Sermon 2: Building High-Performance Systems

2.1 Options for performance tuning

1. Make every line of code as fast as possible?
   Takes a lot of time, maybe tuning stuff that isn't important

2. Tune selectively
   This is what I'm going to argue for.

2.2 Observations

1. Only a few places where performance matters
   90-10 rule: 90% of the time in 10% of the code

2. Difficult to predict performance problems in advance.
3. Tuning takes time and makes system more complicated

Biggest gain is going from non-functional to functional!

2.3 How can you tell which part matters?

1. Measure existing systems -- look at how existing systems are used, see where bottleneck is.

2. Modeling -- back of the envelope calculations (example: 100 threads times 8KB/stack => lots of wasted memory in OS/2)

3. Simulate algorithms ahead of time -- to evaluate file system approaches, use simulator on measured traces, not real system. Lets you try out alternatives and get performance data more quickly.

4. Tuning
   Build simple system
   Get it running
   Measure
   Tune part that's the bottleneck

5. Go top-down. For all of the above, could measure, model, simulate, build everything, but apply 90-10 rule instead -- get 90% of the information you need, with 10% of the effort.

Moral of performance tuning: If it doesn't stink, don't stir it.