Towards a distributed scalable data service for the Grid

Marco Aldinucci
ISTI-CNR, Pisa, Italy

M. Danelutto, G. Giaccherini, M. Torquati, M. Vanneschi
CS dept. Uni. Pisa, Italy
Outline

- ADHOC (Adaptive Distributed Herd of Object Caches)
  - Motivation
  - Features
  - Why it is a Grid-aware software
- Applications & Experiments
  - Apache+ADHOC parallel web server architecture
  - ADHOC-based DSM for ASSIST
  - ADHOC-based Parallel Virtual File System (astFS)
- Ongoing & Future work
ADHOC (Adaptive Distributed Herd of Object Caches)

- A very basic storage facility
  - No hardwired policies for deployment, allocation, data coherence, ...
  - pluggable into different, third-party applications/frameworks
- proving **data management** as external service for applications
  - implemented as high-throughput distributed server
- decoupling computational and storage management in (distributed) application design
  - enforcing a structured development
- and exploiting persistency, scalability, re-configurability
Permanent, shared storage facility

- a facility (distributed server) providing permanent, shared storage to apps (clients)
- clients may dynamically join/leave the storage facility
- HOC set may be hotly enlarged/reduced on need - storage room change accordingly
- interaction with HOCs may be delegated to application-specific protocol (proxy)
Permanent, shared storage facility

- a facility (distributed server) providing permanent, shared storage to apps (clients)
- clients may dynamically join/leave the storage facility
- HOC set may be hotly enlarged/reduced on need - storage room change accordingly
- interaction with HOCs may be delegated to application-specific protocol (proxy)
Permanent, shared storage facility

- a facility (distributed server) providing permanent, shared storage to apps (clients)
- clients may dynamically join/leave the storage facility
- HOC set may be hotly enlarged/reduced on need - storage room change accordingly
- interaction with HOCs may be delegated to application-specific protocol (proxy)
Permanent, shared storage facility

**distributed storage facility**

- A facility (distributed server) providing permanent, shared storage to apps (clients)
- Clients may dynamically join/leave the storage facility
- HOC set may be hotly enlarged/reduced on need - storage room change accordingly
- Interaction with HOCs may be delegated to application-specific protocol (proxy)
Permanent, shared storage facility

- a facility (distributed server) providing permanent, shared storage to apps (clients)
- clients may dynamically join/leave the storage facility
- HOC set may be hotly enlarged/reduced on need - storage room change accordingly
- interaction with HOCs may be delegated to application-specific protocol (proxy)
Permanent, shared storage facility

- A facility (distributed server) providing permanent, shared storage to apps (clients).
- Clients may dynamically join/leave the storage facility.
- HOC set may be hotly enlarged/reduced on need - storage room change accordingly.
- Interaction with HOCs may be delegated to application-specific protocol (proxy).
Permanent, shared storage facility

- A facility (distributed server) providing permanent, shared storage to apps (clients).
- Clients may dynamically join/leave the storage facility.
- HOC set may be hotly enlarged/reduced on need - storage room change accordingly.
- Interaction with HOCs may be delegated to application-specific protocol (proxy).
Permanent, shared storage facility

- A facility (distributed server) providing permanent, shared storage to apps (clients).
- Clients may dynamically join/leave the storage facility.
- HOC set may be hotly enlarged/reduced on need - storage room change accordingly.
- Interaction with HOCs may be delegated to application-specific protocol (proxy).
Permanent, shared storage facility

- A facility (distributed server) providing permanent, shared storage to apps (clients).
- Clients may dynamically join/leave the storage facility.
- HOC set may be hotly enlarged/reduced on need - storage room change accordingly.
- Interaction with HOCs may be delegated to application-specific protocol (proxy).
Permanent, shared storage facility

- A facility (distributed server) providing permanent, shared storage to apps (clients).
- Clients may dynamically join/leave the storage facility.
- HOC set may be hotly enlarged/reduced on need - storage room change accordingly.
- Interaction with HOCs may be delegated to application-specific protocol (proxy).
Permanent, shared storage facility

- a facility (distributed server) providing permanent, shared storage to apps (clients)
- clients may dynamically join/leave the storage facility
- HOC set may be hotly enlarged/reduced on need - storage room change accordingly
- interaction with HOCs may be delegated to application-specific protocol (proxy)
Why using HOC

- is efficient (because essential)
  - HOC provide few primitives and no policies for data integrity (e.g. coherence, consistency, ...):
    - these are application specific and may be deployed upon HOC (at the proxy level)
- is a basic building block for broad class of applications
  - may be considered a storage component
  - massive storage, out-of-core applications, high-throughput data servers, shared memory support
  - extendible with application-specific primitives
- enhances both memory size and throughput by means of parallelism
HOC API

