Capítulo 5

Algoritmo Paralelo

En este Capítulo se presenta el algoritmo paralelo, *Parallel Skew Conscious Radix sort* (PSKC-Radix sort), el cual es *equivalente* al algoritmo secuencial Skew Conscious Radix sort, presentado en el Capítulo anterior.

En la Figura 5.1 se muestran las propuestas realizadas en técnicas de particionado y algoritmos de ordenación paralela, junto con la relación de inclusión existente entre ellas. Cada relación de inclusión se muestra con una flecha cuyo significado es que el destino incluye al origen.

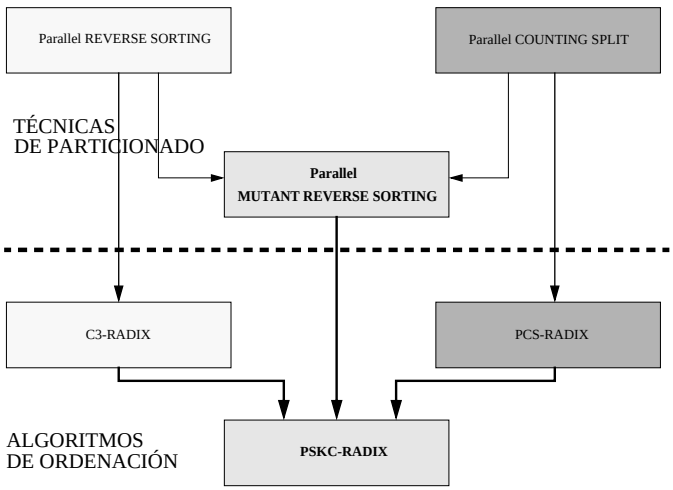

Figura 5.1: Relación de inclusión entre las técnicas de particionado y los algoritmos paralelos de ordenación propuestos en esta tesis.

En la parte superior de la Figura se muestra la relación entre las técnicas de particionado, Parallel Mutant Reverse Sorting, Parallel Counting Split y Parallel Reverse

Sorting, que se explican en el Capítulo 3.

PSKC-Radix sort combina técnicas de particionado y ordenación tal y como se puede observar en la Figura. PSKC-Radix sort combina la técnica de particionado paralela Mutant Reverse Sorting con dos algoritmos paralelos que se han propuesto durante el desarrollo de la tesis. Estos algoritmos son *Communication and Cache Conscious Radix sort* (C3-Radix sort) [23] y *Parallel Counting Split Radix sort* (PCS-Radix sort) [27] que ordenan datos de 32 y 64 bits respectivamente. Los resultados publicados en estos trabajos demuestran que son los algoritmos paralelos de ordenación en memoria más rápidos para estos dos tipos de datos. PSKC-Radix sort, tal y como se verá en los resultados, es el que mejor se adapta a las características de los datos, independientemente del número de bits de los datos.

En la primera sección se explica el algoritmo de PSKC-Radix sort. Después, se analizan el *speed-up* y la escalabilidad de este algoritmo. Finalmente, se comparan PSKC-Radix sort y Load Balanced Radix sort. Load Balanced Radix sort era el algoritmo paralelo de ordenación en memoria más rápido en el momento del desarrollo de esta tesis [40]. El análisis comparativo de los algoritmos PSKC-Radix sort y Load Balanced Radix sort se realiza en un computador basado en p630, con 16 procesadores Power4, y en un SGI O2000 usando hasta 32 procesadores R10K.

5.1. Parallel Skew Conscious Radix Sort

PSKC-Radix sort se muestra en el Algoritmo 16. Éste es, en esencia, el mismo algoritmo que el algoritmo de particionado paralelo Mutant Reverse Sorting (Algoritmo 12 en el Capítulo 3), añadiendo la ordenación local de las particiones globales, una vez que se han comunicado los datos. Los tres primeros pasos mostrados en el Algoritmo 16 son los mismos que los del Algoritmo 12 en el Capítulo 3. Así, PSKC-Radix sort consiste en los siguientes pasos:

1. **Paso (1) - Obtención y comunicación del vector de muestreo a P_0:** Línea 2 del Algoritmo 16.

2. **Paso (2) - Aplicación del Reverse Sorting en P_0 sobre el vector de muestreo recibido:** Línea 4 del del Algoritmo.

3. **Paso (3) - Comprobación del equilibrio de carga y elección:** Se averigua si se podría conseguir un buen equilibrio de carga entre los procesadores tras dos iteraciones de Reverse Sorting. Esto se realiza comprobando el equilibrio de carga que se puede conseguir en el particionado del vector muestreado. Las razones se explicaron en detalle cuando se explicó la técnica de particionado paralelo Mutant Reverse Sorting, Capítulo 3. Así, si se consigue un buen equilibrio de carga, se estará en el Caso (3.1) del Algoritmo 16 (líneas 8 a 11). Por el contrario,

si aplicando Reverse Sorting no se consiguiese un buen equilibrio de carga entre los datos muestreados, se estará en el Caso (3.2) del Algoritmo 16 (Líneas 14 a 18).

- **Caso 3.1: Aplicar Reverse Sorting:** Todos los procesadores particionan el conjunto de datos a ordenar con la técnica de Reverse Sorting paralelo. Una vez hecho el particionado y la comunicación de los datos, cada procesador P_i ordena las particiones globales que le corresponden. La ordenación de las particiones globales se realiza con el algoritmo secuencial que se propuso en la sección anterior, SKC-Radix sort (Línea 10 del Algoritmo 16).

  En la ordenación se debe tener en cuenta que el número de bits de cada clave que resta por ordenar es $b - kb_r$. Donde k es el número de iteraciones de Reverse Sorting efectuadas por parte de PSKC-Radix sort.

  Todo este proceso (líneas 8 a 11 del Algoritmo) corresponde al algoritmo paralelo de ordenación de datos de 32 bits, C3-Radix sort, publicado en [23].

- **Caso 3.2: Aplicar Counting Split:** Los procesadores particionan con Counting Split en paralelo. Para ello se utilizan las claves del muestreo obtenidas en el Paso (1). Una vez realizados el particionado y la comunicación de los datos, cada procesador P_i ordena las particiones globales que le tocan con SKC-Radix sort. Los datos de cada partición tendrán un conjunto de bits iguales. El número de bits iguales puede ser diferente entre distintas particiones.

  Todo este proceso se realiza en las líneas 14 a 18 del Algoritmo 16, que se corresponde con el algoritmo de ordenación de datos de 64 bits, PCS-Radix sort [26, 27].

