Designing a better visualization system for supercomputing users

M.S MABAKANE (1,2), M KUTTEL (2)

(1) Centre for High Performance Computing, CSIR Campus, 15 Lower Hope Street, Rosebank, Cape Town, South Africa

(2) University of Cape Town, 18 University Avenue, Upper Campus, Rondebosch, Cape Town, South Africa

CHPC CENTRE FOR HIGH PERFORMANCE COMPUTING

University of Cape Town • iYunivesithi yaseKapa • Universiteit van Kaapstad

OUTLINE

- INTRODUCTION
- PROBLEMS AFFECTING PERFORMANCE OF PARALLEL PROGRAMS
- WHY PROGRAMS DO NOT PERFORM WELL IN THE SUPERCOMPUTERS?
- WHY IS IT DIFFICULT TO OPTIMIZE PARALLEL PROGRAMS?
- METHODS USED TO ANALYSE PARALLEL PERFORMANCE
- OVERVIEW OF VISUALIZATION TOOLS:
  - Scalasca
  - Vampir
  - Paraver
  - Periscope
- DIFFERENT TYPES OF SUPERCOMPUTING SYSTEMS
- THE STRUCTURE OF CHPC – SUN CLUSTER
- OBJECTIVE OF THE STUDY
- RESEARCH QUESTIONS
- IMPACT OF THE STUDY
- RESEARCH METHODOLOGY
- EXPECTED RESEARCH OUTCOME
- REFERENCE

CHPC CENTRE FOR HIGH PERFORMANCE COMPUTING

University of Cape Town • iYunivesithi yaseKapa • Universiteit van Kaapstad

INTRODUCTION

Different scientific disciplines (such as material science, oceanography and chemistry) utilise parallel applications to execute various calculations in the supercomputing systems. Parallel program is the simulation that can simultaneously perform many different operations in a computing system and run on different types of parallel systems:

- Distributed-memory system
- Shared-memory system
- Vectors system

Some of this parallel programs do not perform up to optimum level (meaning not executing as expected) and fully utilising computational resources (e.g. memory and processors) within the parallel system. In some instances, application users use visualization tools to trace bottlenecks within the program and help to increase application's performance within the computational system.

PROBLEMS AFFECTING PERFORMANCE OF APPLICATIONS

In a supercomputer, the performance of parallel applications is mainly affected by interplays of factors such as (Adve and Vernon, 2004; Ebnenasir and Beik, 2009):

- limited network bandwidth.
- unevenly distribution of message-passing.
- slow read/write requests within the storage.
- logic of the parallel code
- high memory latency in processing nodes.
- High processor utilisation in the execution nodes.

It is therefore important to understand the role of the below computational factors in order to increasing efficiency of the code and overall utilisation of the parallel system.

WHY PROGRAMS DO NOT PERFORM WELL IN THE SUPERCOMPUTERS?

It is inevitable that program developers do not have the same level of parallel programming experience and this may lead to parallel applications not scaling well due to inconsistencies such as embarrassing parallelism within the code of the program.

The continuous changes of hardware architectures in the supercomputing community also contribute to none standard approach for developing applications meant to run in the parallel systems. For example, when a system is equipped with different types of processing systems (e.g. shared-memory and distributed-memory system) then one may need to parallelize the applications using OpenMP in a shared-memory system and apply OpenMPI for distributed-memory system.

Some of the application users do not have sufficient knowledge of optimizing programs to achieve better performance within the system and this may even become more challenging when one does not understand some aspects of programming used to design the code of the program.

WHY IS DIFFICULT TO OPTIMIZE PARALLEL PROGRAMS

The word “Optimization” is referring to a process of rewriting some program instructions with the purpose of increasing the performance and/or efficiency of the code on a particular computing system. it is hard to optimize parallel programs due to factors such as complex structure of the parallel system and many activities (such as message passing tasks, read/write activities and network communication) that need to be analysed in order to optimize the correct area of the parallel code. The visualization tools attempt to simplify the process of optimization but not all of them can be able to provide useful information about the program (Schwartz-Narbonne et al., 2011):

- Identify exact area of the problem within the code.
- Identify all computational activities causing performance bottlenecks.
- Level of parallelism within code of the application.

