Challenges in Scalability of Performance Tools for Petaflops Systems

Patricia J. Teller, Ph.D.
University of Texas-El Paso
Department of Computer Science
pteller@cs.utep.edu

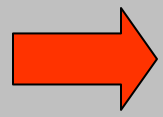
Outline

- Motivation – Why do we need performance tuning tools?
- Foundational Techniques
  - Data Collection
  - Facilitating Data Analysis
- Tradeoffs in terms of Scalability
- Challenges in the Peta(fl)ops Era
  - Data Reduction
  - Time to Insight
- Proposal

Motivation

- Gap between peak and achieved performance in high-end systems
- Tools needed to narrow the gap by
  - Tuning performance (my point of view)
    - identifying performance bottlenecks
    - determining cause of bottlenecks
  - Collecting performance data
  - Identifying relevant information
  - Facilitating analysis of relevant information

Tuning Performance



- **Collecting Data**
  - Foundational Techniques
    - What?
      - Level of detail
    - How?
    - When?
      - Level of overhead (perturbation)
- **Facilitating Data Analysis**

Collecting Data

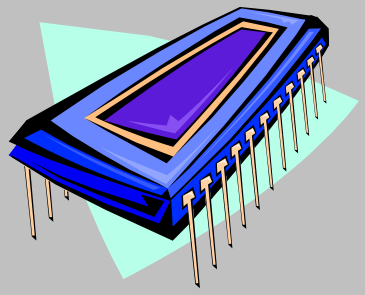
Application Profiles - 1

- Time (e.g., time profiles)
  - Relative execution times
  - Candidates for further analysis
- Time stamped events
  - More detail
    - Subroutine entry/exit information
    - MPI library calls
    - OpenMP parallel region invocations

Collecting Data

Process/Thread Profiles - 1

- Event counts (on-processor chip hardware performance counters)
  - More detail
    - Process/thread level
    - Set of events – concurrent monitoring via multiple performance counters
    - No spatial or temporal information



Collecting Data

Process/Thread Profiles - 2

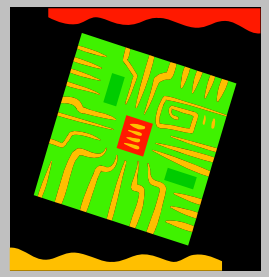
- **Processor event counts**
  - microarchitecture event counts, e.g.,
    - cycles
    - pipeline stalls
    - branch mispredictions
  - instruction event counts, e.g.,
    - floating-point instructions committed
    - loads executed

Collecting Data

Process/Thread Profiles - 3

– **Memory Event Counts**
  - L2 data cache hits
  - TLB misses
  - Cache-consistency related events

– Operating system events, e.g.,
  - L1 instruction cache hits
  - TLB misses



Collecting Data

Process/Thread Profiles - 4

- **Granularity**
  - **smaller granularity ⇒ more detail**
  - **more detail ⇒ more data**
    - whole program
    - time interval
    - phase
    - function
    - loop
    - basic block
    - instruction

Collecting Data

Process/Thread Profiles - 5

- **Interfaces to on-processor performance counters**
  - Manufacturer-specific interfaces
    - Different number of counters
    - Different number and types of events
    - No standard
  - PAPI – cross platform API
    - Common set of events

Collecting Data

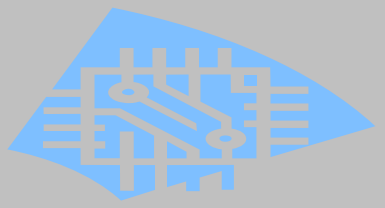
Instrumentation for Profiling

- **Application instrumentation**
  - Automatic or user-guided
  - Graphical (SvPablo, TAU)
  - Static Profiling: records measurements at each invocation of a probe point
    - Data set size increases with number of probe points

Collecting Data

System-wide Profiles - 1

- **Event counts (off-processor hardware performance counters)**
  - On memory chips
    - Contention-related events, e.g., network output blocks (Cray X1 (on main memory) M-chip)
    - Cache evictions, writebacks (Cray X1 (on cache) E-chip)
  - On network interface cards
  - On network switches (e.g., on Myrinet switches)
    - Good packets received on this port
    - Timeouts
    - Bad routes



Collecting Data

System-wide Profiles - 2

- **Enough detail?**
  - If only application in execution
- **Interfaces to off-processor performance counters**
  - Different manufacturer-specific interfaces
    - No standard
  - PAPI team working on interface to counters on Myricom network switches