5.2. Evaluación del Algoritmo

En esta sección se analiza el comportamiento del algoritmo PSKC-Radix sort en el computador basado en p630 y el SGI O2000. Los resultados se dan en CSE medios, es decir, la suma de los ciclos que tardan todos los procesadores que participan en la ejecución, dividido entre el número de claves totales a ordenar, n, y el número de procesadores P. También se muestra el número de CSE del proceso que más ciclos tarda con tal de dar una idea del desequilibrio de carga. Primero se analiza cómo se distribuyen los CSE entre los diferentes pasos del algoritmo: Reverse Sorting o Counting Split, ordenación y comunicación de los datos. Después se verá cómo afectan las diferentes distribuciones de datos al número de CSE de cada paso de PSKC-Radix sort.

Algoritmo 16: Parallel Skew Conscious Radix sort (S, n, b_r)

1: – **Paso (1):** Obtener Muestreo
2: $vector\_muestreo_{P_0} \leftarrow envia\_muestreo(\forall iP_i, q/P)$

3: – **Paso (2):** Reverse Sorting local del vector de muestreo
4: $< D, C > \leftarrow particiona\_conteo(vector\_muestreo_{P_0}, q, b, 2b_r)$

5: – **Paso (3):** Comprobación del equilibrio de carga y elección.
6: **si** $equilibrio\_carga(C)$ **entonces**
7: – **Caso (3.1):** Aplicar Reverse Sorting paralelo y ordenar (C3-Radix sort).
8: $particiones\_globales_{P_i} \leftarrow Reverse\_Sorting\_Paralelo()$
9: **para todo** $particion\_global$ en $particiones\_globales_{P_i}$ **hacer**
10: $SKC\_Radix(sub\_bucket, |sub\_bucket|, b - b_r)$
11: **fin para**
12: **sino**
13: – **Caso (3.2)** Aplicar Counting Split paralelo y ordenar (PCS-Radix sort).
14: $particiones\_globales_{P_i} \leftarrow Counting\_Split\_Paralelo()$
15: **para todo** $particion\_global$ en $particiones\_globales_{P_i}$ **hacer**
16: $sub\_bits \leftarrow b - bits\_comunes(vector\_separadores, sub\_bucket)$
17: $SKC\_Radix(sub\_bucket, |sub\_bucket|, sub\_bits)$
18: **fin para**
19: **fin si**

5.2.1. Particionado y Comunicación

La Figura 5.2 muestra la distribución de los CSE invertidos en las diferentes partes del algoritmo PSKC-Radix sort, al ordenar una distribución Random de datos. En las gráficas de arriba de la Figura 5.2 se muestran los CSE para un computador basado en p630 con 16 procesadores Power4. Los conjuntos que se ordenan van de 1M a 64M (de 64k a 4M datos por procesador) datos de clave y puntero de 32 bits (gráfica izquierda) y 64 bits (gráfica derecha). En las gráficas de abajo se muestran los CSE para el SGI O2000 y 32 procesadores R10K para ordenar de 2M a 128M (de 64k a 4M datos por procesador) datos de 32 bits (gráfica de la izquierda), y de hasta 64M (2M por procesador) datos de 64 bits (gráfica de la derecha). Cada columna de la Figura 5.2 muestra, de arriba a abajo, el número medio de CSE invertidos en comunicación de datos (Com. datos), ordenación local (Ordenación local), comunicación para la realización del particionado (Com. Particionado) y el cálculo para el particionado local que se realiza en cada procesador (Cal. Particionado). Detrás de cada columna se muestran los CSE del procesador más lento.

Para ambos computadores se puede observar que el número de CSE invertidos en el particionado (comunicación y cálculo) es pequeño en comparación a los CSE totales

invertidos. Esto se debe a que el particionado que se realiza para una distribución Random consiste en una única iteración de Reverse Sorting.

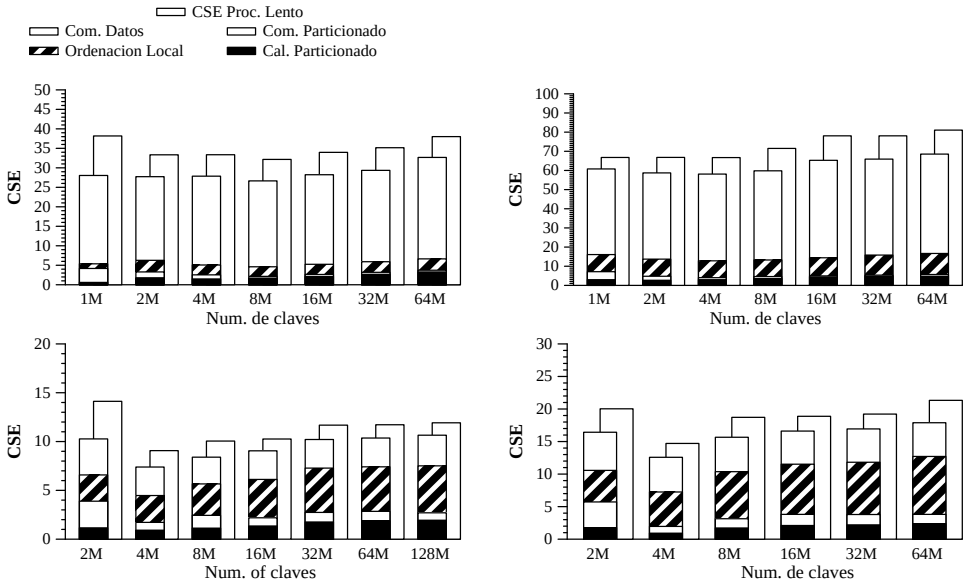

Figura 5.2: Arriba, CSE para PSKC-Radix sort ordenando 1M a 64M datos de clave y puntero de 32 bits (izquierda) y 64 bits (derecha), con 16 procesadores Power4 en un computador basado en p630. Abajo, CSE para 32 procesadores R10K en un SGI O2000 para 2M a 128M datos de 32 bits en la gráfica de la izquierda, y para 2M a 64M datos de 64 bits en la de la derecha. La distribución de datos es Random. La leyenda se debe leer de arriba a abajo y de izquierda a derecha para entender las barras.

Por otra parte, la Figura 5.3 muestra los resultados para una distribución de datos S40 (el 40 % de los bits de la clave es igual a todas las claves) para el mismo rango de elementos a ordenar, tipo de computador y número de procesadores que se mostraron en la Figura 5.2. En el caso del computador con p630, el particionado para la distribución S40 es de un 10 % a un 20 % más costoso que el particionado para la distribución Random, *para el que se particiona con una única iteración de Reverse Sorting*. En el caso del SGI O2000, es de un 20 % a un 30 % más costoso. Esto se debe a:

1. Cálculo: el particionado de los datos locales con Counting Split realiza una búsqueda dicotómica para cada clave. Esto es más costoso que un particionado con una iteración de Reverse Sorting (ver Capítulo 3 para más detalles).

