Oliver Tipton

April 28, 2024

Dynamo: The Data Storage Platform for all of your shopping needs

Novel Idea:

Amazon boasts the largest e-commerce platform in the world with over 310 million users. With that many users, Amazon needs a data storage platform which can handle hundreds of thousands of daily reads and writes, as well as scalability. In, “Dynamo: Amazon’s Highly Available Key-value Store” written by Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall and Werner Vogels, the data storage platform which helps manage Amazon’s expansive data set is described. Dynamo, built by Amazon, is highly available, symmetric, and decentralized with minimal need for manual assistance at any given time. Whereas other data store platforms typically prioritize consistency, Dynamo prioritizes availability through a hashing algorithm and replication. This allows Dynamo to maintain functionality even while storage devices and networks may be failing. It uses a gossip protocol as a way for nodes to keep track of relevant information to the other nodes around them, greatly reducing latency time for reads and writes. Simply put, Amazon has fulfilled its own need for a data store platform that handles

monolithic size, allows for scalability, and runs with quick latency, all while remaining reliable and efficient.

Main Result(s):

Dynamo is usable across a myriad of Amazon's different functionalities. One such functionality is *business log specific reconciliation,* meaning reconciling or merging potentially divergent versions. An example of this functionality which is affected by Dynamo is the shopping cart, where Dynamo ensures an incredibly low probability of divergent versions and keeps track of the data in case of a necessary merge. The platform is also successfully used for *timestamp based reconciliation* and *high performance read engine*. Clients have found usage in the platform's "tunability," where clients can adjust durability, consistency, and availability. These options, coupled with its broad-based usage across multiple functionalities, show how Dynamo is able to be an efficient and accurate data storage platform for the world's largest e-commerce site.

Impact:

As seen in "How To implement any concurrent data structure," simpler algorithms and data storage platforms, despite not being customized, seem to produce the most efficient broad-based solutions to problems across a range of different sectors. While there may be significantly more complex and intricate data store and collection platforms than Dynamo, Amazon manages to use a more simplistic and broad-based solution while maintaining the top position of the global e-commerce and company leaderboard. Examples like Dynamo illustrate an interesting dichotomy in the Computer Science field between complicated, targeted solutions

and simple solutions which can handle more cases. This continues to add value to how Computer Science skills can be applied and even simplified to better fit even more "real-world" necessities.

Evidence:

One experiment conducted by the Amazon team over 24 hours showcased an impressive 99.94% of requests seeing only one version of their shopping cart, 0.00057% of requests seeing 2 versions, 0.00047% of requests seeing 3 versions, and 0.00009% of requests seeing 4 versions. This showcases an extremely low number of divergent versions while using Dynamo, a testament to the efficiency and effectiveness of Dynamo despite not prioritizing consistency and being highly scalable. In addition to this small case study, across the entirety of Amazon's existence, applications have received successful responses for 99.9995% of requests. There has been no data loss to date in the history of Amazon's services. While there was not a lot of information about Dynamo's results versus their competition, these stats alone speak to Dynamo's success in its ability to manage large user data collections in parallel.

Prior work:

There are many Peer-To-Peer systems that have built successful data storage and collection platforms. Within the scope of Dynamo, Bayou, Coda, and Ficus are some of the most important to look at. These three systems are resilient to network partitions and power outages in the same way that Dynamo is. The 4 systems share the ability to continue read and write

operations during power outages and network shutdowns. However, Dynamo builds upon these previously mentioned platforms and differentiates in what it "targets." For starters, Dynamo prioritizes applications which look for "always writable" data storage, meaning no updates are rejected due to concurrency writing issues. It is also built for a network/application where all nodes within the single infrastructure are to be trusted. Applications which use Dynamo also do not require support for hierarchical namespaces as a result of its use of key value pairings and gossip protocol. Lastly, it is built for systems which require a super low latency and nearly instantaneous response time (Amazon). Dynamo builds upon these other decentralized storage systems by adding key value pairings, lower latency, and differentiating what the system itself is designed for.

Reproducibility:

Results generated in section 6 were specifically provided through "aggregated measurements instead of absolute details"(206) due to Amazon's personal interest in protecting its data store. Without having access to the intricate mechanics of the Dynamo system and the exact tests which were performed (as well as absolute details to cross-check solutions), it would be extremely difficult to reproduce these types of results. That being said, simply running front-end tests such as creating 1000 shopping carts and marking the number of times there were misplaced items or overwrites would be a way to get a generic sense on a small-scale of how accurately Dynamo functions.

Question:

At what point does shifting to a simplistic algorithm and platform make more sense in a system than a specifically curated algorithm? Where does that line lie between complicated mechanisms, high-level functionality, and protocols which will continue to grow in size?

Criticism:

While the essay spoke to Dynamo's effectiveness and even compared its differing versions, the paper somewhat lacked empirical comparison to competitors. Section 3 talked about other peer-to-peer data systems, yet there was no hard evidence or numerical comparisons that showed Dynamo running more efficiently, having better data storage, or any empirical comparison at all. A lack of empirical comparison evidence, despite Amazon's success, makes it difficult to see in an analytical practice whether or not Dynamo truly is the best choice of data platform over a different type. In addition to comparisons, more empirical evidence overall would have made the essay stronger. Without numbers, especially in Computer Science, it's hard to get a full grasp of why a protocol, platform, or algorithm is effective or usable (Amazon's data storage platform is an outstanding example given Amazon's success, but it still would have strengthened the paper).

Ideas for Further Work:

This essay made me curious how something like Dynamo could be used for large-scale video games with many users. Video games often experience lag as a result of congestion and high numbers of players. While I don't know how feasible this would be, I'm curious if different characters/locations could be treated as nodes, so movement from place to place (node to node) would be simpler as a result of the gossip protocol. Since Dynamo seems most effective and

conducive to extremely large data sets, it seems like Dynamo could be an effective way to reduce lag time on a simpler video game with a lot of users.