Euro-Par 2004 - Pisa - Italy

Accelerating Apache farms through ad-HOC distributed scalable object repository

Marco Aldinucci, ISTI-CNR, Pisa, Italy
Massimo Torquati, CS dept. Uni. Pisa, Italy

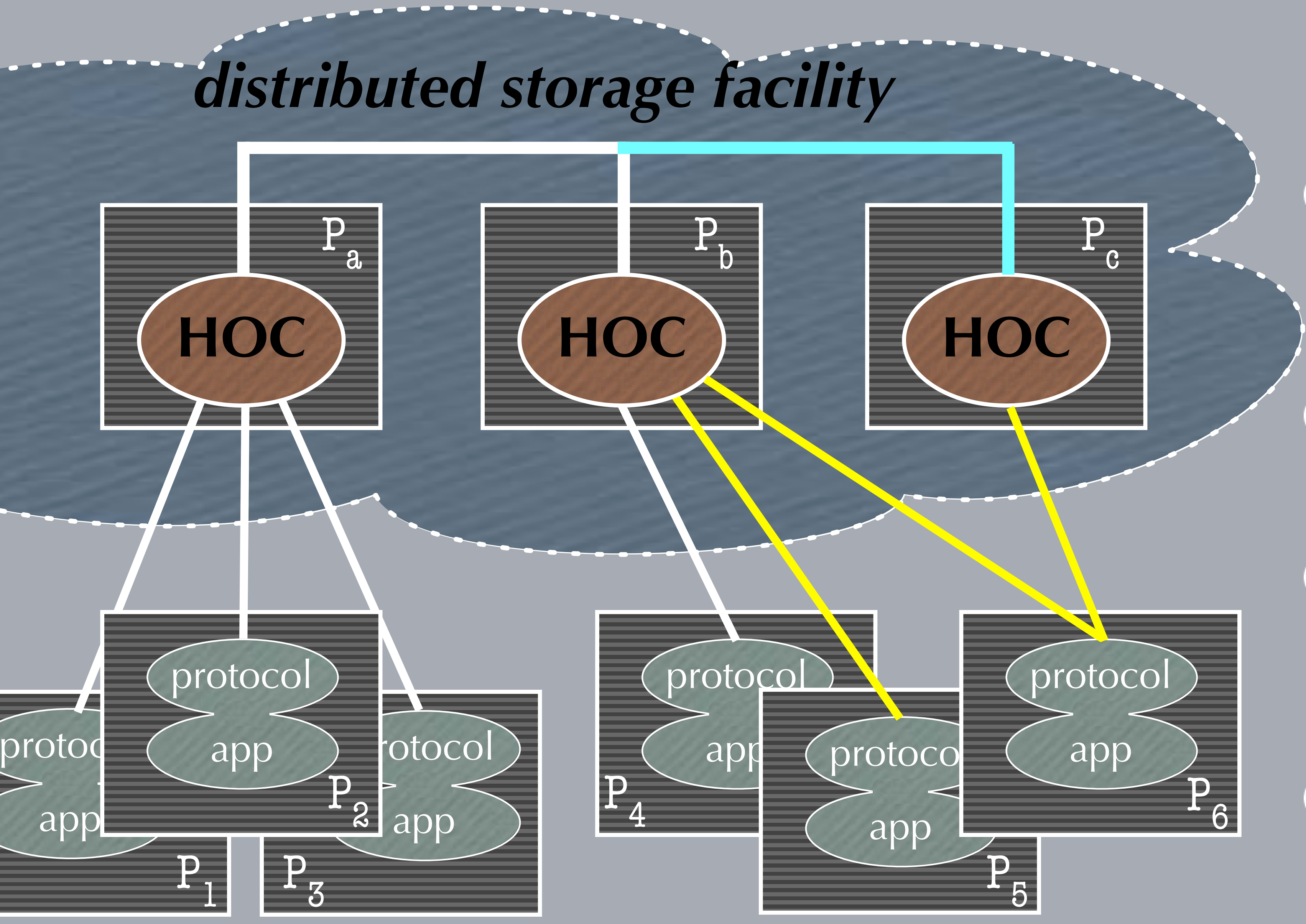
Outline

- HOC (Herd of Object Caches)
  - Motivation
  - Features
- Apache+HOC parallel web server architecture
- Experiments (a lot of)
  - HOC
  - Apache+HOC
- Ongoing & Future work

HOC (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



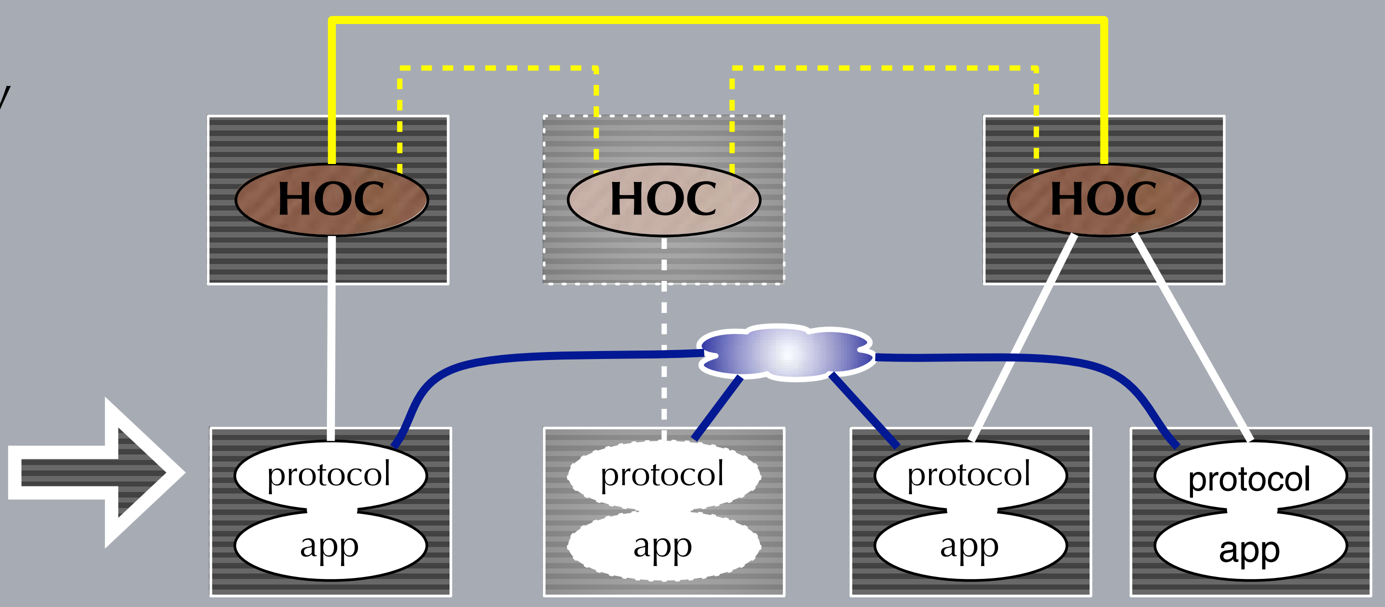
- a facility (distributed server) providing permanent, shared storage to apps (clients)
- clients may dynamically join/ leave the storage facility
- HOC set may be hotly enlarged/ reduced on need - storage room change accordingly
- interaction with HOCs may be delegated to application-specific protocol

Why using HOC

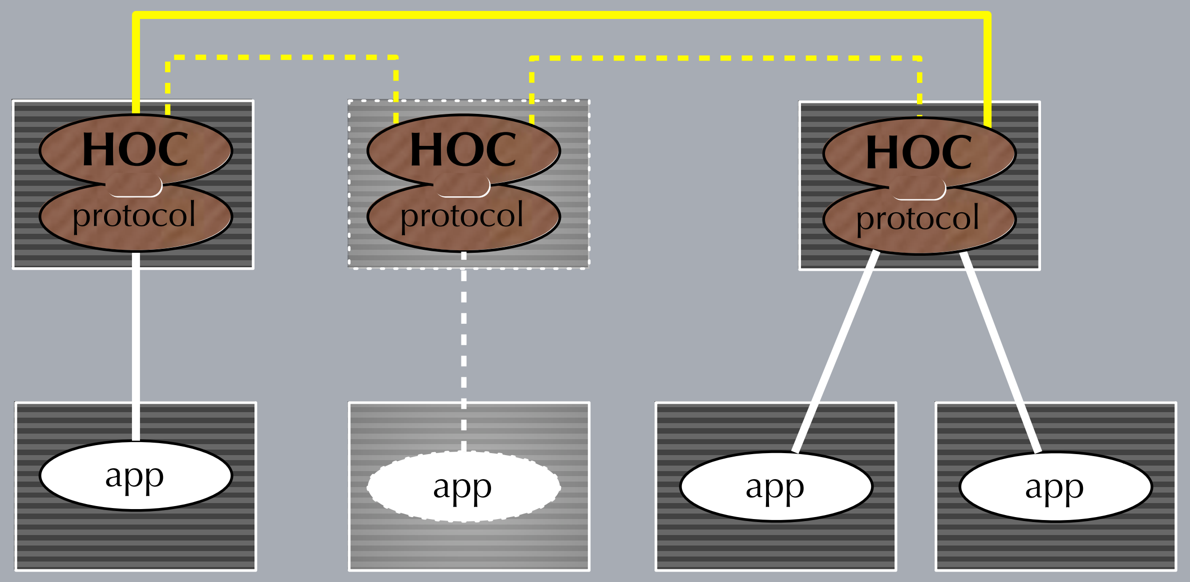
- is efficient (because essential)
  - HOC provide few primitives and no policies for data integrity (e.g. coherence, consistency, ...)
  - these are application specific and may be deployed upon HOC (at the **protocol** level)
- is a basic building block for broad class of applications
  - may be considered 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

... using HOC

- protocol enforces application requirements on data integrity acting as mediator between the application and HOC
- it is linked to the application and use HOC API
- e.g. Apache module

