Metadata

Friday, July 4, 2014

Distributed is not necessarily more scalable than centralized

Centralized is not necessarily unscalable!

Many people automatically associate centralized with unscalable, and distributed with scalable. And, this is getting ridiculous.

In the Spring semester, in my seminar class, a PhD student was pitching me a project for distributed storage: syncing from phone to work/home computers and other phones. The pitch started with the sentence "Dropbox is unscalable, because it is centralized". I was flabbergasted, and I asked a couple of times "Really? Do you actually claim that Dropbox is unscalable?". The student persisted and kept repeating that "Dropbox has a bottleneck because it is a centralized storage solution, and the distributed solution doesn't have that bottleneck". I couldn't believe my ears.

Dropbox already proved it is scalable: It serves files for more than 200 million users, who store 1 billion files every 24 hours. That it has a centralized architecture hosted in the cloud doesn't make it unscalable. As far as I can see there is no bottleneck caused by Dropbox having a more centralized architecture.

(For those who want to nitpick, I know Dropbox is not fully centralized; it uses AWS S3 for storage and Dropbox-company servers for metadata management. Also, it employs data parallelism in the backend for scalability, but, on the spectrum, it is closer to a centralized architecture than a fully decentralized one.)

Distributed is not necessarily scalable!

> Some people when faced with a problem think, I know, I'll use distributed computing. Now they have N^2 problems. -- @jamesiry

Here is the second part. Distributing a system does not necessarily make it scalable. In fact, a fully decentralized architecture can sometimes be a disadvantage for scaling.

Consider Lamport's mutual exclusion (ME) algorithm presented in his seminal "Time, Clocks, and the Ordering of Events in a Distributed System". This ME algorithm is fully decentralized, and requires O(N) messages to be exchanged in response to one ME request. The Lamport ME algorithm employs broadcasts to keep all the nodes informed of all updates and get them on the same (more or less) state.

Now consider a centralized algorithm for ME: there is a centralized coordinator; the nodes send their request to the coordinator, and the coordinator assigns ME accordingly. (For the literalist: You can still have causal ordering in the centralized algorithm. Just use VC when nodes communicate and include VC in the request messages.) The centralized ME algorithm is more scalable: only 1 message is exchanged in response to one ME request. It has less drama and it is easier to maintain and build over.

Single point of failure?

> A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable. -- Leslie Lamport

A common reflex argument about centralized solutions is that it constitutes a single point of failure (SPOF). But if a distributed solution is not designed carefully, it will have multiple points of failures (MPOF). Which one would you rather have?

Let's reconsider the Lamport ME and the centralized ME algorithms. The distributed algorithm does not offer any fault-tolerance advantages. Both algorithms are prone to getting stuck with one crash failure.

In fact, we can argue that it is easier to design fault-tolerance to a centralized solution: You can employ Paxos to replicate the centralized server. In contrast, it is often much

About Me

Murat

I am a computer science and engineering professor at SUNY Buffalo. I work on distributed and networked systems and fault-tolerance. Here is my webpage. You can also follow me on Twitter.

View my complete profile

Blog Archive

- ▼ 2014 (23)
  - ► October (1)
  - ► September (6)
  - ► August (2)
  - ▼ July (2)
    - Hybrid Logical Clocks
    - Distributed is not necessarily more scalable than ...
  - ► June (4)
  - ► March (3)
  - ► February (4)
  - ► January (1)
- ► 2013 (25)
- ► 2012 (18)
- ► 2011 (38)
- ► 2010 (31)
- ► 2007 (1)

Popular Posts

- Paper summary: ZooKeeper: Wait-free coordination for Internet-scale systems
- Hybrid Logical Clocks
- Spanner: Google's Globally-Distributed Database
- Distributed is not necessarily more scalable than centralized
- Ceph: A Scalable, High-Performance Distributed File System
- Paper summary: Tango: Distributed Data Structures over a Shared Log
- My Advice To My Graduate Students
- Scaling Memcache at Facebook
- MDCC: Multi-Data Center Consistency
- Case for RAMClouds: Scalable High-Performance Storage Entirely in DRAM

harder to design and add fault-tolerance to a distributed system. Since a distributed system is complex, it is more prone to introduce corner cases that jeopardize fault-tolerance.

Conclusion

Distributed is not necessarily more scalable than centralized;
And centralized is not necessarily a scalability bottleneck.

As a distributed systems professor, I wouldn't imagine myself defending centralized solutions. But there it is.

To avoid potential misunderstandings, I am not saying fully distributed/decentralized solutions are bad and to be avoided. There are advantages to decentralization, like latency reduction. And some conditions necessitate decentralization, like geographic/political/corporate isolation. We know in the real world it is a mix of centralized, up to where that is manageable and has reasonable cost, and some distributed architecture. This also depends very much on the application/task.

PS: Maybe we should do an XtraNormal animation movie about this "centralized unscalable and distributed scalable" mania. Any takers?

PS2: I thank @tedherman for improvements to the 1st draft.

PS3: Optimistic replication is a great survey of more decentralized replication protocols, their advantages, and challenges.

Bonus Section: Paxos is a relatively centralized approach to distributed consensus

Consensus is usually not an all-hands-on process. That can be hard to scale. Consider our democratic system: It is pretty centralized; we only elect leaders to rule for us.

In the same sense, you can think of Paxos as the more centralized approach to distributed consensus and state machine replication. In Paxos, the participants do not interact with all other participants to decide the order of requests to be accepted, instead the leader dictates the order of requests and the participants just accept them. (A fully decentralized consensus algorithm would be like the synchronous rounds consensus algorithm where in every round each participant communicates with all other participants so that they can converge on the same state.)

Posted by Murat at 4:19 AM

g+1 +8 Recommend this on Google

2 comments:

Anonymous said...

What part of dropbox is centralized? It is massively distributed.

October 14, 2014 at 2:23 AM

Anonymous said...

I think all the "magic" is done in AWS using S3. Dropbox is just some kind of wrapper of S3.

October 14, 2014 at 8:09 AM

Post a Comment

Links to this post

Create a Link

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

Pageviews

235,492

Subscribe To

Posts

Comments

Search This Blog

Search

Follow by Email

Email address...

Submit