SE Computational Systems Seminar

SS 2004

Flash: An efficient and portable Web Server

Vivek S. Pai, Peter Druschel, Willy Zwaenepoel

Presentation by:
Zenina Huskić (zehuskic@cosy.sbg.ac.at)

27. May, 2004

University of Salzburg
Department of Computer Science

Overview

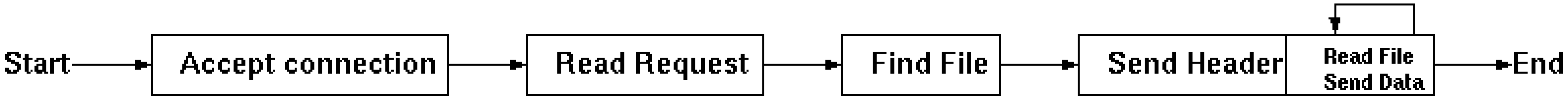
- Motivation
- Server Architectures
- Design comparison
- Flash Implementation
- Performance Evaluation
- Conclusion

Motivation 1

- Goals:
  - → High throughput,
  - → Good portability,
  - → Wide range of workloads.

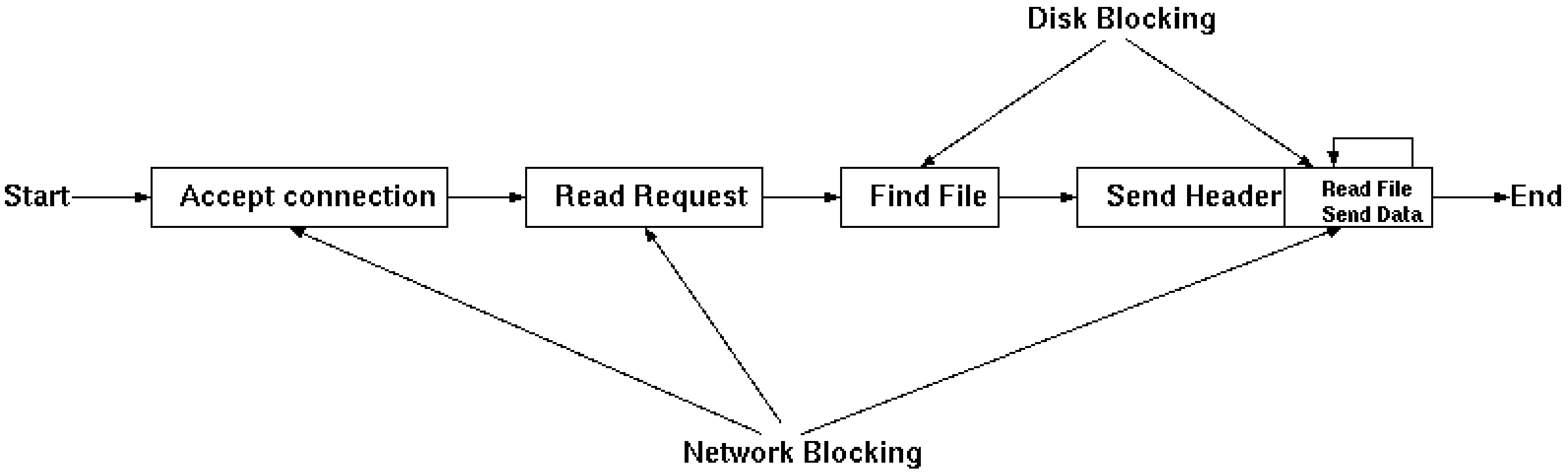
Motivation 2

- Basic processing steps performed by a Web Server



Motivation 3

- Problem: Blocking steps

