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

Globus Toolkit Version 4 : Software for Service-Oriented System

What's the Globus?

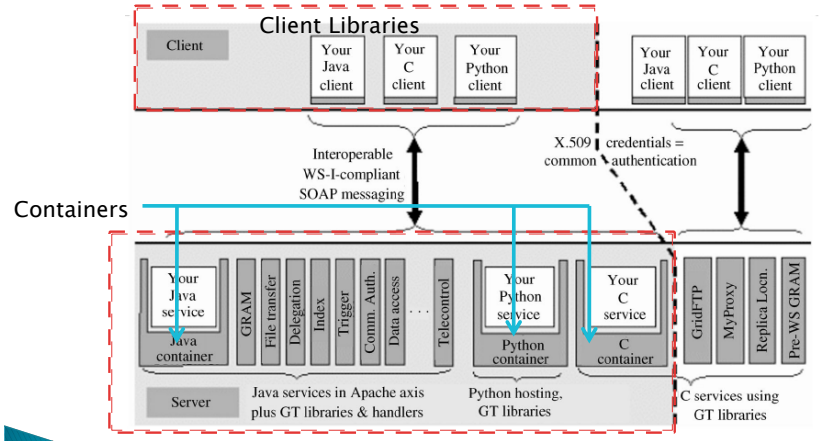
- Globus ?
  - Community : virtual organizations.
  - Software itself. -> GT
  - Infrastructure that supports this community.
    : code repository, list of mails, problem tracking system …
    All accessible at ***dev.globus.org***
- GT(Globus Toolkit)
  - GT2 in 2002 – Introducing OGSA
  - GT3 in 2003 – based on OGSI
  - GT4 in 2004 – based on WSRF
    : Web Service Resource Framework
  - Current version 5.0.3

<Fig 1.> Grid & Web Service
http://blog.naver.com/juno72/120024214344

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



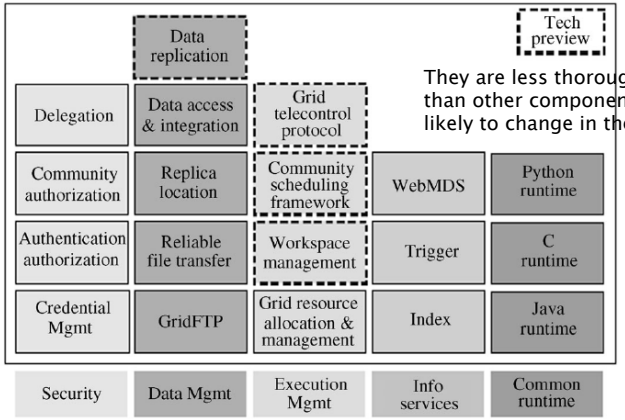
A set of Service Implementation

* White Box : User Code
Shaded Box : GT4 Code

GT4 Architecture

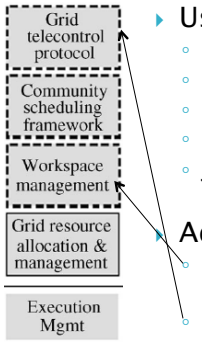
- Communication among components
  - WS-I compliant SOAP messaging
  - X.509 : an ITU-T standard for a PKI, SSO and PMI
- User Programming?
  - Implementing Service
    - Using Container(Java, C and Python)
    - Own Services using GT libraries (C)
  - Implementing Client
    - Using Client Libraries
    - Own Client Program (Communication via X.509)

GT4 S/W Details



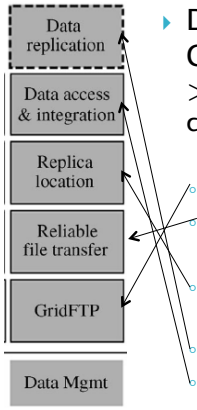
They are less thoroughly tested than other components and likely to change in the future.

How to manage Execution?



