

# Languages, Compilers, and Run-Time Systems

Kai Zhang

CMSC818S

March 16<sup>th</sup>, 18<sup>th</sup>, 2004

# Introduction

- ◆ Chapter 25 of the Grid 2: Languages, Compilers, and Run-Time Systems
  - Ken Kennedy
- ◆ The GrADS Project: Software Support for High-Level Grid Application Development
  - F. Berman, A. Chien, K. Cooper
- ◆ Programming the Grid: Distributed Software Components, P2P and Grid Web Services for Scientific Application
  - Dennis Gannon, Randall Bramley, Geoffrey Fox

# Grid Computing Challenges

- ◆ Grids is significantly more complex than homogeneous parallel computers
  - Heterogeneous, variable latencies and varying bandwidths
- ◆ Decomposition of responsibilities b/w developer and system
  - Developer concentrates on problem itself at high level of abstraction
  - The system, including programming language and compiler, handles mapping the abstraction onto the available Grid resource at any given moment
  - Developer and system together produce a correct and efficient program

# Grid Computing Challenges(2)

- ◆ Goal of application development support software:
  - Easy for programmer to develop Grid app.
  - Portable to different computing configurations.
  - Achieve performance close to using underlying feature of network directly
- ◆ Principal problems for the language, compiler and library:
  - Balancing the load across a heterogeneous configuration – minimize running time, match communication to latencies and bandwidths.
  - Ensure that performance variability remains within certain bounds.

# Lessons From Parallel Computation

- ◆ Automatic Parallelization will not by itself solve the parallel programming problem:
  - The complexity of the automatic parallelization techniques
  - Long compiler running time
  - Small number of successful demonstrations
- ◆ Explicit Communication
  - PVM, MPI, Grid-enabled MPICH-G
  - Explicit communication can be thought of as an assembly language for grids.

# Lessons from Parallel Computation (2)

- ◆ Data-Parallel Language
  - Dividing data domain and assigning subdomains to diff processors
  - Maximizing locality and minimizing communication.
  - Drawback for Grid computing: not allow dynamic load balancing
- ◆ Task-Parallel Language and System Support
  - Components which represent diff functions run in parallel
  - Task-Parallel is well suited to Grid because diff tasks can be allocated to diff Grid nodes
  - Challenge for coarse-grained s/w integration: preventing perf degradation caused by sequential processing of tasks.

# Lessons from Parallel Computation (3)

## ◆ Load Balancing

- Spreading the calculations evenly across processors while minimizing communications.
- Becomes more difficult in the Grid: power of each node must be taken into account.

## ◆ Latency Tolerance and Management

- Compiler approach: latency hiding – communication overlapped with computation, latency reduction – better reuse local data by reorganizing program
- More complex to implement in compiler for the Grid
- More time be spent on estimating running time and communication delay

# Lessons from Parallel Computation (4)

## ◆ Run-Time Compilation

- Compiling after upper bound or array size are known can give better load-balancing
- Inspector/executor method: inspector is executed once to establish a plan according to run-time data, and executor carry out the plan.
- Powerful tool for tailoring a program for heterogeneous computing.

## ◆ Library Encapsulation

- Encapsulate parallelism in libraries
  - Function library: parallelized standard function, more control to program
  - Data structure library: parallel data structure, maximum flexibility to library builder
- Collaboration b/w compiler and library might be useful

# Programming Tools

- ◆ Programming tools for parallel computation
  - Problem
    - Hand programming provides great flexibility but is burdensome on the programmer.
    - Compiler and library approaches don't offer enough flexibility to identify perf bottleneck and overcome them.
  - Programming tool approach
    - Programming support tools can identify perf. bottlenecks (e.g. Jumpshot, Pablo) and help restructure program (e.g. Parascope).
    - A promising approach is to employ tight collaboration b/w tools and compiler.
      - ◆ Example: HPF compiler provides info on program transformations to Pablo which maps the discovered perf problem back to original source, and then reconstructing tool carries out source-level transformation.

# Programming Tools (2)

- ◆ Programming tools for the Grid today
  - MPICH-G and other tools make Grid programming possible but not make it easy enough.
  - Hard work as resource selection, communication management and adapting to varying loads is left to the developer.
  - Workflow systems as Condor are not designed to deal with more tightly coupled applications.

# The GrADS Project

Software Support for High-Level Grid  
Application Development

F. Berman, A. Chien, K. Cooper

# Goal of the GrADS Project

Grid environments require a new approach to design, implementation, execution and optimization of app.

