Dynamic Adaptation of DAGs with Uncertain Execution Times in Heterogeneous Computing Systems

Qin Zheng

Institute of High Performance Computing

Agency for Science, Technology and Research (A*Star), Singapore

Background on DAG Scheduling

- Static algorithms

  e.g., ***HEFT***, which sets the priority of each task with the ***upward rank*** value, which is the length of the critical path from this task (inclusive) to the exit task based on mean computation and communication costs.

- Dynamic algorithms

  e.g., ***Just-In-Time (JIT)***, which schedules a task only when it is ready to start, that is, after all its parents complete.

Performance of Static Algorithms When Task Execution Times Are Uncertain

- In many cases, in a DAG, the actual execution time of a task is different from the expected one [3].
- This may greatly affect the performance of static algorithms [4].
- In [5], the authors point out that considering their average case behavior is not safe, because execution overruns may cause other tasks to miss their deadlines.
- In hard real-time feasibility analysis, it is assumed the worst case execution time as the task computation time.

Our Work

- We study **how to improve the performance of static algorithms when tasks execution times in a DAG are uncertain**.

- The initial schedule, based on the estimated task execution times, is revised or modified (**adapted**) during runtime as necessary when tasks overrun or underrun.

- The objective is to minimize the response time and the number of adaptations.

When Should We Adapt a Task Schedule?

- When each of its predecessor (or only parent) overruns? not scalable and causing overhead. Also, not necessary.

- *We consider adapting a task* ***when all its parents start running***. At this time, except its running parents, this task will not be influenced by its other predecessors. This enables us to focus on uncertainties of its running parents only and reduces the overhead.

Whether This Task Schedule Should Be Adapted?

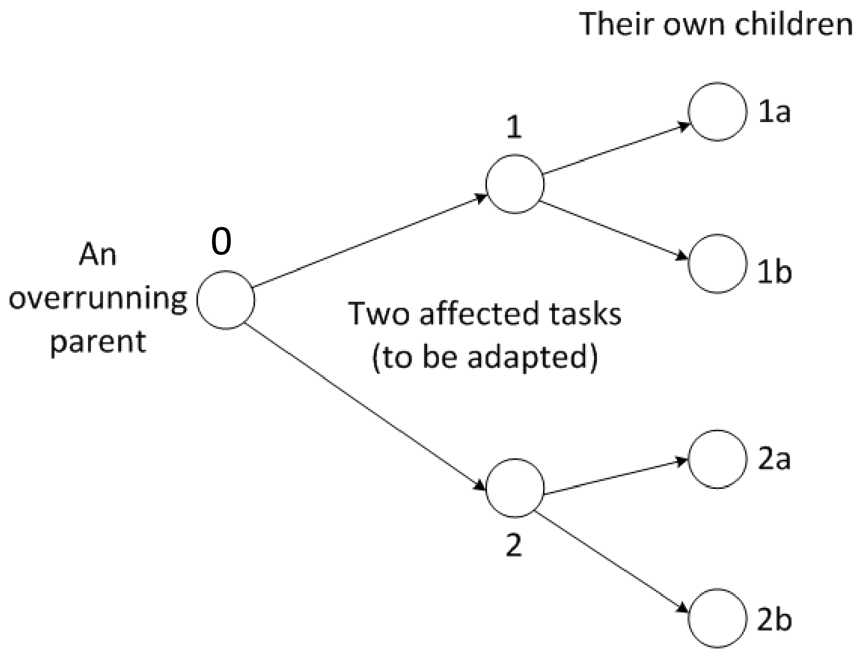
- At this time, its parents either (1) *have completed* or (2) have *actually started and are still running*. Therefore, we can have a better estimation of the expected earliest start time (***e-est***) of this task based on
  - the actual completion times (***act***) of its parents for (1)
  - the actual starting times (***ast***) + the expected running times of its parents for (2)

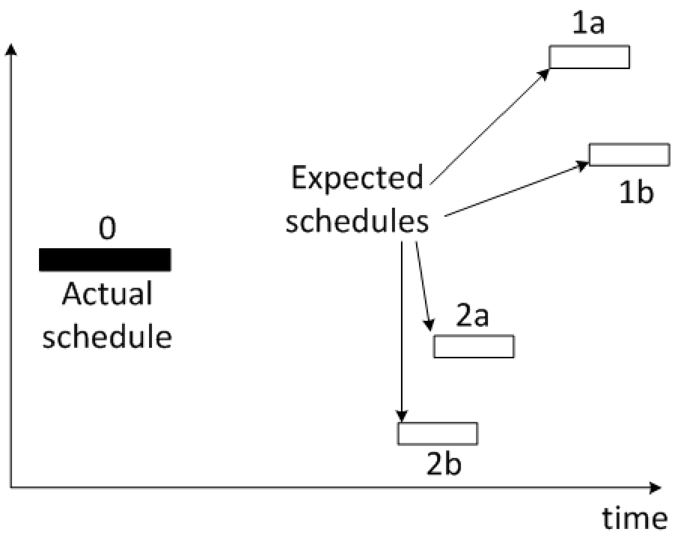
  e-est of this task is the maximum of the above among its parents.

