UniGrids and GPE
A Client Framework for Interoperability

Unicore Summit Sophia Antipolis, October 11-12, 2005
First of all...

What do we mean by interoperability?
Production UNICORE

- Stable, working solution
- No interoperability
New web service and Grid standards will allow better ways of interoperability
Need for Standards

- Defining standards like WSRF is not enough
- Need to define semantics of WSRF enabled services
- GPE defines a set of atomic services for job execution and data management
UniGrids Interoperability

- Atomic service interfaces define mandatory functionality for system, file and job management
- Different protocols and description languages are announced via resource properties

SOAP, WSDL, WSRF, WS-Addressing, WS-Security, JSDL and other standards

UNICORE/GS

Globus Toolkit

Other OGSA-compliant servers

Atomic Services
Three levels of interoperability

- **Level 1: Interoperability between WSRF services**
  - UNICORE/GS passed the official WSRF interop test
  - GPE and JOGSA hosting environments successfully tested against UNICORE/GS and other endpoints
  - **WSRF specification will be finalized soon!**
    - Currently: UNICORE/GS: WSRF 1.3, GTK: WSRF 1.2 draft 1

**WSRF Service API**
- UNICORE/GS
- GTK4
- CGSP

**WSRF Hosting Environment**
- UNICORE/GS-HE
- GTK4-HE
- GPE-HE
- JOGSA-HE

**Advanced services**
- GPE-Workflow
- GPE-Registry
- UoM-Broker
Three levels of interoperability

- **Level 2: Interoperability between atomic service implementations**
  - Client API hides details about WSRF hosting environment
  - Client code will work with different WSRF implementations and WSRF versions if different stubs are being used **at the moment!**
Three levels of interoperability

- **Level 3: GridBeans working with different Client implementations**
  - Independent of atomic service implementations
  - Independent of specification versions being used
  - GridBean run on different atomic service implementations without modifications
  - GridBeans survive version changes in the underlying layers and are easy to maintain
Grid Programming Environment (GPE)

- High-level Grid API
  - Descriptions
    - Resources (CIM)
    - Jobs (JSDL)
    - Workflows (BPEL)
  - Operations
    - Job management
    - File transfers
    - Brokering
    - Steering, etc.

Grid Programming Library

Open Grid Services Architecture (OGSA)

Web Service Resource Framework (WSRF)

- WSRF-enabled Servers
- WSRF-enabled Storage
- WSRF-enabled Network

Applications

- Grid SDK
- Grid Beans
- Client Framework

*Other brands and names are the property of their respective owners © Copyright 2004 Intel Corporation. All Rights Reserved.
Standards

- **JSDL (Job Submission Description Language)**
  - High level job description that can be submitted to all target systems offering a JSDL interface

- **CIM (Common Information Model)**
  - Used to describe resources
  - Usage of CIM management interfaces for Grid administration

- **BPEL (Business Process Execution Language)**
  - Integration of Grid Bean services into larger business process workflows

- **WS* (WS-Addressing, WSRF, WSN, etc.)**
  - Interoperation with other Grid Middleware

- **OGSA (Open Grid Services Architecture)**
  - Share components with other architectures
GPE Components Overview

- API and library for GridBean development
- Dynamically loaded, portable client plug-ins

Grid SDK

GridBeans

Client Framework

- Expert Client
- Application Client
- Portal

WSRF Hosting Environment

- GridBean Service
- Target System Registry
- Workflow Execution Service

- Target System Factory
- Target System Service
- Job Management Service
- Storage Management Service
- File Import Service
- File Export Service

Atomic Services

GPE services

Target System

- Backend (UnicoreGS)

Target System Interface (TSI)

Lightweight Java component to perform the actual work

Native GPE
Globus Toolkit 4.0
UNICORE/GS
or other

*Other brands and names are the property of their respective owners © Copyright 2004 Intel Corporation. All Rights Reserved.
Atomic Services Overview

