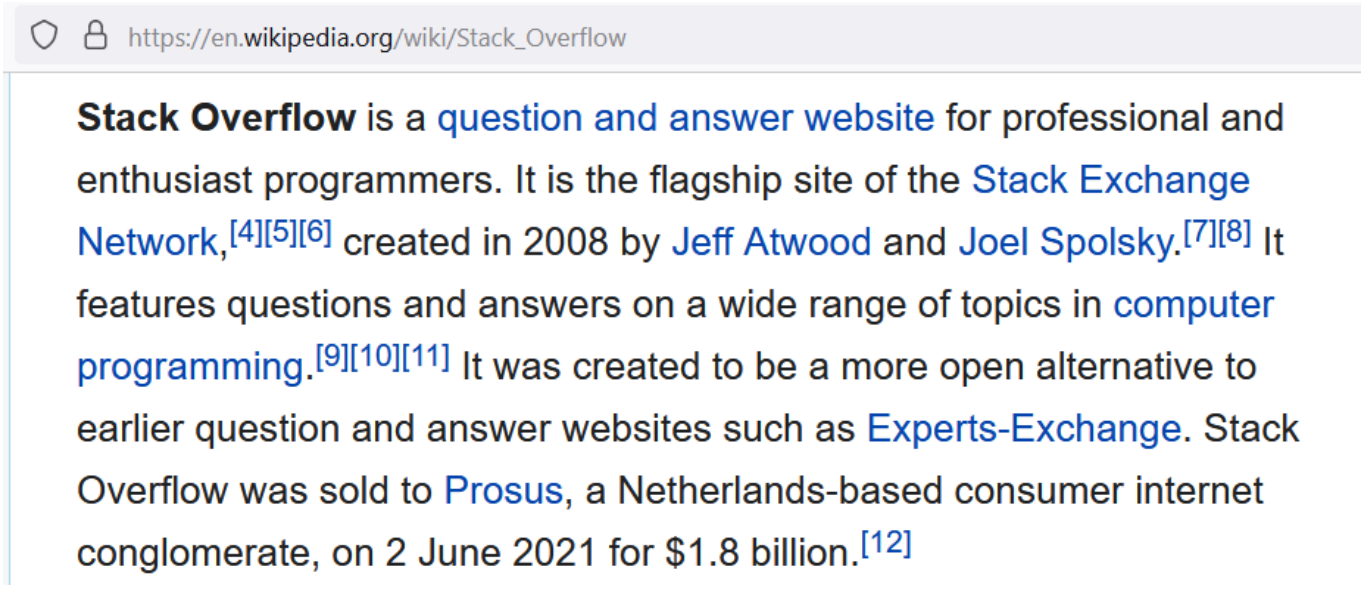
DIGITAL SOFTWARE

Why SIAS Online 3.x the FASTEST School Management System in the Philippines?

