"Flash...", by Pai, Druschel, and Zwaenepoel
USENIX 1999

What are we doing here?
  case studies
  examples of research papers, learn how to write them
  understand content
  dig deeper into details than any one of us could alone
  put work into larger context - does it matter? is it useful?

Discussion format
  I'll start out by asking questions
  Feel free to interrupt me or ask your own questions
    Or to answer each others questions, or discuss each others points
  If you don't understand, ask!
  Feel free to be critical (of me or paper), or to defend

Why we're reading this paper:
  to talk about server structure
  to talk about performance evaluation

What are the paper's main claims?
  new web server architecture, AMPED
  as good as it gets for cached workloads
  as good as it gets for disk-bound workloads

Does anyone care about web server efficiency?
  You do if you have thousands of servers.

What is the basic AMPED idea?
  [draw picture w/ helpers]

What operations does the helper perform?
  disk read()
  how about open()? stat()?

Why is cached vs disk a big issue?

Where might the cache be?
  O/S disk block cache, or
  Maintained by web server[s]
  Also servers cache results of file lookup, file sizes

What's a reasonable number of helpers?
  one or two per disk?
  many per disk for disk-arm scheduling?

Other techniques they discuss:
  MP
  MT (user threads? kernel threads?)
  SPED (event-driven, like 2nd lab)
  Apache == MP
  Zeus == SPED

What performance do we expect?
  Disk-bound: AMPED > MT > MP/Apache >> SPED/Zeus
  Cacheable: SPED/AMPED/Zeus > MT > MP/Apache.

```
Questions we'd like performance section to answer:
  Does AMPED win for both cached and non-cached?
  Are process/thread context switch costs significant?
  Are benefits of single address space significant?
  Are benefits of disk I/O concurrency significant?

What's the test setup?
  Real server and lots of clients. How many clients? Is one enough?
  Clients run fake web browsers that issue concurrent requests.

Figure 6:

Why does b/w go up with file size?

What's the limiting factor for small files?
  Disk? Net? RAM? I/O bus? CPU?
  Client's ability to generate requests?

What's the limiting factor for large files?

Why does the curve have the shape it does?
  x = file size
  a = time to process zero-length request
  b = big-file serving rate, bytes-per-second
  y = bytes/time = x / (a + x/b)

What are a and b?
  Figure 6(b) suggests a is about 1 millisecond.
  Figure 6(a) suggests b is about 100 mbits/second.

Why is there no MT line in Figure 7?

Why is FreeBSD faster than Solaris?
  Same hardware...
  Solaris is a commercial O/S, you'd expect it to be faster?

Why does the paper present Figures 6 and 7?
  Is the workload realistic? no. only one file, no disk...
  What have we learned?
  Apache is slow.

What would we still like to learn about?
  Disk-bound performance.
  "Realistic" performance with typical mix of big/small, cached/disk.
  Effect of various parameters (mem size, # of processes, &c)

Why don't they show us a simple disk-only graph like Figure 6?
  How could we force *all* requests to use the disk?
  Would we want to force all path name lookups to use disk too?
  What would we learn from true disk-bound experiment?
    Probably all servers the same, we'd learn # disk reads / request.
  Best we can do is serve enough files that they don't fit in cache.
  Thus mixed cache/disk workload.

Why is performance only 40 mbits in Figure 8, was ca. 100 in Figure 6?
  avg file size apparently 10 kBytes. or 80 kilobits.
```

Figure 8, why is SPED < MT for CS, but SPED > MT for Owlnet?

What can we conclude from Figure 8?
  Realistic traces. Flash is a bit faster, but not radically.
  Presumably this is a mix of cached/disk requests.
  But actual mix is not known, so we don't really know what we're testing.

How do figures 9 and 10 shed light on cached/disk performance?
  by varying data set size, control how well data fits in ~100 MB disk cache.

How do they vary the data set size?
  How does that affect cache vs disk?

Why is there a sudden decrease at around 100 mbytes in Figure 9?
  Why at 100 mbytes, not 50 mbytes or 200 mbytes?

Why is b/w around 50..100 mbits for large data set sizes?
  How many requests per second? 500 to 1000... assuming 80-kilobit files.
  Is this workload diskbound?
  What would b/w be if diskbound? 8 mbits/second...
  What's the cache hit rate?
    Hit rate must be around 90%
    That is, 10 files served per 10 millisecond disk seek (i.e. per miss)
    Or 1000 files per second
  Do they in fact ever evaluate disk-bound behavior?

Figure 9/10, Flash vs MP.
  Why does Flash beat MP for small data set?
  Why does Flash beat MP for large data set?

Flash vs SPED
  Why are Flash and SPED close for small data set?
  Why does Flash beat SPED for large data set?

Flash vs MT (Figure 10)
  Flash and MT have about the same behavior for all data set sizes.
  Why?
  What does this mean w.r.t. whether Flash is worthwhile?

At right of Figure 9, why is MP < SPED?
  we expect MP to get more I/O concurrency with disk-heavy workload
  so maybe user-level cache is small in MP?

Figure 12, why do most lines go up w/ # clients?
  Why does MP line go down a lot, starting at 50?
    Context switch costs? Partitioned 4MB caches?
  Why does MT line go down a little?

Cynical view:
  Should just use MT, not Flash.

Practical view:

Flash far easier to implement then kernel-supported threads!
Much better use of programmer time.