Trade-off Among Timeliness, Messages and Accuracy for Large-Scale Information Management

René Brunner

Computer Networks and Distributed Systems Group
Computer Architecture Department
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Advisors: Dr. Felix Freitag and Dr. Leandro Navarro

A THESIS PRESENTED TO THE TECHNICAL UNIVERSITY OF CATALONIA IN FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor in Computer Science

AND FOR THE MENTION OF

Doctor Europaeus

Barcelona, Spain
May 2011

Dedicated

To my parents

To my grandparents

ABSTRACT

The increasing amount of data and the number of nodes in large-scale environments require new techniques for information management. Examples of such environments are the decentralized infrastructures of Computational Grid and Computational Cloud applications. These large-scale applications need different kinds of aggregated information such as resource monitoring, resource discovery or economic information. The challenge of providing timely and accurate information in large scale environments arise from the distribution of the information. Reasons for delays in distributed information system are a long information transmission time due to the distribution, churn and failures.

A problem of large applications such as peer-to-peer (P2P) systems is the increasing retrieval time of the information due to the decentralization of the data and the failure proneness. However, many applications need a timely information provision. Another problem is an increasing network consumption when the application scales to millions of users and data. Using approximation techniques allows reducing the retrieval time and the network consumption. However, the usage of approximation techniques decreases the accuracy of the results. Thus, the remaining problem is to offer a trade-off in order to solve the conflicting requirements of fast information retrieval, accurate results and low messaging cost.

Our goal is to reach a self-adaptive decision mechanism to offer a trade-off among the retrieval time, the network consumption and the accuracy of the result. Self-adaption enables distributed software to modify its behavior based on changes in the operating environment. In large-scale information systems that use hierarchical data aggregation, we apply self-adaptation to control the approximation used for the information retrieval and reduces the network consumption and the retrieval time. The hypothesis of the thesis is that approximation techniques can reduce the retrieval time and the network consumption while guaranteeing an accuracy of the results, while considering user's defined priorities.

First, this presented research addresses the problem of a trade-off among a timely information retrieval, accurate results and low messaging cost by proposing a summarization algorithm for resource discovery in P2P-content networks. After identifying how summarization can improve the discovery process, we propose an algorithm which uses a precision-recall metric to compare the accuracy and to offer a user-driven trade-off. Second, we propose an algorithm that applies a self-adaptive decision making on each node. The decision is about the pruning of the query and returning the result instead of continuing the query. The pruning reduces the retrieval time and the network consumption at the cost of a lower accuracy in contrast to continuing the query. The algorithm uses an analytic hierarchy process to assess the user's priorities and to propose a trade-off in order to satisfy the accuracy requirements with a low message cost and a short delay.

A quantitative analysis evaluates our presented algorithms with a simulator, which is fed with real data of a network topology and the nodes' attributes. The usage of a simulator instead of the prototype allows the evaluation in a large scale of several thousands of nodes. The algorithm for content summarization is evaluated with half a million of resources and with different query types. The self-adaptive algorithm is evaluated with a simulator of several thousands of nodes that are created from real data. A qualitative analysis addresses the integration of the simulator's components in existing market frameworks for Computational Grid and Cloud applications.

The proposed content summarization algorithm reduces the information retrieval time from a logarithmic increase to a constant factor. Furthermore, the message size is reduced significantly by applying the summarization technique. For the user, a precision-recall metric allows defining the relation between the retrieval time and the accuracy. The self-adaptive algorithm reduces the number of messages needed from an exponential increase to a constant factor. At the same time, the retrieval time is reduced to a constant factor under an increasing number of nodes. Finally, the algorithm delivers the data with the required accuracy adjusting the depth of the query according to the network conditions.

RESUM

La gestió de la informació exigeix noves tècniques que tractin amb la creixent quantitat de dades i nodes en entorns a gran escala. Alguns exemples d'aquests entorns són les infraestructures descentralitzades de Computacional Grid i Cloud. Les aplicacions a gran escala necessiten diferents classes d'informació agregada com monitorització de recursos i informació econòmica. El desafiament de proporcionar una provisió ràpida i acurada d'informació en ambients de grans escala sorgeix de la distribució de la informació. Una raó és que el sistema d'informació ha de tractar amb l'adaptabilitat i fracassos d'aquests ambients.

Un problema amb aplicacions molt grans com en sistemes peer-to-peer (P2P) és el creixent temps de recuperació de l'informació a causa de la descentralització de les dades i la facilitat al fracàs. No obstant això, moltes aplicacions necessiten una provisió d'informació puntual. A més, alguns usuaris i aplicacions accepten inexactituds dels resultats si la informació es reparteix a temps. A més i més, el consum de xarxa creixent fa que sorgeixi un altre problema per l'escalabilitat del sistema. La utilització de tècniques d'aproximació permet reduir el temps de recuperació i el consum de xarxa. No obstant això, l'ús de tècniques d'aproximació disminueix la precisió dels resultats. Així, el problema restant és oferir un compromís per resoldre els requisits en conflicte d'extracció de la informació ràpida, resultats acurats i cost d'enviament baix.

El nostre objectiu és obtenir un mecanisme de decisió completament auto-adaptatiu per tal d'oferir el compromís entre temps de recuperació, consum de xarxa i precisió del resultat. Autoadaptacío permet al programari distribuït modificar el seu comportament en funció dels canvis a l'entorn d'operació. En sistemes d'informació de gran escala que utilitzen agregació de dades jeràrquica, l'auto-adaptació permet controlar l'aproximació utilitzada per a l'extracció de la informació i redueixen el consum de xarxa i el temps de recuperació. La hipòtesi principal d'aquesta tesi és que els tècniques d'aproximació permeten reduir el temps de recuperació i el consum de xarxa mentre es garanteix una precisió adequada definida per l'usari.

La recerca que es presenta, introdueix un algoritme de sumarització de continguts per a la descoberta de recursos a xarxes de contingut P2P. Després d'identificar com sumarització pot millorar el procés de descoberta, proposem una mètrica que s'utilitza per comparar la precisió i oferir un compromís definit per l'usuari. Després, introduïm un algoritme nou que aplica l'auto-adaptació a un ordre per satisfer els requisits de precisió amb un cost de missatge baix i un retard curt. Basat en les prioritats d'usuari, l'algoritme troba automàticament un compromís.

L'anàlisi quantitativa avalua els algoritmes presentats amb un simulador per permetre l'evacuació d'uns quants milers de nodes. El simulador s'alimenta amb dades d'una topologia de xarxa i uns atributs dels nodes reals. L'algoritme de sumarització de contingut s'avalua amb mig milió de recursos i amb diferents tipus de sol·licituds. L'anàlisi qualitativa avalua la integració del components del simulador en estructures de mercat existents per a aplicacions de Computacional Grid i Cloud. Així, la funcionalitat implementada del simulador (com el procés d'agregació i la query language) és comprovada per la integració de prototips.

L'algoritme de sumarització de contingut proposat redueix el temps d'extracció de l'informació d'un augment logarítmic a un factor constant. A més, també permet que la mida del missatge es redueix significativament. Per a l'usuari, una precision-recall mètric permet definir la relació entre el nivell de precisió i el temps d'extracció de la informació. Alhora, el temps de recuperació es redueix a un factor constant sota un nombre creixent de nodes. Finalment, l'algoritme reparteix les dades amb la precisió exigida i ajusta la profunditat de la sol·licitud segons les condicions de xarxa. Els algoritmes introduïts són prometedors per ser utilitzats per l'agregació d'informació en nous sistemes de gestió de la informació de gran escala en el futur.

ACKNOWLEDGEMENTS

This section was the most difficult and at the same time the most enjoyable to write. The writing of a dissertation can be a lonely and isolating experience, yet it is obviously not possible without the personal and practical support of numerous people. I would like to acknowledge all the countless and wonderful people I met during the past 4 years of the thesis.

First, and foremost, I would like to express my sincere thanks to my advisors Dr. Felix Freitag and Dr. Leandro Navarro for their guidance, encouragements, support, patience and feedbacks throughout my PhD candidature. Their patience in reading draft after draft of every paper, proposal and idea I wrote up continues to be very impressive. I thank them for always being willing to meet me in their office.

I want to acknowledge the Catalan ministry AGAUR for providing me a 3 year graduate fellowship to finish my thesis. In this way, I want to thank all the people from this lovely country and from Barcelona for their warm reception, integration, hospitality and openness.

I am grateful to thank Omer F. Rana for his time and providing a wonderful research exchange at the Cardiff University and his helpful comments and discussions. I would like to thank all the people with who I collaborated during this stay. In this way, I would also thank all the people I met in Cardiff during the stay for their hospitality.

I am grateful to my committee members for their time, their helpful comments and discussions. Furthermore, I have benefited greatly from the advice of the many unknow reviewers of the submitted journals and conferences.

This work was supported in part by the Spanish Ministry of Education and Science and the European Union under Contract P2PGrid (TIN2007-68050-C03-01), Delfin (TIN2010-20140-C03-01), CATNETS (IST-FP6-003769), SORMA (EU IST-FP6-034286) and Grid4All (EU IST-FP6-034567).

I would like to thank many people of the faculty, staff of the UPC, colleagues at the CNDS group, colleagues of the research projects P2PGrid, CATNETS, SORMA and Grid4All, my colleagues in the room C6-E208 and “The Big Bang Theory“ group for the conversations at the Thursdays lunches.

My parents and familiy members have been a constant source of support and encouragement during my life, and this thesis would certainly not have existed without them.

CONTENTS

Abstract i

Acknowledgements v

List of Figures xii

List of Tables xiii

CHAPTER 1 INTRODUCTION 1

1.1. Problem Statement 1

1.2. The Importance of Timely and Accurate Results 2

1.3. Rationale 4

1.4. Outline and Contributions 6

CHAPTER 2 LARGE-SCALE INFORMATION AGGREGATION SYSTEMS 9

2.1. Taxonomy for P2P Information Aggregation Systems 10

2.1.1. Propagation Strategy 10

2.1.2. Information Retrieval Process 13

2.1.3. Approximation Strategy 17

2.1.4. Underlying Frameworks and Implementations 19

2.2. Survey of P2P Information Aggregation Systems 21

2.3. System Comparison 27

2.3.1. Reactive and Hybrid Information Retrieval 27

2.3.2. Proactive Information Retrieval 30

CHAPTER 3 COMMON METHODS 33

3.1. Optimization by Approximations 33

3.2. Success Criteria 35

3.2.1. Quantitative Results 36
3.2.2. Qualitative Analysis 38
3.3. Simulation Infrastructure 38
3.3.1. Simulation Framework 38
3.3.2. Prototype for Real-world Scenarios 40
3.3.3. Simulator for Large-scale Networks 41
3.4. Evaluation of Large-scale Network Topology 42

CHAPTER 4 CONTENT SUMMARIZATION 47
4.1. Introduction 47
4.2. Content Summarization Technique 48
4.3. Example Scenario 53
4.4. System Design 54
4.4.1. System Initialization 55
4.4.2. The Information Dissemination 55
4.4.3. The Information Retrieval 58
4.4.4. The Network-aware Information Discovery 59
4.5. Evaluation 60
4.5.1. Summary Dissemination Costs 60
4.5.2. Information Retrieval Costs 67
4.5.3. Network-aware Resource Discovery 75
4.6. Summary 78
4.7. Discussion 78

CHAPTER 5 SELF-ADAPTIVE APPROXIMATION 83
5.1. Introduction 83
5.2. Hierarchical Approximation 85
5.2.1. Aggregation Abstraction 85
5.2.2. Hierarchical Aggregation Example 89
5.2.3. Approximation Technique 90
5.3. Self-adaptive Trade-off 91
5.3.1. The Analytic Hierarchy Process 91
5.3.2. Assessing the Network Imprecision 94

5.3.3. Assessing the Approximation Imprecision 96
5.3.4. Decision Process 97
5.3.5. AHP-based Pruning for Information Aggregation 98
5.4. Experimental Evaluation 100
5.4.1. Evaluation of Retrieval Time and Network Consumption . 100
5.4.2. Evaluation of AHP-based Trade-off 102
5.4.3. Parameter Analysis 105
5.5. Summary 108
5.6. Discussion 109

CHAPTER 6 INTEGRATION TO GRID MARKETS 111
6.1. Motivation 111
6.2. Requirements 113
6.2.1. Economic Requirements 113
6.2.2. Temporal Information Occurrence 114
6.3. Decentralized Market Information 114
6.4. Integrated Model: Information and Markets 116
6.4.1. Integration with SORMA Framework 116
6.4.2. Integration with Grid4All Framework 118
6.5. Summary 120
6.6. Discussion 120

CHAPTER 7 CONCLUSIONS AND FUTURE WORK 125
7.1. Summary 125
7.2. Conclusions 126
7.3. Future Work and Directions 128

BIBLIOGRAPHY 147

List of Figures

2.1. Taxonomy of strategies for the information propagation. 11
2.2. Taxonomy of the information provision. 14
2.3. Taxonomy of the approximation strategy. 18
2.4. Taxonomy of the framework implementation. 20

3.1. Pastry-based simulation framework 39
3.2. The CDF of the RTTs of 600 PlanetLab nodes. 43
3.3. The behavior of an increasing size of transferred data. 44
3.4. Analysis of PlanetLab attributes. 46

4.1. Concept hierarchy generated after sequence 1. 50
4.2. Concept hierarchy generated after sequence 2. 50
4.3. Concept hierarchy generated after sequence 3-5. 51
4.4. Example scenario. 54
4.5. Job sequence with dependencies. 59
4.6. Number of sent messages per peer 63
4.7. Total size of the summary for a tree depth of 1. 63
4.8. Total size of the summary for threshold of 0.1. 64
4.9. Maximum time for the dissemination of a message. 64
4.10. Total data with a discrete uniform distribution. 65
4.11. Total data with a Gaussian distribution. 66
4.12. Percentage of improvement. 66
4.13. Average hops needed to find the resources for query type 3. 68
4.14. Recall for query type 2. 70
4.15. Precision for query type 2. 70

4.16. Recall for a depth of 4. 71
4.17. Precision for a depth of 4. 72
4.18. Comparing the ratio α for different summarization depths. 73
4.19. Comparing the ratio α for different summarization depths 74
4.20. Cobweb-based algorithm and network-aware algorithm. 75
4.21. Critical workflow path in comparison to the Cobweb threshold. . . 76
4.22. Worst case comparison. 77

5.1. Example of a hierarchical aggregation. 89
5.2. Example of an AHP structure. 92
5.3. Number of sent messages in comparison to baseline systems. . . . 101
5.4. Retrieval time in comparison to baseline systems. 101
5.5. Scores of the two alternatives regarding the number of messages. . 106
5.6. Scores for the criteria of the alternative continue. 106
5.7. Scores for the criteria of the alternative prune. 107

6.1. The overlay layers. 115
6.2. Possible trading places. 116
6.3. Integration to the market framework of SORMA. 117
6.4. Integration to the market framework of Grid4All. 119
6.5. State diagram of trading resources in the Grid4All market. 120

List of Tables

2.1. Information systems with an unbiased approximation. 22
2.2. Reactive information retrieval systems. 23
2.3. Reactive-proactive information retrieval systems. 24
2.4. Proactive information retrieval systems. 25
2.5. Other information retrieval systems. 26
2.6. A categorization of reactive and hybrid information retrieval systems. 27
2.7. A categorization of proactive information dissemination. 30

3.1. Possible optimizations in large-scale information retrieval. 34

4.1. Sequence of sample resources. 49
4.2. Simulation setup data. 62
4.3. Attributes and possible values for a job query. 67

5.1. The reciprocal matrix 93
5.2. Normalized values for the calculation of the alternatives. 96
5.3. Analysis of the offered trade-off. 103

6.1. Economic market parameters. 113

CHAPTER 1

INTRODUCTION

Information retrieval in large-scale applications such as peer-to-peer (P2P) databases, information aggregation systems and monitoring tools have become more and more important over the last few years. Examples are applications for document storage, file sharing and resource monitoring. A problem arising in those large-scale systems is that the information is spread on different nodes. The distribution and decentralization of the information pose hurdles in terms of delays and imprecisions to perform queries in an efficient and accurate way.

1.1 Problem Statement

Jain et al. [JKM$^+$08] and Sacha [SNSP10] argue that distributed aggregation systems suffer under inaccuracy. The inaccuracy is mainly caused by failures that arise in distributed environments. The reasons of inaccuracy are shown by the following observations:

- Failures cause that nodes disconnect involuntarily from the system. A consequence is that a node is not available until the failure is solved or that the system decides to exclude the node.
- The delay of the information provision process means that the information

turns obsolete when it reaches the user who started the query.

- The reorganization of the hierarchical aggregation tree in a P2P-system causes delays. When a peer leaves the system (due to failures or churn), the system has to reorganize the aggregation tree. Thus, the reorganization process causes a certain delay and consequently the result might include inaccuracy [JKM+08].

- Churn causes information loss. For example, if a node is disconnected, the information about the CPU load gets lost. Furthermore, the information of the subtree can be temporally unavailable.

1.2 The Importance of Timely and Accurate Results

In many companies, a trend embodies the outsourcing of their calculations, simulations or data. Computational Grid and Cloud systems help companies and organizations to reduce their costs in IT infrastructures. Examples are data storage (e.g. Freenet [1]) or computational power (e.g. Sorma [BBN+09] and Grid4All [KNB+08]). However, most of these applications need an information service (e.g. Resource monitoring (GIS) [TTF+06] or a Market Information System [BFN08]) to sustain, to support or to improve the core functionalities of the applications.

It is known that it is impossible to obtain totally accurate data in an instant from an environment with millions of users. The terms of totally accurate data means for example the average load of CPUs or the correct lookup for a certain resource within the Seti@Home [2] network. To provide the information, a system needs a retrieval time, which is at least a multiple of the round-trip-time (RTT) between the participants. However, failures increase the risk in regard to delays and inaccuracies in large-scale systems.

Approximation techniques are a promising technique to improve existing information aggregation and discovery systems [SNSP10, ADGK07]. Approximate queries are already successfully applied to solve similar problems in traditional databases. The traditional databases are designed for a centralized usage and ap-

[1] http://freenetproject.org/
[2] http://setiathome.berkeley.edu/

ply approximate queries to speed up the calculations and the queries. Similar approximations provide an efficient information lookup process for distributed and large-scale information systems. In distributed systems, approximations promise a larger reduction of the retrieval time due to the longer message transfer times than in local databases.

Distributed aggregation systems address the retrieval of information [VRBV03, ZSZ03, BMVV04, YD04, CH06, BGMGM03, JKvS04]. The mentioned systems use completely decentralized and hierarchy-based routing overlays. The decentralized overlay avoids single point of failures like it is possible in server applications.

Two different types of information retrieval are considered: information discovery and information aggregation. The information discovery is used to find a certain data within a very large bundle of information. For example, a user or the system looks for the price of a resource, which matches exact constraints such as CPU, memory and disk capacity. In contrast to information discovery, the information aggregation is used to find an aggregated valued such as minimum, maximum or average. To obtain such a value, the system queries for all data, which matches the search constrain and computes the requested value.

The information provision process distinguishes between two requirements that are attribute-driven: **timely information** and **accurate data**. Requirements for timely information are shown by studies reporting that data mining applications have a higher sensitivity for timely data than for accuracy [MGL$^+$10, AGP99, ADGK07, CDN07]. Often, data analysts accept small inaccuracies if the information is retrieved earlier. The Etomic [3] project retrieves information in a large-scale with time intervals consisting of nano seconds, which shows a high time sensitivity. The objective of the project is the measurement of the data changes in the network traffic. Often, the discovery of one matching resource which can execute the required job out of many resources is more important (shorter retrieval time as the system can stop the query and return the result) than the need to find the optimal matching resource (the system needs to query the complete system, which takes longer than stopping after a sufficient match).

Another requirement is accurate data. Academic evaluations such as earthquake or cancer simulations are executed on large-scale simulations on Seti@home

[3]http://www.etomic.org/

or Boinc [4]. These simulations need fully accurate results even when the calculations run over several hours or days. Other examples are companies in the financial sector; they prefer accurate estimations of the markets (e.g., Markov chains) instead of fast and inaccurate estimations. The discovery process of a Computational Grid system needs to discover at least one resource which matches the exact constraints of a job. However, a result has higher probability for incompatibility with the requirements, if the system returns quickly a resource.

1.3 Rationale

The hypothesis of the thesis is that approximations can improve the retrieval time and number of messages while reaching a high level of accuracy. Reducing the retrieval time and the number of sent messages is important to reach the scalability of the information retrieval process in large environments. Nowadays, the increasing number of applications with an increasing amount of users and data requires efficient processes for the information retrieval. Proposing approximations allows reducing the volume of data, which needs to be queried. The intention of such a reduction is to improve the lookup time, to decrease the number of messages and to decrease the transmitted data without degradation of the quality of data.

Besides an increasing volume of data in large-scale applications, a problem arises from the geographical distribution of the users (e.g. P2P) as delays, failures and churn lead to inaccuracy. The inaccuracy is mainly caused by a loss of information or delays after failures. The network distance between each node delays the message exchange, which can lead to obsolete data. Inaccuracy exists in large-scale and geographically distributed environments [CLKB04]. An example for such inaccuracies is the snap shot problem, which is described by Mattern [Mat93, CLKB04].

There is a need to investigate the dealing with such an inaccuracy in combination with approximations. Both, the applied approximation and the distributed network cause inaccuracies. However, approximation contributes to reduce the number of sent messages, the message size and the retrieval time. An important challenge is to find a trade-off among the parameters of retrieval time, network

[4]http://boinc.berkeley.edu/

consumption and accuracy. The work in this thesis proposes a metric (precision-recall metric in Chapter 4) to feed a user-driven decision to regulate the quality of the content summarization with the parameters of summarization depth and threshold. Furthermore, this thesis propose a self-adaptive decision-making (analytic hierarchy process in Section 5), which prunes the query if a minimum of the user's defined priorities for time, messages and accuracy is reached.

The **purpose of the thesis** is to design scalable information retrieval mechanisms for large-scale distributed systems, large in number of messages, nodes and volume of information. In this study, approximation techniques (such as content summaries and hierarchical approximate queries) are used to improve the information retrieval process for large-scale environments. In addition, the thesis offers a (user-driven) trade-off between fast and accurate date. The timely and accurate provision of information is a challenge in large-scale environments [HV03, BA01]. The study reduces the information retrieval time while guaranteeing a reasonable accuracy.

Information discovery can be improved by applying approximation techniques. The use of content summarization reduces the information discovery process from a linear increasing manner to a constant lookup time in regard to the volume of information. The number of needed hops can be reduced to one, by providing the summarized information. The content summarization technique reduces the transferred message size to less than 10% with 500 000 resources.

Some works already address the advantages and the need for approximation within P2P systems. Arai et al. [ADGK07], Hall and Carzaniga [HC09], and Jelasity et al. [JMB05] draw random samples from the overlay's nodes. In their work, the focus is on the efficient selection of the sample peers. Sacha et al. [SNSP10] states that the random sample technique is very inefficient when it is compared to their algorithm of sampling the nodes in an efficient and unbiased way. All of the presented approximate query techniques in this paragraph are based on gossip P2P systems. However, other works show that hierarchical aggregation systems (i) are resistant to churn and failures [VRBV03, ZSZ03, BMVV04, YD04, CH06], (ii) have less message overhead since the structure avoids that peers receive duplicated messages, (iii) are (at least) as fast as gossip based approximation (iv) and also selects randomly the samples by the assignation of the peer identifier. Our algorithms apply approximation techniques that use a structured information provision such as a hierarchical aggregation.

The thesis' **hypothesis** is that the retrieval time of the query and the network consumption can be reduced by the usage of approximation. At the same time, a reasonable accuracy can be guaranteed when applying approximations.

In detail, the following research question arise: Is it possible to reduce the information retrieval time and keep the accuracy with approximation techniques in large-scale aggregation systems? An expectation is that applying content summarization allows reducing the lookup time to one hop and guaranteeing a precision of over 90%. Furthermore, the application of hierarchical approximate queries expects a reduction of the lookup time by 1/3 for an environment, which scales up to 10 000 nodes.

Another research question is: can a trade-off between accuracy and timely information retrieval be found? A trade-off between timely and accurate information retrieval is offered by applying a precision-recall metric on content summarization for the information discovery in Chapter 4. Chapter 5 proposes a self-regulated decision-making to find a trade-off among timeliness, network consumption and accuracy.

1.4 Outline and Contributions

The main contribution of the thesis is the application of approximation techniques to large-scale information retrieval systems. The contributions are divided in three main parts:

- Algorithms use a content summarization technique for the information discovery to reduce the retrieval time, the message size and the network consumption. This thesis provides a precision-recall metric to provide information for a user driven trade-off among the parameters (Chapter 4).

- An algorithm for the aggregation and estimation of an average value with an approximation technique that reduces the retrieval time, the number of sent messages. A self-adaptive algorithm regulates the accuracy in regard to user's priorities in order to find a trade-off between timeliness, messages and accuracy (Chapter 5).

- Chapter 6 describes the experience and evaluation of a prototype of P2P-based information retrieval that has been integrated in larger Grid market applications.

Portions of work presented in this thesis have been partially or completely derived from the following set of publications.

Chapter 4 is partially derived from the following publications:

- *Brunner, R.; Caminero, A. C.; Rana O. F.; Freitag, F. & Navarro, L. Network-aware Summarisation for Resource Discovery in P2P-Content Networks. Future Generation Computer Systems (FGCS), 2011*

Chapter 5 is partially derived from the following publications:

- *Brunner, R.; Freitag, F. & Navarro, L. Self-regulated Trade-off Among Timeliness, Messages and Accuracy for Approximate Queries in Large-scale Information Aggregation. The 14-th International Conference on Network-Based Information Systems (NBiS-2011), 2011*

- *Brunner, R.; Freitag, F. & Navarro, L. Uncertainty Management for the Retrieval of Economic Information from Distributed Markets. Second International Conference on Scalable Uncertainty Management, pp. 106-119, volume 5291, Lecture Notes in Computer Science, Napoli, Italy, 2008*

Chapter 6 is partially derived from the following publications:

- *Brunner, R.; Freitag, F. & Navarro, L. Towards the Development of a Decentralized Market Information System: Requirements and Architecture Parallel and Distributed Computing in Finance (PDCoF'08).Proceedings of the 22nd IPDPS, Miami, FL, USA, 2008*

- *Borissov, N.; Brunner, R.; Neumann, D.; Freitag, F.; Navarro, L. & Weinhardt, C. Fostering Efficiency of Computational Resource Allocation - Integrating Information Services into Markets Proceedings of the 17th European Conference on Information Systems (ECIS' 09), 2009, 2048-2059*

- *León, X.; Vilajosana, X.; Brunner, R.; Krishnaswamy, R.; Navarro, L.; Freitag, F. & Marquès, J. M. Information and regulation in decentralized marketplaces for P2P-Grids COPS in proceedings of WETICE, 2008*

CHAPTER 2

LARGE-SCALE INFORMATION AGGREGATION SYSTEMS

This chapter studies the state of the art of large-scale information aggregation systems. The study outlines a taxonomy of P2P-based databases (P2PDB) and large-scale information aggregation systems in large-scale environments. Afterwards, the studied information aggregation systems are surveyed in Section 2.2. The last subsection gives a comparison of the systems and classifies them in regard to the properties that are given by the taxonomy. Moreover, the comparison includes our proposed algorithms.

Some surveys and taxonomies already address the information management in distributed systems. Lua et al. [LCP$^+$05] studies overlay network schemes of peer-to-peer systems. Risson et al. [RM06] focus their survey on the different search mechanisms of peer-to-peer systems. Liu and Plale [LP03] describe a taxonomy and a survey of P2P-based systems that apply publish-subscribe design pattern. Blanco et al. [BAH$^+$06] survey data management in peer-to-peer systems. In comparison to the above presented surveys and taxonomies, this chapter outlines taxonomy about large-scale information aggregation with a focus on propagation strategies, retrieval process and approximation mechanisms.

2.1 Taxonomy for P2P Information Aggregation Systems

After studying over 40 information aggregation systems for large-scale environments, four main categories for the taxonomy are outlined. First, different propagation strategies are used to disseminate the information. Second, the category of the information retrieval process distinguishes between a reactive or a proactive manner [JMB05] of information retrieval that is based on different query languages. Third, approximation techniques optimize the large-scale information retrieval. Fourth, the implementation of the used framework depends on the requirements and functionalities of the application.

The remainder of the thesis distinguish between read-dominated and write-dominated attributes. The distinction between the attributes is important for the reduction of delays, latencies and inaccuracies of the information retrieval process [YD04]. For example, read-dominated attributes like the number of CPUs per node changes rarely, while write-dominated attributes like the number of processes or jobs change quite often. On one hand, an information provision tuned for read-dominated attributes cause a higher network load when applied to write-dominated attributes because the system would have many queries instead of only pushing the information after updates. On the other hand, an information provision tuned for write-dominated attributes suffer from query latency or imprecision for read-dominated attributes, because the system would push on each update the new information instead of querying when a new request arrives [Yal05].

2.1.1 Propagation Strategy

The taxonomy of the propagation strategy is illustrated in Figure 2.1.

Propagation strategy

Non-tree-based protocols send the query message or the information message to all known nodes or a subset of known nodes without following a tree structure.

- A **server-client model** propagates the information via an unstructured combination of nodes as servers and clients. An example of the server-client model for distributed data aggregation is the Mariposa system [SAP$^+$96].

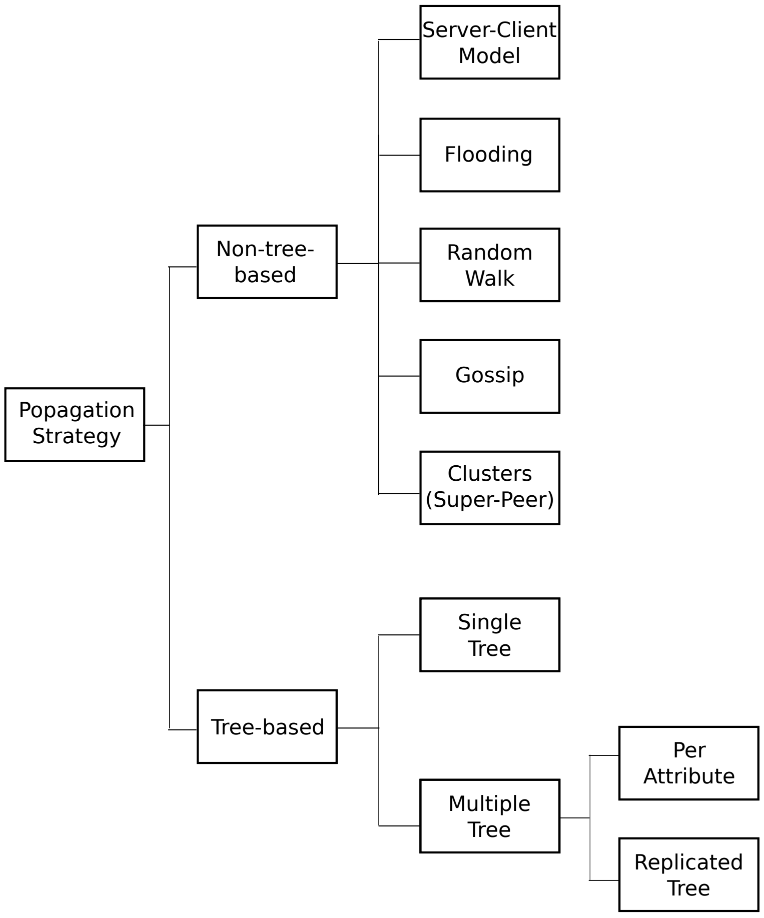


Figure 2.1: Taxonomy of strategies for the information propagation.

Mariposa provides the information dissemination for a wide area network and it shows the feasibility to provide a distributed database in a large distributed environment. Scalability problems can arise in server-client models when each node provides a server for all other nodes.

- A **Flooding** A P2P-based flooding sends the message to all known neighbor nodes. The nodes forward the message to all its neighbor nodes after receiving the message. P2P means that all nodes within the system are equally privileged and have the same capabilities. An advantage of the flooding strategy is a fast information provision in comparison to structured P2P protocols. Moreover, the flooding is resilient against failures and churn. However, a disadvantage is a very high network load. The network load is often caused by nodes (at the beginning of the second hop) that forward the identical message to the same nodes. A consequence of the uncontrolled message replication is an exponentially increase of the message load.

