2. Algoritmos y Estructuras de Datos para Memoria Secundaria (3 semanas = 6 charlas)

Jérémy Barbay

18 November 2010

Contents

1 Modelo de computación en memoria secundaria. Accesos secuenciales y aleatorios

1. Arquitectura de un computador: la memoria

   (a) Muchos niveles de memoria

   - Procesador
   - registros
   - Cache L1
   - Cache L2
   - Memory

- Cache
- Disco Duro magnetico / Memory cell
- Akamai cache
- Discos Duros en la red
- CD y DVDs tambien son "memoria"

(b) Diferencias

- velocidad
- precio de construccion
- relacion fisica entre volumen y velocidad
- volatil o no
- accesso arbitrario en tiempo constante o no.
- latencia vs débito

(c) Modelos formales

- RAM
- Jerarquia con dos niveles, paginas de tamaño B
- Jerarquia con k niveles, de paginas de tamaños $B_1, ..., B_k$
- "Cache oblivious"
- Otros... mas practicas, mas dificil a analizar.

2 Diccionarios en Memoria Externa

1. B-arbol

(a) $(2, 3)$ Arbol: un arbol de busqueda donde

- cada nodo tiene 1 o 2 llaves, que corresponde a 2 o 3 hijos, con la excepcion de la raiz;
- todas las hojas son al mismo nivel (arbol completamente balanceado)
- Propiedades:
  - altura de un $(2, 3)$ arbol?
  - tiempo de busqueda?
  - insercion en un $(2, 3)$ arbol?
  - delecion en un $(2, 3)$ arbol?

(b) $(d, 2d)$ Arbol un arbol donde

- cada nodo tiene de d a $2d$ hijos, con la excepcion de la raiz (significa $d-1$ a $2d-1$ llaves).
- todas las hojas son al mismo nivel (arbol completamente balanceado)
- Propiedades:
  - altura de un $(d, 2d)$ arbol?
  - tiempo de busqueda?
  - insercion en un $(d, 2d)$ arbol?
  - delecion en un $(d, 2d)$ arbol?

(c) B-Arbol, y variantes

- B-Arbol
  - http://www.youtube.com/watch?v=coRJrcIYbF4
  - http://en.wikipedia.org/wiki/B-tree
- B**arbol*
  - otros nodos que la raiz son llenos al menos hasta 2/3 (en vez de 1/2)
  - http://en.wikipedia.org/wiki/B*-tree
- B$^+$*arbol*
  - the leaf nodes of the tree are chained together in the form of a linked list.
  - http://www.youtube.com/watch?v=GsY3-K4-J-8feature=related

2. Van Emde Boas arbol (vEB) http://en.wikipedia.org/wiki/Van$_E$*mde*$_B$*oas*$_t$*ree*

(a) Historia:

- Originalemente (1977) un estructura de datos normal, que suporta todas las operaciones en $O(\lg \lg n)$, inventada por el equipo de Peter van Emde Boas.
- No considerado utiles en practica para "pequeños" arboles.
- Applicacion a "Cache-Oblivious" algoritmos y estructuras de datos
  - optimiza el cache sin conocer el tamaño B de sus paginas
  - => optimiza todos los niveles sin conocer $B_1, ..., B_k$
- otras applicaciones despues en calculo parallelo (?)

(b) Definicion

- Cada nodo contiene un arbol van EmdeBoas sobre $\sqrt{n}$ elementos
- $\lg \lg n$ niveles de arboles
- operadores:
  - Findnext
  - Insert
  - Delete

(c) Analisis

- Busqueda en "tiempo" $O(\lg n / \lg B) acualquierniveli, dondeeltiempoeslacuantidaddeacc$

- Delecion

3. Finger Search Tree: la busqueda doblada de los arboles de busqueda

   (a) Estructura de datos

   (b) algorimo de busqueda

   (c) analisis: busqueda en $O(\lg p)$

3 Colas de prioridad en memoria secundaria. Cotas inferiores.

1. Colas de Prioridades tradicional:

   - que se necesita?
     - Operadores
       - *insert(key,item)*
       - *findMind()*
       - *extractMin()*
       - Operadores opcionales
         - *heapify*
         - *increaseKey, decreaseKey*
         - *find*
         - *delete*
         - *successor/predecessor*
         - *merge*
         - ...

