Flash: An Efficient and Portable Web Server

Vivek S. Pai‡
Peter Druschel†
Willy Zwaenepoel†

Rice University

‡ Department of Electrical and Computer Engineering
† Department of Computer Science

June 10, 1999

Why Another Server?

Goals

- High throughput
- Good portability
- Wide range of workloads

But Also...

Understand server application performance

- Examine optimizations
- Compare concurrency architectures
- Quantify architecture vs. implementation

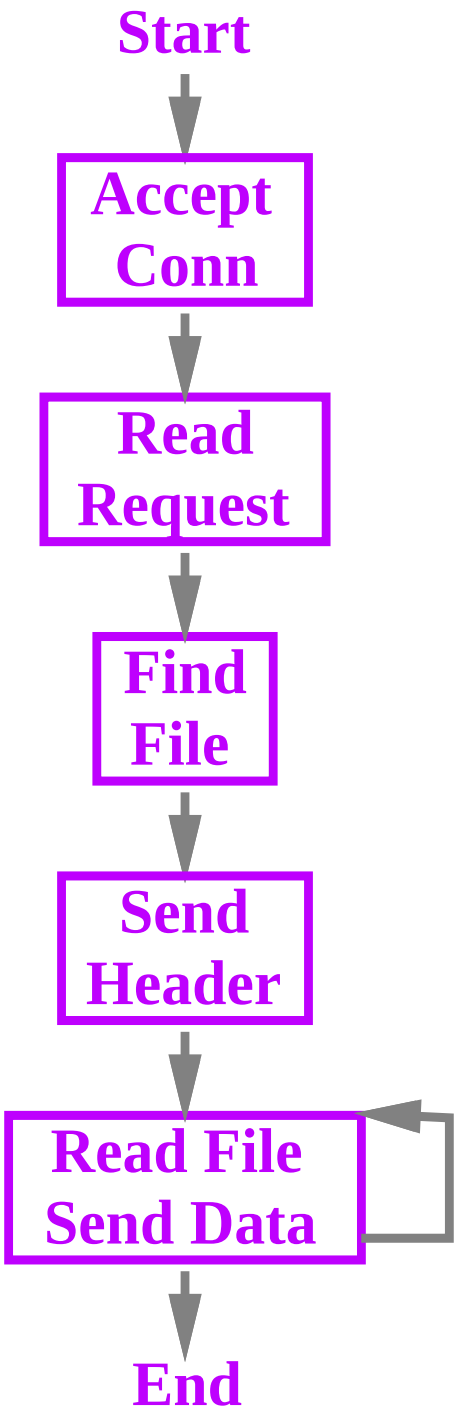
Overview

New server architecture

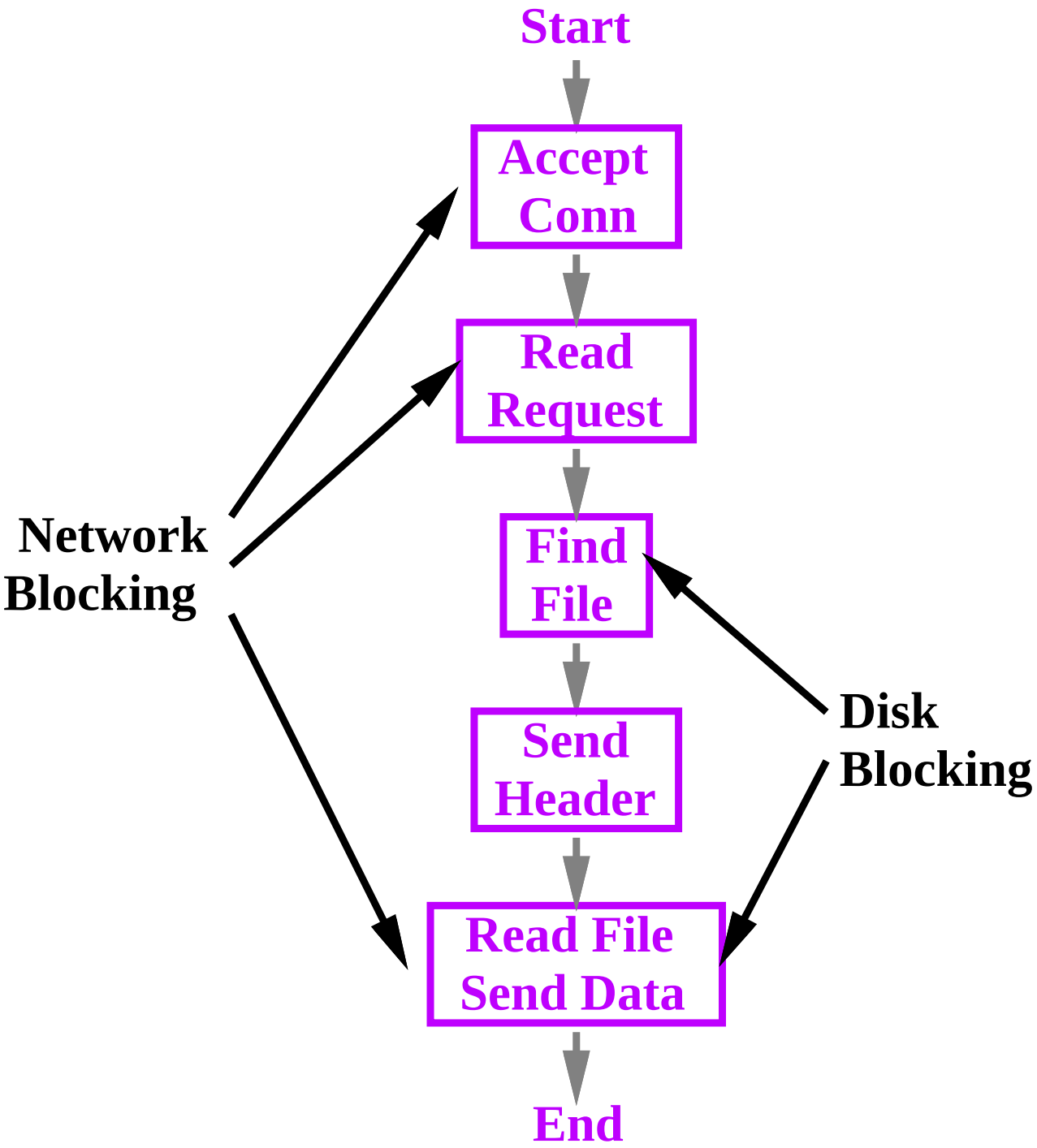
Implementation (Flash)
- aggressive optimizations

