Performance Modeling and Analysis

Striking a Balance between Resolution and Insight

Jeffrey Vetter
Future Technologies Team Leader
Oak Ridge National Laboratory
vetterjs@ornl.gov

The submitted manuscript has been authored by a contractor of the U.S. Government under Contract No. DE-AC05-00OR22725. Accordingly, the U.S. Government retains a non-exclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.

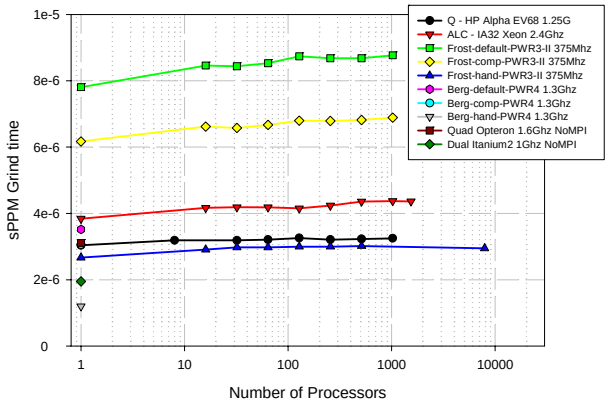
ornl

Highlights

- ORNL has been at the forefront of supercomputing for over 15 years
  - New systems will continue the trend of massively parallel computing
- Performance analysis, modeling, and benchmarking play an important role in the effective use of these systems
- Existing and planned systems stress current techniques
- We need new strategies for dealing with performance analysis and modeling of massively scaled parallel computing
- We are exploring new solutions to various problems including
  - Reducing data volume with statistical profiling of message passing applications
  - Reducing data analysis effort with machine learning and multivariate statistics
  - Automatically extracting performance models from applications

sPPM Across Architectures

- We need more detailed information at scale



Many Factors Influence the Value of HPC Systems

Performance Modeling and Analysis Project Impacts Platforms and Applications

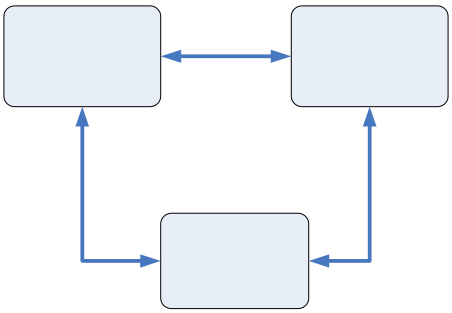
- Use measurement, modeling, and simulation to understand performance of important applications across a range of platforms
- Apply these techniques at every phase of platform and application lifecycles
  - Design
  - Procurement
  - Installation
  - Optimization

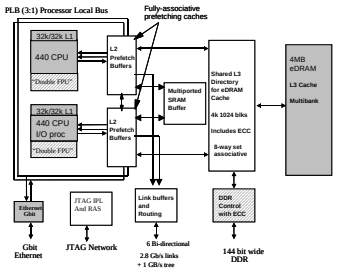
Scalability of Tools is Critical!

- Performance analysis will become increasingly important as processor counts grow
  - 'Scaling performance problems only manifest themselves at scale.' ☺
- Several levels of performance analysis technology do not scale well

Concurrency	O(100)	O(1,000)	O(10,000)	O(100,000)
Instrumentation	OK	OK	OK	OK
Instrumentation management	OK	Hurdle	Hurdle /Barrier	Barrier
Data management	OK	Hurdle	Hurdle /Barrier	Barrier
Data interpretation	Hurdle	Barrier	Barrier	Barrier

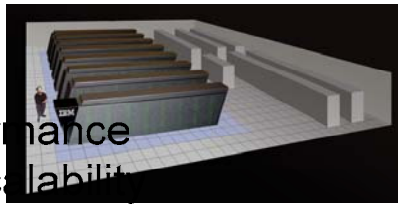
Measurement and Simulation Techniques Require Users & Designers to Balance Conflicting Criteria





- Performance, Scalability
  - minimize latency and turnaround time
- Flexibility
  - maximize scope
- Detail
  - minimize risk

