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

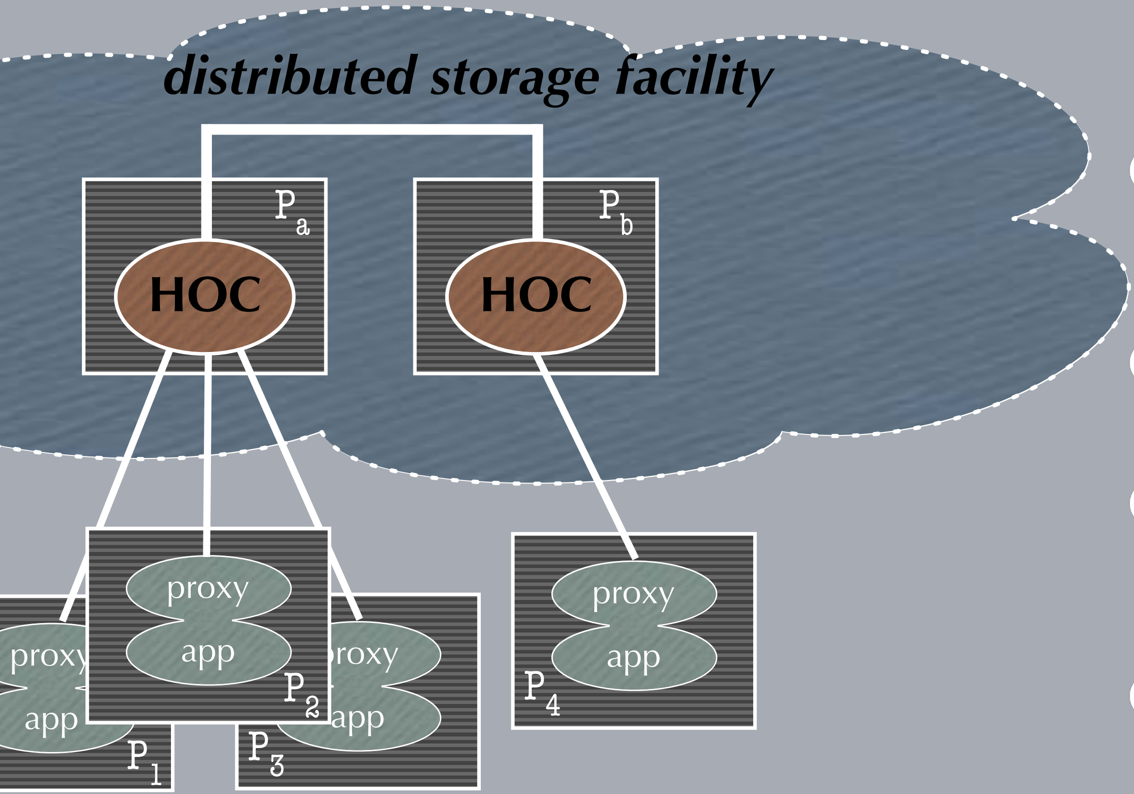
Outline

- 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)

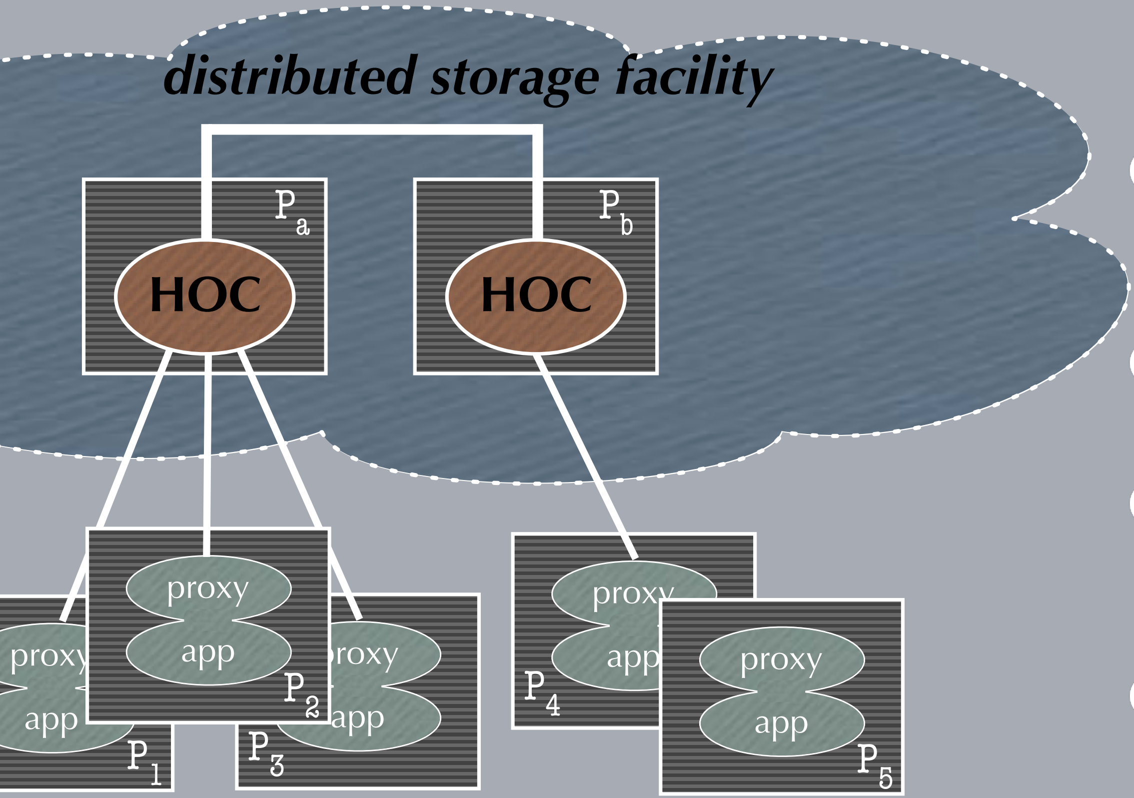
- A very basic storage facility
  - No hardwired policies for deployment, allocation, data coherence, ...
  - pluggable into different, third-party applications/frameworks
- proving ***data management*** as external service for applications
  - implemented as high-throughput distributed server
- decoupling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

Permanent, shared storage facility



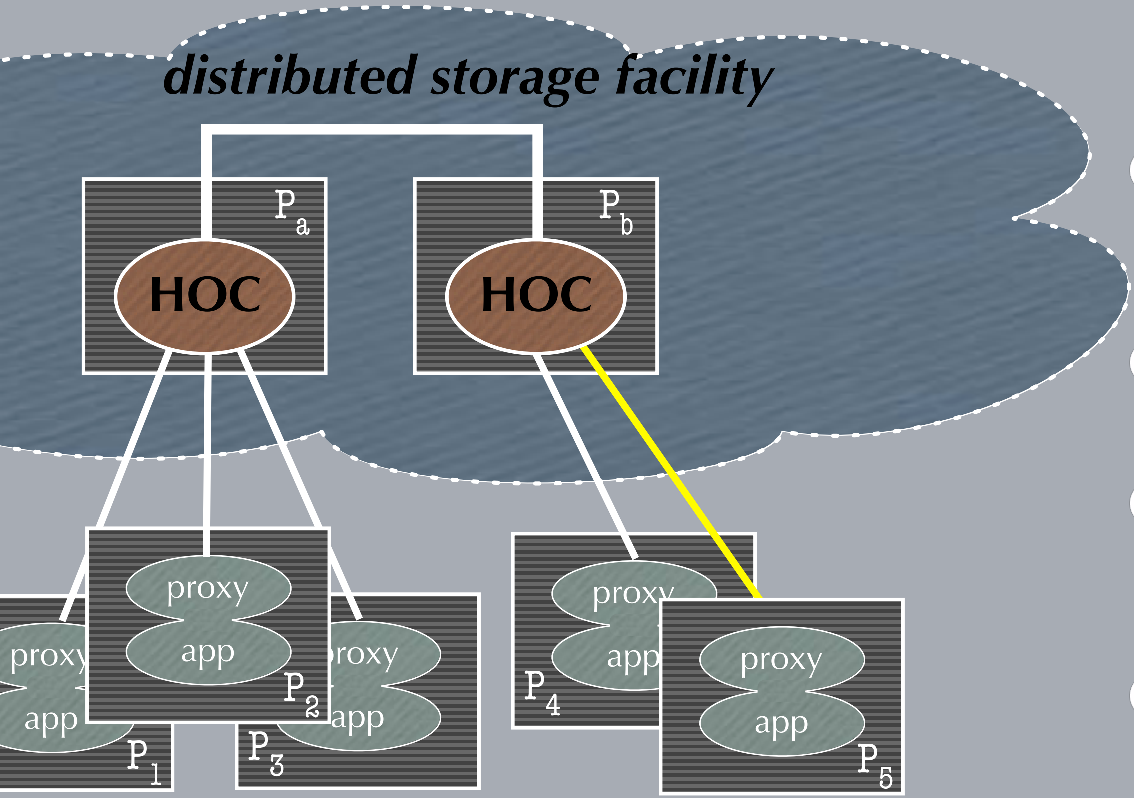
- a facility (distributed server) providing permanent, shared storage to apps (clients)
- clients may dynamically join/ leave the storage facility
- HOC set may be hotly enlarged/ reduced on need - storage room change accordingly
- interaction with HOCs may be delegated to application-specific protocol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

Permanent, shared storage facility



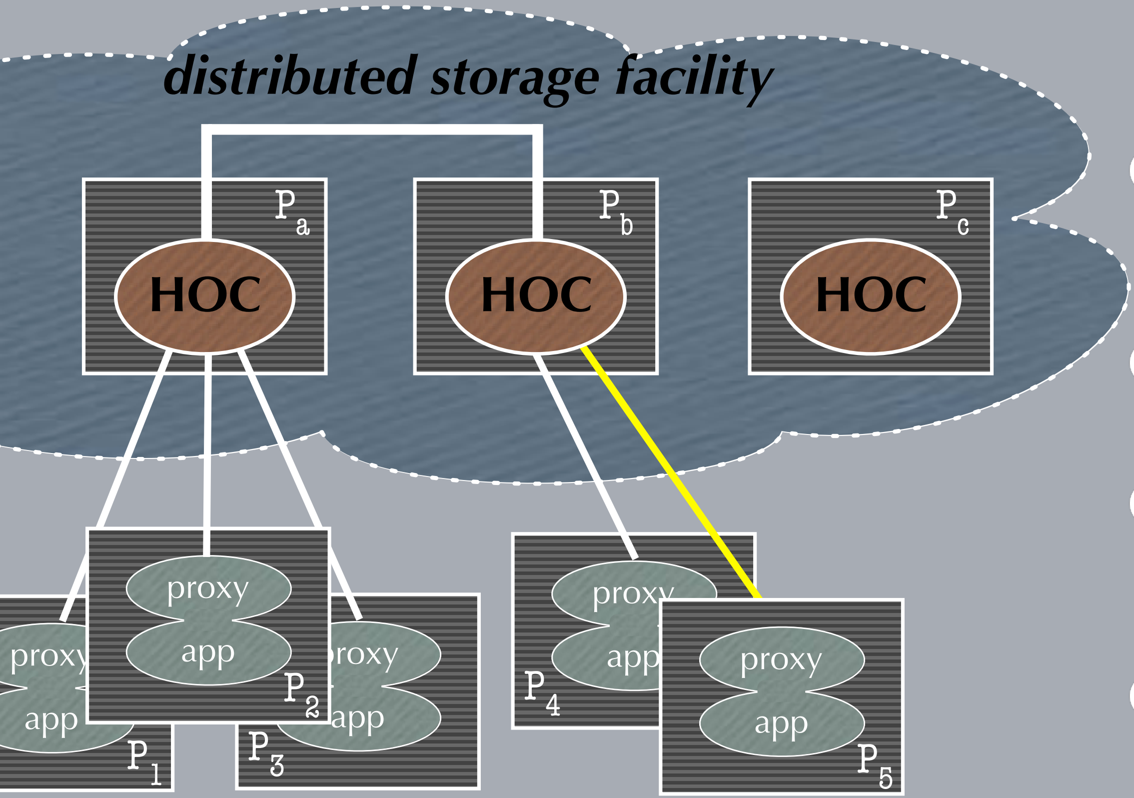
- a facility (distributed server) providing permanent, shared storage to apps (clients)
- clients may dynamically join/ leave the storage facility
- HOC set may be hotly enlarged/ reduced on need - storage room change accordingly
- interaction with HOCs may be delegated to application-specific protocol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

Permanent, shared storage facility



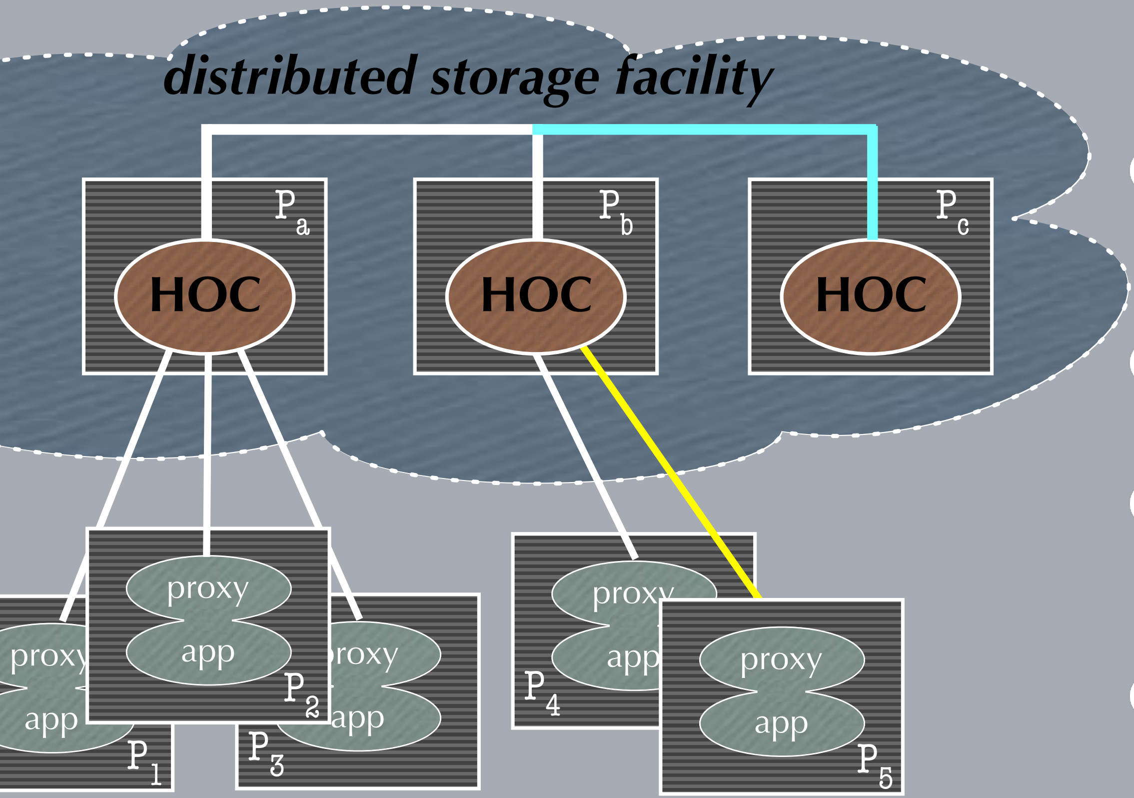
- a facility (distributed server) providing permanent, shared storage to apps (clients)
- clients may dynamically join/ leave the storage facility
- HOC set may be hotly enlarged/ reduced on need - storage room change accordingly
- interaction with HOCs may be delegated to application-specific protocol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

Permanent, shared storage facility



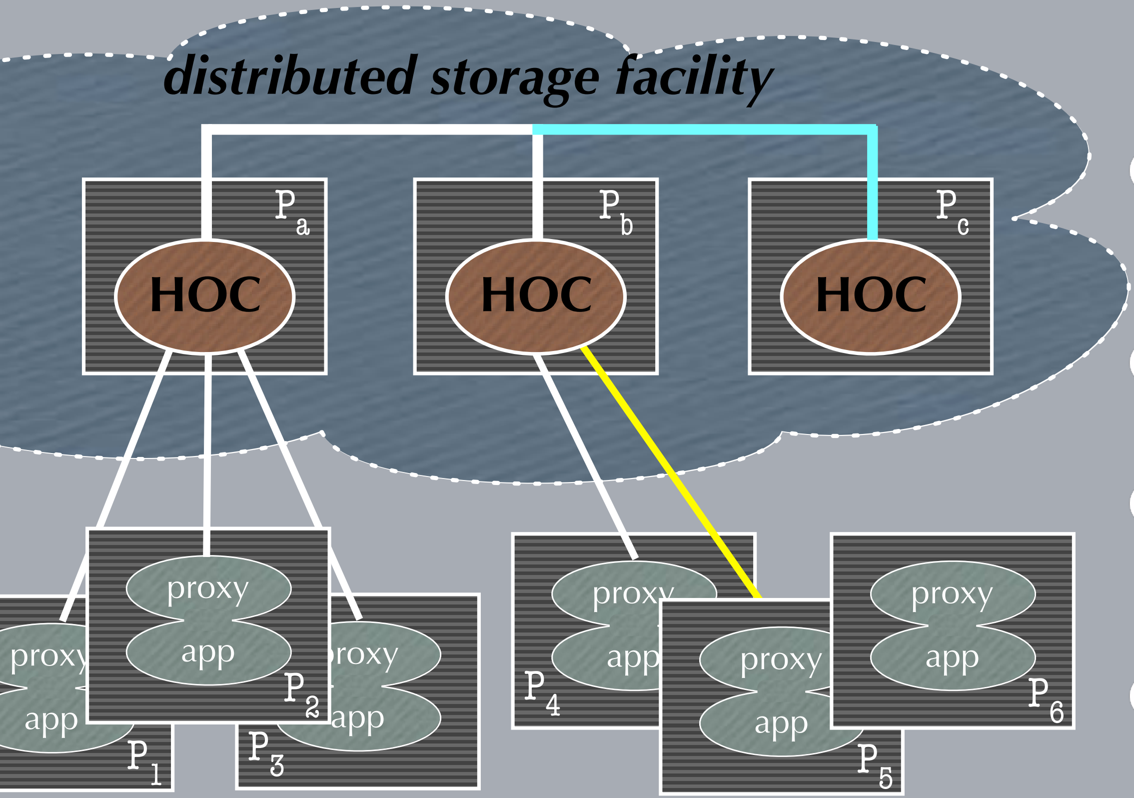
- a facility (distributed server) providing permanent, shared storage to apps (clients)
- clients may dynamically join/ leave the storage facility
- HOC set may be hotly enlarged/ reduced on need - storage room change accordingly
- interaction with HOCs may be delegated to application-specific protocol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

