A Computational Infrastructure for Grid-based Asynchronous Parallel Applications

Zhen Li and Manish Parashar
Electrical and Computer Engineering Department Rutgers University
94 Brett Road Piscataway, NJ 08854

E-mail: {zhljenny,parashar}@caip.rutgers.edu

1. Introduction

Grid computing, based on the aggregation of large numbers of independent hardware, software and information resources spanning multiple organizations, is rapidly emerging as the dominant paradigm for distributed problem solving for a wide range of application domains. Complementary to Grid virtual organizations, Desktop Grids [1] leverage Internet connected computers to support large computations. Desktop Grid systems have been successfully used to address large applications in science and engineering with significant computational requirements, including global climate predication (Climatprediction.net), protein structure prediction (Predictor@Home), search for extraterrestrial intelligence (SETI@Home), gravitational wave detection (Einstein@Home), and cosmic rays study (XtremWeb). While the successes of the above applications do demonstrate the potential of Desktop Grids, current implementations are limited to *embarrassingly parallel* applications based on the Bag-Of-Task paradigm, where the individual tasks are independent and do not require inter-task communications. As a result, these implementations cannot support more general scientific and engineering applications, such as those based on parallel iterative computations and replica exchange simulations, as the parallel formulations of these applications require synchronization and inter-task communications. Parallel asynchronous formulations of computation algorithms relax synchronization and communication requirements, and can tolerate heterogeneous computation powers and unreliable communication channels. These formulations have been proposed to extend Desktop Grids beyond *embarrassingly parallel* applications and support parallel applications, such as computing the lowest eigenvalue and eigenvector of stochastic matrices for Google pageranks and solving linear systems.

Asynchronous parallel applications tend to be computationally expensive and can benefit from the large numbers of processors and computational capacities offered by parallel and distributed computing systems and especially Desktop Grid environments. However, existing implementations either target tightly coupled parallel systems or relatively small homogenous clusters. General formulations of these applications require complex coordination and communication patterns. Coupled with the complexity of the Grid environment, including its scale, its heterogeneity in computational, storage and communication capabilities, its dynamism and its unreliability, Grid-based parallel asynchronous computation presents significant challenges.

Clearly, the complexity of developing Grid-based asynchronous parallel applications must be abstracted from the application scientists/engineers and effectively addressed by a computational infrastructure. Such an infrastructure should support dynamic computation task management and efficient, robust and scalable inter-task communications enable large scale application/system size. This paper presents CometG, a decentralized computational infrastructure for Desktop Grid environments. It provides the abstractions and mechanisms required by asynchronous parallel applications, including mechanisms for dynamic and anonymous task distribution, task coordination and execution, decoupled communication and data exchange.

2. CometG Computational Infrastructure

CometG provides a global virtual shared space abstraction that can be associatively accessed by all system entities without knowledge of the physical locations of the hosts over which the space is distributed. The architecture of CometG, shown in Figure 1, consists of 3 key layers. The communication layer provides scalable content-based messaging services as well as channels for direct communication, and manages system heterogeneity and dynamism. This layer guarantees that content-based messages, specified using flexible content descriptors, are served with bounded cost. The components of this layer include a content-based routing engine and a 1-dimensional structured self-organizing overlay. The routing engine [3] supports flexible content-based routing and complex querying using partial keywords, wildcards, or ranges. It also guarantees that all peer nodes with data elements that match a query/message will be located. The overlay is composed of peer nodes, which may be any node in the Desktop Grid system (e.g., end-user computers, servers, or message relay nodes). The coordination layer provides Linda-like [2] primitives and sup-

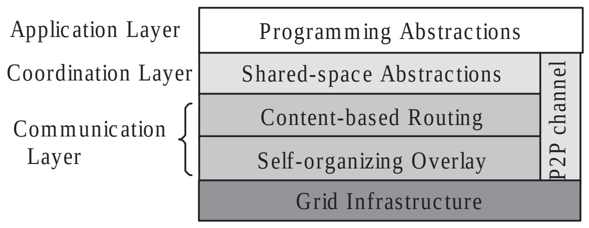

Figure 1. A conceptual overview of the CometG infrastructure.

ports the tuple space coordination model. The components of this layer include a tuple repository, matching engine, and message dispatcher. The application layer provides abstractions required by asynchronous computations, which are described below.

