Using differential execution analysis to identify contention

Mohamed Mosli, François Trahay, Alexis Lescouet, Gauthier Voron, Rémi Dulong, Amina Guermouche, Élisabeth Brunet, Gaël Thomas

ACMES group
Parallel programming

- **High Performance Computing**
  - Extensively used in weather forecasting, molecular modeling, physical simulation, …
  - Need for a lot of computing power

- **Parallelizing an application**
  - Split a problem into sub problems
  - Distribute over several processors
  - Processors communicate their contribution through a network

- **Performance improvement**
  - 4 processors → 4 times faster
  - 128 processors → 128 times faster?
Contestation on shared resources

- Multiple resources are shared
  - Memory hierarchy (caches, NUMA nodes, …)
  - Peripheral devices (hard drive, network card, …)
  - Software resources (locks, …)
Contention on shared resources

- **Multiple resources are shared**
  - Memory hierarchy (caches, NUMA nodes, …)
  - Peripheral devices (hard drive, network card, …)
  - Software resources (locks, …)

- **How to detect the source of a slowdown?**
  - Log resource usage
  - Measure software indicators
  - Use hardware counters

- **If there is a problem, is it bad for performance?**

Available resources on a computer:

- **I/O rate:** 178MB/s
- **Net rate:** 107MB/s
- **Load:** 7.3GB/s
- **Imbalance:** 198%
- **Ctx switch:** 12.10^3/s
- **LLC miss:** 57.10^3/s
- **Stall cycles:** 93.10^6/s
Analyzing memory access patterns

- **Goal:** running threads close to their memory
  - Access latency depends on the placement

- **NumaMMA memory profiler** [ICPP’18]
  - Low overhead collection of memory access
  - Detect the access pattern of threads on memory objects
  - Reports the most accessed objects
  - Available as open-source https://github.com/numamma

- **Allows to improve performance**
  - Select the best NUMA allocation policy
  - Improve the thread binding
  - Up to 28% performance improvement
Detecting thread contention

- **EZTrace** [CCGrid'11]
  - Generate execution trace from an application
  - Available as open-source: http://eztrace.gforge.inria.fr

- **Differential execution analysis** [submitted to TPDS]
  - Detect sequences of event that repeat
  - Compare the duration of sequences
  - Simulate the execution time without contention
  
  → Universal indicator for contention

- **Evaluation on 27 applications**
  - Detect 12 problems (lock contention, disk IO, network, memory placement, ...)
  - Up to 900% speedup once (manually) fixed

- **Offline analysis**
  - Allow users to optimize their application
Runtime decisions based on predictions

- Offline analysis that feeds a runtime

- **Pythia** [ANR JCJC 2018-2021]
  - Collect & analyze execution trace
  - Provide runtime systems with hints on the future

- Allow runtime systems to take wise decisions
  - Decisions based on past event and probable future events
    → Prefetch irregular IO operations
    → Binding thread close to their resources
  - Better decisions → better performance
Thank you!

**Open-source software**
- NumaMMA – https://github.com/numamma/

**Publications**
- F. Trahay, et al. “NumaMMA: NUMA Memory Analyzer”. In ICPP 2018
- F. Trahay, et al. “Selecting points of interest in traces using patterns of events”. In PDP 2015

**PhD students**
- M. Mosli Bouksiaa (2014-2018)

**Projects**
- FUI IDIOM (2018-2020)
- ANR JCJC Pythia (2018-2021)