Computer Science and Engineering

FREERIDE-G: A Grid-Based Middleware for Scalable Processing of Remote Data

Leonid Glimcher
Gagan Agrawal

Abundance of Data

Data generated everywhere:

- Sensors (intrusion detection, satellites)
- Scientific simulations (fluid dynamics, molecular dynamics)
- Business transactions (purchases, market trends)

Analysis needed to translate data into knowledge

Growing data size creates problems

Online repositories are becoming more prominent for data storage

Emergence of Cloud and Utility Computing

- Group generating data
  - use remote resources for storing data
  - Already popular with SDSC/SRB
- Scientist interested in deriving results from data
  - use distinct but remote resources for processing
- Remote Data Analysis Paradigm
  - Data, Computation, and User at Different Locations
  - Unaware of location of other

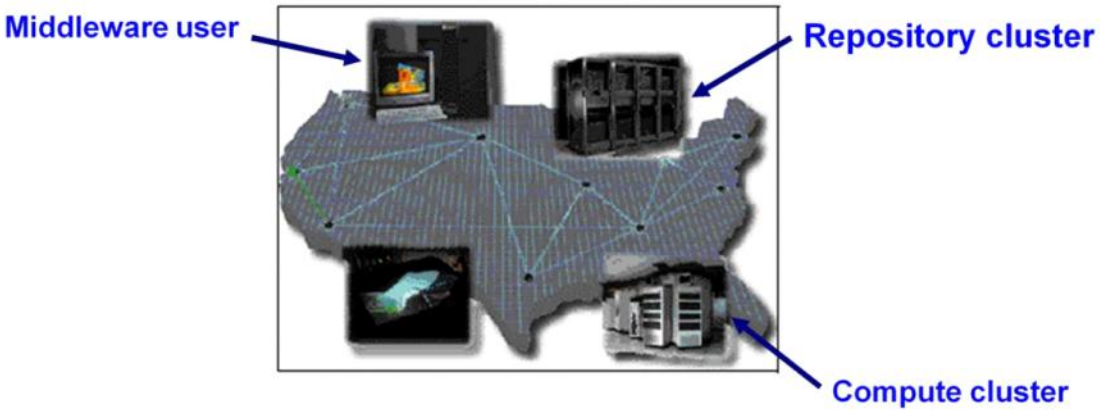
Remote Data Analysis

- Advantages
  - Flexible use of resources
  - Do not overload data repository
  - No unnecessary data movement
  - Avoid caching process once data
- Challenge: Tedious details:
  - Data retrieval and caching
  - Use of parallel configurations
  - Use of heterogeneous resources
  - Performance Issues
- Can a Grid Middleware Ease Application Development for Remote Data Analysis and Yet Provide High Performance ?

Our Work

FREERIDE-G (Framework for Rapid Implementation of Datamining Engines in Grid)

Enable Development of Flexible and Scalable Remote Data Processing Applications



Challenges

- Support use of parallel configurations
  - For hosting data and processing data
- Transparent data movement
- Integration with Grid/Web Standards
- Resource selection
  - Computing resources
  - Data replica
- Scheduling and Load Balancing
- Data Wrapping Issues

FREERIDE (G) Processing Structure

KEY observation: most data mining algorithms follow canonical loop

Middleware API:

- **Subset of data to be processed**
- **Reduction object**
- **Local and global reduction operations**
- **Iterator**

Derived from precursor system FREERIDE

```
While( ) {
  forall( data instances d) {
    (I , d') = process(d)
    R(I) = R(I) op d'
  }
  .....
}
```

FREERIDE-G Evolution

FREERIDE

data stored locally

FREERIDE-G

- ADR responsible for remote data retrieval
- SRB responsible for remote data retrieval