Permanent, shared storage facility



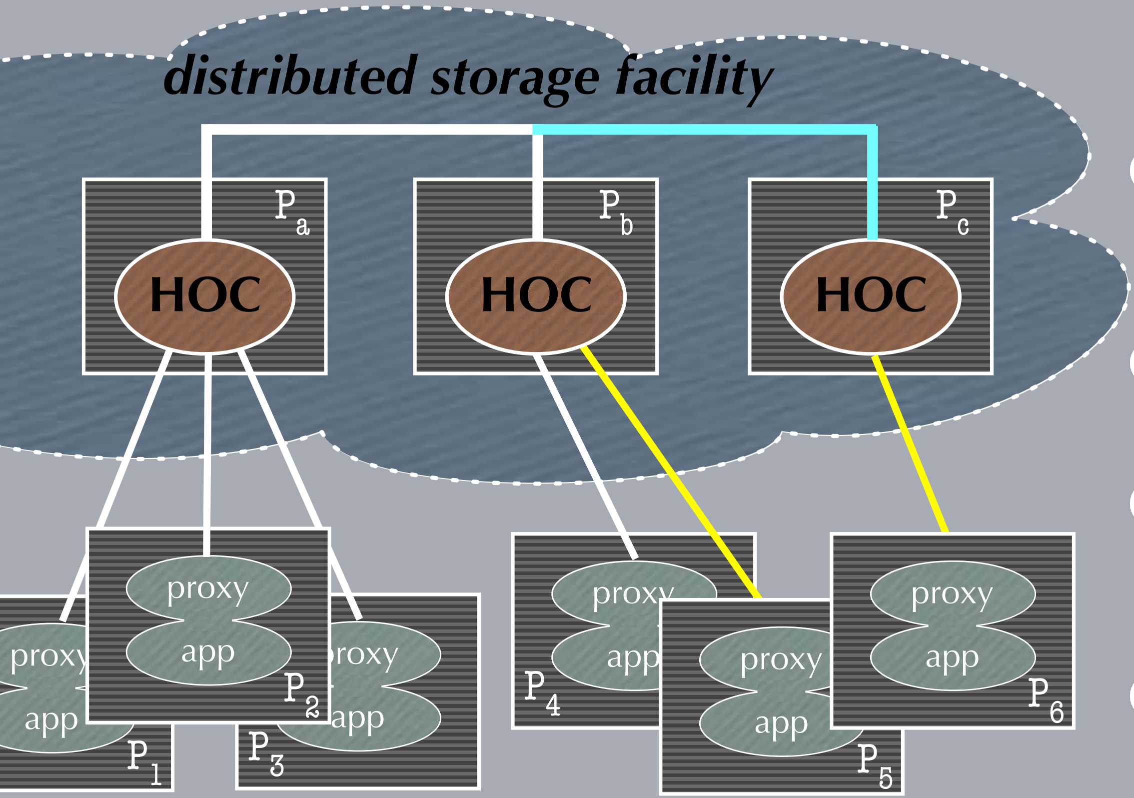
- a facility (distributed server) providing permanent, shared storage to apps (clients)
- clients may dynamically join/ leave the storage facility
- HOC set may be hotly enlarged/ reduced on need - storage room change accordingly
- interaction with HOCs may be delegated to application-specific protocol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

Permanent, shared storage facility



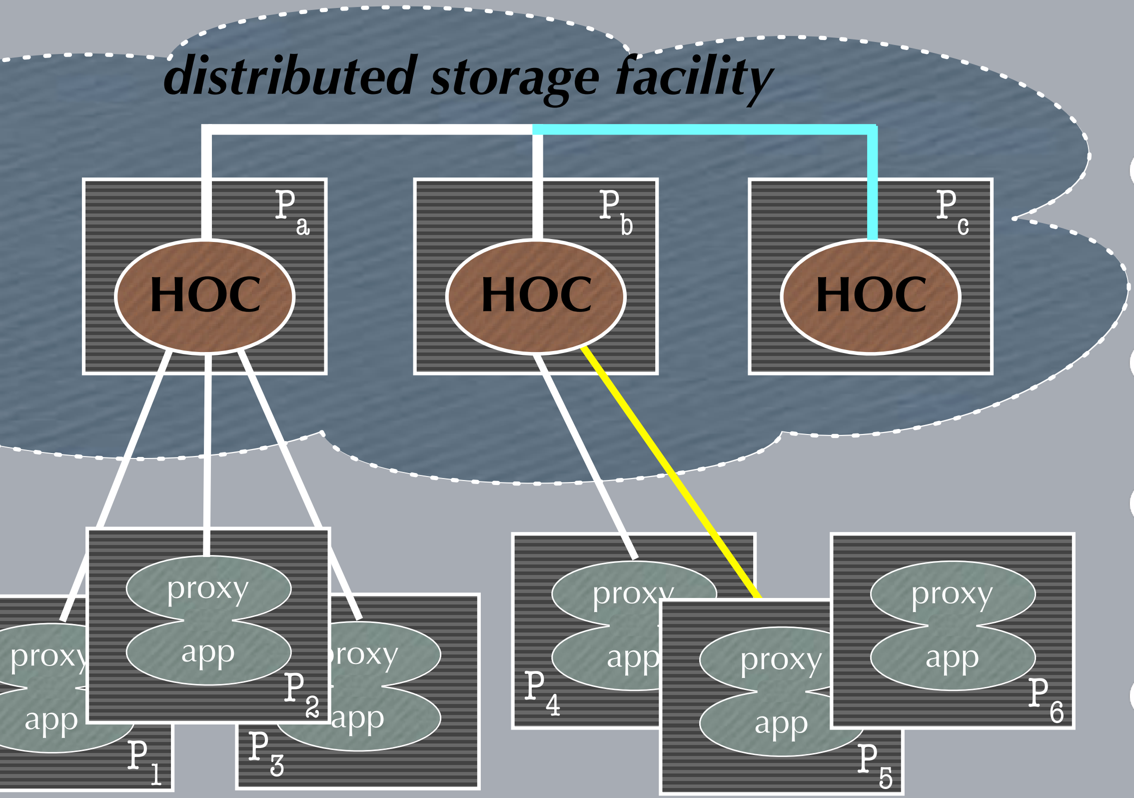
- a facility (distributed server) providing permanent, shared storage to apps (clients)
- clients may dynamically join/ leave the storage facility
- HOC set may be hotly enlarged/ reduced on need - storage room change accordingly
- interaction with HOCs may be delegated to application-specific protocol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

Permanent, shared storage facility



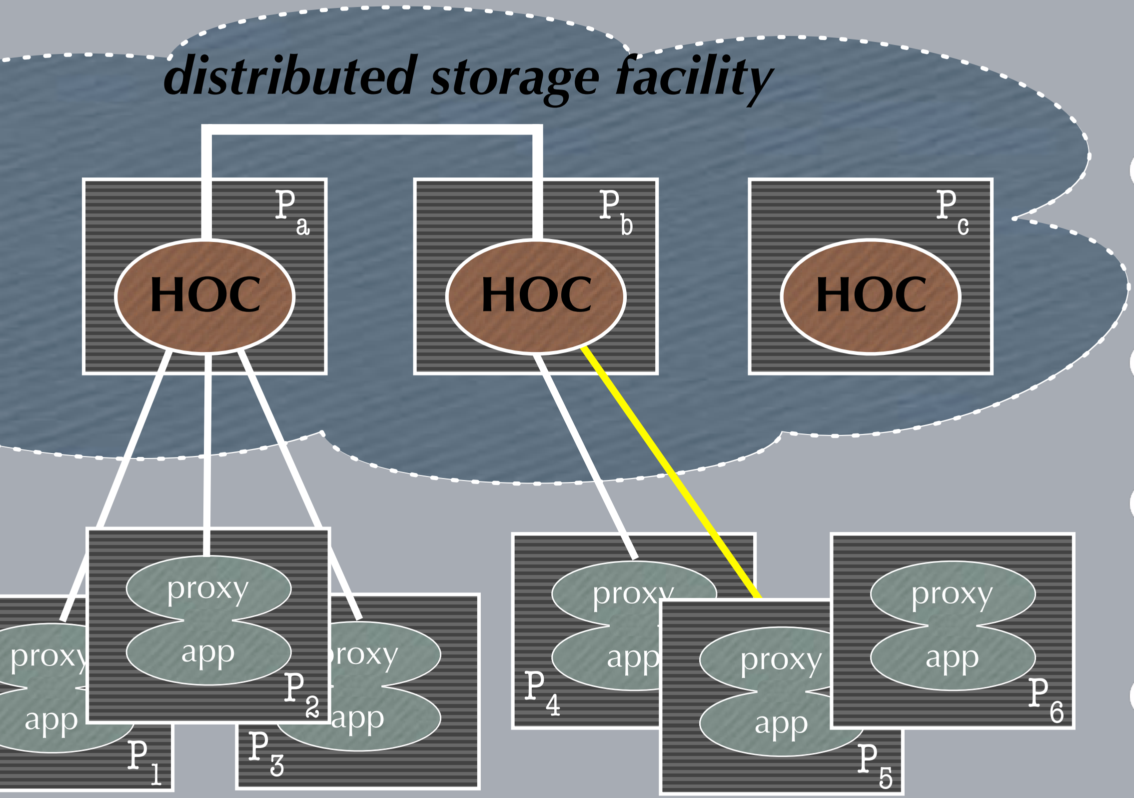
- a facility (distributed server) providing permanent, shared storage to apps (clients)
- clients may dynamically join/ leave the storage facility
- HOC set may be hotly enlarged/ reduced on need - storage room change accordingly
- interaction with HOCs may be delegated to application-specific protocol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

Permanent, shared storage facility



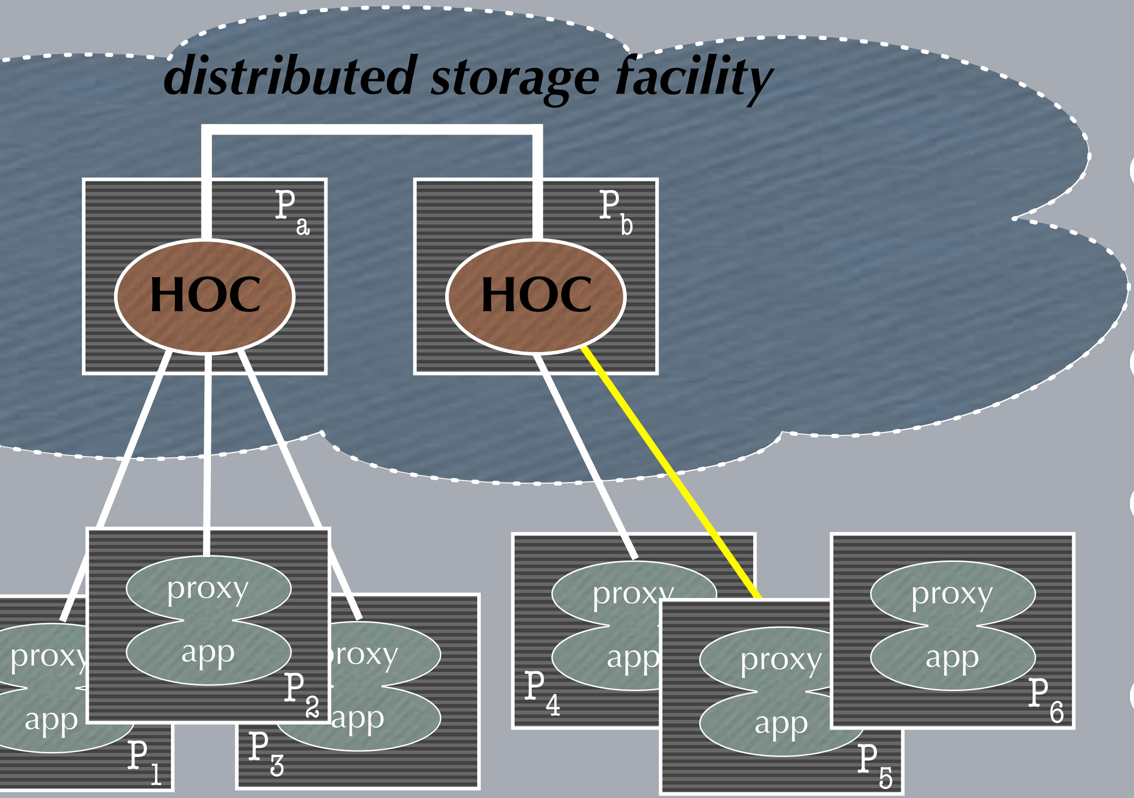
- a facility (distributed server) providing permanent, shared storage to apps (clients)
- clients may dynamically join/ leave the storage facility
- HOC set may be hotly enlarged/ reduced on need - storage room change accordingly
- interaction with HOCs may be delegated to application-specific protocol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

Permanent, shared storage facility



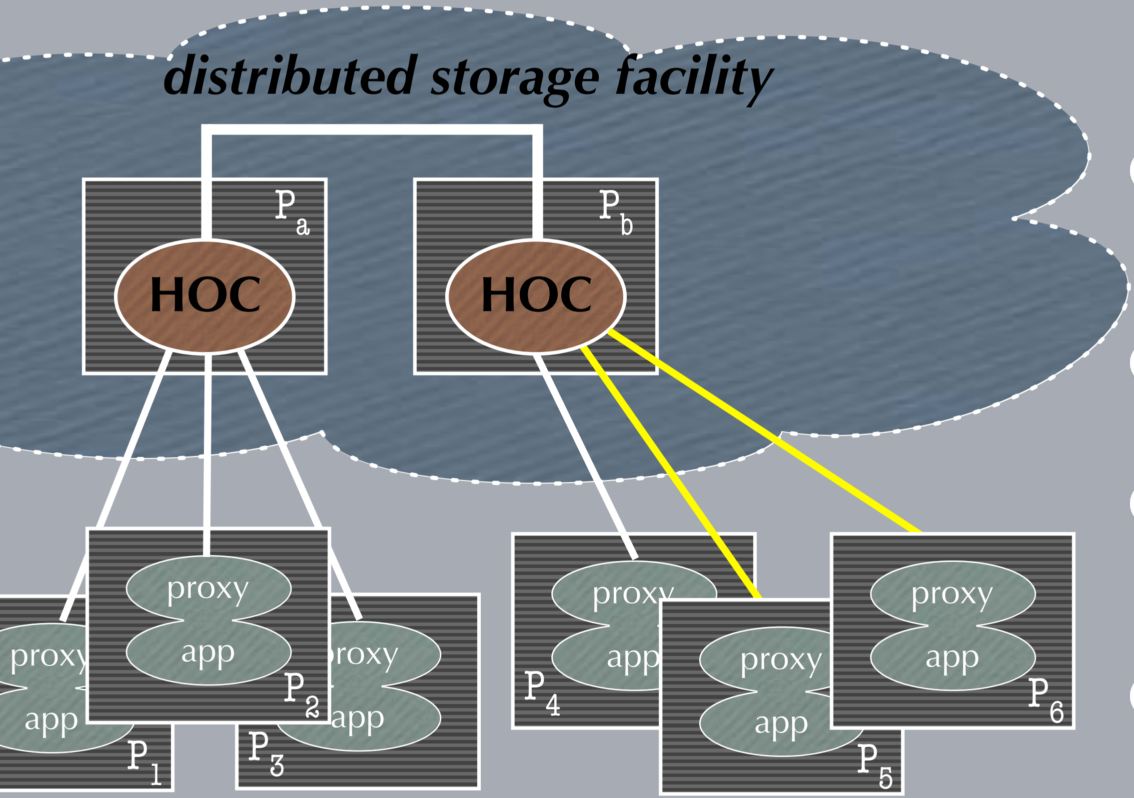
- a facility (distributed server) providing permanent, shared storage to apps (clients)
- clients may dynamically join/ leave the storage facility
- HOC set may be hotly enlarged/ reduced on need - storage room change accordingly
- interaction with HOCs may be delegated to application-specific protocol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

HOC API

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

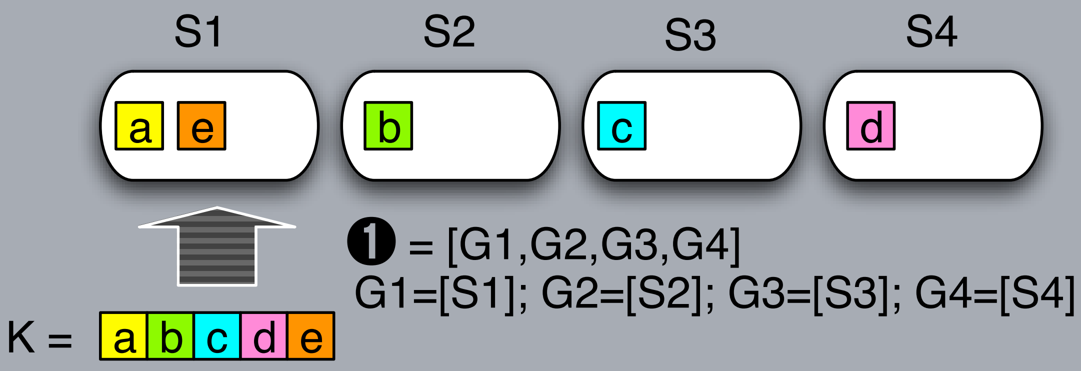
We also believe on such philosophy. As matter of a fact HOC have a four operations API

- `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

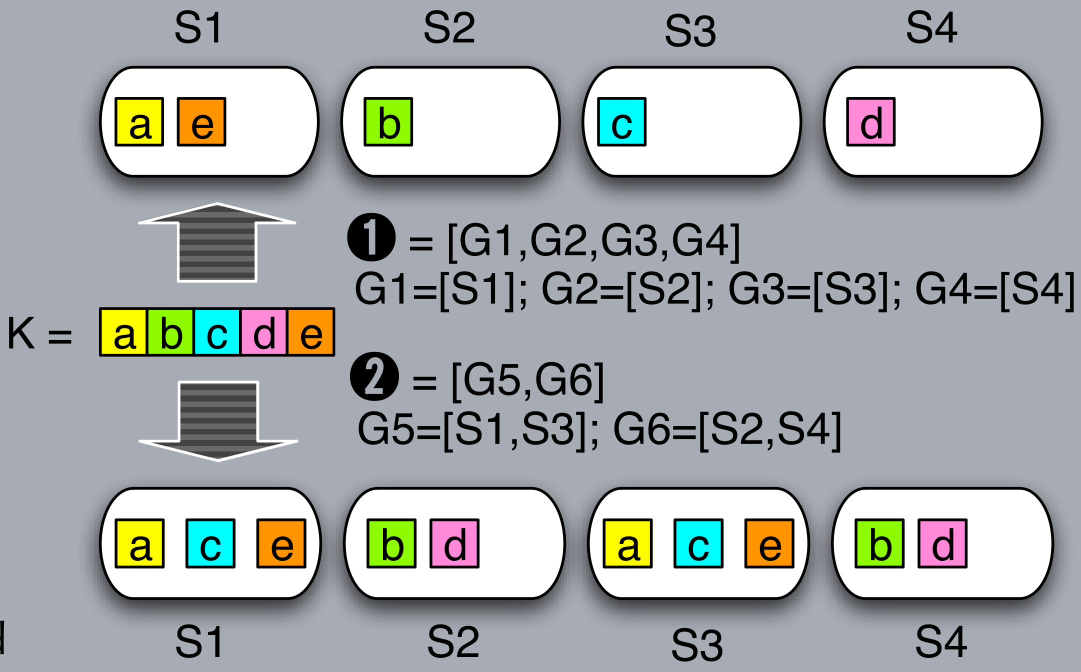
- objects are atomic
- collections distributed and replicated by means of a dynamic schema (as many as you want):
  - spread group
  - replica group
- schemas can be added & changed at runtime



