Enabling ROOT Queries Using a Distributed Data Storage Architecture.

Nuno Almeida

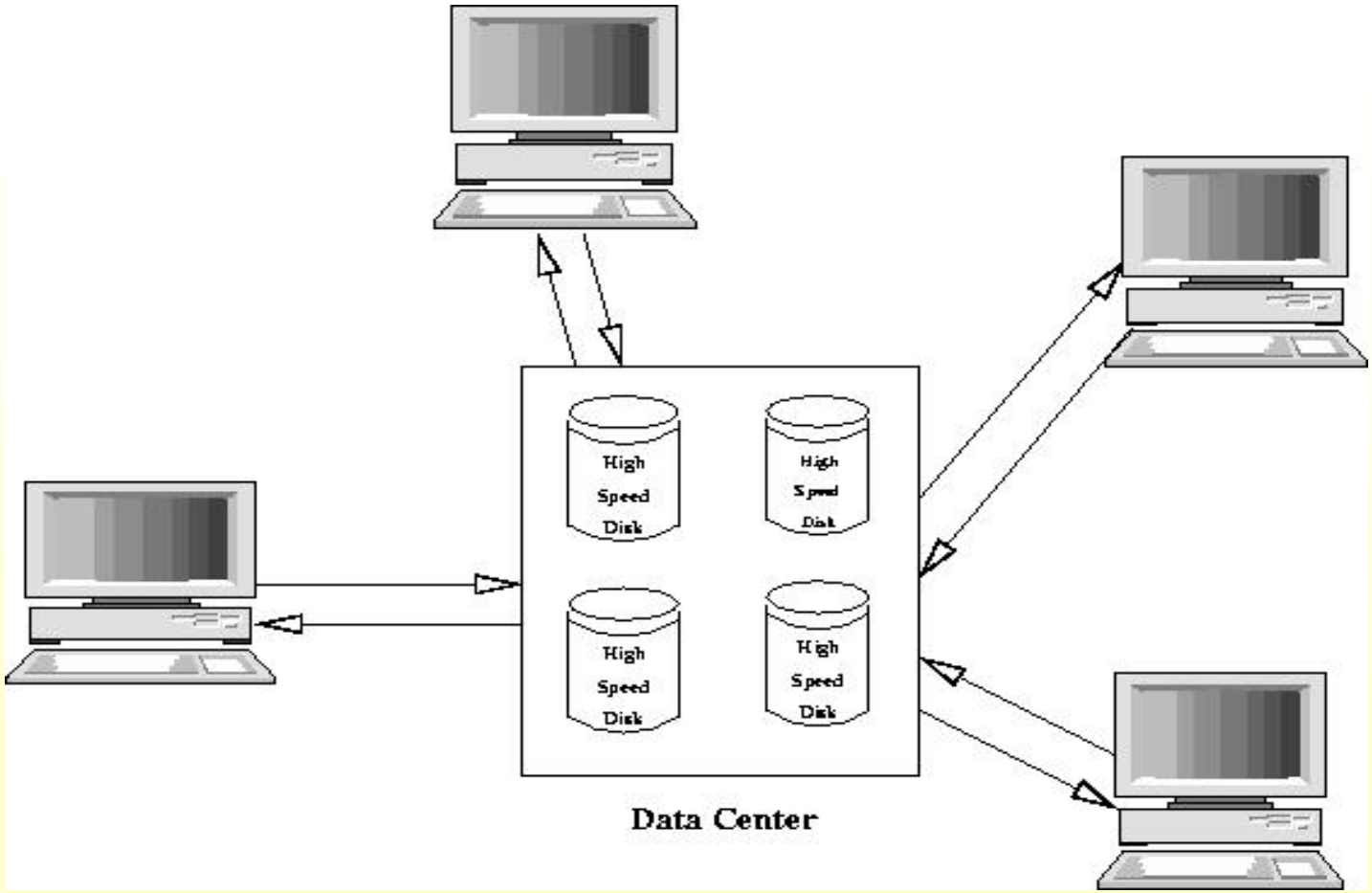
http://adetti.gridpt.org

Main Idea

Do Not Move Data!