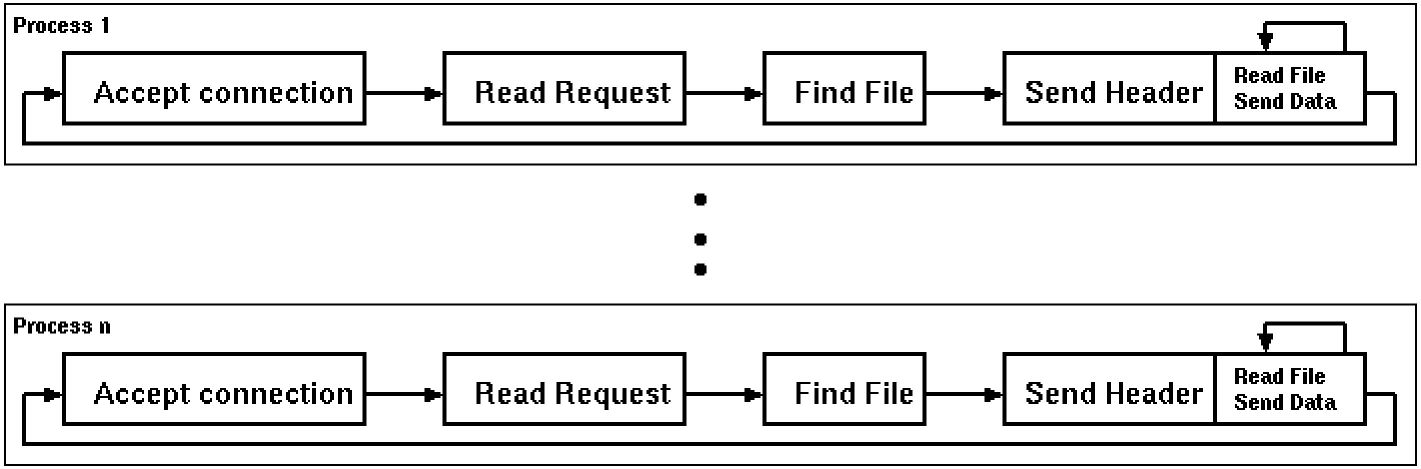


Motivation 4

- Solution:
  - → concurrency architectures
    - overlap CPU processing with disk accesses and network communication.
  - → caching
- Architecture - strategy, used to achieve the interleaving.

Server Architectures 1

- Multi-Process architecture



Process 1

Accept connection → Read Request → Find File → Send Header → Read File Send Data

Process n

Accept connection → Read Request → Find File → Send Header → Read File Send Data

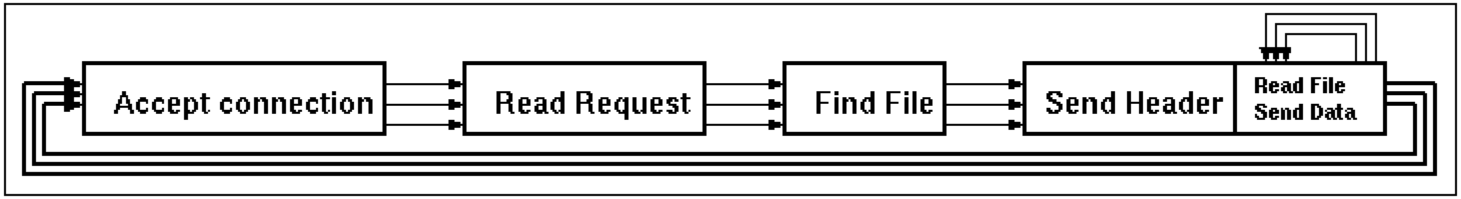
+ many HTTP requests may be served concurrently,

+ relies on OS.

– many processes.

Server Architectures 2

- Multi-Threaded architecture



+ shared global variables.

