Multi-source Event Generation and Analysis for Performance Understanding in Large-Scale Environments (MEGA-PULSE)

Wu Feng[†], Robert J. Fowler[‡], Mark K. Gardner[†], Song Huang[†], Allan Porterfield[‡]
[†]Virginia Tech [‡]Renaissance Computing Institute

Context: High-end systems run with O(10,000) concurrent computational threads.
Problem: Resources through which threads interact and coordinate → *Critical Performance Bottleneck*
Solution: *Resource-centric* monitoring and performance analysis tools that capture interactions and contention for resources.

Resource-Constrained Multicore Systems

Issue: ***Utilization & contention for shared resources.***

Four independent copies of NAS MG or BT yield poor *utilization* efficiencies: ~55% on Phenom, ~43% on Clovertown for MG

LU.A	Execution Time	L3 Cache Misses
One copy of LU.A	213.26 sec	2.915e09
Four copies of LU.A	232.65 sec	1.450e10

Independent copies of NAS NPB LU.A cause *contention* through shared L3 cache on AMD Phenom.

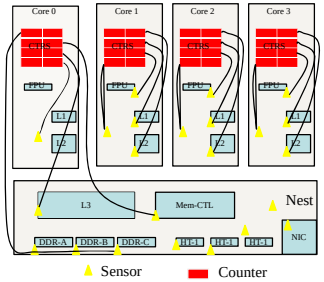
BT	DRAM accesses	DRAM Misses	DRAM Page Hits	DRAM Page Conflicts
One copy	4.59e09	7.06e08	3.27e09	6.27e08
Two copies	8.12e09	1.74e09	4.62e09	1.48e09
	< 2X	2.5X	1.4X	2.4X

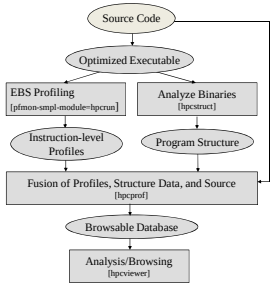
Independent copies of NAS BT cause *contention* in DDR2 memory on dual-core Opteron.

Need: ***Tools that focus on thread interactions at shared resources.***

Resource-Centric Reflection (RCR) Tool

- One core (0) measures events in the "nest."
- Other cores measure events private to cores.
- Core 0 processes the event logs of all cores and runs on-node analysis.
- Other cores run application threads.
- If core 0 is not used for application thread, system overhead to application is eliminated.
- Easy to study and analyze application behaviour without system noise.





Multi-Core RCR Prototype

- **Profiling:** Modified *pfmon* collects instruction-level, event-based sampling histograms in *hpcrun* file format.
- **Structural Analysis, Data Fusion, and Browsing:** Use unmodified HPCToolkit components.
  - ✓ Binary executables analyzed by *hpcstruct* to recover their program structure.
  - ✓ Profile and program structure interpreted and correlated with application source by *hpcprof* into a database.
  - ✓ Database presented and explored by *hpcviewer*.

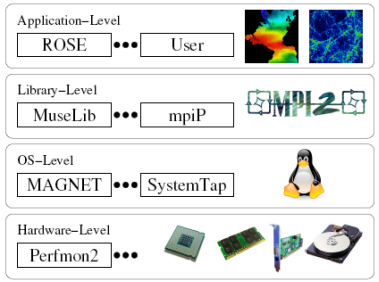
Advantages:

- ✓ *pfmon* consistently allows all 16 counters on a quad-core chip to be used for AMD processors.
- ✓ Non-competing instances of *pfmon* can be created to do simultaneous core/nest monitoring.

MEGA-PULSE Architecture

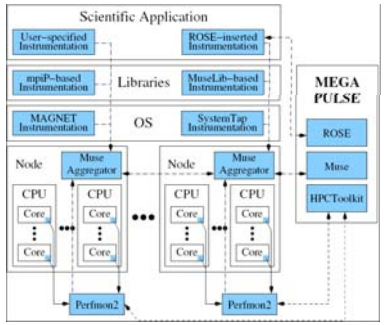
Data Resource Layers and Tools

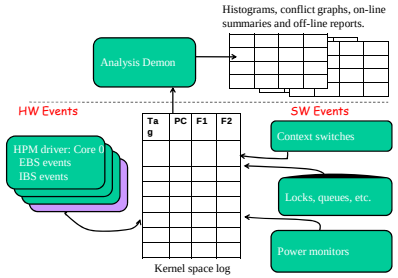
- Application Level: Use Rose to identify shared data structures and track their usage.
- Library Level: Use MuseLib to insert instrumentation points at link time to interpose at the API boundaries.
- OS Level: Use MAGNET to dynamically add and remove instrument-ation points from a running system.
- Hardware Level: Use Perfmon2 in Event-based sampling mode to record samples in per-core, bounded-length, log-structured buffer in kernel space.



Data Merging and Integration

- Extend the Muse data collection system to combine data from multiple sources.
- Extend wavelet analysis to improve compaction and to handle software events with low overhead.
- Perform analysis only at the performance bottleneck to reduce data volume.
- Develop new generation of browser in support of "third-person" view as well as individual view.



Implementation Strategy

- Unified kernel-space mechanism to merge events from multiple sources: hardware and software.
- Analysis of sampling buffer to understand multi-thread interactions and event interleaving
- Merging of additional event streams in user-level daemon.
- Compacted summaries for on-line adaptation, scalable tools.
- Exploration of new approach to present information to users.