Center for Scalable Application Development Software

CScADS Workshop on Performance Tools for Petascale Systems

Plenary Discussion Notes

John Mellor-Crummey
Department of Computer Science
Rice University

Outline

- ☛ Performance Problems at Scale
- Analysis of Applications at Scale
- Measurement Support
- Multithreaded Code
- Understanding Node Performance
- Heterogeneous Platform Issues
- Support for Dynamic Adaptation
- Data Format Issues
- Miscellaneous Issues
- Workshop Outcomes
- Funding Priorities

Underutilization of Processors

- Symptom: block waiting for communication to complete
- Causes
  - serialization
    - trace analysis
      - space time diagrams
      - analysis of partially-ordered execution graphs
    - comparative analysis of costs in context
      - e.g. are there linear relationships between processes?
  - load imbalance
    - computation vs. communication
    - global vs. local
  - lack of communication and computation overlap
  - resource contention
  - runtime system interference

Cost of Sharing Physical Resources

- Network congestion
  - is it a problem (for particular applications)
  - how can we diagnose if it is a problem?
    - what can be observed from a network interface
    - internal to the network itself
  - is adaptive routing needed for performance on these large machines
    - what if we run into performance problems?
    - do we need to provide input into policy choice?
- Collective bandwidth needs of cores sharing memory interface
- File system issues (within and across jobs)
  - access to file metadata
  - sharing of IO nodes
  - sharing of network paths to IO nodes
  - striping of files to IO nodes and disk
  - sharing of disk spindles

Topology Impact on Network Congestion

- Potential problems
  - OS policy for allocation of physical nodes
  - bad mapping of logical to physical topology could bring
    - link congestion
    - path dilation
    - interference from other jobs
- Observability
  - congestion?
  - network counter issues
    - some networks support direct measurement of congestion
    - virtualized counters for access to network by multiple cores
- Diagnosis
  - embedding of logical topology into physical topology
    - information about physical topology
    - information about routes between communication partners
    - identify collective communication among subgraphs in logical topology
    - nearest neighbor communication within logical topology

Outline

- Performance Problems at Scale
- ☛ Analysis of Applications at Scale
- Measurement Support
- Multithreaded Code
- Understanding Node Performance
- Heterogeneous Platform Issues
- Support for Dynamic Adaptation
- Data Format Issues
- Miscellaneous Issues
- Workshop Outcomes
- Funding Priorities

Parallel Performance Analysis Decision Tree

- Is the job running fast enough; if yes = done
- Is the problem that it is not scaling
- If it is scaling, then explore node performance

Methodologies for Investigating Problems

- Understanding communication performance in context
- Examining variability among peers
- Load imbalance problems and approaches
  - balancing aggregate memory b/w demands among processor
  - on heterogeneous systems adaptive assignment based on efficiency
- Comparative analysis of multiple experiments
  - e.g. understand impact of congestion and dilation on communication performance
    - controlled experimentation, simulation, modeling
- Clustering
- Anomaly detection
  - expectation-driven approach
- Knowing when you are done tuning
  - assessing the potential benefits of alternative implementations
    - modeling and simulation are useful here
    - microbenchmarking might also be useful to assess value of alternatives

Outline

- Performance Problems at Scale
- Analysis of Applications at Scale
- ☛ Measurement Support
- Multithreaded Code
- Understanding Node Performance
- Heterogeneous Platform Issues
- Support for Dynamic Adaptation
- Data Format Issues
- Miscellaneous Issues
- Workshop Outcomes
- Funding Priorities

PMU Design

- Current counters often have only limited utility
- Desires
  - architectural counters
  - counters that provide information about losses
    - e.g. lost cycles
  - counters for understanding NUMA memory
    - local access vs. remote access counts Commodity processors
- Issue: HPC community lacks ability to influence designs

Measurement Infrastructure

- libmonitor: a layer to support instrumentation upcalls
  - similar layer being developed by Sandia Livermore (Cranford)
  - Dyninst also has a similar abstraction
  - action item: an opportunity for joint investment in a single implementation
- MPIR interface
  - used by TotalView
  - interface was designed for debuggers
    - ok for 90% dyninst needs
    - malicious for last 10%
  - action item: more fully specify the MPIR interface
    - e.g. 2-3 ways to start the process after attach, which is platform specific
  - issue: MPI2 control interface is different
  - players: LLNL, ANL (Gropp), Wisconsin, OpenMPI group (Graham)
- launchmon
  - job startup and basic process control
    - attach, detach
    - locate application processes
  - uses MPIR interface

