Lecture 11
Practical Examples with Specific Problems

- Memory Queue
- Priorities
- Disk Sub-System
- CPU Scheduling
- Paging

Problems

- Memory queue — Simultaneous resource possession
- Disk subsystem — Complex sub-model
- CPU scheduling — Priorities
- Paging — Dependence on other jobs

- General: non-product form

Non-Product Form Solutions

- Use your imagination and know-how
- Flow equivalence
- Load concealment
- Change model
- Multi-level modeling
- Simulation
- Hybrid simulation

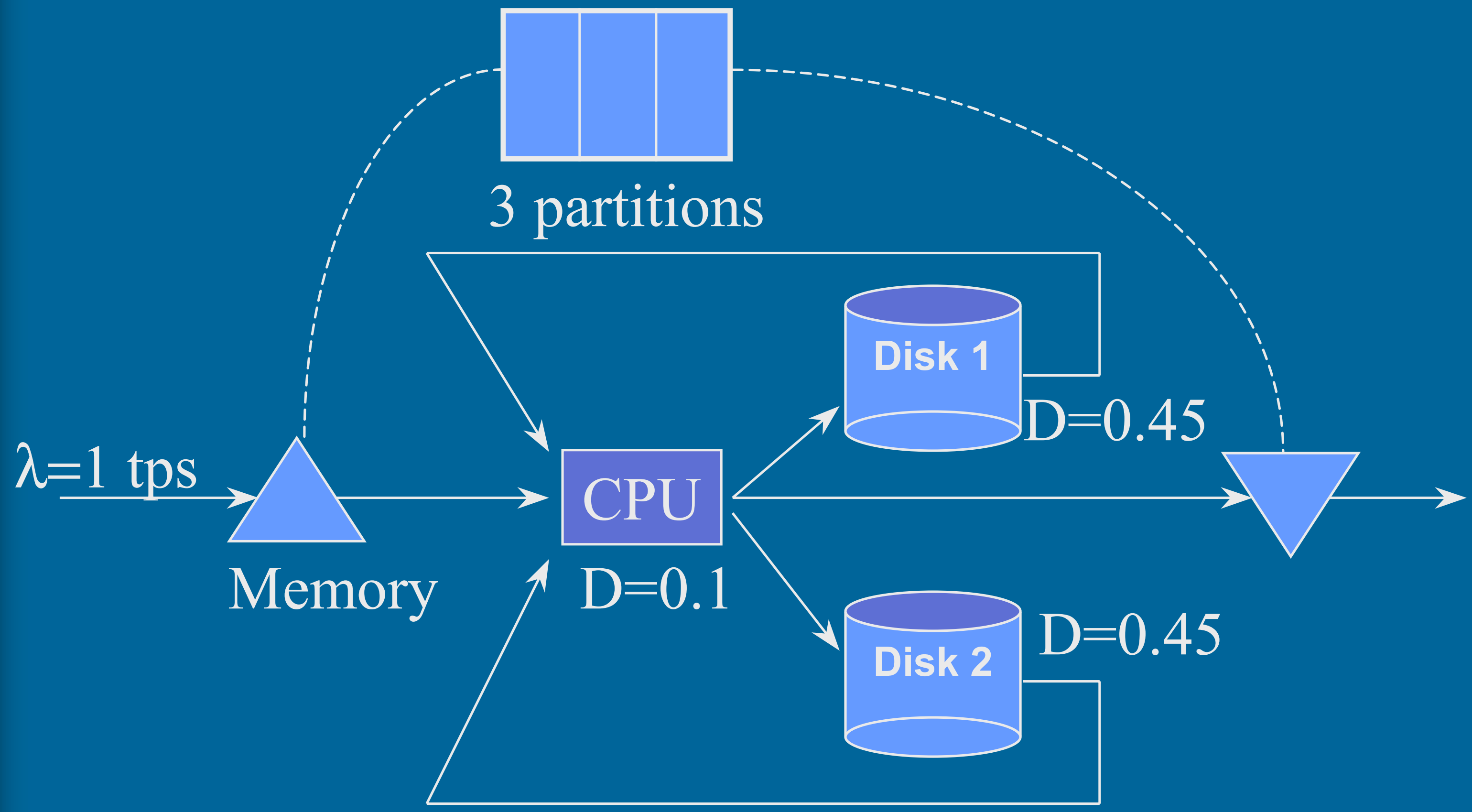
Flow Equivalent Server

- Fig. 8.2
- Approach OK, if sub-model is “busy” part of model
  - many state transitions within sub-model as compared to transitions between sub-model and rest of the original model
- Hierarchical models
  - orig model, sub-model, aggregate model