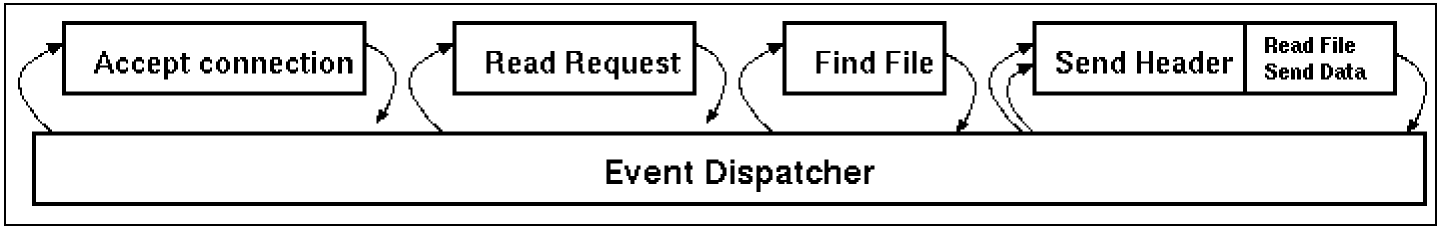
– many threads,

– requires kernel thread support,

– requires synchronization.

Server Architectures 3

- Single-process event-driven architecture (SPED)



+ single address space,

+ no context switching required,

+ no synchronization required.

– in practice, disk reads still block.

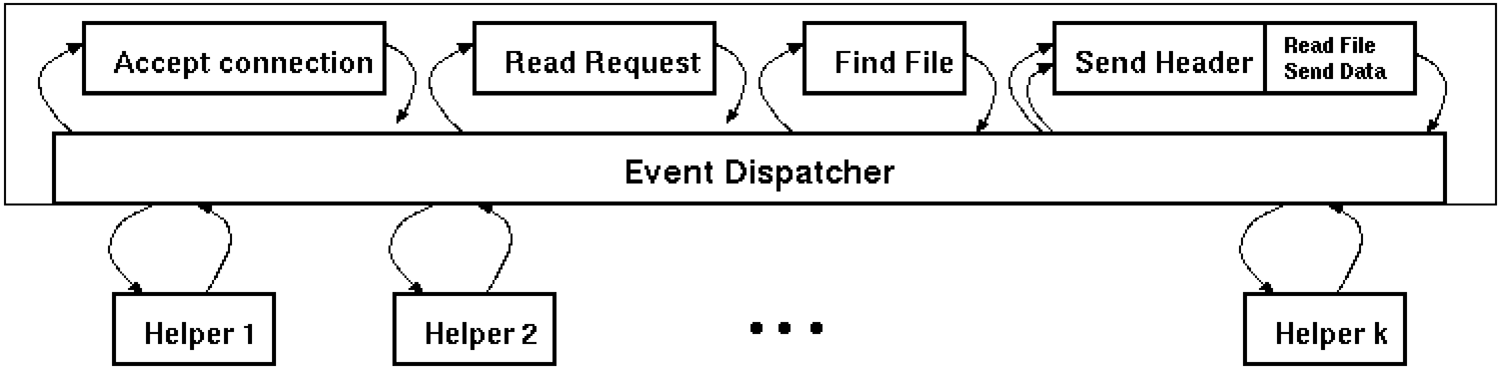
Server Architectures 4

- Desired:
  - → Shared (single) address space,
  - → Good disk behavior,
  - → No synchronization.

Server Architectures 5

- New architecture:

  Asynchronous/Asymmetric Multi-Process Event-Driven architecture (AMPED)



Design comparison 1

- Performance characteristics

	MP	MT	SPED	AMPED
Disk operations	Only the process that causes the disk activity is blocked	Only the thread that causes the disk activity is blocked	The main server process is blocked	Only the helper process that handles the disk activity is blocked
Memory consumption	High memory requirements	Single process memory requirements plus memory requirements for each thread employed	Single process memory requirements	Single process memory requirements plus additional memory for the helper processes
Disk utilization	One disk request per process	One disk request per thread	One disk at a time	One disk request per helper