2. Comunicación: en el particionado con Counting Split se tienen que comunicar claves de muestreo y separadores. En el particionado con Reverse Sorting esto no es necesario. Además, en caso de no conseguirse un buen equilibrio de carga

entre los procesadores con el particionado Counting Split, las particiones frontera se vuelven a particionar. En este caso, los CSE de particionado se incrementan (más detalle en el Capítulo 3).

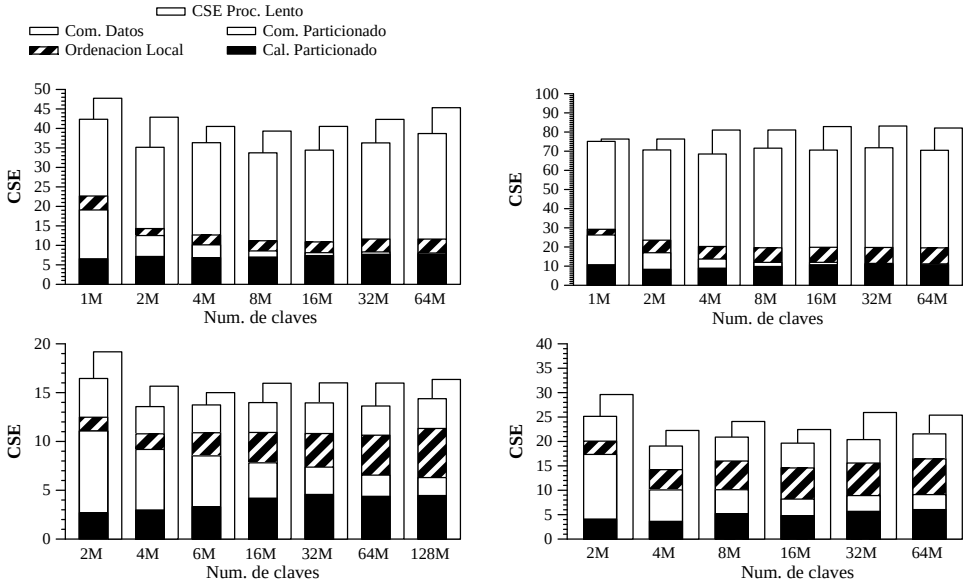

Figura 5.3: CSE para la distribución S40. Arriba para el computador con p630 y abajo para el SGI O2000. La descripción de la gráfica es la misma que para la de una distribución Random (Figura 5.2).

Sin embargo, el rendimiento del particionado con Counting Split para la distribución S40 es mejor que si se hubiera elegido Reverse Sorting paralelo para particionarlos, tal y como se mostró en el Capítulo 3. En este caso, Mutant Reverse Sorting elige Counting Split porque el número de iteraciones de Reverse Sorting necesarios para obtener equilibrio de carga para esta distribución de datos sería demasiado grande.

Desde un punto de vista de equilibrio de carga, se puede observar que la diferencia entre valores medios de CSE y el procesador más lento (columna en blanco, detrás de cada columna con las diferentes partes) va entre un 5 % y un 15 % más lento en media. El número de CSE para el procesador más lento varía según se ordenen claves de 32 ó 64 bits o si el número de datos a ordenar por procesador es pequeño. Si el tamaño del conjunto de datos a ordenar por procesador es pequeño, el desequilibrio de carga puede ser más grande que un 15 % debido al desequilibrio de carga permitido, U.

Por otra parte, se puede observar que hay una diferencia notable entre los resultados de el computador basado en p630 y el SGI O2000. El número de CSE de comunicación de datos es, en relación al total de CSE, mucho mayor que en el caso del p630. Esto se debe a que el ancho de banda entre procesadores de diferentes nodos (módulos SCM

p630) es pequeño. Aquí se trabaja con 4 módulos SCM p630. Cada módulo SCM p630 tiene 4 procesadores. Así, mientras que para la distribución Random, la comunicación ocupa entre un 75 % y un 80 % para ordenar datos de 32 y 64 bits en el computador basado en p630, en el SGI O2000 es de aproximadamente un 30 %. Para la distribución S40, el tanto por ciento de comunicación de datos con respecto al total disminuye en ambos computadores. La comunicación de los datos ocupa un 60 % del tiempo para el computador basado en p630 y de un 20 % en el SGI O2000, tal y como se observa en la Figura 5.3. Sin embargo, el número de CSE de la comunicación de datos es aproximadamente el mismo. La razón está en que el número de CSE invertido en el particionado (comunicación y cálculo) se hace relativamente mayor en este caso. Es necesario recordar que para esta distribución de datos se aplica el particionado local de Counting Split.

5.2.2. Distribuciones de Datos

En esta sección se analiza cuál es el comportamiento de PSKC-Radix sort para diferentes distribuciones de datos. PSKC-Radix sort varía su comportamiento dependiendo de las características de los datos a tratar. La Figura 5.4 muestra los resultados para 16 procesadores variando el conjunto de datos a ordenar de 1M a 64M datos de clave y puntero para el computador basado en p630 (arriba), y para la ordenación de datos de 2M a 64M datos de clave y puntero para el SGI O2000 (abajo). En ambos casos, se muestran resultados de ordenación de claves y punteros de 64 bits. En esta figura se muestran los resultados para las distribuciones Gaussian, S40, D50, D100 y Stagger. Al igual que se comentó en la sección anterior, hay diferencias en el número de CSE según el tipo de particionado elegido por PSKC-Radix sort.

Desde el punto de vista del particionado local se puede observar que, en la ordenación de las distribuciones de datos Gaussian, S40 y D50 se necesita un mayor número de CSE que para la distribución Random que se estudió en las secciones anteriores. Esto es debido a que en el caso de la distribución Gaussian es preciso realizar dos iteraciones de Reverse Sorting para conseguir un buen equilibrio de carga, mientras que en el caso de las distribuciones S40 y D50 se particiona con Counting Split.

