PBS: A Configurable Scheduling Policy

Hanhua Feng[1], Vishal Misra[1,2] and Dan Rubenstein[2,1]
Columbia University

[1]Department of Computer Science

[2]Department of Electrical Engineering

Columbia University in the City of New York

Sigmetrics 2007

Scheduling Policies in Queueing Models

Scheduling is a compromise ...

- not only between individual tasks, but also ...
- between systems with different workload patterns,
- between different performance requirements, including
  - mean response time, mean slowdown, responsiveness, ...
  - fairness measures: seniority, RAQFM, ...

Our work

- Design a flexible scheduling policy to balance these requirements.

Assumptions in this talk

- Single-server queueing model
- Work-conserving, preemption allowed

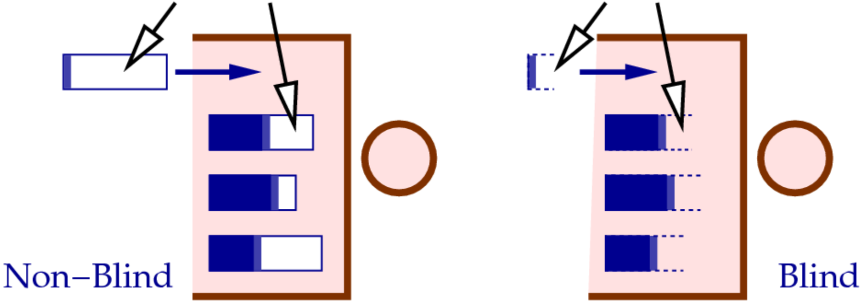
Blind Scheduling Policies

Non-blind policies

Know required and remaining service time when tasks arrive.

Blind policies

No information about remaining service until tasks complete.



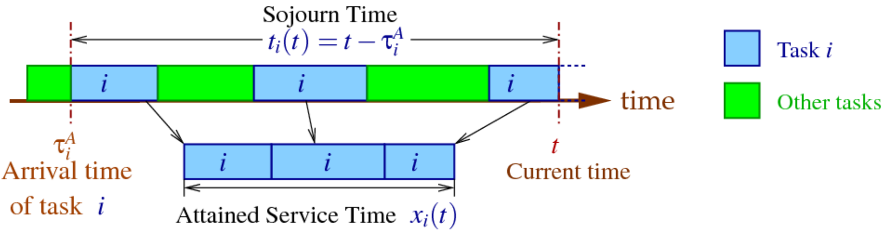
Non-blind policy examples

SJF, SRPT, SMART . . .

Blind policy examples

FCFS, PS, LAS, LCFS, . . .

How Do We Measure Fairness of a Policy?



Fairness criteria [cf. Raz,Levy&Avi-Itzhak 2004]

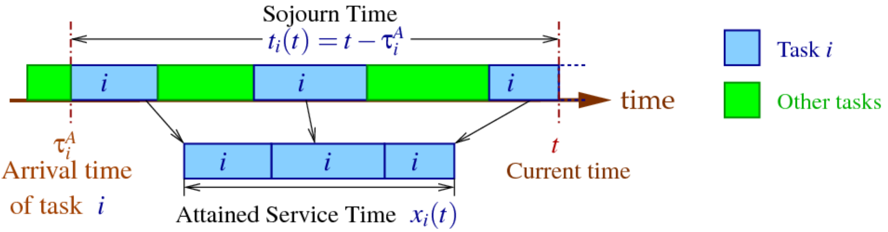
- Task seniority (emphasis on t_i) $\Rightarrow$ FCFS
- Task service requirements (emphasis on x_i)
  - Equal attained service $\Rightarrow$ LAS/FBPS
- Combination of the two: Equal share of processor
  - Current: $dx_i(t)/dt_i(t) \equiv x_i'(t)$ $\Rightarrow$ PS
  - Aggregated: $x_i(t)/t_i(t)$ $\Rightarrow$ GAS