METHODS USED TO ANALYSE PARALLEL PERFORMANCE

Some of the parallel computing users prefer to utilise visualization and analyses tools such as Scalasca and Vampir in an attempt to investigate the performance of parallel programs (Geimer et al., 2010; Subotic et al., 2010).

On the other hand, others manually index and search the syntax of the code to increase performance and efficiency of the program (Schwartz-Narbonne et al., 2011).

OVERVIEW OF VISUALIZATION TOOLS

Different visualization tools utilise various techniques to perform analyses and trace the behaviour of the application executed in the parallel system. Some of these analyses software trace and analyse the performance of the parallel model during its execution (Geimer et al., 2010) while others can also visualize the statistics of the application after the simulation have completed to run in the system (Becker, Frings and Wolf, 2008). This performance analyses tools also support different types of applications; for example; some of the analyses and visualization software support MPI programs while others are capable of analysing OpenMP applications. In this study, we discuss the following:

- Scalasca.
- Vampir.
- Paraver.
- Periscope.

SCALASCA

Scalable Performance Analysis of Large-Scale Applications (Scalasca) is an open-source performance analysis tool widely used to identify both MPI/ OpenMP communication and synchronization barriers within the parallel programs (Geimer et al., 2010; Böhme et al., 2010).

Scalasca provides both runtime summaries and event traces. The runtime summary is a performance report of the program during execution, while event tracing provides detail performance analysis report after run completion. During a run, it reports on the total execution time of the program; hardware (e.g. memory and processor) utilisation, message-passing statistics such as numbers of synchronization, communication and read/write activities like number of bytes transferred, sends/writes versus receives/reads requests.

For more information visit: http://www.scalasca.org.

METHOD OF TRACING EVENTS & RUNTIME

Scalasca have two different ways, namely, event trace and runtime mode of recording the performance of the parallel program in the system. It instruments the target application with its libraries and run the code on a parallel system. During execution, it allows the user to select one of the modes.

In event trace mode, it generate a separate file for each process/event that take place during the simulation of the program. It then load all the trace files (binary files containing information about the events/processes) in the main memory and analyse them using the same number of processors used to simulate the model. Thereafter, the analyses system identifies bottlenecks (such as wait states, late message sender or receiver, wasted memory, processor and network).

As for runtime mode, Scalasca collects the statistics of the parallel application and report immediately without storing any data.

Advantage and disadvantages

The main advantage of Scalasca is to perform the following:

- map wait states with the wasted computational resources.
- identify late message senders versus receivers.
- display load and communication imbalance occurred between the actual system and target application.

However, one of the challenges of this analyses tool is that it creates file for each process/event and this may cause a need for more storage capacity to store those file. It also depend on a local time of the execution nodes to accurately record the movement of the process and associate it with time, of which, this can be disadvantageous for a system that have nodes which contains different time zones.

VAMPIR

Vampir (Visualization and Analysis of MPI Resources) is a timeline visualization and analyses tool (Becker, Frings and Wolf, 2008) mainly used to provide more detail information about the processes and events simulated during the execution of either MPI and/or OpenMP application (Hein et al., 2005; Tang, 2008). This visualization tool was developed at the Centre for Applied Mathematics of Research Centre, Jülich and Centre for High Performance Computing, Technische Universität Dresden and has capability to zoom/ navigate into processes of the simulation. It also provide useful information about execution of the application v/s system (Becker, Frings and Wolf, 2008):

-Total execution time taken to simulate an application in the system.
- Time taken to simulate application's code.
- Time taken to process message passing parameters.
- Time taken to trace overheads.
- Time taken to execute each function of the program.
- How much memory was used for each function and for how long?
- High-level statistics of message passing and read/write activities.