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

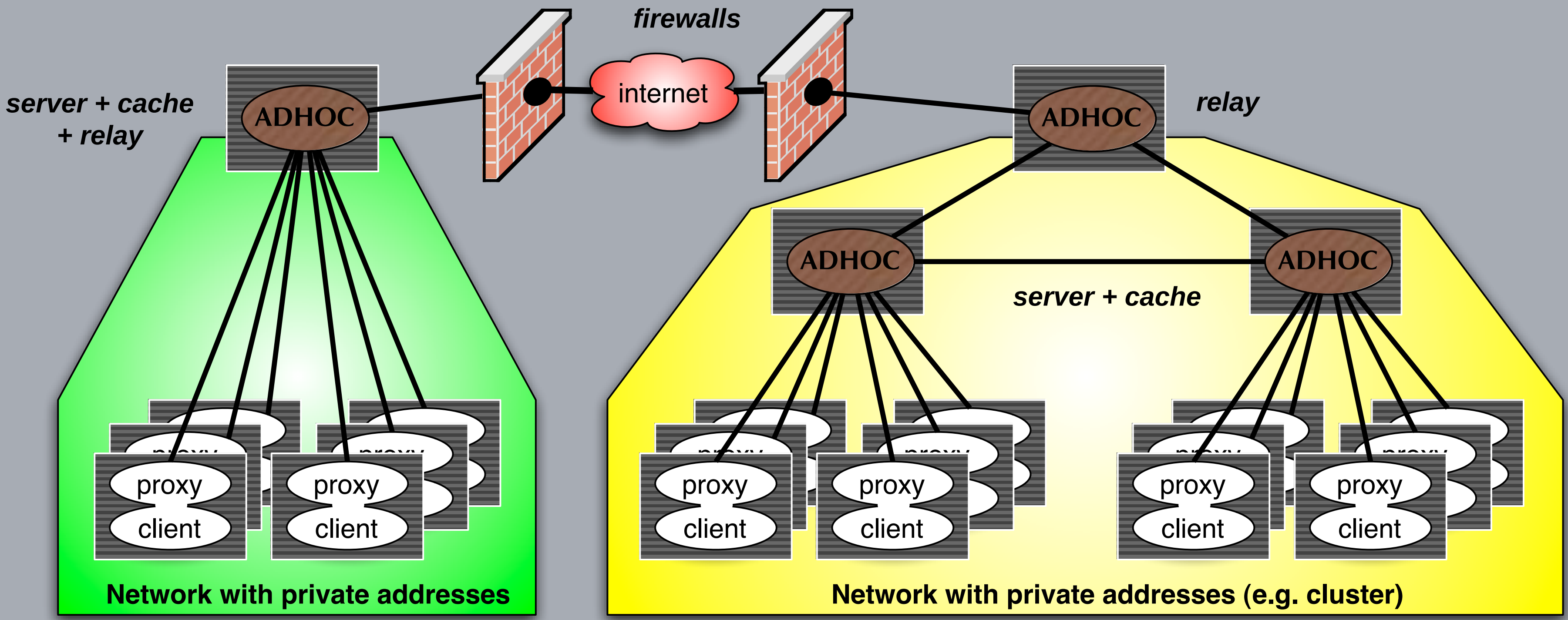


ADHOC is Grid-aware

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

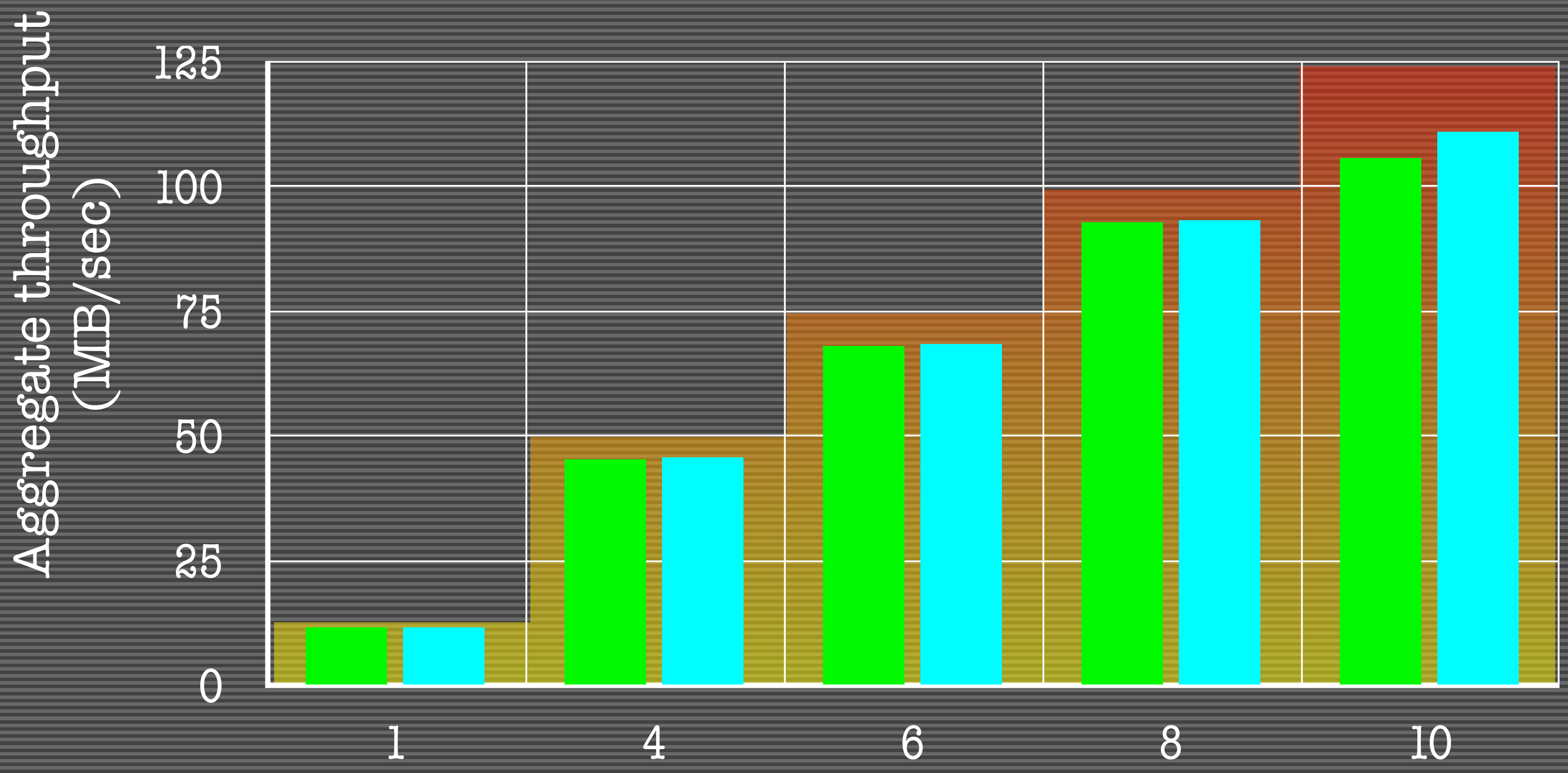
Through the firewalls



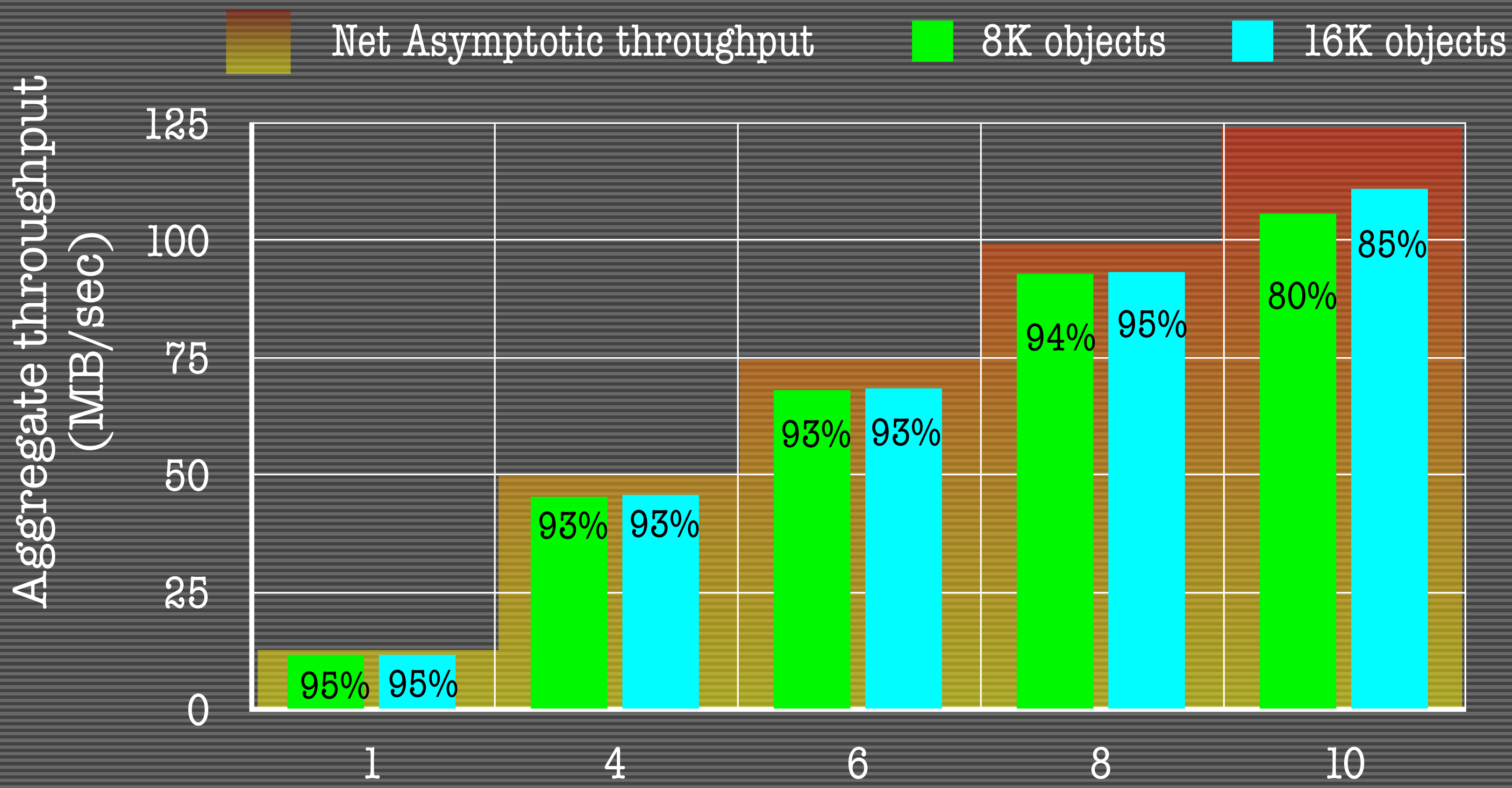
ADHOC performance figures (1PE)

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

Sustained aggregate throughput

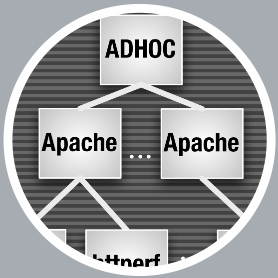


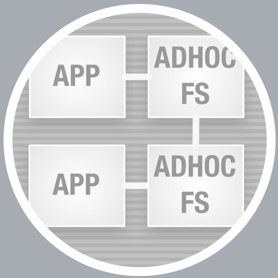
Sustained aggregate throughput



Summarizing ...

- 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, ...

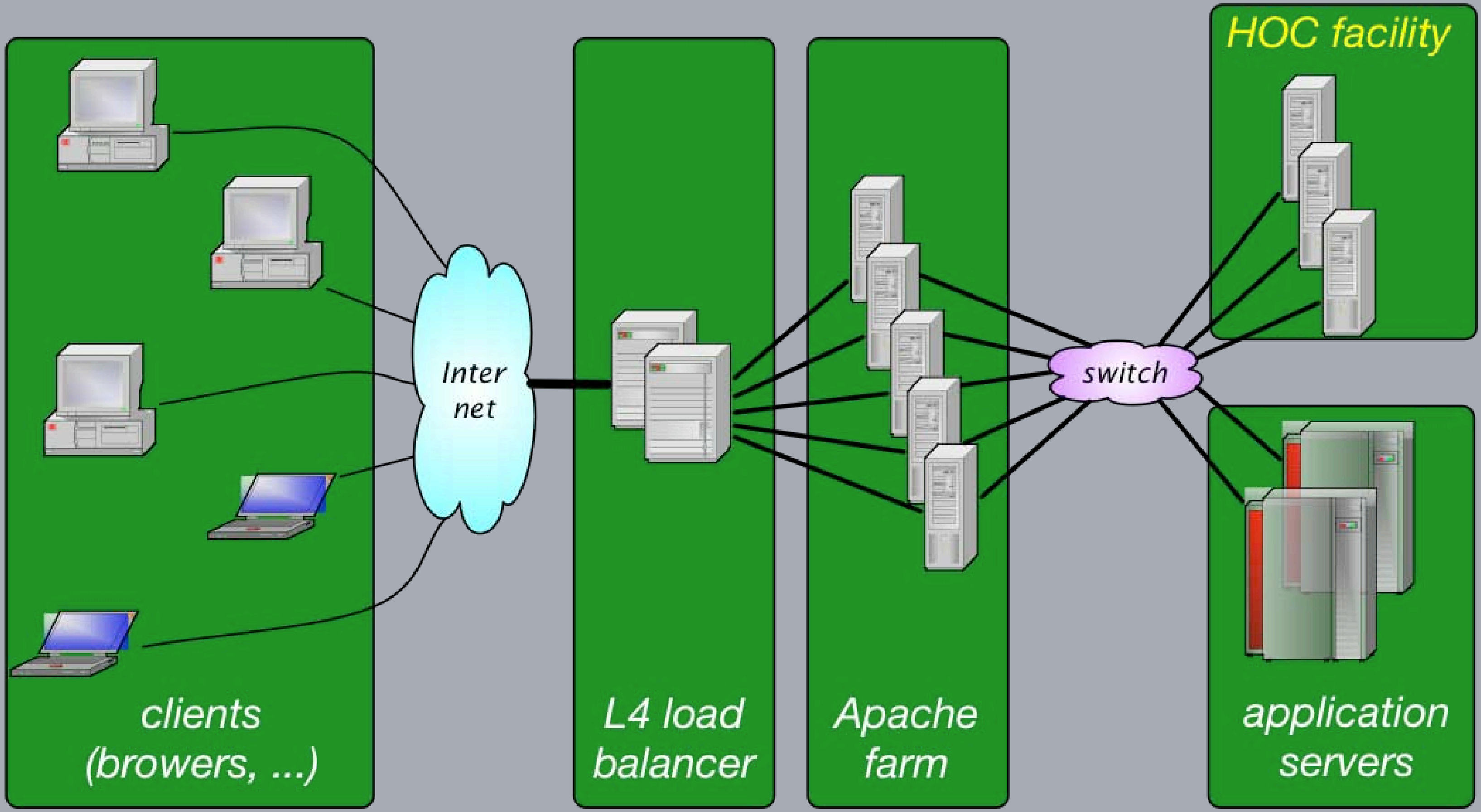
How accelerate a web server/service

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

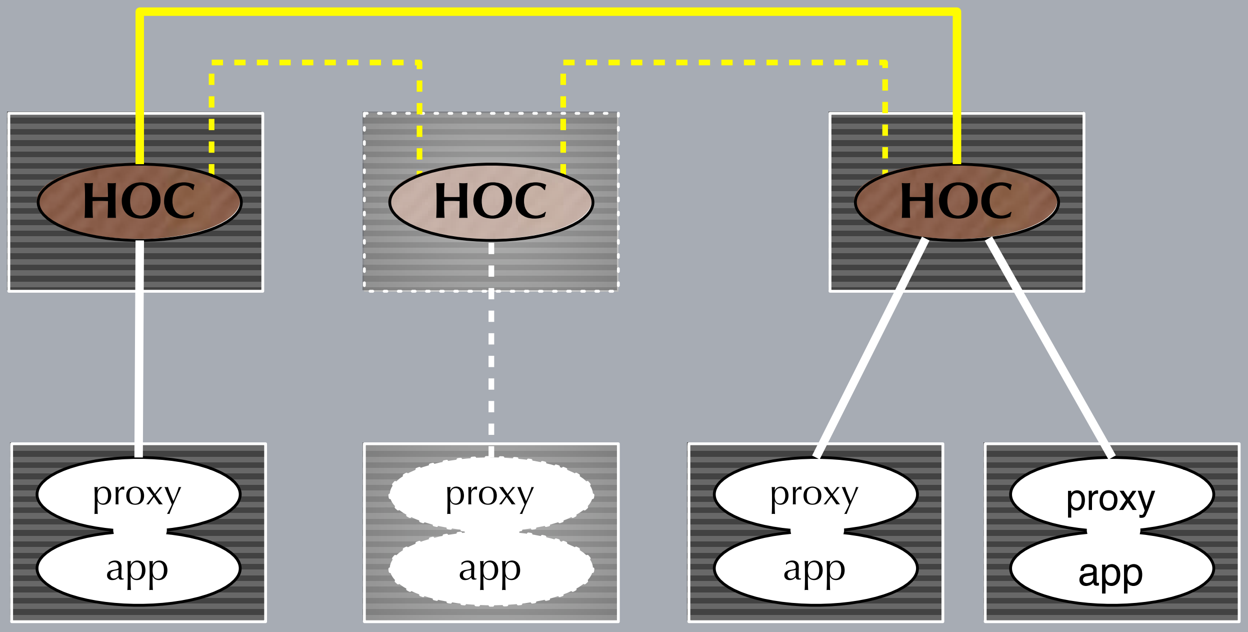
We would like to improve web server performance without changing web server core, thus relying on correctness, people expertise, ...

