Re: What is best searching algorithm for URL

Source: http://coding.derkeiler.com/Archive/General/comp.programming/2006−06/msg00104.html

- *From*: Joe Seigh <jseigh_01@xxxxxxxxxx>
- *Date*: Mon, 05 Jun 2006 06:50:20 −0400

sandeep wrote:

> Our team is developing proxy server(in VC++)which can handle 5000
> clients. I have to implement cache part so when ever a new request com
> from client I have to check the request URL content is in cache of
> proxy and send to client if it is cache, if it is not there then it
> have to get data from web server and store in proxy server cache.
>
> so i am thinking to use binary tree search(or AVL tree) to search
> request URL content in cache if it is not there to insert in it
> is it a good idea so that insertion and searching is faster
>
> I also used hash table and key I has chosen according to first
> character in URL

I would think you'd get a lot of hash collisions that way which
cosidering string compares start with the first letter and could be
expensive.

> So now in that bucket it contain double linked list now I have search
> in it, for that I am thinking to use binary tree

Well, unless you have unlimited storage, you might want to keep that
around to determine least recently used (LRU) URL's to purge when
storage becomes constrained. Also you'll need an ordered list for
URL's expiring from cache.

You also need to worry about locking if you're going to be multi−threaded.
You'll need a global mutex or rwlock which could have contention problems.
A lot of implementations don't bother and trade off stability and integrity
for speed.

You could use lock−free data structures to get around that problem since lock−free is a natural fit for this kind of problem. It's an advanced topic and if you could do that, you wouldn't have posted here in the first place. Linked lists and hash tables are fairly easy to make lock−free. Balanced trees can be made lock−free but I haven't worked out the balancing heuristics to determine how efficient it is. Unmodified AVL and red/black trees algorithms aren't usable for lock−free.

−−
Joe Seigh

When you get lemons, you make lemonade.
When you get hardware, you make software. .