U.S. Department of Energy, Office of Science
Advanced Scientific Computing Research

Knowledge-Based Parallel Performance Technology for Scientific Application Competitiveness

Final Report

August 15, 2007 – May 14, 2011

DOE Agreement: DE FG02-07ER25826

Allen D. Malony and Sameer Shende
Department of Computer and Information Science
University of Oregon

1. Introduction

The primary goal of the University of Oregon's DOE "competitiveness" project was to create performance technology that embodies and supports knowledge of performance data, analysis, and diagnosis in parallel performance problem solving. The target of our development activities was the TAU Performance System and the technology accomplishments reported in this and prior reports have all been incorporated in the TAU open software distribution. In addition, the project has been committed to maintaining strong interactions with the DOE SciDAC Performance Engineering Research Institute (PERI) and Center for Technology for Advanced Scientific Component Software (TASCS). This collaboration has proved valuable for translation of our knowledge-based performance techniques to parallel application development and performance engineering practice. Our outreach has also extended to the DOE Advanced CompuTational Software (ACTS) collection and project. Throughout the project we have participated in the PERI and TASCS meetings, as well as the ACTS annual workshops.

The original award started on August 15, 2007. Progress reports were submitted 90 days before the end of each budget year in May 2008 and 2009 for project Years 1 and 2, respectively. We requested a no-cost extension to the award in August 2010 and were granted a 9-month extension to May 14, 2011. Thus, the final report covers the period from August 15, 2009 to May 14, 2011.

The final project report is organized as follows. The progress reports for Year 1 and 2 provided detailed discussion of the significant progress made in the development of performance database and data mining technologies in TAU, and our PERI and TASCS interactions. In Section 2, we summarize the highlights of this work. We then focus on the achievements in the last project period in Section 3. Section 4 describes our participation in external activities throughout the project. Section 5 concludes the report.

2. Highlights for Project Years 1 and 2

The project proposal emphasized the transfer of knowledge-based performance technology through direct performance engineering engagement with the DOE SciDAC PERI and TASCS efforts. The work involved TAU development efforts to demonstrate advances in performance technology to better characterize parallel performance data, mine multi-experiment results, and understand relationships in factors that help to diagnosis problems. We applied these TAU advances in application performance

engineering efforts (PERI) and in component-based application development (TASCS). During Year 1 of the project, we worked closely with the PERI Tiger Team efforts on the S3D and GTC applications. Our efforts with the Common Component Architecture (CCA) environment focused on the integration of TAU and CCA's computational quality of service (CQoS) objectives. In Year 2, we improved support for multi-dimensional data analysis, analysis workflow scripting, and performance knowledge integration, again with application to PERI and TASCS goals.

2.1 TAU Technology Achievements

The project delivered important new TAU capabilities in the first two years. The key developments are listed below. More detailed information can be found in the earlier progress reports.

PerfDMF: TAU's performance data management framework (PerfDMF) was significantly enhanced to capture metadata about the development environment, platform, and application execution. This came through automatic querying of systems information at runtime, a TAU API for metadata recording by the application, and mechanisms for database metadata updates post-execution. With these improvements, PerfDMF was updated to provide a reference implementation of the PERI DB specification. This included the addition of support for metadata storage with parallel profiles, importers from multiple profile formats, and implementation of XML metadata and profile export. PerfDMF was linked to the PERI DB website interface for performance data/metadata search and query. All of TAU's performance analysis tools were updated to work with PerfDMF.

PerfExplorer: A new version of TAU's performance data mining framework (PerfExplorer) was developed during the project to process knowledge (metadata, scripts, rules) that extended the base-level analysis. Implementation of analysis scripting enabled the automation of multi-step performance analysis procedures in the form of workflows. All analysis components were updated to be modules that could be linked into workflows. Intermediate results and provenance were also implemented in the PerfExplorer architecture. The ability to access metadata within PerfExplorer allowed new meta-analysis capabilities. Also, PerfExplorer analysis supplemented with new classification features. To support performance diagnosis, we incorporated an inference engine in PerfExplorer that could process rules for reasoning about performance problems.

Integration with OpenUH: In collaboration with researchers at the University of Houston, we integrated the TAU measurement system and the PerfExplorer analysis framework in the OpenUH compiler. The union provided OpenUH auto-instrumentation of source code with TAU and automated analysis using PerfExplorer scripts of the performance measurements. PerfExplorer inference rules were developed to recognize and diagnose performance characteristics important for feedback-directed OpenUH modeling and optimization, such as for OpenMP load balancing.

TAU support in Eclipse: The TAU toolset was integrated in the Eclipse Parallel Tools Platform (PTP) to support measurement and analysis of parallel applications developed using the Eclipse IDE. The first phase in performance tool integration resulted in the implementation of plug-ins for several Eclipse IDE configurations, including the C/C++ Development Tools(CDT), Photran (Fortran IDE) and PTP. These plug-ins integrated the functionality of the TAU performance analysis system (configuration, instrumentation, PAPI counters, profile data management, performance analysis, visualization) into the existing workflows of these IDE components. Further work extended these Eclipse-TAU plug-ins to use our External Tools Framework (ETFw), a generalized performance tool integration method for Eclipse. ETFw has been contributed back to the Eclipse PTP project. We demonstrated the capabilities of ETFw by integrating the functionality of several third-party performance tools into Eclipse/PTP, including Valgrind, SCALASCA, and VampirTrace.

2.2 PERI Accomplishments

Our primary objective for work with PERI in Year 1 and 2 was to support the use of TAU in performance engineering practice, especially for petascale applications targeting the DOE leadership class facilities. To this end, our efforts focused on two PERI activities: participating in the PERI DB working group and working with the PERI Tiger Teams.

We substantially contributed to the PERI performance database working group (PERI DB) to design and develop performance database support for PERI performance engineering work. This included helping to define the metadata and performance data schemas. TAU was updated to be a PERI DB reference platform. Throughout the two years, we met regularly with group members, worked together on specification and interoperability issues, and contributed to demonstrations and presentations at the SC conference. Most importantly, we applied the PERI DB work in the course of PERI performance engineering activities.

The substantial part of our efforts were in PERI Tiger Team performance engineering projects. Our work focused on collection and analysis of weak-scaling performance data from S3D and GTC/GTS codes using the TAU on the leadership class facility (LCF) systems at ANL (Intrepid, IBM BG/P) and at ORNL (Jaguar, Cray XT4/5). Performance profiling and tracing experiments were conducted on these applications and the TAU measurements were collected in the performance database for analysis. The new data mining capabilities developed in the first year were applied for multi-experiment investigation and problem investigation.

Our efforts on the Gyrokinetic Toroidal Code (GTC) contributed to a better understanding of the dynamic runtime behavior of the main bottleneck, charged distribution (a scatter operation), true for most particle codes. The GTC development team was already aware of the performance bottleneck, but did not have a good sense as to the reason. TAU's dynamic phase profiling features were effective in characterizing the scatter-gather effects on cache and memory inefficiencies. This confirmed that the particles in the simulation are accessed with good spatial locality, but the grid cells have poor temporal locality. Hand-tuning techniques such as common sub-expression elimination, code movement, loop unrolling, and cache blocking were used to improve performance of the charge deposition routine by around 10 percent. The performance analysis indicated that changes to the data layout (i.e., the particle ordering) would be needed to obtain additional gains in performance.

The predominant amount of our PERI Tiger Team work was with the S3D application. We investigated the scalability of S3D, particularly in issues of load imbalances, on the ORNL Jaguar system. During the transition of Jaguar from a XT3 to XT4 platform, it was used in a hybrid configuration (XT3 + XT4). TAU's measurement, analysis, and visualization capabilities were used effectively at scale to uncover the nature of weak scaling performance. Hybrid runs proved perplexing in this regard because the runs experienced increased *MPI_Wait* times that were different from experiments on the XT3 or XT4 alone. The use of TAU metadata about the type of node used by each process allowed TAU's metadata reasoning to conclude that the faster XT4 memory system enabled it processors to arrive at the *MPI_Wait* earlier than the XT3 processors, leading to an imbalance.

We also looked at the effects of node mapping on the S3D code in the IBM BG/P architecture. TAU was used to collect S3D BG/P performance for jobs ranging from 1 to 30,000 cores. This weak-scaling study made apparent that time spent in communication routines began to dominate as the number of cores increased. In the 30,000 core case, the time spent in routines *MPI_Barrier*, *MPI_Wait* and *MPI_Isend* rose significantly. We further observed deviation between individual threads in time spent in communication routines. The pattern of deviation suggested a load imbalance impacted by node topology. We tested this hypothesis by running an 8000 core test with a random node mapping replacing the default. This resulted in a 6% speedup entirely due to MPI behavior. Other topologies were evaluated. The PerfExplorer tool was able to highlight the effects across application and MPI library routines.