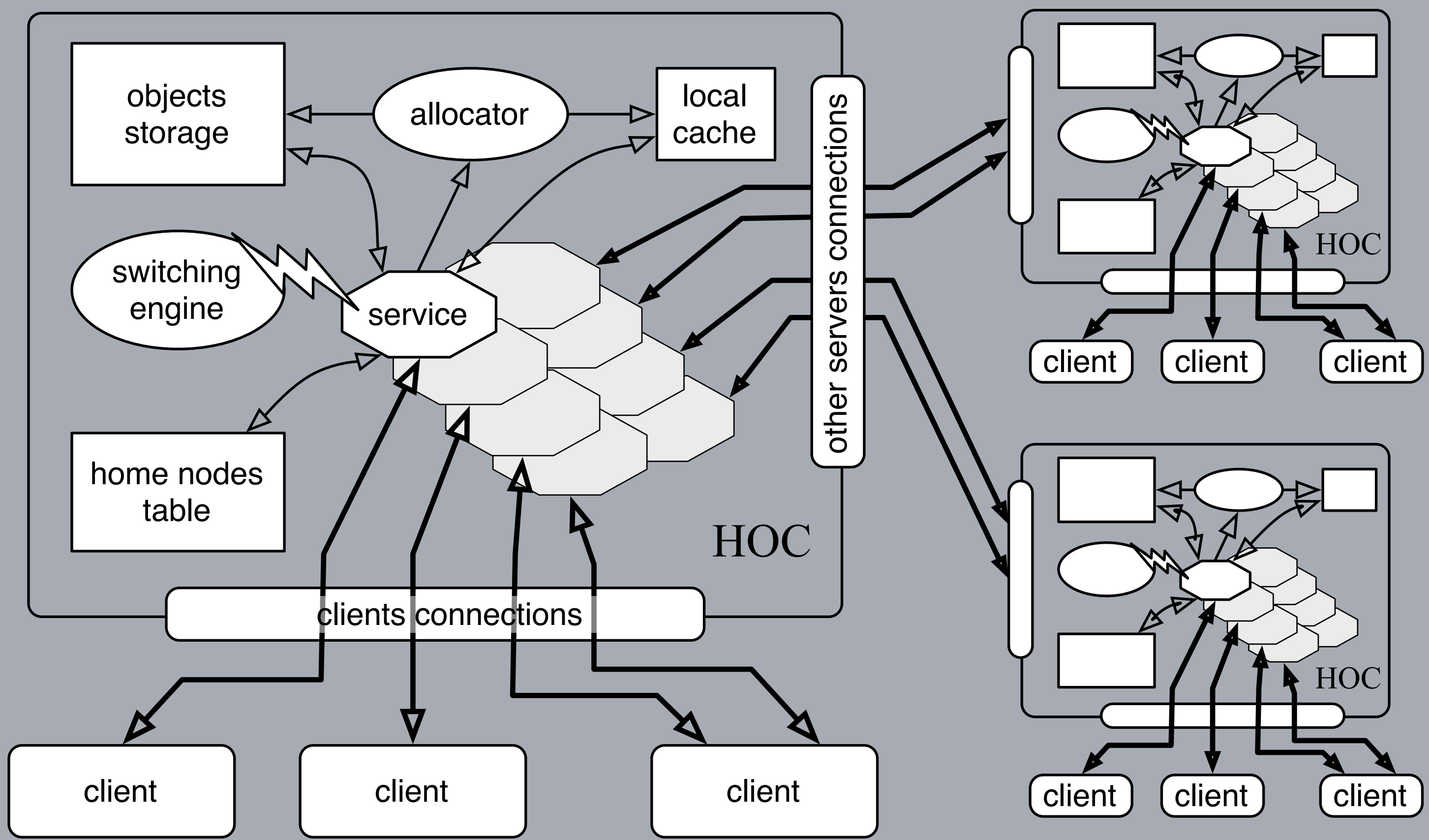


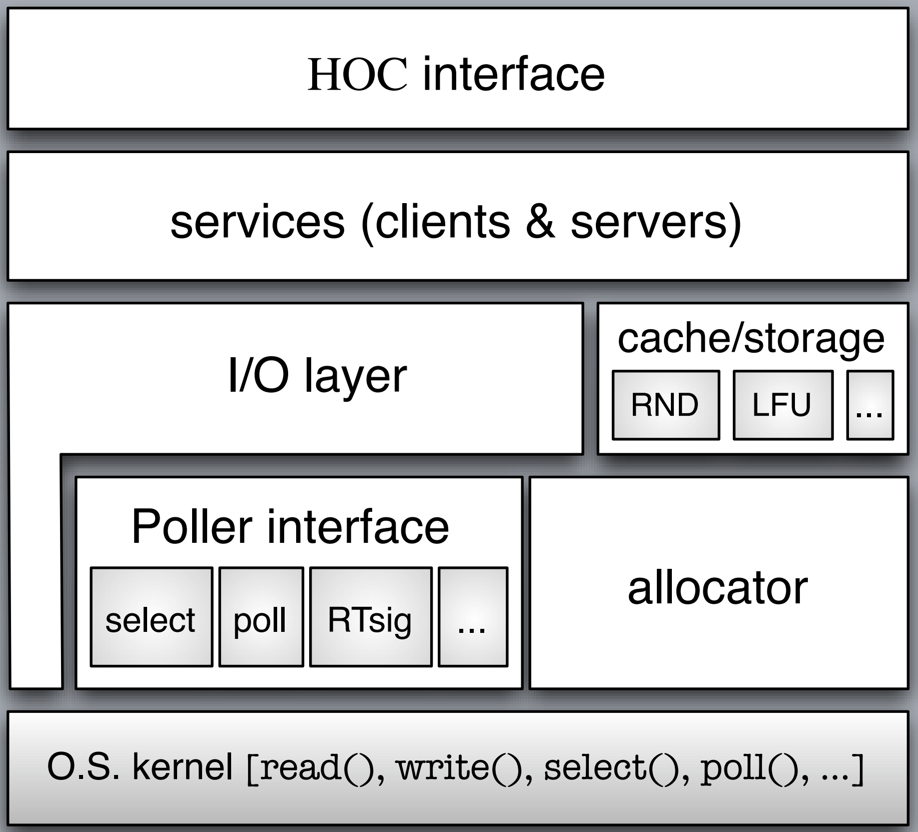
protocol may actually is a distributed application (e.g. reaching consensus, cache invalidation, ...)



HOC API may also be easily extended (provided some knowledge of HOC internals)

HOC internals





- C++, single-threaded, manage concurrent connections using non-blocking I/O based **services** (each of them being a state machine managing a single connection)
  - supporting both level-triggered (`select`, `poll`, ...) and edge-triggered (`RTsignal`, `kqueue`, ...) I/O events
- object storage may be managed either as a memory or a cache, remote objects may be cached in a separate write-through cache. Policies are configurable.
- tested on Linux, MacOS X, and heterogeneous cluster of them

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`**

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

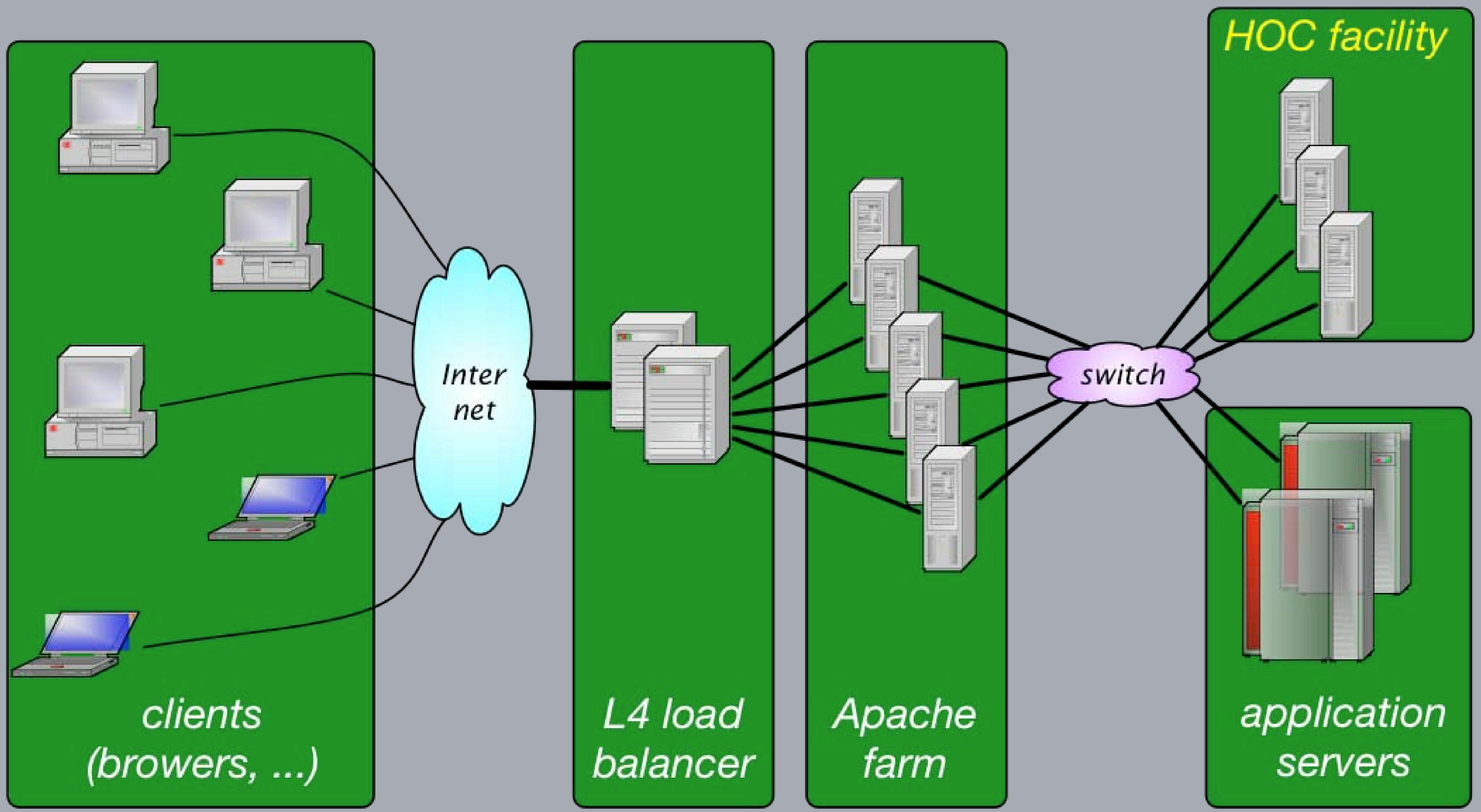
How accelerate a web server/service

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

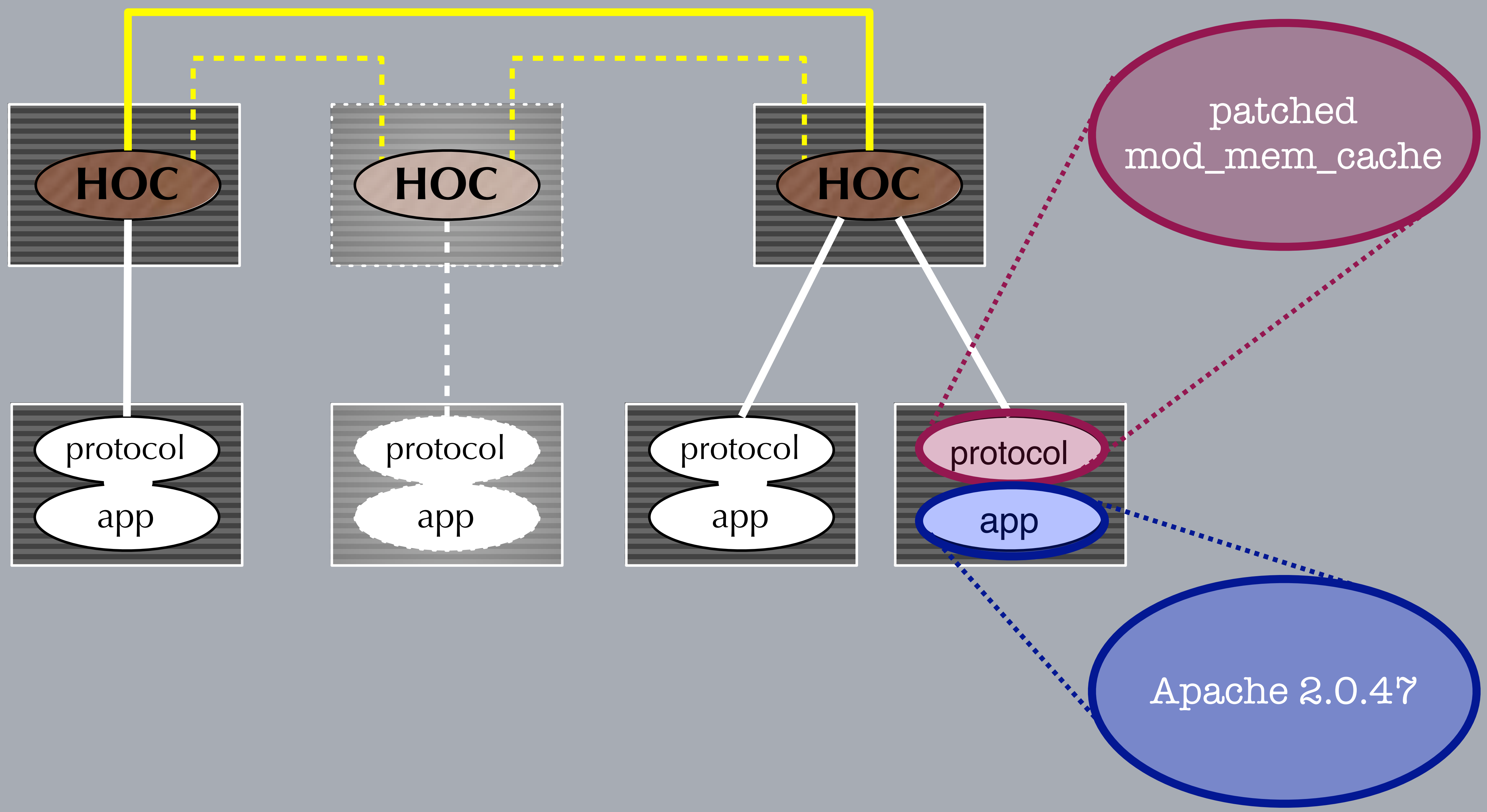
We would like to improve web server performance without changing web server core, thus relying on correctness, people expertise, ...