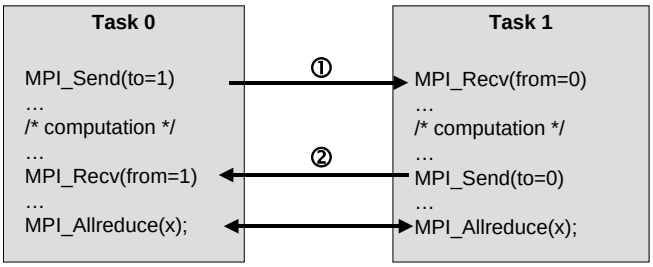
Performance and Scalability



Flexibility

Explicit Message Passing is the Primary Programming Model

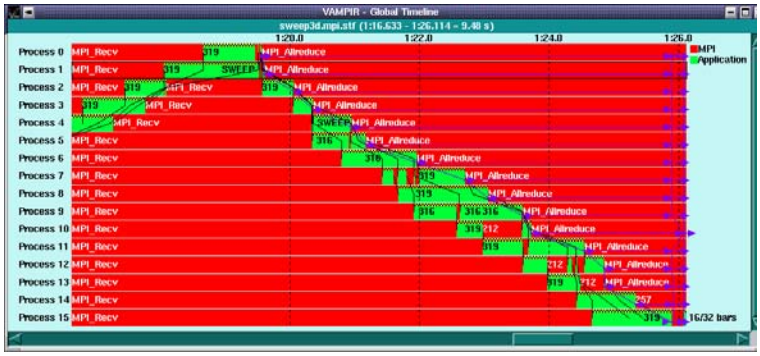
- Most scalable applications use explicit message passing with MPI for coarse grained concurrency
  - Explicit management of communication and synchronization
  - Portability, scalability, tool availability, etc.



Explicit message passing requires that the application manage performance.

Traditional Performance Analysis of MPI

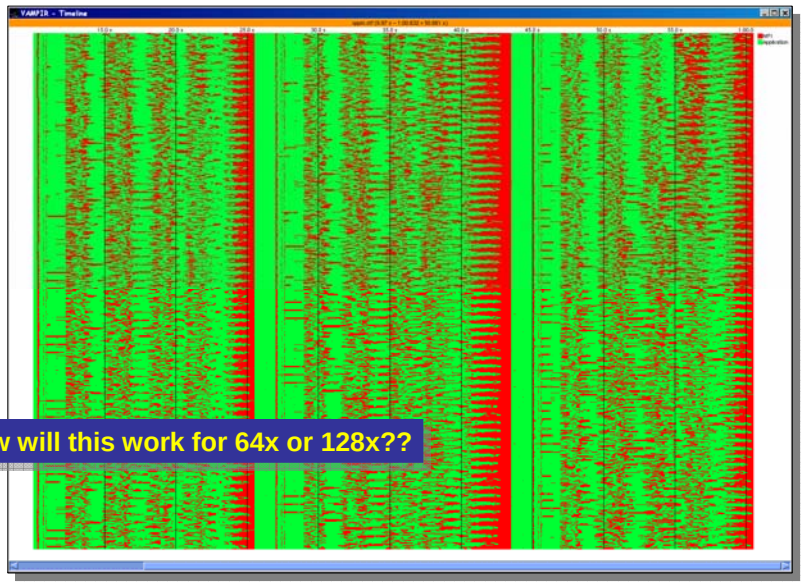
- MPI's profiling layer promotes construction and portability of tools
- Most MPI tools use tracing
  - Produces very detailed information about communication activity



