TAUoverMRNet (ToM): A Framework for Scalable Parallel Performance Monitoring

Aroon Nataraj, Allen D. Malony, Alan Morris
University of Oregon
{anataraj,malony,amorris}@cs.uoregon.edu

Dorian C. Arnold, Barton P. Miller
University of Wisconsin, Madison
dorian.arnold@gmail.com
bart@cs.wisc.edu
Motivation

- Performance problem analysis increasingly complex
  - Multi-core, heterogeneous, and extreme scale computing

- Shift of performance measurement and analysis perspective
  - Static, offline $\Rightarrow$ dynamic, online
  - Support for performance monitoring (measurement + query)
  - Enabling of adaptive applications

- Prerequisites for performance measurement
  - Low overhead and low perturbation
  - Runtime analysis antithetical to performance tool orthodoxy

- Neo-performance perspective
  - Co-allocation of additional (tool specific) system resources
  - Make dynamic, performance-driven optimization viable
Performance Observation Needs

- Performance problem type determines observation approach
  - Translates to requirements for measurement and analysis
- Standard offline performance diagnosis/tuning process
  - Compile-execute-measure-analyze cycle
  - Pre-determined performance experiment (events, measures)
  - Static application execution and optimization
- Standard approach difficult to apply to complex execution
  - Dynamic applications where performance changes
  - Extreme scale, heterogeneous systems with high dimensionality
- Requires extended online performance measurement support
  - Dynamic monitoring and performance feedback
  - Raises vital concerns of overhead and perturbation
    - bigger issue in online systems due to global effects
Performance Observation Modes

- **Post-mortem**
  - Performance data interpreted offline
  - May lack temporal detail (e.g., using profiles only)

- **Post-mortem with temporal detail**
  - Still offline interpretation
  - Can generate prodigious data volumes (e.g., using tracing)

- **Online**
  - Performance data queried, interpreted at runtime
  - Suitable to long running applications (especially at scale)
  - Similar in spirit to real-time visualization

- **Online with feedback** into ...
  - Measurement subsystem (optimize, distribute analysis)
  - Application (steering)
Monitor for Performance Dynamics

- Runtime access to parallel performance data
  - Scalable and lightweight
  - Support for performance-adaptive, dynamic applications
  - Focus on parallel profile data
- Alternative 1: Extend existing performance measurement
  - Create own monitoring infrastructure
  - Integrate with measurement system
  - Disadvantage: maintain own monitoring framework
- Alternative 2: Couple other with monitoring infrastructure
  - Leverage scalable middleware from other supported projects
  - Challenge: measurement/monitor integration
  - TAU over Supemon (ToS) (UO, LANL)
  - TAU over MRNet (ToM) (UO, University of Wisconsin)
Talk Outline

- Motivation
- Performance observation needs
- Performance observation modes
- Monitoring for performance dynamics
- Separation of concerns and MRNet
- TAUoverMRNet (ToM)
  - System design
  - Monitor instantiation problem
  - ToM filters: distributed analysis, reduction
- System characterization
- Future plans and conclusion
Separation of Concerns

- Online *performance monitoring* decomposes into
  - Measurement
  - Access / Transport

- Measurement sub-system
  - Measures application performance
    - parallel profile per context (MPI ranks, processes, threads)
  - Maintains performance state locally (global performance data)

- Access / Transport
  - Query of distributed performance state (frequency, selection)
  - Bridges application (source) with monitors / front ends (sinks)
  - Moves performance data from source to sink
  - Distributed performance data processing (MRNet)
    - distributed performance analysis / reduction also feasible
What is MRNet?