The work with PERI in Years 1 and 2 proved the value of the new features we developed in TAU. In addition to the GTC and S3D Tiger Teams, the TAU tools were also used in PERI performance studies on the MILC and Chroma codes.

2.3 TASCS Accomplishments

Our work with the TASCS SciDAC project continued development support for the Common Component Architecture (CCA) environment, while bringing new performance analysis and database capabilities to bear on the computational quality of service (CQoS) problem. We improved our PDT source analysis tool to support the TASCS OnRamp project to ease the transition of legacy simulation codes to the CCA system. In addition, PDT developments were useful for updating our support for CCA instrumentation and proxy generation. The TAU Performance Component was implemented to enable performance measurement within the CCA framework. It is to date the only component to be produced that is fully CCA compliant.

We also incorporated the TAU PerfDMF and PerfExplorer tools into CQoS infrastructure for CCA to support performance analysis and decision-making for runtime adaptivity. Our approach used classification analysis in PerfExplorer to identify opportunities for CQoS decisions based on prior learned performance classes. For production application runs, the classifier is loaded into a CCA component and the best parameter setting (class) is obtained by querying the classifier with the current values of the application-specific metadata. These values are matched to the classification properties to find the best class selection for the parameters.

There were two specific CQoS investigations that we participated in with the CCA CQoS subgroup. First, TAU was used to characterize performance of linear and non-linear solvers for a variety of parameters so as to make intelligent configuration and runtime adaption decisions based on execution context. These were stored in the PerfDMF database and PerfExplorer evaluated features for data mining and decision rule support based on linear/non-linear solver performance, execution context, and current performance state.

Second, CQoS was incorporated in the GAMESS application. Here the goal was to match high performing algorithms to the characteristics of chemical models. TAU was used to capture metadata describing the chemical properties of a particular GAMESS execution and associate performance profiles which PerfExplorer then analyzed to find those metadata that best partition/cluster the performance space. From this, an efficient CQoS model could be produced for runtime decision control. In particular, it is important in GAMESS to suggest whether to use a direct or conventional method, given the other parameter selections. For each molecule, the associated basis functions roughly correlate with resource demand of the corresponding computations: the greater the number of basis functions, the more demanding the computation is expected to be. The choice of molecules are based on their importance in chemistry and biology as well as on characteristic types of chemical interactions they represent; also, computations of molecules of a similar size (that is, with similar number of atoms and basis functions) are routine in contemporary quantum chemistry. We constructed training experiments to learn the classifiers to choose between the direct and conventional methods, and other execution configuration parameters. A new PerfExplorer CCA component was released to support the classification work.

As with PERI, the University of Oregon was an active participant in the TASCS project. We attended every project meeting and even hosted meetings in Oregon. We also played a strong role in the development and maintenance of the CCA tutorial, including overhauling the performance section of the hands-on guide to include the new CCA components. Our group participated in CCA tutorial presentations at the SC and ACTS meetings.

2.4 Application Successes

An outgrowth of our PerfExplorer developments and CQoS work led to work on the GenIDLEST

application. Here we wanted to understand the scalability of OpenMP implementations and learn the best settings for OpenMP parameters on different platforms. We constructed PerfExplorer scripts to derive the inefficiency metrics, stall rates, and cache behavior. Rules were created to examine the results and derive prescriptions for parameters selection. Data locality was shown to be significant and we looked to first-touch policies on platforms for remedies. Scheduling alternatives and privatization control also were significant strategies to pursue for performance optimization. The work here demonstrated PerfExplorer's ability to automatically deduce causes of performance problems (e.g., cache inefficiencies) and identify possible reasons (e.g., sequential data initialization). The lesson learned here was that we need to provide feedback to the compiler to direct it to high value targets for optimization. The research was report in a SC 2008 paper.

3. Final Period Results

In the final period (Year 3 plus no-cost extension) of the DOE "competitiveness" project, we had three objectives: 1) continue support for PERI and TASCS initiatives, 2) translation of our TAU technology development to other applications, and 3) broaden our outreach to other DOE efforts. Our achievements in these areas are described below.

3.1 PERI Work

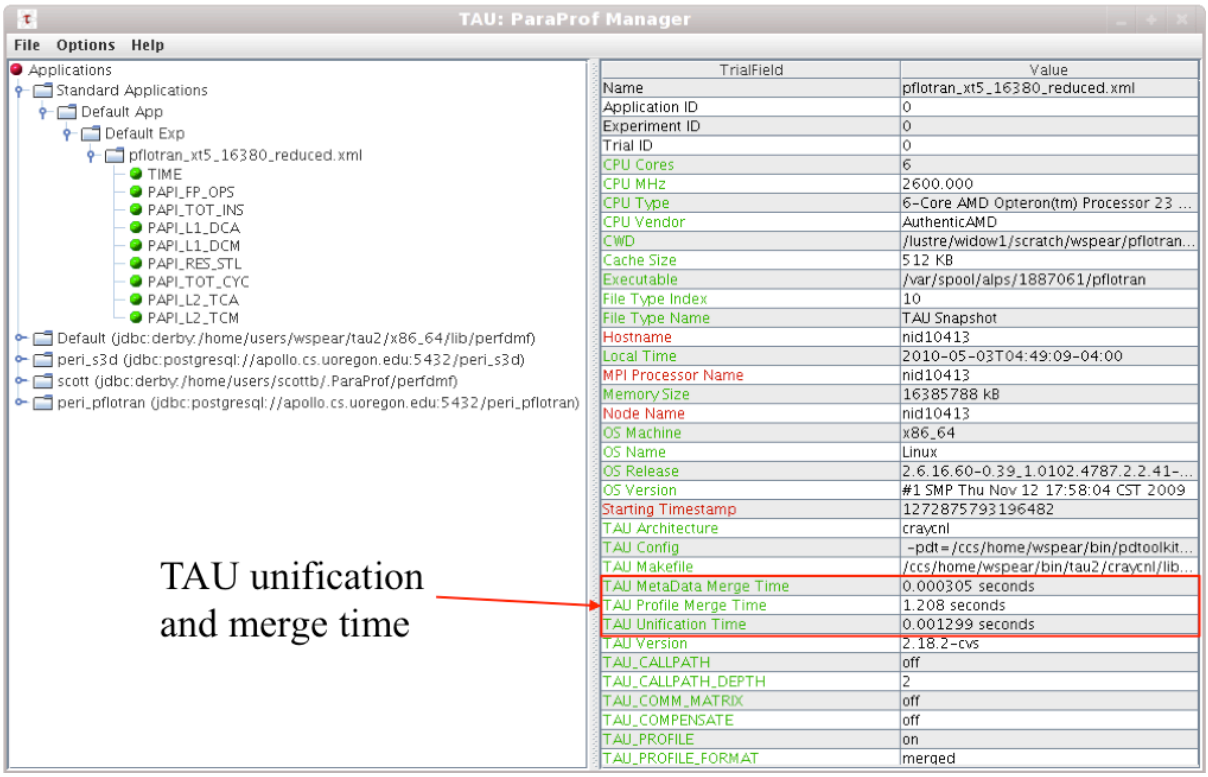
The primary work activity with the PERI initiative was the further involvement with the application Tiger Teams. PFLOTRAN, a reactive flow and transport code that uses PETSc as the basis for its parallel framework, became an important focus. We instrumented PFLOTRAN with TAU and ran experiments on the ORNL Jaguar (Cray XT4), UT Kraken (Cray XT5) platforms, as well as the ANL Intrepid (IBM BG/P) system. Our work was targeted to scaling experiments and I/O performance. Interestingly, during this time, PFLOTRAN also became a test problem for tool evaluation, as part of a Dagstuhl seminar on parallel performance tools. Thus, we tool the opportunity to improve the TAU measurement infrastructure with respect to online processing of performance data and demonstrated this with the PFLOTRAN application. Results from this work are reported below.

The PERI Tiger Team was interested in the scaling properties of PFLOTRAN from 10s of thousands of processors to over 100,000 processors. We conducted scaling experiments on Jaguar, Kraken, and Intrepid for PFLOTRAN at these scales. For TAU, these experiments required performance instrumentation. We use our PDT tool for source instrumentation of both PFLOTRAN and the PETSc library. Instrumentation of the PETSc library was challenging, mostly because of the time needed for recompilation, but PDT was successful in automatically inserting instrumentation for all PETSc routines. Full instrumentation of PFLOTRAN, the PETSc library, and the MPI library resulted in 1131 total active events at runtime. We ran a TAU measurement test on PFLOTRAN to determine those events with exclusive times greater than 1% of the total and created a selective instrumentation file for only those events. This effectively removed insignificant routines from consideration, and resulted in a partial instrumentation consisting of all PFLOTRAN routines, 44 MPI routines, and 19 PETSc routines. Subsequent PFLOTRAN experiments used this partial instrumentation for performance profile measurements with and without callpaths.

