Fixed Priority until Zero Laxity (FPZL)
Schedulability Analysis

Robert Davis and Alan Burns

Real-Time Systems Research Group, University of York
Research scope

- Homogeneous Multiprocessor Real-Time Systems
- Global scheduling
  - Single global run-queue
  - Pre-emption and migration
- Based on fixed task-priority scheduling
  - All jobs of a task have the same fixed priority
- Add minimally dynamic priorities
  - Promote the priority of any job that would otherwise inevitably miss its deadline (zero-laxity)
Motivation

- **Improve upon the effectiveness of global FP scheduling**
  - Dynamic priority algorithms
    - Potentially much more effective than fixed task-priority algorithms in terms of the tasksets that can be scheduled
    - But can have significantly larger overheads e.g. theoretically optimal algorithms with \( n - 1 \) context switches per job release

- **Avoid significant increase in complexity or number of context switches**
  - FPZL: Zero-Laxity rule applied to global FP scheduling
    - When remaining execution time equals time to deadline, task must run or the deadline will be missed - so priority promoted
    - At most **one** change in priority per job release
    - At most **two** pre-emptions per job release
Outline

- System model, terminology, and definitions
- Recap on schedulability tests for global FP scheduling
- Schedulability tests for FPZL
- Improving the tests by bounding execution time in the zero-laxity state
- Empirical results
  - Schedulability test performance
  - Algorithm performance (simulation)
- Comparison with previous work on RMZL
- Summary and conclusions
System model

- **Multiprocessor system**
  - \(m\) identical processors
  - FPZL scheduling (global FP pre-emptive scheduling + priority promotion at zero-laxity)
  - Migration is permitted, but a job can only execute on one processor at a time

- **Sporadic task model**
  - Static set of \(n\) tasks \(\tau_i\) with priorities \(1..n\)
  - Bounded worst-case execution time \(C_i\)
  - Sporadic/periodic arrivals: minimum inter-arrival time \(T_i\)
  - Relative deadline \(D_i\) (Constrained deadlines \(\leq T_i\))
  - Independent
Global FP: Sufficient schedulability tests

- **Fundamental approach (Baker [2])**
  - Problem window in which deadline is missed (e.g. $D_k$)
  - Necessary condition for deadline miss: $m$ processors all occupied for more than $D_k - C_k$
  - Derive upper bound on interference $I^{UB}$ from other tasks
  - Negate the un-schedulability condition to form a sufficient schedulability test for task $\tau_k$
Deadline analysis for global FP

- **Worst-case scenario for task** $\tau_k$
  
  **(Davis & Burns [16], Guan et al. [20])**
  
  - At most $(m - 1)$ higher priority tasks contribute *carry-in* interference

  \[
  I_i^D(L, C_k) = \min(W_i^D(L), L - C_k + 1)
  \]

  \[
  W_i^D(L) = N_i^D(L)C_i \+ \min(C_i, L + D_i - C_i - N_i^D(L)T_i)
  \]

  \[
  N_i^D(L) = \left\lfloor \frac{(L + D_i - C_i)}{T_i} \right\rfloor
  \]

  - Other tasks contribute no carry-in interference

  \[
  I_i^{NC}(L, C_k) = \min(W_i^{NC}(L), L - C_k + 1)
  \]

  \[
  W_i^{NC}(L) = N_i^{NC}(L)C_i \+ \min(C_i, L - N_i^{NC}(L)T_i)
  \]

  \[
  N_i^{NC}(L) = \left\lfloor \frac{L}{T_i} \right\rfloor
  \]
Deadline analysis for global FP

- **Polynomial time test: Deadline Analysis ("DA-LC test") (Davis & Burns [16] based on Bertogna et al. [9], Guan et al [20])**
  - Difference between carry-in and no carry-in interference
    \[
    I^\text{DIFF-D}_i (L, C_k) = I^D_i (L, C_k) - I^{NC}_i (L, C_k)
    \]
  - Include extra interference from \((m - 1)\) tasks with largest difference between carry-in and no carry-in interference
    \[
    D_k \geq C_k + \frac{1}{m} \left( \sum_{\forall i \in hp(k)} I^{NC}_i (D_k, C_k) + \sum_{i \in MD(k,m-1)} I^\text{DIFF-D}_i (D_k, C_k) \right)
    \]
  - Schedulability test for each task \(\tau_k\)
Response Time analysis for global FP

- **Worst-case scenario for task** $\tau_k$ (Guan et al. [20])
  - At most $(m - 1)$ tasks contribute *carry-in* interference

  $$ I_i^R (L, C_k) = \min(W_i^R (L), L - C_k + 1) $$
  $$ W_i^R (L) = N_i^R (L)C_i + \min(C_i, L + R_i^{UB} - C_i - N_i^R (L)T_i) $$
  $$ N_i^R (L) = \left\lceil \frac{(L + R_i^{UB} - C_i)}{T_i} \right\rceil $$