- **Gossip:** In contrast to the flooding mechanism, the **gossip-based** protocol sends the query message only to a subset of nodes, instead to all known neighbor nodes. Afterwards, the receiving nodes forward the message to a subset of their neighbor nodes. The gossip protocol is often called epidemic because the information is spread similar to the way that a viral infection spreads in a biological population. The gossip protocol sends a message to a subset of neighbor nodes to ask for an individual value of an attribute or to ask for an aggregated value such as minimum, maximum or average. The asked node returns its (aggregated) value. After receiving a result, the query's initializer node calculates the aggregated value with other incoming results [JMB05, KDG03, ZHC08].

- **Random walk:** A random walk operates similar to a gossip protocol since the random walk sends a message to a small subset of peers [MMKG06]. The subset is often smaller in a random walk than in a gossip-based protocol. Thereby, the selection of the next nodes is defined by the local node and not by the node, which initialized the query. Nevertheless, previous nodes on the path of the random walk are excluded. The random walk has predefined time to live (TTL) parameter, which defines the duration of the random walk. When the TTL is triggered on a node, it returns the message to the initializer node.

- **Clusters:** Cluster mechanisms organize the peers into groups with similar attributes. Clustered P2P systems are often grouped to clients and super peers [YGM03]. In the cluster model, each client is connected to at least one super-peer and the super peers are neighbors of other super peers to connect the individual clusters. In information aggregation systems, the super-peers provide the information about the entire system. First, the clients send its local information to the super-peers where the information about a cluster is aggregated. Second, the super-peers share and aggregate the information of their cluster with other super-peers. Third, a client can query its assigned super-peer for the global information.

A **tree-based** propagation strategy aggregates the information along a hierarchical structure. The aggregation process is executed bottom-up until the root node(s) obtains the result. The leaf nodes send their information to their parent nodes that aggregate the value of their child nodes and their own value and transmit the result to their parent nodes. An advantage of the tree based structure is that the network load decreases from an exponential factor like in epidemic-based protocols to a linear factor. The lookup time is also limited to log N (N is the number of the nodes of the system). However, a criticism is that the system suffers of maintenance cost in terms of delays, accuracy and reorganization if higher level nodes fail [VRBV03, VRB04, YD04].

- **Single tree:** Hierarchical aggregation systems with a single tree have only one aggregation tree, which contains all nodes. An example is Astrolabe [VRBV03], which executes each query on a single tree. A problem arising from a single tree is the risk of overloading the root node and higher level nodes, and that the root node is a single point of failure. Replication of the root node is one solution to avoid the single point of failure.

- **Multiple trees:** Hierarchical aggregation systems use multiple trees to improve the single tree structure. The provision of several aggregation trees reduces the risk of overloading nodes on a higher level of the aggregation tree and to reduce the risk of a single point of failure. For example, SDIMS [YD04] creates an aggregation tree for each attribute.

2.1.2 Information Retrieval Process

The taxonomy of the information retrieval process, which is partly adapted from Jelasity et al. [JMB05] is illustrated in Figure 2.2. The following three information retrieval processes characterize the different systems: reactive, proactive and hybrid. The expression reactive is also called pull-up and the proactive expression is also called push-up.

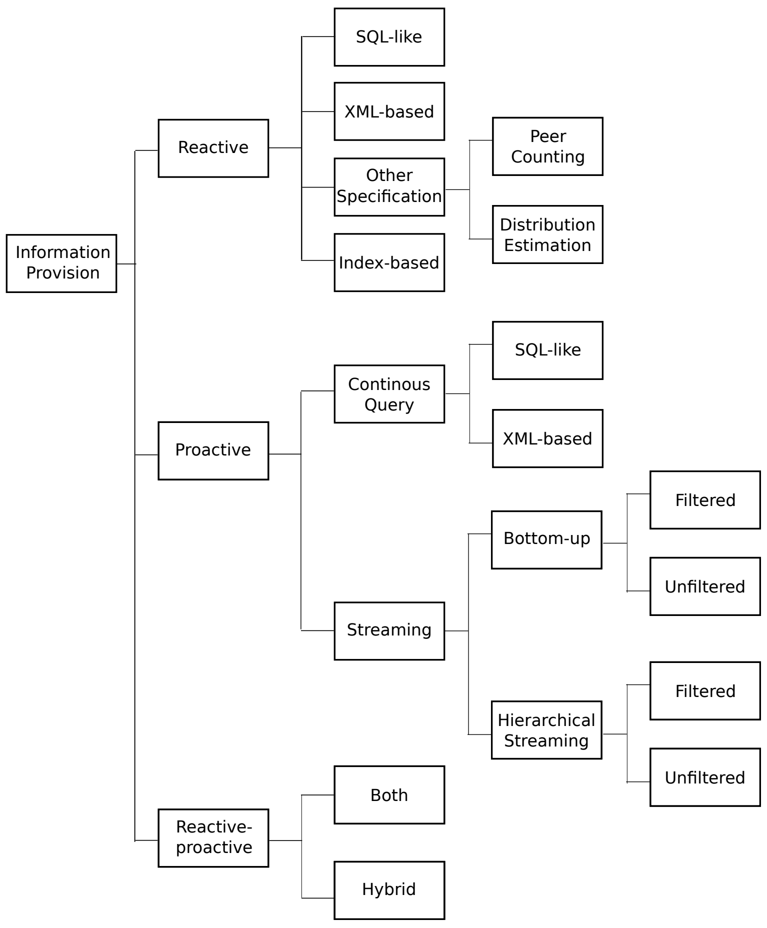


Figure 2.2: Taxonomy of the information provision.

Reactive:

The information in reactive systems is retrieved in a manner of on-demand. On-demand means that a user or an automated trigger initiates the information process when information is required. After the initiation of the retrieval process, the system pulls-up the data to the initializer node. Pull-up means that a query is disseminated to neighbor or child nodes. After the dissemination process is stopped, the information is aggregated and returned to the initializer node. In hierarchical aggregation, the query follows a tree structure to the nodes and aggregates the values upwards. In epidemic networks, the root nodes send a query to a subset of its neighbor nodes. A query generally defines a time to live (TTL) to determine when the query is sent back to the root node. A reactive aggregation is better in terms of network load if the attributes are write-dominated. An example for write-domination is a system where only rarely complex information is required and where the values of the attributes change frequently.

Reactive systems execute a **query** to retrieve the aggregated information. Four different query types can be found: SQL-like [VRBV03], XML-based [CGMS07], Index-based [ZSZ03] or other specifications such as Peer-counting [ADGK07] and distributed estimations [SNSP10]:

- SQL-like queries are the most common queries used in the studied information aggregation systems. The common operators are: minimum, maximum, average, sum and count. Some systems provide grouping operators but the application of all operators of the SQL syntax is still not reached because of its complexity in applying them to a very-large scale [Yal05].
- XML-based query languages are second most used for the information aggregation within the studied systems. The aggregation mechanism follows the tree structure which is provided by the XML syntax.
- Index-based query languages are a combination of a key and value. The query looks for the value of a corresponding key or index.
- Other specification such as Peer-counting [ADGK07] and distributed estimations [SNSP10]. A query language is invented for some information aggregation algorithms. For example, the estimation of the number of peers in the network or the estimation of an attribute's distribution follows another aggregation syntax.

Proactive:

The **proactive** information retrieval process means that the information is disseminated on-change. On-change means that a node sends the new information (push-up) after the local data is updated. The push-up mechanism sends the information to parent nodes or to a set of neighbor nodes. Afterwards, the nodes with an incoming message transfer the message until all nodes are aware of the new information. The proactive information retrieval process is similar to publish-subscribe pattern, where a set of an event's subscribers are notified when such a new event arrives. The process is divided in continuous queries and streaming mechanisms.

Applications with a **continuous query** mechanism send queries at predefined time periods. For example, the system inquiries every second or minute for the arrival of new data. In environments with read-dominated attributes, continuous queries have a higher message load than streaming and reactive propagation mechanisms.

The **aggregated streaming** mechanism disseminates a message to the nodes after the data is updated. In an environment with read dominated attributes, the proactive information provision causes a lower network load since the system aggregates only the information if the local information is updated. In hierarchical streaming, the requested value is aggregated towards the root node. After receiving the new value and aggregating the new result, the root nodes disseminate the new result to all child nodes [VRB04][ZSZ03].

The streaming mechanism can be divided in regard to their filter mechanism. **Filtered streaming** means that at each node, the algorithm decides if the message is of interest to forward the information to other nodes. An example for such a filter is a Bloom filter. **Unfiltered streaming** forwards all data even if the information has no interest for the global result.

The proactive information retrieval is only applicable if many users apply the function for the information aggregation. When many users need a different query type then the proactive system needs to install an aggregation mechanism for each aggregation function and each attribute. For many systems (such as SDIMS and STAR), the installation of these functions would mean the creation of an aggregation tree for each function, which would increase the network load significantly. Considering many complex queries would lead to a problem of installing new aggregation functions to all trees.

Proactive-reactive:

Some systems incorporate a hybrid mechanism, which allows reacting to dynamic environments when changing between read-domination and write-domination. Two types integrate both retrieval mechanisms:

- Both retrieval mechanisms incorporate reactive and proactive information retrieval in one system and allow switching between reactive and proactive information retrieval depending on the attribute's properties [YD04]. Switching has an advantage of a lower network load in an environment with frequently changing attributes between read-domination and write-domination.

- Hybrid mechanism has two algorithms; each follows either the reactive or the proactive retrieval process. The proactive information retrieval makes a pre-selection of fresh updated data and provides the updates to some nodes. However, the system pushes the data only to some point (half way bottom-up). The other way (half way top-down), the reactive information retrieval starts a query for the preselected data when a node receives a new query.

2.1.3 Approximation Strategy

The taxonomy of the approximation strategy is illustrated in Figure 2.3. The figure shows the results of a study of the aggregation systems in regard to the applied approximation and in regard to the sample selection process.

Approximation:

Approximation is used in large-scale information retrieval to exclude unimportant information during the aggregation process. Another reason to apply the approximation is the reduction of queried peers that leads to a smaller sample size. A consequence of a smaller sample size is the reduction of the number of sent messages and the reduction of the retrieval time. However, the reduction of the sample size leads to higher costs in terms of inaccuracy. There are three possibilities in regard to use approximation in large scale aggregation systems:

- **Explicit:** The information aggregation process applies explicitly approximation techniques. The system is aware of the approximation and queries for a sample data based on a subset of nodes. Based on the sample data, the algorithm estimates the result for the entire system [ADGK07, SNSP10]. Another algorithm, which uses explicitly approximation, is proposed by Hayek et al. [HRVM08]. The content summarization technique estimates the availability of certain information on a node or cluster of nodes.

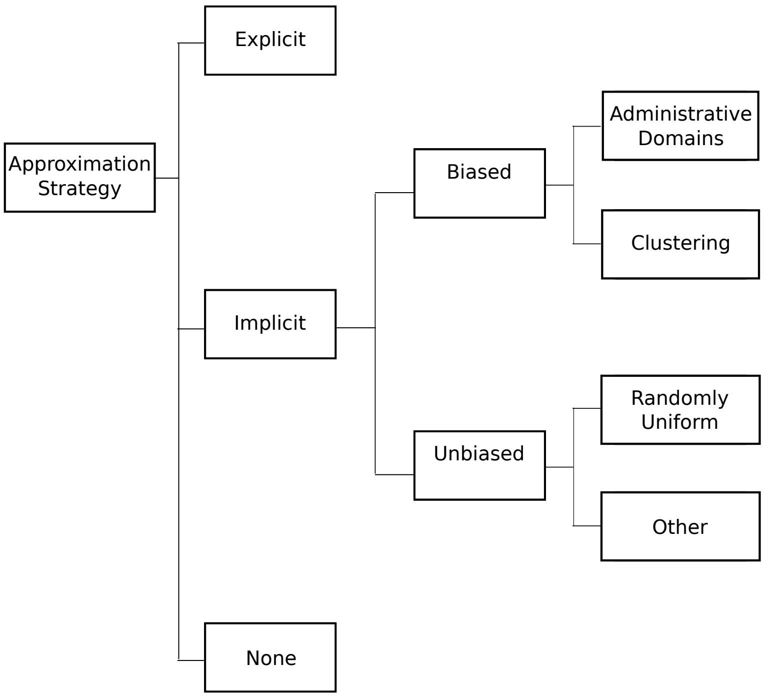


Figure 2.3: Taxonomy of the approximation strategy.

- **Implicit:** Algorithms with an implicit approximation excludes nodes that contain unimportant data or nodes that have high costs to obtain the information. In the case of excluding nodes, the system applies approximation with a sample data which contains only the values of the "good" nodes. An example is the exclusion of failure prone peers from the querying process [JKM$^+$07b].

- **None:** No approximations are used and the system queries all nodes without any constraints. Nevertheless, failures and churn might hide information form the information system. Furthermore, delays in P2P systems can turn the results obsolete and the old data would be used to approximate the current value. Both cases lead to a reduction from querying all nodes to querying only a subset of the entire system. However, this thesis does not consider these cases as explicit or implicit approximation since they are a common occurrence in P2P systems and generally insignificant in comparison to the explicit reduction of the sample size.

Sample selection:

The sample data is the information, provided by a set of nodes (sample nodes). The approximation for the information aggregation is based on the data obtained from the sample nodes. An advantage of querying only sample nodes than the entire system is the reduction of messages and retrieval time. However, the reduction increases the inaccuracy. Therefore, the selection of the sample nodes is important to ensure unbiased approximation. The sample data is either biased or unbiased:

- Biased means that the set of nodes are selected by containing altered preferences (e.g. geographical, political or organizational) that can lead to unbiased estimations. For example, using an isolated administrative domain like in SDIMS or Astrolabe excludes a subset (subtrees) of nodes and has the risk that a whole organization can be excluded from the estimation. The result is biased, if the organization(s) has significantly different data than the rest of the organizations.

- Unbiased sample data is generally randomly chosen and returns unaltered estimations. Besides random sampling, also other techniques ensure unbiased approximations [ADGK07, SNSP10].

2.1.4 Underlying Frameworks and Implementations

The underlying **frameworks and implementations** of the information aggregation systems describe the used P2P protocols and the manner of its development. The taxonomy of the underlying frameworks is illustrated in Figure 2.4. The studied systems often base their aggregation mechanism on existing frameworks. Most of the studied systems use P2P frameworks that are implemented in Java. Surveys on P2P systems [LCP$^+$05] distinguish between two different types: unstructured P2P and structured overlays.

For example, an **unstructured P2P protocol** is the gossip-based protocol such as Gnutella [RF02]. These systems for the information retrieval commonly develop its own protocol or adapt an existing algorithm since the unstructured protocols are easier to develop in comparison to structured protocols.

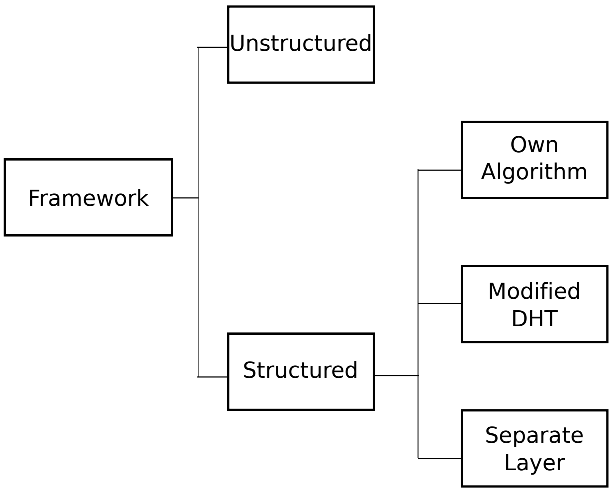


Figure 2.4: Taxonomy of the framework implementation.

Information aggregation systems that use **structured overlays** can be divided in three different types:

- SOMO and Astrolabe develop their **own algorithm**. An advantage of developing an own algorithm is the independence from other ideas and constraints of third parties. Furthermore, the complexity of unstructured P2P protocols is lower than the complexity of structured overlays.

- DAT and SDIMS modify an **existing DHT** and extend their requirements of the aggregation process to the existing processes of the underlying DHT. Modifying an existing DHT allows reusing components, which have already proven a well functioning. An example for such a component is the subscription mechanism to join an overlay or the communication layer among the peers.

- SanFermin uses existing P2P frameworks and builds their aggregation **layer on top of the DHT overlay** without any modifications. Chawathe et al. [CRR$^+$05] argue that a clear separation of the overlay and the functional layers has the advantage of reusing existing services. Furthermore, the clear separation of the layers leads to an easier implementation and maintenance of the system.

2.2 Survey of P2P Information Aggregation Systems

This section lists information aggregation systems for large-scale environments. The following table surveys the characteristics of the systems in regard to the presented taxonomy in Section 2.1. Table 2.1 shows the information aggregation systems that use unbiased approximation for the information retrieval. Afterwards, the tables contain information aggregation systems without approximation or a biased approximation. Table 2.2 lists the reactive information aggregation systems. Table 2.3 contains the reactive-proactive information aggregation systems. Table 2.4 surveys the proactive information provision systems. Within the tables, (-) means that no explicit information is available for the survey.

System	Architecture	Strategy	Implementation	Reactive-Proactive	Query Language	Strategy
Adam2 [SNSP10]	Random walk	Gossip-based	Unstructured	Reactive	Statistical distribution of an attribute's values	Explicit
Arai et al. [ADGK07]	Random walk	Gossip-based	Unstructured	Reactive	SQL-like	Explicit
Massoulié et al. [MMKG06]	Random walk	Random walk	Unstructured	Reactive	Two peer counting approaches	Explicit
DBGlobe: [PAP+03]	Hierarchical	Gossip-based	Unstructured P2P	Proactive	XML-based	Explicit
Kempe et al. [KDG03]	Flooding - based and random walks	Gossip-based (Uniform gossip)	Unstructured P2P	Proactive	Sum, average, quantile aggregation	Explicit (random sample)
Bawa et al. [BGMGM03] (single tree) (Propagate2All)	Single Tree & Multiple-trees	Hierarchical & Flooding	Unstructured P2P	Both	MIN and MAX not suited COUNT and SUM	Explicit (NODE-COUNTING)
Bawa et al. [BGGMM04] (Newscast)	Gossip	Epidemic	Unstructured P2P	Continuous Query	sum, count, average, min, max	Explicit (node counting)
PeerSum [HRVM08]	Super peer	Gossip	DHT-based	Both	Discovery, Count	Explicit
PeerDB [OTZ+03]	Index-based	1. gossip/flooding and structured queries	BestPeer	Reactive	SQL query (data keyword thesaurus)	Explicit TTL to obtained answers
Kashyap et al. [KDN+06]	Gossip-based	Epidemic	Unstructured P2P	Push-Pull	MIN, MAX, SUM, AVERAGE, and RANK	Explicit

Table 2.1: Information systems with an unbiased approximation.

System	Architecture	Strategy	Implementation	Reactive-Proactive	Query Language
DASIS [AAGW04]	Single Tree	Hierarchical	Kademlia-like DHT	Reactive	MIN, MAX, COUNT, and AVG
Cone [BMVV04]	Multiple trees, each attribute has a ring-based DHT	Hierarchical	Prefix-based DHT, CHORD	Reactive	max, min, sum, union
Piazza [HIM+04]	Central node hosts an index structure	Hierarchical	P2P	Reactive	XML- and RDF-based
Piazza [GHI+01]	Spheres of cooperation (gossip)	Hierarchical	Unstructured P2P	Reactive	XML- and RDF-based
PIER [HCH+05]	Structured	Hierarchical	DHT-based	Reactive	SQL-like language
XPeer [CGMS07]	Clusters with super peers	Epidemic	Unstructured P2P	Reactive	Xquery (xml-based)
Wigan [CW08]	Index-based Tracker and seed, (Server-Client) Super-peer	-	BitTorrent	Reactive	SQL-like
Mariposa [SAP+96]	-	Server-Client	-	Reactive	SQL-like
PeerOLAP [KNO+02]	Gossip	Epidemic	Unstructured P2P	Reactive	SQL-like
Seaweed [NDMR08]	Super-peer on top of Pastry	Hierarchical	Pastry	Reactive	(SQL-like)
Ryeng and Norvag [RN08]	Single tree	Hierarchical		Reactive	SQL like
Del Vecchio and Son [DVS05]	Gossip-based (back: same path the query request followed)	Epidemic	Unstructured P2P (Gnutella)	Reactive	-

Table 2.2: Reactive information retrieval systems.

System	Architecture	Strategy	Implementation	Reactive-Proactive	Query Language	Strategy	Sample
Willow [VRB04]	Single trees	Hierarchical	Kademlia-like DHT	Both	SQL	None	Biased
SDIMS [YD04]	Multiple trees (one tree for each attribute)	Hierarchical	Pastry	Both	SQL-like (no complex)	None	Biased (Administrative Isolation)
Shruti [YD07]	Multiple trees (one tree for each attribute)	Hierarchical	Pastry	Both	SQL-like (no complex)	Implicit (Self-Tuning)	Biased (based on SDIMS)
SOMO [ZSZ03]	Single Tree	Hierarchical	Arbitrary DHT	Both	Key-based	None	-

Table 2.3: Reactive-proactive information retrieval systems.

System	Architecture	Strategy	Implementation	Reactive-Proactive	Query Language	Strategy	Sample
Astrolabe [VRBV03]	Single Tree	Hierarchical	Unstructured P2P	Proactive	SQL aggregation queries	None	Biased
STAR [JKM$^+$07b]	Multiple trees (one tree for each attribute)	Hierarchical	Pastry	Proactive (continuous queries)	SQL-like (no complex)	Implicit (Punning unreliable nodes)	Biased
Li et al. [LSL05]	Single Tree	Hierarchical	Arbitrary DHT	Proactive	MIN, MAX, COUNT, SUM, AVG	None	-
DAT [CH06]	Single Tree	Hierarchical	Chord	Proactive	min, max, count, and sum	-	-
In-Network Aggregation [DKR04]	Single Aggregation Tree	Hierarchical	-	Continuous query	SQL (SUM, AVG, COUNT, MIN and MAX)	Filter	-
NiagaraCQ [CDTW00]	Grouping method	Epidemic	Unstructured P2P	Continuous query	XML-QL	-	-
IrisNet [GKK$^+$03]	Hierarchical network overlay (DNS)	Hierarchical data model (XML)	Hierarchical network overlay (DNS)	Both (Query and update)	Xpath (xml-based)	-	-
Jelasty et al. [JMB05]	Unstructured P2P/ Gossip/ epidemic	Epidemic	Unstructured P2P	Proactive	Counting, averages, sums, products, and extremal values	Implicit	Biased
San Fermin [CH08]	Swap forests	Gossip-based / Random walk	Pastry	Proactive	Summing counters, comparison for equals, maximum, and string parsing	Shortest path as sample	Biased

Table 2.4: Proactive information retrieval systems.

System	Architecture	Strategy	Implementation	Reactive-Proactive	Query Language
BATON [JOT+06]	B-tree	Hierarchical	BestPeer	Reactive	Multi-attribute queries
LOT [AWIK06]	Multiple virtual trees	Hierarchical	Structured	Reactive	Range queries SQL-like
Bayeux [ZZJ+01]	Single tree	Hierarchical	Tapestry	Proactive	-
Split Stream [CDmK+03]	Multiple-trees	Hierarchical	Pastry	Proactive	-
Cougar [DGR+03]	Multiple trees	Hierarchical	Unstructured P2P	Both (hybrid pull-push)	SQL-like, MIN and AVG
TAG [MFHH02]	Single tree	Hierarchical	(wireless sensor networks)	Proactive (streaming sensor) Distribution phase pushed down collection phase	SQL-like (GROUP BY)
APPA [VP04]	super-peer network, parallel execution, tree-based on super-peers	Hierarchical	JXTA framework	Reactive	XQuery
Edutella [NWST02]	Super-peers	Epidemic	Unstructured P2P	Hybrid	RDF-based
ActiveXML [ABC+03]	XML tree	Hierarchical	Unstructured	Reactive	XQuery

Table 2.5: Other information retrieval systems.

2.3 System Comparison

This section categorizes the studied large-scale information aggregation systems. First, a discussion presents the advantages of the different reactive processes for the information retrieval. The hybrid information retrieval processes are compared in conjunction with reactive information provision. Second, the proactive information dissemination is compared to the reactive information retrieval.

The focus of the study is on the application of approximation techniques for large-scale information aggregation. Our studied information aggregation systems apply exclusively decentralized solutions. However, the existence of centralized accuracy and approximation management (e.g. Aqua [AGP99] and quasi-copies [OW00]) is known but these information management systems are not considered as scalable.

2.3.1 Reactive and Hybrid Information Retrieval

This section compares reactive systems and reactive-proactive Systems (Table 2.6) systems with our proposed algorithms.

	No approx	Biased	Unbiased
Epidemic	PeerDB, PIER	SeaWeed	Adam2, Arai et al., Massoulié et al., PeerSum
Hierarchical	Cone, SOMO	DASIS, Piazza, SDIMS, Willow	Chapter 4, Chapter 5

Table 2.6: A categorization of reactive and hybrid information retrieval systems.

Epidemic Information aggregation:

Several systems improve the scalability of information provision for large-scale information aggregation: DASIS [AAGW04], XPeer [CGMS07], PIER [HCH$^+$05], PeerDB[OTZ$^+$03] and PeerOLAP [KNO$^+$02]. The mentioned systems propose epidemic information provision. Epidemic information dissemination has the advantage of being scalable in terms of a high fault tolerance. In contrast to epidemic information dissemination, our algorithm is based on a hierarchical structure of the nodes. A hierarchical structure has generally a lower message load by avoiding the transmission of identical messages to the same nodes. Furthermore, traditional databases use also hierarchical structure for an efficient query processing. Another aspect is that the mentioned systems include all nodes of the systems in the query, which should lead to accurate results. However, known imprecision arise in very large systems due to message delay and data dynamics [JKM$^+$08], which leads to inaccurate results. Our algorithm applies approximate queries that compute the data from only a subset of nodes but retrieve the query result much faster than if all nodes would be included.

Biased Gossip-based information aggregation:

Information aggregation systems such as SeaWeed [NDMR08] use approximations to obtain the requested data and are based on an unstructured aggregation process. Using unstructured information aggregation based on gossip protocols or random walks guarantees certain robustness against failures and churn in environments with a large number of nodes. However, applying a cluster mechanism to achieve approximation can lead to biased approximations when peers are grouped by a certain attributes value. In contrast to the mentioned system, our retrieval process is hierarchical based which has in general a lower network load. Furthermore, the hierarchy allows a finer tuning of the lookup process to find automatically a trade off among parameters. Moreover, our approximation technique uses a randomly uniform organization of the nodes within an aggregation tree to provide unbiased results.

Unbiased Gossip-based Aggregation:

Arai et al. [ADGK07], Adam2 [SNSP10], and Massoulié [MMKG06] propose approximation techniques for the information retrieval in large scale systems that consists in selecting the nodes as samples. In the mentioned works, sample peers are obtained through gossip-based message propagation and random walks in unstructured P2P networks. Their approximation techniques are already successfully applied in traditional databases. Applying these approximation techniques to distributed information aggregation leads to faster retrieval times in large environments. The authors concentrate their work on finding an unbiased way to select a sample of nodes. An unbiased set of sample peers is important for the quality of the approximation, which can be measured by the accuracy of the result. In contrast to our work, these systems are designed to obtain the sample nodes in unstructured overlays. A hierarchical structure has generally a lower message load by avoiding the transmission of identical messages to the same nodes. Furthermore, hierarchical structures are successfully used for an efficient query processing in traditional databases.

Hierarchical Aggregation:

SOMO [ZSZ03] and Cone [BMVV04] propose a hierarchical aggregation process for the distributed information retrieval. The hierarchical aggregation provides scalability and it avoids duplicated messages to the same nodes and therefore reduces network load. Their aggregation tree includes all nodes of the system in the retrieval process in order to obtain accurate data. The work in this thesis extends the tree-based aggregation with an approximation mechanism to deal with imprecisions from distributed environments. The use of approximate queries ensures high accuracy and reduces significantly the retrieval time and the network load.

Biased Hierarchical Approximations:

NI [JKM+08], Moara [KYG+08] and Shruti [YD07] include approximations in an aggregation system, which is based on a hierarchical tree structure presented in SDIMS [YD04]. The idea of the authors is the exclusion of unreliable subtrees from the information retrieval process. A consequence from that pruning is a lower network consumption and a lower network imprecision. However, their systems propose an administrative isolation to provide autonomy, security and isolation properties. That administrative isolation can lead to biased approximation. For example, the pruning of only unstable subtrees can lead to an exclusion of a group of nodes with the same isolation property. The exclusion of a large group of nodes can lead to an unrepresentative sample set and lead to imprecise approximation results, if entire organizations are excluded. Another criticism is that these systems cut off unreliable subtrees but the used pruning technique does not necessarily reduce the maximum tree depth. Consequently, the retrieval time would be close to the time needed for the querying of the entire aggregation tree. However, our algorithm limits the total tree depth for the data aggregation, which leads to a significant reduction of the retrieval time.

2.3.2 Proactive Information Retrieval

Many systems such as Astrolabe, STAR and DAT follow a proactive manner, which is categorized in Table 2.7. The information is disseminated after an update of the data or the nodes are continuously queried. The presented summarisation technique and the AHP-based algorithm that are presented in this thesis are limited to a reactive information retrieval. The reasons are explained in the following paragraphs.

	No approx	Biased	Unbiased
Epidemic	NiagaraCQ	Jelasity, Astrolabe	DBGlobe, Kempe et al.
Hierarchical	DAT, IrisNet	STAR	

Table 2.7: A categorization of proactive information dissemination in large scale.

First, continuous streaming systems like STAR have a higher network consumption in contrast to streaming or reactive systems when the time periods between the queries are short. Conversely, a longer time period leads to less accurate results in dynamic environments.

Second, in proactive systems for the large-scale information aggregation, the query structures are installed in advance and a modification or flexibility on the fly is more difficult and cost intensive in terms of maintenance messages. Furthermore, the possibly required information needs to be defined by the installation of the system. Knowing the needed queries in advance is a problem for rare query structures. A consequence of the limitation to static queries is that applying complex queries remains a challenge [Yal05]. In contrast to a proactive information dissemination, a reactive system can create flexibly a query on demand and send the query conditions to a node. The receiving node looks locally for the requested attributes. The node can optionally apply the query operator and return the new result, if the node has the value for the demanded attribute.

Third, the need for information retrieval for read-dominated attributes is shown by example scenarios and by related systems. In read-dominated environments, a reactive information retrieval has a lower consumption of network bandwidth than a proactive information retrieval.

CHAPTER 3

COMMON METHODS

This chapter describes the common methods that are used for the studies in Chapter 4, Chapter 5 and Chapter 6. First, the reasons and the selection mechanism for the studies are described in Section 3.1. Afterwards, Section 3.2 defines the success criteria for the optimizations of the large-scale information provision in terms of quantitative measurements and qualitative evaluations. Section 3.3 describes the simulation infrastructures to evaluate the thesis' hypothesis that approximation can help in providing timely and accurate results under low network consumption. Section 3.4 evaluates the input values for the network topology in regard to obtain data from real network environments.

3.1 Optimization by Approximations

The objective of this thesis is the provision of timely and accurate results with low network consumption. This section elaborates different optimization possibilities. The usage of approximation is discussed to speed up the information retrieval process and to reduce the number of messages while guaranteeing a reasonable accuracy.

Table 3.1 outlines the reasons for using approximation instead of using other techniques to obtain a fast retrieval time, a low number of messages and a high accuracy of the results. The ratings in the table are based on our subjective perception. However, the direction if a feature is positive or negative for the total result is based on the presented examples in the table.

Choosing the approximation technique promises a faster information retrieval time and fewer sent messages in comparison to a baseline system which follows a standard information retrieval process such as Astrolabe [VRBV03]. However, the approximation has a risk of inaccurate results.

Table 3.1: Possible optimizations in large-scale information retrieval.

Technique	*Time*	*Quality*	*Messages*	Examples
Approximation	+++	- -	+++	[DUA04], [ADGK07], [SNSP10]
Routing Mechanisms	++	++	- -	[YD04]
Information Caching	++	- -	++	[TDVK99]
Replication	+	+++	- - -	[ABC$^+$03]
Push / Pull	+	o	+	[YD07]

Improvements of the routing mechanisms such as using a structured information provision instead of an unstructured information provision reduces the network overhead but it can lead to shorter delays or in inaccurate results that are caused by failures. Moreover, structured overlays can lead to higher maintenance costs in terms of the number of sent messages within highly dynamic or failure prone environments. There are already many different routing systems. Therefore, the presented algorithm is based on a common and efficient routing mechanism (see Section 3.3).

