A Study on Performance Monitoring Techniques for Traditional Parallel Systems and Grid

Sarbani Roy (nee Ghosh) [1] and Nandini Mukherjee [2]

[1] Department of Computer Science and Engineering,
St. Thomas College of Engineering and Technology,
West Bengal University of Technology, Kolkata 700 023, India
sarbani_roy77@yahoo.co.in
[2] Department of Computer Science and Engineering,
Jadavpur University, Kolkata 700 032, India
nmukherjee@vsnl.com

Abstract: During the last decade, much research has been done to evolve cost-effective techniques for performance monitoring and analysis of applications running on conventional parallel systems. With the advent of technology, new Grid-based systems emerged by the end of the last decade, and received attention as useful environments for high-performance computing. The requirements for discovering new techniques for monitoring the Grid have also come to the front. The objective of this paper is to study the techniques used by tools targeting traditional parallel systems and highlight the requirements and techniques used for Grid monitoring. We present case studies of some representative tools to get an understanding of both paradigms.

1. Introduction

Computer scientists and researchers have a long-time target to develop effective methods and tools for performance analysis of compute-intensive applications and providing support for automatic parallelisation and performance tuning. In other words, performance engineering of such applications in a cost-effective manner has been a major focus of the researchers during the last decade. Many tools for measuring or predicting performance of serial / parallel programs have been developed. These tools were designed with diverse objectives, targeted different parallel architectures and adopted various techniques for collection and analysis of performance data. The scope of these tools comprises instrumentation, measurement (monitoring), data reduction and correlation, analysis and presentation and finally, in some cases, optimisation.

The second half of the last decade however witnessed radical changes in technology. Clusters of workstations / personal computers / SMPs offered an attractive alternative to traditional supercomputers, as well as MPPs. With the advent of Internet technology, we then entered the era of Grid Computing. Introduction of this new paradigm in the world of high performance computing has forced the computer scientists to look into the performance-engineering problem with an entirely different

H. Jin et al. (Eds.): NPC 2004, LNCS 3222, pp. 38-46, 2004.

view. The term Grid indicates an execution environment in which high speed networks are used to connect supercomputers, clusters of workstations, databases, scientific instruments located at geographically distributed sites. Monitoring such an environment for executing compute-intensive applications and performance engineering these applications is one of the central tasks of Grid computing research. Performance monitoring of Grid and Grid applications, by its very own nature, differs from the underlying principles of traditional performance analysis tools. The heterogeneous nature of computational resources, potentially unreliable networks connecting these resources and different administrative management domains pose the biggest challenge to the performance monitoring / engineering community in Computer Science research.

The objective of this paper is to study the main trends of traditional performance analysis tools and Grid monitoring tools, to compare the techniques used by them, to focus on the major issues of Grid monitoring and applicability of various techniques for monitoring the applications and resources in Grid environment. Traditional performance analysis tools are discussed in Section 2 based on some case studies. The paper then summarises the properties of Grid and compares with conventional distributed systems. It also highlights the main issues that need to be tackled while designing a Grid monitoring system (Section 3). Case studies on representative Grid monitoring tools are furnished in Section 4 followed by a discussion on various aspects of Grid monitoring. Section 5 presents related work and Section 6 concludes.

2. Performance Monitoring and Analysis for Traditional Parallel Systems

This section focuses on the performance monitoring and analysis techniques adapted for use on traditional parallel systems. We present case studies on two performance analysis tools with an emphasis on their important features.

2.1 SCALEA

SCALEA is a performance analysis tool developed at the University of Vienna [17]. It provides support for automatic instrumentation of parallel programs, measurement, analysis and visualization of their performance data. The tool has been used to analyse and to guide the application development by selectively computing a variety of important performance metrics, by detecting performance bottlenecks, and by relating performance information back to the input program.

The components of *SCALEA* include *SCALEA Instrumentation System (SIS)*, *SCALEA Runtime System*, *SCALEA Performance Data Repository*, and *SCALEA Performance Analysis and Visualization System*. Each of these components can also be used as a separate tool. The *SCALEA Instrumentation system (SIS)* enables the user to select code regions of interest and automatically inserts monitoring code to collect all relevant performance information during an execution of the program. An

execution of a program on a given target architecture is referred to as an experiment. Unlike most performance tools, *SCALEA* supports analysis of performance data obtained from individual experiments, as well as from multiple experiments. All the important information about performance experiments including source code, machine information and performance results are stored in a data repository. *SCALEA* focuses on four major categories of temporal overheads including data movement, synchronization, control of parallelism, and additional computation. The classification of overheads in *SCALEA* is hierarchical.

