Uniform Resource Visualization (URV): Software & Services

Diane T. Rover and Kukjin Lee

Department of Electrical & Computer Engineering
Iowa State University

Dagstuhl Seminar – Performance Analysis and Distributed Computing
August 2002
Introduction

• System Visualization
  – Heterogeneity
  – Hierarchy
  – Scale
  – Dynamics
Grid Performance Problems

- **Problems**
  - TCP/IP retransmission
  - HPF array allocation failure
  - Nexus thread scheduling delay
  - Poor SP-2 utilization

- **Questions**
  - What is the root problem?
  - How should the problem be reported to the user?
  - Who is the user (e.g., an application developer, a network manager, and/or library developer)?

Grid Performance Problems

- Grid performance analysis/measurement requirements
  - Data correlation across semantic levels
  - Hierarchical performance visualization
  - Interactive drilldown
  - Dynamic optimization


Rover, Lee  PADC 2002
Grid Performance Problems

Application Code

MPI

HDF

MPI-IO

TCP/IP

IBM SP-2 application subtask

Performance Toolbox, IBM

Vampir, Pallas

Application Code

HPF

Nexus/Globus threads

HDF

MPI-IO

TCP/IP

SGI Origin2000 application subtask

Pablo, UIUC

Vis5D, U of Wisconsin

Rover, Lee

PADC 2002
URV Research

• Target
  – Performance analysis & resource monitoring

• Challenge
  – Information integration in performance visualization
Questions of Interest

• How can heterogeneous resources be presented uniformly?
• Can visualization be reused across domains?
• How can a visualization system be constructed from pre-defined modules?
• How can a visualization system deal with a dynamic environment?
• How can visualization design knowledge be shared?
Visualization Model

- **Resource**
  - Physical entity, e.g., processor
  - Logical entity, e.g., array

- **Resource Mon component**
  - Monitoring and control services for a resource

- **Visualization component**
  - Rendering data

- **Connector**
  - Set of services provided to and required of other components
What URV Does

Without URV

- Resource-specific development
- No reuse

With URV

- New Sensors
- New Views

URV
Uniform Resource Visualization

- URV is a component-based strategy for:
  - Constructing resource visualizations
    - Composing system-level views
    - Sharing visualization design knowledge
    - Describing visualizations and their interfaces uniformly
  - Monitoring and analyzing distributed heterogeneous systems
    - Viewing heterogeneous resources, levels in a coordinated framework
URV Framework

• Provide basic persistent services for structured development of URV views (e.g., on top of Grid middleware services)
  – Metadata service
  – Directory service
  – Connection service
  – Composition service

• Maintain repositories
  – Component repository
  – Connector repository
URV Services

- Metadata Service
  - processing of XML documents
- Directory Service
  - searching for & retrieving resource monitoring information
- Connection Service
  - matching and interfacing individually developed components
- Composition Service
  - integrating multiple performance views
- Component Repository
  - storage of visualization components
- Connector Repository
  - storage of template connectors
Grid Monitoring Architecture

GMA (Global Grid Forum, Performance WG)  
Relation of URV to GMA
Benefits of URV

- Standardized performance visualization development
  - Uniform descriptions of views for hardware/software resources
  - Framework for developers to supply and catalog URV views
- Reusable performance visualization
- Extensible performance analysis tool development
- Integrated monitoring of heterogeneous resources
- Sharable visualization design knowledge
Key Issues

• Connection between producer and consumer of performance data
• Reuse of performance monitoring technology
  – Views
  – Instrumentation
Multi-level Performance Monitoring

VR System

Application

VR Juggler

Distributed Shared Object

Plexus

Hardware

RMC

1. Register

RMC

Performance Monitoring Directory

3. RMC info

2. Search

URV Service

4. Select

Connector

VC

5. Subscribe

URV: Uniform Resource Visualization
RMC: Resource Monitoring Component
VC: Visualization Component
Connection Technology & Services

- Run-time configurable connector
- Component wrapper
- SOAP-enabled interoperable interface
Connection

VR System

Application

VR Juggler

Distributed Shared Object

Plexus

Hardware

RMC

RMC

RMC

RMC

Performance Monitoring Directory

Runtime Configurable Connector

URV Service

Connector

VC
Connection Service

- Provides executable connectors between components, and establishes control of connection.