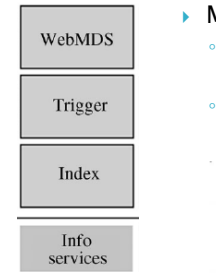
- GRAM(Grid Resource Allocation and Manage ment) Service
- Using GRAM, Client can express followings
  - The type and quantity of desired resource
  - Data to be staged to and from the execution site
  - The executable and its arguments
  - Credentials to be used
  - Job persistence requirements
- Additional Component
  - WMS : Dynamic execution sandbox using virtualization(Xen)
  - GTCP : Managing instrumentations such as earthquake engineering, microscopes…

How to manage Data?

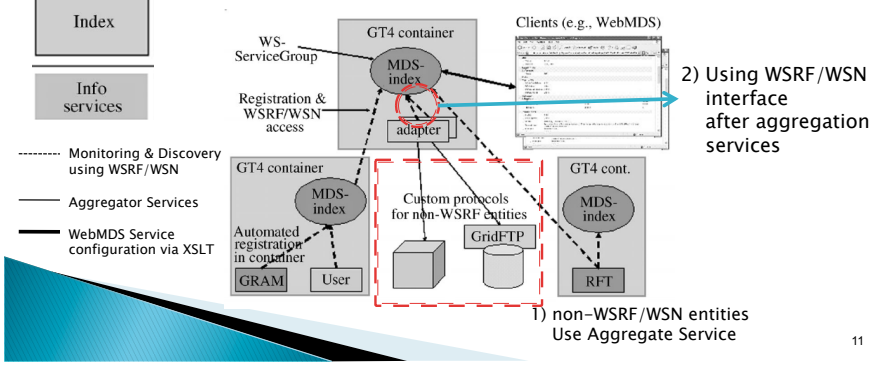


- Data problem is broad and complex in Grid Problems
  > GT Supports reliable and high-performance components for accessing and moving data
  - GridFTP : Data transfer libraries & tools
  - RFT : Manage multiple GridFTP
    - millions files between two astronomy archives
  - RLS : Manage location of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

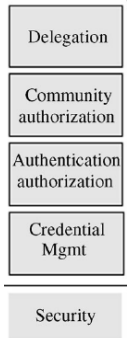
How to manage Service?



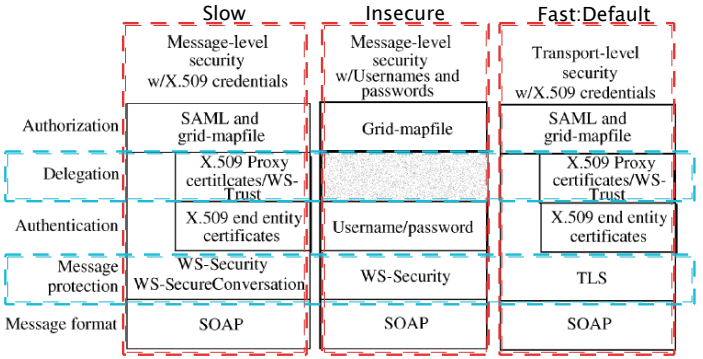
- Monitoring and Discovery of Services
  - Using WSRF/WSN interface
    - Supported by container
  - Aggregator Services(For non-WSRF/WSN entities)
    - Index(register) / Trigger(filter)



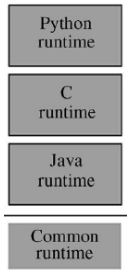
How to manage Security?



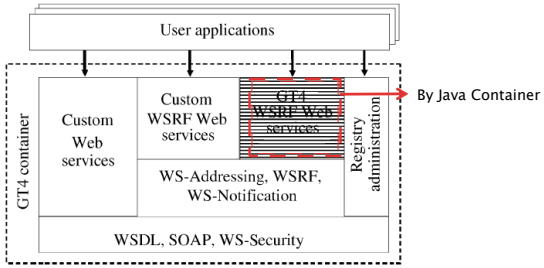
- Three types of security supporting
- Two entities to validate each other's credential
  - Message Protection : establish secure channel
  - Delegation : delegate credential to a remote component



How to build new services?