The information caching mechanism saves final or intermediate values of the queries. Therefore, the system can return the result without querying the system, if the same query is triggered at least twice. In an environment with many identical queries, the caching of the results leads to a faster retrieval time and to a lower network consumption than in heterogeneous environments with flexible and complex queries. However, a high accuracy is difficult to reach since the data in the cache might be obsolete.

The replication of the retrieval process ensures a high accuracy since it reduces the impact of failures. The replicated nodes can transfer the requested message, if a node suffers under a failure within a replicated tree. The replication of the node structure leads to a faster retrieval time as a delay on one single node does not affect the total retrieval process. A criticism is that the replication multiplies the number of messages and leads to additional maintenance messages.

To provide an efficient information delivery process, two different attribute types are distinguished: write-domination and read-domination [JKM+07a, JKM+08]. The write-dominated attributes change often but are queried only a few times. Read-dominated attributes change rarely and are queried often. The adaption to read-dominated or write-dominated attributes of a system improves the number of sent messages and the retrieval time. On one side, the adaption avoids pushing unnecessarily messages to the systems. On the other side, the system pushes the information if there is a high demand and stops the query mechanism, which has a higher cost in terms of messages and retrieval time.

In conclusion, an optimization of the information retrieval is reached with approximations:

- Approximation is a promising optimization for our scenarios in regard to a potential reduction of the information retrieval time, network consumption and guaranteeing accurate results.
- The approximation technique can be combined with other optimizations such as including a replication mechanism.
- To our knowledge, there are no research projects about using approximation techniques for structured aggregation for P2P systems. Furthermore, no project focuses on finding a trade-off for the three factors in Table 3.1 in a large-scale environment.

3.2 Success Criteria

This section describes the success criteria of the presented hypothesis. Therefore, the work in this thesis distinguishes between a quantitative measurement and a qualitative analysis of the developed prototype and simulator. The distinction is required as not all aspects of the hypothesis can be measured by the aim of numbers but rather by a proof of concept and correctly behaving functions.

3.2.1 Quantitative Results

The following quantitative criteria are measured to define the success of the objectives of the thesis to improve the retrieval time and network consumption, while guaranteeing a high accuracy. First, the network consumption is measured with the *number of sent messages* and the *compression rate of the transmitted data*. Avoiding to overload the network capacities provides a large scalability of the application in terms of users and data. A large scalability can be reached by limiting the increase of sent messages (e.g., logarithmically instead of exponentially) and by reducing the message size. Second, the timely information retrieval is measured with the *retrieval time*, which measures the delay until the systems returns the answer to a query request. Third, the accuracy of the obtained results is measured by an *inaccuracy metric* and a *precision-recall metric*. Both metrics define the quality of the results, reaching a high accuracy is preferred. All criteria are chosen to show the improvements of the new algorithms in time, accuracy and network consumption.

The number of sent messages per query process is one indicator for the network consumption of the system. Therefore, the sent messages are compared with the number of total nodes within the analyzed network. The system might encounter problems for a large scalability if the relation between the sent messages and the number of nodes $\frac{messages}{nodes}$ has an exponential increase. A reasonable increase for large-scale applications would be a logarithmic or a constant factor in regard to an increasing number of nodes.

Another success criterion is the measured compression of the summarized content. The reduction is measured in terms of the data size that is in Gigabyte (GB) or Megabyte (MB). The compression of the summarized content is important because it influences the network consumption and the distribution time during the dissemination process. A higher compression leads to a smaller message size and consequently, the network consumption is lower and the distribution is faster.

The retrieval time measures the delay when a query is executed on a node until the result is returned. The retrieval time is compared to the number of nodes within a network by the relation $\frac{time}{nodes}$. The retrieval time is measured in milliseconds (ms) or seconds (sec). The retrieval time stays under a minute during the execution of our experiments. The retrieval time should reach a logarithmic increase or a constant factor in regard to an increasing number of nodes to provide a large scalability in terms of the network consumption. However, many applications as mentioned in the introduction need a fast information provision. Thus, even a short and constant retrieval time is preferred.

The inaccuracy metric measures the difference between the retrieved value and the real value. The inaccuracy metric differs from the previously presented imprecision metrics since it measures the difference to the real value. The average inaccuracy of the results per node is calculated as:

$$inaccuracy = \frac{1}{N}\sum_{i=1}^{N}\frac{|R_i - O_i|}{R_i} * 100\% \tag{3.1}$$

where N is the number of iterations (repeated depending on the update rate), R_i is the real value of the requested attribute at iteration i obtained from the input matrix and O_i is the value obtained from the query process at iteration i. The remainder of the thesis distinguishes between the terms (in)accuracy and (im)precision. The accuracy defines the degree of closeness of the measured values to its real values within the system. The precision shows the quality of the measurement in terms of reproducibility or repeatability.

The information discovery is evaluated by the *recall*, which identifies the set of possible and relevant documents which have been retrieved from the total set R ($\frac{Ra}{R}$), where Ra represents the correctly retrieved documents (each document representing information associated with a particular resource). The *precision* measures the quality of the retrieved results, which is the set of matching documents from the retrieved set A ($\frac{Ra}{A}$). For most applications, a high precision is more important than a high recall. An example is the resource discovery in Computational Grid registries where the discovery of one matching resource is already sufficient.

3.2.2 Qualitative Analysis

The qualitative analysis assesses the degree of the success, which are not measured in numbers. The study defines the qualitative success criteria if the simulations are applicable in real world scenarios and in existing applications. A proof-of-concept evaluates the system and checks the correct behavior of the functions and components. The tests are important to show the application in real applications.

A success criterion for the qualitative analysis is the elaboration and the integration of the developed simulation infrastructure in a real system. The integration is successful if the correct results are returned to the user or its agent. Therefore, a prototype of our simulator is deployed and evaluated in real systems [BBN$^+$09, LVB$^+$08]. The prototype is an information service for the SORMA and Grid4All market places. Therefore, the prototype has the same structures as the simulator which is provided in one Java library.

3.3 Simulation Infrastructure

The simulation infrastructure describes the tools which are developed and used for the evaluation of the hypotheses. The proposed infrastructure has the objective to obtain a model, which presents real world scenarios. The solutions and improvements in terms of algorithms should be valid for an integration to real applications such as monitoring tools for resources in Computational Grid environments [CKFF01] [TTF$^+$06] or Market Information Systems [BFN08]. Therefore, the framework of the simulator and its components are described. Afterwards, a presentation shows how the framework is tested in real applications and how the simulations are executed.

3.3.1 Simulation Framework

Our simulation Framework is built on the FreePastry library[1], which includes Scribe and Past. A reason for using the FreePastry project is a wide usage of the library in many P2P projects. Furthermore, the library provides a clear separation of the functional layers, which provides an easier handling of modifications [CRR$^+$05, LSL05]. The library provides a generic, scalable and efficient sub-

[1]http://www.freepastry.org/

strate for peer-to-peer networks. The decentralization, the self-organization and the fault-tolerance of the FreePastry overlay permit evaluations of our algorithms within the Internet. The following paragraphs describe the architecture of the used framework (bottom-up), which is shown in Figure 3.1.

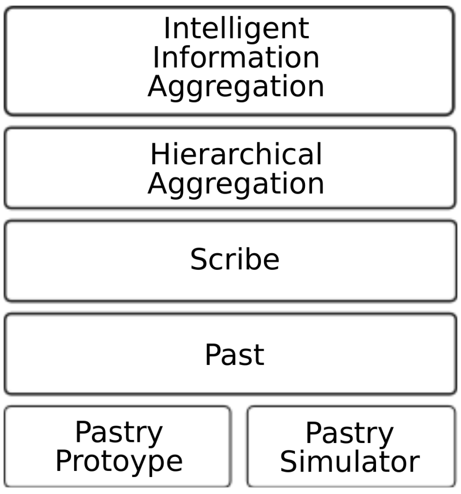


Figure 3.1: Pastry-based simulation framework

The communication layer (Pastry prototype and Pastry simulator) is responsible for the physical message transfer between the nodes. The FreePastry library provides two possibilities for message transfer. First, the communication layer transfers the messages via a real network (Pastry prototype). Second, the communication layer permits to simulate message transfer (Pastry simulator). Therefore, a P2P latency matrix contains the RTTs between the nodes. An advantage of the communication layer is that the upper layers contain identical code and algorithms for both, real evaluations and simulations.

The Past layer provides an efficient routing mechanism for the transmission of messages in P2P networks. The Past protocol provides the routing of messages and the direct sending of messages. The routing mechanism follows a structured approach. The send direct function sends the message direct to target node via IP.

The Scribe layer is a large-scale publish-subscribe protocol for group communication and event notification. Therefore, it provides the functionality of the node's subscription to topics. A notification about new events is sent to the nodes that are assigned to a topic. For example, the events can be updates of new content summaries. Scribe provides a hierarchical multicast mechanism to disseminate the information to the subscribed nodes.

Our hierarchical aggregation layer provides a large-scale aggregation mechanism. Therefore, it reuses and modifies underlying functions such as the multicast mechanism. The multicast mechanism is modified in order to provide a tree-based aggregation. More details and an example of the hierarchical aggregation process are described in Section 5.2.

The intelligent information aggregation layer contains the new algorithms for an efficient information provision. The objective of the layer is to provide the nodes with a self-adaptive decision making about the pruning of the queries. Moreover, the layer contains a user-driven specification of the algorithms that is important to find a trade-off among the timeliness of the information retrieval, the accuracy of the results and low network consumption.

3.3.2 Prototype for Real-world Scenarios

The presented framework provides stable micro benchmark tests that are performed in a real world deployment. First, the developed prototype is successfully deployed with the real communication layer on a distributed set of PCs in our lab. The nodes returned the correct average, minimum and maximum value. Second, in another scenario, the prototype was deployed on five PlanetLab nodes situated on different organizations around the world. The prototype returned the correct average, minimum and average CPU from the resources of the five nodes.

PlanetLab [PMRK06] is an academic test environment, which is deployed on many sites around the world. The PlanetLab research network allows executing test application on hundreds of distributed nodes to gain experience with distributed applications running at a world-wide scale. The architecture of PlanteLab allows among others to test peer-to-peer systems, network mapping and query processing. The infrastructure provides management tools to deploy and control the executed test applications. PlanetLab has the advantage of being well tested and accepted within the research community of network and it allows to test the deploying of novel network services on the internet.

The micro benchmark of the presented framework showed that the developed aggregation mechanism can be deployed in real applications. The benchmark tests show that the simulations are very likely to run in real scenarios since the FreePastry's communication layer is already proven to be executable to several thousands of nodes.

3.3.3 Simulator for Large-scale Networks

The FreePastry simulator is a framework to evaluate large applications which is important for our simulations and FreePastry is proven within the research community to provide reliable results. Furthermore, the framework can change by configurations the underlying communication protocol. The switching between a real communication protocol and the simulation protocol allows testing the implementation in a real environment. After a successful execution, the same code can be simulated in a large-scale environment with 10 000 nodes.

The structured overlay network assigns uniform random keys to data items and organizes its peers with uniform random NodeIDs into a graph, where each data key maps to a peer. The hash map assigns the peers into a large space of identifiers and enables efficient discovery of data items using the given key mappings. An important aspect is the identifier generation process since it defines the selection of the samples. The binary tree used for the hierarchical aggregation depends on its key identifier. Therefore, a random generator, which is implemented by the FreePastry library, generates the key identifier. The randomly generated identifier allows a random generation and consequently a random distribution of the peers. Even if peers are geographically close to each other or situated in the same domain, the peers are assigned to different places within the aggregation tree.

The P2P topology matrix has the size of $n \times m$ where $n, m \leq 10\ 000$. Each transition $n \mapsto m$ is assigned a value obtained from evaluations of PlanetLab. An advantage of using the P2P topology matrix is that it allows introducing failures. Failures are simulated in incrementing a random number, given by a certain percentage, of connection $n \mapsto m$. Thus, the matrix can be varied depending on the envisaged P2P scenarios by changing its failure rate and network type in terms of RTT. In order to simulate a real P2P network the RTT of all PlanetLab nodes are obtained by applying a ping to all nodes. The results return the average maximum and minimum RTT of 10 consecutive pings to each node of the PlanetLab environment. The obtained values are used in the P2P topology matrix of our simulations.

For the simulation of a resource information system for Computational Grid environments, the input values are memory utilization, disk utilization or CPU load. For each scenario, a matrix is setup with the dimensions node n and time t. The values $V_{n,t}$ for each n and t based on the values obtained from PlanetLab. The simulator uses a fix input matrix which allows testing the data and reproducing the results if necessary.

To reach a high number of simulations (and results based on 10 repetitions for each experiment), the experiments are run within a cluster of 73 Nodes of USP Xeon Dual-Core 5148. Each of the 73 nodes has 2 Processors Intel Xeon Dual-Core 2,333GHz, FSB 1333MHz, 4MB Cache and 12 GB memory RAM in 6 modules of 2 GB. The network card is an Intel Pro/1000 Gigabit Ethernet. The experiments are run with Java version 1.6, assigned with the heap size of -Xms1024m -Xmx8192m.

3.4 Evaluation of Large-scale Network Topology

This section describes and explains the initialization of the input data for the simulator. The input data contains the network topology and the values of the retrieved resources' attributes such as the CPU load, the memory usage and the disk usage. The intention of the simulations is to model the experiments like the following:

- Using as much real values as possible
- Deploying the highest amount of network nodes as reasonable or as possible

Most of the related real world evaluations in large-scale information aggregation are applied on internal clusters or on PlanetLab. Thereby, a criticism is that internal clusters do not represent real P2P scenarios with slow nodes or failures. The PlanetLab nodes seem closer to real P2P applications but the nodes are commonly deployed on academic organizations. Therefore, it is limited to a distributed network and not a home user P2P network or a Computational Grid environment. However, after analyzing the existing alternatives, PlanetLab is the most appropriate environment to obtain real results in a large scale. Another criticism point is that real simulations on PlanetLab for the large-scale information aggregation use only between 100 and 200 nodes [JKM$^+$07b]. First, 200 nodes are not sufficient to simulate our large-scale applications. Second, 200 nodes are not sufficient to show the effects of our algorithm, which contains approximation techniques. Even using all nodes in PlanetLab, which are less than 1000, would not be sufficient.

As the evaluation of our algorithm needs several thousands of nodes, real PlanetLab values are analysed to obtain a larger amount of data which is based on real values. Our simulations with the FreePastry execute 10 000 nodes for the evaluations which are more nodes than most related work analyze. Moreover, our algorithm prunes at maximum the queries around 3 000 nodes that makes a network with 10 000 nodes largely sufficient. Over 600 real values are obtained for the P2P topology which are multiplied to obtain a P2P topology with 10 000 x 10 000 nodes.

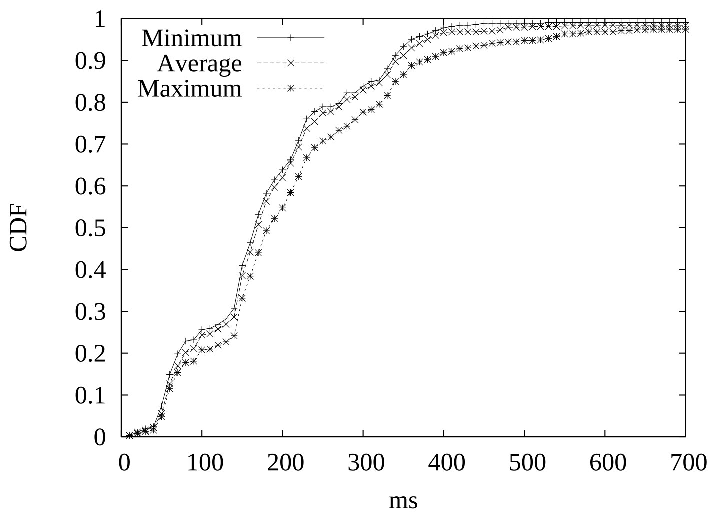


Figure 3.2: The CDF of the RTTs of 600 PlanetLab nodes.

An evaluation of the RTTs of all PlanetLab nodes, by applying a ping to all nodes, obtained 625 successful responses. The results provide the average, maximum and minimum RTT for 10 consecutive pings to each node within the Planet-

Lab environment. Figure 3.2 shows the Cumulative Distribution Function (CDF) for the RTT. The CDF shows that half of the nodes have a RTT faster than 200 ms. The obtained data is the basis for the P2P topology of the executed simulations.

The next experiments test the behavior in regard to an increasing transmitted data size of 6 PlanetLab nodes, which are located in different continents. The results in Figure 3.3 show that the time increases linearly in regard to the data size, which means that the investigated PlanetLab nodes do not suffer from bandwidth throttling. A bandwidth throttling could reduce the effective bandwidth with an increasing data transfer size and falsify the significance simulation. However, the connection is influenced by simultaneous downloads from several users that lead to a sharing of bandwidth among the users. To simulate the data transfers with large data files, the obtained RTT are multiplied with the data size.

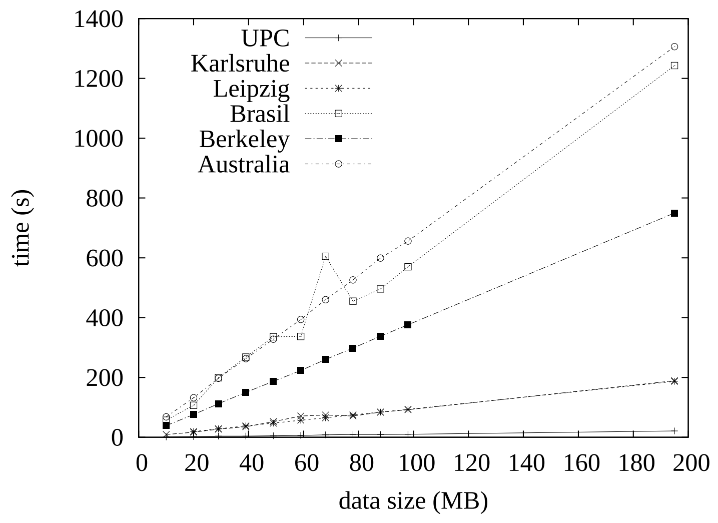
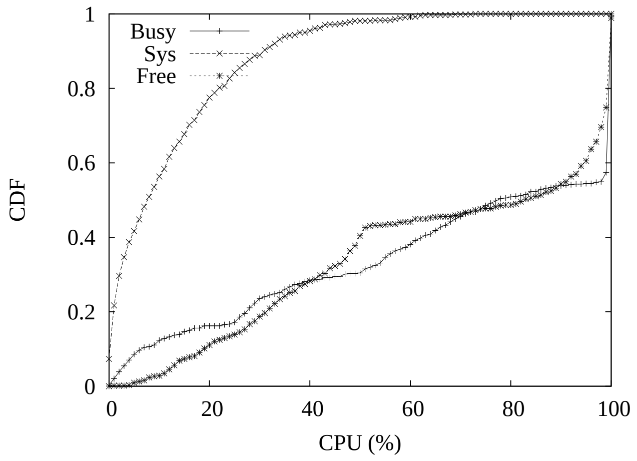


Figure 3.3: The behavior of the network connections of PlanetLab nodes in regard to and increasing size of transferred data.

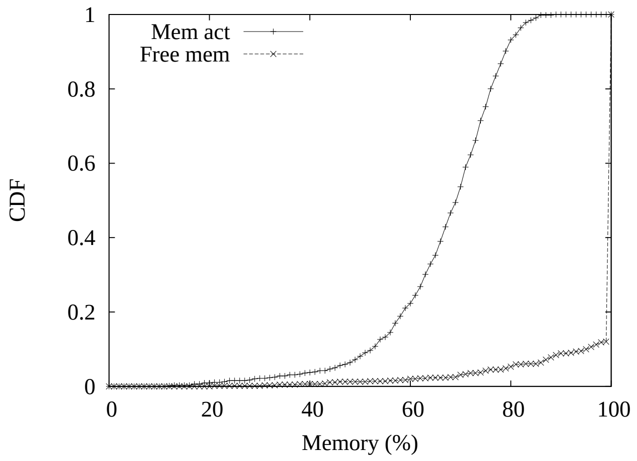
The analysis of the PlanetLab network obtained real input values for the simulated attributes. The characteristics and the usage of the PlanetLab nodes are analysed in terms of CPU, memory and disk in Figures 3.4(a), 3.4(b) and 3.4(c). The obtained results are the basis for several experiments. The results are used as real attributes for the analysis of the algorithm. Moreover, the results serve to obtain the distribution type and value ranges to generate the input data for the summarization of resources.

The values from our PlanetLab analyses for common distribution types showed that most of the attributes have different or no distribution of the attributes. Minitab analyzed the data [2] for the following distributions: Gaussian normal distribution, uniform random, Poisson, Weibull, Pareto, Box-Cox transformation, Lognormal, Exponential, Smallest Extreme Value, Largest Extreme Value, Gamma and Logistic. Therefore, the obtained PlanetLab values are multiplied them to obtain the required number of several thousand nodes. Nevertheless, the simulations also include values with the common distributions. Common distributions are still valid for many real world attributes such as the Poisson distribution or the Pareto distribution. For example, the Pareto distribution describes the social well-fare and it often represents the distribution of products to customers.

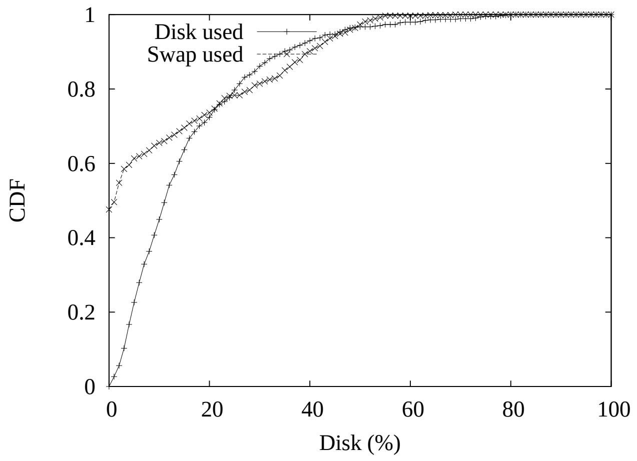
[2]http://www.minitab.com



(a) Analysis of PlanetLab nodes for CPU utilization in percentage.



(b) Analysis of PlanetLab nodes for memory utilization and capacity.



(c) Analysis of PlanetLab nodes for disk capacity and occupation.

Figure 3.4: Analysis of PlanetLab attributes in order to obtain real simulation data.

CHAPTER 4

CONTENT SUMMARIZATION

4.1 Introduction

Large-scale information systems have gained on importance over recent years, often having many concurrent users and managing an increasing amount of data. Different approaches address the scalability challenges that arise within such systems. First, using a P2P infrastructure for the information provision provides scalability in terms of the number of domains and users. Second, summarization techniques help to reduce the amount of information, which is exchanged between nodes fo the system. Therefore, a combination of these two techniques proposes a promising solution for new large-scale information system applications such as Grid-based information systems [CKFF01] [TTF^{+}06].

Many application scenarios have a read-dominated behavior of the information provision. Read-domination means that the information is exposed to few updates and that the users execute frequent queries for new information. Examples include registries of Grid systems, in which the resources are relatively stable in the sense that they provide the same operating system and the same hardware configuration over a long time frame. Although new resources appear in such systems, queries to discover suitable resources to execute a job are likely to reach a much higher number than new resources being added/removed to/from the system. Thus, the data retrieval process has a higher priority in terms of the message size, the number of messages and the retrieval time than the initialization process of creating the summaries.

This chapter proposes a summarization technique which is based on Cobweb clustering. The design of using the summarization technique allows an efficient information retrieval in comparison to the related work. Furthermore, this sections shows the analysis of the scalability of update costs in terms of the number of messages per peer and in terms of a fast dissemination process. Scalability is achieved by the provision of efficient data discovery and by summarization, reducing the amount of data that needs to be exchanged among peers. This chapter specifies and analyzes the behavior of the proposed mechanism using simulations with up to half a million resources, each having several attributes.

4.2 Content Summarization Technique

The summarization technique is based on a clustering algorithm called Cobweb [Fis87], which is an incremental approach for hierarchical conceptual clustering. The system carries out a hill-climbing search through a space of hierarchical classification schemes using operators that enable bidirectional travel through this space. Cobweb uses a heuristic measure called *category utility* (*CU*) to guide search. Gluck and Corter [GC85] originally developed this metric as a means of predicting the basic level in human classification hierarchies. Briefly, basic level categories (e.g., bird) are retrieved more quickly than either more general (e.g., animal) or more specific (e.g., robin) classes during object recognition. More generally, basic level categories are hypothesized to be where a number of inference-related abilities are maximized in humans [MR81].

Category utility (CU) is a trade-off between intra-class similarity and inter-class dissimilarity of objects, where objects are described in terms of (nominal) attribute-value pairs. In [GC85], category utility is defined since the increase in the expected number of attribute values that can be correctly guessed given a partition over the expected number of correct guesses with no such knowledge [Fis87]. CU

Table 4.1: Sequence of sample resources.

Sequence	OS-name	CPU model	LRMS
Resource 1	Linux		
Resource 2	Linux	Pentium4	Condor
Resource 3	Windows		
Resource 4	Linux		Condor
Resource 5	Linux	Pentium4	Condor

is described as:

$$\begin{aligned} CU(C_1, C_2, \ldots, C_n) = {} & \frac{\sum_{k=1}^{n} P(C_k)}{n} * \\ & * \left[\frac{\sum_{i=0}^{a} \sum_{j=0}^{v} P(A_i = V_{i,j} | C_k)^2}{n} - \right. \\ & \left. - \frac{\sum_{i=0}^{a} \sum_{j=0}^{v} P(A_i = V_{i,j})}{n} \right] \end{aligned} \tag{4.1}$$

where n is the number of categories in a classification, a is the number of attributes, v is the number of values, A_i represents an attribute, and $V_{i,j}$ represents the j^{th} value of attribute A_i.

The clustering technique by Gluck and Corter [GC85] works by checking, for each data tuple, which of the following options is the best: (1) inserting the new data tuple into one of the existing categories; (2) creating a new category by merging two existing ones; (3) creating a new category by splitting an existing one. The best option is decided by using the category utility function mentioned before.

For example, each computing resource is described by the operating system (OS-name), the used CPU model and the local resource management system (LRMS). The resources can be represented by the following feature vector:

$$((OS-name)(CPU model)(LRMS))$$

Table 4.1 presents a sequence of summarized resource based on a number of features that has a similar structure to the Cobweb summarization used by

[ISW92]. The 5 resources are added sequentially to the Cobweb-based summarisation. After adding the first resource, a new concept is generated as the hierarchy is empty. Figure 4.1 shows the resulting concept node of resource 1, which is placed at the root. A node represents an object class Ci that is represented by a set of probabilities. That are the probabilities associated with values for each attribute.

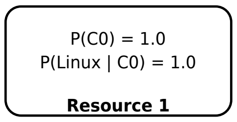


Figure 4.1: Concept hierarchy generated after sequence 1.

In Figure 4.1, only the probabilities of one of the possible values for the `VO` parameter are shown. In Figure 4.1, $P(Linux|C0) = 1.0$ means that the probability for a resource with the attribute `Linux` is 1.0 for class $C0$.

After adding the second resource in sequence 2, the concept hierarchy is split in two nodes, one for each resource, since that has the best category utility in Figure 4.2. The probability for resources with the attribute `Condor` is 0.5 as half the objects in this class have `Condor`. Associated with each object is a probability of belonging to a particular class. In Figure 4.2, this is represented as `P (C1) = 0.5`, which means that 50 % of the objects belong to class $C1$. A class is fully represented by the probabilities of all the attribute values exhibited over objects belonging to it.

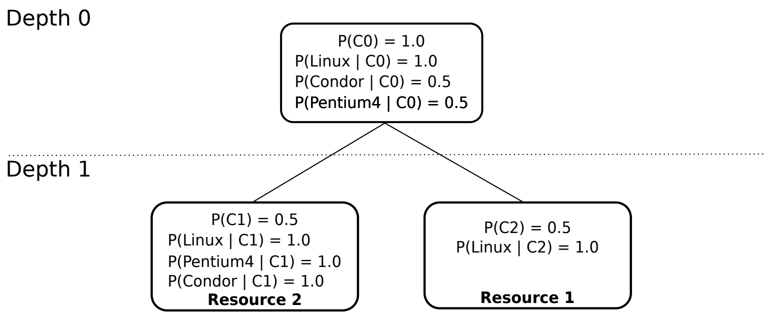


Figure 4.2: Concept hierarchy generated after sequence 2.

As a result of applying Cobweb clustering after sequence 3-5, a category tree is created that encodes the content of the database and whose structure is similar to Figure 4.3. A summary of the database can then be created by *pruning* and *leafing*. The first step leads to branches of the tree being pruned to a given depth. The second step involves generating a summary from the leaves of the pruned tree. Hence the lower the selected depth the more detailed is the summary of the database. For example, a summary of the tree in Figure 4.3 could be created by pruning at depth 0, and this would contain less detail than if depth 1 were chosen. Additionally, as categories are based on the probability of each attribute to take on a particular value, probability values below a given threshold can be filtered out. This third step (named *filtering*) would create a more summarized version of the database.

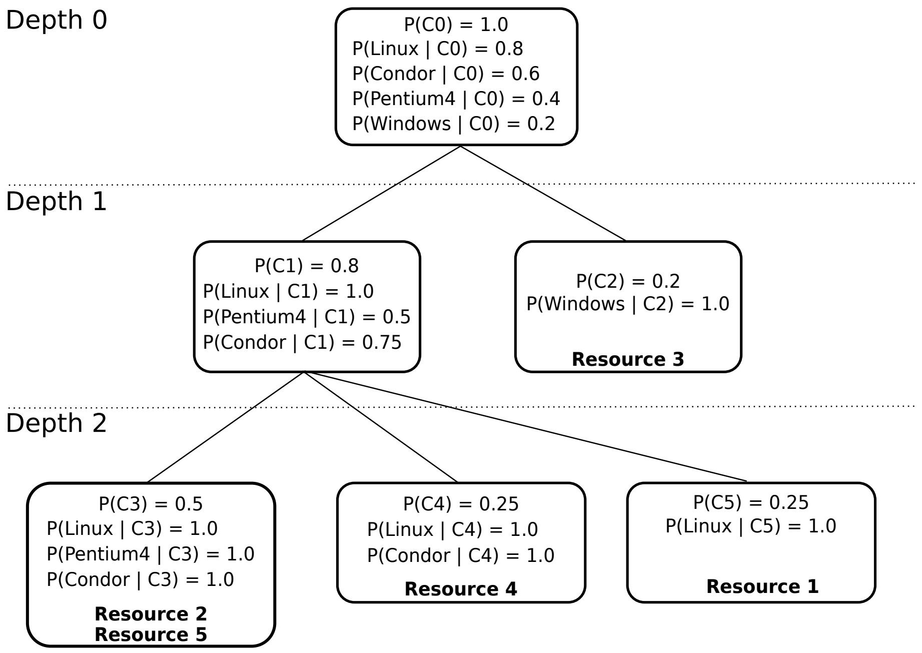


Figure 4.3: Concept hierarchy generated after sequence 3-5.

Once the summary has been created, the next step is the propagation of the summary to other domains, so that they can perform the match-making process. This can involve sending summaries to all the other domains or a subset; in this way, each domain has either information about resources in the whole system or about a subset. The size of this subset affects the match-making process and the scalability of the approach. Sending information to all domains improves match-making but lead to an exchange of large data volumes between domains. Also, keeping such information up-to-date may be difficult, since updates must be forwarded across the entire system. A full sharing across domains is assuemed, in order to improve the accuracy of the match-making process. Once summaries have been propagated, they are used to forward and refer queries to other domains if a suitable resource is not available locally. The technique, which is presented in this chapter, works as follows. First, each domain retrieves information about its local resources. The number and nature of attributes is not fixed so that this approach is general and can perform match-making based on any parameter of interest. A summary is then generated from this retrieved data. This results in the creation of a category hierarchy, from the more general to the more detailed. Then, a summary can be created based on this hierarchy by applying the steps mentioned above (pruning, leafing and filtering). Next, the summary is propagated to the neighbor domains, so that they have information on the computing resources in other domains to perform the match-making between jobs and computing resources.

