Flash: An Efficient and Portable Web Server

Presentation by Nikolaos Giannarakis

Vivek S. Pai, Peter Druschel, Willy Zwaenepoel

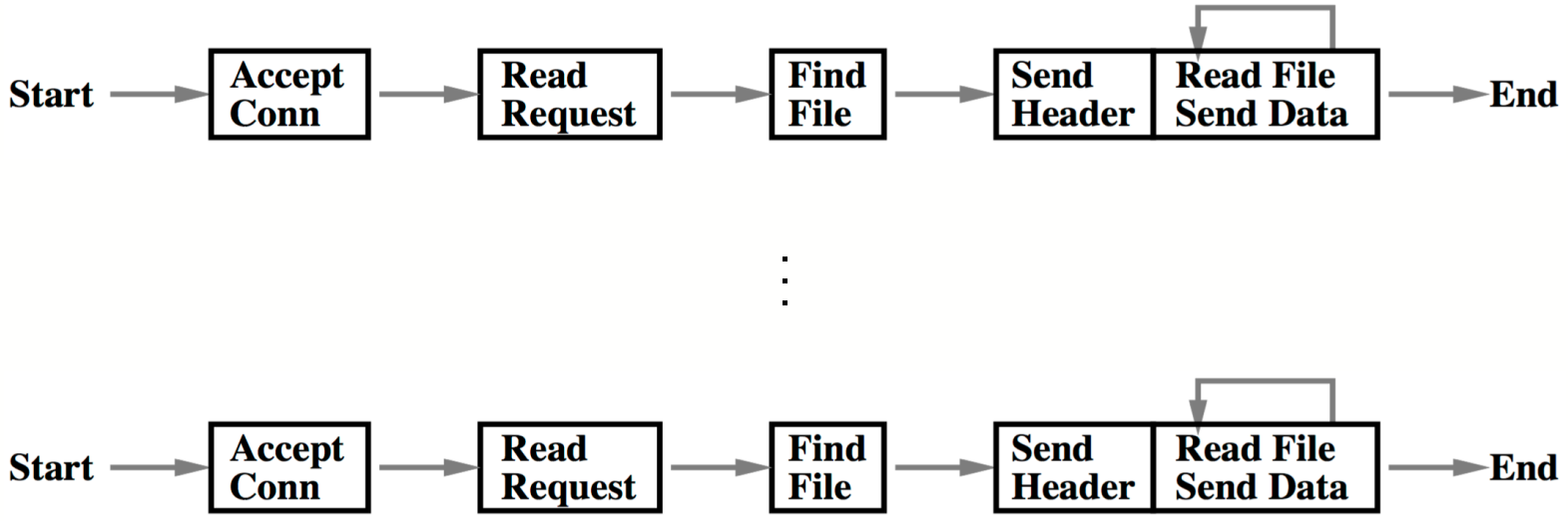
Web Servers

- Fulfill client (e.g. web browser) request for web content.
- To achieve high throughput web servers rely on caching popular content.
- If the requested content is not found in the cache then the server must fetch it from the disk.
- ***Key idea:*** Overlap fetching (and other blocking actions) with serving requests for cached content.

Handling a request

- Each task involves either network communication or disk I/O and hence can block.
- Server must interleave the tasks of many requests to improve performance.
- Different ways to achieve interleaving of tasks.