- diccionarios: demasiado espacio para que se pide
  - menos operadores que diccionarios
    - => mas flexibilidad en la representacion
    - => mejor tiempo y/o espacio
- **binary heap**: una estructura a dentro de muchas otras:
  - sequence-heaps
  - binomial queues
  - Fibonacci heaps
  - leftist heaps
  - min-max heaps
  - pairing heaps
  - skew heaps
  - *van Emde Boas queues*
- **van Emde Boas queues** http://www.itl.nist.gov/div897/sqg/dads/HTML/vanemdeboas.ht
  - Definicion:
    "An efficient implementation of priority queues where insert, delete, get minimum, get maximum, etc. take O(log log N) time, where N is the total possible number of keys. Depending on the circumstance, the implementation is null (if the queue is empty), an integer (if the queue has one integer), a bit vector of size N (if N is small), or a special data structure: an array of priority queues, called the bottom queues, and one more priority queue of array indexes of the bottom queues."
    - rendimiento en memoria secundaria de "binary heap": muy malo?

2. Colas de Prioridades en Memoria Secundaria: diseno

- Reference:
  - http://www.dcc.uchile.cl/ gnavarro/algoritmos/tesisRapa.pdf
    - paginas 9 hasta 16
  - Otras referencias en http://www.leekillough.com/heaps/
- El equivalente de B-Arbol
- Muchas alternativas en practica

(a) Buffer trees

(b) M/B-ary heaps
(c) array heaps
(d) R-Heaps
(e) Array Heaps
(f) sequence heaps
(g) Mas en "An experimental study of priority queues in external memory" http://portal.acm.org/citation.cfm?id=351827.384259
(h) Colas de Prioridades en Memoria Secundaria: cota inferior?
- Cota inferior para dictionaries es una cota inferior por

- No. La reduccion es en la otra direccion.
- Cual es la cota inferior mas simple que se puede imaginar?
  - $\Omega(n/B)$

3. *REFERENCES*

- http://en.wikipedia.org/wiki/Priority_queue
- http://en.wikipedia.org/wiki/Heap_data_structure
- "An experimental Study of Priority Queues in External Memories" by Brengel, Crauser, Ferragina and Meyer

4 Ordenamiento en memoria secundaria: Mergesort. Cota inferior.

1. Un modelo mas fino

(a) Cuantos paginas quedan en memoria local?
- no tan importante para busqueda
- muy importante para applicaciones de computacion con mucho datos.

(b) Nuevas notaciones
- B = Tamano pagina
- N = cantidad de elementos en total
- n = cantidad de paginas con elementos = N/B
- M = cantidad de memoria local
- m = cantidad de paginas locales = M/B

- mnemotechnique:
  - N, M, B en cantidad de palabras maquinas (=bytes?)
  - n, m en cantidad de paginas
  - $n << N$, $m << M$

(c) En estas notaciones, usando resultos previos:

- Insertion Sort (en un B-Arbol)
  - usa dictionarios en memoria externa
  - $N \lg N / \lg B = N \log_B N$
  - Heap Sort
    - usa colas de prioridades en memoria externa
    - $N \lg N / \lg B = N \log_B N$
    - Eso es optimo o no?

