Global Multiprocessor Scheduling

Quentin Delhaye

Université Libre de Bruxelles
ELEC-H410 Real-Time Systems

June 6th 2014

Partitioned and Global Scheduling

Partitioned

- Assign each task to a processor: Bin-Packing problem (NP-Complete)
- Forbid task migration
- Uniprocessor scheduling

Global

- Schedule jobs
- Allow migration

They are incomparable.

Notations

- e Execution time
- D Absolute deadline
- T Period
- τ Set of tasks
- m Number of processors
- U Utilisation

Global and Partitioned are incomparable

Some systems are schedulable using global FJP algorithm, but not partitioned.

	e	T	D
τ_1	2	3	2
τ_2	3	4	3
τ_3	5	12	12

- If m = 2, partitioning fails: $U(\tau_i) + U(\tau_j) > 1 \qquad \forall i \neq j$
- Global scheduling succedes if τ_3 has the lowest priority.

Global and Partitioned are incomparable

Some systems are schedulable using partitioned FJP algorithm, but not global.

	e	T	D
τ_1	2	3	2
τ_2	3	4	3
τ_3	4	12	12
τ_4	3	12	12

- Partitioned: τ_1 and τ_3 on m_1 and the rest on m_2.
- Global fails: τ_1 and τ_2 are scheduled each on one processor, τ_3 or τ_4 migrates to fill the idle units, the other one fails (no job parallelism).

EDF– Processors needed

Any Sporadic Implicit-Deadline System is schedulable using global EDF on m processors if

$$U(\tau) \leq m - (m-1)U_{max} \tag{1}$$

If we suppose $U(\tau_i) \geq U(\tau_{i+1}), \forall i \in [1, n-1]$, we have:

$$m \geq \left\lceil \frac{U(\tau) - U(\tau_1)}{1 - U(\tau_1)} \right\rceil \tag{2}$$

Poor results if $U_{max} \approx 1$, EDF$^{(k)}$ can improve that.

EDF$^{(k)}$– Scheduling Algorithm

- Set of states with the lowest utilization:

$$\tau^{(i)} = \{\tau_i, \tau_{i+1}, \ldots, \tau_n\}$$

- $\forall i < k$, give maximum priority to jobs of τ_i.
- $\forall i \geq k$, assign priorities according to EDF.

A sporadic implicit-deadline system is schedulable on m processors using EDF$^{(k)}$ if

$$m = (k-1) + \left\lceil \frac{U(\tau^{(k+1)})}{1 - U(\tau_k)} \right\rceil \tag{3}$$

As a corollary, we can minimize m:

$$m_{min}(\tau) = \min_{k\in[1,n]} \left\{ (k-1) + \left\lceil \frac{U(\tau^{(k+1)})}{1-U(\tau_k)} \right\rceil \right\} \tag{4}$$

Example

	e	D	U
τ_1	9	10	0.9
τ_2	14	19	0.737
τ_3	1	3	0.333
τ_4	2	7	0.286
τ_5	1	5	0.2
τ_6	1	10	0.1
τ	-	-	2.557

- Equation 4 is minimal for $k = 3$, leading to $m_{min}^{EDF^{(3)}} = 3$.
- Using EDF and equation 2 would have lead to $m_{min}^{EDF} = 16$.

PFAIR

- We now consider a synchronous implicit-deadline periodic tasks system and a quantum-based scheduler (*i.e.* discrete).
- Schedule function: $\mathcal{S} : \tau \times \mathbb{Z} \mapsto \{0, 1\}$.
  $\mathcal{S}(\tau_i, t) = 1$ if τ_i is scheduled in $[t, t + 1)$
- Ideal fair schedule: τ_i receives $U(\tau_i) \times t$ processor units in $[0, t)$.

Lag

Difference between the ideal and real scheduling:

$$\mathrm{lag}(\tau_i, t) = U(\tau_i) \cdot t - \sum_{\delta=0}^{t-1} \mathcal{S}(\tau_i, \delta) \tag{5}$$

Scheduling

- A schedule is PFAIR iff:

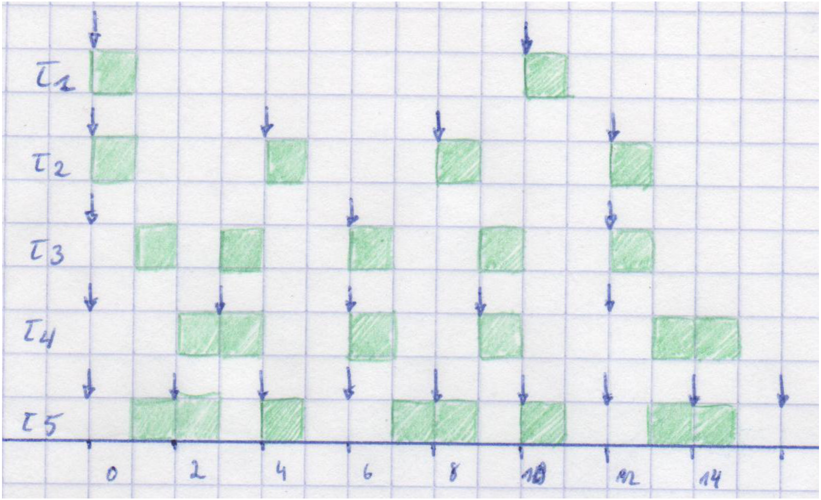
$$-1 < \text{lag}(\tau_i, t) < 1 \qquad \forall \tau_i \in \tau, t \in \mathbb{Z} \tag{6}$$

- At each time unit, schedule the m tasks having the largest positive lag.

Example

	e	D	U
τ_1	1	10	0.1
τ_2	1	4	0.25
τ_3	2	6	0.333
τ_4	1	3	0.333
τ_5	1	2	0.5
τ	-	-	1.52

lag	1	2	3	4	5	6	7
τ_1	-0.9	-0.8	-0.7	-0.6	-0.5	-0.4	-0.3
τ_2	-0.75	-0.5	-0.25	0	-0.75	-0.5	-0.25
τ_3	1/3	-1/3	0	-2/3	-1/3	0	-2/3
τ_4	1/3	2/3	0	-2/3	-1/3	0	-2/3
τ_5	0.5	0	-0.5	0	-0.5	0	0.5
	8	9	10	11	12	13	14
τ_1	-0.2	-0.1	0	-0.9	-0.8	-0.7	-0.6
τ_2	0	-0.75	-0.5	-0.25	0	-0.75	-0.5
τ_3	-1/3	0	-2/3	-1/3	0	-2/3	-1/3
τ_4	-1/3	0	-2/3	-1/3	0	1/3	0
τ_5	0	-0.5	0	-0.5	0	0.5	0



τ_1
τ_2
τ_3
τ_4
τ_5
0
2
4
6
8
10
12
14

Drawbacks

- Lots of preemptions
- Lots of migrations
- Lots of computations

Advantage PFAIR is optimal for a periodic synchronous implicit-deadline system iff

$$U(\tau) \leq m \qquad \text{and} \qquad U_{max} \leq 1 \tag{7}$$

Conclusion

- Field under extensive study.
- Implementation trickier than for uniprocessor.
- PFAIR optimal because we know the release times beforehand, but there exists no online optimal scheduler.

J. Goossens.
Operating systems II, 2013.

S. Funk J. Goossens and S. Baruah.
Priority-driven scheduling of periodic task systems on multiprocessors.
Real-Time Systems: The International Journal of Time-Critical Computing, 2003.

A. Srinivasan and S. K. Baruah.
Deadline-based scheduling of periodic task systems on multiprocessors.
In *Information Proceeding Letters*, 2002.