Finding an Appropriate Profiler for the Weather Research and Forecasting Code

Technical Report FIU-SCIS-2007-09-03
August 2007

Faculty Advisor:

S. Masoud Sadjadi

Graduate Students:

Javier Munoz, Diego Lopez, and David Villegas

Undergraduate REU Students:

Alex Orta, Michael McFail, Xabriel J. Collazo-Mojica, Javier Figueroa

School of Computing and Information Sciences (SCIS)

Florida International University (FIU)

11200 SW 8th St., Miami, Florida 33199, USA

Introduction

This evaluation of profiling tools started because of a need to examine the behavior of the Weather Research and Forecasting Model (WRF). It is currently written to run on a single cluster; our team wished to explore the options for scaling out WRF to a grid environment. This necessitated understanding how WRF works and what its resource usage patterns look like. In order to do this we required a profiler, which led us to creating this document.

We began with a long list of tools, but an in depth investigation for each of them would have been both ineffective and unwarranted. Instead, we broke our evaluation of the field into three passes. In the first pass we discarded programs that did not meet our basic criteria, things like architecture and language support. The second pass was more qualitative. We came up with a list of pros and cons for each tool, and rejected those that did not have features we wanted. Those tools that were still being considered were then given an extensive trial to determine how well they worked for us. We examined things like documentation, ease of installation, whether the tool provided source code correlation and call graphs, etc. It is our hope that this information proves useful to the community, allowing researchers and professionals to learn from our experiences.

First Pass

The First Pass Evaluation Criteria

For the first pass the evaluation criteria were kept simple. Basically, we followed this principle: if we can't use a tool to profile WRF on either the GCB cluster or the UNF cluster, then it is useless to us. Thus, support for Fortran90, support for either x86 or ppc architecture, and support for Linux were absolutely critical. Any tool lacking support for one of these 3 criteria was immediately removed from consideration. For the architecture criteria, a tool without x86 support would pass if and only if it supported ppc, and vice versa.

There is another reason why a tool might be rejected at this pass: an extreme lack of information about the tool. A large number of tools were gathered from tool evaluations done by other organizations, and in particular [1]. However some of these evaluations had been done a number of years ago, and as such many of the tools had stopped being developed or supported long ago. Such tools were removed from consideration.

The Long List of Tools [2]

Vampir

Technically consisting of a visualization component (Vampir) and a tracing component (VampirTrace), Vampir is a commercial tool that was developed at the Center for Applied Mathematics of Research Center Jülich and the Center for High Performance Computing of the Technische Universität Dresden. Originally it was distributed by Pallas GmbH, a German company. However, Pallas was acquired by Intel, and Vampir was integrated into their ITAC tool. A few years later, the cooperation with Intel was ended and development of Vampir continued at the Center for Information Services and High Performance Computing (ZIH) of TU Dresden. [3] It is primarily a tool for analysis of MPI performance, and it is one of the more widely used tools in this regard. Vampir met all the requirements of this pass and was promoted to the next one.

XMPI

XMPI is a GUI for visualizing MPI programs on LAM/MPI systems. Over the course of 18 years, it has been developed by students and faculty at Ohio State University, University of Notre Dame, and Indiana University, where it is still being developed. It is freely available, and meets all the requirements for the First Pass.

MPE/JumpShot

Jumpshot is a Java-based visualization tool for doing postmortem performance analysis for serial, MPI and threaded programs written in C/C++ and FORTRAN. Jumpshot is developed and made freely available from Argonne National Lab. [1] It is a part of the MPE (Multi-Processing Environment) library, which itself is distributed alongside MPICH. This tool meets all the requirements for the First Pass.

XProfiler

Xprofiler is a graphical profiling tool that was formerly provided as part of IBM's Parallel Environment software on its AIX SP platform, but is now part of the AIX operating system software. Xprofiler is actually a GUI for the well-known UNIX gprof profiling utility, which has been enhanced by IBM to work with multi-process MPI programs. [1] Since XProfiler supports only AIX and not Linux, it was rejected at the First Pass.

HPCToolkit

HPCToolkit is a suite of performance-related tools and libraries to assist in application tuning developed by the Advanced Computing Technology Center (ACTC), part of IBM Research in Yorktown Heights, New York.[4] It does not support x86, however it does support Linux on PowerPC, so it meets all the requirements of the First Pass. HPCToolkit is available under a commercial license.

mpiP

mpiP is a lightweight communication profiling library for MPI applications that is developed and distributed by the Lawrence Livermore National Laboratory. [1] It is freely available and meets the requirements of the First Pass.

KOJAK

KOJAK is a performance-analysis tool for parallel applications supporting the programming models MPI, OpenMP, SHMEM, and combinations thereof. It is a is a collaborative project between the Central Institute for Applied Mathematics at Forschungszentrum Jülich and the Innovative Computing Laboratory at the University of Tennessee.[5] It is freely available, and meets the requirements of the First Pass.

Paraver

Paraver is a multi-platform, parallel performance visualization and analysis tool developed by the European Center for Parallelism of Barcelona (CEPBA).[1] It is designed to work with a set of tracing tools also developed by CEPBA, though it can also work with trace files generated by a few other tools, notably TAU. It is free to download and use, but first you must request a license from CEPBA. It meets all the requirements of the First Pass.

gprof

gprof is a basic profiling utility that is available on most UNIX systems. It is developed by the GNU Project, is freely available, and meets the requirements of the First Pass. Additionally, it is a tool which we are already using on the GCB.

TAU

TAU stands for Tuning and Analysis Utilities, and is a joint project between the University of Oregon Performance Research Lab, The LANL Advanced Computing Laboratory, and The Research Centre Julich at ZAM, Germany.[6] It is freely available, and meets the requirements of the First Pass.

ITAC

ITAC stands for Intel Trace Analyzer and Collector. It consists of two components based on VampirTrace and Vampir, which respectively collect data into trace files and present the performance data in a visual manner. It is developed by Intel, and meets the requirements of the First Pass.

