



Datos Masivos I

5. Algoritmos de memoria externa



5. Algoritmos de memoria externa

5.1 Modelo de memoria externa

5.2 Modelo de caché inconsciente

5.3 Cotas fundamentales de operaciones de entrada y salida

5.4 Escaneo

5.5 Ordenamiento

5.6 Búsqueda

5.7 Estructuras de datos estáticos y dinámicos

Aprendizaje de la semana

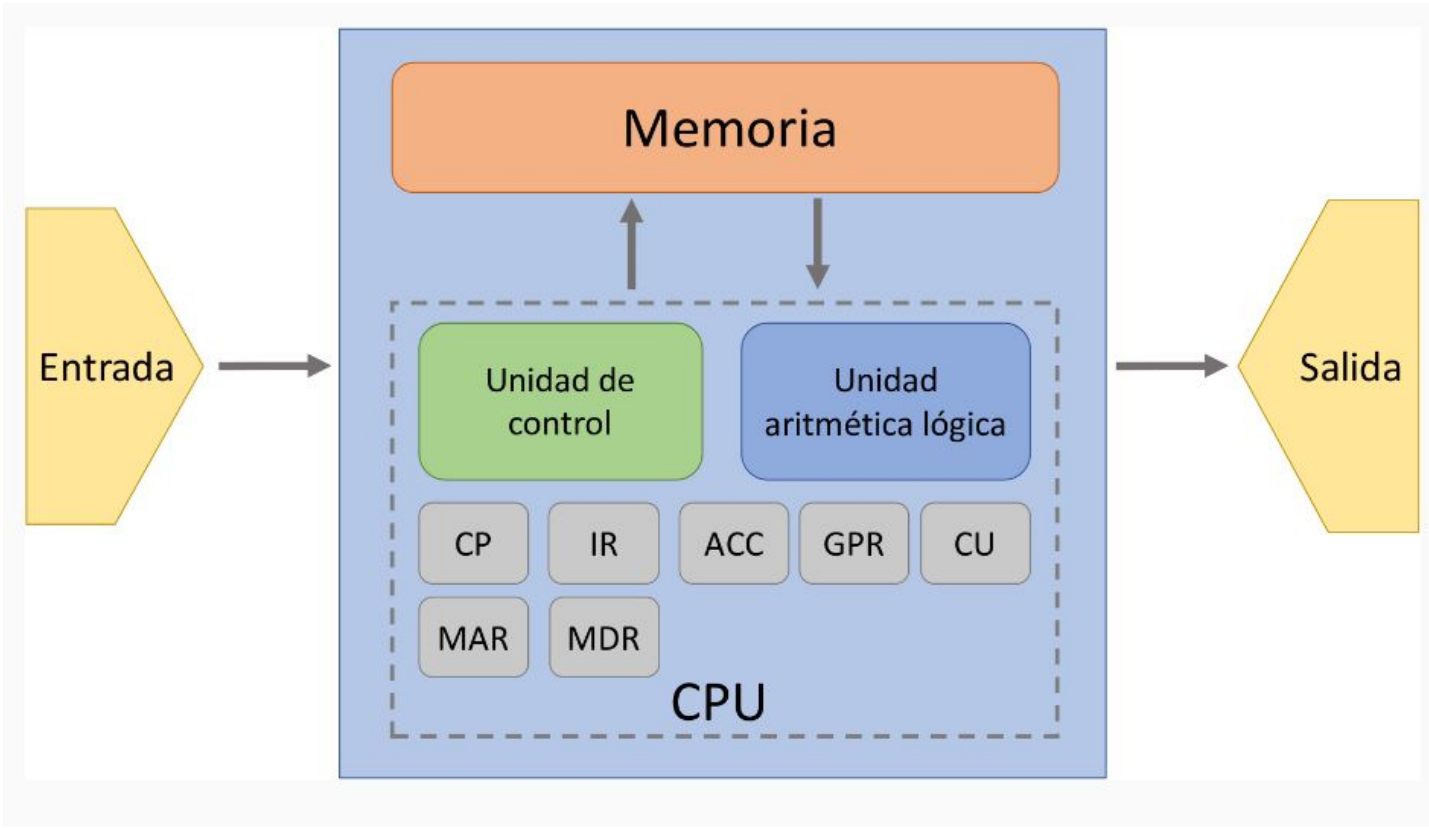
No regalen su trabajo!!!

Modelo de memoria externa

5.1 Modelo de memoria externa

- **Orientado a acceso:** se controlan de forma explícita las solicitudes de transferencia, incluyendo los tipos.
- **Orientado a arreglos:** se accede a través de los tipos reconocidos por el compilador y de las operaciones sobre esos tipos. Se utilizan principalmente para computación científica que ocupa regularmente arreglos.
- **Orientado a marcos de trabajo:** Los programas acceden constantemente a datos de discos y van produciendo resultados.

Arquitectura de Von Neumann



Jerarquía de Memoria

Se refiere a la organización piramidal de la memoria de niveles que tiene los ordenadores. Su objetivo es conseguir el rendimiento de una memoria de gran velocidad basándose en el **principio de cercanía de referencias**.