- Atomic service interfaces define basic set of operations and properties that have to be available on a Grid
- Different implementations of interfaces for different infrastructures

Atomic Services

- **Target System Factory (TSF)**
  - Implementation
  - Current Implementations:
    - Globus Toolkit 4
    - UNICORE/GS
    - Native GPE
    - China Grid Support Package (CGSP)

- **Target System Service (TSS)**
  - Implementation

- **Job Management Service (JMS)**
  - Implementation

- **Storage Management Service (SMS)**
  - Implementation

- **File Import Service (FIS)**
  - Implementation

- **File Export Service (FES)**
  - Implementation

- Add a new target system to the Grid
- Manage target system
- Manage jobs on target system
- Manage files on storage
- Manage imports to storage
- Manage exports from storage
For some users it is sufficient to offer interfaces that are restricted to run and manage a certain application on the Grid.

For this category of users we implemented a thin **Application Client** with a functionality limited to application specific features.

Lightweight Java application that can be run on mobile devices
Client Framework: Expert Client

- Expert users want to:
  - build their own complex workflows to combine different Grid services to complex applications
  - access information and broker services
  - use different identities on different systems.

- The Expert Client:
  - provides a workflow editor to construct Grid specific BPEL workflows
  - manages multiple GridBeans
  - manages multiple certificates
Client Framework: Portal Client

- For the **Grid-unaware user**
  - GPE offers a web portal
    - Provide simple user interface in web browser
    - Hide Grid specific functionality
- GridBeans may provide JSR168 compliant portlets
  - In addition to client plug-ins
- GridBean portlets can be integrated into existing portal solutions
  - UPortal, GridSphere, etc.
Implement portable applications with GridBeans

- GridBeans are the interoperable successors of UNICORE Client plug-ins
GPE as interoperability framework

Expert Client
Application Client
Portal Client

UNICORE/GS
Globus Toolkit 4
China Grid Support Package
Other OGSA-compliant Grid servers

Atomic Service Client API
Atomic Services

*Other brands and names are the property of their respective owners © Copyright 2004 Intel Corporation. All Rights Reserved.
Additional GPE Services: Registry

- Registry keeps track of static and dynamic information
  - Hardware properties, available software, ...
  - Workload, available disk space, ...
- TSS contacts registry on startup and when properties change using WS-Notification
- Clients and services query informations about target systems from registry
- Implemented as WSRF Service Group
Additional GPE Services: Workflow Execution Service

- Use Grid-specific BPEL subset to orchestrate WSRF services in complex workflows
  - Allows integration into larger business processes
- Implemented as WSRF service itself
  - Workflow TSS accepts workflow JSDL descriptions in submit operation
  - BPEL description is extension in JSDL
- Information about workflow (state, input/output files, etc.) is kept in BPEL variables
  - BPEL variables are accessible as resource properties of the Workflow TSS
How does it work in concrete?
Atomic Service Interfaces based on WSRF

Operations
- TSR-EPR create(type, config)
  Extends WS-RP/LT
- Job-EPR submit(JSDL, initialTT)
  Extends WS-RP/LT
  start()
  abort()
  hold()
  resume()

Service
- Target System Factory (TSF)
  create
- Target System Service (TSS)
  create
- Job Management Service (JMS)
  Job1
  Job2
  Job3
- Storage Management Service (SMS)
  Home
  Temp
  Work
  Root
- File Import Service (FIS)
  File Import 1
  File Import 2
- File Export Service (FES)
  File Export 1
  File Export 2

Resource Type
- Memory=1Gb
  Running Jobs=23
  Storage=Temp, Root etc.