... thus we add an HOC-based distributed cache behind the server (or the server farm)

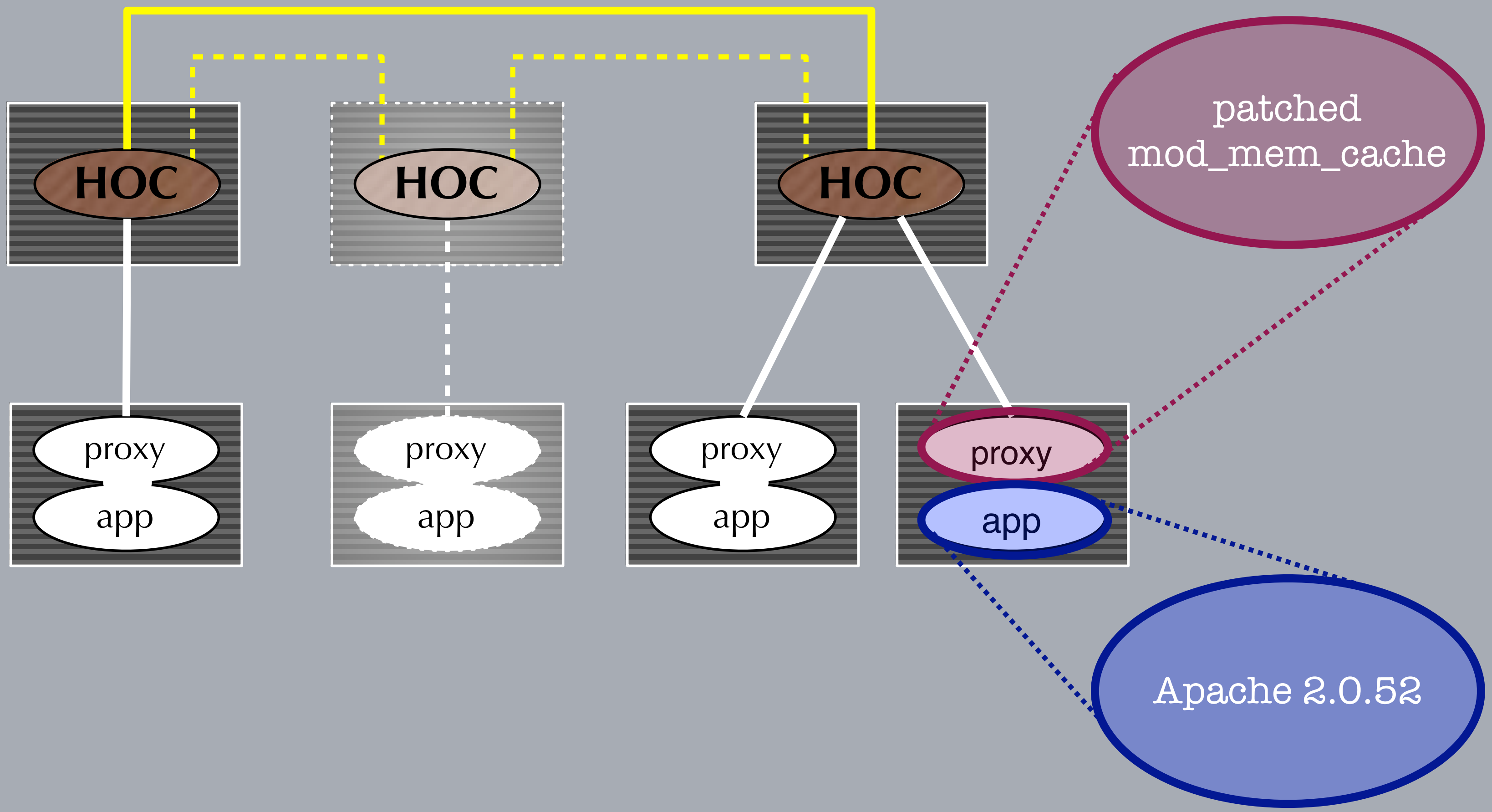
The big picture



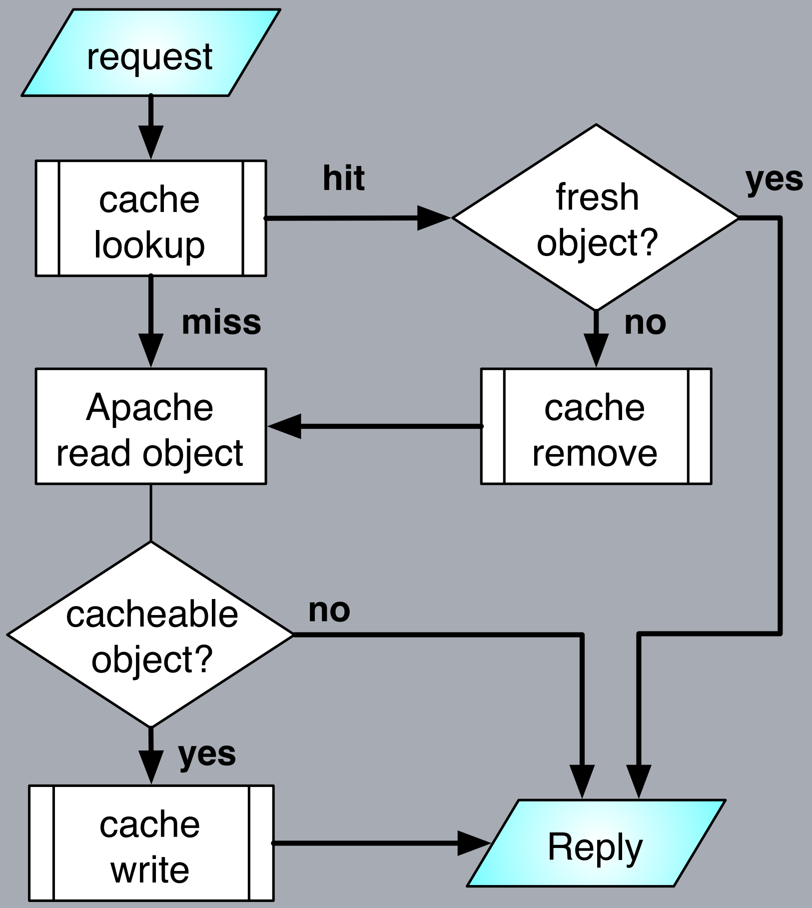
The Apache plug-in for HOC



The Apache plug-in for HOC

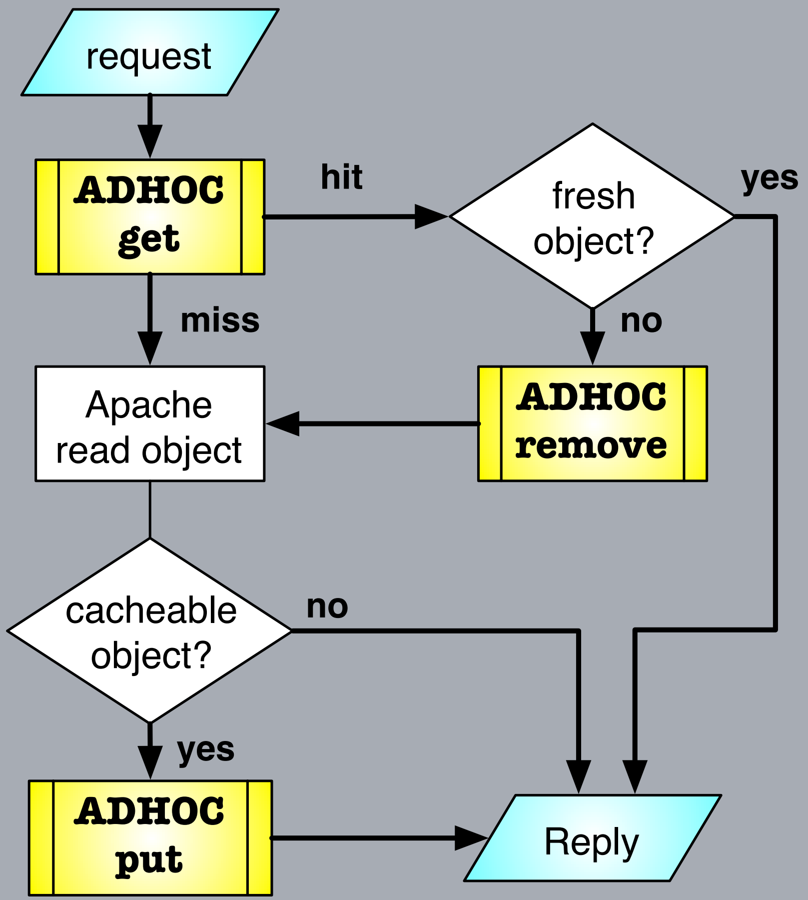


The Apache plug-in for ADHOC



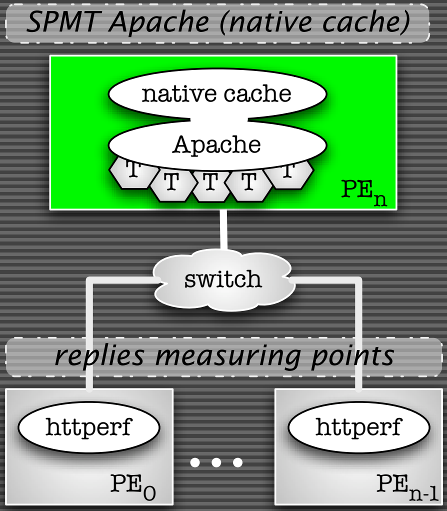
High-level functional behavior of the Apache 2.0.52 native cache module (mod_mem_cache)

The Apache plug-in for ADHOC



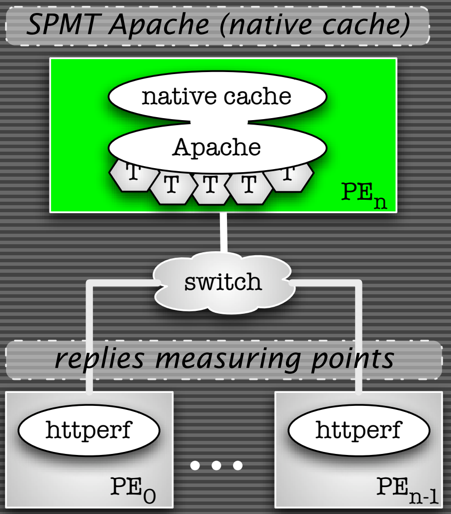
High-level functional behavior
of the protocol for ADHOC+Apache
architecture
(a simple patch to mod_mem_cache)

Comparing reply rate (1-ADHOC/1-Apache/k-httperf)



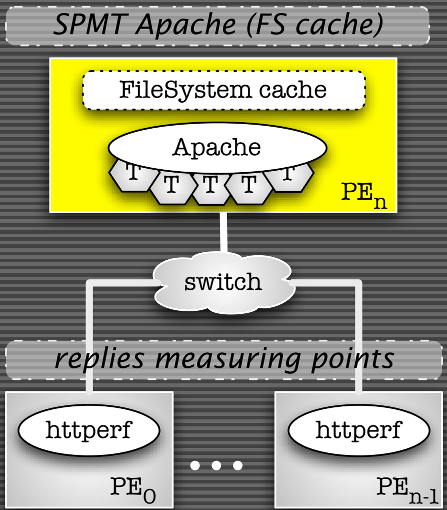
SingleProcessMultiThreaded Apache (900MB shared native cache)

Comparing reply rate (1-ADHOC/1-Apache/k-httperf)



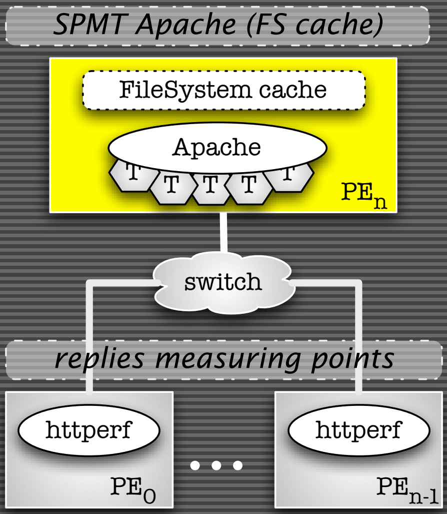
SingleProcessMultiThreaded Apache (900MB shared native cache)

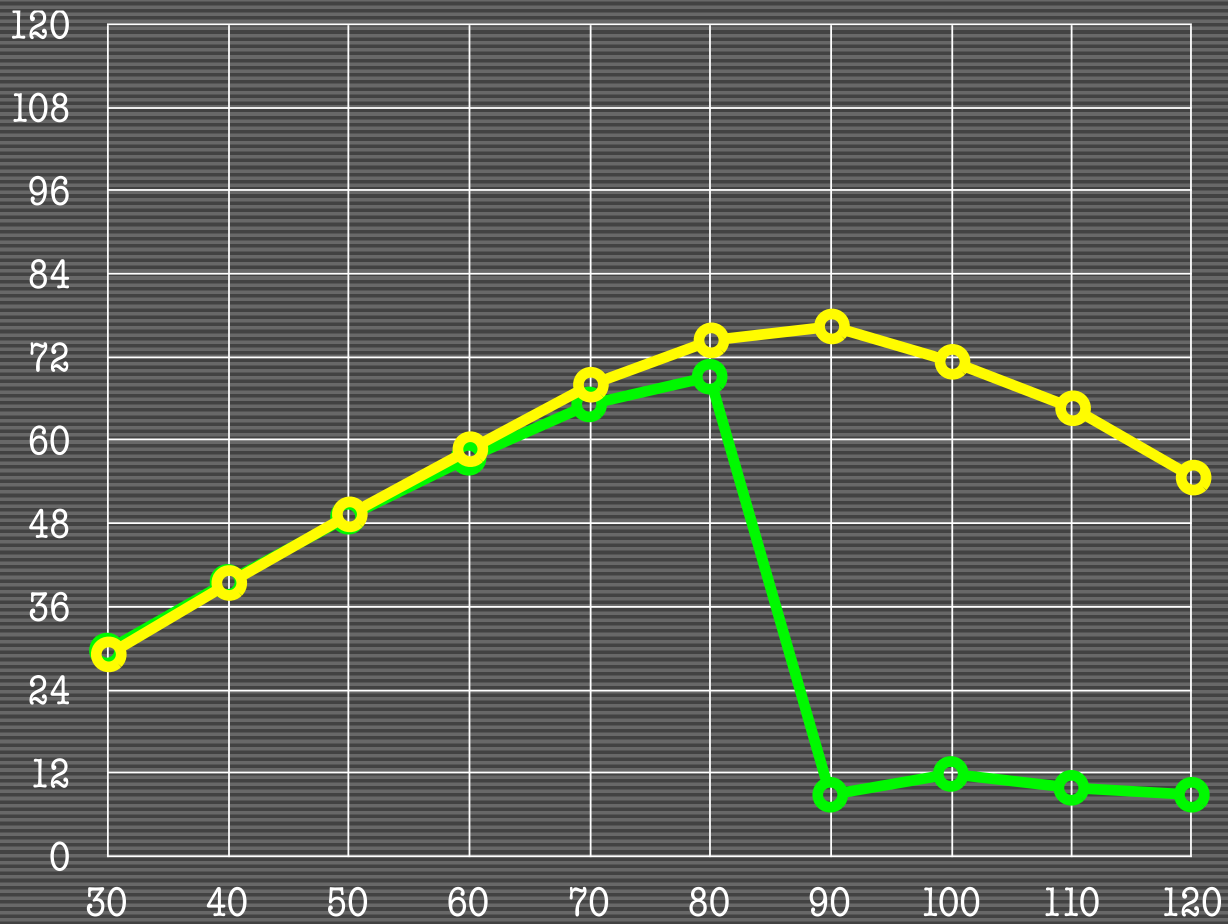
Comparing reply rate (1-ADHOC/1-Apache/k-httperf)



NoCache SPMT Apache (FileSystem buffer behaves as cache)

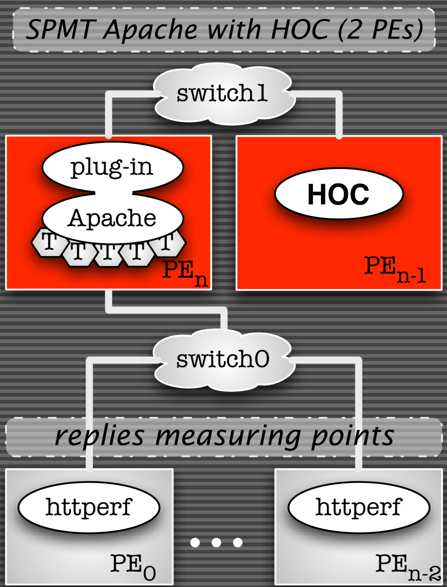
Comparing reply rate (1-ADHOC/1-Apache/k-httperf)

