Apéndice C

Modelos para p630

En este Apéndice se describen los modelos para el algoritmo de Radix sort y las técnicas de particionado que se proponen, en un computador basado en p630 con procesadores Power4. La comparativa entre modelo y ejecuciones reales se realiza en los Capítulos 2 y 3. En el Apéndice B se explican los aspectos que se tienen en cuenta para el modelado cualitativo de los algoritmos. El objetivo de estos modelos es entender el coste de cada una de las partes de los algoritmos y ver cual es la tendencia en el comportamiento de los algoritmos propuestos para este computador en cuestión. Se recomienda leer las características del computador basado en módulos p630 con procesadores Power4 que se explicaron en el Capítulo 1. En la Tabla 1.1 del mismo Capítulo se encuentran resumidas las características más importantes del computador.

C.1. Radix sort

Modelo

La Figura C.1 muestra el grafo de lecturas y escrituras de los pasos de conteo y movimiento de la versión mejorada del Radix sort, descrita en el Capítulo 2. A continuación explicaremos los costes de cada uno de los pasos, basándonos en las dependencias entre las instrucciones y los accesos a memoria para un Power4 (procesador de un módulo p630).

El coste según nuestro modelo para cada uno de los pasos del algoritmo es:

1. **Paso (1)- Inicialización de Contadores:** En este paso se inicializan los 2^{b_i} contadores pertenecientes a cada uno de los d dígitos con b_i bits. Se construye el algoritmo de forma que b_i sea óptimo, por lo que el número de CSE para este paso se puede despreciar ya que en este caso b_i no es muy grande. En el Apéndice A se explica la forma de obtener el tamaño de dígito óptimo.

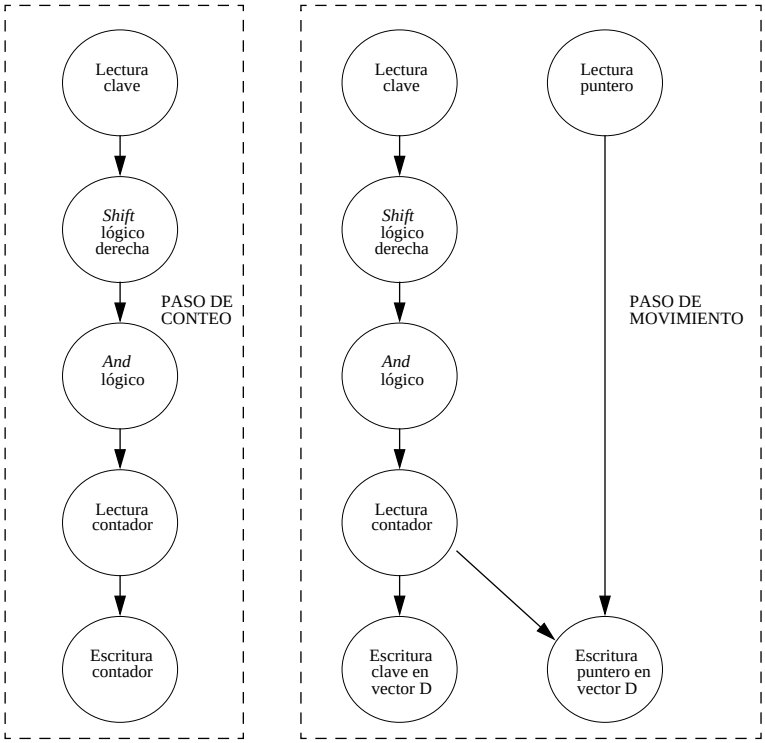

Figura C.1: Grafo de dependencias entre las operaciones de memoria para el paso de conteo y de un movimiento en el Radix sort mejorado.

2. **Paso (2)- Conteo:** Nos centraremos en los CSE para cada iteración del bucle de este paso y lo descompondremos en diferentes aspectos: CSE por accesos al primer nivel de memoria cache, CSE por fallos de acceso a los diferentes niveles de memoria cache, CSE debido a fallos de TLB y CSE por operaciones que no son de lectura/escritura.

   - **Accesos al primer nivel de memoria cache**. Se realiza una lectura para obtener la clave del vector fuente (C_a ciclos). Con esta clave se actualiza el valor de un contador por cada dígito, es decir, d contadores. Esta actualización supone d lecturas que se pueden lanzar a ejecutar de dos en dos ($\frac{d}{2} + C_a - 1$ ciclos) y d escrituras que se tienen que lanzar una a una ($d + C_a - 1$ ciclos). Los CSE debido a accesos al primer nivel de cache son:

$$CSE_{cpu\_accesos} = C_a + (\frac{d}{2} + C_a - 1) + (d + C_a - 1) = \frac{3d}{2} + 3C_a - 2$$

   - **Fallos de memoria cache**. Por un lado, tenemos la lectura de las claves del vector fuente, que es secuencial. Tal y como comentamos, el Power4 dispone de un prefetch hardware que se activa después de 4 fallos en accesos secuenciales y se desactiva cada vez que se accede a una nueva página de memoria. Por consiguiente, los CSE debido a esos 4 fallos por cada vez que se tiene que activar el prefetch hardware son $4(C_{f1} + C_{f2} + C_{f3})/R_{page}$.
     Por otro lado, se debe realizar la actualización de los contadores. Supondremos que las lecturas no fallan ya que el número de contadores es reducido y se acceden frecuentemente; por lo que seguramente están en el primer nivel de cache. En cuanto a las escrituras, aunque la actualización se realiza en el segundo nivel de memoria cache para mantener la coherencia de cache en ese nivel, no contabilizaremos ningún CSE por esta actualización. Esto lo hacemos así porque suponemos que la latencia de las escrituras se pueden solapar con otras operaciones ya que existen buffers de escritura. Así, los CSE según fallos de memoria cache son:

$$CSE_{mem\_mc} = 4(C_{f1} + C_{f2} + C_{f3})/R_{page}$$

   - **Fallos de TLB.** Tenemos la penalización por fallos obligatorios o de capacidad de TLB para leer el vector fuente. Los CSE debido a fallos de TLB son:

$$CSE_{TLB} = C_{fTLB}/R_{page}$$

   - **Operaciones que no son lecturas ni escrituras.** Es un coste despreciable ya que son los accesos al primer nivel de cache para actualizar los contadores los que determinan el coste del conteo.