Binary Rewriting Approaches

- Dyninst
  - difficulty
    - shared libraries get loaded at different locations makes current binary rewriting by Dyninst problematic
  - long term: much better Dyninst
    - on horizon, in deliverables, but behind other deliverables due to logical dependencies
    - needs better code generator
    - plan for the petascale systems is the longer-term route
      - the static rewriter will use whole function rewrites
- Just-in-time code generators for rewriting
  - Pin, valgrind
- Static rewriters
  - Alto, Fit/Diablo (link-time rewriter with block level support?)

Binary Rewriting Issues

- Platform availability
- Thread safety
- Will JIT methods meet performance needs?
  - valgrind does not scale for threaded programs
- A catalogue and characterization of these tools would be good
- Having the right information available at runtime
  - Sun compiler “-xbinop=prepare?” to support transformation at runtime
    - enough info in object files that post-compiler binary xform tools can use it
      - e.g. Sun’s thread analyzer
  - Cray now uses Dwarf3

Compiler Instrumentation

- Some compilers support hooks and interfaces
- Issues: not publicized
- Control over instrumentation insertion
  - e.g. avoid specific functions
- Action item: query vendor partners about standardizing interfaces
  - Oregon (Malony) to take the lead on outlining a proposal
    - circulate to other players: Pathscale, IBM, PGI
  - existing models
    - Sun has some interfaces for instrumentation
      - xprofile=function?
    - PGI mproc=func

Process Control

- ptrace/utrace
  - ptrace lacks support for multithreaded process control
    - responses from signal interface
  - utrace is a kernel internal interface in Fedora core 7
    - want it exposed to tools
- Action item: explore standardization of a debugging interface
  - who should be involved in the discussions
    - petascale system vendors
    - Etnus
    - broader Linux community?
  - Wisconsin/Maryland to take the lead on defining a wish list
    - circulate to Cray (DeRose), ANL (Beckman), IBM (Klepacki)

Outline

- Performance Problems at Scale
- Analysis of Applications at Scale
- Measurement Support
- ☛ Multithreaded Code
- Understanding Node Performance
- Heterogeneous Platform Issues
- Support for Dynamic Adaptation
- Data Format Issues
- Miscellaneous Issues
- Workshop Outcomes
- Funding Priorities

Multithreading Flavors

- Hardware threads
- Software threads
- Is there an issue with visible abstractions of h/w thread contexts?

Performance Issues for Multithreaded Code

- Locality
  - thread affinity
  - co-location of threads
    - same socket or not
      - is data sharing cheap, or will it involve SMP coherence traffic
- Causality for waiting
  - why I was waiting for a lock (who was holding it)
- Resource contention (see next slide)

Understanding Local Resource Contention

- Hardware support for monitoring resource contention?
  - multiple cores sharing cache
    - false sharing
    - true sharing
    - thread migration causes data access patterns to flip
    - observability
  - multiple cores sharing the memory subsystem
  - multiple cores sharing the network
  - number of performance counters available
    - too little information for observability as # of cores increase
- Observability: extracting information from PMU's not provided by cores
  - memory interface
  - accelerators will have separate monitoring capabilities
  - counters in NICS, etc
- Issue: processor virtualization
  - what to do with performance counters under virtualization

Issues Monitoring Multithreaded Code

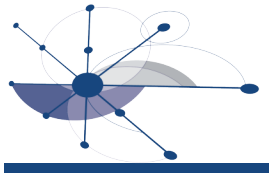
- Runtime-data management when monitoring multithreaded code
  - e.g. maintaining integrity of data structures used by monitoring with locks without impacting performance requires subtle design
- When observing multithreaded code, often want to
  - identify thread so it can be uniquely distinguished
  - nested parallelism: what is my ancestry?

