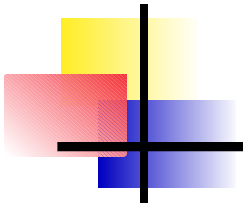
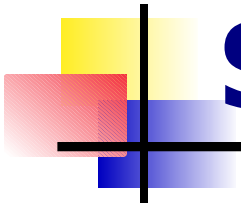


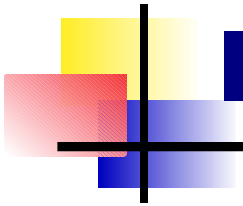
# Java Development and Grid Computing with the Globus Toolkit Version 3

Michael Brown
IBM Linux Integration Center
Austin, Texas



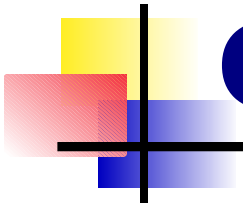
# Session Introduction

- Who am I?
  - mwbrown@us.ibm.com
  - Team Leader for Americas Projects
  - IBM Linux Integration Center, Austin
  - Author on 2 IBM Redbooks about Globus 3
- What will we talk about ?
  - Quick recap of grid and Globus from Intro session
  - Walk through code for two services and a client
  - Run the grid (fingers crossed)



# Introduction to Grid Computing

- Don't panic: it's just the new word for distributed computing
- We want to exploit resources on a wide, loose network
  - Not a cluster, but heterogeneous systems
  - CPU, storage, software, devices, all w/quality-of-service
- We started small in the early days with file copying, printing
- Later we moved to program-program communication
- We have to solve the same problems every time
  - Comms protocol, data format, security, management
- Why can't we just focus on the high-level tasks? Please?
  - Leverage existing standards
  - Follow accepted patterns



# Classes of Grids

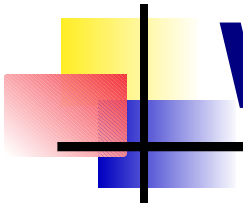
- Computation
  - The obvious one
  - Systems dedicated to tasks, or scavenging (including desktop)
- Data Virtualization
  - Intelligent, cross-network virtual filesystem
- Business Intelligence
  - Single coherent view of enterprise data
- Analytics
  - Computation + business intelligence
- High Availability
  - Provide better uptime due to no single point of failure

# What Is a Web Service?

- Program-program interoperability standard
  - Allows for construction of loosely coupled applications
- Platform and language independent
- You don't need previous knowledge of a service to use it
- Uses standards-based protocols
  - WSDL – Description of the web service interface
  - UDDI – Directory of available services
  - SOAP – Network procedure call
  - HTTP – Transport protocol (gets through ~ anything)
  - XML – Payload data format for messages
- See Hatzidakis, Nash, Lawrence

# Web Services Invocation

- Find a service by querying the UDDI registry
- Registry replies with a list of servers
- Ask Web service to describe its invocation format
- Service replies with WSDL definition
- Send SOAP request message to service
- Service replies with SOAP response message
  - Contains requested data or error info



# What Is the GGF?

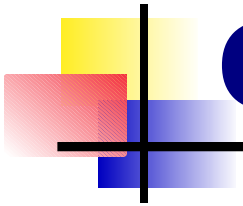
- Global Grid Forum http://www.ggf.org
  - " to promote and support the development, deployment, and implementation of Grid technologies and applications *via* the creation and documentation of "best practices" - technical specifications, user experiences, and implementation guidelines."
- Developed two key specifications for us
  - OGSA
  - OGSI

# Goals of OGSA and OGSI

- Provide a grid framework
- Leverage existing standards
- Services are heterogeneous
- Services are dynamic
- Services are searchable
- Services are callable by any system on the grid
- Services are independent of implementation

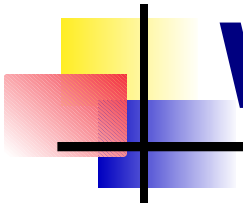
# What Is OGSA?