- ◆ To build software development environment for Grid app that enables the effects of dynamism to be mitigated and controlled.
- ◆ To provide tools, compilers and libraries to help automate the construction of acceptably efficient Grid application from high-level inputs.
- ◆ To develop an experimental methodology for characterizing Grid software that can be used to evaluate and predict their perf, fault tolerance and scalability.

# Introduction to GrADS

- ◆ Discrete steps of application creation, compilation, execution and replaced with a continuous process of adapting application to the changing Grid.
- ◆ Two critical concepts to the working of this system:
  - Application is encapsulated as a *configurable object program* which can be dynamically mapped on Grid resources available at launch time
  - The system relies on performance contracts to specify the expected perf of the application.

# GrADS Program Preparation and Execution Architecture

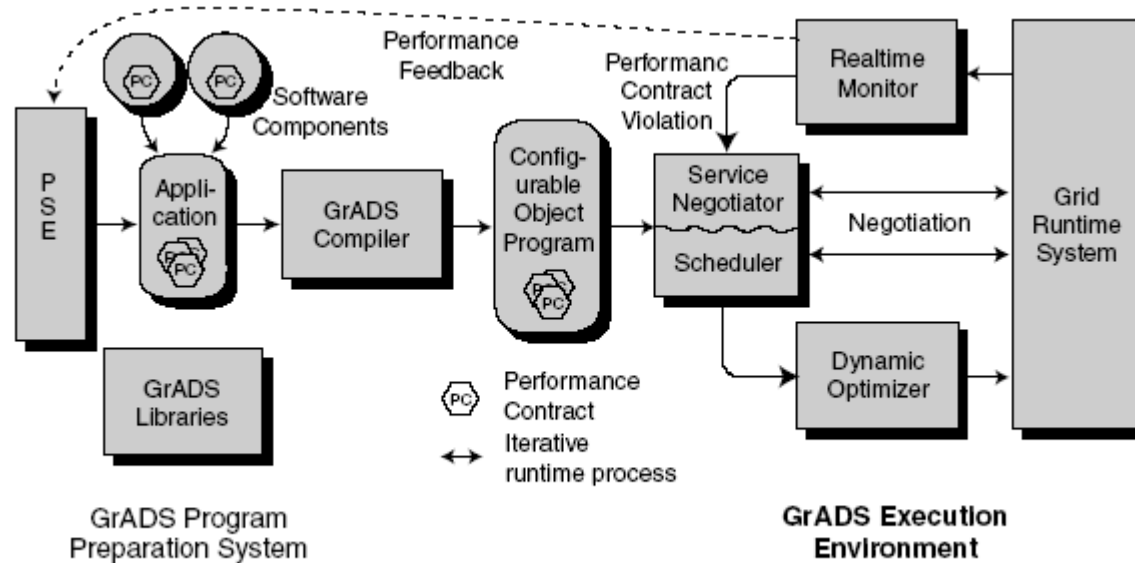


Fig. 1 GrADS preparation and execution architecture

# GrADS Program Preparation System

- ◆ Developer will eventually use high-level problem-solving environments (PSE) to assemble Grid application from toolkits.
- ◆ Efforts to date focused on tools that assist in constructing the perf estimator and mapper which are required components of *configurable object program*:
  - Condor ClassAD is extended to refer to resource sets, a critical capability for Grid scheduling.
  - Automatic construction of mapper
    - Graph-cluster algorithm is used to map communication to fast link
  - Automatic construction of perf estimator
    - Trial executions to determine impact of diff parameters on perf, and curve fitting to produce a final model.
  - Component integration
    - Be able to integrate perf estimator and mapper from components written by experts into effective estimator and mapper for whole program