El número de CSE para este paso es el siguiente:

$$\begin{aligned} CSE_{cont} &= C_a + (\frac{d}{2} + C_a - 1) + (d + C_a - 1) \\ &+ 4(C_{f1} + C_{f2} + C_{f3})/R_{page} \\ &+ C_{fTLB}/R_{page} \end{aligned}$$

3. **Paso (3)- Suma Parcial:** El coste de este paso también lo podemos despreciar ya que el número de bits b_i no es grande.

4. **Paso (4)- Movimientos:** Los CSE para cada movimiento (uno por dígito, en total d dígitos) lo podemos dividir en los mismos puntos en que dividimos los CSE en el paso de Conteo. Los CSE para un paso de movimiento son:

   - **Accesos al primer nivel de memoria cache.** Lectura de la clave y el puntero del vector fuente. Estas dos lecturas se pueden ejecutar en paralelo ya que disponemos de dos unidades funcionales de lectura. Este acceso tarda C_a ciclos. Después, calculamos el valor del dígito y leemos el contador asociado (C_a ciclos). Finalmente escribimos la clave y el puntero en el vector destino y actualizamos el contador. Esto significa que hay tres escrituras independientes en el primer nivel de cache que se traducen en $3 + C_a - 1$ ciclos. Por lo tanto el número de CSE de esta parte es:

   $$CSE_{cpu\_accesos} = 2 + 3C_a$$

   - **Fallos de memoria cache**. En la lectura del vector fuente tenemos la misma penalización vista en el paso de conteo según este mismo punto ($4(C_{f1} + C_{f2} + C_{f3})/R_{page}$ ciclos). No contabilizamos las escrituras de la clave y puntero en el vector destino. Al igual que se supuso en el paso de conteo, estos accesos se solapan con las lecturas y las operaciones que no acceden a memoria gracias a los *buffers* de escritura. Así, los CSE por penalización en los fallos de acceso a la memoria cache son:

   $$CSE_{mem\_mc} = 4(C_{f1} + C_{f2} + C_{f3})/R_{page}$$

   - **Fallos de TLB.** Tenemos la penalización por fallos de capacidad de TLB al leer el vector fuente si las estructuras necesarias para realizar la ordenación de los datos no se pueden mapear por el TLB. Los CSE por estos fallos son C_{fTLB}/R_{page}.
     Al escribir en el vector destino tenemos la misma penalización por fallos obligatorios o de capacidad: C_{fTLB}/R_{page}. Además, si el número de particiones es muy elevada, la probabilidad de fallos de TLB por conflicto (p_f)

en cada escritura al vector destino D puede ser alta. Así, el número de CSE por fallos de TLB es:

$$\begin{aligned} CSE_{mem\_TLB} &= C_{fTLB}/R_{page} + \\ &+ C_{fTLB}/R_{page} + \\ &+ C_{fTLB}p_f \end{aligned}$$

- **Operaciones que no son lecturas ni escrituras.** Este un coste despreciable.

Por lo tanto, los CSE para *un sólo movimiento* son :

$$\begin{aligned} CSE_{mov} &= (2 + 3C_a) \\ &+ 4(C_{f1} + C_{f2} + C_{f3})/R_{page} + \\ &+ C_{fTLB}(\frac{2}{R_{page}} + p_f) \end{aligned}$$

El número total de CSE del algoritmo Radix sort es entonces:

$$\begin{aligned} CSE_{sec\_radix} &= CSE_{cont} + d(CSE_{mov}) \\ CSE_{sec\_radix} &= 3(1+d)C_a + \frac{7}{2}d - 2 \\ &+ 4(1+d)(C_{f1} + C_{f2} + C_{f3})/R_{page} + \\ &+ C_{fTLB}(\frac{2d+1}{R_{page}} + dp_f) \end{aligned}$$

que sale de la suma de todos los costes anteriores.

C.2. Reverse Sorting

C.2.1. Algoritmo Secuencial

Modelo

En esta sección se modelará primero el número de CSE para una iteración de Reverse Sorting. El modelo para calcular el coste de una iteración de Reverse Sorting es similar a la ordenación de un dígito con Radix sort.

El número de CSE de una iteración de Reverse Sorting según nuestro modelo secuencial es el siguiente:

1. **Paso (1) - Inicialización de contadores:** Coste despreciable.

2. **Paso (2) - Conteo:** El razonamiento es muy parecido al que hicimos para el Radix sort. Los CSE de este paso se descomponen en CSE de acceso a primer nivel de cache, fallos de memoria cache, fallos de TLB y operaciones que no son ni lecturas ni escrituras.

   - **Accesos al primer nivel de memoria cache.** Se realiza una lectura para cada clave (C_a ciclos) y se actualiza un contador. La actualización supone una lectura (C_a ciclos) y una escritura (C_a ciclos). El número de CSE para los accesos al primer nivel de memoria cache es:

   $$CSE_{cpu\_accesos} = 3C_a$$

   - **Fallos de memoria cache.** El razonamiento es exactamente el mismo que para el paso de conteo del algoritmo Radix sort. Por lo tanto el número de CSE es:

   $$CSE_{mem\_mc} = 4(C_{f1} + C_{f2} + C_{f3})/R_{page}$$

   - **Fallos de TLB.** Fallos obligatorios y de capacidad accediendo al vector fuente.

   $$CSE_{mem\_TLB} = C_{fTLB}/R_{page}$$

   - **Operaciones que no son lecturas ni escrituras.** Este coste lo podemos despreciar.

   Así el número de CSE del paso de conteo es de:

   $$\begin{aligned} CSE_{cont} &= 3C_a + \\ &+ 4(C_{f1} + C_{f2} + C_{f3})/R_{page} + \\ &+ C_{fTLB}/R_{page} \end{aligned}$$

3. **Paso (3) - Suma Parcial:** El coste de la suma parcial de los contadores es despreciable.

