Outline

**Triana**
- Overview
- Triana services and their distribution
- Distribution policies
- The GAP interface and its relation to the Gridlab GAT

**Workflow**
- Overview of Triana workflow language
- Distributed workflow

**Web Services**
- Web service implementation - discovery, data types, choreography

**Applications**
- Overview of current Triana applications
- Coalescing Binary Scenario using the GAP interface

Matthew Shields, Cardiff University
What is Triana?
Remote Control Triana

Triana GUI and engine are Decoupled:
- use standardized task-graphs
- allows multiple devices to connect/disconnect
- persistent service (TCS)

Workflow, commands

Html/Java

Wap

Triana Controller Service
Persistent Service
→ A client logged on
Triana Distributed Work-flow

- flexible distribution: based around Triana Groups
- HPC and Pipelined distribution

Triana Controlling Service (TCS)

Workflow, e.g. BPEL4WS
Action Commands
Network

Matthew Shields, Cardiff University
Distributed Triana Prototype

- Based around Triana Groups i.e. aggregate tools
- Each group can be distributed
- Distribution policies:
  - HTC - high throughput/task farming
  - Peer to Peer - allow node to node communication
- Each service can be a gateway to finer granularities of distribution:
Current Triana Architecture

- GAP Interface
- Distributed Mechanisms
  - JXTAServe
  - P2PS
  - WServe
  - Gridlab Services
    - JXTA
    - Sockets
    - Web Services
    - Globus Services

Matthew Shields, Cardiff University
Java GAT Prototype

GAP (Java Prototype)

- Jxta
- Web Services
- P2PS
- OGSA (planned)
- Job Submission (GRMS)
- Data Management

- Advertising
- Discovery
- Communication

- Generic Job Submission
- Virtual filename data access

- Set of generic Java interfaces
- High level abstractions to Grid services
- Factory design - dynamic pluggable services

GridLab GAT (www.gridlab.org)

Matthew Shields, Cardiff University
Triana is inherently flow based
- Data flow - data arriving at component triggers execution
- Control flow - control commands trigger execution

Decentralised execution
- Data or Control messages sent along communication “pipes” from sender to receiver causes receiver to execute
- Synchronous or Asynchronous messaging (Implementation dependant)
- Multiple inputs can block or trigger immediately (Component designer defined)
Components and Definitions

- Component is unit of execution
- Components are defined in XML files:
  - Naming information
  - Input and output ports
  - Parameter information
- Why Components?
  - To simplify the application design process and to speed up application development
- The component model provides an infrastructure for the interaction of components
Taskgraph

- Internal object based workflow graph representation
  - Taskgraph - DAG
  - Tasks
  - Connections

- External XML representation
  - Simple XML syntax
  - List of participating Task definitions
  - Parent/Child connection
  - Hierarchical (Compound components)

- Alternative Languages & Syntax
  - e.g. BPEL4WS
  - Available through pluggable readers & writers.
Triana Pluggable Architecture

Plug-in Applications
- flexible: apps can use Triana in various ways, as a:
  - GUI
  - remote control GUI
  - or in full inc. GAP/GAT

3rd Party Application

Interactive
Interactive/Offline
Communication Channels
Application's Insert Points

Matthew Shields, Cardiff University
Workflow

- No explicit language support for control constructs
- Loops and execution branching handled by components
  - Loop component - controls loop over sub-workflow
  - Logical component - control workflow branching
- Unlike BPEL4WS or similar
- Flexibility of control - constraint based loops etc...
<?xml version="1.0" encoding="UTF-8"?>
<tool>
    <toolname>powerspectrum</toolname>
    <inportnum>0</inportnum>
    <outportnum>0</outportnum>
    <parameters>
        <param name="popUpDescription" type="unknown">
            <value>No description for tool</value>
        </param>
    </parameters>
    <tasks>
        <task>
            <toolname>FFT</toolname>
            <package>SignalProc.Algorithms</package>
            ...
        </task>
        <task>
            <toolname>SqMag</toolname>
            <package>Math.Functions</package>
            ...
        </task>
    </tasks>
</tool>
Example XML Workflow

```
<task>
  <toolname>OneSide</toolname>
  <package>SignalProc.Algorithms</package>
  <unitName>OneSide</unitName>
  <unitPackage>SignalProc.Algorithms</unitPackage>
  <inportnum>1</inportnum>
  <outportnum>1</outportnum>
  <inparam />
  <outparam />
  <input>
    <type>triana.types.ComplexSpectrum</type>
    <type>triana.types.Spectrum</type>
    <type>triana.types.TimeFrequency</type>
  </input>
  <output>
    <type>triana.types.ComplexSpectrum</type>
    <type>triana.types.Spectrum</type>
    <type>triana.types.TimeFrequency</type>
  </output>
```

