Summary White Paper: Hardware Performance Monitor Design and Function

Peter F. Sweeney (IBM), Olaf Lubeck (LANL), Mike Lang (LANL), Rob Fowler (Rice U), Greg Johnson (LANL), Gabriel Marin (Rice U)

1. Introduction

The one-day workshop on **Hardware Performance Monitor (HPM) Design and Functionality** was attended by about 55 participants from industry, academia, and national laboratories. Speakers representing 14 different groups gave position briefs. The workshop (distinct from a mini-conference) consisted of a truly participatory working group where audience member discussions and spontaneous reactions were as valuable as the scheduled briefs. Major industrial participants included Intel, IBM, AMD, Sun, and HP. Design engineers from the Opteron, PowerPC, Pentium, Xeon, and Itanium processor groups added a great deal of reality to the discussions and attested to the industrial interest in the goals of this workshop. An international group of researchers and engineers were able to discuss their uses and needs for improved hardware performance monitors. One of the often-repeated statements coming from the participants was that this "was the first time where many different users and designers discussed goals and issues together in one room". The major complaint was that we did not have enough time to fully discuss the ideas. This white paper together with the slides from the talks is a summary of this workshop. Section 2 summarizes the multiple dimensions that comprise the user requirements for HPM's. Section 5 discusses the return on investment (ROI) of using precious chip real estate for HPM's. Section 6 discusses standardization issues. Section 7 discusses issues that are going to be challenges in the future. Section 8 are conclusions and a vision for the future of HPM's

Workshop Themes and significant issues:

In this section, we attempt to briefly highlight the major themes that were brought out by speakers and participants alike. We provide references to the talks (most of the slides are available on this web site) and discussions that initiated the themes. Some of theses themes became persistent issues during the day and are discussed in more detail in Sections 3-7.

1. HPM's are being used for a wide variety of purposes and goals. Section 2 discusses the dimensions of their applicability and use. Workshop participants are relying on HPM's to supply the basic data needed to tune and model application and system-wide

performance, adapt to dynamic shared resource requirements, and model and manage power consumption.

2. HPM's may be "second-class" citizens in the priorities of processor design cycles (Brinkley talk). Functionality, correctness and validation of mainstream processor functions take precedence. Is this the reason that users see little documentation and description of HPM's and associated events? What's the return on investment for HPM's? Many chip designers who were present disagreed with the assertion and felt it was somewhat overstated.

3. At a minimum, all HPM's should supply a holistic, big picture of application performance, system performance and power management:

- For application performance: the big picture is a comprehensive, hierarchically organized accounting of cycles where the processor is making progress and where it is stalled. A participant summarized this area by stating that we need to not only count the stall cycles, but also attribute the reason for the stalls. It was recognized that stalls can occur for multiple reasons but can be attributed to a single cause with a priority scheme (as in the Itanium).
- For system performance: event information must be able to be attributed to processes and threads that are contending for shared resources (HP talk and Peter's discussions).
- For power issues: the big picture includes activity measurements for functional units – population counts of major associated queues (Margaret's student talk). With this data, accurate predictive power consumption is possible.

4. Monitoring memory system performance is still the key issue. Most of the processor stall time is a direct result of memory latency and bandwidth capabilities. Dynamic memory allocation and data layout have a huge impact on system performance. The information needs to be mapped back to the program and thread that the developer is working on. For example, counters counting cache misses is often not enough. The problem may be due to the layout of data structures and the same code may be used to traverse a number of data structures with different access patterns. TLB and cache eviction data is required. (Irina, Christos & Mustafa talks)

5. New processor directions in SMT and CMP will require information about contention for shared resources (Rob's and Christos's talk). These include queue populations for shared resources, tagging for contention characterization, and cross-thread eviction data.

6. In-order and out-of-order instruction execution models will require different event semantics and it may not be possible to "standardize" events across processors. For example, the basic concept of a stall may be defined as cycles where there are no instruction issues for in-order execution, but stalls for out-of-order should be related to functional unit completions. (discussions from Alex, Nidhi, Jim).

