Hard Real-Time Multiprocessor Scheduling

Marcus Völp
The Challenge

Die
Core
Cache

Die
Core
Cache

Bus / Crossbar

Memory
The Challenge

Die

FPU

GPU

GPU

GPU

GPU

GPU

GPU

CPU

Cache

Cache

Cache

Cache

Cache

network

Memory

Memory
Outline

Today:
- Terminology and Notations
- Anomalies and Impossibility Results
- Partitioned Scheduling
- Global Scheduling
- Optimal MP Scheduling
- Practical Matters

Next week:
- Exercise:
  - UP Resource Protocols
- Caches / Memories
- Resources
- Dynamic Tasks
- Peek and Poke into other Bleeding Edge Research

12/4/2013
Dr.-Ing. Marcus Völpe
Outline

Today:
- Terminology and Notations
- Anomalies and Impossibility Results
- Partitioned Scheduling
- Global Scheduling
- Optimal MP Scheduling
- Practical Matters

Next week:
- Exercise:
- UP Resource Protocols
- Dynamic Tasks
- Peek and Poke into other Bleeding Edge Research

[Davis '09]
R. Davis, A. Burns,
Terminology

- number of priorities
  - FTP: fixed task priority
  - FJP: fixed job priority
  - Dynamic: priorities may change during execution
- migration
  - no migration (partitioned)
  - task-level (at job boundaries)
  - job-level (arbitrary)
Terminology

- deadlines
  - implicit: \( D_i = P_i \)
  - constrained: \( D_i \leq P_i \)
  - arbitrary

- metrics
  - utility
  - density

\[
\partial_i = \frac{C_i}{\min(D_i, P_i)}
\]

\[
U_i = \frac{C_i}{P_i}
\]
Lessons Learned from UP

Fixed Task Priority:
- Rate monotonic / Deadline monotonic
- Liu-Layland Criterion  \( U_{RMS}(n) = n\sqrt{2} - 1 \leq 0.693 \)

Fixed Job Priority:
- EDF is optimal

Simultaneous release is critical instance:
- Simultaneous arrival sequence (sporadic)
- Response Time Analysis

Response times depend on set but not on the order of high-priority tasks
Simultaneous release is not a critical instance [Lauzac ’98]
Anomalies

Response time depends on priority ordering of higher priority threads
Anomalies

Response time depends on priority ordering of higher priority threads
Response time depends on priority ordering of higher prioritized threads
Sustainability

How robust is the schedule against parameter changes?

- decrease execution time of task (predictability)
  otherwise, WCET won’t work for admission
- increase minimal interrelease time
  otherwise, assuming more frequent releases is no safe approximation
- increase relative deadline
  otherwise, assuming earlier deadlines is not safe

Global FTP and Global EDF are not sustainable if no. of CPUs > 1

All pre-emptive FJP / FTP algorithms are predictable
No optimal online multiprocessor scheduling algorithm for arbitrary jobs unless all jobs have the same deadlines.

This holds even if execution times are known precisely.

No optimal online scheduling algorithm for sporadic tasksets with constrained or arbitrary deadlines.
Partitioned vs. Global

Partitioned Scheduling

- split workload into chunks of total utilization $U \leq n^{\sqrt{2}} - 1 \leq 0.693$
- reuse uniprocessor scheduler $U \leq 1$

\[
\begin{align*}
\text{CPU}_0 & \lessdot \lessdot \lessdot \\
\text{CPU}_1 & \lessdot \\
\text{CPU}_2 & \\
\end{align*}
\]
Partitioned Scheduling

- split workload into chunks of total utilization \( U \leq n^{\frac{n}{2}} - 1 \leq 0.693 \)
- reuse uniprocessor scheduler \( U \leq 1 \)
Partitioned vs. Global

Partitioned Scheduling

- split workload into chunks of total utilization
- reuse uniprocessor scheduler

\[ U \leq n^{\frac{n}{2}} - 1 \leq 0.693 \]
\[ U \leq 1 \]
Partitioned vs. Global

