Experimental evaluation of optimal schedulers based on partitioned proportionate fairness

Davide Compagnin, Enrico Mezzetti and Tullio Vardanega
University of Padua - Italy

27th EUROMICRO Conference on Real-Time Systems (ECRTS)
Lund, July 9th, 2015

This project and the research leading to these results has received funding from the European Community’s Seventh Framework Programme [FP7 / 2007-2013] under grant agreement 611085

www.proxima-project.eu
Outline

- Motivation of our work
- Brief recall of RUN and QPS algorithms
- Implementation and evaluation
- Conclusions and future work
## Introduction

<table>
<thead>
<tr>
<th>RUN</th>
<th>QPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduction to UNiprocessor (RTSS-11)</td>
<td>Quasi-Partitioning Scheduling (ECRTS-14)</td>
</tr>
</tbody>
</table>

Optimal multiprocessor scheduling
Based on partitioned proportionate-fairness
Designed to reduce # of preemptions and migrations

On periodic task-sets               Also on sporadic task-sets
Motivation

RUN

QPS

Implemented\(^1\)
on top of LITMUS\(^\text{RT}\)

Confirming
moderate run-time overhead
in between that of P-EDF and G-EDF

---

\(^1\) Compagnin, D.; Mezzetti, E.; Vardanega, T., "Putting RUN into Practice: Implementation and Evaluation," (ECRTS-14)
Recall of the algorithms

**RUN**

- **Off-line phase**
  - Multiprocessor scheduling problem
  - QPS
    - Uniprocessor scheduling problems
    - decomposition

**QPS**

- **On-line phase**
  - The multiprocessor schedule is “derived” from the corresponding uniprocessor schedule
Recall of the algorithms /2

RUN

Off-line phase

Reduction tree

Processor hierarchy

Unitary processor capacity can be exceeded

External servers reserve capacity for exceeding parts on a different processor
Recall of the algorithms /3

RUN

On-line phase

QPS

ECRTS, Lund (S), 9 July 2015

D Compagnin et al.
Recall of the algorithms /3

RUN

On-line phase

QPS

ECRTS, Lund (S), 9 July 2015
D Compagnin et al.
## Notable differences

### Global scheduling
- Virtual scheduling
- Compact tree representation
- CPUs are assigned to level-0 servers
- Timers trigger budget consumption events
- Node selection is performed
- Release queue and lock

### Local scheduling + Processor synchronization
- Uniform representation of tasks and servers
- Budgets consistently updated
- Timer triggers budget consumption events
- Per-hierarchy release queue and lock

### Local scheduling
- With EDF
Global scheduling
- Virtual scheduling
- Compact tree representation
- CPUs are assigned to level-0 servers
- Timers trigger budget consumption events
- Node selection is performed
- Release queue and lock

Local scheduling
- With EDF

Notable differences

Local scheduling + Processor synchronization

P_3 notifies P_1 of the S_1's execution
Evaluation

- **Empirical evaluation** instead of simulation

- Focus on **scheduling interference**
  - Cost of scheduling primitives
  - Incurred preemptions and migrations

- Evaluation limited to **periodic task**
  - *External servers* are always “active”
  - Sporadic activations would normally have lower utilization
    - Thus reducing the number of preemptions/migrations
Experimental setup

- **LITMUS\textsuperscript{RT}** on a 16-cores AMD Opteron 6370P

- Exhaustive measurements over the two algorithms
  - Thousand of automatically generated task sets
  - Harmonic and non-harmonic, with global utilization in 50\%-100\%
  - Stressing both the off-line and the on-line phases

- Two-step experimental process
  - Preliminary empirical determination of system overheads

  
  ![Diagram](https://via.placeholder.com/150)

  - collect measurements on overheads
  - determine per-job upper bound
  - perform actual evaluation
Primal overheads and empirical bound

- Expectation was confirmed
  - QPS has lighter-weight scheduling primitives
  - And does not need Tree Update Operations (TUP)

- Empirical upper bound on the scheduling overhead
  - Based on theoretical bounds on the scheduling structures (RUN tree and QPS hierarchy)
Per-job scheduling interference

- Determined by preemptions and migrations
- In relation to reduction-tree and processor hierarchy depth
Scheduling primitives

- Maximum observed cost of core scheduling primitives
  - *Release* and *Schedule*
  - Variation under increasing system utilization
Overall per-job overhead

**Heavy tasks (utilization [0.5;0.9])**

- QPS-harmonic
- QPS-non-harmonic
- RUN-harmonic
- RUN-non-harmonic

**Medium tasks (utilization [0.1;0.5])**

- QPS-harmonic
- QPS-non-harmonic
- RUN-harmonic
- RUN-non-harmonic

- QPS is more susceptible to packing than RUN
- Lighter-weight tasks ease the partitioning problem
  - And lead to less complex scheduling structures
Conclusions and future work

- QPS benefits from **partitioned scheduling**
  - Hence improves over RUN for cost of scheduling primitives
- ... but is more susceptible to the **off-line** phase
  - QPS’s need for processor synchronization hits performance badly with higher processor hierarchies
- RUN exhibits an almost constant overhead
  - Induced by its global scheduling nature
  - Which in turn may penalize it at lower system utilization
- Future work
  - Mainly interested in evaluating how this class of algorithms may behave when the number of processing units increases
  - Considering also how different implementation may affect the algorithm scalability
Experimental evaluation of optimal schedulers based on partitioned proportionate fairness

Davide Compagnin, Enrico Mezzetti and Tullio Vardanega
University of Padua - Italy

27th EUROMICRO Conference on Real-Time Systems (ECRTS)
Lund, July 9th, 2015

This project and the research leading to these results has received funding from the European Community’s Seventh Framework Programme [FP7 / 2007-2013] under grant agreement 611085