Experiments
- microbenchmarks, traces
- apples-to-apples comparisons

Processing Steps



Blocking Steps



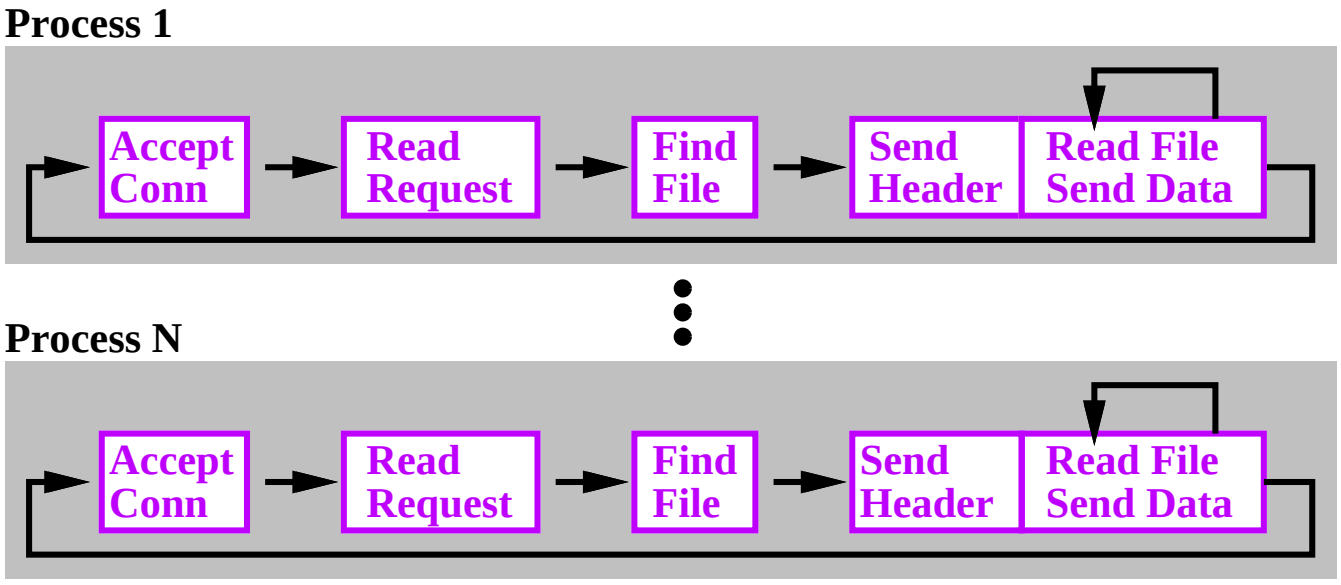
Concurrency Architecture

Overlap disk, network, & application-level processing

Architecture ⇒ how steps are overlapped

note: implications for performance

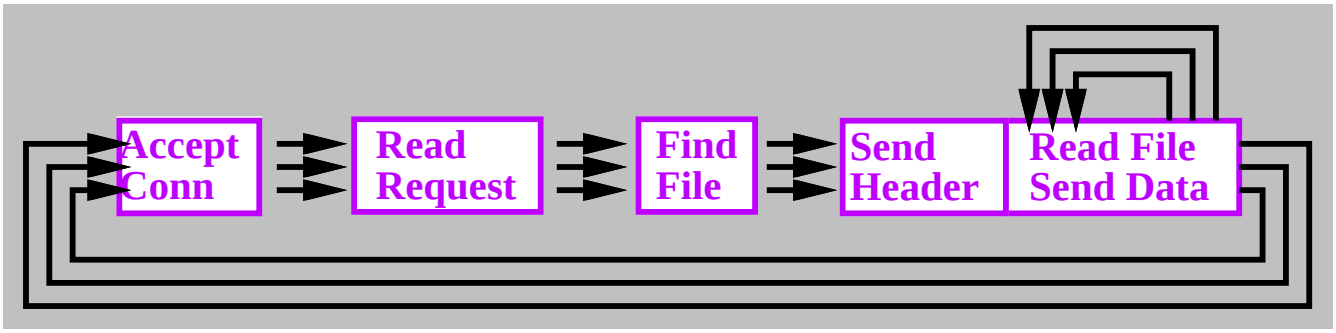
Multiple Processes (MP)