Para la distribución D100 también se aplica Counting Split, pero tal y como se mostró en el Capítulo 3, esta distribución de datos hace que el número de CSE, debido a la búsqueda dicotómica del particionado local con Counting Split, sea menor. De ahí que el número de CSE del particionado sea menor también en la gráfica. En el caso de la distribución Stagger, al igual que la distribución Random, con una iteración de Reverse Sorting se consigue particionar los datos; con lo que el número de CSE es también reducido. Todo lo que sea más de una iteración de Reverse Sorting significa mayor cantidad de contadores a comunicar. En el caso de particionar con Counting Split, se deberán comunicar, además, separadores. En cualquiera de los dos casos, que se necesite más de una iteración de Reverse Sorting o se necesite aplicar Counting Split,

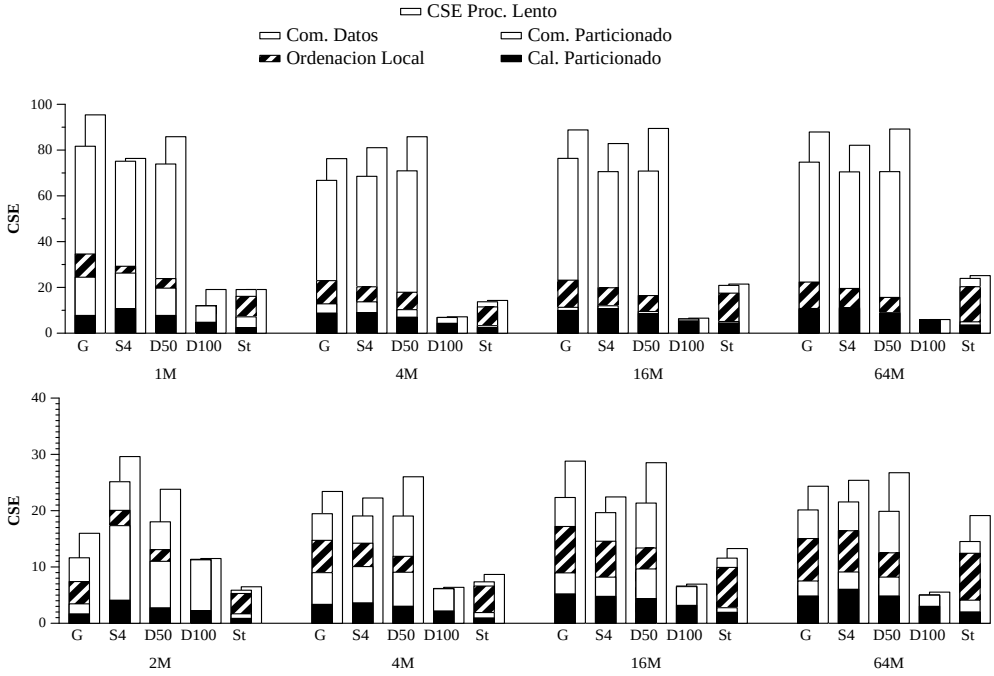

Figura 5.4: Comparativa de PSKC-Radix sort, variando el número de datos (64 bits) para las distribuciones Gaussian (G), S40 (S4), D50, D100 y Stagger (St). Los CSE son para un computador basado en p630 con 16 procesadores (arriba) y un SGI O2000 con 32 procesadores (abajo).

a este *overhead* de comunicación y cálculo hay que restarle el ahorro de ordenación que se hace posteriormente en la ordenación local. Esto se puede observar para las distribuciones S40, D50 y D100 en las figuras. La ordenación local en el caso de las distribuciones S40 y D50 es menor que en la Gaussian. En el caso de D100, el algoritmo no tiene que ordenar nada después del particionado.

Desde un punto de vista de comunicación de datos, en el computador basado en p630, los CSE dedicados son un 75 % a 80 % del número total de CSE; siempre y cuando no se pueda reducir la cantidad de datos comunicados como sucede para las distribuciones D100 y Stagger. En el caso de la distribución Stagger se consigue reducir significativamente la cantidad de datos a comunicar. Sabemos que en la programación con paso de mensajes se le asocia una identificación lógica (número) a cada procesador. La distribución Stagger es un caso atípico en el que las claves y punteros están asignados inicialmente a procesadores que no les corresponde como destino final en la ordenación. Con una reasignación de esos identificadores lógicos, haciendo que sí les correspondan como destinos finales, todo a nivel de programa, reducimos la comunicación de datos entre procesadores. En el caso de tener una distribución D100, además del ahorro de comunicación de cualquier envío de datos, también se ahorra trabajo en la ordenación local.

La comunicación de datos no tiene tanto peso en el SGI O2000 como en el computador basado en p630.

Finalmente, se quiere ver como varía el rendimiento global de PSKC-Radix sort para diferentes distribuciones de datos. En la gráfica de la izquierda de la Figura 5.5 se muestra el número de CSE que invierte PSKC-Radix sort para ordenar 4M datos en el computador basado en p630. En la gráfica de la derecha de la Figura se muestran los CSE para ordenar 2M datos en el SGI O2000. Las claves a ordenar son de 64 bits. En las gráficas se muestran los resultados de la ordenación de conjuntos con distribución de datos Random, Partial, S40, D50, D100 e Inverse para el procesador más lento. El comportamiento para el resto de distribuciones es parecido a una u otra distribución de las mostradas en la Figura.

Para el computador basado en módulos p630 se pueden distinguir dos tipos de resultados:

- Distribuciones de datos para las que no se puede reducir el volumen de datos comunicados. En este caso, el número de CSE aumenta cuando se pasa de 4 a 8 procesadores. La razón está en la disminución del ancho de banda entre procesadores de diferentes nodos (módulos SCM p630).

- Distribuciones de datos para las que se puede reducir el volumen de datos comunicados (Inverse) o incluso evitar la comunicación de ningún dato de tipo clave y puntero (D100). Esta reducción de comunicación se hace gracias a la reasignación de los identificadores de procesadores lógicos. Por consiguiente, se va reduciendo

el número de CSE a medida que aumenta el número de procesadores. En el caso de la distribución D100, que el número de CSE sea tan reducido se debe a que no se tienen que comunicar ni ordenar datos.

Para el computador SGI O2000, PSKC-Radix sort tiene un número de CSE similar para las diferentes distribuciones de datos. Al igual que antes, también se observa una importante reducción del número de CSE para la distribución de datos D100. Las razones son las mismas.

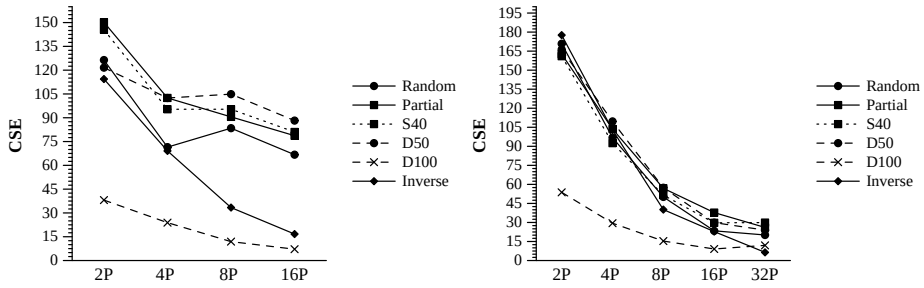

Figura 5.5: CSE totales en la ordenación con PSKC-Radix sort para diferentes distribuciones de datos. Los resultados son para ordenar 4M y 2M datos de clave y puntero de 64 bits, hasta 16 procesadores en el computador basado en p630 (izquierda) y 32 procesadores en el SGI O2000 (derecha) respectivamente.

