Designing an Architecture for Distributed Shared Data on the Grid

Dacian Tudor[1] , Vladimir Cretu[1] and Wolfgang Schreiner[2]

[1] "Politehnica" University of Timisoara, Computer Science and Engineering Department,
Vasile Parvan Street, No. 2, 300223, Timisoara, Romania
{dacian, vretu} @ cs.upt.ro
[2] Research Institute for Symbolic Computation (RISC)
Johannes Kepler University, 4040 Linz, Austria
Wolfgang.Schreiner@risc.uni-linz.ac.at

Abstract. Despite the continuous advances of the last years in grid computing, the grid computing programming paradigms are dominated by the message passing concept. There is little support for other paradigms such as shared data or associative programming. In this paper we analyze some of the existing solutions for grid shared data programming and highlight some of their drawbacks. We propose a new architecture and its core features as well as new evaluation means of its behavior in various scenarios including the next generation grid systems. In addition to the simplicity of our solution, we believe that it would allow us to easily apply further extensions.

Keywords: grid computing, distributed shared data, programming model.

1 Introduction

Although the number of networked machines has been constantly increased, the number of new distributed applications is still much lower. Some of the core issues that are faced by distributed applications are due to latencies, synchronization and partial failures. Only on dedicated grids ideal conditions can hold during the entire application lifetime. Next, the increasing heterogeneity and the greater difficulty to replace large spread legacy systems impose an important break on grid application development. One of the answers we believe to these challenges is in the grid programming model and more specifically in grid shared data programming.

In the grid landscape, there are very few solutions for large scale data sharing models. Solutions like the LOTS system [1], SMG [2] or Teamster-G [3] that addresses the shared memory problem at the grid level does not provide important information like detailed design, replication policies, mutual exclusion handling, and memory consistency specification. Besides the missing information, there is little evidence of their suitability or behavior in large scale grid computing. A new direction towards dependable distributed computing systems that aim to improve both data and service availability is aimed by Dedisys [4], which appears to focus on availability and fault tolerance at the system level rather than performance. Most advanced solution for grid shared data programming that came to our knowledge is JuxMem [5], which is based on peer-to-peer middleware. One of its main drawbacks is the fixed replication scheme that bounds data replicas at creation time or when fault occur, and which does not consider the system dynamics such as data usage patterns.

We have noticed that there are few shared memory systems designed for the grid. Many of these systems were tested in particular environments that represent ideal scenarios of fast connected machines most of the times being grouped as high performance clusters. In search for a better approach, we aim to investigate the problem of distributed shared memory for grid systems and provide a system specification that addresses the following main points we found missing in most of the existing solutions:

(1) **Large scale system over large latency connections**, which are dominant between machines located at large distances.
(2) **Relaxed consistency and type coherence,** as we expect that relaxed consistency does not carry sufficient information on data usage.
(3) **Object oriented architecture,** as the most appealing concept for grid application programmers.
(4) **Quantifiable system validation and verification**, through formal methods as a proof of concept for the system model.

2 Abstract Model

Some of the previous attempts in designing distributed shared memory systems for the grid used logical mappings over one single large machine group. We believe that another split is necessary. We see this mapping as part of the system deployment, rather than a predefined mapping. In order to address thousands of nodes, we decompose the system into a federation of groups of abstract machines called universes. A universe is a logical collection of machine nodes which provides a hosting environment for distributed objects. Nodes are homogeneous and have a data storage capacity in memory and code execution capabilities. Each node can hold a certain number of objects so that the sum of all object weights held by the node shall not exceed the node's capacity. All existing universes form together the Grid Universe. Each universe is a continuously evolving entity together with its connections to the other universes. A universe groups together more physical machines which share the same communication paths, thus communication channels within universes are homogeneous and have known and constant characteristics. Communication between universes is unpredictable, unknown and dynamic.

We propose an object oriented model which provides interfaces for data encapsulation and a natural and convenient way to abstract data sharing objects. It supports the idea of objects residing in architectural different run-time systems like nodes in universes. The grid universe acts as a container for grid objects and provides means to create delete and locate grid objects based on a unique object identifier. The users do not operate directly on objects, but rather on object references. A grid object reference is a handle to a concrete grid object that provides the same interface as the object provides. A Grid Object has two identifiers associated with: the GID, which is associated by the system, and the OID which is given by the creator as a human friendly identifier. The OID is used to lookup a certain grid object. In order to decrease access time to grid objects from different universes, we make use of data replication concepts. If some configurable system conditions are satisfied, a grid object is replicated to other universe nodes, assuming that object state can be transferred from one process to another across a communication path.