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

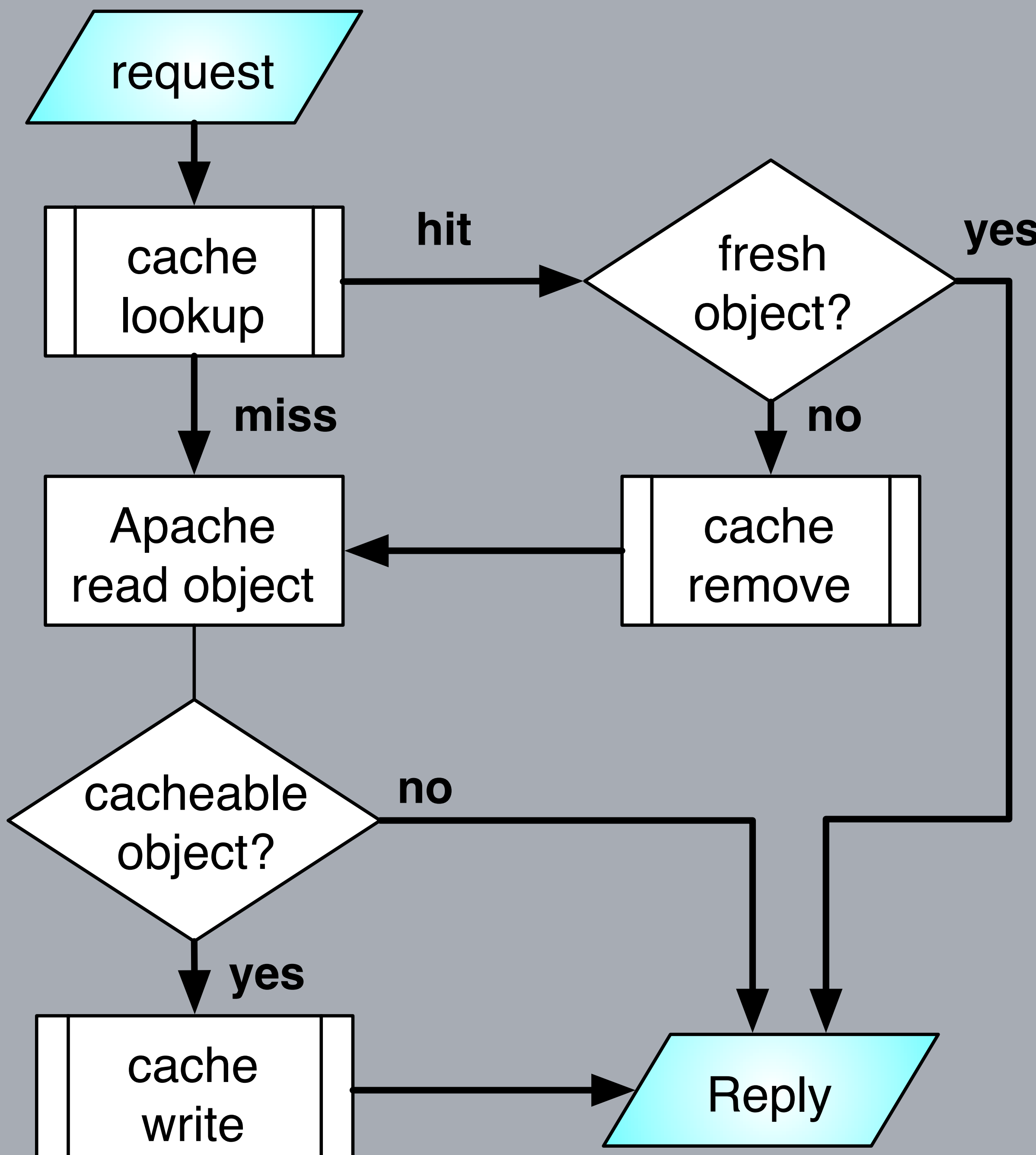
The big picture



The Apache plug-in for HOC

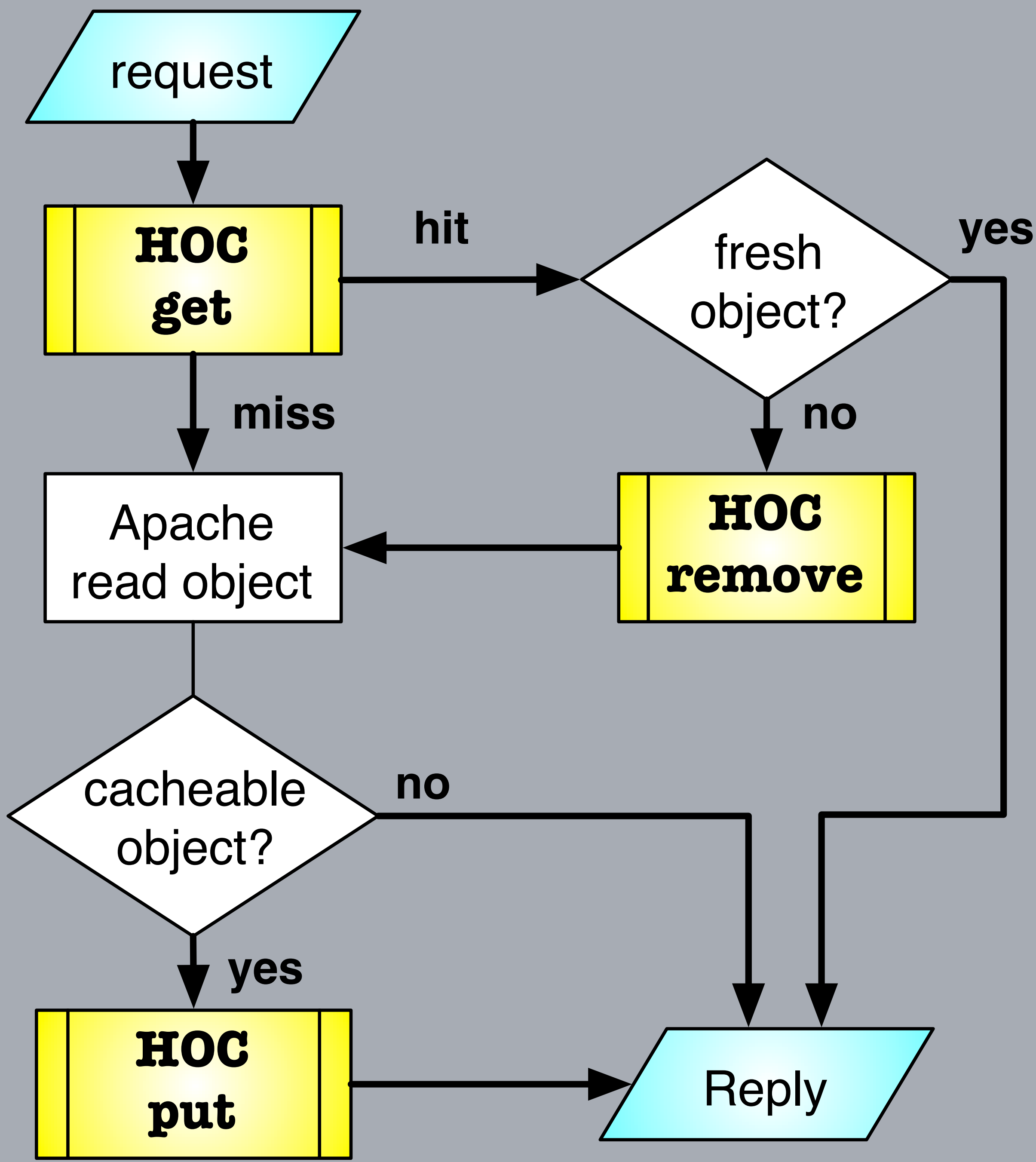


The Apache plug-in for HOC



High-level functional behavior of the Apache 2.0.47 native cache module (mod_mem_cache)

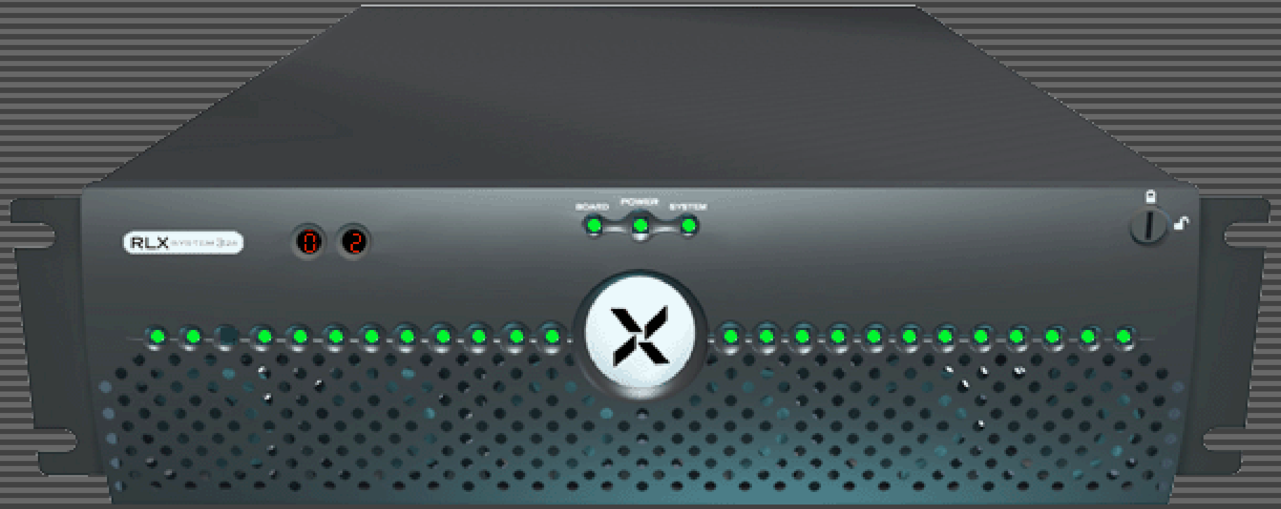
The Apache plug-in for HOC



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

Experiments

RLX blade - 24 P4@800MHz
(outside the room ...)