The Cray XT5 experiments highlight our achievements. For all of our measurement runs, we used PAPI metrics: total cycle counts as an execution time metric, and PAPI counters (*FP OPS*, *TOT IN*, *L1 DCA/DCM*, *L2 TCA/TCM*, *RES STL*) for observing hardware effects. Here we consider two experiments: one with 16K cores and one with 131K cores. With full instrumentation, a total of ~1.5 GB of performance profile data is created for a flat profile with 16K cores; ~27 GB is created for 131K cores. With partial instrumentation, a total of ~80 MB (16K) and ~1.4 GB (131K) is produced. Clearly, partial instrumentation can reduce the amount of data generated. However, turning on callpath profiling will results in more performance data being produced.

In general, we are concerned with performance data management as applications scale. One of TAU's scaling problems at this time was the fact that each core's profile was being written to a separate file. Also, the post-processing of parallel profiles to unify the performance event identifiers was causing verbose information to be stored with the measurement. These two problems were addressed in our work. First, we developed a "merged" parallel profile output format that allows all core profiles to be written to a single file. The performance data sizes above are for a merged profile file for 16K and 131K cores. The second step was to implement event unification before writing the merged file.

Figure 1. ParaProf's manager shows the PFLOTRAN experiment and metrics (left)



and metadata (right) where information about event unification and profile merge performance is recorded.

TAU performance measurement infrastructure is scalable because all performance data is kept local to the nodes. However, TAU assigns event identifiers dynamically as measured events occur during execution. Hence, IDs for the same events can be different between nodes. Unifying event IDs post-mortem requires information to associate each node's event name to its ID, and this full event information has to be kept in the performance data until then. If event unification can take place at the end of the application execution before the merged file is written, additional savings in data volume can be achieved. We implemented a scalable, distributed event unification algorithm using MPI that runs during TAU finalization. Remarkably, this produced a compact performance data size of 300 MB for the 16K full profile (vs. 1.5 GB) and 600 MB for the 131K core full profile (vs. 27 BG). Furthermore, the parallel event unification is fast, as shown in Figure 1 for the 16K case. The profile merge algorithm has a sequential I/O bottleneck because a single file is being written, but it is still just a few seconds. (For 131K, the event unification took 0.00041 seconds, and 12.96 seconds for profile merging.)

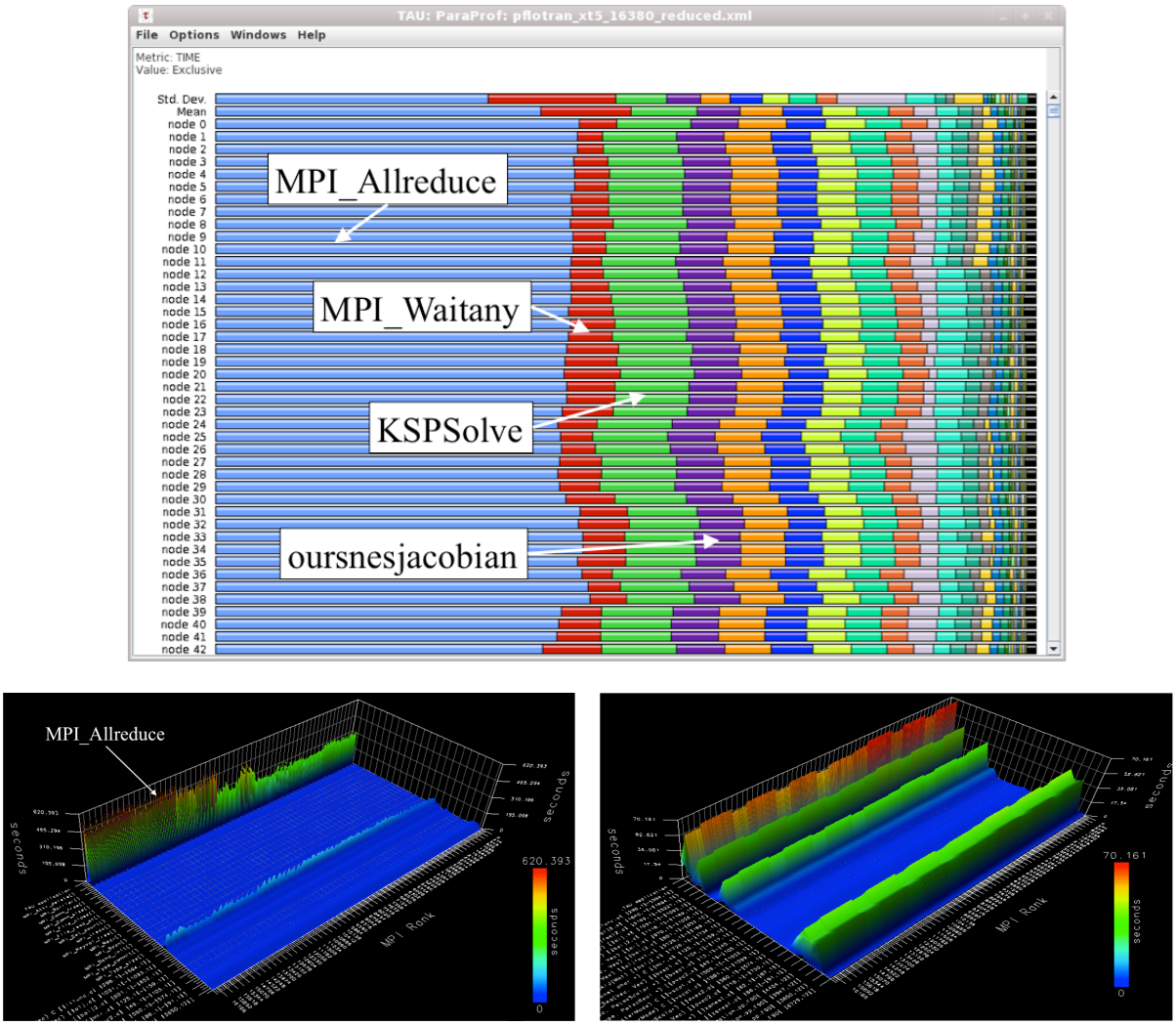


Figure 2. ParaProf's bargraph view of a 16K PFLOTRAN parallel profile on a Cray XT5 (top). A 3D view of the full parallel profile showing *MPI_Allreduce* dominance (bottom left). Removal of this event allows other events to be seen more clearly (bottom right).

The ParaProf "bargraph" display of the first 43 processes in a 16K PFLOTRAN parallel profile is shown in Figure 2. This interactive display allows the user to navigate the profile data set to investigate areas of interest. Here we see immediately the dominance of *MPI_Allreduce* and *MPI_Waitany* in the execution, suggesting communication overheads in collective operations. We also see relative uniformity in performance of events, although we are only seeing a subset of the processes. The "full profile" views in Figure 2 bear out this impression. When we compare the Cray XT5 MPI performance on PFLOTRAN with that of the IBM BG/P, we see significantly better relative behavior due to the BG/P's high-performance interconnection network and support for collective communication. However, it is also the case that the computation times are slower due to the differences in processor performance. This results in a more balanced performance perspective across the PFLOTRAN application.

As seen in Figure 1, all of the experiments we ran with TAU collected detailed performance counters. This information is important for understanding the performance interactions within the nodes. We can also turn on callpath profiling to get a better sense of how performance is distributed across the PFLOTRAN code.

Our positive experience with event unification on PFLOTRAN led us to consider the implementation of other profile analysis operations at the end of application execution. As a matter of course, anytime a parallel profile is loaded in ParaProf, the average, minimum, maximum, and standard deviation is computer for all profile events. Why not compute these values online? We developed parallel, scalable algorithms for this purpose. Again, the speed of the performance was impressive. This functionality is now turned on by default in TAU. More information about this work can be found in our TAU monitoring paper at the PROPER workshop.

3.2 TASCS Work

Our interactions with the TASCS project continued along two lines in the final project period. First, there was a significant push in TASCS to engage potential users of CCA, those integrating CCA methodology and components, and those componentizing their libraries. In addition to maintaining the CCA-related TAU technology, we contributed substantially to the CCA educational activities. These efforts include regular participation in the CCA tutorials, presented at SC and ACTS. These tutorials included hands-on training in CCA and we applied our LiveDVD technology (developed with NSF and DOE SBIR funding) to facilitate CCA software installation and usage in training sessions. See more discussion of ACTS and our LiveDVD support below.

Second, building on the positive results in TASCS CQoS activities, we continued to improve our capabilities in the TAU for sophisticated profile data analysis for informing high-level decision analysis. In particular, our colleagues at Argonne were requesting better mechanisms for specifying certain analyses to be applied within ParaProf and PerfExporer. Instead of building a new analysis component for each request, we decided that the best way to implement this would be to enable users to specify their own analysis operations through a programming interface.