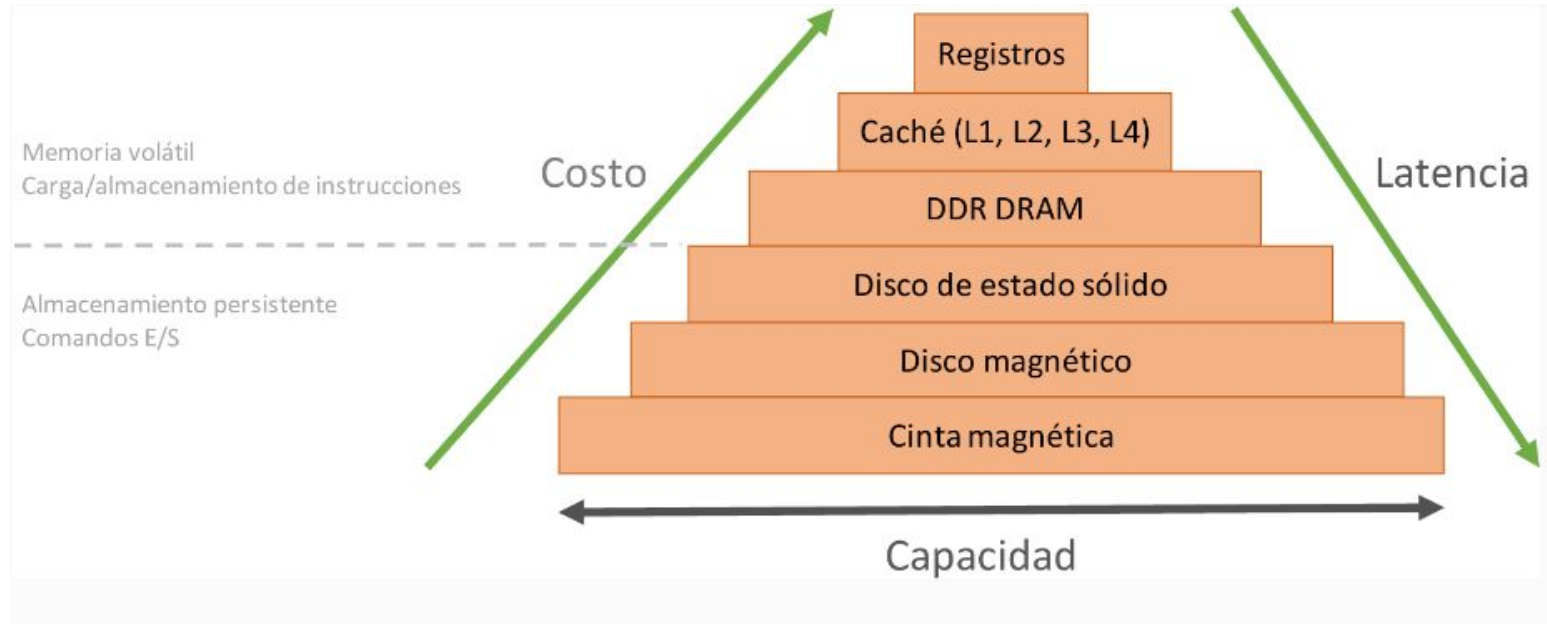
Los puntos básicos relacionados con la memoria pueden resumirse en:

- Capacidad - cuanta más memoria disponible, más puede utilizarse.
- Velocidad - velocidad óptima de los tiempos de espera entre cálculo y cálculo para extraer o guardar resultados.
- Costo por bit - El coste de la memoria no debe ser excesivo.

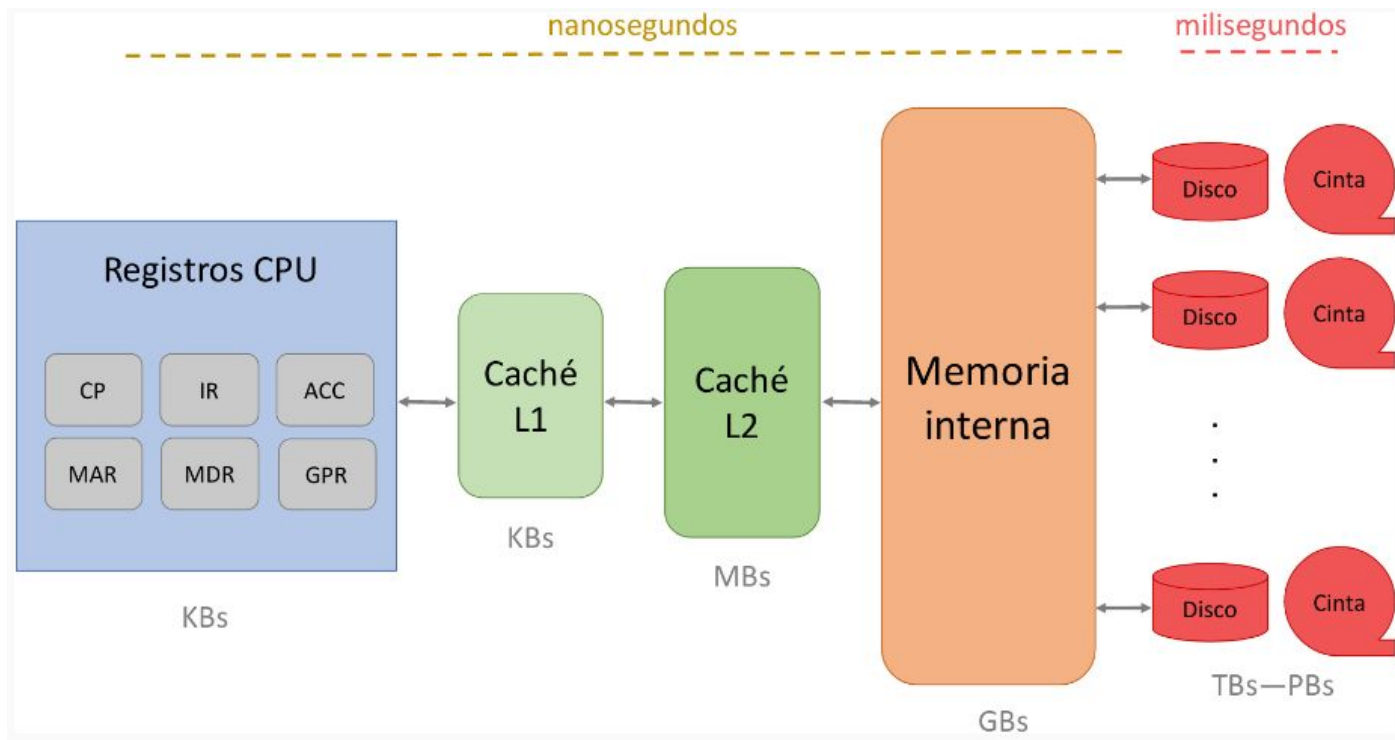
Jerarquía de Memoria

- A menor tiempo de acceso, mayor coste.
- A mayor capacidad, menor costo por bit.
- A mayor capacidad, menor velocidad.