Multi-process architecture

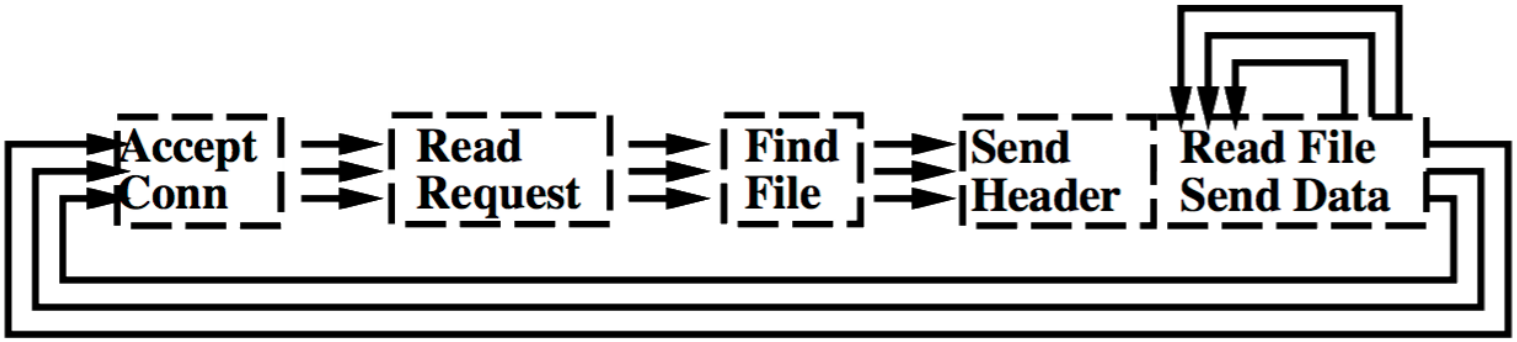


- Each process handles the tasks of a request sequentially.
- Multiple processes are used to serve many requests concurrently.

+ Interleaving in this model occurs naturally, as the OS will context switch when a process blocks.

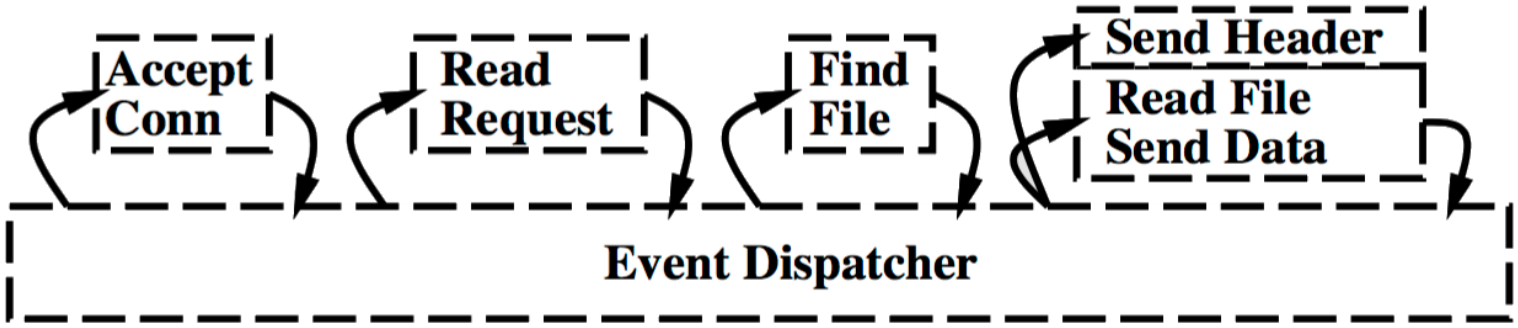
– High overhead from multiple processes, less aggressive optimizations.

Multi-threaded architecture



- Multiple threads are used to serve many requests concurrently.
- \+ Less overhead compared to MP.
- \+ Shared address space allows more aggressive optimizations.
- − Requires synchronization for accessing shared data.
- − Only feasible with OS that supports kernel threads.