4. **Paso (4) - Movimiento:** Es el mismo paso que describimos para Radix sort pero suponiendo una probabilidad de fallo de TLB por conflicto igual a 0, es decir $p_f = 0$. Esto lo podemos suponer porque el dígito de Reverse Sorting, b_r, se eligió de tal forma que se redujeran los fallos de memoria cache y se minimizaran los fallos de TLB. Por consiguiente, el número de CSE para un movimiento es de:

$$\begin{aligned} CSE_{mov} &= 2 + 3C_a \\ &+ 4(C_{f1} + C_{f2} + C_{f3})/R_{page} + \\ &+ C_{fTLB}\frac{2}{R_{page}} \end{aligned}$$

El número de CSE de *una iteración* de Reverse Sorting es de:

$$\begin{aligned} CSE_{sec\_una\_it\_rev} &= CSE_{cont} + CSE_{mov} \\ CSE_{sec\_una\_it\_rev} &= 2 + 6C_a + \\ &+ 8(C_{f1} + C_{f2} + C_{f3})/R_{page} + \\ &+ C_{fTLB}\frac{3}{R_{page}} \end{aligned}$$

El número total de CSE de particionado con Reverse Sorting depende de la distribución de los datos con la que nos encontremos (más detalle en el Capítulo 3). Después de una iteración de Reverse Sorting, algunas de las particiones pueden ser demasiado grandes. En este caso se aplica una nueva iteración de Reverse Sorting a esas particiones. Este proceso se realiza recursivamente tal y como podemos observar en el Algoritmo 6 en el Capítulo 3. Esto puede suponer que el coste total de particionado sea elevado. El número total de CSE con el particionado con Reverse Sorting, CSE_{sec_rev}, lo expresamos como el número total de ciclos invertidos en el particionado con Reverse Sorting ($Ciclos_{sec\_rev}$) dividido entre el número total de elementos a ordenar (n). Así, CSE_{sec_rev} se expresa como:

$$\begin{aligned} Ciclos_{sec\_rev}(n) &= CSE_{sec\_una\_it\_rev} * n + \sum_{i=0}^{2^{b_r}} (Ciclos_{sec\_rev}(n_i)), \quad (n > R_{TLB}) \vee (n > R_{cache_i}) \\ Ciclos_{sec\_rev}(n) &= 0, en\ otro\ caso \\ CSE_{sec\_rev}(n) &= Ciclos_{sec\_rev}(n)/n \end{aligned}$$

donde n_i es el número de elementos de la partición de la cual tenemos que hacer el Reverse Sorting otra vez debido a que es demasiado grande ($(n > R_{TLB}) \vee (n > R_{cache_i})$). R_{TLB} es la cantidad de memoria, expresada en número de elementos de clave y puntero, que puede mapear el TLB. R_{cache_i} es la capacidad en memoria, expresada en número de elementos del nivel de cache del cual queremos explotar la localidad temporal de los datos.

Nótese que los CSE de los pasos de inicialización de contadores y suma parcial los hemos despreciado en el cómputo de $CSE_{sec\_una\_it\_rev}$ suponiendo que $2^{b_r} << n$. Sin embargo, el coste de estos dos pasos podría ser significativo si se realiza un número elevado de iteraciones de Reverse Sorting.

C.2.2. Algoritmo Paralelo

La descripción del modelo paralelo que vamos a seguir está en el Apéndice B. El modelo paralelo distingue entre CSE de comunicación (CSE_{com}) y de cálculo (CSE_{cal}) para cada uno de los pasos de los algoritmos descritos.

Modelo

El número de CSE para cada uno de los pasos del particionado paralelo con Reverse Sorting son los siguientes:

1. **Paso (1) - Reverse Sorting local:** El coste acumulado para las n_{rev} iteraciones de Reverse Sorting necesarias para obtener un buen equilibrio de carga entre los procesadores es el siguiente:

$$\begin{aligned} CSE_{cal} &= \sum_{i=1}^{n_{rev}} (\sum_{j=0}^{(2^{b_r})^{(i-1)}} (\frac{n_i^j}{n} CSE_{sec\_una\_it\_rev})) \\ CSE_{cal} &= CSE_{sec\_una\_it\_rev} \frac{1}{n} \sum_{i=1}^{n_{rev}} \sum_{j=0}^{2^{b_r(i-1)}} n_i^j \end{aligned}$$

   donde n_i^j es el número de elementos de la partición j-ésima después de i iteraciones de Reverse Sorting. Después de i iteraciones, el número de particiones creadas son $2^{b_r(i-1)}$. Nótese que $CSE_{sec\_it\_rev}$ son los CSE para *una iteración* de Reverse Sorting calculados en el apartado anterior suponiendo n elementos a ordenar. Por lo tanto, los CSE de una iteración de Reverse Sorting para una partición con n_i^j elementos contribuye con $\frac{n_i^j}{n}$ CSE, al total de CSE para n elementos.

2. **Paso (2) - Comunicación de contadores:** Después de la iteración i-ésima del Reverse Sorting, se tienen 2^{ib_r} contadores, uno para cada partición. Por lo tanto, usando la fórmula de la comunicación colectiva *transpose* para m datos en la Tabla B.1 del Apéndice B y sabiendo que se hacen n_{rev} iteraciones de Reverse Sorting, el número total de CSE es de:

$$\begin{aligned} CSE_{com} &= \sum_{i=1}^{n_{rev}} \tau/n + \alpha T_e(2^{ib_r}P - 2^{ib_r})/n \\ CSE_{cal} &= \sum_{i=1}^{n_{rev}} O(2^{ib_r}/n) \end{aligned}$$

El número de CSE invertidos en la comunicación de los contadores lo podemos suponer como el número de CSE gastados por un procesador en leer la información que recibe del resto de procesadores $(P-1)$. Cada procesador envía 2^{ib_r} contadores. Por lo tanto, eso hace un total de $2^{ib_r}(P-1)\alpha T_e$ ciclos. Los CSE de cálculo son los invertidos en procesar el total de contadores que se reciben, 2^{ib_r}. Los valores para α y τ se especifican en el Apéndice B. T_e aquí es 4 bytes.

En nuestra implementación *br* es 9 en la primera iteración de Reverse Sorting local, 5 para el resto de iteraciones, si son necesarias. Para el resto de iteraciones reducimos *br* a 5 para no aumentar considerablemente el coste de comunicación de los contadores.

