Speed Scaling to Manage Temperature
Joint work with Guillaume Aupy (ENS Lyon)
Daniel Cole, Kirk Pruhs (University of Pittsburgh)

Leon Atkins
University of Bristol

April 19, 2011
Overview

Speed Scaling to Minimise the Maximum Temperature

Our Results:

- In the batched release model, provide optimal algorithms, with a known or unknown optimal maximum temperature ($T_{\text{max}}$)
- In the general online model, provide a $\frac{e}{e-1}(\ell + 3e^\alpha)$-competitive algorithm when the optimal maximum temperature is known ($\ell \leq 2$)
What is Speed Scaling?

- Allows frequency/voltage on a processor to be lowered/raised so that the processor runs more slowly/quickly.
- Reducing the speed reduces the rate work is completed, but also reduces the energy required per unit work.
- In general, power consumed at a rate $P(s) = s^\alpha$, where $\alpha > 1$ and $0 \leq s \leq \infty$ is the processor speed.
- Temperature change at time $t$ is modelled as $T'(t) = aP(t) - bT(t)$. 
Deadline Scheduling

- Each job $i$ has an associated release time $r_i$, deadline $d_i$, and processing requirement (work) $p_i$.
- For each job, $p_i$ units of work must be scheduled between $r_i$ and $d_i$.
- Work is $\int s \, dt$.
Deadline Scheduling

- Each job $i$ has an associated release time $r_i$, deadline $d_i$, and processing requirement (work) $p_i$
- For each job, $p_i$ units of work must be scheduled between $r_i$ and $d_i$
- Work is $\int s \, dt$

- Work is released online - the scheduler has no knowledge of future jobs
- When a job is released, the scheduler learns the deadline and processing requirement of the job
- Measure the success of our algorithm by examining the competitive ratio
First: some definitions

We will make extensive use of the following functions in the analysis

- $UMaxW(0, t_1, T_0, T_1)(t) = \left( \frac{T_1 - T_0 e^{-bt_1}}{e^{-bt_1} - e^{-bt_1}} \right)^{\frac{1}{\alpha}} \left( \frac{b}{\alpha - 1} \right)^{\frac{1}{\alpha} - 1} \left( 1 - e^{-\frac{bt}{\alpha - 1}} \right)$

- $MaxW(0, t_1, T_0, T_1)(t) =$
  \begin{align*}
  &UMaxW(0, \gamma, T_0, T_1)(t) & : t \in [0, \gamma) \\
  &UMaxW(0, \gamma, T_0, T_1)(\gamma) + (bT_1)^{\frac{1}{\alpha}} (t - \gamma) & : t \in (\gamma, t_1]
  \end{align*}

  Where $\gamma$ is defined implicitly by
  \[
  \frac{1}{\alpha - 1} T_0 e^{-\frac{b\gamma\alpha}{\alpha - 1}} + T_1 - \frac{\alpha}{\alpha - 1} T_{\max} e^{-\frac{b\gamma}{\alpha - 1}} = 0
  \]

**Intuitively:** Follow a $UMaxW$ curve when increasing $T$, follow $(bT)^{1/\alpha}$ when maintaining it
Batched Release: Known $T_{\text{max}}$

**Batched Release**

All jobs released at time $t = 0$.

We consider this as testing the feasibility of a schedule $S$ with constraints in the form $W(S, d_i) \geq w_i$, where $W(S, d_i)$ is the total work of $S$ by $d_i$.
**Batched Release: Known $T_{\text{max}}$**

**Batched Release**
All jobs released at time $t = 0$.

We consider this as testing the feasibility of a schedule $S$ with constraints in the form $W(S, d_i) \geq w_i$, where $W(S, d_i)$ is the total work of $S$ by $d_i$.

**Algorithm:** iteratively construct schedules $S_0, ..., S_n$

- By definition, the schedule $S_0$ is defined by $\text{Max}W(0, d_n, 0, T_{\text{max}})(t)$.
- Use $S_{i-1}$ as $S_i$ unless $S_{i-1}$ breaks $i$-th work constraint $(W(S_{i-1}, d_i) < w_i)$
- *What to do then?*
Consider constraints s.t for any $j < i$, $W(S_{i-1}, d_j) = w_j$ - these are tight constraints.

$S_{i,j}$ is the set of all possible schedules where, during $[d_j, d_i]$, $S_{i,j}$ is speeding up to meet the $i$th work constraint, whilst minimising the temperature at time $d_i$.

Calculate the temperature at $d_i$ in the new schedule by solving a $UMaxW$ equation.

New schedule $S_i$ follows $S_{i-1}$ until $t = d_j$, follows a $UMaxW$ curve from $d_j, d_i$, and then follows a $MaxW$ curve from $d_i, d_n$. 

Batched Release: Known $T_{\text{max}}$
Now imagine we don’t have the $T_{\text{max}}$ in the first step of the previous algorithm.

We could guess $T_{\text{max}}$ and then binary search to find the optimal, and then carry on as normal from there?
Unknown Maximum Temperature: Algorithm Overview

Now imagine we don’t have the $T_{\text{max}}$ in the first step of the previous algorithm.

We could guess $T_{\text{max}}$ and then binary search to find the optimal, and then carry on as normal from there?