Collecting Data with more Detail

Event Traces

- **Event records**
  - **Spatial and temporal information**
    - Memory address (instruction/data)
    - Timestamp
  - **Communication/synchronization events**
    - MPI event trace records
      - MPI point-to-point and collective communication
    - OpenMP parallelism changes
    - Parallel constructs
    - Synchronization events

Collecting Data with more Detail and Larger Footprint

- Time-based application profiles
- Time-stamped application events
- Architecture-related event counts associated with processes/threads
- Architecture-related event counts that could be associated with process/threads
- Communication/synchronization events with temporal and spatial information
- Architectural (in particular, memory) events with temporal and spatial information

Performance Tuning

- **Collecting** Data
- **Facilitating** Data Analysis
  - Some foundational techniques
    - Human involvement (tool autonomy)
    - Overhead (time to insight)

Facilitating Data Analysis

- Database of profile data (TAU's PerfDBMS)
  - Facilitate data access and analysis
- Multivariate statistical analysis (Ahn & Vetter)
  - Facilitate data collection and analysis
- Pattern matching (KOJAK communication event analysis)
  - Facilitate identification of performance issues
- Visualization
  - Real-time (SvPablo); Post-mortem (KOJAK)

Facilitating Data Analysis

Multivariate Statistical Analysis - 1

- Use scatterplot/correlation matrices, Principal Component Analysis, cluster analysis (hierarchical and K-means), factor analysis to determine
  - what events to capture
  - which events cause the most variability in thread behavior
  - what aspects of the code to analyze
  - estimate the impact of code changes and detect load imbalances

Facilitating Data Analysis

Multivariate Statistical Analysis - 2

Cluster analysis (in particular) can be used to determine which threads have similar behavior

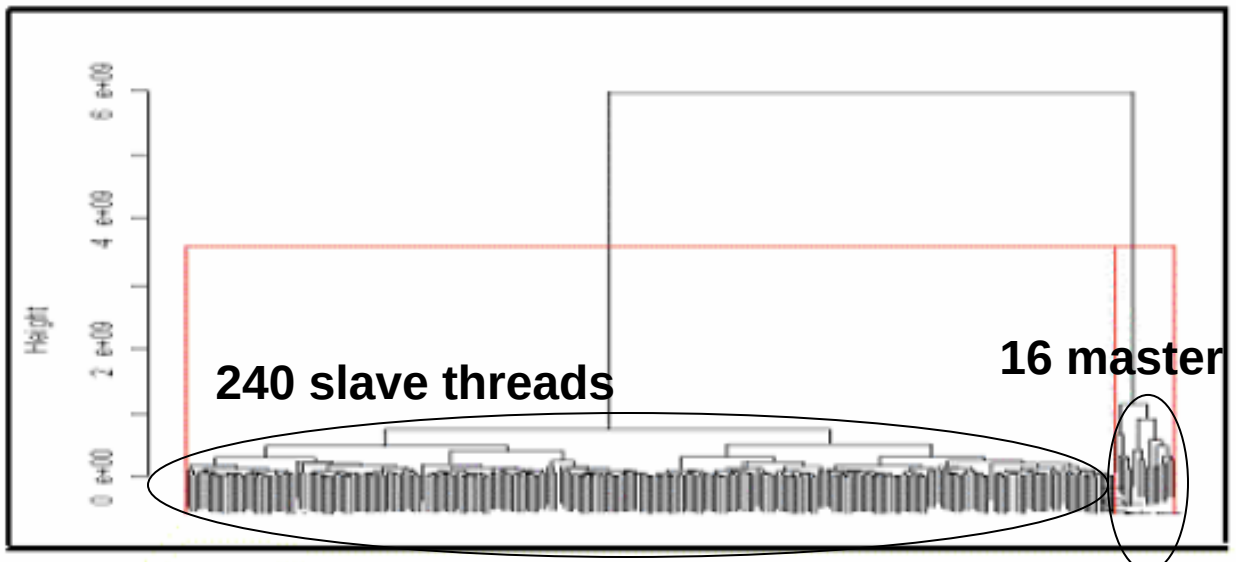


Figure 2: Dendrogram for a section of sPPM using 23 raw counter metrics (task numbers elided)

