Optimality, Fairness and Robustness in Speed Scaling Designs

Lachlan Andrew

Joint work with
Minghong Lin, Caltech
Adam Wierman, Caltech
Kevin Tang, Cornell

Supported by ARC, NSF
Systems must balance delay and energy use

→ Speed scaling
Speed scaling balances energy vs delay

- Dynamic voltage scaling
- Spin down
- Radio power
- Applications
- Power off
Goal: $\min \beta \cdot E[\text{Delay}] + E[\text{Energy}]$

$= 1$ in this talk
A simple model

state-dependent controllable service rates

Goal: \( \min \beta \cdot \mathbb{E}[\text{Delay}] + \mathbb{E}[\text{Energy}] \)

\[ P(s) = \text{power to run at speed } s \]

 CPUs, disks:
\[ P(s) = s^\alpha \] (often \( \alpha \approx 2 \))
A simple model

state-dependent controllable service rates

Goal: \( \min \beta \cdot E[Delay] + E[Energy] \)

Many other objectives possible:
- Minimize energy subject to QoS guarantee
- Minimize delay subject to energy budget
- Delay \( \times \) energy
A simple model

What order to serve jobs

The speed of the server

?
Fundamental questions about speed scaling

Can on-line speed scaling be optimal?
   What structure do optimal algorithms have?

How does speed scaling interact with scheduling?

Are sophisticated speed scalers better?

Are there drawbacks of speed scaling?
A simple model

What order to serve jobs

The speed of the server

What can we say about the optimal design?
A simple model

What order to serve jobs → The speed of the server

SRPT
Shortest Remaining Processing Time

?
A simple model

- What order to serve jobs
  - SRPT
  - ...but in practice often...
  - PS
  - Processor Sharing

- The speed of the server
  - ?
An example – batch arrivals under SRPT

$n$ jobs of size $1$ arrive at time 0 (no other arrivals)

$$\text{Cost} = \left(\frac{1}{s_n} + \frac{1}{s_n} P(s_n)\right) + \left(\frac{1}{s_n} + \frac{1}{s_{n-1}} + \frac{1}{s_{n-1}} P(s_{n-1})\right) + \ldots$$

$$= \left(\frac{n}{s_n} + \frac{1}{s_n} P(s_n)\right) + \left(\frac{n-1}{s_{n-1}} + \frac{1}{s_{n-1}} P(s_{n-1})\right) + \ldots$$

$P(s_n) = s^\alpha \implies s_n = P^{-1}\left(\frac{n}{\alpha - 1}\right)$

Observation 1: Optimal speeds only change at arrival/departure instants.

“Energy-proportional” computing
What if the jobs have different sizes?

$n$ jobs of size $x_i$ arrive at time 0 (no other arrivals)

\[
Cost = \left( \frac{x_n}{s_n} + \frac{x_n}{s_n} P(s_n) \right) + \left( \frac{x_n}{s_n} + \frac{x_{n-1}}{s_{n-1}} + \frac{x_{n-1}}{s_{n-1}} P(s_{n-1}) \right) + \ldots
\]

\[
= x_n \left( \frac{n}{s_n} + \frac{1}{s_n} P(s_n) \right) + x_{n-1} \left( \frac{n-1}{s_{n-1}} + \frac{1}{s_{n-1}} P(s_{n-1}) \right) + \ldots
\]

\[P(s_n) = s^\alpha \implies s_n = P^{-1}\left( \frac{n}{\alpha - 1} \right)\]

Observation 2: Speeds only depend on number of jobs in system (not on sizes)
What about under PS?