- GT4 Support common development components
  - Containers!
- Containers(Java, C, Python) can supports
  - 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…

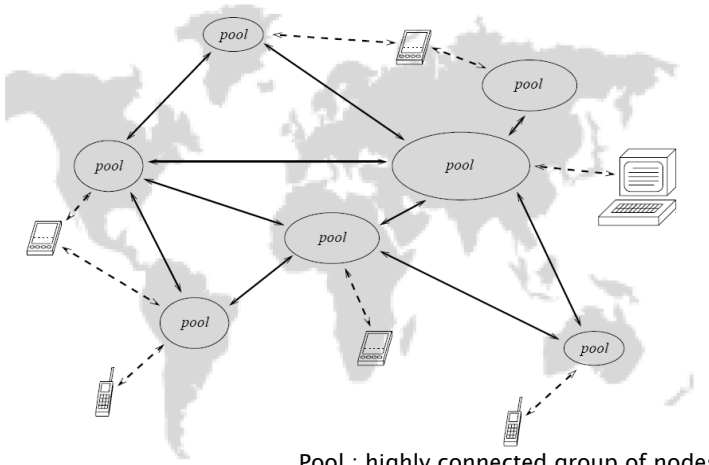
- Official Homepage of OceanStore
  - http://oceanstore.cs.berkeley.edu
  - last modified on … … 07/08/2002
  - Members are busy? ☺
- The paper appeared in 2000, and the prototype was under development at that time, but
  - "Pond: the OceanStore Prototype" was published in 2003
- Now, they use "Tapestry" 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

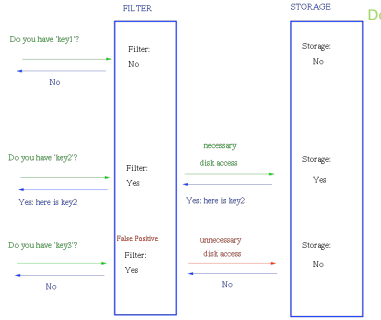
Naming & Access Control

- 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 authentificate using specified ACL

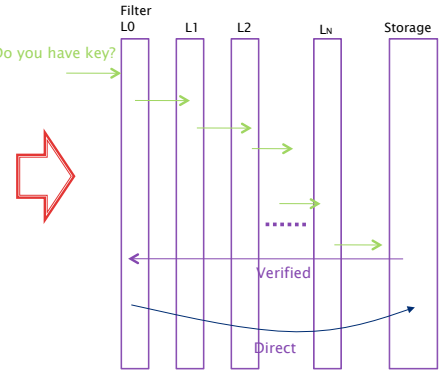
Data Location & Routing

- 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?