Pro: simple programming - rely on OS

Cons: too many processes
caching harder

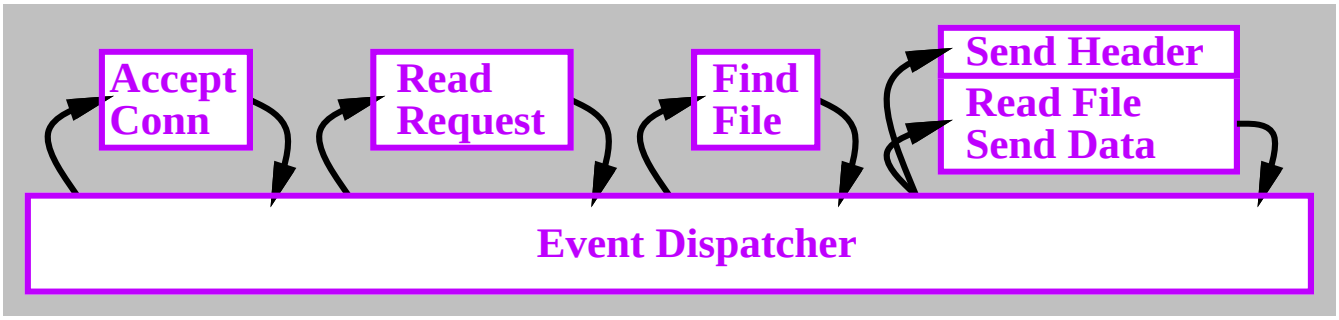
Multi-Threaded (MT)



Pros: shared address space
lower "context switch" overhead

Cons: many threads
requires kernel thread support
synchronization needed

Single Process Event Driven (SPED)



Pros: single address space
no synchronization

Con: in practice, disk reads still block

Disk Access Options

Non-blocking file descriptors

- Blocks on most Unix systems
- No metadata support (open, stat)
- Does not work with mmap()

Async interfaces

- Not well-integrated
- Portability
- Metadata support? (open, stat)
- Interoperation with mmap()