7. There is a need to define a standard interface to access event counters in OS kernels (Stephane talk). Section 6 discusses this in more detail. Benefits from a standard interface include: uniform level of functionality across platform; focus on the tool instead of the infrastructure; increased tool reuse that results in faster development and improved capability.

8. "Where's the science?" – asked by one of the chip designers (Jim talk). One answer is that we can use data from HPM's to develop and validate performance models (statistical or queuing theory). Another response to this question is that we need to carefully layout "evidence" of need when requesting features in HPM's. (initiated by Jim and discussions by many).

9. Toolkit developers would like to see more HPM counters that allow for simultaneous monitoring and reading of events. (Rob talk and discussions from Peter) This may be limited by hardware design considerations (see Section 3).

10. The level of abstraction of HPM events is relatively low-level. Better metrics need to be discovered that aggregate the events (Martin LLNL talk). Section 7 discusses new challenges facing HPM designers.

2. Hardware Performance Monitor Requirements

An important theme of the workshop was that there are many different ways in which performance analysts are using hardware performance monitors, HPM's. Each way has its own unique set of requirements on the HPM design and functionality. Instead of attempting to enumerate each way, we have identified a number of key orthogonal dimensions. Each dimension is a spectrum that is identified and defined by its end points. Each point along a dimension places different requirements on the design and functionality of the HPM's. The subsequent sections present the dimensions, each dimension's end points, and how the dimension affects HPM design and functionality.

Chip Designer versus Software Performance Analyst

On the one hand, hardware performance analysts are interested in understanding low level aspects of a microprocessor's behavior (Talks: Nidhi, Mericas, Mucci, Sprunt). The HPM events that they are interested in may have little semantic meaning with respect to the software. This group uses software to observe hardware behavior. They typically write small kernel programs to investigate hardware behavior giving them complete control over the software. Their kernel program's access to hardware functionality is well-defined and regular. After running a kernel program to collect HPM values, they use the values to determine if the hardware is working as expected (discussion).

On the other hand, software performance analysts are interested in using the HPM's to understand the behavior of software (Talks: Callister, Eranian, Fowler, Itzkowitz, Schultz, Sprunt). This group has little or no control over the complex software that they are analyzing, as this software is usually written by others. In this world, the access patterns to hardware resources are neither well-defined nor regular.

These two groups put different requirements on HPM design. The software performance analysts require that an HPM event monitors only one possible outcome in the underlying hardware. If an event monitors multiple outcomes, a software analyst may not know how to attribute the event's occurrences to the different outcomes due to the complex behavior of the software that is running, rendering the event useless for performance analysis. For the hardware performance analyst, requiring that an event monitors only one outcome is not a necessity, because a kernel program is written, when needed, such that only one of the many possible outcomes will occur.

Application versus System-wide Performance Analysis

Application performance analysis is interested in an application-centric view of the behavior of the hardware; that is, understanding the hardware's behavior from the perspective of an application (Talks: Chihaia, Fowler, Mucci, Tiker, Schultz). They typically run an application in isolation to study its behavior. The application is typically single threaded.

Alternatively, system performance analysis is interested in a system-centric view of the behavior of the hardware where the "application" could be considered a virtual machine,

an operating system or a hypervisor with one or more layers of software running on top of it and with one or more applications running in each layer. This view is interested in the interaction between and within layers.

System performance analysis requires that HPM events that monitor shared resources distinguish how those resources are shared (Talks: Antonopoulos, Collard, Itzkowitz, Mucci). For example in an simultaneous multithreaded (SMT) machine, where a core supports multiple hardware threads that share a single L1 cache, it is imperative that the HPM events that monitor the L1 cache can distinguish between hardware threads. With a chip multiprocessor (CMP) machine, where a chip has multiple cores that share a single L2 cache, it is imperative that HPM events that monitor the L2 cache distinguish between cores.