- The task is ***invalid*** if its ***scheduled start time*** < the ***act*** of any parent.
- The task ***is likely to be invalid*** if its ***scheduled start time*** < its ***e-est***.

Early release

Which Task Schedule to Adapt First if There Are More Than One of Them?





What to Do When An Underrun Occurs?

- ***We consider adapting a task when its parent underruns so as to reduce the response time for the DAG.***
- This task may be able to start on earlier slots because of two reasons.
  - Firstly, its *est* may become earlier.
  - Secondly, there may exist an earlier schedule due to the early release of that parent.
- A task is considered only after all its parents have started.
- Among the children, which task schedule to adapt? Critical task

The Maximum Number of Tasks Adapted

- For task adaptation due to overruns, if we only adapt task schedules that become invalid, then in the worst case it equals to the number of invalid tasks in the DAG.

- For task adaptation due to underruns, in the worst case, it equals to the number of underruning tasks in the DAG.

The Case with No Initial Static Schedule – Dynamic Planning and Adaptation

- During runtime, schedules are planned for tasks when all their parents have started.

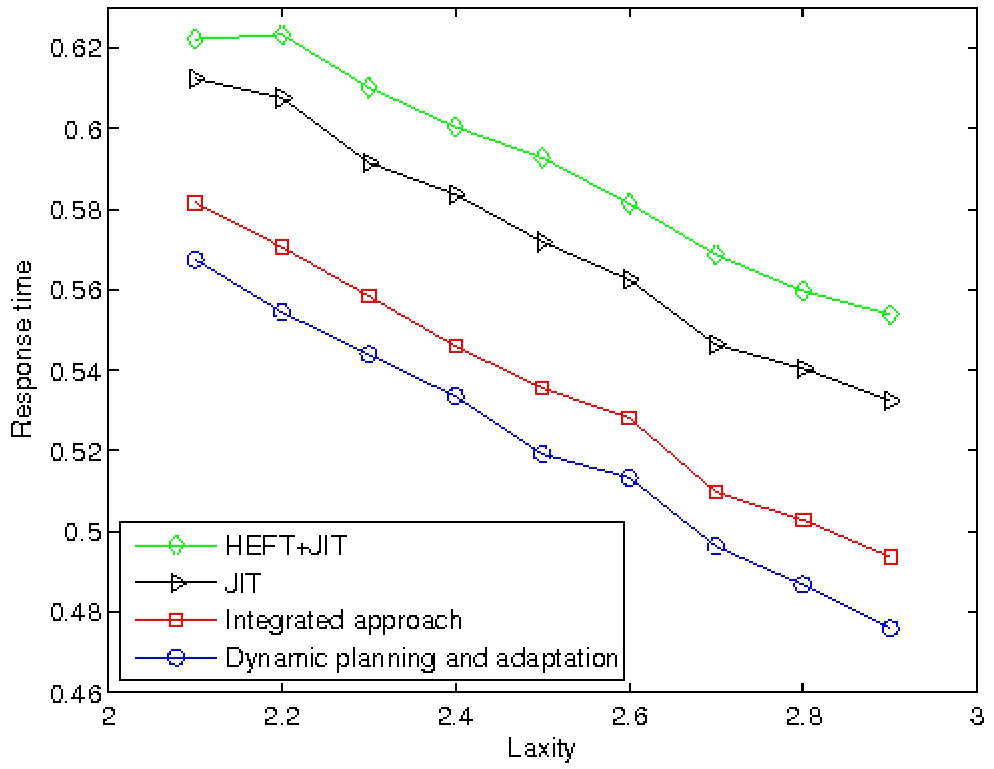
- Which task to plan first if there are more than one of them? Base on their potential impact on the response time, i.e., ***e-est*** (at this time) + ***upward rank*** value

- When a task underruns, which child schedule should be adapted?
  - The critical path may change during runtime, which needs to be taken into consideration in order to optimize the response time.
  - The child on the current critical path first, then the child with the largest potential impact on the response time.