- Others contribute no carry-in interference (as before)

  $$ I_i^{NC} (L, C_k) = \min(W_i^{NC} (L), L - C_k + 1) $$
  $$ W_i^{NC} (L) = N_i^{NC} (L)C_i + \min(C_i, L - N_i^{NC} (L)T_i) $$
  $$ N_i^{NC} (L) = \left\lceil \frac{L}{T_i} \right\rceil $$
Response Time analysis for global FP

- **Pseudo-polynomial time test: Response Time Analysis ("RTA-LC test")** (Guan et al [20], based on Bertogna & Cirinei [8])

  - Difference between carry-in and no carry-in interference

\[ I_i^{DIFF-R}(L, C_k) = I_i^R(L, C_k) - I_i^{NC}(L, C_k) \]

  - Include extra interference from \((m - 1)\) tasks with largest difference between carry-in and no carry-in interference

\[ R_k^{UB} \leftarrow C_k + \frac{1}{m} \left( \sum_{i \in hp(k)} I_i^{NC}(R_k^{UB}, C_k) + \sum_{i \in MR(k, m-1)} I_i^{DIFF-R}(R_k^{UB}, C_k) \right) \]

Recall dependency on response time upper bounds of higher priority tasks – need to evaluate schedulability in priority order – highest priority first
FPZL Schedulability analysis

- **Differences w.r.t. analysis for global FP**
  - Up to $m$ tasks may be deemed unschedulable but still meet their deadlines due to the zero-laxity rule
  - Tasks executing in the zero-laxity state have an impact on the schedulability of other tasks (assume $Z_{UB}^j = C_j$)

\[
I_j^Z(L, C_k) = \min(W_j^Z(L), L - C_k + 1)
\]
\[
W_j^Z(L) = N_j^Z(L)Z_{UB}^j + \min(Z_{UB}^j, L - N_j^Z(L)T_j)
\]
\[
N_j^Z(L) = \left\lfloor \frac{L}{T_j} \right\rfloor
\]

- Zero-laxity execution immediately proceeds the deadline
  - Equations similar to “no carry-in” case
  - Need only consider lower priority zero-laxity tasks (no increase in interference from higher priority zero-laxity tasks – already of higher priority)
FPZL Schedulability Analysis

- **Deadline Analysis for FPZL (DA-LC test)**

\[
D_k \geq C_k + \frac{1}{m} \left( \sum_{i \in hp(k)} I_i^{NC}(D_k, C_k) + \sum_{i \in MD(k, m)\setminus hp(k)} I_i^{DIFF-D}(D_k, C_k) + \sum_{j \in lpzlj(k)} I_j^L(D_k, C_k) \right)
\]

- If inequality holds, task is schedulable without priority promotion, otherwise it is a zero-laxity task
- At most \( m \) zero-laxity tasks in a schedulable system
- Dominates equivalent test for global FP
- Schedulability needs to be checked lowest priority first to identify which tasks are zero-laxity tasks
- Polynomial time \( O(n^2) \) test of taskset schedulability
**FPZL Schedulability Analysis**

- **Response Time Analysis for FPZL (RTA-LC test)**
  
  $$R_k^{UB} \leftarrow C_k + \left[ \frac{1}{m} \left( \sum_{i \in hp(k)} I_i^{NC} (R_k^{UB}, C_k) + \sum_{i \in MR(k,m-1)} I_i^{DIFF-R} (R_k^{UB}, C_k) + \sum_{j \in pzl(k)} I_j^{Z} (R_k^{UB}, C_k) \right) \right]$$

- **As before:**
  - If $R_k^{UB} \leq D_k$, task is schedulable without priority promotion, otherwise it is a zero-laxity task
  - At most $m$ zero-laxity tasks in a schedulable system
  - Dominates equivalent test for global FP

- **Problem:**
  - Response time upper bound depends on response times of higher priority tasks and the zero-laxity status of lower priority tasks
FPZL Schedulability Analysis

- **RTA Solution**
  - Response time (and hence zero-laxity status) is monotonically non-decreasing in the response times of higher priority tasks and the zero-laxity status / zero-laxity execution times of lower priority tasks
  - Whenever a zero-laxity task is found – must repeat response time calculations

```plaintext
1. countZL = 0
2. Initialize all $R_k^{UB} = C_k$ and $Z_k^{UB} = 0$
3. repeat = true
4. while (repeat) {
5.     repeat = false
6.     for (each priority level $k$, highest first) {
7.         Determine $R_k^{UB}$ according to (18)
8.         if ($R_k^{UB} > D_k$) {
9.             $R_k^{UB} = D_k$
10.            Compute $Z_k^{UB}$
11.            if ($	au_k$ not marked as a ZL task) {
12.                mark $	au_k$ as a ZL task
13.                repeat = true
14.                countZL = countZL + 1
15.            if (countZL > m) {
16.                repeat = false
17.                break (exit for loop)
18.            }
19.        }
20.     }
21. 
22. [if ($R_k^{UB}$ or $Z_k^{UB}$ differ from prev. values) repeat = true]
23. 
24. if (countZL > m)
25.     return unschedulable
26. else
27.     return schedulable
```
Bounding zero-laxity execution time