Profiling Multithreaded Code

- Sample sources
  - itimer (delivered on a per-thread basis)
    - can start timers on per thread basis on Linux, not on BSD
  - hardware counters based on SIGIO
    - Perfmon driver posts SIGIO
    - requires virtualization of hardware counters on a per thread basis
- OS requirements
  - need to be able to direct SIGIO to specific threads
  - action item: is this an issue with Cray and Blue Gene OSes?
  - Solaris delivers SIGPROF with ticks in various states
    - executing code is delivered tick representing CPU time
    - blocked thread is delivered tick when thread unblocks that describes (a) how long spent waiting, (b) what it spent its time waiting for
  - action item: investigate how Linux handles timer-based profiling of blocked threads

Call Path Profiling

- Characteristics
  - instrumentation-based vs. asynchronous event-based
  - function level (Julich, Oregon) vs. PC based (Sun, Rice)
  - temporal information (Sun) vs. no temporal information (Rice, Oregon)
  - asynchronous sampling (Rice, Sun) vs. synchronous sampling only (Oregon, Julich)
    - synchronous + asynchronous sampling: Sun, Rice (soon)
- The product of monitoring
  - calling context tree (Rice)
  - temporal vector of events with context (Sun)
  - vector of unique contexts labeled with their costs (Oregon)



Instrumentation of Multithreaded Computations

- Timing issues are critical
- Want to instrument every single event in the threading runtime

OpenMP Threading Performance Issues

- Monitoring OpenMP performance
  - see the paper http://www.compunity.org/futures/omp-api.html
- Diagnosing OpenMP inefficiencies
  - identify time waiting in barrier
    - shows imbalance
  - when a region is under-parallelized
    - slave threads in OMP_idle
- Attributing performance with nested parallelism
  - a proposal for managing thread ancestry
    - non-nested parallelism
      - thread identifier 0 to n-1
    - nested parallelism
      - vector of thread ids
        - should vector include inactive regions or not?
        - include them because it is consistent with static information (Wolf)

Outline

- Performance Problems at Scale
- Analysis of Applications at Scale
- Measurement Support
- Multithreaded Code
- ☛ Understanding Node Performance
- Heterogeneous Platform Issues
- Support for Dynamic Adaptation
- Data Format Issues
- Miscellaneous Issues
- Workshop Outcomes
- Funding Priorities

Decision Tree for Memory Hierarchy

- Historical
  - Intel Itanium PMU design has nice decision tree
  - old Cray vector machines
    - output from flowtrace/flowview
      - text describes each routine
        - read bandwidth, store bandwidth, MFLOP ratings per routine
- What is the CPI?
  - if CPI is bad and b/w is only 10%
    - likely using long latency computational instructions
- Look at counters to see how much data is being moved
  - if it is close to peak b/w, investigate whether it is being used well
  - if it is running at 10%, b/w is not a problem
- Levinthal's work
  - http://www.devx.com/go-parallel
  - bubble events on Itanium
  - microinstruction front end and back end dispatch and graduation rates
    - Itanium and Core-2

Memory Hierarchy Analysis Methods

- Profiling
  - explore memory bandwidth limits across whole program
  - and per routine
  - instruction issues
- Issues
  - prefetch streams, open pages, associativity
- Problems to look for
  - lots of read/write turnarounds
  - page conflicts
  - bandwidth in and out of each level of memory hierarchy
    - if more data going into cache than going out, severe gather problem
- Expectations for applications
  - how many cache misses do you expect?
  - minimum memory traffic
  - use performance counters to assess match with expectations
- Is it worth improving? 4 quadrants to consider
  - low effort vs. high effort to make changes
  - small improvements vs. large improvements
  - want to work in (low effort, large improvement) quadrant

Outline

- Performance Problems at Scale
- Analysis of Applications at Scale
- Measurement Support
- Multithreaded Code
- Understanding Node Performance
- ☛ Heterogeneous Platform Issues
- Support for Dynamic Adaptation
- Data Format Issues
- Miscellaneous Issues
- Workshop Outcomes
- Funding Priorities

Heterogeneous Processor Difficulties

- Usually not a single source view for heterogeneous programming models and tools
- Benefit of computation acceleration offset by data transfer costs
- Hard to assess opportunities for performance improvements
  - role for modeling
- Asynchronous operation complicates measurement and performance assessment
- Hard to map accelerator performance metrics to application source
  - need to interfacing performance tools better with programming environment
- Lack of portability

