Investigating and improving the scalability of task-based programming models

Dana Akhmetova, Michael Schlirp, Orljan Ekeberg, Erwin Laure
KTH Royal Institute of Technology

Introduction

A task-based program is a program formulated in terms of "tasks" (work to be done). A task is the set of instructions that occur between two interactions with the runtime system. Tasks are dynamically created, and then assigned by a task scheduler to CPUs for their execution.

A promising way to program Exascale applications: applications are divided into a myriad of small tasks; the system is performance-tuned with tasks (Yeasuk & Noorco).

Task-based programming models (TBPMs): Cilk/Cilk++/Intel Cilk Plus, OpenMP 3.0. OmpSs, StarPU, Intel TBB, Qthreads and others.

A few keywords to expose parallelism (Fig. 1): parallel work is created when the keyword "spawn" precedes the invocation of fib(n-1); the parent task can continue to execute in parallel with its child task (the semantics of spawning); the fib() function cannot safely use the values returned by its children until it executes "sync" (a local "barrier").

The poster presents our data-locality sensitivity study of TBPMs.


def fib(int n):
    if n <= 2:
        return 1
    a = spawn fib(n-1)
    b = spawn fib(n-2)
    sync
    return a + b;

Figure 1: Pseudo-code of task-based Fibonacci and its representative DAG for Fib(4).

Motivation

The modern computing is cheap and massively parallel, while energy and performance costs are impacted by data movement.

Task-based programming is very favorable for the future Exascale computing.

A NUMA architecture (Fig. 2) connects different NUMA nodes (Nodi 1-4) - typically multi-core CPUs (C1-C4), and their local memories - via interconnect links such as AMD's HyperTransport or Intel's QuickPath technology (Link), to enable a single logically shared global memory. Memory access times (latency) and possibly bandwidth between the cores vary depending on the physical location. For instance, they are reduced when Node 1 accesses memory associated with Node 3, due to overhead introduced by the link. This effect is typically referred to as NUMA effect.

Data locality - the probability of a memory reference being "local" to prior memory accesses. It affects performance of task-based programs: task stealing often results in data migration (when threads steal tasks, they also often take a working set of these tasks); task size affects locality as larger tasks are more likely to have larger memory footprints, so task granularity should be tuned for efficient use of the memory hierarchy.

To manage these issues, it is necessary to make the runtime system aware of data access patterns so that it can apply data-locality-aware scheduling strategies (for example, last level shared caches have larger space, which can be well used for groups of tasks that share some data (constructive cache sharing)).

Classical random work-stealing TBPMs (Cilk and OpenMP) have no notion of the location of data. Hierarchical work-stealing policies (Qthreads) check a task to be stolen first among cores in a local NUMA domain and only then among remote domains, but do not naturally extend to scheduling data-flow graphs while preserving possibly efficiency in terms of scheduling overheads and effectiveness of load balancing.

There is a need to study how TBPMs allocate memory and to what extend data-locality sensitivity of TBPMs affects execution time.

Results

Figure 4: Execution time and a number of minpage faults for one of the kernels, depending on an employed scenario. Black lines represents results for a serial version.

Conclusions and future work

The poster presented our study of data-locality sensitivity of TBPMs.

The studied TBPMs should be aware of data placement, e.g. by providing a mechanism to schedule tasks in a way that minimizes data movement.

Real-life applications are currently being studied for data-locality sensitivity.

New TBPMs are being deployed.

Energy and power measurements are to be collected and analyzed by accessing the RAPL counters as data movement incurs energy costs.

Acknowledgments

This work was funded by the European Union’s Horizon 2020 Research and Innovation program through the INTERTHink project under the Grant Agreement no. 647302 (www.interthink-project.eu). The simulations were performed on resources provided by the Swedish National Infrastructure for Computing (SNIC) at the PDC Center for High-Performance Computing (PDC-HPC).

References


[2] The picture was taken from http://www.iue.tuwien.ac.at/phd/weinbub/dissertation amendments.html