- Multicast Reduction Network
  - Software infrastructure, API, utilities (written in C++)
  - Create and manage network overlay trees (TBON model)
  - Efficient control through root-to-leaf multicast path
  - Reductions (transformations) on leaf-to-root data path
  - Packed binary data representation

- Uses thread-per-connection model
  - Supports multiple concurrent “streams”

- Filters on intermediate nodes
  - Default filters (e.g., sum, average)
  - Loads custom filters through shared-object interface

- MRNet-base tools (Paradyn, STAT debugger, ToM)
TAU Transport Abstraction Layer

- Application calls into TAU (**TAU.DB_DUMP()**)
  - Application specific intervals
    - example: per-iteration or phase
  - Regular periodic intervals
- Configuration specific
  - Compile or runtime
  - One per thread
- Develop abstract transport interface
  - Adaptors to alternative monitor infrastructure
- Push-Pull model
  - Source pushes and sink pulls
MRNet Back-End Adapter

- Adapter responsibilities
  - Initialization
  - Finalization
  - Control
  - Performance data output

- TAU MRNet Back-End
  - Two streams
    - data
    - control
  - Packetization
  - Non-blocking receive for control
Components and Data/Control Flow

- Components
  - Back-End (BE) adapter
  - Filters
    - reduction
    - distributed Analysis
    - up / down stream
  - Front-End (FE)
    - unpacks, interprets, stores

- Data path
  - Reverse reduction path

- Control path
  - Forward multicast path
Monitor Instantiation Problem

- How to co-allocate nodes (cores) for monitoring?
  - Monitor performs transport and analysis
  - General problem when utilizing additional resources
    - tool specific

- Important especially in non-interactive (batch) environments
  - Set of allocated nodes not known a priori
  - Multi-step setup procedures difficult / awkward
  - Environments vary widely
    - command-line / script interfaces and capabilities

- Need an approach ...
  - To instantiate application, transport, and front-end
  - ... that is independent of batch environment
  - ... that requires no changes to application
Monitor Instantiation: Required Steps
Monitor Instantiation: Required Steps

- Calculate (monitor + application) and request total resources
- Apportion resources based on role (monitor, application)
- Construct transport topology (Front-End, Filters)
- Help Back-Ends discover and connect to parents
- Do so transparently to application
- Do so transparently to queue manager and scheduler
Monitor Instantiation: Required Steps

- Calculate (monitor + application) and request total resources
- Apportion resources based on role (monitor, application)
- Construct transport topology (Front-End, Filters)
- Help Back-Ends discover and connect to parents
- Do so transparently to application
- Do so transparently to queue manager and scheduler

- Total resource calculation easy
  - Do so manually or through script (based on FanOut)
- MRNet already does transport topology construction and filter instantiation for us
Transparent Monitor Instantiation

- Solution for MPI Applications
- Based on interception of MPI Calls
  - PMPI interface
- Separate roles
  - Tree: Rank-0 and Ranks 1..k-1
  - Application: Ranks k..N+k
- Three parts to method:
  - Initialization
  - Application execution
  - Finalization

Front-End  
Intermediate Ranks 1..k-1  
Application Ranks k..N+k
**Transparent Monitor: Initialization**

- `COMM_WORLD` split based on *role* of rank
- Intermediate nodes register with ToM on Rank-0 using MPI
- Rank-0 uses MRNet API to instantiate transport
- Rank-0 MPI *bcasts* tree info to application BEs to join
Transparent Monitor: Application Execution

- Application on Back-Ends proceeds normally
- MPI calls on COMM_WORLD are converted
  - Intercepted to use *userComm*
- MPI jobs on Ranks 0 to k-1 idle
- MRNet processes active

<table>
<thead>
<tr>
<th>Rank 0</th>
<th>Tree Ranks 1 to (K-1)</th>
<th>Application Ranks</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAU MPI_Init() Wrapper</td>
<td>TAU MPI_Init() Wrapper</td>
<td>Other TAU MPI Wrapper</td>
</tr>
<tr>
<td>S0 : Call PMI_Init()</td>
<td>S3 : Call MPI_Irecv(fini)</td>
<td>S0 : if(comm ==</td>
</tr>
<tr>
<td>S1 : Split Tree/App Comm</td>
<td>S4 : sleep 5</td>
<td>MPI_COMM_WORLD)</td>
</tr>
<tr>
<td>S2 : Send Hostname to Rank0</td>
<td>S5 : if(MPI_Test() == false)</td>
<td>comm = userComm;</td>
</tr>
<tr>
<td>S3 : Return</td>
<td>goto S4</td>
<td>S1 : Call PMI routine</td>
</tr>
<tr>
<td>S4 : Return</td>
<td></td>
<td>S3 : return</td>
</tr>
</tbody>
</table>
Transparent Monitor: Finalization

- Application ranks call `MPI_Finalize`
- ToM tree destruction initiated
- Eventually Ranks 0..k-1 also call `MPI_Finalize` to end job.