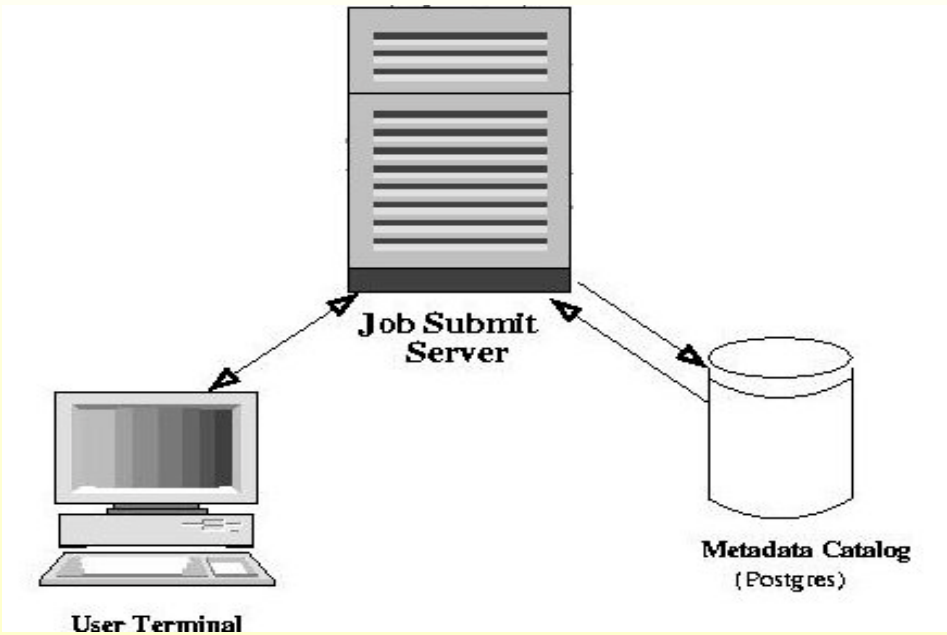
Each Node Stores And Is Reponsible By a Subset of The Whole Data.

Usual Way


High
Speed
Disk
High
Speed
Disk
High
Speed
Disk
High
Speed
Disk
Data Center

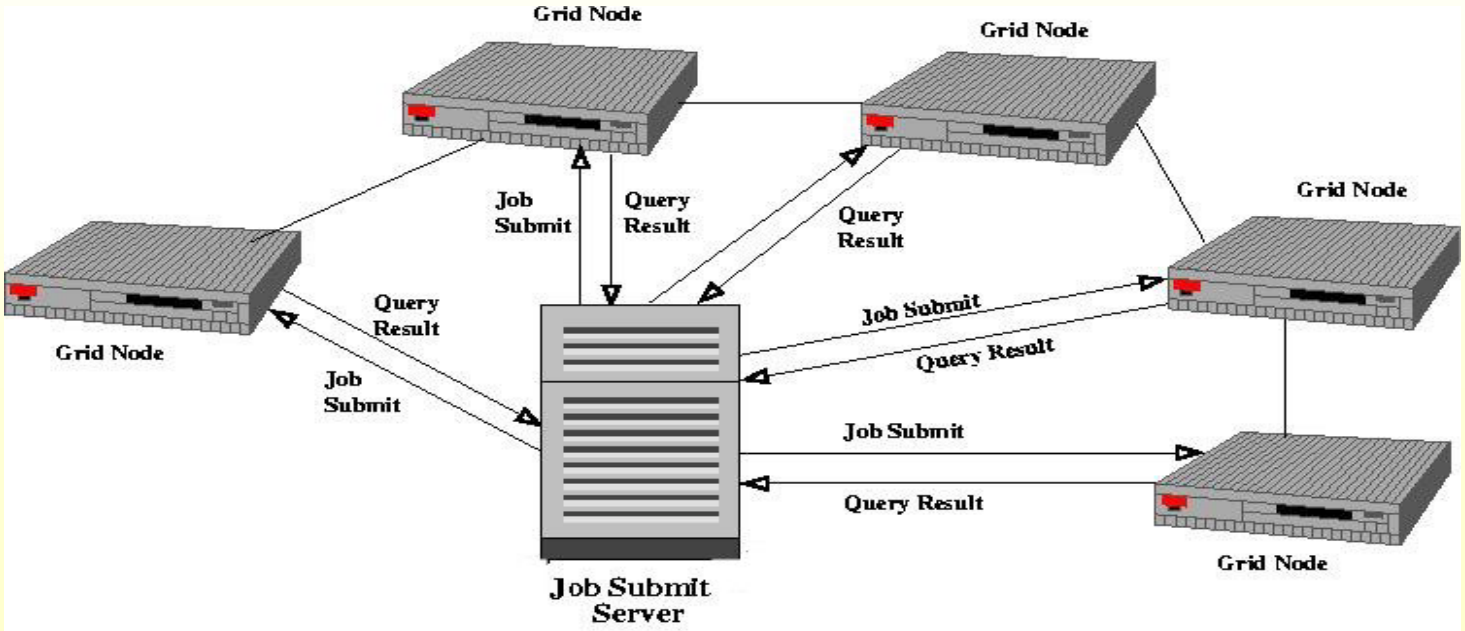
Arquitecture Overview

- User submit a ROOT query through a interface to Job Submit Server.
- Job submit information will be stored in the Metadata Catalog.



