Grid Computing:

An Introduction

Keith R. Jackson

Lawrence Berkeley National Laboratory

Overview

- What is a Grid?
- Grid Components
- Grid Evolution
- Web Services
- Next Generation Grids
- Conclusion

The Grid Problem

- Enable flexible, secure, coordinated resource sharing among dynamic collections of individuals and institutions
  - Grid Resource = Any entity that is to be shared
    - computers, storage, data, instruments (e.g.: microscope, telescope, etc.)
- Enable communities ("virtual organizations") to share geographically distributed resources as they pursue common goals -- assuming the absence of…
  - central location
  - central control
  - existing trust relationships

The Grid Problem

- Resource sharing
  - Computers, storage, sensors, networks, …
  - Sharing always conditional: issues of trust, policy, negotiation, payment, …
- Coordinated problem solving
  - Beyond client-server: distributed data analysis, computation, collaboration, …
- Dynamic, multi-institutional virtual organizations
  - Community overlays on classic org structures
  - Large or small, static or dynamic

Technical Requirements

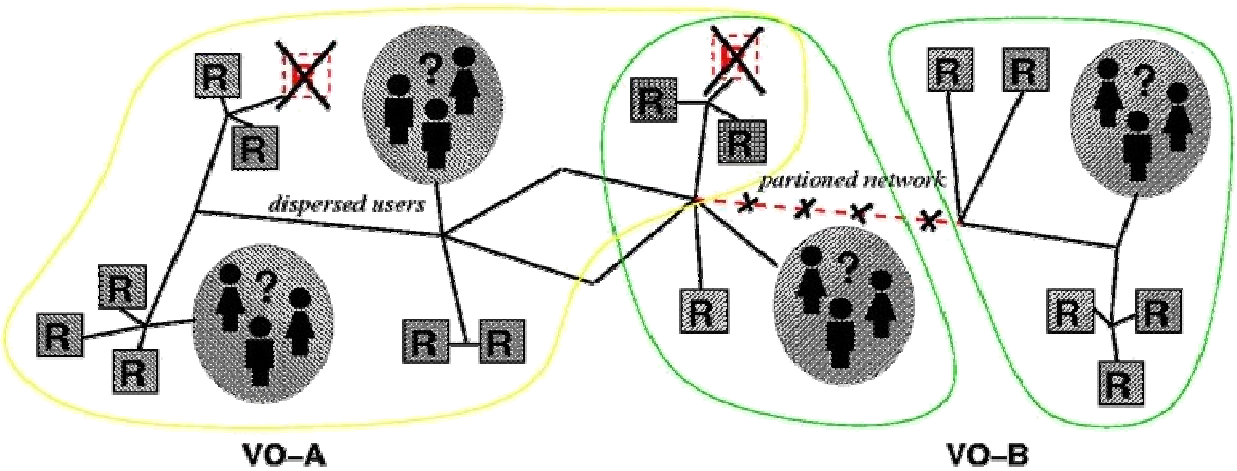
- Dynamic formation and management of virtual organizations
- Discovery & online negotiation of access to services: who, what, why, when, how
- Flexible security mechanisms
- Configuration of applications and systems able to deliver multiple qualities of service
- Autonomic management of distributed infrastructures, services, and applications
- Management of distributed state
- Open, extensible, evolvable infrastructure
- Minimize the burden to setup and manage

What is a Grid

- Dynamically link resources/services
  - From collaborators, customers, infrastructure, … (members of evolving “virtual organization”)
- Into a “virtual computing system”
  - Dynamic, multi-faceted system spanning institutions and industries
  - Configured to meet instantaneous needs, for:
- Multi-faceted QoS for demanding workloads
  - Security, performance, reliability, …

What is a Grid?

Distributed infrastructure must be resilient in the face of frequent failures



"The Anatomy of the Grid", Foster, Kesselman, Tuecke, 2001

Brief History of the Grid

- Distributed computing research in the 1980's
  - Project Athena at MIT
  - Andrew at CMU
  - Condor project started in 1988 (Livney U. Wisc)
- Late 80's early 90's sees a dramatic increase in network capacity.
- I-WAY (SC 95)
  - 17 Sites
  - 60+ Applications
  - Catalyzed the community
  - Start of the Globus Project

Examples of Production Grid Deployments

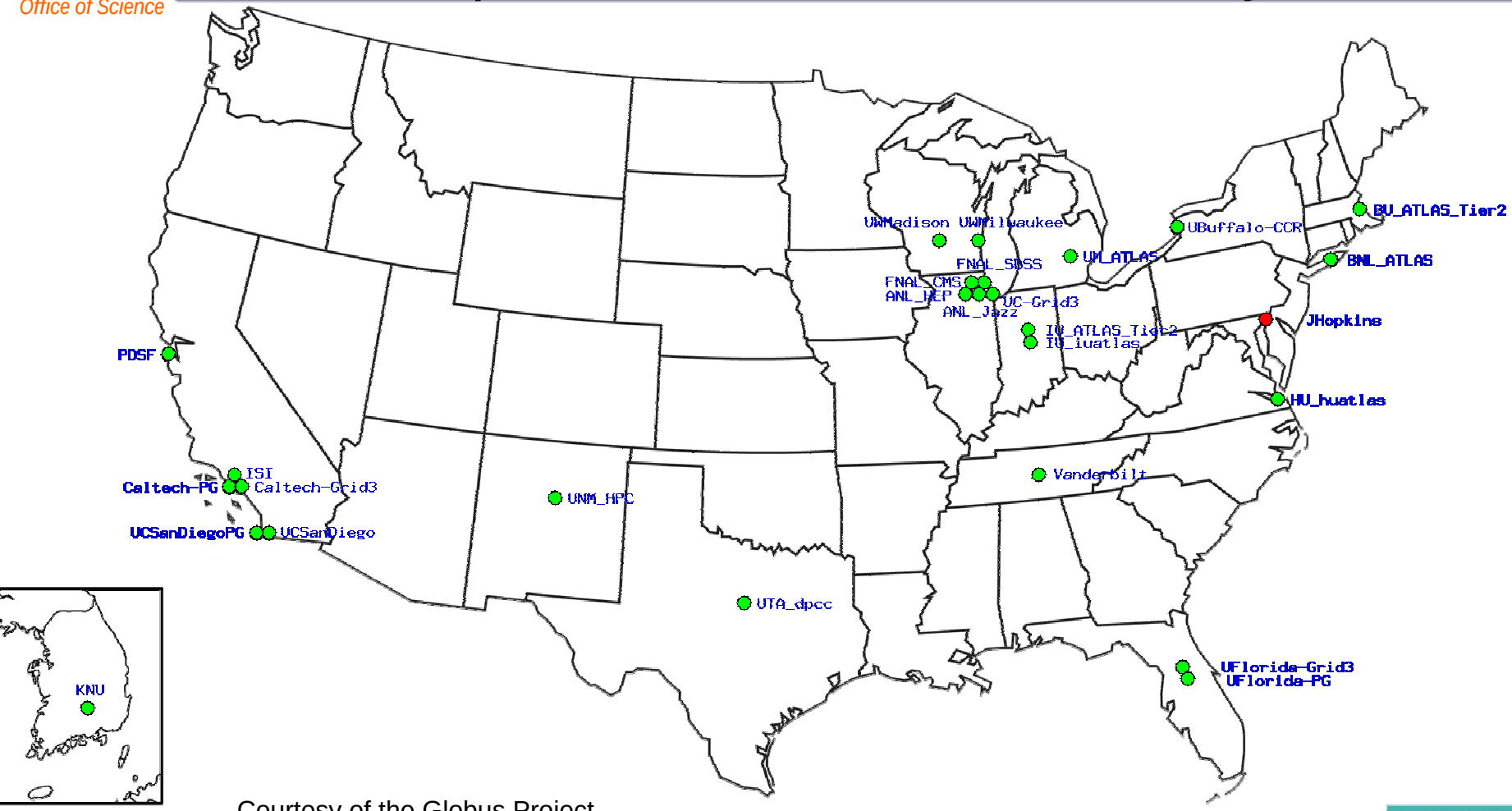
"Persistent deployment of Grid services in support of a diverse user community"

P re-WS
W eb Services

- Grid2003/OSG P
  - 26 sites, O(3000) CPUs, 2 countries
- LHC Computing Grid P
  - 25 sites, international
  - High energy physics
- NorduGrid P
  - 24 clusters, 724 CPUs, 6 countries; physics
- NASA IPG P
  - 4 sites, O(3000) CPUs
  - Aeronautics
- NEESgrid (prod. 2004) W
  - Instruments, data, compute, collaborative
  - Earthquake eng.
- TeraGrid P
  - 5 sites, expanding

Courtesy of the Globus Project

Grid2003 System for Computation-Intensive Data Analysis



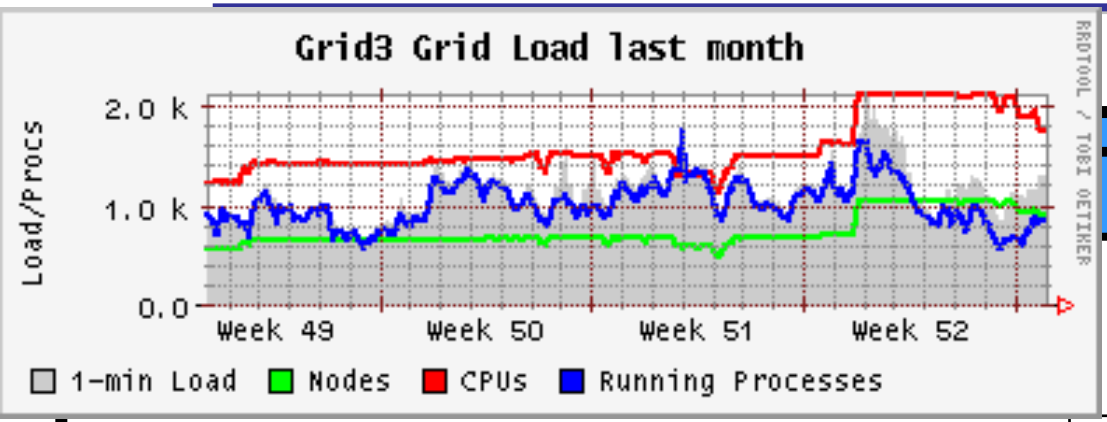
Courtesy of the Globus Project

Status on 11/19/03 (http://www.ivdgl.org/grid2003)

Grid2003 Components

- **Computers & storage** at 28 sites (to date)
  - 2800+ CPUs
- **Uniform service environment** at each site
  - Globus Toolkit provides basic authentication, execution management, data movement
  - Pacman installation system enables installation of numerous other VDT and application services
- **Global & virtual organization services**
  - Certification & registration authorities, VO membership services, monitoring services
- **Client-side tools** for data access & analysis
  - Virtual data, execution planning, DAG management, execution management, monitoring
- **IGOC**: iVDGL Grid Operations Center

Grid2003 Metrics



	Target	Achieved
	400	2762 (28 sites)
Number of users	> 10	102 (16)
Number of applications	> 4	10 (+CS)
Number of sites running concurrent apps	> 10	17
Peak number of concurrent jobs	1000	1100
Data transfer per day	> 2-3 TB	4.4 TB max

Grid2003 Software Stack ("Virtual Data Toolkit")

Application

Chimera Virtual Data System

DAGMan and Condor-G

Globus Toolkit GSI, GRAM, GridFTP, etc.

