Investigating and improving the scalability of task-based programming models

Dana Akhmetova, Michael Schliephake, Örjan 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 overprovisioned with tasks ($Ntasks \gg Ncores$).
- **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.**

```
fib(int n)
{
  if (n < 2)
    return n;

  a = spawn fib(n-1);

  b = spawn fib(n-2);

  sync;

  return a + b;
}
```

```
fib(int n)
{
  if (n < 2)
    return n;

  a = spawn fib(n-1);

  b = spawn fib(n-2);

  sync;

  return a + b;
}
```

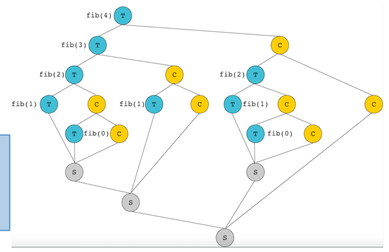

Figure 1: Pseudo-code of task-based Fibonacci and its representative DAG for Fib(4).

Motivation

- The modern computing is cheap and massively parallel, while **energy and performance costs are impacted by data movement**.
- Task-based programming is very favourable for **the future Exascale computing**.
- (1) proposed a promising approach for future Exascale programming models: to employ a best-effort local scheduler and a sophisticated (data-locality/workload-aware) remote scheduler.
- **A NUMA architecture** (Fig. 2) connects different NUMA nodes (Nodes 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 also 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 firstly 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 provably 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.**

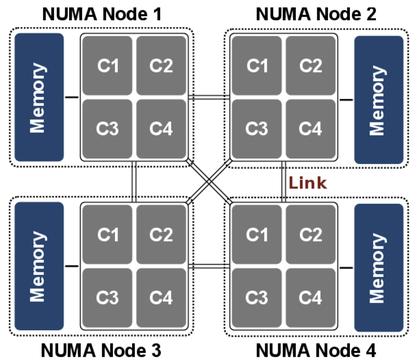

Figure 2: Typical NUMA architecture (2).

Method

- The codes are in Cilk, OpenMP, Qthreads + serial versions, with the same algorithmic structure, as we wanted to compare these TBPMs and to abstract from different algorithmic variants.
- A Tegnér node at the PDC Center for High-Performance Computing: 4x12 core E7-8857v2 (Ivy Bridge) (total 48 cores per node), 1TB RAM per node, 2x Nvidia Quadro K420 GPU.
- 6 scenarios (Fig. 3) to run applications on different NUMA domains of a Tegnér node: binding execution and memory on different NUMA nodes.
- Measuring hardware performance counters (approx. 20).

Test case	Computation	Memory
n0	NUMA 0 (CPUs 0-11)	NUMA 0
n1	NUMA 0 (CPUs 0-11)	NUMA 1
n2	NUMA 0 (CPUs 0-11)	NUMA 2
n3	NUMA 0 (CPUs 0-11)	NUMA 3
nn	NUMA 0 (CPUs 0-11)	no pinning
no pinning	no pinning	no pinning

Figure 3: Scenarios of running experiments.

Results

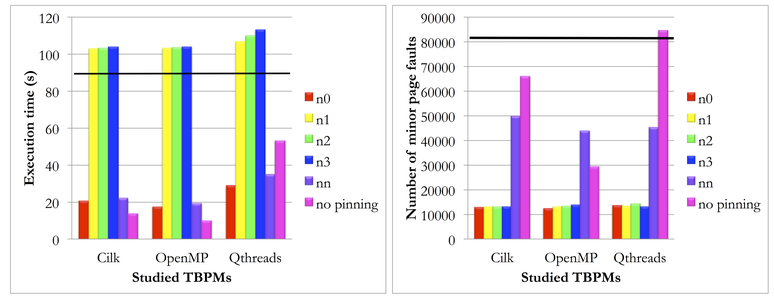

Figure 4: Execution time and a number of minor page faults for one of the kernels, depending on an employed scenario. Black lines represents results for a serial version.

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 induces energy costs.

Acknowledgments

This work was funded by the European Union's Horizon 2020 Research and Innovation program through the INTERTWinE project under the Grant Agreement no. 671602 (www.intertwine-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

[1] Akhmetova, Dana, et al. *On the application task granularity and the interplay with the scheduling overhead in many-core shared memory systems*. Cluster Computing (CLUSTER), 2015 IEEE International Conference on. IEEE, 2015.

[2] The picture was taken from
`http://www.iue.tuwien.ac.at/phd/weinbub/dissertationsu16.html`