3. **Paso (3) - Cálculo de distribución de las particiones:** En este paso se suman los elementos de cada partición local para cada procesador con el objetivo de obtener el número total de elementos de cada partición global. Después, con un recorrido por todos esos contadores podemos saber si después de esa iteración de Reverse Sorting se consiguió equilibrio de carga entre los procesadores. Por consiguiente, el número total de CSE de la evaluación, para todas las iteraciones de Reverse Sorting necesarias, es el número de CSE de realizar la suma de las particiones:

$$CSE_{cal} = \sum_{i=1}^{n_{rev}} \frac{1}{n} O(2^{ib_r})$$

4. **Paso (4) - Renombramiento:** Para cada uno de los P conjuntos de particiones, buscamos el procesador que:

   - Tenga el mayor número de elementos locales pertenecientes a ese conjunto de particiones
   - No esté asignado todavía a un grupo de particiones.

   El número de CSE para este paso es de :

$$CSE_{cal} = O(\frac{P^2}{n})$$

5. **Paso (5) - Comunicación de los datos:** Suponiendo un equilibrio de carga perfecto en la comunicación de los datos, cada procesador envía y recibe $n/P - \frac{n}{P^2}$ elementos en una comunicación de todos con todos. El número de CSE de esta operación *All to All*, usando la fórmula que aparece en la Tabla B.1 del Apéndice B, es de:

$$\begin{aligned} CSE_{com\_datos} &= \tau/n + \alpha T_e(n/P - \frac{n}{P^2})/n \\ CSE_{cal\_datos} &= O(\frac{n}{nP}) = O(\frac{1}{P}) \end{aligned}$$

donde T_e aquí es 8 o 16 bytes dependiendo de si hablamos de elementos de clave y puntero de 32 o 64 bits respectivamente.

Finalmente, el número total de CSE del algoritmo de Reverse Sorting paralelo es:

$$\begin{aligned}
CSE_{Par_rev} &= CSE_{Par_rev_{cal}} + CSE_{Par_rev_com} \\
CSE_{Par_rev_{cal}} &= CSE_{sec_una_it_rev} \frac{1}{n} \sum_{i=1}^{n_{rev}} \sum_{j=0}^{2^{b_r(i-1)}} n_i^j \\
&+ \frac{1}{n} O(\sum_{i=1}^{n_{rev}} 2^{(i+1)b_r}) \\
&+ O(\frac{P^2}{n}) \\
&+ O(\frac{1}{P}) \\
CSE_{Par_rev_{com}} &= \tau \frac{(n_{rev}+1)}{n} + \\
&+ \alpha T_e (\frac{1}{P} - \frac{1}{P^2} + \frac{P-1}{n} \sum_{i=1}^{n_{rev}} 2^{ib_r})
\end{aligned}$$

En la Figura C.2 se muestran los CSE_{Par_rev} para los casos en que se realizan 1, 2, y 3 iteraciones de Reverse Sorting para ordenar datos de 64 bits. En la gráfica de arriba de la Figura C.2, mostramos los CSE para la ordenación de 4M datos entre 2 a 16 procesadores. La gráfica de abajo de la Figura C.2 muestra los CSE para la ordenación de $4P$M datos entre 2 a 16 procesadores, donde P es el número de procesadores. Las conclusiones que se pueden extraer son las mismas que se comentaron en el Capítulo 3 para las ejecuciones reales.

Finalmente, en la Figura C.3 mostramos cuál sería el número de CSE de nuestro algoritmo si el computador basado en p630 tuviera un mismo ancho de banda para la comunicación entre procesadores de diferentes nodos que dentro de un mismo nodo. En este caso, el rendimiento del algoritmo no se degrada al pasar de 4 a 8 procesadores, o de 8 a 16 procesadores, es decir, el número de CSE sigue disminuyendo para un número de iteraciones al aumentar el número de procesadores.

En la misma Figura mostramos el número de CSE invertidos por el algoritmo para ese mismo computador si éste tuviera 32 procesadores. En este caso, cuando el número de iteraciones de Reverse Sorting es 3, los CSE invertidos en comunicación de contadores se incrementa significativamente Esto es debido al elevado número de contadores (cantidad de memoria) que se deben comunicar con 32 procesadores. El número total de contadores a comunicar es proporcional al número de procesadores según se ve en el modelo para el paso de comunicación de contadores del algoritmo.