5.2.3. Comparación con Load Balanced Radix sort

En esta sección se comparan PSKC-Radix sort y Load Balanced Radix sort [40]. Load Balanced Radix sort mejora otros métodos de ordenación en paralelo en memoria, demostrando ser hasta *dos veces* más rápido que los métodos más rápidos hasta 1998 [2, 17], el momento de iniciar el trabajo de esta tesis.

Las distribuciones de datos que se han elegido para la comparativa son:

- **Random:** Distribución con datos aleatorios que normalmente se usa para analizar el comportamiento de métodos de ordenación.
- **D100**: Distribución de datos donde todas las claves tienen el mismo valor. Es una de las distribuciones para las que Load Balanced Radix sort ofrece mejor rendimiento ya que el número de CSE invertidos en comunicación es muy pequeño.
- **Bucket y Stagger:** A parte de la distribución D100, éstas son las dos distribuciones de datos para las que Load Balanced Radix sort tiene mejor rendimiento.

Además, PSKC-Radix sort, a diferencia de lo que pasaba con D100, debe comunicar y ordenar los datos con estas distribuciones de datos. Por lo tanto se estarán comparando comportamientos más parecidos para ambos algoritmos ya que ambos deben comunicar y ordenar los datos para estas distribuciones.

- **S40:** Distribución de datos para la que PSKC-Radix sort tiene que aplicar Counting Split (más costoso que Reverse Sorting) y para la que se ahorra menos trabajo y comunicación de datos. Hay poco sesgo, pero el suficiente como para tener que particionar con Counting Split.

En las gráficas de arriba de la Figura 5.6 se muestra el *tiempo de ordenación en segundos* variando el número de procesadores de 2 a 16 procesadores Power4 en el computador basado en módulos p630 y variando el número de datos a ordenar de 1M datos de clave y puntero, entre todos los procesadores, hasta 4M datos por procesador. En las gráficas de abajo se muestra el tiempo invertido variando el número de procesadores R10K de 2 a 32 en el SGI O2000 y el número de datos de 1M datos de clave y puntero, entre todos los procesadores, hasta 2M datos por procesador. En las gráficas de la izquierda, cada curva representa el tiempo de ordenación de un número fijo de datos variando el número de procesadores. En las gráficas de la derecha, cada curva representa el tiempo de ordenación invertido por un número fijo de procesadores variando el número de datos a ordenar. Los resultados son para la ordenación de datos de 64 bits y una distribución Random, con Load Balanced Radix sort (línea discontinua) y PSKC-Radix sort (línea continua). En las Figuras 5.7 y 5.8 se muestra el mismo tipo de experimentos que en la Figura 5.6, pero para distribuciones de datos D100 y Bucket.

En general, se puede observar que PSKC-Radix sort precisa menos de la mitad de tiempo y número de procesadores de lo que necesita Load Balanced Radix sort para ordenar la misma cantidad de datos (véase las gráficas de la izquierda).

Para un computador basado en p630, para una distribución de datos Random y utilizando 16 procesadores, por ejemplo, Load Balanced Radix sort invierte el mismo tiempo en ordenar 1M datos que PSK-Radix sort ordenando 8M datos con la misma cantidad de procesadores. Esta equivalencia en tiempo de ordenación se muestra con dos marcas (rombos) más grandes que el resto de marcas de la gráfica superior derecha de la Figura 5.6. También, otro ejemplo lo tenemos para una distribución Bucket y 4 procesadores, PSKC-Radix sort es capaz de ordenar 4M datos de clave y puntero en el mismo tiempo que Load Balanced Radix sort ordena 1M datos. Esta equivalencia se muestra con dos cruces más grandes en la gráfica superior derecha de la Figura 5.8.

En el caso del SGI O2000, PSKC-Radix sort es capaz de ordenar 2 veces más datos que Load Balanced Radix sort en el mismo tiempo, tanto para la distribución Random como para la distribución Bucket. Además, para ambos algoritmos con 32 procesadores, el decremento más significativo del tiempo de ordenación se produce cuando reducimos el número de datos a ordenar de 8M a 4M datos (véase las gráficas

de la derecha para la curva de 32 procesadores). Esto es debido a que la parte de los datos que le corresponde a cada procesador empieza a caber en cache cuando el conjunto total de datos es menor o igual a 4M datos. Por consiguiente, los procesadores pueden explotar mejor la jerarquía de memoria.

Para ambos computadores y ambos algoritmos, el tiempo invertido en ordenar una distribución D100 es el más pequeño. PSKC-Radix sort también es más rápido que Load Balanced Radix sort en este caso.

La razón por la que PSKC-Radix sort es capaz de ordenar tanto más rápido que Load Balanced Radix sort es debido a varios aspectos como la reducción de comunicación de datos, manteniendo un desequilibrio de carga moderado y la explotación de la localidad de los datos. Estos aspectos se analizan a continuación.