These two types of analysis put different demands on the set of HPM events that are supported. System-wide analysis requires a richer set of events that can distinguish how resources are shared by different applications or by threads of the same application.

Novice versus Expert Performance Analyst

Novice performance analysts are interested in a general understanding of the underlying hardware's behavior (Talks: Itzkowitz, Mucci). Their view of the hardware is at a high, abstract level. They are interested in a limited number of events that have semantics that are defined at the instruction level. For example, events that monitor memory hierarchy behavior are identified by loads and stores that hit and miss in cache, and events that characterize program behavior are identified by the number and type of instructions that are executed.

Alternatively, expert performance analysts are intimately familiar with the idiosyncrasies of the underlying hardware and are interested in a rich set of events that can help them understand these idiosyncrasies (Talks: Chihaia, Fowler, Sprunt). The expert wants an event defined for any underlying hardware artifact that could possibly affect an application's behavior.

These two users groups put different demands on the set of HPM events that are supported. Novice users want a small set of events that are closely tied to the instruction set and expert users want a large set of events that reveal all possible causes of performance loss.

Offline versus Online Performance Analysis

Offline performance analysis is concerned with analyzing HPM values after an application has stopped executing. In this approach, an application may be run multiple times: each time HPM values are collected for a different set of events. This traditional use of HPM's tends to be application-centric as an application can be run multiple times.

Alternatively, online performance analysis is interested in understanding the underlying hardware behavior of an application or system as it runs (Talks: Antonopoulos, Schultz,

Sprunt). To be able to accomplish this feat, all the events that are necessary to understand behavior must be monitored simultaneously at run-time. There is no after-the-fact (offline) opportunity to analyze the events, nor is there an opportunity to run the application or system multiple times to collect all the events. Applications that run continuously, such as an operating system or web servers, may require online performance analysis.

Unlike offline analysis, online analysis requires many HPM's with no restrictions on the events that can be counted simultaneously. Although multiplexing the monitoring of events help alleviate some of the requirement for many HPM's, the higher the multiplexing factor, the greater the loss of precision. Online analysis also requires that HPM events are hierarchically defined. That is, the value of a child's event can be computed from the values of its parent and siblings events. Given HPM events that are hierarchically defined, a child's event only needs to be monitored if the value of its parent event is unusual. Otherwise, a child's event never needs to be monitored.

6/3/05 9:55 AM
Comment:

3. The Hardware Designers' Perspective

On-Chip Resources and Constraints (talks by Jim, Alex, Nidhi)

Like everything else on the chip, HPM structures are composed of wires and transistors. These take up chip area, consume power, and require time and effort to design and verify. Furthermore, the propagation of signals along a wire is a function of the power with which the signal is driven, the size of the wire (Length and width determine resistance and capacitance.), and the number and size of transistors being driven. If one tries to drive a long, skinny wire with a minimal transistor, the time to propagate the signal will be large. HPM facilities are hooked into components that are spread across the chip, so there is necessarily a tradeoff among time (cycles), area, and power in their implementation.

The sub-components of a HPM facility include:

- Wires connect everything.
- Detectors are the points at which the HPM hardware is hooked to the hardware it measures. The design of a detector depends upon the easy availability of the signal. It may not be possible to add additional load to a signal that's already on the critical path of the chip. Other problems arise if it is difficult to add a wire into the interior of a functional block, or if the event of interest is not intrinsically represented by an existing signal. If an ``event'' needs to be defined by combining probes into different parts of the chip, difficulties arise because of distances, timing, and the need for additional logic. These difficulties are part of the reason why events on different chips might have similar, but slightly different definitions.
- Multiplexors are used to select the events that will be measured. The number of potential events is typically much larger than the number of physical counters.
- Storage components including registers and latches are used to deal with multi-clock events, and to record processor state information. The processor state information is intended to record the instruction(s), data addresses, etc to which some triggering event is attributed in profiling (trap on overflow) mode.
- Counters count the number of times an event occurs.
- Because the events that need to be measured occur in components spread across the chip and the HPM detectors must be located where the events occur, the HPM subsystem must span the chip. One design strategy, epitomized by the Pentium 4 design, is a distributed design in which the HPM is composed of several

