Grids and Clouds Interoperation: Development of e-Science Applications Data Manager on Grid Application Platform

WeiLong Ueng
Academia Sinica Grid Computing
wlueng@twgrid.org
Outline

• Introduction to GAP (Grid Application Platform).
• Principles of e-Science Distributed Data Management
• Putting it to Practice
• GAP Data Manager Design
• Summary
Grid Application Platform (V3.1.0)

• Grid Application Platform (GAP) is a grid application framework developed by ASGC. It provides a vertical integration for developers and end-users

– In our aspects, GAP should be

• Easy to use for both end-users and developers.

• Easy to extend for adopting new IT technologies, the adoption should be transparent to developers and users.

• Light-weight in terms of the deployment effort and the system overhead.
The layered GAP architecture

Reduce the effort of developing application services
Reduce the effort of adapting new technologies
Concentrate efforts on applications
Advantages of GAP

• Through GAP, you can be a
  • Developer
    – Reduce the effort of developing application services.
    – Reduce the effort of adopting new distributed computing technologies.
    – Concentrate efforts on implementing application in their domain.
    – Client can be developed by any Java-based technologies.
  • End-user
    – Portable and light-weight client.
    – User can run their grid-enabled application as simple as using a desktop utility.
Features

- Application-oriented approach focuses developers effort on domain-specific implementations.
- Layered and modularized architecture reduces the effort of adopting new technology.
- Object-oriented (OO) design prevents repeating tedious but common works in building application services.
- Service-oriented architecture (SOA) makes the whole system scalable.
- Portable thin client gives the possibility to access the grid from end-users desktop.
The GAP (Before V3.1.0)

• Can’s
  • simplify User and Job management as well as the access to the Utility Applications with a set of well-defined APIs
  • interface different computing environments with customizable plug-ins

• Cannot’s
  • simplify Data management
Why?

• Distributed data management is a hard problem
• There is no one-size-fits-all solution (otherwise Condor/Globus/gLite/ yourfavoritegrid would’ve done it!)
• Solutions exist for most individual problems (learn from RDBMS or P2P community)
• Integrating everything into an end-to-end solution for a specific domain is hard and ongoing work
• Many open problems!!
  • ..and not enough people..
Data Intensive Sciences depend on Grid Infrastructures

Characteristics: any one of the following

- Data is inherently distributed
- Data is produced in large quantities
- Data is produced at a very high rate
- Data has complex interrelations
- Data has many free parameters
- Data is needed by many people

A single person / computer alone cannot do all the work.
Several Groups Collaborating in Data Analysis
The Data Flood

- Instrument data
  - Satellites
  - Microscopes
  - Telescopes
  - Accelerators
  - ..

- Simulation data
  - Climate
  - Material science
  - Physics, Chemistry
  - ..

- Imaging Data
  - Medical imaging
  - Visualizations
  - Animations
  - ..

- Generic Metadata
  - Description data
  - Libraries
  - Publications
  - Knowledge base
  - ..
High-Level Data Processing Scenario

Data Source

Preprocessing
- Formatting
- Data descriptors

Storage
- Security

Distributed Data Management

Distribution
- Transfer
- Replication
- Caching

Analysis
- Computation
- Workflows

Science Data

Science Library
- Indexing

Interpretation
- Publications
- Knowledge
- New ideas

Academia Sinica Grid Computing
High-Level Data Processing Scenario

Distributed Data Management

Data Source → Preprocessing • Formatting • Data descriptors → Storage • Security → Distribution • Transfer • Replication • Caching → Analysis • Computation • Workflows → Science Data

Science Library • Indexing → Interpretation • Publications • Knowledge • New ideas

COMPLEXITY
Principles of Distributed Data Management

- Data and computation co-scheduling
- Streaming
- Caching
- Replication
Co-Scheduling: Moving computation to the data

- Desirable for very large input data sets
- Conscious manual data placement based on application access patterns
- Beware: Automatic data placement is domain specific!
Complexities

• It is a good idea to keep the large amounts of data local to the computation
• Some data cannot be distributed
• Metadata stores are usually central