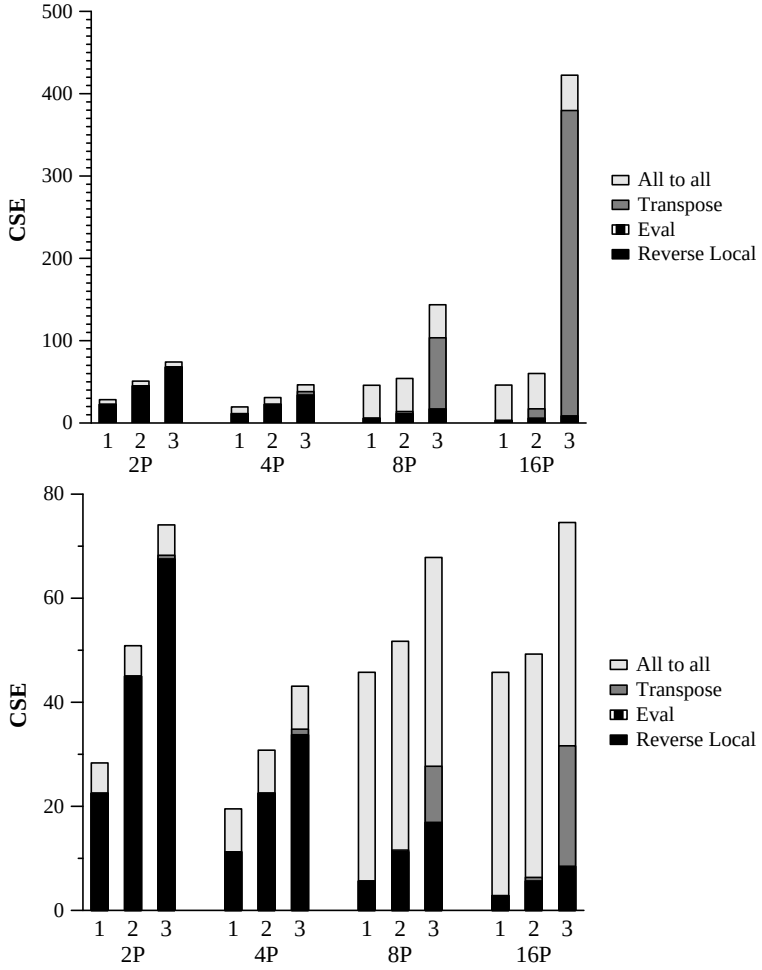

Figura C.2: CSE invertidos, según el modelo, en los pasos del Reverse Sorting paralelo para 4M (arriba) y 4PM (4M *por procesador*, abajo) datos de claves y punteros de 64 bits con el computador basado en p630 usando de 2 a 16 procesadores. Se muestran los resultados para cuando el Reverse Sorting debe hacer 1,2, ó 3 iteraciones.

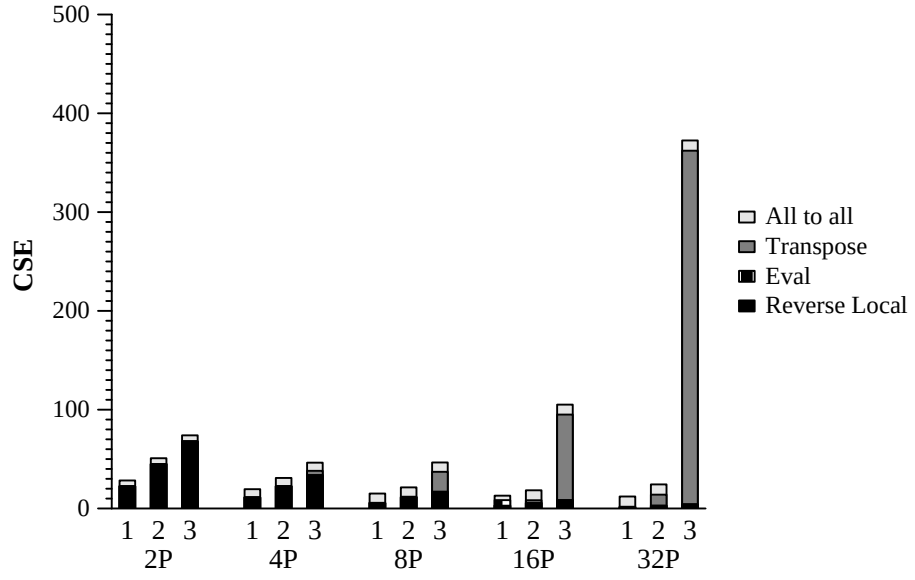

Figura C.3: CSE invertidos, según el modelo, en los pasos del Reverse Sorting paralelo para 4M de claves y punteros de 64 bits con el computador basado en p630 usando de 2 a 32 procesadores. Se muestran los resultados para cuando el Reverse Sorting debe hacer 1, 2, ó 3 iteraciones.

C.3. Counting Split

C.3.1. Algoritmo Secuencial

Modelo

Los CSE según el modelo para cada uno de los pasos del Counting Split, suponiendo que $q << n$ son los siguientes:

1. **Paso (1) - Muestreo.** Coste despreciable.
2. **Paso (2) - Ordenación del muestreo.** Coste despreciable.
3. **Paso (3) - Obtener separadores.** Coste despreciable.
4. **Paso (4) - Algoritmo de Conteo.** Aplicación del algoritmo de conteo con búsqueda binaria y el vector *indice_particion*. Este paso consta de 4 subpasos que describimos en el Capítulo 3 con el Algoritmo 8. El coste de los subpasos 4.1 (inicialización de contadores) y 4.3 (suma parcial de los contadores) se puede despreciar. Se modelarán los subpasos 4.2 y 4.4. Al igual que hicimos cuando se analizó el Radix sort, nos centraremos en los CSE debidos a los accesos al

primer nivel de memoria cache, fallos de memoria cache, fallos de TLB, y a operaciones que no son de lectura ni escritura. Para poder realizar este análisis se tiene en cuenta las operaciones que se realizan en una iteración del bucle y las dependencias entre las operaciones. La Figura C.4 muestra un grafo con las instrucciones y dependencias existentes entre ellas para los subpasos 4.2 (Conteo) y 4.4 (Movimiento).

Los CSE para los subpasos 4.2 y 4.4 son:

- **Paso (4.2) - Conteo.**
  - **Accesos al primer nivel de memoria cache.** Por un lado tenemos el acceso al vector fuente (C_a ciclos). Por otro lado, tenemos la escritura en el vector *indice_particion* y la actualización del contador correspondiente al valor del dígito. La escritura en el *inidice_particion* puede realizarse a la vez que se lee el contador (C_a ciclos). Después se debe escribir el valor actualizado del contador (C_a ciclos). Así, el número de CSE para los accesos al primer nivel de cache es de :

$$CSE_{cpu\_accesos} = 3C_a$$

  - **Fallos de memoria cache.** Sólo contabilizamos los accesos al vector fuente S. La penalización es la misma que para el Reverse Sorting. Los CSE de las escrituras en el contador y el vector *inidice_particion* no los contabilizamos porque consideramos que se pueden solapar con el resto de accesos y operaciones gracias a los *buffers* de escritura. Por lo tanto, los CSE para este punto son:

$$CSE_{mem\_mc} = 4(C_{f1} + C_{f2} + C_{f3})/R_{page}$$

  - **Fallos de TLB.** Éstos se pueden contar como los fallos obligatorios al acceder al vector fuente y al vector *indice_particion*. Los fallos obligatorios que se producen al acceder al vector *indice_particion* son despreciables. La probabilidad de fallar accediendo a este vector es de 1/4096 ya que cada elemento del vector es de 1 byte y cada página de memoria es de 4K bytes.
    Así pues, los CSE por fallos de TLB son:

