sOMP: Simulating OpenMP Task-Based Applications with NUMA Effects

DAOUDI Idriss[1], VIROULEAU Philippe, THIBAULT Samuel, GAUTIER Thierry, AUMAGE Olivier

idriss.daoudi@inria.fr

September 23, 2020

Context and objectives

Context:

- anticipating applications behavior, studying, and designing algorithms
- experiment with various scheduling algorithms in a reproducible context
- predictive tools to evaluate applications performances on existing and non-existing systems

SimGrid $\rightarrow$ simulation of distributed infrastructures

- MPI, tasks, threads...
- OpenMP?
- In order to simulate OpenMP:
  1. parallel tasks with data dependency
  2. parallel loops

Objectives:

- predict the performances of task-based applications
- take into account Non-Uniform Memory Access (NUMA) effects

State of the art

- Distributed memory simulators:
  - SimGrid, Dimemas, BigSim, xSim...
- Shared-memory simulators:
  - for specific architectures: Aversa et al. (hybrid MPI/OpenMP on SMP), Simany (multicore)...
  - full system simulators: SimNUMA...
  - for task-based applications: HLSMN tool (without considering task dependencies)...
- **None of these tools takes into account task dependency and NUMA effects!**

Methodology

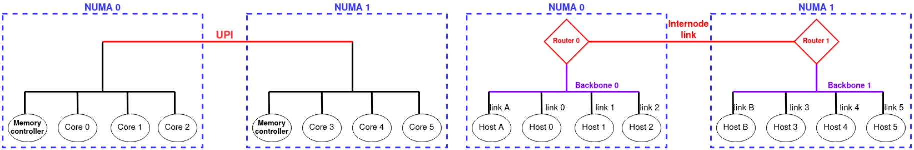
1. OpenMP application runtime data collection (tracing)
2. NUMA architectures modeling
3. Building a task-based application simulator with SimGrid
4. Construction of a performance model to take into account memory accesses
5. Simulation of the behavior of OpenMP applications using the performance model

Tracing with **TiKKi**

- uses the OMPT API integrated in OpenMP 5.0
- captures all the events required
  → to build an OpenMP application task graph
- generate several output forms of execution traces
  → task graph, Gantt chart, sOMP trace format
- sequence of parallel regions
  → where the events of each task is recorded
- one trace file per encountered parallel region

Modeling NUMA architectures with SimGrid

- exploits the possibilities offered by SimGrid's S4U API
- **NUMA node → cores + memory controller + links**
- exploit links parameters → model contention and concurrency
- **compromise between precision and simulation cost!**
  → around 70 times faster than real execution



(a) NUMA machine modeling

(b) NUMA machine modeling using SimGrid components

Task-based applications simulator: **sOMP**

Concept:

- exploit traces collected from a **sequential execution**
- execute the application on platforms modeled in the SimGrid sense using the collected traces
- simulate the execution time for a large number of cores

Components:

- **parser** for trace and platform files
- **scheduler** to submit jobs
- single **worker** per simulated core to execute the tasks
- **performance model** to refine the simulations

Communications-based performance model

- **Default** task execution **model** in sOMP:
  - task execution time obtained via the trace files
  - advance SimGrid's simulated clock
- **Communications-based model** using SimGrids' communications:
  - trace file provide the list of **memory operations performed by each task** (R, RW);
  - take into account those memory accesses;
  - memory access → communication to the memory controller
  - the size of each communication (in bytes) impact SimGrids' internal clock
    → **contention and concurrency** on the crossed links;

Evaluation

Architectures:

- dual socket **Intel Xeon Gold 6240** (CascadeLake)
  $\rightarrow$ 2 sockets, 2 NUMA nodes, 36 cores
- dual socket **AMD EPYC 7452** (AMD Infinity)
  $\rightarrow$ 2 sockets, 16 NUMA nodes, 64 cores

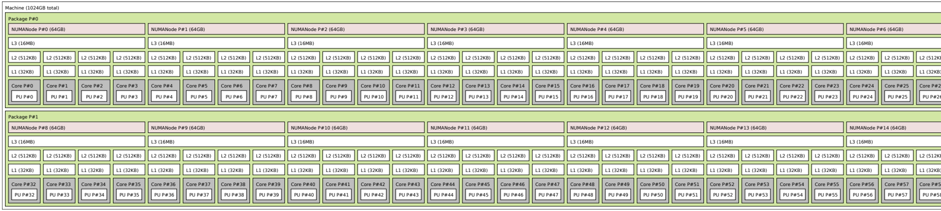
Kastors/Plasma[IWOMP2014]:

- benchmark suite to evaluate the implementation of the OpenMP task dependency paradigm;
- several matrix factorization algorithms: Cholesky, QR, LU.

