AstroPortal: A Science Portal to Grid Resources

Ioan Raicu
Distributed Systems Laboratory
Computer Science Department
University of Chicago

January 5th, 2006
Introduction

• Science Portals: gateway to Grid resources
• Potential Applications Characteristics
  – Large data sets
  – Large number of users
  – Easy parallelization
• Applicable fields:
  – Astronomy
  – Medicine
  – Others
Astronomy Field

- Astronomy datasets (i.e. SDSS) are the crown-jewels
  - SDSS DR4
    - 500K images
      - 300M+ objects
      - 1TB+ compressed images (2MB x 500K)
      - 3TB+ raw images (6.1MB x 500K)
    - 100K worldwide potential users

- Applications:
  - Stacking
  - Montage
Medical Field

• Medium to large medical datasets are hard to acquire
  – Typical medium size data set (of CT images)
    • 1000 patient case studies
      – 100K images (1000 cases x 100 images)
        » 1M+ objects (i.e. organs, tissues, abnormalities, etc…)
        » 0.4TB+ raw images (4MB x 100K)
    • 10K+ potential users from 1K+ of different institutions (research labs, hospitals, etc…)

• Applications:
  – Making datasets available to trusted parties
  – Allowing image processing algorithms to be dynamically applied
  – Normal tissue classification in CT images
  – Lung cancer image databases
Medical Field (cont)

  - Information eXtraction from Images (IXI): Image Processing Workflows Using A Grid Enabled Image Database
  - Information eXtraction from Images (IXI): Grid Services for Medical Imaging
- University of Oxford
  - Grid-based Federated Databases of Mammograms: Mammogrid and eDiamond experiences
- Universidad Politécnica de Valencia Spain
  - A Middleware Grid for Storing, Retrieving and Processing DICOM Medical Images
- University of the West of England, Frenchay, Bristol & CERN, Geneva, Switzerland
  - A Grid Information Infrastructure for Medical Image Analysis
Generic Overview

1/5/2006 AstroPortal 6

Generic Overview

Data

Repository

2-way communication via WS

Portal WS

SITES

2-way communication via WS

Data

Manager

Resource Provisioning via GRAM

Local FS

Client 1

Client 2

Client n

Compute

Nodes

2-way communication via WS

2-way communication via WS

User Interface

Resource Provisioning via GRAM

Local FS

Data

Manager

Data

Repository
Functionality Overview

• Input
  – A set of {band ra dec} tuples plus operation to be performed (GetAll, SumAll, etc…)

• Work
  – GetAll: crop ROIs
  – SumAll: crop ROIs and stack them

• Output
  – GetAll: A set of images corresponding to the above tuples
  – SumAll: 1 image corresponding to the summation of the above tuples
Current Implementation

User Data Repository

AstroPortal WS

Query SQL DB via HTTP

GridFTP

AstroData (AD) Manager

Resource Provisioning via qsub

Local GPFS (LAN)

Data Location DB

Site

Compute Nodes

Query & Response via WS

Query & Response via WS

Query & Response via WS

Data Repository

GridFTP

AstroClient AC_1

AC_i

AC_n
Summary FIT Client Performance
Response Time

- JAVA 1crop
- JAVA 10crops
- C 1crop
- C 10crops

File System:
- Local
- ANL GPFS
- NFS
- PVFS
- TG GPFS
Summary FIT Client Performance Throughput

- Local ANL GPFS
- Java 1crop
- Java 10crops
- C 1crop
- C 10crops

Throughput (crops/sec) - log scale
Time to complete O(100K) Crops

File System

LOCAL
NFS
PVFS
ANL GPFS
TG GPFS

JAVA 1crop
JAVA 10crops
C 1crop
C 10crops

Time (sec) - log scale

10000
1000
100
10
1

1/5/2006 AstroPortal 12
Raw FS Performance
Time to complete O(100K) Crops
Target Implementation

Data Location DB

MDS4 / RLS Index

AP WS Lookup & Location

USER

Query & Results

RLS Query

Query & Results

MDS4 Registration

Query & Results

Site

AstroPortal WS

Query & Response

Query & Response

Query & Response

Site

AstroPortal WS

Browse & Query

Query & Results

State Exchange

Query Forwarding

Target Implementation

Compute Nodes

Local GPFS (LAN)