Single-process event-driven architecture



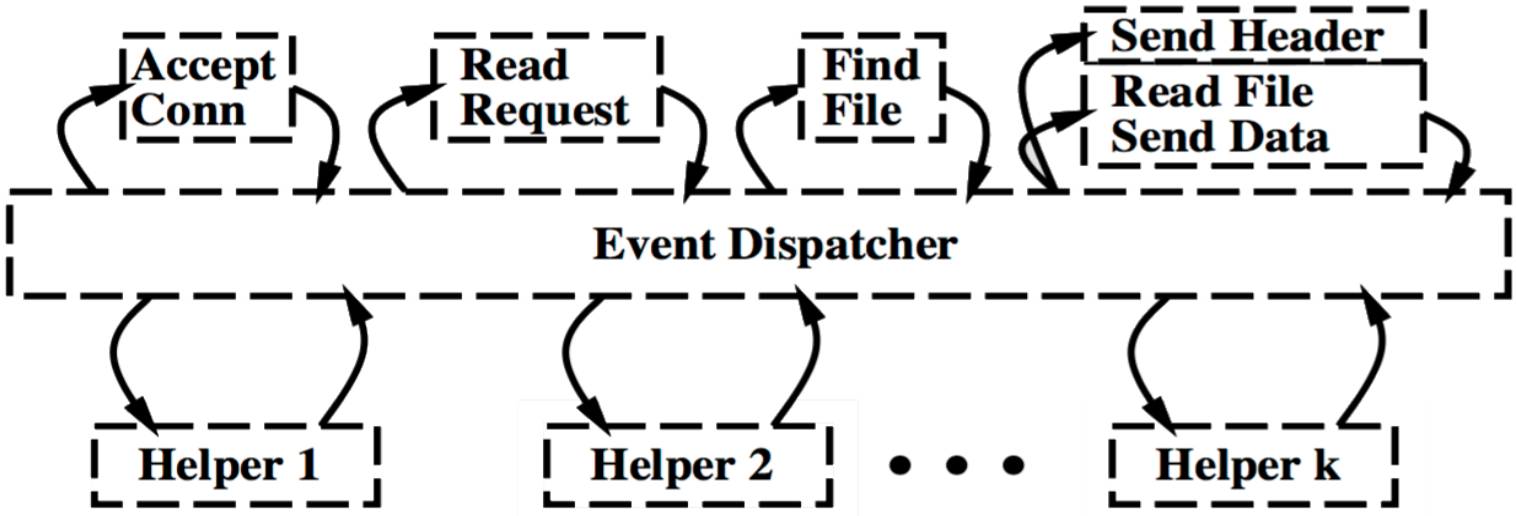
- Single process, interleaving of tasks is achieved using non-blocking I/O.
- Issues I/O and proceeds to next request. Uses polling to check for completion of I/O.

\+ Minimal overhead, no context switches or synchronization required.

– Missing abstraction of sequential execution of the tasks, control flow and reasoning become more complicated.

– In practice I/O operations may block.

Asymmetric multi-process event-driven architecture



- Combines the previous approaches.
- A single process runs as an event loop but delegates blocking actions to helper threads/processes.
- \+ Less overhead compared to MP/MT, performance of single process on cached requests.
- \+ Admits the same optimizations as SPED.
- – Only provides I/O concurrency, not CPU concurrency.

Design comparison

Disk Blocking

- The cost of disk operations differ between the architectures depending on whether they can cause the system to block.
- In MP, MT and AMPED only one request gets blocked, the system can serve other requests.
- In SPED if one task blocks the whole server blocks as well.

Memory Consumption

- The memory consumption of the server is important because the server caches requests. The more memory, the more requests it can cache.
- SPED has very low memory consumption. It doesn't require any memory to keep track of children processes/threads.
- MT has some additional cost due to per-request threads.
- AMPED also incurs some overhead, but notice that AMPED's helpers are per concurrent I/O task and not per request.
- MP has the highest memory overhead due to per-request processes.

Design comparison

Disk Utilization

- Multiple disks may improve performance if the server can generate multiple I/O operations concurrently.
- SPED can only generate one I/O operation hence it cannot benefit from multiple disks (assumes it blocks?).
- MT, MP and AMPED can generate concurrent I/O operations.

Design comparison

Information Gathering

- Servers perform profiling to uncover possibilities for optimizations.
- Trivial in SPED and AMPED because of single thread.
- MP and MT require communication/synchronization.

Application-level Caching

- Caching of frequently accessed data (request responses, file mappings, etc.)
- Single cache for AMPED and SPED.

Flash: AMPED implementation

- Implements helpers as processes for portability.
- Three types of application-level caching (filename translations, response headers, file mappings)
- Smart trick to keep data aligned by padding response headers accordingly.

Performance Evaluation

Comparing architectures

- Compare four servers based on Flash: Flash, Flash-MT, Flash-MP, Flash-SPED.

Comparing web servers

- Compare Flash and two state-of-the-art web servers Apache and Zeus.