Example Scaling Issues w/ Public Tool: SMG2000 Experiment Results with Tracing

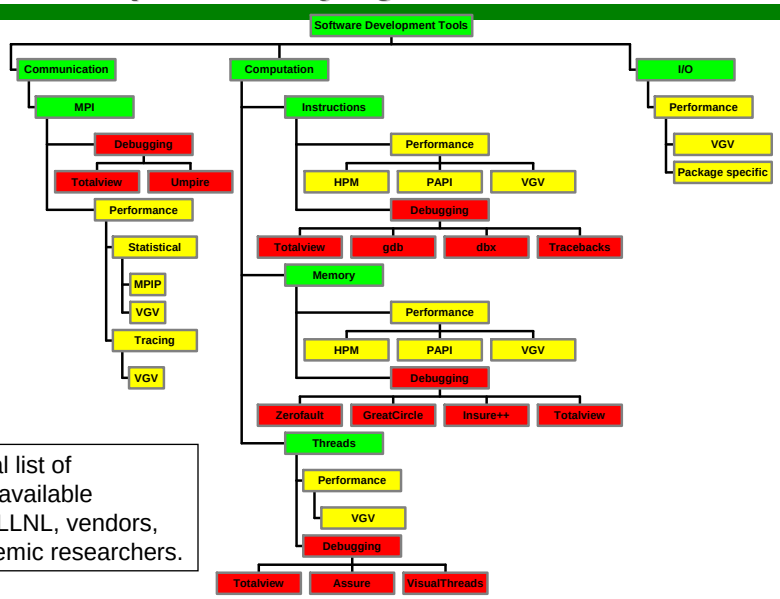
- 48 task experiment
  - ~26 seconds
  - ~16,000 messages per task per solve
  - ~768,000 messages per solve
- Tracing
  - 225MB global tracefile
  - 154% increase in execution time
    - Excludes initialization and finalization including merging the tracefiles
- In real use
  - Applications invoke linear solvers frequently
  - SMG scales to over 4,000 tasks
  - Tracing has prohibitive costs

sPPM at 1024 tasks with VGV



How will this work for 64x or 128x??

Assortment of Tools to Address Different Topics at Varying Levels of Detail



Partial list of tools available from LLNL, vendors, academic researchers.

Scale of Systems Demands New Analysis and Prediction Strategies

- Instrumentation
  - How do we limit the amount of data generated?
  - How do we maintain correct performance observations?
    - That is, small perturbation and overhead
- Data management and analysis
  - How do we gain insight from this massive performance dataset?
- Interpretation and prediction of performance data
  - How to we make accurate and efficient predictions about application performance?

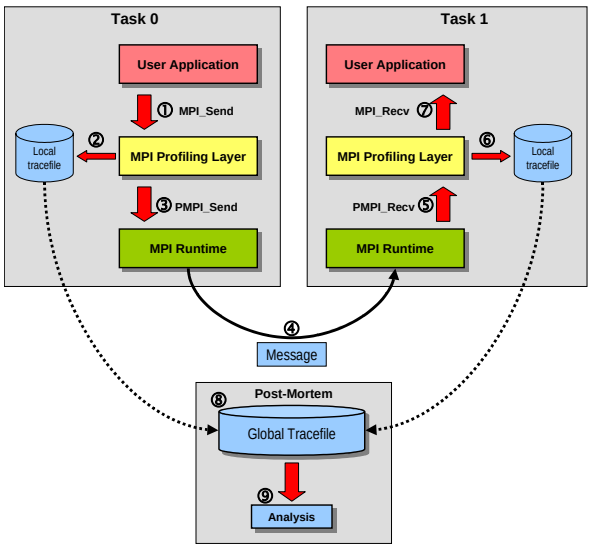
Dynamic statistical sampling of MPI communication

MPI Performance with Sampling and Runtime Analysis

- Sampling and runtime analysis are widespread
  - E.g., Instruction sampling: DPCI, gprof
- Extend this model to message passing applications for profiling communication activity
  - Eliminate management of huge data files
  - Reduce perturbation on target application including pathological measurement situations
- With this additional capability, we can
  - Decide which messages to observe
  - Analyze data at runtime, jettisoning raw data immediately if desired

