Incorporating Deadline-Based Scheduling in Tasking Programming Model for Extreme-Scale Parallel Computing

CS@UH

Albert M. K. Cheng and Panruo Wu

University of Houston, Texas, USA

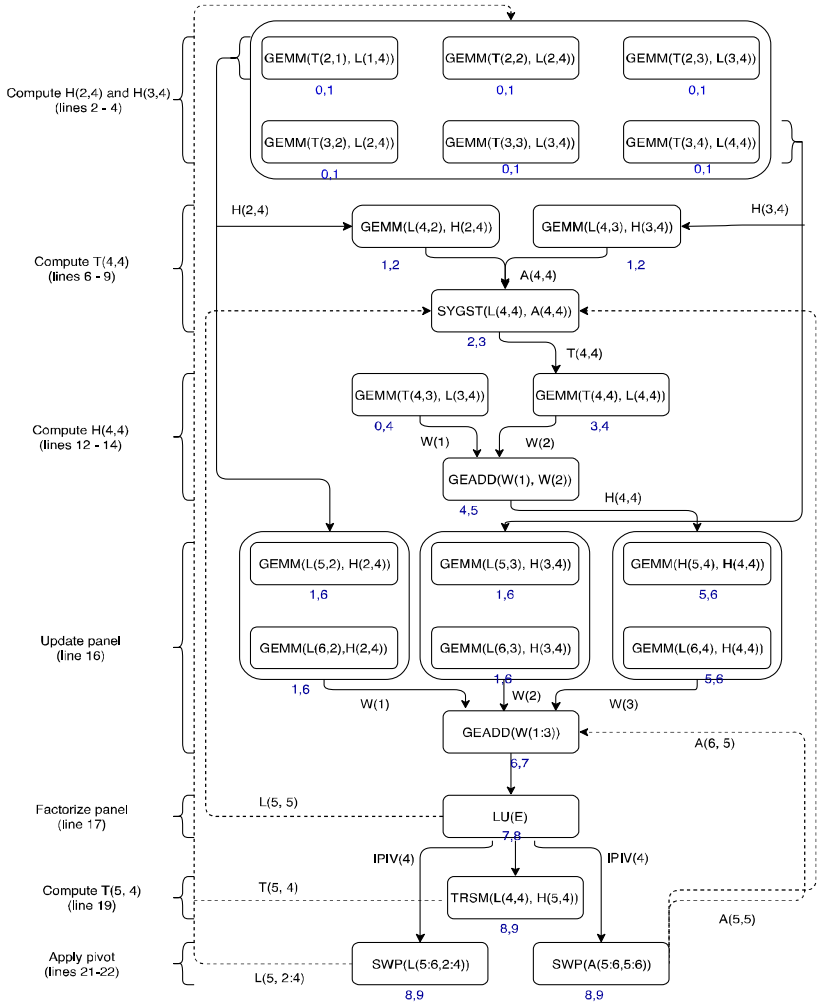
- Introduction
- Deadline-based Scheduling in Tasking Programming Model
- Virtual Deadline and its Derivation from Task Dependencies
- Use of Performance Model for More Truthful Deadlines
- On-line Profiling of Tasks