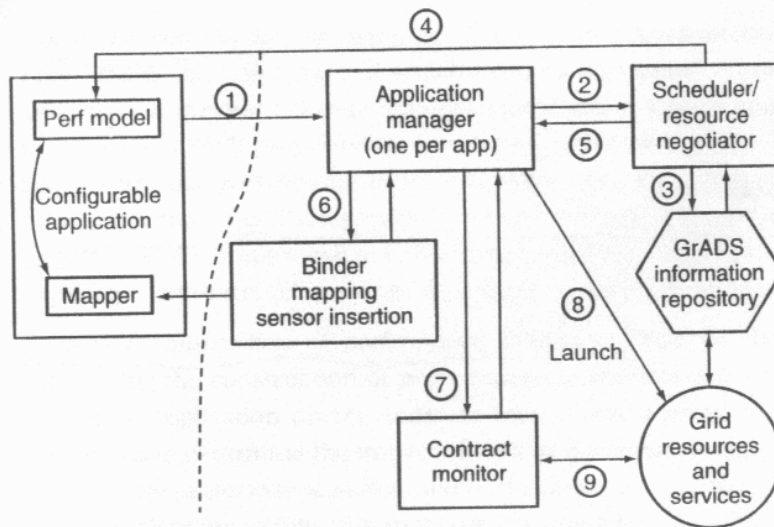
# Configurable Object Program

- ◆ Portable program that can be executed on diff Grid resources
- ◆ Components for automatic mapping and load balancing are required in addition to code:
  - A parallel application of MPI program
  - A mapper – maps computation and communication onto a given set of resources, determine load matching for the best performance.
  - A perf estimator – approximates perf that will be achieve on a given set of resource, serve as objective function for scheduler/negotiator.
- ◆ Configurable Object Program can be viewed as an application on an abstract parallel machine which is the GrADS execution environment

# GrADS Execution Environment

- ◆ Principal task is to automatic the process of mapping and executing a configurable object program on a set of Grid resources
- ◆ Uses the callback mapping and perf estimation function to handle resource allocation and load balancing.
- ◆ Manages the complex task of adapting the application to the dynamically changing Grid resources.

# GrADS Execution Environment(2)



Program launch in the GrADS execution system.

1. Instantiate an application manager(AM)
2. AM consulting with mapper provides an initial space of resources
3. Scheduler/negotiator requests sets of feasible resources
4. Scheduler/negotiator solves an optimization problem using perf estimator to find the best set of available resources
5. Resource are reserved and program launch begin
6. Binder instantiates the mapping and tailoring of application to the resources, and inserts sensor for the perf monitor
7. Contract monitor process is launched on the Grid
8. Application itself is launched on the Grid
9. Application continuously contact with contract monitor via sensors during execution. If perf sufficiently falls below estimates, AM can reschedule or migrate the app onto a diff set of resources

# Component Technologies

Component technologies for supporting the GrADS app development and execution framework:

## ◆ Performance Estimation

- Must be done interprocedurally so to take account of entire program
- Some must be done at launch time and some must be done at run-time
- May still be modified after program runs for some period of time

## ◆ Mapping

- Based on graph clustering, automatically produce task-graph mappers that assign computation and communication in such way that entire computation finishes as early as possible.
- Basic idea behind the mappers is to assign the computations at the endpoints of expensive communication edges to the same processor.

# Component Technologies(2)

- ◆ Whole-Program Compilation and Integration
  - Though provides better program performance and correctness, whole-program compilation significantly complicates the compilation environment and increases compile time.
  - Interprocedural compilation will become necessary when compiling for distributed heterogeneous Grids.
- ◆ Run-Time Compilation
  - Recompile after some needed run-time data are read
  - Recompile to update load balancing when needed to keep good performance of the program

# Component Technologies(3)

## ◆ Libraries

- High-level language strategy pursued by GrADS requires libraries of component software to be used to optimize performance.
- Extensive work is needed to make libraries be effectively integrated and optimized by interprocedural compiler into object program.

## ◆ Programming Support Tools

- The strategies envision for application development establish a complex relationship b/w the source version of the program and the version that runs on the Grid.
- Performance-improving changes must typically be made in terms of the program source. Tools must understand the perf of a program and be able to make transformations based on that understanding.

# GrADS Accomplishments

- ◆ GrADS project has succeeded in constructing prototype execution system that includes all the critical components except rescheduling and migration.
- ◆ GrADS has successfully run six application
- ◆ GrADS infrastructure is currently a prototype, and it is designed to be a complete and robust Grid middleware which exists on top of the Globus Toolkit.



# Project Links



<http://hipersoft.cs.rice.edu/grads/>

<http://www-pablo.cs.uiuc.edu/Project/GrADS/>



# **Programming the Grid: Distributed Software Components, P2P and Grid Web Services for Scientific Applications**

Dennis Gannon

Randall Bramley

Geoffrey Fox

# Introduction – Computational Grids

- ◆ Security services, which support user authentication, authorization
- ◆ Information services, which allow users to see available resources.
- ◆ Job submission services, which allow a user to submit a job to any compute resource that the user is authorized to use
- ◆ Co-scheduling services, which allow multiple resources be scheduled concurrently

