Data Grids

Lidan Wang

April 5, 2007

Outline

- Data-intensive applications
- Challenges in data access, integration and management in Grid setting
- Grid services for these data-intensive application
- Architectural approaches
- The Data Grid

Data-Intensive Applications

- Bioinformatics tasks in deciphering genes, large scale collaboration. Creation, management, and exploiting structured collections.
- Comparisons between species and integration with different protein databases.
- “Virtual observatory” - combining observations from different sources to create one unified view
- Large-scale movement of data and information integration is essential in these tasks

Challenges

- Diverse usage cases, e.g. updatable vs read-only data, binary formatted vs. relational data.

- Different storage systems, data formats.

- Access, manipulation, and analysis of large quantities of data
  - Analysis may require teraflops of computing power and access to data distributed across files and databases
  - Applications may require integration and querying of large quantities of data
  - How to discover the relevant data

Challenges (contin.)

- Need an infrastructure:
  - Shared data, storage and computing resources can be delivered to data analysis
  - Should have an integrated, and flexible manner
- Types of data
  - Structured and unstructured data
  - We focus on structured data. So collaborators know the structure of the data, and operation on the data can be carried out
  - Grid itself uses structure data collection for its operation and administration

Architectural Approaches

Data-oriented services are partitioned into four classes:

- Resource-level services for data sources
  - Data access service (structured data)
  - Data movement service (oblivious to the structure of data)
- Collective services for managing data
  - Managing data across more than 1 data source
  - E.g. Data discovery, data transformation, data transfer.
- Collective services that federate data sources
  - Integrating two or more data sources at functionality level.
  - E.g. virtual database.
- Domain-specific services
  - Data management, processing, and analysis operations for specific application domains.

Data Source Services

- Data Access
  - Interested in interfaces, and data access performance characteristics.

- Format of Data Sources
  - Data sources as either file or database
  - We want integrated services that accommodate both

- Integrating Grid and Databases
  - Remote DB access is a challenging task in Grid setting:
    - Grid has a uniform model in creating tools and services
    - Individual database management can be vastly different, so need extend this uniform technology to data management components.

Integrating Grid and Databases (contin.)

- Need to develop consistent authentication and authorization mechanisms.
- DBMS may use Grid's management regimes to recover resources
- DBMS may also use Grid infrastructure to implement distributed databases and Grid services to expose data and functionality.
- DBMS' security, management, diagnostic facilities may in turn influence the development of Grid services.

Integrating Grid and Databases (contin.)

- Another motivation: needs for functionality not yet provided by simple extensions to DB technology

- E.g. Combining computations with operations on data drives needs for new optimizations:

  - Short-term optimizations, data location and movement, scheduling of data operations and computations can be brought into the same framework as longer-term optimizations on how and where to store data.

  - If DB needs to be created/moved as part of a Grid application, then DBMS must have their location and lifetime managed by Grid technology.

File Access Service - GridFTP

- GridFTP: a data access and data transport service.
  - Uniform interface to different storage systems, disk systems.
  - Assumptions: incompatible data access protocols used by different storage systems ==> partition of the datasets on the Grid
  - As a result, applications need to specify a subset of storage systems, or use multiple methods to retrieve data.

- GridFTP functionalities:
  - GridFTP includes and extends FTP
  - As a data access protocol: user-written code that processes data prior to transmission/storage.
  - As a data transport protocol: a third-party initiates and monitors data transfer between two other sites.

Database Oriented Access

- GridFTP is file-oriented, but we want direct query interface that facilitates more complex specifications and more uniform access to data sources.
  - Example query: get all temperature readings that are between -10F and 30F only.

- Some standard mechanisms: JDBC. Remote connection.

- OGSA-DAI: an open source implementation of the specifications for accessing and integrating structured data.
  - Relational or XML databases, to be accessed via web services.
  - An OGSA-DAI web service allows data to be queried, updated, transformed and delivered.

More Details on OGSA-DAI

- Follow a sequence of steps to get services:
  1. Client uses data registry to locate a Grid data service factory (GDSF)
  2. Client activates a GDSF with its Grid service handle (GSH)
  3. Ask GDSF to produce a Grid data service (GDS) that provide the required access to data resources
  4. Ask GDS to perform a sequence of operations: update, query, load, etc. The GDS can be the data resource itself or a proxy for the data resource.

Collective Data Management Services

- Collective services: define functions whose operations can span multiple resources or services, including storage, management, and computational services.
- Examples:
  - Data transport services
  - Data discovery services
  - Workflow management, planning, and scheduling

Data Transport Services (some examples)

- Multiple data object transfer service
  - Users can submit and monitor a large number of simultaneous data transfer operations.