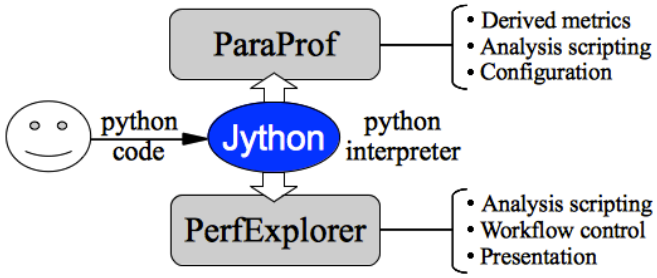


Figure 3. Design approach for embedding a scripting language (Python) interpreter in ParaProf and PerfExplorer, and functionality that is being developed with this support.

As shown in Figure 3, the idea is to open up the internal functionality of ParaProf and PerfExplorer for external use by allowing users to program new programming operations for performance data processing in Python. However, all of TAU's performance data analysis tools are written in Java. To allow Python scripts to be processed, we needed to embed a Java-based Python interpreter in the tools. The Jython interpreter was chosen because it is a robust, powerful implementation of Python in Java. Within ParaProf and PerfExplorer, we needed to expose the performance data accessors and analysis components through Python-callable interfaces. This was a straightforward process given the design of the tools and our earlier project work.

To demonstrate the productivity gains in performance analysis afforded by work, consider the request of our Argonne colleague (Boyana Norris) to derive new performance metrics from measured data. The general idea is to create new metrics that characterize performance data in different ways. These “derived metrics” can then be interpreted relative to and in combination with each other to expose performance anomalies. Figure 4 shows the output from a Python script written by Norris and input to PerfExplorer. This script is being used to analyze a specific measurement trial of the FLASH application where profile

events are processed with respect to metric expressions to determine "Top 10" events for different derived metric "features." The features of interest are:

P_WALL_CLOCK_TIME intensive events

floating-point intensive events (relative, high ratio of FP to non-FP instructions for this event only)

FP-bound events (absolute, high ratio of this event's FP instructions to all program instructions)

memory-bound events (relative, events with high memory costs per cycle per event)

memory-bound events (absolute, events with high memory costs per cycle for all program cycles)

floating point inefficiency events (relative)

floating point inefficiency events (absolute)

Each one of these (except for the first) is derived from a Python metric expression. Once the derived metrics are calculated, the Python script then determines the top events manifesting the features. These are then listed for each derived metric. Again, the goal is to generate different perspectives on the performance data through sophisticated analysis expressions that are meaningful to the user and useful in performance problem classification. In the case of FLASH, they were looking for performance symptoms that would allow them to pinpoint computations that were the best candidates for locality-improving optimizations, based on certain measures of FP inefficiencies. These identified regions where there was a relatively high cache miss rate, as well as proportionately high (to the whole computation) number of floating-point operations.

The outcome for FLASH of this new TAU capability was important for defining new directions for auto-tuning. Building this support in the TAU tools will also be significant for extending their functionality in the future.

3.3 Application Case Study: NWChem

Consistent with the project's goals of building knowledge-based support for analysis, there is a need to enhance the measurement capabilities to capture performance features that reflect evolving parallel programming models. The use of global address space languages and one-sided communication is gaining attention, but there is a lack of good evaluative methods to observe multiple levels of performance, from programming interfaces to hardware counters. This makes it difficult to isolate the cause of performance deficiencies on current systems, and to understand the fundamental limitations of system design for future improvement. During the final period of the project, we were presented the opportunity to apply several achievements of the DOE project to an important application utilizing global address space and one-sided communication functionality, NWChem. NWChem also presented a challenge to TAU to improve its support for one-sided communication. The following describes our work with PNNL and ANL, which will appear in Concurrency and Computation: Practice and Experience.

```
Main Event:  FLASH [{Flash.F90} {31,1}-{45,17}]


Top 10 P_WALL_CLOCK_TIME intensive events:
1.1687393E7    HYDRO_1D [{hydro_1d.F90} {153,1}-{677,25}]
3081856.0        *** custom:hy_block
2840283.0        *** custom:hy_ppm_sweep
1664883.0        SIMULATION_INITBLOCK [{Simulation_initBlock.F90} {45,1}-{306,35}]
1183784.0        *** custom:eos gc
1082294.0        AMR_GUARDCELL [{mpi_amr_guardcell.F90} {87,7}-{473,34}]
1046692.0        AMR_1BLK_GUARDCELL_SRL [{amr_1blk_guardcell_srl.F90} {10,7}-{856,43}]
996254.0         MPI_AMR_LOCAL_SURR_BLKS [{mpi_amr_local_surr_blks.F90} {88,7}-{529,44}]
822190.0         GR_SANITIZEDATAAFTERINTERP [{gr_sanitizeDataAfterInterp.F90} {67,1}-{208,41}]
732730.0         MPI_AMR_1BLK_RESTRICT [{mpi_amr_1blk_restrict.F90} {117,7}-{1406,42}]


Top 10 floating-point intensive events (relative, high ratio of FP to non-FP instructions for this event only):
0.460917         GR_UPDATEDATA [{gr_updateData.F90} {25,1}-{105,28}]
0.397660         GR_MARKREFINEDEREFINE [{gr_markRefineDerefine.F90} {45,1}-{504,36}]
0.299210         HY_PPM_UPDATESOLN [{hy_ppm_updateSoln.F90} {90,1}-{775,33}]
0.269704         MAP1 [{umap.F} {498,9}-{824,11}]
0.244240         MAP2 [{umap.F} {1389,9}-{1900,11}]
0.208284         RIEMAN [{rieman.F90} {90,1}-{480,21}]
0.170483         STATES [{states.F90} {107,1}-{625,21}]
0.165641         HYDRO_COMPUTEDT [{Hydro_computeDt.F90} {60,1}-{372,30}]
0.161455         MPI_SETUP [{mpi_lib.F90} {265,7}-{376,30}]
0.133053         DETECT [{detect.F90} {45,1}-{136,21}]


Top 10 FP-bound events (absolute, high ratio of this event's FP instructions to all program instructions):
0.002911         RIEMAN [{rieman.F90} {90,1}-{480,21}]
0.002908         MONOT [{monot.F90} {40,1}-{109,20}]
0.002589         INTERP [{interp.F90} {44,1}-{101,23}]
0.002046         STATES [{states.F90} {107,1}-{625,21}]
0.001107         INTRFC [{intrfc.F90} {111,3}-{338,21}]
0.000975         DETECT [{detect.F90} {45,1}-{136,21}]
0.000946         HY_PPM_UPDATESOLN [{hy_ppm_updateSoln.F90} {90,1}-{775,33}]
0.000942         SIMULATION_INITBLOCK [{Simulation_initBlock.F90} {45,1}-{306,35}]
0.000907         HYDRO_1D [{hydro_1d.F90} {153,1}-{677,25}]
0.000724         AMR_RESTRICT_UNK_GENORDER [{amr_restrict_unk_genorder.F90} {14,7}-{235,46}]


Top 10 memory-bound events (relative, events with high memory costs per cycle per event):
5.106236         AMR_FLUX_CONSERVE [{mpi_amr_flux_conserve.F90} {90,7}-{132,38}]
4.663727         AMR_REFINE_DEREFINE [{mpi_amr_refine_derefine.F90} {91,7}-{681,40}]
4.473107         MPI_AMR_WRITE_RESTRICT_COMM [{mpi_amr_store_comm_info.F90} {482,7}-{555,48}]
4.419078         MPI_AMR_WRITE_PROL_COMM [{mpi_amr_store_comm_info.F90} {168,7}-{234,44}]
4.207310         AMR_RESTRICT_BND_DATA [{mpi_amr_restrict_bnd_data.F90} {15,7}-{410,42}]
4.200861         MPI_AMR_READ_GUARD_COMM [{mpi_amr_store_comm_info.F90} {88,7}-{162,44}]
4.125719         MPI_AMR_READ_PROL_COMM [{mpi_amr_store_comm_info.F90} {240,7}-{317,43}]
4.112978         AMR_1BLK_TO_PERM [{amr_1blk_to_perm.F90} {15,7}-{264,37}]
4.102125         GR_SANITIZEDATAAFTERINTERP [{gr_sanitizeDataAfterInterp.F90} {67,1}-{208,41}]
3.972938         GR_UPDATEDATA [{gr_updateData.F90} {25,1}-{105,28}]


Top 10 memory-bound events (absolute, events with high memory costs per cycle for all program cycles):
0.018345         GR_SANITIZEDATAAFTERINTERP [{gr_sanitizeDataAfterInterp.F90} {67,1}-{208,41}]
0.015368         *** custom:hy_ppm_sweep
0.012984         *** custom:hy_block
0.012354         HYDRO_1D [{hydro_1d.F90} {153,1}-{677,25}]
0.012107         AMR_FLUX_CONSERVE [{mpi_amr_flux_conserve.F90} {90,7}-{132,38}]
0.011350         STATES [{states.F90} {107,1}-{625,21}]
0.011254         AMR_GUARDCELL [{mpi_amr_guardcell.F90} {87,7}-{473,34}]
0.008848         RIEMAN [{rieman.F90} {90,1}-{480,21}]
0.008125         EOS_GETDATA [{Eos_getData.F90} {82,1}-{256,26}]
0.006954         INTRFC [{intrfc.F90} {111,3}-{338,21}]


Top 10 floating point inefficiency events (relative):
0.216254         GR_UPDATEDATA [{gr_updateData.F90} {25,1}-{105,28}]
0.203574         GR_MARKREFINEDEREFINE [{gr_markRefineDerefine.F90} {45,1}-{504,36}]
0.134752         HY_PPM_UPDATESOLN [{hy_ppm_updateSoln.F90} {90,1}-{775,33}]
0.105361         HYDRO_COMPUTEDT [{Hydro_computeDt.F90} {60,1}-{372,30}]
0.095866         MPI_SETUP [{mpi_lib.F90} {265,7}-{376,30}]
0.095160         MAP1 [{umap.F} {498,9}-{824,11}]
0.084138         MAP2 [{umap.F} {1389,9}-{1900,11}]
0.061689         GR_SANITIZEDATAAFTERINTERP [{gr_sanitizeDataAfterInterp.F90} {67,1}-{208,41}]
0.050163         RIEMAN [{rieman.F90} {90,1}-{480,21}]
0.039648         MPI_AMR_GLOBAL_DOMAIN_LIMITS [{mpi_amr_global_domain_limits.F90} {60,10}-{109,49}]


Top 10 floating point inefficiency events (absolute):
0.000048         MONOT [{monot.F90} {40,1}-{109,20}]
0.000044         INTERP [{interp.F90} {44,1}-{101,23}]
0.000035         INTRFC [{intrfc.F90} {111,3}-{338,21}]
0.000016         SIMULATION_INITBLOCK [{Simulation_initBlock.F90} {45,1}-{306,35}]
0.000012         RIEMAN [{rieman.F90} {90,1}-{480,21}]
0.000010         HYDRO_1D [{hydro_1d.F90} {153,1}-{677,25}]
0.000008         EOS_WRAPPED [{Eos_wrapped.F90} {93,1}-{198,26}]
0.000005         *** custom:hy_block
0.000005         SIM_FIND [{Simulation_initBlock.F90} {524,1}-{561,23}]
0.000005         STATES [{states.F90} {107,1}-{625,21}]
```