PGI-CDK

PGI-CDK (Cluster Development Kit) is a suite of tools developed by the Portland Group which includes all the software for building and programming a turn-key Linux cluster. It is a commercial product which includes all of PGI's compilers, their profiling tool PGPROF, their debugging tool PGDBG, and a set of open source cluster software developed outside of PGI are also included. All are designed to run on Linux Clusters. It meets all the requirements of the First Pass.

OPT

Developed by Allinea, OPT (Optimization and Profiling Tool) is a commercial profiling tool. It meets all the requirements of the First Pass.

AIMS

AIMS (Automated Instrumentation and Monitoring System) is a software toolkit for analyzing C and FORTRAN77 parallel programs supporting a variety of communication libraries.[1] It was developed by NASA, but its website seems to have disappeared, and no other information could be found on the NASA website. It was rejected at the First Pass for lack of information.

HPM Toolkit

HPM Toolkit is another set of performance analysis software developed by IBM. It is now a part of HPCToolkit, and does not need to be considered separately, so it was rejected at the First Pass.

PE BenchMarker

The PE Benchmarker Toolset comprises a suite of tools used to collect and analyze program event trace and hardware performance data, developed by IBM.[1] Since it supports only AIX and not Linux, it was rejected at the First Pass.

PerfSuite

PerfSuite is a collection of tools, utilities, and libraries for software performance analysis where the primary design goals are ease of use, comprehensibility, interoperability, and simplicity.[7] It is developed at the National Center for Supercomputing Applications at the University of Illinois. It is freely available and meets all the requirements of the First Pass.

Open|SpeedShop

Open|SpeedShop is an open source multi platform Linux performance tool which is initially targeted to support performance analysis of applications running on both single node and large scale IA64, IA32, EM64T, and AMD64 platforms. It is a project co-funded by the Department of Energy (DOE), managed by the Tri-laboratories of LLNL, LANL, and Sandia, and developed by SGI. It is freely available and meets all the requirements of the First Pass.

DynaProf

Dynaprof is a parallel performance analysis tool designed to dynamically insert all of its performance measurement instrumentation directly into an application's address space at run-time. It is a software project at the University of Tennessee's Innovative Computing Lab.[1] However the last update to it was in September of 2003, and it only supports Pentium III and below as well as Power3 and below architectures. There is a Pentium4 version with limited features, available at request. Since the clusters we are working with are Xeons and Power5's, this software is unusable and was rejected at the first pass. Furthermore, there is very little current support, documentation, and/or development of DynaProf.

MPX

MPX is not a performance analysis tool per se, but is software for multiplexing hardware performance counters on IBM systems. It is developed by the LLNL, however the latest revision is from 2003. There is very little information we could find about this tool, and since PAPI now does its own multiplexing of hardware performance counters, there appears to be little active development, use, or support of MPX. It was rejected at the First Pass for this reason.

VTune

VTune is a sophisticated and full featured performance analysis tool from Intel for use on 32-bit and 64-bit (Pentium/Itanium) Linux and Windows systems.[1] It is a commercial tool, and meets all the requirements of the First Pass.

DEEP/MPI

DEEP/MPI is a commercial parallel program analysis tool from Veridian/Pacific-Sierra Research. DEEP stands for DEvelopment Environment for Parallel programs.[1] It meets all the requirements of the First Pass.

VProf

The Visual Profiler, VProf, is a basic profiling tool that can be used with serial and parallel MPI programs written in C/C++ or FORTRAN. VProf is developed and distributed by Sandia National Laboratory.[1] Unfortunately, it does not seem to be actively developed or supported anymore. For this reason, it was rejected at the First Pass.

HiProf

Hiprof (Hierarchical instruction profiler) is used to generate profiles of a program's execution time based on its procedure call graph.[1] Unfortunately, it is only available for Tru64 UNIX, and so is useless to us. It was rejected at the First Pass for this reason.

IPM

IPM is a lightweight MPI communications profiler developed at NERSC.[1] It is freely available and meets all the requirements of the First Pass.

PapiEx

PapiEx is a performance analysis tool designed to transparently and passively measure the hardware performance counters of an application using PAPI. It was written by Philip J. Mucci of the Innovative Computing Laboratory and SiCortex Inc. Major contributions and enhancements were made by Tushar Mohan, also of SiCortex Inc.[8] It is freely available, and meets all the requirements of the First Pass.

lperfex

lperfex is a performance analysis tool for the IA32 architecture, and is developed at the Ohio Supercomputer Center. It is based on perfex, which reports the hardware counts of selected events on SGI R10000 platforms. Unfortunately, not much information could be found about the tool, and the newest update is from January of 2002. For this reason, it was rejected at the First Pass.

Pixie

Pixie is another HP/Compaq profiler tool, similar to prof, gprof and hiprof.[1] Unfortunately, like HiProf, it is only available for Tru64 UNIX. It was rejected at the First Pass for this reason.

SCALEA

SCALEA is a performance instrumentation, measurement, analysis, and visualization tool for parallel FORTRAN programs.[1] It is developed by a team of 18 researchers from the Distributed and Parallel Systems Group led by Thomas Fahringer at the Institute of Computer Science, University of Innsbruck, Austria. Unfortunately, it is only available for the Sun Sparc, and so is rejected at the First Pass.

TProf

The tprof command is an AIX profiling utility developed by IBM that reports CPU usage for individual programs and/or the system as a whole.[1] Unfortunately, it has no support for Linux, and so was rejected at the First Pass.

Performeter

Performeter is (or once was) the real-time performance monitor distributed with PAPI.[1] Unfortunately, very little information could be found about it, so it was rejected at the First Pass.

RootCause

RootCause, based on Aprobe, is a sophisticated tracing tool that, thru a GUI, a user can selectively choose the data to be collected or navigated.[1] It is a commercial tool developed by OC Systems. However, it was rejected at the First Pass for lack of Fortran Support.

SIGMA

SiGMA (Simulation Guided Memory Analyzer) is a toolkit for analyzing bottlenecks and inefficiencies due to the memory hierarchy.[1] It is developed by IBM, and is now included as part of the HPCToolkit. Standalone, it is only available for the AIX OS. For this reason, it was rejected at the First Pass.