1. Stack Overflow



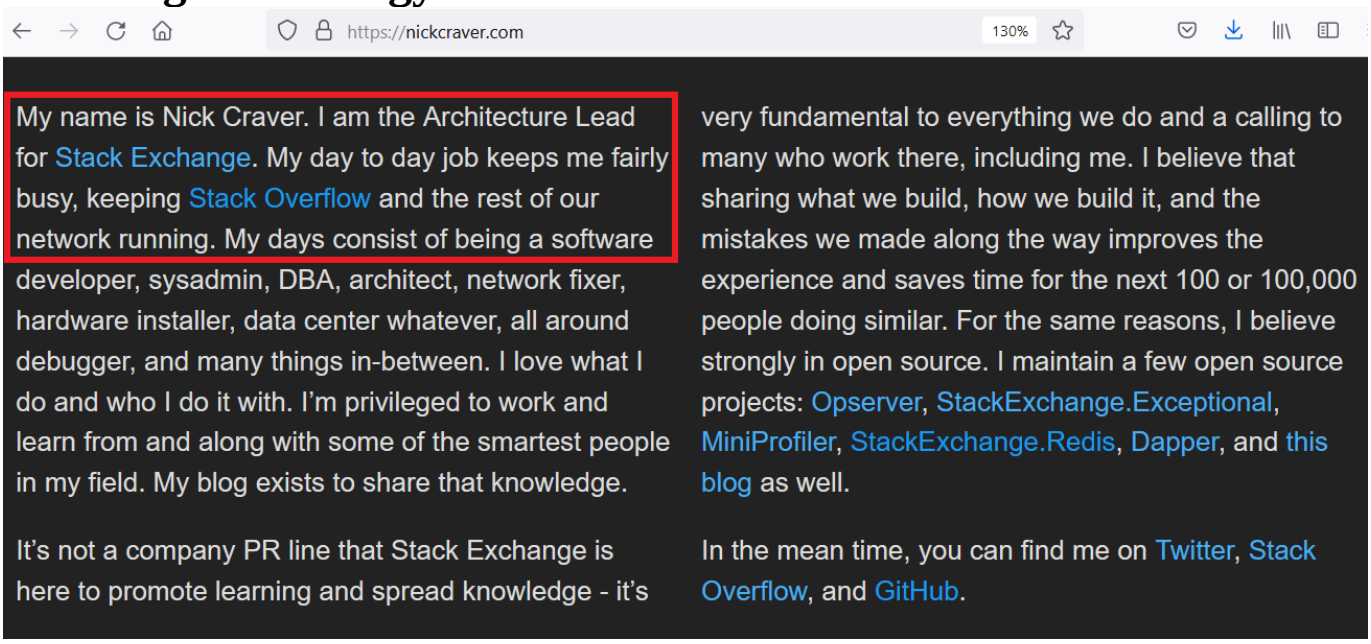
https://en.wikipedia.org/wiki/Stack_Overflow

Stack Overflow is a question and answer website for professional and enthusiast programmers. It is the flagship site of the Stack Exchange Network,[4][5][6] created in 2008 by Jeff Atwood and Joel Spolsky.[7][8] It features questions and answers on a wide range of topics in computer programming.[9][10][11] It was created to be a more open alternative to earlier question and answer websites such as Experts-Exchange. Stack Overflow was sold to Prosus, a Netherlands-based consumer internet conglomerate, on 2 June 2021 for $1.8 billion.[12]

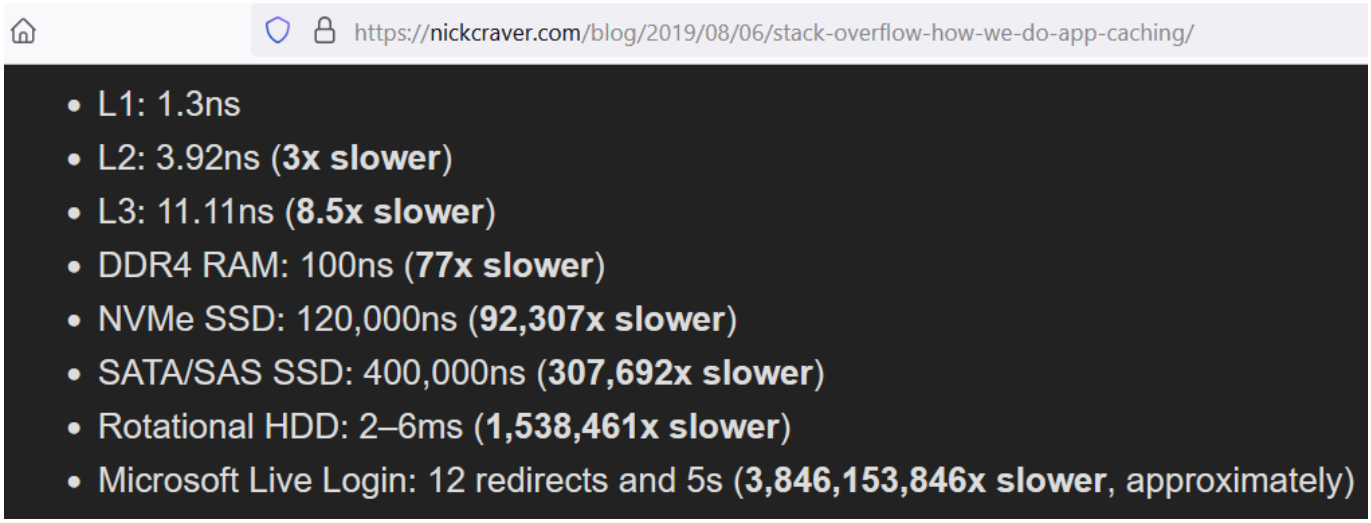
https://en.wikipedia.org/wiki/Stack_Overflow