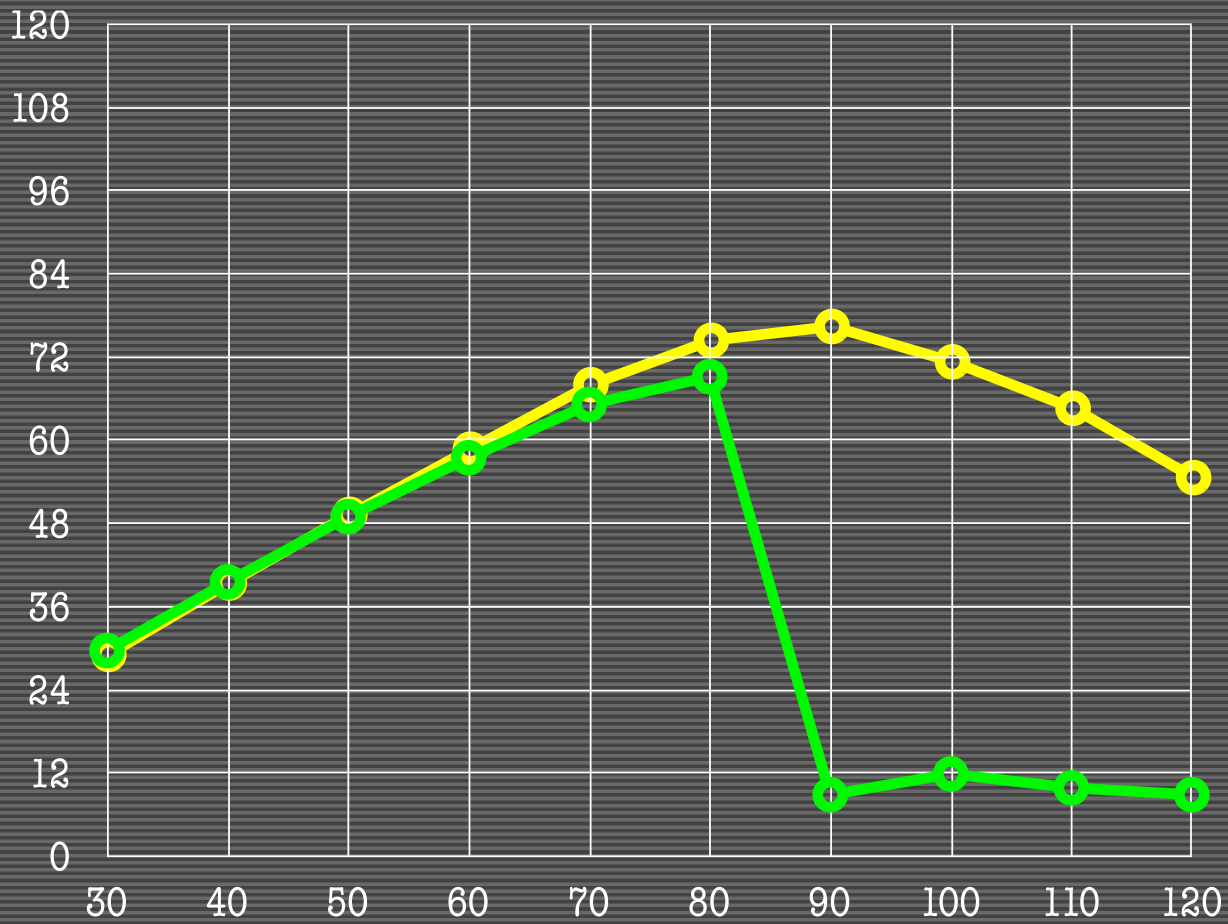




NoCache SPMT Apache (FileSystem buffer behaves as cache)

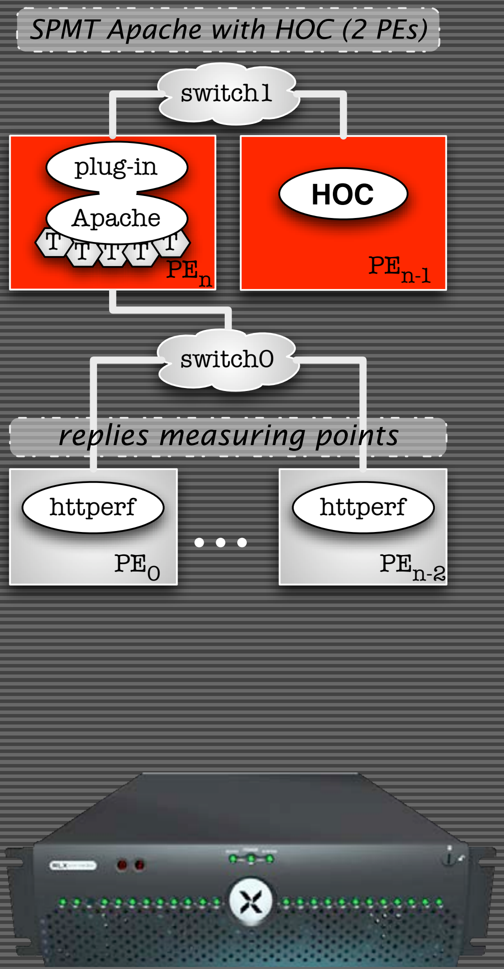
Comparing reply rate (1-ADHOC/1-Apache/k-httperf)

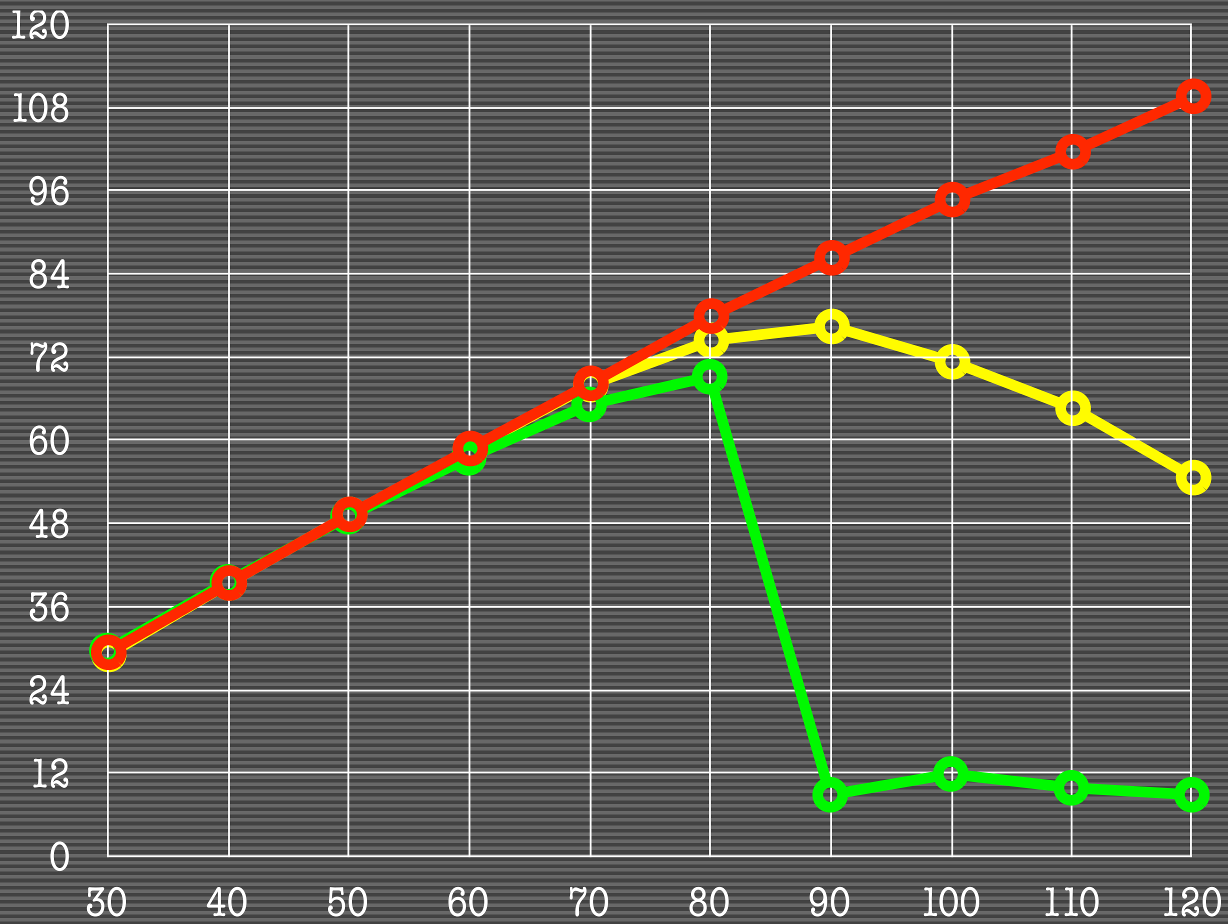




SPMT Apache with **900MB ADHOC on 2 boxes**

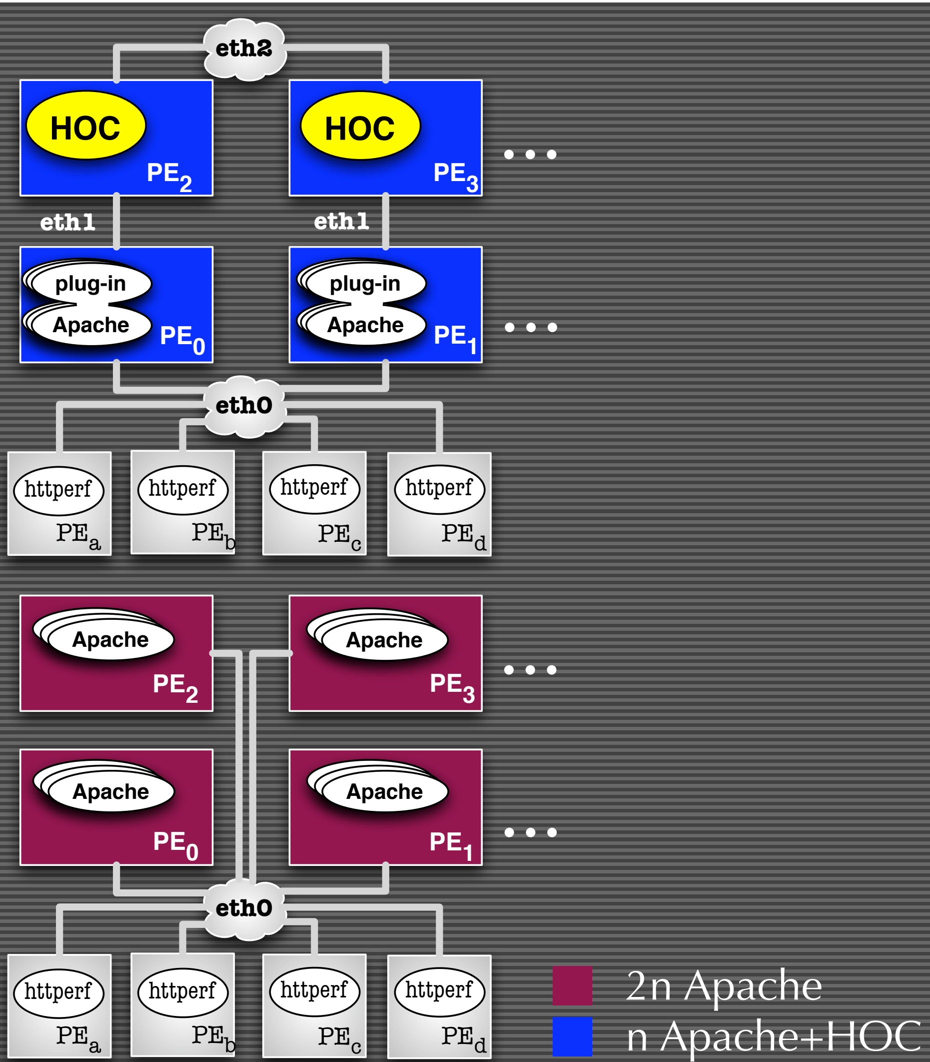
Comparing reply rate (1-ADHOC/1-Apache/k-httperf)



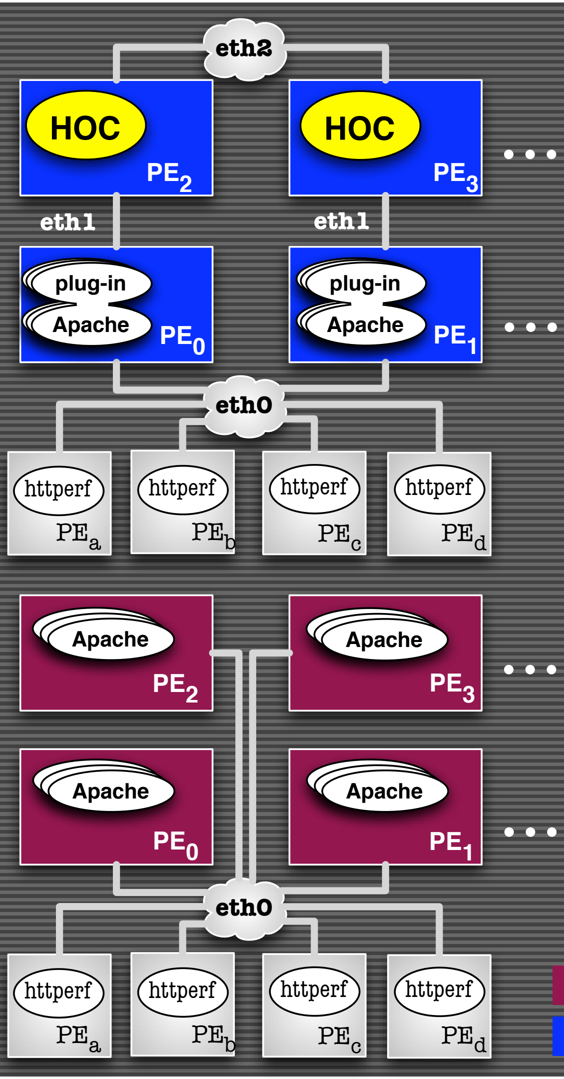


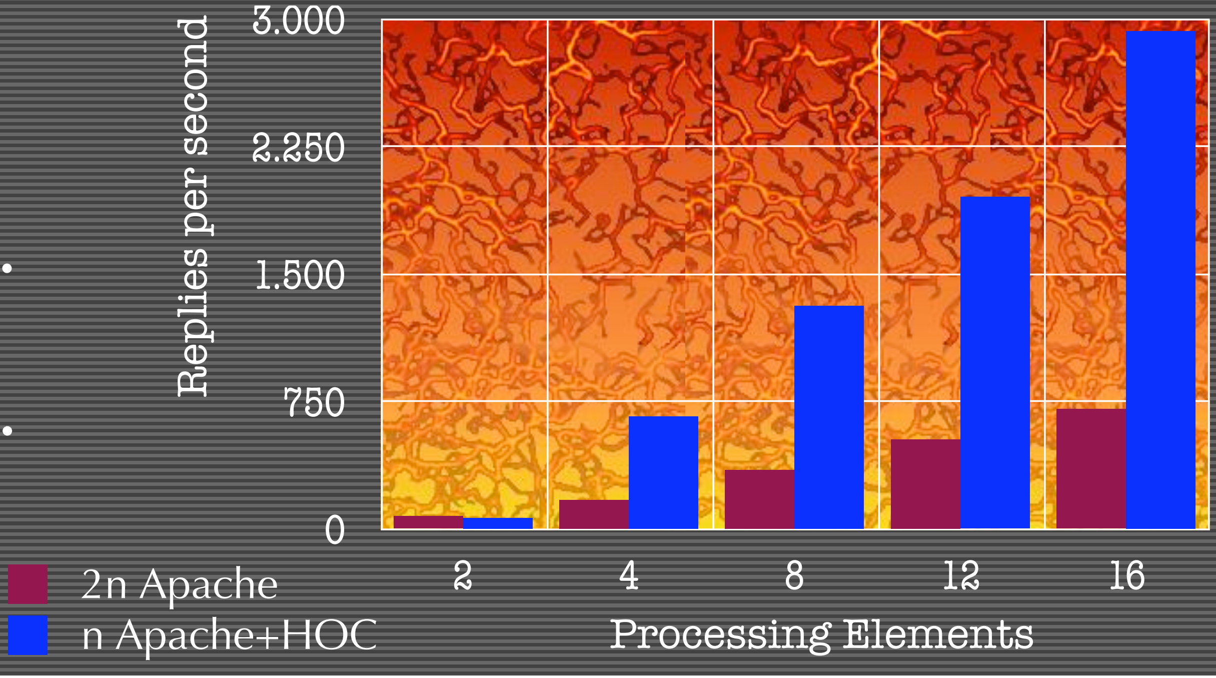
SPMT Apache with **900MB ADHOC on 2 boxes**

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

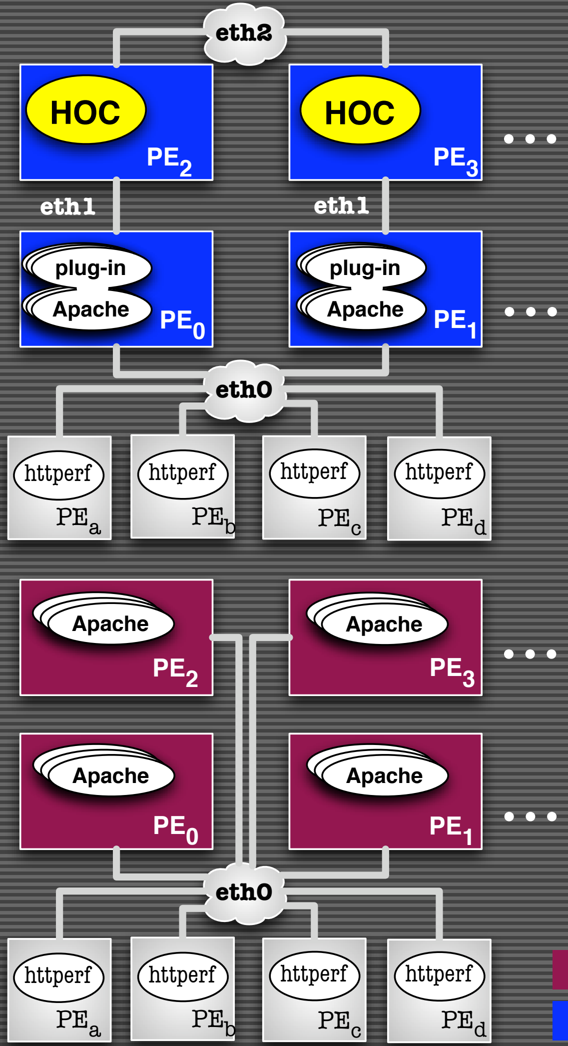


Apache ***2n***-farm *vs* Apache+ADHOC ***n***-farm

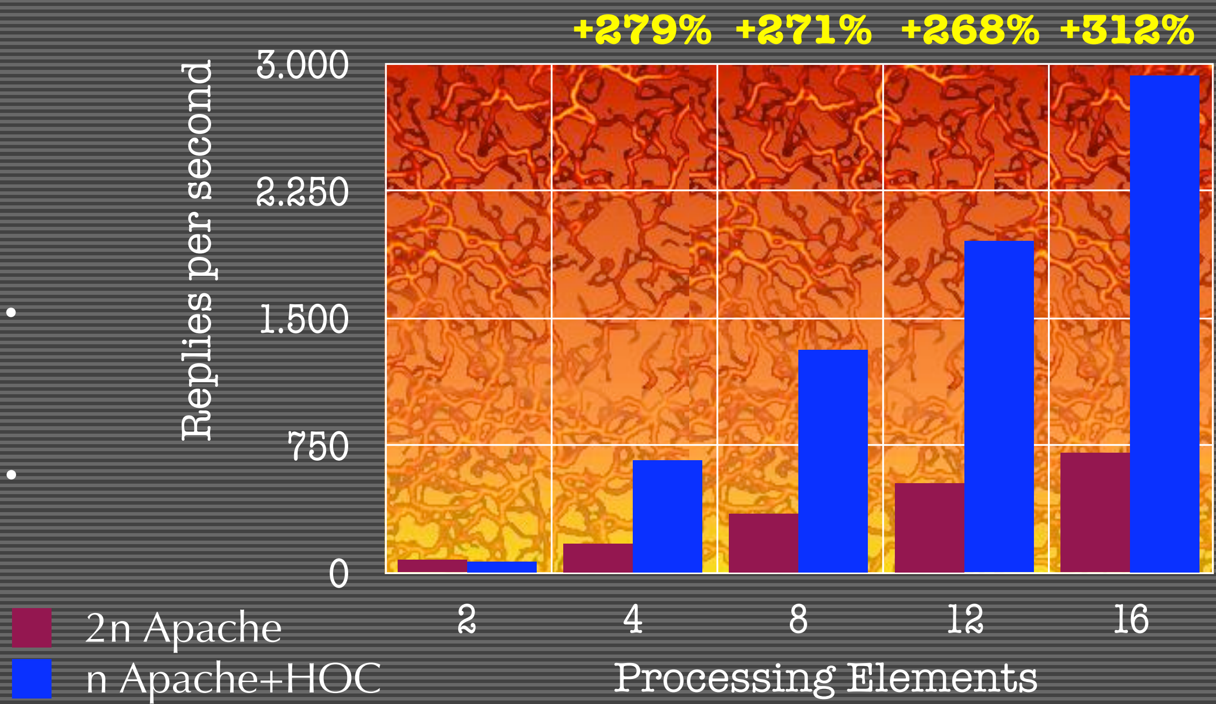


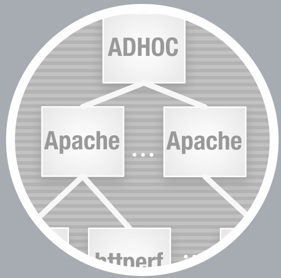


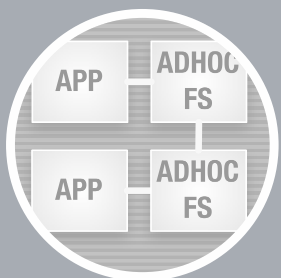
Apache ***2n***-farm *vs* Apache+ADHOC ***n***-farm