OptiPath

OptiPath was a planned commercial MPI Acceleration Tool from PathScale. However, there seems to be no current mention of it on PathScale's website, and seems to have been abandoned. For this reason, it was rejected at the First Pass.

ParaDyn

The Paradyn parallel performance analysis tool comes from of an ongoing research and software development project originating at the University of Wisconsin.[1] It is freely available, and meets all the requirements of the First Pass.

ParaGraph

ParaGraph is a visualization tool developed by Michael T. Heath of the University of Illinois at Urbana-Champaign and Jennifer E. Finger of the University of Tennessee at Knoxville. It depends on the MPICL instrumentation library, though it is not clear whether MPICL and ParaGraph support Power5 or IA32 architectures. It will be tentatively promoted through the First Pass.

SvPablo

SvPablo is a graphical performance analysis tools that originates from the Pablo Research Group at the University of Illinois, Urbana-Champaigne, with funding provided by NASA, DOE and DARPA.[1] It is freely available, and meets all the requirements of the First Pass.

SeeWithin/Pro

Verari System's SeeWithin/Pro scalable performance analysis tool works with MPI applications written in C and FORTRAN on Linux and Windows NT/2000/XP platforms including Intel and AMD clusters.[1] There do not seem to be any current references to the software on Verari's website, and little other information about it could be obtained. For this reason, it was rejected at the First Pass.

Oprofile

OProfile is a system-wide profiler for Linux systems, capable of profiling all running code at low overhead. OProfile is released under the GNU GPL.[9] It is developed by John Levon and Philippe Elie, along with a team of contributors listed on their website. It meets all the requirements of the First Pass.

MicroGrid

The MicroGrid is a tool that provides online simulation of large-scale (20,000 router, thousands of resources) network and Grid resources.[10] It is developed as part of the GRADS project, and is freely available. Unfortunately, it seems as though Fortran is not supported, and was rejected at the First Pass.

TotalView

TotalView Debugger is a commercial debugging tool designed to work in High Performance Computing environments. It was developed by Etnus LLC, which is now called TotalView Technologies. It is highly regarded as a debugging tool, and also has some features that aid in performance analysis. It meets all the requirements of the First Pass.

Second Pass

For the second pass we wanted a more thorough look at what features each tool offered to us. This information, as well as information from profiling tool reviews done by other organizations, would be used to select from this list a much smaller number of tools which we would like to study further in a third pass. There was no specific criteria for moving a tool on to the next pass: rather a list of pros and cons was made for each tool based on the information we gathered. Of particular help at this stage were [1] and [11], as well as the websites of each individual tool. We originally had intended to have a tool selected as soon as the second pass was done, but with the number of tools that had made it this far, it was decided that we should pick a much smaller number to look into comprehensively. We wanted to keep the number of tools that went on to the third pass to a minimum, and after much discussion at meetings, five tools stood out as the most interesting. In the next section we'll discuss the tools involved in the second pass, starting with the five that were eventually chosen to move on to the third pass.

The Short List of Tools [10]

Paraver

Paraver supports MPI, OpenMP, and hybrid programming environments on AIX, Tru64, Linux, and Irix platforms, in C/C++ and FORTRAN. Some of Paraver's key features include both quantitative and qualitative displays for message passing activity, hardware performance counters, and operating system activity. [1] Another thing that makes Paraver interesting to us is that we have some contacts in Barcelona who are willing to help us out in using Paraver. This, along with the features and free availability of Paraver led us to include it in the third pass.

TAU

TAU is a profiling, tracing and visualization toolkit for parallel codes. It supports MPI, threads (OpenMP and pthreads), and hybrid (MPI+threads) programs written in C, C++, FORTRAN, Python and Java. TAU is very portable and has been ported to a number of platforms including SGI IRIX, Intel x86 Linux, Sun Solaris, IBM AIX, HP HP-UX, HP Alpha Tru64, NEX SX, Cray X1, T3E, SV-1, Hitachi SR8000, Apple OS X and Microsoft Windows. It is probably one of the most full featured HPC performance analysis tools. TAU combines the technologies of several other performance analysis tools including Dyninst dynamic instrumentation, PAPI hardware counters, Opari OpenMP instrumentation, and Vampir and Paraver trace visualization.[1] Due to the wide range of features and free availability of the tool, we decided to include it in the third pass.

PGI-CDK

There are several reasons why PGI-CDK is a highly interesting tool for our project. First of all, it supports MPI, OpenMP, and MPI+OpenMP. Support for profiling a hybrid MPI+OpenMP approach affords us great flexibility in trying to adapt WRF to a grid computing environment. Should we find out that OpenMP may be of great use in such an effort, we can use it in conjunction with MPI and the Portland Group's profiling tool PGPROF would be able give us performance information in such an environment. Another feature is that it can measure scalability between multiple execution runs with varying number of processes/threads. This is similar to the benchmarking that we've been doing. Although we already have scripts and such that do benchmarking for us, it is good to know that this functionality is built into PGPROF, in the case that we need it for something. PGPROF also supports both sample-based and instrumentation based-profiling. This is another great feature for flexibility. What this means is that if we want the detailed information that an instrumentation-based approach provides, we can go ahead and do that, but that if we don't need such detailed information and would prefer to avoid the extra work that instrumentation entails, then we can go instead with a sample-based approach, similar to gprof's. And perhaps one of the greatest reasons that we decided to include PGI-CDK in the third pass is that it includes PGI's compilers. These compilers are known to work with WRF as well as TAU, which itself is one of the leading tools in this evaluation. For all these reasons, we felt that PGI-CDK was worthy of further investigation in the third pass.

HPCToolkit

HPC toolkit is a good performance analysis tool for one running on a Power platform. It provides MPI and OpenMP support for Linux and AIX on Power, as well as Blue Gene. In addition to the command line tools it supplies there is also a graphical front end for viewing the output trace files. One major downside is its inability to support the x86 architecture, which is very commonly used. It was chosen for evaluation in the third pass mainly due to our interest in using the PowerPC-based cluster at UNF.