Facilitating Data Analysis

Multivariate Statistical Analysis - 3

Scatterplot /correlation matrix to identify redundant counters

High correlation – just measure one, predict others

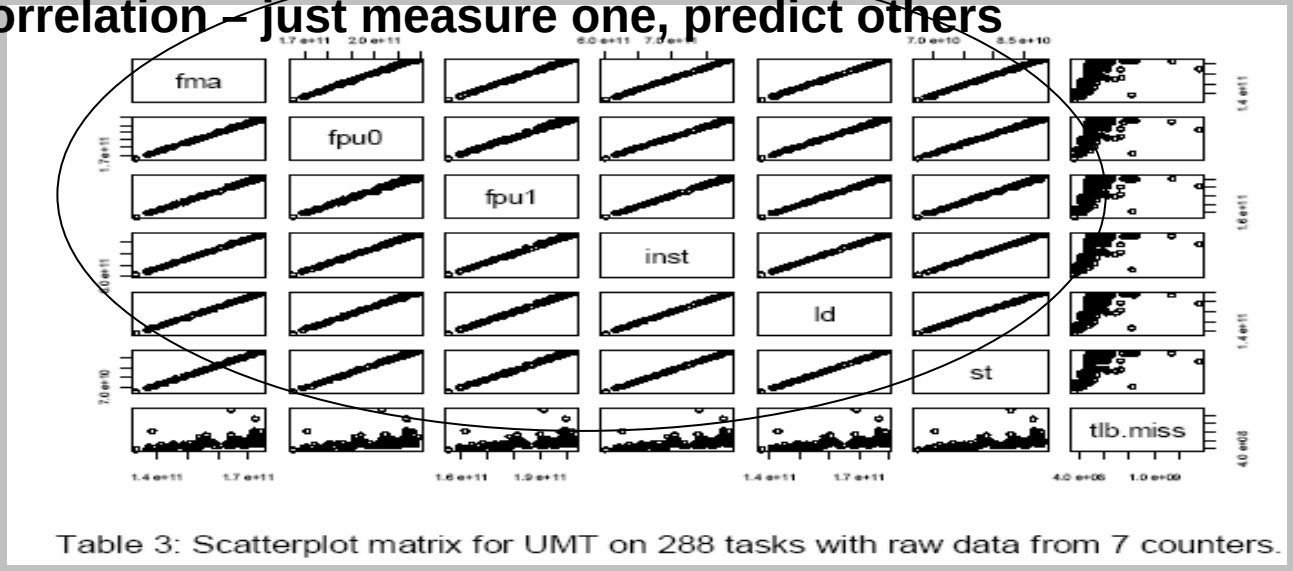


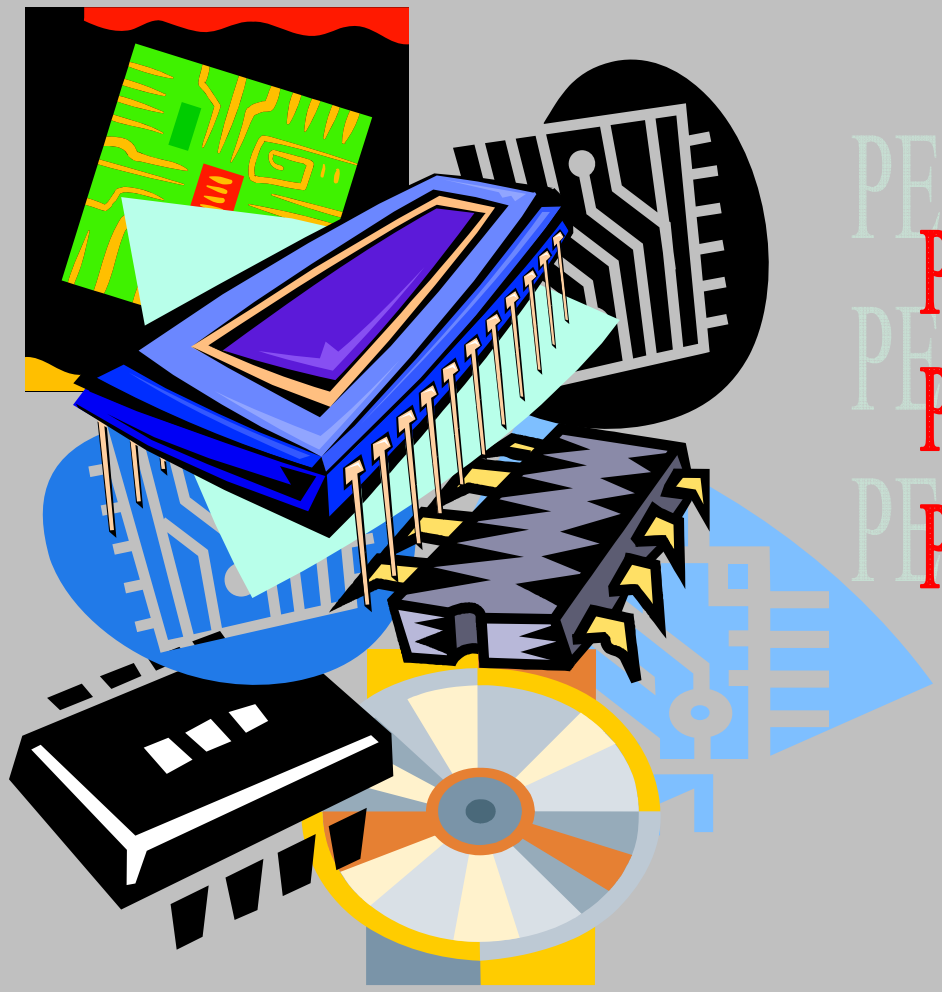
Table 3: Scatterplot matrix for UMT on 288 tasks with raw data from 7 counters.