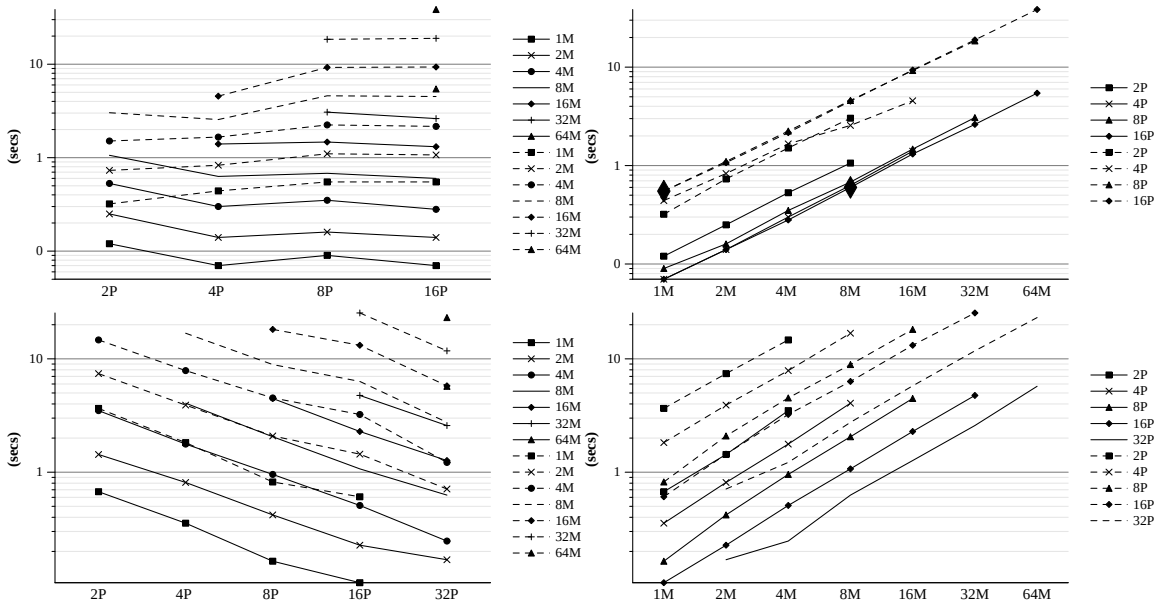

Figura 5.6: Tiempo en segundos para PSKC Radix sort (línea continua) y Load Balanced Radix sort (discontinua) para una distribución Random. En las gráficas de la izquierda cada línea corresponde con un número fijo de elementos a ordenar variando el número de procesadores. En las gráficas de la derecha cada línea corresponde con un número fijo de procesadores variando el número de elementos a ordenar. Resultados para el computador basado en p630 en las gráficas de arriba. Abajo para el SGI O2000.

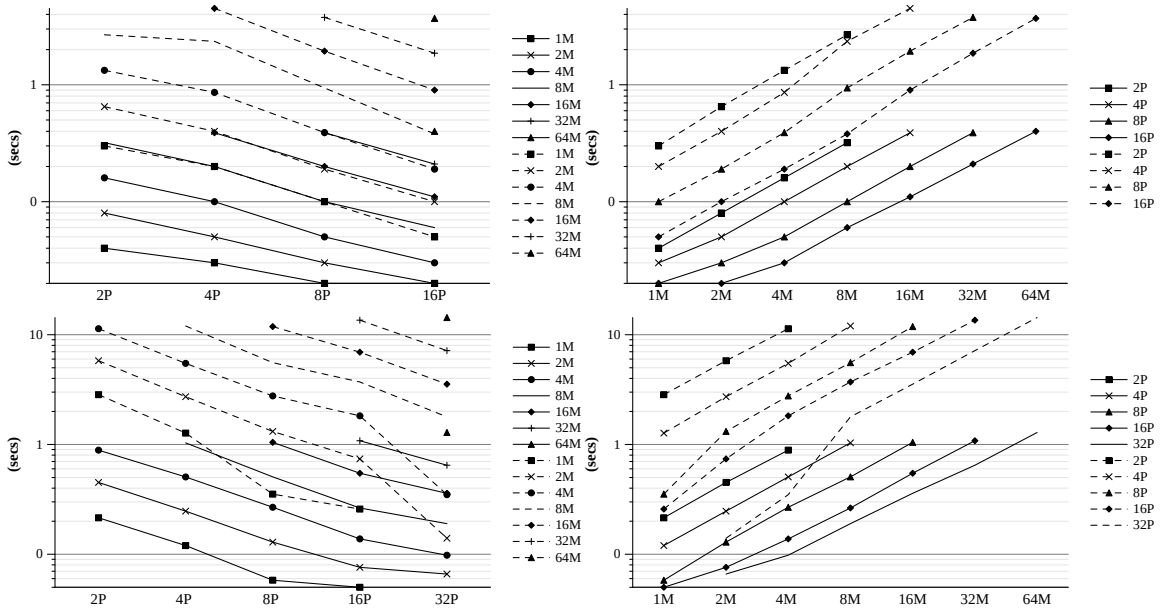

Figura 5.7: Tiempo en segundos para PSKC Radix sort (línea continua) y Load Balanced Radix sort (discontinua) para una distribución D100. Resultados para el computador basado en p630 en las gráficas de arriba. Abajo para el SGI O2000.

Comunicación

Para entender los resultados anteriores se utilizan las Figuras 5.9 y 5.10 donde se comparan Load Balanced Radix sort (BR en las Figuras) y PSKC-Radix sort (P en las Figuras) para el computador basado en p630 y el SGI O2000, utilizando 16 y 32 procesadores respectivamente y variando la cantidad de datos a ordenar. Los diagramas de barra muestran cómo los CSE se distribuyen en los diferentes pasos de los algoritmos para una distribución Random (R), S40 (S) y Bucket(B) (gráfica de arriba de cada figura) y D100 (D) y Stagger (St) (gráfica de abajo). En las gráficas se muestra la ordenación para datos de 64 bits.

El primer aspecto a destacar es que la comunicación de datos es aproximadamente 8 veces más rápida para PSKC-Radix sort que para Load Balanced Radix sort. Esto es porque Load Balanced Radix realiza ocho pasos de broadcast de contadores y de comunicación al ordenar datos de 64 bits. Hay tantos pasos de comunicación de contadores y datos como pasos de movimiento en el algoritmo Radix sort. Así, como Radix sort trabaja con dígitos de 8 bits, se tienen $\frac{64}{8} = 8$ dígitos y, por lo tanto, se realizan 8 pasos de movimientos. En cambio, PSKC-Radix sort sólo realiza 1 comunicación de datos, la comunicación de los muestreos tomados, y el broadcast de contadores y separadores según la técnica de particionado que se decida.