Site schedulers and file systems

Clusters and storage systems

Grid2003 Applications To Date

- CMS proton-proton collision simulation
- ATLAS proton-proton collision simulation
- LIGO gravitational wave search
- SDSS galaxy cluster detection
- ATLAS interactive analysis
- BTeV proton-antiproton collision simulation
- SnB biomolecular analysis
- GADU/Gnare genome analysis
- Various computer science experiments

www.ivdgl.org/grid2003/applications

Grid Components

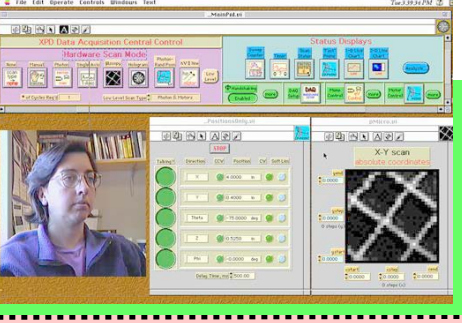
Core Grid Services

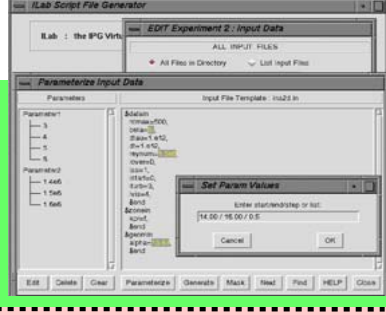
- A common security model for Grid services and Grid applications
  - Provides uniform authentication, authorization, and privacy
  - Basis of the cyber-trust that enables collaboration among the many organizations of a large science project
  - Must support local control of security policy
- Standardized interfaces to computing systems and data storage systems
- Tools and services supporting construction and management of virtual organizations

Core Grid Services

- Services for dynamic construction of execution environments supporting complex distributed applications
  - locating and co-scheduling many combinations of instruments, compute systems, and data archives at multiple locations
- Management of dynamic pools of underlying resources
  - Automatic resource registration and de-registration
  - Resource discovery

Architecture of a Grid





Science Portals

Portal Toolkits (to build application portals)

Applications (Simulations, Data Analysis, etc.)

Grid Services (Workflow management, Visualization, Data Publication/Subscription, Brokering, Job Mg'mt, Data Grid Services, Fault Mg'mt, Grid System Admin., etc.)

Grid Core Functions: Standardized Services and Resources Interfaces

- Grid Information Service
- Uniform Computing Access
- Uniform Data Access
- Unix and OGSA hosting
- Co-Scheduling
- Global Event Services, Auditing, Monitoring
- Collaboration and Remote Instrument Services
- Systems management and access
- Communication Services
- Authentication Authorization
- Security Services

Distributed Resources

(Operational services are provided by Globus, SRB, Condor

clusters

National Supercomputer facilities

Condor pools of workstations

network caches

tertiary storage

scientific instruments

Supernova Observatory

High Speed Communication Services

Early Grid Software

- JINI based Grids
  - Tend to be closed systems
- Legion
  - University of Virginia
  - Strong OO design
  - Single global naming scheme
  - Early advocate for usage of WS protocols
- Unicore
  - Corporate and German government partnership
  - Currently transitioning to an OGSA compliant Grid based on WS-RF

Primary Grid Software

- Condor (Univ. Wisconsin)
  - Cycle-sharing and distributed resource management
  - Supports runtime I/O
  - Condor-G provides a powerful job management interface for Globus based Grids
  - Supports DAG based workflows
- Globus Toolkit (ANL, ISI)
  - Open-source
  - Most recent release based on WS and WS-RF protocols
  - Several independent implementations in several languages, e.g., Java, C, Python, Perl, .Net

Current Globus Software

- The Globus Toolkit v2 (GT2) centers around four key protocols
  - *Security*: Grid Security Infrastructure (GSI)
  - *Resource Management*: Grid Resource Allocation Management (GRAM)
  - *Information Services*: Grid Resource Information Protocol (GRIP)
  - *Data Transfer*: Grid File Transfer Protocol (GridFTP)
- Info Services, Replica Management, etc.

Core Grid Services Security

Grid Security Problem

- Need to support secure communications amongst components
- Must allow local control over security policies
  - Integrate with local mechanisms, do not replace them
- Unique names across a VO
  - Map to local names
- Support single-sign on for ease of use

Authentication

- The process of proving the identity of a person or computer
  - Typically done through the proof of possession of a secret, e.g., a password
  - Your identity is attested to by a trusted outside party
    - Your drivers license can be used as an authentication token because the DMV is trusted

Authorization

- The process of establishing what "rights" an authenticated entity has.
  - With my drivers license I can use a credit card in the name of "Keith Jackson", but not (unfortunately) one in the name of "Deb Agarwal"
  - Current Grids often use simple access control lists to manage authorization
    - The grid-mapfile is an ACL that maps DNs to local accounts
  - Longer term, more of this will be handled by dedicated authorization servers

Confidentiality/Integrity

- Confidentiality is the property that only the authenticated entities can see the data they exchange
- Integrity is the property that the authenticated entities can be guaranteed that the data they exchange has not been modified by anyone

Brief Intro to PKI

- Based on asymmetric cryptography
  - Two keys that are mathematically related are used.
    - Data encrypted with one key can only be decrypted by the other
    - Given one key, it is not computationally feasible to calculate the other
    - The “public key” is published and available to the world
    - The “private key” is kept secret by the user

Certificate Authority

- A trusted third-party that attests to the binding between a public key and unique identifier
  - The unique ID is called a “Distinguished Name” (DN)
    - /C=US/O=Lawrence Berkeley National Lab/CN=Keith R. Jackson
  - A public key signed by a CA is called a certificate
  - CAs provide the root of all trust in PKI based Grids

Proxy Certificates

- A short-lived credential derived from your long-lived identity
  - Can be used equivalently to your long-lived credential
- Useful to support single-sign on
  - Uses an unencrypted private key
    - Allows it to be used without the user typing a passphrase
- Recently became an IETF standard

Transport Layer Security

- TLS is an IETF standard for establishing a secure channel between two parties
  - Uses certificates to authenticate the communicating parties
  - Can provide integrity and/or confidentiality
  - Operates at the transport layer of the network stack
    - Potentially a problem if using a message system that has the ability to route messages separately from the underlying transport

Delegation

- Allows another entity to perform actions on your behalf, e.g.,
  - A job is submitted to a broker which then submits it to a compute resource and transfers the input and output files
- Creates a new proxy credential for the delegatee
  - This proxy is then used for authentication/authorization

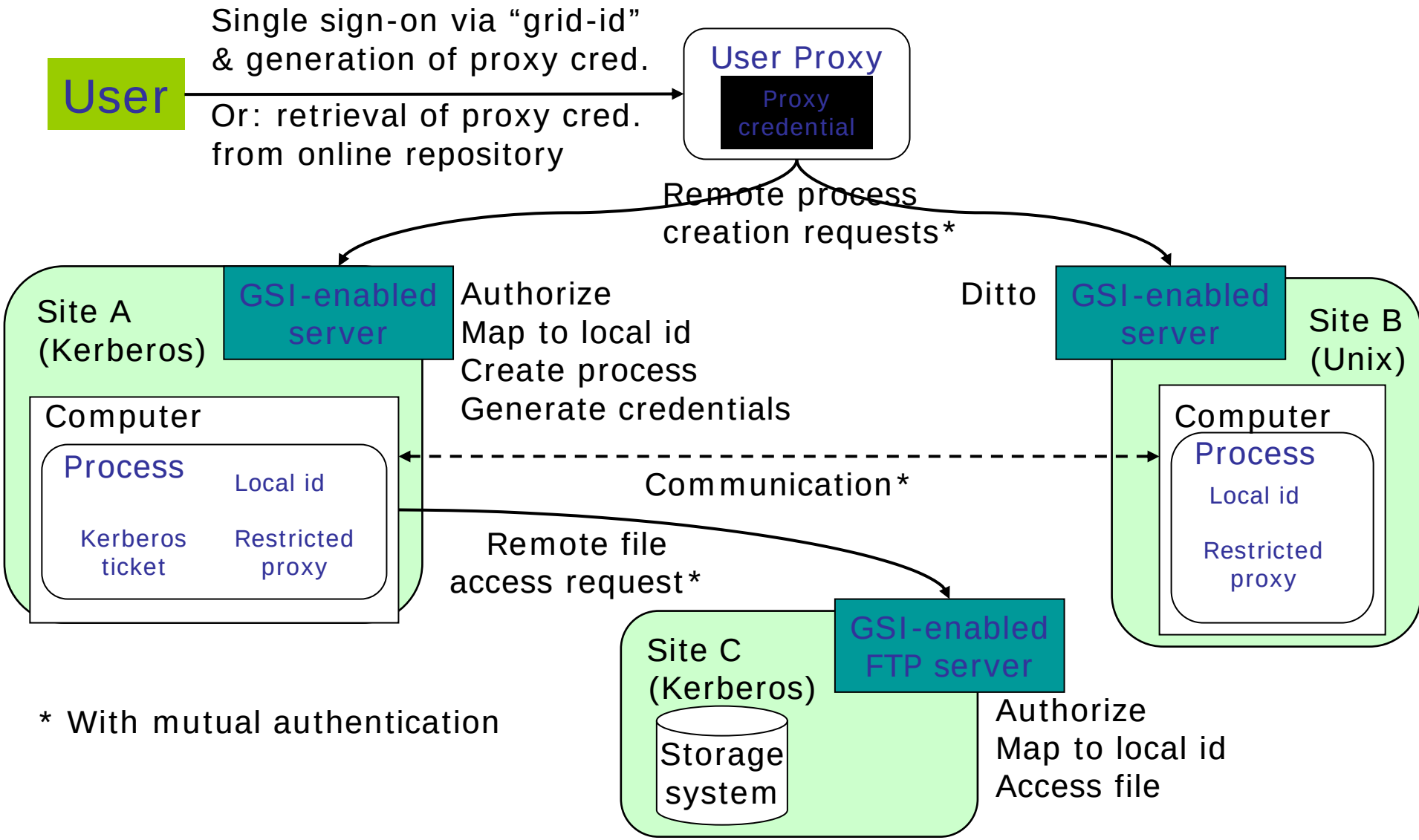
Grid Security Infrastructure (GSI)

- Extensions to standard protocols and APIs
  - Based on Public Key Cryptography
  - Standards: SSL/TLS, X.509 & CA, GSS-API
  - Extensions for single sign-on and delegation
- Globus Toolkit reference implementation of GSI
  - SSLeay/OpenSSL + GSS-API + SSO/delegation
  - Tools and services to interface to local security
    - Simple ACLs; SSLK5/PKINIT for access to K5, AFS; …
  - Tools for credential management
    - Login, logout, etc.
    - Smartcards
    - MyProxy: Web portal login and delegation
    - K5cert: Automatic X.509 certificate creation

Globus Authorization

- Simple Access Control List (ACL) in the grid-map-file
  - Maps DNs to local uid's
- Difficult to maintain for even moderately large deployments
- Hard for VOs to manage community based rights