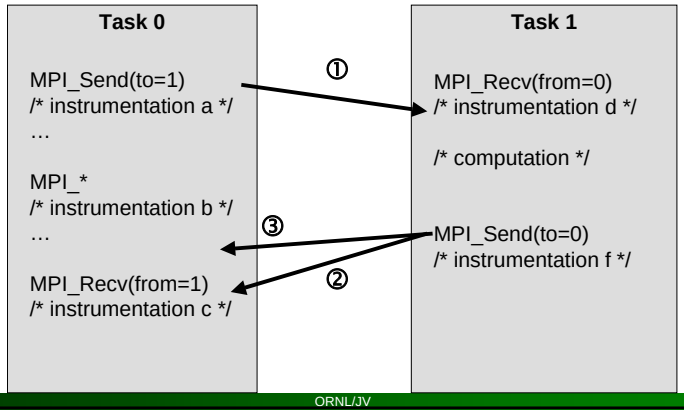
Traditional Techniques Force Post-mortem Analysis

- Tracer stores performance information to task local tracefiles
- Task tracefiles merged at finalization
  - Reconcile sends with receives
  - Align collective operations
- Performance analysis performed on global logfile post-mortem



Perturbation can Change Observations

- Additional instrumentation can alter the behavior of the performance analysis
- Traditional techniques introduce considerable instrumentation everywhere



Observations and Implications with Traditional Techniques

- Tracing provides very detailed information necessary for understanding certain performance phenomena
- Data management issues with collection and management of local and global tracefiles
  - As machines scale up, so does the management problem
- Mandatory perturbation
  - May introduce perturbation that changes performance characteristics
- Many metrics are not available at runtime
  - Limited analysis; must collect data conservatively
  - E.g., Message latency

Design Alternatives

1: Send an extra message after each sampled message containing additional information

NO: additional messages, race conditions

2: Exchange performance data at collective operations

NO: high overhead, local logging, complex communicators

3: Use derived types to append performance data to messages

NO: derived types are slow, msg envelope does not contain datatype

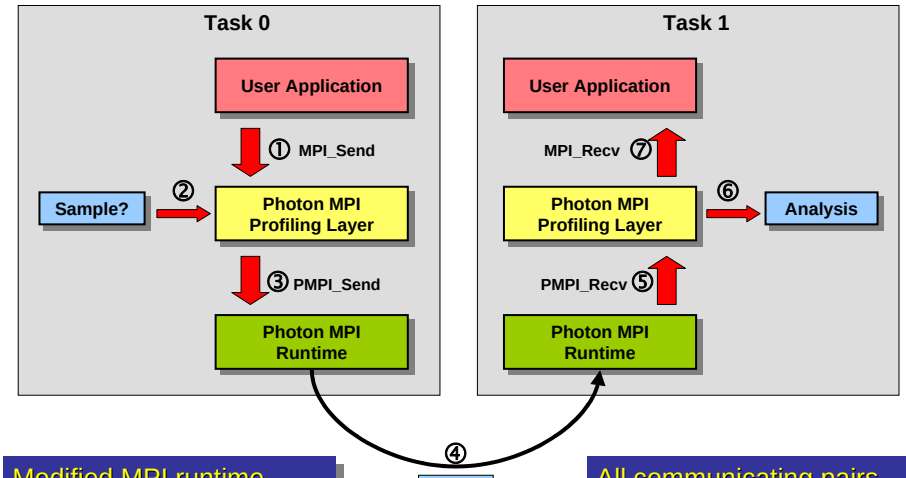
4: Modify MPI implementation to carry additional information

- Include additional data in each message header
  - Originating timestamp, Source code location tuple
- Augment message protocols to pack/unpack this data
- Sacrifice portability

New Solution: Enable Message Sampling and Runtime Analysis

- Sampling and runtime analysis are widespread
  - E.g., Instruction sampling: DCPI, gprof
- Extend this model to message passing applications for profiling communication activity
- With this additional capability, we can
  - Decide which messages (including their corresponding send and receive operations) to observe
  - Analyze data at runtime, jettisoning raw data immediately if desired