FREERIDE-G grid service

Grid service featuring

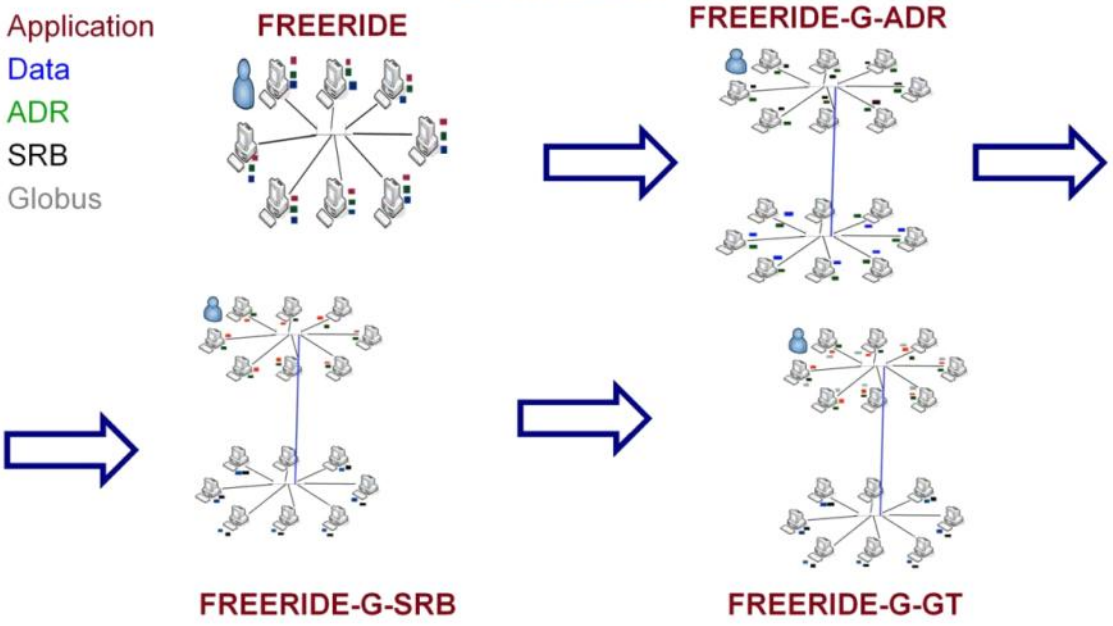
- Load balancing
- Data integration

College of Engineering
The Ohio State University

Computer Science and Engineering

Evolution

Application
Data
ADR
SRB
Globus



Classic Data Mining Applications

Tested here:

- Expectation Maximization clustering
- Kmeans clustering
- K Nearest Neighbor search

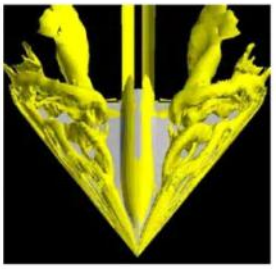
Previously on FREERIDE:

- Apriori and FPGrowth frequent itemset miners
- RainForest decision tree construction

Scientific data processing applications

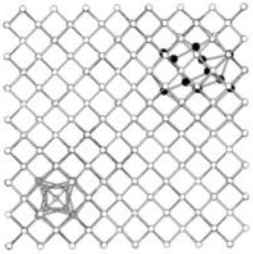
VortexPro:

- Finds vortices in volumetric fluid/gas flow datasets



Defect detection:

- Finds defects in molecular lattice datasets
- Categorizes defects into classes



Middleware is useful for wide variety of algorithms

FREERIDE-G Grid Service

Emergence of Grid leads to computing standards and infrastructures

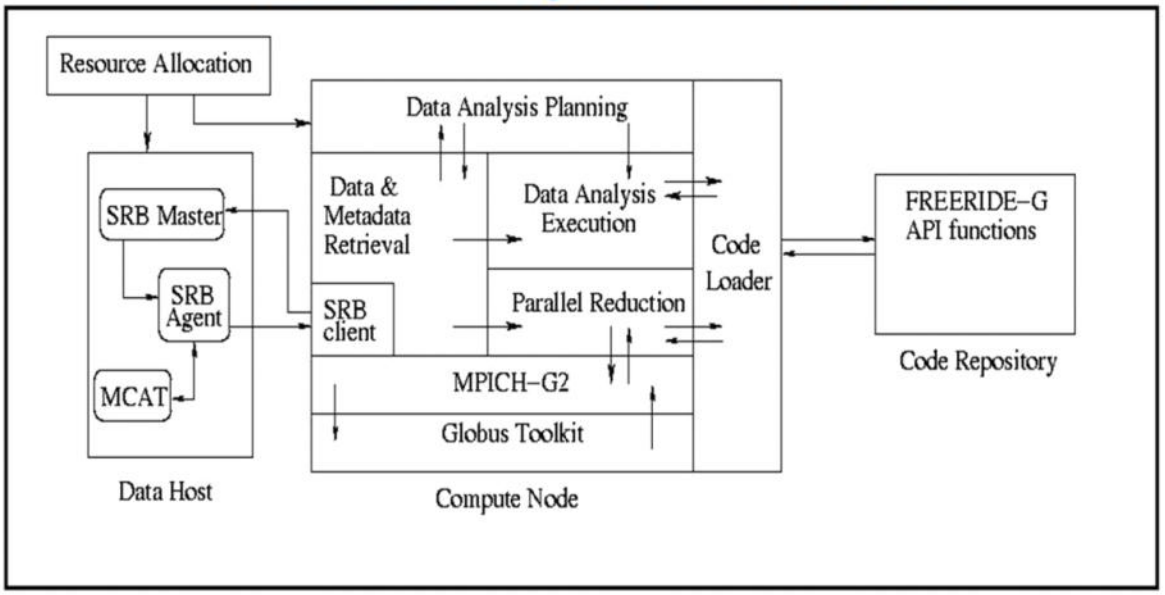
Data hosts component already integrated through SRB for remote data access