geographically localized detector-to-counter subsystems that can measure only events that occur in that corner of the chip (Nidhi talk). The other main design strategy, epitomized by the Power 5 design, is a hierarchical design in which multiple levels of multiplexor and buffers are used to bring event and processor state information to a centralized set of counters and state registers (Alex). In the distributed design, there are a large number of counters, each of which is multiplexed among a small number of events. The hierarchical design has fewer counters that can be used more flexibly, but a full crossbar from events to counters is not possible.

Architectural Complexity versus HPM Design

While the programming model used for today's processors is still very close to the classic sequential model, the underlying implementation exhibits a prodigious amount of parallelism in a distributed collection of components. This parallelism is implemented using pipelining, multiple functional units, out-of-order execution, and multiple thread contexts.

This internal parallelism complicates the HPM designer's job by making it difficult to attribute an event to one of the many instructions that is currently in execution. When the designers of Digital's Alpha processors moved to an out-of-order execution model in the Alpha EV6 chip, they recognized the difficulty of attributing events to individual instructions. To remedy the situation, they dramatically reduced the number of countable events relative to the previous in-order generations of the Alpha. When the Alpha EV67 chip came out, it retained a small number of events, but it introduced ProfileMe as an alternative mechanism. The MIPS R10000/R120000 processors retained conventional event counts, but without precise attribution of events; for example, the count for level 2 cache misses was incremented when the miss was satisfied, not when it was first raised. The program counter often pointed to an instruction not related to the miss.

The designers of the HPM systems in the IBM Power 5 (Alex talk) have gone to considerable effort to provide precise attribution of events. Given that the recording of an event on this chip can take several cycles, there is a danger that if several events occurred within a few cycles of one another that only the first one would ever be recorded in profile mode. To avoid the danger and related problems on this out-of-order chip, a variety of mechanisms (continuous, queue based, and random marking of instructions; event-based tagging of instructions that caused monitored events; and speculative counting and ex post facto attribution of costs) are used to count events such as wait cycles and to allow the HPM to figure out reasons later.

In contrast, the in-order architecture of the Itanium (Jim talk) made it a lot easier for the designers to provide precise attribution through the event address registers, the branch trace buffer, and other circular buffers for tracing events after they've occurred. The

problem of closely-spaced events shadowing one another is still an issue, so randomization is used to decide whether or not to sample a particular event.

The cost of tracking and attribution is continuing to grow with chip/processor complexity. For example, the introduction of hardware multi-threading introduces a need to tag or track instructions by threads (HP talk).

4. Good HPM Design from the User's Perspective

There are many ways in which performance analysts use HPM's (see Section 2). Nevertheless, a number of requirements are common to all users. This section presents HPM design guidelines for these common requirements from a user's perspective. Although modern microprocessors provide one form of HPM functionality or another , their functionality differs. While we think that it is generally not a good idea for hardware designers to standardize their offerings because this may stifle innovation, a small set of common events widely used by all classes of performance analysts should be provided on all architectures. As an example, events for program characterization should minimally include:

- Total cycles
- number of cache/TLB misses
- numbers of instructions retired by category
- stall cycles categorized by reason.

As discussed in Section 6, standardization for this set of events may be achieved through OS interface and middle-ware software layers.

In addition to a common set of events, each new microprocessor design needs to provide HPM functionality to enable evaluation of the impact that new hardware features have on performance.

The introduction of the Itanium hierarchical organization of stall events in a tree like structure is popular with performance analysts (AMD talk and Jim talk).
Such a design enables a systematic approach to program tuning. One can start by analyzing the events at the top of the tree, then drill down for a more detailed view of performance data in subcategories that indicate a performance problem.
A hierarchical structure is useful for program characterization events also. For example we could have a top level event for measuring floating point instructions, and sub-events for measuring different types of floating point instructions, such as adds, multiplies, divisions, etc.