Design comparison 2

- Cost/Benefits of optimizations

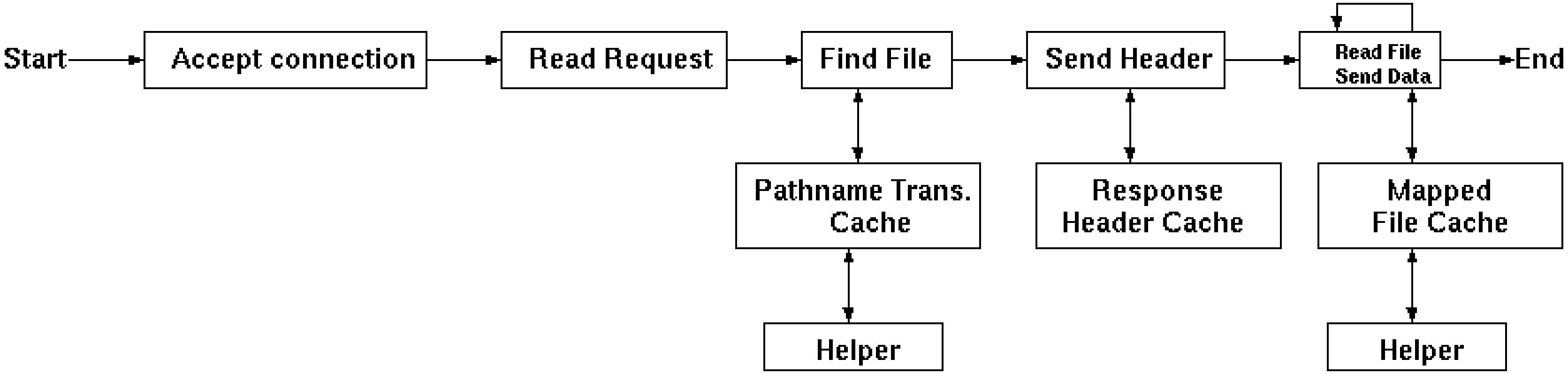
	MP	MT	SPED	AMPED
Information gathering	Requires some form of IPC in order to consolidate data	Requires synchronization on global variables	Simple information gathering since all requests are processed in a centralized fashion	Simple information gathering since all requests are processed in a centralized fashion
Application level caching	Each process may have its own cache	Single cache with synchronization	Single cache without synchronization	Single cache without synchronization
Long-lived connections	Overhead of an extra process for each connection	Overhead of an extra thread for each connection	Overhead of a file descriptor, application-level information and some kernel state for the connection	Overhead of a file descriptor, application-level information and some kernel state for the connection

Flash Implementation 1

- High performance implementation of the AMPED architecture.
- Various optimizations.

Flash Implementation 2

- Three types of caching:
  - → pathname translation caching,
  - → response header caching,
  - → mapped files caching.



Performance Evaluation 1

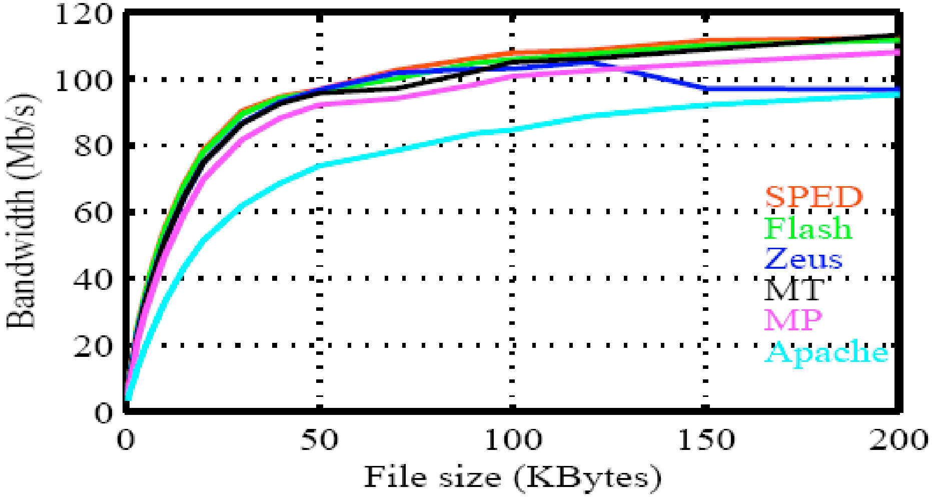
- Test environment
  - → Server Hardware:
    - 333 MHz Pentium II
    - 128 MB memory
    - Five 100 Mbit/s Ethernet interfaces
  - → Operating Systems:
    - FreeBSD 2.2.6
    - Solaris 2.6

