

Model-Based Enhancement of Software Performance for Precision Critical Systems

Naeem Muhammad
DistriNet Research Group,

Department of Computer Science
Katholieke Universiteit Leuven,
Leuven, Belgium

Naeem.Muhammad@cs.kuleuven.be

Nelis Boucke
DistriNet Research Group,

Department of Computer Science
Katholieke Universiteit Leuven,
Leuven, Belgium

Nelis.Boucke@cs.kuleuven.be

Yolande Berbers
DistriNet Research Group,

Department of Computer Science
Katholieke Universiteit Leuven,
Leuven, Belgium

Yolande.Berbers@cs.kuleuven.be

ABSTRACT

Architectural level analysis of a software system for its quality attributes is a proven cost-effective approach. This is particularly significant for performance, which defines multiple aspects of the quality of the system.

In this paper we outline the contribution of a PhD, which provides architecture viewpoint based modeling and analysis support for parallelism and flow latency aspects of the performance, in legacy systems. The main contribution of the PhD includes Parallelism Viewpoint and Flow Latency Viewpoint.

We use the proposed viewpoints to find parallelism and flow latencies specific performance bottlenecks of an industrial case, a precision critical electron microscope software system.

The preliminary results of using Parallelism Viewpoint for our example case show that the viewpoint provides a profound insight into the thread-model of the system, which helps in reducing the excessively used parallelism in the system.

Categories and Subject Descriptors:

C.0 [Computer Systems Organization]: General— System architectures, Systems specification methodology; C.4 [Computer Systems Organization]: Performance Of Systems— Design studies, Modeling techniques; D.2.11 [Software Engineering]: Software Architectures— *Domain-specific architectures, Architecture viewpoint*; D.4.1 [Operating Systems]: Process Management— *Concurrency, multitasking, Threads*; D.4.8 [Operating Systems]: Performance— Modeling and prediction, Operational analysis.

General Terms: Performance, Design.

Keywords: Architecture Viewpoint, Software Performance, Parallelism Viewpoint, Flow Latency Viewpoint.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

"ECSA 2010, August 23-26, 2010, Copenhagen, Denmark.

Copyright (c) 2010 ACM 978-1-4503-0179-4/10/08 ...\$10.00."

1. INTRODUCTION

Quality attributes are characteristics of a system which are used to evaluate the quality of the system. Because quality attributes are often linked with more than one functional component of the system, modelling and analyzing quality attributes is a difficult task. The difficulty increases further with system complexity, which is the case for our problem domain: electron microscope systems. Electron microscopes are highly complex devices whose design combines quite a number of disciplines: (high-voltage) electron physics, electromagnetism and electron optics, image sensors and image processing, mechatronics, instrument conditioning (vacuum, heat), electronics (amplifiers, sensors, embedded control) and software engineering. Quality attributes for the electron microscope precision critical systems are mainly performance, evolvability reliability and accuracy. Precision critical systems, which can be categorized as soft real-time systems require nano-level precision in the measurements of their quality attributes. Modelling, prediction and analysis of these quality attributes at architectural level is a difficult task but very cost effective.

Performance is one of the most important quality attribute as multiple aspects are involved in it. Among these aspects are throughput, efficiency, response time, concurrency and flow latencies. Although various techniques for analyzing the performance at the architectural level are already available (e.g ATAM, SAAM, CBAM) [1] a more precise and realistic analysis can be achieved by utilizing the system implementation, in the case of legacy systems.

The primary aim of this doctoral study is to provide modelling and analysis support for parallelism and flow latencies aspects of the performance quality attribute, by utilizing the implementation of the system. To achieve this aim the following objectives have been set:

1. Selection of suitable modelling languages
2. Development of conceptual architectural models of the system
3. Modelling the parallelism behaviour of the system
4. Modelling system flows and their latencies

5. Provide analysis support for analyzing performance bottlenecks associated with parallelism and system flows
6. Developing tool support for modelling and analyses activities
7. Application of the developed techniques and methods on an industrial case: the FEI electron microscope software

The remainder of this paper is organized as follows. In the next section we outline the proposed approach that we devised to achieve the aforementioned objectives. In section 3 we describe the Parallelism Viewpoint and provide preliminary results of its application on an industrial case. We describe the Flow Latency Viewpoint in section 4. Finally, in section 5 we present a summary of our research work.

2. THE PROPOSED APPROACH

Using architecture views and viewpoints to describe an architecture description of a system is a common approach [1][3]. They describe the architecture of the system for some related concerns of a set of stakeholders. According to ISO 42010 [4]:

“An architecture view is a work product expressing a system’s architecture from the perspective of its concerns.”

“An architecture viewpoint establishes the conventions for constructing, interpreting and analyzing an architecture view addressing concerns framed by the viewpoint.”