Facilitating Data Analysis

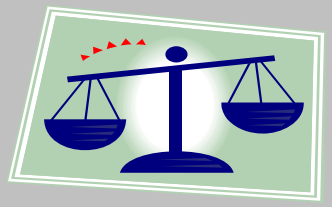
Pattern Matching

- KOJAK: searches communication event traces for execution patterns that indicate inefficient performance behavior
- UTEP and U. of Tenn.-Knoxville (supported by DoD) are investigating whether multivariate statistical analysis can be used for this purpose as well

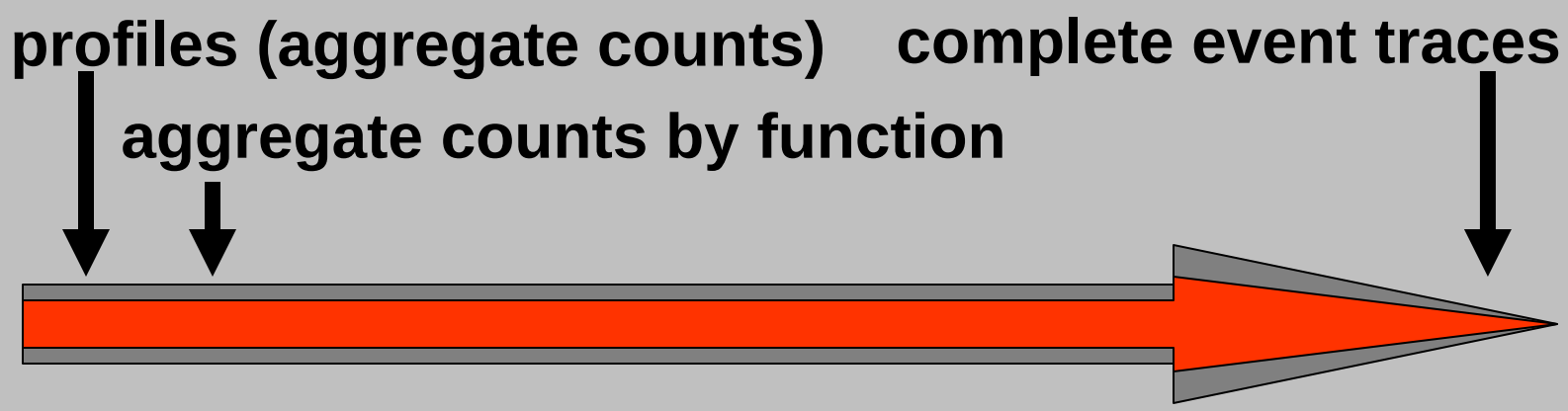
Peta(fl)ops, Exabytes, Terabytes



PERFORMANCE DATA
PERFORMANCE DATA
PERFORMANCE DATA
PERFORMANCE DATA

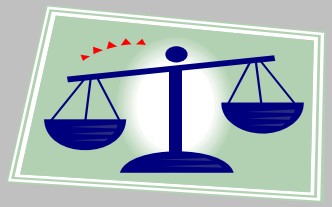


Scalability Tradeoffs – Collection/Analysis - 1



less storage
less overhead
lower level of detail

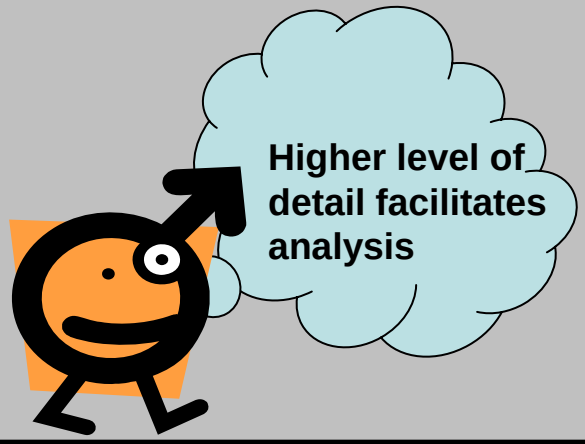
more storage
more overhead
higher level of detail