Example XML Workflow

<parameters>
  <param name="minOut" type="internal">
    <value>0</value>
  </param>
  <param name="popUpDescription" type="unknown">
    <value>Converts two-sided spectrum to one-sided</value>
  </param>
  <param name="maxIn" type="internal">
    <value>1</value>
  </param>
  <param name="maxOut" type="internal">
    <value>2147483647</value>
  </param>
  <param name="defaultOut" type="unknown">
    <value>1</value>
  </param>
  <param name="minIn" type="internal">
    <value>1</value>
  </param>
  <param name="toolVersion" type="internal">
    <value>3</value>
  </param>
  <param name="defaultIn" type="unknown">
    <value>1</value>
  </param>
</parameters>
Example XML Workflow

<connections>
    <connection>
        <source taskname="FFT" node="0" />
        <target taskname="SqMag" node="0" />
    </connection>
    <connection>
        <source taskname="SqMag" node="0" />
        <target taskname="OneSide" node="0" />
    </connection>
</connections>
Distributing Triana Taskgraphs

Mapping tasks or groups of tasks to resources

Two stages:

1. Taskgraph annotation, XML definition for each task or group of tasks extended to specify resources and message channels
2. Data distribution, annotated sub-sections of taskgraph passed to resources
Custom Distribution

The workflow is cloned/split/rewired to achieve the required distribution topology.

Custom distribution units allow sub-workflows to be distributed in parallel or pipelined.

Distribution units are standard Triana tools, enabling users to create their own custom distributions.
Remote Deployment

- User can distribute any task or group of tasks (sub-workflow)

- Using the GAP Interface, Triana automatically launches a remote service providing that sub-workflow.

- Input, Output and Control Pipes are connected using the current GAP binding (e.g. JXTA Pipes)
Deploying and Connecting To Remote Services

- Running services are automatically discovered via the GAP Interface, and appear in the tool tree.

- User can drag remote services onto the workspace and connect cables to them like standard tools (except the cables represent actual JXTA/P2PS pipes).
Web Services

Choreographing web services
- Discovering web services
- Automatic interface for complex input/output types
- Deploying Triana services
- Connecting Triana services
Web Service Discovery 1

- Triana allows users to query UDDI repositories.

- Alternatively, users can import services directly from WSDL.
Discovered/Imported Web Services are converted into Triana tools
(service name = tool name)
(input message parts = in nodes)
(output message parts = out nodes)
etc...

Web Service tools are displayed in the user's Tool Tree (alongside local tools)
Connecting Workflows

- Web Service tools can be dropped onto the user’s workspace and connected like local tools.
- A workflow can contain both local and Web Service tools.
Complex Data Types

- Users can build their own interface for creating/mediating between complex types
- Alternatively, Triana can dynamically generate an interface from the WSDL2Java generated bean class
Converting the Bible into French

Simple but powerful example:
- `read_bible` - extracts verses from the bible
- `BabelFish` - translates between English and French (and other languages)
- Result = The Bible translated into French!

Red Boxes – Web Services

Matthew Shields, Cardiff University
Applications

User Interface Integration (direct plugin)
- Used GriPhyN to represent work-flow
- GEO 600 - interface with GEO++ monitoring system

GEMSS:
- Application Work-flow

Galaxy Formation
- Galaxy formation group at Cardiff for visualization

GEO 600 Signal Processing
- Quick Look data analysis
  - GEO 600 - rigorous unit testing (over 200 signal processing units)
- Data mining
- Coalescing Binary Code - will compare against MPI code
  - Uses 50+ Triana units to simulate the search
  - Full search complete ...
GEMSS: Maxillo-facial Surgery Simulation
GEO 600 Inspiral Search

**Background**
- Compact binary stars orbiting each other in a close orbit among the most powerful sources of gravitational waves
- As the orbital radius decreases a characteristic chirp waveform is produced - amplitude and frequency increase with time until eventually the two bodies merge together

**Computing**
- Need 10 Gigaflops to keep up with real time data (modest search..)
- Data 8kHz in 24-bit resolution (stored in 4 bytes) -> Signal contained within 1 kHz = 2000 samples/second
  - divided into chunks of 15 minutes in duration (i.e. 900 seconds) = 8MB

**Algorithm**
- Data is transmitted to a node
- Node initialises i.e. generates its templates (around 10000)
- fast correlates its templates with data
Coalescing Binary Search
Coalescing Binary Scenario

Controller

- Submit Job
- Optimised Mapping

Logical File Name

GW Data

Distributed Storage

GW Data

Email, SMS notification

GAT (GRMS, Adaptive)

GAT (Data Management)

CB Search

Gridlab Test-bed

Matthew Shields, Cardiff University
Triana Submits Itself …

GAP (using GRMS service)

Testbed
Thanks!

All this and more ... at:

http://www.trianacode.org/

http://www.gridlab.org/

Matthew Shields, Cardiff University