Dynamic Load Balancing Of Large-Scale Simulations

Jeff Amelang¹, Nathan Barton²


Motivation and Aim

As we push towards exascale computing, load balancing (or the task of keeping all of the processors working most of the time) becomes an increasingly difficult challenge. Even with an exascale machine, an improperly-balanced simulation may only achieve a fraction of the capacity of the computer.

Traditionally, load balancing for Single Program Multiple Data (SPMD) simulations has been attempted by static or dynamic domain decomposition techniques, which work fairly well for modestly-sized simulations but can perform poorly as the number of cores in the calculation reaches the order of 100,000.

Recent work at Lawrence Livermore National Laboratory shows the efficacy of using the Multiple Program Multiple Data (MPMD) parallel style in the co-op/Babel/Adaptive Sampling project, achieving a speedup of on the order of 50-100 with respect to wall clock time. Unfortunately, this technique fell short of its potential, hampered by a high degree of load imbalance.

The goal of this project is to decrease simulation wall clock time by better utilizing the available computation cores.

Problem Setup

We will be performing multi-scale embedded polycrystal plasticity calculations for the expansion of a metal ring under high pressure. In the coarse-scale material model, the visco-plastic part of the material response is based on parameters determined from polycrystal-scale fine-scale calculations.

Computationally, we will be performing this calculation on a large, distributed memory machine such as terra at LLNL. Each polycrystal finite-scale evaluation is computationally expensive and fine scale evaluations dominate (>99%) the computation time.

Adaptive Sampling

Adaptive Sampling is a technique developed at Lawrence Livermore to reduce wall clock time by caching the results of fine scale calculations and using them to interpolate later requests. Two key ingredients are needed:

1. Some way to efficiently store and retrieve the previous results – Metric Tree Database
2. An interpolation method – Kriging is used because it includes an error estimate

Pseudocode for the adaptive sampling might be:

```java
AssModule: RequesstRecoverdInputs
  krig a response from the previous calculations
  if the kriged response is low enough return it
  else do a fine scale evaluation (long)
```

Adaptive Sampling does a phenomenal job of reducing the number of fine scale evaluations required, but it unfortunately creates a computational load with high spatial and temporal variation.

High spatial variation: only those quadrature points that are not krig-able will have to be evaluated, and it’s very difficult to predict which quadrature points will be the sources of the evaluations. This makes domain decomposition techniques very difficult to apply.

High temporal variation: with each new “type” of deformation in the simulation, the database has to be populated with “new” values and the number of evaluations can change by orders of magnitude over only a few timesteps.

Computational Load Induced by Adaptive Sampling

Load Balancing through MPMD

For computational loads with high spatial variation, domain decomposition techniques in SPMD programs produce the style of work-time graph shown on the left. By changing to MPMD, the load can be balanced and the total runtime can be reduced (right).

Solution: Multi Farm Idea

To achieve 100% utilization at the highest efficiency, we stack multiple farms with varying levels of parallelism on top of the same cores. When a group of requests arrive, we assign requests to the farm with the largest number of servers that will keep the cores 100% utilized.

Results

The Multi Farm Servers successfully keep the fine-scale cores fully utilized and decrease the runtime by a factor of slightly more than 2 (for this problem).

Acknowledgements

This material is based upon work supported by the Department of Energy National Nuclear Security Administration under Award Number DE-FG02-08NA28613.