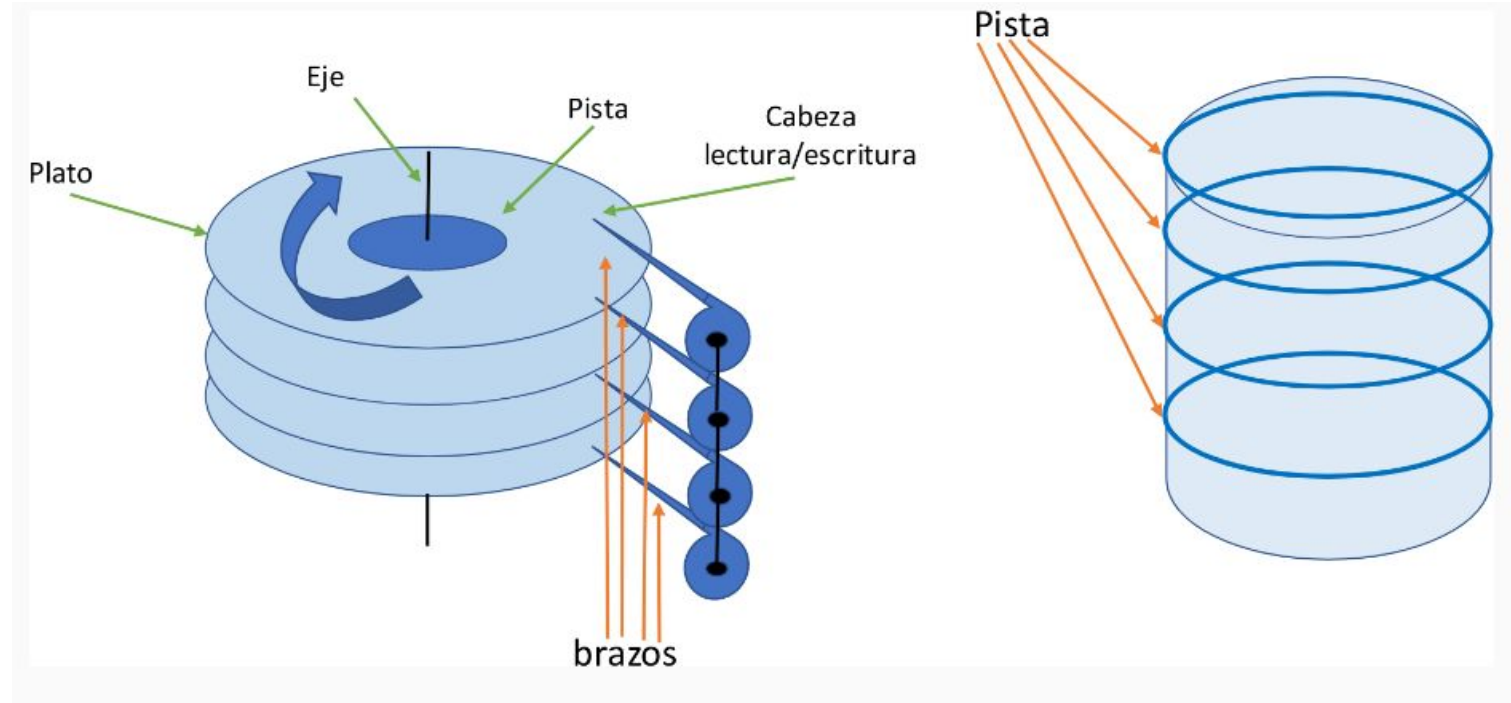
Jerarquía de Memoria



Sistemas Multidisco



Componentes de un disco magnético



Organización de Almacenamiento en Discos Magnéticos

- A Pista
- B Sector circular
- C Pista de un sector
- D Bloque

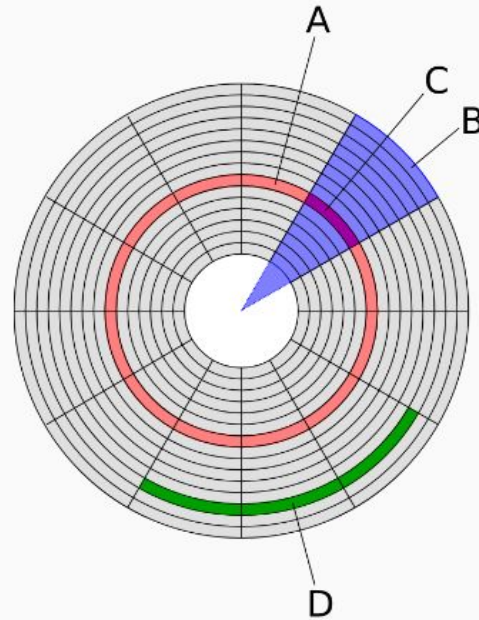


Imagen tomada de Wikipedia (*Data Cluster*)

Acceso a disco

Tiempo de operaciones de lectura y escritura (E / S) no es constante.

- Tiempo de búsqueda: 0 - 10 ms
- Tiempo de rotación: 0 - 3 ms
- Tiempo de transferencia: 0.2 ms por 8 kb

El acceso a disco en varios órdenes de magnitud es más lento que el acceso a memoria interna.

Paginación y precarga

Estrategias de sistemas operativos para minimizar el cuello de botella de operaciones E / S.

Precarga instrucciones o datos del disco a la memoria RAM para que estén disponibles cuando sean accedidos.

Se almacena y lee en tamaños fijos de bloques (llamados páginas).

Transferencia de bloques

- Acceso secuencial a disco es más rápido (8 - 156 kb) que acceso aleatorio.
- Operaciones E / S en disco son más eficientes si se realizan en fragmentos grandes bloques contiguos.
- Bloques completos se transfieren en fragmento grandes bloques contiguos.
- Cada transferencia por bloque es 1 operación E / S.

Modelo de discos Paralelos (PDM)

Explota dos mecanismos en sistemas multidiscos.

1. Localidad de referencia, la cual aprovecha la transferencia por bloques.
2. Acceso paralela a múltiples discos.

Presuposiciones:

- En cada operación E / S cada una de los D discos puede transferir un bloque de B elementos contiguos.
- Transferencia síncrona de los D bloques (toman el mismo tiempo).

Parámetros del modelo de discos paralelos

- **P** — procesadores comparten D discos.
- **N** — número de elementos (del mismo tamaño) del problema.
- **M** — número de elementos que se pueden almacenar en la memoria RAM.
- **B** — número de elementos por bloque de disco.
- $M < N$ y $1 \leq B \leq M/2$

Parámetros del modelo de discos paralelos

Para las consultas, se definen dos parámetros adicionales.