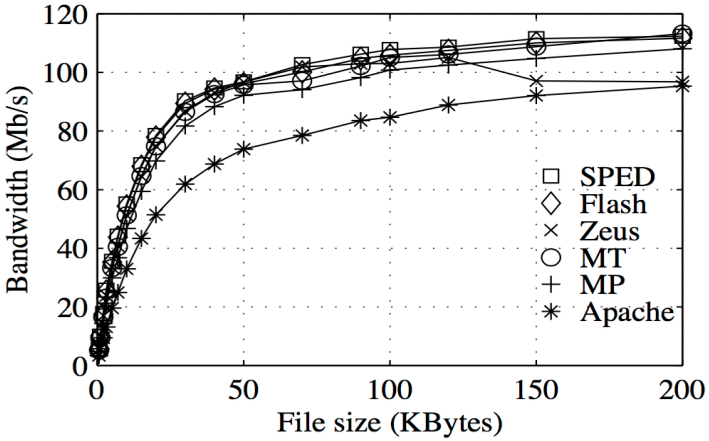
Evaluating optimizations

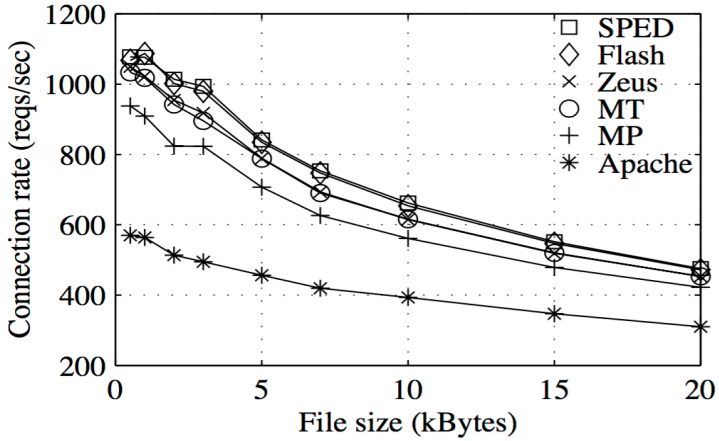
- Evaluate the impact of the various optimizations implemented in Flash.

Evaluation environment

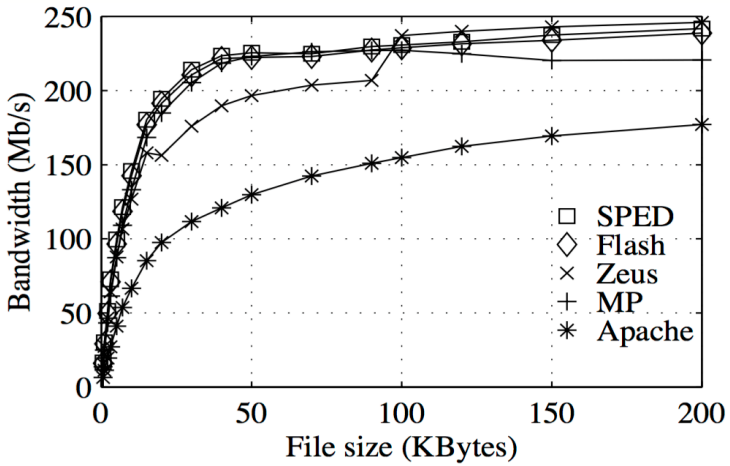
- Same hardware (Pentium II, 128MB RAM).
- Two different operating systems (Solari, FreeBSD).
- 100Mbit/s ethernet connections.

