**Globus TK ver. 4 & OceanStore**

Mar / 01 / 2011

by Youngil Kim

---

**Agenda**

- **Globus**
  - What’s Globus
  - New features in GT4
  - GT4 Architecture
  - GT4 S/W Details

- **OceanStore**
  - Overview of OceanStore (Introduction)
  - System Architecture
  - Data Location & Routing in OceanStore
  - Update Model in OceanStore
  - Introspection

---

**What’s the Globus?**

- **Globus**
  - Community: virtual organizations.
  - Software itself, → GT
  - Infrastructure that supports this community:
    - code repository, list of mails, problem tracking system ...
    - All accessible at [dev.globus.org](http://dev.globus.org)

- **GT (Globus Toolkit)**
  - GT2 in 2002 – Introducing OGSA
  - GT3 in 2003 – based on OGSi
  - GT4 in 2004 – based on WSRF
  - Web Service Resource Framework
  - Current version 5.0.3
GT4

- **New Features**
  - Full-featured Web service in Java, Python and C
  - Tools and libraries for WSRF and WS compliant client
  - A new GRAM implementation optimized for flexibility,
  - A new improved GridFTP Service
  - ...
  - GT4 provides a set of *infrastructure services*:
    - standard implementations for common purpose.

GT4 Architecture

- Communication among components
  - WS-I compliant SOAP messaging
  - X.509: an ITU-T standard for a PKI, SSO and PMI

- User Programming?
  - Implementing Service
    - Using Container (Java, C and Python)
    - Own Services using GT libraries (C)
  - Implementing Client
    - Using Client Libraries
    - Own Client Program (Communication via X.509)

GT4 S/W Details

They are less thoroughly tested than other components and likely to change in the future.
How to manage Execution?

- GRAM (Grid Resource Allocation and Management) Service
- Using GRAM, Client can express followings
  - The type and quantity of desired resource
  - Data to be staged to and from the execution site
  - The executable and its arguments
  - Credentials to be used
  - Job persistence requirements

Additional Component
- WMS: Dynamic execution sandbox using virtualization (Xen)
- GTCP: Managing instrumentations such as earthquake engineering, microscopes...

How to manage Data?

- Data problem is broad and complex in Grid Problems
  - GT supports reliable and high-performance components for accessing and moving data
    - GridFTP: Data transfer libraries & tools
    - RFT: Manage multiple GridFTP
    - RLS: Manage location of replicated files and data sets
    - DRS: GridFTP + RLS, Used for data replication
    - OGSA-DAI: relational and XML data

How to manage Service?

- Monitoring and Discovery of Services
  - Using WSRF/WSN interface
    - Supported by container
  - Aggregator Services (For non-WSRF/WSN entities)
    - Index (register) / Trigger (filter)

2) Using WSRF/WSN interface after aggregation services

How to manage Security?

- Three types of security supporting
  - Two entities to validate each other’s credential
    - Message Protection: establish secure channel
    - Delegation: delegate credential to a remote component

- Fast-Default
  - Transport-level security w/X.509 credentials
  - SAML and grid-mapfile
  - X.509 Proxy certificates/WS-Fed
  - Username/password
  - TLS

- Slow
  - Message-level security w/X.509 credentials
  - Grid-mapfile
  - X.509 Proxy certificates/WS-Fed
  - WS-Security
  - SOAP
How to build new services?

- GT4 Support common development components
  - Containers!
- Containers (Java, C, Python) can support
  - Basic WS specification
  - State management specifications
  - Enhanced registry and management capability
  - Java Container: GT4 Java Web Services
    - GRAM, RTF, DRS, Delegation, Index, and Trigger

OceanStore:
An Architecture for Global-Scale Persistent Storage

Some useful backgrounds...

- Official Homepage of OceanStore
  - http://oceanstore.cs.berkeley.edu
  - last modified on … 07/08/2002
  - Members are busy? 😊

The paper appeared in 2000, and the prototype was under development at that time, but
  “Pond: the OceanStore Prototype” was published in 2003

Now, they use “Tapestry” for the locality of object location
  - Ben Y. Zhao is a member of this project.

Why OceanStore?

- In Ubiquitous Computing Environment...
  - Need geographically distributed servers and caching close to clients
    - > Like COMA style design
    - > More efficient but, how to find where the data is?

- Two keywords of OceanStore
  - Untrusted infrastructure
    - Servers are lack of trust -> avoid server/client
  - Nomadic data
    - promiscuous caching: data can be cached anywhere, anytime

※ NUMA: have data in “home node”, when process need to access data, copy to local cache
COMA: no home node, when remote node request access cause migration of data to the node.
OceanStore?

Pool: highly connected group of nodes data is allowed to “flow” freely.