3 System Architecture Selection

The main issue that our architecture needs to address is the problem of realizing mutual exclusion. We have chosen entry consistency as replica consistency specification for our system and we evaluated several possible solutions to realize the abstract model. In the following table, we summarize the characteristics of each of the four remaining candidates, where we highlight the negative characteristics of each solution by marking them in italic style.

Table 1. Solution Selection Criteria

Criteria	Centralized/N-T	Martin/N-T	Suzuki-Kasami/N-T	Grid N-T
Universe Scalability	High	High	High	High
Local scalability	*Low/Medium*	*Low*	Medium	High
Local obtaining time	*Low/Medium*	Medium/High	Medium	Low
Local resource demand	Low	*Medium*	*High*	*Low/Medium*
Independent processing	High	*Low*	*Low*	*Low*
Complexity	Low	*Medium*	*Medium*	*High*
Local dynamics	High	*Low/Medium*	*Low/Medium*	Low

. The first solution is our main contribution and refers to a centralized algorithm inside each universe and a multi-token algorithm between universes derived from the Naimi-Trehel [6] algorithm that was adapted to satisfy entry consistency. The core motivation for this choice is its high local dynamics, higher capability to perform independently, a low resource demand and low complexity. We have traded the local scalability for all other characteristics as we believe that universes will have a limited number of nodes for typical deployment scenarios. The next two considered solutions are the compositional approaches described [7]. These are similar to the previous solution, the only difference is the mutual exclusion algorithm applied inside a universe. The last choice is the adapted Naimi-Trehel algorithm described in [8] which is applied on the grid scope. This solution requires a gateway node in each cluster in order to keep track if the token is held remotely or not. From this point of view, this design approach resembles our proposal. Based on the measurements of [8], it appears that the Naimi-Trehel algorithm is the most suitable algorithm between universes and it provides a reasonable trade-off between different classes or applications (highly parallel vs. low parallel applications) which supports the idea of our architecture proposal.

Conclusions

In this paper we have highlighted the problem of shared data programming on the grid and have pointed out that there is little research in this direction. We have introduced an abstract programming model that transparently defines the grid shared data items as grid shared objects. In addition to the relaxed entry consistency semantics, we consider different object types in order to exploit different synchronization schemes and reduce communications costs. We have proposed a mutual exclusion algorithm based on the Naimi-Trehel algorithm that is easy to adapt and extend in order to accommodate different interaction patterns.

A model of the presented system is currently being developed in order to be simulated and verified using a probabilistic model checker that would provide quantifiable results on the behavior of our system in various conditions. At the same time, a prototype implementation is being developed in order to confirm the findings through model verification and simulation.

References

1. Cheung, B.W.L. Cho-Li Wang Lau, F.C.M. LOTS: a software DSM supporting large object space, IEEE International Conference on Cluster Computing, Pp. 225-234, ISBN: 0-7803-8694-9, 2004
2. J.P. Ryan, B.A. Coghlan, SMG: Shared memory for Grids, In: Proceedings of 6th IASTED International Conference on Parallel and Distributed Computing and Systems. 2004, pp. 439-451. http://www.cs.tcd.ie/coghlan/pubs/pdcs04-06072004-v1.pdf
3. Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, San Francisco (1999)
4. J. Osrael, L. Froihofer, and K.M. Goeschka. A Replication Model for Trading Data Integrity against Availabilit, The 12th Int. Symp. on Pacific Rim Dependable Computing (PRDC'2006), IEEE CS Press, 2006
5. Antoniu, Gabriel and Bougé, Luc and Jan, Mathieu. JuxMem: An Adaptive Supportive Platform for Data Sharing on the Grid. Scalable Computing: Practice and Experience, Volume 6, Pp. 45-55, September 2005
6. M. Naimi, M. Trehel, and A. Arnold. A log (N) distributed mutual exclusion algorithm based on path reversal. JPDC, 34(1) : 1–13, 1996
7. Julien Sopena, Fabrice Legond-Aubry, Luciana Arantes, Pierre Sens. A Composition Approach to Mutual Exclusion Algorithms for Grid Applications. Proceedings of the 2007 International Conference on Parallel Processing (ICPP 2007), Volume 00, Page: 65, ISBN 0-7695-2933-X, 2007
8. Bertier, M.; Arantes, L.; Sens, P. Hierarchical token based mutual exclusion algorithms. IEEE International Symposium on Cluster Computing and the Grid, 2004. CCGrid 2004, ISBN: 0-7803-8430-x, Page 539- 546, 19-22 April 2004.