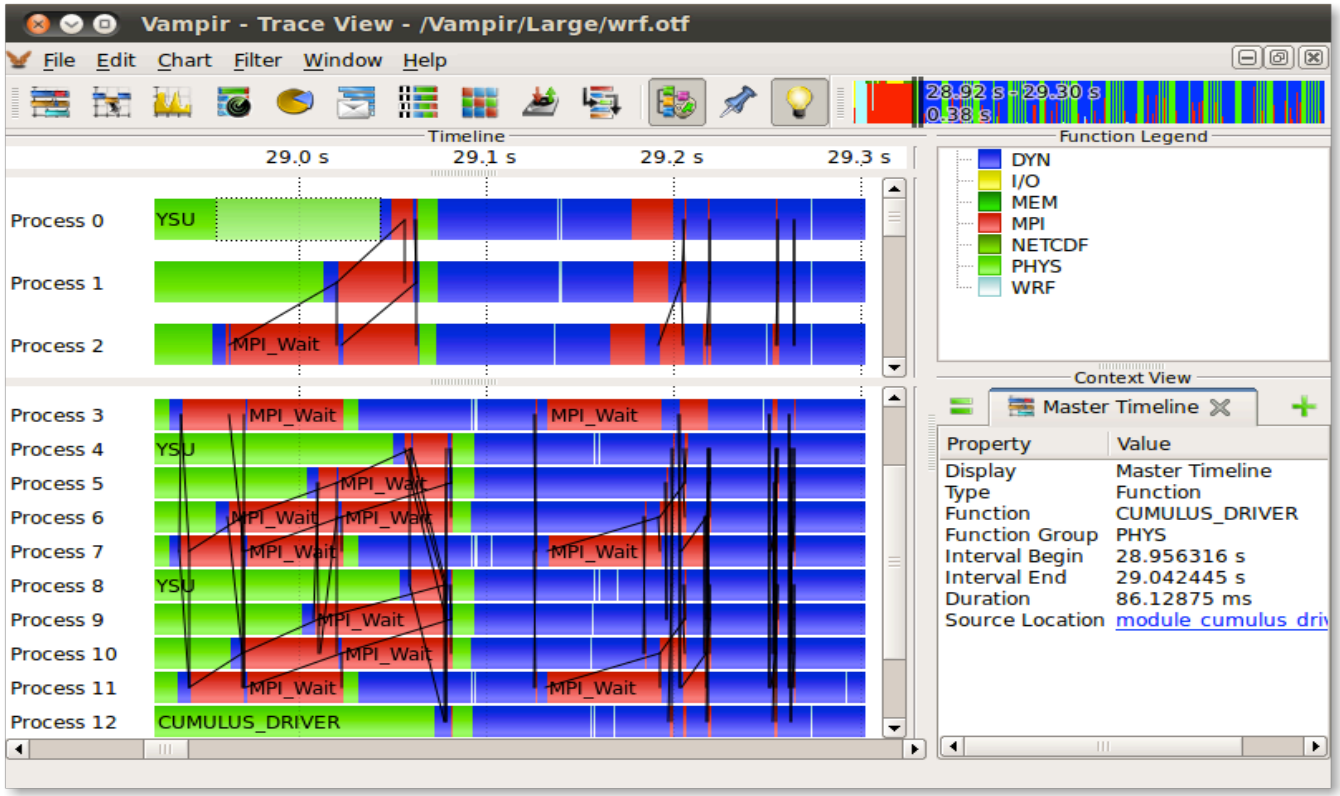
METHOD USED TO TRACE PARALLEL APPLICATIONS

The Vampir use a built-in library called Vampir Trace to instrument the code of the parallel program in the supercomputing system. Thereafter, it runs an executable to simulate the program and collect information about the performance behaviour of the application. The information about the performance of the application is then stored in a file that supports OTF (Open Trace Format).

It also has capability to convert data files of other third-party analyses tools (such as Scalasca, Paraver) and visualize performance behaviour of the program simulated in the computational. It is understood that PAPI (Performance Application Programming Language) is used to gather information related to hardware utilisation and pass-it to Vampir for reporting purpose. (Knüpfer, Brunst and Nagel, 2005).