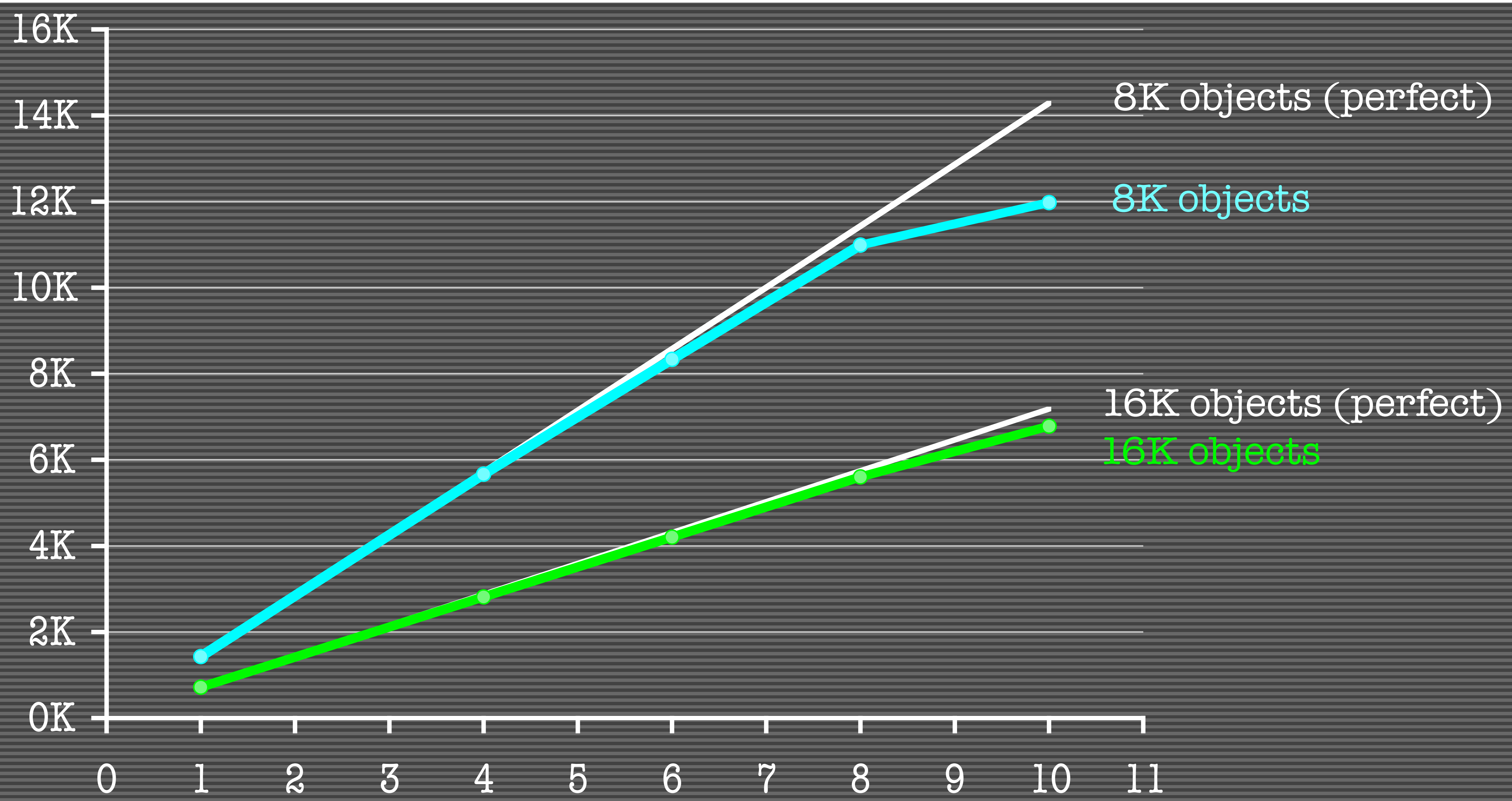


- experimenting HOC
- experimenting Apache+HOC

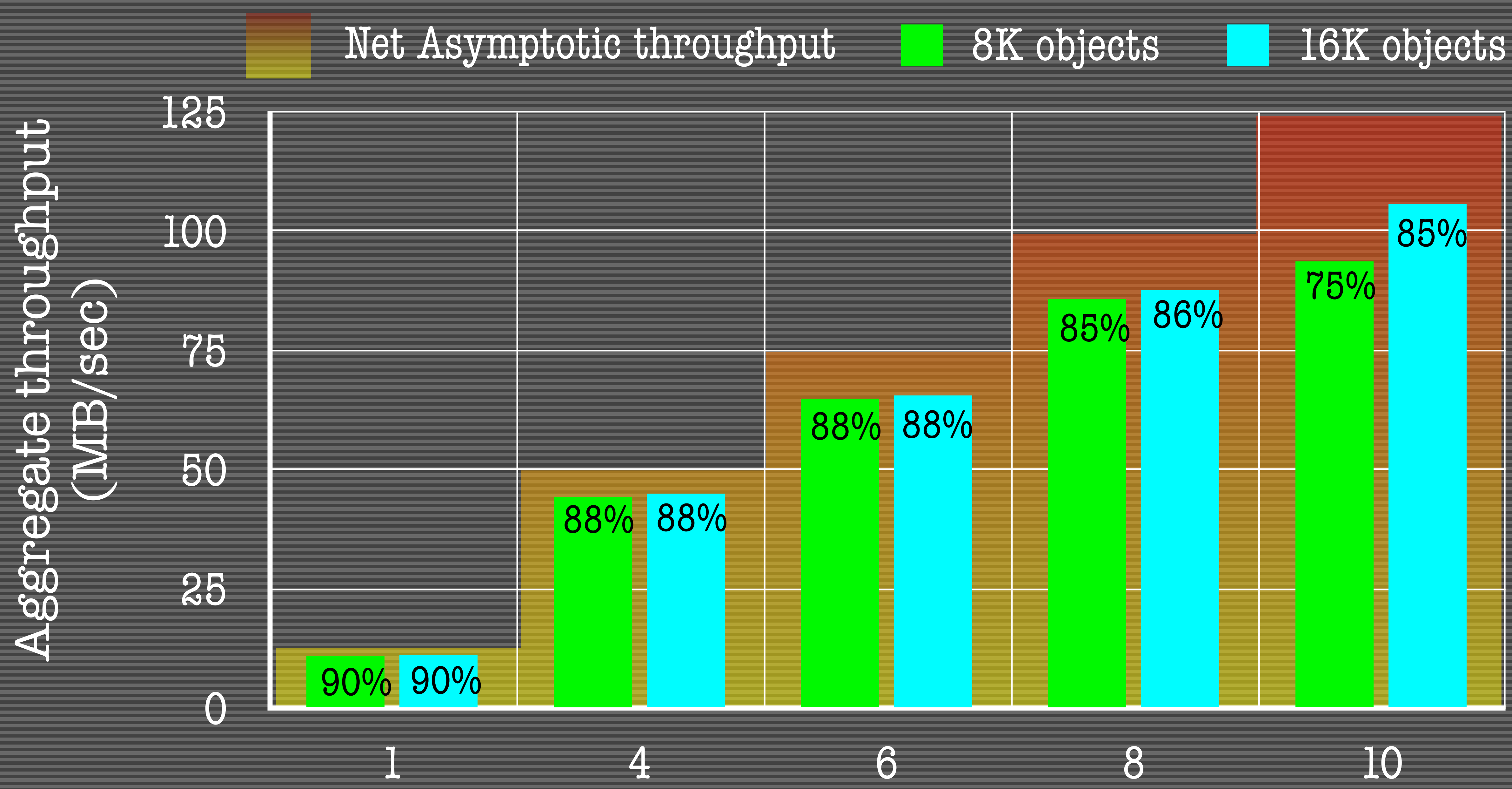
Performance figures (1PE)

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%

Speedup (Hit per sec VS N. servers)



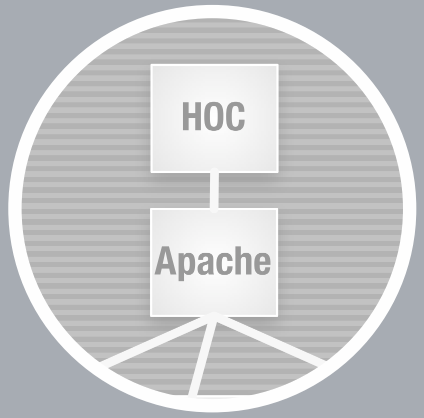
Sustained aggregate throughput

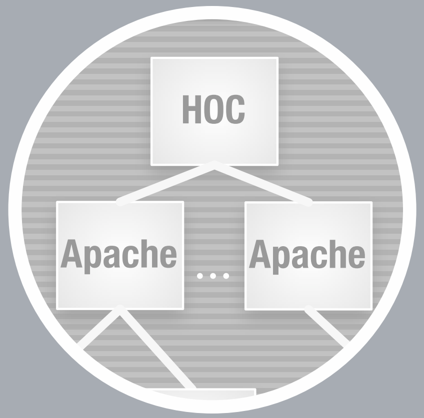


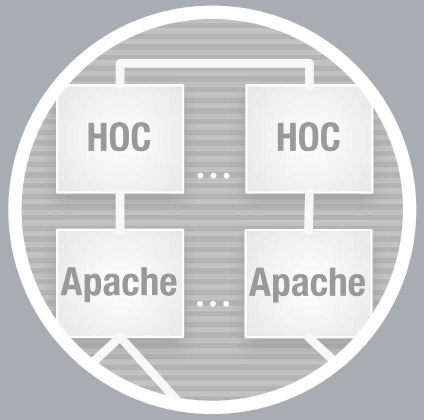
Summarizing

- HOC is a building block for storage-oriented components
  - distributed caches, distributed memories, parallel repositories
  - configurable, hot-pluggable,
- very good performances
  - close-to-ideal net throughput over thousands of concurrent connections
  - close-to-ideal speedup

Hoc+Apache architecture







Three architectures experimented:

- 1-1 1 HOC - 1 Apache
- 1-n 1 HOC - n Apaches
- n-n n HOCs - n Apaches

Experimental environment summary

Raw data set

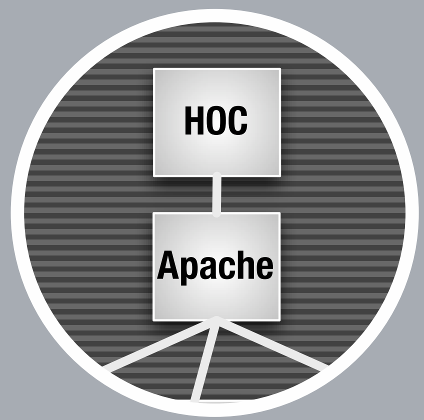
Total size	4GB
N. of files	100K
N. of requests	250K
N. of files <	~100K
Static pages	100%