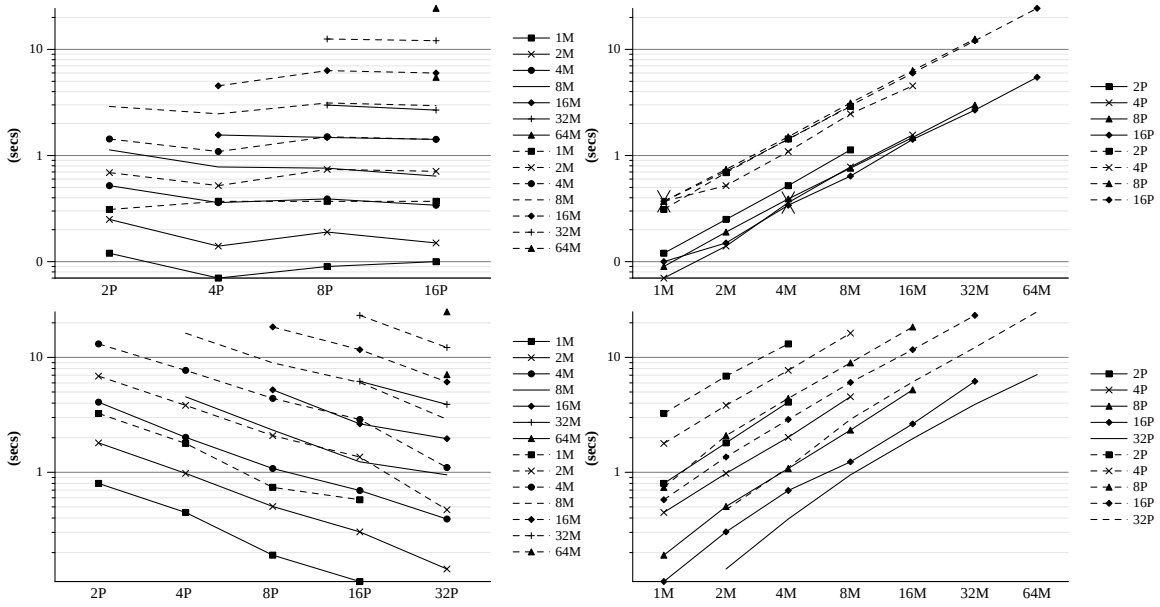

Figura 5.8: Tiempo en segundos para PSKC Radix sort (línea continua) y Load Balanced Radix sort (discontinua) para una distribución Bucket. Resultados para el computador basado en p630 en las gráficas de arriba. Abajo para el SGI O2000.

Además, PSKC-Radix sort intenta reducir el volumen de datos en el paso de comunicación. Esto se puede observar para la distribución Stagger en las Figuras 5.9 y 5.10 donde la comunicación para Stagger es mínima. La razón para esto se halla en la propia distribución que hace que los datos pertenecientes a un grupo de particiones globales se concentren en unos determinados procesadores. Esto permite ahorrarnos comunicación de datos con la reasignación de los identificadores de procesadores lógicos. Load Balanced Radix sort no se preocupa de reducir la cantidad de datos a comunicar en cada paso de comunicación.

Equilibrio de Carga

El equilibrio de carga se trata de forma diferente en PSKC-Radix sort que en Load Balanced Radix sort. PSKC-Radix sort permite un cierto desequilibrio de carga con el objetivo de poder tener particiones completas y con valores disjuntos en cada procesador; consiguiendo hacer un único paso de comunicación de datos y, posteriormente, explotar la jerarquía de memoria en la ordenación de las particiones. Load Balanced Radix sort, por otra parte, obtiene un equilibrio de carga perfecto pero sin preocuparse de conseguir particiones completas en cada procesador. Sin embargo, aunque PSKC-Radix sort permite desequilibrio de carga, las Figuras 5.9 y 5.10 muestran que

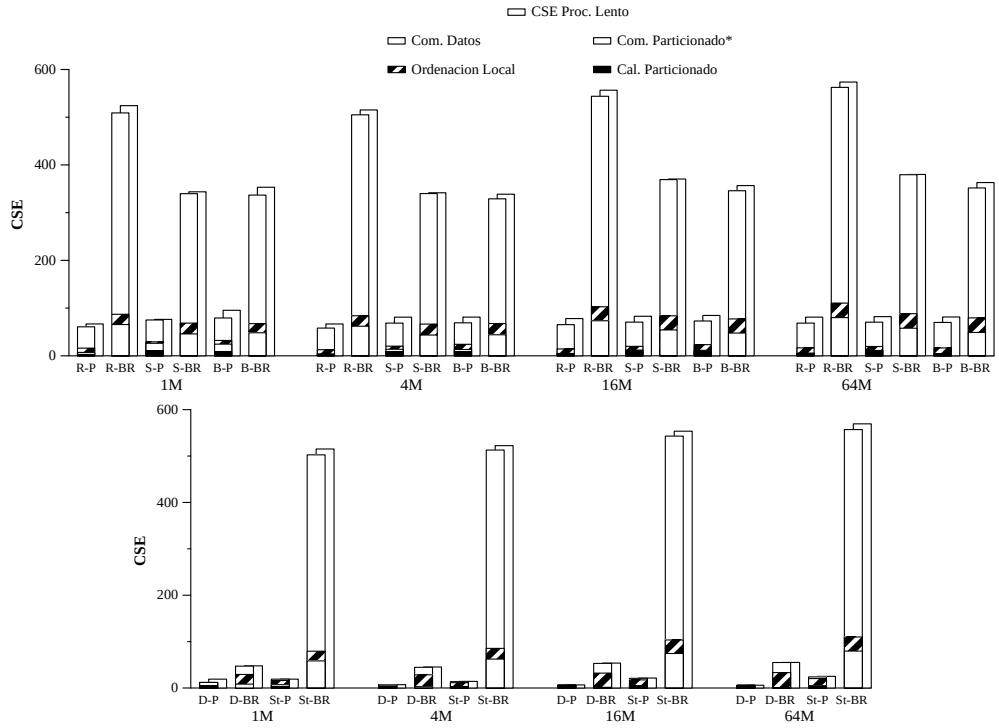

Figura 5.9: Comparativa entre PSKC-Radix sort (P) y Load Balanced Radix sort(BR) para las distribuciones Random (R), S40 (S) y Bucket (B) (arriba), y D100 (D) y Stagger (St) (abajo). Los resultados son para un computador basado en p630 y 16 procesadores. (*) Com. Particionado es la comunicación de contadores para Load Balanced Radix sort.