- Open Grid Services Architecture
- Defines what Grid Services are
  - Enhanced Web Services
- Defines a set of core and higher-level features
  - Factory, Registry, Discovery, Lifecycle, Authentication
  - Policy, Monitoring, Management, SLA, Hierarchy, QOS,
  - Query Service Data, Notification, Reliable Invocation
- Architecture layers can be implemented by different products, ISVs, open source communities, *etc*.
- Pick and choose the “best” implementation



# OGSA Programming Model

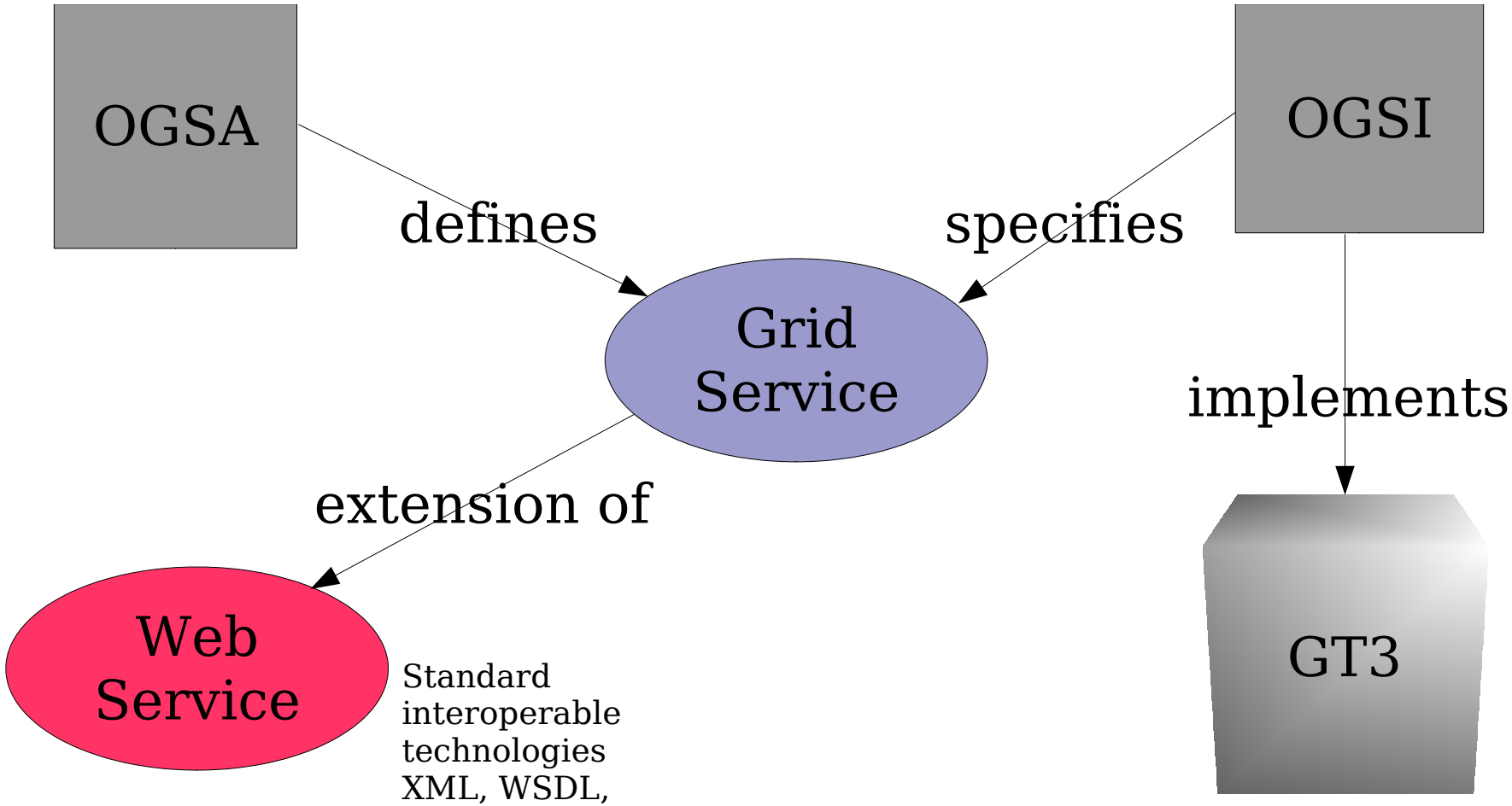
- All OGSA services adhere to specific service interfaces and behaviours
- Service interfaces defined by GWSDL
  - Grid WSDL
  - OGSA has identified several WSDL extensions