Figure 4. Derived metric analysis used in the FLASH application.

NWChem is a computational chemistry suite supporting electronic structure calculations using a variety of chemistry models. NWChem employs Global Arrays (GA) as the underlying one-sided programming model. GA provides a global address space view of the distributed address spaces of different processes. Aggregate Remote Memory Copy Interface (ARMCI) is the communication substrate that provides the remote memory access functionality used by GA. Being able to observe the performance characteristics of ARMCI in support of Global Arrays and in the context of NWChem is a key requirement for future development and optimization. Of specific interest to multicore parallelism in NWChem is that one-sided communication requires remote agency, leading communication runtime systems to often spawn a thread for processing incoming one-sided message requests. In ARMCI this thread is known as the *data-server*. It is also important for performance optimization to observe the role of core assignment between computation and communication.

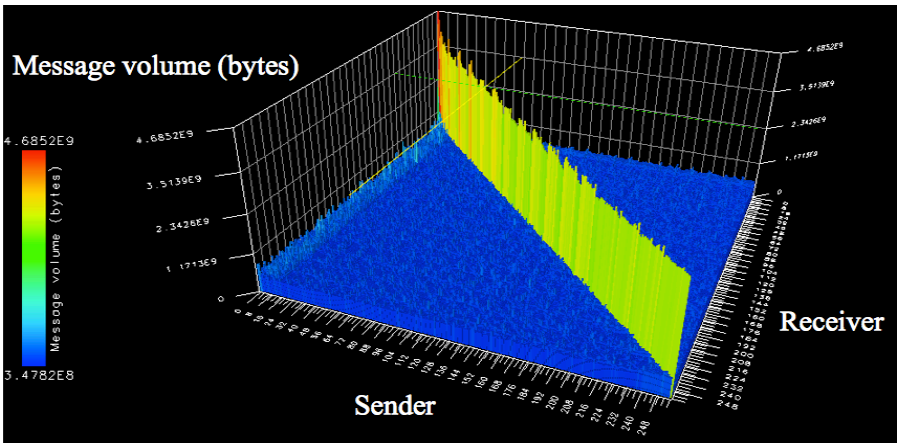


Figure 5. Derived metric analysis used in the NWChem application.

An important challenge for profiling the NWChem software was to capture events associated with the use of Global Arrays and the ARMCI communication substrate. The ARMCI instrumentation approach we developed is similar to what is used in the MPI library whereby an alternate "name- shifted" interface to the standard routines is created (called PMPI in the case of MPI, where 'P' stands for "profiling") and a new library is provided to substitute for the original calls. In the spirit of PMPI, we call the profiling interface for ARMCI, *PARMCI*. (PARMCI is now included as part of the ARMCI distribution.) We use PARMCI to create a TAU-instrumented library for ARMCI that captures entry/exit events and make performance measurements of time as well as communication statistics (e.g., bytes transmitted) between sender and receivers. The TAU ParaProf tool was enhanced to display communication statistics to highlight gross patterns of interaction, as shown in Figure 5. Utilizing TAU in NWChem required additional enhancements of instrumentation and measurement support, as described in the paper.

An important goal in analyzing one-sided communication in NWChem was to understand the interplay between the data-server and compute processes as a function of scale. However, as the job is strong-scaled to larger numbers of nodes, not only does the computational work per node decrease, but the fragmentation of data across the system leads to an increase in the total number of messages. We conducted scaling experiments on three systems in this study:

Fusion: A 320-node Linux cluster with dual-socket Intel Nehalem-series quad-core processors (Xeon X5550) connected by QDR IB (Mellanox Technologies MT26428). This machine is operated by the Argonne Laboratory Computing Resource Center (LCRC).

Chinook: A 2310-node Linux supercomputer with dual-socket AMD Barcelona-series quad-core processors (AMD Opteron 2354) connected by DDR IB (Mellanox Technologies MT25418

NICs and Voltaire switches). This machine is operated by Pacific Northwest National Laboratory's Molecular Science Computing Facility.

Intrepid: The Argonne Leadership Computing Facility operates Intrepid and Surveyor, which are 40- and 1-rack Blue Gene/P systems, respectively. In contrast to the Linux clusters above, the implementation of ARMCI on Blue Gene/P does not use the data server due to the existence of active-message functionality within DCMF. True passive progress is achieved either with a communication helper thread continuously polling for incoming active-messages or by operating system interrupts, which are extremely lightweight in the BG/P compute node kernel (CNK) relative to Linux.

Our experiments varied the number of nodes, cores-per-node, and whether or not memory buffers were "pinned", that is, registered with the NIC and ineligible for paging. On BG/P, paging is disabled in the kernel and memory-registration of the entire address space is trivial, hence there is no comparison to be made with respect to buffer pinning. There is an important trade-off between using all available processing power for numerical computation and dedicating some fraction of the cores to communication. Understanding these trade-offs as a function of scale is critical for adapting software for new platforms which may have widely varying capability for hardware offloading of message-processing, such as support for contiguous and/or non-contiguous RDMA operations. The strong scaling experiments with Fusion exposed the effects of growing effects of communication (see Figure 6) in NWChem when care is not taken to properly configure the application for the communications system. Use of a separate thread for the data-server only stalls the onset. It is only when the communication buffers are pinned that performance scaling in achieved (see Figure 7).

To test the generality of our experiments on Fusion, we performed similar tests on Chinook. A key difference is that the newer Intel Nehalem processors have approximately twice the memory bandwidth per core and per node, and the system software runs HPC-oriented Linux and communication stack. Figure 8 compares the total execution time and relative efficiency of different experiments on Fusion and Chinook.