How to Measure Fairness of a Policy? (cont'd)

Fairness measures in the literature

- Comparison vs FCFS [Wang & Morris 1985]
- RAQFM: Comparison vs PS [Raz,Levy&Avi-Itzhak 2004]
  - A quantitative measure.
  - Difficult to analyze: with results for FCFS, LCFS, PLCFS, and Random in $M/M/1$.
  - $G/D/m$ [Raz,Levy&Avi-Itzhak 2005]
- Expected slowdown for given required service $E[S|X = x]$ compared with PS [Wierman&Harchol-Balter 2004]
  - A classification: always fair/unfair, sometimes fair.
  - Assume $M/G/1$.
  - Extended in [Wierman&Harchol-Balter 2005].
- SQF [Avi-Itzhak,Brosh&Levy 2007]

Balance Between Two Fairness Criteria



Two fairness criteria (cont'd)

- Seniority — Prefer larger sojourn time $t_i(t)$
- Service requirements — Prefer smaller attained service $x_i(t)$

Our idea: A configurable balance

- Schedule a task with maximal $t_i(t) - \alpha x_i(t)$.
- More general: $g(t_i(t)) - \alpha g(x_i(t))$, e,g., $\log t_i(t) - \alpha \log x_i(t)$.

Our Parameterized Scheduler: PBS

The PBS policy with a single server

- For every task i, compute its **priority value**

$$p_i(t) = \log t_i(t) - \alpha \log x_i(t), \quad \text{Equivalent to} \quad P_i(t) = \frac{t_i(x)}{[x_i(t)]^\alpha}$$

- α is a configurable parameter in $[0, \infty)$.
- At time t, serve the task with the highest priority p_i (or P_i).
  - Randomly choose among equal-priority tasks.
  - Preempt low-priority tasks, if currently been served.
- Can be used in continuous time (theory)
  or in discrete time (practice).

PBS: Priority-based Blind Scheduling (cont'd)

Why PBS?

- Tunable: Parameter α can be changed from 0 to ∞.
  - Emulate well-known policies: $P_i = t_i / x_i^{\alpha}$
  - $\alpha = 0$: First-come first-serve (FCFS) $P_i = t_i$
  - $\alpha \to \infty$: Least attained service (LAS), a.k.a. Foreground-Background Processor-Sharing (FBPS) $P_i \sim 1/x_i$
  - $\alpha = 1$: Greatest Attained Slowdown (GAS), closely emulate Processor-Sharing (PS). $P_i = t_i / x_i$
  - $\alpha =$ other values: Hybrid policies.
- Blind: Using only past information (t_i, x_i)
- Simple: Easy to implement.
- Dimensionless: Not dependent on scale of time unit (minute, second).

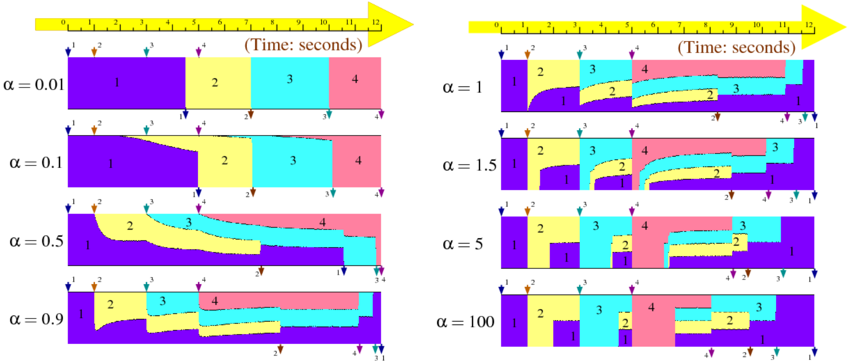
Behavior of PBS

An example

- Four tasks in 4 colors
- Arrival time: 0s,1s,3s,5s
- Service: 4.5s,2.5s,3s,2s

How to read the graphs

- X-axis: Time
- Y-axis: CPU utilization per task.
- Area: Service received.