The *broker* is an entity, which is present within each domain and it is responsible for the creation of the summary, which is based on the Cobweb algorithm. After the summary creation, the broker handles the communication of the summary among the domains, such as the propagation and reception process. Moreover, the brokers perform the match-making between job requirements and resource features.

Match-making based on summarized information leads inevitably to a loss of accuracy and leads to incorrect referrals or undiscovered resources. For example, consider the resources depicted in Table 4.1 and the hierarchy tree depicted in Figure 4.3, for the case when the local domain (let us call it d_1) only shares the top category (depth 0) with the other domains, and no probabilities are filtered out. In this case, if another domain (let us call it d_2) receives a query asking for $< OS-name = MAC >$ and $< LRMS = Condor >$, d_2 may decide that the probability of d_1 to have resources meeting the given requirements is $P(OS-name = MAC|C0) * P(LRMS = Condor|C0) = 0.0625$ and $P(OS-name = MAC|C0) = 0.25$ (this data does not appear in the figures). If d_2 eventually chooses d_1 to forward the query to, this decision would be inaccurate, since d_1 has no resources with $< OS-name = MAC >$ and $< LRMS = Condor >$.

Although a number of other clustering approaches exist (such as k-means, quality threshold clustering, etc), Cobweb clustering is very suitable in the context of resource discovery since it is used to cluster concepts. A category utility metric is used to decide which element should be placed in which category and falls within the general area of a “category goodness” measure. Other approaches to summarization utilize the notion of domain ontology in order to find concepts that are related to each other and separated by a “semantic distance”. This work does not apply a generally used domain ontology for resource management as no generally used ontology exists in distributed systems (although there are some more specific data models such as the Glue Schema [ABE$^+$09], common information model (CIM) [For10]).

4.3 Example Scenario

Figure 4.4 illustrates an example scenario for our information provision system. In a computational Grid scenario, users want to execute their jobs, thus they have to find resources that meet a set of requirements. Examples of such requirements are the availability of particular software, processor architecture or operating system. Besides the application in a computational Grid environment, the resource discovery process could also be applied for the management of Cloud services that are deployed on a large number of resources like in [CR11].

When users want to execute a job, they ask the broker at their local administrative domain for a computing resource, and provide a set of requirements that the computing resource must meet. On receiving a query, the broker at the local administrative domain searches for a computing resource, which matches the requirements of the job among the resources at the local domain. The job is accepted for execution on a resource, if at least one resource matches the requirements of the job. Otherwise, the broker performs a query to decide which of the other domains is more likely to have resources matching the requirements of the job. In order to perform such search, the local broker uses summaries of the other domains. Depending on the level of detail of the summaries, the search is more or less accurate. It depends on the accuracy of the results if the query is likely to be forwarded to a domain that actually has resources matching the requirements of the job or not. Once the local broker has decided which of the other domains is more likely to have resources meeting the requirements of the job, the query is forwarded to the broker of that domain. On receiving the query, this broker proceeds in the same way as explained for the local broker. The summary of a domain is made of the categories at the Cobweb tree at a given depth. The higher this depth is, the more detailed is the summary. Consequently, the level of detail of the summary decides how efficiently a query is forwarded, thus deciding on the likelihood that a computing resource meeting the requirements of a job can be found and how soon this happens.

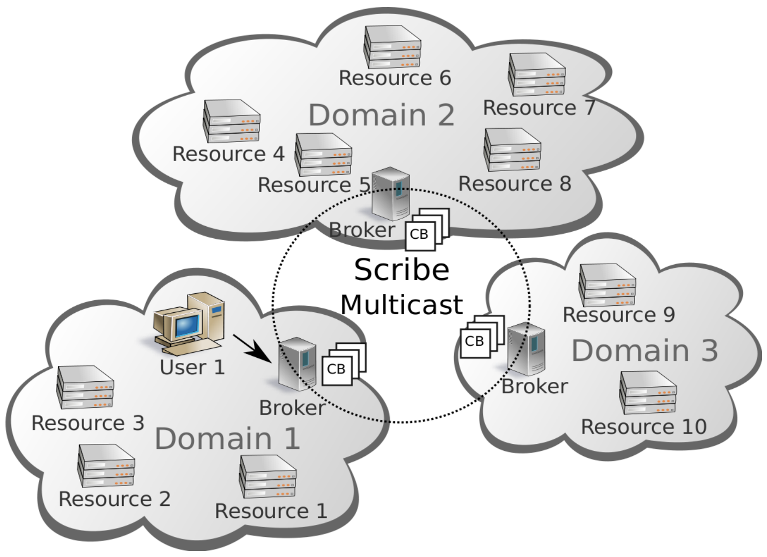


Figure 4.4: The resources in the example scenario are clustered into domains. The broker at each domain is connected to the other brokers of the other domains via a P2P overlay and disseminates the summary of the local resources to other domains.

4.4 System Design

The proposed summarization technique needs to scale in the number of resources and domains as well as it has to deal with an increasing amount of data. This thesis proposes a combination of P2P protocols to ensure the scalability of the system in terms of number of domains, and the utilization of the Cobweb summarization technique to reduce the amount of information transferred through the network.

Large-scale information systems distinguish between write-dominated and read-dominated attributes [YD04]. This chapter analyzes read-dominated data, which means that the query for an attribute is more frequent than the updates of the attributes. For example, a resource within a Grid system may have the same configuration for several days or months. However, hundreds of jobs could be submitted to it within minutes. Therefore, the main objective is to reduce the lookup costs associated with the discovery of a suitable resource to execute a batch of jobs. To ensure an effective lookup process, the proposed system provides summarized information of all domains. The summarization helps to get an overview of the system, such as knowing which type of resources are likely to exist. Moreover, the summarization helps to reduce the number of sent messages and the time required for the querying process because the brokers know which domain is likely to have a resource meeting certain requirements. Besides the setup process, the information provision is divided into two processes: the *information dissemination process* and the *information retrieval process*.

4.4.1 System Initialization

To start the initialization process, each domain has to assign at least one broker. The brokers are selected by the administrator of a domain, which can consider different criteria such as bandwidth, CPU occupation or network proximity.

Our system provides a list of peers (via a URL) for the bootstrap process of a broker (see Algorithm 1). If a new peer joins the system, it retrieves the list of peers containing `<IP, port>` and selects a peer to boostrap from. Afterwards, the new peer joins the multicast group and is able to receive the summarization trees and updates from other peers. A new peer that has joined the network sends its own summarized tree.

4.4.2 The Information Dissemination

The information dissemination process sends the files containing the summarizations of attributes (called indices) to the other domains. The indices are created with the WEKA library [HFH$^+$09]. Three different ways for the maintenance of

Algorithm 1 Bootstrap process.

```
1:  get list with bootstrap peers from URL
2:  while not connected to a peer do
3:      try to connect $< IP; port >$
4:  end while
5:  join Cobweb multicast group
6:  while incoming message m do
7:      if m = new summary then
8:          add to existing summaries
9:      end if
10:     if m = update summary then
11:         merge difference to existing summary
12:     end if
13: end while
```

the indices can be outlined:

- **Centralized:** a server with a global index over the shared information provides the querying services. However, problems of scalability and a single point of failure arise in a server-based architecture.

- **Decentralized:** a completely distribution of the nodes and indices are disseminated to all the peers in the system.

- **Hybrid**: central points (i.e. a broker of a domain) hold the summarized data of the assigned nodes.

The proposed algorithm uses a hybrid approach to harness the summarization technique, enabling summarization over multiple resources while it also provides scalability by means of decentralization.

Scribe [RKCD01], a P2P-based publish-subscribe system, is used for the dissemination process. The publish-subscribe system provides a structured multicast mechanism to send the summaries to interested peers. The advantage of using a structured publishing process in regard to a flooding mechanism is that the structured publishing process avoids the transmission of duplicated messages to the same peer [CCR05]. The Scribe protocol structures the subscribed nodes in a tree.

The randomness of the Scribe identifiers leads to a balanced tree as the tree is ordered by the identifier [RKCD01]. However, minor variances might occur during the simulation process if a tree is not totally balanced.

Algorithm 2 describes the dissemination process. The broker at each domain retrieves information of the resources of its local domain (line 3) and then creates the summary of such information (line 4) following the three steps (pruning, leafing and filtering) mentioned in Section 4.2. Afterwards, the summary is disseminated to the other domains via Scribe's multicast mechanism (line 6).

Algorithm 2 Summary dissemination.

```
1: INPUT: Cobweb threshold
2: INPUT: summary depth
3: Broker: gets information from resources
4: Broker: creates summary using Cobweb(threshold, depth)
5: for $p$ = parent or neighbour nodes do
6:    send summary to $p$
7: end for
```

The dissemination process is invoked when a new domain joins the system or when the configuration of resources changes. The *update process* follows the same principle as the initial dissemination process. To ensure the freshness of the information, an update is executed after each change of the summary. However, it has to be noted that changes of resource configuration may not affect the summary, especially if the chosen tree depth is low. For example, no dissemination is executed if the new probability of an attribute to take a value is still lower than the required threshold. Moreover, sending only differences of the summary (e.g. the categories having at least one attribute whose probabilities of taking each value have changed) reduces the amount and size of transferred messages for an update.

The dissemination of the summaries is based on Scribe [RKCD01], which uses a subscription mechanism using Pastry [RD01] as a structured DHT. The experiments use a Scribe tree which avoids bottlenecks on individual nodes and leads to an equal distribution of the transmission load to the peers. The tree structure is based on the peer ID, which is randomly generated. The randomness ensures a certain balance of the tree but at the same time the randomness leads to smaller deviations in terms of the number of sent messages and the retrieval times on each simulation run.

4.4.3 The Information Retrieval

Algorithm 3 Information retrieval.

```
 1: while incoming query to broker do
 2:     check local resources
 3:     if query matches a local resource then
 4:         reply local resource to user
 5:     else
 6:         order domains for cobweb probability
 7:         while getNextBest Domain d do
 8:             if d not already visited and probability > threshold and TTL > life-
                time of message then
 9:                 forward query to d
10:                 break
11:             end if
12:         end while
13:         if no unvisited domain with good quality found then
14:             reply to user no resource found
15:         end if
16:     end if
17: end while
```

The information retrieval process needs a higher efficiency than the information dissemination process since the number of executions is clearly higher. Therefore, the information retrieval uses a lookup process which goes through the peers with the highest probability of having resources meeting the requirements of a job, according to the Cobweb-based summary. The algorithm orders the nodes in regard to their probability, which allows a high effectiveness in number of messages and the shortest lookup time since peers with a lower probability of having a resource are avoided.

Algorithm 3 presents the way how the information retrieval is performed. On the reception of a query (coming from a user or from another peer), the peer (recall that a peer is the broker of a domain) checks the features of the resources that are available at its local domain (line 2). The query is returned to the user who issued the query if a resource matches the requirements (line 4). Otherwise, the peer must forward the query to that peer, which is more likely to have resources meeting the requirements. The other peers are ordered based on their probability of having resources meeting the requirements of the job (line 6). Afterwards, the peer with the highest probability is chosen to receive the query (line 9). On the reception of the query, the peer acts following the same algorithm, until all the peers have been visited (line 14). The time-to-live (TTL) limits how many times a query is forwarded between peers, which avoids delays arising from failures. The retrieval process is stopped when a resource is found, since a user needs only one resource, which matches the requirements of a job.

4.4.4 The Network-aware Information Discovery

The network-awareness involves choosing domains which are *closest* in terms of the RTT, which is then used to filter the results from the summarization technique. The RTT is used to order the search results – so that brokers which are located within a particular domain with an RTT below a particular bound are selected. The latency between resources that co-exist within a domain is assumed to be negligible compared to the latency between resources of two different domains. For instance, when placing jobs J1 and J6 in Figure 4.5, the shortest chosen path is

$$min(\sum(RTT(res(J1), res(J6)))$$

. Note that the RTT is used as a basis for evaluating the total time to transfer particular sizes of data between jobs.

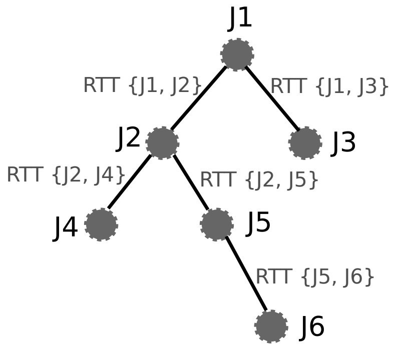


Figure 4.5: Job sequence with dependencies.

The summary-based resource discovery is extended to consider dependencies (and the critical path) between jobs identified in the workflow. The objective is to both increase the accuracy of matched resources and to discover resources that are *close* to each other in terms of RTTs.

The network aware resource discovery is explained in Algorithm 4, which considers network delays to perform the resource discovery. The algorithm works in the following manner. The broker receives a list of jobs, each of them with a certain set of requirements. For each job in the list, local resources are checked, and a reference to a resource is sent back to the user in the event that a resource matching the requirements of the job is found (line 2).

Otherwise, an information retrieval process is started over the other domains in the system, similar to the process presented in Algorithm 3 with an extension for network-awareness. So, not only the probability of the domain having resources that meet the requirements of job is taken into account, but also the RTT between domains. More precisely, domains are ordered based on their quality (line 6), and subsequently the domain with the lowest RTT is chosen to execute the job (line 10). At the end, the algorithm returns a list of resources which have been chosen to execute all the jobs (line 14).

4.5 Evaluation

This section presents evaluations of the two processes which are necessary for an efficient and fast information provision. The first process is the information dissemination, which spreads data about resources existing in each domain (its evaluation is presented in Section 4.5.1). The second is the information retrieval process, which involves generating queries looking for resources meeting a set of requirements (its evaluation is presented in Section 4.5.2).

4.5.1 Summary Dissemination Costs

The experiments evaluate the scalability in terms of the reduction of the network consumption of the proposed dissemination process of the summaries. The evaluation includes up to 1 000 domains, each containing approximately 500 resources. The 500 000 resources are randomly assigned to the domains. Each domain has to disseminate its summary, which is based on the Cobweb clustering. The dis-

Algorithm 4 Network-aware resource discovery.

```
Require: JOBLIST
 1: for all (jobs in JOBLIST) do
 2:     check local resources
 3:     if query matches a local resource then
 4:         send local resource reference to user
 5:     else
 6:         order domains with quality where quality $>$ threshold;
 7:         minRtt = VALUE, which is higher than maximum RTT;
 8:         for all $d$ = domains do
 9:             if $d$.getRtt() $<$ minRtt then
10:                 minRtt = $d$.getRtt();
11:                 best domain = $d$;
12:             end if
13:         end for
14:         add $d$ to results;
15:     end if
16: end for
```

semination requires analyzing the summary for its size, the number of messages that are sent during the dissemination process in relation to the number of existing domains.

Figure 4.6 shows the number of messages sent per peer to disseminate summaries, for 50, 250, 500, 750 and 1 000 domains, where each domain is represented by a peer. The graph shows that the number of sent messages increases in a linear manner with an increasing number of peers.

Using Cobweb-based summarization leads to a reduction in the size of the transferred messages, which reduces the network load of the system and is an important feature to ensure scalability. Figure 4.7 shows the total size of transmitted messages in relation to a varying threshold and a tree depth of 1. Recall that a threshold means that only values of pair attributes with a probability higher than the threshold is included in the summary (the filtering step of the creation of a summary). As a result, the total message size decreases when the threshold increases. However, the increase of the summary's depth as shown in Figure 4.8 increases the total size of the summary and consequently the size of the transmitted messages. Therefore, a user-driven parameterization needs to define a reasonable trade-off

Parameter	Value
Number of total resources	∼500 000
Number of domains (peers)	1 000
Number of queries	Each peer sends one query
Matching nodes per query	Depending on query type from 3% to 95%
Cobweb Threshold	0 - 0.5
Summary tree depth	1 - 4
time-to-live (TTL)	3 hops
RTT	Gaussian normal distribution with a mean of 400 ms and a variance of 200 ms

Table 4.2: Simulation setup data.

between depth, threshold and accuracy.

In addition to the number of total sent messages per peer, the maximum time of the dissemination process is also important. A fast dissemination process avoids that information becomes obsolete, thereby ensuring that correct referrals can be made to other domains. Obsolete data would have an amplification effect for the occurrence of inaccuracies (false positives/negatives) in the resource information retrieval for a particular job. In Figure 4.9, the maximum dissemination time per KB is almost constant for all the numbers of peers studied.

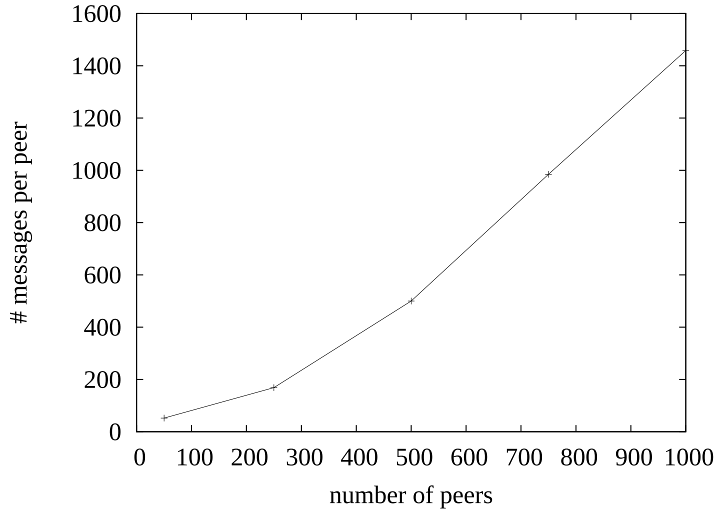


Figure 4.6: Number of sent messages per peer for the summary dissemination process.

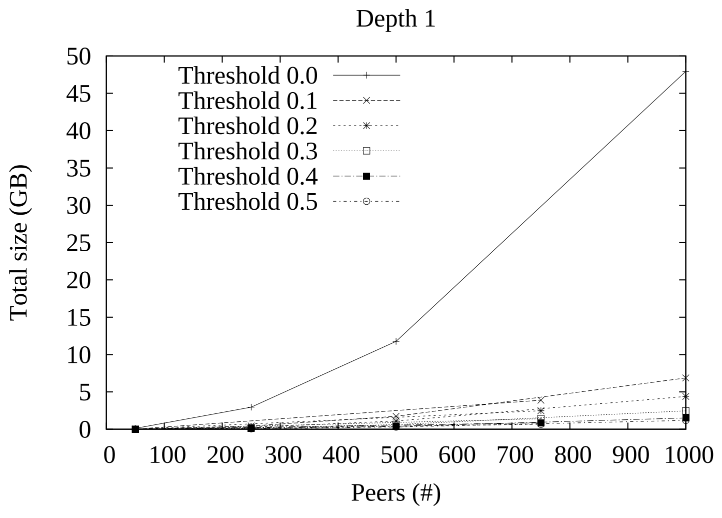


Figure 4.7: Total size of the summary for a tree depth of 1.

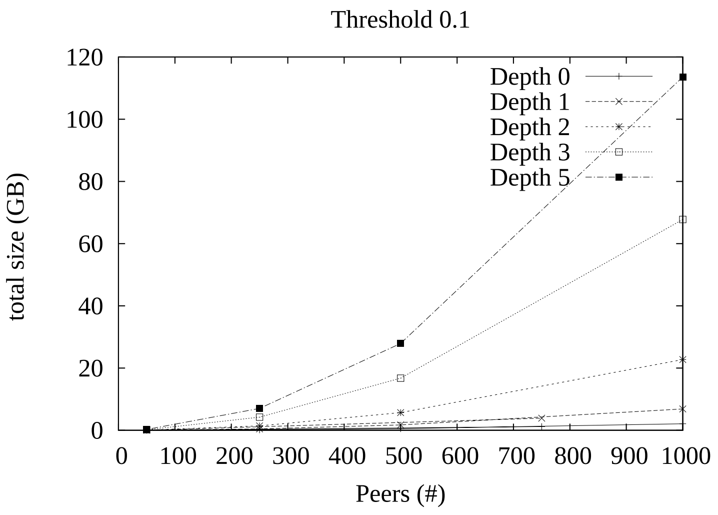


Figure 4.8: Total size of the summary for threshold of 0.1.

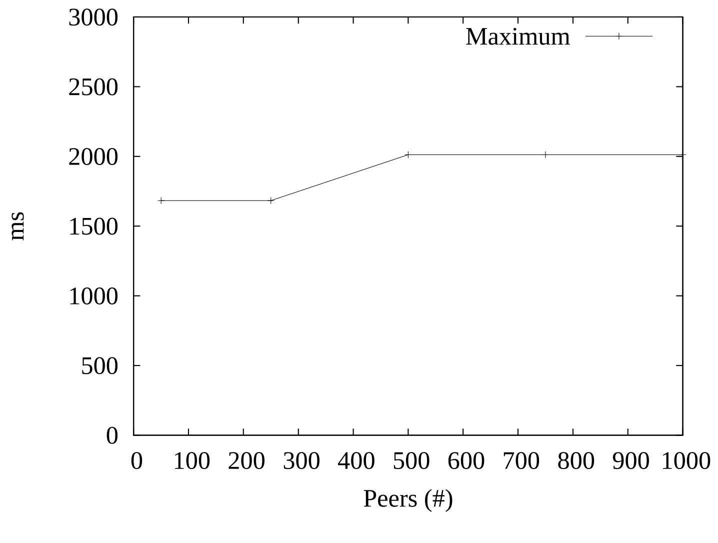


Figure 4.9: Maximum time for the dissemination of a message of 1 KB in a simulated P2P network.

Figures 4.10 and Figure 4.11 show the average size of the summaries in gigabytes (GB) for the corresponding depth and threshold. In Figure 4.10, the resources and their attributes are randomly assigned to each broker based on a discrete uniform distribution. In Figure 4.11, a Gaussian normal distribution is used. The graphs show that the summary size increases while the depth is increased. However, the summary size significantly decreases while the threshold is increased.

Comparing both distributions, the Gaussian normal distribution (Figure 4.11) presents a total size of summary significantly smaller than the discrete uniform distribution (Figure 4.10). For a tree depth of 2 and a threshold of 0.1 to 0.5, the summary size is reduced to less than half.

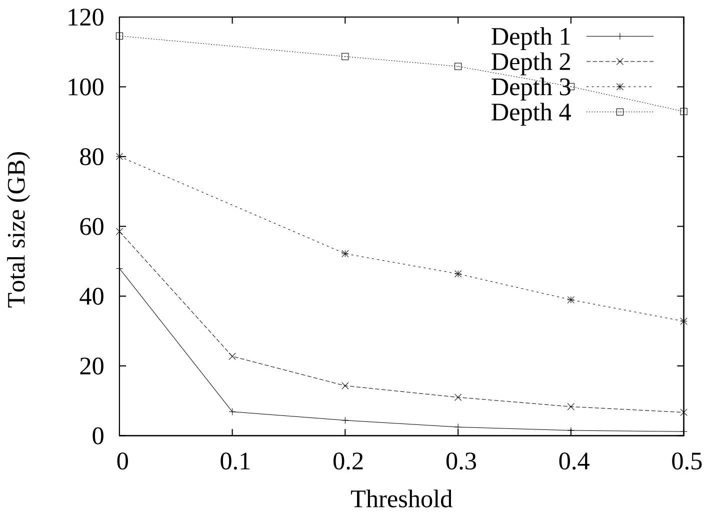


Figure 4.10: Total data of the summaries based on a discrete uniform distribution of the resources. The transmitted data is structured in an XML file, which is provided to all nodes. Each XML file contains millions of attributes.

Figure 4.12 shows the percentage of improvement per tree depth and threshold, when changing thresholds. The basis for the calculation is the average summary size per peer. The highest improvement is for a small depth like 1 and a small threshold. In example, summarizing with a depth of 2 and changing the threshold from 0.1 to 0.2 has an improvement of 45% of the message size.

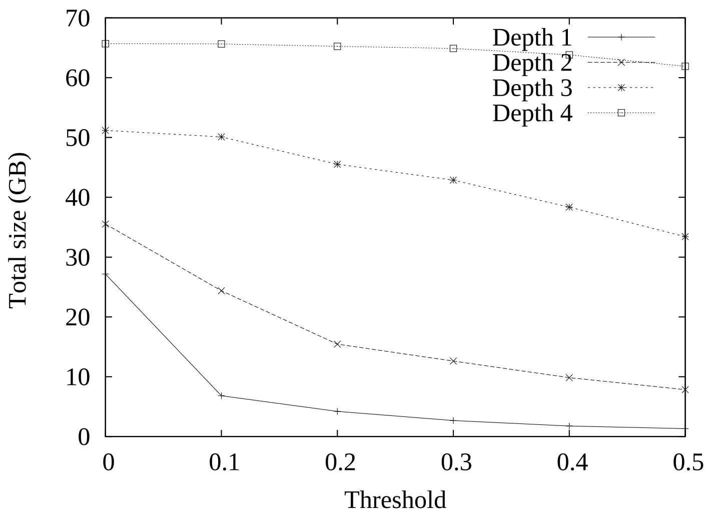


Figure 4.11: Total data of the summaries based on a Gaussian distribution of the resources. The transmitted data is structured in an XML file, which is provided to all nodes. Each XML file contains millions of attributes.

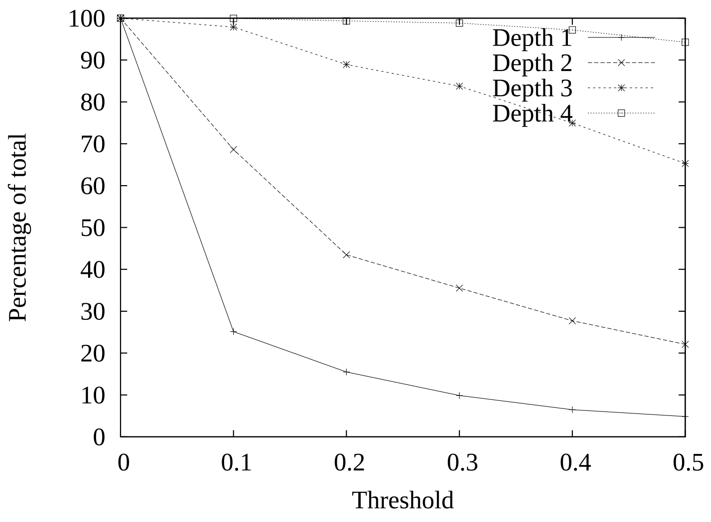


Figure 4.12: Percentage of improvement of the previous value of the threshold over the total summary size.

Attribute	Values	Count
`Architecture`	$< x64, i386, ... >$	4
`Operating system (OS)`	$< Linux, Windows, ... >$	5
`OS-version`	$< 2.6.0, 2.6.30, ... >$	3
`CPU model`	$< Pentium4, Opteron, ... >$	6
`LRMS type`	$< Condor, PBS, ... >$	4

Table 4.3: Attributes and possible values for a job query.

4.5.2 Information Retrieval Costs

After evaluating information dissemination, the overall information retrieval process is analyzed in this section. Section 4.4.3 describes the algorithm for the query process. For the setup for the evaluations generates four different query types – representing generic query types which match different percentages of resources in the system. Each query type has different constraints associated with resource attributes such as CPU, memory and OS.

Each query type is generated by a combination of the tuples presented in Table 4.3. An example query is the following:

`SELECT resource FROM domain WHERE architecture = 'x64' AND os = 'Linux' AND os-version = '2.6.0';`

The query attributes for the experiments are randomly generated, where the combination of the different query types are the following:

type 1: `CPU`

type 2: `architecture + os-name`

type 3: `architecture + os-name + CPU model + os-version`

type 4: `architecture + os-name + CPU model + os-version + lrms-type`

The combination of the presented query types leads different percentages of matching resources. These are around 97 % of resources in the system for type 1, around 55 % for type 2, around 10 % for type 3, and around 2 % for type 4. The creation of different query types allows varying the query for rare and common attributes.

Figure 4.13 shows the average number of needed hops to find a matching resource for query type 3 (around 10% of resources match) and for 1 000 queries executed. No hops are needed if a matching resource exists in the local domain. Otherwise, one hop is counted for querying each domain. The figure shows that the average hop count is lower than 1. A hop number of 1 means that when a suitable resource does not exist in the local domain, it is found only after one hop. For example, when a broker receives a query for a resource which is not matched by any resource in the local domain, the first domain to which the query is forwarded normally has a resource matching those requirements.

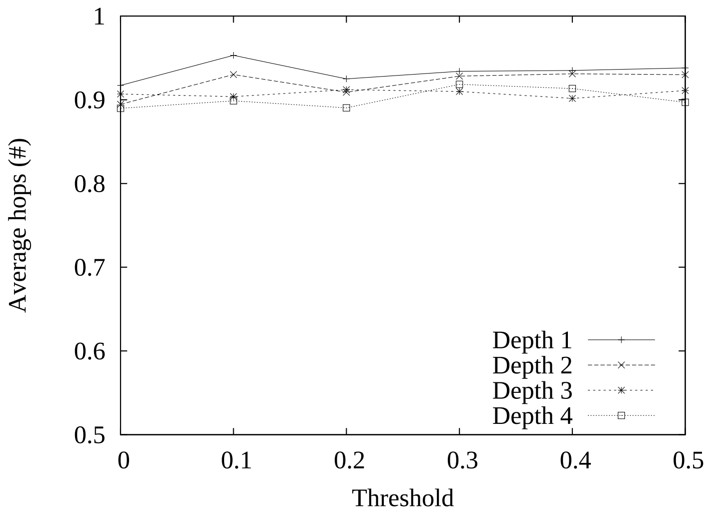


Figure 4.13: Average hops needed to find the resources for query type 3.

The information retrieval process is also evaluated by means of two metrics, called *recall* and *precision*, which is explained in the next section.

Recall and Precision

The retrieval strategy is evaluated with the *recall* metric, which identifies the set of resources obtained from the content summarization which is a subset from all resources R ($\frac{Ra}{R}$), where Ra represents the correctly retrieved documents (each document representing information associated with a particular resource). The *precision* metric measures the quality of the retrieved results, which is the set of matching documents from the retrieved set A ($\frac{Ra}{A}$). For most applications, a high precision is more important than a high recall. An example is the resource discovery in Grid systems registries where the discovery of one matching resource is already sufficient. Consequently, obtaining many resources which possibly contain false information is less desired as messages and time might be spent on looking for non-matching resources. For a real application of the system, returning only one resource for each query would be sufficient. However, to calculate the precision and the recall metric all resources are considered that are obtained by the content summarization.

Figure 4.14 presents the recall metric for the query type 2 and shows that the recall decreases while the threshold increases. The result means that not all the possible matching resources are found. However, Figure 4.15 (depicting the precision for the query type 2) indicates that the precision improves with an increasing threshold, which means that the results are more accurate. Consequently, increasing the threshold leads to fewer but more accurate results. However, increasing the threshold to a limit where the recall reaches zero matches means that no results are returned and the precision falls to zero. Therefore, a user-driven parameterization can adjust the given summarization depth and threshold. A strategy to automatically adjust these parameters is given in Chapter 5.

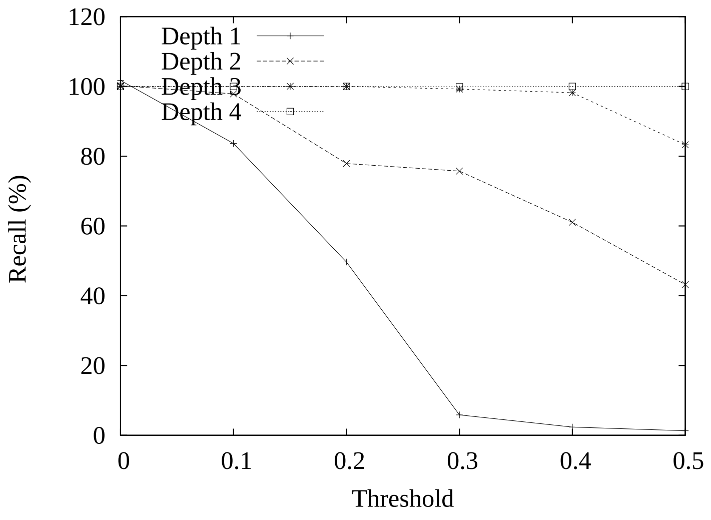


Figure 4.14: Recall depending on the threshold and tree depth for query type 2.