2.2 Paradyn

Paradyn, from the Computer Sciences Department of University of Wisconsin, Madison, is a tool for measuring and understanding the performance of serial, parallel and distributed programs [8]. It consists of a data collection facility (the Data Manager), an automated bottleneck search tool (the Performance Consultant), a data visualization interface (the Visualization Manager), and a User Interface Manager. The central part of the tool is a multi-threaded process and communication between threads is defined by a set of interfaces constructed to allow any module to request a service of any other module.

The performance consultant module of *Paradyn* automates the search for a predefined set of performance bottlenecks based on a well-defined model, called the *W^3 search model*. It attempts to find the answer of three questions: firstly why is the application performing poorly, secondly where is the performance problem located, and finally when does the problem occur. *Paradyn* uses *dynamic instrumentation* to instrument only those parts of the program relevant for finding the current performance problem. Dynamic instrumentation defers instrumenting the program until it is executed and dynamically inserts, alters and deletes instrumentation during program execution. What data to be collected is also decided during the execution of the program under the guidance of the Performance Consultant.

2.3 Discussion

In this section, we summarise the salient features of performance analysis tools discussed above. Although these tools are only representatives of the vast number of performance analysis tools available for traditional parallel architectures, the following discussion will give some idea about the techniques adopted by majority of them.

- The important **components** of performance analysis tools are (a) tools for instrumentation and data collection, (b) tools for analysing and identification of performance problems, and (c) tools for visualisation.
- Monitoring data is **collected** through source code instrumentation and trace libraries are used for recording monitoring events. For example, *SCALEA* uses SISPROFILING, and PAPI (does profiling using timers, counters, and hardware parameters) libraries.

- **Source code instrumentation** is generally guided by the user (in an interactive manner). Modern tools employ dynamic instrumentation techniques to dynamically change the focus and performance bottlenecks are identified during run-time.
- The tools are designed to work on divers **hardware platforms.** For example, the platform dependent part of *Paradyn* is available for SPARC, x86 and PowerPCs.
- One requirement of these tools is to support different **programming paradigms**. *SCALEA*, for example, supports performance analysis of HPF, OpenMP and MPI codes.
- Each tool defines their own **data storage formats**, thus limiting the portability of monitoring data.

3. High Performance Computing and Grids

Grids are persistent environments that enable software applications to integrate computational and information resources, instruments and other types of resources managed by diverse organizations in widespread locations. Computational resources on a Grid together can solve very large problems requiring more resources than is available on a single machine. Thus harnessing the power of these distributed computational resources "computational power grids" may be created for high performance computing. This concept is often referred to as a “Computational Grid”.

Grids are usually viewed as the successors of distributed computing environments; although from the user's point of view conventional distributed systems and Grids are intrinsically semantically different. The resources in a Grid are typically heterogeneous in nature with fast internal message passing or shared memory [1]. A relatively slow wide area network externally connects the individual resources.

3.1 Performance Analysis in Grid Environment

Grid performance analysis is distinctively different from the performance analysis for traditional parallel architectures. The fundamental and ultimate goal of parallel processing is speeding up the computation, i.e. executing the same task in shorter time. In a Grid environment, however, speed of the computation is not the only issue. Mapping application processes to the resources in order to fulfill the requirement of the application in terms of power, capacity, quality and availability forms the basis of Grid performance evaluation. The huge amount of monitoring data generated in a Grid environment are used to perform fault detection, diagnosis and scheduling in addition to performance analysis, prediction and tuning [2]. The dynamic nature of Grid makes most parallel and distributed programming paradigms unsuitable and the existing parallel programming and performance analysis tools and techniques are not always appropriate for coping with such an environment.

The important elements that make performance analysis for Grids different from the performance analysis for SMPs or MPPs may be summarised as below [10]:

- **Lack of prior knowledge about the execution environment:** The execution environment in a Grid is not known beforehand and also the environment may vary from one execution to another execution and even during the same execution of an application.
- **Need for real-time performance analysis:** Due to the dynamic nature of a Grid-based environment, offline analysis is often unsuitable. Sometimes it becomes necessary to predict the behaviour of an application based on some known characteristics of the environment. However, in the absence of precise knowledge about the future shape of the Grid, this prediction-based analysis is also of little use.
- **Difficulty in performance tuning of applications:** It may not be possible to exactly repeat an execution of a given application in a dynamic Grid environment. Therefore, performance tuning and optimisation of the application is difficult.
- **Emergence of new performance problems:** The very different nature of a Grid infrastructure gives rise to new performance problems that must be identified and tackled by the performance analysis tool.
- **Requirement of new performance metrics:** Usual performance metrics used by traditional performance analysis tools are inappropriate and therefore new performance metrics must be defined.
- **Overheads due to Grid management services:** Grid information services, resource management and security policies in a Grid environment add additional overhead to the execution of an application.