Why does the web work so well?
A language with few verbs (get, put, post) ...
Gannon said ... (Europar04, invited talk)

- get, put, remove arbitrary length objects. Each object is identified by a key and a home node
- `execute(key, op, data)` remotely execute method `op` with parameter `data` on object identified by `key`
Why does the web work so well? A language with few verbs (get, put, post) ... Gannon said ... (Europar04, invited talk)

We also believe on such philosophy. As matter of a fact HOC have a four operations API

- **get**, **put**, **remove** arbitrary length objects. Each object is identified by a key and a home node

- **execute(key, op, data)** remotely execute method **op** with parameter **data** on object identified by **key**
Objects & collections of them

- objects and collections of them
- both indexed by fixed length key
- objects are atomic
- collections distributed and replicated by means of a dynamic schema (as many as you want):
  - spread group
  - replica group
- schemas can be added & changed at runtime

\[ \{ \text{G1}, \text{G2}, \text{G3}, \text{G4} \} \]

\[ \text{G1} = [\text{S1}]; \text{G2} = [\text{S2}]; \text{G3} = [\text{S3}]; \text{G4} = [\text{S4}] \]
Objects & collections of them

- Objects and collections of them
- Both indexed by fixed length key
- Objects are atomic
- Collections distributed and replicated by means of a dynamic schema (as many as you want):
  - Spread group
  - Replica group
- Schemas can be added & changed at runtime

K = \{a, b, c, d, e\}

1 = \{G1, G2, G3, G4\}
G1 = [S1]; G2 = [S2]; G3 = [S3]; G4 = [S4]

2 = \{G5, G6\}
G5 = [S1, S3]; G6 = [S2, S4]
ADHOC is Grid-aware

not because the paper is fulfilled of typical Grid buzzwords, but because it addresses underlying features of grid platforms ...

- connectivity: through the firewalls, multi-tier networks
- parallelism (speed), distribution (memory size), replication (availability), caching (self-optimization) by means of a dynamic & flexible object keys creation mechanism
- add/remove nodes with no data loss) (adaptivity), data migration (load balancing), data robustness (fault-tolerance)
- heterogeneous platforms, it may be deployed through standard middleware (standards), tolerate job schedulers through a lazy wiring mechanism, it can be wrapped by means of WS
Through the firewalls

Network with private addresses

Network with private addresses (e.g. cluster)
## ADHOC performance figures (1PE)

<table>
<thead>
<tr>
<th>Arch/Net/OS</th>
<th>concurrent connections</th>
<th>Msg size (Bytes)</th>
<th>Replies/Sec</th>
<th>net throughput (Bytes/Sec)</th>
<th>net throughput w.r.t. ideal</th>
</tr>
</thead>
<tbody>
<tr>
<td>P4@2GHz</td>
<td>2048</td>
<td>1 M</td>
<td>91</td>
<td>91 M</td>
<td>96%</td>
</tr>
<tr>
<td>Mem 512MB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GigaEth</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Linux ker.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.4.22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3@800MHz</td>
<td>3072</td>
<td>512</td>
<td>20 M</td>
<td>10 M</td>
<td>11%</td>
</tr>
<tr>
<td>Mem 1GB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FastEth</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Linux ker.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.4.18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3@800MHz</td>
<td>1024</td>
<td>8 K</td>
<td>1429</td>
<td>11.2 M</td>
<td>90%</td>
</tr>
<tr>
<td>Mem 1GB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FastEth</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Linux ker.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.4.18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3@800MHz</td>
<td>1024</td>
<td>16 K</td>
<td>718</td>
<td>11.2 M</td>
<td>90%</td>
</tr>
<tr>
<td>Mem 1GB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FastEth</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Linux ker.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.4.18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Sustained aggregate throughput

Processing Elements

Aggregate throughput (MB/sec)
Sustained aggregate throughput

![Graph showing sustained aggregate throughput with different processing elements and object counts.]
ADHOC is a building block for storage-oriented components.

- Distributed caches, distributed memories, parallel repositories
- Configurable, hot-pluggable, grid-aware

- Very good performances
  - Close-to-ideal net throughput over thousands of concurrent connections
  - Close-to-ideal speedup

- See M. Aldinucci, M. Torquati paper @ EuroPar 2004, LNCS 3149
- **ADHOC cache plugin for Apache**
  - Big picture & features
  - Performances & scalability
    (both of them very good)

- **ADHOC-based DSM**
  - Big picture & features
  - Performances

- **ASTFS (ADHOC-based FS)**
  - Big picture & features
  - Performances
The Apache Web server

- Worldwide most used Web server
  - broadly accepted, well-known, well supported
  - opensource

- MultiThread-MultiProcessor Web server
  - good performance, nevertheless several attempts to improve yet more performances
  - usually used in farm configurations

