Performance analysis for parallel applications

- Robert Roy
- Department of Computer and Software Engineering

*Monitoring Distributed Systems for Diagnostic Purposes Workshop at Ericsson, Montréal January 29-30, 2008*
Outline

- Parallel vs. distributed computing
- Specific performance problems
  - using MPI or OpenMP
  - some parallel tools
- RQCHP: the HPC network
- DRAP laboratory
- Developer's remarks
Parallel vs. distributed computing

- **Parallel systems:**
  - Often SPMD programs:
    - using OpenMP or MPI
  - Real-time performance:
    - dedicated nodes
    - No workload sharing
    - 1-to-1 mapping
      - (Processor, Thread)

- **Distributed systems:**
  - Often MPMD programs:
    - client-server, P2P, ...
  - ...

Using MPI and OpenMP standards

- **Message passing interface**
  - data distribution and task scheduling:
    - controlled by developer
  - explicit communications
  - often predictable

- **Open Multiprocessing**
  - shared memory and fork/join model:
    - synchronized by developer
  - implicit communications
  - … less predictable
  - Memory access?
Some MPI performance problems

- **Point-to-point communication**
  - Late sender: Idle receiver waiting for message
  - Late receiver: Idle (blocked?) sender

- **Collective communication**
  - Late broadcast/scatter: Root not ready to send
  - Early reduce /gather: Waiting for non-root

- Any I/O: MPI-IO, PVFS, database?
MPI perf analysis tools

- tracing
  - MPE/Jumpshot, Trace Collector/Analyzer, KOJAK

- profiling
  - TAU, mpiP, hpcprof
    - many tools use
  - PAPI, PMPI, DynInst
Now OpenMP performance problems

- Load Imbalance
  - In parallel section  Work per thr. or # of thr.
  - In parallel loop   Iterations (wait, no wait)

- Synchronization, parallel control
  - Contention in critical section  Competing threads
  - Serialization  Parallel loop with critical section
  - Bad scheduling  Loop overhead
  - False sharing  Overhead to get same cache line
OpenMP perf analysis tools

- tracing
  - Thread Checker, OPARI

- profiling
  - TAU, Thread Profiler, ompP
OpenMP performance analysis

- Quick (and cheap) look:
  - use hardware counters to identify problems
- Identify region and search the cause:
  - true/false sharing invalidations
  - coherence misses (?)

« Performance of a multiprocessor depends on the performance of the system when sharing data. »
Hennessy & Patterson, Computer Architecture, IV ed., MK(2007)
Cache coherence problems

- Detect coherence miss:
  - access data that was in the proc.cache previously, but was invalidated by a write from another proc.
  - Not for hardware counter!
- SMP cache simulator: ccSIM (Rsim, SimOS...)
  - Huge tracing files: online compression (LZO, ...)

« Source-code-correlated cache coherence characterization of OpenMP benchmarks »
RQCHP: HPC network

http://rqchp.ca

• researchers from:
  • Universités: de Montréal, de Sherbrooke, Concordia, Bishop, École Polytechnique,
  • FCI funding
RQCHP: HPC network

http://rqchp.ca

- **Mammouth (Mp) @ Université de Sherbrooke**
  - Most powerful supercomputer in Canada
  - Dell 1425SC: 576 nodes dual-Xeon - 8 GB mem
  - Network interface (communication) InfiniBand

- **Altix 4700 @ Université de Montréal**
  - SGI 384 nodes dual-Itanium II - 4 GB mem
  - Shared memory image: NUMAlink
Analysts use tools:
- `gprof` GNU/tool
- `pfmon` HP dev.
- `pgprof` Portland gr.
- `histx` SGI
- Trace Analyzer (ex-Vampir)

Plus, for debug:
- `gdb`, `pgdb`, `idb`
- `Totalview`

And monitoring:
- `ganglia`
Altix 4700 – common problems

- Code run well with small #(proc)
  - Automatic parallelization, etc.

- When #(proc) is increased, be aware of:
  - any memory leak
    - TotalView or MemoryScape can help
  - cache coherency misses
  - doing too much I/O
The DRAP laboratory

http://www.polymtl.ca/drap

- **Design and Realization of Parallel Applications**
  - Parallel codes:
    - DRAGON/DONJON codes (nuclear engineering)
      - Varin, Dahmani, He
    - LBM and DEM codes (chemical engineering)
      - Leclaire, Vidal
  - Tools for parallel systems:
    - AdélieLinux
    - Clone+Napalm
    - ..and other projects: MPI2XML, PolyJob
AdélieLinux

• Idea:
  – diskless node – get OS by network (Adélie/SSI)
  – with optimized parallel tools (Adélie/HPC)

• Single-system image
  – integrated in Gentoo distr. (32 & 64-bit versions)
  – limits: scalability, I/O

• Benoît Morin, B.Ing.,M.Sc.A., now working at IREQ
Clone+Napalm

- **Idea:**
  - light virtual nodes – boot/transfer OS clones
  - coupling virtualization (Xen) and UnionFS
- **Single-filesystem image**
  - transferring kernel and early-userspace module
  - using initramfs under Genkernel
- Jean-François Richard, B.Ing., M.Sc.A., now in Norway
Clone+Napalm

- Heartbeat & monitoring of clones:
  - gmond
  - gmetad
- XML data
  - ganglia
Clone+Napalm

- Performance of user-level programs:
  - Processor-bound: No virtualization overhead
  - Network (netperf): No visible overhead
    - (here only 1 clone <= 1 NIC)
  - I/O: 10-15% overhead
    - Most overhead is from UnionFS
Steps in performance analysis

• instrumentation/measurement
  • source, wrapper library, binary
  • hardware counters, trace points
  • must ensure:
    − correctness, scalability

• analysis
  • online, automatic, post-mortem
  • timeline diagrams, charts, metric panels, source code
Developer's remarks

• usability:
  • less work to do => most likely to use the tool!
    − clear documentation, quick start, tutorial

• efficiency:
  • Is tool any better than ad-hoc techniques?
  • Does tool help to find performance problem?
  • Is tool able to analyze and find it? ... solve it?
My bucket list

- Tracing/profiling in the VMM:
  - anyone remembers **SIMMON**?

- Parallel analysis of trace files:
  - next generation **Vampir**?