AD Manager

Data Repository

GRAM -- Advanced Reservations

Local GPFS (LAN)

GRAM -- Advanced Reservations

Local GPFS (LAN)

TG GPFS (WAN)
Some Design Choices

• all the communication is implemented over WS with the exception of the query to the database for translating \{band ra dec\} to \{path/filename\}, which is done over HTTP / TCP

• AP WS can support an arbitrary number of users and workers dynamically

• users must know where the AP WS is; ideally this would be done via MDS4

• workers must know where the AP WS is; ideally this would be trivial if the AP WS were to dynamically start the worker clients via GRAM
Some Design Choices (cont)

- requests/results are bundling together to send several queries/results in a single WS call
- polling (as opposed to notifications) the AP WS is used as the primary mechanism for workers to get requests, and for the users to get the results back
  - Polling: should yield the best performance for a heavily utilized AP WS since the poll call also retrieves results/work if there is any, and there would always be something to do
  - Notifications: should be more efficient for a lightly utilized AP WS, since WS calls would only be made when there was a need
Key Features Missing: Implementation Future Work

• Use GRAM to make resource provisioning dynamically
• Use MDS to register the AP WS to MDS4, and have the user (client code) automatically find the AP WS via MDS4
• Make transition from polling to notifications
  – Necessary to give the AP WS better resource management control over the worker nodes
• Add non-volatile state support (for crash recovery)
• Use RLS API to keep track of data location
• Add GUI for monitoring entire system
Open Research Questions

• Cluster level
  – advanced reservations
  – resource allocation
  – resource de-allocation

• Data management
  – Data location and replication
  – Data caching hierarchies

• Resource management
  – Distributed resource management between various sites
Open Research Questions: Cluster Level

• leverage techniques used in large clusters
• Find heuristics will apply for managing efficiently the set of resources depending on the workload characteristics, number of users, data set size and distribution, etc…
• how to perform efficient state transfer among worker resources while maintaining a dynamic system
Open Research Questions: Data Management

- very large data set distributed among various sites
- Replication strategies to meet the desired QoS
- Data placement based on past workloads and access patterns
Open Research Questions
Resource Management

• The inter-site communication among the AP WS and its effects on the overall system performance is very interesting

• Workload management, moving the work vs. moving the data

• Algorithms, the amount of state information, and the frequency of state information exchanges will affect the performance of the overall system
Questions?

Slides: http://people.cs.uchicago.edu/~iraicu/research/AstroPortal/astro_portal.presentation_v2.pdf
Terminology

- **Site**: A TeraGrid site, such as UC/ANL, SDSC, NCSA, PSC, ORNL, TACC, etc…
- **User**: user from the astronomy domain who wants to query the data set with a 5-tuple (path & file name, x-coordinate, y-coordinate, height, and width)
- **AstroPortal Web Service (AP WS)**: A WS that gives users an entry point into accessing TG resources to process the user’s queries
- **MDS4 Index**: A standard MDS4 Index used for resource (AP WS) discovery by the users
- **Compute Nodes - AstroClient (AC)**: dedicated nodes in TG that are reserved in advance to be used for processing queries from the AP WS
- **Data Repository**: the original data set in compressed format that can be accessed via GridFTP
- **AstroData (AD) Manager**: A data resource manager that keeps the data set up to date between the data repository, and the corresponding file systems (Local GPFS, TG GPFS, etc…); in the distributed version, the AD Manager could also use RLS to manage data replication; the AD Manager also communicates with the AP WS in order to keep the AP WS data set index updated with the latest data set location
- **Local GPFS**: Refers to site local GPFS accessed over a LAN
- **TG GPFS**: TeraGrid wide GPFS accessed over a WAN
- **RFT**: Used to update the working data set on GPFS from the data repository
- **GRAM**: Used to make advanced reservations of AC compute nodes by being scheduler independent
- **RLS**: used to keep track of the data replicas in the distributed AP architecture