$n$ jobs of size 1 arrive at time 0 (no other arrivals)

Observation 3: Optimal speeds for PS match those for SRPT (in this case)
2 design choices

What order to serve jobs

The speed of the server

SRPT
or
PS

$s_n = ?$
Two styles of analysis

<table>
<thead>
<tr>
<th></th>
<th>Stochastic analysis</th>
<th>Worst-case analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(M/GI/1)</td>
<td>(no assumptions)</td>
</tr>
<tr>
<td><strong>SRPT</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[Pruhs et al. 04]</td>
<td>[Albers, Fujiwara 06]</td>
</tr>
<tr>
<td></td>
<td>[Albers, Fujiwara 06]</td>
<td>[Bunde 06]</td>
</tr>
<tr>
<td></td>
<td>[Bunde 06]</td>
<td>[Bansal et al. 09]</td>
</tr>
<tr>
<td><strong>PS</strong></td>
<td>[George, Harrison 01]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[Bradley 05]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[Wierman, Andrew, Tang 09]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[Chan, Edmonds, Pruhs 09]</td>
</tr>
</tbody>
</table>
## Two styles of analysis

<table>
<thead>
<tr>
<th></th>
<th>Stochastic analysis (M/GI/1)</th>
<th>Worst-case analysis (no assumptions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRPT</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>PS</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

The table shows the comparison between stochastic and worst-case analysis styles for different scenarios (SRPT and PS). The diagrams illustrate the flow or transitions between these scenarios.
$s_n = P^{-1}(n)$ seems to work well

Model introduced [Yao et al. 1995]

$(SRPT, P^{-1}(n))$ proposed [Pruhs et al. 03]

Unit sized jobs, $P(s) = s^a$ (unbounded $s$)

$$
\left(\frac{3 + \sqrt{5}}{2}\right)^\alpha - \text{competitive} \quad [\text{Albers, Fujiwara 06}]
$$

$$
4 - \text{competitive} \quad [\text{Bansal et al. 06}]
$$

General jobs sizes, $P(s) = s^a$

$$
O\left(\frac{\alpha}{\log(\alpha)}\right) - \text{competitive} \quad [\text{Lam et al. 07}]
$$

$$
\quad [\text{Bansal et al. 08; bounded $s$}]
$$

General jobs sizes, General $P(s)$: $(s_n = P^{-1}(n))$

$$
3 - \text{competitive} \quad [\text{Bansal et al. 09}]
$$

Is 3 the best possible?
Robustness guarantee:
“Good” performance regardless of workload

Theorem:
\((\text{SRPT}, P^{-1}(n))\) is 2-competitive for all \(P\).

Allows \(P\) to be bounded, non-continuous, non-convex, etc.
Theorem:
(SRPT, P^(-1)(n)) is 2-competitive for all P.

Proof:
1) assume convex, continuous, unbounded P
   -- Potential function:
   \[
   \phi(t) = \int_0^\infty \sum_{i=1}^{n(q,t)-n_{opt}(q,t)} 2P'(P^{-1}(i)) \, dq
   \]

2) extend to general P
   -- Result not quite (SRPT, P^(-1)(n))

Allows P to be bounded, non-continuous, non-convex, etc.
Theorem: exactly
(SRPT, P^{-1}(n)) is \overset{\wedge}{2}\text{-competitive} (for unbounded power functions)
Proof (lower bound):

1 Job of size 1 arrives every $1/r$ seconds

Optimal speed scaling:
Serve at rate $r$.

**(SRPT,$P^{-1}(n)$):**
Must have $P(r)$ jobs before serving at rate $r$

Cost / job $\approx \frac{P(r) + P(r)}{r}$

$\geq 2$ - competitive
Theorem: For all “natural” algorithms there exists a $P$ where the algorithm is $= 2$-competitive.

Our Results

Theorem: $(SRPT, P^{-1}(n))$ is $\leq 2$-competitive (for unbounded power functions).

All algorithms that use either
- a policy that serves 1 job at a time, or
- speeds that grow slower, faster, or proportional to $P^{-1}(n)$
Theorem: \textit{exactly} (SRPT, P^{-1}(n)) is $\leq 2$-competitive (for unbounded power functions)

Theorem: For all "natural" algorithms there exists a $P$ where the algorithm is $= 2$-competitive.