GSI in Action



Core Grid Services
Resource Management

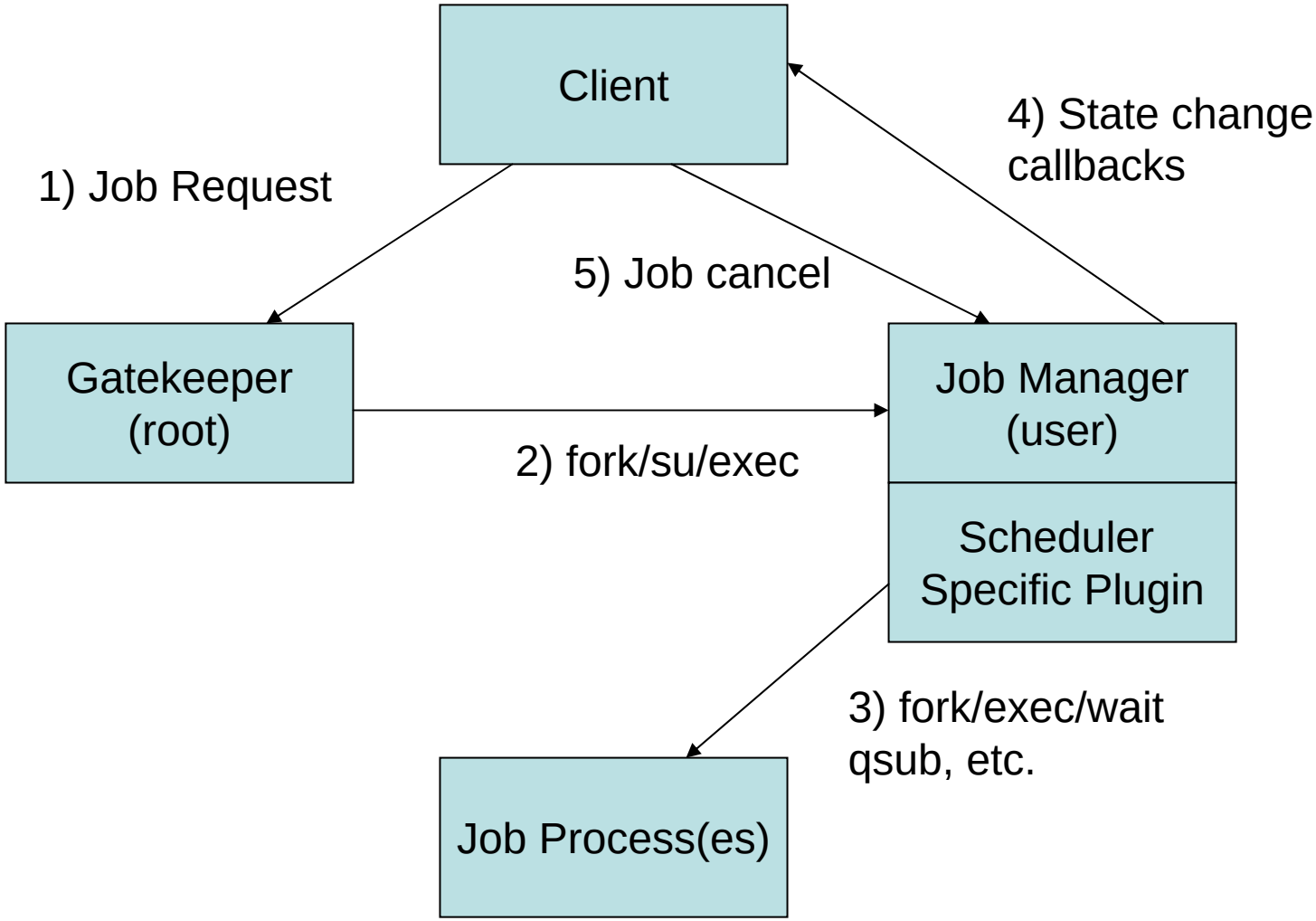
Resource Management Problem

- Submitting and managing jobs remotely is a fundamental problem for the Grid
  - Starting and stopping jobs
  - Interrogating the state of a running job
  - Moving input and output files
  - Cleaning up after the job
  - Interacting with local resource management systems
    - Batch queuing systems

Globus Resource Allocation Manager (GRAM)

- Provides services for starting, querying, and monitoring a job
  - Gatekeeper
  - Job Manager
- Given a Resource Specification Language (RSL) file the job is submitted to the resource
  - Supports many underlying scheduling systems, e.g., PBS, LSF, SGE, Condor, fork, etc.
- Custom HTTP based protocol

GRAM Architecture



Grid Information Service (GIS)

GIS Problem

- System information and monitoring data is critical to operation of the grid and construction of applications
  - What resources are available?
    - Resource discovery
  - What is the “state” of the grid?
    - Resource selection
  - How to optimize resource use?
    - Application configuration and adaptation?
  - Fault Detection, debugging, performance tuning
- We need a general information infrastructure to answer these questions

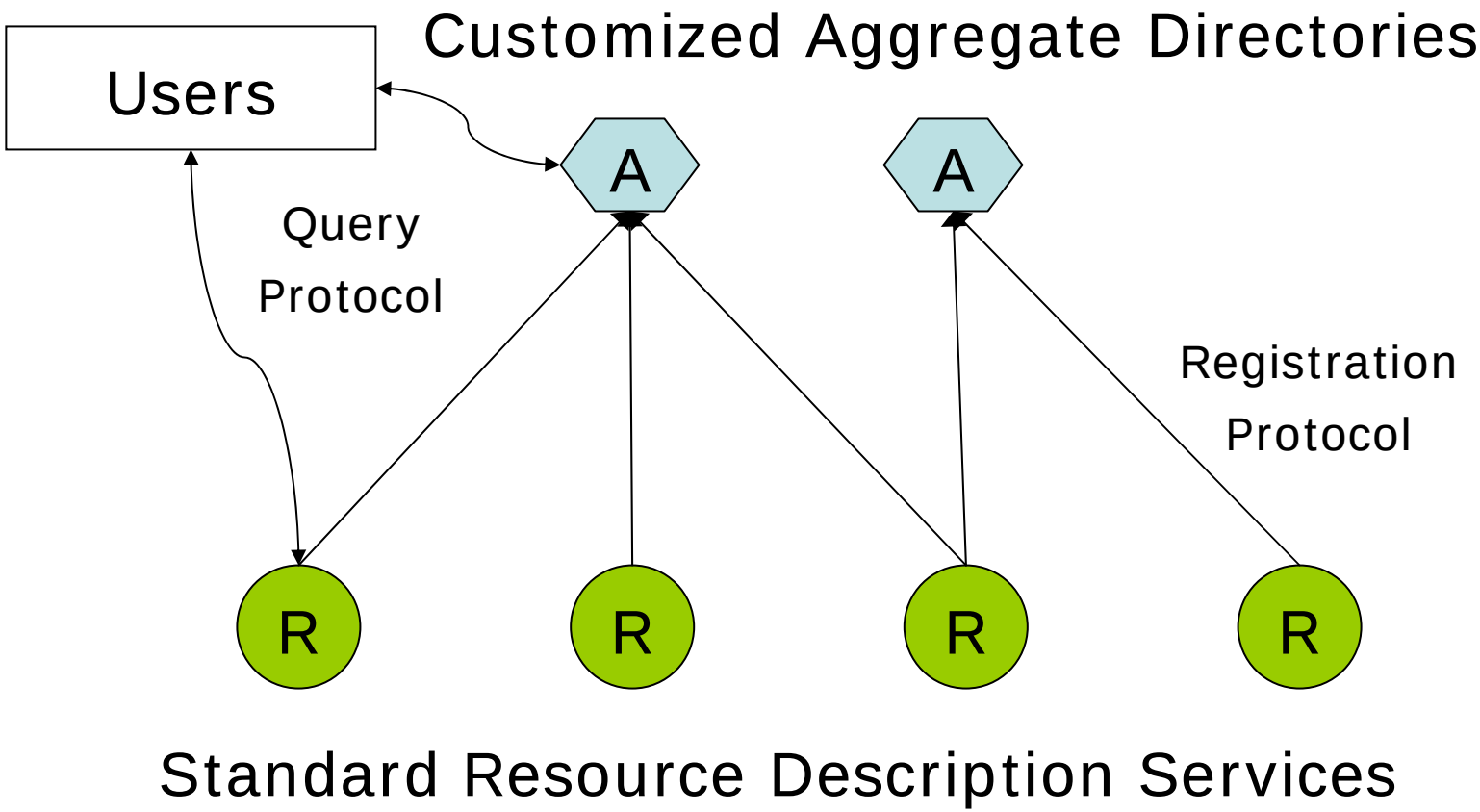
GIS Problem

- Information is always old
  - Time of flight, changing system state
  - Need to provide information quality metrics
- Distributed state hard to obtain
  - Complexity of global snapshot
- Components will fail
- Scalability and overhead Issues
- Many different usage scenarios
  - Some data is cacheable, some must be "real time"

GIS Requirements

- Provide access to static and dynamic information on system components
- Uniform, flexible access to information
- Scalable, efficient access to dynamic data
- Access to multiple information sources
- Decentralized maintenance

GIS Architecture



Data Management

Data Management Problem

- Allow scientists to collaborate over and analyze large data sets
  - Must ensure the integrity of data and derived products
  - Must meet reasonable performance guarantees
    - Must support today's high-speed, high bandwith-delay product networks

Data Management Functionality

- High-speed, reliable access to remote data
- Automated discovery of “best” copy of data
- Manage replication to improve performance
  - Automatic replica creation or migration
- Co-scheduling of compute and storage resources
- Enforce access control on data

Replica Management

- Maintain a mapping between logical names for files and collections and one or more physical locations
- Important for many applications
  - Example: CERN Physics data
    - Multiple petabytes of data per year
    - Copy of everything at CERN (Tier 0)
    - Subsets at national centers (Tier 1)
    - Smaller regional centers (Tier 2)
    - Individual researchers will have copies

GridFTP

- GridFTP
  - An implementation of many optional FTP specifications and other new features to support high-performance data transfer
  - Supports:
    - Parallel streams
    - Striped transfers
    - Third-party transfers
    - Tunable network parameters
    - Proxy certificates
    - Restart markers

Making Grids Today Easier to Use!

Higher-Level Toolkits

- Encapsulate Grid functionality into higher-level interfaces
- Commodity Grid Kit project (CoG)
  - Java CoG
    - Pure Java re-write of the client side Globus APIs
    - Protocol level compatibility with GT2.x
  - Python CoG
    - Wraps the underlying C code and exposes it in Python
    - Provides a series of higher-level OO abstractions to make Grid programming easier

Grid Portals

- Provide a domain-specific web portal that supports the usage of Grid resources
  - Allow scientists to use their browser to interact with the Grid
  - Converging on a technology based on Java Portlets
  - GridSphere and OGCE are two of the main toolkits

Next Generation Grids

Grid Evolution

- GT2 implementations based on a heterogeneous protocol base
  - Used standard protocols where applicable
    - TLS, LDAP, HTTP, etc.
  - Invented new protocols
    - GRAM, GRIP, GSI, GridFTP
- Along the way problems were discovered:
  - Lack of compatibility between versions
    - Protocols kept changing
  - Protocols were mainly only defined in the code
  - Programming interfaces are complex
  - Complex to setup and to run

Grid Evolution

- Ubiquitous adoption demands open, standard protocols
  - Internet and Web as guides
  - Enables innovation/competition on end points
  - Avoid product/vendor lock-in
- At the same time industry was beginning to standardize on Web Service protocols
  - Could provide a common protocol base for the Grid, but … Web services are not sufficient to support Grid computing

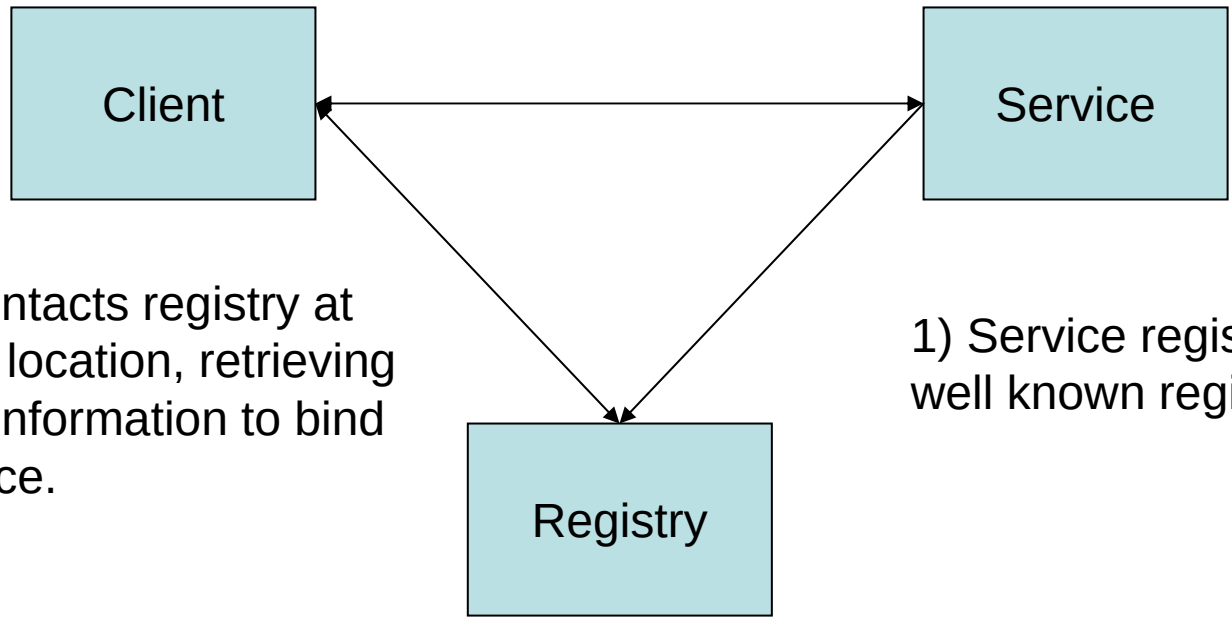
Web Services

What is a Web Service?