<table>
<thead>
<tr>
<th>Rank 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAU MPI_Init() Wrapper</td>
</tr>
<tr>
<td>S9 : Send fini to Tree-ranks</td>
</tr>
<tr>
<td>S10 : Call PMPI_Finalize()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tree Ranks 1 to (K-1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAU MPI_Init() Wrapper</td>
</tr>
<tr>
<td>S3 : Call MPI_Irecv(fini)</td>
</tr>
<tr>
<td>S4 : sleep 5</td>
</tr>
<tr>
<td>S5 : if(MPI_Test() == false) goto S4</td>
</tr>
<tr>
<td>S6 : Call PMPI_Finalize()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Application Ranks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calls MPI_Finalize()</td>
</tr>
<tr>
<td>{ Send ToM FIN to FE</td>
</tr>
<tr>
<td>Call PMPI_Finalize()</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>
ToM Filters

- Ideally there would be no need for filtering
  - Retrieve and store *all* performance data provided
  - Acceptability depends on performance monitor use

- High application perturbation, transport and storage costs
  - Need to trade-off queried performance data granularity
  - Which events, time intervals, application ranks?

- Reduce performance data as it flows through transport
  - Distribute Front-End analysis out to intermediate filters

- Three filtering schemes developed for ToM
  - Each builds upon and extends previous
  - Progressively provide increased temporal and spatial detail

- Upstream and downstream filters
Summary Statistics Filter

- Global summary statistics
  - Across ranks (N)
  - For each profile event
  - N parallel profiles reduced to E event statistics
  - Functions:
    - mean, min, max
    - standard deviation

- Single phase (A)
  - Up-stream filter

- Intermediate node
  - Summarize children’s data
  - Recursively arrive at FE
Example: Summary Statistics Filter

FLASH Sod 2D | N=1024 | Allreduce

Full profiles are generated (N*300 profiles)
Example: Summary Statistics Filter

FLASH Sod 2D | N=1024 | Allreduce

Sudden spike at iteration 100

Full profiles are generated (N*300 profiles)
Example: Summary Statistics Filter

FLASH Sod 2D | N=1024 | Allreduce

Temporal information only
Spatial information lacking

Sudden spike at iteration 100

Full profiles are generated (N*300 profiles)
Histogram Filter

- Maintain specified level of spatial information (# bins)
- Accurate histogram needs global min/max (range)
- Global unknown below root
- Three Phase (A, B, C)
  - A: Push up min/max; buffer
  - B: Push min/max to DSF
  - C: Histogram recursively
- Model
  - Non-blocking, pipelined
  - Data parallel

KEY

- FE : Front End
- BE : Back End
- USF : UpStream Filter
- DSF : DownStream Filter
Example: Histogram Filter

FLASH Sod 2D | N=1024 | Allreduce

Hist Filter
- 1024 MPI Ranks
- FLASH 2D Sod
- ToM Fanout=8
- Offload performance every iteration

No. of Ranks

Application Iteration #

Total Event Runtime (secs)
Example: Histogram Filter

FLASH Sod 2D | N=1024 | Allreduce

Hist Filter
- 1024 MPI Ranks
- FLASH 2D Sod
- ToM Fanout=8
- Offload performance every iteration

Temporal information
Spike at Iteration 100
Example: Histogram Filter

FLASH Sod 2D | N=1024 | Allreduce

Hist Filter
- 1024 MPI Ranks
- FLASH 2D Sod
- ToM Fanout=8
- Offload performance every iteration

Temporal information
Spike at Iteration 100

Spatial information
Unevenness of across ranks
Evolution of unevenness over iterations

No. of Ranks
What was the cause for the unevenness in last example?
Are there “classes” of ranks performing specific roles?
Can we identify them from the performance profile?