Q : número de consultas

Z : número de elementos en la respuesta

Parámetros del modelo de datos paralelos expresados en términos de bloques de

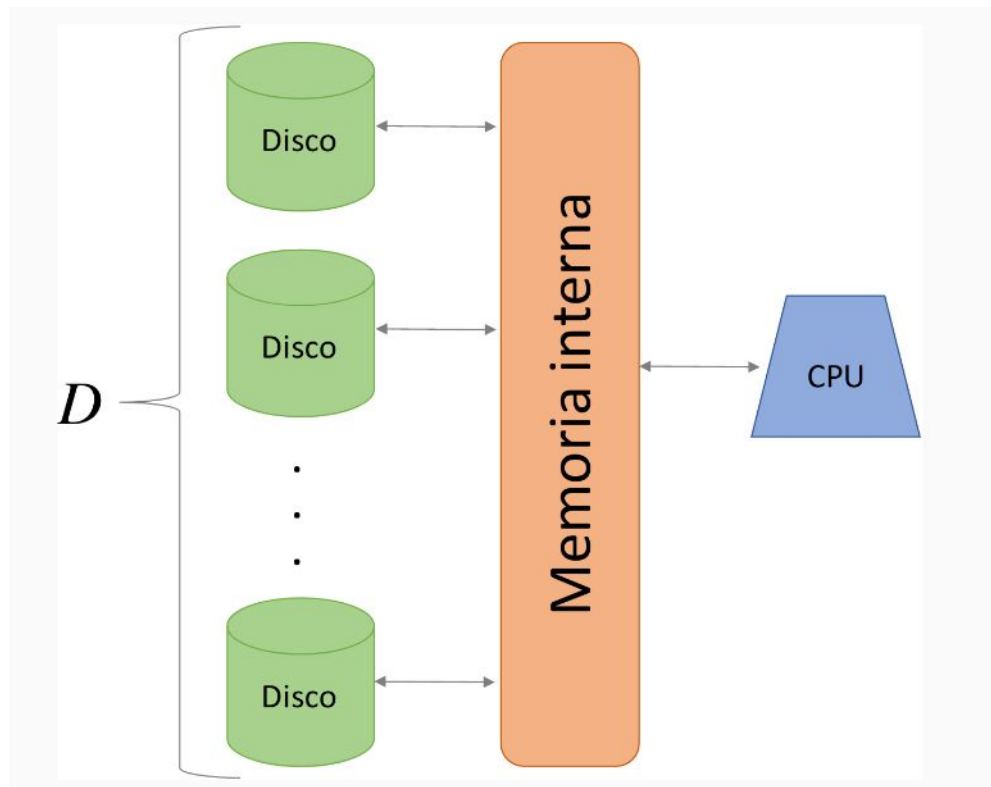
$$n = \frac{N}{B} \quad m = \frac{M}{B} \quad q = \frac{Q}{B} \quad z = \frac{Z}{B}$$

Parámetros del modelo de discos paralelos

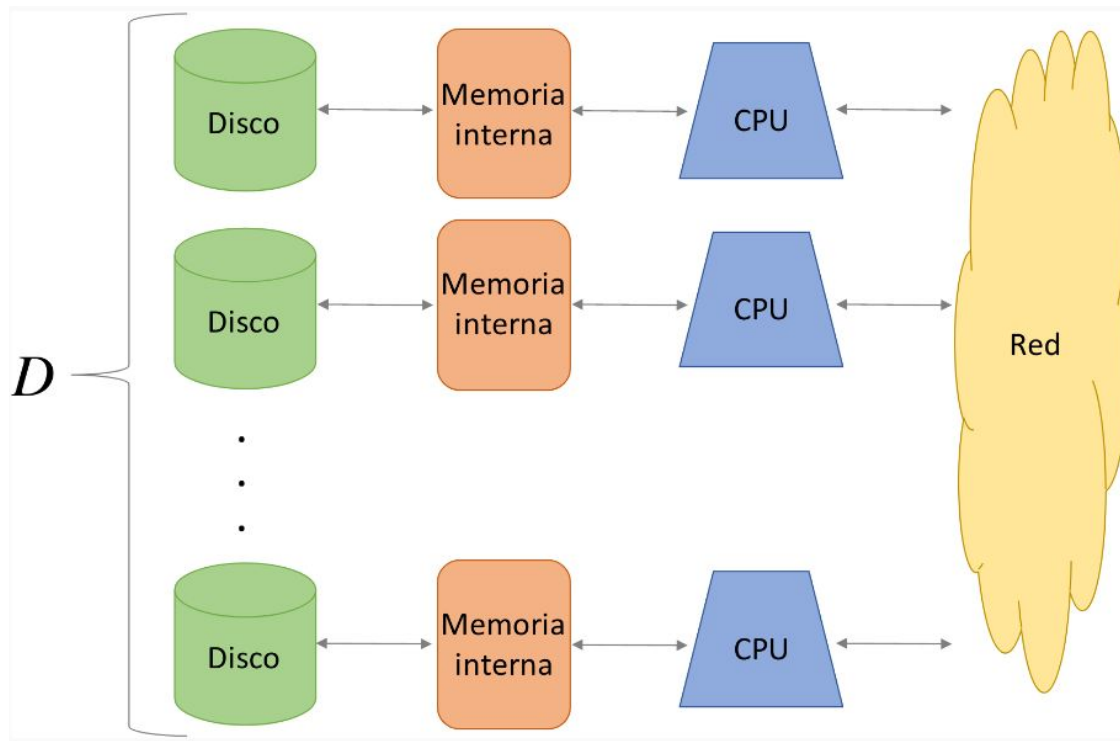
Considera los datos alineados

	D_0		D_1		D_2		D_3		D_4	
Línea 0	0	1	2	3	4	5	6	7	8	9
Línea 1	10	11	12	13	14	15	16	17	18	19
Línea 2	20	21	22	23	24	25	26	27	28	29
Línea 3	30	31	32	33	34	35	36	37	38	39

Modelo de Memoria Paralela



Modelo de Memoria Paralela



Complejidad E / S

Una operación E / S involucra leer o escribir un bloque de B elementos contiguos de o al disco.

La complejidad E / S para un algoritmo es el número de bloques que transfiere entre memoria y disco (las operaciones internas no se consideran en el modelo).

Tamaño del bloque al menos 512 bytes (dependiendo del hardware), aunque visualmente se usan al menos de 8kb.

