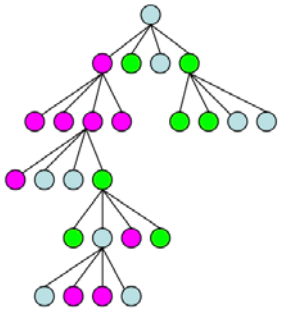
Locality Awareness in Task-Parallel Computation on Shared-Memory Systems

Department of Computer Science **University of North Carolina at Chapel Hill** **Spring 2014**

The Challenge

Within a parallel computer, multiple compute nodes are connected through a high speed network. Compute nodes have evolved rapidly in recent years to provide high-bandwidth and high-capacity shared memory systems serving multiple, multi-core, processors within the node. Heterogeneous nodes further expand performance by integrating computational accelerators like GPUs. Shared-memory compute nodes now offer the greatest performance opportunities for novel parallel algorithms and programming models.

Largely based on ideas from the Cilk language at MIT in the 1990's, the parallel task model enables the expression of parallel algorithms as a series of tasks, each of which may in turn spawn other tasks, and wait for these tasks to complete. For example, a parallel divide-and-conquer algorithm is easily expressed using parallel tasks, as are computations on spatial decompositions. The run time system dynamically schedules ready tasks onto threads (cores) and is charged with maintaining load balance across cores and termination detection. The task parallel programming model is supported OpenMP directives in C/C++/Fortran compilers.



Parallel execution of a tree-structured adaptive spatial decomposition computation using three threads. Each node is a task, and is colored according to the thread that processed it. Edges show spawning and completion dependencies between tasks.

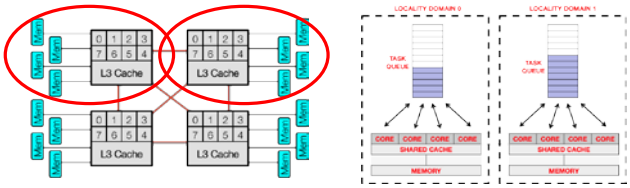
Observation of running times of tasks show that in some applications the same task requires dramatically different times to complete its work depending on when and where it is scheduled, we termed this behavior work-time inflation. It is the consequence of cache coherence protocols and communication latency among the multiple cores operating on values in shared memory. We have investigated scheduling techniques to mitigate these effects to enable efficient parallel execution of applications programmed using the task parallel model.

The Approach

Shared-memory nodes are organized into a number of locality domains, one for each socket. Within a socket multiple cores share a third level (L3) on-chip cache. Cache coherence protocols operate very efficiently between cores within a single locality domain. But between different locality domains cache coherence incurs higher costs as cache lines have to move between different L3 caches. We constructed a locality-based run-time scheduler using the Sandia Qthreads API. A separate scheduler runs in each locality domain, and work-stealing is used to move tasks between domains only when a scheduler's task queue is empty and no local tasks are pending (i.e. waiting on completion of a task by another local thread).

Highlights

- **Stephen Olivier (Ph.D. 2012, now at Sandia) awarded Supercomputing 2012 best student paper for his research on mitigating work-time inflation in task parallel programming.**
- **DOE INCITE awarded 60 million processor-hours annually 2014-2016 on the top ranked US Supercomputer Titan (Oak Ridge National Lab) to advance heterogeneous parallel methods for the simulation of multiphase flow and transport phenomena in porous media.**
- **Locality-based scheduling is a candidate for incorporation into the OpenMP standard.**



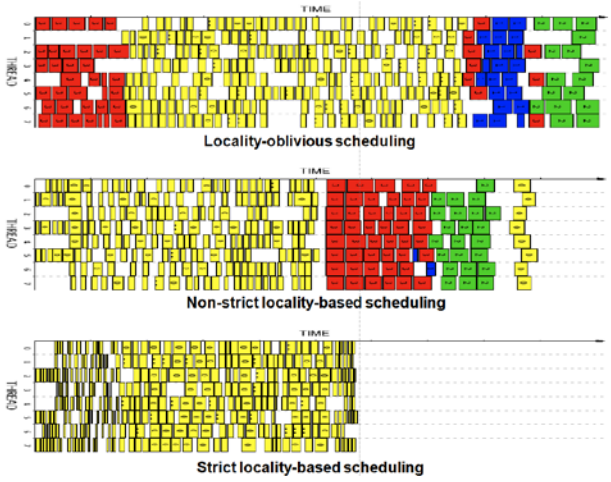
Left: Locality domains in a shared-memory node with four sockets and 8 cores per socket. Right, per-domain task schedulers flexibly schedule tasks to different cores, while performance is maintained through shared caches.

Results

Without any source level changes, the locality-based scheduler obtained 5%-300% performance improvement across a suite of task-parallel benchmarks (BOTS) when compared to the leading OpenMP runtime (from Intel). This was corroborated by lower work-time inflation measurements, fewer L3 cache misses, and an order of magnitude or more reduction in data rates across the socket-level interconnects (QPI).