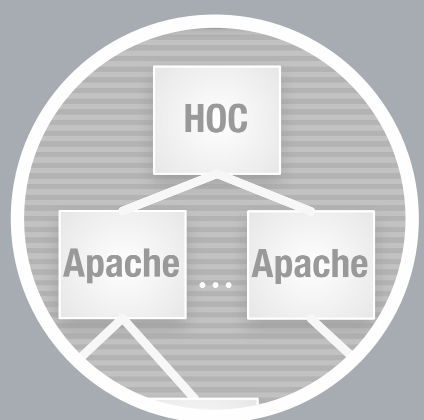
Access log

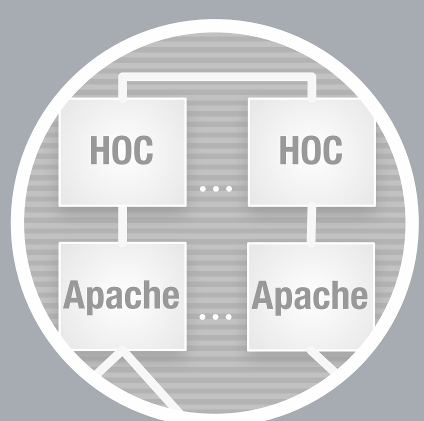
Data transfered	~9GB
N. of distinct files requested	~75K (2.8GB)
Avg. file size	~37KB
N. of distinct files < 256 KB	~100K
Static pages	100%

Apache 2.0.47 MPM worker configuration (hybrid multi-threaded multi-process)

StartServers	4	ThreadPerChild	64
ServerLimit	8	MaxRequestsPerChild	0
MaxClients	512	Log level	Notice
MinSpare Threads	32	Access log	None





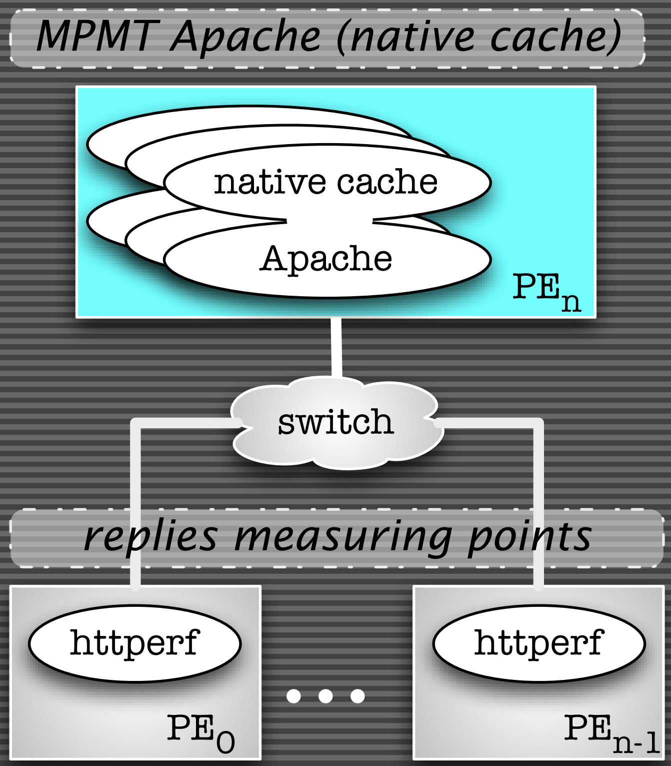


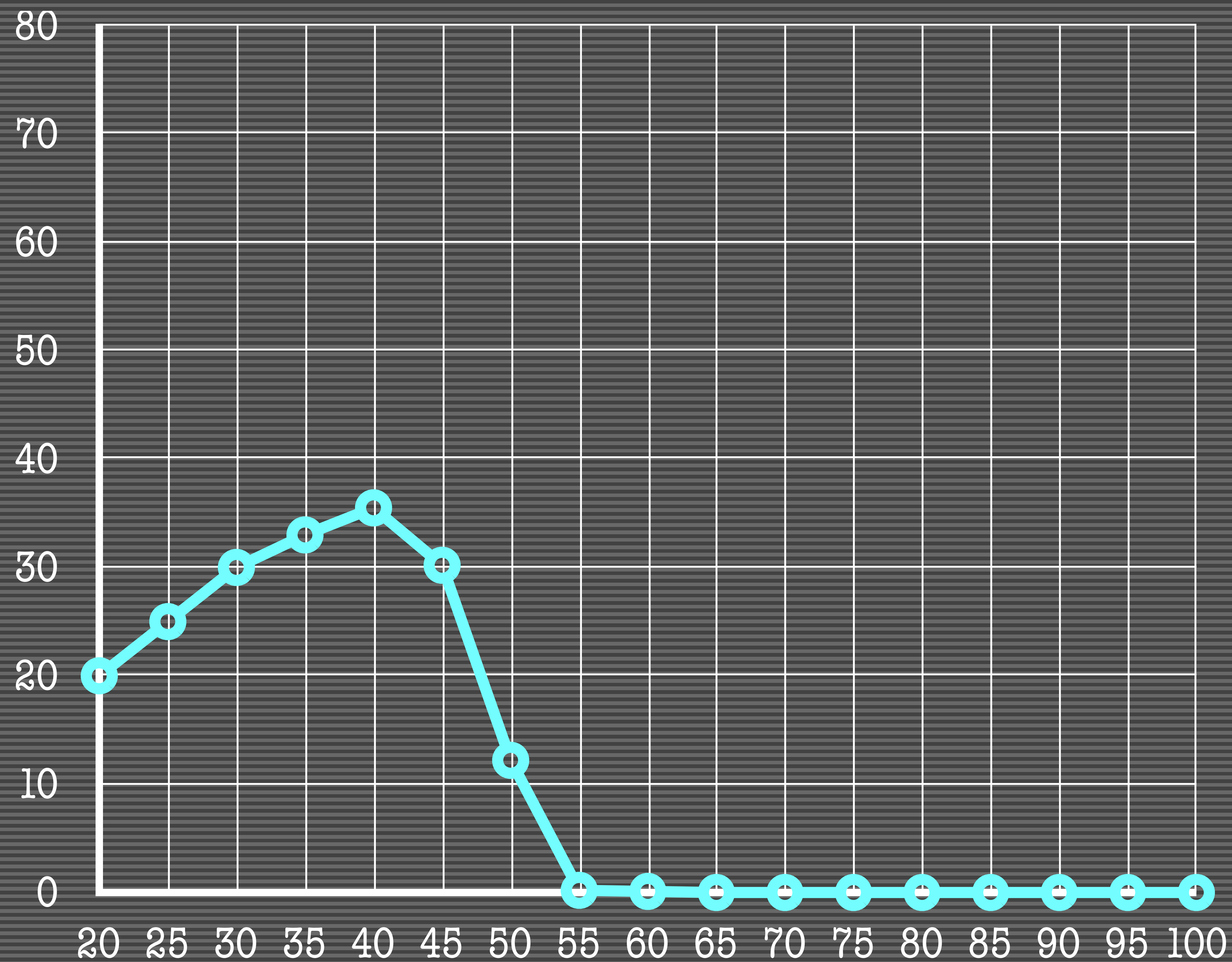
1 HOC -1 Apache, compared architectures:

1. MPMT Apache native cache (per process)
2. SPMT Apache native cache (shared)
3. MPMT with no cache
4. MPMT Apache with HOC cache (on the same PE)
5. MPMT Apache with HOC cache (on different PEs)

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

1-1

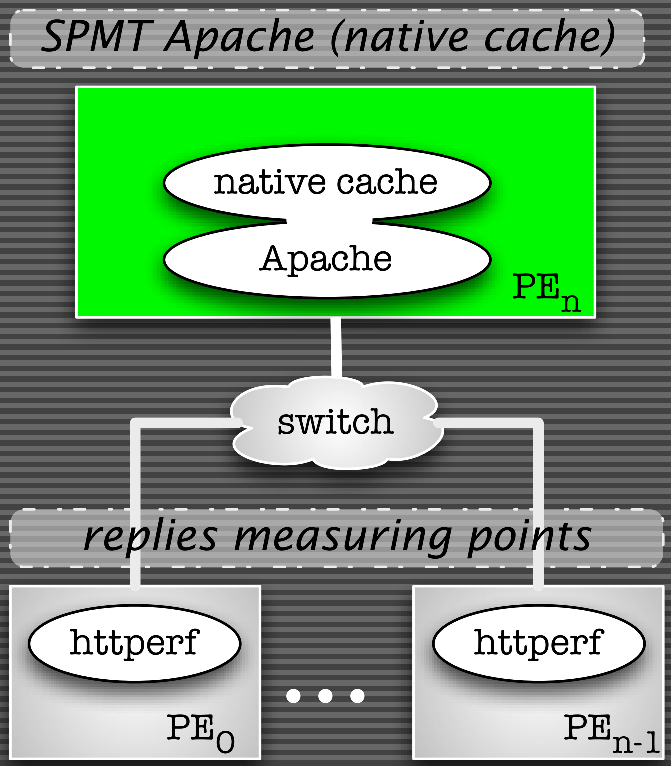


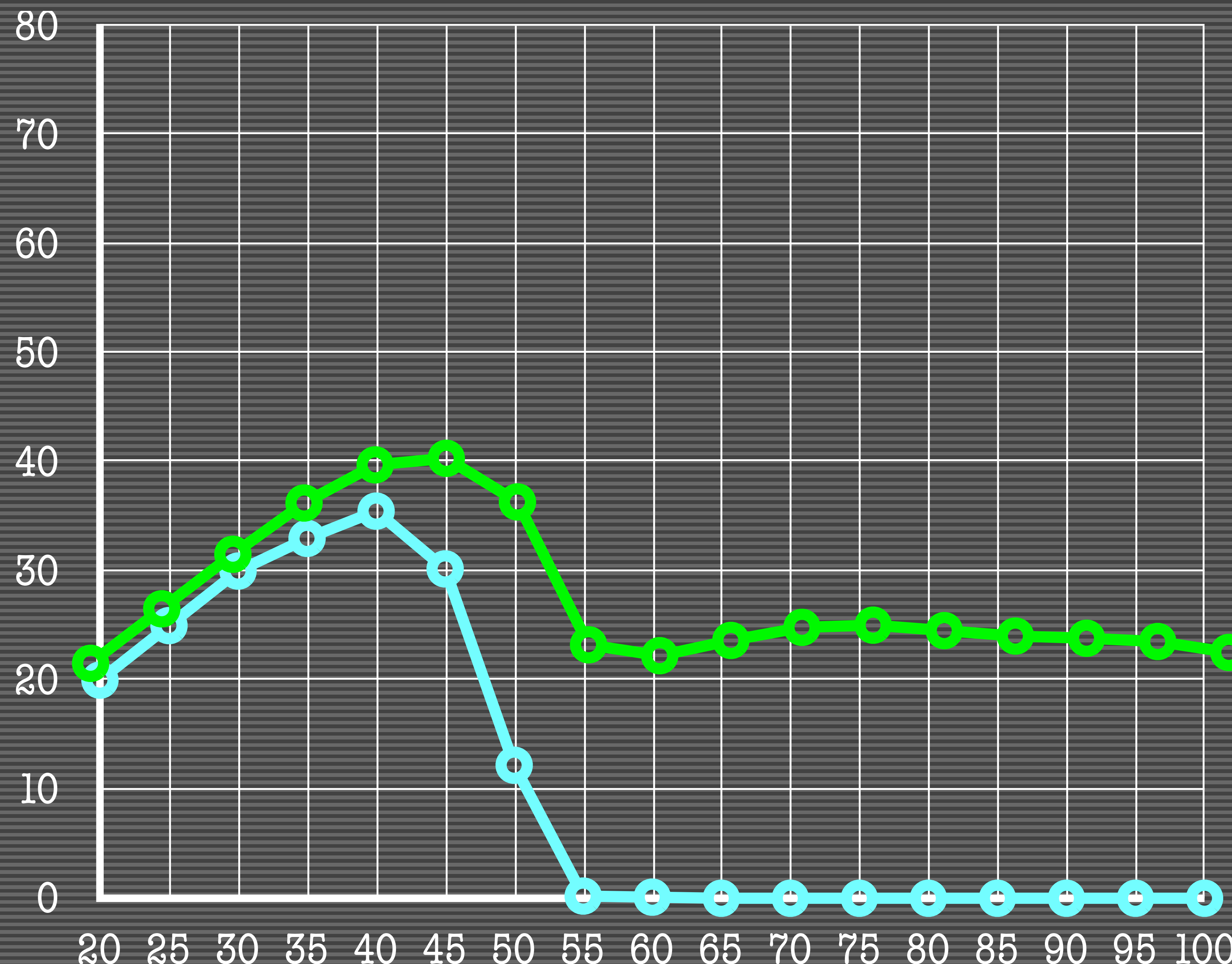


MultiProcessMultiThreaded (150MB native cache per process)

