Outline

- Introduction
  - Grid Performance Problems
- Uniform Resource Visualization
  - Components & Connector
- URV Framework
- Related Work
- Summary
Outline

- Introduction
  - Grid Performance Problems
- Uniform Resource Visualization
  - Components & Connector
- URV Framework
- Related Work
- Summary
Introduction

- System Visualization
  - Heterogeneity
  - Hierarchy
  - Scale
  - Dynamics

**Devices**
- CPU
- Network IC
- Storage

**Operating System**
- Process Mgt.
- I/O
- Memory Mgt.

**Distributed System Management**
- Load Balancing
- Resource Allocation

**APIs & Run-Time Services**
- Scalable I/O
- Data Management
- Security

**Application**
- Distributed Application

**LEVEL**
- RESOURCE
- NODE
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)?


K.Lee, D.Rover, MSU, 6/2001
Grid Performance Problems

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

Grid Performance Problems

Application Code

MPI

HDF

MPI-IO

TCP/IP

IBM SP-2 application subtask

Vampir, Pallas

Nexus/Globus threads

HDF

MPI-IO

TCP/IP

SGI Origin2000 application subtask

Pablo, UIUC

Performance Toolbox, IBM

Vis5D, U of Wisconsin

K.Lee, D.Rover, MSU, 6/2001
Our Focus

Objective

- Performance analysis & resource monitoring

Challenge

- Information integration in performance visualization
Question

- 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 **dynamic behaviors**?
- How can visualization design knowledge be **shared**?
Outline

- Introduction
  - Grid Performance Problems
- Uniform Resource Visualization
  - Components & Connector
- URV Framework
- Related work
- Summary
Uniform Resource Visualization

- URV is a component-based strategy
  - 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
Visualization Construction

- Resource
- Two independently executing components
  - Resource-monitoring component
  - Visualization component
  - Described with formal descriptors
- Connector
  - Connects components
Visualization Model

Resource

- Physical entity (e.g., router, sensor)
- Logical entity (e.g., process, scheduler)
Visualization Model

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

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

- Introduction
  - Grid Performance Problems
- Uniform Resource Visualization
  - Components & Connector
- URV Framework
- Related work
- Summary
URV Framework

- Provide basic persistent services for structured development of URV views (on the top of Grid middleware services)
  - Metadata service
  - Directory Service
  - Connection Service
  - Composition Service
- Maintain repositories
  - Component repository
  - Connector repository
Formal Descriptor

- Description of a component
  - Concept: what the component does
  - Content: how the concept is implemented and can be specialized
  - Context: the domain in which the component may be applied

- Searchable XML-based component specifications

- For designing uniform interfaces for accessing and manipulating heterogeneous resources
Visualization Descriptor

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

- **Content**
  - How to visualize

- **Context**
  - How to identify and create

```xml
<componentName>Plotting</componentName>...
<graphicModel>
    <datatype>quantative</datatype>....
    <nVar>1</nVar>....
    <dimension> 2 </dimension>...
    <xDim>point</xDim>...
</graphicModel>...
<dataModel>
    <data> . . . </data>
</dataModel>

<interfaceDef>...
    <port>
        <methods>getData</methods>...
        <parameters>yvalue</parameters> ...
    </port>
</interfaceDef>...

<creationInfo>
    <hostName>pgrt.egr.msu.edu</hostName>
    <platform>windows</platform>...
    <creationMethod>java</creationMethod>
    <creationEnv>...</creationEnv>
</creationInfo>
```
Metadata Service & Directory Service

- **Metadata service**
  - XML-based descriptor parsing

- **Directory service**
  - Component discovery and registry
Connection Service

- Defines a connector to establish control of connection between components.