As of March 2021 Stack Overflow has over 14 million registered users,[17] and has received over 21 million questions and 31 million answers.[18] Based on the type of tags assigned to questions, the top eight most discussed topics on the site are: JavaScript, Java, C#, PHP, Android, Python, jQuery, and HTML.[19] Stack Overflow also has a Jobs section to assist developers in finding their next opportunity.[20] For employers, Stack Overflow provides tools to brand their business, advertise their openings on the site, and source candidates from Stack Overflow's database of developers who are open to being contacted.[21]

2. Caching Technology used in Stack Overflow



https://nickcraver.com/blog/2019/08/06/stack-overflow-how-we-do-app-caching/



https://nickcraver.com/blog/2019/08/06/stack-overflow-how-we-do-app-caching/ 130%

Layers of Cache at Stack Overflow

We have our own "L1"/"L2" caches here at Stack Overflow, but I'll refrain from referring to them that way to avoid confusion with the CPU caches mentioned above. What we have is several types of cache. Let's first quickly cover local and memory caches here for terminology before a deep dive into the common bits used by them:

- **"Global Cache"**: In-memory cache (global, per web server, and backed by Redis on miss)
  - Usually things like a user's top bar counts, shared across the network
  - This hits local memory (shared keyspace), and then Redis (shared keyspace, using Redis database 0)
- **"Site Cache"**: In-memory cache (per site, per web server, and backed by Redis on miss)
  - Usually things like question lists or user lists that are per-site
  - This hits local memory (per-site keyspace, using prefixing), and then Redis (per-site keyspace, using Redis databases)
- **"Local Cache"**: In-memory cache (per site, per web server, backed by *nothing*)
  - Usually things that are cheap to fetch, but huge to stream and the Redis hop isn't worth it
  - This hits local memory only (per-site keyspace, using prefixing)

What do we mean by "per-site"? Stack Overflow and the Stack Exchange network of sites is a multi-tenant architecture. Stack Overflow is just one of many hundreds of sites. This means one process on the web server hosts all the sites, so we need to split up the caching where needed. And we'll have to purge it (we'll cover how that works too).

Redis

Before we discuss how servers and shared cache work, let's quickly cover what the shared bits are built on: Redis. So what is Redis? It's an open source key/value data store with many useful data structures, additional publish/subscriber mechanisms, and rock solid stability.

3. Caching Technology used in SIAS Online 3.x

1st Layer: BitFaster.Caching

https://github.com/bitfaster/BitFaster.Caching

README.md

Method	Mean	Error	StdDev	Ratio	Gen 0	Allocated
ConcurrentDictionary	16.76 ns	0.322 ns	0.285 ns	1.00	-	-
FastConcurrentLru	18.94 ns	0.249 ns	0.220 ns	1.13	-	-
ConcurrentLru	21.46 ns	0.204 ns	0.191 ns	1.28	-	-
FastConcurrentTLru	41.57 ns	0.450 ns	0.376 ns	2.48	-	-
ConcurrentTLru	43.95 ns	0.588 ns	0.521 ns	2.62	-	-
ClassicLru	67.62 ns	0.901 ns	0.799 ns	4.03	-	-
RuntimeMemoryCacheGet	279.70 ns	3.825 ns	3.578 ns	16.70	0.0153	32 B
ExtensionsMemoryCacheGet	341.67 ns	6.617 ns	6.499 ns	20.35	0.0114	24 B