A viewpoint essentially includes modelling formalisms, modelling and analysis techniques, and design rules. In this PhD, to provide support for analyzing the issues associated with the parallelism and flow latencies of the system, we develop the Parallelism and Flow Latency architecture viewpoints. Furthermore, we also study the impact of the parallelism overheads on the latencies of the system. The highlighted part in figure 1 shows the contribution and the overall approach of the PhD. The Parallelism and Flow Latency viewpoints are part of the execution view of an architecture description.

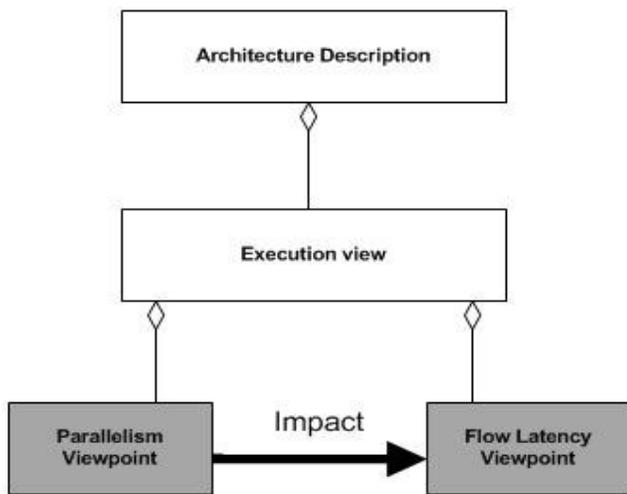


Figure 1 Research contribution and approach

3. PARALLELISM VIEWPOINT

The use of parallelism (multithreading) for software systems is becoming increasingly more important because of the potential benefits it provides. Multithreaded applications are considered to be more efficient because of their better software and hardware resource utilization caused by the parallel execution of tasks. Nowadays most of the widely used hardware and operating systems support multithreaded applications, therefore the use of parallelism has increased substantially. However, overheads and difficulties associated to parallelism amplify dramatically as the number of threads increase. Among them are context switches overhead, incorrect distribution of Read/Write operations and a complex thread management structure. Especially in legacy systems, where design decisions are usually not available, excessive use of multithreading proves to be problematic. Identification and mitigation of these issues in such cases is extremely hard.

One of the goals of this PhD is to provide support for modelling parallelism behaviour of the system to analyze these issues. In this perspective we have proposed an architecture viewpoint called Parallelism Viewpoint.

The Parallelism Viewpoint is a domain-specific form of the concurrency viewpoint which is used to describe the concurrent structure of a system. The concurrency viewpoint mainly provides support for describing concerns related to the communication and synchronization mechanisms of concurrent systems [8]. We extend this support for concurrent systems by describing parallelism related concerns with our viewpoint.

Essentially, a viewpoint must explicitly describe the concerns of a particular domain, identify the stakeholders of these concerns and specify a set of model kinds.

Table 1 contains parallelism specific concerns and their corresponding stakeholders. We find these concerns necessary to identify the overheads associated with context switches, read/write operations and complex thread management. These concerns were identified with the inputs from the stakeholders of the electron microscope software and researchers from the parallelism domain.

Table 1 Parallelism specific concerns and corresponding stakeholders

Concerns	Descriptions	Stakeholders
Time Allocation	Represents the total time used by a thread.	Software architect, Developer, Tester, End User
Task Types	The nature of the tasks performed by threads e.g. file/registry read and write operations	Developer, Tester, System Maintainer
Task Distribution	Number of tasks performed by each thread.	Developer, System Maintainer
Active Time	The time when a thread is using the CPU to perform a task.	Software architect, Tester, System Maintainer

Waiting Time	The time when a thread is waiting for its turn to get the CPU time.	Software architect, Tester, System Maintainer
Execution Elements Management	A way of managing a thread's life cycle e.g. a thread creation and deletion mechanism	Developer, Testers

For the proposed viewpoint we introduced a set of 5 models (Time distribution, Task distribution, Task types, Thread