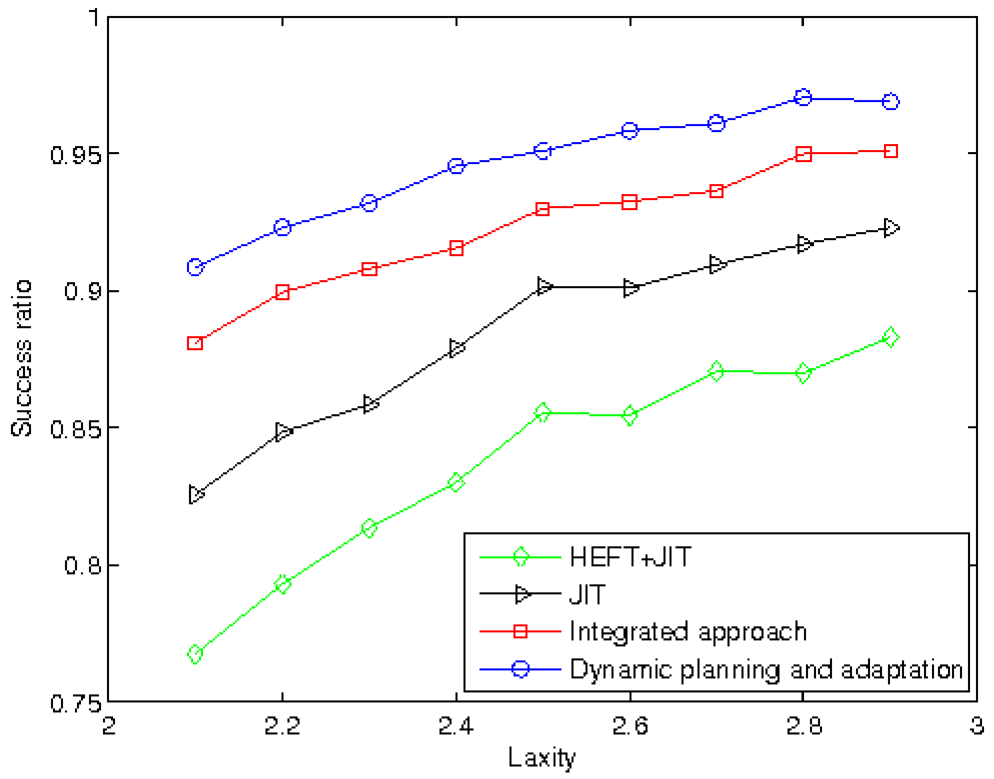
Performance Study

- Event-driven simulations using traces collected from the Hilbert system at A*STAR Computational Resource Center (ACRC) [24].
- The Hilbert system consists of 128 quad processor Opteron with aggregated 2 TeraByte memory space.
- Above 200,000 requests were processed over the period between Jan 2008 and April 2008.

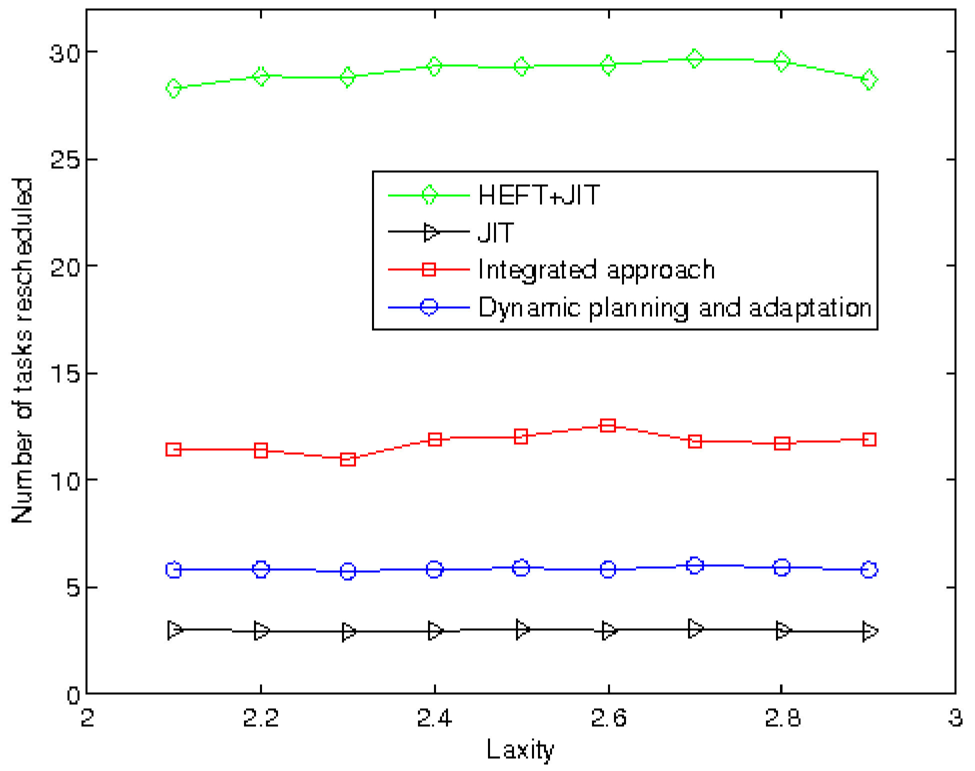
Response Time



Success Ratio



The Average Number of Tasks Rescheduled



Conclusion

- In this paper, we presented efficient heuristics to handle task overruns and underruns for DAGs during runtime.
- The objective is to minimize the response time and the number of tasks adapted.
- We considered both cases with and without an initial static schedule.
- The proposed heuristics can be used together with existing static and dynamic DAG scheduling algorithms to deal with uncertainties and improve the response time.

Q&A