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



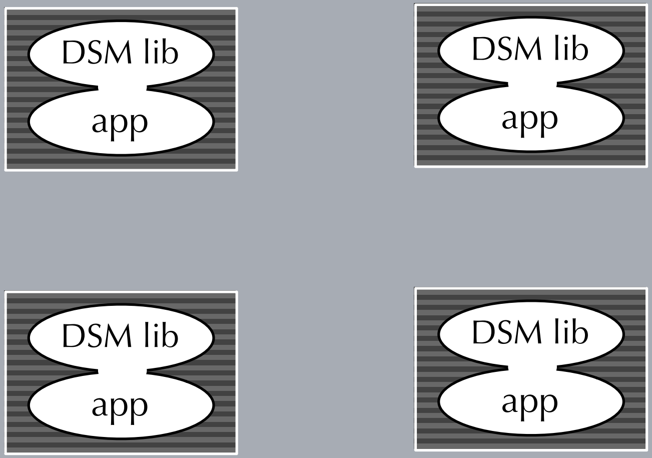




- ADHOC cache plugin for Apache
  - Big picture & features
  - Performances & scalability (both of them very good)
- ADHOC-based DSM
  - Big picture & features
  - Performances
- ASTFS (ADHOC-based FS)
  - Big picture & features
  - Performances

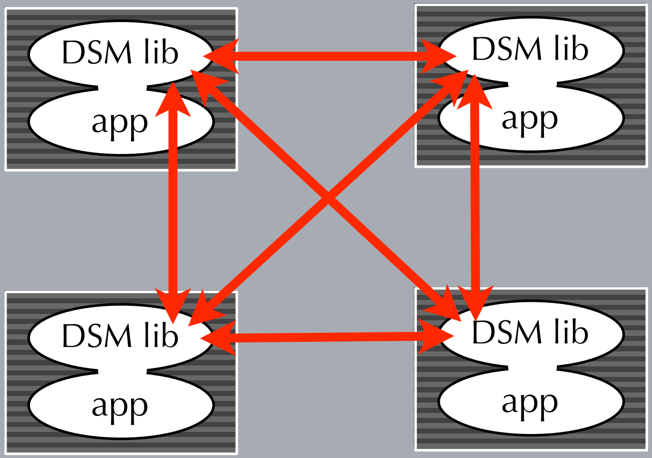
Standard DSM vs ADHOC-DSM

Standard DSM



Standard DSM vs ADHOC-DSM

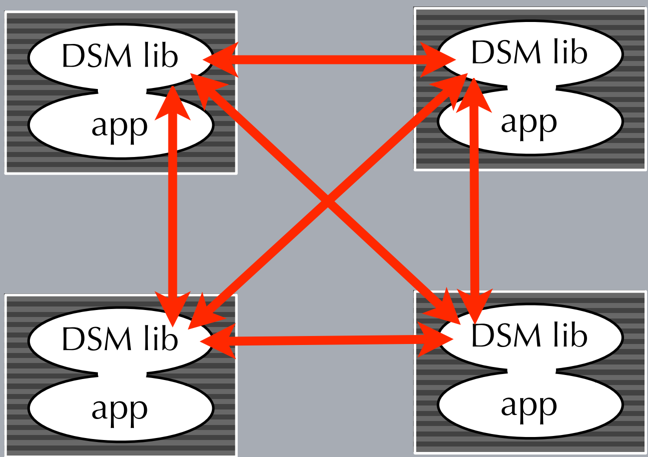
Standard DSM



- 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

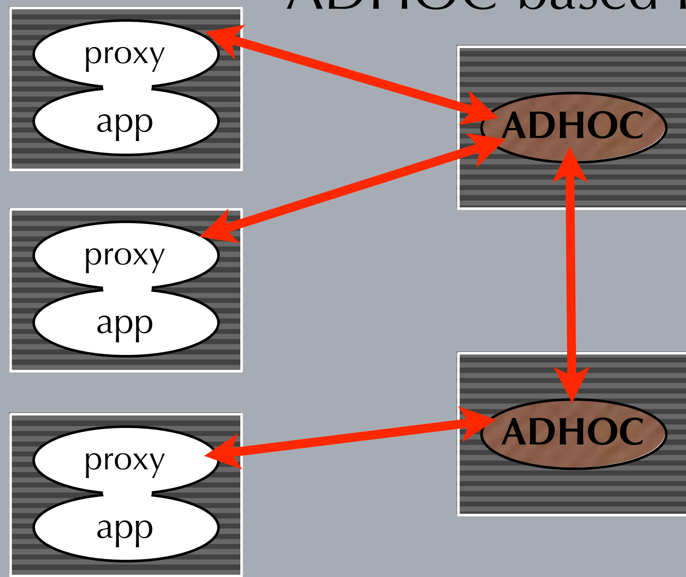
Standard DSM vs ADHOC-DSM

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

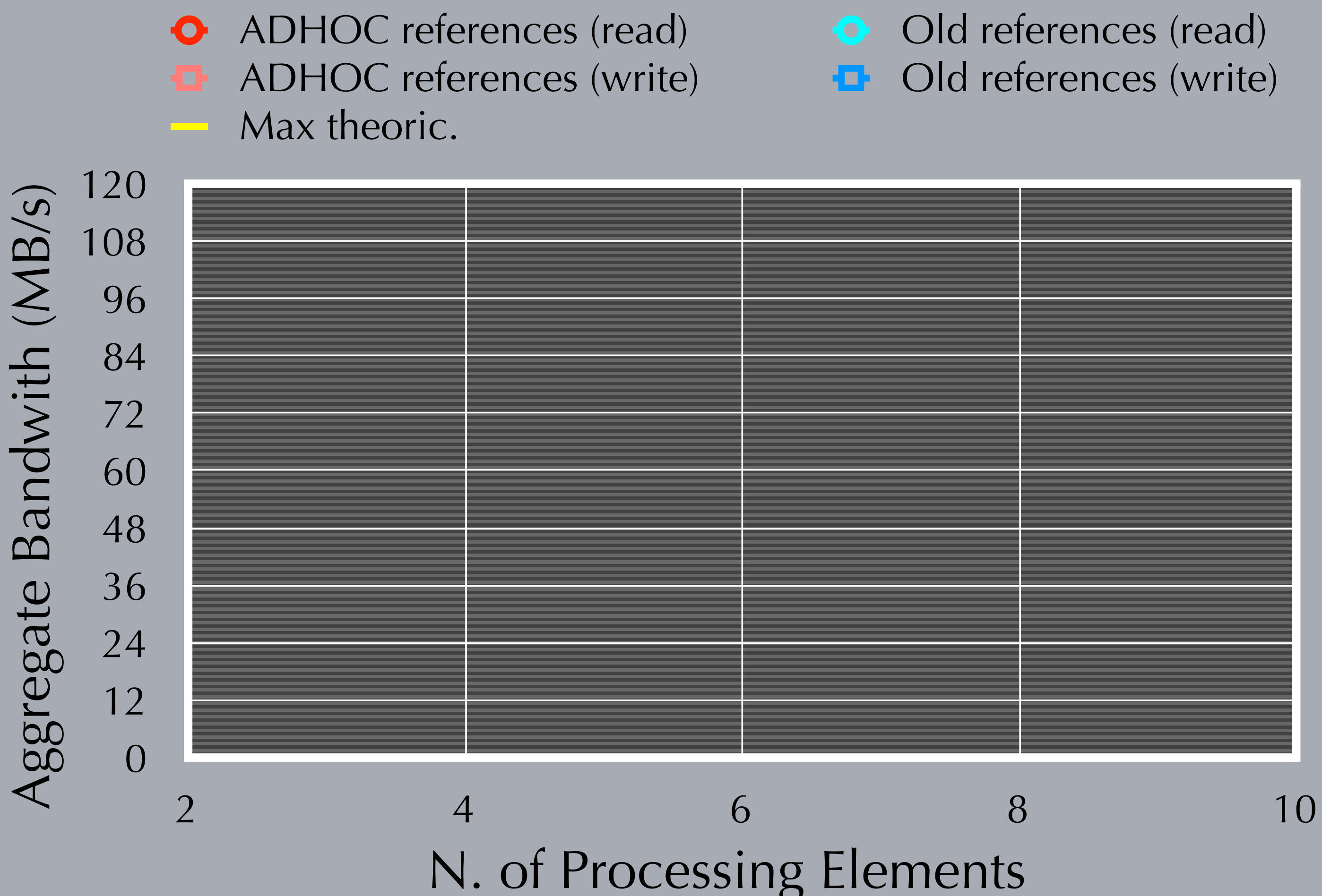
ADHOC references (read)
ADHOC references (write)
Max theoric.
Old references (read)
Old references (write)

Aggregate Bandwith (MB/s)