ADVANTAGES AND DISADVANTAGES

The most powerful feature of Vampir is to zoom into application processes and able to visualize data generated by other performance analyses tools. The only obstacle is its limited depth (width and height) that cannot be able to display visualization information of programs that are communication intensive (Becker, Frings and Wolf, 2008).



PARAVER

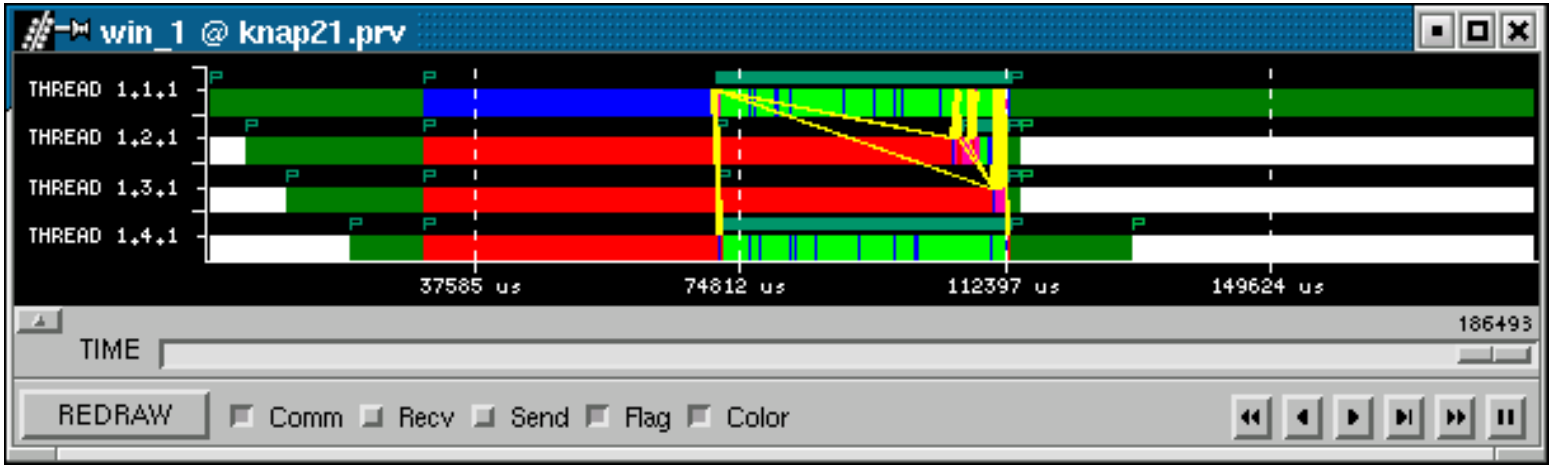
Paraver is described as the visualization analysis tool designed to analyse performance of applications that are communication and computationally intensive when running in the parallel system (Dorta, Leon and Rodriguez, 2006; Subotic et al., 2010; Truong et al., 2001) and was developed at the Barcelona Supercomputing Centre, Computer Science. The tool provides two different types of visualization information, namely, timeline display and statistical summary.

Timeline display arranges statistics according to processes (processing threads) and objects such as mpi task, processor, memory and node utilisation where one can be able to zoom and get more details about performance of a particular object (Dorta, Leon and Rodriduez, 2006). It is also able to provide useful information programs that are communication intensive, for example, it can determine level of parallelization of the application, processor utilisation per task, late message senders v/s late receivers, communication load per task, instructions per cycle executed by each thread, load balance of different parallel loops.