Modeling Memory

- Fig. 8.1
- Original model not product form
- Use FESC, max mpl 5
  - short cut sub-system to closed model
  - solve for all mpl={1,2,3,4,5}
  - create service times for FESC: $S^{FESC}(k) = 1/X^{SUBSYS}(k)$
  - solve new model (or models), Fig. 8.3
    - solve open class first, slow down FESC
    - solve closed class

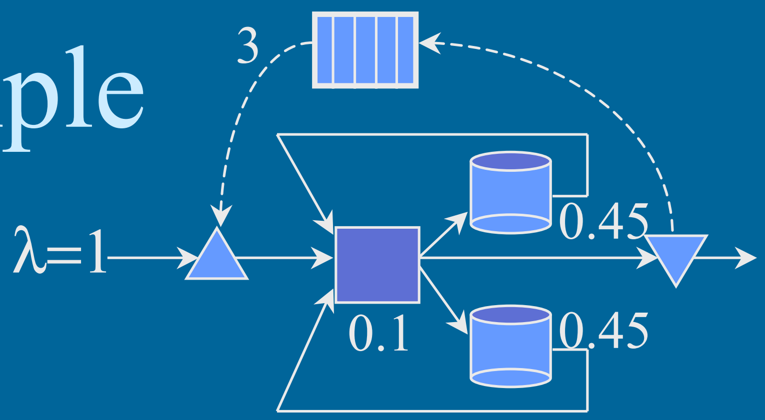
Memory Constraint, One Class



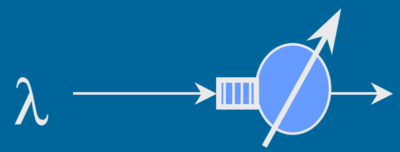
Not product form! Why? Simultaneous resource possession

source:[Men 94, p. 239]

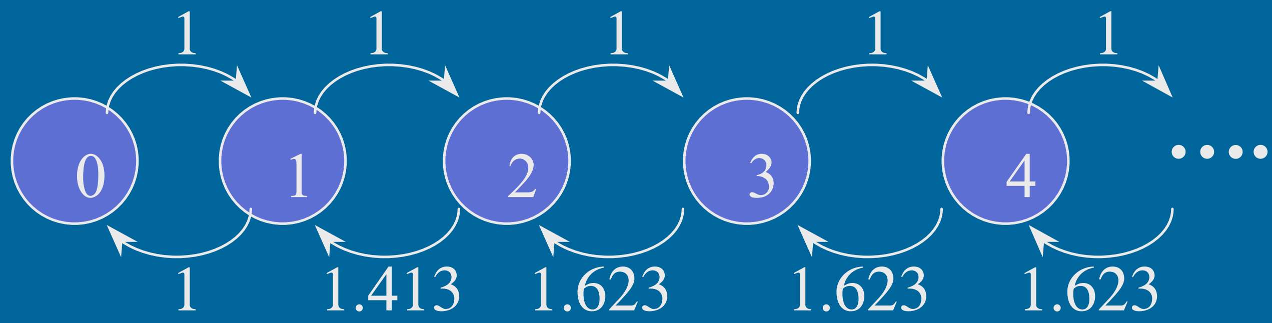
One Class Example



- Isolate system, use shortcuts
- Solve pprduct form model for mpl={1,2,3}
  - use MVA: X= {1, 1.413, 1.623}
- Create FESC: S(n) = {1, 1/1.413, 1/1.623, 1/1.623, ...)
  = {1, 0.708, 0.616, 0.616, …}
- Create new model
  - birth-death process
  - state dependent serv. rate: X= {1, 1.413, 1.623}