- Easy to extend via plug-in modules
  - already existing “native” memory-based cache module
How accelerate a web server/service

- farming servers out
- caching, typically reverse proxy (in front of the server)
  - worsen requests latency (miss)
  - complex as much as the web server
How accelerate a web server/service

- farming servers out
- caching, typically reverse proxy (in front of the server)
  - worsen requests latency (miss)
  - complex as much as the web server

We would like to improve web server performance without changing web server core, thus relying on correctness, people expertise, ...
How accelerate a web server/service

- farming servers out
- caching, typically reverse proxy (in front of the server)
  - worsen requests latency (miss)
  - complex as much as the web server

We would like to improve web server performance without changing web server core, thus relying on correctness, people expertise, ...

... thus we add an HOC-based distributed cache behind the server (or the server farm)
The Apache plug-in for HOC
The Apache plug-in for HOC

Apache 2.0.52

patched mod_mem_cache
The Apache plug-in for ADHOC

High-level functional behavior of the Apache 2.0.52 native cache module (mod_mem_cache)
High-level functional behavior of the protocol for ADHOC+Apache architecture (a simple patch to mod_mem_cache)
Comparing reply rate (1-ADHOC/1-Apache/k-httpperf)

SingleProcessMultiThreaded Apache (900MB shared native cache)
Comparing reply rate (1-ADHOC/1-Apache/k-httperf)

SingleProcessMultiThreaded Apache (900MB shared native cache)
Comparing reply rate (1-ADHOC/1-Apache/k-httperf)

NoCache SPMT Apache (FileSystem buffer behaves as cache)
Comparing reply rate (1-ADHOC/1-Apache/k-httperf)

NoCache SPMT Apache (FileSystem buffer behaves as cache)
Comparing reply rate (1-ADHOC/1-Apache/k-httperf)

SPMT Apache with 900MB ADHOC on 2 boxes
Comparing reply rate (1-ADHOC/1-Apache/k-httperf)

SPMT Apache with 900MB ADHOC on 2 boxes
Apache $2n$-farm vs Apache+ADHOC $n$-farm

```
eth0
PEa
httpperf
Apache
PE2
HOC
PE3
eth1
plug-in
PE0
httpperf
Apache
PE1

2n Apache

n Apache+HOC
```
Apache 2n-farm vs Apache+ADHOC n-farm

Replies per second

Processing Elements

2n Apache
n Apache+HOC
Apache 2n-farm vs Apache+ADHOC n-farm

HOC+Apache farm outperform standard farm by 3x with equal HW resources

+279%  +271%  +268%  +312%

Replies per second

0  750  1.500  2.250  3.000

Processing Elements

2n Apache  n Apache+HOC
○ ADHOC cache plugin for Apache
  ○ Big picture & features
  ○ Performances & scalability (both of them very good)

○ ADHOC-based DSM
  ○ Big picture & features
  ○ Performances

○ ASTFS (ADHOC-based FS)
  ○ Big picture & features
  ○ Performances
Standard DSM vs ADHOC-DSM

Standard DSM

DSM lib
app

DSM lib
app

DSM lib
app

DSM lib
app
Standard DSM vs ADHOC-DSM

Standard DSM

- The DSM is a library
- All-to-all connections
- Difficult to manage:
  1. Heterogeneity of nodes
  2. Data mapping & migration
  3. Nodes hot-add & remove
  4. Data persistency
Standard DSM vs ADHOC-DSM

**Standard DSM**

- The DSM is a library
- All-to-all connections
- Difficult to manage:
  1. Heterogeneity of nodes
  2. Data mapping & migration
  3. Nodes hot-add & remove
  4. Data persistency

**ADHOC-based DSM**

- The DSM is external
- More general (subsumes the DSM as library)
- Eases the management of 1,2,3,4: data & computations may be independently mapped/migrated
ADHOC-DSM scalability

Intel Linux boxes, fast Eth connected, equal number of connections

- ADHOC references (read)
- ADHOC references (write)
- Max theoric.

Aggregate Bandwidth (MB/s)

N. of Processing Elements

Old references (read)
Old references (write)
ADHOC-DSM scalability

Intel Linux boxes, fast Eth connected, equal number of connections

- ADHOC references (read)
- ADHOC references (write)
- Old references (read)
- Old references (write)

Max theoric.