Scalability Tradeoffs – Collection/Analysis - 2

but …
with low level of detail

- Performance problems such as short delays in execution, e.g., in streaming video can be missed
  - Problems highly visible to user
  - But statistically insignificant and, thus, ignored by traditional profilers
- Intermittent bugs hard to find
  - Debuggers change the timing of events, often masking the problem



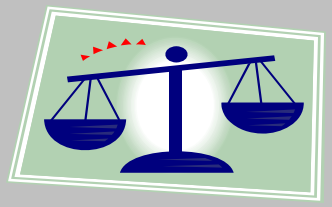
Scalability Challenges

Collection/Analysis

- **Reduction** of overhead
  - **Time for data collection (perturbation)**
  - **Storage requirements**
- **Time to insight (effective analysis)**
  - Level of detail of data (depends on nature of performance problem)
  - **Data access time**
  - **Number of data collection/analysis iterations**
  - **Number of steps of analysis refinement**
  - Level of user interaction (autonomy of tool)
  - Data visualization

Data Reduction

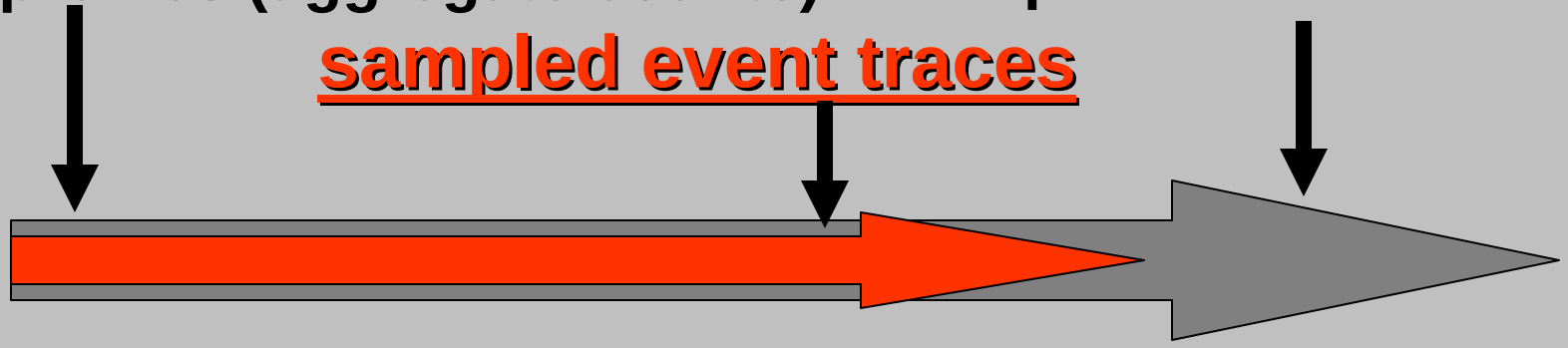
- **Application instrumentation (automatic or user-guided)**
  - **Statistical Sampling**: captures program state at regular intervals
  - **Dynamic Profiling**: collects measurements via insertion of instrumentation into application while it is executing – implemented much like a debugger
  - **Dynamic Control of Static Profiling**: dynamic (de)activation of statically inserted instrumentation



Data Reduction and Time to Insight

profiles (aggregate counts) **complete event traces**

sampled event traces



less storage
less overhead
lower level of detail

more storage
more overhead
higher level of detail

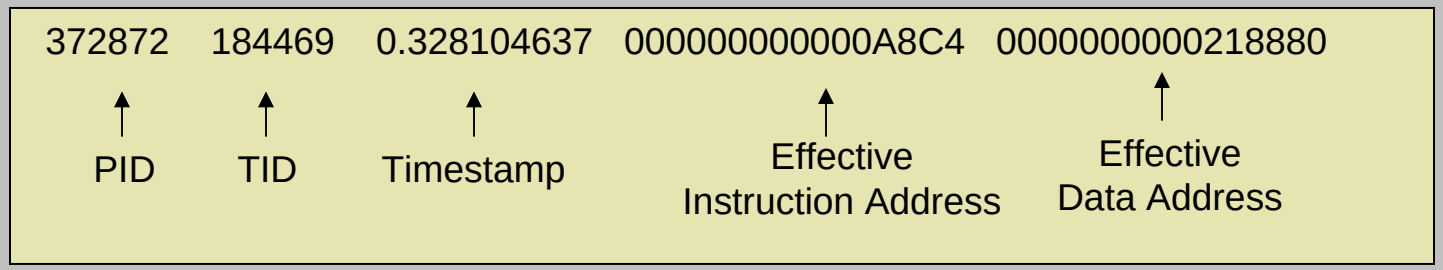
Level of detail may be higher than concommitant need for storage and accrued overhead – study in progress