Solve Birth-Death Model (5)



$P_0 = 0.260$, $P_1 = 0.260$, $P_2 = 0.184$, $P_3 = 0.113$,
$P_{N>3} = 1 - 0.260 - 0.260 - 0.184 - 0.113 = 0.183$

average jobs in mem $= 0 * P_0 + 1\ P_1 + 2\ P_2 + 3\ (P_3 + P_{N>3})$
$= 0.260 + 2 * 0.184 + 3 * 0.296 = 1.516$

average number of waiting for memory jobs:

$$N_w = \sum_{i=1}^{\infty} i P_{i+3} = \frac{P_0\, \lambda^3}{\mu_1 \mu_2 \mu_3} \frac{\lambda / \mu_3}{[1-(\lambda / \mu_3)]^2} = 0.474$$

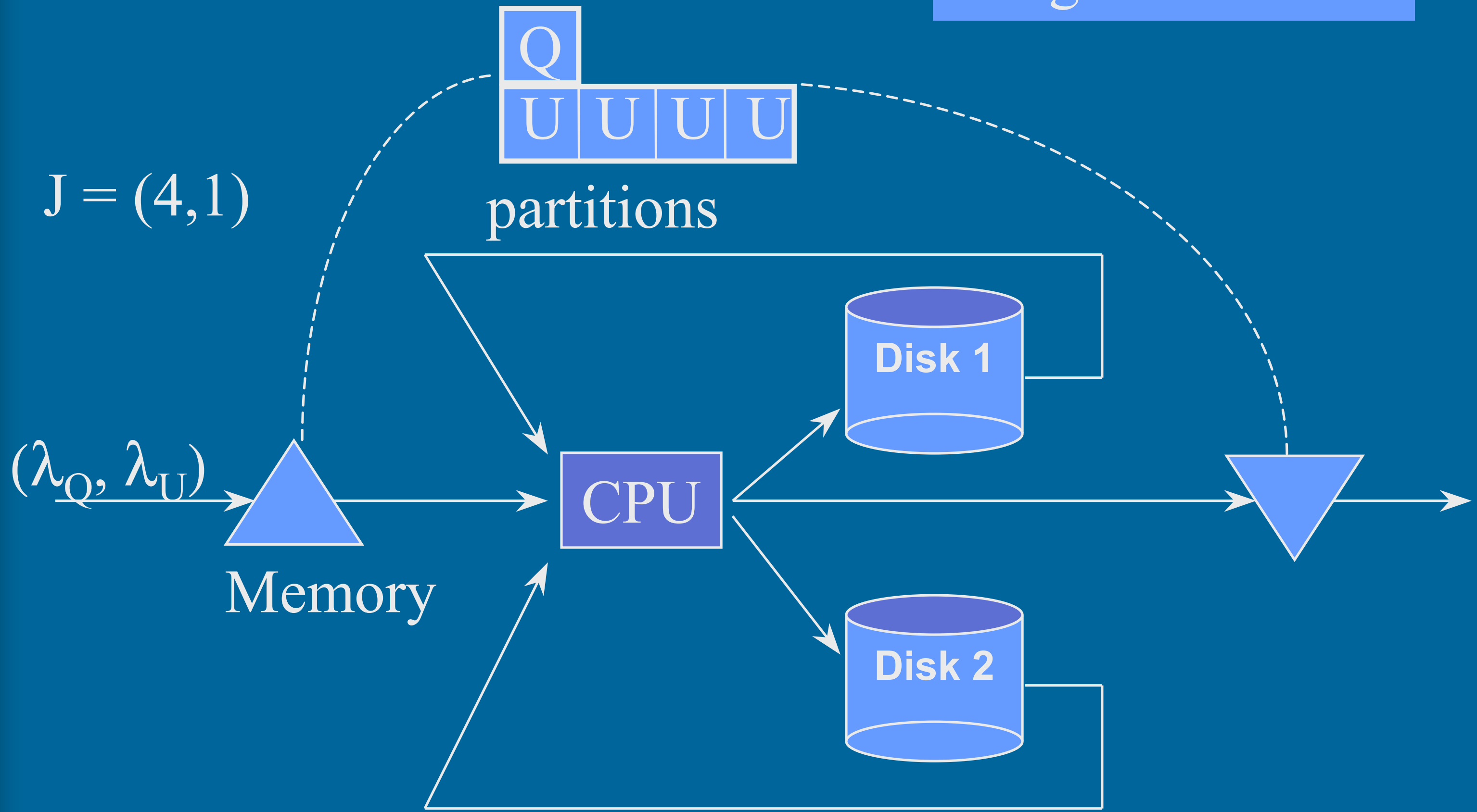
total popul $= 1.516 + 0.474 = 1.99$

time spent in memory queue: $0.474/1.99 = 23.8\%$

Memory Constraint, Many Classes

- Each class c has its own constraints J_c
  - each class in its own domain
  - classes are independent

4 segm for class 1
1 segm for class 2



Multiple Domain Solution

- Assumptions
  - Class r population is independent of population in other classes
  - Throughput in class r depends only on *average* propulations in other classes:

$$X_r(n_r) = f(\bar{n}_1, \bar{n}_2, \ldots, n_r, \ldots, \bar{n}_R)$$

- Iterative solution, one class at a time
  - solution for each class is not quite simple...

Model with Memory, Multiple Classes Iterative Solution

- Guess initial average populations $\overline{N}$