The application layer of CometG provides coordination space abstractions and programming modules to support master-worker parallel formulations of asynchronous computations. Specifically, two coordination spaces, *TaskSpace* and *BorderSpace*, are provided. *TaskSpace* stores task tuples representing application tasks. *BorderSpace* allows the workers exchanging data tuples between tasks. The programming modules include master and worker modules. A master module is responsible for partitioning application data, generating tasks, collecting results, and terminating the application when it completes. A worker module contains the application-specific computational component associated with a retrieved task. Workers use the tuple space abstractions to retrieve tasks and exchange borders.

CometG supports large application/system scales using multiple coordination groups. A coordination group includes application specified coordination spaces, and groups of masters and workers. A group can support multiple applications with logically separate shared spaces. An application can span multiple groups, each of which would handle a part of the application. An application is hierarchically partitioned, first across coordination groups, and then across masters within each coordination group. Task with communication dependencies should be mapped to the same coordination group if possible, as communications across groups can be expensive. Workers within a coordination group communicate using the shared *BorderSpace*. Masters within and across coordination groups communicate using direct communication channels.

CometG provides application level fault tolerance mechanisms to address the unreliability inherent in Grid environments. These mechanisms assume a fail-stop failure model and timed communication behavior. Under these assumptions, possible failures include inter task communication failure, worker failure, master failure, and task loss. Inter task communication failures can be simply handled using timeout, due to the resilient nature of asynchronous algorithms. Master failures are handled using checkpoint-restart. The runtime system periodically checkpoints the local state of each master, including its task table and intermediate results, to a stable storage. Users are currently responsible for detecting the failure of a master node and can recover its state from the stable storage and resume the computation. Finally, worker failures and task loss are handled using timeout-regeneration, a retrieval-submission protocol, and a garbage-collection mechanism [4].

Two prototype applications have been built using CometG: (1) parallel asynchronous iterative computations and (2) asynchronous formulation of replica exchange simulations. Parallel asynchronous formulations of iterative algorithms relax synchronization and communication requirements, and can tolerate heterogeneous computation powers and unreliable communication channels. Potential applications of parallel asynchronous iterative computation span a range of scientific and engineering disciplines, such as solving partial differential equation (PDE), high-performance linear algebra, and optimization problems. A PDE application has been developed using CometG and deployed on PlanetLab using more than 200 machines [4]. Replica exchange is an effective sampling algorithm that has been proposed in various disciplines, such as bimolecular simulations where it allows for efficient crossing of high energy barriers that separate thermodynamically stable states. CometG extends the asynchronous formulation of replica exchange to Desktop Grid environments. CometG-based replica exchange allows computation units on different nodes to negotiate and perform exchanges in a decoupled, dynamic and asynchronous manner, which effectively addresses environment dynamism and improves simulation efficiency.

3. Conclusion and Future Work

This paper presented the CometG decentralized computational infrastructure that extends Desktop Grid environments to robustly support asynchronous parallel computations. CometG provides scalable communication/coordination mechanisms and programming abstractions to support parallel asynchronous iterative computations and replica exchange simulations. The current CometG system can effectively support hundreds of workers. Its scalability can be further improved to thousands or even millions of workers using two enhancements: (1) separating the space nodes from end nodes, where the space nodes provide coordination services and the end nodes host the workers; (2) employing relatively powerful peers, i.e., super-peers, with larger memory capacity and network bandwidth, as space nodes. These enhancements are currently being explored.

4. Acknowledgments

The research presented in this paper is supported in part by National Science Foundation via grants numbers CNS 0305495, CNS 0426354, IIS 0430826 and ANI 0335244, and by Department of Energy via the grant number DE-FG02-06ER54857.

5. References

[1] D. Kondo, M. Taufer, C. Brooks, H. Casanova, and A. Chien, Characterizing and Evaluating Desktop Grids: An Empirical Study, In *Proceedings of the IPDPS*, 2004.

[2] N. Carriero and D. Gelernter, Linda in context, *Communications of the ACM*, 32(4):444-458, 1989.

[3] C. Schmidt and M. Parashar, Enabling Flexible Queries with Guarantees in P2P Systems, *IEEE Internet Computing, Special issue on Information Dissemination on the Web*, 8(3), June 2004.

[4] Z. Li, A scalable, Decentralized Coordination Infrastructure for Grid Environments, Ph.D. Thesis, Rutgers University, May 2007.