Data Reduction/Time to Insight

Sampled Event Traces - 1

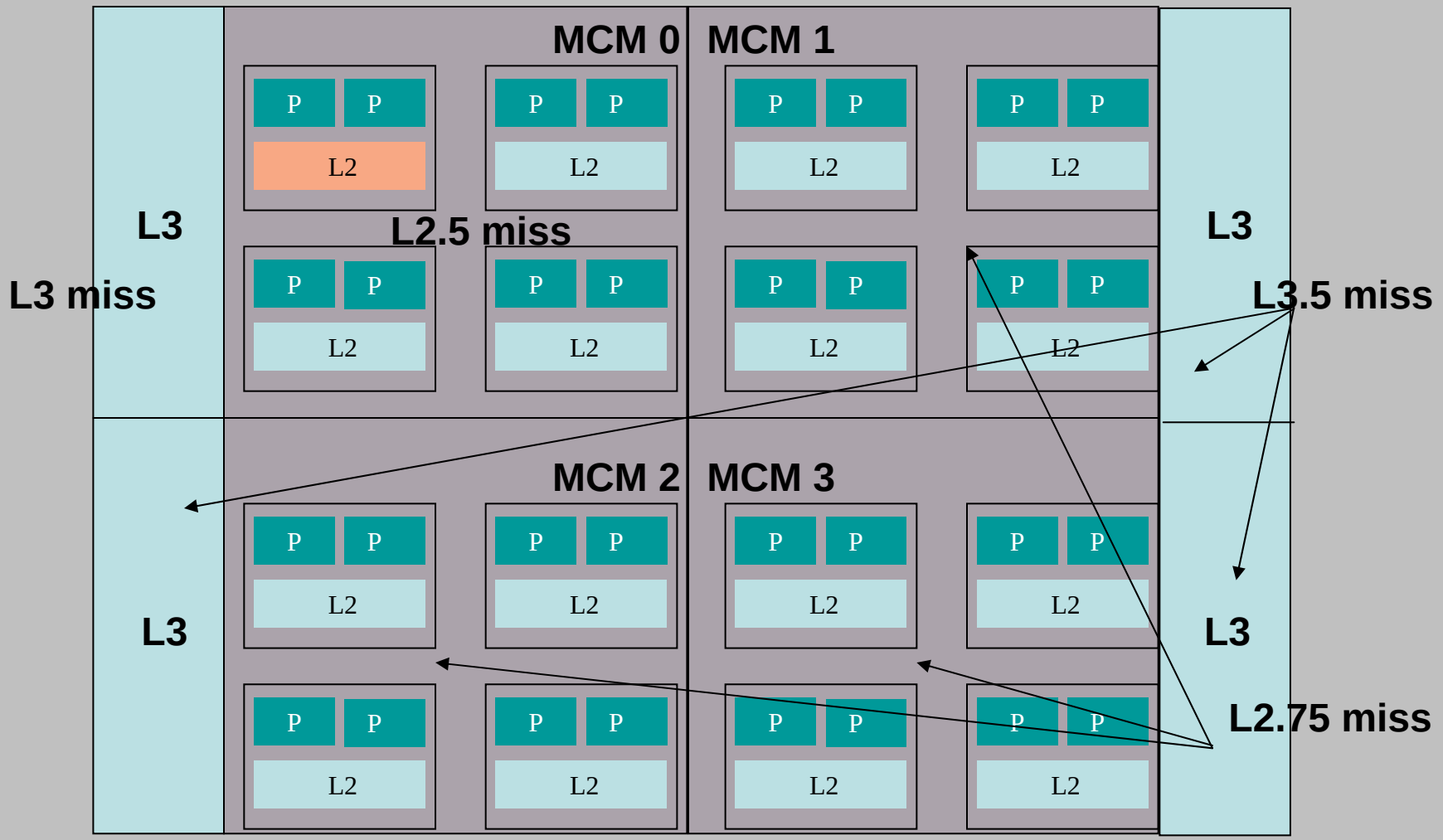
in Collaboration with IBM-Austin (Bret Olszewski, AIX Performance)
IBM faculty Award and NPSC/IBM Ph.D. Scholarship (Diana Villa)