An important component of HPM design, which is often overlooked by hardware vendors, is comprehensive documentation. The documentation needs to organize events by functionality and to target description to different levels of user ability. For example, novice users are interested in a quick guide for how to use a subset of the HPM events to determine the main sources of inefficiencies. Expert users are interested in understanding where every cycle is spent and in evaluating the impact of advanced hardware features on performance. For this latter class, a more thorough, systematic, complete guide of all the HPM events is needed.

Hardware validation is one step of the design process that is often rushed (Brinkley talk). Chip designers' priorities are first to ensure functional correctness of the processor, then

to verify that achieved performance meets expectations, and third to validate the correctness of hardware events. Post-silicon validation of HPM's is limited: many events are not validated. Without this last step, much of the effort that is put into the HPM design is wasted.

5. Justification for HPM's

The design and implementation of HPM's in a microprocessor is, like most engineering problems, affected by both economic and technical considerations. In the end, decisions are driven by the expected return on investment (ROI) (Brinkley talk and discussion).

ROI can be evaluated under two scenarios. What are the costs and benefits of the HPM implementation? What are the costs and benefits to exposing an HPM event to vendors and selected partners, to customer, and to competitors? Costs of HPM design include:

- The cost of design, implementation, and validation of the HPM facilities.
- The opportunity cost of devoting on-chip resources to the HPM rather than some other feature that directly improves performance. The cost in terms of increasing the size or decreasing the yield of the chip.
- Potential delay in the delivery schedule for the chip.
- If the HPM is exposed as part of the architecture, the recurring costs of support and of reimplementation in future generations of the processor and of system software that uses the facilities.
- Potential exposure of design flaws to competitors and customers. Detailed performance information can give competitors insight into details of design successes and failures.

Benefits of HPM design include:

- Faster debugging and validation of the processor, leading a faster time to market.
- Improved performance of standard software and benchmarks improves both perceived and actual value of the chip to all potential users. This can improve both sales price and volume. Performance counter hardware is very desirable for users. A study by HP (Jim talk) concluded that the performance increase from using counters is more effective than simply waiting for a faster processor.
- Performance tuning facilities make the chip more attractive to both independent software vendors (ISVs) and customer programmers. Libraries and OS's that adapt based on performance parameters depend on HPM's
- Improved competitive position by satisfying instrumentation requirements in procurement specifications of some customers.
- Performance measurements of today's systems can inform the design of the next generation. The benefits of specific architecture/implementation features can be measured.

6. Standardization

The current state of hardware performance monitors is both good and bad. The good is that every major microprocessor vendor provides HPM's with their microprocessors. The bad is that every vendor has its own naming convention for events, registers, and concepts; and its own way to program the HPM interface. Such diversity makes it difficult to reuse HPM functionality across microprocessors from different vendors, or even to reuse HPM functionality across different implementations of the same micro-architecture from the same vendor.

There was a general consensus among HPM users for the need of standardization across micro-architectures and across implementations of the same micro-architecture. The need arises because the world of computing is heterogeneous (most people have access to and use multiple micro-architectures) and it is not uncommon for machines to be replaced with next generation implementations within a 2 to 5 year period. Using HPM requires a significant investment in understanding the events and programming the HPM functionality. Without standardization, this investment cannot be reused.

There are three levels where standardization is possible.

- At the hardware level, an industry-wide set of events could be defined with a precise semantics (Talks: Callister, Mericas). Every platform would be expected to implement these events. See Section 4.
- Above the hardware level, a common API to access the HPM functionality could be defined and every platform would be expected to implement this API (Talk: Mucci).
- At the OS level, a common interface is needed in the kernel. (Talk: Eranian).

Each level would provide a solid footing across platforms for HPM reuse by users and tools.