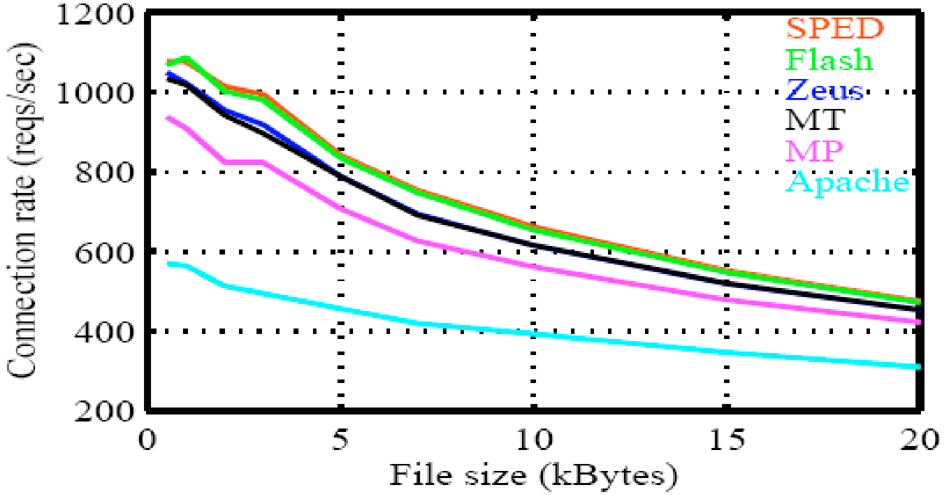
Performance Evaluation 2

- Test environment
  - → Server Software:
    - Apache 1.3.1 - MP
    - Zeus 1.30 - SPED
    - Flash - AMPED
    - Flash-SPED
    - Flash-MP
    - Flash-MT