Diseño de algoritmos de memoria externa

Métricas de rendimiento

- Complejidad E / S
- Máximo espacio en disco (número de bloques) activo en un tiempo dado.
- Tiempo de procesamiento interno.

Objetivos

- Tiempo comparable a algoritmos en memoria interna.
- Acceso a bloques con tantos datos útiles como sea posible.
- Aprovechamiento máximo de datos que se encuentran en memoria interna.

Operaciones Fundamentales

Complejidad E / S de muchos algoritmos depende de los siguientes operaciones.

- Escaneo: Lectura secuencial de N elementos.
- Ordenamiento: Acomodar N elementos en un orden dado.
- Búsqueda: encontrar elementos específicos en N elementos ordenados.
- Salida: regresar los Z elementos de consulta.

En el momento de discos paralelos, un algoritmo de memoria interna normalmente realiza $\Omega(N)$ operaciones E / S.



Cotas fundamentales de operaciones de entrada y salida



Cotas fundamentales

Las cotas de complejidad nos ayudan a clasificar los algoritmos de acuerdo a su tiempo de ejecución.

\mathcal{O} → cota superior, se utiliza normalmente para acotar el peor caso de un algoritmo.

Ω → cota inferior, para acotar el caso de menor tiempo de ejecución de un algoritmo.

Θ → cota ajustada, para ajustar el tiempo de ejecución de un algoritmo.



Escaneo

Escaneo en Memoria Externa

Encontrar un elemento x en un conjunto de tamaño N .

1. Para $i = 1$, hasta n
 - 1.1. Leer bloque i a memoria.
 - 1.2. Si el bloque contiene el elemento x , regresa verdadero.
2. Regresa falso

La ejecución de la lectura de los bloques contribuye al tiempo de ejecución, lo demás no se toma en cuenta.

La complejidad está dada por $O(n) = O(N/B)$ operaciones de entrada salida

Ejercicio

Diferencias entre N y N/B

Para escanear un arreglo de N elementos en disco tendríamos una complejidad E/S.

- $O(N)$, si los elementos contiguos se encuentran en bloques distintos.
- $O(N/B)$, si los elementos contiguos se encuentran en un mismo bloque.

Considera un arreglo de 600,000,000 elementos (N) almacenados en bloques de 6,000 (B) y un tiempo de acceso a disco de 1 milisegundo.

- N operaciones E/S tomarían 600,000 segundos (~166 horas)
- N/B operaciones E/S tomarían 100 segundos.

Modelo de Caché Inconsciente

Caché

- **Almacena elementos** (ej. resultados anteriores a datos) que van a ser utilizados por el CPU para que su acceso sea más rápido.
- **Asociatividad limitada**

Cada bloque se puede almacenar en solo c de M partes.

Valores típicos de c son 1 y 2, aunque algunas caches pueden tener 4 u 8.

Localidad en acceso a memoria

El caché aprovecha la localidad en el acceso a memoria.

Temporal: elementos tienden a ser utilizados múltiples veces en tiempos cercanos.

Espacial: elementos requeridos tienden a estar localizados en partes cercanas.

La jerarquía completa de memoria se puede ver como distintos niveles de caché, cada uno transfiriendo datos a sus niveles adyacentes en bloques.

Acierto y Fallo de Caché

Se llama *acierto* de caché si un bloque se encuentra disponible en caché cuando es requerido.

El *fallo* en caché ocurre si un bloque no puede ser accedido cuando se necesita y se tenga que copiar de la memoria interna, provocando un tiempo de espera.

Estrategias de Reemplazo

Cuando ocurre un fallo y de la caché está llena, se reemplaza un bloque.

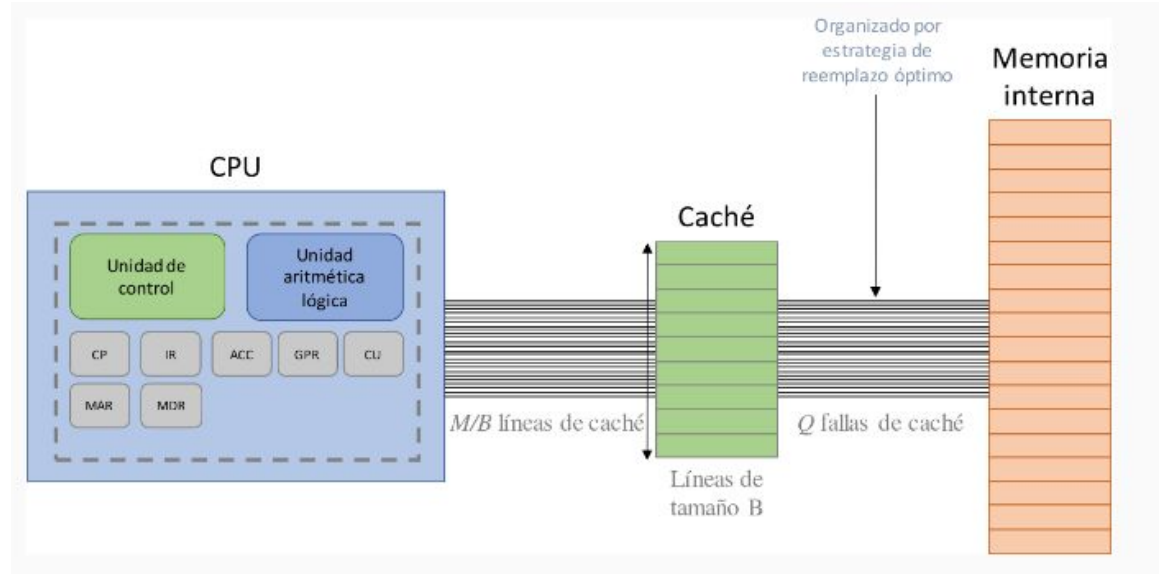
Estrategias para reemplazar un bloque

- El menos utilizado recientemente (LRU - Least Recently Used)
- El menos frecuente utilizado (LFU - Least Frequently Used)
- Primero en entrar, primero en salir (FIFO - First in First Out)

Modelo Inconsciente de Caché

Modelo de diseño de algoritmos para jerarquías de memoria multi nivel que no requiere conocimiento del tamaño del caché M ni del bloque de transferencia B .

Presupone el caché ideal.