Arquitecture Overview

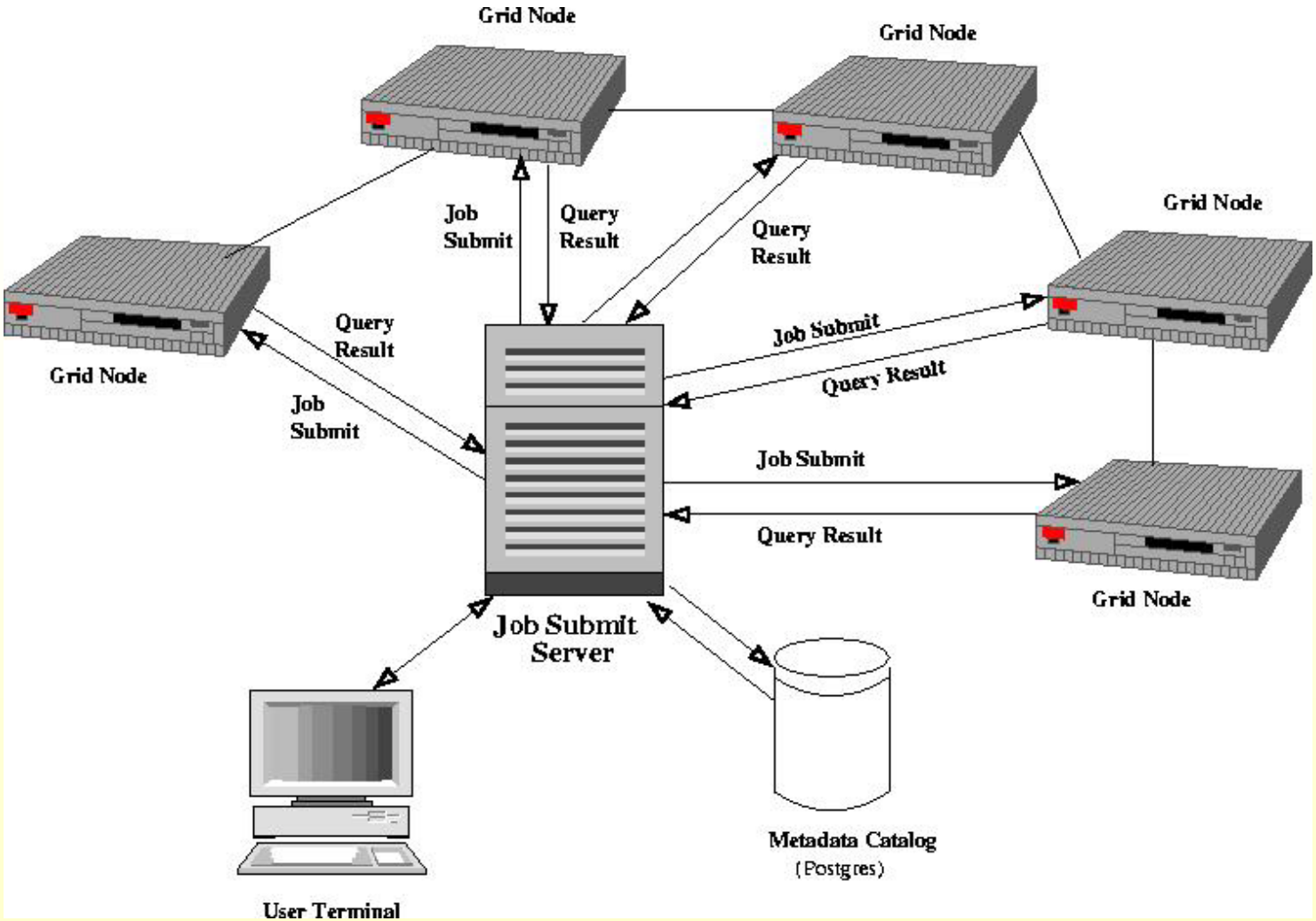
- The job is submitted to the grid nodes using Globus API functions.
- All the nodes query their own information with ROOT and retrieve a ROOT file with a TTree.