Global Scheduling

- just one ready queue

\[ \text{CPU}_0 \quad \text{CPU}_1 \quad \text{CPU}_2 \]
Partitioned Scheduling

[Anderson ’01]

m CPUs, m+1 Tasks

Period \( P_i = 2 \)

WCET \( C_i = 1 + \varepsilon \)
Partitioned Scheduling

[Anderson '01]

m CPUs, m+1 Tasks

Period $P_i = 2$

WCET $C_i = 1 + \varepsilon$

Optimal utilization bound for partitioned schedulers:

$$U_{opt} = \frac{m + 1}{2}$$
Global EDF

m+1 CPUs

m+1 Tasks with
Period $P_i = 1$, $U_i = \varepsilon$

1 Task with
Period $P_0 = 2$, $U_0 > (2 - \varepsilon) / 2$

Utilization bound: $U_{GEDF} = 1 + m\varepsilon$

Dhall Effect does not occur if $U_i < 41\%$
Global Scheduling

Optimal utilization bound for global schedulers: $U_{opt} = \frac{m + 1}{2}$

m CPUs, m+1 Tasks

Period $P_i = 2$

WCET $C_i = 1 + \varepsilon$
$U_{opt} = m$?

PFAIR [Baruah ‘96]

- Divide timeline into quanta $q$ of length $t$. At each quanta, allocate tasks such that the accumulated processor time is either $\lceil tu_i \rceil$ or $\lfloor tu_i \rfloor$.

Optimal **utilization bound** for PFAIR schedulers and periodic implicit deadline constraint tasksets: $U_{opt} = m$

Very high preemption and migration costs.
Design Space

- Dynamic job priority / partitioned
- Fixed job priority / partitioned
- Fixed task priority / partitioned
- Dynamic job priority / task level migration
- Fixed job priority / task level migration
- Fixed task priority / task level migration
- Dynamic job priority / job level migration
- Fixed job priority / job level migration
- Fixed task priority / job level migration

More dynamic migration
Design Space

\[ U_{opt} = m \]
\[ U_{opt} = \frac{(m+1)}{2} \]

- dyn. job prio. / job level migration
- dyn. job prio. / task level migration
- fixed job prio. / partitioned
- fixed job prio. / task level migration
- fixed task prio. / partitioned
- fixed task prio. / task level migration

A → B => A can schedule any taskset that B can schedule and more
A ↔ B => dominance is not yet known
[Lakshmanan ‘09]
Can we improve on Anderson’s utilization bound by migration some (few) jobs?

- **PDMS**: partitioned deadline monotonic scheduling
- **HPTS**: highest priority task split
- **DS**: allocate tasks according to highest density first
PDMS-HPTS-DS

\[
\partial_i = \frac{C_i}{\min(D_i, P_i)}
\]

\[
t_1 = (4,3,1) \quad u_1 = 0.25 \quad \delta_1 = \frac{1}{3} = 0.33
\]

\[
t_2 = (6,2,2) \quad u_2 = 0.33 \quad \delta_2 = 1
\]

\[
t_3 = (4,4,1) \quad u_3 = 0.25 \quad \delta_3 = \frac{1}{4} = 0.25
\]

\[
t_4 = (6,4,2) \quad u_4 = 0.33 \quad \delta_4 = \frac{1}{2} = 0.5
\]

\[
t_5 = (6,5,1) \quad u_5 = 0.16 \quad \delta_5 = \frac{1}{5} = 0.2
\]

\[
u_{\text{sum}} = 1.33 \quad => \quad u_{\text{sum}} / 2 = 0.66%
\]
$\tau_1 = (4, 3, 1)$
$\tau_2 = (6, 2, 2)$
$\tau_3 = (4, 4, 1)$
$\tau_4 = (6, 4, 2)$
$\tau_5 = (6, 5, 1)$

$u_1 = 0.25 \quad \delta_1 = 1/3 = 0.33$
$u_2 = 0.33 \quad \delta_2 = 1$
$u_3 = 0.25 \quad \delta_3 = 1/4 = 0.25$
$u_4 = 0.33 \quad \delta_4 = 1/2 = 0.5$
$u_5 = 0.16 \quad \delta_5 = 1/5 = 0.2$