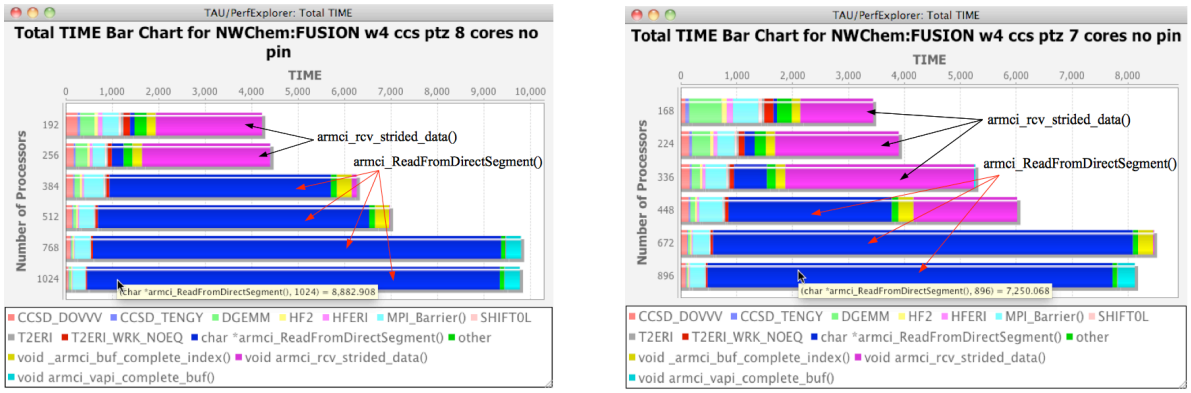


Figure 6. PerfExplorer graphs are automatically generated from scaling experiment profiles, highlighting the top contributors to execution time. ARMCI communication overheads increase with increasing number of processors, with and without a help thread for communication.

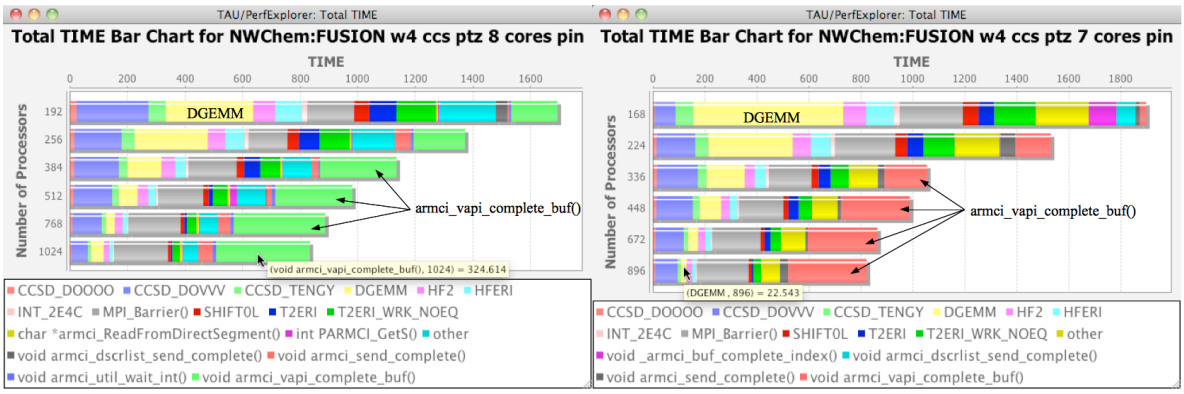


Figure 7. Pinning communication buffers on Fusion results in performance scaling advantages.

Here we see a reduced dependence on the use of pinning versus the number of compute cores used on Chinook. On the other hand, it is much more sensitive to how cores are allocated. In particular, using all 8 cores per node for computation increases the time for certain procedures markedly. With 32 nodes and 8 cores/node computing, the time spent generating atomic integrals is 3 and 9 times greater than when only 6 cores per node are used for the pinned and non-pinned cases, respectively. Similarly, on 128 nodes, using 8 cores per node for computation increases the wall time by approximately 3 times the 6- and 7-compute core per node cases.

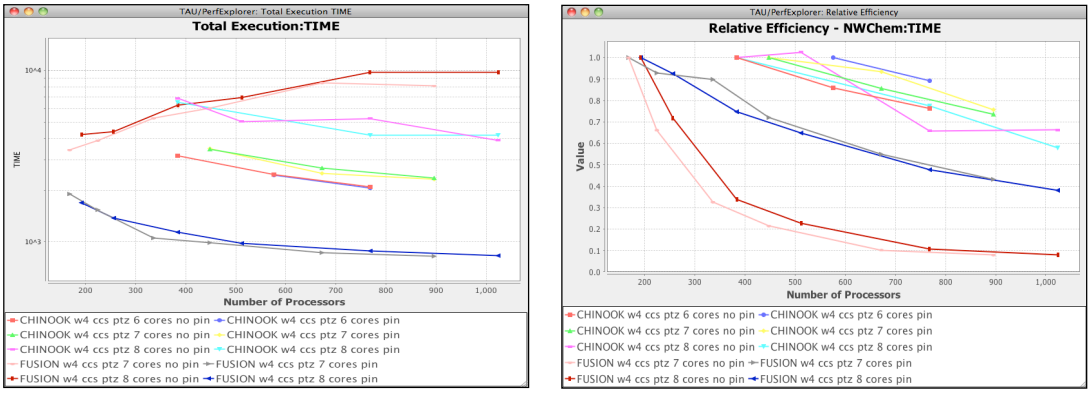


Figure 8. Total time and relative efficiency of different numbers of cores and pinning on Chinook and Fusion.

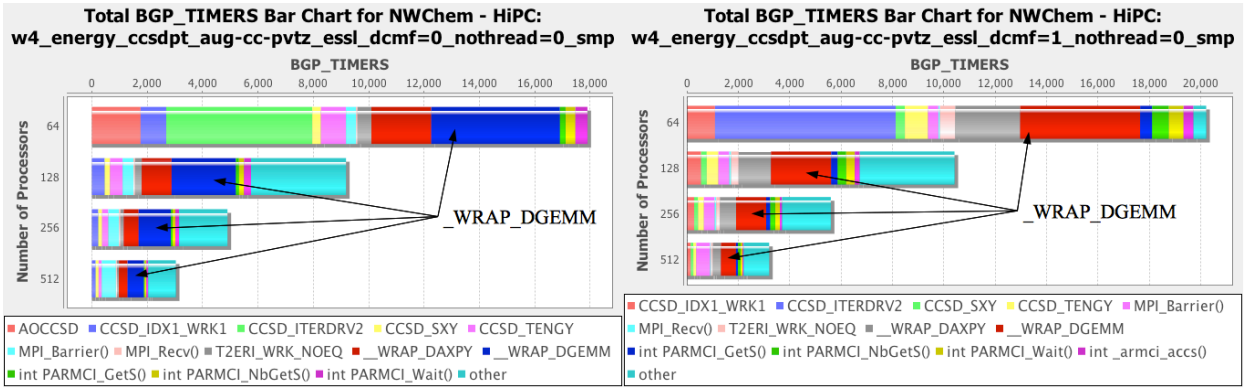


Figure 9. Scaling results are compared for cases with (dcmf=0) and without (dcmf=1) a communication helper thread.

When we look at the Intrepid BG/P results in Figure 9, we see the influence of an entirely different

systems context for our performance scaling experiments. BGP not only has very different hardware — low-memory, slow-clock-rate processors, no local disk, and an extremely low-latency network with modest bandwidth network — but it uses a lightweight operating system which does not permit Linux-style oversubscription, nor does it support SysV shared-memory. As such, the implementation of ARMCI is quite different and relies heavily upon active-message capability within DCMF, which is enabled by lightweight operating system interrupts. Preliminary investigations of ARMCI performance on BG/P resulted in a primitive communication helper thread (CHT) being added to the ARMCI implementation. The performance results demonstrate the utility of this approach to asynchronous progress relative to interrupts. Both the CHT and interrupt-mode provide a means to achieve passive- target progress in one-sided communication, although only the CHT requires a dedicated core. We compare these two contexts for running NWChem across a range of node counts (64, 128, 256 and 512) which is approximately comparable to the range used on Chinook and Fusion. Our NWChem calculations on BG/P executed in SMP and DUAL mode – 1 and 2 processes per node respectively.

What is remarkable in these results is that the ARMCI calls are barely noticeable, whereas normally negligible BLAS2 operations (e.g., DAXPY) show up in an unusually significant way. This result is not surprising due to the low clock-frequency of the BG/P processor and relatively low bandwidth from L2 cache. As a consequence, the scaling on BG/P is excellent, since the absence of pathological IB communication behavior allows for nearly halving of total execution time with a doubling of node count.

The NWChem performance analysis exercised the full capabilities of TAU enhancements made possible by the DOE "competition" project. The role of automated instrumentation, measurement, and analysis in TAU was invaluable for understanding the performance behavior of a complex code such as NWChem. The NWChem code base is millions of lines, in addition to the tens of thousands of lines of code active in GA and ARMCI for a given interconnect. Without automated source instrumentation and profiling hooks to both MPI and ARMCI, it would not have been possible to reliably identify the performance issues described. In particular, understanding the performance effects of one-sided communication is non-trivial. While the remote target is passive from a programmer perspective, passive-target progress requires significant resources on every node, especially since remote accumulate requires floating-point computation on top the memory operations required for packing and unpacking of non-contiguous messages. More rudimentary profiling techniques are not useful for analyzing the behavior of an asynchronous agent such as the ARMCI data server. Advanced performance tools will become more important with increased interest in one-sided programming models in both GA and PGAS languages.