6/7/05 8:39 AM
Formatted: Indent: Left:

There was a general consensus among the HPM designers that standardization must not hinder new innovative HPM design: individual vendors need to be able to innovate and differentiate their processors from each other.
PAPI (Performance Application Programming Interface) is the best in its class in trying to support a common set of events and providing a common API. PAPI defined a set of standard PAPI events, which was the most common events available on the architectures of the time, and then proceeded to map HPM events of different vendors into these PAPI events. PAPI also provided an API to access HPM functionality across platforms. The API provides access to a subset, the least common denominator, of the HPM events available on many of the platforms, the rest of the events are accessed through a "native event" interface. Another challenge due to lack of standards are that the events on different architectures may not always have the same meaning. On one micro-architecture, the instruction count event may include no-ops, while on another, it may not. Unless vendors mutually agree upon the HPM functionality that is provided, it is

difficult to develop a common set of events and an API that encompasses the different HPM functionality that is available.

The third level of standardization is a common operating system interface. Standardization at this allows users to immediately access new features without having to wait for patches to the OS. A common model of register based access was suggested at the workshop [13]. Tools would especially benefit from a common operating system interface. Porting to new operating systems would become trivial. System architects would not be hindered by a common interface, because innovation and proprietary functionality in the HPM does not require changes to this layer. Another benefit is that these changes would be more easily accepted in to main stream kernel development, whether it is Linux or a proprietary Unix.

The hardware designers argue that standardization will not be an easy task to achieve. First of all, vendors consider their HPM functionality a competitive advantage that they are not willing to give up easily. Partly for proprietary reasons, HPM functionality has intentionally not been architected. But just as important, hardware designers do not want to be shackled with previous HPM functionality in their new implementation of a micro-architecture. For example, Intel's Pentium 4 has a trace cache, an architectural feature that does not exist in the Pentium 3 or in Pentium M. If the trace cache was architected, how would the trace cache events be emulated in future microprocessor designs that don't support it?

7. New challenges

As processor design evolves, new challenges in performance monitoring come to the forefront and these themes were prevalent at the workshop:

- Power management.
- Multi-core or CMP (chip multi-processing).
- Multi-thread or SMT (simultaneous multi-threading).
- Out of core - monitoring of board or system level events outside of the CPU core
- Virtualized hardware and hyper-visors.

Power has become more and more important as frequencies rise. Research in this area is leading to new requirements for performance monitoring hardware. Power management researchers need to be able to measure usage of functional units. These would then be managed by software to adaptively change power going to functional units in real time. Units would be powered on and off, and the operating frequency would be stepped up or down depending on what is happening in the executing code.

SMT processors have been around for a while now. Most processors in the near future will have either SMT or CMP. Issues for Multi-threaded processors and multi-core issues are similar, and are focused on resource contention. Performance tools need the ability to relate the event information directly back to the process or thread. Most workshop attendees wanted a complete HPM for each core. If there is not a HPM for each core then there must be new hardware added to track which of the cores' processes is affecting resources. This is major concern with shared caches, one core's process may pollute the cache of one of the other cores. HPM users would like to see tags for resources under contention, for example cache eviction tags for which thread causes the eviction. Currently the cores share HPM's and users are not allowed to have two active HPM processes simultaneously [4].

In many cases, research is now moving off of the CPU core and on to other components. Information is needed for the performance of bus transactions, main memory interactions and interconnects. HPC (high performance computing) users want to look at a whole supercomputer with a system view, this would aggregate interconnect counters, interconnect switch counters and data from all other individual HPM's the system has access to. The "owl"[14] system design takes this to an extreme with FPGA's across the system (Martin talk).

Virtualized hardware is an active area. Hypervisors such as ZEN or rHype are revisiting the features of multi-domain IBM mainframe OS's without the overhead. Vendors are now adding features to processors for support of virtualized hardware. New problems include access to HPM's from separate instances of the OS, and security of information that can be inferred from access to the HPM [6].

8. Conclusions

This section summarizes a number of the hopes and aspirations that the HPM community of users has for the direction of the design and functionality of hardware performance monitors.