$u_{\text{sum}} = 1.33 \Rightarrow u_{\text{sum}} / 2 = 0.66\%$

$\partial_i = \frac{C_i}{\min(D_i, P_i)}$
PDMS-HPTS-DS

\[ \tau_1 = (4,3,1) \]
\[ \tau_2 = (6,2,2) \]
\[ \tau_3 = (4,4,1) \]
\[ \tau_4 = (6,4,2) \]
\[ \tau_5 = (6,5,1) \]

\[ u_1 = 0.25 \quad \delta_1 = 1/3 = 0.33 \]
\[ u_2 = 0.33 \quad \delta_2 = 1 \]
\[ u_3 = 0.25 \quad \delta_3 = 1/4 = 0.25 \]
\[ u_4 = 0.33 \quad \delta_4 = 1/2 = 0.5 \]
\[ u_5 = 0.16 \quad \delta_5 = 1/5 = 0.2 \]

\[ \sum u = 1.33 \Rightarrow \sum u / 2 = 0.66\% \]

\[ \partial_i = \frac{C_i}{\min(D_i, P_i)} \]
\[
\begin{align*}
\tau_1 &= (4,3,1) & u_1 &= 0.25 & \delta_1 &= 1/3 = 0.33 \\
\tau_2 &= (6,2,2) & u_2 &= 0.33 & \delta_2 &= 1 \\
\tau_3 &= (4,4,1) & u_3 &= 0.25 & \delta_3 &= 1/4 = 0.25 \\
\tau_4 &= (6,4,2) & u_4 &= 0.33 & \delta_4 &= 1/2 = 0.5 \\
\tau_5 &= (6,5,1) & u_5 &= 0.16 & \delta_5 &= 1/5 = 0.2 \\
\end{align*}
\]

\[u_{\text{sum}} = 1.33 \implies u_{\text{sum}} / 2 = 0.66\%\]
PDMS-HPTS-DS

\(\tau_1 = (4,3,1)\)
\(\tau_2 = (6,2,2)\)
\(\tau_3 = (4,4,1)\)
\(\tau_4 = (6,4,2)\)
\(\tau_5 = (6,5,1)\)

\(u_1 = 0.25\)  \(\delta_1 = 1/3 = 0.33\)
\(u_2 = 0.33\)  \(\delta_2 = 1\)
\(u_3 = 0.25\)  \(\delta_3 = 1/4 = 0.25\)
\(u_4 = 0.33\)  \(\delta_4 = 1/2 = 0.5\)
\(u_5 = 0.16\)  \(\delta_5 = 1/5 = 0.2\)

\(u_{\text{sum}} = 1.33 \Rightarrow u_{\text{sum}} / 2 = 0.66\%\)

\(\partial_i = \frac{C_i}{\min(D_i, P_i)}\)
\[ \tau_1 = (4,3,1) \]
\[ \tau_2 = (6,2,2) \]
\[ \tau_3 = (4,4,1) \]
\[ \tau_4 = (6,4,2) \]
\[ \tau_5 = (6,5,1) \]

\[ u_1 = 0.25 \quad \delta_1 = 1/3 = 0.33 \]
\[ u_2 = 0.33 \quad \delta_2 = 1 \]
\[ u_3 = 0.25 \quad \delta_3 = 1/4 = 0.25 \]
\[ u_4 = 0.33 \quad \delta_4 = 1/2 = 0.5 \]
\[ u_5 = 0.16 \quad \delta_5 = 1/5 = 0.2 \]

\[ u_{\text{sum}} = 1.33 \Rightarrow u_{\text{sum}} / 2 = 0.66\% \]
**PDMS-HPTS-DS**

\[ t_1 = (4,3,1) \]
\[ t_2 = (6,2,2) \]
\[ t_3 = (4,4,1) \]
\[ t_4 = (6,4,2) \]
\[ t_5 = (6,5,1) \]