SvPablo

SvPablo seems like a really solid product as it's documentation is up to date and well written. Although it doesn't support OpenMP, it's main purpose is in the use of big, long-running MPI programs. One of the big features of SvPablo is it code-graph correlation. You can also choose to automatically or manually instrument the code. A combination of the features it offered and the solid documentation it had caused us to include it in the third pass.

KOJAK

KOJAK is a set of utilities to identify bottlenecks in programs based on OpenMP and MPI. It includes utilities for all the profiling stages from instrumentation to data presentation. Unfortunately, as stated by one of it maintainers, KOJAK is not particularly good for profiling long-running programs. WRF is generally run for long periods of time, due to both the amount of calculations it performs and the amount of data involved in a run. For this reason, KOJAK was not really considered as a candidate for the third pass.

Vampir/ITAC

Vampir and ITAC provide very similar functionality at very similar costs, since ITAC is based largely on an older version of Vampir. Vampir is one of the more highly regarded MPI performance analysis tools. It works by storing detailed information about a running program into tracefiles, which are later read by a visualization component in order to aid in performance analysis. The resulting call trees are very detailed and informative. Vampir can do binary instrumentation, which provides the instrumentation necessary to collect the trace data without requiring a programmer to manually alter any code. However, Vampir is a commercial tool, does not support OpenMP, and is said to have a steep learning curve. The five tools in the third pass provide similar functionality at less cost (except for PGI-CDK) and with less of a learning curve, so it was left out of the third pass.

XMPI

XMPI is an X/Motif based graphical user interface for running, debugging and visualizing MPI programs. Extensive MPI information is extracted from a running application on demand, or from a cumulative log of communication. Both sources are tightly integrated with an application overview window and any number of single process focus windows. [13] It is specifically designed to be used with the LAM/MPI implementation of the MPI standard. However, XMPI is generally considered more of an MPI teaching aid than a useful MPI profiling tool. Also, its basic interface and simple features do not lend themselves well to large scale parallel runs.[1] For these reasons, XMPI was removed from consideration for the third pass.

mpiP

mpiP is a lightweight communication profiling library for MPI applications that is developed and distributed by LLNL. mpiP’s lightness is attributed to the fact that it only collects statistical information about MPI functions, and therefore generates considerably less overhead and much less data than tracing tools. Furthermore, all the information captured by mpiP is task-local. It only uses communication during report generation, typically at the end of the execution, to merge results from all of the tasks into one output file. [1] mpiP scales well, but does not support OpenMP and has no call graphs. mpiP is a tool which we are currently using, but we may eventually need more detailed information than mpiP can provide. In either case, since it's already in use, there's no real need to evaluate it further, and so it was left out of the third pass.

gprof

gprof is a standard sample-based profiling command that comes with UNIX. It only requires a simple recompile to be used, and so does not demand any changes in the code. However, gprof really only provides somewhat basic timing information. We already use it, and it is a great tool, but we may need something more detailed for future work with WRF, especially in regard to MPI calls. Since it is already in use, there is no real need to evaluate it more thoroughly, and so it was left out of the third pass.

OPT

Allinea OPT is a commercial program for performance analysis on large, parallel systems and software. It has a number of features that are interesting to us, including support for MPI and OpenMP profiling. It is cross-platform software, available for both the x86 and ppc architectures, among others. OPT can compare multiple runs to assess code scalability [15], somewhat like the benchmarking that we have been doing. Another nice feature is that there is no need to instrument code, which reduces the time and effort that must be put into using the tool. Allinea also says that OPT is "Grid-enabled", though it is not exactly clear what exactly they mean by that. The main disadvantage of this tool is that it is commercial software. The price is not listed anywhere on their website, it seems to be something someone must inquire about through email. By the time we were deciding whether or not to include OPT in the third pass, we already had five tools put through into the third pass, two of which were commercial tools. Five was already more than we really wanted to have in the third pass, so in spite of its interesting features, we left it out of the third pass. If one or more of the final five tools doesn't work out (and especially a commercial one), this may be an interesting solution to look into.

MPE/Jumpshot

MPE/Jumpshot's main feature is logging and tracing of MPI data, which makes it a nice tool for analysis of MPI performance. It is based on java, and so it is cross platform, all you need is a java2 runtime in order to use it. Another good thing about it is that it already comes with MPICH, so if you have MPICH installed there should be nothing special required to use it. Its logging/tracing utilities scale well across large runs, and it comes with a system for auto-instrumentation. Some downsides are a lack of OpenMP support, and the fact that it is mostly an MPI profiler. It was left out of the third pass because tools such as TAU and Paraver provide the same features, and more.

Paradyn

First of all, Paradyn's documentation seemed to be pretty solid. Quality documentation can make a good amount of difference in productivity. Paradyn supports MPI profiling and scales quite well across large runs. And one of the more interesting features that is has is dynamic binary instrumentation. This give the advantages inherent in using instrumentation without requiring any modifications to the code. It has a few downsides though, one being a lack of support for OpenMP profiling. Also, it is said to be somewhat buggy and have a steep learning curve. The final five tools seemed to provide a larger number of features with fewer or less serious downsides, so Paradyn was left out of the third pass.

VTune

VTune is a commercial tool from Intel that supports profiling of both MPI and OpenMP. It scales quite well, which is important for running large software on clusters as we are doing. It is also programming language and compiler independent, meaning that it can work with any configuration of the two. Another nice thing about VTune is that it does not require any modification to the code. Some downsides are the price and the fact that VTune is limited to Intel-based platforms. At this point, PGI-CDK was the most interesting of the commercial performance analysis tools, and so VTune was left out of the third pass.

TotalView

TotalView is a highly scalable debugging tool that supports multiple platforms, compilers, and programming languages. It supports both MPI and OpenMP, and has a number of desirable features that are outlined at [14]. At the bottom of that page, there is a summary of the performance analysis features that TotalView contains. It seems impressive, however, it also seems to be mainly a debugging tool. Since our (current, anyway) main interest is profiling/performance analysis, and not debugging, I cannot see TotalView as being worth the cost. For this reason it was left out of the third pass. However, should we ever need a debugging solution for WRF, or some other cluster application, then TotalView is highly recommended for consideration.