- Reliable data transfer service
  - Closely monitor the status of data transfer operations, restart failed transfers.
  - Augment basic data transfer service GridFTP

- Globus Toolkit's reliable file transfer service
  - Able to monitor and control third-party data transfer between two GridFTP servers.

Data Discovery Services

- First step is to discover relevant data, before we can access, integrate and analyze it.
- Use attributes that describe the data to discover the relevant data. Names for the logical/physical data can also be used.
- Some example attributes: creator, data size, how the data were generated.
- OGSA mechanisms for publication and discovering service via registries can be utilized, but it can be complicated:
  - How to describe, query, match service properties?
    - Potentially large number of data objects we may want to discover
    - How to model various subsets and views of a dataset

Workflow Management, Planning, and Scheduling

- Data analysis is likely to span multiple data sources
- Two possible approaches:
  - Move data from its storage to a remote service that performs the analysis
  - Move computation to the data
- Decompose tasks into subtasks, then come up with an execution sequence (a plan) that consists of data movements, computation resources to be used.
- Then workflow management system executes the plan, monitors status etc.
- Some applications have well-established workflows, so no need to create new ones for them.

Federation Services

- Federated DB: a collection of databases that contribute data and resources to a multidatabase federation. Each DB with full local autonomy.

- Loosely coupled federated DB: schemas of each DB is distinguishable

- Tightly coupled federated DB: a global schema hides underlying differences in schemas.

- Federation in Grid setting: much more than just integrating different databases.

Federation Services (contin.)

- Virtual DB: a single federated schema.
  - Hide different schemas from contributing DBs, and their physical locations.
  - But hard to manage: autonomous changes in a DB, costly update and maintenance of the virtual view.

- Current status: combining specific queries for limited purpose. Difficult to have a general approach due to the diversity of usage scenarios.

- An alternative: loose federation.

Transparencies in Loose Federation

- Location transparency
  - Data location is immaterial in ability to access data
- Name transparency
  - Does not need to know the name or location of the data
  - Use registry queries
- Distribution transparency
  - Querying w/o knowing the distributed nature of the data
- Replication
  - No need to know if replica or cache are being used

Transparencies in Loose Federation (contin.)

- Ownership and cost
  - No involvement in negotiating access
- Heterogeneity transparency
  - Implementation of data source is immaterial in ability to access data
- Schema change transparency
  - Individual DB can change their schemas, applications should not be affected by it.

Benefits of Using Loose Federation

- Although weaker forms of federation, it has several benefits
- Customized integration for creating a virtual DB
- More specifically, we can combine different types of transparencies to create cost/performance/functionality trade-off.
- Can be more practical than tightly coupled federated database

Benefits of Using Loose Federation

- Although weaker forms of federation, it has several benefits
- Customized integration for creating a virtual DB
- More specifically, we can combine different types of transparencies to create cost/performance/functionality trade-off.
- Can be more practical than tightly coupled federated database

Other Issues in Federating Data

- Data mediation: transparency with respect to data models
- Simple renaming attributes, e.g., schema renaming students==>graduate students
- Complex semantic mappings, e.g. rule-based approaches for mapping terms
- Benefit: uniform access, relieve applications from too much details

Other Issues in Federating Data (contin.)

- Replica management services
  - Create copies and update location services to identify location of replica
- Replica location services
  - Locate replicas by mapping a data name to its storage services
- Consistency services
  - Ensure consistency among different copies

Data Grid

- Application environment
  - Geographic distribution of users and resources
  - Heterogeneous behavior and policies
  - Large quantities of queries, each requiring access to large quantities of data

- Data Grid: a data management architecture. Defines the requirements, components and APIs

- Two core services: storage systems, metadata management

Data Grid Design

- Mechanism neutrality
  - Independent of low-level mechanisms
    - Data storage
    - Data transfer
    - Metadata storage
  - Achieved by defining interfaces that hide these details
    - Data access
    - Third-party data mover
    - Catalog access

Data Grid Design (contin)

- Policy neutrality
  - Design decisions with significant performance implications are exposed to users, rather than encapsulated in “black box”.
  - Provide basic operations for data movement, replica cataloging
  - Replica policies can be substituted with application specific code

Data Grid Design (contin)

- Compatibility with Grid infrastructure
  - Grid infrastructure can be useful:
    - Authentication
    - Resource management
  - More specialized data grid tools are compatible with Grid mechanisms

- Uniformity of information infrastructure
  - Uniform and convenient access to information
  - Use same data model and interface as the underlying Grid information infrastructure