N. of Processing Elements

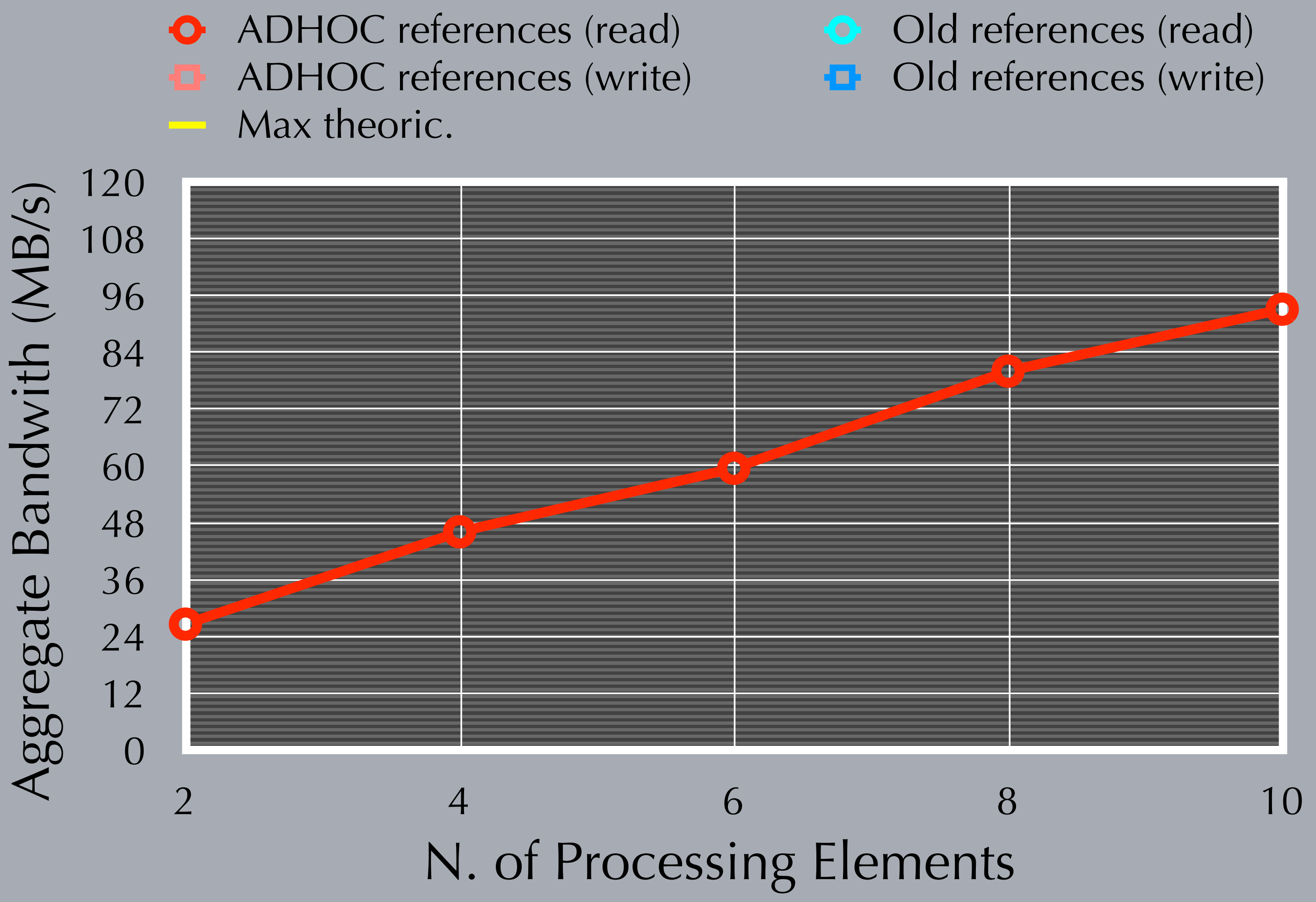
ADHOC-DSM scalability

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



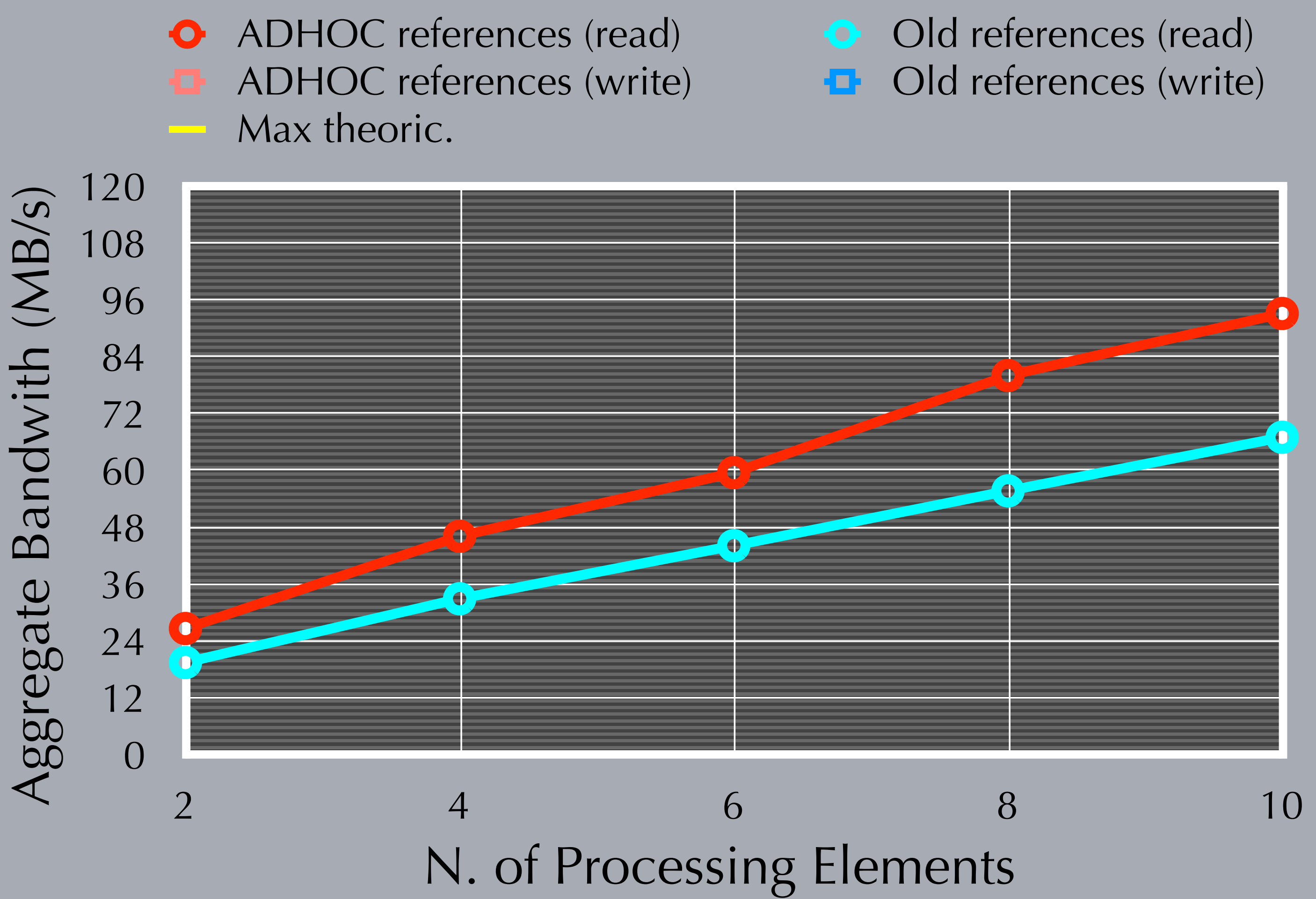
ADHOC-DSM scalability

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



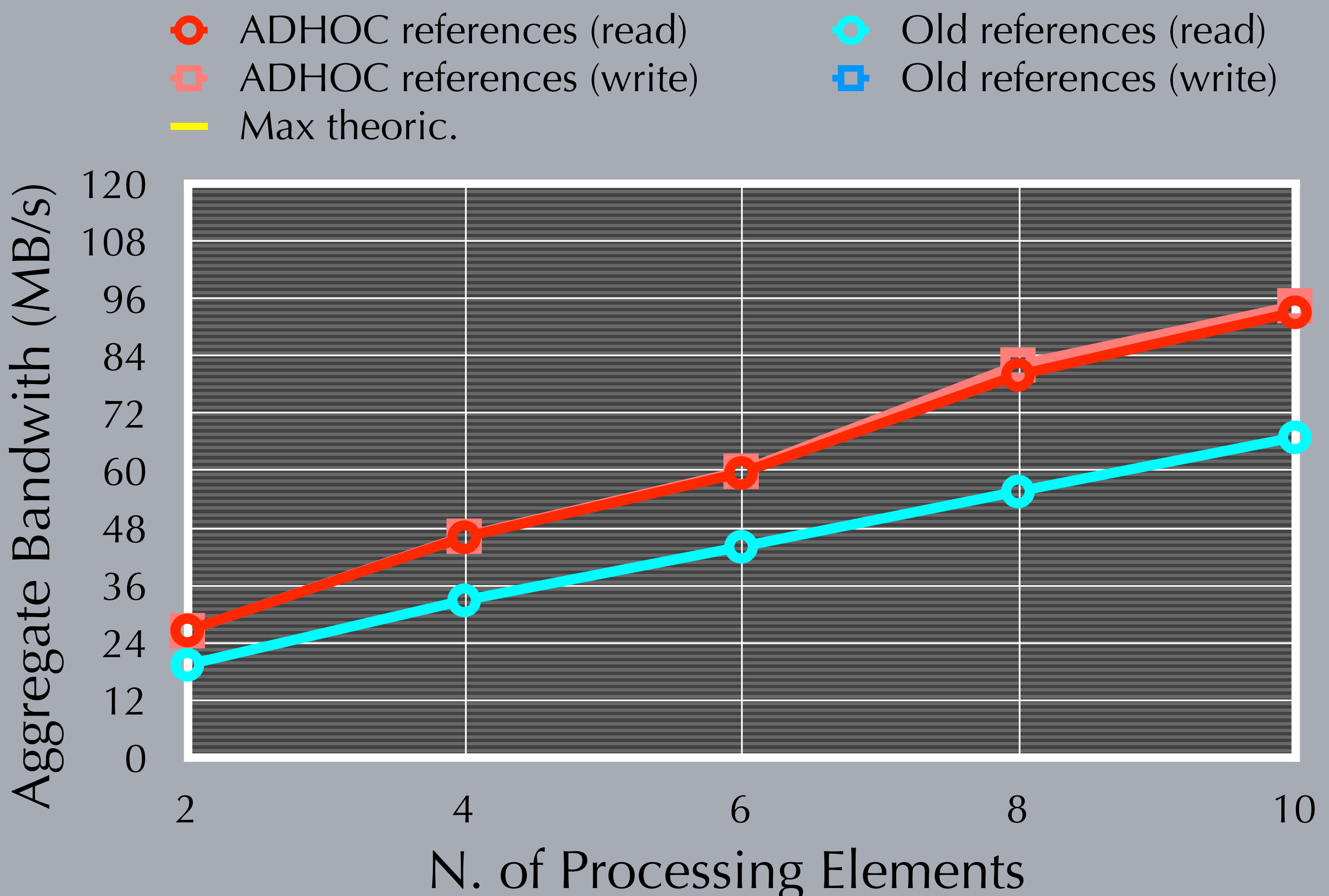
ADHOC-DSM scalability

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



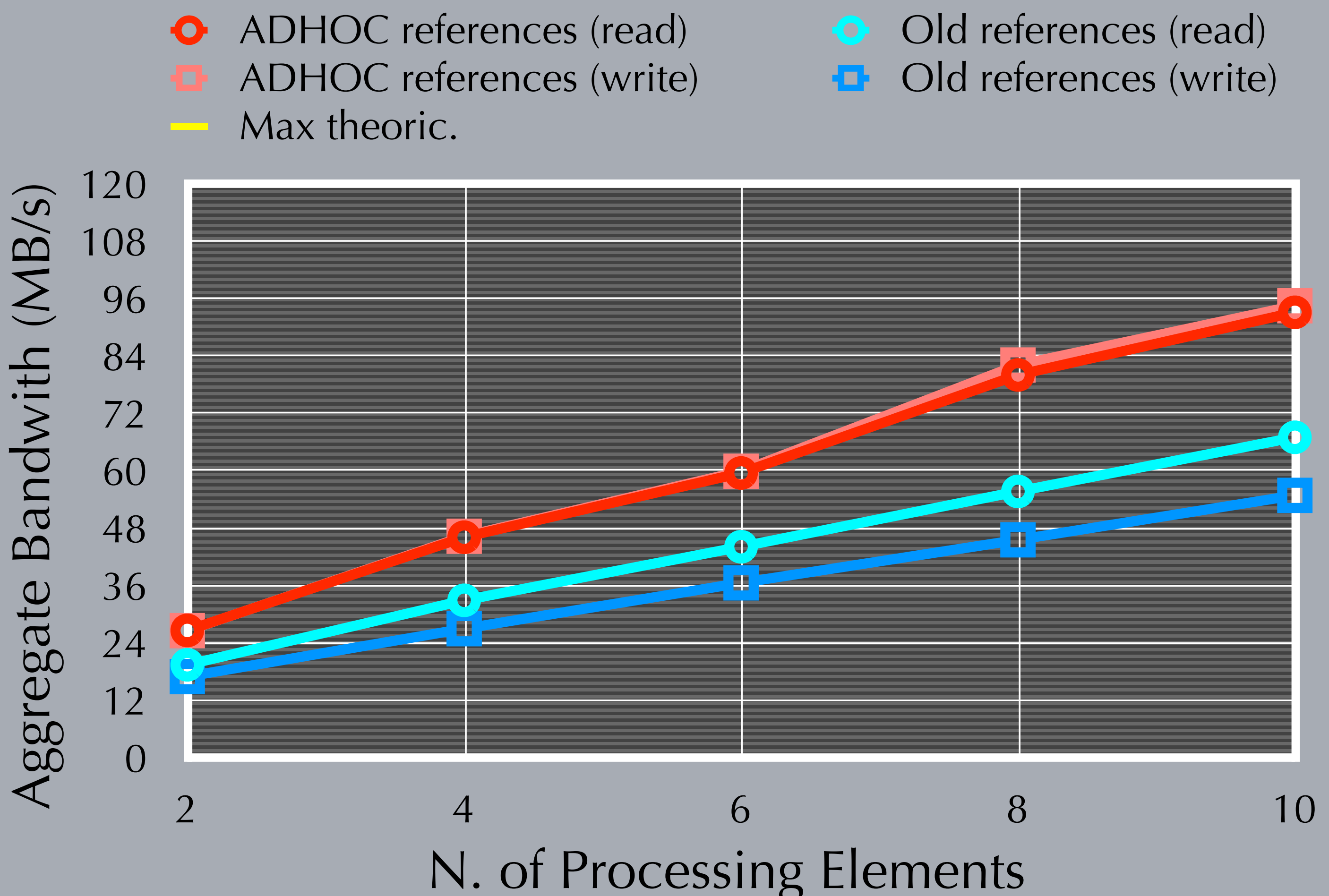
ADHOC-DSM scalability

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



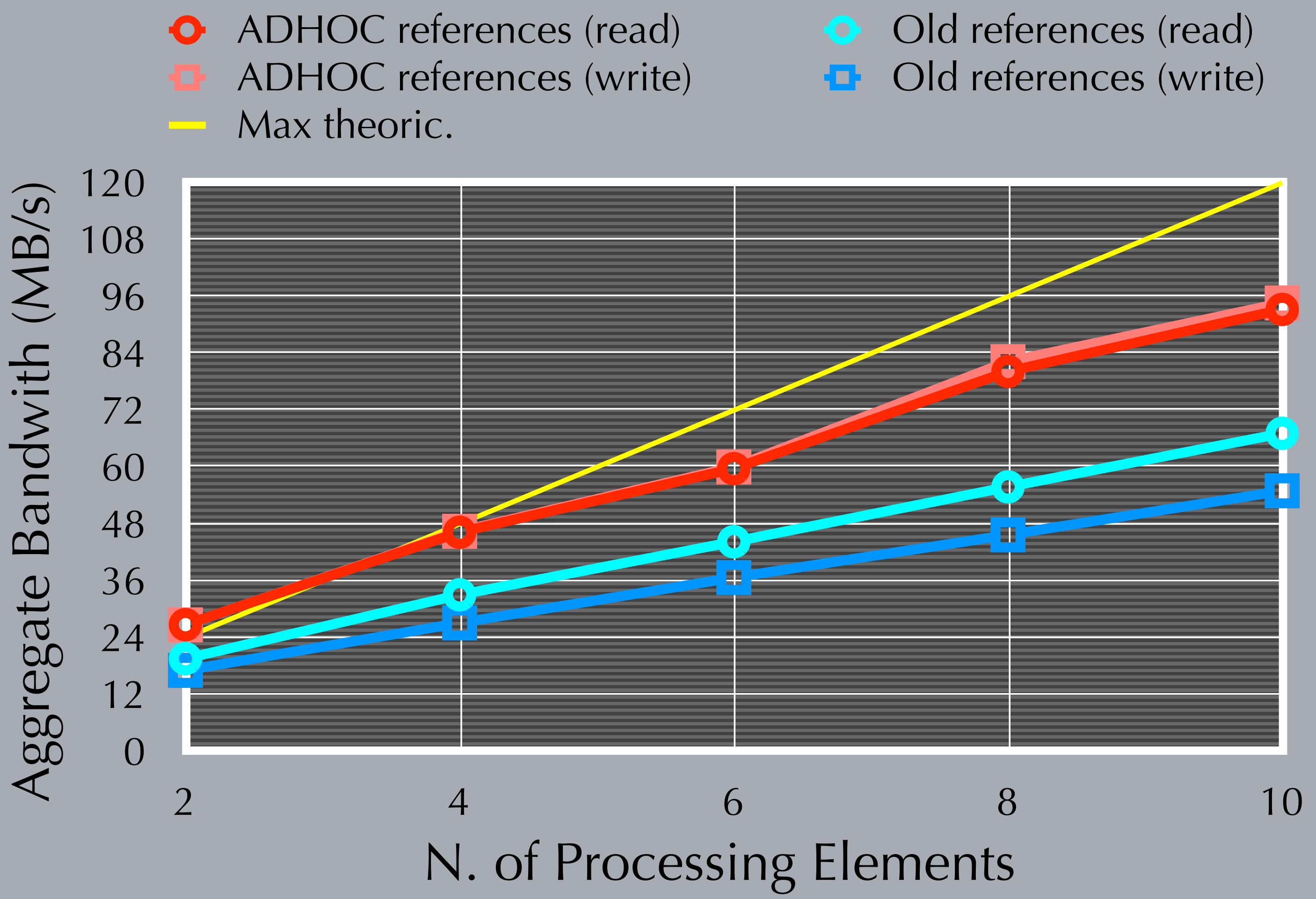
ADHOC-DSM scalability

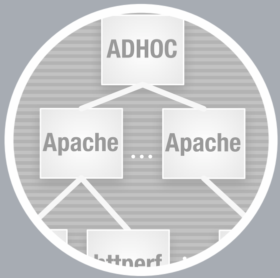
Intel Linux boxes, fast Eth connected, equal number of connections

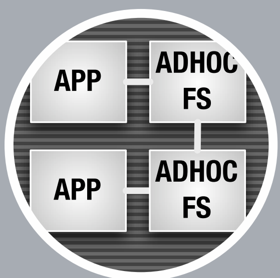


ADHOC-DSM scalability

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

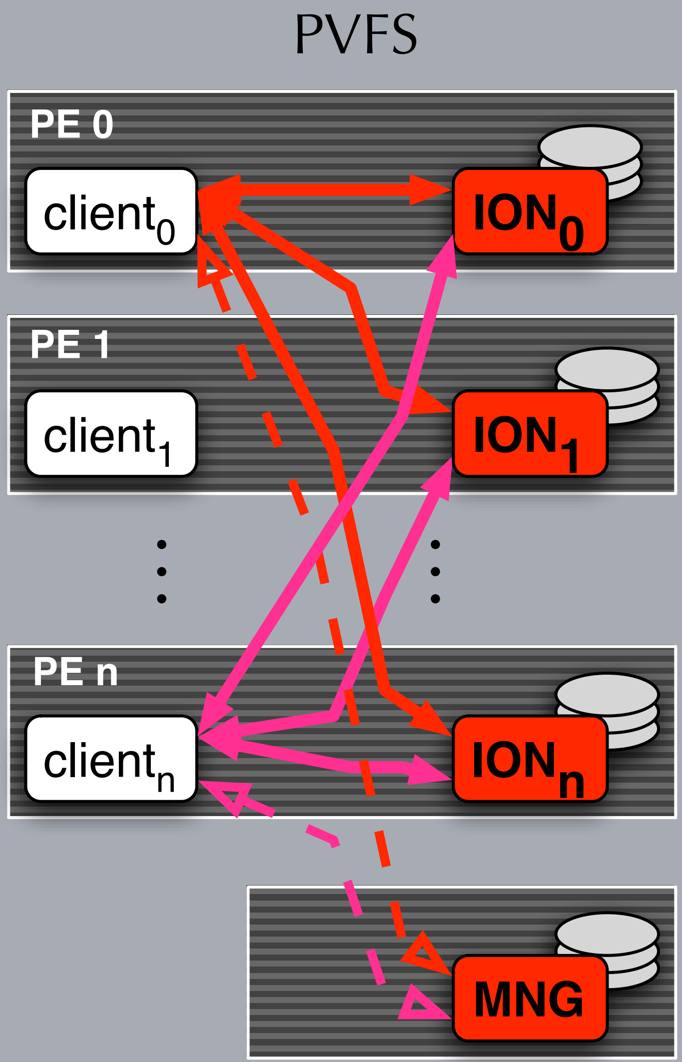






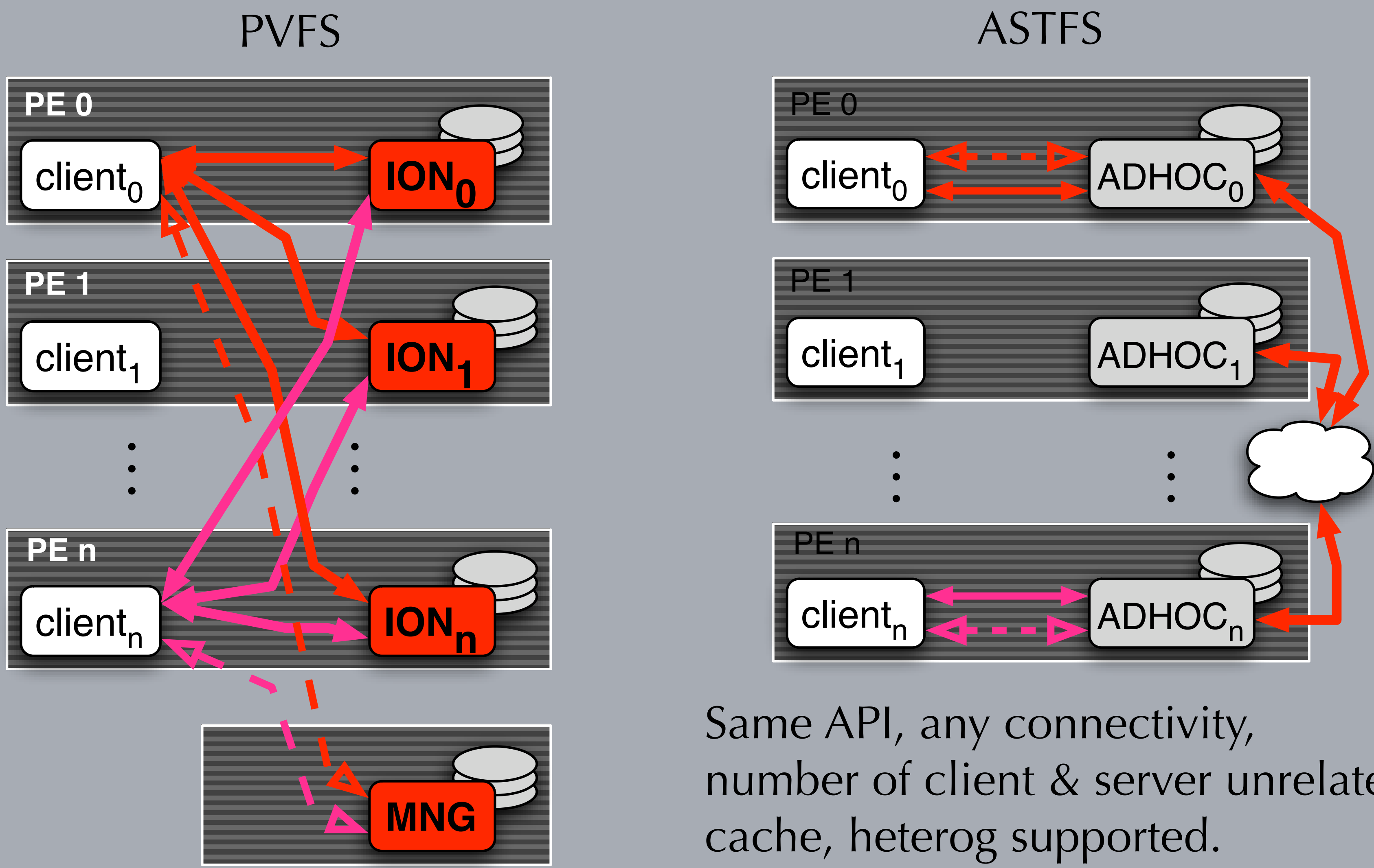
- ADHOC cache plugin for Apache
  - Big picture & features
  - Performances & scalability (both of them very good)
- ADHOC-based DSM
  - Big picture & features
  - Performances
- ASTFS (ADHOC-based FS)
  - Big picture & features
  - Performances