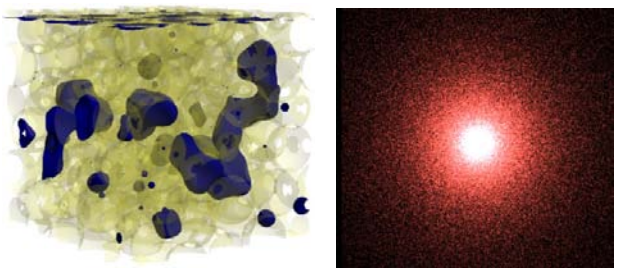
When a locality domain runs out of work it steals task(s) from other domains in order to improve load balance. But for certain classes of applications this can turn out to be counter-productive. A typical example is a simulation of a health care delivery system over time, modeling different locations, services provided, and a population of patients. The simulation itself has an inherent notion of locality, so that, for example, grouping together service providers and patients in a geographic region leads to substantial reuse of information as local patients prefer

local services (but may not always find them). In this setting a coarse top level decomposition of tasks by geographic region may lead to similar amounts of work for each region, although the detailed load balance will vary with each simulation time step and may become substantially unbalanced over time. We observed the scheduler's objective to minimize load imbalance through task stealing conflicts with minimizing work inflation because (1) a stolen task typically generates additional tasks operating on other non-local data and (2) attendant cache misses are repeated at the beginning of the next time step, following the original decomposition. For these sorts of problems a *strict* locality-based scheduler that never steals tasks from another domain can achieve better performance.



Socket-level view of locality-based scheduling. There are 8 cores executing tasks and each box corresponds to a single task. Yellow-colored tasks operate on data local to the socket. The width of each task is proportional to its run time. Task execution is shown in the longest running socket in the simulation time step.

Applications on Heterogeneous nodes. When computational accelerators are present in a node, tasks may be further segregated according to the preferred mode of computation and scheduled at run time according to the observed overall load balance. We are working on two different applications with different notions of locality and



Left: Multiphase fluid flow and mass transfer in porous media using lattice Boltzmann methods. The geometry of the porous medium is fixed while the fluid state evolves. Right: n-body gravitational simulation using the adaptive fast multipole algorithm. The locus and local density of the masses evolve in time necessitating an adaptive spatial decomposition and local adjustment of near-field vs. far-field computation to maintain load balance between CPU cores and GPUs.

Current Project Members

Jan Prins
Sridutt Balachandra

Collaborators

James McClure (Virginia Tech)
Casey Miller (UNC)
Alan Porterfield, Rob Fowler (RENCI)
Bronis de Supinski, Martin Schulz (LLNL)

Research Sponsors

Department of Energy: XPRESS eXascale Programming Environment and System Software (DE- FC02-l2ER26102).
National Science Foundation: FRG - Advanced Algorithms and Software for Problems in Computational Bio-Fluid Dynamics (DMS-0854961).
Department of Energy INCITE award: Multiphase Flow and Transport in Porous Medium Systems.
National Science Foundation: Revolutionary Advances in Modeling Transport Phenomena in Porous Medium Systems (CDI-0941235).

Recent Publications

J.E. McClure, J.F. Prins, C.T. Miller, "A novel heterogeneous algorithm to simulate multiphase flow in porous media on multicore CPU-GPU systems", *Computer Physics Communications* **185**, *in press*, 2014, http://dx.doi.org/10.1016/j.cpc.2014.03.012.

S. Olivier, B. de Supinski, M. Schulz, J.F. Prins, "Characterizing and Mitigating Work Time Inflation in Task Parallel Programs", Proc. Supercomputing (SC 12), Nov. 2012 (**Named Best Student Paper**). Reprinted in *Scientific Programming* **21**(3):123-136, 2013.

R.E. Overman, J.F. Prins, L.A. Miller, M.L. Minion, "Dynamic Load Balancing of the Adaptive Fast Multipole Method in Heterogeneous Systems", *Proc. ASHES workshop* (IPDPSW), IEEE, 2013.

A. Porterfield, S. Olivier S., S. Balachandra, J.F. Prins, "Power Measurement and Concurrency Throttling for Energy Reduction in OpenMP Programs", *Proc. HP-PAC workshop* (IPDPSW), IEEE, 2013.

S. Olivier, A. Porterfield, K. Wheeler, M. Spiegel, J. Prins, "OpenMP Task Scheduling Strategies for Multicore NUMA Systems", *International Journal of High Performance Computing Applications* (HPCA) **26**(2):110-124, 2012.

Keywords

Task Parallelism; Load balancing; Run Time Systems; Heterogeneous Computing, Parallel Programming Languages and Models.

For More Information

Jan Prins
Phone (919) 590-6213
E-mail: prins@cs.unc.edu
http://www.cs.unc.edu/~prins