- **DC-Sustainability**
  - A schedulability test is *DC-Sustainable* provided that
    - Any task that is *schedulable* according to the test with parameters \((D, C)\) remains schedulable when \(D\) and \(C\) are reduced by the same amount \(x\) to \((D-x, C-x)\)
    - Any task that is *unschedulable* according to the test with parameters \((D, C)\) remains unschedulable when \(D\) and \(C\) are increased by the same amount to \((D+x, C+x)\)
  - Both FPZL schedulability tests (DA-LC and RTA-LC) are DC-Sustainable
    - Proofs in the paper
Bounding zero-laxity execution time

- **Execution time in the zero-laxity state**
  - DC-Sustainability of the schedulability tests means
    - For each zero-laxity task, we can use a binary search to find the min value of $x$ such that the task is schedulable with parameters $(D-x, C-x)$ without priority promotion
    - $x$ is then an upper bound on the execution time in the zero-laxity state
  - Response Time Analysis
    - Iterative calculation - also need to re-start calculations whenever the response times or execution times in the zero-laxity state change
Empirical Investigation

- **Taskset parameters**
  - Task utilisations generated via UUnifast-Discard
  - Task periods chosen from a log-uniform distribution with a range from min to max period of 1000 (e.g. 1ms to 1 sec)
  - Execution times set from task utilisation and period values
  - Task deadlines chosen from a uniform distribution between execution time and period
  - Total utilisation varied from 0.025\textit{m} to 0.975\textit{m} in steps of 0.025\textit{m}
  - 1000 tasksets generated for each total utilisation level
  - Graphs plot the percentage of tasksets that are schedulable according to each schedulability test against total utilisation
Empirical Investigation

- **Sufficient schedulability tests**
  - Global FP: (DA-LC test and DMPO)
  - Global FP: (DA-LC test and OPA)
  - Global EDF: (EDF-RTA test)
  - EDZL: (EDZL-I test)
  - FPZL: (DA-LC test and OPA)

- **LOAD* necessary infeasibility test**

- **Simulations**
  - Global FP (DMPO, DCMPO)
  - FPZL (DCMPO)
  - EDF
  - EDZL
Empirical results: 8 Processors 40 tasks $D \leq T$

Percentage of tasksets schedulable vs. Utilisation

- LOAD* infeasible
- FPZL Sim (DCMPO)
- EDZL Sim
- FP Sim (DCMPO)
- EDF Sim
- FP Sim (DMPO)
- FPZL-LZ DA-LC (OPA)
- FP DA-LC (OPA)
- EDZL (I)
- FP DA-LC (DMPO)
- EDF (RTA)
Empirical results: 4 Processors 20 tasks $D \leq T$
Empirical results:
2 Processors
10 tasks D≤T
RMZL and FPZL

- Related research on RMZL
  - Originally published in Japanese by Shinpei Kato
  - Now available as a technical report in English
  - RMZL is the same zero-laxity rule applied to global FP scheduling for the “Rate Monotonic” case ($D=T$)
    - Algorithm is the same as FPZL
    - Analysis is simpler but only applicable to the implicit deadline case with RM priority order
    - RMZL analysis assumes every lower priority task can be a zero-laxity task
    - Unfortunately this leads to declining schedulability test performance with an increasing number of tasks
  - FPZL schedulability test dominates the equivalent RMZL test
Summary and conclusions

- **Motivation**
  - To improve on current state-of-the-art in terms of techniques that enable the efficient use of processing capacity in hard real-time systems based on multiprocessors.
  - Aimed to improve upon the effectiveness of global FP scheduling without introducing significant additional overheads (e.g. large numbers of context switches)
  - Therefore investigated a minimally dynamic priority algorithm FPZL
Summary and conclusions

- **Contribution**
  - Introduced polynomial and pseudo-polynomial time schedulability tests (Deadline Analysis and Response Time Analysis) for FPZL
  - Improved these tests via calculation of the maximum execution time in the zero-laxity state
  - Test dominate the equivalent tests for global FP
  - Empirical results show that FPZL schedulability tests make a useful improvement on those for global FP particularly in the implicit deadline case
  - Simulation results show that FPZL (and EDZL) are highly effective – still a large gap between simulation and schedulability analysis potentially due to pessimism in the analysis