Photon MPI Design and Distributed Operation

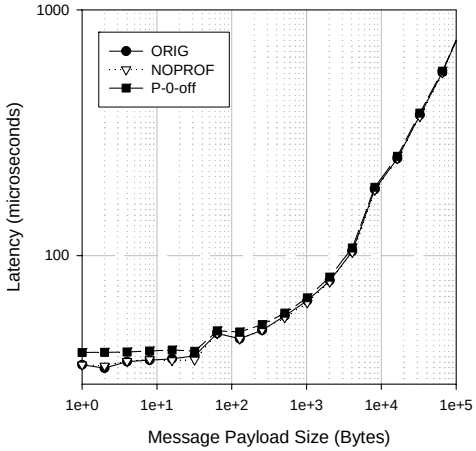


Modified MPI runtime carries tiny amount of data with each message.

All communicating pairs in the system sample and analyze independently.

Photon Overhead Measurements Demonstrate Low Cost

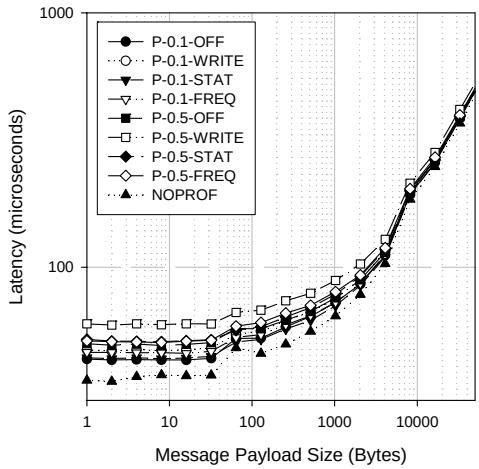
- Overhead of Photon modifications on MPI operation
  - Must be low to allow changes to production MPI
- Virtually imperceptible latency difference for NOPROF over ORIG
- Installation of disabled profiling layer impacts latency for messages below 1K bytes
  - ~3 microsecs
- BW did not vary noticeably



Analysis Overhead has Variable Cost – as necessary

STAT: update simple stats
FREQ: update distribution
WRITE: log event to file

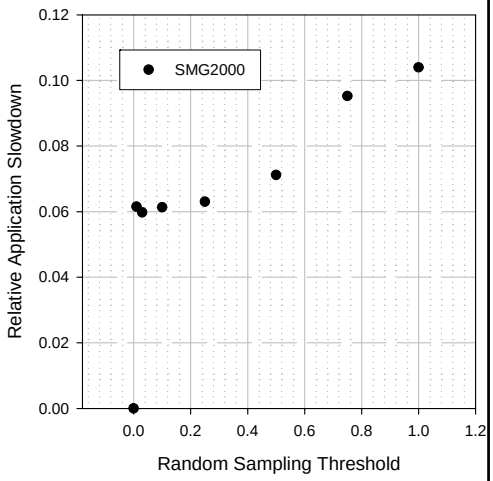
At 4 bytes,
- P-0.5-WRITE is highest at 59.9 us
- FREQ, STAT are much lower
  - ~40-45 us range



- Sampling rate can clearly compensate for expensive analysis

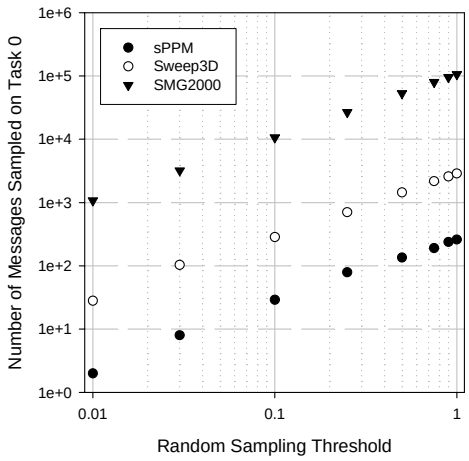
Impact of Sampling Rate on Application Runtime is a Major Improvement

- Overall application runtime is controllable and small
- sPPM and Sweep3D were imperceptible for sampling rates of 50% or less
  - Less than 1% for all cases
- SMG2000 sends many more messages
  - Perturbation of solve phase is ~6% slowdown for sampling less than 50%
- This is much better than tracing's 154% slowdown!