Issues for Particular Technologies

- FPGA issues
  - HW counters nascent if non-existent for accelerators
  - use timers outside the device
  - next steps: synthesis of counters for monitoring?
  - best approach: co-design for performance
- GPU issues
  - difficulty of programming
- Multithreaded units (Tera model)
  - requires completely different model for performance tuning
    - usual: first scalar performance, then communication
    - MTA:must avoid serial regions
      - rely a lot on traces (based on sampling)
      - need accumulation to make sense of performance
      - traceview shows availability of processors and your utilization
      - navigate from utilization view to CANAL to explore lack of parallelism
      - bprof used the least
- Vectors
  - relatively well understood

Outline

- Performance Problems at Scale
- Analysis of Applications at Scale
- Measurement Support
- Multithreaded Code
- Understanding Node Performance
- Heterogeneous Platform Issues
- ☛ Support for Dynamic Adaptation
- Data Format Issues
- Miscellaneous Issues
- Workshop Outcomes
- Funding Priorities

Support for Runtime Introspection

- MPI peruse interface (Terry Jones, project lead)
  - conceived as an interface to look inside an MPI implementation to get insight into factors affecting performance
    - e.g. message queue info
  - poorly defined
    - makes assumptions about implementation
    - problem: MPI implementations are widely different
  - implemented by OpenMPI

Outline

- Performance Problems at Scale
- Analysis of Applications at Scale
- Measurement Support
- Multithreaded Code
- Understanding Node Performance
- Heterogeneous Platform Issues
- Support for Dynamic Adaptation
- ☛ Data Format Issues
- Miscellaneous Issues
- Workshop Outcomes
- Funding Priorities

Data Format Questions

- Opportunities for standardization?
  - human readable formats (e.g. XML)
  - descriptive language for metrics
    - each tool describes measurements using common language
      - e.g. CPU time means something different
    - Paradyn has a description format (MDL for describing how to make measurements)
- Obstacles to standardization?
- Plans for standardization?
  - what players need to be involved?
    - development of shared requirements

Trace Formats

- OTF, OTF'
  - relatively high level (tied to semantics of OpenMP and MPI)
  - common event model (program structure & sys structure as well)
  - common read format for OTF and epilog
- Paraver trace format
- Idea: share API that read/write multiple formats

Call Path Profile Data Format

- Differences
  - different types: function level (Julich, Oregon, eventually Rice) vs. PC level (Rice)
  - synchronous (Oregon, Sun, Rice) vs. asynchronous collection (Sun, Rice)
  - no temporal information (CCT)
  - sampled events with context (Sun, Oregon)
    - Sun has temporal trace of contexts; Oregon collapses common contexts into single event and loses temporal knowledge
- Want hooks for extensibility
- Attempt to agree on common schema for the data?
- Interest in human-readable format for tool interchange

Outline

- Performance Problems at Scale
- Analysis of Applications at Scale
- Measurement Support
- Multithreaded Code
- Understanding Node Performance
- Heterogeneous Platform Issues
- Support for Dynamic Adaptation
- Data Format Issues
- ☛ Miscellaneous Issues
- Workshop Outcomes
- Funding Priorities

OS Issues

- Support for performance measurement
  - An API request for Linux per thread user CPU time
    - API: how much time has this thread accumulated so far
      - getrusage is not fast enough
        - Linux 2.4 and earlier - accurate info
        - 2.6 and onward - mapping pthreads to kernel threads is problematic
      - pthread vs. tid
        - pthreads might migrate to different kernel threads n x m threading
      - Solaris 9 abandoned n x m threading in favor of 1 to 1
      - DEC pushed n x m threading
- Support for performance tuning
  - OS memory management (e.g. page coloring)
- Security

Compiler/user Interaction

- Explanations for what the compiler was able to accomplish and what it wasn't able to do would be enormously helpful
  - compilers don't externalize data
- Old SGI compilers would ask questions and pragmas could be used to provide answers
- Opportunity: enable users to offer pragmas as tuning guidance
  - e.g. is a vector a permutation