3.4 DOE ASCR Outreach – ACTS

Throughout the DOE "competition" project, we have looked for avenues to translate our efforts to the broader DOE ASCR community. In particular, we have been actively involved in the DOE Advanced CompuTational Software (ACTS) collection and project. As seen in Figure 10, TAU is a part of the ACTS collection, and has been so for over five years. As part of the collection, we validate TAU with every one of the other ACTS packages to guarantee compatibility on all the platforms where ACTS software is installed.

Our ACTS involvement does not end there. We have actively participated in every one of the ACTS workshops, giving tutorials on TAU and assisting in hands-on training. The TAU project created a LiveDVD that configures parallel software development environments and performance tools for a single Linux distribution which can boot up on a user's laptop or workstation, natively or on a virtual machine. Our technology has been used for the ACTS workshops for the last three years. The LiveDVD disk for the 2010 ACTS workshop is shown in Figure 11. The different packages are listed and include several performance tools, the Eclipse/PTP framework, and CCA.

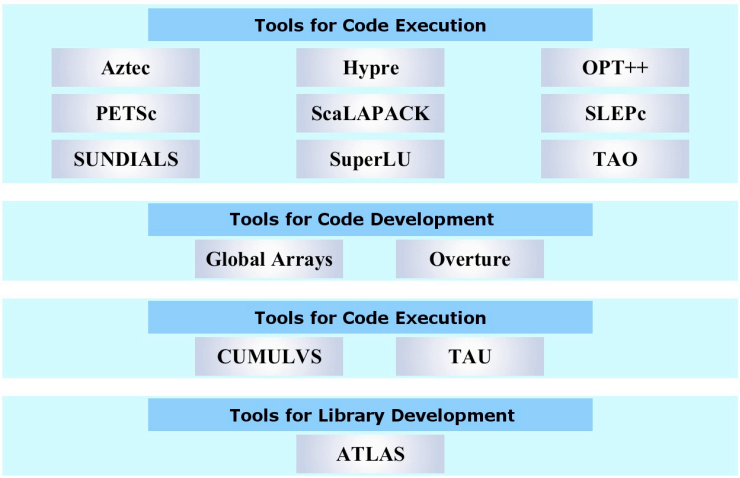


Figure 10. ACTS software collection architecture and packages.

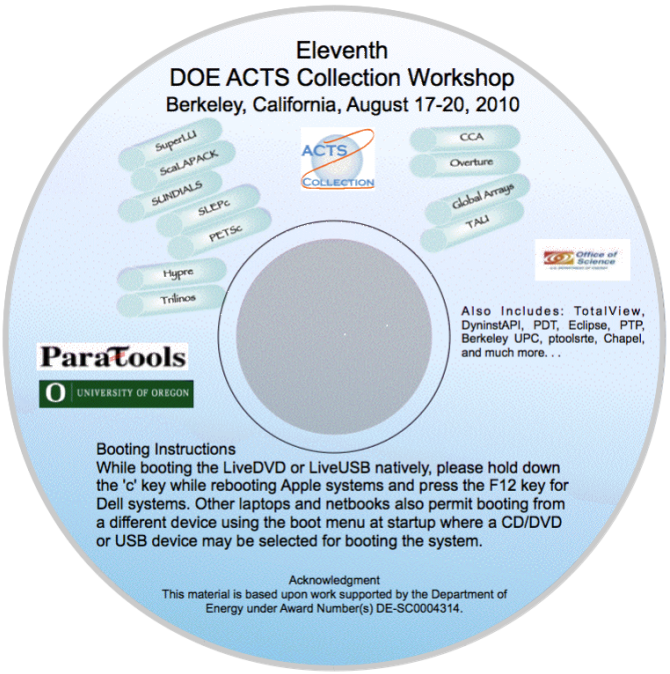


Figure 11. ACTS software collection architecture and packages.

4. Other Information

During its lifetime, the project supported two Ph.D. graduate students, two Master's graduate students, three staff members at the Performance Research Laboratory (PRL), a PRL system administrator, and the project PIs. One graduate student, Kevin Huck, completed his Ph.D. degree during the second year and went to work at the Barcelona Supercomputing Center.

It was important for our project to maintain frequent and consistent connections to the PERI and TASCS groups through meeting participation, conferences calls, and visits to DOE laboratories. For both PERI and TASCS, we hosted meetings in Oregon at various points during the project period. We are currently scheduled to host a PERI meeting in Eugene, Oregon in September 2011. Similarly, we regularly attended the ACTS workshops and will be doing so again August 16-19, 2011 at LBL.

Lastly, we participated in all of the DOE SciDAC Center for Scalable Application Development Software

(CScADS) summer workshops on performance tools and applications.

4. Conclusion

The DOE "competitiveness" project at the University of Oregon has achieved all of its objectives to develop advanced performance technology that supports knowledge-based analysis of parallel performance measurements and to translate those achievements to the PERI and TASCS activities. All of the technology is delivered in the open source TAU performance system for distribution. In this way, results from the project more broadly benefit DOE projects. In addition to PERI, TASCS, and ACTS, we were involved with the DOE SciDAC CFRFS and FACETS projects during the project where our TAU performance tools were applied.

Presentations:

We gave the following presentations during the project period:

1. S. Shende, "TAU Performance System," LLNL, May 3, 2007.

2. S. Shende, "Parallel Performance Evaluation Tools for HPC Systems," tutorial, *Linux Cluster Institute Conference*, South Lake Tahoe, May 14, 2007.

3. S. Shende, "Parallel Performance Evaluation using the TAU Performance System Project," *DOE CScADS Workshop on Performance Tools for Petascale Computing*, Snowbird, UT, July 17, 2007.

4. S. Shende, "TAU Performance System," *DOE CScADS Workshop on Petascale Architectures and Performance Strategies*, Snowbird, UT, July 24, 2007.

5. S. Shende, "Performance Evaluation using the TAU Performance System," *TACC Summer Institute*, U. Texas, Austin, August 17, 2007.

6. K. Huck, "Knowledge Support for Parallel Performance Data Mining, Code Instrumentation and Modeling for Parallel Performance Analysis," *Dagstuhl Seminar*, Dagstuhl, Germany, August 19-24, 2007.

7. S. Shende, "TAU Performance System," *DOE ACTS Workshop*, Berkeley, CA, August 24, 2007.

8. K. Huck, "Scalable Performance Analysis with TAU, PerfDMF and PerfExplorer," Forschungszentrum, Juelich, Germany, August 28, 2007.

9. S. Shende, "TAU Performance System," Performance Tools BOF, *International Conference for High Performance Computing, Networking, Storage, and Analysis (SC07)*, Reno, NV, November 13, 2007.

10. K. Huck, "Scalable, Automated Performance Analysis with TAU and PerfExplorer," *Parallel Computing Conference*, Aachen and Juelich, Germany, September 4-7, 2007.

11. S. Shende, "TAU: Performance Technology for Productive, High Performance Computing," seminar at ORNL, February 5, 2008.

12. K. Huck, "PERI Database Working Group: Status Update," *PERI Meeting*, UCSD, February 25-26, 2008.

13. S. Shende, "TAU: Performance Technology for Productive, High Performance Computing," seminar at MCS, ANL, May 2, 2008.

14. S. Shende, "Future Tool Overview and Plans for Future Tools (What they will do)," *Petascale Summer Workshop*, *TeraGrid 2008*, Las Vegas, June 9-13, 2008.

15. K. Huck, "Using the TAU Performance Analysis System on the Blue Gene/P," *ALCF INCITE Workshop*, ANL, Argonne Leadership Computing Facility, May 7-8, 2008.

16. T. Drummond, O. Marques, J. Roman, and S. Shende, "Computational Tools for Rapid and Reliable Development of High Performance Applications," tutorial, *International Meeting on High Performance Computing for Computational Science (VECPAR 2008)*, Toulouse, France, June 24-27, 2008.

17. K. Huck, "Integrating Knowledge, Automation, and Persistence with PerfExplorer and PerfDMF," *DOE CScADS Workshop on Performance Tools for Petascale Computing*, Snowbird, UT, July 21-24, 2008.

18. S. Shende, "TAU Parallel Performance System," *Leap to Petascale Workshop*, ANL, Argonne Leadership Computing Facility, July 28-31, 2008.

19. S. Shende, "TAU Performance System," *DOE ACTS Workshop*, Berkeley, CA, August 19-22, 2008.

20. K. Huck, "Knowledge Support for Parallel Performance Data Mining," Doctoral Showcase, *International Conference for High Performance Computing, Networking, Storage and Analysis (SC08)*, Austin, TX USA, November 20, 2008.

21. K. Huck, "Capturing Performance Knowledge for Automated Analysis," Technical Session, *International Conference for High Performance Computing, Networking, Storage and Analysis (SC08)*, Austin, TX USA, November 20, 2008.