Performance Evaluation 3

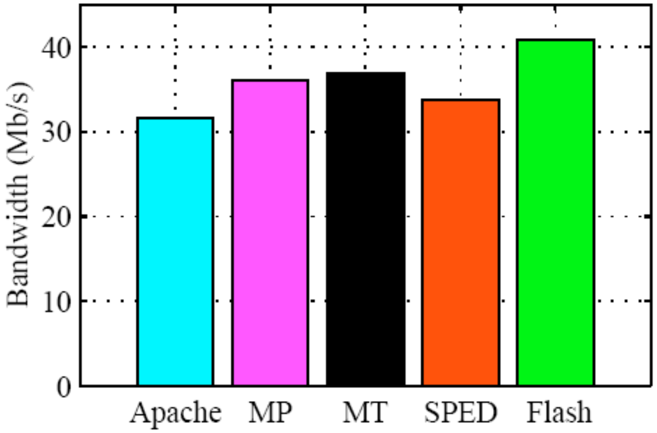
Solaris:
single file test



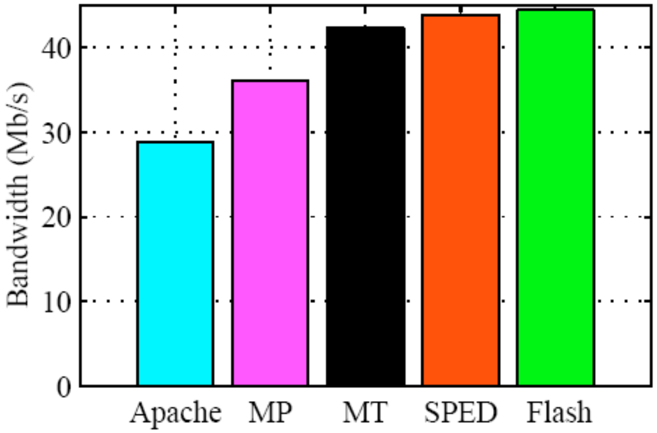


Performance Evaluation 4

Solaris: Rice Server Traces



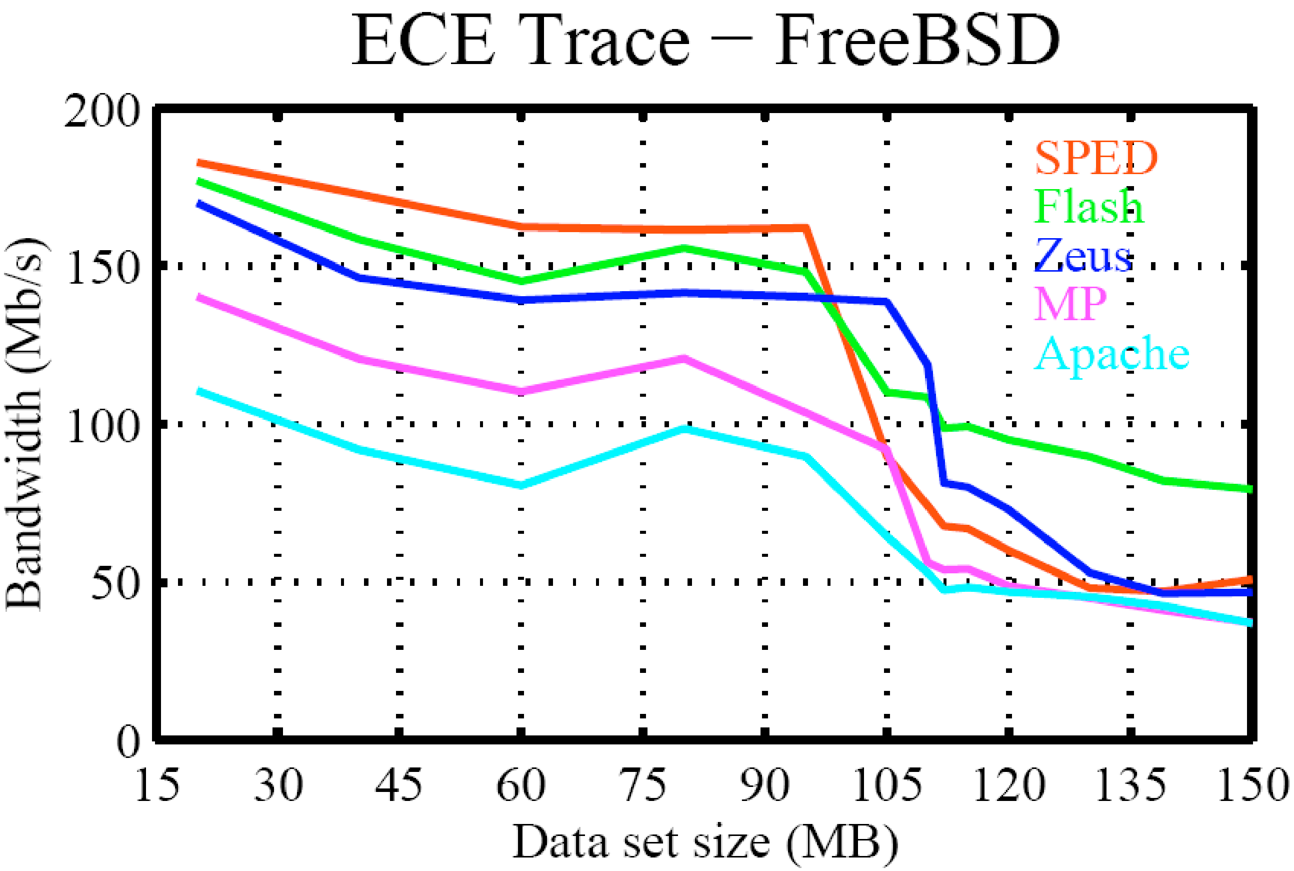
CS trace



Owlnet trace

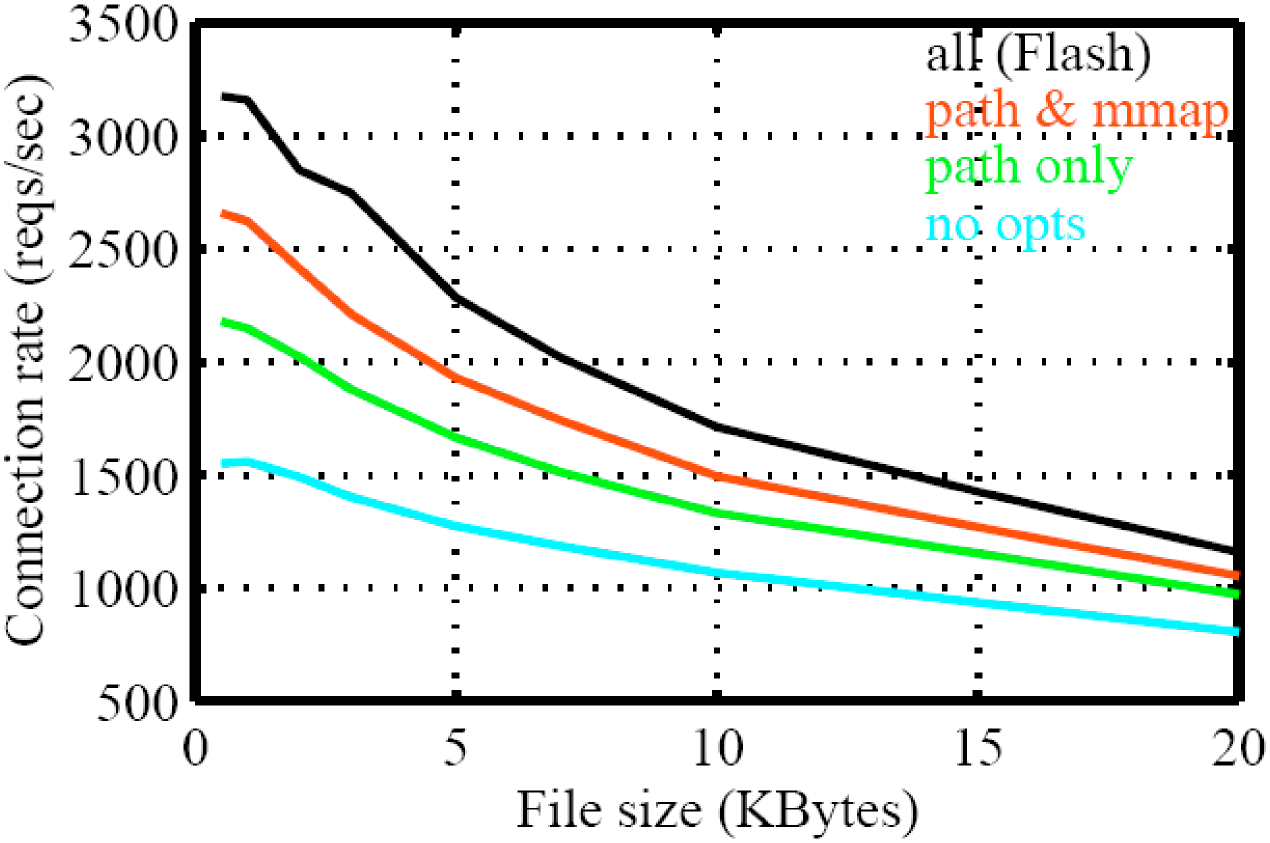
Performance Evaluation 5

FreeBSD:
Real
workload



Performance Evaluation 6

Optimization contributions



Conclusion

- Goals achieved?
- Good performance on real workloads

  → up to 30% faster than Zeus,

  → up to 50% faster than Apache.

Thank you for your attention!