$$CSE_{mem\_TLB} = C_{fTLB}/R_{page}$$

  - **Operaciones que no son de lectura ni escritura.** En el Counting Split este aspecto tiene un peso significativo. La búsqueda binaria es costosa ya que se tiene que hacer $\log(s)$ comparaciones (s es el número

de particiones que se quieren realizar). Cada una de estas comparaciones puede significar una rotura del flujo de ejecución donde una predicción incorrecta de un salto puede suponer un número elevado de ciclos de penalización. En el caso del Power4 esta penalización es de al menos de 12 ciclos. En la fórmula, el número de ciclos medio por comparación y penalización por predicciones incorrectas lo hemos expresado a través del factor k. k se ha calculado empíricamente. Los CSE para estas operaciones son de:

$$CSE_{cpu\_opers} = k\log(s)$$

La suma de CSE para el paso de Conteo es de:

$$\begin{aligned} CSE_{cont} &= 3C_a + \\ &+ 4(C_{f1} + C_{f2} + C_{f3})/R_{page} + \\ &+ C_{fTLB}/R_{page} + \\ &+ k\log(s) \end{aligned}$$

- **Paso (4.4)- Movimiento.** Este paso se modela con el mismo número de CSE que el paso de movimiento del Reverse Sorting aunque se diferencian en la forma de obtener a que partición debe ir un elemento.

  Con Reverse Sorting, el número de partición se obtiene a partir del valor del dígito de Reverse Sorting de la clave. Este cálculo tiene un coste despreciable. Counting Split, sin embargo, accede al vector *indice_particion* para averiguar el número de partición. De todas formas, el acceso al primer nivel de cache para obtener el elemento de este vector se puede solapar con el acceso al vector fuente. Los fallos obligatorios o de capacidad de memoria cache y de TLB, accediendo a este vector, son despreciables por las mismas razones que se comentaron el el paso 4.2, arriba.

  Por lo tanto, el número de CSE de este paso es el mismo que el del paso de movimiento de Reverse Sorting, es decir:

$$\begin{aligned} CSE_{mov} &= (2 + 3C_a) \\ &+ 4(C_{f1} + C_{f2} + C_{f3})/R_{page} + \\ &+ C_{fTLB}(\frac{2}{R_{page}}) \end{aligned}$$

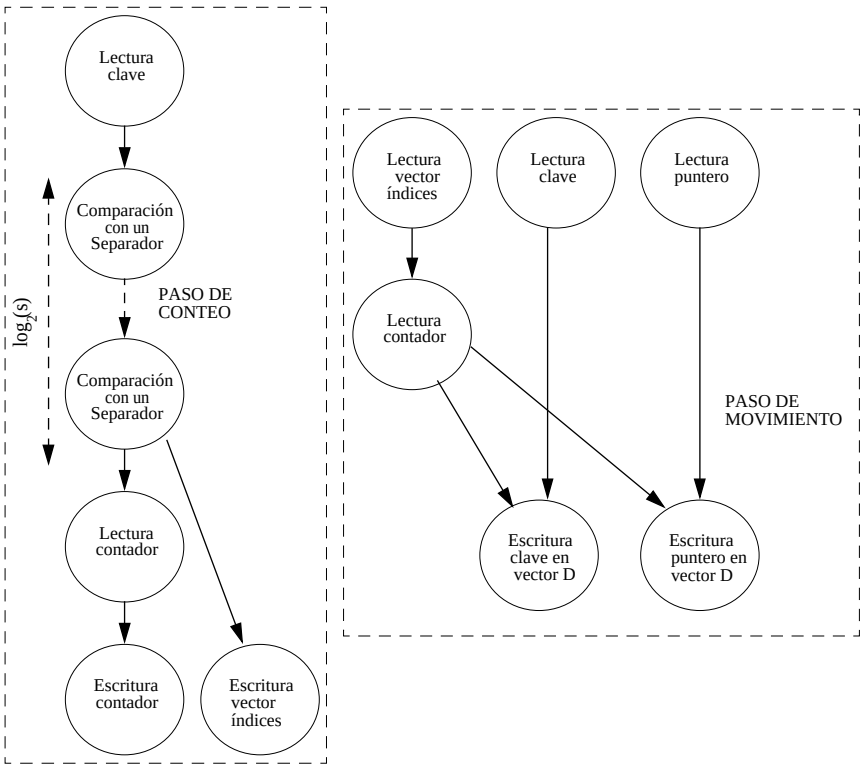

Figura C.4: Grafo de dependencias entre las operaciones de memoria para el paso de conteo y de movimiento en el Counting Split.

Así, el número total de CSE es:

$$\begin{aligned} CSE_{sec\_cs} &= CSE_{mov} + CSE_{cont} \\ CSE_{sec\_cs} &= 2 + 6C_a + \\ &+ 8(C_{f1} + C_{f2} + C_{f3})/R_{page} + \\ &+ C_{fTLB}(3/R_{page}) + \\ &+ k \log s \end{aligned}$$

C.3.2. Algoritmo Paralelo

Para modelar el particionado paralelo con Counting Split seguimos el modelo descrito en el Apéndice B.

Modelo

El número de CSE para cada uno de los pasos del Counting Split paralelo son los siguientes:

1. **Paso (1) - Muestreo del primer particionado**. En este paso todos los procesadores envían sus muestreos al procesador 0 mediante una comunicación de tipo *gather*. El número de CSE de la operación de comunicación *gather* se muestra en la tabla B.1 del Apéndice B. Así, el procesador 0 recibirá $q - q/P$ claves. Para este paso, los CSE son:

$$\begin{aligned} CSE_{com} &= \tau/n + \alpha T_e(q - q/P)/n \\ CSE_{cal} &= O(q/n) \end{aligned}$$

   donde T_e es 4 u 8 bytes según las claves sean de 32 o 64 bits respectivamente.

2. **Paso (2) - Ordenación con Radix sort.** El coste para esta operación es de:

$$CSE_{cal} = \frac{q}{n} CSE_{sec\_rad}$$

   CSE_{sec_rad} es el número de CSE de la ordenación con Radix sort, visto en la Sección C.1 de este mismo Apéndice. Este coste se multiplica por q/n para normalizarlo a los n elementos que se están ordenando, ya que la ordenación es de sólo q claves. Por consiguiente, para $n >> q$, este coste es despreciable.