METHOD USED TO COLLECT PERFORMANCE EVENTS

Paraver collects performance events of the application by instrumenting and executing the all areas of the parallel code. This visualization tool will then generate a trace file that contains the following fields:

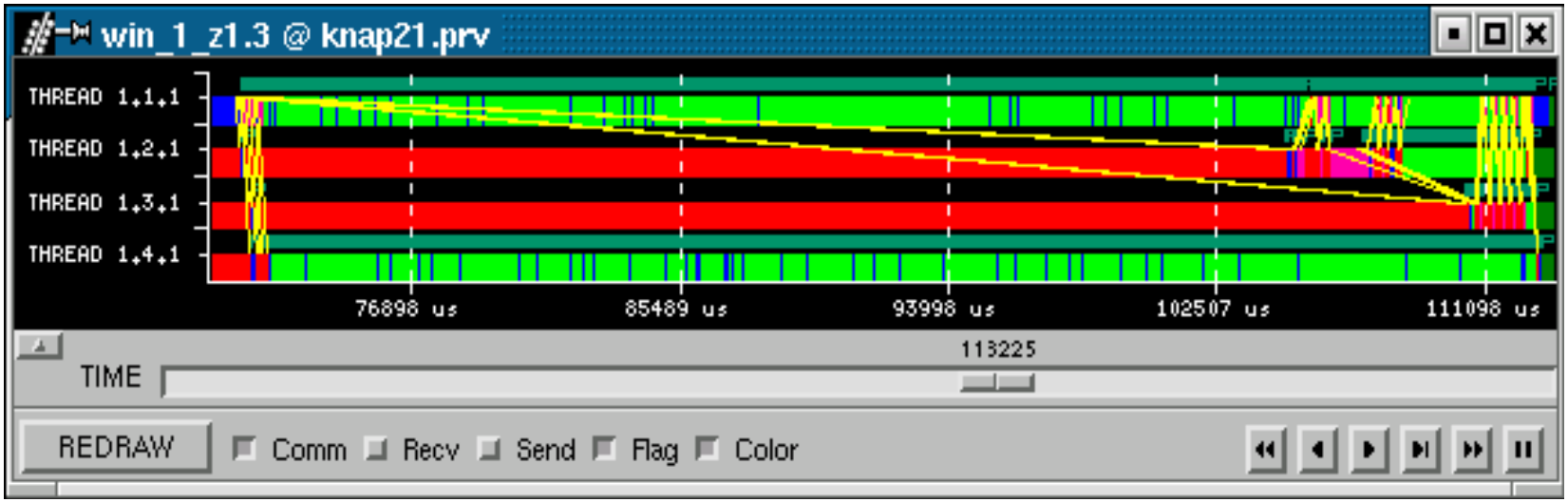
- communication (relationship between the events).
- state status of processing thread.
- events (event of the object).



(Dorta, Leon and Rodriguez, 2006)

ADVANTAGES AND DISADVANTAGES

Paraver has capability to simultaneously trace performance of more than one application and provide statistics about various parts of the applications and system. However, it is not able to map performance bottlenecks to the exact area of the application's code.



(Dorta, Leon and Rodriguez, 2006)

PERISCOPE

Periscope is a distributed performance analysis tool that search performance bottlenecks and map them to the exact area of the application's code running in the computational systems (Benedict, Petkov and Gerndt, 2010; Petkov and Gerndt, 2010) and is currently under-development at Technische Universitat Munchen in Integrated Systems and Application Analyses.

It does not have features to present data in a graphical way; instead; it provides the application performance statistics using tables with rows and columns. In summary, it can be able to display number of processing threads per function, threads/loops cycles that was lost and reason for lost of computational cycles (Gerndt and Ott, 2009).

For more information about periscope visit: http://www.lrr.in.tum.de/~periscop.