PBS: Smoothly Move Across Hybrid Policies

The smoothness of PBS with respect to α

- α varies from 0 to ∞.
- X-axis: Time
- Y-axis: CPU utilization per task.
- Area: Service time received.

Properties of PBS for $0 < \alpha < \infty$

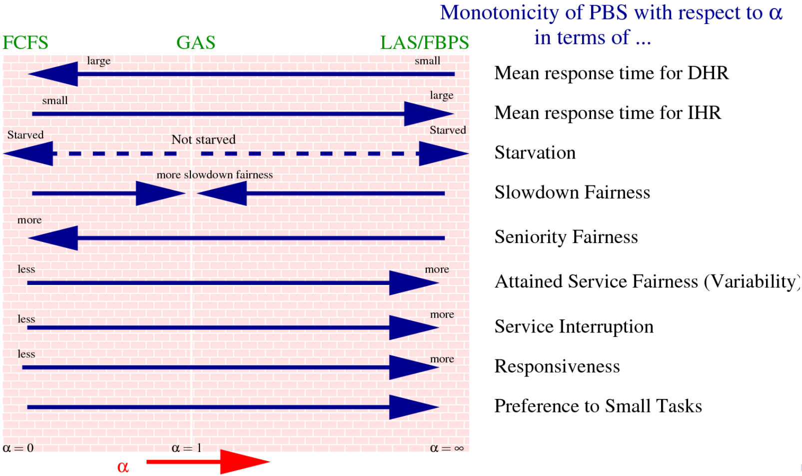
Some properties of PBS proved in the paper

- A new task immediately receives service after arrival.
  - Small CPU fraction for $\alpha < 1$
  - Large CPU fraction for $\alpha > 1$.
- **Seniority**: Earlier tasks get more attained service.
- **Time-shared**: CPU may be shared by two or more tasks.
  - **Hospitality**: A new task always gets a CPU share.
- **Convergence**: Converge to PS in a long run for long jobs.
  - Converge to DPS with an offset to log formula,
- **No Starvation**: Priority values of temporarily blocked tasks increase towards infinity, and will become highest-priority task.
  - For α close to 0 (FCFS) or ∞ (LAS), tasks may be blocked for a long time.

PBS Tunability: A Graphical Conclusion

PBS is monotonic in many aspects

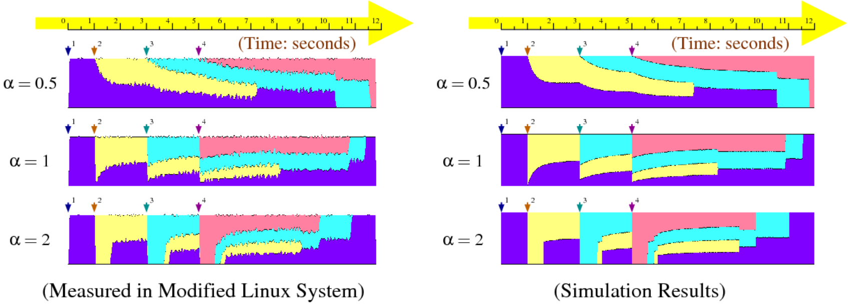
- Guidelines for tuning α manually.



Implementation in Linux Kernel

CPU utilization measurement

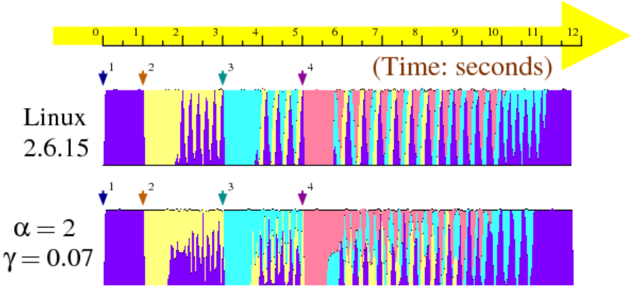
- Discrete time implementation in Linux 2.6.15.
- 50ms moving average of measured CPU utilization per task.
- Measurement results are close to simulation results.
- Difference is the roughness on small time scales.