# Grid Portal Architecture

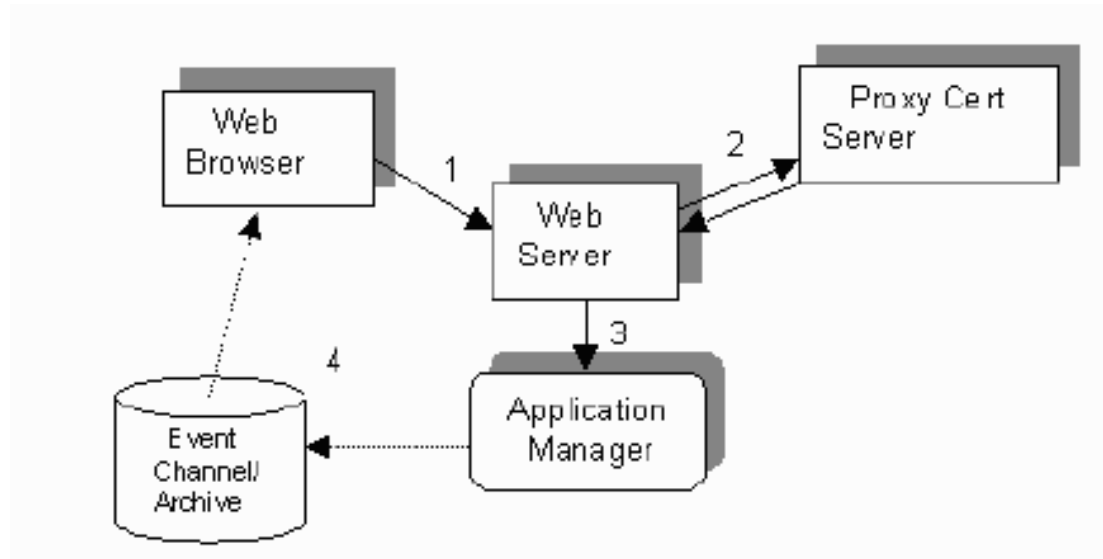


Figure 1. Grid Portal Architecture. 1. The User makes a secure connection from the web browser to the portal server. 2. The portal server then obtains a certificate from a proxy certificate server and uses that to authenticate the user with the Grid. 3. When the user completes defining the parameters of the computation the portal web server launches an *application manager*, which is a process that controls and monitors the actual execution of the grid computation. 4. In some systems, the application manager publishes an event/message stream to a persistent event channel-archive.

# Software Component Models

- ◆ Common Component Architecture concepts
  - *Provides ports* are component access points that provide an interface of functions that the component will evaluate on behalf of its client.
  - *Uses ports* are component features that represent a reference to an external object from within the component.

# Software Component Models(2)

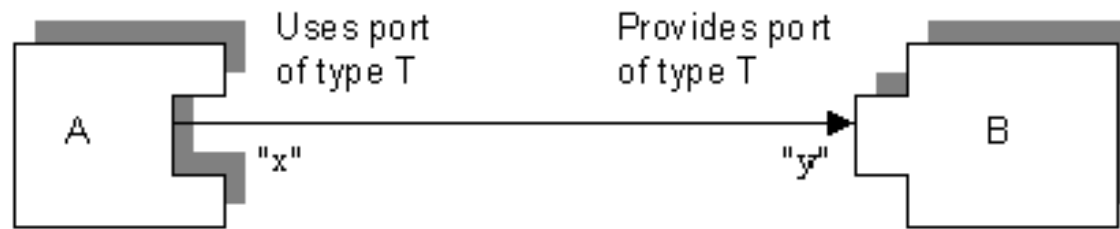


Figure 2. Connecting a uses port named "x" of type T of component A to a provides port named "y" of type T on component B means that the uses port object identified by name "x" has become a remote reference in A for invoking methods on interface "y" of B.

# Events and Messaging

- ◆ Component port connections often are not the most convenient or robust model for communicating information between the parts of a Grid application.
- ◆ An asynchronous messaging/event system is needed to solve this problem.
- ◆ A simple XML event system is built using XSOAP.
  - An event is an XML object that has the following required fields: a namespace, a type name, a timestamp and a source name.
  - The event system follows a simple publish-subscribe model.

# Events and Messaging (2)

## ◆ Generic Event Channel

- A mechanism to store the events being published and allow filtering and query of the events.
- Components using events to communicate need not know each other's location, and instead just need the location of an event channel.
- This robustness is crucial for Grid-distributed applications which use resources and components that are potentially unreliable.