Outline

- Performance Problems at Scale
- Analysis of Applications at Scale
- Measurement Support
- Multithreaded Code
- Understanding Node Performance
- Heterogeneous Platform Issues
- Support for Dynamic Adaptation
- Data Format Issues
- Miscellaneous Issues
- ☛ Workshop Outcomes
- Funding Priorities

Workshop Outcomes

- Exchange of Ideas
  - new challenges
  - data collection and current analysis strategies
  - needs for new analysis strategies
  - barriers to tool adoption
- Identified opportunities for community interaction
  - planning to increase collaboration
- Identified opportunities for better OS support

Workshop Outcomes

Opportunities for community interaction

- Understanding executables
  - began dialogue about requirements analysis, refining, and sharing
    - symtab API
    - instruction API
    - OpenAnalysis
  - frame understanding for stack unwinding
    - Sun might provide unwinding code
    - guerilla implementation of precise stack unwinding on x86
      - explore using Intel's Xed
  - long term use of Xed for x86 implementation of instruction API?
    - is closed source acceptable?

Workshop Outcomes

Opportunities for community interaction

- Measurement infrastructure
  - identified opportunities for collaboration
    - libmonitor
    - wrapping tool
  - identified need for refining MPIR interface
  - identified need for better process control

Workshop Outcomes

Opportunities for community interaction

- Analysis and presentation
  - explore componentizing presentation & visualization modules
  - terminology for types of displays?
- Developed taxonomy of data format approaches
  - who, what, why (capabilities)
  - next steps: interoperability rather than standardization seems in order
    - there are good reasons for differences in data recorded

Workshop Outcomes

Feedback for the OS community

- Need a standard interface to performance counters
  - Perfmon2 is our choice
- OS support for performance measurement
  - want Linux API for per thread user CPU time accumulated so far
  - BSD needs support to start itimer on a per thread basis
  - need virtualization of hardware counters on a per thread basis
    - can start timers on per thread basis on Linux, not on BSD
  - hardware counters based on SIGIO (Perfmon driver posts SIGIO)
    - need to be able to direct SIGIO to specific threads
  - for multithreaded code, follow Solaris's lead and deliver SIGPROF with ticks in various states
    - executing code is delivered tick representing CPU time
    - blocked thread is delivered tick when thread unblocks that describes (a) how long spent waiting, (b) what it spent its time waiting for

Outline

- Performance Problems at Scale
- Analysis of Applications at Scale
- Measurement Support
- Multithreaded Code
- Understanding Node Performance
- Heterogeneous Platform Issues
- Support for Dynamic Adaptation
- Data Format Issues
- Miscellaneous Issues
- Workshop Outcomes
- ☛ Funding Priorities

Funding Priorities

New techniques

- Anomaly detection
- Mining large performance database archives
- Analysis techniques for new architectures – GPUs, multi-core
- Integration of analysis into visualization
- Integration of system level data into application level analysis
- Data reduction - for both analysis and visualization

Funding Priorities

- Support for augmentation & componentization of existing infrastructure
  - rewarding developers whose software is reusable/reused
- Support for standardization
- Support for international collaborations
  - joint programs with other countries?
    - supercomputing infrastructure is a priority in the EU
  - travel and coordination represent a minimum investment which should yield good ROI
    - APART model
- Training support
  - fund engagement with application teams
  - workshops to help application teams learn to use tools productively
    - benefit for tool developers: having users adopt tools provides ROI to DOE

Funding Issues

Model for long term maintenance of software?

- Tools built by academic groups: how to support them in the long term?
  - tension in academia: innovation vs. support
  - who owns the tool?
    - awkward if owner is not original developer
  - co-locating innovators with maintainers is essential
    - maintenance programmers will burn out if divorced from innovation
- Examples: measurement support
  - support PAPI group to port and maintain on leadership class machines?
    - pro: the UTK folks are best qualified
    - con: should be vendor's responsibility (LLNL)
  - standard OS interface is important too, e.g. perfmon
- Model for government and industry partnership in funding?
  - government + HPC vendors
  - alternate HPC revitalization message
    - those who buy the machines own the problem

Next Steps

- Coordinate with PERI to propose workshop on processor performance counters?