Data Grid Design (contin)

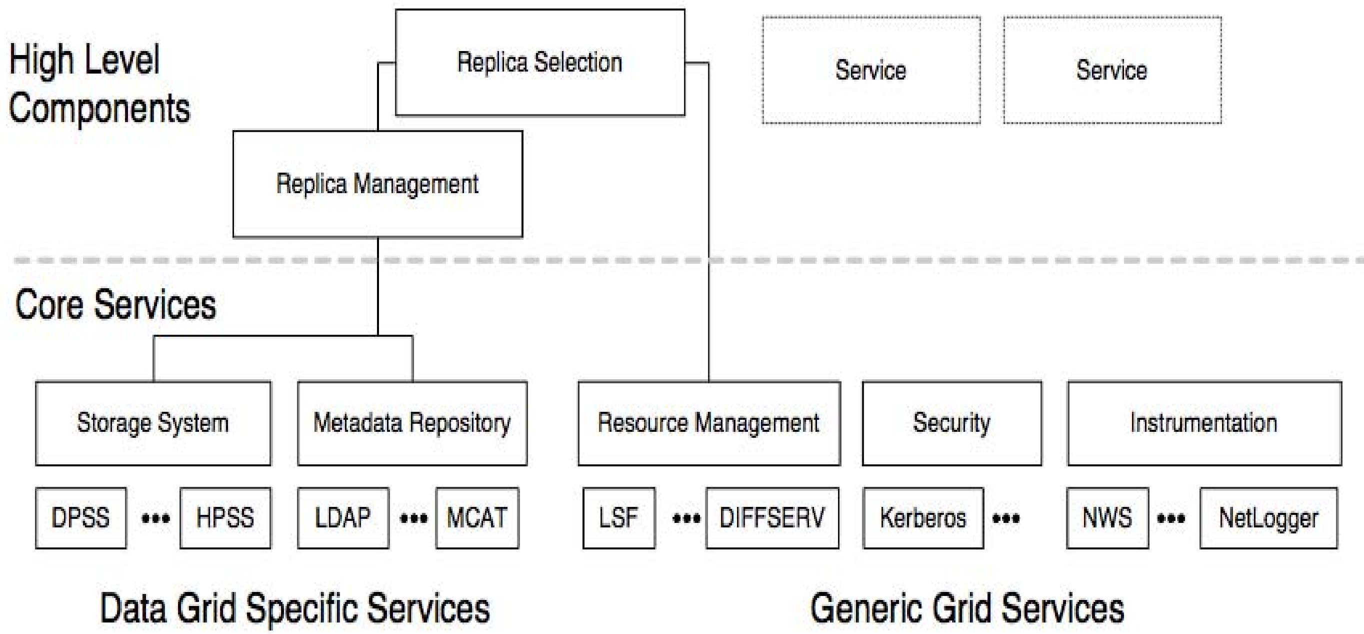

Figure 1: Major components and structure of the data grid architecture

Core Services

- Two basic services

- Data access (for data stored in storage systems)
  - Access
  - Management
  - Initiating third-party transfers of data

- Metadata access (for information about data stored in storage systems)
  - Access
  - Management

Storage Systems and Data Access

- Storage system is a data abstraction
  - Provides functions for creating, destroying, reading, writing file instances
  - File instance
    - A basic unit of information in a storage system
    - Consists of named, un-interpreted sequence of bytes
    - Associates a set of properties (name, attributes, size) with each of the file instances it contains

Storage Systems and Data Access (contin.)

- Data access API
  - Possible operations in storage systems and file instances
  - E.g. reading, writing a file instance
  - Finding out file instance attributes such as size
  - Third party transfer
  - Additional functional requirements
    - Access functions are integrated with security environment at each site
    - Charactering and monitoring the performance (by both the storage system and applications)
    - Data movement functions should be able to detect and report errors.

Metadata Service

- Metadata
  - Information about data grid, file instances, content of file instances
  - Types:
    - Application metadata: defines logical structure/semantics to be applied to a file instance. E.g. information content of a file instance.
    - Replica metadata: e.g. mapping a file instance to a particular storage system location. Management of replicated objects.
    - Configuration metadata: describes the data grid itself, e.g. network connectivity and details about storage systems

Higher-Level Data Grid Components

- Replica management
  - Create or delete copies of file instances.
  - Repository: associate with each logical file with one or more replicas or file instances.
- Replica selection and data filtering
  - Replica selection: the process of choosing a replica that will provide applications with desired performance in terms of speed, cost, security, etc.
  - Data filtering: provide access to subsets of a file instance