<u>Open|SpeedShop</u>

Open|SpeedShop is an open source multi platform Linux performance tool which is initially targeted to support performance analysis of applications running on both single node and large scale IA64, IA32, EM64T, and AMD64 platforms. It is explicitly designed with usability in mind and targets both application and computer scientists. Open|SpeedShop's base functionality includes metrics like exclusive and inclusive user time, MPI call tracing, and CPU hardware performance counter experiments. In addition, Open|SpeedShop is designed to be modular and easily extendable. It supports several levels of plugins which allow users to add their own performance experiments.[16] Open|SpeedShop was added later on in the profiler evaluation process, and was being put through the second pass as the third pass tools were being decided on. It was left out of the third pass because its second pass evaluation had not yet been conducted.

<u>Oprofile</u>

Oprofile is one of the more interesting tools that I came across. Its low overhead makes it an extremely scalable piece of software, which is good for large parallel software running for long periods of time across many nodes. It supports gprof-style call-graph profiling data, hardware performance counter profiling, and system wide profiling. With system wide profiling, all code running on the system is profiled, enabling analysis of system performance. [9] One of Oprofile's most interesting features though is its unobtrusiveness. It has no need for a kernel patch, special recompilations, wrapper libraries or anything of the like. Some downsides are a lack of MPI or OpenMP profiling, since the software focuses mostly on profiling of hardware counter data. MPI support is (as we currently see it) particularly important for our project, so Oprofile was left out of the third pass in favor of other tools. However, if there ever is a need for unobtrusive profiling of hardware counters, this tool is highly recommended. Addtionally, it seems that PGI-CDK can be used in conjunction with Oprofile, so if PGI-CDK is selected Oprofile may be good to look into again.

<u>PapiEx</u>

PapiEx uses Monitor to to effortlessly intercept process/thread creation/destruction. It measures the entire run of an application. By default this includes all subprocesses. It is not a tool for selective instrumentation, for that you should use TAU or possibly DynaProf. PapiEx's goal is to be a Linux substitute for the perfex command found in SGI's Speedshop. PapiEx is fairly simple to build, install and use. For a more full featured tool, please consider the excellent PerfSuite distribution from Rick Kufrin at NCSA. [8] Some good things about PapiEx are that it supports MPI profiling, it has no external dependencies other than Monitor and PAPI, and that is fairly simple to build, install, and use. However, we may want the additional features that tools such as TAU provide, possibly including the noted selective instrumentation feature. For this reason, we found other tools more compelling, and left PapiEx out of the third pass.

<u>IPM</u>

IPM is a portable profiling infrastructure for parallel codes. It provides a low-overhead performance summary of the computation and communication in a parallel program. The amount of detailed reported is selectable at runtime via environment variables or through a MPI_Pcontrol interface. IPM has extremely low overhead, is scalable and easy to use requiring no source code modification. IPM brings together several types of information important to developers and users of parallel HPC codes, such as: MPI data, PAPI performance events, memory information, and switch information such as communication volume and packet loss. [17] IPM was added later on in the profiler evaluation process, and was being put through the second pass as the third pass tools were being decided on. It was left out of the third pass because its second pass evaluation had not yet been conducted.

DEEP/MPI

DEEP/MPI provides an integrated GUI for performance analysis of shared memory (threads), distributed memory MPI, and hybrid (shared + MPI) parallel programs. To use DEEP/MPI, one must first compile the MPI program with the DEEP profiling driver mpiprof. This step collects compile-time information and also instruments the code. After executing the program in the usual manner, the user can view performance information using the GUI. The DEEP/MPI GUI includes a call tree viewer for program structure browsing and tools for examining profiling data at various levels. It displays whole program data such as the wallclock time used by procedures. After identifying procedures of interest, the user can bring up additional information for those procedures, such as loop performance tables. The DEEP Performance Advisor suggests which procedures or loops the user should examine first. Clicking on a loop in a loop performance table or on an MPI call site takes the user to the relevant source code. CPU balance and message balance displays show the distribution of work and number of messages, respectively, among the processes. DEEP provides PAPI hardware counter support and can do profiling based on any of the PAPI metrics. [1] It's only obvious downsides are the price and the lack of OpenMP support. DEEP/MPI was added later on in the profiler evaluation process, and was being put through the second pass as the third pass tools were being decided on. It was left out of the third pass because its second pass evaluation had not yet been conducted.

PerfSuite

PerfSuite is a collection of tools, utilities, and libraries for software performance analysis where the primary design goals are ease of use, comprehensibility, interoperability, and simplicity. This software can provide a good "entry point" for more detailed performance analysis and can help point the way towards selecting other tools and/or techniques using more specialized software if necessary (for example, tools/libraries from academic research groups or third-party commercial software). [7] Thus, the PerfSuite tools seem to provide a great amount of breadth in its performance analysis capabilities, but lacks the depth that more specialized tools might provide. The only other obvious downside is a lack of OpenMP support. PerfSuite was added later on in the profiler evaluation process, and was being put through the second pass as the third pass tools were being decided on. It was left out of the third pass because its second pass evaluation had not yet been conducted.

Notes

As an afterthought, it seems as though quite a few tools were left out of the third pass simply in order to keep the number of tools in the third pass down to a more manageable level. If we had more time to work with, perhaps we would have put a few more through to the third pass. Of course, this may have necessitated a fourth and perhaps even a fifth pass, which is something we did not see as possible to finish by the end of the summer.

Third Pass

The Third Pass Evaluation Criteria

For the purposes of this in depth evaluation we came up with a set of Evaluation Criteria that will allow us to focus on the features that matter the most to our project and compare how each tool fares for each criteria. Here's the current list of criteria, followed by a small explanation of each.