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

1-1

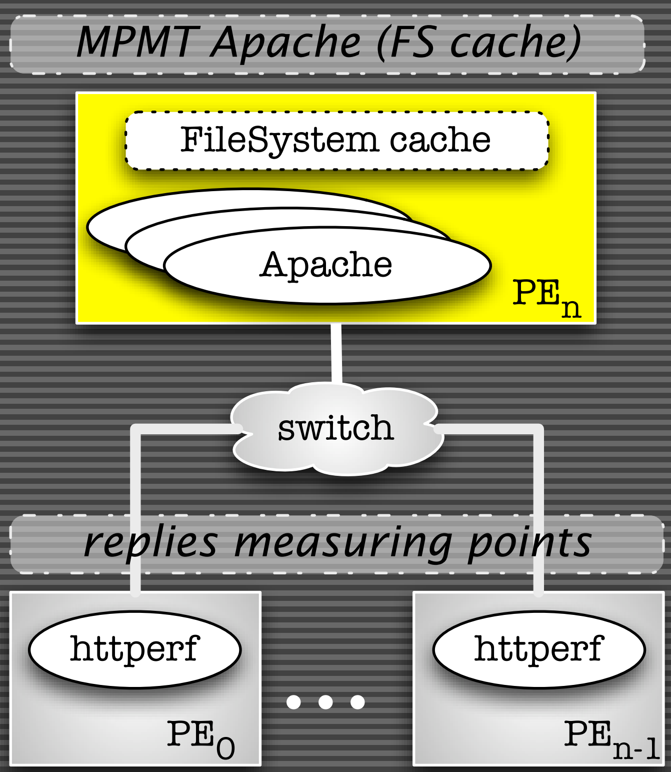


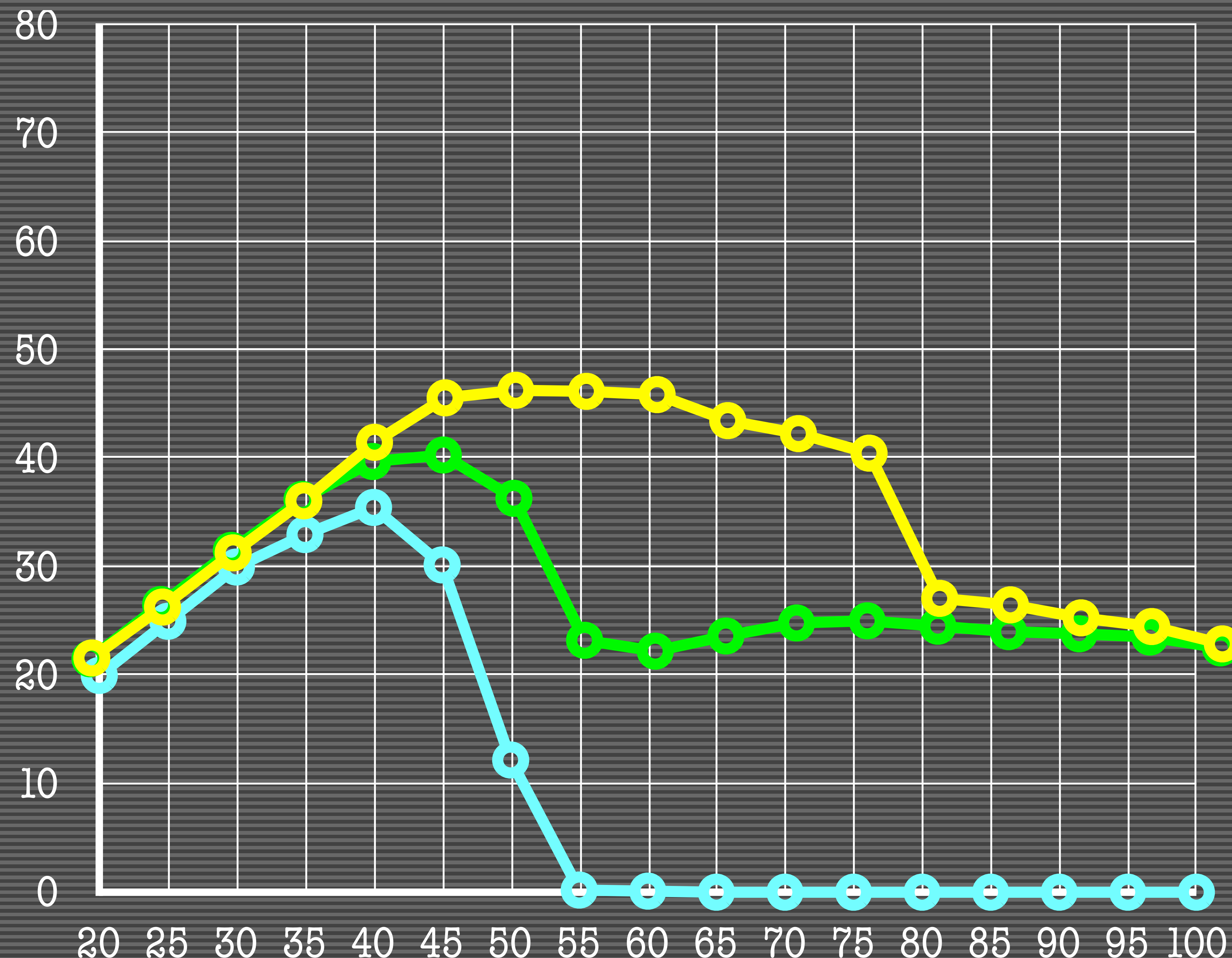


SingleProcessMultiThreaded Apache (900MB shared native cache)

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

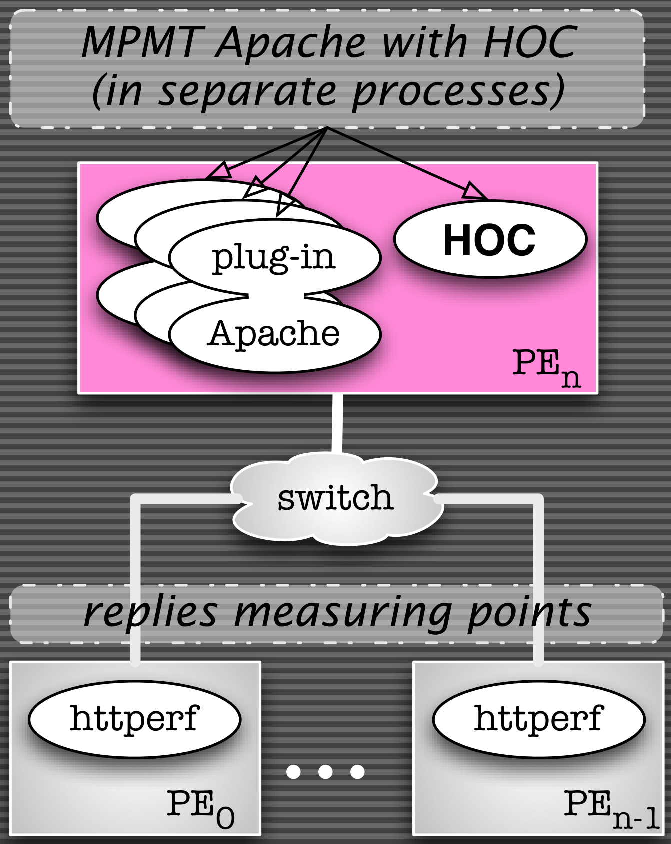
1-1

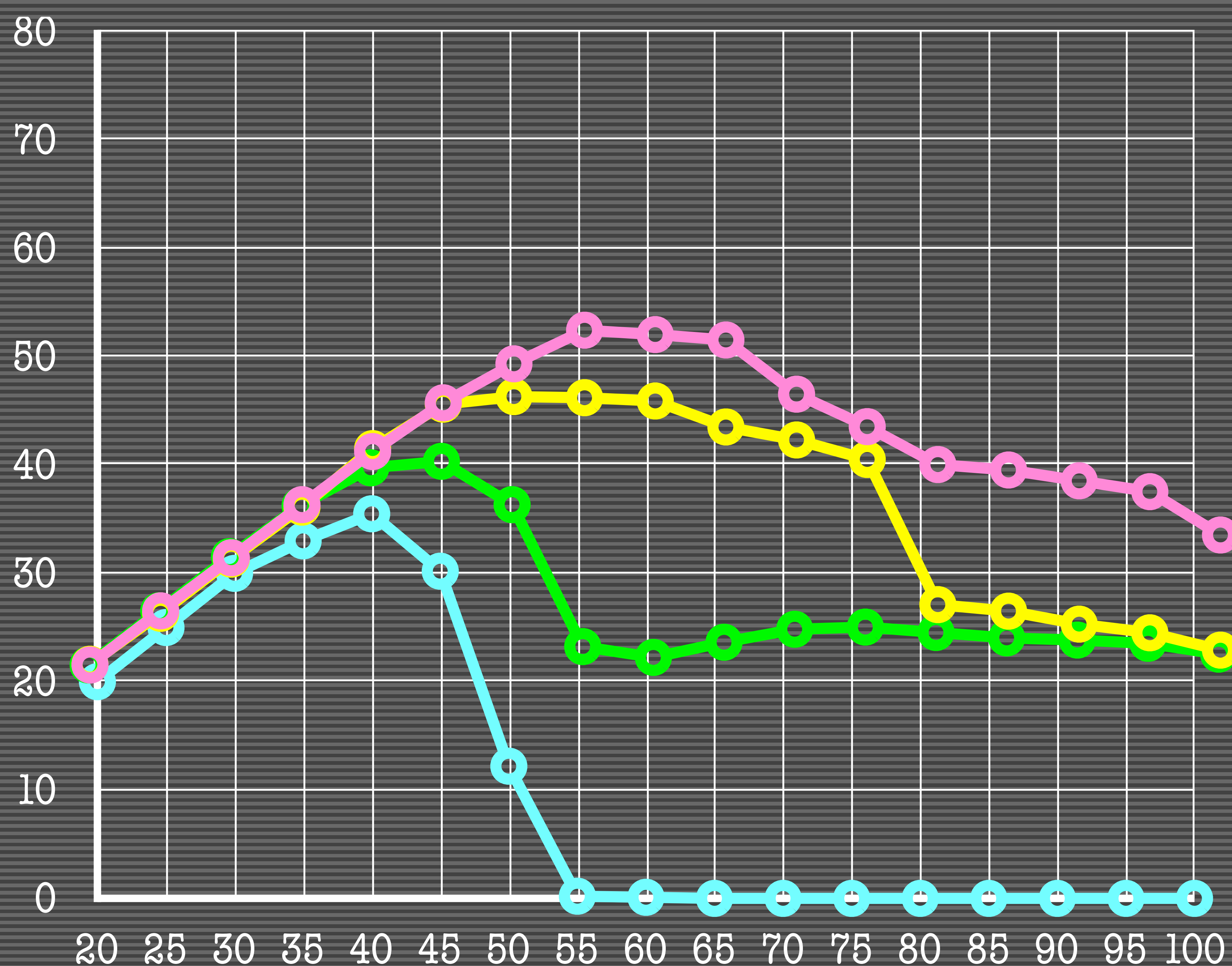




NoCache MPMT Apache (FileSystem buffer behaves as cache)