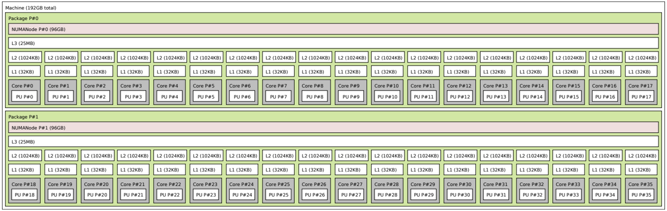
Metric:

- precision error (%) $= \frac{T_{native} - T_{simulated}}{T_{native}}$ x 100
- positive $\rightarrow$ optimistic simulation, negative $\rightarrow$ pessimistic simulation

Architectures overview with HWLoc



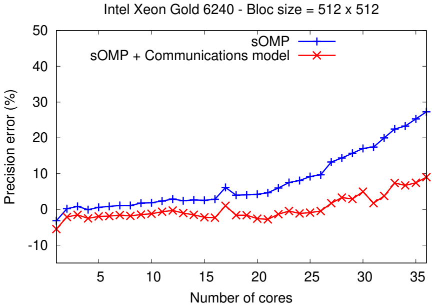
AMD EPYC 7452



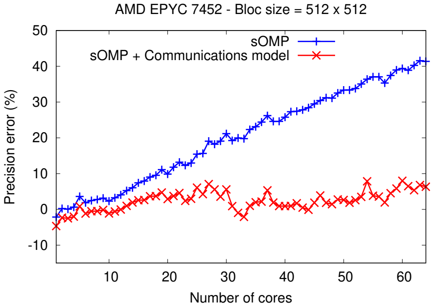
Intel Xeon Gold 6240

Results: Msize = 16384 x 16384, Bsize = 512 x 512

→ **Cholesky** algorithm on **different architectures**



(c) on Intel

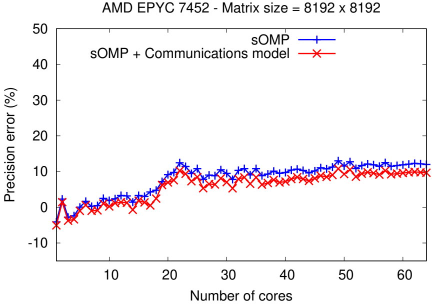


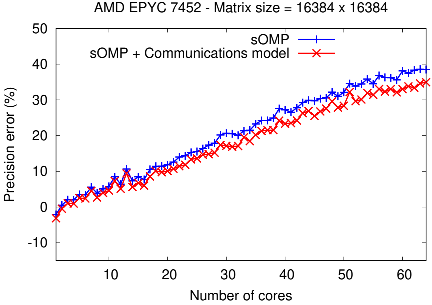
(d) on AMD

→ **less than 10%** precision error with the communications-based model
→ ideal for memory-bound applications
→ similar results on **LU** algorithm

Results: Bsize = 768 x 768, on AMD machine

→ **QR** algorithm for **different matrix sizes**





→ impact of granularity and number of threads
→ communications-based model is less efficient: QR is compute-bound

Summary

- We developed sOMP for **reproducible** researches!
- **TiKKi**
  → obtaining sequential trace of application
- **sOMP**
  → simulate the application on **modeled NUMA architectures**
- **Communications-based model**
  → improve simulations by taking into account NUMA effects
- **Result**
  → very good precision for some applications!

Future work

- Model data movements inside a cache
  $\rightarrow$ introduce a level of cache (L3) in the simulator
- Take into account various hardware affinity policies
- Introduce other scheduling policies
- Combine this work with MPI and GPUs simulations to model hybrid MPI/OpenMP applications

Sources:

- **TiKKi**: https://gitlab.inria.fr/openmp/tikki/-/wikis/home
- **sOMP**: https://gitlab.inria.fr/idaoudi/omps/-/wikis/home