Caché Ideal

- Reemplazo de página óptimo: bloque reemplazado es aquel que accederá al último en el futuro.
- Caché completamente asociativo de M elementos: cuando se carga un bloque de nivel $\ell + 1$, este puede reducir en cualquier segmento de nivel 2.
- Cada bloque tiene B elementos.
- El caché es más alto que ancho: el número de bloques M/B es más grande que el tamaño B de los bloques, esto es:

$$M = \Omega(B^2)$$

Algoritmos inconscientes de caché

Desempeño equivalente en cualquier par de niveles adyacentes de la jerarquía de memoria.

No gestionan de forma explícita las solicitudes de lectura y escritura.

Tratan de evitar el número de fallos de caché: cada bloque que se copia contiene tanta información útil como sea posible.

Algoritmos inconscientes de caché

Métricas de rendimientos

- Número de fallas en caché ($Q(N; M, B)$)
- Complejidad temporal

Estos algoritmos aprovechan el caché disponible.

Propiedad de inclusión

El modelo inconsciente de caché trabaja en dos niveles de memoria que tienen la propiedad de inclusión.

Los datos no pueden estar en el nivel ℓ a menos que también estén presentes en el nivel $\ell + 1$.

El tamaño del nivel ℓ en la jerarquía de memoria es estrictamente más pequeño que el nivel $\ell + 1$.

Justificación del modelo Inconsciente de Caché

1. Si un algoritmo tiene $Q(N; M, B)$ fallos (T transferencias) con un caché ideal, entonces sufre a lo mucho $2Q(N; M, B)$ fallos con reemplazo LRU o FIFO.
2. Un caché con reemplazo LRU o FIFO se puede mantener usando una memoria de tamaño $O(M)$ tal que cualquier acceso a un bloque tome un tiempo esperado de $O(1)$.

Condición de Regularidad

La condición de regularidad de una cota de complejidad de caché es

$$Q(N; M, B) = O(Q(N; 2M, B))$$

$$\text{Número de transferencias } T(M) = O(T(M/2))$$

Si un algoritmo con cota regular tiene $Q(N; M, B)$ fallos ($T(M)$ transferencias)

usando reemplazo óptimo, entonces tiene $\Theta(Q(N; M, B))$ fallos y

$\Theta(T(M))$ transferencias usando LRU o FIFO.

Escaneo en el modelo inconsciente de Caché

Problema: Recorrer todos los elementos de un arreglo.

Modelo de memoria externa: se almacenan los elementos en $[N / B]$ bloques de tamaño B y se recorren bloque por bloque.

Se requieren $[N / B]$ transferencias

Modelo inconsciente de caché: se almacenan los elementos en un segmento contiguo de memoria y se recorren uno por uno en el orden en el que fueron almacenados.

Se requieren $[N / B] + 1$ transferencias

Divide & Vencerás

El diseño de algoritmos inconscientes de caché hace amplio uso de divide y vencerás.

Varios algoritmos basados en esta estrategia se pueden considerar inconscientes de caché.

Al dividir el problema en subproblemas de forma recursiva, en algún nivel los subproblemas caben en caché (M) y en divisiones posteriores en B .



Ordenamiento para Memoria externa

Ordenamiento

Es el acomodar ya sea de forma descendente o ascendente los datos, ya sea de una lista o de un archivo.

Los algoritmos de ordenamiento se clasifican por:

- Lugar en donde se realiza la ordenación.
- Por el tiempo que tarda en realizar la ordenación.

Ordenamiento para memoria externa

- Lugar en donde se realiza la ordenación.
 - Algoritmos de ordenamiento interno.
 - Algoritmos de ordenamiento externo.
- Por el tiempo que tarda en realizar la ordenación.
 - Algoritmos de ordenación natural
 - Algoritmos de ordenación no natural.
 - Por estabilidad.

Cuando los elementos son indistinguibles, la estabilidad no interesa.

Algoritmos de ordenamiento interno

Usan la memoria primaria.

Su uso es con listas simples, los datos son de un solo tipo.

- Ordenamiento Burbuja
- Ordenamiento Shell
- Ordenamiento Quick Sort
- Ordenamiento Radix

Algoritmos de ordenamiento externo

Para su uso utiliza memoria secundaria.

Se usan cuando se tienen registros y/o archivos, ya que su uso es para el manejo de datos ya sea del mismo tipo y estos pueden quedar almacenados ya ordenados.

Se encuentran almacenados en un dispositivo externo.

Utilizan la técnica de divide y vencerás.

Algoritmos de ordenamiento externo

Intercalación/ Mezcla directa/ Merge

Este método se llama mezcla porque combina dos o más secuencias en una sola secuencia ordenada por medio de la selección repetida de los componentes accesibles en el momento.

Puede usarse un arreglo individual en lugar de dos secuencias, se considera como de doble extremo. En este caso se tomarán elementos de los dos extremos del arreglo para hacer la mezcla. El destino de los elementos combinados se cambia después de que cada par ha sido ordenado para llenar uniformemente las dos secuencias que son el destino.

Después de cada pasada de los extremos del arreglo, intercambian de papel, la fuente se convierte en el nuevo destino y viceversa.

Intercalación/ Mezcla directa/ Merge

El más utilizado.

Realización sucesiva de una partición y una fusión que produce secuencias ordenadas de longitud cada vez mayor.

En la primera pasada, la partición es de longitud 1 y la fusión o mezcla produce secuencias ordenadas de longitud 2.

En la segunda pasada, la partición es de longitud 2 y la mezcla produce secuencias ordenadas de longitud 4.

Intercalación/ Mezcla directa/ Merge

1. Inicio.
2. Dividir la secuencia F en dos mitades denominadas F1 y F2.
3. Mezclar F1 y F2 combinando cada elemento en pares ordenados.
4. Llamar F a la secuencia mezclada, repetir 1 y 2, combinando los pares en cuadriples ordenados.
5. Repetir los pasos anteriores duplicando cada vez la longitud de las secuencias combinadas hasta que quede ordenada la secuencia original.
6. Fin del algoritmo.

Se repite hasta que la longitud de la secuencia para la partición sea $\left(\frac{n+1}{2}\right)$ entero.

Ejemplo

Supongamos que se desea ordenar las claves del archivo F. Para realizar tal actividad se utilizan los archivos auxiliares a los que se denominará F1 y F2.

F: 09 75 14 68 29 17 31 25 04 05 13 18 72 46 61

Mezcla Natural (Mezcla equilibrada)

Optimización del método de mezcla directa.