22. A. Malony, F. Wolf, and D. Skinner, "Tools for High Productivity Supercomputing," BOF, *International Conference for High Performance Computing, Networking, Storage, and Analysis (SC08)*, Austin, TX USA, November 20, 2008.

23. S. Shende, A. Malony, R. Kufrin, R. Reddy, S. Moore, "Parallel Performance Evaluation Tools," Tutorial, *LCI Conference*, Boulder, CO, March 9, 2009.

24. B. Mohr, M. Schulz, D. Gunter, K. Huck, X. Wu, "A Proposal for a Profiling Data Exchange Format," *DOE CScADS Workshop on Performance Tools for Petascale Computing*, Tahoe City, CA, July 20-23, 2009.

25. A. Malony, "Performance Measurement and Analysis of Heterogeneous Systems: Tasks and GPU Accelerators," *DOE CScADS Workshop on Performance Tools for Petascale Computing*, Tahoe City, CA, July 20-23, 2009.

26. W. Spear, "Parallel Performance Evaluation with TAU," *DOE CScADS Workshop on Leadership-class Machines, Petascale Applications, and Performance Strategies*, Tahoe City, CA, July 27-30, 2009.

27. S. Shende, "TAU Performance System," DOE *ACTS Workshop*, Berkeley, CA, August 18-21, 2009.

28. A. Malony, "PFLOTRAN Meets TAU," *Dagstuhl Seminar on Program Development for Extreme-Scale Computing*, Proceedings 10181, May 2-7, 2010.

29. A. Malony, "Hybrid Parallel Performance Measurement and Analysis," *Dagstuhl Seminar on Program Development for Extreme-Scale Computing*, Proceedings 10181, May 2-7, 2010.

30. L. Drummond, O. Marques, S. Shende, and J. Roman, "Computational Tools for Rapid and Reliable Development of High Performance Applications," tutorial, *International Meeting on High Performance Computing for Computational Science (VECPAR 2010)*, June 23-25, 2010.

31. A. Malony, "TAU Potpourri and Working with Open Components, Interfaces, and Environments," *DOE CScADS Workshop on Performance Tools for Petascale Computing*, Snowbird, UT, August 2-5, 2010.

32. S. Shende, "TAU Performance System," DOE *ACTS Workshop*, Berkeley, CA, August 17-20, 2010.

33. M. Geimer, A. Knupfer, R. Kufrin, S. Moore, and S. Shende, "Hands-on Practical Parallel Application Performance Engineering using PAPI, PerfSuite, Scalasca, Vampir, and TAU," tutorial, *International Conference for High Performance Computing, Networking, Storage, and Analysis (SC10)*, November 15-18, 2010.

34. A. Malony, "Performance Visualization, Integrated Profiling, and Kernel Measurement," *DOE CScADS Workshop on Performance Tools for Petascale Computing*, Tahoe City, CA, August 1-4, 2011.

References:

1. D. Gunter, K. Huck, K. Karavanic, J. May, A. Malony, K. Mohror, S. Moore, A. Morris, S. Shende, V. Taylor, X. Wu, and Y. Zhang, "Performance Database Technology for SciDAC Applications," *Journal of Physics: Conference Series*, Vol. 78, 24--28 June 2007, Boston Massachusetts, USA.

2. Y. Zhang, R. Fowler, K. Huck, A. Malony, A. Porterfield, D. Reed, S. Shende, V. Taylor, and X. Wu, "US QCD Computational Performance Studies with PERI," *Journal of Physics: Conference Series*, Vol. 78, 24–28 June 2007, Boston Massachusetts, USA.

3. Li Li, "Model-based Automated Parallel Performance Diagnosis," Ph.D. thesis, University of Oregon, June 2007.

4. K. Huck, A. Malony, S. Shende, and A. Morris, "Scalable, Automated Performance Analysis with TAU and PerfExplorer," *Parallel Computing: Architectures, Algorithms, and Applications*, C. Bischof, M. Bucker, P. Gibbon, G.R. Joubert, T. Lippert, B. Mohr, and F. Peters (Eds.), NIC Serices, Vol. 38, Aachen, Germany, pp. 629–636, 2007. Reprinted in: *Advances in Parallel Computing*, Vol. 15, 2007.

5. S. Moore, "Porting and Performance Improvement of GTC_S," PERI GTC Tiger Team report, July 11, 2007.

6. H. Chen, A. Choudhary, B. de Supinski, M. DeVries, E. R. Hawkes, S. Klasky, W. K. Liao, K. L. Ma, J. Mellor-Crummy, N. Podhorski, R.Sankaran, S. Shende, C. S. Yoo, "Terascale Direct Numerical Simulations of Turbulent Combustion using S3D," (to appear) IOP Journal, 2008.

7. N. Trebon, A. Morris, J. Ray, S. Shende, A. Malony, "Performance Modeling of Component Assemblies," Concurrency and Computation: Practice and Experience, Vo. 19 No. 5, pp. 685-696, 2007.W. Spear, A. Malony, A. Morris, and S. Shende, "Performance Tool Workflows," *International Conference on Computational Science (ICCS 2008)*, Springer-Verlag, pp. 276–285, May, 2008.

8. K. Huck, A. Malony, S. Shende, and A. Morris, "Knowledge Support and Automation for Performance Analysis with PerfExplorer 2.0," *Journal of Scientific Programming*, special issue on Large-Scale Programming Tools and Environments, 16(2-3):123–134, 2008.

9. A. Malony, S. Shende, A. Morris, S. Biersdorff, W. Spear, K. Huck, and A. Nataraj, "Evolution of a Parallel Performance System," *2nd International Workshop on Tools for High Performance Computing*, High Performance Computing Center Stuttgart (HLRS), Eds. M. Resch, R. Keller, V. Himmler, B. Krammer, and A. Schulz, Springer-Verlag, pp. 169–190, July, 2008.

10. A. Morris, W. Spear, A. Malony, and S. Shende, "Observing Performance Dynamics using Parallel Profile Snapshots," *European Conference on Parallel Processing (EuroPar 2008)*, Springer, LNCS 5168, pp. 162–171, August, 2008.

11. K. Huck, W. Spear, A. Malony, S. Shende, and A. Morris, "Parametric Studies in Eclipse with TAU and PerfExplorer," *Workshop on Productivity and Performance (PROPER 2008), EuroPar 2008*, Las Palmas de Gran Canaria, Spain, August, 2008.

12. V. Bui, B. Norris, K. Huck, L. Curfman McInnes, L. Li, O. Hernandez, and B. Chapman, "A Component Infrastructure for Performance and Power Modeling of Parallel Scientific Applications," *Component-Based High Performance Computing (CBHPC 2008)*, October, 2008.

13. K. Huck, O. Hernandez, V. Bui, S. Chandrasekaran, B. Chapman, A. Malony, L. Curfman McInnes, and B. Norris. "Capturing Performance Knowledge for Automated Analysis." *International Conference for High Performance Computing, Networking, Storage, and Analysis (SC08)*, 2008.

14. H. Jagode, J. Dongarra, S. Alam, J. Vetter, W. Spear, A. Malony. "A Holistic Approach for Performance Measurement and Analysis for Petascale Applications." *International Conference on Computational Science (ICCS 2009)*, Baton Rouge, LA, 2009.

15. W. Spear, S. Shende, A. Malony, R. Portillo, P. Teller, D. Cronk, S. Moore, D. Terpstra. "Making Performance Analysis Tuning Part of the Software Development Cycle." *UGC 2009*, San Diego, CA, June 15-18, 2009.

16. L. Li, J. Kenny, M. Wu , K. Huck, A. Gaenko, M. Gordon , C. Janssen, L. Curfman McInnes, H. Mori, H. M. Netzloff, B. Norris, and T. Windus. "Adaptive Application Composition in Quantum Chemistry"," submitted to *5th International Conference on the Quality of Software Architectures (QoSA 2009)*, East Stroudsburg University, Pennsylvania, USA, June 22-26, 2009.

17. S. Shende, A. Malony, and A. Morris, "Improving the Scalability of Performance Evaluation Tools," *International Workshop on Applied Parallel Computing (PARA 2010)*, June, 2010.

18. C.W. Lee, A. Malony, and A. Morris, "TAUmon: Scalable Online Performance Data Analysis in TAU," *Workshop on Productivity and Performance Tools for HPC Application Development (PROPER 2010)*, August 2010.

19. A. Morris, A. Malony, S. Shende, and K. Huck, "Design and implementation of a hybrid parallel performance measurement system," in *International Conference on Parallel Processing (ICPP 2010)*, September, 2010.

20. J. Hammond, S. Krishnamoorthy, S. Shende, N. Romero, A. Malony, "Performance Characterization of Global Address Space Applications: A Case Study with NWChem," to appear in *Concurrency and Computation: Practice and Experience*, 2011.

21. U. Oregon, "TAU Performance System," URL: http://tau.uoregon.edu, 2011.