* http://en.wikipedia.org/wiki/Bloom_filter

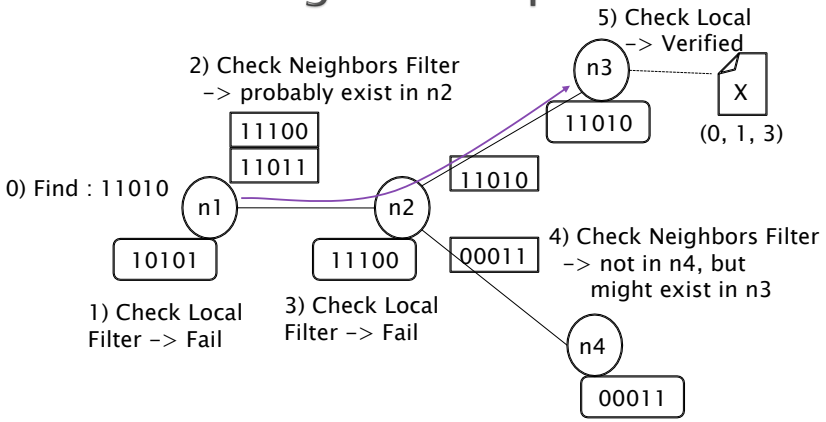


Overall answer speed is better with Bloom filter than without Bloom Filter

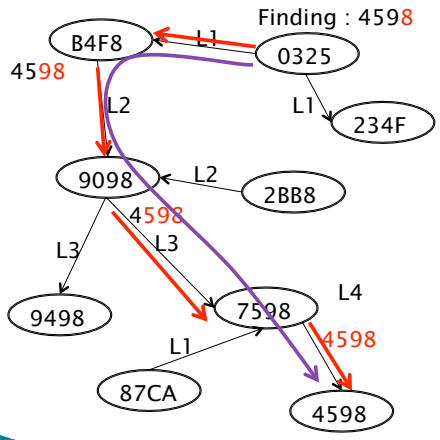
Attenuated Bloom Filter :
Query propagates through nodes if there is no local matching

* Image is simplified

Fast Routing – Example



Global Algorithm-Slower Routing



Routing Table for 0325

	0	…	8	…	F
L1	XXX0	…	B4F8	…	XXXF
L2	XX05	…	XX85		XXF5
L3	X025	…	X825		XF25
L4	0325	…	8325	…	F325

Routing Table for B4F8

	0	…	9	…	F
L2	XX08	…	9098		XXF8

Routing Table for 9098

	0	…	5	…	F
L3	X098	…	7598		XF98

Routing Table for 7598

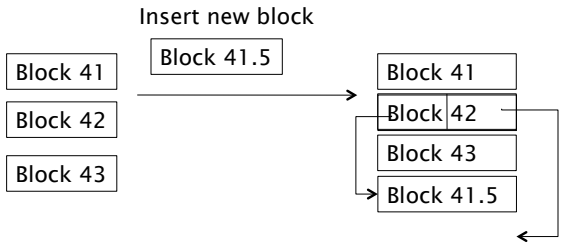
	0	…	4	…	F
L3	0598	…	4598		F598

Update

- Designed based on Conflict Resolution
  - Introduced in Bayou System(http://www2.parc.com/csl/projects/bayou)
  - Need computation in server side (Serializing…)
- 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 ciphetext[47]
  - *delete-block* : replace with empty pointer block
  - *insert-block* : see next page

[47] D.Song, D.Wagner, and A. Perrig. Search on encrypted data.

Update : *insert-block*

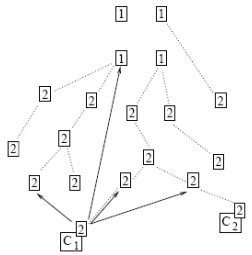


Update – Serializing updates

- 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

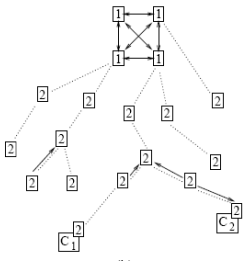
Update – Serializing updates

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

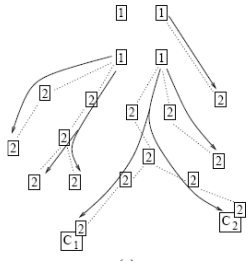


Update – Serializing updates

- 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



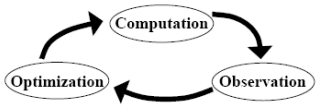
Update – Serializing updates



- 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*

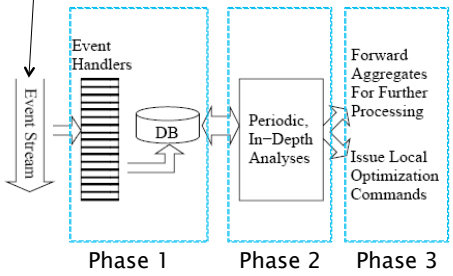


The Cycle of Introspection

- 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 i nformation in DB
  - Analysis and incorporation historical information
- Response to the events
  - Forward it to parent
  - Issue local optimization commands

Question?