Desired Properties

Shared (single) address space

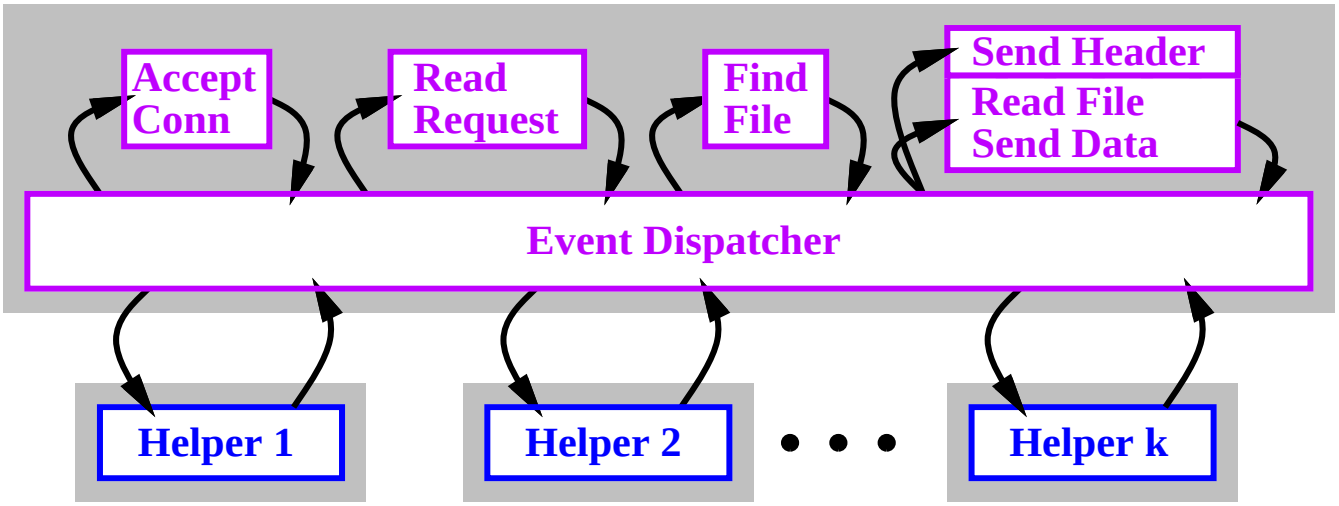
Good disk behavior

No synchronization

Low resource requirements

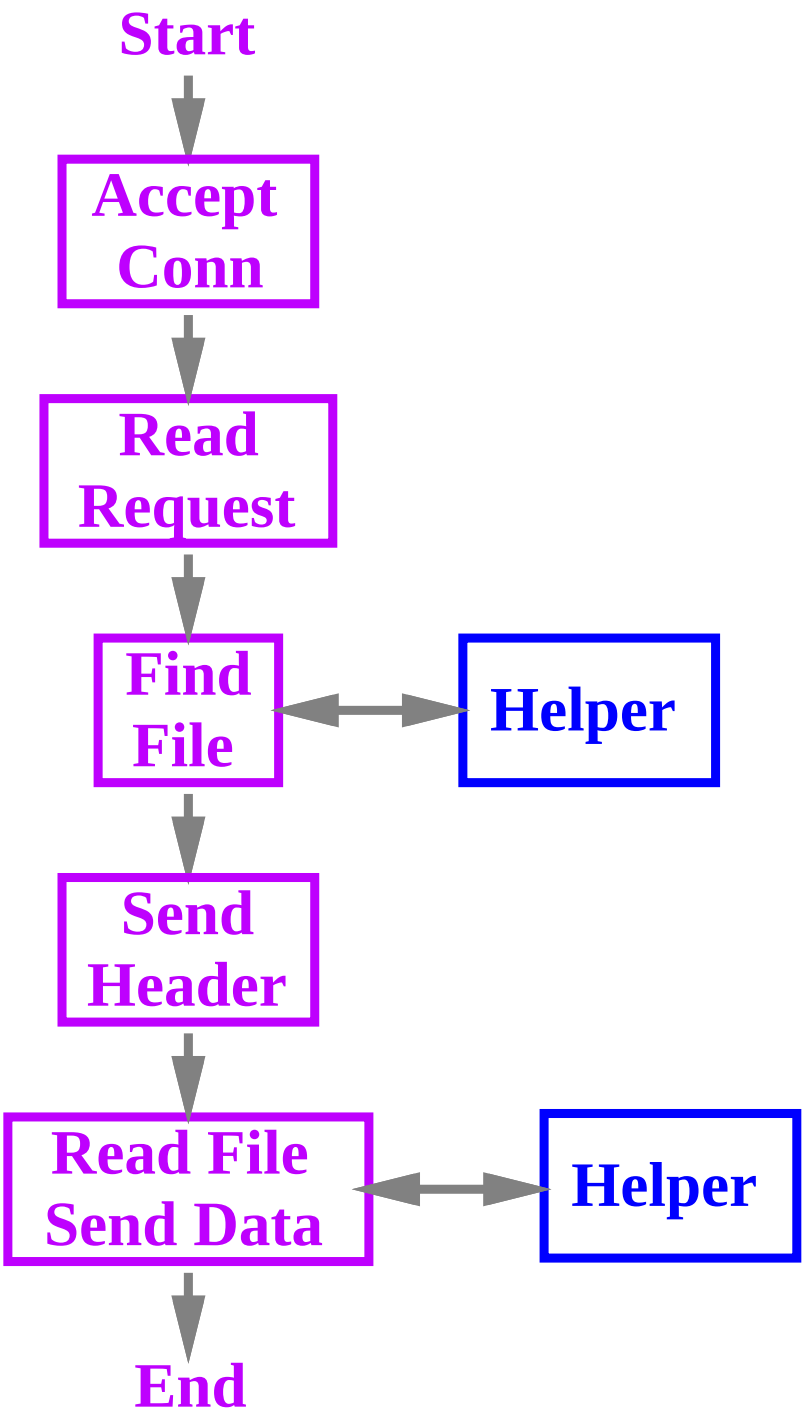
New Architecture - AMPED

Asymmetric Multiple Process Event Driven



Helpers are threads or processes

AMPED Steps


Start
Accept Conn
Read Request
Find File
Helper
Send Header