- Solve one class *r* at a time for population $\overline{N} = (\overline{n}_1, \overline{n}_2, \ldots, n_r, \ldots, \overline{n}_R)$
  - Get new average population for each class r: $\overline{n}_r, X_r, R_r, U_{kr}$

$$\overline{N} = (\overline{n}_1, \overline{n}_2, \ldots, \overline{n}_r, \ldots, \overline{n}_R)$$

- Iterate until "convergence"

$$\overline{N} = (2.1, 5.3, 2)$$

Iterative Memory Solution

- 1. Initialize
  - solve network without memory queue
  - get average popul. for each class: $\overline{n}_r^{nomem}$
  - set initial $\overline{n}_r = \min(\overline{n}_r^{nomem}, J_r)$
- 2. Create transformed model
  - remove memory constraint
  - make all memory constrained classes c closed (batch) job classes

Iterative Memory Solution (contd)

- 3. Solve the model for all constrained classes c, one class at a time:
  - solve it for populations
    - $\overline{N} = (\overline{n}_1, \overline{n}_2, \ldots, 1, \ldots, \overline{n}_R)$ (class c)
    - $\overline{N} = (\overline{n}_1, \overline{n}_2, \ldots, 2, \ldots, \overline{n}_R)$
    - …
    - $\overline{N} = (\overline{n}_1, \overline{n}_2, \ldots, J_c, \ldots, \overline{n}_R)$

    Approx MVA... why?

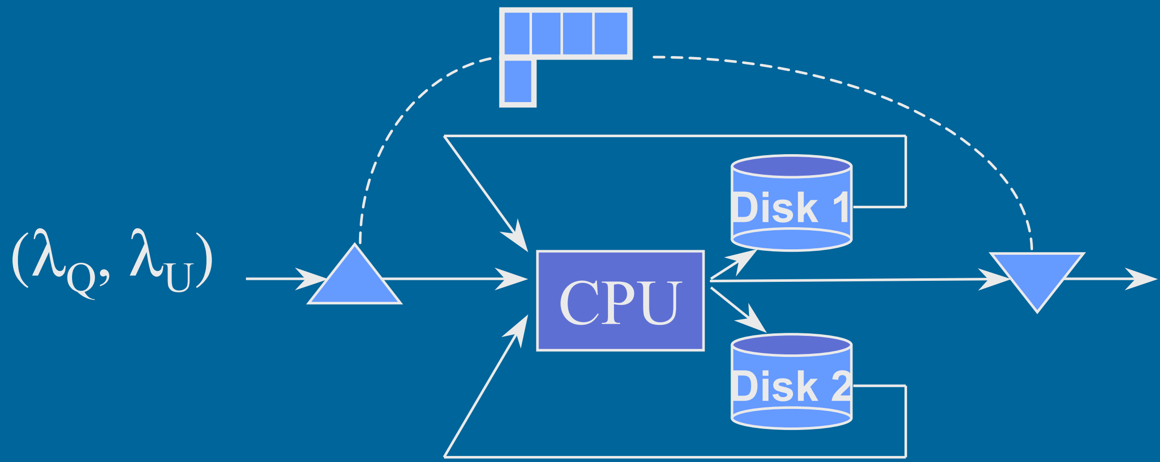
    population not integers!