OGSA – previous standard for Grid Services

WSRF – standard that merges Web and Grid Service definitions

Globus Toolkit is most fitting infrastructure for Grid Service conversion of FREERIDE-G

FREERIDE-G System Architecture



Compute Node

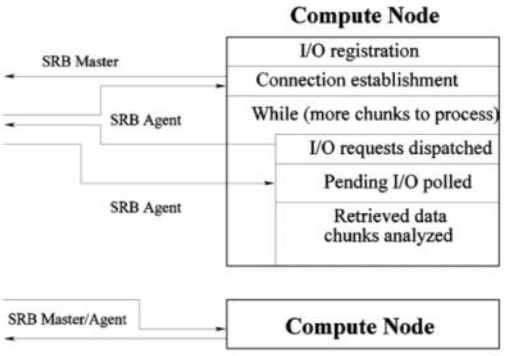
More compute nodes than data hosts

Each node:

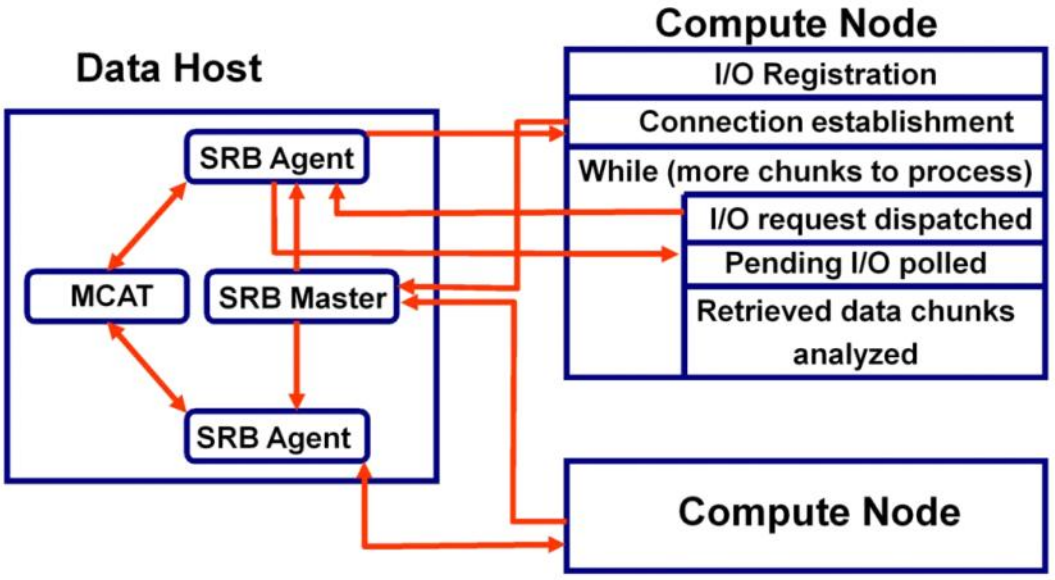
1. **Registers IO (from index)**
2. **Connects to data host**

While (chunks to process)

1. **Dispatch IO request(s)**
2. **Poll pending IO**
3. **Process retrieved chunks**



FREERIDE-G in Action



Implementation Challenges

- Interaction with Code Repository
  - Simplified Wrapper and Interface Generator
  - XML descriptors of API functions
  - Each API function wrapped in own class
- Integration with MPICH-G2
  - Supports MPI
  - Deployed through Globus components (GRAM)
  - Hides potential heterogeneity in service startup and management

Experimental setup

Organizational Grid:

- Data hosted on Opteron 250 cluster
- Processed on Opteron 254 cluster
- Connected using 2 10 GB optical fibers

Goals:

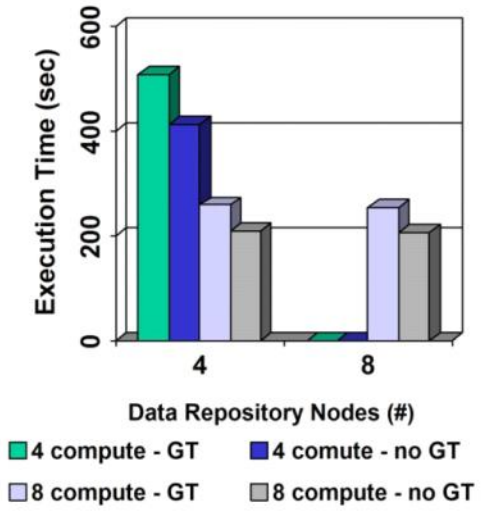
- Demonstrate parallel scalability of applications
- Evaluate overhead of using MPICH-G2 and Globus Toolkit deployment mechanisms

Deployment Overhead Evaluation

Clearly a small overhead associated with using Globus and MPICH-G2 for middleware deployment.

Kmeans Clustering with 6.4 GB dataset: 18-20%.

Vortex Detection with 14.8 GB dataset: 17-20%.



Previous FREERIDE-G Limitations

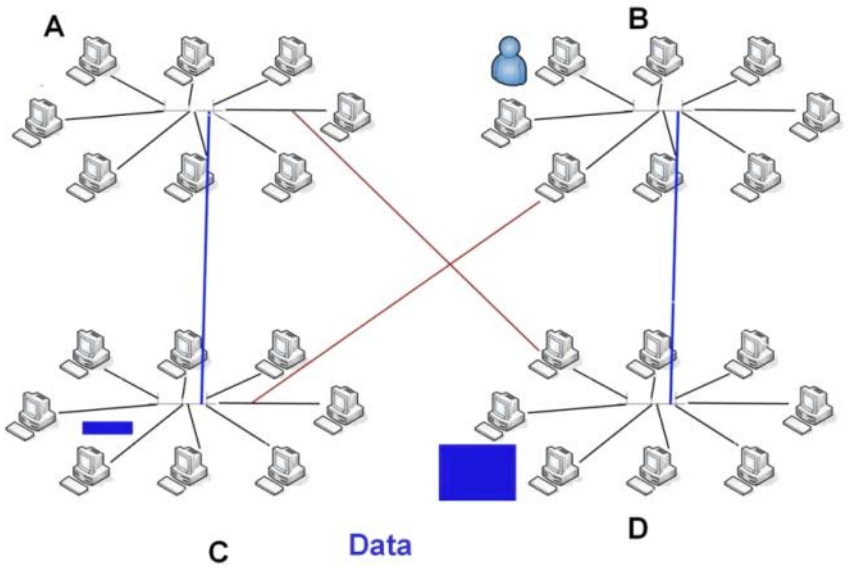
Data required to be resident on a single cluster

Storage data format has to be same as processing data format

Processing has to be confined to a single cluster

Load balancing and data integration module helps FREERIDE-G overcome limitations efficiently

Motivating scenario



Run-time Load Balancing

1. Based on unbalanced (initial) distribution figure out an partitioning scheme (across repositories)
2. Model for load balancing based on two components (across processing nodes):
   - partitioning of data chunks,
   - data transfer time.

   weights used to combine two components

Load Balancing Algorithm

```
Foreach (chunk c)
 If ( no processing node assigned to c )
   Determine Transfer Costs across all attributes
  //Compare processing costs across all nodes:
   Foreach (processing node p)
     Assume chunk is assigned to p,
     Update processing cost and transfer cost
          Calculate averages for all other processing nodes (except p)
       for processing cost and transfer cost
     Compute difference in both costs between p and averages
     Add weighted costs together to compute total cost
     if (cost < minimum cost)
       update minimum
   Assign c to processing node with minimum cost
```

Approach to Integration

Bandwidth available also used to determine level of concurrency

After all vertical views of chunks are re-distributed to destination repository nodes, use wrapper to convert data

Automatically generated wrapper:

- Input – physical data layout (storage layout)
- Output – logical data layout (application view)

Horizontal View

X_1	Y_1	Z_1
…	…	…
X_n	Y_n	Z_n

Vertical View

Experimental Setup

Settings:

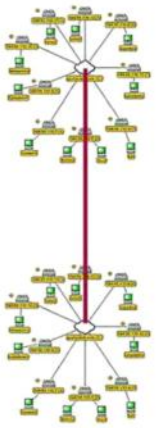
- Organizational Grid
- Wide Area Network

