PROXIMA

Putting RUN into practice

Implementation and evaluation

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

26th EUROMICRO Conference on Real-time Systems (ECRTS)
Madrid, July 9th, 2014
Outline

- Motivation of our work
- Brief recall of Reduction to UNiprocessor
- RUN implementation and evaluation
- Conclusions and future work
Balancing good theoretical properties and viability requirements

- Low interference and high system utilization
- Standard RTOS support and reasonable scheduling overheads
Multiprocessor scheduling state-of-the-art

Partitioned approaches
- Reduce to single-core scheduling with well-known solutions
- Bin-packing $\rightarrow$ NP-hard
- In general cannot guarantee high utilization (50% bound)

Global approaches
- Work-conserving
  - Sustain relatively higher utilizations
- Large shared scheduling structures
- Larger scheduling overheads (e.g., job migration)

Hybrid approaches
- Flexibility to attenuate the drawbacks of P- and G- approaches
- Much more difficult to implement
- May require non-standard RTOS support

- Reduction to UNiprocessor (RUN)
  - **Optimal** for implicit-deadline periodic independent tasks
  - Low interference with **few job migrations**
  - Reduces to P-EDF when a perfect partitioning exists
Recall of the RUN algorithm

- Reduction to UNiprocessor (RTSS’11)
  - Semi-partitioned algorithm
  - Optimal without resorting to proportionate fairness

- Reduction principles
  - Duality
    \[ \tau_i(T_i, u_i) \overset{\text{dual}}{\longleftrightarrow} \tau_i^*(T_i, 1 - u_i) \]
    \[ \text{SCHED}(\mathcal{T}_n, U, m) \equiv \text{SCHED}(\mathcal{T}_n^*, n - U, n - m) \]
  - Fixed-rate tasks and servers
    \[ \tau_i \overset{\text{def}}{=} (\mu_i, D_i) \Rightarrow S(\sum_{\tau_i \in S} \mu_i, \bigcup_{\tau_i \in S} D_i) \]

- Scheduling decision taken on reduction tree

- Doubts
  - Can we implement it on standard RTOS support?
  - Does handling the reduction tree incur unacceptably large overhead?
Scheduling on RUN

- Off-line: *reduction tree*
  - Dual + Pack
- On-line: *EDF rules*
  - Virtual scheduling of servers
    - Virtual jobs
    - Proportionate execution
RUN implementation

- Successfully implemented
  - On top of LITMUS\textsuperscript{RT} Linux test-bed (UNC)
  - Thus relying on standard RTOS support

- Main implementation choices and challenges
  - \textit{Scheduling on the reduction tree}
    - How to organize the data structure
    - How to perform virtual scheduling and trigger tree updates
    - Intrinsic influence of the packing policy
  - \textit{Mixing global and local scheduling}
    - Global release event queue vs. local \textit{level-0} ready queue
    - Handling simultaneous scheduling events
      - Job release, budget exhaustion (possibly from different sub-trees)
  - \textit{Meeting the full-utilization requirement}
    - Variability of tasks’ WCET and lower utilization
Empirical evaluation instead of simulation-based

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

Compared RUN against P-EDF and G-EDF
- RUN shares something in common with both
- Had a preliminary evaluation on Pfair (S-PD² in LITMUS\textsuperscript{RT})
  - Inferior performance in terms of preemptions and migrations
Experimental setup

- **LITMUS**\textsuperscript{RT} on an 8-cores AMD Opteron\textsuperscript{TM} 2356

- Collected measurements for the three algorithms
  - Hundreds of automatically generated task sets
  - Harmonic and non-harmonic, with global utilization in 50%-100%
  - Representative of small up to large tasks

- Two-step process
  - Preliminary empirical determination of overheads

### Diagram
- Collect measurements on overheads
- Determine per-job upper bound
- Perform actual evaluation
- Expectations confirmed
  - P-EDF needs lighter-weight scheduling primitives
- Tree update (TUP) triggered upon
  - Budget exhaustion event
  - Job release $\rightarrow$ REL includes TUP
- Empirical upper bound on RUN scheduling overhead
  - $OH_{RUN}^{Job} = REL + \widehat{SCHED} + CLK + k \times (TUP + \widehat{SCHED} + max(PRE, MIG))$
  - $k = \lceil (3p + 1)/2 \rceil$ and $\widehat{SCHED} = SCHED + CSW + LAT$.  

![Graph showing time in microseconds for REL, SCHED, CSW, CLK, LAT, and TUP categories for different scheduling methods: RUN, GEDF, PEDF. The graph indicates variations in time across different scheduling methods and categories.]
Task sets exhibiting at least one miss
RUN suffered **no misses**
  - Optimality and tailored overhead
Kernel interference

Observing average preemptions and migrations

![Graph showing average preemptions and migrations with different utilisation caps.](image)

![Bar chart showing the ratio of different utilisation levels.](image)
Scheduling cost

- Average cost of core scheduling primitives

![Graph showing scheduling cost](image)
Per-job scheduling overhead

Harmonic task set

Non-harmonic task set
Conclusions and future work

- Good news on RUN from this evaluation
  - It can be practically and efficiently implemented
  - It may exhibit very modest kernel overhead
    - Acceptable even on non-harmonic task sets
  - It causes a tiny amount of migrations
    - Hence low inter-task interference

- Essential improvements
  - Handle sporadic task sets
  - Allow sharing of logical resources

- Further work
  - Better understanding of the role of packing policies
    - Affecting the reduction tree, hence preemptions/migrations
  - Further comparisons against other optimal solutions
    - High interest in Quasi-Partitioned Scheduling (QPS)
Putting RUN into practice

Implementation and evaluation

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

26th EUROMICRO Conference on Real-time Systems (ECRTS)
Madrid, July 9th, 2014

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
Evaluation against S-PD^2

**Observed preemptions and migrations**

**Per-job kernel overhead**

09/07/2014
Reduction tree data structure