The current state of hardware performance monitors is both good and bad. The good part is that every major microprocessor vendor provides HPM's with their microprocessors. The bad news is that every vendor has its own naming convention for events, registers, concepts; and their own way to program the functionality. Our hope is that the vendors will start to standardize the vocabulary that they use to discuss HPM concepts, standardize the operating system access to the HPM functionality, and standardize the model to program this functionality. We would like an industry standard on terminology and on the semantics of HPM events.

Most applications fall far short of a machine's peak performance. Our hope is that the HPM's would provide the insight into this deficiency by identifying which parts of the machine is inhibiting performance and the reason why. Part of this vision is that an algebra on the HPM events could be defined that would provide an analytic modeling basis for the insight. The current state of the art rarely allows HPM events to be meaningfully combined into formulas.

Our hopes are that as new microarchitectural designs, such as simultaneous multithreading (SMT) and chip multiprocessing (CMP), become more widely available, and as new microarchitectural features, such as power and temperature, become more important, the HPM events provide the performance analysts with the means to understand these new architectural features and tune their applications. The current state of the art is that new architectural features are incorporated into microprocessors and released without any means to understand their implications on performance. For the full potential of these architectural features to be met, HPM support is required to understand and evaluate their performance impact on software.

The current state of affairs for the evolution of HPM functionality is that performance analysts want more and more (Fowler, Isci) and the HPM designers and implementers want more and more proof that the functionality is useful (Callister). What is needed is a way to evaluate the benefit and cost of HPM functionality and that this evaluation becomes the basis for how HPM functionality evolved. As one workshop attendee questioned "Where is the science?" Our hope is that a science is developed that provides a means by which to evaluate the benefit and cost of HPM functionality.

References:

1. Summary and Wrap Up
   Towards a Flexible and Realistic Hardware Performance Monitor Infrastructure, Phil Mucci
2. What we need to be able to count to tune programs
   What we need to be able to count to tune programs, Mustafa M. Tikir, Bryan R. Buck, Jeffery K. Hollingsworth – University of Maryland
3. Hardware Event Counters
   Using Hardware Event Counters for Continuous, Online System Optimization: Lessons and Challenges, Christos D. Antonopoulos & Dimitrios S. Nikolopoulos
4. Position Paper - Hardware Event Counters
   Position Paper - Using Hardware Event Counters for Continuous, Online System Optimization: Lessons and Challenges, Christos D. Antonopoulos and Dimitrios S. Nikolopoulos
5. Performance Monitoring on Pentium 4 Processors
   Performance Monitoring on Pentium 4 Processors, Nidhi - IA 32 Performance Architect
6. NUMA Instrumentation Challenges
   NUMA Instrumentation Challenges, William C. Brantley, PhD Advanced Micro Devices
7. Efficient Collection of Information on the Locality of Accesses
   Efficient Collection of Information on the Locality of Accesses, Irina Chihaia Tuduce and Thomas Gross - Laboratory for Software Technology ETH Zurich, Switzerland
8. PowerPC Performance Monitor
   The PowerPC Performance Monitor, Alex Mericas - IBM
9. Performance Hardware
   Performance Hardware – If I ran the world. Observations on the current state and future needs for hardware performance instrumentation, Rob Fowler - Rice University.
10. Performance Monitoring Hardware will Always be A Low Priority, Second Class Feature in Processor Designs
    Performance Monitoring Hardware will Always be A Low Priority, Second Class Feature in Processor Designs Until ..., Brinkley Sprunt – Bucknell University
11. Confessions of a Performance Monitor Hardware Designer
    Confessions of a Performance Monitor Hardware Designer, Jim Callister - Intel Corporation
12. Hardware Performance Monitoring
    Hardware Performance Monitoring: Sun's Perspective, Marty Itzkowitz - Sun Microsystems, Inc.
13. “Experiences with Perfmon” Stephane Eranian – HP

14. “A Vision for Next Generation System Monitoring” Martin Schulz – Lawrence Livermore National Laboratory, Brian White, Sally A. McKee – Cornell, HsienHsin Lee – Georgia Tech.