More

FAIRNESS

WHERE WE ARE

- Last 2 lectures: One set of desirable properties of schedulers
- This lecture: More of the same
- Next lecture: Switch tracks, schedulers in practice

Switch to time vs space sheet here.
Space Sharing  Vs  Time Sharing

→ Not Useful
CFS

- Runs in rounds: all runnable processes run at least once each round. $T_R \leq$ total time for a round

- Each process $P$ (petit) has a weight ($W_p$) and a counter ($V_{run_p}$) tracking how much CPU time the process has already used.

- At the beginning of a round $R$, the scheduler computes a quanta ($Q_p$) for each process $P$: $Q_p = \left( \frac{W_p}{\sum_{p \in R \cap \text{run}} W_p} \right) \cdot T_R$

  and add $P$ to $\text{run}$

Scheduling Loop

- Pick process $P$ from $\text{run}$ s.t. $V_{run_p}$ is minimal
- Set preemption timeout for $Q_p$
Run process P.

If P blocks:

- Update VRUN
- ADD to BLOCKED

Else if P preempted:

- Update VRUN

P unblocked:

If runtime in round (P) < \( Q_p \)

- Add P to RUNQ
- Update \( Q_p \)

Properties:

- Fair when used to schedule single processor

  → Sharing Incentive (\( Q_p \))

  → Strategy proof (how unblocking works)

  → Pareto efficient

  → Envy free

Q: How many cores, how to extend?

The kernel developer's answer

\[ \frac{T_1}{T_2} \quad \frac{CFS_1}{CFS_2} \quad \frac{C_0}{C_1} \]
Problem: Everything depends on placement

Sharing Incentive (QP depends on placement)

Strategy Proof

× Pareto Efficient
× Envy Free (switching cores helps)

Bottom line: Composing schedulers is tricky

Dominant Resource Fairness

• Slightly different problem
  → Space Sharing
  → My job needs 1 core, 1 GPU and 2 GB of RAM before it even thinks about running.
  → Common for space partitioning
  → Increasingly common for time sharing
Possible Approaches

* Schedule 1 Resource Only (maybe bottleneck) (Switch Sheet)

* Treat All Resources The Same and Divide Fairly
  1 lottery ticket (of n) ~ \( \frac{1}{N} \cdot C \) cores, \( \frac{1}{N} \cdot M \) memory, ...

[Asset Fairness]: No Sharing Incentive

System: (30, 30)

\[ J_1 : (1, 3) \rightarrow J_1 : (6, 18) [6 \cdot (1, 3)] \]

\[ J_2 : (1, 1) \rightarrow J_2 : (12, 12) [12 \cdot (1, 1)] \]

* Come up with an exchange rate b/w resources

[CEEI]: Strategy Proof?

Does this matter in practice?
DRF

Job 1 \((a_1, b_1, c_1)\)
Job 2 \((a_2, b_2, c_2)\)
\vdots
Job K \((a_k, b_k, c_k)\)

FAIRLY ALLOCATE DOMINANT RESOURCE.

(SWITCH SHEET)

CAMPUSWIRE QUESTION:

"How to mix space & time?"
Real Ans: No Single Solution

Point Solution: DRFO

Side Quest: Memory vs Memoryless

If $P$ blocks:

- Update VRUN
- Add to blocked

Else if $P$ preempts

- Update VRUN

Do we need VRUN?

Is VRUN a problem?

Start Time Fair Queuing

- Another way to fairly schedule single resource
  (like CFS & lottery)
    → Designed for non-preemptible tasks

$P_i \square T_{i1} \square T_{i2} \ldots \square \text{GDF}$
**Basic Idea**

- **Track "Virtual Time"**
  - Time that would model an idealized version with infinitesimal quanta

- **When Task Ti Arrives**
  - Assign it a start & finish time

- **Pick Task with Smallest Start Time.**

**DRFQ**

- **Core Idea:**
  - Change how finish time \( F_{ij} \) is computed

Remember \( F_{ij} = S_{ij} + c \cdot L(T_{ij}) \)  
\[ \text{Time task used core.} \]
Change this to:

\[ F_y = S_y + c \cdot L \left( \text{Dominant } C_{T_y} \right) \]

**Task Uses Dominant Resource**

- **Pick Task with Smallest Start Time**

"Pick Smallest Start Time"

![Diagram](image)
Problem with Different Ratios

Job 0 <1 core, 4 GB>

Job 1 <3 core, 1 GB>
Different Resources and Ratios

Memory and CPU demands for Hadoop at Facebook. (From DRF paper)
Follows. Let total of DRF will equalize users' dominant shares, giving the al-
total memory, so user user user user user's dominant resource is CPU. Each task from
user's dominant resource is memory. Each task from user user user user user user user user user user
user as well as the user's dominant share, note that DRF need not always equalize users' domi-
nant shares of users. The algorithm tracks the total resources allocated to each
user, and competitive equilibrium from § 4.1) In Section 5, we present
algorithm for DRF allocation in the next section.

\[ A \] \begin{align*}
R &= \langle r_1, \ldots, r_m \rangle \quad \triangleright \text{total resource capacities} \\
C &= \langle c_1, \ldots, c_m \rangle \quad \triangleright \text{consumed resources, initially 0} \\
s_i \quad (i = 1..n) & \quad \triangleright \text{user } i \text{'s dominant shares, initially 0} \\
U_i &= \langle u_{i,1}, \ldots, u_{i,m} \rangle \quad (i = 1..n) \quad \triangleright \text{resources given to} \\
& \quad \text{user } i, \text{initially 0} \nonumber
\end{align*}

\textbf{pick} user \( i \) with lowest dominant share \( s_i \) \\
\( D_i \leftarrow \text{demand of user } i \text{'s next task} \) \\
if \( C + D_i < R \) then \\
\quad \begin{align*}
C &= C + D_i \\
U_i &= U_i + D_i \\
s_i &= \max_{j=1}^m \{ u_{i,j} / r_j \}
\end{align*}

\textbf{else} \\
\quad \text{return} \\
\textbf{end if} \\
\quad \triangleright \text{the cluster is full}