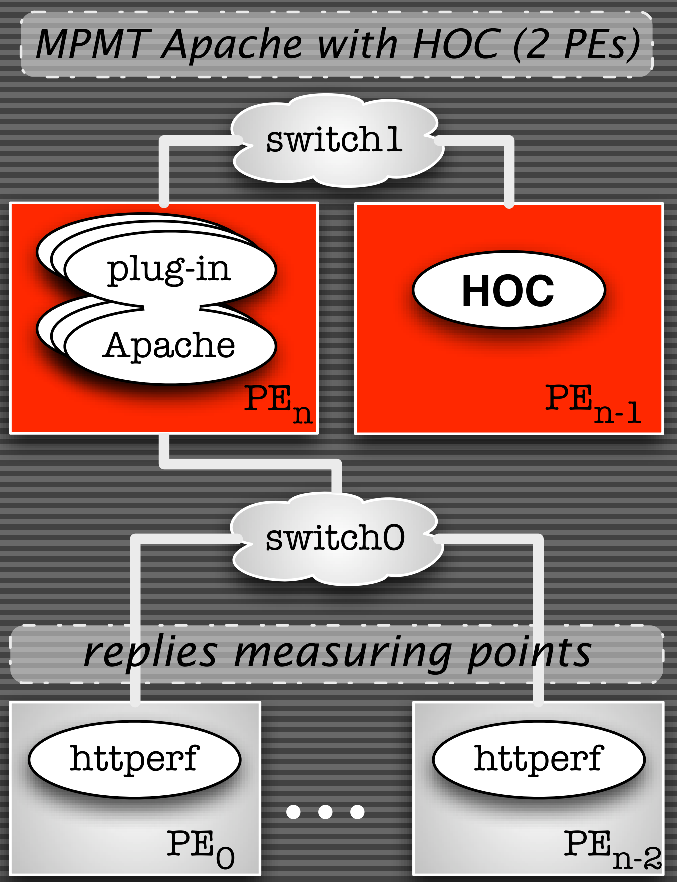
Comparing reply rate (1-Hoc/1-Apache/k-httperf)

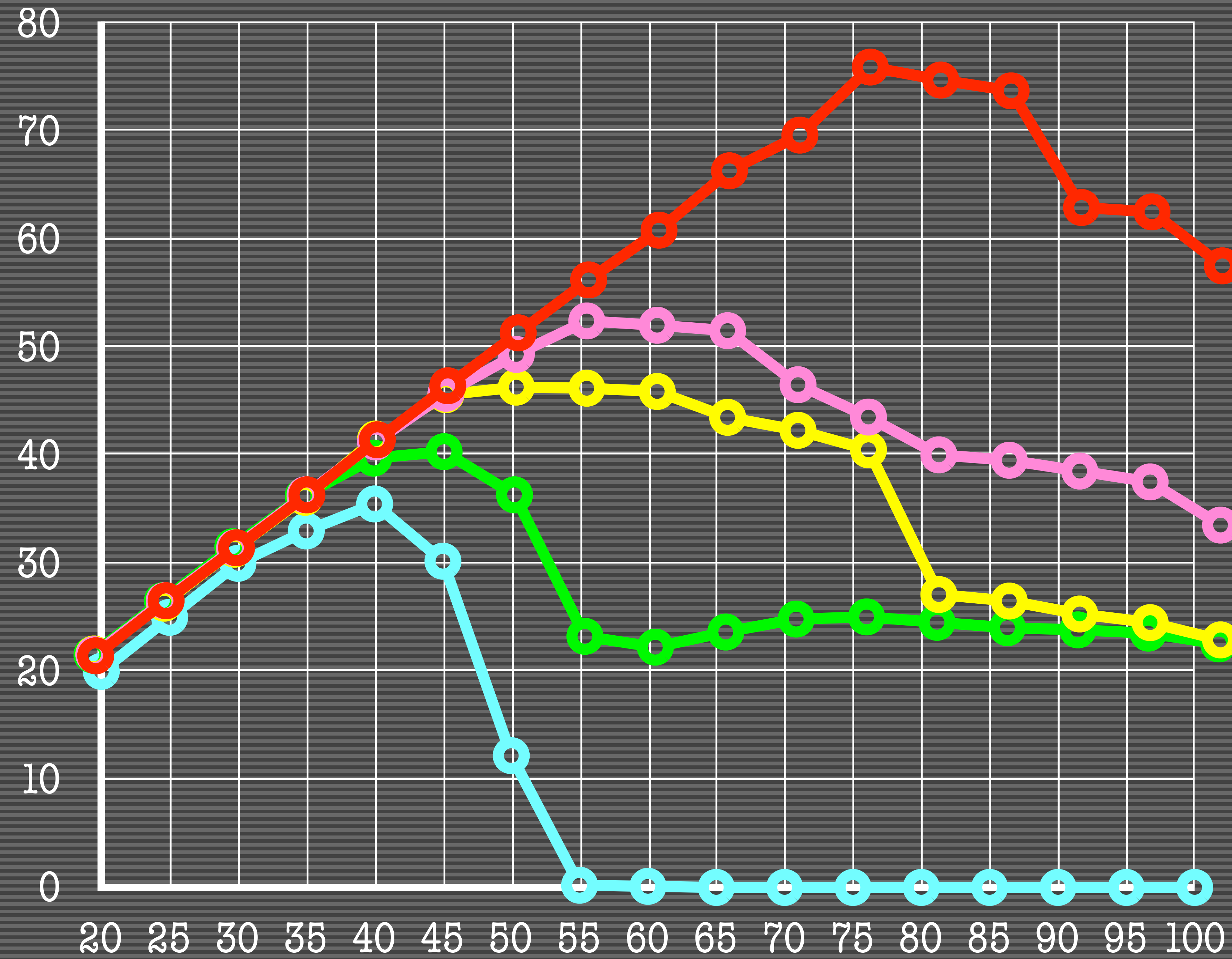




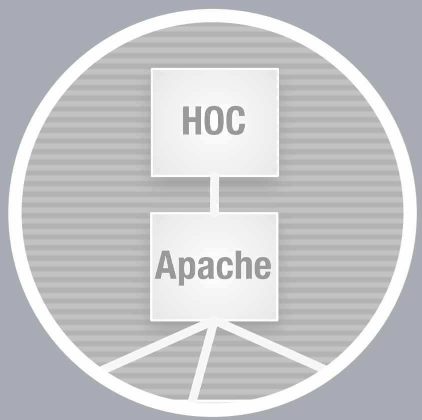
MPMT Apache with **450MB HOC on the same box**

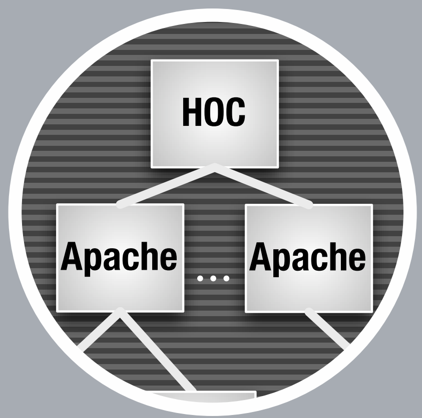
Comparing reply rate (1-Hoc/1-Apache/k-httperf)

