Holding slide prior to starting show
Processing Scientific Applications in the JINI-Based OGSA-Compliant Grid

Yan Huang
Presentation Outline

- Introduction to JISGA
- SWFL and its extensions
- Implementation issues
- Current and future works
JISGA

• is a Jini-Based Service-Oriented Architecture.
• extends a Jini system into an OGSA-compliant system for Grid computing by introducing Web service techniques into the Jini system
• All its services are presented as both Jini and Web service.
JISGA

• Service Workflow Language (SWFL) is the XML-based description language for composite applications and serves.
• workflow engine serves as the execution environment for the SWFL-described Grid applications.
• By using multiple distributed Job Processors and JavaSpaces as shared memory, It supports parallel processing for compute-intensive applications.
The Architecture of JISGA

Job Processor

JavaSpaces

SWFL Document

Workflow Engine
A JISGA Virtual Machine

Grid Services

JobProcessor (CPU)

JobProcessor (CPU)

JobProcessor (CPU)

JobProcessor (CPU)

JobProcessor (CPU)

Workflow Engine(API)

Workflow Engine(API)

Workflow Engine(API)

JavaSpaces (Shared Memory)

JavaSpaces Housekeeper

UDDI Registry

Lookup Service
Service Work Flow Language (SWFL)

• Is XML based for describing interacting Web services.

• extends Web Service Flow Language (WSFL) by
  – supporting the application of all the conditional and loop control constructs of the Java language to the composition of Web services.
  – Allows more general data mapping.
Why SWFL?

- Both Web Services and OGSA services present their implementation-free interfaces in a platform-free, XML-based language such as WSDL
- Presenting service composite applications in traditional programming languages limits the applications to a range of specific services and a specific time
Why SWFL?

• Representing applications in an XML-document allows applications be run in any environment where a set of services are available.

• Having a graph-based approach, SWFL gives the flexibility to dynamically partition and schedule services at run time in a distributed environment.
Limitations of SWFL

• In the original definition of a parallel for loop, there is no mechanism provided to allow message-passing between the parallel processes, which assumes that all the processes are running the same code but on different data with no communication between them.
Extensions to SWFL

• A label is specified for each parallel process.
  – A particular job can be assigned to a particular parallel process to support MIMD parallelism.
  – A message-passing path can be specified

• Allow specification of *shared* and *non-shared* variables to support shared-memory applications
  – Accessing a *shared* variable requires mutual exclusion
Extensions to SWFL

• Extensions to data mapping specification in SWFL to specify how a data structure is to be distributed across the processes.
  – In order to adequately support SPMD (Single-Program Multiple-Data) parallelism.
  – The mapping is implemented in two steps
    • a single-to-multiple data decomposition mapping
    • Mapping sub-data to processes
Mapping a data structure to parallel processes.
Extensions to SWFL

• A “multiple-to-single” data mapping mechanism in SWFL to support data merging. It can be
  – data reduction in which all the data sub-parts are combined using a binary operation such as
    addition, multiplication, max, min, logical and, etc.
  – combining a data sub-parts into a larger data structure by allocating each element in the sub-
    parts to a specific location in the target data structure.
Extensions to SWFL

Processes

Sub-Data

Data merging
JISGA Parallel Job Processing

Job Partition Algorithm

JavaSpace

write write
write take
take take
take

take

To be continued
Implementation Issues

- Point-to-point message passing:

  Process A → JavaSpaces
  Process B → JavaSpaces
  Process C → JavaSpaces
  Process D → JavaSpaces

- JavaSpaces

- Write

- take
Implementation Issues

• Broadcasting:

  * Process A
  * Process B
  * Process C
  * Process D

  - JavaSpaces
    - Notify
    - Read/Register
    - Write
Implementation Issues

- Allow multiple identical sub-jobs to be represented in a single sub-job description and submitted to the Sub-Job Queue.

![Diagram of multiple identical sub-jobs being submitted to a sub-job queue]
Implementation Issues

• Shared Data: only read and update operations allowed.
  – Update is combined take-and-write operations.

• Non-shared Data: only read and write operations.
Implementation Issues

- Data decomposition and merging
Current and Future Research

• SWFL 2.0 and JISGA 2.0
• Development of a Visual Service Composition software to build service composite application in an easy, graphic environment
• More experiments in integration Jini services, CORBA services and Globus services by using workflow engine
• A new version of workflow engine supporting other job description languages such as BPEL4WS and WSCI.