

Programming Models for Grid Applications and Systems

Extended Abstract

Thilo Kielmann

Vrije Universiteit, Amsterdam, The Netherlands
kielmann@cs.vu.nl

Keywords: GAT, Ibis, SAGA, Satin

1 Introduction

History repeats itself. Since the invention of the programmable computer, numerous computer scientists keep dedicating their professional lives to the design of “the single, best” programming model, whereas programmers “vote” by choosing their favourite languages and tools.

Interestingly, these choices have always been guided by non-functional properties. For programming a single computer, the most widely used models have become object-oriented and component-based programming, a choice driven by their high abstraction level, leading to high programmer productivity. For parallel computers, the winner turned out to be message passing, providing by far not the highest-possible abstraction level, but the closest match between machine architecture and programming model, leading to efficient program execution.

For grids, the race is still open. Here, additional non-functional properties like fault-tolerance, security, and platform independence enter the scene. In this work, we explore the scope of grid programming problems and argue for a palette of programming abstractions, each suitable for its respective problem domain.

2 Grids

Grid computing had once been introduced via the analogy to the electrical power grid: the idea of “plug and run” suggested that grids could execute user programs on some unknown, remote resource, without any further user intervention or effort. Reality turned out to be different. Users typically have to spend lots of efforts to make their applications suitable for grids.

Grids turned out to be highly complex environments, with frequent variations of resource availability and operativeness. Due to the combination of multiple, independent administrative domains, along with different security policies and deployed middleware,

writing and deployment of grid applications turned out to be a challenging task.

To address this problem, many grid programming environments have been introduced, the authors of many, if not all of them, are claiming that their tool provides the single, right way of writing grid applications.^{1 2}

This is an extended abstract of a full paper, published in [4]. In Section 3, we list typical application requirements. In Section 4, we map the different types of grid applications and tools to appropriate programming environments.

3 Requirements of grid applications

Figure 1 shows the perspective of the application developer. We distinguish required functional properties (“What applications need to do.”) from non-functional properties (“What else needs to be taken care of.”).

3.1 What applications need to do

- Job submission, spawning, and scheduling
- Access to file and data resources
- Inter process communication
- Application monitoring and steering

3.2 What else needs to be taken care of

- Performance
- Fault tolerance
- Security and trust
- Platform independence

¹This, of course, also includes the author.

²However, these claims typically need to be taken with a “grain of salt.”

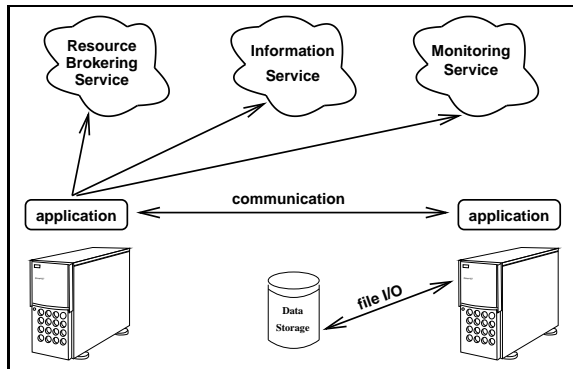


Figure 1: The application developer's view on an application and its interaction with the grid environment.

4 Types of grid programs and their programming environments

Different kinds of grid applications require different programming models, interfaces, and environments. We discuss possible application types in top-down order.

- Legacy codes
are such applications that have been written without grids in mind and which can not be modified to become grid-aware or at least parallelized by a grid-enabled environment. Running such codes in a grid environment requires virtualization of the machine for which the codes once had been written, as, e.g., proposed in [2].
- Parallel applications
are written in order to achieve fast execution by using many compute resources together. For running parallel applications in grids, they need to be programmed by explicitly using a grid-enabled, parallel programming environment. Prominent examples of such environments are MPI, ProActive [5], or Satin [6].
- Grid-aware codes
are those application programs that are programmed (or re-programmed) to use grid resources explicitly, like spawning off jobs to other machines or using remote files and data bases. These programs are best written using a simplified API like SAGA [3] that restricts the programming environment to application needs, abstracting away such features needed to monitor or manage services and resources.
- Support tools
More experienced application developers frequently start writing their own application support tools. As such tools are operating across

multiple services and resources, they need to abstract from the individual entities in a grid. They still need detailed control, for example of security policies or performance information. These tools are thus best programmed using a service and resource abstraction layer like the GAT [1].

- Service and resource management
Software for management and administration of resources and services needs full control over status and features of the grid middleware. This type of software will best be programmed directly using the respective service API's.

5 Conclusions

Programming for grids is hard. There is no single, "one size fits all" programming model for grid software. Different problems require different approaches in order to cope with non-functional aspects of programming that in fact determine the right choice of tools. There will always be some trade-off between programmer productivity (by good abstractions) and program efficiency or resilience to failures and changing environments. What is the best grid programming model? "It depends!"

Acknowledgments

This work is partially supported by the *CoreGRID* Network of Excellence, funded by the European Commission's FP6 programme (contract IST-2002-004265). The author would like to thank his many colleagues from the network for fruitful and stimulating discussions.

References

- [1] G. Allen, K. Davis, T. Goodale, A. Hutanu, H. Kaiser, T. Kielmann, A. Merzky, R. van Nieuwpoort, A. Reinefeld, F. Schintke, T. Schütt, E. Seidel, and B. Ullmer. The Grid Application Toolkit: Towards Generic and Easy Application Programming Interfaces for the Grid. *Proceedings of the IEEE*, 93(3):534–550, 2005.
- [2] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the Art of Virtualization. In *ACM SOSP*, pages 164–177, 2003.
- [3] T. Goodale, S. Jha, H. Kaiser, T. Kielmann, P. Kleijer, A. Merzky, J. Shalf, and C. Smith. A Simple API for Grid Applications (SAGA). Grid Forum Document, GWD-R.90, 2007. Open Grid Forum.
- [4] T. Kielmann. Programming Models for Grid Applications and Systems: Requirements and Approaches. In *John Vincent Atanasoff International Symposium on Modern Computing (JVA 2006)*, pages 27–32, Sofia, Bulgaria, 2006. IEEE.
- [5] ProActive. <http://www.inria.fr/oasis/ProActive>.
- [6] R. V. van Nieuwpoort, T. Kielmann, and H. E. Bal. Efficient Load Balancing for Wide-area Divide-and-Conquer Applications. In *Proc. PPOPP '01: ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 34–43, 2001.