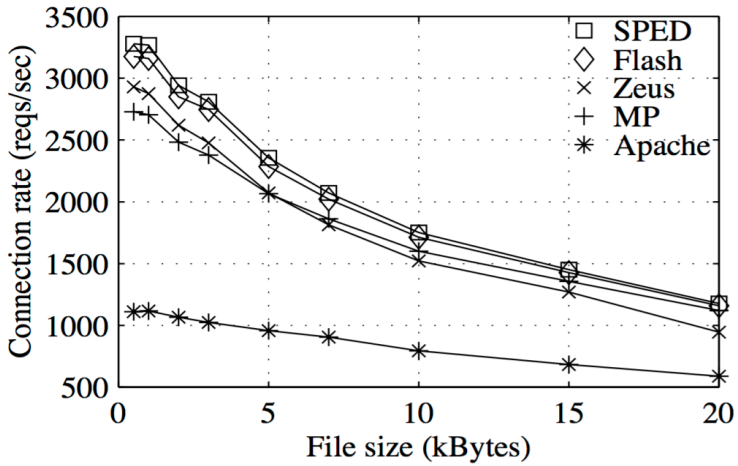
Synthetic workload





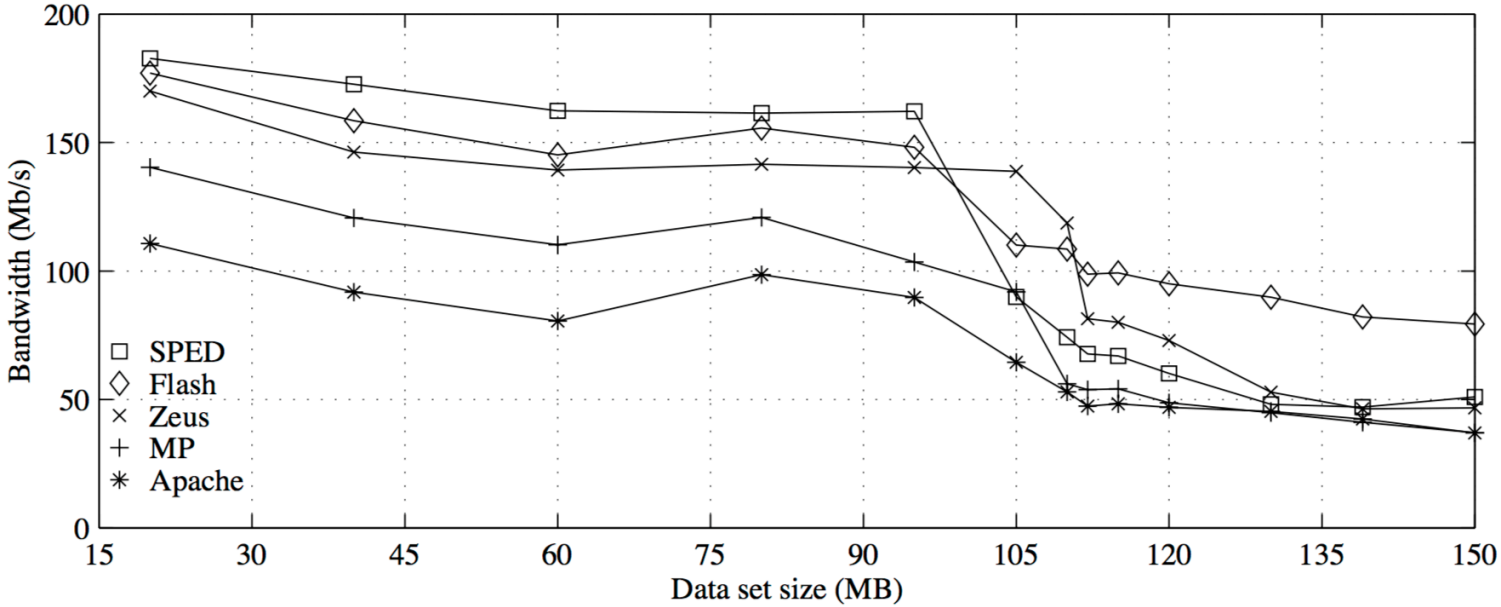
Cache-bound workload on Solaris





Same workload on FreeBSD

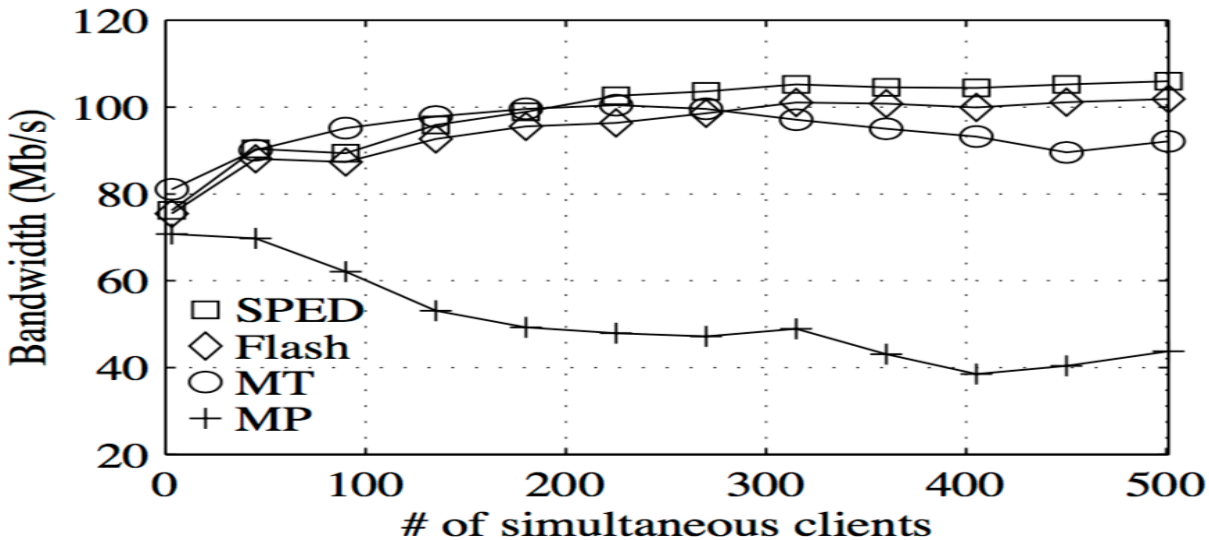
Real workload



Real workload on FreeBSD