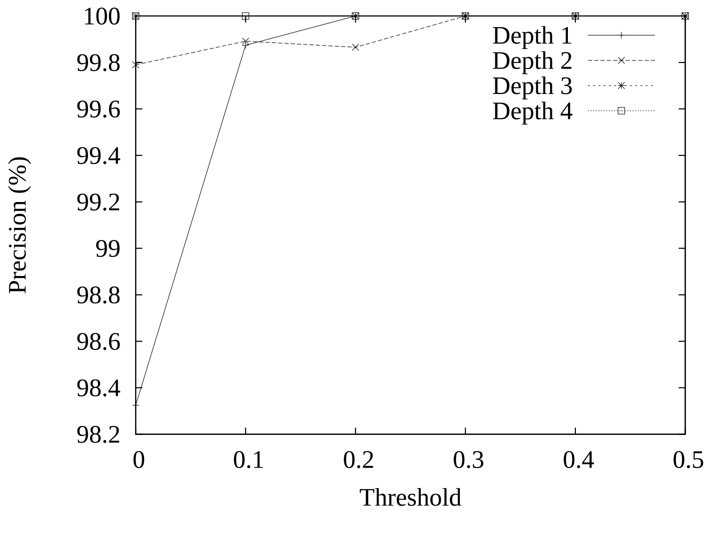


Figure 4.15: Precision depending on the threshold and tree depth for query type 2.

Figure 4.16 compares the recall to the different query types to show the difference between querying for commonly occurring resources and querying for rare resources. Considering a tree depth of 4, Figure 4.16 shows that commonly occurring resources return a recall close to 100 percent. A recall of 120 percent means that 20 percent of the results are summarization inaccuracies (e.g. the retrieval of non-matching resources), which leads to a precision of around 80 %. The precision of 80 % is shown in Figure 4.17 and the precision improves when the threshold is increased. The precision of the information retrieval is higher with commonly occurring resources, which is the same as for the recall.

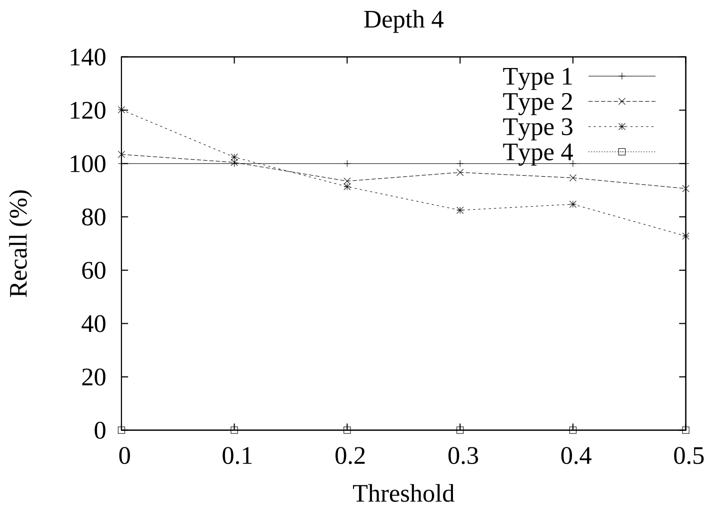


Figure 4.16: Recall for a depth of 4 with a variation of the query type and threshold size.

The ratio $\alpha = \frac{precision}{recall}$ is used for a better evaluation of the accuracy of the query response. Figure 4.18 shows the results for α with a tree depth of 2, 3 and 4 in the corresponding subfigure, for resources following a discrete uniform distribution. The query type varies within each graph of the subfigures 1-4. The subfigures show the trend that the ratio increases when the threshold increases as well. On one hand, it can be seen that the accuracy is better with a higher threshold when the sample size increases (compare to Figure 4.10). However, the risk of returning no result increases at the same time with a higher threshold. On the other side, a lower tree depth leads to a better α but the risk of returning no

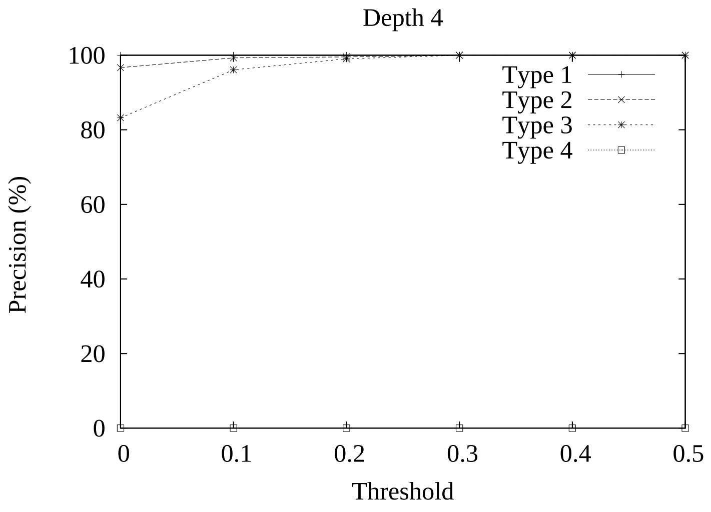
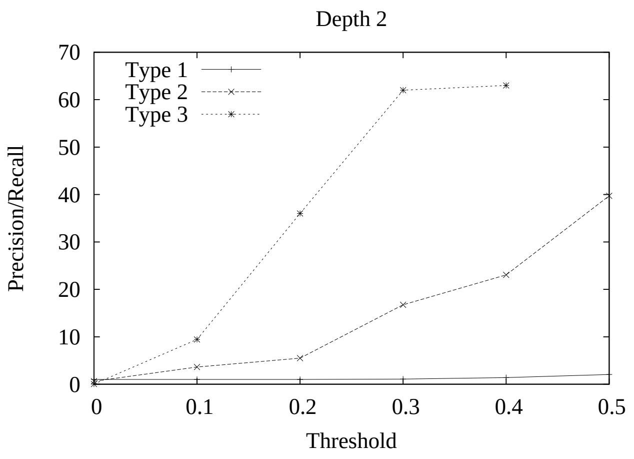


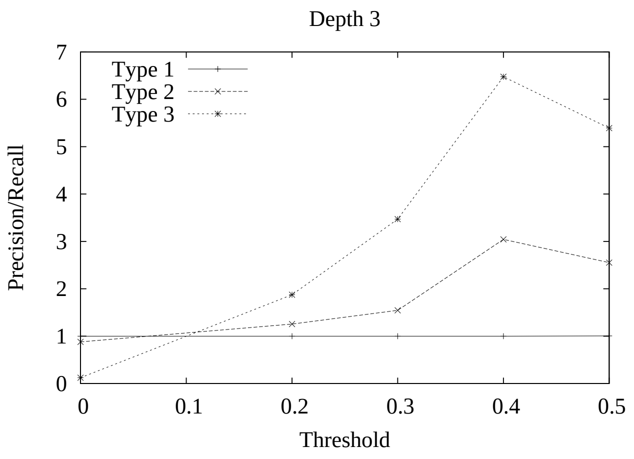
Figure 4.17: Precision for a depth of 4 with a variation of the query type and threshold size.

result fails with a higher depth.

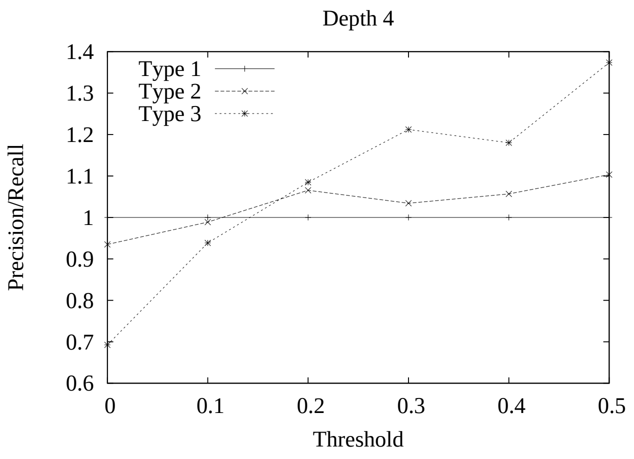
Figure 4.19 shows the accuracy of the results with resources based on a Gaussian normal distribution. In contrast to Figure 4.18, which shows the results of simulations with a discrete uniform distribution of resources and attributes per domain, the simulations with Gaussian normal distribution return more accurate data in terms of a significantly higher α ratio. The result is obtained with a smaller total summary size (In Figure 4.7 and in Figure 4.8 a smaller depth and a higher threshold leads to a smaller summary size), which means that the information retrieval based on a Cobweb clustering is more efficient with a Gaussian normal distributed data than for discrete uniform distributed data.



(a) α for depth 2

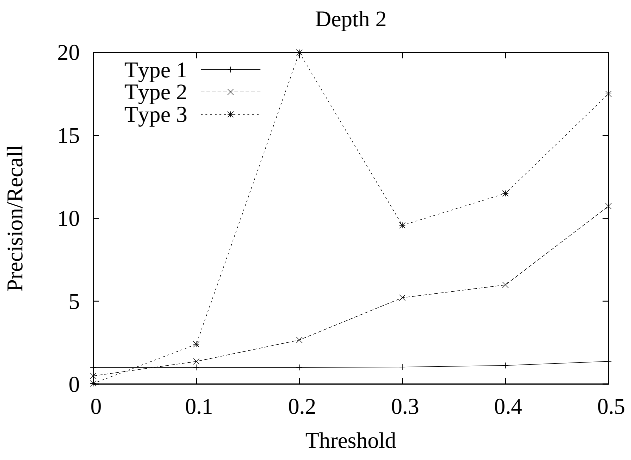


(b) α for depth 3

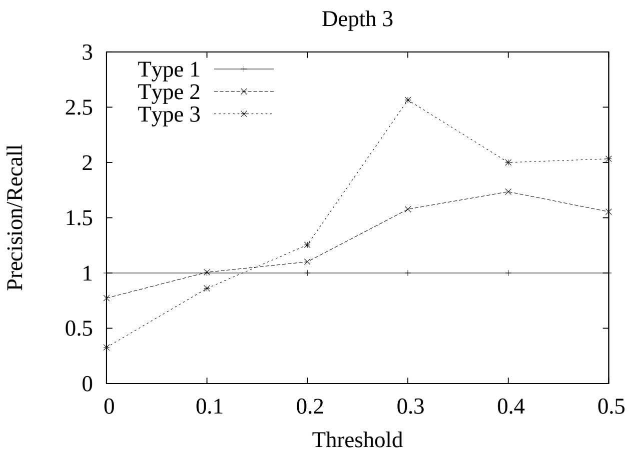


(c) α for depth 4

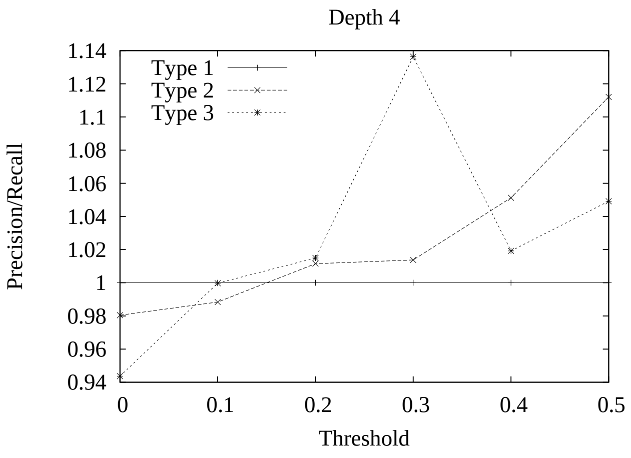
Figure 4.18: Comparing the ratio $\alpha = \frac{precision}{recall}$ for different summary depths and a discrete uniform distribution of the resources and their attributes.



(a) α for depth 2



(b) α for depth 3



(c) α for depth 4

Figure 4.19: Comparing the ratio $\alpha = \frac{precision}{recall}$ for different summary depths and a Gauss distribution of the resources and their attributes.

4.5.3 Network-aware Resource Discovery

The evaluations includes two different approaches for the resource summarization. One is the selection of a resource based on the highest Cobweb probability – referred to as Cobweb (FIFO) (see Algorithm 3). The algorithm considers the first found domain, which is returned by the broker if two domains have the same probability of containing a required resource. To optimize the accuracy of the lookup process when multiple resources are returned as a response to a query, a network awareness is introduced (see Algorithm 4) to reduce the data transfer time between dependent jobs.

Figure 4.20 compares both algorithms together with a worst case scenario and it shows that both approaches perform significantly better in terms of time needed for resource discovery than the worst case. The Cobweb (FIFO) experiments are considered as baseline results. The network awareness can reduce significantly the RTT in comparison to standard summarization-based resource discovery services.

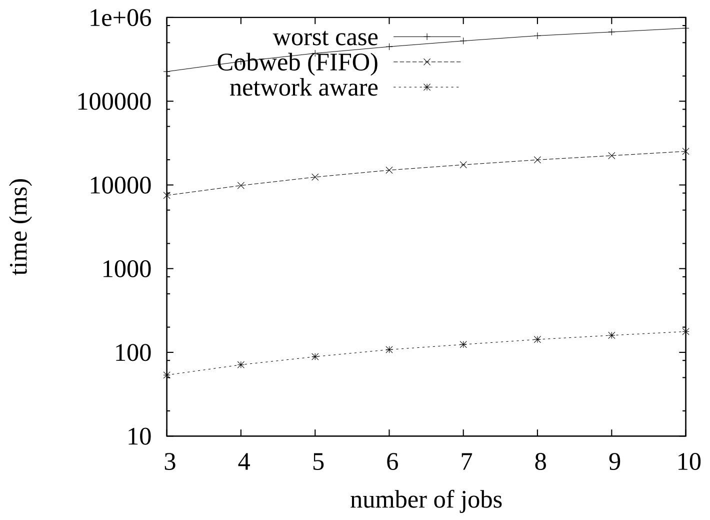


Figure 4.20: Comparing the presented Cobweb algorithm 3 and the network-aware algorithm 4 for the resource discovery with the worst case RTT within the same Cobweb threshold.

Figure 4.21 shows the average RTT for the dependencies between the jobs for a sequence of 6, 8 and 10 jobs, a Cobweb depth of 2 and a query type 2. During the experiments, the RTT stays stable while varying the Cobweb threshold. A consequence of the stability is that the threshold can be modified for an improved precision without modifying significantly the RTT for a job sequence.

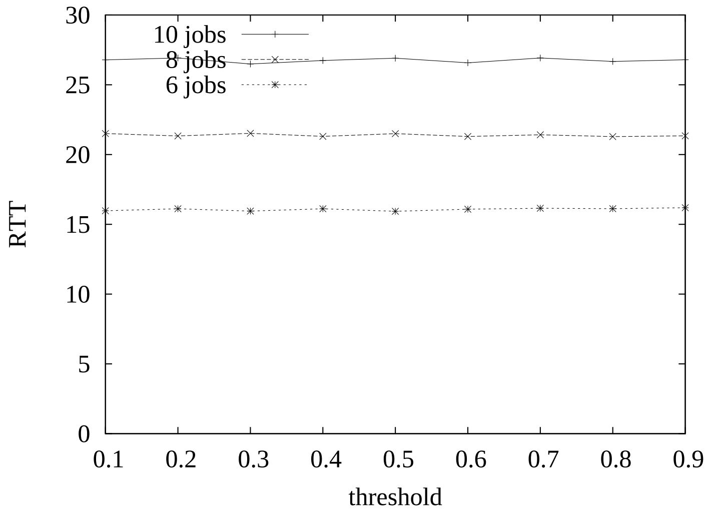


Figure 4.21: Analysis of the critical workflow path in comparison to the Cobweb threshold.

The obtained results in Figure 3.3 for the increasing transmission time enables us to multiply the RTT with a certain file size in MB – as a first approximation for determining the data transfer time. Figure 4.22 shows the transmission time for a file which is uniform randomly distributed between 1 MB and the given maximum size. The job sequence is 10, the Cobweb threshold is 0.5 and the depth is 2.

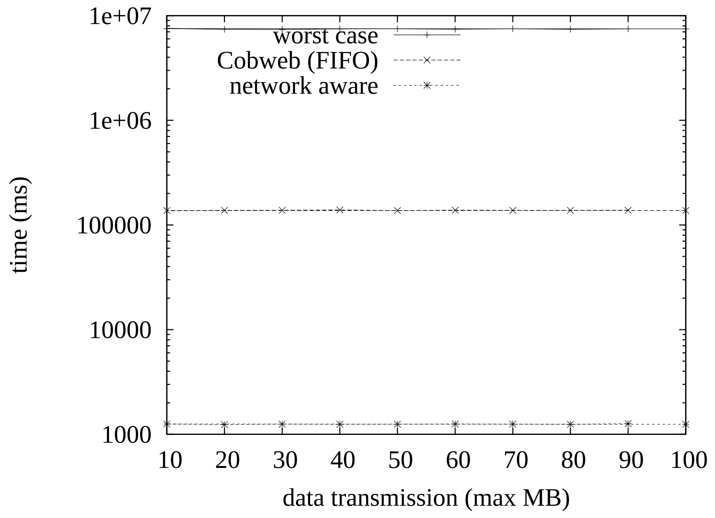


Figure 4.22: Comparing the presented Cobweb and network-aware algorithm with the worst case, where the threshold is 0.5 and the depth is 2 and the job sequence (critical workflow path) is 10.

Consequently, the results with the network-aware algorithm shows an improved network time of the critical workflow path of 0.01% in comparison to the worst case or 1% for a general summary-based scenario. In comparison to Section 4.5.2, the precision improves while the network-awareness ensures a lower RTT than a general summary-based algorithm.

4.6 Summary

The introduced content summarization is important to reduce the time for the information discovery and the network consumption. The reduction of time, message size and the number of sent messages improves the scalability of the system. However, the Cobweb-based algorithm summarizes the content with inaccuracies, which is normal for approximation techniques. Therefore, a precision recall-metric offers a user-driven trade-off between the accuracy of the resource discovery and the summarization level.

The evaluations of of the algorithms give insights for a user-driven trade-off between the retrieval time, the network consumption and the accuracy of results. The network consumption is analyzed in Section 4.5.1, the retrieval costs and the precision of the results are analyzed in Section 4.5.2. The users can decide the degree of the content summarization based on the results of the experiments. For example, a high summarization of the content leads to a low network consumption during the setup process but it leads also to a lower precision.

4.7 Discussion

Over the years, several systems have been developed for resource discovery in distributed systems [TTF$^+$06]. In Grid systems, one of the most popular is the Globus Monitoring and Discovery System (MDS) [CKFF01]. MDS allows users discovering resources that belong to a VO and to monitor those resources. However, most of the resource discovery systems are limited in their scalability. The presented algorithm improves scalability by reducing the amount of data transferred and by improving the efficiency of resource discovery.

The proposed algorithms of this section are designed for read-dominated systems and provide a very efficient way for the reduction of the retrieval time and the network consumption of the lookup process. The quality of the results in terms of accuracy is also reasonable, because the discovery of one correct resource is already sufficient. The presented algorithm reduces the network consumption of the information dissemination by using a content summary technique and by providing a super-peer model from the perspective of each resource (similar to [HRVM08]). Other possibilities for further optimization in regard to updates are out of the scope in regard to the thesis hypothesis. For example, after an update, not the total content summary needs to be distributed because the transmission of the differences of the summary file would be sufficient. Moreover, not every update of the data leads to a new summary file.

SaintEtiq represents a P2P-based summary management system. Despite using super-peers, SaintEtiq is deployed over an unstructured overlay. The usage of a structured overlay improves the performance of the discovery mechanism in comparison to an unstructured overlay used by SaintEtiq. Given that the number of hops of SaintEtiq is closer to a central index server than to a completely decentralized system, the average number of hops still increases with the number of resources. For instance, our results show that the average hops to find a resource within the system is smaller than 1 for 500 000 resources distributed to 1 000 domains and that the value stays stable with an increasing number of domains Figure 4.13. Hayek et. al [HRVM08] presented results for a flooding approach as baseline experiments, which needs nearly 2 000 messages per query within 1 000 peers and their hybrid approach needs at least several messages while it increases with the number of peers.

Caminero et al. [CRCC10] describe the use of hierarchical routing indices as a means to route jobs to particular nodes within a Peer-2-Peer system. Routing indices are used to prevent flooding the network and to forward requests to those resources that are likely to have the capability to process requests. The work demonstrates that effective use can be made of the underlying network resources when forwarding queries. The main drawback of this work is that it only considers numeric parameters (such as effective bandwidth or number of processes) to perform the resource discovery.

Cardosa and Chandra [CC10] propose a clustering technique that is based on the aggregation of resource bundles for the resource discovery in Grid systems. Their clustering technique is important to ensure a large scalability and the robustness against failures. Our algorithm differs in stopping the discovery process if one matching resource is found. In that case, the lookup process is reduced to one hop with the summarization algorithm. In contrast to their clustering technique, the aggregation of the resource bundles follows a linear increasing function in regard to the network size. Our design separates the information process into a dissemination process and a discover process. The separation of the discovery process produces promising results in speeding up the information discovery and reducing the network load, even in simulation with a ten times higher network size.

Doulamis et al. [DKDN09] identify the notion of a "semantic proximity" for locating content within a P2P network. In this work, proximity refers to a "closeness" of the nodes in terms of common interests and possessed content. The authors demonstrate that the search overhead for content is reduced by semantically close associations between the nodes. Similarity in the system is based on comparing user defined "filesets", which correspond to a representative number of files that are possessed by the user, enabling grouping (clustering) of common filesets. An entropy measure is then used to identify the difficulty of discovering a particular type of content. The entropy indicates the expected number of queries needed during the content search, with the objective of minimizing the overhead. A scheme is proposed to calculate the network partitioning that leads to the minimum value of the search entropy. Our work differs in two fundamental ways: (i) in not considering semantic relationships between nodes or their content and (i) in focusing not specifically on clustering nodes, which are based on their possessed files.

Volckaert et al. [VTL$^+$04] introduce a network-aware Grid scheduler to obtain quality jobs, resulting in low throughput time executed on network efficient resources. However, the evaluation considers only 12 Grid sites. Our work goes further in introducing network-aware resource discovery for workflows on a large scale, which is increased through a combination of P2P protocols and summarization techniques.

Besides the usage for resource discovery in Computational Grid and Cloud systems, our algorithm has a potential for the usage in visual summarization. The visual summarization like described in [ADD09, EZS$^+$09, DDK00, DD01, SCSI08] follows a similar approach like the content summarization. The visual summarization reduces the number of pixels to reduce the image size. Our Cobweb-based algorithm summarizes the information to obtain a smaller message size. The reduction of the image or message size is important for providing a faster information transmission and for reducing the network consumption. The challenge of both summarization techniques is to find a trade-off between a small message or image size (high summarization level) and the provision of a high quality of the summarization output. In visual summarization, the challenge is to summarize an image until it is still understandable by human users. Different to the visual summarization, this thesis introduces a precision-recall metric in the H2M interface. The precision-recall metric allows the users to define the trade-off in a measurable value.

CHAPTER 5

SELF-ADAPTIVE APPROXIMATION

This chapter proposes for hierarchical aggregate queries a self-adaptive trade-off among the retrieval time, the network consumption and the accuracy of the results. The system needs to adapt itself to a dynamic environment. The attributes of the application can change during run-time. Furthermore, adjusting the system before run-time would be very complicated or even impossible due to the unpredictable behavior and complexity. An algorithm, which can adapt itself to new situations and constraints to find a trade-off in a dynamic environment.

5.1 Introduction

Data management for large-scale applications such as Peer-to-Peer networks and Grid Computing gained importance over the last several years. The applications include information and monitoring systems for computational Grids and Clouds [ZS05], large-scale information aggregation systems [SNSP10][ADGK07] and distributed databases on a very large scale [CW08][RN08][CW09].

The decentralization and the scale of such applications lead to failure-proneness, delays and uncertainties. The use of distributed and heterogeneous components within many such applications also leads to important scalability challenges. As the number of nodes and data sets increases, in order to support scalability it is necessary to reduce the network consumption from an exponential growth to a linear, logarithmic, or constant factor. Approximation techniques allow for reducing the time of the information retrieval process, but at the same time approximate queries also reduce the accuracy of the results. A conflict arises when attempting to optimize time, number of exchanged messages and accuracy at the same time: reducing the network load and the retrieval time can increase the inaccuracy of the results. In addition to the contradictory requirements of these three factors, other difficulties arise due to limits of static configuration within such applications (due to the wide range of application scenarios and network conditions).

Timely and accurate results are important in the retrieval of aggregated information. It has been observed that real-time and interactive responses are considered to be more important factors than accuracy in most typical data analysis applications and data mining applications [AGP99, ADGK07, CDN07, MGL$^+$10]. Often, data analysts are willing to overlook small inaccuracies in the provided answer if the answer is delivered fast enough. The need for fast information is a major motivation for recent developments of approximate query processing techniques for aggregation queries in traditional databases and decision support systems [ADGK07].

A promising solution for a scalable system that considers both the user's requirements and technical implementation is a recent trend towards self-adaptive architectures [JPR09]. Self-adaptive systems include methods and means for reducing the human burden of managing computing systems. A challenge in the design of large-scale hierarchical aggregation systems lies in achieving a trade-off among timely information retrieval, accurate results and low network consumption [HV03, BA01]. Besides difficulties that arise from the contradictory requirements of the three factors, other difficulties arise from the consideration of a wide range of application scenarios and network types.

The contribution of this chapter is a novel self-management algorithm for a hierarchical aggregation system that offers a trade-off among timeliness, accuracy and network consumption. First, the system assesses a reciprocal matrix with the user's preferences, which feeds the criteria of the decision's alternatives. Second, the analytic hierarchy process (AHP)-based algorithm decides on each node in a self-adaptive manner about applying approximate queries, which allows adaptation to a dynamic environment. The algorithm applies approximate queries for the reduction of the retrieval time and for the reduction of the network load, while guaranteeing the required level of accuracy.

5.2 Hierarchical Approximation

This section describes the aggregation process and the applied approximation technique. After a presentation of an abstraction of the used aggregation, examples explain the hierarchical aggregation and approximation in detail.

5.2.1 Aggregation Abstraction

Aggregation allows providing a global view over the system for each node in large systems. The usage of an aggregation abstraction is in accordance with Astrolabe [VRBV03], SDIMS [YD04] and TAG [MFHH02], in order to provide scalability in large-scale information systems.

Our prototype and simulation framework assigns randomly the node IDs, which is described in Chapter 3. A consequence is an unbiased distribution of the data within the aggregation tree. In contrast to Astrolabe [VRBV03] and SDIMS [YD04], our aggregation abstraction has no administrative isolation mechanism. The administrative isolation groups the nodes of institutions and organization to a virtual cluster. An advantage of the grouping can be faster transmission among the groups as they are geographically close to each other. Furthermore, the grouping provides an easier application of privacy among the groups. However, the grouping to administrative isolation leads often to a biased data, which would turn general approximation to unrepresentative results.

The use of approximation with unbiased data is preferable to an administrative isolation. First, in a hierarchical aggregation with administrative isolation, reducing the transmission time still leads to a logarithmic increase of the retrieval time in regard to an increasing number of nodes. The retrieval time is the maximum depth of the aggregation tree, which is for example log_n in a balanced binary tree. Second, besides the administrative isolation, other techniques provide also privacy models. For example, the provision of anonymous data is an examples for such a technique [BFN08].

Our aggregation mechanism stores on the physical node the information as a set of tuples:

```
 < <attribute, value>; ...; <attribute, value> >
```

An example for such a tuple set is $<< cpu, 2GHz >; < memory, 4GB >; < diskspace, 20GB >>$. Each physical node can have many tuple sets that can lead to many matching tuples for the query on each node. In the case of several matches, the aggregated value of all values is included in the query. The aggregation function is applied to the value, if the attributes match the query. A physical node can manage several resources, which are not directly connected to the aggregation overlay.

The hierarchical aggregation follows a tree-based structure and assigns for each node and each (sub-)tree an aggregation function f [Yal05]. Each attribute defines an aggregated value $v_{i,attribute}$ where i defines the level on the aggregation hierarchy. The pairs at the leaf nodes are located at level 0 and defined as $v_{0,attribute}$; the root of the hierarchical aggregation is defined as $v_{n,attribute}$ where n is the highest level of the hierarchical aggregation (in hierarchical aggregation with balanced binary trees, n would be $logN$ and in balanced k-ary trees n would be $log_k N$). $v_{i,attribute}$ is NULL if the tuple (or the tuples of the subtree) are not matching the query, which is also the case if a node has no information about that attribute. The aggregated value for each node at level i is calculated with the node's children c as:

$$v_{i,attribute} = f(V^1_{i-1,attribute}, V^2_{i-1,attribute}, \ldots, V^c_{i-1,attribute}) \tag{5.1}$$

Our system supports the hierarchical computation property, which is applied by TAG [MFHH02] and SDIMS [YD04, Yal05]:

$$f(v_1, \ldots, v_n) = f(f(v_{1,1}, \ldots, v_{1,k}), f(v_{2,1}, \ldots, v_{2,k}), \ldots, f(v_n)) \tag{5.2}$$

where v_n is the value of an attribute at node n and $N = 1, \ldots, n$ of nodes.

The aggregation expressions that are provided by our system are: `MINIMUM`, `MAXIMUM`, `SUM`, `COUNT` and `AVERAGE`. These particular expressions are commonly used in distributed information aggregation systems like in [VRBV03, Yal05, CH06]. The count expression defines the number of tuples that match the query constraints. The query constraints filters the values to include only required information in the aggregation process.

The minimum expression is defined by the following equation:

$$MINIMUM(v_1,...,v_n) = \forall v \in V, \quad \exists v_{minimum} \leq v \tag{5.3}$$

where V are all values of an attribute that matches the query constraints.

The maximum expression is defined by the following equation:

$$MAXIMUM(v_1,...,v_n) = \forall v \in V, \quad \exists v_{maximum} \geq v \tag{5.4}$$

where V are all values of an attribute that matches the query constraints.

The sum is calculated by summarizing all tuple sets from each physical node.

$$SUM(v_1,...,v_n) = \sum_{i=1}^{n} v_i. \tag{5.5}$$

where n is the number of tuple sets that matches the query constraints.

The average value of the tuple set which are distributed over the physical nodes is calculated as:

$$AVERAGE(v_1,...,v_n) = \frac{1}{n}\sum_{i=1}^{n} v_i. \tag{5.6}$$

where n is the number of tuple sets that matches the query constraints. Our implementation divides the result of sum expression by the result of the count expression.

While the above presented aggregation functions 5.3-5.5 have a single value, the average aggregation function needs a pair of values for a distributed aggregation [MFHH02]. For example, considering f as the merging function for AVERAGE leads to the aggregation expression, which consists of a pair of value: SUM (S) and COUNT (C). f is specified by two state records $< S_1, C_1 >$, $< S_2, C_2 >$, ..., $< S_n, C_n >$:

$$f(< S_1,C_1 >,< S_2,C_2 > ... < S_n,C_n >) =< S_1+S_2+...+S_n >,< C_1+C_2+...+C_n > \tag{5.7}$$

To obtain the result of the aggregation expression AVERAGE, the SUM is divided by the COUNT: ($\frac{S}{C}$).

The query of the aggregation abstraction is based on SQL-like many other systems (see Chapter 2 Section 2.1.2). A general abstraction of our query aggregation is:

```
SELECT {aggregation (expression), attribute}
FROM tree
WHERE attribute {operator} value
```

Where the aggregation expression can be `AVERAGE`, `MINIMUM`, `MAXIMUM`, `SUM` or `COUNT`. The *tree* defines the aggregation tree in which the query is executed. The aggregation mechanisms use different trees to avoid the overcharging of root or higher level nodes. For example, SDIMS [YD04] has a tree for each attribute to avoid a single point of failure. Assigning a root node for each attribute is an improvement in regard to other existing systems but assigning a root node for each attribute would still lead to high network consumption and also to delays due to many maintenance messages. Therefore, our algorithm constructs the aggregation trees with keywords, which is similar to SCRIBE [RKCD01]. Each keyword creates a tree. The main reason for using the keyword-based root node generation is to offer a high flexibility to the system's administrator, who can choose between assigning the keyword to each attribute like in SDIMS, or assigning the keyword to a category. Examples for such a category are *high-performance computing, private nodes* or *non-profit organizations*. The WHERE clause filters the tuples which match the constraints of the query. The possible operators to define the constraints of the query are $<, \leq, =, \geq$ and $>$. An applied example is the query for the average CPU from the nodes of PlanetLab, where the free disk capacity is smaller than 2 GB:

```
SELECT {AVGERAGE(cpu)}
FROM cluster1
WHERE disk space > 2GB
```

In large-scale information systems with millions of users and attributes, it is difficult (mostly impossible) to retrieve the exact aggregation of an attribute

[Mat93, JKM+08, SN95]. Therefore, our systems would provide week consistency guarantees like in [VRBV03, YD04]. The week consistency guarantee motivates the usage of approximation techniques as the impression from approximations has a lower influence to the results if there are already imprecision resulting from the network.