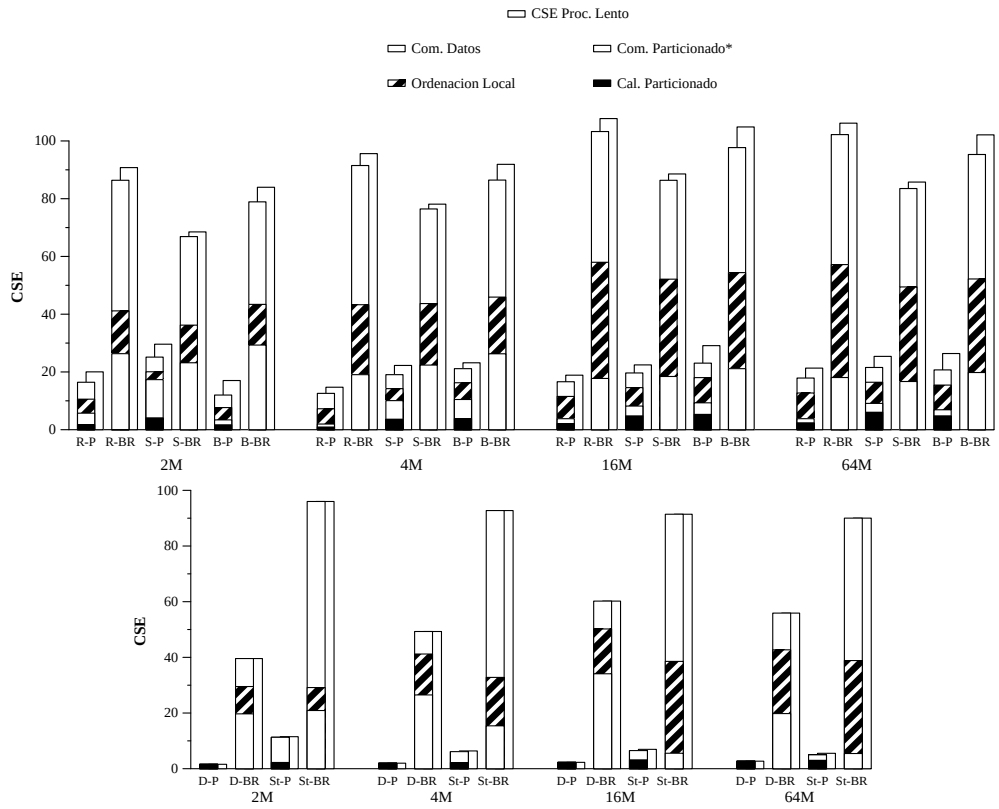

Figura 5.10: Comparativa entre PSKC-Radix sort (P) y Load Balanced Radix sort(BR) para las distribuciones Random (R), S40 (S) y Bucket (B) (arriba), y D100 (D) y Stagger (St) (abajo). Los resultados son para el SGI O2000 y 32 procesadores. (*) Com. Particionado es la comunicación de contadores para Load Balanced Radix sort.

la diferencia entre el número de CSE medio y el número de CSE del procesador más lento (debida al desequilibrio) no es muy alta y no varía mucho entre Load Balanced Radix sort y PSKC-Radix sort; aunque para conjuntos pequeños de datos, el desequilibrio de carga se hace mayor para PSKC-Radix sort. El que haya mayor desequilibrio de carga para estos conjuntos de datos se debe a que se está permitiendo un desequilibrio de carga fijo. Si se quisiera conseguir un equilibrio de carga perfecto con PSKC-Radix sort esto supondría hacer muchas iteraciones de Reverse Sorting o, en el caso de Counting Split, reparticionar muchas particiones frontera. En ambos casos, el coste de particionado podría ser excesivo (para más detalles, ver Capítulo 3).

Ordenación local

Otro aspecto a destacar son las diferencias en CSE que hay en la ordenación local de las particiones. El número de CSE para la ordenación local con Radix sort en Load Balanced Radix sort varía con el tamaño del conjunto de datos a ordenar, sobretodo en el caso de el SGI O2000 (Figura 5.10). Esto es debido a dos aspectos:

1. Excesiva Comunicación: Es probable que las claves se comuniquen de un procesador a otro para cada dígito de las claves que se quiere ordenar. Por consiguiente, no es fácil explotar la localidad temporal de esos datos para el siguiente dígito a ordenar ya que es posible que se hayan enviado a otro procesador, donde se trabajará con ellos. Además, todas las claves se tienen que procesar otra vez para cada dígito, lo que hace más difícil explotar la localidad temporal si el conjunto de datos es grande. A todo esto, y en el caso de datos de 64 bits, el número de dígitos y, por lo tanto, de movimientos y comunicación de datos, es elevado (8 movimientos).

   En el caso de PSKC-Radix sort se puede explotar la localidad temporal de los datos ya que sólo es preciso comunicar los datos una vez. Como efecto colateral, al particionar los datos para reducir la comunicación, también se reduce el tamaño de cada una de las particiones que se ordenarán posteriormente. Esto ayudará a que se explote mejor la localidad de los datos.

2. Radix sort no explota la jerarquía de memoria: Cuando se pasa de ordenar 4M datos a ordenar 16M datos de 64 bits, para el SGI O2000 con procesadores R10K, los datos locales de cada procesador no caben en el segundo nivel de memoria cache ni se pueden mapear por el TLB. Así, al ordenar estos datos con el algoritmo Radix sort, tal y como hace Load Balanced Radix sort, no se consigue explotar la jerarquía de memoria, tal y como se explica en detalle en el Capítulo 2. En este caso el número de fallos de memoria cache y de TLB por elemento aumenta.

En PSKC-Radix sort, los datos se ordenan localmente con SKC-Radix sort, que es capaz de explotar la jerarquía de memoria del procesador (ver Capítulo 4).

Speedup

Finalmente, en la Figura 5.11 se muestra una comparativa de los speedups para PSKC-Radix sort y Load Balanced Radix sort para distribuciones Random y Bucket. En la gráfica de la izquierda se muestra el speedup de los dos algoritmos en el computador basado en p630 para ordenar 4M datos de clave y puntero de 64 bits entre 2 a 16 procesadores. En la gráfica de la derecha se muestra para el SGI O2000 y la ordenación de 2M datos de clave y puntero de 64 bits entre 2 a 32 procesadores. En ambos casos, el speedup se obtiene con respecto al número de CSE invertidos por un procesador que ordena los datos con Quicksort. La clara diferencia entre los dos algoritmos muestra que PSKC-Radix sort se adapta mejor que Load Balanced Radix sort a las características de los datos a ordenar y al computador sobre el que se ordena.

La degradación de speedup en el computador basado en p630 se debe a la disminución del ancho de banda cuando se pasa a tener comunicación entre diferentes nodos.

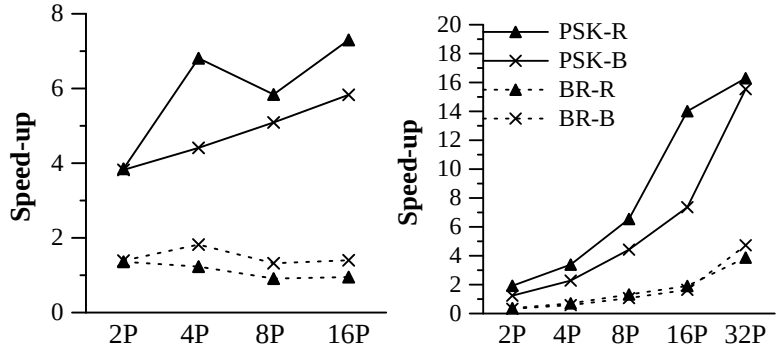

Figura 5.11: Speedup de los algoritmos PSKC-Radix y Load Balanced Radix sort respecto Quicksort para el computador basado en p630 y 4M datos de 64 bits (izquierda) y el SGI O2000 y 2M datos de 64 bits(derecha). En las gráficas, R significa Random y B significa Bucket.