Properties
- Status=Running
  Original JSDL=<JSDL..>
  Execution JSDL=<JSDL..>
  etc.
- Available File Space=127kb
  Supported Protocols={GridFTP, UPL, scp}
  etc.
- Sourcefile=c:/tmp/test.txt
  DestFile=/tmp/test.txt
  Transferred=300kb
  etc.
- Sourcefile=/tmp/output.txt
  DestFile=c:/tmp/output.txt
  Transferred=12kb
  etc.

*Other brands and names are the property of their respective owners © Copyright 2004 Intel Corporation. All Rights Reserved.
Hiding platform-specific information with Application Resources

- Abstract job concept borrowed from UNICORE
  - No concrete platform specific information in job description (paths, libraries, etc.)
  - Job will be **incarnated** on target system
  - For security and portability reasons!

- Available applications can be queried from target system resource properties

- Use JSDL Posix extensions to specify required application resources in submitted job
File Transfers

- Atomic Services support different protocols
  - FTP
  - GridFTP
  - plain HTTP(s)
  - SOAP with Attachments (parallel)
  - Baseline file transfer
  - <add your own here...>

- Storage management announces available protocols via its resource properties
  - Client queries available protocols and selects appropriate one
    - GridFTP for large high-performant transfers
    - HTTP, SOAP w/a to work with firewall limitations
Adding a target system to a Grid

- Start Target System Interface
- Define Static Properties and Incarnation Rules
- Start Service Container

Configuration file describing hardware and software properties
Configuration file defining systems specific paths to commands, libraries, etc.

Static Properties
Incarnation Database (IDB)
Target System Interface (TSI)
Lightweight Java component to perform the actual work
Service container
TSF  TSS  JMS  SMS  FIS  FES
Alternative setup

- Start Target System Interface
- Define Static Properties and Incarnation Rules
- Invoke Target System Factory in service container to add new target system
Outlook: Virtualization in Grid Computing

- Security: Protection of sensitive user data
- Reliability
  - Other partitions on the same machine will remain unaffected if one partition crashes
  - Virtual machines can migrate during run-time
- „Configurability“
  - Current model: Static OS and applications
  - With virtualization: Dynamically deployed OS images and applications on user request

Virtual Machines are dynamically created and configured according to user request.
Virtualization Architecture in GPE

1. send resource request

2. request matching OS image

3. deploy OS image

4. create VM with OS image

5. reference to deployed VM

6. query, submit, store, shutdown, pause, migrate, ...

CLIENT

VIRTUAL TARGET SYSTEM FACTORY

Virtual Target System Service

OS Image Repository

Real Machine

Virtual Machine

Management Interface

Virtual Machine

OS

TSI

submit job

manage VM

TSI = Target System Interface

Other brands and names are the property of their respective owners © Copyright 2004 Intel Corporation. All Rights Reserved.
What is the current state and what are the next steps?
GPE Alpha Release Available for Download

Available at UNICORE SourceForge project

Application Client
- Lightweight client to load and run one application at a time

WSRF Hosting Environment
- Based on Axis 1.2 (RC3)
- Deploy services to Tomcat or run standalone server application
- Complete WSRF implementation

Example GridBeans
- Use source code as template for your own implementation

Admin Client
- Graphical administration interface
- Embedded standalone server

Complete Atomic Service Implementation
- File transfers based on SOAP with attachments
- „UNICORE-style“ Java TSI as execution back-end
- Runs on Windows and Linux/Unix
NEW: GPE4GTK!

- New SourceForge project
  - https://sourceforge.net/projects/gpe4gtk/
- Use Globus GASS server for file transfers
  - http/https transfers to work with firewall restrictions
- GridFTP for efficient file transfers
  - Needs opens port range in firewalls
- Includes Expert Client and BPEL Workflow engine
- UnigridsGS port running
Summary

Intel GPE...

...enables applications to run on and across different Grid infrastructures including UNICORE/GS and GTK

...provides a client framework to give users access to the infrastructure

...provides the GridBean concept and a programming API for Grid developers

...will support future virtualization and management concepts

...is available under BSD license
Thank you!