Combination of all of the above
Accessing Remote Data:

**Streaming** data across the wide area

- Avoid intermediary storage issues
- Process data as it comes
- Allow multiple consumers and producers
- Allow for computational steering and visualization
Caching data in local data caches

- Improve access rate for repeated access
- Avoid multiple wide area downloads
Distributing Data: Replication

Data is **replicated** across many sites in a Grid

- Keeping Data close to Computation
- Improving throughput and efficiency
- Reduce latencies
File Transfer

• Most Grid projects use GridFTP to transfer data over the wide area

• Managed transfer services on top:
  • Reliable GridFTP
  • gLite File Transfer Service
  • CERN CMS experiment’s Phedex service
  • SRM copy

• Management achieved by
  • Transfer Queues
  • Retry on failure

• Other Transfer Mechanisms (and example services):
  • http(s) (slashgrid, SRM)
  • UDP (SECTOR)
Putting it to Practice

• Trust
• Distributed file management
  • Distributed Cluster File Systems
  • The Storage Resource Manager interface
    • dCache, SRB, NeST, SECTOR
  • Clouds File System
    • HDFS
• Distributed database management
File System

- Distributed File Systems
- Managed, Reliable Transfer Services
- Distributed Caching and P2P Systems

Transfer Protocols: FTP, http, GridFTP, scp, etc..

Storage
Trust

Trust goes both ways

• Site policies:
  • Trace what users accesses what data
  • Trace who belongs to what group
  • Trace where requests for access come from
  • Ability to block and ban users

• VO policies:
  • Store sensitive data in encrypted format
  • Managing user and group mappings at VO level
File Data Management

- Distributed Cluster File Systems
  - Andrew File System AFS, Distributed GPFS, Lustre

- Storage Resource Manager SRM interface to File Storage
  - Several implementations exist: dCache, BeStMan, CASTOR, DPM, StoRM, Jasmine, Storage Resource Broker SRB, Condor NeST..

- Other File Storage Systems
  - iRODS, SECTOR, .. (many many more)
Managed Storage Systems

**Basics**
- Stores data in the order of Petabytes
- Total-throughput scales with the size of the installation
- Supports several hundreds to thousands of clients
- Adding / removing storage nodes w/o system interruption
- Supports posix-like access protocols
- Supports wide area data transfer protocols

**Advanced**
- Supports quotas or space reservation, data lifetime
- Drives back-end tape systems (generates tape copies, retrieves non cached files)
- Supports various storage semantics (temporary,
Storage Resource Manager Interface

- SRM is an **OGF interface standard**
- One of the few interfaces where **several** implementations exist (>5)

**Main features**

- **Prepares for data transfer** (not transfer itself)
  - Transparent management of hierarchical storage backends
  - Make sure data is accessible when needed: Initiate restore from nearline storage (tape) to online storage (disk)
- Transfer between SRMs as managed transfer (**SRM copy**)
- **Space reservation** functionality (implicit and explicit via space tokens)
Storage Resource Manager Interface

**SRM v2.2 interface supports**

- Asynchronous interaction
- **Temporary, permanent and durable** file and space semantics
  - Temporary: no guarantees are taken for the data (scratch space or `/tmp`)
  - Permanent: strong guarantees are taken for the data (tape backup, several copies)
  - Durable: guarantee until used: permanent for a limited time
- Directory functions including file listings.
- Negotiation of the actual data transfer protocol.
Hadoop File System (HDFS)

- Highly fault-tolerant
- High throughput
- Suitable for applications with large data sets
- Streaming access to file system data
- Can be built out of commodity hardware
HDFS Architecture
File system Namespace

• Hierarchical file system with directories and files
• Create, remove, move, rename etc.
• Namenode maintains the file system
• Any meta information changes to the file system recorded by the Namenode.
• An application can specify the number of replicas of the file needed: replication factor of the file. This information is stored in the Namenode.
Data Replication