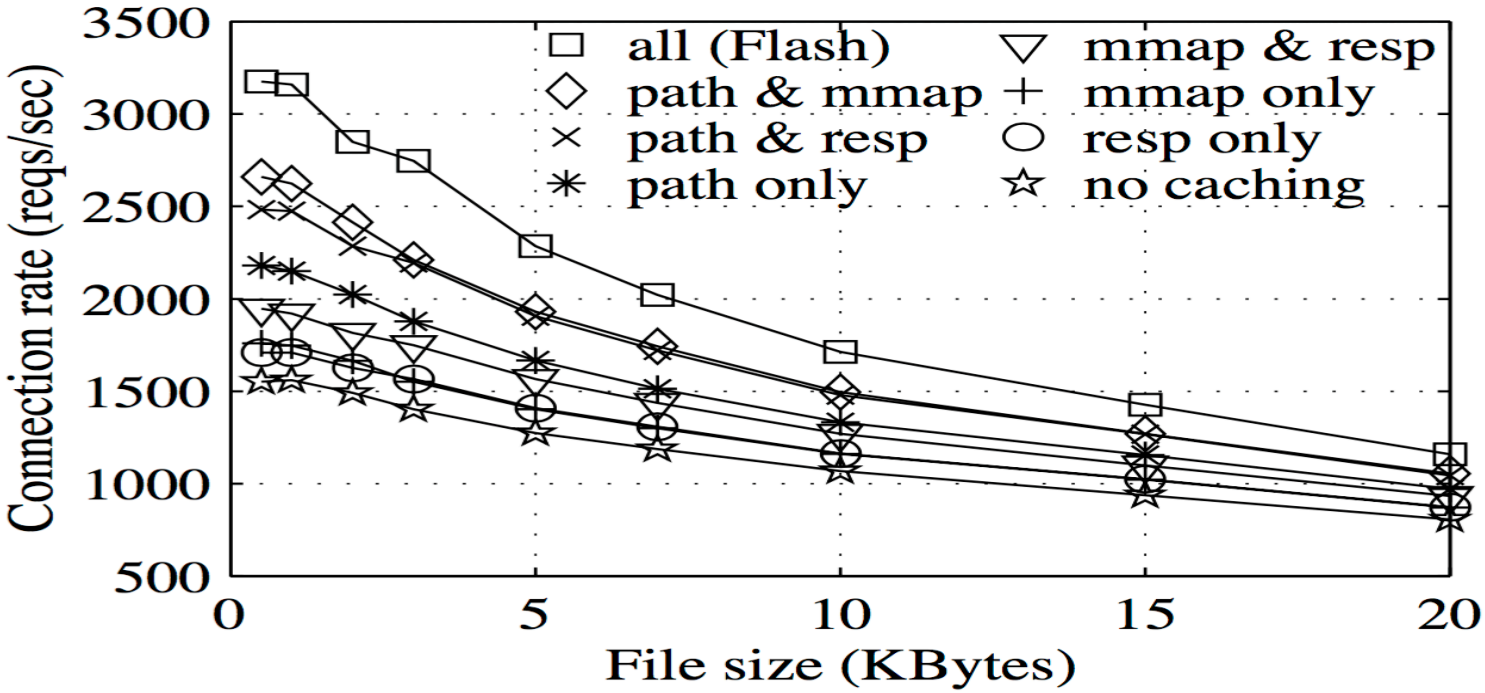
Evaluating in a WAN

- Previous evaluations were done in a LAN setting.
- This does not accurately model the increased connection times due to packet losses and limited bandwidth that occur in a WAN.
- Idea: Use persistent connections