DIGITAL SOFTWARE

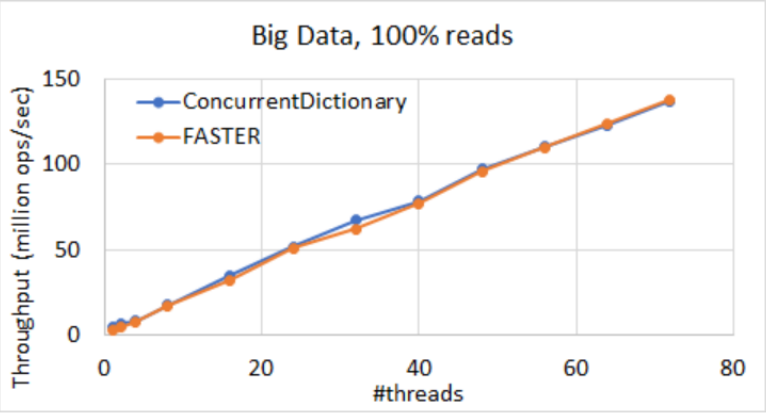
2nd Layer: Microsoft.FASTER.Core

https://github.com/Microsoft/FASTER/wiki/Performance-of-FASTER-in-C%23

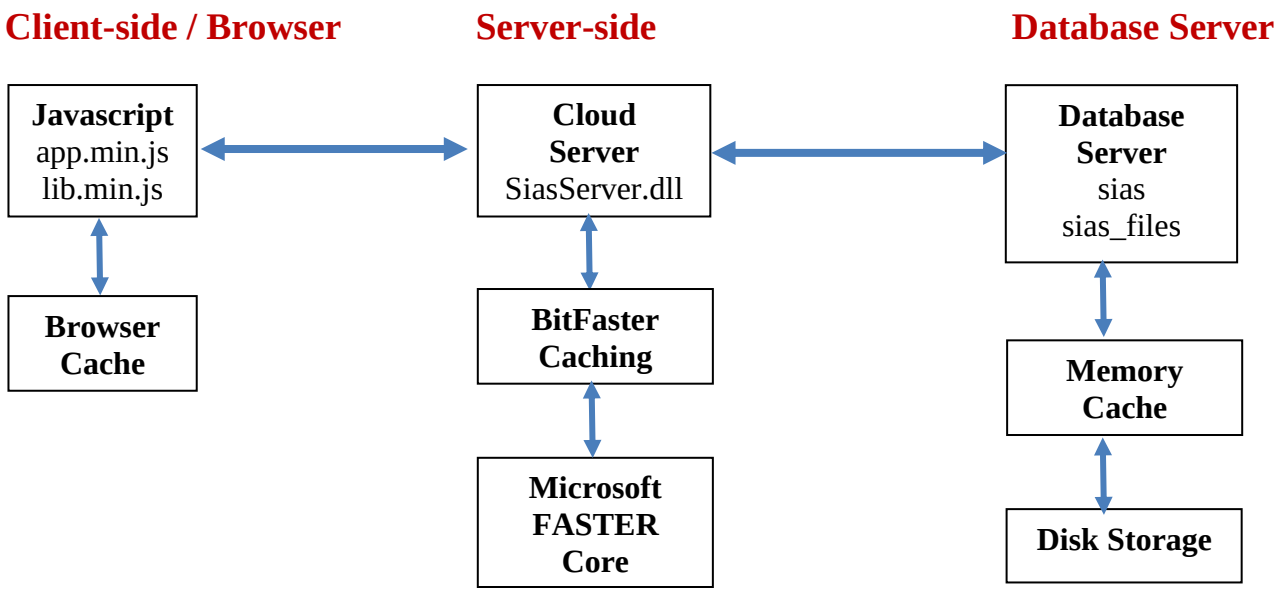
90-99% Reads

Interestingly, we found that with 90% reads, at 72 threads, the performance of FASTER was 139M ops/sec, while ConcurrentDictionary achieved 1.32M ops/sec. At 95% reads, FASTER gets 137M ops/sec, whereas ConcurrentDictionary achieved 3.62M ops/sec. This shows that with even a small fraction of updates, FASTER becomes a good option to consider if it fits your usage scenario. With a very high read fraction of 99%, ConcurrentDictionary gets 14.87M ops/sec, while FASTER achieves 139M ops/sec. As expected, with very few updates, ConcurrentDictionary starts doing better as it avoids the overheads and contention related to updates.

100% Reads



4. SIAS Online 3.x Architecture



5. Multi-Campus Cache Backplane