**Better way:**

- From before: optimal schedule is a concatenation of $UMaxW$ curves $C_1, \ldots, C_{k-1}$, with a possible single $MaxW$ at the end.
- Each $C_i$ begins at the time of the $(i - 1)$st tight work constraint
- On the $i$-th iteration, our algorithm computes $C_i$ from $C_1, \ldots, C_{i-1}$
- Which is good news! We only need to find $C_1$
Unknown Max Temp: Finding \( C_1 \) (Overview)

1. Classify all \( n \) work constraints as \( UMaxW \) or \( MaxW \)
1. Classify all $n$ work constraints as $UMaxW$ or $MaxW$
2. From all the $UMaxW$ constraints, pick the candidate $C_1$ (call this the $UMaxW$-winner)
Unknown Max Temp: Finding $C_1$ (Overview)

1. Classify all $n$ work constraints as $UMaxW$ or $MaxW$
2. From all the $UMaxW$ constraints, pick the candidate $C_1$ (call this the $UMaxW$-winner)
3. Extend this curve to run until $d_n$
Unknown Max Temp: Finding $C_1$ (Overview)

1. Classify all $n$ work constraints as $UMaxW$ or $MaxW$
2. From all the $UMaxW$ constraints, pick the candidate $C_1$ (call this the $UMaxW$-winner)
3. Extend this curve to run until $d_n$
4. Rule out all $MaxW$ constraints as candidates for $C_1$. (Check if any $MaxW$ are not satisfied by $UMaxW$-winner)
Unknown Max Temp: Finding \( C_1 \) (Overview)

1. Classify all \( n \) work constraints as \( UMaxW \) or \( MaxW \)
2. From all the \( UMaxW \) constraints, pick the candidate \( C_1 \) (call this the \( UMaxW \)-winner)
3. Extend this curve to run until \( d_n \)
4. Rule out all \( MaxW \) constraints as candidates for \( C_1 \). (Check if any \( MaxW \) are not satisfied by \( UMaxW \)-winner)
5. If so, create \( MaxW \) curves to satisfy all deadlines, pick the best as your complete schedule
Unknown Max Temp

![Graph showing work completed over time](image-url)
Unknown Max Temp

- Time
- Work Completed

Leon Atkins (University of Bristol)
Unknown Max Temp
Unknown Max Temp
Now an algorithm for the general arrival model

**Algorithm Description**

- Algorithm runs at a constant speed \((\ell b T_{\text{max}})^{1/\alpha}\) until it determines this would result in a missed deadline.
  - \(\ell = (2 - (\alpha - 1) \ln(\alpha/(\alpha - 1)))^{\alpha} \leq 2\)
  - \(T_{\text{max}}\) is the optimal maximum temperature

- At this point, \(A\) runs according to the online algorithm OA (Optimal Available) until it is able to switch back.
General Online Algorithm: Proof Sketch

- Temperature optimal schedule does less than \((\ell b T_{\text{max}})^{1/\alpha}(t)\) work in any period \(> 1/b\), so we’re \(\ell \leq 2\)-competitive over these periods.
General Online Algorithm: Proof Sketch

- Temperature optimal schedule does less than \((\ell b T_{\text{max}})^{1/\alpha}(t)\) work in any period > \(1/b\), so we’re \(\ell \leq 2\)-competitive over these periods.
- Algorithm switches to OA for < \(1/b\) time.
General Online Algorithm: Proof Sketch

- Temperature optimal schedule does less than \((\ell b T_{\text{max}})^{1/\alpha}(t)\) work in any period \(> 1/b\), so we’re \(\ell \leq 2\)-competitive over these periods.
- Algorithm switches to OA for \(< 1/b\) time.
- Known that OA is \(\alpha\)-competitive for energy.
General Online Algorithm: Proof Sketch

- Temperature optimal schedule does less than \((\ell b T_{\text{max}})^{1/\alpha}(t)\) work in any period \(> 1/b\), so we’re \(\ell \leq 2\)-competitive over these periods.
- Algorithm switches to OA for \(< 1/b\) time.
- Known that OA is \(\alpha^\alpha\)-competitive for energy.
- Energy and temperature over \(1/b\) interval are related:

\[
\frac{C[S]}{e} \leq T[S] \leq \frac{e}{e-1} C[S]
\]
General Online Algorithm: Proof Sketch

- Temperature optimal schedule does less than \((\ell b T_{\text{max}})^{1/\alpha}(t)\) work in any period \(> 1/b\), so we’re \(\ell \leq 2\)-competitive over these periods.
- Algorithm switches to OA for \(< 1/b\) time.
- Known that OA is \(\alpha\)-competitive for energy.
- Energy and temperature over \(1/b\) interval are related:
  \[
  \frac{C[S]}{e} \leq T[S] \leq \frac{e}{e-1} C[S]
  \]
- At most 3 fast periods over an interval of \(1/b\), gives \((\ell + 3e\alpha)T_{\text{max}}\) energy.
General Online Algorithm: Proof Sketch

- Temperature optimal schedule does less than $(\ell \cdot b \cdot T_{\text{max}})^{1/\alpha}(t)$ work in any period $> 1/b$, so we’re $\ell \leq 2$-competitive over these periods.
- Algorithm switches to OA for $< 1/b$ time.
- Known that OA is $\alpha^\alpha$-competitive for energy.
- Energy and temperature over $1/b$ interval are related:
  \[
  \frac{C[S]}{e} \leq T[S] \leq \frac{e}{e-1} C[S]
  \]
- At most 3 fast periods over an interval of $1/b$, gives $(\ell + 3e\alpha^\alpha) T_{\text{max}}$ energy.
- Competitive ratio of $\frac{e}{e-1} (\ell + 3e\alpha^\alpha)$
In the batched release model, we have an optimal algorithm in $O(n^2)$ time.

More complicated in the general model - fast periods harm our competitive ratio.

How could we do better?

- OA isn’t the best algorithm - key property is knowing it completes jobs at their deadlines.
- If we could predict when we needed to switch to a better ‘backup’ algorithm, we could do a lot better (even with the same analysis technique).
- Would also be useful to bound how bad we can be by never cooling when there are jobs active.