a scalable proxy cache for grid data access

Cristian Cirstea, Jan Just Keijser, Ronald Starink, Oscar Koeroo, Jeff Templon
CHEP 2012, NYC, 24 May 2012
Associated Problem

- “User” Access to grid data
  1. Avoid having to know where data is (job failures)
  2. Avoid having to install entire gLite stack
  3. Use standard protocols (e.g., posix or http)
  4. Convenient browsing (file system view)
- Anticipate “grad student” use case
  - Repeated access to selected set of data
The vision
(René’s version)

100 Gbits/s

This is us …
Amsterdam DM Workshop, June 2010

See also I. Bird talk “TEG recommendations” Sunday, slides 4 & 5 … and René’s plenary talk

Focus of this talk

Analysis Clusters

LCG CDN

proxy disk

T1

T0

T2

J. Templon
Nikhef
Amsterdam
Physics Data Group

Jeff Templon, CHEP 2012, NYC
Design Choices

1. Access via http(s)
2. Provide webDAV
   -> works on user desktop w/o mw
3. Support horizontal scaling
4. Avoid writing code if possible
   (not completely possible : must deal with data on grid : GSI, LFC, SRM)
5. VOMS aware across all clients
System Overview

Diagram showing the flow of protocols and interaction between components:
- User communicates via HTTPS protocol to WebDAV Server.
- WebDAV Server uses WebDAV protocol and communicates via HTTPS to Distributed System.
- Distributed System includes Cache Nodes 1, 2, and n, each communicating via HTTPS.
- Cache Nodes 1 and 2 communicate with Worker Node and Storage Element via Grid-specific protocol.
- WebDAV Server also communicates via Grid-specific protocol to LFC.
WebDAV seq diagram

1: authenticate user
2: PROPFIND (browse directory)
3: PROPFIND response (directory content)
4: GET (open file)
5: REDIRECT to Cache Node
6: authentication + GET (open file)
7: send the file

2.1: request the directory content
2.2: content of the directory

Note: redirect is to a random cache node!
HTTP Client

1: authentication + GET (download file)

Instance 1: Cache Node

1.1: check the cache (file is not cached)
1.2: check if the file is cached
1.3: file is NOT cached
1.4: require the file from SE
1.5: file content
1.6: cache the file
1.7: file content

Instance n: Cache Node

Storage Element
File Cached

HTTP Client

1: authentication + GET
  (download file)

Instance 1: Cache Node
1.1: check the cache
  (file is not cached)
1.2: check if the file is cached
1.3: file is cached

Instance n: Cache Node

Storage Element

2: REDIRECT to Cache Node
Instance n

3: authentication + GET
  (download file)

4: file content
Load Balancing & Passive Stability

- webDAV redirects to a random CN
- If that node doesn’t have file, queries others
- Burst of requests: multiple CNs start to retrieve file …
- Hot file: CN hosting file does not answer quickly enough, a 2nd node retrieves file
coding

- About 700 lines total code
- Mostly standard stuff: apache2 plus gridsite, NginX
- Hand-crafted stuff:
  - Cache system
  - WebDAV server (support redirects, grid interface)
  - Wdfs kernel module mod (support redirects)
  - Htproxyput mod (CN string format)
Deployment tests

Deployment environment

- tbn21.nikhef.nl
  - Cache Node

- tbn22.nikhef.nl
  - Cache Node

Test environment

- Worker Nodes (13 machines)
  - tbn23.nikhef.nl
  - 10Gb/sec

Grid Environment

- Storage Element
  - hooimaand-01
    - hooimaand-02

- lfc.grid.sara.nl
  - LFC Server

- px.grid.sara.nl
  - MyProxy Server
“hot file” results, single CN

![Graph showing average duration of a job vs. concurrent jobs]

- WebDAV (data is not cached)
- lcg-cp

![Graph showing average duration of a job vs. concurrent jobs]

- WebDAV (data is cached)
- lcg-cp
“hot file”, 3 CNs
Different files, 1 CN

- Breakpoint: \# files x 1GB = RAM size
- SE RAM & Disk specs much better than CN
• Breakpoint now at $60 = 3 \times 20$
• Verifies horizontal scaling!!
Same test but smaller files

![Graph showing average duration of a job (sec) vs concurrent jobs. The graph compares WebDAV (data is not cached), WebDAV (data is cached), and lcg-cp.]
Solved Associated Problems?

- “User” Access to grid data
  1. Avoid having to know where data is (job failures) YEP
  2. Avoid having to install entire gLite stack YEP
  3. Use standard protocols (eg posix or http) YEP
  4. Convenient browsing (file system view) YEP

- Grad stud case: = for 1 CN, better if > 1
Conclusions

• Achieved working prototype at scale
• Possible with only 700 loc
• System generally outperforms “lcg-cp” in hot file use case
• System is not significantly worse than “lcg-cp” when files are not reread, as long as total active data volume < total CDN RAM
Much more technical detail

- Also includes what was tried and did not work
- Details on modifications, gotchas (webDAV clients), directions for future work.