# What Is OGSI?

- Open Grid Services Infrastructure
- Formal specification of OGSA concepts
- What do we need to build grid services?
  - Stateful web services
  - Life cycle functions
  - Naming functions
  - Service Data
  - Notification of state change

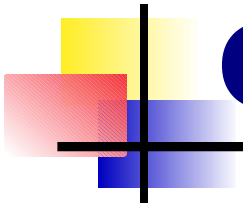
# OGSA/OGSI/GT3 Relationship




From GT3 Tutorial, Sotomayor

# What Is the Globus 3 Toolkit?

- Here we are finally!
- Open source implementation of OGSI 1.0
- Usable Web services-based grid services toolkit
- Runtime environment called the container
- Written in Java
- Some GT2 components have no GT3 equiv
  - Full support for GT2.4, written in C
  - GT3 job manager launches as separate process
  - Deprecating over time



# GT3 Core Services

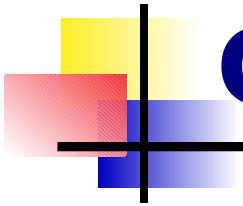
- Implements all OGSI interfaces
  - GridService, Factory, Notification, HandleResolver
  - Admin, Logging, Management, *etc*.
- Run time environment for services
  - Between application and plumbing
  - Embedded: library for any J2SE application
  - Standalone: lightweight server (test, development)
  - Web: runs in a J2EE servlet engine
  - EJB: expose stateful Entity and Session beans as services

# GT3 Security Services

- Transport- and message-level security
  - Transport being deprecated
  - Message based on WS-Security, XML Signature
  - At SOAP level
  - Per-session or per-message
- SSL, X.509 certificates
- Based on JAAS
  - Authentication, Authorization Service Framework

# GT3 Base Services

- Database services
  - Persisting Service Data in native XML db
- Managed Job Service
  - Notify, subscribe, pause, stop, query jobs
- Index Service
  - Query services around grid
- Reliable File Transfer
  - Guaranteed large file transfers *via* restart
- Replica Location Service



# GT3 Factories

- A service that creates other services
- Client:Service can be
  - 1:1 is stateful, transient
  - N:1 is stateful, persistent
- Defined by Deployment Descriptor
  - Java stubs generated by ant and used in client code
- Client gets a reference to the service factory
- Client creates an instance of the service
- Client calls methods on object like it was local

# GT3 Notification

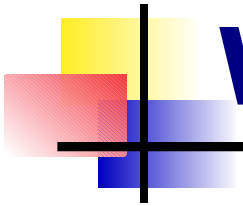
- Notification source fires events
- Notification sink receives events
- Associate with some dummy Service Data
- Add "ogsi:NotificationSource/Sink" to GWSDL
- Sink calls addListener() on source's Service Data
- Source calls notifyChange() to fire notification
- Sink implements deliverNotification() to catch

# GT3 Service Data

- Structured data associated with a grid service
  - Zero or more Service Data Elements per instance
- SDEs are described by an XML grammar
  - Java Bean is generated for each SDE type
- Client calls findServiceData("SDE1");
- Helper methods extract populated Bean instance
- Use in conjunction with Notification
  - Add payload to event

