Nomadic Migration: A New Tool for Dynamic Grid Computing
Gerd Lanfermann, Gabrielle Allen, Thomas Radke, Edward Seidel

Abstract—We describe the design and implementation of a technology which provides an application with the ability to seek out and exploit remote computing resources by migrating tasks from site to site, dynamically adapting the application to a changing Grid environment. The motivation for this migration framework, dubbed "The Worm," originated from the experience of having an abundance of computing time for simulations, which is distributed over multiple sites and split in time chunks by queuing systems. We describe the architecture of the Worm, describing how new or more suitable resources are located, and how the payload simulation is migrated to these resources following a trigger event. The migration technology presented here is designed to be used for any application, including large-scale HPC simulations.

I. INTRODUCTION

GRID computing involves utilizing computational resources, connected by networks, as needed to solve problems. Recent advances in Grid computing are such that applications are now in a position to begin to exploit a wide range of available computer resources, simultaneously, sequentially, or both, enabling many different new and innovative Grid usage scenarios.

Adding up the theoretically available computing time across a pool of standard computers, such as idle workstations, or summing the total computing time granted to a research group by several independent supercomputing sites will typically yield an impressive capacity of processing power. But these resources are neither available on a homogenous architecture base nor are they all continuously accessible over a long period of time.

Here we have focussed on a new type of Grid computing appropriate for its dynamic character: self-determined migration of a simulation from one site or collection of sites to any other. We present a migration technology, dubbed "the Worm," designed for parallel applications with high IO and memory requirements, driven by the need for performing large scale simulations on HPC machines. The technology is also applicable for the efficient use of idle cycles from small machine pools.

II. PROTOTYPE EXPERIENCES

A prototype implementation of the Worm was already demonstrated at Supercomputing 2000, running across the machines of the EGrid Testbed. The Worm was implemented in the Cactus programming framework, designed for parallel applications with high IO and memory requirements, driven by the need for performing large scale simulations on HPC machines. The technology is also applicable for the efficient use of idle cycles from small machine pools.

III. WORM TECHNOLOGY IN A DYNAMIC GRID ENVIRONMENT

Based on the early prototype experiences, a more sophisticated Worm framework was designed to provide the user with range of migration policies and to overcome numerous technical challenges when dealing with heterogeneous grid environments. With the new worm technology, application migration can be initiated by a variety of cases, which range from the user's manual triggering of a migration event to fully automatic application relocation: by monitoring the simulation performance and profiling the current hardware with small benchmarking programs a detailed profile of the current execution environment is generated.

Fig. 1. We show the main components of a resource aware, self-replicating Worm. The user payload is encapsulated by the Worm Kernel, acting as a contact point to the resource detector and various application information servers (AIS). The transfer units stage executables to machines and provides storage for checkpoint files during hibernation.

The participating EGrid sites published characteristic system profiles and load information to a central Resource Manager. While the prototype Worm simulation was running on machines in the Grid, it was able to query this central resource service, seeking available machines of a certain configuration. Using this information it could then migrate to another site according to some predefined criteria.
storage", "less queue waiting" or "better network".

Since the Worm migrates between machines in a non-predictable fashion, reacting to the dynamic nature of a Grid, it requires a mechanism for tracking the current and past locations of the Worm. This is handled with different degrees of finesse, for example, by publishing the information to a centralized Application Information Service (AIS) or an email/SMS notification server.

Before migrating, the Worm must have located the next resource by querying a remote Resource Broker (RB), which tracks available computing resources, obtaining load and other information from registered sites. Different RB formats developed by e.g. GrADS [8] and groups within the EGrid are understood. If a suitable resource cannot be found, the simulation hibernates by writing a checkpoint which is stored until appropriate machines become available to host the restarted simulation.

The Worm must provide the capability to access the different sites without user interaction to copy checkpoint and parameter files, start processes and handle output data. It is essential to provide a secure but easy way to interface these resources. Our Worm supports methods as Globus GSI technology [6] or more simply secure shell and secure copy.

The Worm application (including the user provided payload simulation) must be available on the different heterogeneous machines of the user's virtual grid. We support repositories of pre-built binaries and automated compiling "on-the-fly" before execution. To restore the simulation state in a heterogeneous machine environment, the checkpoint files are coded in an architecture independent format. We use HDF5 [11] and the CactusCode framework [1] to meet these requirements.

IV. CONCLUSIONS

Automating and optimizing the usage of multiple resources is an essential challenge to Grid-enabling application software. We have described a technology that enables an application on its own to seek out and exploit computational resources on the Grid. This "Worm" approach not only provides applications with the ability to make decisions about resource usage and to self-migrate to new machines if necessary, it also takes into account the heterogeneous nature of resources as well as their dynamic availability in time.

Note that although we have spoken in terms of migrating an entire application from site to site, future Grid applications will be able to take advantage of work flow parallelism: e.g. analysis tasks may be spawned off to other grid resources. The Worm technology paves the ground for these advanced and intelligent Grid applications. There are many possible uses that will be discussed elsewhere [10].

ACKNOWLEDGMENTS

The development of the EGrid Worm is a highly collaborative effort, and we are indebted to a great many experts at different institutions, especially on the EGrid for their advice and support. It is a pleasure for us to thank, above all Tom Goodale and John Shalf, as well as Ian Foster, Sridhar Gullapalli, Steve Fitzgerald and the Globus team at ANL for their Globus and Data Grid work; Mike Folk and his HDF5 development group at NCSA; Computing resources and technical support have been provided by the EGrid, AEI, NCSA, ANL, and ZIB. We have also benefited from close association with and partial support of the ASC project, NSF PHY-9979985.

REFERENCES


Fig. 2. The timeline of migration events between three sites is shown. The first two relocations involve hibernation of the application. In the third case an advanced reservation scheduler is used to request overlapping resources, which allows to directly stream checkpoints.

430