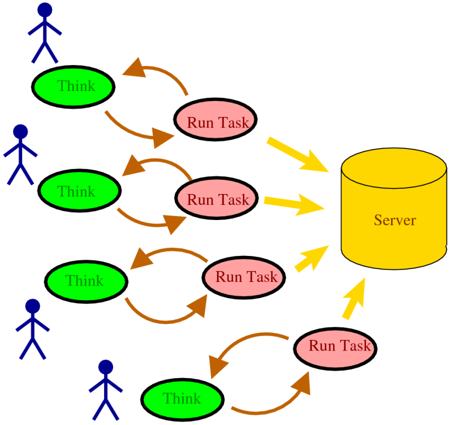
Emulating Existing Linux Scheduler

A small tweak

- Add a bonus priority γ to the current task in order to limit context switch.
- With $\alpha = 2$ and $\gamma = 0.07$, PBS looks close to Linux native scheduler.



Experimental model

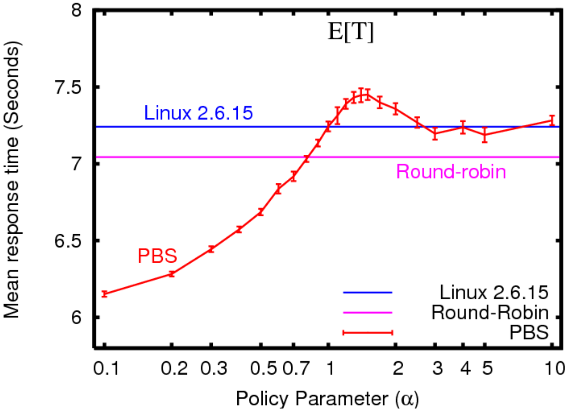


A closed model

- A fixed number of users.
- Each user submits a task after thinking.
- Exponentially distributed thinking time.
- Response time of every task is measured.

Experimental Results (Set A)

- Computational tasks with almost deterministic CPU usage.
- About 3-second processing for each task.
- 8 users, 25s average thinking time.

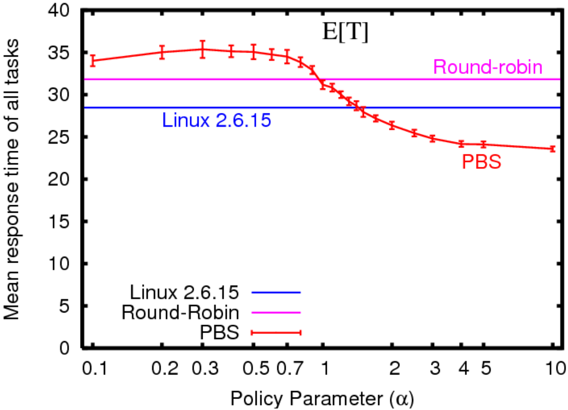


For this work load,

- small α works best.
- PBS ($\alpha < 0.7$) outperforms Linux and Round-robin.

Experimental Results (Set B) (1/2)

- Apache web server 2.0, dynamic pages with heavy processing.
- Overloaded with 30 users, 10s average thinking time.
- Processing time is heavy-tailed.



For this workload,

- big α works best.
- PBS ($\alpha > 2$) outperforms Linux and Round-robin.

Conclusion

Different α's are better for different workloads.

Conclusion and Future Work

Conclusion of contribution

- We introduce a novel configurable policy, PBS.
- By varying the single parameter, we can tune for various performance and fairness requirements.
- Demonstrate properties and advantages of PBS by analysis, simulations, implementation, and experiments.

Current/Future work

- Closed form of mean response time in $M/G/1$.
- Design an automatic mechanism to dynamically adapt α to workload.
- Extend PBS to multi-core systems.

The End