Read File Send Data
Helper
End

Performance Expectations

Main server process
- good in-core behavior

Helper processes
- good disk-bound behavior

Low overheads
- memory residency checking, IPC

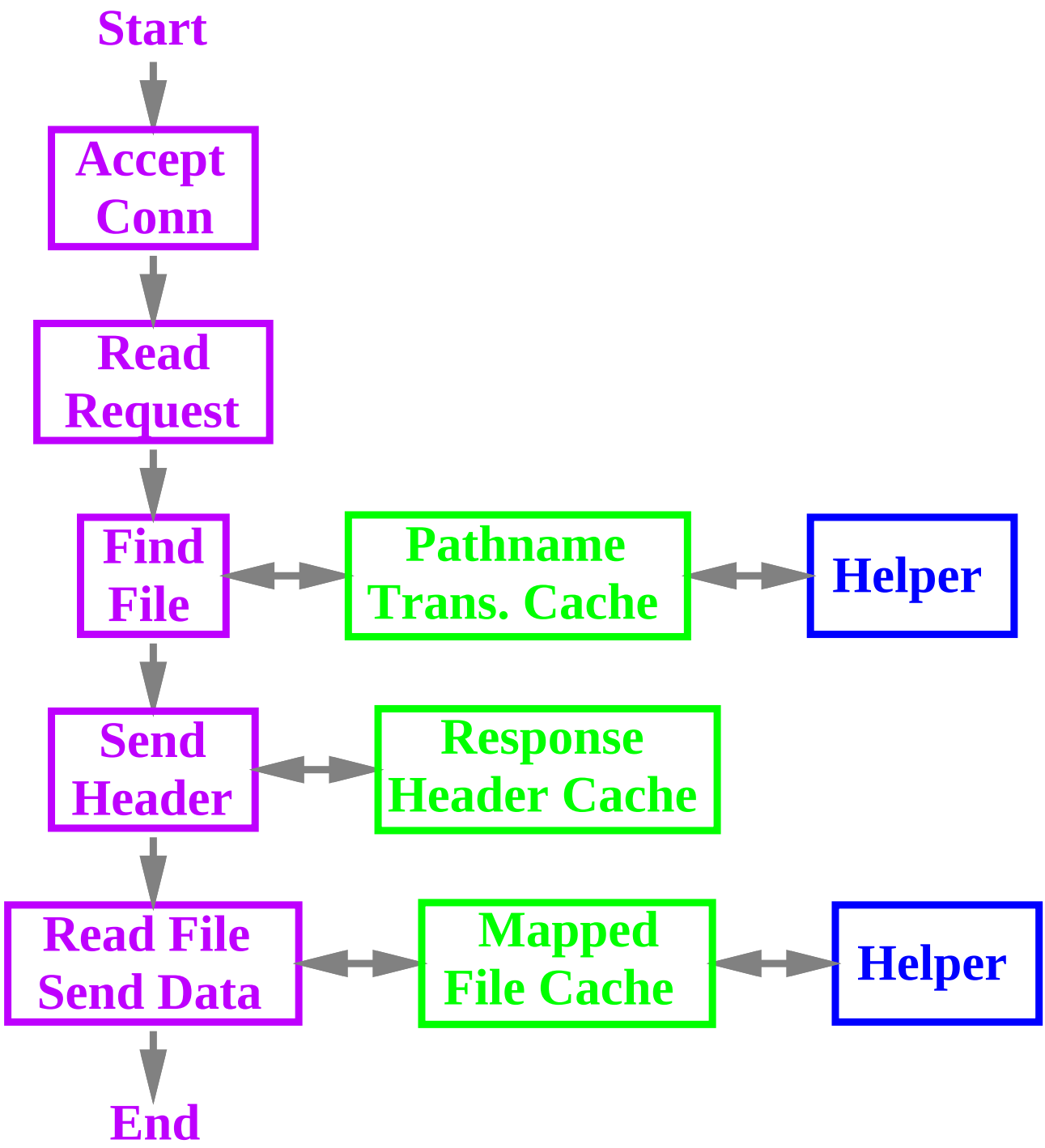
Implementation: Flash

AMPED architecture

Various optimizations

- Memory-mapped files
- Gather writes (writev)
- Aggressive application-level caching