3. **Paso (3) - Broadcast del vector de separadores** $vector\_separadores_{P_0}$. El procesador P_0 hace el broadcast de $s - 1$ claves al resto de procesadores. Para simplificar el modelo se ha calculado los CSE suponiendo el broadcast de s

separadores. Para calcular el coste de este paso se utiliza la fórmula mostrada en la Tabla B.1 del Apéndice B para la operación de broadcast.

El número de CSE para este paso es de:

$$\begin{aligned} CSE_{com} &= 2(\tau/n + \alpha T_e(s - s/P)/n) \\ CSE_{cal} &= O(s/n) \end{aligned}$$

donde T_e es 4 u 8 bytes dependiendo de si se ordenan claves de 32 o 64 bits respectivamente.

4. **Paso (4) - Counting Split Local.** El número de CSE para este paso es de:

$$CSE_{cal} = \frac{1}{P} CSE_{sec\_cs}$$

donde CSE_{sec_cs} es el número de CSE del particionado secuencial Counting Split. Aquí se calculan estos CSE para el Counting Split local a cada procesador. Estos CSE del Counting Split se multiplican por $\frac{1}{P}$ para normalizar los CSE por el número total de elementos a ordenar, es decir n. En realidad se multiplican por $\frac{n/P}{n}$ ya que cada procesador hace el particionado con Counting Split de $\frac{n}{P}$ elementos pero el total de elementos ordenados es n, por lo que este factor se queda en $\frac{1}{P}$.

5. **Paso (5) - Comunicación de los contadores.** Deben notar que tenemos que hacer la comunicación de $2s$ contadores, un contador para cada partición que obtenemos. Tenemos las particiones de duplicados (impares) y no duplicados (pares).

   El número de CSE para este paso es de:

$$\begin{aligned} CSE_{com} &= \tau/n + \alpha T_e(2sP - 2s)/n \\ CSE_{cal} &= O(2sP/n) \end{aligned}$$

   donde T_e es 4 bytes, el tamaño de cada contador.

6. **Paso (6) - Cálculo de la distribución de las particiones.** Es el mismo cálculo que hacemos para el Reverse Sorting para obtener una buena distribución de las $2s$ particiones. Este coste es despreciable.

7. **Paso (7) - Comunicación de los muestreos de las particiones fronteras.** Se calcula el número de CSE suponiendo que tendremos $P - 1$ particiones fronteras que *NO* son de duplicados, es decir, el peor caso que nos podemos encontrar ya que en este caso tenemos que comunicar el máximo número de muestreos.

Cabe recordar que el número de muestreos para este paso es q' ($q' < q$, ver Capítulo 3 para más detalles) y que los muestreos se comunican de todos los procesadores al procesador P_0.

Los CSE para este paso son:

$$\begin{aligned} CS_{com} &= (P-1)\tau/n + \alpha T_e(q' - q'/P)/n \\ CS_{cal} &= (P-1)O(q'/n) \end{aligned}$$

donde T_e es 4 u 8 bytes dependiendo de si son claves de 32 o 64 bits respectivamente.

8. **Paso (8) - Ordenación con Radix sort de los muestreos.** Para cada partición frontera tenemos que ordenar los datos de la muestra. El muestreo para cada partición frontera tiene q' claves. Así, el número total de CSE de este paso es la suma de CSE de cada una de las ordenaciones de cada muestreo. Hay un total de P-1 vectores de muestreo, por lo que el número de CSE es:

$$CS_{cal} = (P-1)(\frac{q'}{n} CSE_{sec\_radix})$$

9. **Paso (9) - Broadcast de los vectores de separadores.** Para cada partición frontera, el procesador P_0 hace el broadcast de un vector de separadores. En el peor de los casos, se hace el broadcast de $P-1$ vectores de separadores. Cada vector tiene s' separadores. Por lo que el número de CSE es de:

$$\begin{aligned} CSE_{com} &= (P-1)2(\tau/n + \alpha T_e(s' - s'/P)/n) \\ CSE_{cal} &= (P-1)O(s'/n) \end{aligned}$$

donde T_e es 4 u 8 bytes dependiendo de si son claves de 32 o 64 bits respectivamente.

10. **Paso (10) - Counting Split local para cada partición frontera.** Suponiendo el peor caso, cada procesador debe hacer el particionado con Counting Split de $P-1$ particiones frontera. El número de separadores que se utilizan son s'. Por lo tanto, el número de CSE de este paso para cada procesador es:

$$CSE_{cal} = \sum_{i=1}^{P-1} (\frac{n_i}{n} CSE_{sec\_cs}) = \frac{CSE_{sec\_cs}}{n} \sum_{i=1}^{P-1} n_i$$

donde n_i es el número de elementos de la partición frontera i local a cada procesador.

11. **Paso (11) - Comunicación de los contadores para cada partición frontera.** En el particionado de las particiones fronteras, en el paso anterior, no se distinguen entre particiones de duplicados y no duplicados. Por lo tanto, para cada partición frontera comunicaremos s' contadores, y no $2s'$.

    Los CSE para este paso son:

$$\begin{aligned} CSE_{com} &= (P-1)(\tau/n + \alpha T_e(s'P - s')/n) \\ CSE_{cal} &= (P-1)O(s'P/n) \end{aligned}$$

    donde T_e es 4 bytes.

12. **Paso (12) - Renombramiento.** El mismo tipo de renombramiento que hacemos para el Reverse Sorting paralelo. Este coste es despreciable.

13. **Paso (13) - Comunicación de los datos.** Suponiendo que todos los procesadores envían y reciben la misma cantidad de elementos, $\frac{n}{P} - \frac{n}{P^2}$, los CSE se calculan de la misma forma que calculamos los CSE de comunicación de datos para el particionado con Reverse Sorting Paralelo:

$$\begin{aligned} CS_{com} &= \tau/n + \alpha T_e(n/P - \frac{n}{P^2})/n \\ CS_{cal} &= O(n/(nP)) \end{aligned}$$

    donde T_e es 8 o 16 bytes dependiendo de si se está ordenando, respectivamente, elementos de clave y puntero de 32 o 64 bits cada uno.