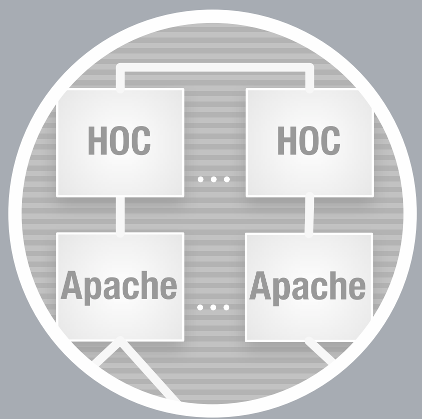




MPMT Apache with **900MB HOC on 2 boxes**



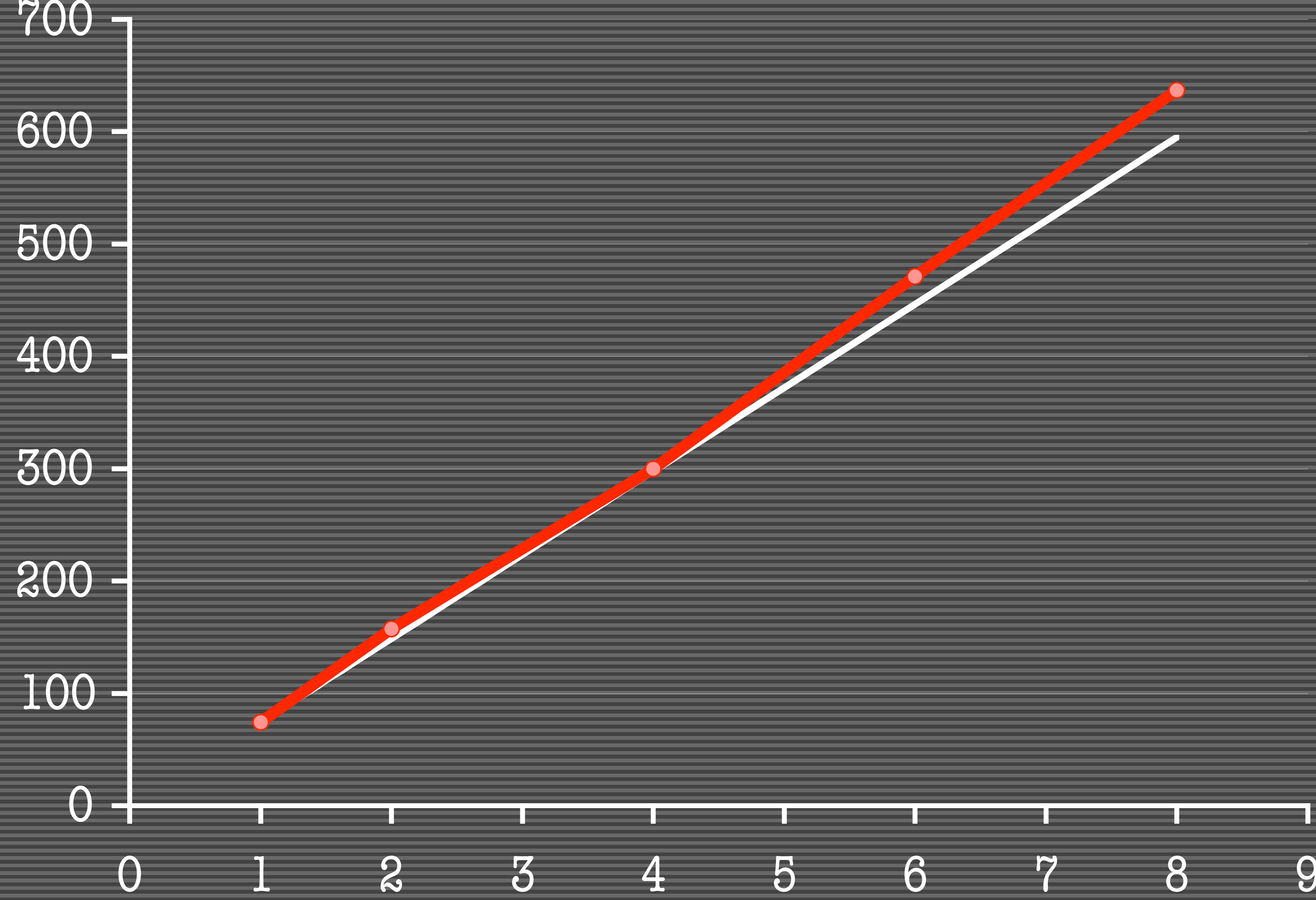


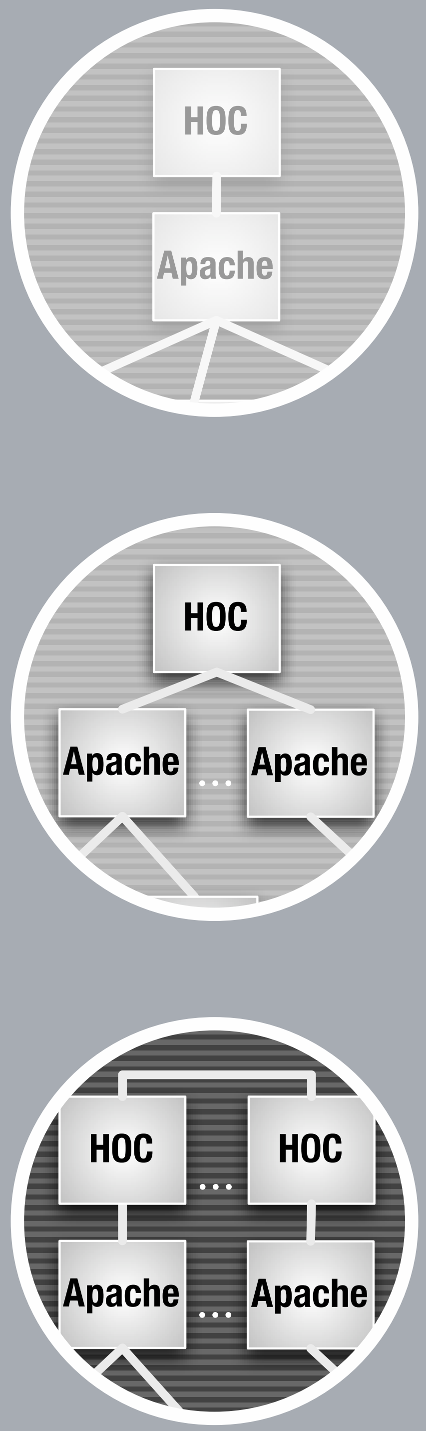


1 HOC -n Apache

- A single HOC acting as external, shared cache for many Apaches (Apache farm). Speedup measure.
- How many Apaches a single HOC may support? Does "optimal number n" exist?

Speedup

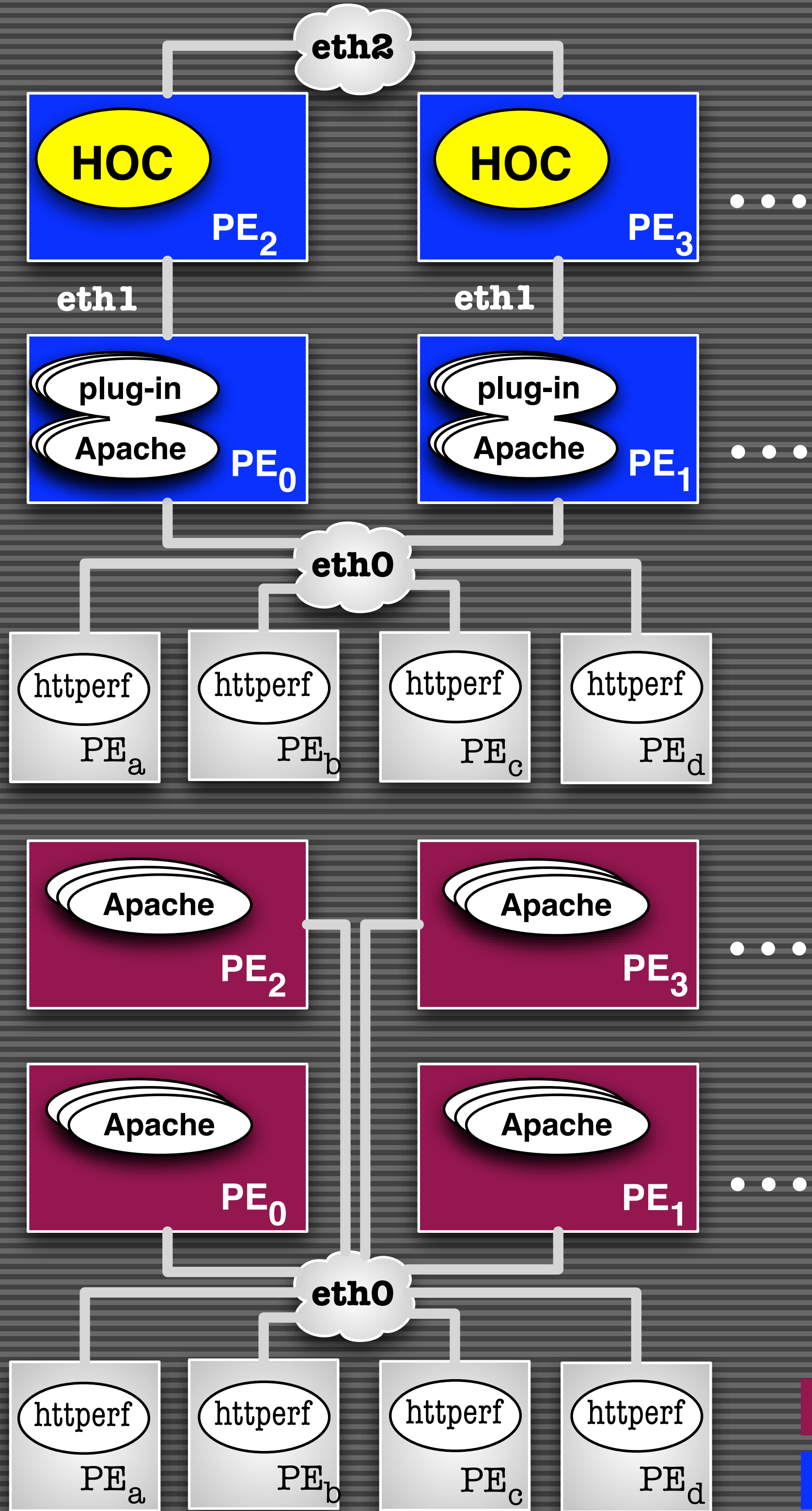




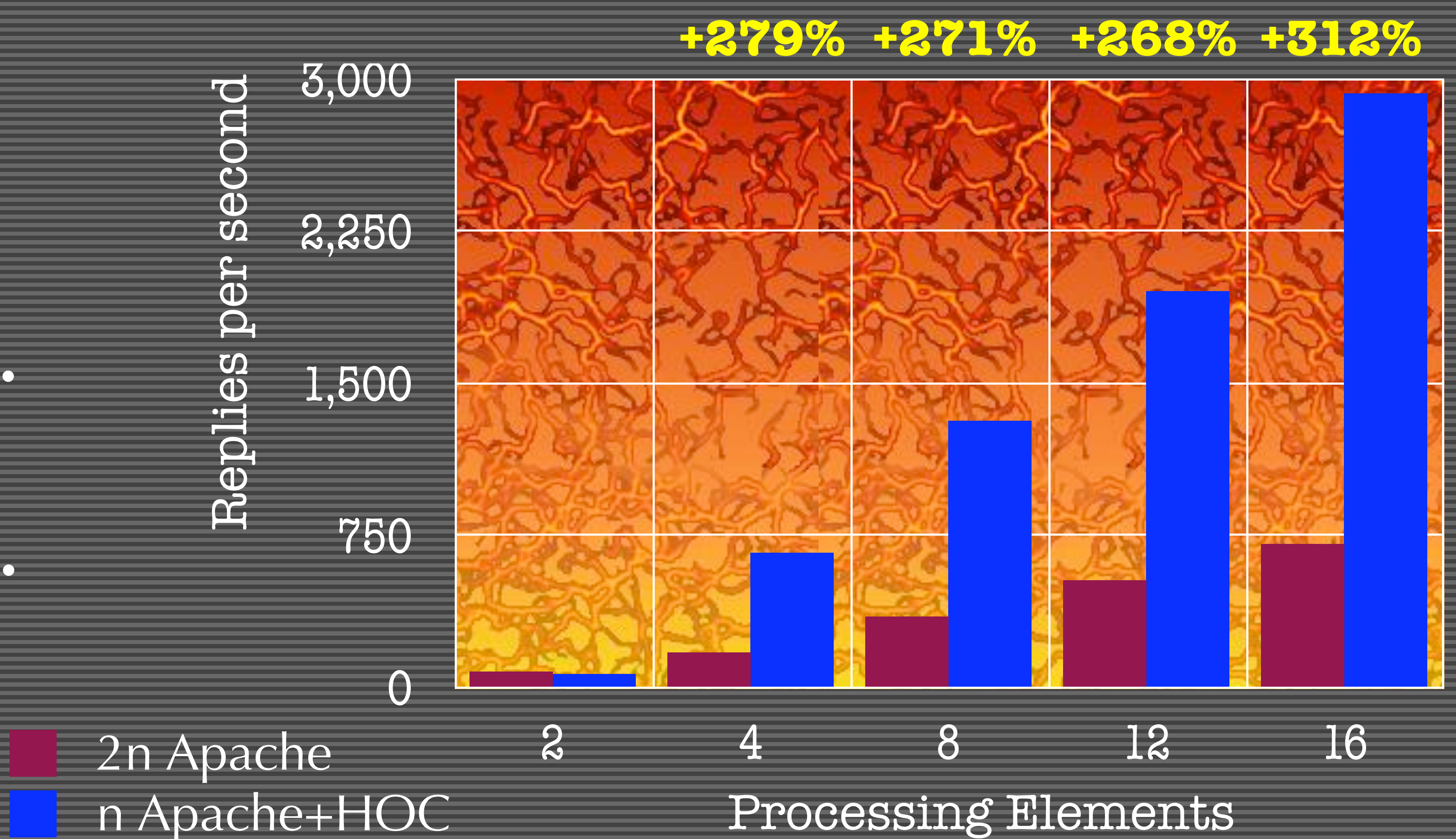
n HOC -n Apache

- Many HOC acting as external, shared cache for many Apaches (Apache farm). Speedup measure.
- How many Apaches a single HOC may support? Does “optimal number n” exist?

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



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



Current & future work

- Supporting heterogeneous cluster (done!)
- integration with the ASSIST environment (ongoing)
  - ASSIST has “external shared objects” at the language level
  - supporting dynamic reconfiguration (state migration)
- distributed in-memory File System PVSF-like (beta)
- SMP scalability (multi-threading) (in agenda)
- web-services interface (in agenda)

Conclusions

Thank you!
Questions?

- HOC is fast and scalable storage component running heterogeneous clusters
  - hot-pluggable, sustain thousands of flowing concurrent connections
- easily adaptable for different I/O bound applications, e.g. Apache, FS, ...
- Apache+HOC improves Apache performances without any change to the Apache core code
  - in the sequential architecture (20% on the same PE, 100% with an additional PE)
  - in several flavors of parallel architectures:1-n, n-n, n-m (with a 300-400% gain with equal resource cost)
- **HOC is open source, and come with the ASSIST package**

Thanks to Alessandro Petrocelli and the whole Pisa HPClab people