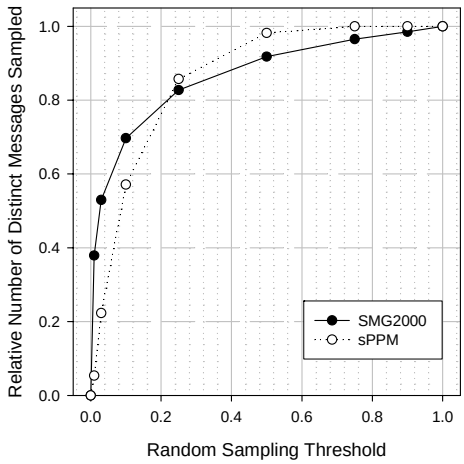
Number of Messages Sampled Scales as Expected

- Random sampling allows control of the number of messages sampled
- Our applications span a wide rage of message characteristics
  - SMG2000 sends ~100,000
  - SPPM sends ~260



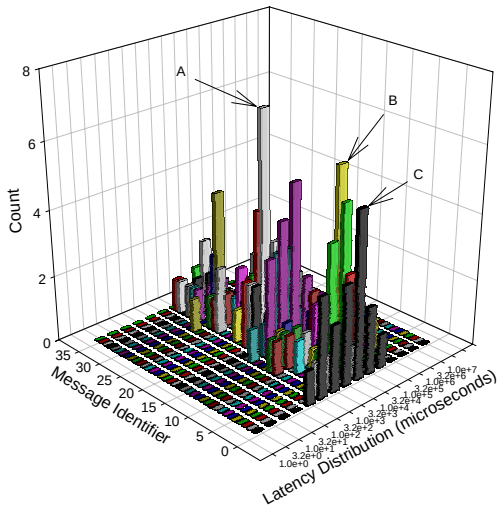
Sample Space of Message Population

- As the global random sampling threshold increases
  - Coverage increases dramatically



One Possible Runtime Analysis: Example Frequency Distribution

- Message latency now available at runtime
- Update a frequency distribution for latency and message identifier
- Inexpensive
- Raw data discarded after tabl update
- Combine w/ topology information to infer performance problems



Advantages of Message Sampling

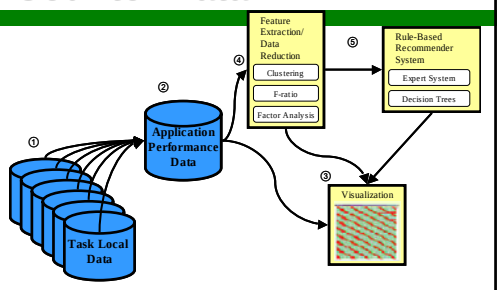
- Reduced overhead
  - Only a subset of operations are sampled, so overhead is reduced on overall application
- Improved accuracy
  - Instrumentation impact can be randomized
  - Fewer perturbations due to instrumentation and data management
- Reduced data volume
  - Record only a subset of operations
  - Analyze data at runtime
- Frugal modifications to MPI runtime have negligible performance implications
  - Separation of sampling and analysis strategies from MPI implementation allows efficiency and flexibility
- Sampling methods
  - Prefer random sampling, need to improve overhead
- Works for all protocols and operations

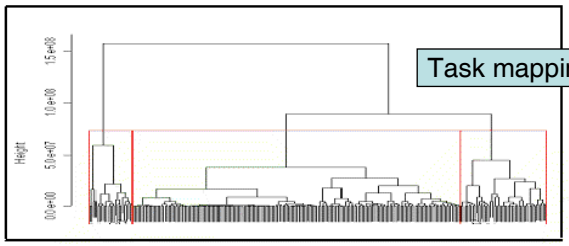
ORNL is Actively Researching and Developing New Productivity Tools