Goals are to evaluate:

- Scalability
- Load balancing accuracy
- Adaptability to scenarios
  - compute bound,
  - I/O bound,
  - WAN setting

Setup 1: Organizational Grid

Repository cluster (bmi-ri)



Compute cluster (cse-ri)

Data hosted on Opteron 250's

Processed on Opteron 254's

2 clusters connected through 2 10 GB optical fibers

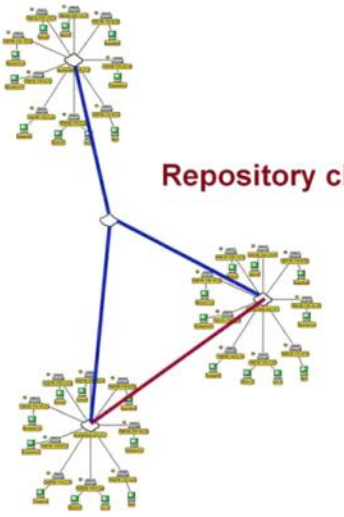
Both clusters within same city (0.5 mile apart)

Evaluating:

- Scalability
- Adaptability
- Integration overhead

Repository cluster (Kent ST)

Setup 2: WAN



Repository cluster (OSU)

Compute cluster (OSU)

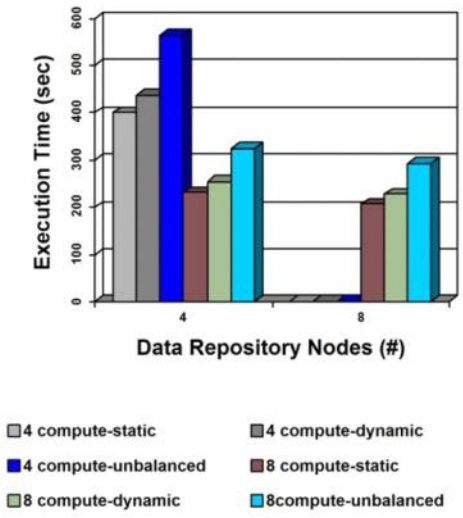
Data Repository:
- Opteron 250's (OSU)
- Opteron 258's (Kent St)

Processed on Opteron 254's

No dedicated link between processing and repository clusters

Evaluating:
- Scalability
- Adaptability

Scalability in Organizational Grid

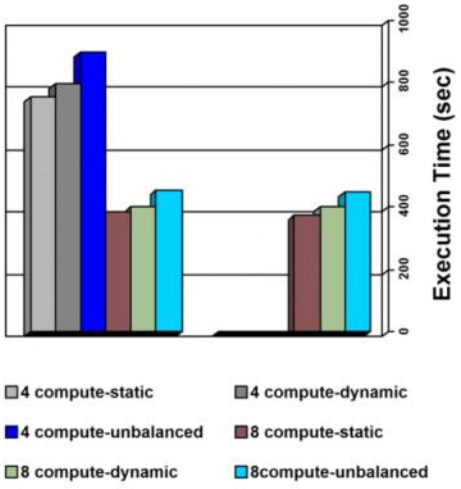


Vortex Detection (14.8 GB)

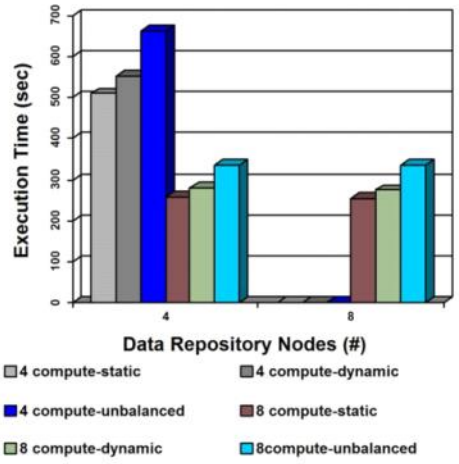
- Linear speedup for even data node - compute node scale-up
- Near-linear speedup for uneven compute node scale-up

- Our approach outperforms initial (unbalanced)
- Comes close to (statically) balanced configuration

Evaluating Balancing Accuracy



EM (25.6 GB)



Kmeans (25.6 GB)

Conclusions

FREERIDE-G – middleware for scalable processing of remote data

- Integrated with grid computing standards
- Supports load balancing and data integration
- Support for high-end processing
- Ease use of parallel configurations
- Hide details of data movement and caching
- Fine-tuned performance models
- Compliance with remote repository standards