Global Grid Forum proposed a Grid Monitoring Service Architecture [6, 15] which is sufficiently general and may be adapted in variety of computational environments including Grid, clusters and large compute farms. The essential components of the architecture as described in [6] are as follows:

- *Sensors:* for generation of time-stamped performance monitoring events for hosts, network processes and applications. Error conditions can also be monitored by the sensors.
- *Sensor Manager:* responsible for starting and stopping the sensors.
- *Event Consumers:* request data from sensors.
- *Directory Service:* for publication of the location of all sensors and other event suppliers.
- *Event Suppliers or Producers:* for keeping the sensor directory up-to-date and listening to data requests from event consumers.
- *Event archive:* for archiving the data that may be used later for historical analysis.

In the next Section we present case studies on two Grid monitoring tools and highlight their important characteristics.

4. Performance Monitoring and Analysis Tools for Grid

Currently available Grid Monitoring Tools may be classified into two main categories: Grid infrastructure monitoring tools and Grid application monitoring tools. Although the primary focus of majority of the tools is on monitoring and analysis of

Grid infrastructure, many tools are nowadays concentrating on a unified approach. In the following sections, we present brief overviews of three such tools. It must be noted that there is no specific motive behind the selection of these three tools apart from rendering some basic idea about the requirements of Grid monitoring tools and their major components. A detail survey of existing performance monitoring and evaluation tools may be found in [5] which has an objective of creating a directory of tools for enabling the researchers to find relevant properties, similarities and differences of these tools.

4.1 SCALEA-G

SCALEA-G [18], developed at the University of Vienna is the next generation of the *SCALEA* System. SCALEA-G is a unified system developed for monitoring Grid infrastructure based on the concept of Grid Monitoring Architecture (GMA) [15] and at the same time for performance analysis of Grid applications [6]. SCALEA-G is implemented as a set of Grid services based on the Open Grid Service architecture (OGSA) [4]. OGSA-compliant grid services are deployed for online monitoring and performance analysis of a variety of computational resources, networks, and applications.

The major services of SCALEA-G are: (a) *Directory Service* for publishing and searching information about producers and consumers of performance data, (b) *Archival Service* to store monitored data and performance results, (c) *Sensor Manager Service* to manage sensors. Sensors are used to collect monitoring data for performance analysis. Two types of sensors are used: *system sensors* (for monitoring Grid infrastructure) and *application sensors* (to measure the execution behaviour of Grid applications). XML schemas are used to describe each kind of monitoring data and any client can access the data by submitting Xpath / Xquery-based requests. The interactions between sensors and Sensor Manager Services also take place through the exchange of XML messages.

SCALEA-G supports both source code and dynamic instrumentation for profiling and monitoring events of Grid Applications. The source code instrumentation service of SCALEA-G is based on *SCALEA* Instrumentation System [17]. Dynamic instrumentation (based on Dyninst from [3]) is accomplished by using a *Mutator Service* that controls the instrumentation of application process on the host where the process is running. An XML-based Instrumentation Request Language (IRL) has also been developed for interaction with the client.

4.2 GrADS and Autopilot

Researchers from different universities led by a team at Rice University are working in the GrADS project [12] that aims at providing a framework for development and performance tuning of Grid applications. As a part of this project, a *Program Execution Framework* is being developed [7] to support resource allocation and reallocation based on performance monitoring of resources and applications. A

performance model is used to predict the expected behaviour of an application and actual behaviour is captured during its execution from the analysis of measured performance data. If there is a disagreement between the expected behaviour and observed behaviour, actions like replacement or alteration of resources or redistribution of the application tasks are performed.

The monitoring infrastructure of GrADS is based on Autopilot, which monitors and controls applications, as well as resources. The Autopilot toolkit supports adaptive resource management for dynamic applications. Autopilot integrates dynamic performance instrumentation and on-the-fly performance-data reduction with configurable, malleable resource management algorithms and a real-time adaptive control mechanism. Thus, it becomes possible to automatically choose and configure resource management policies based on application request patterns and system performance [13]. Autopilot provides a flexible set of performance sensors, decision procedures, and policy actuators to accomplish adaptive control of applications and resources. The policy implementation mechanism of the tool is based on fuzzy logic.

4.3 Discussion

The Section describes some representative tools for Grid monitoring and performance tuning of Grid Applications. It is evident that on a large, complex and widely distributed system similar to Grid, post-mortem performance analysis is of no use. Thus, all tools perform real-time analysis and some use these analysis data for automatic performance tuning of applications. The essential ingredients of real-time analysis are

- *dynamic instrumentation and data collection mechanism* (as performance problems must be identified during run-time),
- *data reduction* (as the amount of monitoring data is large and movement of this amount of data through the network is undesirable),
- *low-cost data capturing mechanism* (as overhead due to instrumentation and profiling may contribute to the application performance), and
- *adaptability to heterogeneous environment* (for the heterogeneous nature of Grid, the components along with the communicating messages and data must have minimum dependence on the hardware platform, operating system and programming languages and paradigms).