- "A Web service is a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols."
  - W3C WS-Arch WG

Service Oriented Architecture

3) Using the returned information the client can interact with the service.



2) Client contacts registry at well known location, retrieving necessary information to bind to the service.

1) Service registers with a well known registry.

XML

- A self-describing text based format for data publication and interchange.
- Provides a framework for creating domain-specific vocabularies.
  - BIOML, IML, AML, etc.
- A simplified version of SGML.
- Standardized by the W3C.
- Most programming languages support standard XML parsing tools.

XML Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Edited with XML Spy v4.2 -->
<note>
        <to imID="steve1">Steve</to>
        <from>Jane</from>
        <heading>Reminder</heading>
        <body>Don't forget me this
   weekend!</body>
</note>
```

WSDL

- Web Services Description Language
  - XML vocabulary for describing the interface and bindings of a Web Service.
  - Can either contain XML Schema or refer to schema documents
  - Defines the messages used by the service
  - Defines the possible operations on a service
  - Clean separation between the interface and the bindings to the interface
    - Can support multiple bindings to a service

SOAP

- The protocol formerly known as: Simple Object Access Protocol
  - XML based protocol for exchanging messages between applications
  - Platform and language independent
  - Consists of three main parts:
    - Envelope defines a framework for describing the contents of a message and how to process it.
    - A set of data encoding rules.
    - A convention for expressing RPC request/response sematics.
  - Typically bound to HTTP for the network transport

WS-Addressing

- Provides transport-neutral mechanisms for addressing Web Services and messages
- Supports message transport through active network intermediaries
  - Firewalls, gateways, etc.
- Not yet standardized.
  - Proposed by Microsoft and IBM

Globus and Web Services

Globus Toolkit and WS

- Web services have some advantages for Grids
  - Standard interface definition
  - Good commercial tooling (eventually)
- **Not a silver bullet or complete solution!!**
  - No standard mechanisms for managing state
  - No notification model
- Globus Alliance working to advance specs ...
  - OGSI/WSRF, OGSA-DAI, WS-Agreement, etc.
  - WSDL 2.0, WSDM, WS-Security, etc.
- … and implementation
  - WS-based interface to existing services
  - New WS-based services
  - Implementations of low-level specifications

Web Services and "State"

- "State" appears in almost all applications
  - Data in a purchase order
  - Current usage agreement for resources
  - Metrics associated with work load on a server
- There are many possible ways Web services might model, access and manage state
  - OGSI v1.0 defined one approach
  - WS-Resource Framework proposes an evolution of that approach
  - Ad-hoc approaches can be used per-application

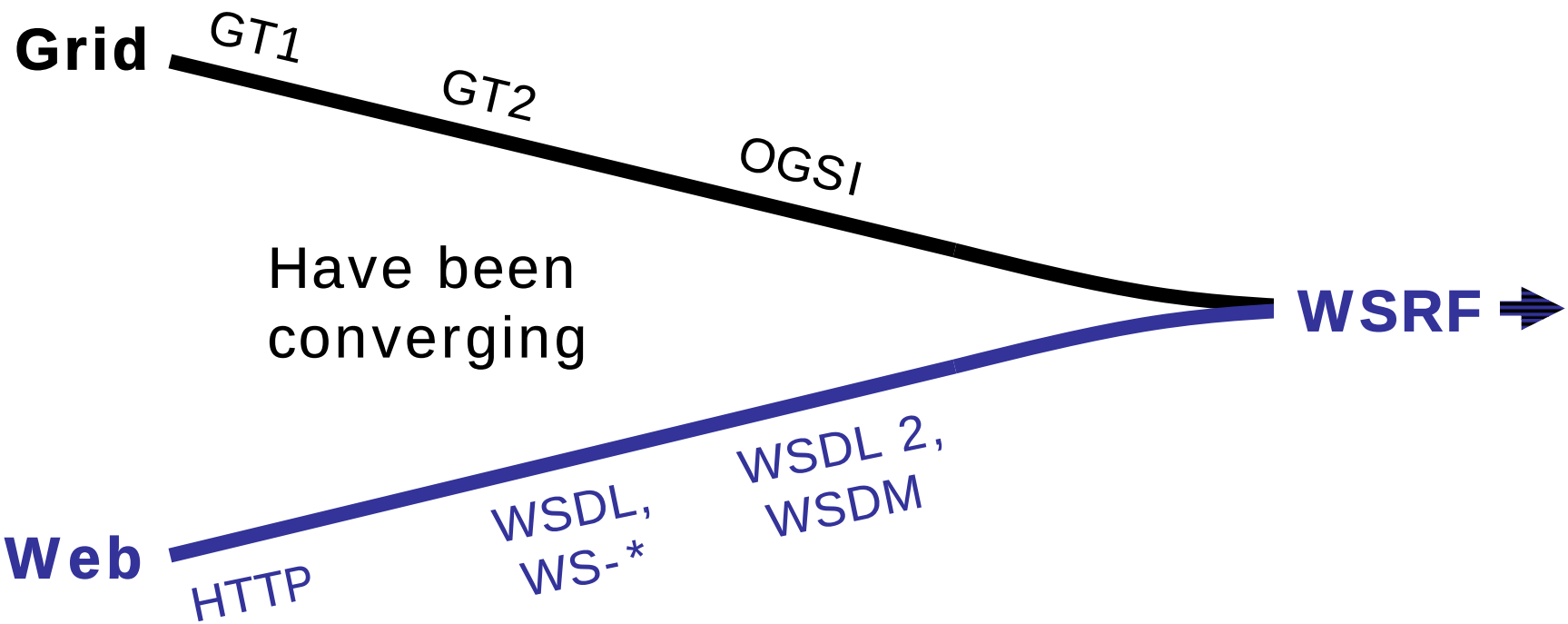
Why Standardize An Approach?

- Building large-scale systems by composition of many heterogeneous components demands that we extract and standardize common patterns
  - Approach to resource identification
  - Resource lifetime management interfaces
  - Resource inspection and monitoring interfaces
  - Base fault representation
  - Service and resource groups
  - Notification
  - And many more…
- Standardization encourages tooling & code re-use
  - Support to build services more quickly & reliably

Open Grid Services Architecture (OGSA)

- Refactor Globus protocol suite to enable common base and expose key capabilities
- Service orientation to virtualize resources and unify resources/services/information
- Embrace key Web services technologies for standard IDL, leverage commercial efforts
- Result: standard interfaces & behaviors for distributed system management: the <u>Grid service</u>

WS-RF and Web Services



The definition of WSRF means that Grid and Web communities can move forward on a common base

WS Resource Framework

- A family of seven Web services specifications
  - A design pattern to specify how to use Web services to access “stateful” components
  - Message-based publish-subscribe to Web services

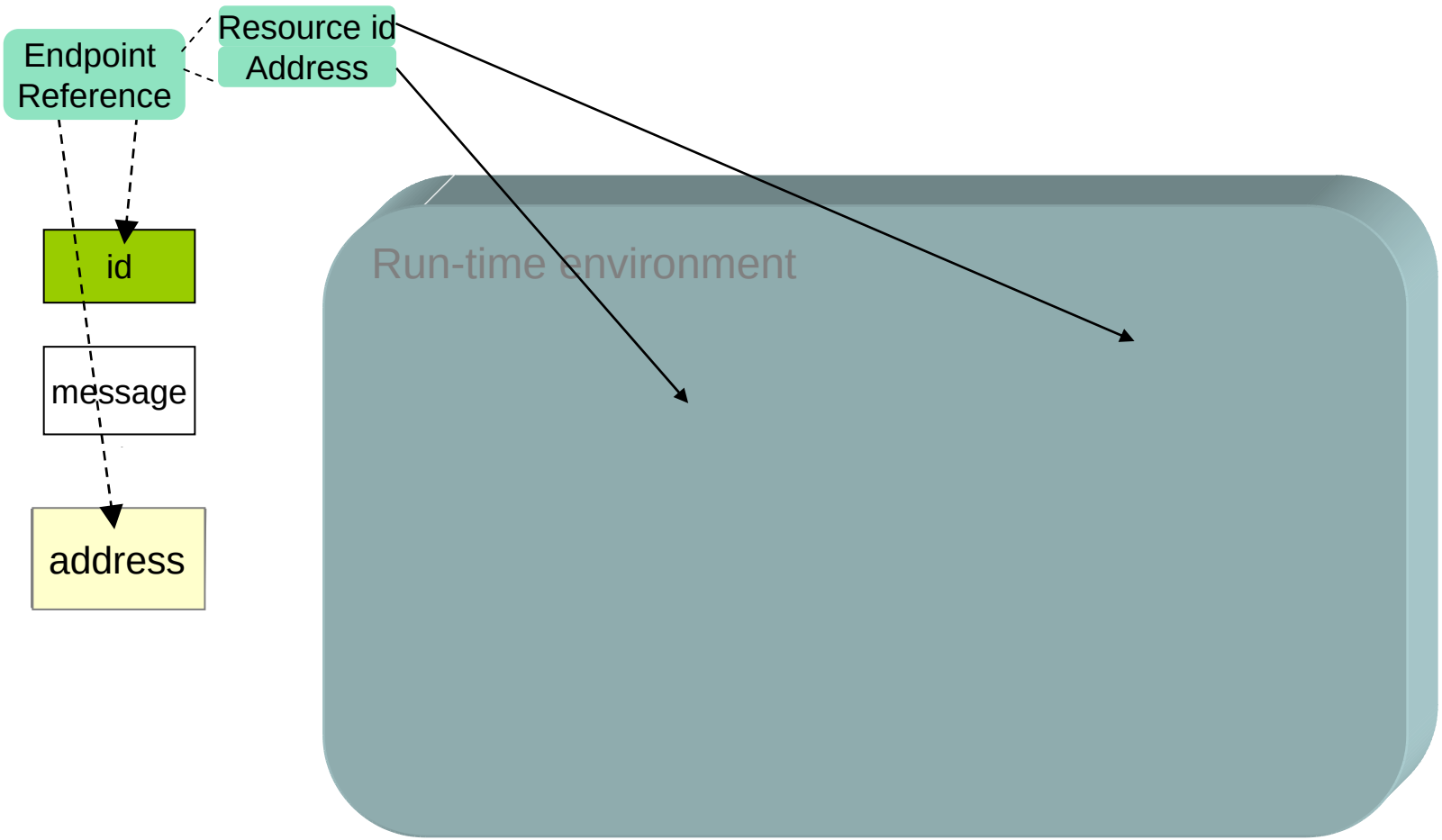
- WS Resource Properties
- WS Resource Lifetime
- WS Service Group
- WS Base Faults
- WS Base Notification
- WS Topics
- WS Brokered Notification

WS-RF and WS-N Overview

- Naming and bindings (basis for virtualization)
  - Every resource can be uniquely referenced, and has one or more associated services for interacting with it
- Lifecycle (basis for fault resilient state management)
  - Resources created by services following factory pattern
  - Resources destroyed immediately or scheduled
- Information model (basis for monitoring & discovery)
  - Resource properties associated with resources
  - Operations for querying and setting this info
  - Asynchronous notification of changes to properties
- Service Groups (basis for registries & collective svcs)
  - Group membership rules & membership management
- Base Fault type

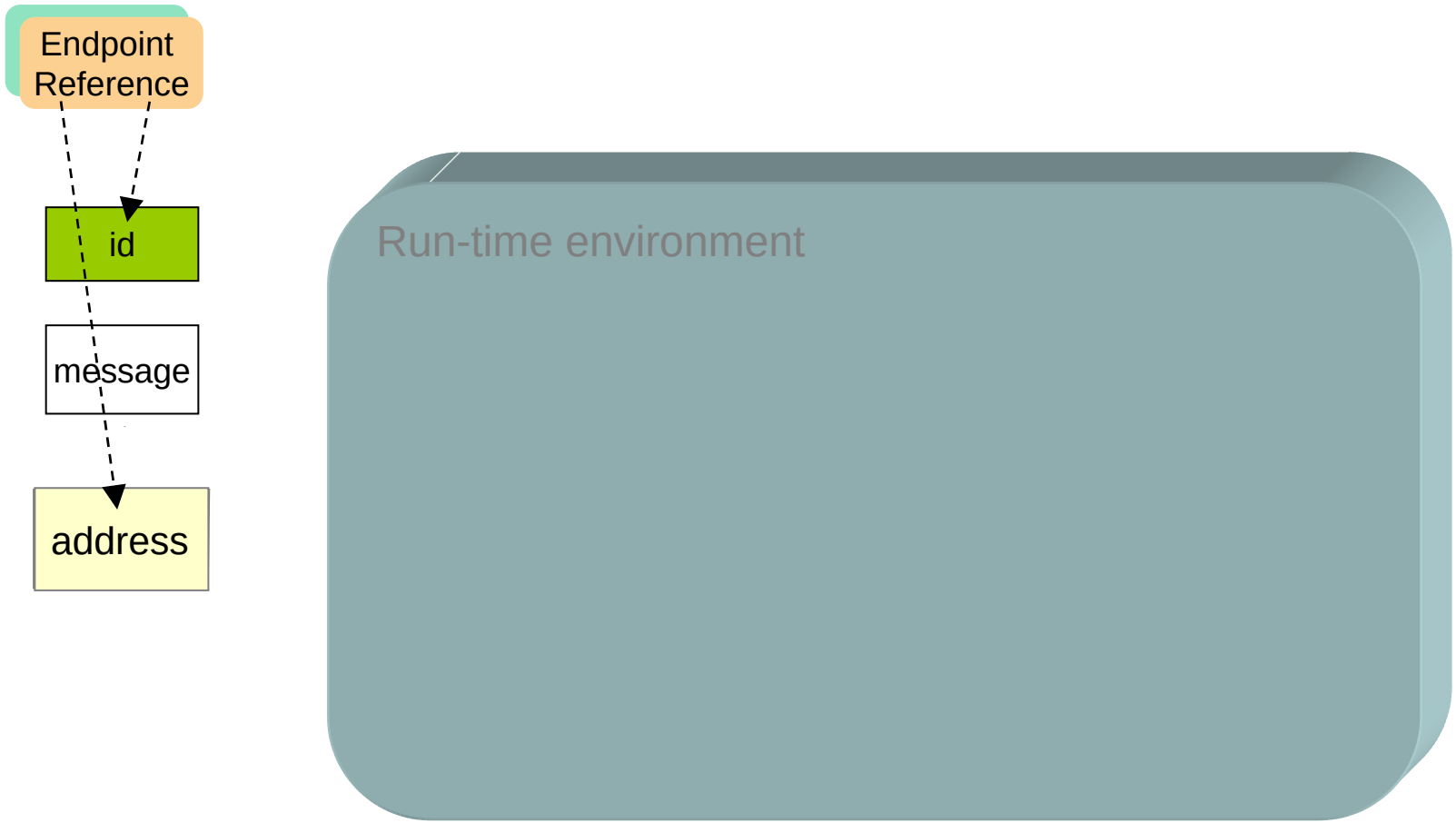
WS-RF Model

Using a Web service to access a resource

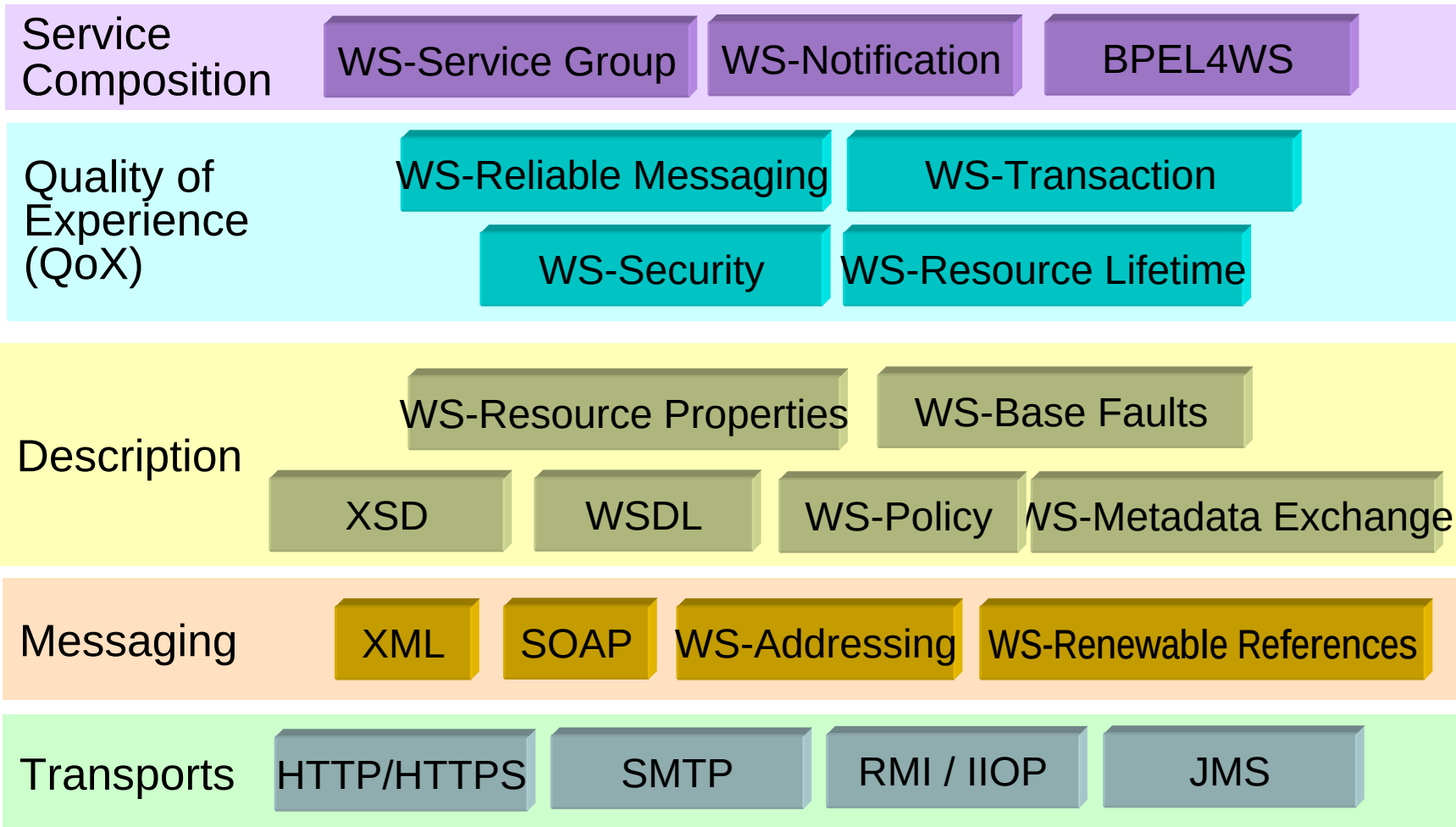


WS-RF Model

Using a Web service to access a resource



WSRF & the Web Services Stack



Next Generation Core Grid Services (GT4)

Grid Security

- Similar to GT2 security
- Updated the proxy certificate format to support the new PKIX standard
- Based on WS-Security specifications
- Support standard callouts for authorization decisions
- Slightly different delegation protocol to lower the delegation overhead
- Introduced message-level security protocols

Message Layer Security

- WS-Secure Messaging
  - Supports the signing of individual SOAP messages
  - Useful if only exchanging a small number of messages
- WS-Secure Conversation
  - Similar to TLS, but done at the application layer instead of the transport layer

Resource Discovery

- GT4 Index Service
  - Information is provided as WS-Resource Properties
  - XPath is used as a query language
  - Supports a pub/sub model using WS-Notification
  - Service Groups used to create aggregating servers

Resource Acquisition/Management

- GT 4 GRAM
  - Provides a single service for starting, querying, and monitoring a job
  - Given a Resource Location Specification (RLS) file the job is submitted to the resource
    - Supports many underlying scheduling systems, e.g., PBS, LSF, SGE, fork, etc.
  - Job properties are represented as a WS-Resource Property document
  - WS-Notification is used to support a pub/sub model for receiving changes

Data Management

- GridFTP
  - New server written from scratch
    - Supports plugable transport layers
- Reliable File Transfer (RFT)
  - Uses GridFTP to reliably transfer files
  - Supports automatic restarts in the face of a variety of faults
  - Can manage 100s to 1000s of submissions
- Replica Management
  - Has not yet been updated

Conclusion

Status of WS-RF Software

- Beta release of GT-4 contains Java, C, and Python implementations of WS-RF
  - Final release in April
- WSRF.Net available from U. of Virgina
- Early version of the Perl implementation available from U. of Manchester
- IBM WebSphere
- Several other companies are working on implementations

What Should I Use?

- WSRF Grids are the future, but are not ready yet
- If working today, hide as much of the underlying changes as possible
  - CoG Kits in Java and Python do this for you
  - Otherwise write abstractions to hide your application from the underlying changes

Questions To Ask

- What is your timeline?
  - If going production in the next year, GT2.x.
  - If later then that, GT4
- What Grid are you running on?
  - Building your own
    - Need to deploy all of the Globus Toolkit
    - Large management overhead
  - Using OSG or other Grid
    - Need to install the appropriate versions of tools

Questions to Ask

- What type of problem do you have?
  - Data management
    - GridFTP from Gt4
    - Reliable File Transfer Service from GT4
    - RLS from GT2 or the Lightweight data replicator from the LIGO project
  - Coordinated resource usage
    - GT2 GRAM
  - MPICH-G2 jobs
    - GT2 GRAM
  - Cycle scavenging
    - Condor
  - Remote instrument control
    - GT4 NEESGrid Teleoperation Control Protocol

More Information

- http://www.globus.org
- http://www.cogkits.org
- http://dsd.lbl.gov/gtg/projects/pyGridWare/
- http://www.cs.virginia.edu/~gsw2c/wsrf.net.html
- http://www.sve.man.ac.uk/Research/AtoZ/ILCT
- http://www-106.ibm.com/developerworks/library/ws-resource/
- http://www.w3schools.com/
- KRJackson@lbl.gov

WSDL Example

- Define messages the service will exchange