Caches in Flash



Test Environment

Machine

- 333 MHz Pentium II CPU
- 128 MB memory
- Five 100 Mb/s Fast Ethernet cards

Operating systems

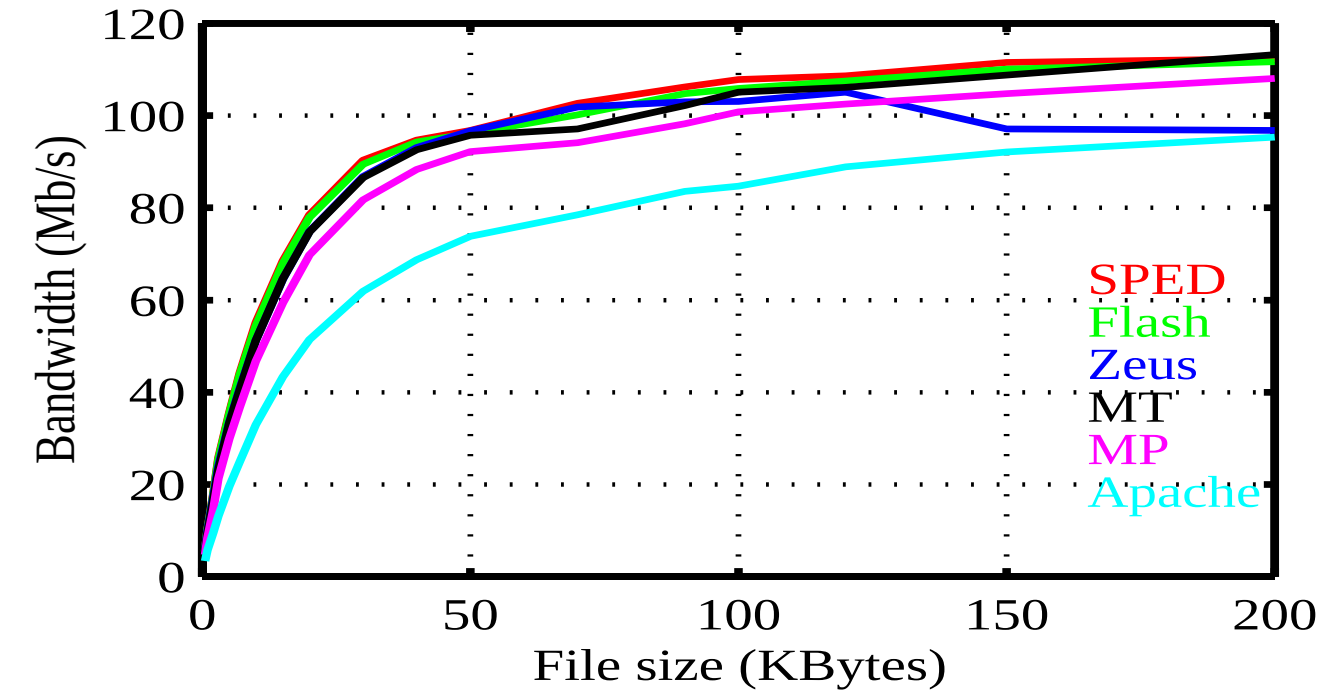
- FreeBSD 2.2.6
- Solaris/x86 2.6

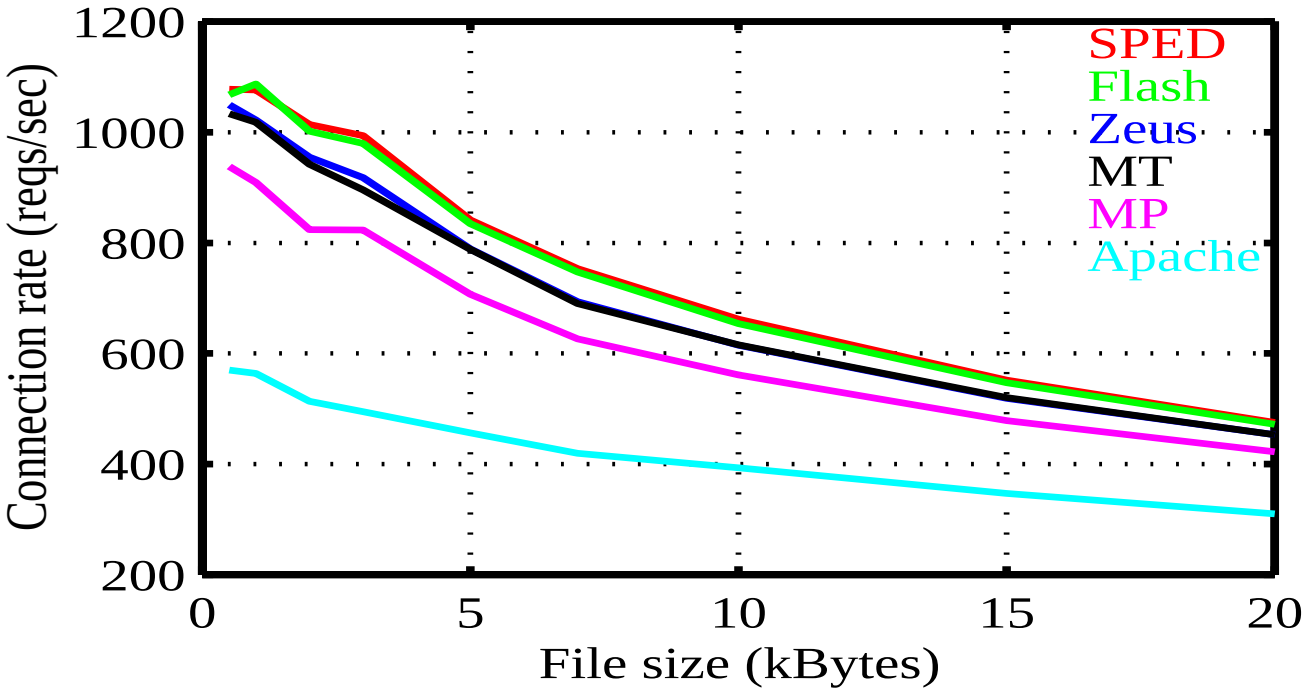
Server Software

Six server applications total

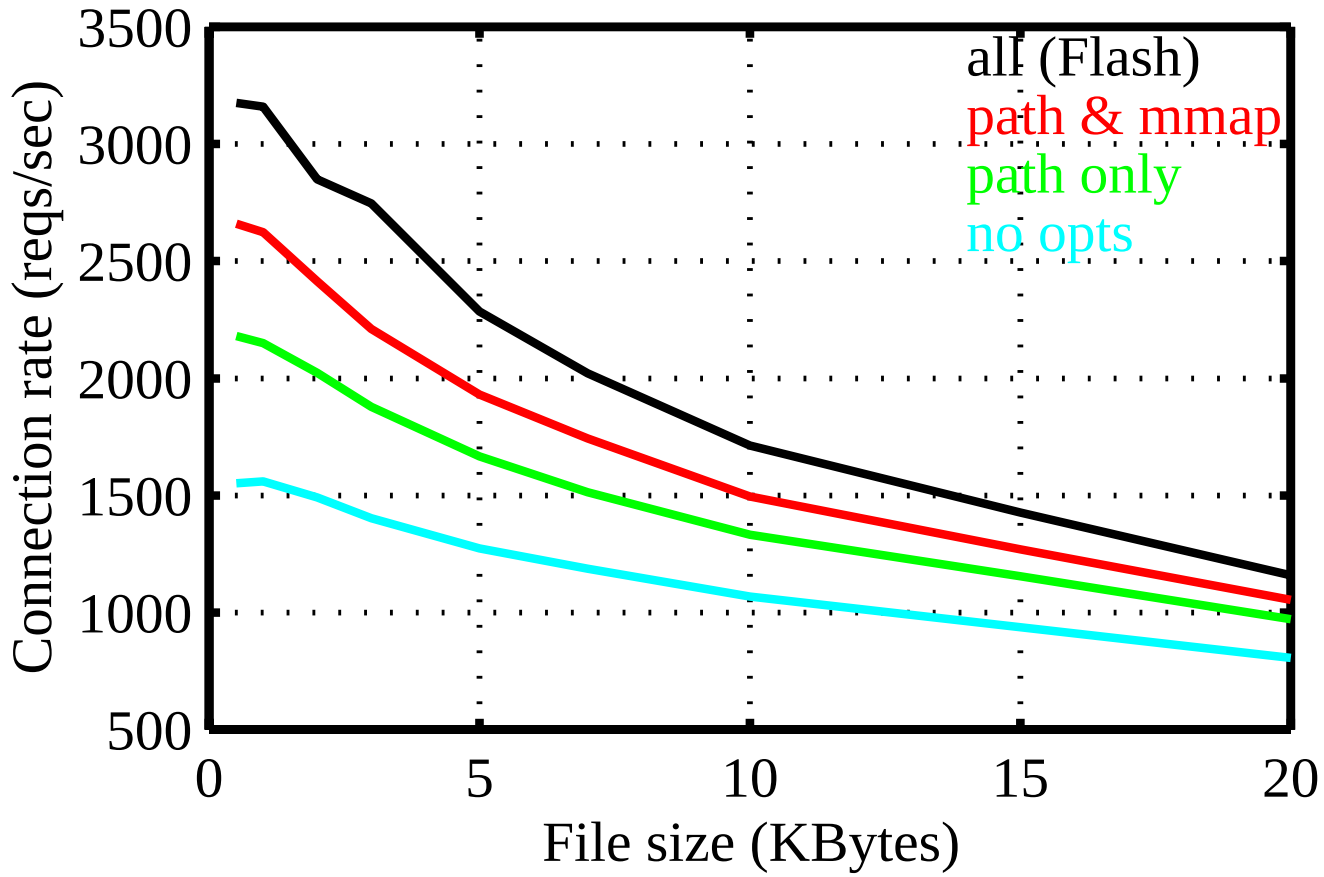
- Apache 1.3.1 (MP)
- Zeus 1.30 (SPED)
- Flash (AMPED)
- Flash-SPED
- Flash-MP
- Flash-MT

