Lecture 9: Online Data

Fan Xin

1 FIF is Always Better

Proof:

Let the Cache have $a_1, a_2, \ldots\ldots, a_k$ and a new page a_0 comes in. (suppose the incoming pages are $a_0, \ldots, a_{k-1}, \ldots, a_k$)

Let a_k is removed by FIF, but algorithm A removes a_{k-1} (it does not have to be a_{k-1}, any a_i rather than a_k is ok, just suppose it is a_{k-1}) to take in a_0.

We establish an Algorithm B. In the first step, B removes a_k to take in a_0 as FIF does.

After that, algorithm B

- Operate the same way as A does if neither a_{k-1} nor a_k is involved.
- If A removes a_k to take in some page a_i, B removes a_{k-1} to take in a_i. Both are in the same state and we apply inductive hypothesis.
- If A takes in a_{k-1} to remove a_j, B takes in a_k to remove a_j, and apply inductive hypothesis.

Goal: $\text{cost}(\text{FIF}) \leq \text{cost}(\text{B}) \leq \text{cost}(\text{A})$

Before a_k comes in, B has the same cost as A, but when a_k comes in, B has no page fault while A has one or zero page fault, therefore $\text{cost}(\text{B}) \leq \text{cost}(\text{A})$. (As a_{k-1} must come in before a_k, when A takes in a_{k-1}, B will take in a_k, so B must contain a_k in its cache before a_k comes in, and A may have removed a_k or not.)

Algorithm A has at least one page fault before a_k comes in (to take in a_{k-1}), because B has the same number of page faults as A, so B has at least one page fault as well. But for FIF, it has no page fault before a_k and one page fault when a_k comes in. Totally, B has at least one page fault and FIF has exactly one page fault, this means $\text{cost}(\text{FIF}) \leq \text{cost}(\text{B})$.

2 LRU with Augmented Resource

LRU for k cache against FIF for h cache has a competitive ratio of $\text{k}/(\text{k} - \text{h} + 1)$.

Break input into subsequences $\sigma_1, \sigma_2, \ldots, \sigma_b$ such that σ_i is the longest sequence where k pages appeared in the input.

- During each phase σ, there are k pages requested but only h slots for FIF.
- At least k - h + 1 faults (including the transition point) are incurred during this phase for FIF.
- As in the above discussion, k faults are incurred in one phase for LRU.

For example:

h=3, k=5

$\sigma_1 = a_1, a_2, a_3, a_1, a_2, a_4, a_5$, $\sigma_2 = a_6, a_5, a_4, a_3, a_2$

from $\emptyset \rightarrow \sigma_1$: 5 page faults, suppose page in cache: a_1, a_4, a_5

from $\sigma_1 \rightarrow \sigma_2$: at least k-h+1=3 page faults. (The first page and other k-h pages will cause page fault.)

$$\text{competitive ratio} = \lim_{b \to \infty} \frac{k + (b-1) * k}{k + (b-1) * (k - h + 1)} = \frac{k}{k - h + 1}$$

Corollary: Competitive Ratio is two if we choose h = k/2.

Extra hardware resource to resolve algorithm design difficulties.

3 System scheduling of unknown processing time jobs

Inputs of n jobs with unknown processing jobs

Objective: execute all jobs with the minimum total completion time.

Minimize $\sum_{i=1}^{n} C_i$, C_i is the completion time of job a_i

n jobs of length $\epsilon_1 < \epsilon_2 \ldots < \epsilon_{n-1} < 1$.

Worst total completion time: $\text{n} * 1 + (\text{n} - 1) * \epsilon_{n-1} + \cdots + \epsilon_1$.

Best total completion time (shortest job first): $\text{n} * \epsilon_1 + (\text{n} - 1) * \epsilon_2 + \cdots + 1$.

3.1 Shortest Job First (SJF)

SHF is the optimal schedule.