Ease of Use
- Documentation
- Tutorials/Examples
- User Friendliness
- Learning Curve
- Community/Tech Support

OpenMP
MPI
Price
Call Graphs
Source Code Correlation

Ease of use was suggested early on as a criterion, however as time went by we realized that it was too broad to be of much use as a single criterion. Therefore, we made it a category comprised of several criteria beneath it.

- *Documentation*, which is one such criterion, is where we describe the quality of the documentation that comes with a profiling tool. Good documentation is well organized, easy to follow, and describes how to use the tool it documents.

- *Tutorials/Examples* is where we describe the availability/helpfulness of any tutorials and/or examples for a profiling tool. Good tutorials and/or examples can really cut down on the amount of time spent trying to figure a tool out.
- *User Friendliness* describes how easily one can make some initial use of a profiling tool without doing too much prior research/reading. A tool that allows one to get straight to work without requiring a lot of reading beforehand is preferable.
- *Learning Curve* describes how much time and effort it takes to learn how to use a profiling tool to its fullest ability. If a tool takes a long time or a large amount of effort to learn how to make full use of its features, we may prefer to go with another one.
- *Community/Tech Support* describes the availability, responsiveness, and quality of the tool's technical support. The tech support can come in the form of an official contact (emails of the development team, etc.) or possibly in the form of an extremely knowledgeable community (forums, mailing lists, etc.) Technical support and development teams are often swamped with work, so sometimes communities are much better equipped to answer the large number of inquiries about a certain piece of software. However, it is also desirable to have an official response, as they are sometimes more definitive.

The *OpenMP*, *MPI*, and *Call Graphs* criteria simply describe the level of support that the tool being evaluated provides for each of these respective features.

The *Price* criterion simply states the cost of using the tool being evaluated on a cluster such as the GCB.

The *Source Code Correlation* criterion describes how conductive the profiling tool being evaluated is to looking at the information it's giving you and figuring out what part of the source code it pertains to

About the Final Five Tools

In this section we'll be talking about TAU, SvPablo, Paraver, HPCToolkit, and PGI-CDK.

SvPablo

Ease of Use	
Documentation	SvPablo's Documentation is quite good. They have a long PDF covering everything from the GUI to code instrumentation. It is up to date (March 2006). There is also a 26-page paper on SvPablo.
Tutorials/Examples	The PDF has examples from the basic PI example to some more complicated ones. There is a full-length tutorial available online.
User Friendliness	Installation is not straight forward, we encountered a variety of problems. We also experienced UI bugs, making it unsuitable for our purposes.
Learning Curve	
Technical Support	They have not responded yet to an email inquiry from a few weeks ago.
OpenMP	As of now SvPablo only has experimental support for OpenMP
MPI	MPI support is SvPablo's main feature.
Fortran	It supports g95, F77 and F90
Price	Free
Call Graphs	Supported, and they seem easier to follow than the call graphs in Paraver.
Source Code Correlation	Inside the GUI you can click on a line of code and it will tell you which trace is that line and vice versa.

Although SvPablo had some very good and up-to-date documentation, it does not seem like a tool we can use for profiling WRF. The biggest reason for this is that the program itself is rather buggy. Trying to do much of anything would often result in crashing the program. Additionally, the technical support never responded to any of our inquiries. This combination rendered SvPablo rather useless for performance analysis purposes.

For the rest of the final five tools, we could not finish an evaluation as we had hoped, due to various technical difficulties. However, here we will document what problems we ran into for each tool, and what we did to try and solve these problems.

PGI-CDK

The primary system we were attempting to get PGI-CDK to work on was the GCB Cluster, since it does not list PowerPC as one of the architectures that it supports. To download PGI-CDK, you must register with PGI's website. However, this is simple, free, and only requires a valid email address. You also must ask them through email for an evaluation license that will unlock the features of PGI-CDK for a limited time, you can use their tech support email (trs at pgroup dot com) for this purpose. Once you register, you can log into their website and then go to the downloads section to download PGI-CDK. While waiting for my cdk evaluation license to get approved, I was also told in an email that:

"you can get very close by first installing the Workstation product on the master node, and then downloading and installing the MPICH Kit. If you install MPICH as root, the script will attempt to create machines.LINUX for you.

1. untar the 32 or 64-bit download into some area like /tmp/pgi
2. untar the MPICH kit download into /tmp/pgi as well.
3. run /tmp/pgi/install to install the compilers.
4. run /tmp/pgi/installmpich"

We got approved for a PGI-CDK evaluation fairly quickly anyway (a few hours), so we downloaded it and ran the install script. We chose /share/apps/pgi as the directory in which we would install it. PGI-CDK is supposed to come with compilers, however, David (our system admin) could find no evidence that the PGI compilers had been installed when we tried to use them. We asked the tech support about this, and they said:

"before you run the installcdk script, download the current 7.0-6 workstation complete into the same directory (pgilinux86-64-706.tar.gz). When you run installcdk, it integrates the current compiler release(pgilinux86-64-706.tar.gz) with the CDK components (pgicdk-linux86-64-70.tar.gz) , to create an up to date CDK installation. "

With "installcdk" being the install script that comes with PGI-CDK. We tried this, however, it did not work as they said it would. So then we tried uninstalling everything, then installing PGI Workstation Complete, then installing PGI-CDK, and then putting the license file into the install directory as license.dat. This appeared to work at first, since it would compile Fortran code. However, then we tried to compile an MPI program with the PGI compilers to run on the GCB. It compiled without any problems, but when we tried to run it with multiple processors, we were getting ssh errors. We asked PGI tech support about this, and they said that they were putting up a new version of their mpich kit that should fix ssh problems. When they said they had this new version out, we downloaded and installed it, however by that time our original evaluation license had expired. So we asked for and got a new one, however when we tried it again, we were getting the same ssh errors. This is where we are currently at with PGI-CDK. Their tech support has been very responsive and friendly, which is one nice thing about PGI-CDK. However, we are still stuck on this ssh error.