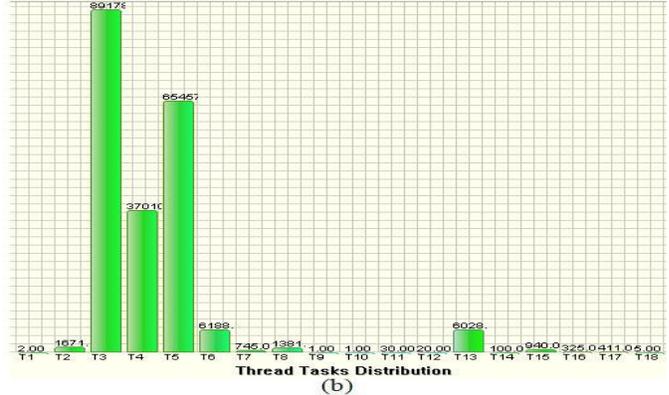
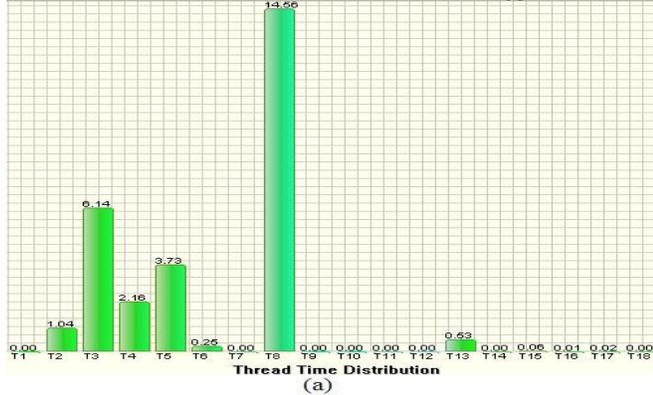


Figure 2 Time and Task distribution model

In this research work we also explicitly identified parallelism specific execution elements to retain focus on parallelism related issues. For this purpose we proposed a parallelism specific metamodel of software execution. A detailed description of the metamodel is given in [6].

Figure 3 shows the methodology for developing models of the Parallelism Viewpoint.

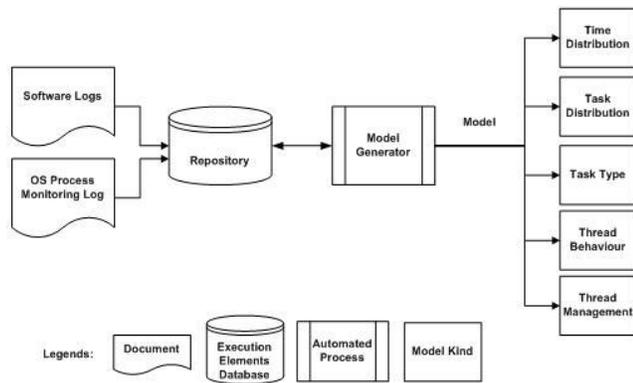


Figure 3 Parallelism Viewpoint model kinds generation

In our approach, we use process logs and logs maintained by the software system under investigation to harvest information and subsequently store it in an architectural repository. We harvest parallelism specific execution information based on the aforementioned metamodel. Afterwards, by using the information in the repository a model generator generates parallelism-specific model kinds. A model kind is a set of notations and conventions which are used to develop domain-specific models for a viewpoint.

In one of the applications of the Parallelism Viewpoint we explored its use to optimize the use of threads in the electron

behaviour and Thread management) to describe these concerns. Figure 2 shows example time and task distribution models of the electron microscope software system. The horizontal axes in the subfigures show threads in the system, whereas vertical bars in figure 2.a represent the total amount of time and in figure 2.b the total number of tasks performed by each thread.

microscope software system [7]. We put forward the use of a thread pool instead of thread-per-job. Based on the time utilization and number of tasks performed characteristics of the thread, we identified threads appropriate to be replaced with a small sized thread pool. In this study we successfully identified 15 of 17 total threads suitable to be replaced with a thread pool of size 2. The total gain is encouraging. We also found that the gain is proportional to the number of threads in the system.

The results of the study indicate that the Parallelism Viewpoint provides a profound insight into the thread structure of the system. Such insight can be used to analyze a system for many threading related concerns.

In this section we briefly discussed the fundamental building blocks of the proposed Parallelism Viewpoint, whereas a comprehensive description covering all aspects of the viewpoint is given in [6].

4. FLOW LATENCY VIEWPOINT

As a part of this PhD research study we are developing a Flow Latency Viewpoint to describe latencies of a flow-intensive system. It includes identifying stakeholders of the system, their flows related concerns and developing a set of models to model and analyze issues with the flow latencies.

Currently, in our research work we have devised an approach for identifying various levels of abstractions in flows by creating connections among them [5]. A flow-intensive system consists of multiple data and control flows. Each flow has a single starting and ending point but may internally consist of many distinct flows. Together these distinct flows compose a single flow at higher abstraction, which is called a composite flow. Such composite flows are most interesting from an architecture abstraction point of view since they often represent more abstract flows of information through the system.

We selected the Architecture Analysis and Design Language (AADL) as a modelling formalism for developing flow models of the system. AADL is one of the best known and most actively used architecture description language for embedded systems [2]. Though AADL is suitable for modelling flows we find its support insufficient for establishing connections among them, which is required in this case to connect distinct flows. We extended AADL to provide such support and provided tool support to analyze latencies of the composite flows of the system. In our solution we connected adjacent flows by introducing a new AADL property called Link_Flow. The new property holds a string value representing the identifier of the flow that is to be linked. Subsequently, this value is manipulated by the tool we developed, to analyze the latencies of the composite flows.