    - get $X_c(n_c)$ for this class $\forall\, n_c \in \{1,\ldots,J_c\}$
  - create single class memory queue birth-death model for class c, with system as FESC
    - solve it, get new $\overline{n}_c =$ "average number of jobs in memory"
  - iterate until "done"

Iterative Memory Solution (contd)

- 4. Iterate step 3 until convergence
- 5. Get performance results for constrained classes c from the latest solutions for each such class
- 6. Solve model for unconstrained classes, using fixed $\overline{n}_c$'s for constrained classes

Multiple Class Example



- Two class model, Tbl 8.1
- Step 1. Solve open model without memory constraint

$$\overline{n}^{nomem} = (6.0192, 0.8540)$$

max pop = max mpl

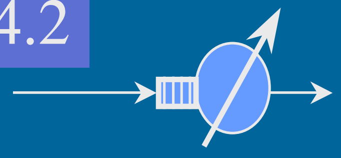
$$\text{init } \overline{n} = (\overline{n}_Q, \overline{n}_U) = (4.0, 0.8540)$$

Multiple Class Example (contd)

- Steps 2 & 3.

Single class open model

$\lambda_Q = 4.2$



$$\bar{n} = (1.0, 0.8540) \xrightarrow{appr.MVA} X_Q(1) = 2.542$$
$$\bar{n} = (2.0, 0.8540) \xrightarrow{appr.MVA} X_Q(2) = 3.577$$
$$\bar{n} = (3.0, 0.8540) \xrightarrow{appr.MVA} X_Q(3) = 4.115$$
$$\bar{n} = (4.0, 0.8540) \xrightarrow{appr.MVA} X_Q(4) = 4.434$$

$$S = \left(\frac{1}{2.542}, \frac{1}{3.577}, \frac{1}{4.115}, \frac{1}{4.434}, \frac{1}{4.434}, \ldots \right)$$