Single File - Solaris

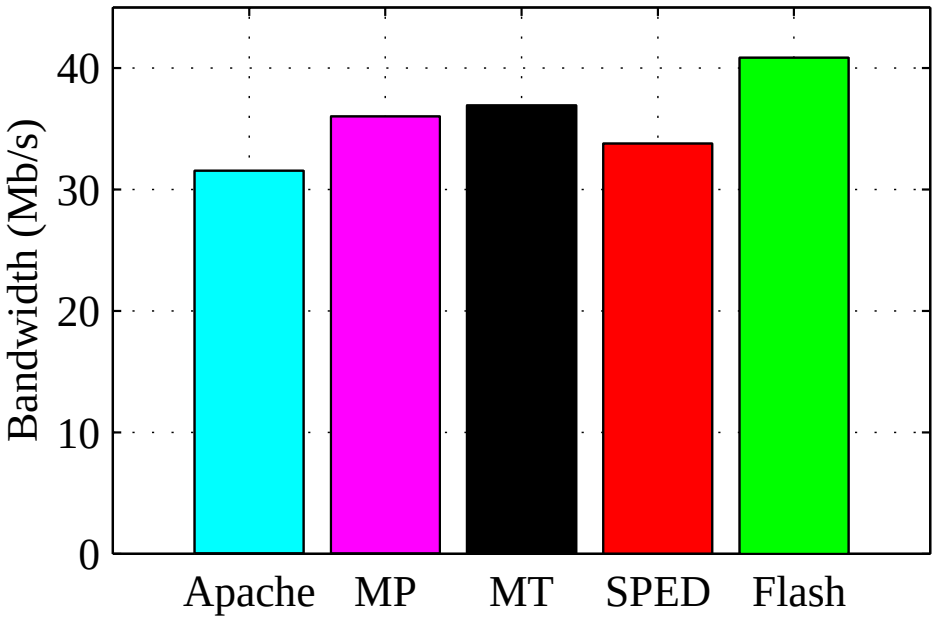




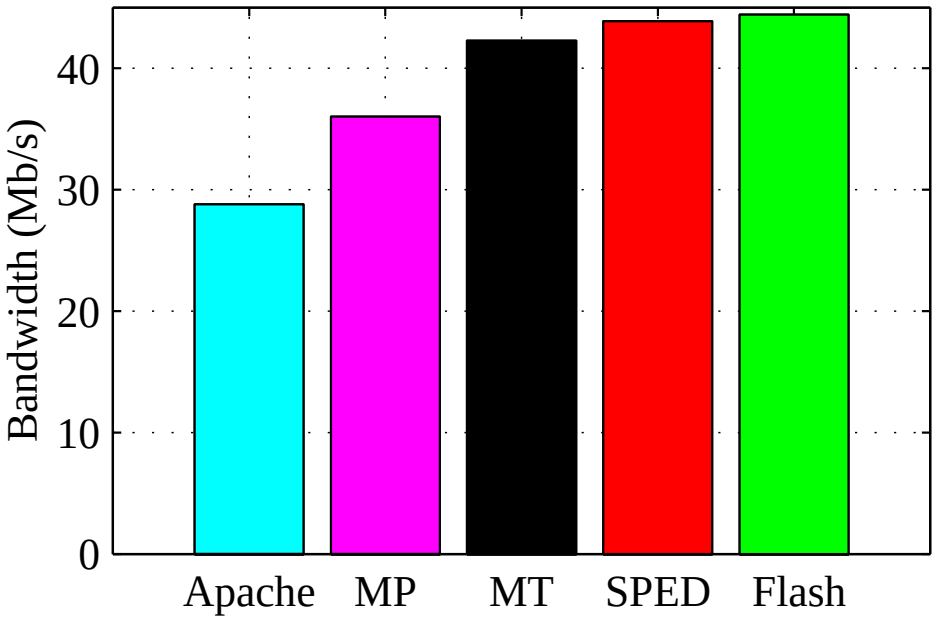
Optimization Contributions



Trace-based Tests



CS trace



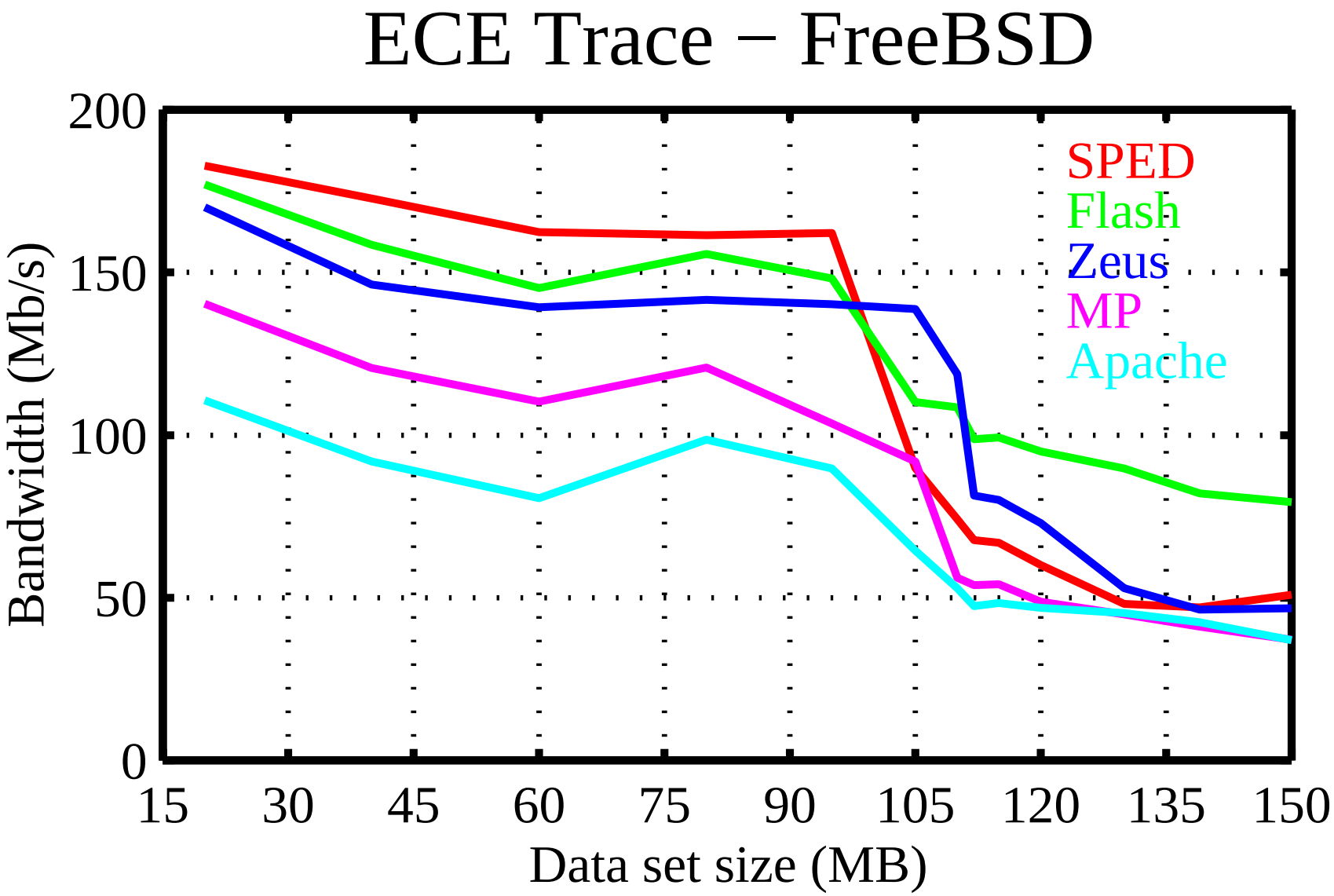
Owlnet trace

Detailed Trace Tests

ECE department access log

Truncate to vary data set size

Clients replay sub-traces in loop


ECE Trace − FreeBSD
Bandwidth (Mb/s)
200
150
100
50
0
15
30
45
60
75
90
105
120
135
150
Data set size (MB)