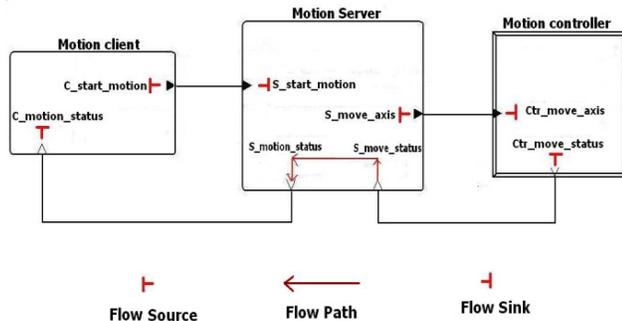


Figure 4 Flows in motion-subsystem

We successfully applied the proposed solution to an industrial case of a motion-subsystem, a subsystem of the electron microscope software. Figure 4 shows three components of the motion-subsystem and three distinct flows flowing in them, starting with a flow source and ending with a flow sink. These distinct flows contribute to a single composite flow. In the case of our example system these flows contribute to a single flow of a command send by the client to move the specimen in the electron microscope. Our proposed solution will enable system architects to model and analyse system flows at different abstraction levels. We will extend this research work to develop the Flow Latency Viewpoint.

5. SUMMARY

The goal of this PhD is to develop state of the art methods and techniques for modelling and analyzing the performance of the system at architectural level by utilizing its implementation. The focus of this research is on parallelism and flow latencies aspects of the performance.

To provide support for modelling parallelism behaviour of the system to analyze associated overheads we have proposed an architecture viewpoint called Parallelism Viewpoint. Results show that the Parallelism Viewpoint provides a profound insight into the parallelism behaviour of the system, required to identify and mitigate the associated performance overheads.

Currently, we are developing the Flow Latency Viewpoint to analyze issues with the latencies of the system flows. The viewpoint will explicitly identify the flows related concerns, associated stakeholders and a set of models to model these concerns.

The Parallelism Viewpoint and Flow Latency Viewpoint will be a part of an architecture description of the electron microscope software. Another goal of this PhD is to trace links between the elements of both viewpoints. In particular, we are interested in utilizing the outcome of the Parallelism Viewpoint to understand the effects of the parallelism overheads on flow latencies of the system.

6. ACKNOWLEDGMENTS

This work has been carried out as a part of the Condor project (<http://www.esi.nl/Projects->Condor>) at FEI company under the responsibilities of the Embedded Systems Institute (ESI). This project is partially supported by the Dutch Ministry of Economic Affairs under the BSIK program.

We would also like to thank our colleagues Sjjr van Loo and Auke van Balen for their valuable inputs and feedback for this research work.

7. REFERENCES

- [1] Clements, P., Kazman, R., Klein, M. 2002. *Evaluating Software Architecture: methods and case studies*. Addison-Wesley Longman Publishing Co., Inc.
- [2] Feiler, P.H.; Lewis, B.A.; Vestal, S. 2006. The SAE Architecture Analysis & Design Language (AADL) A Standard for Engineering Performance Critical Systems. In *Proceedings of the IEEE International Symposium on Computer-Aided Control Systems Design*. 1206-1211.
- [3] Hofmeister, C., Nord, R., and Soni, D. 2000. *Applied Software Architecture*. Addison-Wesley Longman Publishing Co., Inc.
- [4] ISO/IEC 42010. 2007. *Systems and Software Engineering -- Recommended Practice For Architectural Description Of Software-Intensive Systems*.
- [5] Naem Muhammad, Yves Vandewoude, Yolande Berbers, Sjjr van Loo. 2009. Modelling composite end-to-end flows with AADL. In *Proceedings of the workshop on the definition, evaluation, and exploitation of modelling and computing standards for Real-Time Embedded Systems* (Dublin, Ireland. July 1-3, 2009)
- [6] Naem Muhammad, Nelis Boucke, Yolande Berbers. 2010. *Parallelism Viewpoint: An Architecture Viewpoint to Model Parallelism Behaviour of Parallelism-Intensive Software Systems*. Technical Report. Report Number: CW589. Department of Computer Science, KULeuven, Leuven, Belgium. <http://www.cs.kuleuven.be/publicaties/rapporten/cw/CW589.abs.html>
- [7] Naem Muhammad, Nelis Boucke, Yolande Berbers. 2010. Using the Parallelism Viewpoint to Optimize the Use of Threads in Parallelism-Intensive Software Systems. In *Proceedings of the ICSCCT Conference on Software and Computing Technology* (Kunming, China. October 18-19, 2010).
- [8] Rozanski, N. and Woods, E. 2005. *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives*. Addison-Wesley Professional