# Example

## ◆ Wrapping and Coupling Applications: Chemical Engineering

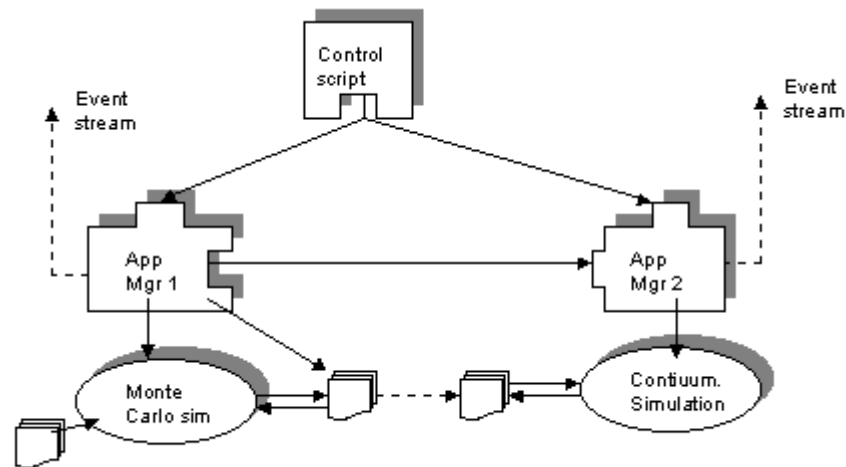


Figure 5. Two coupled Chemical Engineering simulation programs. Application Manger 1 signals Application Manger 2 when the Monte Carlo simulation completes a time step and the associated output state files have been migrated. Upon receipt of the message Application Manager 2 runs the continuum simulation. When this terminates control is returned to the first AM.

# Web Services and Grids

- ◆ Standards which defined web services:
  - Web Services Description Language (WSDL)
  - Universal Description Discovery and Integration (UDDI)
  - other standards like WSFL.
- ◆ Why are Web Services interesting?
  - Web service standards are simple, they are based on standard web technologies, and they are focused on making interoperability possible and easy.

# Grid Application Factory Service

- ◆ Web service + CCA model together

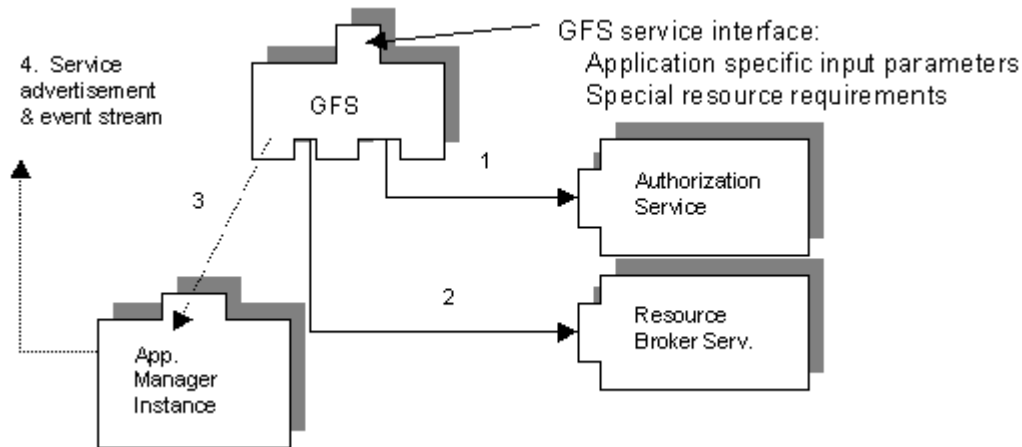


Figure 9. Client supplied application specific parameters and resource requirements to GFS. GFS service first (1) contacts an authorization service to verify that the user. Next (2) it contacts a resource broker service to find a suitable execution host. It then (3) launches an application manager instance and returns a handle to that instance. When the client does run it may publish an event stream and a WSDL service advertisement the client can use to contact it.

# Peer to Peer Grid Concepts

- ◆ Peer-to-Peer (P2P) systems can be divided into two categories:
  - File sharing utilities provide a global namespace and file caching and a directory service for sharing files in a wide-area distributed environment. Each user client program, which can access local files, is also a data server for files local to that host.
  - CPU cycle sharing of unused user resources usually managed by a central system, which distributes work in small pieces to contributing clients. Examples of this include Seti@home, Entropia and Parabon.

# Peer to Peer Grid Concepts(2)

- ◆ Compelling features of P2P system:
  - Purely user-space based and require no system administrator.
  - Designed to be very dynamic, with peers coming and going from the collective constantly.
- ◆ JXTA and the concept of PeerGroups
  - PeerGroup is a distributed collection of people that want to collaborate.
  - JXTA includes a series of protocols for PeerGroups including Discovery, Membership, Sharing, etc.



# Related Links

The Common Component Architecture Technical specification, See <http://www.cca-forum.org>.

Project JXTA, <http://www.jxta.org>