PAPI Breakout Report

Four of us, David Skinner, Mark Krentel, Marcus Geimer and Dan Terpstra met to discuss issues related to PAPI-C and components for PAPI-C. A number of topics were discussed:

- Concern was expressed that the recently introduced perf_events infrastructure for hardware counter access in the kernel has limited bandwidth for handling overflow interrupts. It was agreed that Rice would investigate whether these limits posed a problem in practical applications, and the PAPI team would explore whether the *actual* overflow count could be exported at the end of each sampling period to correct for any potential throttling. The PAPI team would also explore the provision of an interface for setting sampling periods by rate as well as by threshold as exposed by perf_events.
- Since the perf_events interface sources event categories beyond the hardware cpu counters, there was discussion about the best way to expose these new performance events through PAPI. It was decided that this would best be done by additional components rather than within the perf_events cpu component. This approach was confirmed by Stephane Eranian who pointed out that several categories only made sense in system-wide counting mode.
- The issue of component counter read semantics was discussed. CPU counter reads can be assumed to have extremely low latency, since you are typically reading registers on the cpu itself. Components have no such constraints and can have read latencies that span many orders of magnitude. It is undesirable for a high speed component read to stall on a very low speed component. Mechanisms to address this should be explored. David Skinner and the PAPI team will work together to address this issue.
- Component versioning and dependency tracking will become an increasingly important issue as the numbers and types of components rise. A request was made to provide a listing of needed libraries and paths for a given configuration of PAPI and its components. Such a listing could be provided by configure or by a separate shell script from the make files. The PAPI team will investigate.
- PAPI components can measure and report data from a wide range of sources. It's desirable to encourage a broad range of component contributions from the user community. But just because something *can* be done doesn't mean it *should* be done. There was a preliminary discussion on which types of measurement s are most appropriate for PAPI components and which are not. No definitive conclusions were reached.
- The observation was made that the current OFED driver for Infiniband requires write access in order to read the counters. This is unacceptable and probably unnecessary for large systems in sensitive environments. David Skinner will attempt to contact the appropriate OFED people to change this restriction.
- The idea of a web-based Component Repository for PAPI-C was introduced in the PAPI talk on Tuesday. This idea was strongly endorsed in small group discussion and will be undertaken by the PAPI team.
- An effort was made to identify potential components that are either under development or desired, with the idea of using them to "seed" the repository and generate some momentum toward contributing other components:
    - External energy measurement component under development by Juelich.
    - Gemini network component may be possible thru Cray. Concerns about exposing unreleased software interface may require exploration of binary rather than source distribution mechanisms.
    - PBS component could report job time remaining, topology, other info. David Skinner will pursue this.
    - MPIT component can provide detailed performance information from MPI implementations and also possibly unify parameter naming. Martin Schulz is pursuing this. PAPI and possibly OpenMPI teams to support.
- Speculation was offered about possible user bases outside HPC who might be interested in exploring the PAPI-C concept. Such user bases might include sysadmins for system health

monitoring; gamers or game developers; even other languages, such as Haskell: http://hackage.haskell.org/trac/ghc/wiki/Debugging/LowLevelProfiling/PAPI

- There was additional discussion about the possibility of using PAPI-C component information dynamically to drive autotuning efforts. CPU counter information can certainly be used in such an effort; what other kinds of information could be also used? Network traffic? Memory high water marks? MPIT information?
- Another final thought that came up outside the formal discussion involved the development of a shared xml formatted file to collect configuration information for a variety of tools in this community. If such a shared format and readers were developed, a shared access library could also be developed to read and write to such a configuration file.