(almost) A fundamental limit on robust optimality of a scheduler + speed scaler

...recall w/o speed scaling on-line could be optimal
Our Results

Theorem: *exactly* \((\text{SRPT}, P^{-1}(n))\) is \(\leq 2\)-competitive (for unbounded power functions).

Theorem: For all "natural" algorithms there exists a \(P\) where the algorithm is \(= 2\)-competitive.

Proof:
Consider \(P(s) = s^\alpha, \; \alpha \to \infty\).
No natural scheme can be < 2-competitive for even the periodic & batch cases.
<table>
<thead>
<tr>
<th></th>
<th>Stochastic analysis (M/GI/1)</th>
<th>Worst-case analysis (no assumptions)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SRPT</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>$\text{(SRPT, } P^{-1}(n))$</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>exactly 2-competitive</td>
</tr>
<tr>
<td><strong>PS</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PS vs **SRPT**

When just delay is concerned:

-- **PS** is $O(n/\log(n))$-competitive
-- **SRPT** is 1-competitive

So we expect PS to be worse...
Prior work

Theorem:
No blind policy is constant competitive for (delay+energy) across all P(s).

[Chan et al. 2009]
Theorem:
\((PS, P^{-1}(n))\) is \((4a-2)\)-competitive for typical \(a\) when \(P(s) = s^a\)

**Note:** PS is \(O(n/\log(n))\)-competitive in constant speed model.

Energy proportionality works for PS too.

Speed scaling reduces the impact of scheduling.
<table>
<thead>
<tr>
<th></th>
<th>Stochastic analysis (M/GI/1)</th>
<th>Worst-case analysis (no assumptions)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SRPT</strong></td>
<td></td>
<td>(SRPT, $P^{-1}(n)$) Exactly 2-competitive</td>
</tr>
<tr>
<td><strong>PS</strong></td>
<td></td>
<td>(PS, $P^{-1}(n)$) $(4\alpha-2)$-competitive</td>
</tr>
</tbody>
</table>
Dynamic PS

"Textbook" stochastic control problem:

Can “solve” numerically with dynamic programming

[Bertsekas] [Ross] [George & Harrison 01] …
Theorem:
In an $M/GI/1$ queue with $a=2$, the optimal speeds satisfy

$$\rho + \sqrt{n-2\rho} \leq s_n \leq \rho + \sqrt{n + \min\left(\frac{\rho}{2n}, \rho^{1/3}\right)}$$

For large $n$, $s_n \sim \rho + \sqrt{n}$

(see the paper for case of general $a$)
<table>
<thead>
<tr>
<th></th>
<th>Stochastic analysis (M/GI/1)</th>
<th>Worst-case analysis (no assumptions)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SRPT</strong></td>
<td>$s_n \approx \rho + \sqrt{n}$ $(\alpha = 2)$</td>
<td>(SRPT, $P^{-1}(n)$) Exactly 2-competitive</td>
</tr>
<tr>
<td><strong>PS</strong></td>
<td>$s_n \approx \rho + \sqrt{n}$ $(\alpha = 2)$</td>
<td>(PS, $P^{-1}(n)$) (4\alpha-2)-competitive</td>
</tr>
</tbody>
</table>

[Wierman, Andrew, Tang 09]
Switching to SRPT can’t improve performance much!

How well does it work?
Fundamental questions about speed scaling

Can on-line speed scaling be optimal?
What structure do optimal algorithms have?

Can’t be optimal!

no “natural” algorithm can do better than 2-competitive

Open question: Can we drop “natural”?

Can be 2-competitive by using SRPT and

\[ s_n = P^{-1}(n) \]

Theoretical support for energy proportionality
Fundamental questions about speed scaling

Can on-line speed scaling be optimal?

What structure do optimal algorithms have?

How does speed scaling interact with scheduling?

They can be decoupled with little performance loss.

Energy proportionality works well under SRPT & PS.

Scheduling doesn’t matter as much as it used to.

PS can be constant competitive when energy is considered.