Consiste en realizar las particiones tomando secuencias ordenadas de máxima longitud en lugar de secuencias de tamaño fijo previamente determinadas.

Luego se realiza la fusión de las secuencias ordenadas, en alternada sobre dos archivos.

Mezcla Natural (Mezcla equilibrada)

Se necesitan 4 archivos, el archivo original (A) y tres archivos auxiliares (A, B, C), de estos, 2 serán considerados entradas y 2 de salida, de forma alternada para realizar la fusión - partición.

El proceso termina cuando en una fusión - partición el segundo archivo quede vacío.

Se basa en la combinación de subsecuencias ordenadas. Las subsecuencias ordenadas de la cinta fuente, se distribuyen en dos cintas destino auxiliares A y B.

Seguidamente, se mezcla una subsecuencia ordenada de cada cinta auxiliar.

Ejemplo

Ordenar C.

C: 9 10 3 4 5 25 26 13 14 1 2



Estructuras de datos estáticos y dinámicos



Datos Estáticos



El tamaño ocupado en memoria se define antes de que el programa se ejecute y no puede modificarse dicho tamaño durante la ejecución del programa.

Ocupan solo una casilla de memoria, por lo tanto una variable simple hace referencia a un único valor a la vez.

Datos Dinámicos

El tamaño ocupado en memoria se puede modificar durante la ejecución del programa.

Las variables que se crean y están disponibles durante la ejecución del programa se llaman variables continuas.



Búsqueda para Memoria externa

Búsqueda de elementos

Dado un arreglo de N elementos, determinar si un elemento de consulta q está en el arreglo.

Recorriendo todos los elementos tomaría $O(N)$ operaciones E / S.

Si están ordenadas podríamos usar búsqueda binaria y tomaría $O(\log_2 \frac{N}{B})$



Algoritmos de búsqueda para memoria externa



Algoritmos de búsqueda para memoria externa

Los métodos de búsqueda se pueden clasificar en internos y externos, según la ubicación de los datos sobre los cuales se realizará la búsqueda.

- Búsqueda interna.
 - Los elementos se encuentran en la memoria principal.
- Búsqueda externa.
 - Si los elementos están en memorias secundarias.

Búsqueda Interna

Los más usados son:

- Secuencial o lineal
- Binaria
- Por transformaciones de claves
- Árboles de búsqueda

Búsqueda Externa

- Datos en archivos.
- Todo componente o registro de un archivo tiene generalmente un campo que lo identifica llamado campo clave.
- Ocupa la misma posición relativa en todos los registros de un mismo archivo.

Búsqueda Secuencial

El método consiste en recorrer el archivo comparando la clave buscada con la clave del registro en curso.

Utiliza la búsqueda lineal.

Búsqueda Secuencial mediante bloques

Toma bloques de registros de tamaño arbitrario y depende del número de elementos del archivo.

\sqrt{N}

Generalmente

El archivo debe estar ordenado.

Búsqueda Secuencial mediante bloques

La búsqueda se realiza al comparar la clave en cuestión con el último registro de cada bloque. Si la clave resulta menor, entonces se busca en forma secuencial a través de los registros del bloque. En caso contrario se continúa con el siguiente bloque.

Ejemplo

Considera las siguientes claves:

204, 311, 409, 415, 439, 450, 502, 507, 600, 623,
679, 680, 691, 692, 695, 698, 730, 850, 870, 889

$K = 623$

Búsqueda binaria externa

Funciona de la misma forma que la búsqueda binaria convencional, el archivo debe estar ordenado y se debe conocer N.

Requiere accesos a diferentes posiciones del dispositivo en el que está almacenado el archivo.



Alto costo en tiempo de acceso

Búsqueda por transformación de claves (Hash)

Los archivos se encuentran organizados en cubetas, las cuales se encuentran formadas por cero, uno o más bloques de registros.

La función hash aplicada a una clave, dará como resultado un valor que hace referencia a una cubeta en la cual se puede encontrar el valor buscado.

La elección de una adecuada función hash y de un método para resolver colisiones es fundamental para lograr mayor eficiencia en la búsqueda.

Disminuir colisiones

- Los bloques tienen un número fijo de registros.
- Las cubetas no tienen un límite en cuanto al número de bloques que pueden almacenar.

Lo anterior funciona sí y sólo si, el número de cubetas no crece considerablemente.

Si lo anterior ocurre, el número de bloques que se deben recorrer en la cubeta es grande → el tiempo será significativo.

Hashing dinámico por transformación de Claves

El dinamismo se utiliza para variar el número de cubetas en función de su densidad de ocupación.

- Por medio de expansiones totales.
- Por medio de expansiones parciales

Método de expansiones totales

El más utilizado

Consiste en duplicar el número de cubetas, en medida que estas superen su densidad de ocupación previamente establecida.

$$N \rightarrow 2N \rightarrow 4N \dots$$

También se da en sentido contrario.

A medida que la densidad de ocupación de las cubetas disminuye, se reduce el número de estas.

La densidad de ocupación se calcula como el cociente entre el número de registros ocupados & el número de registros disponibles.

Método de expansiones parciales

Consiste en incrementar el 50% el número de cubetas, haciendo que dos expansiones parciales equivalgan a una total.

$$N \rightarrow 1.5 N \rightarrow 2N \rightarrow 3N \dots$$

Hashing extendible

Trabaja con directorios.

Un directorio consiste de una tabla con 2^d referencias, donde $d \geq 0$

La ubicación en la tabla de un elemento está dada por los d bits menos significativos de su valor hash.

$$hash_d(x) = hash(x) \bmod 2^d$$

Hashing extendible

Cada ubicación de la tabla contiene una referencia a un bloque donde se almacenan los elementos.

Una ubicación comparte el mismo bloque con las ubicaciones con los mismos k bits menos significativos.

$$hash_k(x) = hash_d(x) \bmod 2^k$$

Hashing extendible

d es la profundidad global.

- se elige el número más pequeño con el cual cada ubicación de la tabla tiene a lo mucho B elementos.

k es la profundidad local.

- se elige el número más pequeño para que los elementos asignados entren en un solo bloque.

Hashing extendible

Las búsquedas requieren 2 operaciones E /S si el directorio reside en disco.

- Acceso a ubicación de la tabla
- Acceso al bloque asignado a la ubicación.