```
<wsdl:message
  name="GetEndorsingBoarderRequest">
  <wsdl:part name="body"
  element="esxsd:GetEndorsingBoarder"/>
  </wsdl:message>
<wsdl:message
  name="GetEndorsingBoarderResponse">
  <wsdl:part name="body"
  element="esxsd:GetEndorsingBoarderRespo
  nse"/>  </wsdl:message>
```

WSDL Example (2)

- Abstract definition of the methods

```
<wsdl:portType
  name="GetEndorsingBoarderPortType">
  <wsdl:operation
  name="GetEndorsingBoarder">
  <wsdl:input
  message="es:GetEndorsingBoarderRequest"/>
  <wsdl:output
  message="es:GetEndorsingBoarderResponse">
<wsdl:fault
  message="es:GetEndorsingBoarderFault"/>
  </wsdl:operation>  </wsdl:portType>
```

WSDL Example (3)

- Concrete binding of the abstract definition

```
<wsdl:binding
  name="EndorsementSearchSoapBinding"
  type="es:GetEndorsingBoarderPortType">
  <soap:binding style="document"
  transport="http://schemas.xmlsoap.org/soap
  /http"/>    <wsdl:operation
  name="GetEndorsingBoarder">
  <soap:operation
  soapAction="http://www.snowboard-
  info.com/EndorsementSearch"/>
```

WSDL Example (4)

- Finally bind to a network address