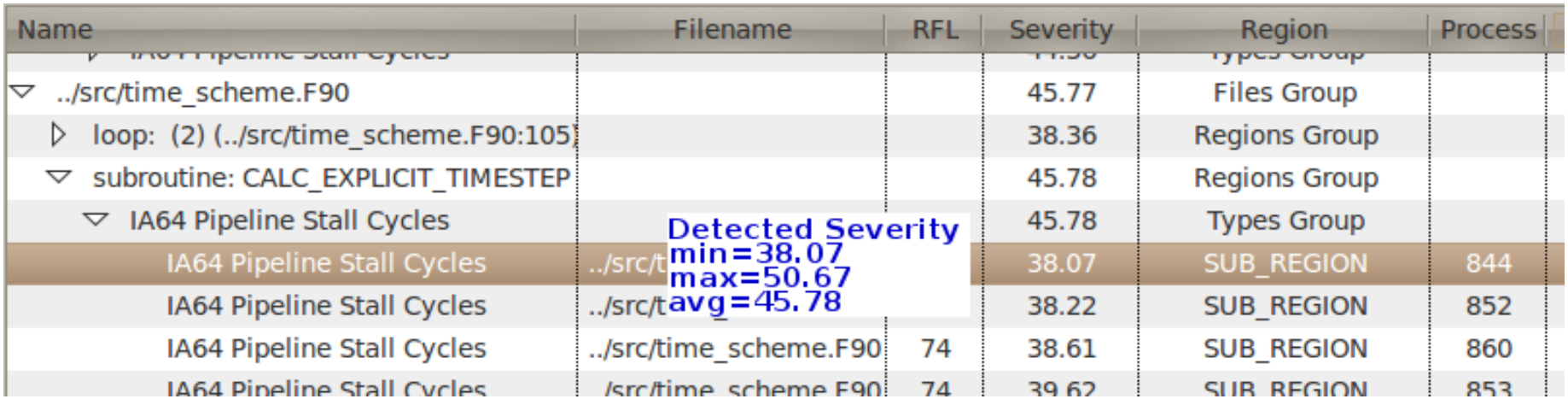
METHOD USED TO INSTRUMENT AND TRACE APPLICATION

Periscope instruments the application and store data about the source code (e.g. main routine, subroutines, loops, data structure, etc) of the application, which can later be used to transform and re-compile the code of the application (Petkov and Gerndt, 2010). It also allows the simulation users to start, stop and resume the analyses of the application at any time.

It also has capability to detect and map problem to the specific area of the code using its capability to index and highlight syntax errors within the code of the program (Benedict, Petkov and Gerndt, 2010). This performance analyses tool is designed to identify errors within the code of the program and analyse level of parallelism applied in the parallel program.

ADVANTAGES AND DISADVANTAGES

The advantage of this analyses tool is to be able to map performance bottlenecks (e.g. imbalanced communication between the message passing activities, excessive memory and processor utilisation) to certain area of the program's code. However, it does not have a graphical interface and this may be a challenge to other users who prefer graphs to visualize performance of the parallel program in the supercomputers.



(Petkov and Gerndt, 2010)

DIFFERENT TYPES OF SUPERCOMPUTERS

In this study, we classify different architectures of parallel systems as follows: MISD, MIMD and SIMD systems:

- MISD (Multiple Instruction Single Data) - involve multiple processors applying different instructions to a single data.

- MIMD (Multiple Instructions Multiple Data) - multiple processing systems that are able to simultaneously execute various instructions of different data such as distributed-memory system (clusters) and shared-memory system (symmetric multiprocessors).

- SIMD (Single Instructions Multiple Data) - multiple processors simultaneously executing the same instruction of different data level e.g. vector.

THE STRUCTURE OF CHPC – SUN CLUSTER

The Centre for High Performance Computing, South Africa is currently hosting the biggest supercomputing system (Sun Cluster) in the African region and is performing up to 61.5 Teraflops/s (Linpack benchmark). The CHPC - Sun cluster consists of different computational architectures, namely, Nehalem (Intel Xeon X5570 processors), Westmere (Intel Xeon X5670 processor), Harpertown (Intel Xeon E5450 processors) and M9000 (SMP - Sparc processors). The system is equipped with a total of 6976 processors connected to the Storage Area Networks (SAN) of 480 Terabytes running Lustre file system.

For more information about Sun cluster visit: http://www.chpc.ac.za/sun.

CHPC CENTRE FOR HIGH PERFORMANCE COMPUTING

University of Cape Town · iYunivesithi yaseKapa · Universiteit van Kaapstad

OBJECTIVE OF THE STUDY

The main aim of the study is to design a visualization system in the CHPC - Sun cluster that will help supercomputing users to optimize parallel programs to achieve better performance and efficiency within the system.