La Figura C.5 muestra los CSE según el modelo desarrollado. En la figura se muestran los CSE del algoritmo variando el parámetro q (256P (1 en la Figura), 512P (2), 1024P (3), 2048P (4) y 8192P (5)). En la gráfica de arriba se muestran los CSE cuando el número total de elementos a ordenar es siempre $4M$ datos de 64 bits. En la gráfica de abajo se muestran los CSE cuando se ordenan $4PM$ datos de 64 bits. P es el número de procesadores. En las gráficas se muestran los CSE para 2 a 16 procesadores. Los CSE se muestran divididos en una misma columna y significan, de arriba a abajo, CSE de preparación de los mensajes (prepare), CSE acumulado de los Pasos (7) a (11) (adicional), CSE para el Paso (13) de comunicación de datos (All to All), CSE de la comunicación de los contadores del Paso (5) (Transpose), CSE del particionado local con Counting Split (Counting Split local), CSE para el broadcast del vector de separadores del Paso (4) y finalmente, CSE para la ordenación y la comunicación del muestreo, Pasos (2) y (1) del Algoritmo 9 respectivamente.

Para el caso de ordenar $4M$ datos, a medida que se aumenta el número de muestreos a enviar, q, los CSE para los pasos (1) y (2) van aumentando. Este incremento es más significativo cuando pasamos de 2048 muestreos por procesador a 4096.

De la experiencia sabemos también que, cuanto menos valores muestreados, con mayor probabilidad tendremos que volver a particionar algunas de las particiones frontera con tal de obtener un buen equilibrio de carga. Esto significa pagar el coste adicional que se muestra en la Figura. Por lo tanto, observando la evolución del coste adicional y el de los Pasos (1) y (2) se ha optado por un número de muestreos igual a 2048P. Otra variante del algoritmo presentado sería aumentar el número de muestreos con tal de reducir el coste de los pasos del algoritmo que contribuyen al coste adicional. Si aumentamos el número de muestreos podríamos extraer del vector de muestreo los vectores de separadores para cada partición frontera. Sin embargo, en este caso siempre estaríamos penalizando con CSE el paso de comunicación y ordenación del muestreo. Además, no podríamos asegurar que finalmente vayamos a particionar de forma equitativa las particiones frontera con ese muestreo extra.

Para el caso de ordenar $4PM$ datos (gráfica de abajo de la Figura C.5), se puede observar que el coste de comunicación y ordenación de los muestreos es poco significativo. Lo mismo sucede con el coste de comunicación de los datos.

En cualquier caso, ya sea ordenando conjuntos pequeños (gráfica de arriba) o grandes de datos (gráfica de abajo), cuando pasamos de 4 procesadores, el ancho de banda de la comunicación se reduce, haciendo que el coste de comunicación de los datos sea mucho mayor. Esto es lo mismo que sucede para el Reverse Sorting paralelo. Si hiciéramos que α fuera igual indistintamente del número de procesadores, el efecto sería el mismo que observamos en el Reverse Sorting Paralelo; no habría aumento del número de CSE al pasar de 4 a 8, o de 8 a 16 procesadores.

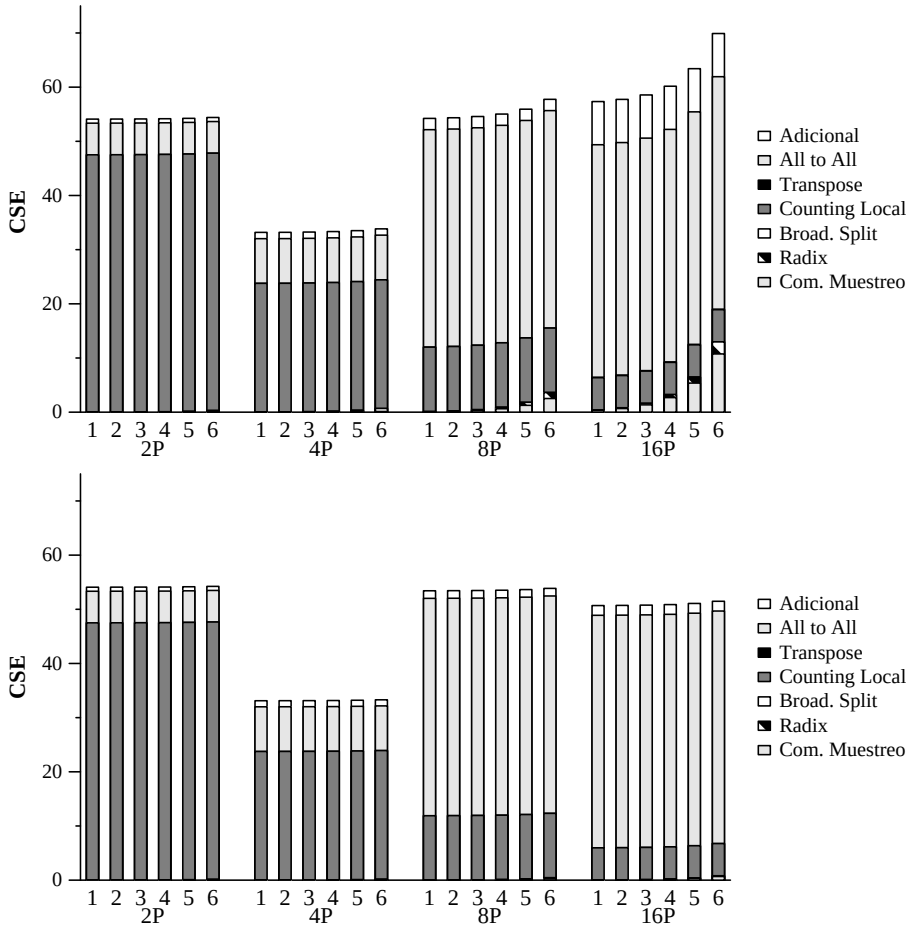

Figura C.5: CSE, según el modelo, del particionado con Counting Split variando q ($256P$ (1), $512P$(2), $1024P$(3), $2048P$(4), $4096P$(5) y $8192P$) y P (de 2 a 16) para 4M (arriba) y $4P$M (abajo) de datos a ordenar.