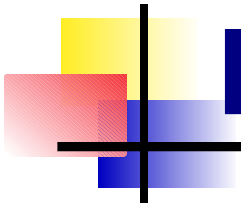
# SDEs in Every Grid Service

- GridServiceHandle – array of GSHs
- FactoryLocator – factory of this service
- TerminationTime – expected time of death
- ServiceDataNames – names of all SDEs
- Interfaces – names of all interfaces in service



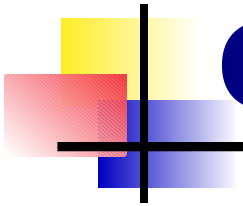
# Writing a Grid Service

- Define all aspects of the service interface
  - Java Interface and/or GWSDL
- Generate grid service support code
  - Server and client stub classes
- Tweak WSDL
- Implement the service guts
- Add overhead code for
  - Notification
  - Service Data
- Fortunately this process is very automated
  - ant is your friend



# Deploy the Service

- Write a deployment descriptor
  - Tells the Web server how the service is published
  - Extra fields define extra grid attributes
- Create a GAR
  - Grid ARchive, a JAR with grid-specific extras
  - Best to copy/paste an existing ant task
  - target=”makeGar”
- Deploy the GAR into a hosting environment
  - ant deploy -Dgar.name=<path to GAR>