\[ u_1 = 0.25 \]
\[ \delta_1 = 1/3 = 0.33 \]
\[ u_2 = 0.33 \]
\[ \delta_2 = 1 \]
\[ u_3 = 0.25 \]
\[ \delta_3 = 1/4 = 0.25 \]
\[ u_4 = 0.33 \]
\[ \delta_4 = 1/2 = 0.5 \]
\[ u_5 = 0.16 \]
\[ \delta_5 = 1/5 = 0.2 \]

\[ u_{\text{sum}} = 1.33 \Rightarrow u_{\text{sum}} / 2 = 0.66\% \]

\[ \partial_i = \frac{C_i}{\min(D_i, P_i)} \]
\[ \begin{align*}
\tau_1 &= (4,3,1) \\
\tau_2 &= (6,2,2) \\
\tau_3 &= (4,4,1) \\
\tau_4 &= (6,4,2) \\
\tau_5 &= (6,5,1) \\
\end{align*} \]

\[ \begin{align*}
u_1 &= 0.25 \quad &\delta_1 &= \frac{1}{3} = 0.33 \\
u_2 &= 0.33 \quad &\delta_2 &= 1 \\
u_3 &= 0.25 \quad &\delta_3 &= \frac{1}{4} = 0.25 \\
u_4 &= 0.33 \quad &\delta_4 &= \frac{1}{2} = 0.5 \\
u_5 &= 0.16 \quad &\delta_5 &= \frac{1}{5} = 0.2 \\
u_{\text{sum}} &= 1.33 \quad \Rightarrow \quad \frac{u_{\text{sum}}}{2} = 0.66\% \\
\end{align*} \]

\[ \partial_i = \frac{C_i}{\min(D_i, P_i)} \]
\[ U_{PDMS-HPTS-DS} = m \times 69.3 \% \text{ if all tasks have a utilization } U_i < 41\% \]
impossibility result [Hong ‘88]:
no optimal MP scheduling algorithm if not all tasks have the same deadline.

- deadline partitioning
impossibility result [Hong ‘88]:
no optimal MP scheduling algorithm if not all tasks have the same deadline.

- deadline partitioning
impossibility result [Hong ‘88]:
no optimal MP scheduling algorithm if not all tasks have the same deadline.

- deadline partitioning
- zero laxity
- fluid rate curve
jobs who twine themselves around the fluid rate curve are somehow in good shape
\[ S = m - \sum_{i=0}^{n} U_i \]

- allocate work proportional to \( U_i \)
- arrange jobs in any order (no preemptions)
DP-Wrap

- allocate work proportional to $U_i$
- arrange jobs in any order (no preemptions)
- at chunk boundary, wrap around to fill other CPUs

$S = m - \sum_{i=0}^{n} U_i$
DP-Wrap

- allocate work proportional to $U_i$
- arrange jobs in any order (no preemptions)
- at chunk boundary, wrap around to fill other CPUs

$$S = m - \sum_{i=0}^{n} U_i$$
Today:
- Terminology and Notations
- Anomalies and Impossibility Results
- Partitioned Scheduling
- Global Scheduling
- Optimal MP Scheduling
- Practical Matters

Next week:
- Exercise:
  - UP Resource Protocols
- Caches / Memories
- Resources
- Dynamic Tasks
- Peek and Poke into other Bleeding Edge Research
References

- [Burns '09]

- [Colette]

- [Edmonds]

- [Levin '10]

- [Hong '88]

- [Anderson '01]

- [Lakshmanan '09]
  K. Lakshmanan, R. Rajkumar, J. Lehoczky, “Partitioned Fixed-Priority Preemptive Scheduling for Multi-Core Processors”, ECRTS, 2009
References

- [Fischer '07]

- [Dertouzos '89]

- [Dhall]

- [Baruah '06]
  S. Baruah, A. Burns, “Sustainable Scheduling Analysis”, RTSS, 2006

- [Corey '08]

- [Carpenter '04]

- [Whitehead]

- [Brandenburg '08]
References

- [Gai '03]

- [Block]

- [MSRP]

- [MPCP]

- [Lauzac '98]

- [Baruah '96]

- [Brandenburg'11]