SPED
Flash
Zeus
MP
Apache

WAN/P-HTTP Effects

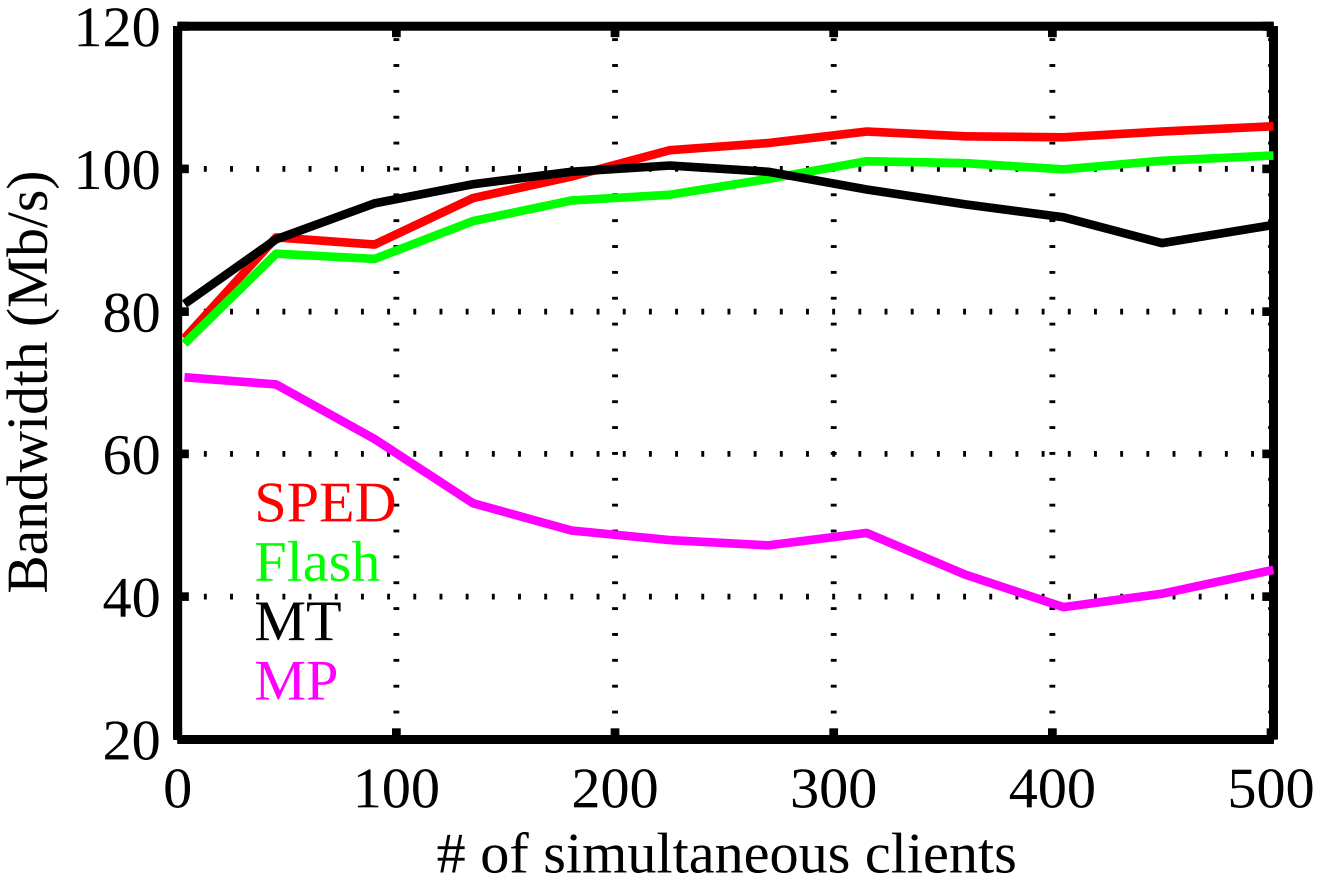
Longer client connections

More simultaneous connections

Per-client overhead an issue

Adding Clients

ECE Trace



Summary

Concurrency architecture
- affects real workloads
- SPED, MT models good for in-core

AMPED architecture

Flash server implementation

Good performance on real workloads
- up to 30% faster than Zeus
- up to 50% faster than Apache