Open question: How generally do these hold?
2 design choices

What order to serve jobs

SRPT or PS

The speed of the server

Dynamic

Gated static
# Gated static speed scaling

<table>
<thead>
<tr>
<th></th>
<th>Stochastic analysis (M/GI/1)</th>
<th>Worst-case analysis (no assumptions)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SRPT</strong></td>
<td>$\frac{\beta E[N]}{(1 - \rho/s) \log(1/(1 - \rho/s))} = P(s)$ (heavy traffic)</td>
<td>( \times )</td>
</tr>
<tr>
<td><strong>PS</strong></td>
<td>$s = \rho + 1$ (( \alpha = 2 )) [Wierman, Andrew, Tang 09]</td>
<td>( \times )</td>
</tr>
</tbody>
</table>

\[ EN \] 

\[ Ps \] 

\[ br \] 

\[ 1 \] 

\[ log \] 

\[ 0 \]
Gated Static vs. Dynamic (under SRP)

Why is dynamic speed scaling used?

Gated Static vs. Dynamic (under SRP) ...same holds under PS
Robustness

Why is dynamic speed scaling used?
Cost per job

gated static

real ?

design ?

Dynamic (worst-case control)

Dynamic (stochastic control)

Robustness/Optimality Tradeoff
Theorem:
For $P(s)=s^2$, speeds optimal for PS with $\lambda$ are $2(2\lambda^2 + 2\lambda + 1)$-competitive under SRPT and $6(2\lambda^2 + 2\lambda + 1)$-competitive under PS.

Tradeoff between optimality and robustness through tuning estimated load.

worst-case result for optimal stochastic control.
Fundamental questions about speed scaling

Can on-line speed scaling be optimal?
   What structure do optimal algorithms have?

How does speed scaling interact with scheduling?

Are sophisticated speed scalers better?

→ Gated static is almost as good as dynamic

Dynamic speeds give robustness to uncertain load

Open question: What is the impact of switching costs?
Fundamental questions and answers about speed scaling

Can on-line speed scaling be optimal?
   What structure do optimal algorithms have?

How does speed scaling interact with scheduling?

Are sophisticated speed scalers better?

Are there drawbacks to speed scaling?

   Speed scaling can “magnify/create” unfairness

   Job types that are often served when the queue is large get faster service
Which jobs are run fast?

...the ones run when the queue is big

- SRPT: Magnifies the bias towards small jobs
- FCFS: Creates a bias towards big jobs
- PS: Remains fair
Theorem:
With dynamic speed scaling:

$$\lim_{x \to \infty} \frac{E[T_{SRPT}^x(x)]}{x} = \lim_{x \to \infty} \frac{E[T_{PS}^x(x)]}{x}$$

[Sigman, Harchol-Balter, Wierman 02]

SRPT is “fair” to large job sizes

With constant speeds:

$$\lim_{x \to \infty} \frac{E[T_{SRPT}^x(x)]}{x} = \lim_{x \to \infty} \frac{E[T_{PS}^x(x)]}{x}$$

Bias against large jobs is magnified
Fundamental questions about speed scaling

Can on-line speed scaling be optimal?
What structure do optimal algorithms have?

How does speed scaling interact with scheduling?

Are sophisticated speed scalers better?

Are there drawbacks of speed scaling?

→ Speed scaling can “magnify/create” unfairness

Open question: How significant is the unfairness?
Optimality, Fairness and Robustness in Speed Scaling design

For delay only: SRPT has all three
Optimality, Fairness and Robustness

(SRPT, $P^{-1}(n)$)
Optimality, Fairness and Robustness

(SRPT, P^{-1}(n))
(PS, P^{-1}(n))
Optimality, Fairness and Robustness

(SRPT, gated-static)

(SRPT, $P^{-1}(n)$)
(PS, $P^{-1}(n)$)
Optimality, Fairness and Robustness in Speed Scaling design

Lachlan Andrew
Minghong Lin
Adam Wierman