- Sampling with IBM POWER4 PMU
  - 8- and 32-processor eServer pSeries 690 (p690)
  - Record periodic occurrences of an event
    - *Smaller, more manageable event traces*
  - 100 events/sec/CPU (default); 10 minutes TPC-C steady-state
- Event record (traces organized by CPU ID)



- Average number of samples collected/event (TPC-C)
  - 238,448/212,396 for 8-/32-processor p690 data

IBM p690



PETrAT

Performance **E**vent **Tr**ace **A**nalysis **T**ool

Diana Villa's Master's Thesis

Data Collection Environment

TPC-C

p690

Sampled Event Traces

PID TID Timestamp Instr.Addr. DataAddr.
PID TID Timestamp Instr.Addr. DataAddr.
PID TID Timestamp Instr.Addr. DataAddr.

Load DB Java Tool

Database

Queries

Report Generation Java Tool

Reports

5 BufferPool 56893 29384
6 Data,BSS,Heap 8799 4855
1 Kernel 23485 9840

Microsoft® Excel

Graphs

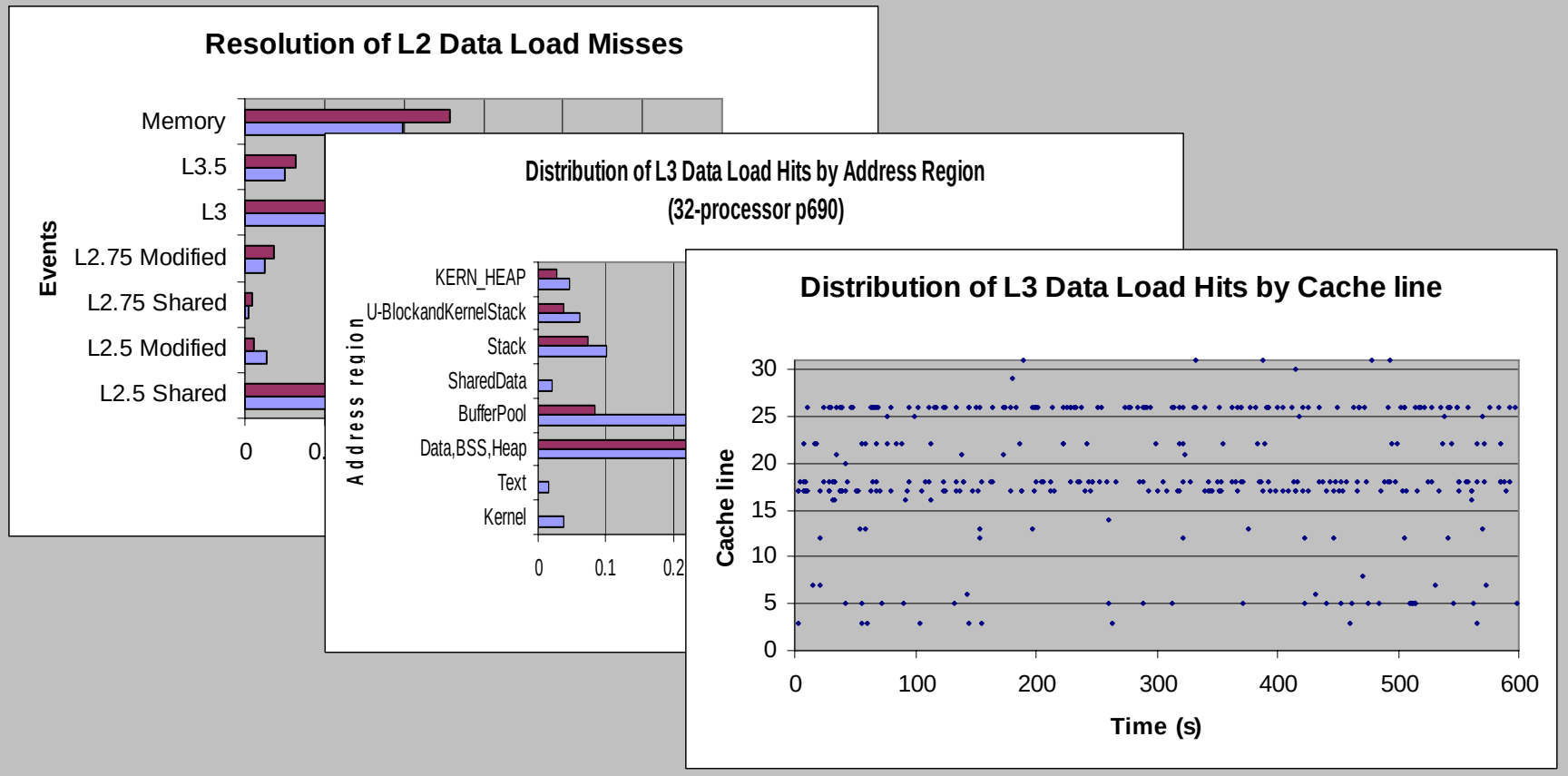
Data Reduction/Time to Insight
Sampled Event Traces - 2