5.2.2 Hierarchical Aggregation Example

Our solution uses a tree structure for the data aggregation. In the example, a binary tree reduces the maximum number of hops to the depth of the aggregation tree which is $log_k N$, where N is the number of all peers and k is the number of leaves for each parent of a tree. Consequently, the maximum retrieval time is reduced. Figure 5.1 shows an aggregation tree with a 4-bit identifier that is a simplified example for the design of our solution. The identifiers are randomly assigned to the nodes and leave spaces between the hash values within the 4-bit space unassigned. The aggregation tree is randomly balanced. Each time a new node joins the aggregation tree, the node is inserted next to the node with the most similar peer identifier. The peer identifier is a unique 128 bit hash value. The randomness of the node IDs enables that the tree is balanced [RKCD01].

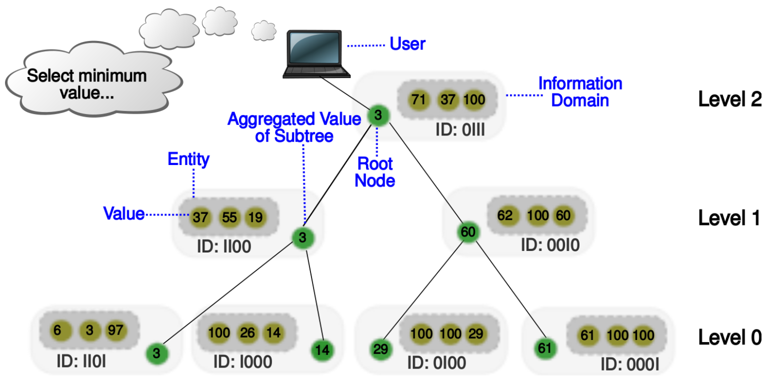


Figure 5.1: Binary tree-based aggregation mechanism with 4-bit node identifier and a depth of 3 (level 0 - level 2) as an example for the design of our solution. The values are randomly chosen real values from the busy CPU value of PlanetLab nodes.

In the example, a user invokes a query for the MINIMUM value of an attribute to the root node of the aggregation tree. The root node is the closest node to the identifier of the keyword within the 4-bit space. Afterwards, the root node transfers the query to the child nodes. When a leave node receives the query it returns the aggregated value for `MINIMUM` to its parent node. When the root node receives the aggregated value it returns the result to the user.

5.2.3 Approximation Technique

Approximate queries are commonly applied within traditional databases which contain a large amount of data. The main reason for applying approximate queries is the reduction of the retrieval time and the reduction of computing power. Moreover, large data centers have a large amount of data and the computation of complex queries (such as queries with many join constraints) could increase the available computational power. Approximate queries allow querying a subset of nodes as a *sample data* which reduces the number of retrieved data.

In the view of layered system architecture, the approximation should run on top of the distributed information or database system, which is similar to the approximation architecture Aqua [AGP99]. The difference to our approach is that Aqua is designed for a centralized database management system. A separation of the approximation techniques provides flexibility for using other approximation techniques.

The approximation function f_a extends the above presented aggregation function 5.6 for the average as:

$$AVERAGE f_a(v_1, ..., v_a) = \frac{1}{n-a} \sum_{i=1}^{n-a} v_i. \tag{5.8}$$

where $1 \leq a \leq n$. a is the level of the hierarchical aggregation and defines that all nodes which are above that level is included within the aggregation sample.

Our solution uses a probability distribution as approximation technique. The approximate queries need to retrieve a sample data on which the approximation can be applied. The sample data is a randomly chosen set of representative values. The sample data of the query are the values beginning with the root node until a certain depth of the aggregation tree. The randomness is obtained by the random assignment of the peer identifier. For example, the aggregation tree in Figure 5.1 has a maximum depth of 2 and the following values at each node: 71, 37, 100, 37, 55, 19, 62, 100 and 60. These values calculate a mean of 60.11 and a variance of 27.58. A confidence interval of 95% would return a result between 44.28 and 75.93 (the average for all values is 56.04) for the average of a value such as CPU load.

Our algorithm uses the approximation technique to collect the `AVERAGE` value of an attribute. However, the algorithm considers that similar techniques can be applied for the other aggregation operators such as `minimum`, `maximum`, `count` and `sum`.

5.3 Self-adaptive Trade-off

This section describes the AHP-based decision-making to find a trade-off among timeliness, accuracy and network consumption in large-scale information retrieval. The aim is to advance the system's ability to adapt to a dynamic environment such as a P2P network. The AHP-based algorithm applies approximations to reduce the retrieval time and the network load while guaranteeing a high level on accuracy.

The following section distinguishes between the terms *(in-) accuracy* and *(im-) precision*. The accuracy defines the degree of closeness of the measured valued to its real value within the system. The precision shows the quality of the measurement in terms of reproducibility or repeatability.

5.3.1 The Analytic Hierarchy Process

In reviewing prioritization algorithms for multi-criteria decision-making, a common method is the AHP [Saa90]. The goal of the AHP is the achieving of an information retrieval which fulfills the requirements of the users. This determination is important as the preferences can vary depending on the application type. An efficient mechanism is required to treat with a large complexity that arises from the variety of the network types that have different RTTs and reliabilities. AHP is a mechanism that helps to make a decision by reducing the human input of managing complex environments and by proposing a simplified mechanism to assess the user's priorities. Once the system is fed with the user's priorities, our algorithm decides in a self-adaptive manner about the information retrieval.

First, the AHP priorities for the pruning of the aggregation tree needs to be set up. The pruning reduces the retrieval time and the network consumption. The result is more accurate if more nodes are queried for the estimation of the result. Figure 5.2 shows the hierarchical structure of the criteria that is used for our evaluations. In our example, the decision-making process has two alternatives: *prune* and *continue*. The three main criteria are timeliness, network consumption and accuracy. The hierarchy divides the accuracy into two sub criteria: approximation imprecision I_A and network imprecision I_N. I_A defines the imprecision, which results from the error of estimation. I_N is the imprecision that arises from P2P networks. I_N is divided in two sub criteria: the time imprecision I_T and the risk imprecision I_R. I_T defines the imprecision which is caused by an obsolete data retrieval, which results from natural delays in P2P systems. I_R defines the probability that a failure delays the information retrieval in terms of a time-out.

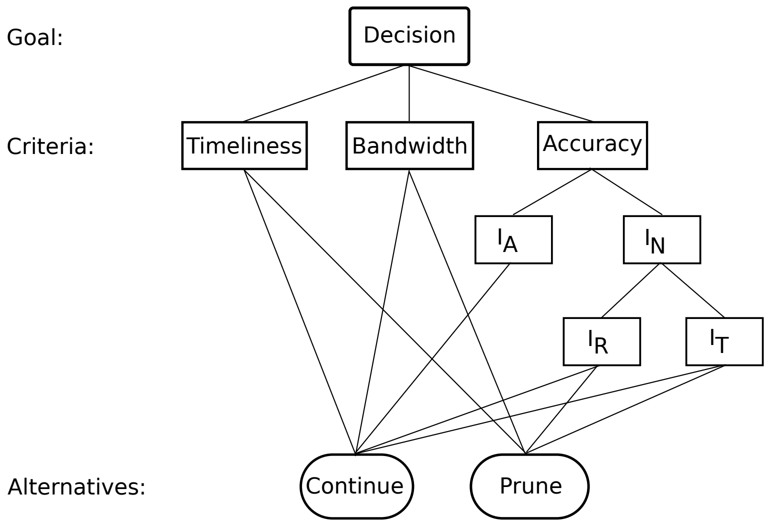


Figure 5.2: AHP hierarchy to find the decision for a trade-off among timeliness, network consumption and accuracy.

An advantage of the AHP is extensibility of the decision making criteria with additional ones. The need for additional criteria might arise from applying the algorithm in different applications or network types. An example criterion for a further extension is the consideration of peers with a high network connection or a high reliability. Such an extension could add two sub criteria to the timeliness and the users could define preferences about querying preferably fast nodes. On the other hand, considering only fast nodes could increase the retrieval time but including only fast nodes for the query could lead to biased approximation as a geographical closeness could be included. The application of further criteria depends on the individual scenario and within the scope of this thesis and are not further elaborated.

After defining the hierarchy of the criteria, the user has to judge the weight of each criterion. Therefore, a judgment value > 1 between the criteria C_1 and C_2 means that criterion C_1 has a higher priority for the user. After the user judges all combination of the criteria, the reciprocal matrix verifies the consistency of the judgment. Table 5.1 shows values that represent the case that data analysts prefer fast data instead of totally accurate data [AGP99, ADGK07, CDN07]. The user's priorities, which are assessed before executing the query, provide the values to feed the self-adaptive decision making.

	Time	Messages	Accuracy	Sum	PV
Time	1	4	2	7	0.54
Messages	0.25	1	0.33	1.58	0.12
Accuracy	0.5	3	1	4.5	0.34
Sum	1.75	8	3.33	13.08	1
Sum * PV	0.94	0.97	1.14	3.05	λ_{max}
				0.02	CI
				0.04	CR

Table 5.1: The reciprocal matrix compares the importance of the three criteria (n=3) after the users defined the priorities by comparing each criterion. The reciprocal value of the timeliness to the accuracy is 1/2 if the judgment value is 2. The timeliness has priority over accuracy if the value is higher than 1. The score of the priority vector (PV) is obtained by the sum of a criterion (e.g. 7 in the case of time) devided by the sum of the sums (i.e., 13.08). λ_{max} is the sum of $(Sum * PV)$. The consistency index (CI) = (λ_{max}-n)/(n-1).

The score of the priority vector (PV) in Table 5.1 weighs the importance of each criterion. In our example, the timeliness is the most important, which is followed by the accuracy and the network consumption. The consistency ratio (CR) is obtained by the division of the CI by the random consistency index which is 0.58 for three criteria. The random consistency index is a constant obtained from [Saa90]. The CR checks the consistency of the pair wise comparison, which is 0.04 in our example. A CR of 0.04 is reasonable in accordance with Saaty [Saa90] who affirms that the CR should be equal to or less than 0.2 to ensure the consistency. The same process to produce the reciprocal matrix is applied for each set of sub criteria.

5.3.2 Assessing the Network Imprecision

This section shows the assessment of the metrics for the calculation of the values for the previously-defined imprecision criteria. The definition of the failure rate γ is the number of failures per second from the total number of nodes within the system (events/second)[GT06]. The literature outlines two possibilities to estimate the failure rate γ, which has the range [0..1]. First, γ can be calculated from a global perspective over all nodes. Second, every node can calculate a local γ since the number of failures per node and per second (events/node/second) in a local view [CCR04]. The proposed algorithm follows the second calculation like in [CCR04, NH07] because a local self-awareness algorithm needs also a local perspective for the failure estimation. The work, which is reported in [GT06, NH07, JSG$^+$04], proposes the following metric for the failure estimation as :

$$\gamma = \frac{-ln(1-\frac{K}{M})}{T} \tag{5.9}$$

where K is the number of failures and M is the number of node samples. T defines the interval between the most recent failure and the current time.

The probability of a failure per node is defined as n ($P^n_{failure}$) [0..1]. The probability of a single node failing until the time t is defined as in [GT06, NH07]:

$$P^n_{failure}(t) = 1 - e^{-\gamma t} \tag{5.10}$$

The tendency of the variation Δ of an attribute V (e.g. memory or disk capacity) is defined by the requested value for a time interval $t_0...t_n$ (e.g. milliseconds or minutes):

$$\Delta(n) = \frac{\frac{|V_{t_n} - V_{t_0}|}{V_{t_0}}}{t_n - t_0} \tag{5.11}$$

The quality of the network is defined by the average message latency between two peers (T^{hop}_{P2P}), which is calculated with the total retrieval time divided by the maximum number of traversed hops h ($h \in 0..N$) as:

$$T^{hop}_{P2P} = \frac{\sum_{i=0}^{N} t_{i,hops} - t_{i+1,hops-1}}{log_N} \tag{5.12}$$

A network imprecision metric I_N is based on the temporal imprecision I_T which is proposed by Jain et al. [JKM$^+$08]. The metric I_N defines the imprecision, which arises from the delay starting at the occurrence of an event until the result is obtained by the source node. The $I_N(h)$ is defined at hop number h with the previously obtained metrics T^{hop}_{P2P} and Δ. Since approximate queries need to pass twice a peer (the first time to distribute the query and the second time to return and aggregate the result), the total time until the root node obtains the result includes the already passed peers h and additionally the messages to and from the leaf node $(h+2)$.

$$I_N(h) = T^{hop}_{P2P} * \Delta * (h+2) \tag{5.13}$$

The risk imprecision arises from the risk of failures $I_R(h)$ at hop h. The metric $I_R(h)$ depends on timeouts, the probability of node failure, the tendency of variation of the value of interest Δ (a higher Δ would lead to a higher imprecision in the case of a failure) and the depth h of the query. The risk imprecision has the following equation:

$$I_R(h) = timeout * P^n_{failure} * \Delta * h \tag{5.14}$$

For the timeout, Lam and Liu [LL04] proposed a timeout delay (seconds/hop) between 5 and 10 seconds. The Kad protocol [Bru06] waits for 30 seconds until it considers a node dead, with the consequence of continuing the lookup process with an alternative node.

	Range	NW	PV	Prune	Continue
Timeliness	0-10	0.1	0.56	Query's lifetime	Query's lifetime + average hop time
Messages	10000	0.00001	0.12	2^n	2^{n+1}
Accuracy	-	1	0.32	$I_A * PV + I_N * PV$	$I_A * PV + I_N * PV$
I_A	0-2	0.5	0.17	I_A calculated with 2^n	I_A calculated with 2^{n+1}
I_N	-	1	0.17	$I_R * PV + I_T * PV$	$I_R * PV + I_T * PV$
I_R	0-1	1	0.085	0	I_R
I_T	0-2	0.5	0.085	I_T with query's lifetime	I_T calculated with Query's lifetime + average hop time

Table 5.2: Normalized values for the calculation of the alternatives.

5.3.3 Assessing the Approximation Imprecision

Approximate queries are commonly applied within traditional databases that contain a large amount of data. The main reason for applying these techniques is the reduction of the retrieval time and the reduction of required computing power. Moreover, data centers have a large amount of data and the computation of complex queries could increase the available computational power. Approximate queries allow querying a subset of nodes as *sample data* which reduces the number of retrieved data.

In the view of layered architecture, the approximation should run on top of the distributed information or database, which is similar to the approximation architecture Aqua [AGP99]. The difference compared to our algorithm is that Aqua is designed for a centralized database. A separation of the approximation techniques provides flexibility for using other approximation techniques.

Our algorithm uses an approximation technique to estimate the average value of an attribute. The approximate queries need to retrieve sample data on which the approximation can be applied. The sample data of the query are the values beginning with the root node until a certain depth of the aggregation tree. The randomness in the samples is obtained by the random assignment of the identifier to the nodes.

The approximate queries cause the approximation imprecision I_A, which depends on the sample size and the probability distribution of the data. The proposed definition of I_A is used for attributes with a Gaussian distribution. However, our algorithm is flexible to use other approximations since the equation could be replaced by expressions for other distributions. I_A depends on the size of the sample data n, obtained in h hops and its standard deviation σ. The probability distribution defines $t_{(h-1,CI)}$ indicates the t-distribution, which depends on the confidence interval (CI). I_A is defined as:

$$I_A(h) = \frac{t_{(h-1,CI)} * \sigma}{\sqrt{h}} \tag{5.15}$$

5.3.4 Decision Process

The decision process balances the criteria of the alternatives in order to choose the best alternative. The following example in Table 5.2 describes the possible number of messages, which is 10 000, the total number of nodes. The maximum timeliness are 10 seconds. This large difference between the ranges of timeliness and message number requires that the criteria's values are converted to a normalized weight (NW). According to our evaluations of PlanetLab, an average RTT of 200 ms is multiplied with $2 * logN$ hops dissemination time and retrieval time that results in 5.6 seconds on average. PlanetLab has peers with a RTT of over 3 seconds that leads to set a reasonable maximum time of 10 seconds. Our evaluations showed that the maximum inaccuracies of I_A in our example are 2, when considering a minimum sample size of 8 nodes. Table 5.2 presents the NW according to the presented values. The real values of the results are multiplied by the NW and used for the decision making.

The final decision about continuing the query process or pruning the query process is based on the alternatives' score. The score for each alternative is calculated by the multiplication of the criteria's value, the normalized weight and the PV. The presented scenario contains the alternatives *prune* and *continue*; each is based on a different calculation of the scores, which is shown in Table 5.2. The calculation of the alternative *prune* is based on the current live time of the query and 2^n messages, where n is the number of nodes passed by the query. The alternative continue is based on the values which would arise if the query continues. The basis input values are the current live time added to the average hop time and 2^n messages. In contrast to the alternative *prune*, the alternative *continue* arises the risk of failures I_R since querying more nodes can include failure-prone nodes.

In the simulations presented in Section 5.4, the input values for the decision-making are based on values from related systems, from evaluations of PlanetLab and from experiences of simulations. The input values are created with a high objectivity. However, considering a vast range of applications and dynamic P2P networks, the users have their own preferences and priorities. To reach a high adaptability the AHP-based decision-making mechanism allows configuring the user's preferences and proposes a generalized algorithm for many scenarios.

5.3.5 AHP-based Pruning for Information Aggregation

The AHP-based queries integrate a decision-making algorithm that decides on each node between the alternatives of *pruning* or *continuing* the query. Each node within the aggregation tree of the query is aware of the criteria's values (e.g. I_A, I_N and I_R).

Algorithm 5 is the pseudo code for the implemented pruning of the approximate queries. The user feeds the algorithm with preferences for the individual criteria that are obtained by the input of the reciprocal matrix (line 1). Based on the reciprocal matrix, the algorithm calculates the PV for each criterion (line 2) and checks the CR (line 3). The algorithm iterates through the alternatives (i.e., prune or continue) to calculate the score of each alternative if the matrix is consistent and a new query message arrives. To calculate the score for each alternative, the algorithm iterates through all criteria per alternative. The score for each criterion is obtained by the product of the criterion's value, the normalized weight and the PV (line 9). The score of the alternatives is calculated by the product of each alternative's criteria (line 10). After obtaining the score for each alternative, the algorithm decides for the alternative with the lowest score (as the costs are minimized). If the alternative prune has the lowest score, then the query message is returned to the parent node (line 15). Otherwise, if the alternative continue has the lowest score, then the message is forwarded to the child nodes.

Algorithm 6 describes the aggregation process when Algorithm 5 prunes the

Algorithm 5 AHP-based Query (downwards).

```
1: INPUT Reciprocal matrix;
2: Calculate PV;
3: Check CR;
4: while incoming query message do
5:    for all alternatives do
6:       calculate criteria's values;
7:       initialise $scoresCriteria$
8:       for all criteria do
9:          score = $valueCriteria * normalisedWeight * PV$;
10:         $scoresCriteria = scoresCriteria + score$;
11:      end for
12:      alternative $\mapsto$ $scoresCriteria$;
13:   end for
14:   if alternative with minimum score equals 'prune' then
15:      return message with value to parent node;
16:   else
17:      send query message to child nodes;
18:   end if
19: end while
```

query process or if the maximum depth of the aggregation tree is reached. The message is sent back to the root node and the requested values are aggregated. A node waits for new messages until all child nodes have returned a message containing the information about the requested value and RTTs, or it waits until a timeout is passed. The usage of a timeout is a common technique in P2P systems to avoid longer delays after possible failures. Each incoming aggregation message of the child nodes are saved in a message list (line 2). The own value is added to the obtained message list when all child nodes have answered the query or a timeout has been passed (line 4). The nodes take the local values of the requested attribute that returned the shortest delay. Afterwards, the requested value is aggregated from the incoming values that contain the own value (line 5). After executing the aggregation process, the result is sent back to the parent node.

Algorithm 6 Aggregation (upwards).

```
1: while incoming aggregationInformation OR timeout do
2:    add messageInformation to messageList;
3:    if incoming messages $\geq$ child nodes OR timeout then
4:       add own value to messageList;
5:       aggregate messageList;
6:       return message with value to parent node;
7:    else
8:       wait;
9:    end if
10: end while
```

5.4 Experimental Evaluation

In this section, the experimental evaluation of the AHP-based algorithm is presented. The default settings for the RTT between nodes are real RTT obtained from PlanetLab. The number of peers is in accordance with other works (e.g Arai et al. [ADGK07]). For the timeout after failures, the default value of 5 seconds is used following Lam et al. [LL04]. The input data has no tendency Δ as default. The update rate α is one second. The default settings for the experiments are 10 000 peers with real RTT obtained from PlanetLab evaluations. The number of peers is in accordance with other works (e.g Arai et al. [ADGK07]). A higher number would lead to similar results since the experiments show already the effect of the algorithm with a lower number of peers.

5.4.1 Evaluation of Retrieval Time and Network Consumption

An improvement of the self-regulated pruning algorithm is the reduction of the total number of messages as shown in Figure 5.3, which shows the number of sent messages for one query in relation to the number of nodes N. The algorithm reduces the querying process from linearly-increasing number of nodes like the baseline systems to a constant factor. Furthermore, our algorithm reduces the retrieval time to a linear increase like shown in Figure 5.4. The reduction to a constant factor is reached by approximate queries that allow pruning the retrieval process by reducing the queried nodes to a subset (sample data). The size of the sample data is independent of the number of nodes and depends on the quality of the queried data (e.g. its standard deviation).

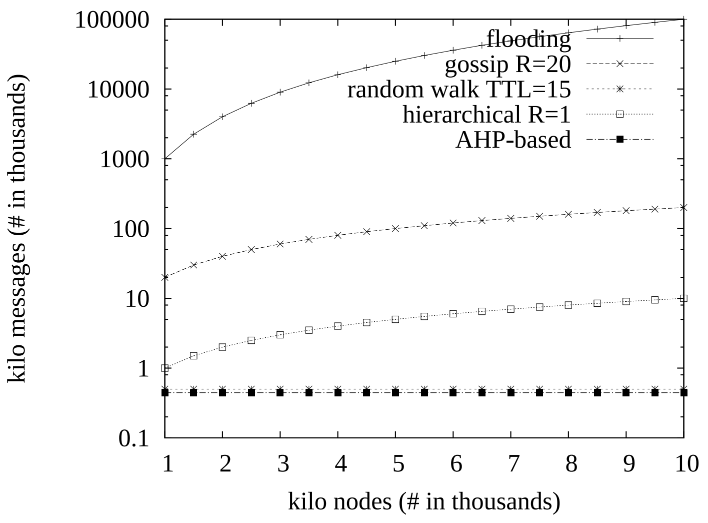


Figure 5.3: Number of sent messages in comparison to baseline systems.

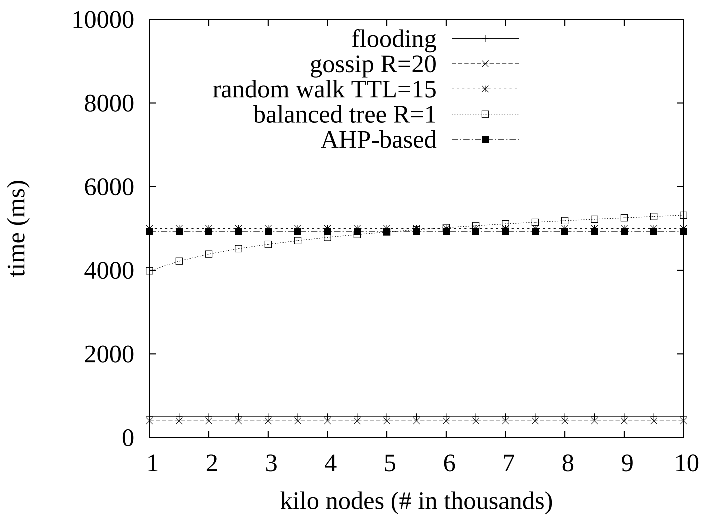


Figure 5.4: Retrieval time in comparison to baseline systems.

The presented algorithm is compared to different baseline systems. A basic flooding mechanism (e.g. Propagate2All [BGGMM04]) sends messages from all nodes to all nodes which leads to an exponential increase of sent messages. The retrieval time in a flooding environment is theoretically very low since the node transmits the new information directly to all other nodes. The used gossip protocol (e.g. Newscast [BGGMM04]) sends a message to only a subset of 20 nodes (R=replication=20). In contrast to the flooding mechanism, the gossip mechanism reduces the number of messages significantly although it still has a linear increase. The retrieval time for the gossip mechanism is constant and similar to the flooding approach. The retrieval time of a random walk (e.g. Adam2 [SNSP10]) has a constant factor for the number of sent messages and for the retrieval time. The retrieval time is higher than in the flooding-based or gossip-based information retrieval. A balanced tree without replication (R=1) like used by SDIMS [YD04] has the lowest number of sent messages and the message load is equally distributed over the nodes. However, the retrieval time follows a logarithmic increase with respect to the number of nodes. An observation is that a linear increase of sent messages can cause high network consumption in very large-scale environments. Even a short retrieval time or information dissemination time does not justify such large network consumption in regard to our algorithm which reduces both the retrieval time and network consumption to a constant factor.

5.4.2 Evaluation of AHP-based Trade-off

After evaluating the improvements in terms of time and number of messages by introducing the AHP-based algorithm, this section evaluates the balancing of the inaccuracy in this section. The results for a trade-off among the three factors of time, messages, and inaccuracy are shown in Table 5.3. The presented values are the average value (with the standard deviation in square brackets) of ten repetitions of the experiment. In the experiments, the query is for computing the average value of an attribute from its values distributed on the nodes. The first column contains the parameters that are altered to analyze the behavior of the algorithm in a changing P2P network and application environment. The first row contains the results of simulations with the default values. The experiments show that the time and the number of messages are reduced to a smaller value than querying all nodes. At the same time when the retrieval time and number of messages are reduced, the inaccuracy stays at a value of under 0.5 %.

The three rows after the default value contain the simulation results for queries

Parameter	Inaccuracy (%)	Messages (#)	Time (ms)
Default	0.44 [0.07]	420 [95]	5046 [484]
Memory usage	0.37 [0.26]	520 [135]	5120 [477]
Disk usage	0.64 [0.45]	457 [125]	4998 [422]
CPU load	1.31 [2.45]	905 [342]	5496 [625]
$\sigma = 5$	0.30 [0.03]	235 [60]	4622 [406]
$\sigma = 15$	0.55 [0.09]	536 [100]	4936 [349]
$\sigma = 20$	0.68 [0.12]	663 [106]	5166 [220]
$\Delta = 0.00001$	0.91 [0.05]	78 [10]	2850 [350]
$\Delta = 0.0001$	1.70 [0.06]	22 [1]	2078 [302]
$\Delta = 0.0005$	2.83 [0.10]	8 [0.1]	1349 [290]
$\gamma = 0.1\%$	0.39 [0.11]	451 [150]	8430 [7219]
$\gamma = 0.5\%$	0.44 [0.42]	559 [4]	23611 [234]
$\gamma = 1\%$	0.17 [0.09]	569 [115]	20838 [5980]
Poisson	0.14 [0.02]	155 [39]	3829 [0.398]
Uniform	0.16 [0.01]	216 [50]	4275 [0.294]
Pareto	0.41 [0.08]	445 [128]	4923 [0.370]

Table 5.3: The trade-off among the retrieval time, the number of messages and the inaccuracy in regard to different input values. The input values observe real data from PlanetLab and changing parameters to analyze the behavior of the algorithm to a vast range of application. The standard deviation after ten repetitions of the simulation is given in the square brackets.

using real attribute values. The real values are the memory usage, the disk usage and the CPU load that were obtained from nodes in PlanetLab. The algorithm performs with the real values similar to the default values although the values vary in terms of the distribution of the values and the standard deviation. All three factors are significantly higher for the attribute CPU than the default attribute, the memory usage and the disk usage. The difference is caused by a different distribution of the values and their standard deviation. Nevertheless, the results show that our algorithm achieves a reasonable trade-off among the retrieval time, number of messages and the accuracy for all three real attributes. After analyzing real values, the following paragraphs evaluate the behavior of our algorithm with respect to changing input parameters in order to analyze a wider range of possible application scenarios.

The next three rows contain the results of the simulations with increasing standard deviation σ for Gaussian distributed attributes. A CPU load measured by a Grid monitoring tool will have a higher σ than the price for a resource in Grid or Cloud applications. The analysis of a σ between 5 and 20 covers a range of applications. The results show that the inaccuracy, the retrieval time and the number of messages decrease with a smaller σ and increase with a higher σ. In conclusion, the number of messages and the retrieval time are low even with a higher σ and the inaccuracy stays reasonably low in regard to the user's priorities.

The next three rows compare a different tendency Δ with the default values. The values of the attributes can change over time which represents an increasing CPU load or an increasing price for a service. When the retrieval time takes longer than several seconds, the results can become obsolete before the query returns the aggregated value. Therefore, a parameter Δ defines the percentages of the value that are incremented every second. The results show that an increasing tendency increases unavoidable the inaccuracy that is caused by the obsolete results. However, the algorithm reacts autonomously to stop the query process if the querying of more nodes does not lead to a reduction of the inaccuracy. Having an increasing tendency, the earlier pruning of the query leads to a shorter retrieval time and to fewer messages.

The next three rows contain the results of the simulation with a failure rate γ. P2P networks are failure-prone, which can cause delays and timeouts. Therefore, the simulations consider a timeout of 5 seconds per failure, which was defined in Section 5.3.2. γ defines the percentage of nodes with a failure within the whole network. An observation of the results is that the retrieval time is higher, caused by the timeouts of the peers. However, querying all nodes would even cause a higher delay when considering failure on all nodes of the network. The number of messages and the inaccuracy stays reasonably low in comparison to the default values.

The last three rows of Table 5.3 show the evaluation with different distributions for the input values. The experiments use common distribution types and create the input values with a Poisson, random uniform and Pareto distribution. The algorithm provides the queried data with a reasonable trade-off among timeliness, messages, and inaccuracy. The comparison shows that the input data following the mentioned distributions are similar to the default configuration with the Gaussian normal distribution. The Poisson and random uniform distributions perform even significantly better than the default configuration for the three factors inaccuracy, messages, and time.

Table 5.3 showed that the algorithm achieves an inaccuracy lower than 1%. The inaccuracy under a higher tendency is caused by the fact that the data becomes obsolete since the real value has changed. In the meantime, the number of messages is reduced by querying less than 10% of the 10 000 nodes in all scenarios. In addition, the retrieval time is reduced to less than one fourth of the time needed when querying all 10 000 nodes.

5.4.3 Parameter Analysis

This section describes the parameterization of the presented algorithm. Figure 5.5 shows the internal scores for each alternative in regard to the number of queried nodes. The AHP-based algorithm decides between the two alternatives *prune* and *continue*. Our algorithm chooses the alternative with the lower score that fits better with the user's priorities. Figure 5.5 shows the alternative of continuing the query, which has a lower score and fewer than 400 nodes. The trade-off, which is based on the user's priorities, is reached at the intersection of the curves at around 400 queried nodes. Afterwards, the alternative of pruning the query has a lower score and consequently pruning is the better choice.

After showing the decision-making, the next figures show the values of the individual scores of each criterion per alternative. Figure 5.6 shows the score for each criterion of the alternative *continue*. The timeliness, which has the highest PV increases with an increasing number of nodes and I_A decreases with an increasing number of queried nodes. The score of the number of messages increases with a larger number of queried nodes.

The scores of the criteria for the alternative *prune* in Figure 5.7 are similar to the alternative *continue*. The score of I_A is generally higher than the score of I_A for *continue* because the sample size is smaller. On the other hand, the score of the number of messages and the retrieval time are smaller as fewer nodes are queried if the query is pruned.