```
<wsdl:service
 name="EndorsementSearchService">
 <wsdl:port
 name="GetEndorsingBoarderPort"
 binding="es:EndorsementSearchSoapBindin
 g">
 <soap:address
 location="http://www.snowboard-
 info.com/EndorsementSearch"/>
 </wsdl:port>  </wsdl:service>
 </wsdl:definitions>
```

SOAP Example Request

```
<SOAP-ENV:Envelope  xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/e
nvelope/"  SOAP-
ENV:encodingStyle="http://schemas.xmls
oap.org/soap/encoding/">  <SOAP-
ENV:Body>    <m:GetEndorsingBoarder
xmlns:m="http://namespaces.snowboard-
info.com">
<manufacturer>K2</manufacturer>
<model>Fatbob</model>
</m:GetEndorsingBoarder>  </SOAP-
ENV:Body></SOAP-ENV:Envelope>
```

SOAP Example Response


```
<SOAP-ENV:Envelope  xmlns:SOAP-
  ENV="http://schemas.xmlsoap.org/soap/en
  velope/"  SOAP-
  ENV:encodingStyle="http://schemas.xmlso
  ap.org/soap/encoding/">  <SOAP-
  ENV:Body>
  <m:GetEndorsingBoarderResponse
  xmlns:m="http://namespaces.snowboard-
  info.com">      <endorsingBoarder>Chris
  Englesmann</endorsingBoarder>
  </m:GetEndorsingBoarderResponse>
  </SOAP-ENV:Body></SOAP-ENV:Envelope>
```

Grid Troubleshooting and Performance Analysis

Brian L. Tierney
BLTierney@lbl.gov

Distributed Systems Department
Lawrence Berkeley National Laboratory

The Problem

- Assume a Grid job is:
  - submitted to a resource broker, uses a reliable file transfer service to copy several files, then runs the job.
- This normally takes 15 minutes to complete. But…
  - two hours have passed and the job has not yet completed.
- What, if anything, is wrong?
  - Is the job still running or did one of the software components crash?
  - Is the network particularly congested?
  - Is the CPU particularly loaded?
  - Is there a disk problem?
  - Was a software library containing a bug installed somewhere?

Grid Debugging Issues

- Multiple Log file formats
- Inconsistent logging levels
- What resources / hosts are being used?
  - Grid's partially hide this

- Naïve Solution
  - Log on to each host that is part of your Grid job (if possible)
  - locate the log files
  - grep for error messages

- Very Tedious process!!

Better Solution

- End-to-End Monitoring
  - All components between the application endpoints are instrumented and monitored.
- This includes:
  - software (e.g., applications, services, middleware, operating systems)
  - end-host hardware (e.g., CPUs, disks, memory, network interface)
  - networks (e.g., routers, switches, or end-to-end paths)

Key Components

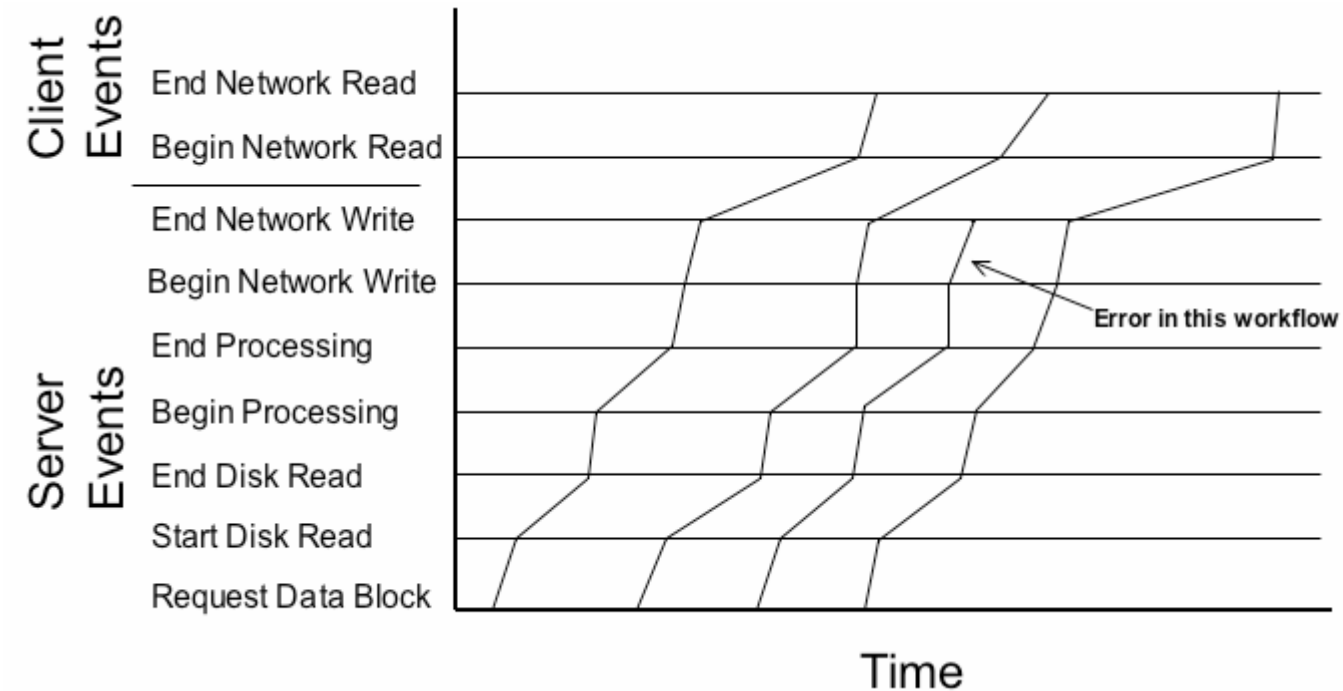
- Instrumented Applications and Middleware
  - Preferably with a common log format
- Time Synchronized Hosts
  - Run NTP everywhere (www.ntp.org)
- Log file collect / merge / sort / visualization tools
- Example solution:
  - DSD NetLogger Toolkit
    - Over 10 years of development and use

NetLogger Methodology

- NetLogger is both a **methodology** for analyzing distributed systems, and a **set of tools** to help implement the methodology.
  - You can use the NetLogger methodology without using any of the LBNL provided tools.
- The NetLogger methodology consists of the following:
  1. All components must be instrumented to produce monitoring These components include application software, middleware, operating system, and networks. The more components that are instrumented the better.
  2. All monitoring events must use a common format and common set of attributes and a globally synchronized timestamp
  3. Log all of the following events: Entering and exiting any program or software component, and begin/end of all IO (disk and network)
  4. Collect all log data in a central location
  5. Use event correlation and visualization tools to analyze the monitoring event logs

NetLogger Analysis: Key Concepts

- NetLogger visualization tools are based on time correlated and object correlated events.
- If applications specify an “object ID” for related events, this allows the NetLogger visualization tools to generate an object “**lifeline**”
- In order to associate a group of events into a “lifeline”, you must assign an “object ID” to each NetLogger event
  - Sample Event ID: file name, block ID, frame ID, Grid Job ID, etc.



Sample Instrumentation

```
log = netlogger.LogOutputStream(“my.log”)
done = 0
while not done:
    log.write("EVENT.START",{"TEST.SIZE”:size})
    # perform the task to be monitored
    done = do_something(data,size)
    log.write("EVENT.END”,{})
```

- Sample Event:

```
t DATE=20000330112320.957943
s HOST=gridhost.lbl.gov
s PROG=gridApp
l LVL=Info
s NL.EVNT=WriteData
l SEND.SZ=49332
```

How to Instrument Your Application

- You'll probably want to add instrumentation code to the following places in your distributed application:
  - before and after all disk I/O
  - before and after all network I/O
  - entering and leaving each distributed component
  - before and after any significant computation
    - e.g.: an FFT operation
  - before and after any significant graphics call
    - e.g.: certain CPU intensive OpenGL calls
- This is usually an iterative process
  - add more instrumentation points as you zero in on the problem

Activating Instrumentation

- Can easily get swamped with instrumentation data
- Should be easy to turn instrumentation ON/OFF
  - use shell environment variables, command line options, etc
- Often will want different “levels” of instrumentation
  - Errors, Debug, Tuning, etc
- Suggested default log level:
  - component enter/exit
  - error messages

Troubleshooting Example: Step 1: Generate Grid Job "Lifeline"

Application Events

Copy output data

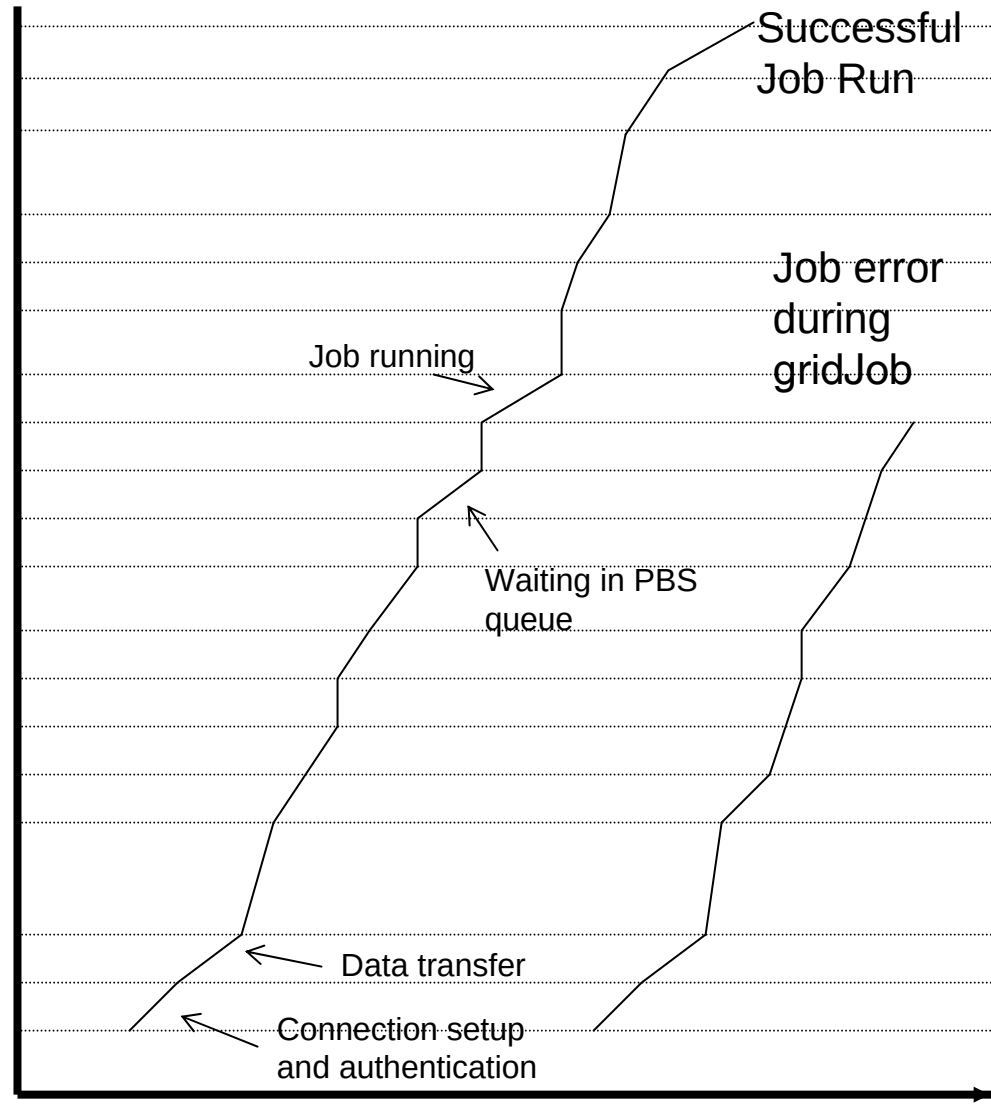
GlobusUrlCopy.put.end
GlobusUrlCopy.put.transferStart
GlobusUrlCopy.put.start

Run Grid Job

GlobusJobRun.end
jobManager.end
jobManger.jobState.done
gridJob.end
gridJob.start
jobManager.jobState.active
jobManager.jobState.pending
akentiAuthorization.end
akentiAuthorization.start
gateKeeper.end
jobManager.start
gateKeeper.start
GlobusJobRun.start

Copy input data

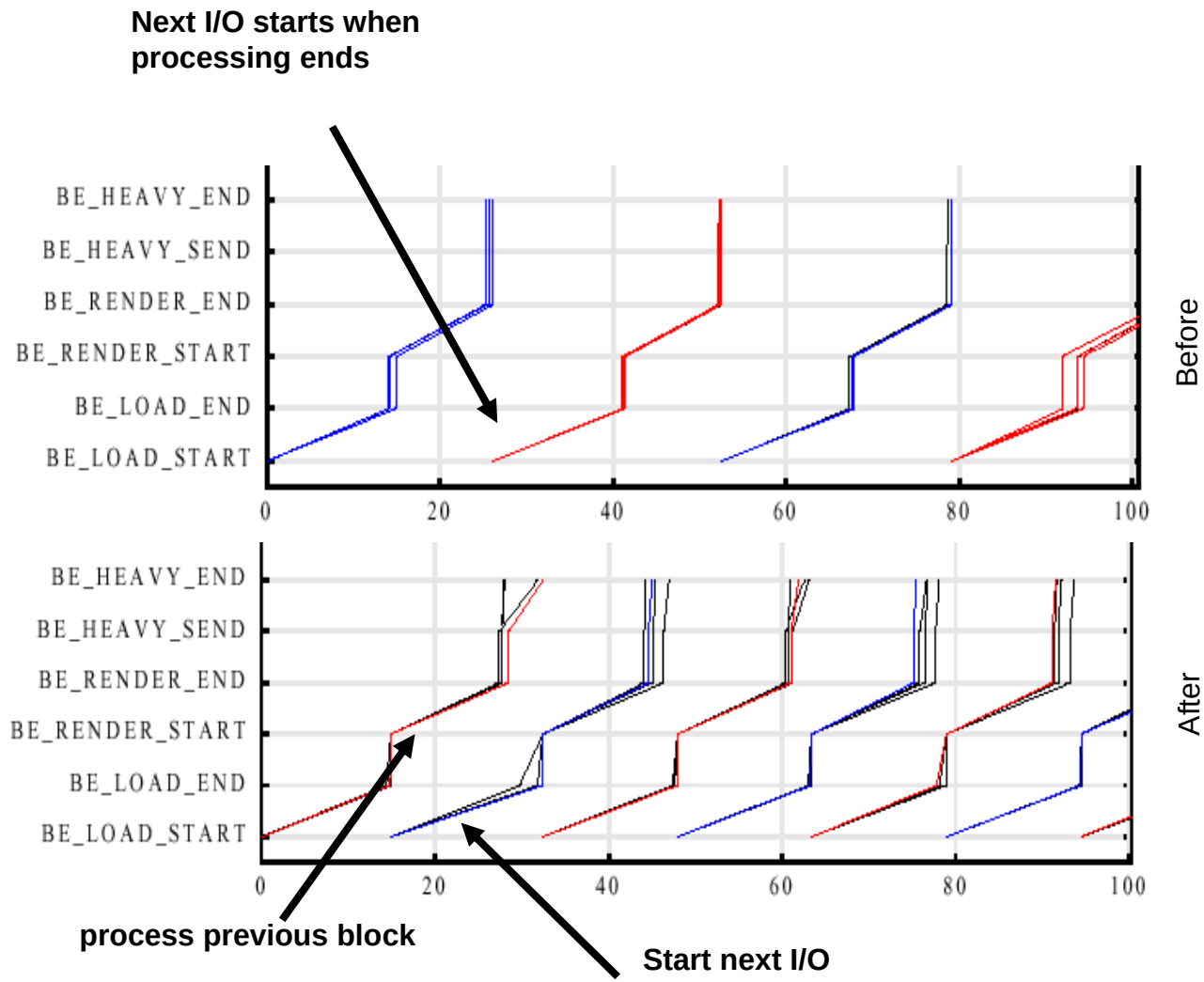
GlobusUrlCopy.get.end
GlobusUrlCopy.get.transferStart
GlobusUrlCopy.get.start



Time

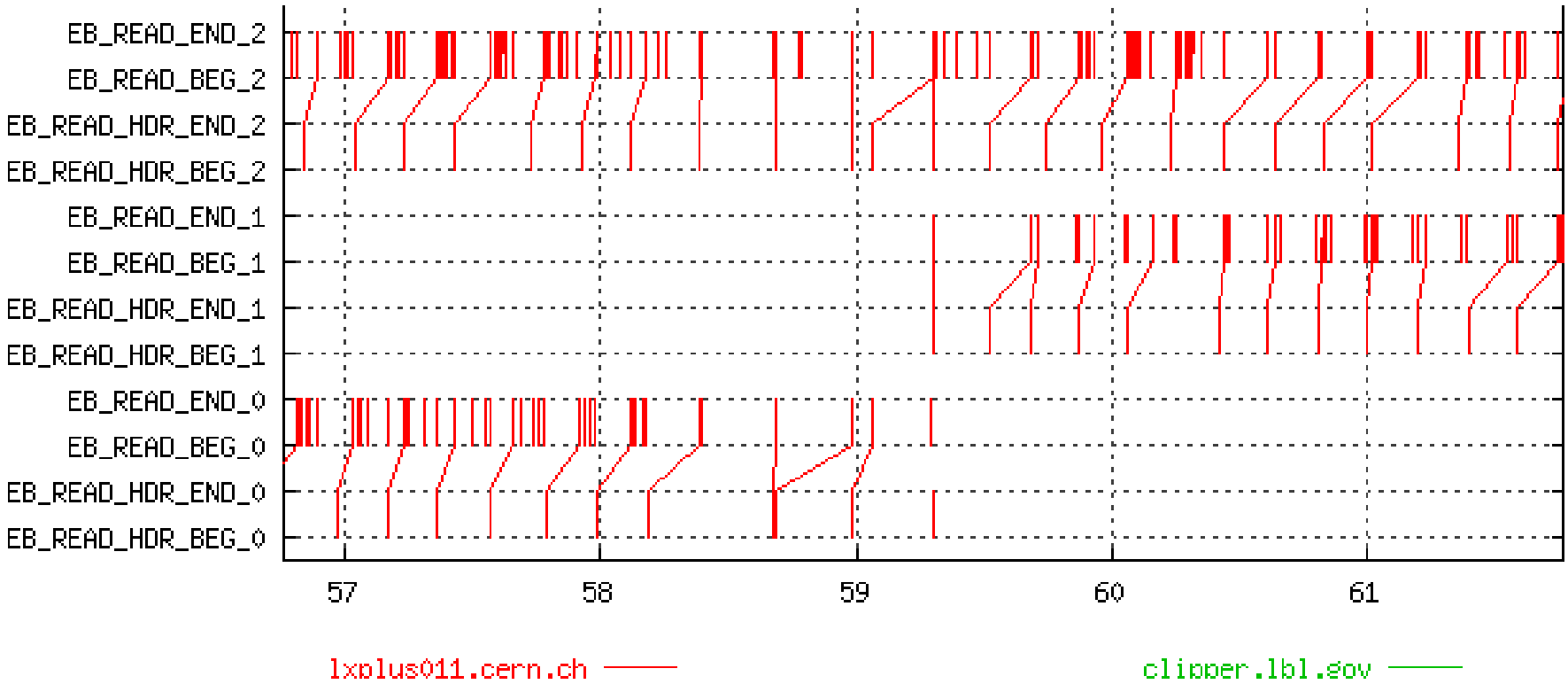