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 vs astFS



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

astFS scalability

Intel Linux boxes, fast Eth connected

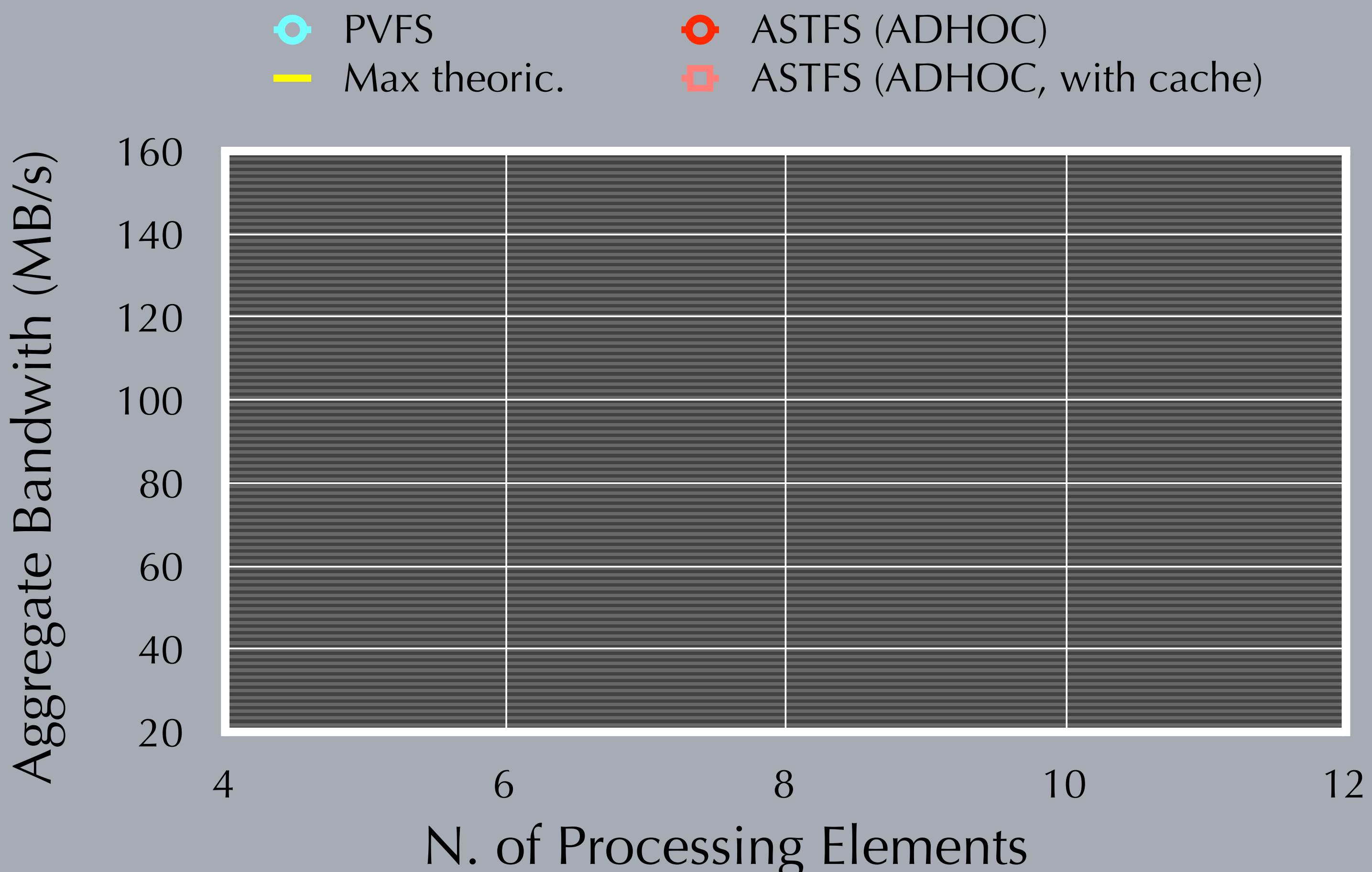
PVFS
Max theoric.
ASTFS (ADHOC)
ASTFS (ADHOC, with cache)

Aggregate Bandwith (MB/s)

N. of Processing Elements

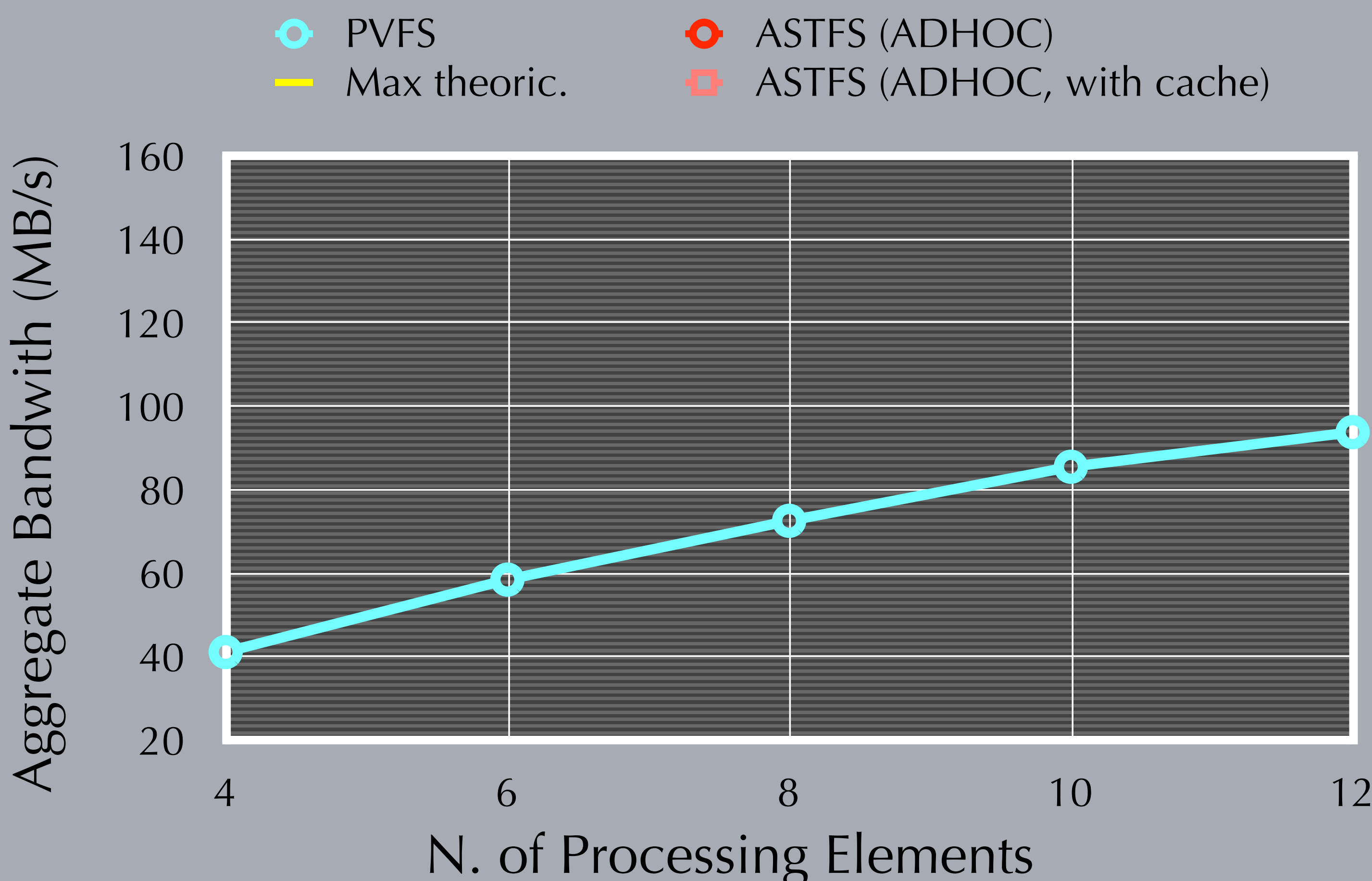
astFS scalability

Intel Linux boxes, fast Eth connected



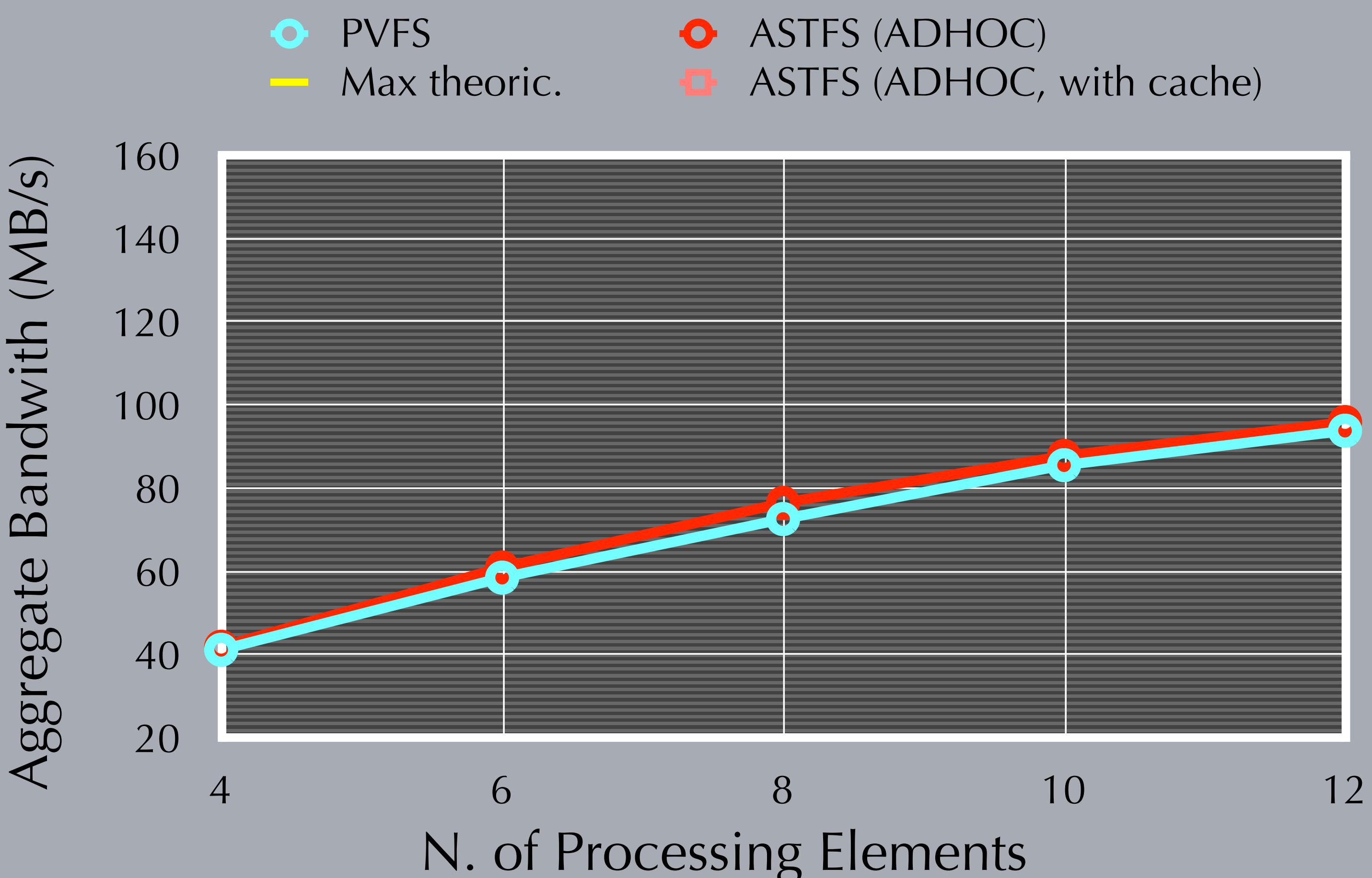
astFS scalability

Intel Linux boxes, fast Eth connected



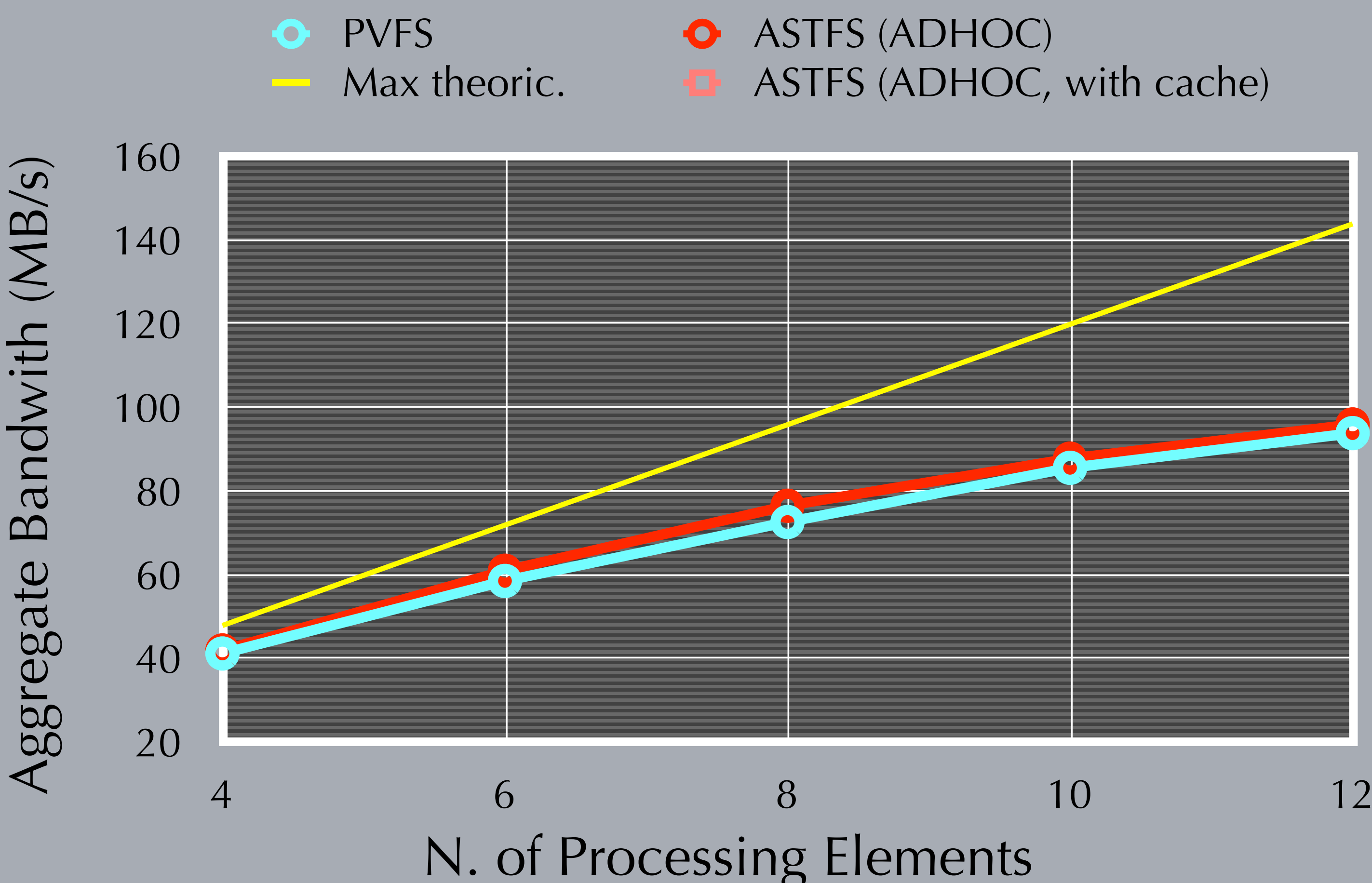
astFS scalability

Intel Linux boxes, fast Eth connected



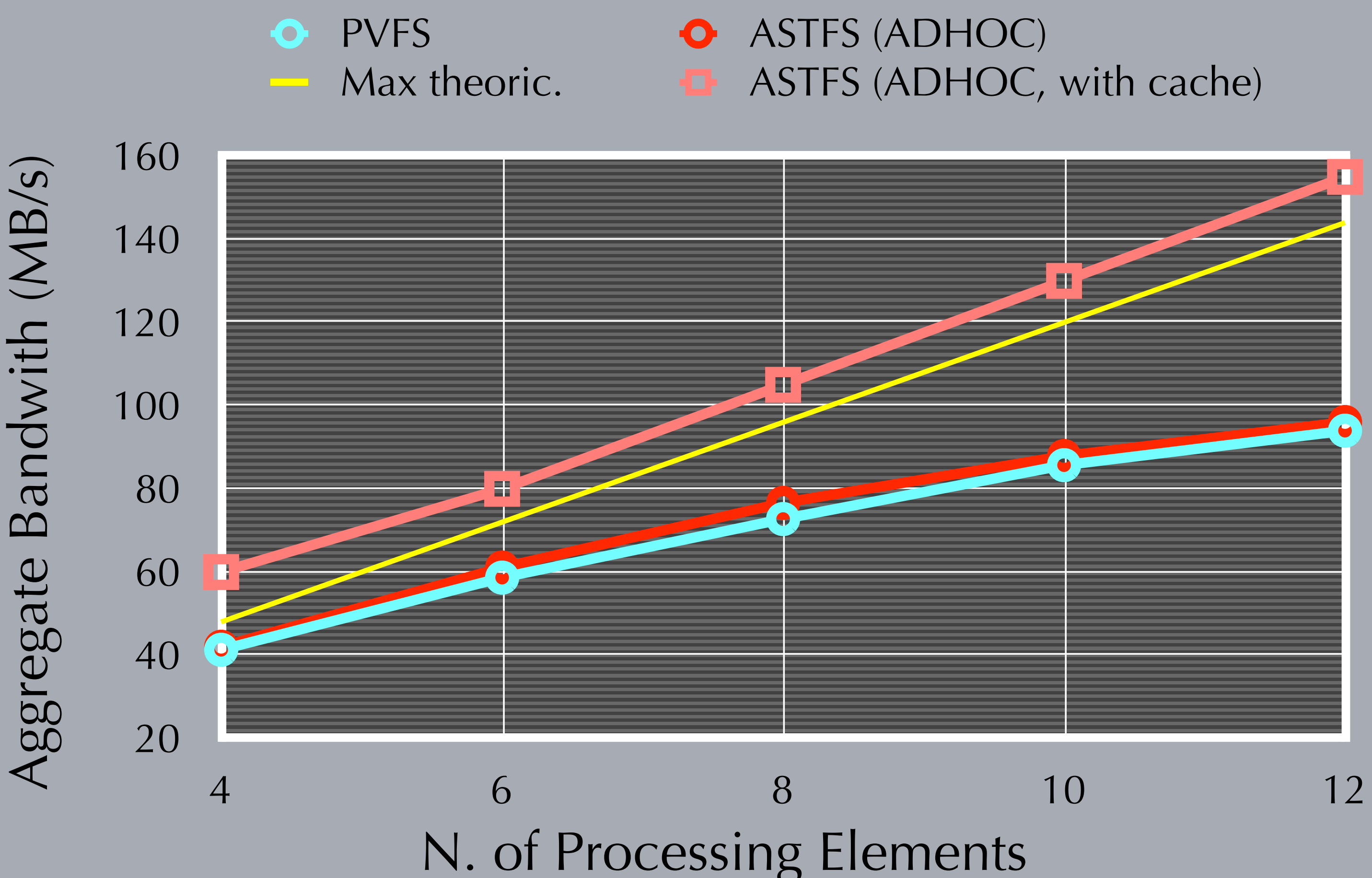
astFS scalability

Intel Linux boxes, fast Eth connected



astFS scalability

Intel Linux boxes, fast Eth connected



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 ;-)

Thank you :-)

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