Arquitecture Overview

- Job Submit Server receive the ROOT Files from the Grid nodes and produce a final ROOT Final with the result of the query.
- User can download or consult the final file because it is a TTree.
- User can also visualize the state of the job in each Grid node.

Arquitecture Overview



The Technologies

The Technologies involved in the prototype are:

• ROOT • Globus • Postgres	• LDAP • PHP

Relevant Features Used

ROOT

- TObject
- TTree
- CINT
- Filtering data from TTree
- ROOT I/O
- TChain

Relevant Features Used

Carrot

- Browsing ROOT files
- Histograming variables

Globus

- Toolkit that provides GRID API functions.

Postgres

- Metadata Catalog Implementation.

Relevant Features Used

LDAP

- Query GRID node information.

PHP

- Create interface between web browser and system.
- And also users preform query through this interface.

ROOT Work In The System

- ROOT has a very important work in the system.
- Stores the information in each node using TTree's.
- Filter the information in each node and retrieve a result file that include a TTree.
- Join all the result files in the Job Submit Server using a TChain and produces a final TTree that is the query result.
- View the final result file with a TBrowser or with Carrot.

Reading ROOT Files

- **Reading Events From TTree's Or From Files**
  - Events Can Be Stored In Different Ways:
    - In a Ttree
    - Event by Event in The File (Using TObject method "Write()")
  - Why Use The Second Way ?
    - Because the Events Are Processed One By One
    - They Are Organized in a Better Way Inside The File
    - The Size of The File is Almost The Same as Using TTree

Reading ROOT Files

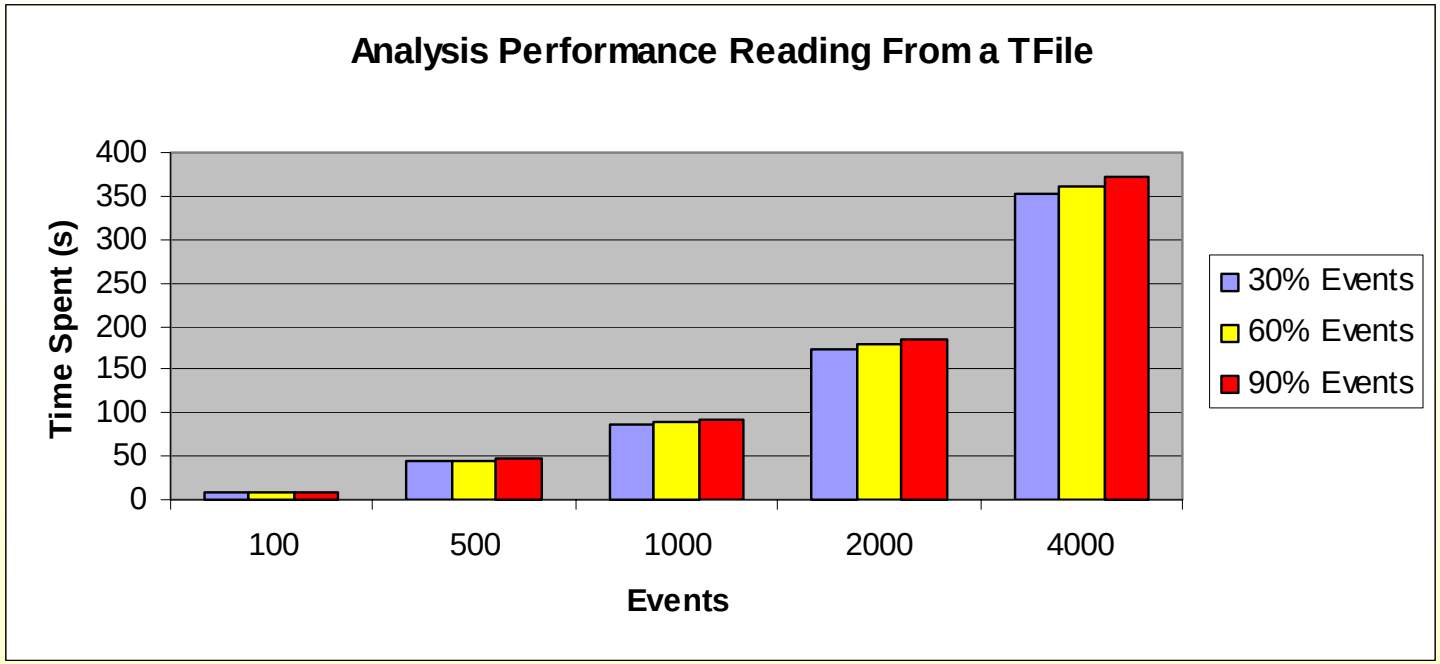
- Problems Found
  - Reading Event by Event is Much More Slower Comparing With TTree's because has much more I/O transactions.