Birth-death

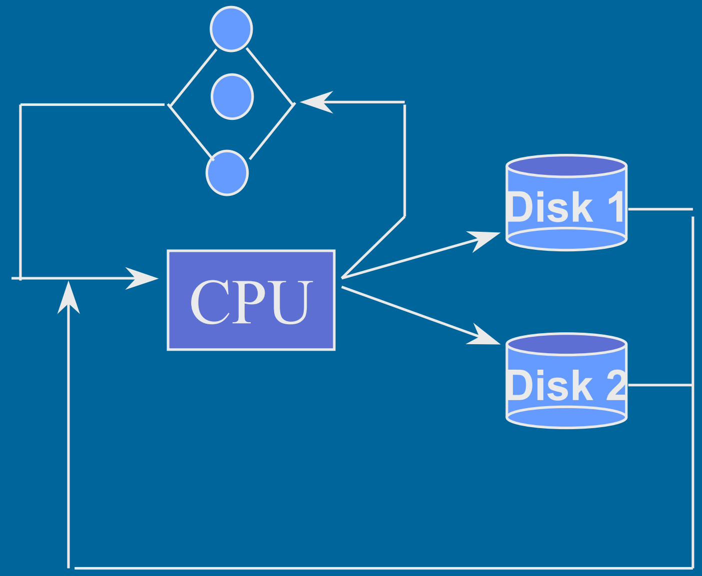
$\bar{n}_Q = 3.648$

Steps 4 & 5

Tbl 8.2

Tbl 8.3

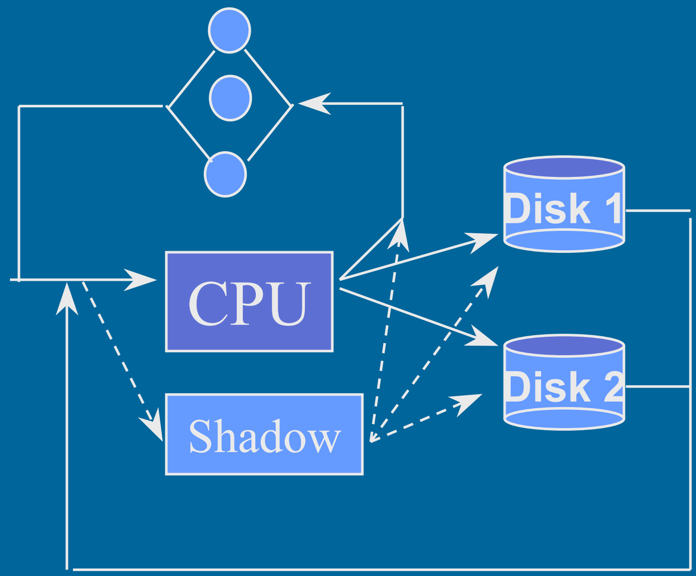
Priorities with Shadow Server



- Two job classes: Tbl 8.6
- No priorities: R = (2.69, 8.19) appr. MVA
  (2.37, 6.74) from PMVA

PMVA listing fig.8.6a.out

Shadow Server



- Class P: CPU "just for it"
- Class D: sees a "shadow" CPU as slower device
  - how much slower? $1/(1-U_{CPU,\,P}) = 1/(1-0.291)$
  - inflate demands D_{kD} this much for class D
- Get: model with no priorities

PMVA listing fig.8.6b.out

Fig 8.6 [Men 94]

PMVA listing fig.8.6.out

Many Priority Levels

- Generalized solution method for many priority levels
- Each level (but the one with highest priority) will get their own shadow server

Alg. 11.1 [LZGS 84]

- Shadow server utilizations of no use

Another Simple Example

(from Distr. OS homework)

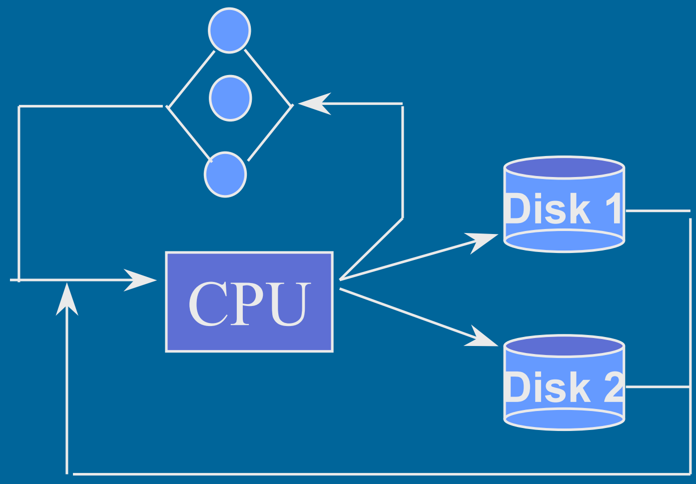
- We consider a supermarket with one check-out ("kassa"). A client arrives once every 3 minutes, and the average service time is 2.5 minutes. During a day, how long is the check-out clerk idle? On the other hand, how long will a client spend in the queue?
- Every fifth client has only one purchase, and for him/her the service time is only half a minute. The manager wants to improve the service for these "express clients". Two alternatives are considered: 1) an "express client" may pass the queue (but he/she is or she is not allowed to interrupt an ongoing service), 2) a new check-out is established for the "express clients".

- How would these alternatives affect the performance of the check-out service? Which alternative is better?
- How would it be possible to guarantee "express service" for "express clients" that the total delay is less than one minute?

Another Simple Example (contd)

- Basic 1-class solution — slides ASE 1-3
- Priority pre-emptive solution — slide ASE 4
- Priority non pre-emptive solution — slides ASE 5-7
- 2-server solution — slide ASE 8
- Basic 2-class solution — slides ASE 9-13

Paging



- Page fault rate depends on behaviour of other jobs, and total number of jobs in system
- Is paging disk the same unit as for files?