1. Identify interface
2. Define a connector
3. Retrieve a connector
4. Insert Connector Repository
Connection

VR System

Application

VR Juggler

Distributed Shared Object

Plexus

Hardware

RMC

RMC

RMC

RMC

Performance Monitoring Directory

URV Service

Connector

VC

RMC Component Wrapper
Component Wrapper

• Provides uniform interfacing and dynamic bridging between components

• Supports:
  – Definition of common communication interface
  – Concise syntax for SOAP-enabled interface
Connection

VR System

Application

VR Juggler

Distributed Shared Object

Plexus

Hardware

RMC

RMC

RMC

RMC

Performance Monitoring Directory

URV Service

Connector

VC

Interoperable Interface
SOAP-enabled Interfacing

- **SOAP**: Simple Object Access Protocol
  - Use of XML and HTTP to access services, objects and servers in a platform-independent manner
    - XML-based object invocation protocol
    - Originally developed for distributed applications to communicate over HTTP
  - Simple remote object accessing mechanism
    - Lightweight remote procedure call or message passing

- **Benefits**
  - Platform independent development
  - Interoperable interaction
Reuse Technology & Services

- URV descriptors: knowledge representation & processing for views and instrumentation
- Visual design schema
- Visual design composition
Reuse

VR System

Application
VR Juggler
Distributed Shared Object
Plexus
Hardware

Performance Monitoring Directory

VC Component Descriptor

URV Service
Connector
VC
URV Descriptors

- RMC Descriptor, VC Descriptor
- Description of a component
  - Searchable XML-based specification
  - Concept: what the component does
  - Content: how the concept is implemented and specialized
  - Context: the domain in which the component may be applied
VC Descriptor

- **Concept**
  - What to visualize, how to interact with other components

- **Content**
  - How to visualize

- **Context**
  - How to identify, create, launch


Rover, Lee  
PADC
Visual Design Schema

• Represents visual design knowledge
• Each schema corresponds to a family of related performance views

TAU, U of Oregon  Vampir, Pallas  JumpShot, ANL

A family of Gantt performance views
Reuse

VR System

- Application
- VR Juggler
- Distributed Shared Object
- Plexus
- Hardware

Performance Monitoring Directory

- URV Service
- Connector
- VC

VR System

VC Composition Service

Reuse
Visual Design Composition

- Composition service as part of URV framework
- Composition creates system-level views representing multiple resources.
- Two types
  - Union $\oplus$: set of concrete transformation rules
  - Synthesis $\otimes$: design activity supported by a framework
Union

- Aggregate of RMC1 and RMC2 unchanged
- Connector replaced via transformation
- Visualization replaced via transformation
  - Aggregate of separate visualization components VC1 and VC2
  - Replaced by new visualization VC3
Performance Analysis Scenario
Union Example

- Two typical performance monitoring levels
  - Middleware level: remote object access request over time
  - Network level: TCP/IP transmission over time
Union Example

- Middleware level
  - Remote object access over time
  - Time-series performance view
Union Example

- Network level
  - TCP/IP transmission over time
  - Time-series performance view

Application Code

DSO

Plexus

TCP/IP

Rover, Lee

PADC 2002
Union Example
Union Example

RMC₁, Dr₁, Dr₂, RMC₂

VC₁, Dv₁, Dv₂

VC₂

Composition Service

Dv₁, Dv₂

Dv₃
Union Example
Union Example
Union Example
URV Challenges

• Systematic selection of compatible components
• Dynamic bridging of individually developed components
• Formal representation of visual design knowledge
• Systematic adaptation of design knowledge to build a new visualization
• Dynamic integration of performance views at multiple levels
• Data integration vs. view composition vs. performance semantics across resources
Related Work

• On-line performance monitoring
  – PGRT & EPIRA (http://www.egr.msu.edu/Pgrt)

• Automated design
  – Sage (http://www.cs.cmu.edu/~sage/)
  – ViA: A personal visualization assistant
    (http://www.csc.ncsu.edu/faculty/healey/projects/ViA.html)

• Component-based framework
  – CCAT (Common Component Architecture Toolkit)
    (http://www.extreme.indiana.edu/ccat/)

• Performance Monitoring in Grid
  – WP3: Information and Monitoring Services under DataGrid Project