Overview of System

- Main components of OceanStore
  - Fundamental Unit - Naming & ACL
    - Persistent objects named by GUID
  - Replicas - Location & Routing
    - Floating replicas of data on multiple servers
    - Located through two methods (probabilistic/deterministic)
  - Modification Object: By updating
  - Object Existence form
    - Active / Archival form (Deep archival storage)
  - Introspection
    - Optimization & Complement other components

Naming & Access Control

- OceanStore Objects
  - Identified by GUID, pseudo-random, fixed-length
  - Secure hash of “Owner’s key” and “Name”
  - Can act directory: Mapping GUIDs to “Name”
  - Server & Archival fragments has GUID
    - Server: hash of its public key
    - A.F.: hash of its data

- Access Control
  - Two primitive types of access control
    - Reader restriction
      - Distribute encryption key to the user with read permission
      - Remove replica or re-encrypt with new key
    - Writer restriction
      - All writes are signed and verified
      - Well-behaved nodes authenticate using specified ACL

Data Location & Routing

- Messages for location & routing data contains
  - Dest. GUID and predicates
    - => no IP Based routing.

- Two-Level Routing
  - Probabilistic Algorithm (Fast)
    - Attenuated Bloom Filter
  - Reliable hierarchical method (slow)
    - Based on Plaxton’s data structure
Bloom Filter?

Overall answer speed is better with Bloom filter than without Bloom Filter

Attenuated Bloom Filter:
Query propagates through nodes if there is no local matching

* Image is simplified

Fast Routing – Example

1) Check Local Filter -> Fail
2) Check Neighbors Filter -> probably exist in n2
3) Check Local Filter -> Fail
4) Check Neighbors Filter -> not in n4, but might exist in n3
5) Check Local Filter -> Verified

Routing Table for 0325

<table>
<thead>
<tr>
<th>0</th>
<th>8</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>XXX0</td>
<td>84F8</td>
</tr>
<tr>
<td>L2</td>
<td>XX05</td>
<td>XX85</td>
</tr>
<tr>
<td>L3</td>
<td>X025</td>
<td>X825</td>
</tr>
<tr>
<td>L4</td>
<td>0325</td>
<td>8325</td>
</tr>
</tbody>
</table>

Routing Table for 0908

<table>
<thead>
<tr>
<th>0</th>
<th>9</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>L2</td>
<td>XX08</td>
<td>2098</td>
</tr>
</tbody>
</table>

Routing Table for 4598

<table>
<thead>
<tr>
<th>0</th>
<th>4</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>L3</td>
<td>0598</td>
<td>4598</td>
</tr>
</tbody>
</table>

Finding : 0325

Update

- Designed based on Conflict Resolution
  - Introduced in Bayou System (http://www2.parc.com/csl/projects/bayou)
  - Need computation in server side (Serializable...)
- Format & Semantics
  - Client Request “Update” to primary replica tier and secondary tier
  - Replica tests update’s predicates
  - “Commit” or “Abort”
- Set of predicates
  - compare-version, compare-size: dealing unencrypted data
  - compare-block, replace-block, append: position dependent block cipher
  - search: directly on ciphertext[47]
  - delete-block: replace with empty pointer block
  - insert-block: see next page

Update: insert-block

Insert new block

Block 41 — Block 41.5 — Block 42 — Block 43 — Block 41.5

Update — Serializing updates

- Serializing Updates
  - Start of conflict resolution
  - Select total order among updates
  - Need a master → no trusted node
    → Use primary tier: group of master replica

- Two ways for serializing
  - Decision is depend on applications
  - Consistent way: Decision made by primary tier
  - Less consistent way: Propagate through secondary tier

C1 generates updates
After generating updates sends it to primary tier and propagate to near nodes

Primary tier perform Byzantine agreement protocol to select serializing update
Secondary replicas propagates updates
Update – Serializing updates

- C1 generates updates
- After generating updates sends it to primary tier and propagate to near nodes
- Primary tier perform Byzantine agreement protocol to select serializing update
- Secondary replicas propagates updates
- Once Primary tier finished its agreement protocol, the result is multicast down the dissemination tree to all of the secondary replicas

Introspection

- OceanStore will consist of millions of servers
- Servers & Networks load will vary
- How to solve it? -> introspection

- Usage
  - Cluster Recognition
  - Identify and group closed related files
  - Replica Management
  - Manage proper number of replica
  - Can be used in many aspects
  - supplements for routing structure, dissemination tree, archival fragments...

Introspection

Events:
- Any incoming messages and noteworthy physical measurements
- Generated at high rate

Event Handler & Local DB
- Event handler summarize local events and store them to DB

Periodic processing the information in DB
- Analysis and incorporation historical information

Response to the events
- Forward it to parent
- Issue local optimization commands

Question?