$D = D^{paging} + D^{file}$

nr of page faults * S^{paging}

Paging (contd)

- Nr of page faults?
  - Fig 8.8
  - Fig. 9.5 from [LZGS 84]
  - Nr page faults:

$$\frac{D_{cpu}}{IFT(f)} = \frac{D_{cpu}}{IFT(\frac{NP}{n})} = \frac{6 \text{ sec}}{1.5 \text{ sec / fault}} = 4 \text{ faults}$$

total Nr of Pages

nr of frames in average

Paging (contd)

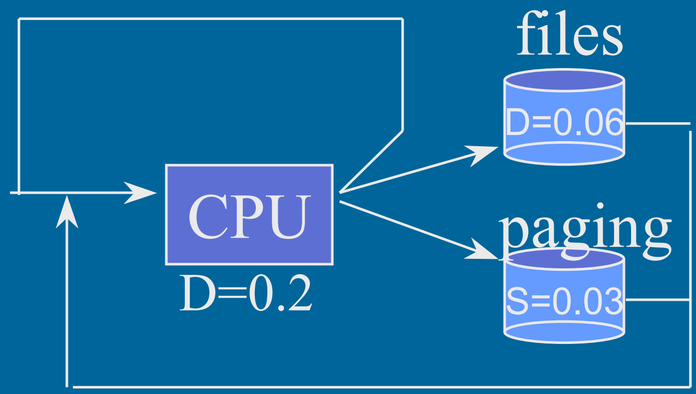
$$D_{disk}^{paging} = \frac{D_{cpu}}{IFT(f)} S_{disk}^{paging} = \frac{D_{cpu}}{D_{cpu} \Big/ \left(1 + \left(\frac{a}{NP/n}\right)^2 - \left(\frac{a}{F}\right)^2\right)} S_{disk}^{paging}$$

$$= \left[1 + \left(\frac{a}{NP/n}\right)^2 - \left(\frac{a}{F}\right)^2\right] S_{disk}^{paging}$$

F = nr of frames in virtual addr space

$$D_{disk} = \begin{cases} D_{disk}^{file} + D_{disk}^{paging} & \text{if } nF > NP \\ D_{disk}^{file} & \text{o / w} \end{cases}$$

Paging Example



page frame size

paging mem

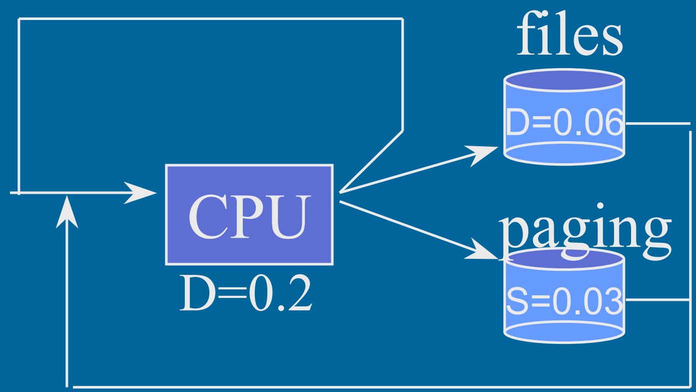
- Memory NP=10M/1K =10K = 10240
- Virt. addr. space = 2 MB
  - nr virt pages F = 2MB/1K = 1K = 2048
- match IFT(f) to data: magic a=4000

$$D_{d2}(n) = D_{d2}^{paging}(n) = \left[1 + \left(\frac{4000}{10240/n} \right)^2 - \left(\frac{4000}{2048} \right)^2 \right] * 0.03$$

$$= \left[1 + 0.153n^2 - 3.81 \right] * 0.03 \quad \text{if } n > \frac{NP}{F} = \frac{10240}{2048} = 5$$

$$D_{d2}(n) = 0 \qquad \text{o/w (i.e., } n \le 5 \text{)}$$

Paging Example (contd)



- Modify MVA to account for varying demand
- Solve, get system throughput as fn of load
  - Fig. 8.9
- X, R, U as function of load:
  - Figs. 9.6-9.8 from [LZGS 84]