RESEARCH QUESTIONS

- Can an improved visualization system help users to optimize and improve the performance of their parallel application?

- How will local and international supercomputing community benefit on the newly improved visualization system?

- Can an improved visualization system provide useful information that lead the user to an exact area of the problem within the parallel code?

- How will visualization system safe storage capacity when simulate parallel models that solve large problem?

- Can a newly improved visualization system enable users to quickly filter and search information about the execution behaviour of the parallel program?

IMPACT OF THE STUDY

- Benefit of visualization for optimizing

Visualization system will enable application users to optimize parallel programs within a short period of time.

- More effective use of supercomputers

The supercomputing users will be able to scale parallel applications and utilise parallel systems up to optimum level.

- More understanding of the source of bottlenecks

The visualization system will enable both application users and developers to obtain useful information that lead to exact area of the problem within the parallel code.

- Value of understanding the performance of parallel applications in the CHPC cluster

Users need to understand the performance of applications in order to safe computational resources and at the same time increase system utilisation.

RESEARCH METHODOLOGY

Methodological approach that will be used to design the visualization system in the CHPC Sun cluster:

To perform computational experiment, visualization system will be install in the CHPC - Sun cluster. Different ways of visualizing information (e.g. display information in a circular or rectangular way) will be tested and applied on the selected visualization system. Moreover, the method of compressing data will be used to reduce the trace files that are generated during the execution of the model. To valid the system, parallel applications will be optimized using the newly improved visualization tool in an attempt to increase the performance in the parallel system.

EVALUATION

The visualization server will be installed in the Sun cluster (login node) that utilise Moab scheduler/Torque resource manager to distribute and compute parallel applications in the execution nodes. To obtain more information about the behaviour of the visualization system, one will run parallel jobs using applications that are memory-intensive and demand more processing capabilities due to the nature of their problems.

The analyses results of the applications will also be used to understand the way visualization system present and trace execution of parallel programs that perform many calculations. Various efficient ways of presenting visualized data and compressing generated trace files will be adopted in the system. Furthermore, parallel applications will be instrumented and analysed using libraries of the newly improved visualization tool. This will further determine the benefit of visualization system to the users of the supercomputers.

VISUALIZATION TOOL AND PARALLEL PROGRAMS. WHY SELECTED?

Scalasca has been selected as the performance analyses tool that will be used to perform computational experiment in the Sun cluster. The selection process is based on the fact that the tool is an open-source and has some interesting problems such as generating excessive trace files, not able to adequately filter and present visualization information of parallel models that simulate large problem sizes.

The following parallel applications, namely, DL_POLY (Smith, Yong and Rodger, 2002; Smith, 2006) and NAMD (Phillips et al., 2005; Huang et al., 2008) will be used to optimized using the improved visualization tool (Scalasca) and achieve better performance in in the selected supercomputing system. The reason for selecting this applications lies on the fact that both programs are open-source and run most frequently in the CHPC Sun cluster.

PROCESS TO TEST, ANALYSE, VISUALIZE AND OPTIMIZE APLICATIONS

The parallel applications will be simulated and visualized using the libraries of the improved visualization system in order to identify bottlenecks affecting the performance and efficiency of codes in the Sun cluster. Some certain areas of the parallel codes will be optimized in an attempt to improve their performance and efficiency in the selected supercomputing system. The process of optimization will involve analysing the execution behaviour of the parallel applications using the selected visualization tool.

EXPECTED RESEARCH OUTCOMES

It is expected that the study will produce an improved visualization system that will help supercomputing users to address problems affecting the performance of their parallel models. It is also anticipated that new knowledge and skills of optimizing applications will be shared across the HPC community.

REFERENCE

Adve, V.S., and Vernon, M.K. (2004). Parallel program performance prediction using deterministic task graph analysis. ACM Transactions on Computer Systems. 22(1): 94-136.