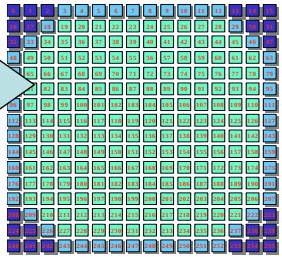
- Research is funded byOffice of Science, DARPA, DoD
- Automated analysis of performance data to pinpoint performance problems with machine learning and multivariate statistical analysis
- Automated techniques for performance modeling and application characterization
- Sampling MPI P2P messages to reduce instrumentation and data overhead
- Umpire automatically identifies errors in MPI applications
- Performance assertions to allow expression of performance expectations in evolving applications
- Many others…

Multivariate Statistical Analysis of Hardware Counter Data

- Hardware counters produce huge amounts of data on large systems
- Multivariate statistical techniques help distill important features
- Clustering, Factor analysis, PCA







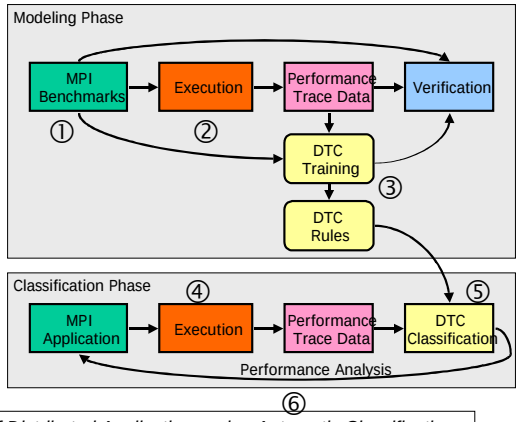
D.H. Ahn and J.S. Vetter, "Scalable Analysis Techniques for Microprocessor Performance Counter Metrics," Proc. SC 2002, 2002.

Automatic Classification for MPI Trace Analysis

- Use decision tree classification (a supervised learning technique) to classify application's messages automatically
- Compare an application's message operations to 'normal' communication for a particular MPI configuration

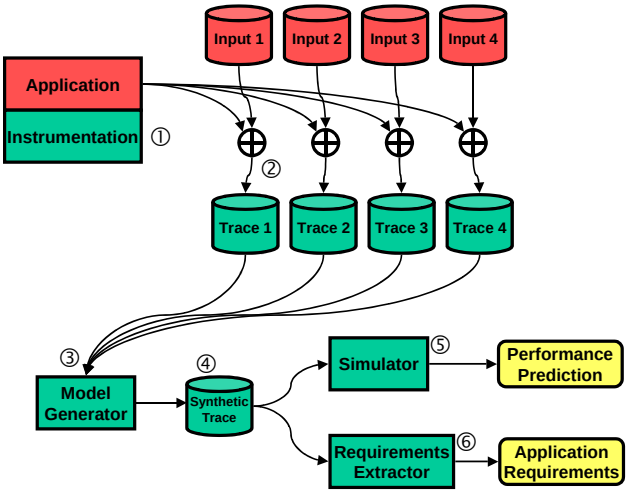
➢Modeling Phase (once)
- –Use benchmarks to generate decision tree
- –Both efficient and inefficient

➢Classification Phase (many)
- –Execute application
- –Analyze application trace with classifier based on decision tree



J.S. Vetter (2000). *Performance Analysis of Distributed Applications using Automatic Classification of Communication Inefficiencies*. Proc. ACM Int'l Conf. Supercomputing (ICS).

Sequoia Toolkit for Application Modeling and Characterization



J.S. Vetter and F. Mueller, "Communication Characteristics of Large-Scale Scientific Applications for Contemporary Cluster Architectures," Proc. International Parallel and Distributed Processing Symposium (IPDPS), 2002. [Best Paper Award]

Conclusions

- Performance analysis and modeling require complex choices among
  - Level of detail
  - Flexibility
  - Scalability and performance
- We need new strategies for performance analysis at massive scale (e.g., 131k processors)
- Dynamic statistical profiling of MPI applications is one approach to enable performance analysis at massive scale
- ORNL is working on other new strategies to address these problems.

J.S. Vetter (2002), *Dynamic Statistical Profiling of Communication Activity in Distributed Applications*. Proc. SIGMETRICS: International Conf. Measurement and Modeling of Computer Systems