Strategy: Run a Scalable, Available, and Cheap Static Site on S3 or GitHub

Monday, August 22, 2011 at 9:10AM

Todd Hoff in Example, Strategy

One of the best projects I've ever worked on was creating a large scale web site publishing system that was almost entirely static. A large team of very talented creatives made the art work, writers penned the content, and designers generated templates. All assets were controlled in a database. Then all that was extracted, after applying many different filters, to a static site that was uploaded via ftp to dozens of web servers. It worked great. Reliable, fast, cheap, and simple. Updates were a bit of a pain as it required pushing a lot of files to a lot of servers, and that took time, but otherwise a solid system.

Alas, this elegant system was replaced with a new fangled dynamic database based system. Content was pulled from a database using a dynamic language generated front-end. With a recent series of posts from Amazon's Werner Vogels, chronicling his experience of transforming his All Things Distributed blog into a static site using S3's ability to serve web pages, I get the pleasure of asking: are we back to static sites again?

It's a pleasure to ask the question because in many ways a completely static site is the holly grail of content heavy sites. A static site is one in which files (html, images, sound, movies, etc) sit in a filesystem and a web server directly translates a URL to a file that is directly reads the file from the file system and spits it out to the browser via a HTTP request. **Not much can go wrong** in this path. Not much going wrong is a virtue. It means you don't need to worry about things. It will just work. And it

will just keep on working over time, bit rot hits program and service heavy sites a lot harder than static sites.

Here's how Werner makes his site static:

- S3 - stores the files and serves the website, creating a website without servers. S3 is not your only option, but an obvious one of course for him. We'll talk more about using Github and Google App Engine too.
- Disqus - for comments.
- Bing - site search. Google wants $100 a year for a site search feature. I remember when Google was free...
- DropBox - used to sync website files to whatever computer he is on so they can be edited locally. Then a static site generator is run on the files. And then the files are copied to S3 which makes them available on the Internet using S3.
- Jekyll - static website generator. Written in Ruby and uses YAML for metadata management and uses the Liquid template engine to manipulate the content.
- s3cmd - push files to S3.
- http://wwwizer.com - free service that gets around S3's requirement that your website include www in the domain name. This service will redirect a naked domain name to www.domain so it all works as expected. Joseph Barillari has a good discussion of this issue.

The articles describing his journey are: New AWS feature: Run your website from Amazon S3, Free at Last - A Fully Self-Sustained Blog Running in Amazon S3, and No Server Required - Jekyll & Amazon S3.

Using DropBox is the clever bit here for me. DropBox makes it so the files follow you so you can edit them on any of your machines. This is

also the downside of the approach. You need a local machine with a complete tool set, which is a pain. Ironically, that is why I prefer cloud based approaches. I want to run a blog from any web enabled device, like an iPhone or an iPad, I don't want to mess with programs.

Static sites are **scalable** sites. The web server or OS can easily cache popular pages in memory and serve them like mint juleps at the Kentucky Derby. If a single server is overwhelmed then a static site can easily be served out of a CDN or replicated out to a load balanced configuration. For this reason static sites are **fast**. If you use a distributed file system underneath then you can even avoid the problem of a slow disk becoming the hot spot.

Content is editable with your **favorite text editor**. Nice and **simple**.

Filesystems tend to be **reliable**. Using S3 is even more reliable and **cheap** too. If there is a failure you can just republish or restore your site with a simple command. Databases tend to spike memory, fill up tables, hit slow queries, and have a myriad of other annoying failure modes. All that can be skipped in a static site. Web servers that go much beyond simple file serving can also act the diva.

The problem with static sites is that they are, well, static. Once the Internet was entirely static, except for the blink tag and animated gifs of course. Then CGI changed all that and the web has never sat still since.

So what we do is **outsource** all the dynamic bits to services and leave the content static. Comments can be handled by a service like Disqus. Search can be handled by Bing. Ad serving is already a service. Like buttons are all bits of includible javascript. And **security** concerns (hacking, SQL Injection, etc) are blissfully minimized. This is the mashup culture.

And it mostly works. I seriously considered this approach when I had to

move HighScalability off shared hosting.

Some of the downsides:

- You can't do a lot with .htaccess. If you have a lot security checks and URL mapping magic then you can't do that with S3.
- No PHP or any other language induced dynamism using a language invoked by the web server. You are of course perfectly free to create a service and mash it into your site. Google App Engine is still a great platform for this kind of mini service layer.

The big downside for me was:

- **Not multi-user**. This limitation hits all aspects of the site. I want multiple people to be able to add content to HighScalability. I want to give special privileges to users. I want to assign roles. I want to control what certain users can see. SquareSpace has that capability as do other content management systems. A site generated by a static site generator does not have these capabilities.
- **Engagers**. These are tools to get users engaged with your site so hopefully they'll stick longer. Features like the top posts of all times, read counts on articles, most recent post lists, tag clouds. These are harder to do with static generators.
- **Monetizers**. These are features that help you make money. They often include engagers, but can include features like email list signup, related content recomendations, white paper matching, sign up for consulting services, sponsored text links, yhat sort of thing. Difficult to implement on a static system. An obvious solution is to have a common CMS meta data service that all mashup services can work across, but that probably won't fly.

For building a static website S3 is not the only game in town. Github can

also be used to host a static website. A blog can be generated and updated with a simple git push. This reduces the required set of installed programs to a more manageable level. Now you don't even need git. You can edit files directly using their web interface. The way it works is Github will automatically build your website every time you push changes to the repository. More details here:GitHub Pages, Publishing a Blog with GitHub Pages and Jekyll, and Github as a CDN.

Google App Engine is also an alternative for a static site. More details at: DryDrop, manage static web site with GAE and Github.

Now there's a bit of push to move blogs over to **social networking** sites like Google+. The advantage is you have a built in mechanism for attracting more readers, great discussion features, increased possibility of engagement, no cost, excellent device availability, and no maintenance worries. For blogs that don't need to monetize this is an excellent option. Though I do worry about what happens when you want to hop on to the next trendy social network and all the old content is simply dust.

Wrapping up:

- If your blog is strictly about content then a static web site approach is scalable, fast, cheap, flexible, and reliable. We have a rich set of tools now for making static websites a reality.
- If your blog isn't your precence on the web then invest time in social network (including StackExchage, Quora, etc) sites instead of blog.
- If want to increase user engagement or otherwise creatively monetize your blog then a CMS is a better option.
- If you want to have multiple users and content creators on your blog then a CMS is a better option.

So more links on creating static web sites:

- Hosting a static website on CloudFront by Jean-Michel Lacroix
- Publishing a static website on CloudFront by Jean-Michel Lacroix
- ponyHost - dead simple s3 websites
- Hyde - Hyde is a static website generator powered by Python & Django
- Nanoc - is a Ruby web publishing system that runs on your local computer and compiles documents written in formats such as Markdown, Textile, Haml… into a static web site consisting of simple HTML files, ready for uploading to any web server.
- blogging tools by joshua schachter
- Cactus - static website generator.
- jekyll vs. hyde - a comparison of two static site generators
- jekyll-s3 - Push your Jekyll Blog to Amazon S3!