2. Cotas Inferiores en Memoria Secundaria

- para buscar en un diccionario?
  - en modelo RAM? (de comparaciones)
    - $\lg N$
  - en modele Memoria Externa? (de comparaciones)
    - $\lg N / \lg B = \log_B N$ (ajustado)
    - para fusionar dos arreglos ordenados?
  - en modelo RAM?
    - N
  - en modelo Memoria Externa con paginas de tamaño B?
    - $N/B = n$ (ajustado)
    - para fusionar k arreglos ordenados?
  - en modelo RAM?
    - N
  - en modelo de Memoria Externa con M paginas de tamaño B?
    - $N/B = n$ (si $M > kB$)
    - para Ordenar
- en modelo RAM de comparaciones
  - $n \lg n$
- en modelo Memoria Externa con n/B paginas de tamaño B

- $\Omega(N/B \frac{\lg(N/B)}{\lg(M/B)})$
- que se puede notar mas simplamente $\Omega(n \lg_m n)$

- Prueba:
  - en vez de considerar el problema de ordenamiento, supponga que el arregla sea una permutacion y considera el problema (equivalente en ese caso) de identificar cual permutacion sea.
  - inicialemente, pueden ser n! permutaciones.
    - supponga que cada bloque de B elementos sea ya ordenado (impliqua un costo de al maximo n=N/B accessos a la memoria externa).
    - queda $N!/((B!)^n)$ permutaciones posibles.
    - para cada accesso a una pagina de memoria externo,
  - con M entradas en memoria primaria
  - B nuevas entradas se pueden quedar de $\binom{M}{B} = \frac{M!}{B!(M-B)!}$ maneras distintas
  - calcular la union de los $M+B$ elementos reduce la cuantidad de permtuaciones por un factor de $1/\binom{M}{B}$
  - despues de t accessos (distintos) a la memoria externa, se reduci la cuantidad de permutaciones a $N!/((B!)^n \binom{M}{B}^t)$
  - cuanto accessos a la memoria sean necesarios para que queda al maximo una permutacion?
    - $N!/((B!)^n \binom{M}{B}^t)$ debe ser al maximo uno.
    - usamos las formulas siguientes:
      - $\log(x!) \approx x \log x$
      - $\log \binom{M}{B} \approx B \lg \frac{M}{B}$
  - BONUS: Para ordenar strings, un caso particular (donde la

- $\Omega(N_1/B \log_{M/B}(N_1/B) + K_2 \lg_{M/B} K_2 + N/B)$
- donde
  - N_1 es la suma de los tamanos de las caldenas mas cortas que B
  - K_2 es la cuantidad de caldenas mas largas que B

3. Ordenar en Memoria Externa N elementos (en $n = N/B$ paginas) http://en.wikipedia.org/wiki/External_$sorting$

- Usando dictionarios o colas de prioridades en memoria externa

- $N \lg N / \lg B = N \log_B N$
- No es "ajustado" con la cota inferior
- impliqua
  - o que hay un mejor algoritmo
  - o que hay una mejor cota inferior
  - Queda un algoritmo de ordenamiento: MergeSort
- usa la fusion de $m-1$ arreglos ordenados en memoria externa:
  (a) carga en memoria principal $m-1$ paginas, cada una la primera de su arreglo.
  (b) calcula la union de estas paginas en la pagina m de memoria principal,
    - botando la pagina cuando llena
    - cargando una nueva pagina (del mismo arreglo) cuando vacilla

  (c) La complejidad es n accessos.
- Algoritmo:

(a) ordena cada de las n paginas $\rightarrow n accessos Cada nodo calcula la union de m arreglos y escribe$
(b) Analisis:
  - Cada nivel de recurencia costa n accessos
  - Cada nivel reduce por $m-1$ la cantidad de arreglos
  - la complejidad total es de orden $n \log_m n$ accessos. (ajustado)

4. **BONUS** cota inferior para una cola de prioridad?

   - una cola de prioridad se puede usar para ordenar (con N accessos)
   - hay una cota inferior para ordernar de $n \log_m n$
   - entonces????

5 Resultados de Aprendisajes de la Unidad Dos

- Comprender el modelo de costo de memoria secundario
- Conocer algoritmos y estructuras de datos básicos que son eficientes en memoria secundaria,
- y el analisis de su desempeño.