Definition of class

- *Class-id*: hash of concatenated event-names
- Ranks with same *class-id* belong to same class
- Application-specific or tailored to observer’s wishes
- *Class-id* generated based on call-depth or only for MPI events

Histograms generated within class

- Output: set of histograms per-event, one for each class

More detail than simple histograms

- Trade-off detail from classification scheme against the costs

---

**Classified Histogram Filter**

STHEC 2008, Kos, Greece  
TAUoverMRNet (ToM)  
23
Example: Uintah (Hot Blob)

- Uintah Computational Framework UCF (University of Utah)
- Computational Fluid Dynamics (CFD) code
  - 3 dimensional validation problem
- Spatial domain decomposition
  - Patch - unit of partitioning
  - 8 outer patches at AMR level 0
  - Inner cubes selected at level 1

- TAU instrumentation strategy
  - Map low-level performance to patches
  - Mapping expressed through event-name
    - Patch index + AMR Level 0 ➞ “Patch 2 -> 0”
Example: Uintah (Hot Blob)

Classification scheme

- *Default*: all event names used for class-id
- *Patch Only*: only high-level Patch events used
- *AMR L0 Patch Only*: only "* -> 0" type events
- *MPI Only*: only MPI events

Depending on scheme ...

- Different number of classes generated
- Different reduction ractor = unreduced bytes / reduced bytes

Classification scheme allows control of trade-off

- Savings from reduction
- Performance detail
Example: Classified Histogram Filter | Uintah

64 MPI Ranks | # Bins = 5
Characterization

- Performance monitoring parameters
  - Frequency of interaction
  - Performance data granularity and size
  - # of processors

- In what circumstances is doing reduction beneficial?
  - No free lunch - requires extra work and resources

- Characterization methodology to optimize trade-off
  - Monitoring overhead
  - Additional resource assignment

- Compare reduced (ToM Hist) vs. non-reduced (ToM) runs
  - Amount of data is usually less (that’s the point)
  - Need a better metric
Characterization: Metric, Benchmark

- Average time for global offload

```c
  time = get_time();
  for(i=0; i<iterations; i++) {
      work(usecs);
      TAU_DB_DUMP();
      MPI_Barrier();
  }
  tot.time = get_time()-time;
  tot.dump.time = time - work_time - barrier_time;
  dump.time = tot.dump.time/iterations;
```

- Increasing offload rate (function of usecs above)
  - Overtakes service rate of ToM (and underlying system)
  - Eventually lead to queueing and blocked send() call
  - Reflected in the average time for offload (dump.time)

- Stress test of ToM
Characterization: $N=64$, $FO=8$

Benchmark Performance (msecs)

- ToM
- ToM Reduce

Profile Period
Fan-Out

usec

STHEC 2008, Kos, Greece

TAUoverMRNet (ToM)
Characterization: $N=64$, $FO=8$

Benchmark Performance (msecs)

ToM
ToM Reduce

Profile Period
Fan-Out

usec

ToM-Reduce out performs ToM
Characterization: \( N=64, \ FO=8 \)

Benchmark Performance (msecs)

- ToM
- ToM Reduce

Load overtaking service rate

ToM-Reduce outperforms ToM
Characterization: \( N=64, \ FO=8 \)

Benchmark Performance (msecs)

Load overtaking service rate

Cost of reduction overtakes savings

ToM-Reduce outperforms ToM
Characterization: Large $N$ (256, 512)

Graph showing benchmark performance (msecs) vs. profile period for different configurations:
- N=512 FO=8
- N=512 FO=8 Reduce
- N=256 FO=16
- N=256 FO=16 Reduce

Profile Period

Benchmark Performance (msecs)
At relatively large $N$
- ToM-Reduce significantly out-performs ToM
- Even at much larger fan-outs than before
Conclusion and Future Work

- High *return on investment* from additional resources
  - Fan-out of 64 is only 1.5% extra resources
- Have only scratched the surface
  - Interesting distributed performance analysis to explore
  - Support of feedback into application
    - based on performance dynamics
  - Load-balancing and resource (re-)allocation
- Interest in experimentation on very large scales
  - Looking for candidate applications
- Would like to hookup system to real-time visualizations
Credits

- University of Oregon
  - Aroon Nataraj
  - Alan Morris
  - Allen D. Malony
  - TAU group members

- University of Wisconsin
  - Dorian C. Arnold
  - Michael Brim
  - Barton P. Miller