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

Connection Service

Resource Monitoring Component

Connector

Visualization Component

Descriptor

Connection Repository
Composition Service

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 $RMC_1$ and $RMC_2$ unchanged
- Connector replaced via transformation
- Visualization replaced via transformation
  - Aggregate of separate visualization components $VC_1$ and $VC_2$
  - Replaced by new visualization $VC_3$
Synthesis

- Conscious activity by a human designer that cannot be automated fully
- Indexing of visualization components according to a classification scheme
- Designer specifies a set of search attributes to guide the search for a component
Performance Analysis Scenario

IBM SP-2 application subtask:
- Application Code
- MPI
- HDF
- MPI-IO
- TCP/IP

SGI Origin2000 application subtask:
- Application Code
- HPF
- Nexus/Globus threads
- HDF
- MPI-IO
- TCP/IP

K.Lee, D.Rover, MSU, 6/2001
Union Example

Two typical performance monitoring levels

- Application level: HDF I/O request over time
- Network level: TCP/IP retransmission over time
Union Example

- Application level
  - HDF I/O request over time (remote data access)

Diagram:

- Application Code
  - HPF
  - Nexus/Globus threads
  - HDF
  - MPI-IO
  - TCP/IP

SGI Origin2000 application subtask

K.Lee, D.Rover, MSU, 6/2001
Union Example

- Network level
  - TCP/IP retransmission over time

SGI Origin2000 application subtask
Union Example

\[ \text{Composition Service} \]

\[ \begin{align*}
\text{RMC}_1 & \quad D_r_1 \\
\text{VC}_1 & \quad D_v_1 \\
\text{RMC}_2 & \quad D_r_2 \\
\text{VC}_2 & \quad D_v_2 \\
\end{align*} \]

\[ D_v_3 \]
Union Example

RMC₁

Dr₁

Dv₁

Dr₂

Dv₂

RMC₂

Dv₁

Dv₂

Composition
Service

DV₁

DV₂

DV₃

VC₁

VC₂

VC₃

Composition
Service

K.Lee, D.Rover, MSU, 6/2001
Union Example

RMC_1 \quad Dr_1 \quad Dr_2 \quad RMC_2

VC_1 \quad Dv_1 \quad Dv_2 \quad VC_2

Dr_1 \quad Dr_2 \quad Dv_3

Composition Service

 VC_3

RMC_1 \quad RMC_2

VC_3

Connection Service

K.Lee, D.Rover, MSU, 6/2001
Synthesis Example

IBM SP-2 application
subtask

- Application Code
- MPI
- HDF
- MPI-IO
- TCP/IP

SGI Origin2000 application
subtask

- Application Code
- HPF
- Nexus/Globus threads
- HDF
- MPI-IO
- TCP/IP

K.Lee, D.Rover, MSU, 6/2001
Synthesis Example

- Application level
  - HDF I/O request over time (remote data access)

Application Code

- HPF
- Nexus/Globus threads
- HDF
- MPI-IO
- TCP/IP

SGI Origin2000 application subtask
Synthesis Example

- Process level
  - Message overhead over time

```
Application Code
   ↓
MPI
   ↓
HDF
   ↓
MPI-IO
   ↓
TCP/IP
IBMP SP-2
application
subtask
```

![Graph showing message overhead over time](image)
Synthesis Example

HDF I/O request (remote data access)

Composition Service

Message overhead

K.Lee, D.Rover, MSU, 6/2001
Development and Support Tools

![Component Integration Window]

- Browser
- LDAP Browser Window
- XML Viewer
- Invocation result

K.Lee, D.Rover, MSU, 6/2001
Outline

- Introduction
  - Grid Performance Problems
- Uniform Resource Visualization
  - Components & Connector
- URV Framework
- Related work
- Summary
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/)
Outline

- Introduction
  - Grid Performance Problems
- Uniform Resource Visualization
  - Components & Connector
- URV Framework
- Related work
- Summary
Benefits of URV

- Standardized performance visualization development
  - Uniform descriptions of views for hardware/software components
  - 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
Thank you for your attention

Uniform Resource Visualization

Questions, Comments, Research Collaboration…

http://www.egr.msu.edu/urv
leekukji@msu.edu