- Data Collection
  - Storage: relatively small
    - Issue for Peta(fl)ops Era
  - Overhead/perturbation: small
    - Timer interrupts
    - Writes to fixed-size buffer
  - Usefulness of data:
    - High level of detail (temporal and spatial relationships among events)
    - Used to enhance performance of AIX and that of DBMS

Data Reduction/Time to Insight
Sampled Event Traces - 3

- Data Analysis
  - Large event set
    - Trace includes records for eight events (eight POWER4 counters)
  - Data access: facilitated by database
  - Refined analysis:
    - Minimization of collection/analysis iterations
      - Data as a guide to further collection / further analysis?
    - Hone in on performance problem through further queries
  - Effective visualization: nothing fancy – graphs
  - Autonomy: GUI (Juan Ulloa, Master's project – tech transfer within IBM)

Sampled Event Traces Data Analysis & Results



Data Reduction/Time to Insight

Sampled Event Traces - 4

- LACSI 2003: pinpoint application-specific sources of performance degradation
  - study L2-cache data load misses w.r.t. resolution sites across p690 memory hierarchy
  - refine analysis (from high to low level of detail): identify heavily-hit resolution sites/concentrated areas of locality of reference at high-penalty resolution sites
    - "hot" address regions, segments within regions, pages within segments, cache lines within pages, instructions/data structures

Data Reduction/Time to Insight

Sampled Event Traces - 5

- ICPADS 2004/MASCOTS 2004: diverse set of memory performance issues studied using one set of traces
  - Study effectiveness of design and policies associated with p690 memory hierarchy w.r.t workload demands
    - Characterize behavioral difference between private and shared data loads
      - Application / architecture matches/mismatches
    - Identify memory hits due to compulsory data cache misses
    - Identify false sharing
  - Study “cost” of intra-MCM migrations in terms of L2.5 data load hit events

Further Data Reduction – 1
Event Traces

- Data set size depends on
  - Number of processors/processes
    - Monitor one of each "process class"
  - Number of events monitored
    - Monitor only "relevant" events
  - Monitoring (time) interval (granularity of data collection), e.g., phase, iteration, function
    - Employ the "correct" granularity for each code section of interest
  - Sampling frequency
    - For on-processor event traces - study in progress to determine impact of sampling frequency

Further Data Reduction – 2
Event Traces

- Employ a two-phase process
  - First phase:
    - event profiling to select events to record
    - call-path profiling to exclude performance-irrelevant but frequently visited call paths
    - off-line simulations, models, multivariate statistical analysis to determine what processes, events, phases, functions, iterations to trace/analyze
  - During second phase
    - dynamic data reduction: for communication/synchronization traces, use stack-based on-line mechanism to delete trace data (from buffer before written to file) of call-path visits not satisfying a criteria, e.g., lack of synchronization or communication operations

Further Data Reduction – 3
Event Traces

- Employ a dynamic adaptive process
  - Use dynamic instrumentation or dynamically-controlled static instrumentation to collect data
  - Employ adaptive performance monitoring
  - Customize data collection and analysis
    - Processes
    - Events
    - Granularity
    - Frequency
  - Employ statistical sampling, call-graph analysis, and, perhaps, machine learning to identify parameters

Time to Insight

- How should the data be stored?
  - Database of performance event counts (TAU PerfDBMS)
  - Database of sampled performance event traces (PETrAT)
  - Communication trace files (Epilog)
- What analysis techniques should be used?
  - DB queries / report generators
  - EARL interface to read and analyze EPILOG trace records
  - Multivariate statistical analysis, call-path profiling, event profiling, models
- Should the process be autonomous?

Addressing the Scalability Challenges of Performance Data Collection & Analysis

- Reduced sampled on-processor performance event traces (w/o losing relevant information)
- Reduced communication event traces
- Off-processor performance event data
- Power consumption measurements
- Scalable analysis of trace data (database queries, EARL modules)
- Integrated analysis of power consumption data with hardware performance counter data
- Scalable visualization of analytic results