General thoughts:
- - - - - - - - - - - - - - - - -

A good way to read papers is to think about the following things and what this paper does for these things:

.a. What problem are they solving (pretty obvious ;)

.b. At what level is the solution being proposed (compiler, OS, hardware, ether)
- -- Chip level integration
- -- software restructuring
- -- magic disks
- -- OS scheduling changes

.c. How is the evaluation being carried out (SPEC?, OLTP? nothing close to real?)
- -- Real Workloads (Apache, Zeus, JAWS) vs. Benchmarks (TPC-X, SURGE, SPLASH)

.d. How does this solution extend/affect other workloads (what might work for www.winamp.com may not work for www.bankone.com)

- -

Questions posed in class:
- -

.a. What attributes of server workloads or just workloads do we care about?
- -- Concurrency/Independence of requests
- -- State
- -- I/O activity
- -- programmability (threads please!)
- -- Code size and complexity
- -- Games
  - o The economics of game consoles (totally unrelated rattle)
- -- Vertical MT vs. SMT
- -- Characteristics of Multimedia applications

.b. Cohort scheduling [hardly discussed!]
- -- Background:
  - o Continuations
  - o Anderson's Scheduler Activations
  - o SEDA (a must read poem)
  - o Ousterhout -- why threads suck.

- -- Motivation
  - o Utilization of resources
  - o Locality (the less-frequent case)
  - o Threads are high overhead (Ousterhout would agree)

- -- Overview
  - o Two big components - stage
    - cohort

  - o What is the paper all about?
    - -- A new way to structure applications
    - -- Instead of threads, use stages
    - -- Two examples, one for I/O and one for computation
    - -- Present design patterns or logical *stages* that the application can be broken down into.
    - -- How generic is this? i.e, are design patterns part of all development processes? i.e, how many man years for figuring out stages in a web-server?
    - -- Software engineering, WRITE GOOD CODE!

  - o Good things
    - -- A new way to structure applications..
    - -- Deal with the uncommon case of cache/BTB locality as opposed to SEDA
    - -- Point out that s/w engineering is as hard as designing good chips

  - o Concerns
    - -- What about backward compatibility?
    - -- Threads are easier to program -- no doubt there?
    - -- Benchmark - SURGE. Cant remember why I thought this was a concern.
    - -- Cohort scheduling is not really the focus of this paper, stages are.

o Specifics
    -- Work in progress
    -- Come talk to me if you are dying to contribute