TAU (Tunning and Analysis Utilities)

TAU (Tuning and Analysis Utilities) is a portable profiling and tracing toolkit for performance analysis of parallel programs written in Fortran, C, C++, Java, Python. TAU is capable of gathering performance information through instrumentation of functions, methods, basic blocks, and statements. All C++ language features are supported including templates and namespaces. The API also provides a selection of profiling groups for organizing and controlling instrumentation. The instrumentation can be inserted in the source code using an automatic instrumentor tool based on the Program Database Toolkit (PDT), dynamically using DyninstAPI, at runtime in the Java virtual machine, or manually using the instrumentation API.

TAUs profile visualization tool, paraprof, provides graphical displays of all the performance analysis results, in aggregate and single node/context/thread forms. The user can quickly identify sources of performance bottlenecks in the application using the graphical interface. In addition, TAU can generate event traces that can be displayed with the Vampir, Paraver or JumpShot trace visualization tools.[6]

In our case we downloaded TAU and installed it in GCB /share/apps/tau. We installed PDT in /share/apps/pdt. We first ran into an issue with the c++ and c compilers. We will show our workaround while explaining the installation of the programs. First, TAU was not using the right version of the compilers so we had to modify some of the configure and make files, for both the pdt and tau installations. To have automatic instrumentation TAU uses PDT tool kit. To install PDT we ran the ./configure with flags -intel -prefix=/share/apps/pdt to specify the installation directory and the system's architecture. Before running the configure script we modified it to use g++4 and gcc4. The following are the changes on the file:

```
[wrf@gcb tmp]$ diff /share/apps/usr1/pdtoolkit-3.11.1/configure pdtoolkit-3.11.1/configure
183,184c183
<     #path_gxx=$dir/g++
<     path_gxx=$dir/g++4
---
>&nbsp;    path_gxx=$dir/g++
```

We did the same thing with the PDT tool kit Makefile:

```
[wrf@gcb pdtoolkit-3.11.1]$ diff /share/apps/usr1/pdtoolkit-3.11.1/ductape/Makefile ductape/Makefile
102c102
< PDT_GXX=g++4 #EOC##GXX#
---
> PDT_GXX=g++ #EOC##GXX#
```

After having successfully installed PDT, we proceeded to install TAU. After decompressing the file we run the configure script with the following flags: -gnu -fortran=intel -mpiinc=/share/apps/mpich-1.2.7p1/include/ -mpilib=/share/apps/mpich-1.2.7p1/lib/ -pdt=/share/apps/pdt -prefix=/share/apps/tau. Since we had the same problem regarding the c++ and c compiler, we had to modify the configure file as well as the makefile as follows:

```
wrf@gcb tau-2.16.5]$ diff /share/apps/usr1/tau-2.16.5/configure configure
290c290
<     if [ $myarg = cc -o $myarg = gcc -o $myarg = gcc4 -o $myarg = KCC -o $myarg = pgcc -o $myarg = guidec -o $myarg =
xlc -o $myarg = ecc -o $myarg = icc -o $myarg = powerpc64-linux-gcc -o $myarg = pathcc -o $myarg = fcc -o $myarg = orcc -o
$myarg = qk-pgcc -o $myarg = scgcc -o $myarg = scpathcc -o $myarg = mips64el-gentoo-linux-gnu-gcc ]
---
>     if [ $myarg = cc -o $myarg = gcc -o $myarg = KCC -o $myarg = pgcc -o $myarg = guidec -o $myarg = xlc -o $myarg = ecc
-o $myarg = icc -o $myarg = powerpc64-linux-gcc -o $myarg = pathcc -o $myarg = fcc -o $myarg = orcc -o $myarg = qk-pgcc -o
$myarg = scgcc -o $myarg = scpathcc -o $myarg = mips64el-gentoo-linux-gnu-gcc ]
308c308
<     if [ $myarg = CC -o $myarg = KCC -o $myarg = g++4 -o $myarg = g++ -o $myarg = cxx  -o $myarg = NCC -o $myarg =
pgCC  -o $myarg = egcs -o $myarg = FCC -o $myarg = guidec++ -o $myarg = aCC -o $myarg = c++ -o $myarg = ecpc -o
$myarg = icpc -o $myarg = powerpc64-linux-g++ -o $myarg = pathCC -o $myarg = orCC -o $myarg = qk-pgCC -o $myarg =
scg++ -o $myarg = scpathCC -o $myarg = mips64el-gentoo-linux-gnu-g++ ]
---
>     if [ $myarg = CC -o $myarg = KCC -o $myarg = g++ -o $myarg = cxx  -o $myarg = NCC -o $myarg = pgCC  -o $myarg =
egcs -o $myarg = FCC -o $myarg = guidec++ -o $myarg = aCC -o $myarg = c++ -o $myarg = ecpc


-o $myarg = icpc -o $myarg = powerpc64-linux-g++ -o $myarg = pathCC -o $myarg = orCC -o $myarg = qk-pgCC -o $myarg =
scg++ -o $myarg = scpathCC -o $myarg = mips64el-gentoo-linux-gnu-g++ ]
345c345
<     if [ $myarg = gnu -o $myarg = sgi -o $myarg = ibm -o $myarg = ibm64 -o $myarg = hp -o $myarg = cray -o $myarg = pgi -
o $myarg = absoft -o $myarg = fujitsu -o $myarg = sun -o $myarg = compaq -o $myarg = kai -o $myarg = hitachi -o $myarg =
intel -o $myarg = nec -o $myarg = absoft -o $myarg = lahey -o $myarg = nagware -o $myarg = pathscale -o $myarg = gfortran -
o $myarg = open64 -o $myarg = g77 ]
---
>     if [ $myarg = gnu -o $myarg = sgi -o $myarg = ibm -o $myarg = ibm64 -o $myarg = hp -o $myarg = cray -o $myarg = pgi -
o $myarg = absoft -o $myarg = fujitsu -o $myarg = sun -o $myarg = compaq -o $myarg = kai -o $myarg = hitachi -o $myarg =
intel -o $myarg = nec -o $myarg = absoft -o $myarg = lahey -o $myarg = nagware -o $myarg = pathscale -o $myarg = gfortran -
o $myarg = open64 ]
356,357c356,357
<       c_compiler=gcc4
<       cxx_compiler=g++4
```

```
---
>       c_compiler=gcc
>       cxx_compiler=g++
1862,1867d1861
<    *g++4)
<       echo "Default C++ compiler will be " \
<          `gcc4 -v 2>&1 | tail -1 | sed 's/gcc/g++/g'`
<       gnu=yes
<       # No fixmakeargs needed because it is the default
<       ;;
```