Introduction

CS@UH

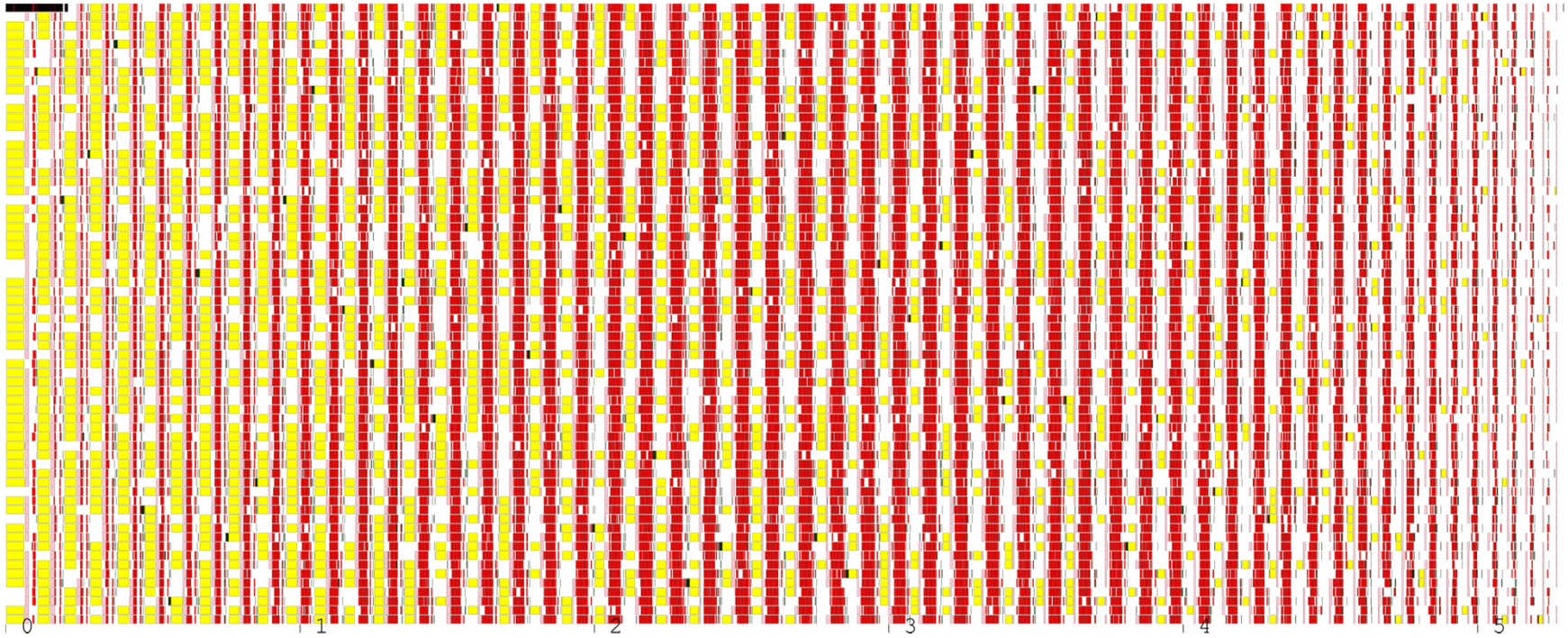
- Processing and analyzing big data sets updated in real time (such as severe weather prediction and particle-physics experiments) require the computational power of extreme-scale high-performance computing (HPC) systems.
- Current scheduling strategies for massive task/thread sets on extreme-scale systems rely on improving centralized, distributed, and parallel scheduling algorithms as well as virtualization developed for HPC systems which aim to reduce the makespan and balance the load amongst computing nodes.
- However, these HPC schedulers provide no guarantees on meeting timing constraints such as deadlines that are required in an increasing number of these real-time science workflows.
- Our approach brings real-time scheduling to address the demands of real-time science workloads: introduces deadline-based scheduling in the tasking programming model.

Figure 1. Task graph of LDLt Block Aasen algorithm [Yamazaki et al], with added annotations for an example set of virtual deadlines



- Compared to fork-join and bulk synchronous parallel (BSP) paradigms, tasking has gained popularity as a new paradigm in HPC.
- The tasking paradigm decomposes a problem into independent tasks that can be executed serially with dependencies among tasks. The example here is the LDLt decomposition in numerical linear algebra.
- The idea of our project here is to introduce a virtual deadline informed by the dependency between tasks for better schedules.
- Virtual deadline and its derivation from task dependencies: Unlike real-time systems for which there are external real deadline requirements, the virtual deadlines for the tasks in a Asynchronous Many-Tasks (AMT) runtime system are internal and for influencing the scheduling only.
- Virtual time: use relative task execution time as the unit for the virtual time and deadline.

Figure 2. Visualization of execution trace of LDLt Block Aasen routine (PLASMA 17 [Dongarra et al], DSYTRF) on Intel Xeon Phi, 68 cores, 1.4GHz, MCDRAM



- This shows an execution trace of all the tasks in a particular run on a size 20,000 matrix.
- Colored chunks indicate different computation and communication tasks, and white gaps between them indicate idling processor time, part of which can be reduced by a better scheduling strategy.
- An example with a fairly regular and static task characteristics.
- For an irregular and dynamic application, scheduling will be more challenging.

Project Summary

CS@UH

- Initial effort toward enhancing extreme-scale high-performance computing by incorporating deadline-based scheduling in the tasking programming model to meet emerging real-time and near-real-time applications.
- **Two new abstractions in this in-progress work:**
- (1) Break the batch job abstraction in the scientific computing workflow, where a job can reside in a queue in an indeterminate amount of time; the new abstraction of scientific workflow requires co-existence of real-time and non-real-time batch jobs to facilitate feedback-guided experimentation and urgent computational jobs.
- New abstraction will be validated by producing schedules that meet deadlines without significantly slowing down batch jobs.
- (2) Add a deadline property to the tasking programming model, thus breaking the old model.
- Newly added deadline will be easier to determine by the programmer compared to more abstract properties such as priority, and will help better schedule tasks to reduce task migration and execution time/energy consumption.
- Validate by using a variety of real-time scheduling techniques in the tasking programming and execution models and by the resulting superior scheduling of tasks in benchmarks and real-world HPC applications.