- How We Read Events
  - Here's How We Read Event's From a TFile:

```
TFile *f =new TFile("Evento.root");
Event *e =new Event();
e = (Event*)f->Get("Event;1");
```

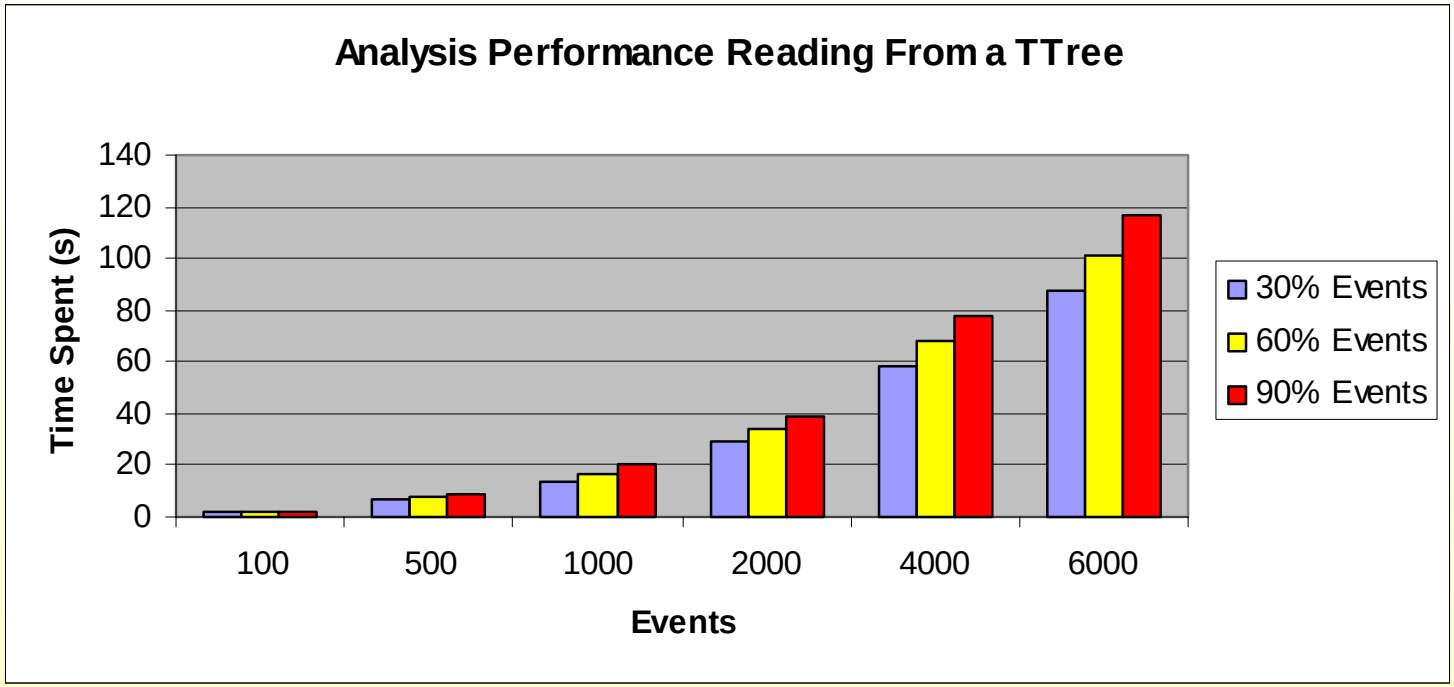
Reading ROOT Files

- Results Obtained



Reading ROOT Files

- Results Obtained



Pros and Cons

Advantages

- Commodity Data Storage.
- Huge Scalability (400 GB/node).

Problems

- Load balancing → Suitable storage policy
- Fault tolerance. → Data replication or Backup

Future Developements

- Develop a storage mechanism to submit more work to the best nodes.
- Provide to user several interfaces to submit work.