And the TAU Makefile:

```
[wrf@gcb tau-2.16.5]$ diff /share/apps/usr1/tau-2.16.5/Makefile Makefile
18,19c18,19
< CONFIG_CC=gcc4
< CONFIG_CXX=g++4
---
> CONFIG_CC=gcc
> CONFIG_CXX=g++
22c22
< TAUROOT=/share/apps/usr1/tau-2.16.5
---
> TAUROOT=/tmp/tau-2.16.5
```

We also made the following changes in this other Makefile:

```
wrf@gcb tau-2.16.5]$ diff /share/apps/usr1/tau-2.16.5/src/Profile/Makefile src/Profile/Makefile
28c28
< TAUROOT=/share/apps/usr1/tau-2.16.5
---
> TAUROOT=/tmp/tau-2.16.5
48c48
< DEFINES = -DTAU_LIBRARY_SOURCE -DTAU_DOT_H_LESS_HEADERS
---
> DEFINES = -DTAU_LIBRARY_SOURCE
143c143
< #GNU#AR_SHFLAGS              = -shared       #ENDIF#
---
> AR_SHFLAGS           = -shared       #ENDIF##GNU#
221c221
<               $(TAU_CXX) -fPIC -shared $(TAU_ARFLAGS) $(TAU_DISABLE_SHARED) TauDisable.o
---
>               $(TAU_CXX) $(TAU_ARFLAGS) $(TAU_DISABLE_SHARED) TauDisable.o
```

Notice that almost everything that was using the CXX and CC variable had to be replaced manually. We also believe that using the 'export CC=gcc4' and 'export CXX=g++4' commands would probably have done this automatically. There are different approaches to the compiler issue, we decided to go manually to ensure the functionality of TAU. Finally we tested TAU on a FORTRAN Hello World program which used MPICH. The program ran perfectly and TAU produced the trace files to be later used in paraprof. The next step is to test TAU with WRF which we are currently working on.

Paraver

Although there were initial compatibility issues with the x86 platform and Paraver, these were promptly fixed. We were able to run a test case with Fortran code, but runs with anything more than 1 node would never end. We asked for help from Paraver's help-desk and they were responsive. As of today we still haven't been able to run anything related to MPI, but communications are going on to see if can finally solve this issue and evalaute Paraver as we should.

HPCToolkit

The HPC toolkit was provided to us as an RPM file. Our testbed system was Linux running on a Power platform. The toolkit had some common, easy to satisfy dependencies, but it also required the use of a special high performance library. A search for the library proved fruitless, but inside one of the documentation files for the RPM was a link to download this library. We also had problems with MPI on our system, an issue separate from the HPC tookit. However, this issue led to the evaluation of TAU being finished with a positive result before the HPC toolkit was successfully evaluated.

Key Terms & Acronyms

API - Application Programming Interface
CEPBA - European Center for Parallelism of Barcelona
GCB - An 8 node, 16 cpu Intel-based rocks cluster at FIU that is available for our use.
GRADS - The GRid Applications Development Software project, led by Rice University
HPC - High Performance Computing
LLNL - Lawrence Livermore National Laboratory
MPI - Message Passing Interface, a library specification for message-passing, proposed as a standard by a broadly based committee of vendors, implementors, and users
NERSC - National Energy Research Scientific Computing Center
OpenMP - an API that supports multi-platform shared-memory parallel programming in C/C++ and Fortran on all architectures, including Unix platforms and Windows NT platforms
PAPI - Performance Application Programming Interface, a portable and efficient API to access the hardware performance counters found on most modern microprocessors
UNF - the University of North Florida, which has a 32 node, 128 cpu PPC-based cluster which we hope to use
WRF - The Weather Research and Forecasting Model, the Fortran-based mesocale numerical weather prediction system that we are working with

References

[1] http://www.llnl.gov/computing/tutorials/performance_tools/HighPerformanceToolsTechnologiesLC.pdf - An evaluation of HPC tools, done by the LLNL
[2] http://docs.google.com/Doc?docid=dctc7vch_15thp5fc&hl=en_GB - The First Pass Table
[3] http://www.vampir-ng.de/Historie.html - Some information on the history of Vampir
[4] https://domino.research.ibm.com/comm/research_projects.nsf/pages/actc.index.html - The HPCT Website
[5] http://icl.cs.utk.edu/kojak/index.html - KOJAK Website
[6] http://www.cs.uoregon.edu/research/tau/home.php - TAU Website
[7] http://perfsuite.ncsa.uiuc.edu/ - PerfSuite Website
[8] http://icl.cs.utk.edu/~mucci/papiex/papiex.html - PapiEx Website
[9] http://oprofile.sourceforge.net/about/ - Oprofile Website
[10] http://docs.google.com/Doc?docid=dfpq2cn6_1grh5w5&hl=en_GB - The Second Pass Table
[11] http://www.hcs.ufl.edu/upc/archive/toolevals/ - A set of tool evaluations done by the HCS at the

University of Florida
[12] http://www.pgroup.com/products/pgprof.htm - Website of PGPROF, the profiling tool that comes with PGI-CDK
[13] http://www.lam-mpi.org/software/xmpi/ - XMPI Website
[14] http://www.totalviewtech.com/Debugger/tvfeatures.htm - The features section of the TotalView Website
[15] http://www.allinea.com/OPT.pdf - A pdf describing the features of Allinea OPT
[16] http://oss.sgi.com/openspeedshop/ - Open|SpeedShop Website
[17] http://ipm-hpc.sourceforge.net/ - IPM Website