Impact of concurrent HTTP connections

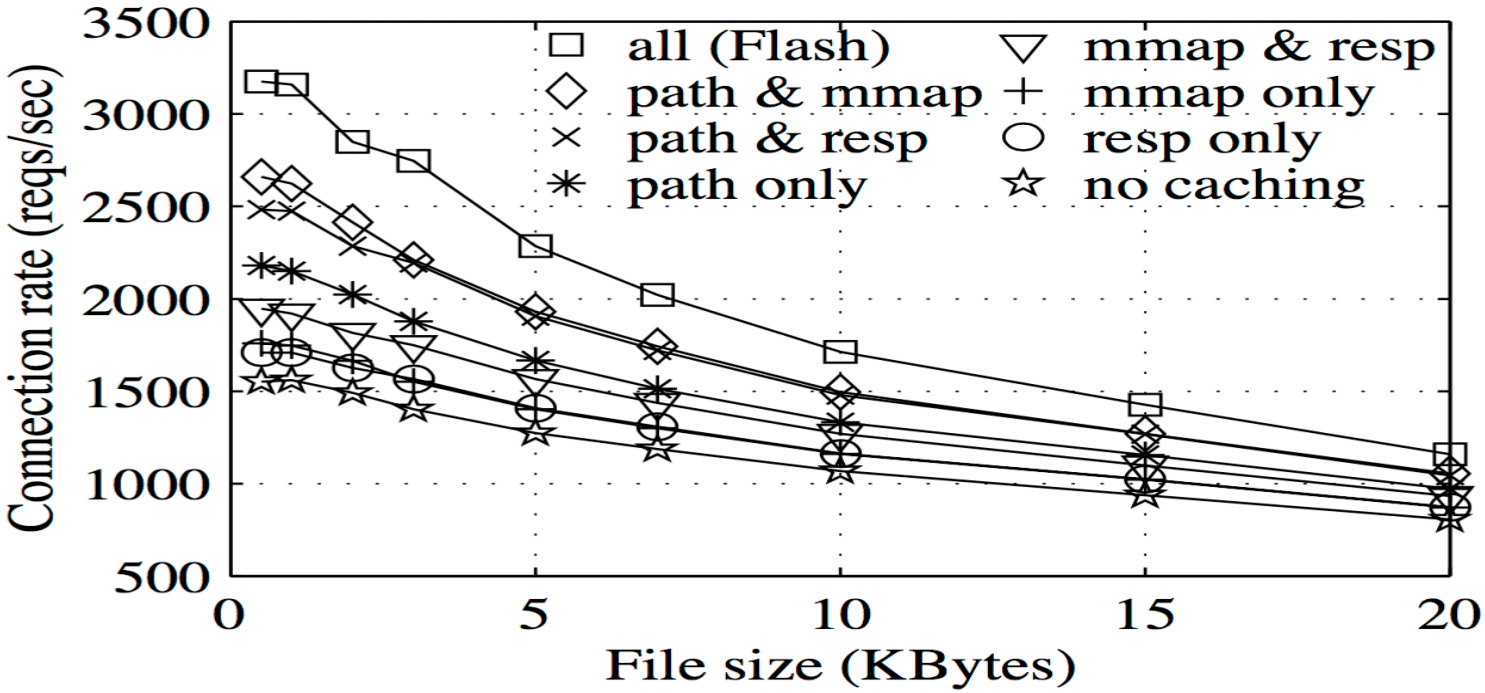
Optimizations impact



Impact of optimizations

- Each optimization (or cache hit) avoids one request.
- Performance may double thanks to optimizations.

Optimizations impact



Impact of optimizations

- Each optimization (or cache hit) avoid one request.
- Performance may double thanks to optimizations.

Conclusions

- Choice of server architecture is really important for performance.
- The evaluation proved that the choice of OS is/was important.
- Flash matches the performance of SPED architectures on cache-bound workloads and exceeds the performance of MP/MT architectures on disk-bound workloads.
  - Flash exceeds the performance of Zeus and Apache by a good margin.
- Only focuses on I/O concurrency.