Becker, D., Frings, W., and Wolf, F. (2008). Performance evaluation and optimization of parallel grid computing applications. Proceedings of the 16th euromicro conference on parallel, distributed and network-based processing. Washington, DC: IEEE computer society, pp. 193-199.

Benedict, S., Petkov, V., and Gerndt, M. (2010). Periscope: An online-based distributed performance analysis tool. Proceedings of the 3rd international workshop on parallel tools for high performance computing, edited by: M.S. Müller, M.M. Resch, A. Schulz and W.E. Nagel. Berlin, Heidelberg: Springer, pp. 1-16.

Böhme, D., Geimer, M., Wolf, F., and Arnold, L. (2010). Identifying the root causes of wait states in large-scale parallel applications. 39th international conference on parallel processing. San Diego: IEEE, pp. 90-100.

Dorta, I., Leon, C., and Rodriguez, C. (2006). Performance analysis of branch-and-bound skeletons. 14th Euromicro international conference on parallel, distributed and network-based processing, edited by: J.D. Cantarella. New York: IEEE Computer Society, pp. 8.

Ebnenasir, A., and Beik, R. (2009). Developing parallel programs: a design-oriented perspective. Proceedings of the 2009 IEEE 31st international conference on software engineering. Vancouver: IEEE Computer Society, pp. 1-8.

REFERENCE (cont…)

Geimer, M., Wolf, F., Wylie, B.J.N., Abraham, E., Becker, D., and Mohr, B. (2010). The scalable performance toolset architecture. Concurrency and Computation: Practice and Experience. 22(6): 702-719.

Gerndt, M., and Ott, M. (2009). Automatic performance analysis with periscope. Concurrency and Computation: Practice and Experience. 22(6): 736-748.

Hein, J., Reid, F., Smith, L., Guest, M., and Sherwood, P. (2005). On the performance of molecular dynamics applications on current high-end systems. Philosophical Transactions of the Royal Society. 363(1833): 1987-1998.

Knüpfer, A., Brunst, H., and Nagel, W.E. (2005). High performance event trace visualization. Proceedings of the 13th euromicro conference on parallel, distributed and network-based processing. Washington, DC: IEEE computer society, pp. 258-263.

Petkov, V., and Gerndt, M. (2010). Integrating parallel application development with performance analysis in periscope. IEEE international symposium on parallel & distributed processing, workshops and Phd forum. Atlanta: IEEE Computer Society, pp. 1-8.

Schwartz-Narbonne, D., Lui, F., Pondicherry, T., August, D., and Malik, S. (2011). Parallel assertions for debugging parallel programs. 9th IEEE/ACM international conference on formal methods and models for codesign, edited by: J. Brandt. New Jersey: IEEE Xplore Digital Library, pp. 181-190.

REFERENCE (cont…)

Subotic, V., Sancho, J.C., Labarta, J., and Valero, M. (2010). A simulation framework to automatically analyse the communication-computation overlap in scientific applications. IEEE international conference on cluster computing, edited by: P. Kellenberger. New York: IEEE Computer Society, pp. 275-283.

Tang, E. (2008). DL_POLY 3.0: Performance study of a Si02/Water system. Edinburgh: The University of Edinburgh, pp. 1-96.

Truong, H., Fahringer, T., Madsen, G., Malony, A.D., Moritsch, H., and Shende, S. (2001). On using SCALEA for performance analysis of distributed and parallel programs. Proceedings of the ACM/IEEE SC2001 conference. Denver, Colorado: IEEE Xplor Digital Library, pp. 37.

ACKNOWLEDGEMENT

The author would like to acknowledge the following:

CHPC, CSIR, Meraka Institute for their support and funding of these project.

Dr. Happy Sithole and Dorah Thobye for their continuous efforts on development of this research project.

Dr. Daniel Moeketsi for providing useful information and valuable advises in this study.

Prof. Jack Dongarra for his commitment to strengthen the collaboration between University of Tennessee and Centre for High Performance Computing.