Prove by contradiction:

Suppose $a_1, a_2, \dots, a_i > a_{i+1}, a_{i+2}, \dots, a_n$ is the best schedule.

$$C_{i+1} = C_i + a_{i+1}$$
$$C_i = C_{i-1} + a_i$$

Switch $a_i \& a_{i+1}$

$$C_j{}' = C_j,\ \mathrm{j} = 1,2,\dots,\mathrm{i}-1,\mathrm{i}+1,\dots,\mathrm{n}$$
$$C_i{}' = C_{i-1} + a_{i+1} < C_i$$

then

$$\sum_{i=1}^{n} C_i{}' < \sum_{i=1}^{n} C_i$$

We have a better schedule, which is a contradiction.

3.2 Round robin

Execute all jobs together, each for ϵ time in turn.

We don't know execution time of jobs but shortest job finishes first.

Does a shortest job a_1 complete as fast as in SJF? No, it takes n times bigger.

SJF finishes a_1 in time a_1----------full information

RR finishes a_1 in time $\mathrm{n} * a_1$---------no information

Suppose all jobs are of the same execution time T_0

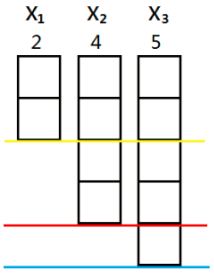
$$\mathrm{SJF}(\dots) = \frac{n(n+1)}{2} T_0$$
$$\mathrm{RR}(\dots) = \mathrm{n} * \mathrm{n} T_0 = n^2 T_0 < 2 * \mathrm{SJF}(\dots)$$

For jobs $\mathrm{x}_1 < \mathrm{x}_2 \dots < \mathrm{x}_n$, the total completion time is

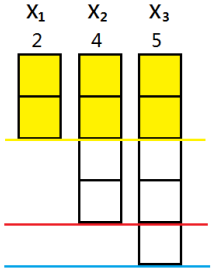
$$\mathrm{RR}(\mathrm{x}_1, \dots, \mathrm{x}_n) = \mathrm{n}\mathrm{x}_1 + \dots + [(\mathrm{n}-\mathrm{i})\mathrm{x}_i + \mathrm{x}_i + \mathrm{x}_{i-1} \dots + \mathrm{x}_1] \dots + [\mathrm{x}_n + \dots + \mathrm{x}_1].$$
$$\mathrm{OPT}(\mathrm{x}_1, \dots, \mathrm{x}_n) = \mathrm{n}\mathrm{x}_1 + (\mathrm{n}-1)\mathrm{x}_2 + \dots + \mathrm{x}_n.$$

For example:

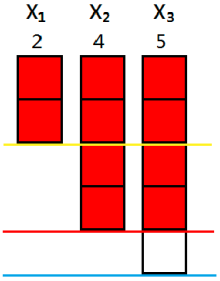
There are three jobs $\mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_3$ and their execution times are 2,4,5



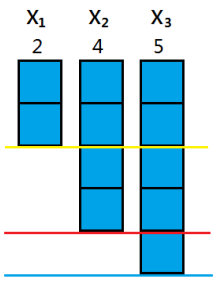
(a) Three jobs



(b) Completion time of x_1



(c) Completion time of x_2



(d) Completion time of x_3

For RR:

The completion times of x_1, x_2, x_3 are the sum of the blocks above the yellow, red, blue line respectively, as shown in the above figure.

The completion time of x_1: $x_1 * 3 = 6$

the completion time of x_2: $x_1 + 2 * x_2 = 10$

and the completion time of x_3 is $x_1 + x_2 + x_3 * 1 = 11$

the total completion time is 6+10+11=27

If use SJF, the total completion time is 2+(4+2)+(5+4+2)=19

$\mathrm{RR}(\vec{X}) \leq 2\mathrm{OPT}(\vec{X})$

Extra time x_i waited in RR is the same as the actual time x_j has to wait for x_i in OPT.