Aggregate Bandwidth (MB/s)

N. of Processing Elements
ADHOC-DSM scalability

Intel Linux boxes, fast Eth connected, equal number of connections

- Red circles: ADHOC references (read)
- Orange squares: ADHOC references (write)
- Blue circles: Old references (read)
- Blue squares: Old references (write)
- Yellow line: Max theoretical

Graph showing Aggregate Bandwidth (MB/s) vs. N. of Processing Elements.
ADHOC-DSM scalability

Intel Linux boxes, fast Eth connected, equal number of connections

- ADHOC references (read)
- ADHOC references (write)
- Old references (read)
- Old references (write)
- Max theor. (represented by a yellow line)

Graph showing Aggregate Bandwidth (MB/s) vs. N. of Processing Elements.
ADHOC-DSM scalability

Intel Linux boxes, fast Eth connected, equal number of connections

- **ADHOC references (read)**
- **Old references (read)**
- **ADHOC references (write)**
- **Old references (write)**
- **Max theoric.**

**Graph Details:**
- **Aggregate Bandwidth (MB/s)**
- **N. of Processing Elements**

- Data points for ADHOC references (read) show an increasing trend with an aggregate bandwidth reaching 120 MB/s as the number of processing elements increases.
- Data points for Old references (read) also show an increasing trend, but at a slightly lower rate.
- Data points for ADHOC references (write) follow a similar trend to read references but at a lower aggregate bandwidth.
- Old references (write) show a lower aggregate bandwidth compared to read references.
- The max theoric line indicates the maximum theoretical bandwidth that could be achieved.
ADHOC-DSM scalability

Intel Linux boxes, fast Eth connected, equal number of connections

- Red circles: ADHOC references (read)
- Red squares: ADHOC references (write)
- Blue circles: Old references (read)
- Blue squares: Old references (write)
- Yellow line: Max theoric.

Graph showing Aggregate Bandwidth (MB/s) vs. N. of Processing Elements.
ADHOC-DSM scalability

Intel Linux boxes, fast Eth connected, equal number of connections

- ADHOC references (read)
- ADHOC references (write)
- Old references (read)
- Old references (write)
- Max theoric.

Aggregate Bandwidth (MB/s)

N. of Processing Elements
- **ADHOC cache plugin for Apache**
  - Big picture & features
  - Performances & scalability (both of them very good)

- **ADHOC-based DSM**
  - Big picture & features
  - Performances

- **ASTFS (ADHOC-based FS)**
  - Big picture & features
  - Performances
PVFS vs astFS

- state-of-the-art parallel FS
- POSIX-like API
- uses aggregate bandwidth,
- requires full connectivity (forget firewalls)
- 1 client = 1 server
- centralized FAT (MNG)
- heterogeneous nodes

PVFS

PE 0
client₀
ION₀

PE 1
client₁
ION₁

PE n
clientₙ
IONₙ

MNG
PVFS vs astFS

PVFS

PE 0
client₀
ION₀

PE 1
client₁
ION₁

PE n
clientₙ
IONₙ

ASTFS

PE 0
client₀
ADHOC₀

PE 1
client₁
ADHOC₁

PE n
clientₙ
ADHOCₙ

Same API, any connectivity, number of client & server unrelated, cache, heterog supported.
astFS scalability

Intel Linux boxes, fast Eth connected

- PVFS
- Max theoretic.
- ASTFS (ADHOC)
- ASTFS (ADHOC, with cache)

Aggregate Bandwidth (MB/s)

N. of Processing Elements
Intel Linux boxes, fast Eth connected

- PVFS
- Max theor. (AGHOC)
- ASTFS (ADHOC, with cache)
Intel Linux boxes, fast Eth connected

Graph showing astFS scalability with different processing element counts and aggregate bandwidth in MB/s.

- PVFS
- ASTFS (ADHOC)
- Max theoretic.
- ASTFS (ADHOC, with cache)
Intel Linux boxes, fast Eth connected

- PVFS
- ASTFS (ADHOC)
- Max theoret.
- ASTFS (ADHOC, with cache)
astFS scalability

Intel Linux boxes, fast Eth connected

- PVFS
- ASTFS (ADHOC)
- Max theoret.
- ASTFS (ADHOC, with cache)

N. of Processing Elements vs. Aggregate Bandwidth (MB/s)
astFS scalability

Intel Linux boxes, fast Eth connected

- PVFS
- ASTFS (ADHOC)
- Max theoretic.
- ASTFS (ADHOC, with cache)

Aggregate Bandwidth (MB/s)

N. of Processing Elements
Conclusions

- ADHOC is building block for various kind of data management for clusters and Grid
  - excellent performance & scalability
  - enable the decoupled management of data and computations, usable as plugin (you don’t need to change the target app code)
  - support heterogeneous platforms, cope with firewalls, private networks, job schedulers, persistency, hot-adaptivity, fault-tolerance ...
- Is general enough to target different storage needs
  - indeed, we presented 3 different applications
  - just 10 student-months developing time (ADHOC excluded)
    - documentation excluded as well ;-)


ADHOC has been developed as part of the ASSIST programming toolkit ...