The basis of most monitoring tools is the general Grid Monitoring Architecture proposed by the Global Grid Forum. However, when the tools extend their functionality to incorporate real-time performance analysis and tuning, they have to confront with complex issues like fault diagnosis and application remapping.

Many of the already existing libraries and performance monitoring tools for traditional parallel systems form parts of the Grid monitoring tools. For example, SCALEA-G uses the instrumentation system of *SCALEA* [17] and Dyninst [3] and one of the current foci of GrADS project is to incorporate the instrumentation of SvPablo [14].

5. Related Work

Performance tools for conventional parallel systems are in use for quite some time. Therefore, many researchers have already studied, discussed and compared these tools. Some of the recent studies focus on the survey of Grid monitoring tools [2, 5]. In particular, [5] provides a directory of recently available tools along with a comparative analysis of them.

The objective of this paper is not to prepare a survey report, nor does it aim at the comparative analysis of performance tools. The main intention is to study the requirements for Grid monitoring tools in comparison to the standard performance monitoring and analysis techniques applicable to conventional parallel architectures.

6. Conclusion

This paper discussed and highlighted different monitoring techniques for traditional parallel architectures and Grids. The observations that surfaced in this paper grow an understanding of the recent road-map of performance engineering tools for developing high performance applications targeting modern systems. In addition, a direction for new generation tools is also set up.

References

1. R.J. Allan, J.M. Brooke and F. Costen and M.westhead, *Grid based high Performance Computing*, Technical Report of the UKHEC Collaboration, June 2000.
2. Z. Balaton, P. Kacsuk, N. Podhorszki and F. Vajda, *Comparison of Representative Grid Monitoring Tools*, Report of the Laboratory of Parallel and Distributed Systems, Computer and Automation Research Institute of the Hungarian Academy of Sciences, 2000.
3. *Dyninst: An Application Program Interface (API) for Runtime Code Generation*, http://www.dyninst.org.
4. I. Foster, C. Kesselman, J. M. Nick, S. Tuecke, *Grid Services for Distributed System Integration*, IEEE Computer, Pages 37-46, June 2002.
5. M. Gerndt et al, *.Perforrmance tools for the Grid: State of the Art and Future- APART White Paper*, http://www.fz-juelich.de/apart.
6. Grid Performance Working Group, Global Grid Forum, *White Paper: A Grid Monitoring Service Architecture*.
7. Ken Kennedy et al, *Toward a Framework for Preparing and Executing Adaptive Grid Programs*, Proceedings of the International Parallel and Distributed Processing Symposium Workshop (IPDPS NGS), IEEE Computer Society Press, April 2002.
8. B. P. Miller et al, *The Paradyn Parallel Performance Measurement Tools*, IEEE Computer 28(11), (November 1995). Special issue on performance evaluation tools for parallel and distributed computer systems.
9. Norbert Padhorszki and Peter Kacsuk, *Presentation and analysis of Grid Performance Data*, Proceedings of EuroPar-2003, Klagenfurt, Austria, 2003.

10. Z. Nemeth, *Performance Evaluation on Grids: Directions, Issues, and Open Problems,* Report of the Laboratory of Parallel and Distributed Systems, Computer and Automation Research Institute of the Hungarian Academy of Sciences, 2002
11. *Paradyn Parallel Performance Tools*. http://www.cs.wisc.edu/paradyn/
12. D. Reed, C. Mendes and S. Whitmore, *GrADS Contract Monitoring*, Pablo Group, University of Illinois, http:// www- pablo. cs. uiuc. edu
13. R. L. Ribler, H. Simitci, D. A. Reed, *The Autopilot Performance-Directed Adaptive Control System*, Future Generation Computer Systems, 18[1], Pages 175-187, September 2001.
14. L. D. Rose and D. Reed, *SvPablo: A Multi-Language Architecture-Independent Performance Analysis System*, Proceedings of the 1999 International Conference on Parallel Processing, p.311, September 1999
15. B.Tierney, *A Grid Monitoring Architecture*, http://www-didc.lbl.gov/GGF-PERF/GMA-WG/papers/GWD-GP-16-2.pdf.
16. B. Tierney, B. Crowley, D. Gunter, M. Holding, J. Lee, M. Thompson, *A Monitoring Sensor Management System for Grid Environments*, Proceedings of the IEEE High Performance Distributed Computing conference, August 2000.
17. Hong-Linh Truong and Thomas Fahringer, *SCALEA: A Performance Analysis Tool for Parallel Programs*. Concurrency and Computation: Practice and Experience, 15(11-12): Pages 1001-1025, 2003.
18. Hong-Linh Truong and Thomas Fahringer, *SCALEA-G: a Unified Monitoring and Performance Analysis System for the Grid*, 2nd European Across Grids Conference, Cyprus, January 2004.