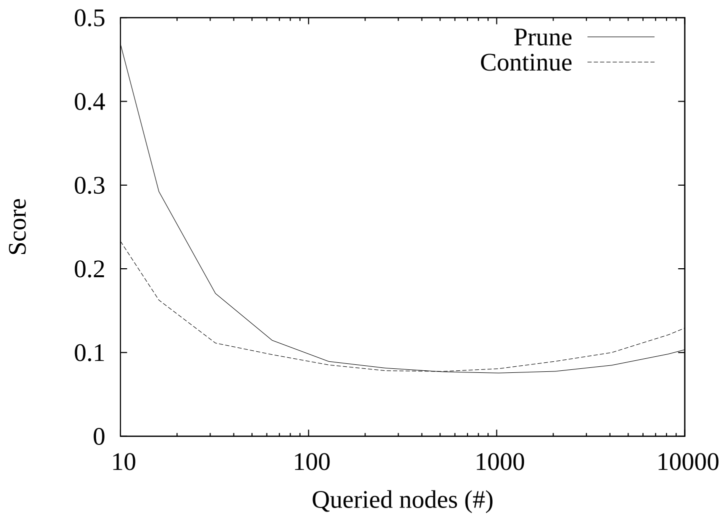


Figure 5.5: Scores of the two alternatives regarding the number of messages.

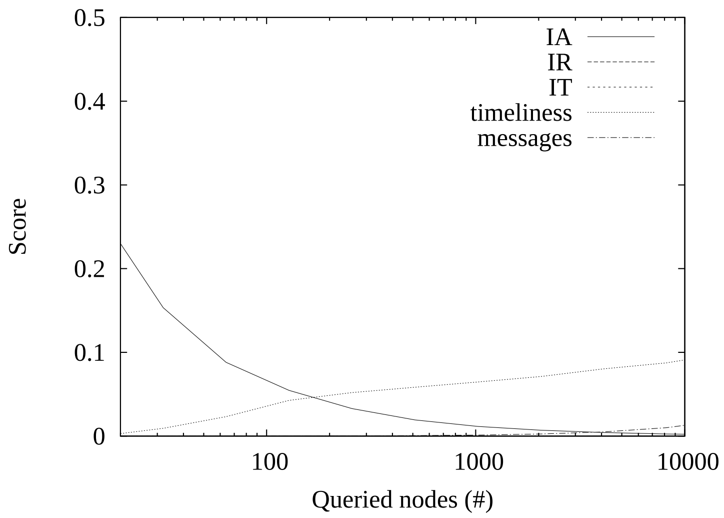


Figure 5.6: Scores for the criteria of the alternative continue.

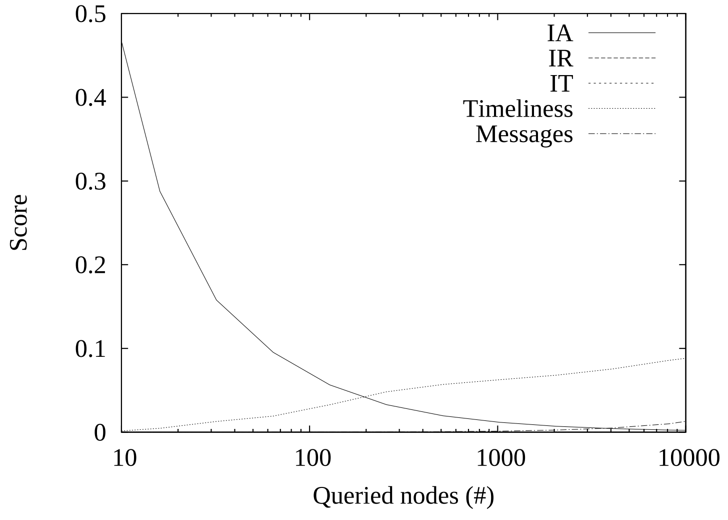


Figure 5.7: Scores for the criteria of the alternative prune.

5.5 Summary

Approximation is an important technique to reduce the retrieval time and network consumption. However, inaccuracies arise by the application of approximation techniques. Inaccuracies are already prevalent in distributed environment like P2P systems that have to scale to thousands or millions of users. Therefore, this work proposes an AHP-based decision-making algorithm, which offers a trade-off among the retrieval time, the network consumption and the accuracy of the results. The proposed AHP-based decision-making allows assessing the heterogeneous preferences and priorities of the large-scale environments. On the one hand, the consequence of the new self-adaptive algorithm is a significant reduction of the retrieval time and the network consumptions. On the other hand, the AHP-based algorithm guarantees autonomously that the inaccuracy stays within a controlled relation to the gains in time and messages.

In contrast to the baseline systems, the number of sent messages is reduced from an exponential increase to a constant factor (see Figure 5.3). For instance, the number of sent messages is reduced to less than $\frac{1}{100}$ in a system with 10 000 nodes. This reduction of the network consumption is important to guarantee a large scalability. The algorithm reduces the retrieval time to a constant retrieval time (see Figure 5.4), which is a critical factor for many distributed applications. The results are achieved by excluding many nodes from the query process but the approximation checking guarantees a high accuracy while meeting the user's priorities.

A consequence of reducing the retrieval time and reducing network consumption by approximate queries is a decreasing accuracy of the results. However, the AHP decision-making process guarantees a high accuracy because it allows an assessment of human preferences to provide a self-adaptive decision-making for complex and dynamic environments. The results show that the AHP-based algorithm guarantees a reasonable accuracy (see Table 5.3). For instance, the algorithm guarantees an inaccuracy lower than 1%. The level of accuracy is defined by the user's priorities, which allows the algorithm to adapt to different application properties and network environments.

5.6 Discussion

The main problem in providing a trade-off between timeliness, number of sent messages and accuracy is the exact assessment of the users' preferences and priorities. As described in Section 1, the different requirements for the trade-off depend on the application type, kind of users, or network types. The variety of the requirements leads to the need for fast results, accurate information and a low network consumption. In addition to the conflicts among the requirements, a user might prefer a ratio of the requirements such results that are fast, but not so fast as to be inaccurate. Existing works [SNSP10, ADGK07, JKM$^+$07b, MMKG06] focus rather on the approximation or self-adaptive aspect of the information retrieval mechanism but does not consider user priorities and fine grained tuning of the preferences.

In large-scale information aggregation systems, to our knowledge only a few works focus on offering an autonomous trade-off among a fast information retrieval, a low network load and accurate results. STAR [JKM$^+$07b] presents a self-tuning algorithm that adaptively sets numeric precision constraints to accurately and efficiently answer queries by excluding unreliable nodes. The authors show the improvements of the communication costs in terms of the number of sent messages. Furthermore, the retrieval time as a important metric to improve the scalability in large-scale environments. A consequence of the pruning of certain subtrees from the aggregation tree is a reduction of the number of sent messages. However, the pruning of subtrees does not guarantee the reduction of the longest path (i.e., reducing the maximum depth of the aggregation tree). In contrast to our work, the approximation applied by STAR has a biased sample selection since they exclude complete administrative domains. Furthermore, this chapter introduce a user-driven metric to define the level of inaccuracy. The objective of STAR is the reduction of the communication costs of the information retrieval. However, users or systems might have different preferences for a trade-off between the accuracy, the number of messages, and the retrieval time, which means that they might prefer more accurate data at the cost of more transferred messages.

One of the major observations is that the retrieval time and network consumption can be reduced significantly by applying approximation techniques. Approximation allows for reducing the queried nodes and estimates the result from a sample set of nodes. A challenge is to guarantee a reasonable level of accuracy when approximations are applied. Our results show that a reasonable accuracy can be reached under various conditions, even under delays due to failures and dynamic data (see Table 5.3).

The incompatibility between the requirements (i.e., fast information retrieval low message cost and accuracy) requires an algorithm which is flexible to different users' preferences and can provide a fine tuning of the preferences. Moreover, the algorithm needs the ability to assess the users' priorities in a precise manner and transform them to the technical implementation. Our results show that the presented algorithm can extend related systems such as from Arai et al. [ADGK07] Massoulié et al. [MMKG06] and Sacha et al. [SNSP10]. These works motivate that the timely information retrieval is more important than the accuracy. Their approach provides a technique to reduce the retrieval time; however, they do not address the difficulty of fulfilling the wide range of user and application requirements. Introducing the AHP-based algorithm is a step towards the assessment of users' preferences that they feed automated processes.

The AHP-based approximate queries can be used for a wide range of applications, such as large-scale aggregation systems, P2P databases and distributed data-mining applications[1]. Approximation techniques are already successfully applied in traditional databases that are deployed on a central server. Therefore, following Sacha et al. [SNSP10] and Arai et al. [ADGK07], approximations are a promising technique for information retrieval in decentralized databases. The delays and network imprecision in large-scale distributed systems [HPR90, JKM+08] are reasons for a more important role of approximations for decentralized applications than centralized databases.

[1]http://www.distributeddatamining.org/

CHAPTER 6

INTEGRATION TO GRID MARKETS

The objective of this chapter is to combine the results and solutions provided by the economics research community with solutions proposed from the research community of distributed computing in order to evaluate an integration study of a Decentralized Market Information System (DMIS) into distributed market frameworks. There is no general approach focusing on the development and integration of a decentralized market information system, which underlines the need for new solutions or adapting existing solutions. Therefore, the study in this chapter addresses these demands and provide economic information in distributed systems. The integration study is a qualitative analysis of the requirements, the integration process and the functions. This chapter also supports the motivation of the quantitative analysis of the previous chapters.

6.1 Motivation

Grid and Cloud Computing gained on popularity in research and in industry. Prominent examples are Sun Microsystem's Network.com, Amazons Elastic Compute Cloud (EC2) and its Simple Storage Service (S3) and SimpleDB Service. The companies frequently offer a fixed pay-per-use price for static resource configurations. Fixed prices can lead to inefficient utilization, little or no reward and bad usability, as fixed prices do not reflect the dynamics of the market's supply and demand. Efficient provision and usage of computational resources as well as pricing in environments like computational Grid and Cloud is not manually manageable. Such processes should be automated with no or minimal human interaction. Hence, market mechanisms and strategic behavior play an important role for the design of the environment. Self-organization and automatic adaptation to the changing market conditions are key propositions for efficient offering and consuming of computational resources.

An efficient allocation of consumers' jobs to providers' resources is a complex task, which includes agent decisions on resource provisioning, market-based allocation and information processing. Moreover, the wide heterogeneity of computational resources challenges the process of finding an appropriate resource for given consumer preferences. Since demand and supply of computational resources fluctuates in the course of time, information about current and future resource utilizations and prices are often not known a priori to the participants. In this case, consumer and provider agents try to maximize their utilities by generating bids based on available information. This enables strategic behavior both on provider's as well as on consumer's side.

Information about computational resource markets is essential for sophisticated and efficient negotiation strategies of traders. There are several aspects that point out the need for a decentralized market information system provided by distributed markets. It might increase the usability for users to trade in markets. Moreover, the need for information can be seen in central markets and adopted to distributed markets.

Existing information systems for markets prove the need for a market information system. The New York Stock Exchange, the marketplace with the highest volume, shows the existence of various stock information systems. For example, Google Finance[1], Yahoo! Finance[2] and Reuters Stocks Information[3] provide a free information service. Some institutions offer additional payable services like Stockgroup[4]. All mentioned services base on a central market.

[1]http://www.google.com/finance

[2]http://finance.yahoo.com/

[3]http://www.reuters.com/finance/

[4]http://www.stockgroup.com/

6.2 Requirements

This section outlines requirements for the DMIS. First, requirements are conducted from the demands of economic information on a market. Second, time-sensitiveness introduces new categories of requirements.

6.2.1 Economic Requirements

The analysis of the economic market parameters are based on different work and resources of the economic field [GS03] [BS99] [BV07] and suggests us to classify the parameters in the following categories. According to the level of aggregation the categories are divided into *single* and *aggregated* parameters. Depending on their complexity they are divided into the four categories: *basic*, *composed*, *complex* and *comments*. *Basic* parameters are simple values like the price, identified by its currency or the quantity measured by units. *Composed* values are constructed by two or more basic values. *Complex* values are more sophisticated economic measurements in the sense that they need to be computed from several composed values. Finally, the *comments* are user generated information like reviews. Table 6.1 summarizes an extract of required parameters classified into the proposed categories.

Table 6.1: Economic market parameters classified depending on their complexity and aggregation.

Category	*Single*	*Aggregated*
Basic	price, quantity, reputation, quantity, distance, budget, shout, offer, free capacity	maximum, minimum, average
Composed	volume, variation of values, trader payoff	top/flop, standard variation
Complex	Price-Earnings Ratio, identical seller/buyer	ROI, Pareto efficiency
Comments	Expert reviews, advertisements, recommendation of alternatives	

6.2.2 Temporal Information Occurrence

The time-related occurrence of the information is an important aspect for the technical implementation. The information might be made persistent for later requests or the information system has to inform the traders immediately after the occurring of a market event. The economic information is differntiated in regard to their time sentivity in the Table 6.1 in the categories *time-sensitive*, *current* and *historical* information:

- **Time-sensitive information** is pushed from the market to interested traders. This has to *subscribe* a notification service which sends a message after an occurring event. A new event is for example the entering of a new product into the market or a price fall of over 20%.
- **Current information** describes the actual state of the market and should be pulled by the traders with a *query*. For example requests for the minimum price of a product or for the average volume of traded products can acquire actual information.
- **Historical information** is mainly *archived* data. Price charts for a certain product about the last 6 month or statistics about the behavior of participants like the preferred trading time of the agents belong to this category.

6.3 Decentralized Market Information

Auction-based and bargaining-based distributed marketplaces require an economic information provision to enable fair and equilibrated prices. Examples for such markets are non-centralized trading places for computational Grid and Cloud services like envisioned in the projects Grid4All [KNB+08, LVB+08] and SORMA [NAB+08, MT10, BCRL09, CLB+08]. Another scenario is the application of a market information system in socially oriented marketplaces like [CBCR11]. These trading places enable resource providers and service providers to sell their products such as resources or computing services on a computational Grid and Cloud market. However, the buyers and sellers need to obtain information about the market in order to optimize their trading strategy, which mostly results to higher benefits.

Figure 6.1 shows the overlays of the DMIS. Using several layers can provide a higher flexibility and an easier maintenance for modifications [CCR05]. For example, different applications have different attribute types which need a layer to convert the query of the application to the query provided by the information system. A user sends a query to the application layer, which converts the query and forwards it to the DMIS. The DMIS aggregates the requested value by querying the application clients for the data. Therefore, the DMIS uses functionality from the overlay.

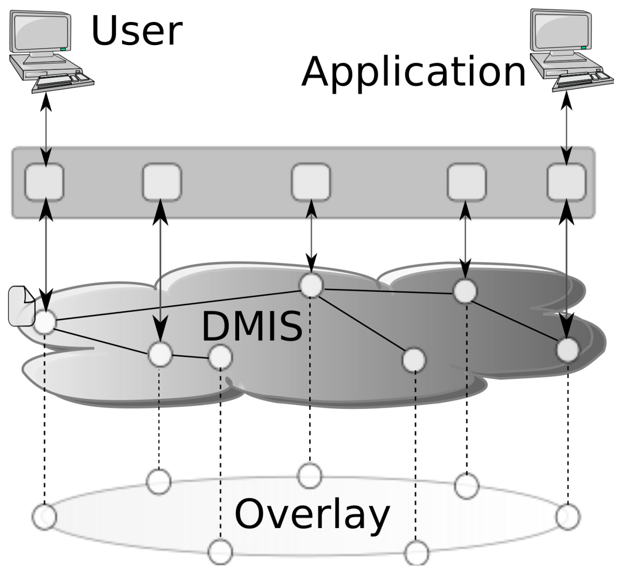
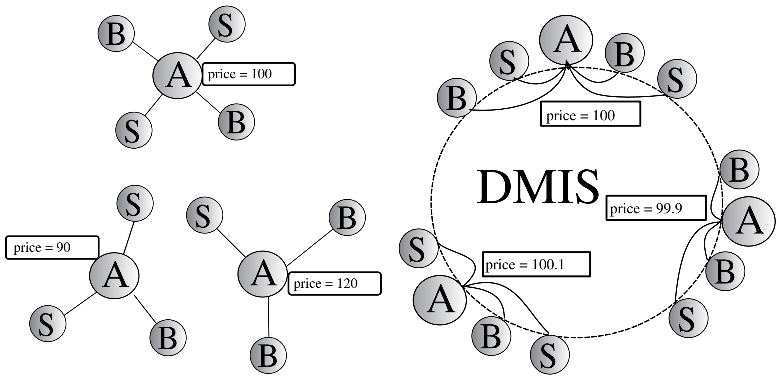


Figure 6.1: The overlay layers.

Figure 6.2 (a) shows a scenario, which motivates the use of on economic information system like the Distributed Market Information System (DMIS). Coordinated by auctioneers, the sellers and buyers trade on different marketplaces. An auctioneer uses for example an English Auction or a Continuous Double Auction (CDA). The separation of the marketplaces leads to an interruption of the information flow. More reasons for such a separation of markets arise from different currencies, geographical locations, privacy, and trust constraints or political aspects.

The integration of the DMIS and a market (see Fig. 6.2 (b)) enables an explicit information exchange among all participants. Traders can now obtain information



(a) Without global information. (b) DMIS information provision.

Figure 6.2: Possible trading places containing auctioneers A, buyers B and sellers S.

from other traders or directly from every auctioneer. Alternatively, an auctioneer can be distributed on several nodes, depending on its type and implementation. Interested participants could execute queries for certain values or could subscribe to new events such as the arriving of new products or concluded trades.

6.4 Integrated Model: Information and Markets

This section describes the integration of the framework in Figure 3.1 into a broader market-based resource allocation system.

6.4.1 Integration with SORMA Framework

Bidding strategies and prediction techniques require market information in order to bid efficiently in selected markets. However, in distributed environments like SORMA [SOR08] such information is not locally accessible. Bergemann and Valimaki [BV02] demonstrate the importance of economic information disclosure and show the increased attention paid to economic information acquisition. Traders require information that allows deducing entry prices for available markets and trading times. Centralized markets are able to furnish the current information, which is necessary for simple or sophisticated bidding strategies - such as ZIP agents [PvT98] or human traders. Therefore, bidding strategies are an issue for distributed and segmented markets. Distribution and segmentation of markets result in loss of information such as prices, products and effective supply

[BFN08]. An efficient information system should allow participants to choose a compromise between exact global information and partial information.

Bidding strategies need information about a state of the market, commonly through the offered resource type and price dynamics in time. The main focus here is the treatment of market information by agent's bidding strategies applying statistical price prediction techniques. Statistics are used in markets to measure current conditions as well as to forecast financial or economic trends. Indicators are used extensively in technical analysis to predict changes in stock trends or price patterns. Economic indicators quantifying current economic and industry conditions are used to provide insight into the future demand for commodity goods.

An integrated architecture for market-based allocation for computational resources is presented in Figure 6.3. The components of the market are clearly separated to achieve an economically efficient allocation of applications to needed computing resources. Different independent systems [BW08] [BFN08] allow a clear separation of code, functionalities, easier development, maintenance and fault detection. The buyer agent, the market information system, the seller agent and the resource manager are the main components, building an infrastructure for market-based allocation of computing services.

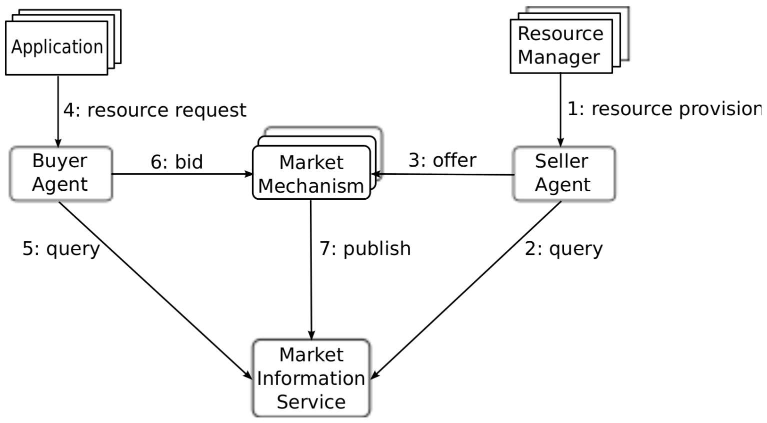


Figure 6.3: Integration to the market framework of SORMA.

The usage of the proposed model in a real application is described in detail in [NAB$^+$08]. The application component expresses batch jobs or real (web or desktop) applications, which has to be executed on demanded computing resources. Therefore, the consumer of the application submits a resource request to the buyer agent. On the other side, the resource manager manages the resources of the provider and is responsible for the execution of the allocated applications. In order to offer a free resource, the resource manager submits a request to the seller agent for the resource provision.

The bids are submitted to the Trading Manager (TM) component, which implements and runs market mechanisms for technical and economic matching. The TM is a platform, which defines the interfaces and rules for implementing market mechanisms and the conversation protocol. The bids and offers (e.g. simplified example is a CDA for CPUs with 2GHz), which are generated by the seller and buyer agent, are submitted to the TM via well defined interfaces of the selected market mechanism. When there is a match, the agents receive and informs the consumer or provider to execute its application on the allocated resource.

The Market Information Service (MIS) [BFN08] obtains economic data from the Trading Manager and provides it to the agents. The architecture has been designed to meet both the economic information requirements and that of scalability and robustness of distributed environments. Aggregation mechanisms are used to reach scalability in number of data and agent requests. Many of the introduced bidding strategies like Zero Intelligence are exploiting prediction techniques, which require public market information. The MIS retrieves and aggregates public market information from the TM to provide them to the BidGenerator.

6.4.2 Integration with Grid4All Framework

The principal issue for resource allocation is that of arbitrating resource assignment when demand and supply do not match. Straightforward arbitration policies such as those based on priorities may be abused by users if they are free to set their own priorities. Hence, mechanisms are required that provide users incentives to regulate their demand or to self-limit. On the other hand, resource owners need incentives such as economic compensation to share their resources. Pricing of resources establishes common scales of value across different resources. Market institutions such as auctions may be used to price resources and to allocate them to who value them the most.

Markets are an efficient and fair allocation mechanism and accommodate diversity in demand and supply. They are recognized as suitable for heterogeneous environments such as the Grid. Besides, markets are efficient, adapt to fluctuating supply and demand and provide feedback on prices. Our work identifies key issues in current decentralized market architectures and proposes an approach based on efficient flow of global market information that allows participating traders to react to market situations.

Figure 6.4 shows the different layers of the Grid market infrastructure and the integration of the Information Service. The bottom layer, P2P Overlay, has to deal with basic requirements of distributed systems such as scalability and robustness against failure and churn. The middle layer is divided into the market component and resources, which use the same communication framework for their interaction.

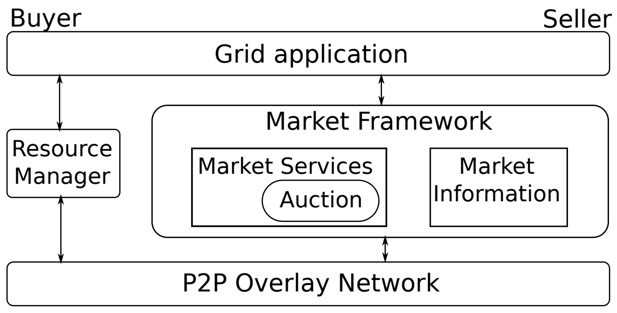


Figure 6.4: Integration to the market framework of Grid4All.

The market information service component provides information to individual market participants such as sellers, buyers or auctioneers. Thus, it uses a distributed information aggregation mechanism to handle the load and volume of participants, requests and events in a potentially large-scale system.

As a result of complex markets, the challenges might be handled entirely by the presented market infrastructure as depicted in Figure 6.5. This interaction diagram depicts a use case for buying a resource from a provider. This way, buyer agents may request from the market information component an average or minimum price, depending on its negotiation strategy. After obtaining the price from this information, it sends a bid for the resource to the Auction component. Sim-

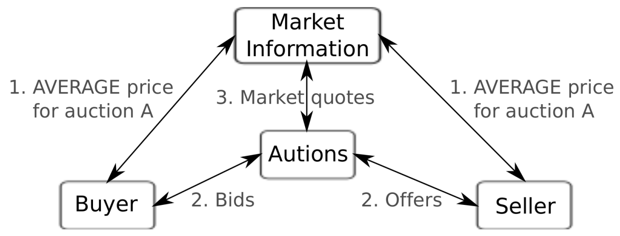


Figure 6.5: State diagram of trading resources in the Grid4All market.

ilarly, the Seller agent publishes an offer for its resources to the market. After the auction successfully matches offer with bids and sends feedback to the market information, an agreement must be reached through the Auction component. Depending on the negotiated settlement, the Auction component sends the agreement to both traders informing of the price and the settlement process begins by paying for resources.

6.5 Summary

This section described the integration process of the simulation framework to existing projects. The objective is a qualitative evaluation of the provided prototype. A successful integration in real applications is shown. Furthermore, this chpter showed that the information provision is coherent, when applied in larger projects. A demonstration on a real testbed showed the integration of the prototype to the SORMA project [5]. Quantitative evaluations showed a successful integration with the components of the Grid4All project [LVB$^+$08]. Previous chapters described the quantitative evaluation with the simulator; this chapter describes a qualitative evaluation of the prototype, which has the same components like the simulator.

6.6 Discussion

Observations (lessons learned) during the integration process showed that it is a challenge to implement Java interfaces and classes for the queries. The construction of the query constraints requires a high flexibility, which is already described by Carzaniga [Car98]. Many different query constructions have to be considered

[5]https://portals.rdg.ac.uk/sorma/mwiki/index.php/SORMA_v1.0_Videos

to cover all needs, such as range queries [AWIK06][Car98] XML-based [CFF+02] [ABC+03] or SQL-based [VRBV03]. During the time, new requirements arise requiring a restructuring or adaption of the code. To handle such a complexity, a XML-based document is used to describe the search requirements and the entry data. Using such a standard allows reducing the implementation and maintenance effort.

Another lesson learned is that the complex queries need advanced structures such as the join functions from SQL. An example is to return the standard deviation of the price from a product where the rating is equal to the median of all products that have at least five ratings. To obtain such complex queries several consecutive queries are required. In this thesis, complex queries in large-scale environments are not considered. The limits to simplified queries with the standard operators such as minimum, maximum, average, count and sum are common for very-large information aggregation systems (see Chapter 2).

Finally, a technical challenge during the evaluation with (not only) other real applications or larger-projects is the evaluation of the scalability with many thousands of nodes. First, many simulation environments are limited to a few hundred or a few thousand nodes such as PlanetLab, Emulab [6] or Grid5000 [7]. Second, the SORMA and Grid4All testbed is deployed on only tens of nodes. Also other simulations within the information aggregation, deployed on PlanetLab are often limited on a few hundreds such as SDIMS [YD04] [Yal05], which has 69 machines of the PlanetLab testbed, and on 256 nodes in the Emulab testbed. Third, real applications which have millions of users such as eMule [8], BitTorrent [9] or Skype [10] have privacy constraints and would need to force the users to update their version for the testing purpose.

The knowledge about a market is essential for the design of efficient bidding strategies. Examples are computational approaches incorporating game theory that allow predicting the future through forecasting or learning rules on former or actual trading information. Bergemann's survey [BV07] shows that the economic

[6]http://www.emulab.net/
[7]www.grid5000.fr
[8]www.emule.com
[9]www.bittorrent.com
[10]www.skype.com

aspect of information acquisition in market mechanisms got more attention by the economic research community. Moreover, the study demonstrates the importance of the economic information disclosure for market participants. The need for this information lies in both being able to apply efficient economic strategies and to feed business models, which are behind these strategies.

CATNETS [ERF$^+$05] proposes a middleware architecture to provide economic and market-based resource management. Scale is addressed by completely decentralized auctions. Their mechanism avoids the knowledge of global information such as an average price for a certain product in the market. Additional services for the regulation of choice such as global reputation and regulation of demand such as currency are missing.

A problem arising from distributed markets is the gathering of information about the market, its prices, products and the participating traders. The knowledge about the market is essential for sophisticated and efficient negotiation strategies. Examples are computational approaches like the game theory, predicting the future through forecasting or using learning rules on former or actual trading information. However, there is currently no completely researched system to provide and consult an overall knowledge of economic information in distributed markets.

Trader agents that solve either simple or sophisticated problems need at least some information about the markets or about offers from other traders. Providing this information causes no problem in a central organized auction because an auctioneer with an overall knowledge can transmit information to the trader agents. However, in decentralized auctions or bargaining, other strategies need to be applied to ensure scalability of information dissemination. Examples of such strategies are dividing auctions in subgroups [OV02] or approaches like Catallaxy-based bargaining [ERS$^+$05]. In such contexts, the overall knowledge gets lost and no accurate information is available (e.g., about the price of all offers in the market).

Sandholm and Lai [SL07] apply different mechanisms to predict the future demand of computational resources. They conclude to deduct the price of the resources from different real workload traces; however, markets with real market are not analyzed. The prediction of high peaks allows the consumers to avoid these to get lower prices or other benefits. This is especially important as sophisticated strategies change the market's behavior. Moreover, different market mechanisms lead to different peaks and distributions of the allocations. Cardosa and Chandra [CC10] analyze statistical aggregation for the resource allocation. The information retrieval aggregates historical data, which builds the basis for the prediction mechanisms. The authors provide a further breakup of commodity goods like analyzed in most market-allocation mechanisms into resource bundles, however, economic mechanisms are not considered.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1 Summary

Existing and future information services need to provide a timely and accurate information retrieval under a low network consumption to ensure the scalability to millions of users and data. A problem lies in the contradiction of these three factors because the reduction of one factor often means the increase of another factor. Furthermore, from the heterogeneity of the applications arises new challenges to satisfy the users' needs. For example, the range of the needs can vary among fast information retrieval, accurate results or even a balance among these factors can be necessary.

Different techniques intend to optimize one of the three factors. If we aim at offering a trade-off according to user's priorities, however, we have to consider all, the three factors, a dynamic network and heterogeneous applications. This dissertation investigates the trade-off among the timeliness, the accuracy and the number of messages in large-scale information systems.

Approximation techniques reduce the retrieval time and the network consumption. However, approximation techniques can also increase the inaccuracy in large-scale systems. Often a certain level of inaccuracy is preferable as inaccuracies in large-scale systems can also result from obsolete values and information loss due to failures. Therefore, our experiments analyze the effect of the approximation techniques in regard to the accuracy. The approximation technique applied in this thesis includes content summarization techniques for information discovery such as for information about resources in Grid Computing systems.

The proposed algorithms help the users in finding a trade-off among the timeliness, the accuracy, and the network consumption in a self-adaptive manner. The results of the summary-based algorithm show that the precision-recall metric can feed a user-driven decision for the network consumption, the quality of the results and the retrieval time. The second algorithm provides a self-adaptive decision-making in order to obtain the same trade-off among timeliness, messages and accuracy for approximate queries in large-scale information aggregation.

7.2 Conclusions

The main contribution of the thesis consists in a Cobweb-based algorithm and a AHP-based algorithm that reduce the retrieval time and the number of sent messages, while guaranteeing a reasonable level of accurate results. The algorithms use approximation techniques to reduce the message load and to guarantee a faster retrieval time. Different mechanisms allow self-regulating the accuracy by adjusting the quality of the approximations in terms of modifying the summarization depth and the size of the sample data. Furthermore, the presented evaluation framework is tested for an application in real world systems and it has shown a successful information exchange for existing applications.

The facts of a reducing the retrieval time and network consumption, while guaranteeing a high level of accuracy confirm the hypothesis of the thesis, which was presented in the introduction. The hypothesis assumes that the retrieval time of the query and the network consumption can be reduced by the usage of approximation. At the same time, a reasonable accuracy should be guaranteed when applying approximations. The following paragraphs explain how the hypothesis is confirmed in each chapter by applying algorithms that base on approximations.

In Chapter 4, the thesis presents a network-aware summarization technique for efficient information retrieval in terms of message size, number of messages, maximum retrieval time and network dependencies within large-scale P2P content networks. An environment was simulated with up to half a million randomly distributed resources as a completely decentralized Grid system. The Cobweb-based summary tree allows reducing significantly the number of the disseminated messages. Furthermore, the content summarization enables a discovery mechanism that has a number of hops per information retrieval process, which is close to the minimum number of peers and has a constant factor of needed hops for the information discovery in regard to an increasing number of resources.