- HDFS is designed to store very large files across machines in a large cluster.
- Each file is a sequence of blocks.
- All blocks in the file except the last are of the same size.
- Blocks are replicated for fault tolerance.
- Block size and replicas are configurable per file.
- The Namenode receives a Heartbeat and a BlockReport from each DataNode in the cluster.
- BlockReport contains all the blocks on a Datanode.
Hadoop properties

- NOT for Wide Area yet
- Built with reliability in mind for commodity hardware
- Optimized for streaming access, not generic posix
- Built in java
- Built for large files
- Write once read many patterns best
- Very new, changing fast
- Watch out for scaling
The Data Manager Framework

Objective

• Integrate different storage resources.
  • Cluster File System.
  • gLite / SRM / Storage Element.
  • Hadoop File System.

• Hope to meet
  • Different user requirements
Data Manager Framework Development

- Data Manager Framework development consists of
  - Interfacing underlying difference storage resources
  - Implementing Data Management logics
  - Designing Well-Define interfaces

Many efforts can be reused to speedup the development
How do I benefit from Data Manager Framework?

grid application

Cluster FS  
SRM  
HDFS

modified
How do I benefit from Data Manager Framework?

DM object

unique interface

hide the difference

DM object

unique interface

hide the difference
GAP Data Manager Design Goal

- Single namespace
- Single interface to difference DM solutions
- Support variety of storage types
  - Grids and Clouds
- Support non-structure and structure data
- Job management integration
- Authentication and Authorization
- Replication
GAP Data Manager Architecture

Authentication

Authorization

GAP Data Manager APIs

File Systems

DAV

HDFS

GridFTP

SRM

Table

HBase

...
GAP DM Server Main Components

Driver Based Framework

Academia Sinica Grid Computing
How to implement new Driver

- Inherit *Driver* Interface
- Implement all related functions
  - define Type Name
    - setDriverType, getDriverType
  - allocation mechanism
    - allocation a physical path (by some rule)
  - connection mechanism
    - *User* object as authentication
  - handle host list information
    - end-point SE host selected
  - basic file transfer action
    - upload, download, delete... (use Native API of SE)
Summary

- Integrate different storage resources on GAP to provide more options of heterogeneous data management mechanisms.
- This work also demonstrated lots of viable alternatives to Grid Storage Element, especially in terms of scalability, reliability, and manageability.
- Enhances the capability of parallel processing and also versatile data management approaches for Grid.
- GAP could be a bridge between Cloud and Grid infrastructure and more computing framework from Cloud would be integrated in the future.
Thank you for your attention and great inputs!
Backup slides
Example 1: dCache

- Developed at DESY and FERMILAB for HEP community
- Manages petabytes of storage, distributed among thousands of storage nodes
- dCache autonomously manages the number and location of the internal copies to optimize overall data throughput and to avoid hotspots
- For data transport, supports a variety of posix-like and wide area protocols. (gsiFtp,dCap,xRoot)
- Consistent security model, applied through all protocola
- Can drive a tertiary (e.g. tape) storage back-end.
- Name space is managed by NFS2/3/(4) and ftp.
- Supports also SRM v2.2 interface.
dCache File System view and Pools
dCache difficulties

- Requires a lot of effort to install and maintain in production environments
- High system complexity
- Heterogeneous set of modules due to community coding (different groups with different approaches)
- Configuration and logging can be confusing
Example 2: Storage Resource Broker SRB

- Using a single interface and authorization mechanism to access data across:
  - Multiple hosts
  - Multiple OS platforms
  - Multiple resource type (UNIX FS, HPSS, UniTree, DBMS ..)

- Global Logical Name space
  - Data organization
  - UNIX like directories (collections) and files (data)
  - Mapping of logical name to physical attributes - host address, physical path.
  - UNIX like API and utilities

- Single Global User Name Space
  - Single sign-on
  - No need for UNIX account on every system
SRB as a Data Grid

- Data Grid has arbitrary number of servers
- Complexity is hidden from users
SRB difficulties

- All or nothing approach
- Needs central DB system (Oracle preferably) to run reliably
- High system complexity
- Scaling, performance issues in heterogeneous setups