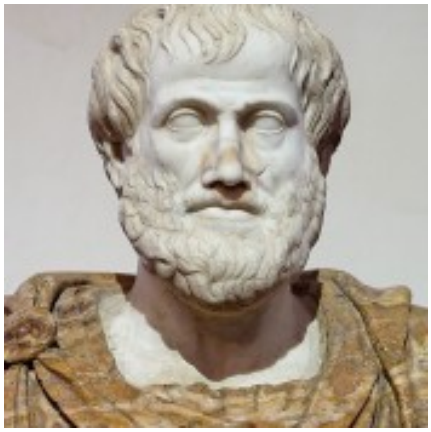
This stuff isn't taught, you learn it bit by bit as you solve each problem.

Wednesday, February 23, 2011 at 9:30AM

Todd Hoff in Strategy



"For the things we have to learn before we can do them, we learn by doing them." -- Aristotle

A really nice Internet moment happened in the HackerNews thread Disqus: Scaling the World's Largest Django Application, when David Kitchen crafted an awesome response to a question about how you learn to build scalable systems. It's so good I thought I would reproduce it here.

Question: asked by grovulent:

Not like this is a problem I have to worry about. But where on earth does one learn this stuff? The talk is useful - as an overview of what they use - but I know nothing of how to implement a single step.

Answer: answered by David Kitchen of buro9:

It's called experience.

Which perhaps sounds rude, but it's not meant to be.

This stuff isn't taught per se, you learn it bit by bit as you solve each problem that you face.

I learned about HAProxy when my site load exceeded that which a single web server could manage.

I learned about heartbeat when I had to update my HAProxy and it knocked the site offline.

I learned about master/slave replication of databases when a site I worked on had considerably more reads than writes and scaling vertically (buying a bigger box) cost more than scaling horizontally (adding cheap read slaves).

I learned of sharding when I worked on a graph stored in an Oracle database and performing calculations on the whole graph exceeded that one physical box.

I learned of one-hop replication to solve the problem of a sharded graph.

I learned of partitioning to solve the problem of having one big database and the computability not being maxed but the storage being maxed.

I learned of memcached when I wanted to reduce page generation times and realised going to the database was more expensive than keeping it in cheap RAM elsewhere on the network: http://www.buro9.com/blog/2010/11/18/numbers-every-developer...

I learned of reverse proxy caches when I wanted to make sure requests for things already served never reached the web layer again.

I learned about Varnish when I considered that most reverse proxies use disk storage for their cache.

We can go on and on here, but the message is that you learn these things one at a time solving real problems that you come up against. There is always the next hurdle to jump through, and when you get there you too will learn how to get past it.

I'd emphasise that you cannot attempt to do this prematurely, that premature optimisation quote really applies well to architecture too. Keep things as simple as they can be and just know that when you get to a hurdle that someone else has already solved it and you've just got to find out where it's written down (if anywhere), what they used, how they approached it, the upsides, downsides, what they'd do differently, etc.

You could try sites like http://highscalability.com/, but I would urge you not to implement things without knowing why you're implementing them. Don't cargo cult this stuff, it's really key to do only what you need to do, when you need to do it.

Related Articles

- Learning Programming Languages with Koans by Mario Aquino
- Ruby Koans