The dissemination of information in systems with read-dominated queries helps to reduce the retrieval time and the network consumption of the information retrieval process. One would generally expect that the more information is spread in the system, the better is the accuracy of the retrieved results. Dissemination of information, however, is costly in terms of the size of the stored data and requires time to reach consistency if the attribute values change. Thus, a precision-recall metric helps offering a trade-off between the amounts of disseminated data and the overall accuracy of the discovered resources (and thereby increase the efficiency of the dissemination and discovery process).

The presented information provision technique is simulated for Grid systems and experimentally evaluated using real PlanetLab data, but it can also be applied to other large-scale information systems. Moreover, the presented architecture provides a separation between a logical P2P overlay and the associated information dissemination process, leading to flexibility with regard to underlying P2P protocols. As a result of the separation, the proposed architecture for the Cobweb-based summarization technique could be used alongside other P2P overlays.

In Chapter 5, the thesis proposed an algorithm to offer a trade-off among fast information retrieval, low message bandwidth, and highly accurate results. The algorithm applies the AHP-based decision-making to approximate queries for a large-scale information aggregation. In contrast to related baseline systems, the algorithm reduces the retrieval time from linearly increasing to a constant retrieval time (see Figure 5.4). We conclude from presented studies that the retrieval time is one of the most important factors in data management. Furthermore, the number of sent messages is reduced from an exponential increase to a constant factor (see

Figure 5.3). The reduction of the network bandwidth is important to increase the scalability. For example,the number of messages was reduced to $\frac{1}{20}$ and the retrieval time to $\frac{1}{5}$ for a system with 10 000 nodes in simulations of queries for an average value. These results show that the thesis' hypothesis of reducing the retrieval time and network consumption can be reached.

A consequence of reducing the retrieval time and reducing network consumption by approximate queries is a decreasing accuracy of the results. However, the AHP-based decision-making process guarantees a high accuracy because it adapts the approximation quality to the human priorities. The self-adaption stops the level of the approximation, when the accuracy is too low. The results show that the AHP-based algorithm guarantees the control of the accuracy on a reasonable level (Table 5.3 shows a accuracy higher than 99% for the most common cases). These supports the hypothesis that a reasonable accuracy can be guaranteeing while applying approximation techniques.

7.3 Future Work and Directions

An efficient approximation technique is important to optimize the information provision for large-scale systems. The thesis showed the improvements and the application of autonomous mechanism in order to obtain an efficient information provision with standard approximation techniques. Nevertheless, there are many approximation techniques in the area of statistics and mathematics such as Chebyshev's approximation. Therefore, other approximations could be more efficient than the provided one depending on the scenario. The evaluation of more specific or efficient approximation is a future direction. Our focus is on the provision of an autonomous algorithm to offer a trade-off between timeliness, network consumption and accuracy with a flexibility to adapt other approximation algorithms.

An interesting aspect is the combination of the approximation techniques with prediction mechanism. Prediction mechanisms are based on historic data and try to predict the future values. Examples for prediction mechanism are moving averages. Applying a forecast mechanism to an attribute (e.g. market price or the average CPU load) may increase the accuracy of the results. A higher accuracy would allow a further reduction of the retrieval time and network consumption.

The presented algorithm is compared to large-scale aggregation systems such as Astrolabe [VRBV03], SDIMS [YD04], Willow [VRB04] and DAT [CH06]. However, we are confident that our (AHP-based and cobweb-based) algorithms can also apply to large-scale databases such as Piazza [HIM$^+$04] and PeerDB [OTZ$^+$03] as they address similar problems arising from large-scale systems. Therefore, a future direction is towards the integration and application of the algorithms into P2P databases. However, the P2P databases need more investigation to reach the functionality of traditional databases that are applied on a central server. It is promising that the P2P databases will have an important role in the future to cope with the increasing amount of distributed applications, users and information. Future P2P database systems need possibly to apply approximation mechanisms like traditional databases already do.

BIBLIOGRAPHY

[AAGW04] Keno Albrecht, Ruedi Arnold, Michael Gähwiler, and Roger Wattenhofer. Aggregating information in peer-to-peer systems for improved join and leave. In *Peer-to-Peer Computing*, pages 227–234, 2004.

[ABC$^+$03] Serge Abiteboul, Angela Bonifati, Grégory Cobéna, Ioana Manolescu, and Tova Milo. Dynamic xml documents with distribution and replication. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, SIGMOD '03, pages 527–538, New York, NY, USA, 2003. ACM.

[ABE$^+$09] Sergio Andreozzi, Stephen Burke, Felix Ehm, Laurence Field, Gerson Galang, Balazs Konya, Paul Millar Maarten Litmaath, and JP Navarro. Glue specification v. 2.0. Technical report, http://forge.ogf.org/sf/projects/glue-wg, 2009.

[ADD09] Vasileios Anagnastopoulos, Nikolaos D. Doulamis, and Anastasios D. Doulamis. Edge-motion video summarization: Economical video summarization for low powered devices. In *WIAMIS'09*, pages 284–287, 2009.

[ADGK07] Benjamin Arai, Gautam Das, Dimitrios Gunopulos, and Vana Kalogeraki. Efficient approximate query processing in peer-to-peer networks. *IEEE Transactions on Knowledge and Data Engineering*, 19(7):919–933, 2007.

[AGP99] Swarup Acharya, Phillip B. Gibbons, and Viswanath Poosala. Aqua: A fast decision support systems using approximate query answers. In *VLDB*, pages 754–757, 1999.

[AWIK06] André Allavena, Qiang Wang, Ihab Ilyas, and Srinivasan Keshav. Lot: A robust overlay for distributed range query processing. Technical report, 2006.

[BA01] Werner Bier and Henning Ahnert. Trade-off between timeliness and accuracy. *Economisch Statistische Berichten (ESB)*, 2001.

[BAH$^+$06] Rolando Blanco, Nabeel Ahmed, David Hadaller, L. G. Alex Sung, Herman Li, and Mohamed Ali Soliman. A survey of data management in peer-to-peer systems, 2006.

[BBN$^+$09] Nikolay Borissov, René Brunner, Dirk Neumann, Felix Freitag, Leandro Navarro, and Christof Weinhardt. Fostering Efficiency of Computational Resource Allocation - Integrating Information Services into Markets. In *Proceedings of the 17th European Conference on Information Systems (ECIS' 09)*, pages 2048–2059, Verona, Italy, June 2009.

[BCRL09] N. Borissov, S. Caton, O. Rana, and A. Levine. Message Protocols for Provisioning and Usage of Computing Services. In *6th International Workshop on Grid Economics and Business Models*, pages 160–170, 2009.

[BFN08] René Brunner, Felix Freitag, and Leandro Navarro. Towards the development of a decentralized market information system: Requirements and architecture. In *Parallel and Distributed Computing in Finance (PDCoF'08).Proceedings of the 22nd IPDPS, Miami, FL, USA*, 2008.

[BGGMM04] Mayank Bawa, Aristides Gionis, Hector Garcia-Molina, and Rajeev Motwani. The price of validity in dynamic networks. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international* *conference on Management of data*, pages 515–526, New York, NY, USA, 2004. ACM.

[BGMGM03] Mayank Bawa, Hector Garcia-Molina, Aristides Gionis, and Rajeev Motwani. Estimating aggregates on a peer-to-peer network. Technical report, Stanford University, 2003.

[BMVV04] Ranjita Bhagwan, Priya Mahadevan, George Varghese, and Geoffrey M. Voelker. Cone: A distributed heap approach to resource selection. Technical report, 2004.

[Bru06] René Brunner. A performance evaluation of the kad-protocol. Master's thesis, Institut Eurécom and Universität Mannheim, 2006.

[BS99] Martin Bichler and Arie Segev. A brokerage framework for internet commerce. *Distrib. Parallel Databases*, 7(2):133–148, 1999.

[BV02] Dirk Bergemann and Juuso Valimaki. Information acquisition and efficient mechanism design. *Econometrica*, 70(3):1007–1033, May 2002. available at http://ideas.repec.org/a/ecm/emetrp/v70y2002i3p1007-1033.html.

[BV07] D. Bergemann and J. Valimaki. Information in mechanism design. In Whitney Newey Richard Blundell and Torsten Persson, editors, *Proceedings of the 9th World Congress of the Econometric Society*, volume Cambridge University Press 2007 of *Chapter 5*, pages 186–221, 2007.

[BW08] Nikolay Borissov and Niklas Wirström. Q-Strategy: A Bidding Strategy for Market-Based Allocation of Grid Services. In *On the Move to Meaningful Internet Systems: OTM 2008 (Grid computing, high-performAnce and Distributed Applications (GADA'08)), Monterrey, Mexico, Nov 13 - 14, 2008*, pages 744–761, October 2008.

[Car98] Antonio Carzaniga. *Architectures for an Event Notication Service Scalable to Wide-area Networks*. PhD thesis, POLITECNICO DI MILANO, 1998.

[CBCR11] K. Chard, K. Bubendorfer, S. Caton, and O. Rana. Social cloud computing: A vision for socially motivated resource sharing. *IEEE Transactions on Services Computing*, 2011.

[CC10] Michael Cardosa and Abhishek Chandra. Resource bundles: Using aggregation for statistical large-scale resource discovery and management. *IEEE Transactions on Parallel and Distributed Systems*, 21:1089–1102, 2010.

[CCR04] Miguel Castro, Manuel Costa, and Antony Rowstron. Performance and dependability of structured peer-to-peer overlays. In *Proceedings of the 2004 International Conference on Dependable Systems and Networks*, page 9, Washington, DC, USA, 2004.

[CCR05] Miguel Castro, Manuel Costa, and Antony Rowstron. Debunking some myths about structured and unstructured overlays. In *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, pages 85–98, Berkeley, CA, USA, 2005. USENIX Association.

[CDmK$^+$03] Miguel Castro, Peter Druschel, Anne marie Kermarrec, Animesh Nandi, Antony Rowstron, and Atul Singh. Splitstream: High-bandwidth multicast in a cooperative environment. In *In SOSP'03*, 2003.

[CDN07] Surajit Chaudhuri, Gautam Das, and Vivek R. Narasayya. Optimized stratified sampling for approximate query processing. *ACM Trans. Database Syst.*, 32(2):9, 2007.

[CDTW00] Jianjun Chen, David J. DeWitt, Feng Tian, and Yuan Wang. Niagaracq: a scalable continuous query system for internet databases. *SIGMOD Rec.*, 29:379–390, May 2000.

[CFF+02] Chee-Yong Chan, Wenfei Fan, Pascal A Felber, Minos Garofalakis, and Rajeev Rastogi. Tree pattern aggregation for scalable XML data dissemination. In *VLDB 2002 - 28th International Conference on Very Large Data Bases, August 20-23, 2002 , Hong Kong, China*, Aug 2002.

[CGMS07] Giovanni Conforti, Giorgio Ghelli, Paolo Manghi, and Carlo Sartiani. Scalable query dissemination in xpeer. In *IDEAS '07: Proceedings of the 11th International Database Engineering and Applications Symposium*, pages 199–207, Washington, DC, USA, 2007.

[CH06] Min Cai and Kai Hwang. Distributed aggregation schemes for scalable peer-to-peer and grid computing. Technical report, 2006.

[CH08] Justin Cappos and John H. Hartman. San fermin: aggregating large data sets using a binomial swap forest. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'08, pages 147–160, Berkeley, CA, USA, 2008. USENIX Association.

[CKFF01] Karl Czajkowski, Carl Kesselman, Steven Fitzgerald, and Ian T. Foster. Grid information services for distributed resource sharing. In *Proc. of Intl. Symposium on High Performance Distributed Computing (HPDC)*, San Francisco, USA, 2001.

[CLB+08] Pablo Chacín, Xavier León, René Brunner, Felix Freitag, and Leandro Navarro. Core services for grid markets. In *CGSYMP in proceedings of Europar*, 2008.

[CLKB04] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases. In *Data Engineering, 2004. Proceedings. 20th International Conference on*, pages 449–460, 2004.

[CR11] Simon Caton and Omer Rana. Towards Autonomic Management for Cloud Services based upon Volunteered Resources. *Concurrency and Computation: Practice and Experience*, 2011. Special Issue on Autonomic Cloud Computing: Technologies, Services, and Applications.

[CRCC10] Agustín C. Caminero, Omer F. Rana, Blanca Caminero, and Carmen Carrión. Network-aware heuristics for inter-domain metascheduling in grids. *Journal of Computer and System Sciences*, 2010.

[CRR$^+$05] Yatin Chawathe, Sriram Ramabhadran, Sylvia Ratnasamy, Anthony LaMarca, Scott Shenker, and Joseph Hellerstein. A case study in building layered dht applications. *SIGCOMM Comput. Commun. Rev.*, 35(4):97–108, 2005.

[CW08] John Colquhoun and Paul Watson. A peer-to-peer database server. In *BNCOD '08: Proceedings of the 25th British national conference on Databases*, pages 181–184, Berlin, Heidelberg, 2008. Springer-Verlag.

[CW09] John Colquhoun and Paul Watson. Evaluating a peer-to-peer database server based on bittorrent. In *BNCOD 26: Proceedings of the 26th British National Conference on Databases*, pages 171–179, Berlin, Heidelberg, 2009. Springer-Verlag.

[DD01] Nikolaos D. Doulamis and Anastasios D. Doulamis. Efficient video transmission over internet based on a hierarchical video summarization scheme. In *ICME'01*, pages –1–1, 2001.

[DDK00] Anastasios D. Doulamis, Nikolaos D. Doulamis, and Stefanos D. Kollias. A fuzzy video content representation for video summarization and content-based retrieval. *Signal Processing*, pages 1049–1067, 2000.

[DGR$^+$03] Alan Demers, Johannes Gehrke, Rajmohan Rajaraman, Niki Trigoni, and Yong Yao. The cougar project: A work-in-progress report. *ACM SIGMOD Record*, 32:53–59, 2003.

[DKDN09] N. D. Doulamis, P. N. Karamolegkos, A. D. Doulamis, and I. G. Nikolakopoulos. Exploiting semantic proximities for content search over P2P networks. *Computer Communications*, 32(5):814–827, 2009.

[DKR04] Antonios Deligiannakis, Yannis Kotidis, and Nick Roussopoulos. Hierarchical in-network data aggregation with quality guarantees. In *In EDBT*, 2004.

[DUA04] Zoran Despotovic, Jean-Claude Usunier, and Karl Aberer. Towards peer-to-peer double auctioning. In *HICSS '04: Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 9*, page 90289.1, Washington, DC, USA, 2004. IEEE Computer Society.

[DVS05] David Del Vecchio and Sang H. Son. Flexible update management in peer-to-peer database systems. In *IDEAS '05: Proceedings of the 9th International Database Engineering & Application Symposium*, pages 435–444, Washington, DC, USA, 2005. IEEE Computer Society.

[ERF$^+$05] Torsten Eymann, Michael Reinicke, Felix Freitag, Leandro Navarro, Oscar Ardaiz, and Pau Artigas. A hayekian self-organization approach to service allocation in computing systems. *Advanced Engineering Informatics*, 19(3):223–233, 2005.

[ERS$^+$05] Torsten Eymann, Michael Reinicke, Werner Streitberger, Omer Rana, Liviu Joita, Dirk Neumann, Björn Schnizler, Daniel Veit, Oscar Ardaiz, Pablo Chacin, Isaac Chao, Felix Freitag, Leandro Navarro, Michele Catalano, Mauro Gallegati, Gianfranco Giulioni, Ruben Carvajal Schiaffino, and Floriano Zini. Catallaxy-based grid markets. *Multiagent Grid Syst.*, 1(4):297–307, 2005.

[EZS$^+$09] G. Evangelopoulos, A. Zlatintsi, G. Skoumas, K. Rapantzikos, A. Potamianos, P. Maragos, and Y. Avrithis. Video event detection and summarization using audio, visual and text saliency. In *Proc. of the Intl. Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Washington, DC, USA, 2009.

[Fis87] Douglas H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139, 1987.

[For10] Distributed Management Task Force. Common information model infrastructure. Technical report, http://www.dmtf.org/standards/cim/, 2010.

[GC85] Mark A. Gluck and James E. Corter. Information, uncertainty, and the utility of categories. In *Proc. of the Seventh Annual Conference of the Cognitive Science Society*, Hillsdale, NJ, 1985.

[GHI$^+$01] Steven Gribble, Alon Halevy, Zachary Ives, Maya Rodrig, and Dan Suciu. What can peer-to-peer do for databases, and vice versa? In *In Proceedings of the WebDB*, 2001.

[GKK$^+$03] Phillip B. Gibbons, Brad Karp, Yan Ke, Suman Nath, and Srinivasan Seshan. Irisnet: An architecture for a worldwide sensor web. *IEEE Pervasive Computing*, 2:22–33, October 2003.

[GS03] Jens Grossklags and Carsten Schmidt. Interaction of human and artificial agents on double auction markets - simulations and laboratory experiments. In *IAT '03: Proceedings of the IEEE/WIC International Conference on Intelligent Agent Technology*, page 400, Washington, DC, USA, 2003. IEEE Computer Society.

[GT06] G. Ghinita and Yong Meng Teo. An adaptive stabilization framework for distributed hash tables. *Parallel and Distributed Processing Symposium, International*, 0:12, 2006.

[HC09] Cyrus Hall and Antonio Carzaniga. Uniform sampling for directed P2P networks. In *Euro-Par 2009*, number 5704 in LNCS, pages 511–522, Delft, The Netherlands, August 2009. Springer-Verlag.

[HCH$^+$05] Ryan Huebsch, Brent N. Chun, Joseph M. Hellerstein, Boon Thau Loo, Petros Maniatis, Timothy Roscoe, Scott Shenker, Ion Stoica, and Aydan R. Yumerefendi. The architecture of pier: an internet-scale query processor. In *CIDR*, pages 28–43, 2005.

[HFH$^+$09] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11:10–18, November 2009.

[HIM$^+$04] Alon Y. Halevy, Zachary G. Ives, Jayant Madhavan, Peter Mork, Dan Suciu, and Igor Tatarinov. The piazza peer data management system. *IEEE Transactions on Knowledge and Data Engineering*, 16(7):787–798, 2004.

[HPR90] J. Helary, N. Plouzeau, and M. Raynal. Computing particular snapshots in distributed systems. In *Computers and Communications, 1990. Conference Proceedings., Ninth Annual International Phoenix Conference on*, pages 116–123, Mar 1990.

[HRVM08] Rabab Hayek, Guillaume Raschia, Patrick Valduriez, and Noureddine Mouaddib. Summary management in unstructured p2p systems. *Ingénierie des Systèmes d'Information*, 13(5):83–106, 2008.

[HV03] Qi Han and Nalini Venkatasubramanian. Addressing timeliness/accuracy/cost tradeoffs in information collection for dynamic environments. In *RTSS '03: Proceedings of the 24th IEEE International Real-Time Systems Symposium*, page 108, Washington, DC, USA, 2003.

[ISW92] Yannis E. Ionnidis, Tomas Saulys, and Andrew J. Whitsitt. Conceptual learning in database design. *ACM Transactions on Information Systems*, 10:265–293, 1992.

[JKM$^+$07a] N. Jain, D. Kit, D. Mahajan, P. Yalagandula, M. Dahlin, and Y. Zhang. Known unknowns in large-scale system monitoring. In *In review.*, October 2007.

[JKM$^+$07b] N. Jain, D. Kit, P. Mahajan, P. Yalagandula, M. Dahlin, and Y. Zhang. STAR: Self-tuning aggregation for scalable monitoring. In *33rd International Conference on Very Large Data Bases (VLDB 2007)*, September 2007.

[JKM$^+$08] N. Jain, D. Kit, D. Mahajan, P. Yalagandula, M. Dahlin, and Y. Zhang. Network imprecision: A new consistency metric for scalable monitoring. In *OSDI 2008*, December 2008.

[JKvS04] Márk Jelasity, Wojtek Kowalczyk, and Maarten van Steen. An approach to massively distributed aggregate computing on peer-to-peer networks. In *Proceedings of the 12th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP'04)*, pages 200–207, A Coruna, Spain, 2004. IEEE Computer Society.

[JMB05] Márk Jelasity, Alberto Montresor, and Ozalp Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst.*, 23(3):219–252, August 2005.

[JOT$^+$06] H. V. Jagadish, Beng Chin Ooi, Kian-Lee Tan, Quang Hieu Vu, and Rong Zhang. Speeding up search in peer-to-peer networks with a multi-way tree structure. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, SIGMOD '06, pages 1–12, New York, NY, USA, 2006. ACM.

[JPR09] Shantenu Jha, Manish Parashar, and Omer Rana. Self-adaptive architectures for autonomic computational science. In *SOAR*, pages 177–197, 2009.

[JSG$^+$04] Jinyang Li Jeremy, Jeremy Stribling, Thomer M. Gil, Robert Morris, and M. Frans Kaashoek. Comparing the performance of distributed hash tables under churn. In *In Proc. IPTPS*, 2004.

[KDG03] David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. In *FOCS '03: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, page 482, Washington, DC, USA, 2003. IEEE Computer Society.

[KDN$^+$06] Srinivas Kashyap, Supratim Deb, K. V. M. Naidu, Rajeev Rastogi, and Anand Srinivasan. Efficient gossip-based aggregate computation. In *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '06, pages 308–317, New York, NY, USA, 2006. ACM.

[KNB$^+$08] Ruby Krishnaswamy, Leandro Navarro, René Brunner, Xavier León, and Xavier Vilajosana. Grid4all: Open market places for democratic grids. In *Grid Economics and Business Models, 5th International Workshop, GECON 2008, Las Palmas de Gran Canaria, Spain, August 26, 2008. Proceedings*, volume 5206 of *Lecture Notes in Computer Science*, pages 197–207. Springer, 2008.

[KNO$^+$02] Panos Kalnis, Wee Siong Ng, Beng Chin Ooi, Dimitris Papadias, and Kian-Lee Tan. An adaptive peer-to-peer network for distributed caching of OLAP results. In *SIGMOD Conference*, 2002.

[KYG$^+$08] Steven Y. Ko, Praveen Yalagandula, Indranil Gupta, Vanish Talwar, Dejan Milojicic, and Subu Iyer. Moara: flexible and scalable group-based querying system. In *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, Middleware '08, pages 408–428, New York, NY, USA, 2008. Springer-Verlag New York, Inc.

[LCP$^+$05] Keong Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *Communications Surveys & Tutorials, IEEE*, pages 72–93, 2005.

[LL04] Simon S. Lam and Huaiyu Liu. Failure recovery for structured p2p networks: protocol design and performance evaluation. *SIGMETRICS Perform. Eval. Rev.*, 32(1):199–210, 2004.

[LP03] Ying Liu and Beth Plale. Survey of publish subscribe event systems. Technical report, Indiana University, 2003.

[LSL05] Ji Li, Karen Sollins, and Dah-Yoh Lim. Implementing aggregation and broadcast over distributed hash tables. *SIGCOMM Comput. Commun. Rev.*, 35(1):81–92, 2005.

[LVB$^+$08] Xavier León, Xavier Vilajosana, René Brunner, Ruby Krishnaswamy, Leandro Navarro, Felix Freitag, and Joan Manuel Marqués. Information and regulation in decentralized marketplaces for p2p-grids. In *COPS in proccedings of WETICE*, 2008.

[Mat93] Friedemann Mattern. Efficient algorithms for distributed snapshots and global virtual time approximation. *J. Parallel Distrib. Comput.*, 18(4):423–434, 1993.

[MFHH02] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. In *IN OSDI*, 2002.

[MGL$^+$10] Sergey Melnik, Andrey Gubarev, Jing Jing Long, Geoffrey Romer, Shiva Shivakumar, Matt Tolton, and Theo Vassilakis. Dremel: interactive analysis of web-scale datasets. *Proc. VLDB Endow.*, 3:330–339, September 2010.

[MMKG06] Laurent Massoulié, Erwan Le Merrer, Anne-Marie Kermarrec, and Ayalvadi Ganesh. Peer counting and sampling in overlay networks: random walk methods. In *Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, PODC '06, pages 123–132, New York, NY, USA, 2006. ACM.

[MR81] Carolyn B. Mervis and Eleanor Rosch. Categorization of natural objects. *Annual Review of Psychology*, 32(1):89–115, 1981.

[MT10] Rana O. Smith G. Guitart J. Macías, M. and J. Torres. Maximizing revenue in grid markets using an economically enhanced resource manager. *Concurrency and Computation: Practice and Experience*, 22:1990–2011, 2010.

[NAB$^+$08] J. Nimis, A. Anandasivam, N. Borissov, G. Smith, D. Neumann, N. Wirstroem, E. Rosenberg, and M. Villa. Sorma - business cases for an open grid. In *The 5th International Workshop on Grid Economics and Business Models (Gecon 2008), Las Palmas de Gran Canaria, Spain*, pages 173–184, 2008.

[NDMR08] Dushyanth Narayanan, Austin Donnelly, Richard Mortier, and Antony Rowstron. Delay aware querying with seaweed. *The VLDB Journal*, 17:315–331, March 2008.

[NH07] Lei Ni and Aaron Harwood. A comparative study on peer-to-peer failure rate estimation. In *ICPADS '07: Proceedings of the 13th International Conference on Parallel and Distributed Systems*, pages 1–7, Washington, DC, USA, 2007. IEEE Computer Society.

[NWST02] Wolfgang Nejdl, Boris Wolf, Steffen Staab, and Julien Tane. Edutella: Searching and annotating resources within an rdf-based p2p network. In *Semantic Web Workshop, WWW02 Conference, Hawai, USA*, 2002.

[OTZ$^+$03] Beng Chin Ooi, Kian-Lee Tan, Aoying Zhou, Chin Hong Goh, Yingguang Li, Chu Yee Liau, Bo Ling, Wee Siong Ng, Yanfeng Shu, Xiaoyu Wang, and Ming Zhang. Peerdb: Peering into personal databases. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, San Diego, California, USA,*, page 659. ACM, June 9-12 2003.

[OV02] Elth Ogston and Stamatis Vassiliadis. A peer-to-peer agent auction. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 151–159, New York, NY, USA, 2002. ACM Press.

[OW00] Chris Olston and Jennifer Widom. Offering a precision-performance tradeoff for aggregation queries over replicated data. In *in VLDB*, pages 144–155, 2000.

[PAP$^+$03] Evaggelia Pitoura, Serge Abiteboul, Dieter Pfoser, George Samaras, and Michalis Vazirgiannis. Dbglobe: A service-oriented p2p system for global computing. *SIGMOD Record*, 32:77–82, 2003.

[PMRK06] Larry Peterson, Steve Muir, Timothy Roscoe, and Aaron Klingaman. PlanetLab Architecture: An Overview. Technical Report PDN–06–031, PlanetLab Consortium, May 2006.

[PvT98] Chris Preist and Maarten van Tol. Adaptive agents in a persistent shout double auction. In *ICE '98: Proceedings of the first international conference on Information and computation economies*, pages 11–18, New York, NY, USA, 1998. ACM Press.

[RD01] Antony Rowstron and Peter Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, November 2001.

[RF02] Matei Ripeanu and Ian T. Foster. Mapping the gnutella network: Macroscopic properties of large-scale peer-to-peer systems. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 85–93, London, UK, 2002. Springer-Verlag.

[RKCD01] Antony I. T. Rowstron, Anne-Marie Kermarrec, Miguel Castro, and Peter Druschel. Scribe: The design of a large-scale event notification infrastructure. In *NGC '01: Proceedings of the Third International COST264 Workshop on Networked Group Communication*, pages 30–43, London, UK, 2001. Springer-Verlag.

[RM06] John Risson and Tim Moors. Survey of research towards robust peer-to-peer networks: search methods. volume 50, pages 3485–3521, New York, NY, USA, 2006. Elsevier North-Holland, Inc.

[RN08] Norvald H. Ryeng and Kjetil Norvag. Robust aggregation in peer-to-peer database systems. In *IDEAS '08: Proceedings of the 2008 international symposium on Database engineering and applications*, pages 29–37, New York, NY, USA, 2008. ACM.

[Saa90] T.L. Saaty. How to make a decision: The Analytic Hierarchy Process. *European Journal of Operational Research*, 48:9–26, 1990.

[SAP$^+$96] Michael Stonebraker, Paul M. Aoki, Avi Pfeffer, Adam Sah, Jeff Sidell, Carl Staelin, and Andrew Yu. Mariposa: A wide-area distributed database system. *VLDB Journal*, 5:48–63, 1996.

[SCSI08] Denis Simakov, Yaron Caspi, Eli Shechtman, and Michal Irani. Summarizing visual data using bidirectional similarity. In *Proc. of the Conference on Computer Vision and Pattern Recognition (CVPR)*, Anchorage, USA, 2008.

[SL07] Thomas Sandholm and Kevin Lai. A statistical approach to risk mitigation in computational markets. In *HPDC '07: Proceedings of the 16th international symposium on High performance distributed computing*, pages 85–96, New York, NY, USA, 2007. ACM.

[SN95] Ambuj Shatdal and Jeffrey F. Naughton. Adaptive parallel aggregation algorithms. *SIGMOD Rec.*, 24:104–114, May 1995.

[SNSP10] Jan Sacha, Jeff Napper, Corina Stratan, and Guillaume Pierre. Adam2: Reliable distribution estimation in decentralised environments. In *Proceedings of the 30th IEEE International Conference on Distributed Computing Systems (ICDCS)*, June 2010.

[SOR08] SORMA. Sorma project. http://www.iw.uni-karlsruhe.de/sormang/, 2008.

[TDVK99] Renu Tewari, Michael Dahlin, Harrick M. Vin, and Jonathan S. Kay. Design considerations for distributed caching on the internet. In *International Conference on Distributed Computing Systems*, pages 273–284, 1999.

[TTF$^+$06] Paolo Trunfio, Domenico Talia, Paraskevi Fragopoulou, Charis Papadakis, Matteo Mordacchini, Mika Pennanen, Konstantin Popov, Vladimir Vlassov, and Seif Haridi. Peer-to-peer models for resource discovery on grids. Technical Report TR-0028, CoreGRID, 2006.

[VP04] Patrick Valduriez and Esther Pacitti. Data management in large-scale p2p systems. In *VECPAR*, pages 104–118, 2004.

[VRB04] Robbert Van Renesse and Adrian Bozdog. Willow: Dht, aggregation, and publish/subscribe in one protocol. In Geoffrey M. Voelker and Scott Shenker, editors, *IPTPS*, volume 3279 of *Lecture Notes in Computer Science*, pages 173–183. Springer, 2004.

[VRBV03] Robbert Van Renesse, Kenneth P. Birman, and Werner Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM Trans. Comput. Syst.*, 21(2):164–206, May 2003.

[VTL$^+$04] B. Volckaert, P. Thysebaert, M. De Leenheer, F. De Turck, B. Dhoedt, and P. Demeester. Network aware scheduling in grids. In *Ninth European Conference on Networks and Optical Communications, NOC 2004*, Eindhoven, The Netherlands, 2004.

[Yal05] Praveen Yalagandula. A scalable information management middleware for large distributed systems. Technical report, Laboratory for Advanced Systems. Research Department of Computer Sciences. The University of Texas at Austin, 2005.

[YD04] Praveen Yalagandula and Mike Dahlin. A scalable distributed information management system. *SIGCOMM Comput. Commun. Rev.*, 34(4):379–390, 2004.

[YD07] Praveen Yalagandula and Michael Dahlin. Shruti: A self-tuning hierarchical aggregation system. In *SASO*, pages 141–150, 2007.

[YGM03] Beverly Yang and Hector Garcia-Molina. Designing a super-peer network. *Data Engineering, International Conference on*, 0:49, 2003.

[ZHC08] Runfang Zhou, Kai Hwang, and Min Cai. Gossiptrust for fast reputation aggregation in peer-to-peer networks. *IEEE Trans. on Knowl. and Data Eng.*, 20(9):1282–1295, 2008.

[ZS05] Serafeim Zanikolas and Rizos Sakellariou. A taxonomy of grid monitoring systems. *Future Generation Computer Systems*, 21(1):pp. 163–188, 2005.

[ZSZ03] Zheng Zhang, Shu M. Shi, and Jinh Zhu. Somo: Self-organized metadata overlay for resource management in p2p dht. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, Berkeley, CA, USA, 2003.

[ZZJ$^+$01] Shelley Q. Zhuang, Ben Y. Zhao, Anthony D. Joseph, Randy H. Katz, and John D. Kubiatowicz. Bayeux: an architecture for scalable and fault-tolerant wide-area data dissemination. In *NOSSDAV '01: Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video*, pages 11–20, New York, NY, USA, 2001. ACM Press.