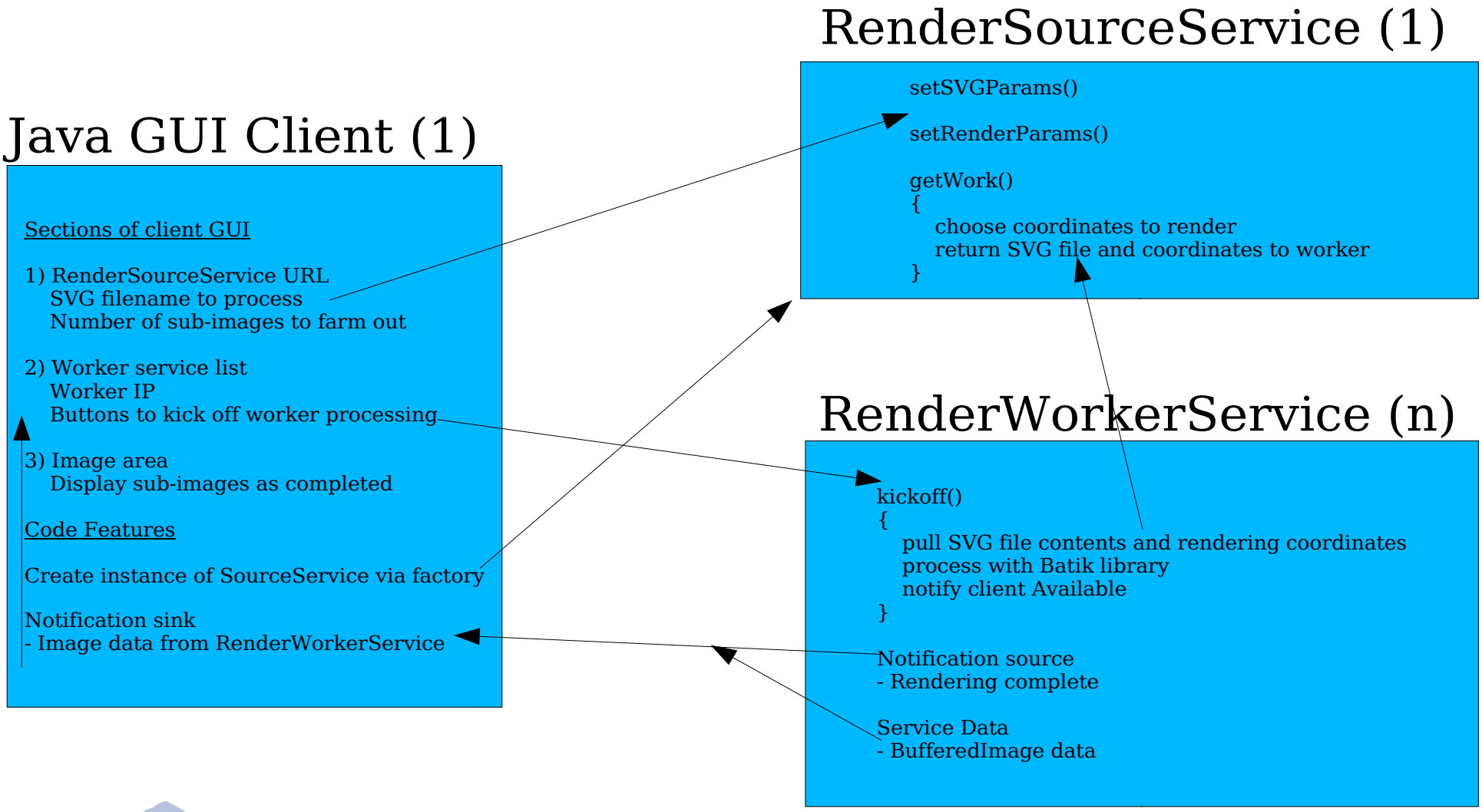
# Our Grid Project

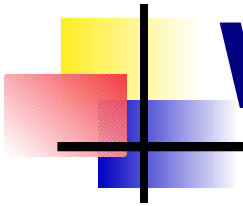
- Distributed SVG Rendering
- RenderSourceService
  - Farms out the SVG file and a rendering area to each worker
- RenderWorkerService
  - Builds BufferedImage from the specified area of the SVG file
- GUI Java application
  - Creates RenderSourceService *via* factory
  - Triggers worker services to get work from source service
  - Watches for status notifications & finished BufferedImages
  - Displays all status and tiles the sub-images as completed

# GT3 Features in Our Grid

- Factories
- Notification sinks & sources
- Service Data
- Standalone client
  - Driving processing
  - Displaying status
- Could it do more? Sure!
  - Security, Index Service, GridFTP, Logging, ...
  - Exercise for the reader

# Render Grid Architecture



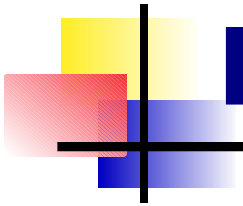


# What Are Our Roles?

- Me
  - Run Java GUI app to drive the grid
  - Host the RenderSourceService
- Y'all
  - Start the WorkerService on your wireless laptops
  - Tell me your IP addresses

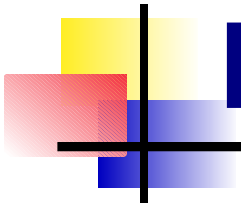
# Let's Look at the Service Code

- RenderSourceService
  - Instantiated *via* factory
  - Responds to requests for chunks of work
- RenderWorkerService
  - Responds to invitation to join grid
  - Pulls chunks of work from RenderSourceService
  - Renders BufferedImages *via* Apache Batik SVG library
  - Notification Source sends rendering complete to client
    - Pushed BufferedImage as ServiceData



# Let's Look at the Client Code

- Create RenderSourceService *via* factory
- Enter IPs and kick off RenderWorkerServices
- Notification Sink
  - Render completion status
  - Gets Service Data from workers *via* Push
- Tile completed sub-images into full image



# Prepare the Servers

- Install GT3 core
  - See separate instructions
- Deploy appropriate GAR
- Start the container
  - ant startContainer
- Create an instance

# Test the Render Grid

- Launch the client
- Enter URI of RenderSourceService
- Enter SVG filename to render
- Enter number of sub-images
- Enter RenderWorkerService IP addresses
- Click OK to kick off RenderWorkerServices
- Display sub-images as they are rendered