Hashing lineal

No necesita directorios.

Cada ubicación de la tabla corresponde a un bloque donde se almacenan elementos.

Requieren usualmente una solo operación E /S.

Árboles B

Son árboles de búsqueda (aplicado a ficheros) pero a diferencia del árbol binario, cada nodo puede poseer más de dos hijos.

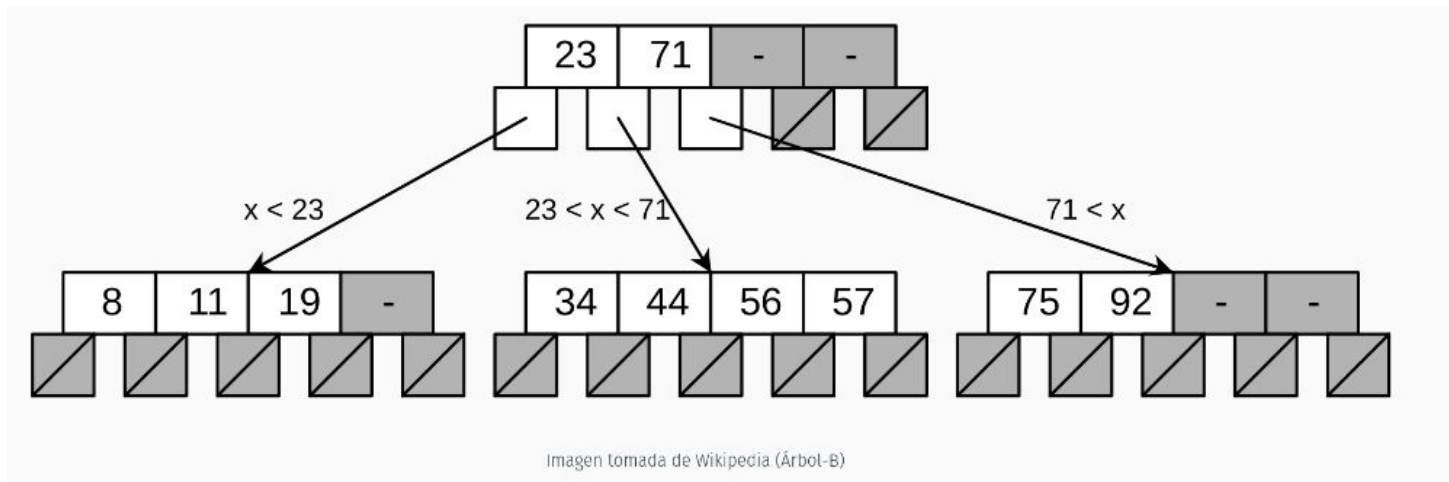
- Mantienen los datos ordenados
- No suelen ser balanceados

Son árboles de búsqueda de m ramas y cada nodo puede almacenar un máximo de $m-1$ claves.

Árboles B - Características

- El orden de un árbol-B es el número máximo de ramas que pueden partir de un nodo.
- Si de un nodo de un árbol-B parten n ramas, ese nodo contendrá $n-1$ claves.
- El árbol está ordenado.
- Todos los nodos terminales (hoja) están en el mismo nivel.
- Todos los nodos intermedios excepto la raíz deben tener entre $\frac{m}{2}$ y m ramas no nulas.
- El número máximo de claves por nodo es $m-1$.
- El número mínimo de claves por nodo es $\frac{m}{2}-1$
- La profundidad (h) es el número máximo de consultas para encontrar la clave.

Árboles B



Árboles B

Cuanto mayor sea el orden, menor será la profundidad.

Interesa que el número de accesos a disco sea lo más pequeño posible.

El orden se ajusta al tamaño del bloque más pequeño que se lee o se escribe en una operación de acceso al disco.

Las operaciones que se pueden realizar en un árbol-B:

- Insertar una clave.
- Eliminar una clave.
- Buscar una clave.

Árboles B - Inserción

1. Búsqueda del nodo hoja correspondiente en orden a la clave a insertar, x , la cual será insertada si no existe en el árbol.
2. Si el nodo hoja correspondiente no está lleno, es decir, el número de claves o entradas es menor que " $m-1$ ", se insertará en ese nodo respetando el orden de las claves.
3. Si el nodo hoja está lleno (número de claves igual a " $m-1$ ") se procede a la división celular:
 - a. Se crea un nuevo nodo repartiéndose el contenido del nodo lleno entre los dos nodos y la clave intermedia sube un nivel, es decir, se añade una nueva entrada al padre.
 - b. Si cabe en el padre, se inserta ahí.

Ejemplo

Insertar los siguientes items en un árbol-B inicialmente vacío de orden $m = 5$.

20, 40, 10, 30, 15, 35, 7, 26, 18, 22, 5, 42, 13, 46, 27, 8, 32, 38, 24, 45, 25

Árboles B - Eliminación

Eliminación de un nodo hoja.

1. Se busca el elemento a eliminar.
2. Si el valor se encuentra en un nodo hoja, se elimina.
3. Si se queda con pocos elementos se rebalancea.

Eliminación de un nodo hoja.

1. Encuentra un nuevo separador que sustituya al elemento.
2. Toma el elemento más grande del hijo izquierdo o del derecho, eliminándolo del subárbol correspondiente y reemplaza con este el elemento a eliminar.
3. Rebalancea el subárbol si es necesario.

Ejemplo

Del árbol anterior, eliminar los elementos

25, 45, 24, 38, 32, 8, 27, 46, 13, 42

Si el nodo tiene dos hijos, sustituir por el mayor de la izquierda.

Árboles B - Búsqueda

1. Se recorre el árbol de la raíz a las hojas.
2. Si el elemento está almacenado en el nodo de la búsqueda, termina, sino se continúa con el subárbol correspondiente al valor a buscar.
3. Si estamos en un nodo hoja y el elemento no está almacenado, entonces no se encontró.

Ejercicios

1. Insertar los siguientes elementos en un árbol-B inicialmente vacío de orden $m = 6$.

1, 9, 32, 3, 53, 43, 44, 57, 67, 7, 45, 34, 23, 12, 23, 56, 73, 65, 49, 85, 89,
64, 54, 75, 77

2. Eliminar los elementos

9, 32, 12, 49, 85, 53

3. Programar el método de búsqueda secuencial mediante bloques.