Step 2: Add detailed application instrumentation

- I/O followed b
  processing
- overlapped I
  processing

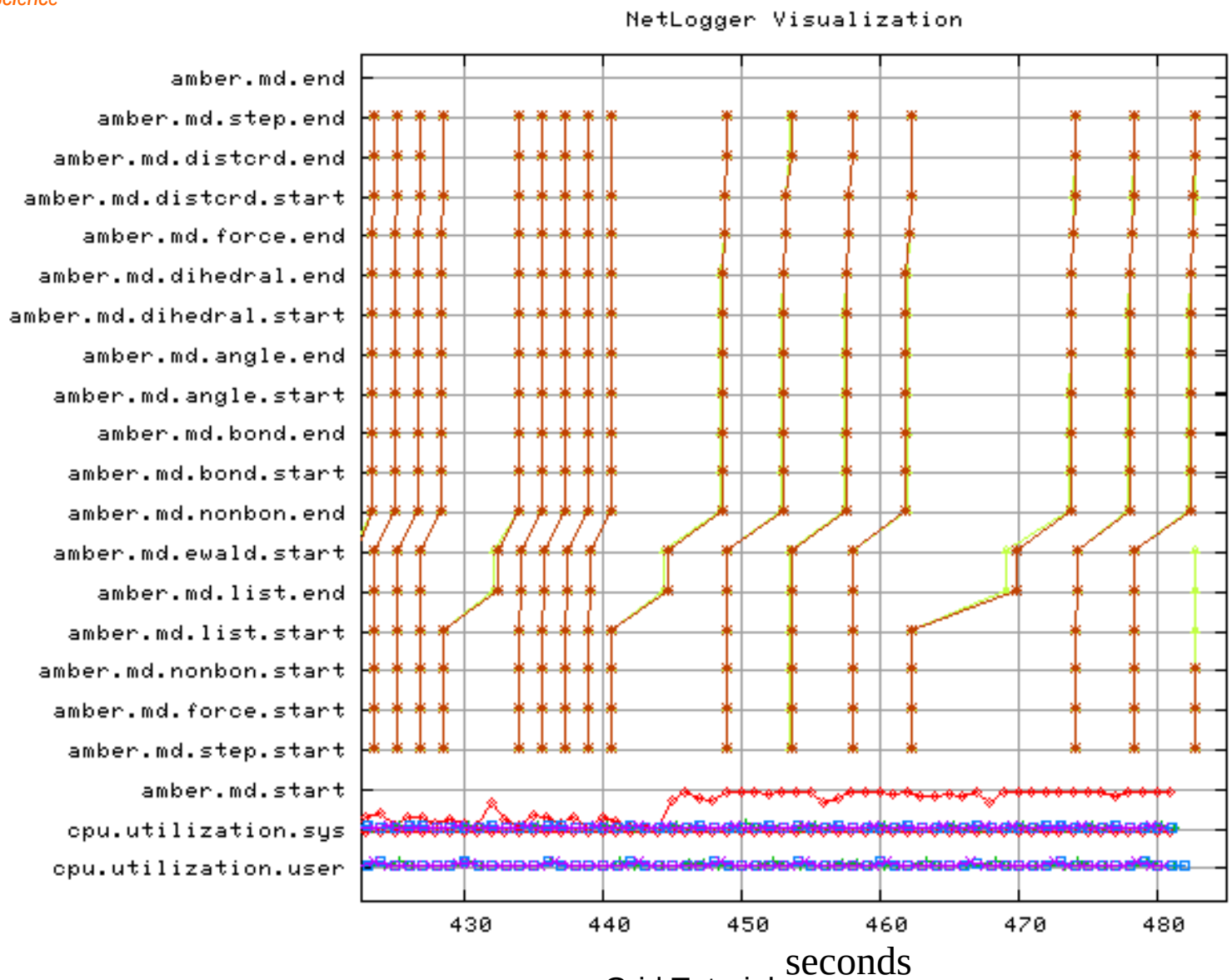


almost a 2:1 speedup

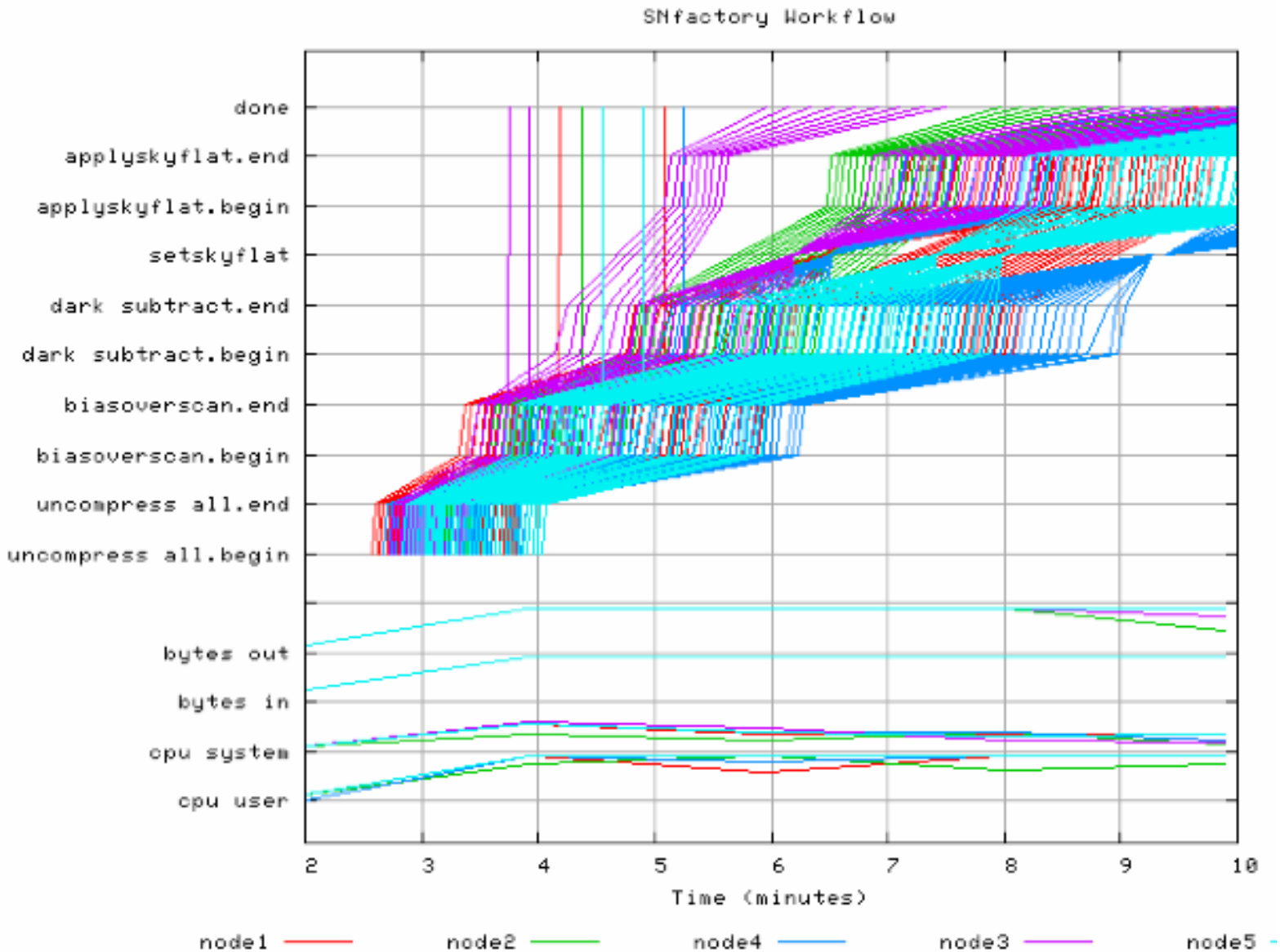
Step 2: 2nd example Parallel Stream Bug



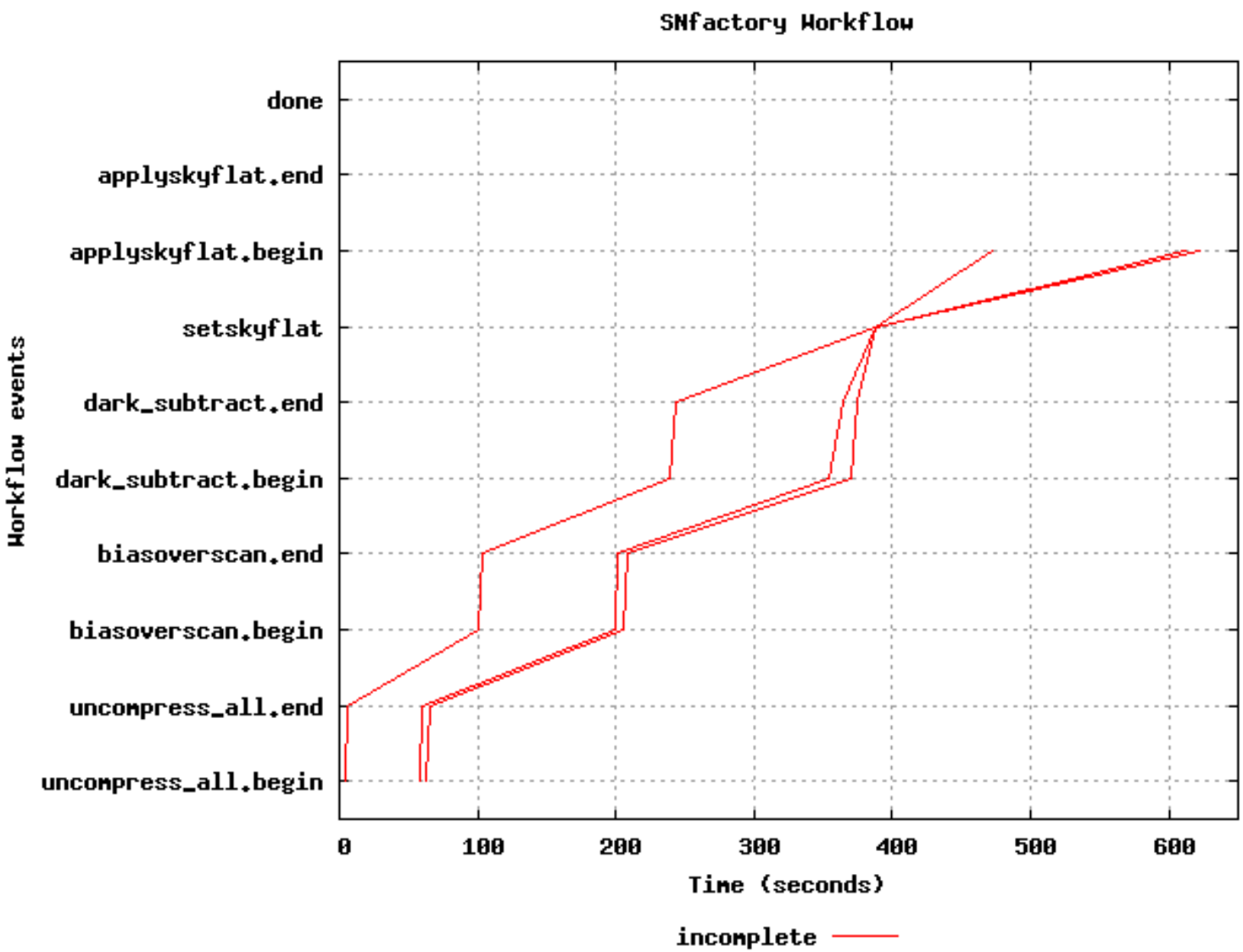
Step 3: Add host monitoring (e.g.: CPU load)



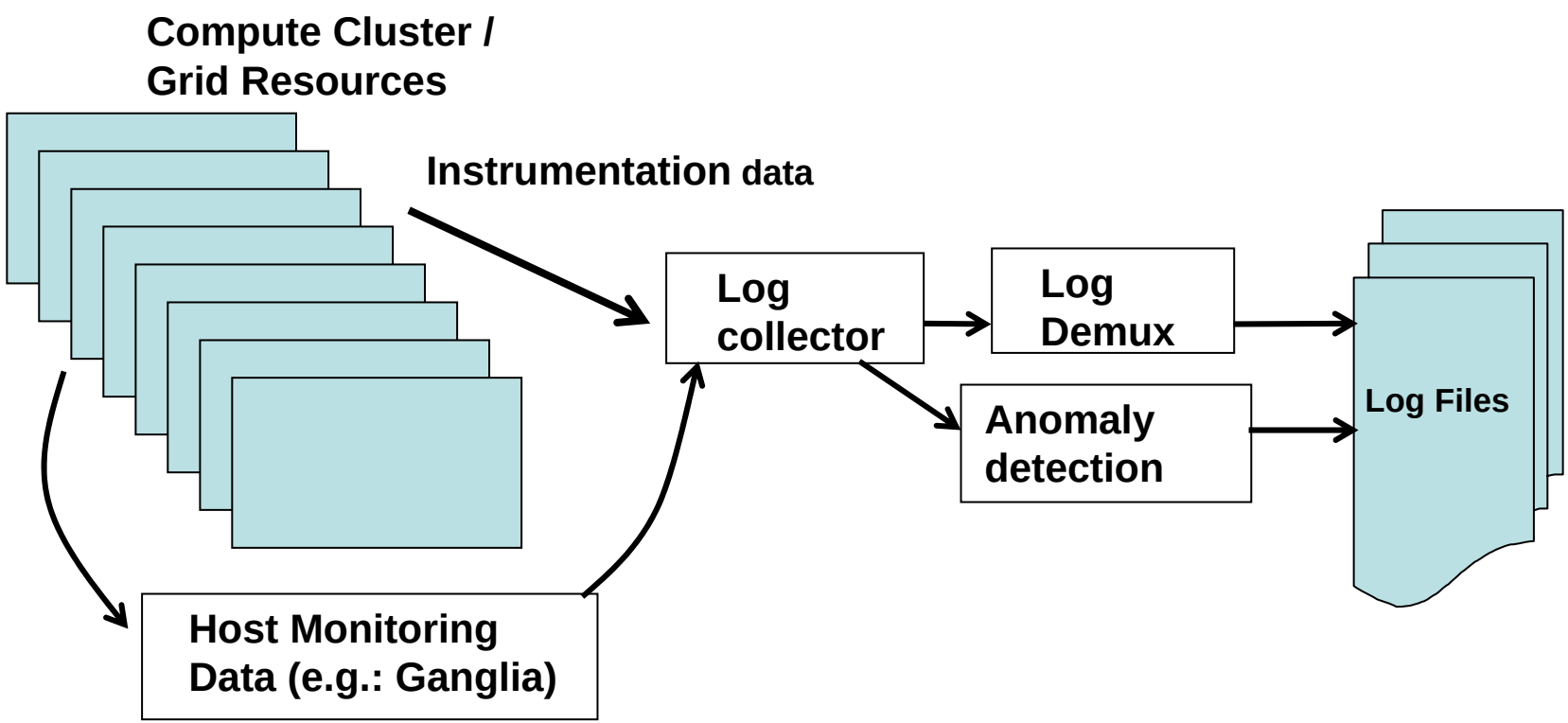
Issue: Too Much Data



Solution: Lifeline Anomaly Detection



Sample Data Collection



Other Tools

- Instrumentation
  - Log4j : Java logging package
  - Python logger : python logging package
- Host / Cluster Monitoring
  - Ganglia (http://ganglia.sourceforge.net/)
- Log collection
  - MonALISA (http://monalisa.cacr.caltech.edu/)
- Visualization Tools
  - Gnuplot, “R”, Chainsaw (log4j), excel

Steps Needed for Grid Debugging

- Add instrumentation code to software components
- Configure NTP
- Install Ganglia (optional)
- Install log file collection tools
  - Syslog, netlogd, etc.
- Log file analysis tools
  - E.g.: anomaly detection
- Extract data for visualization tools
  - Extract data for gnuplot, R, matlab, excel, etc.
- Visualize / Analyze Data
  - requires tool configuration

Conclusions

- **End-to-end, top-to-bottom instrumentation is essential**
  - Need ability to “drill down” from high level view into each component
- **Other important requirements**
  - Central collection point
  - Common data format
  - Data analysis tools
- **Instrumenting user applications is essential**
  - Often applications are the source of the problems

For More Information

- http://dsd.lbl.gov/NetLogger/
  - All software components are available for download under DOE/LBNL open source license (BSD-style)

- email: BLTierney@LBL.GOV