Modern Peer-to-peer Storage: Dynamo and Pond

Greg Hill

Cornell University

October 29, 2013

"The internet has transformed communication for distributed applications: each new system need not implement its own network, but can simply assume a shared global communication infastructure. A similar transformation might be possible for storage, allowing distributed applications to assume a shared global storage infrastructure." - DHash++ (a p2p storage system from MIT based on Chord) [4]

Total overview

↓		
Network	*Storage*	*Cloud*
Chord	*DHash++*	*Dynamo*
Can	*CFS*	*Cassandra*
Pastry	*PAST*	*GoogleFileSystem?*
Tapestry	*POND*	
Bamboo		

Background - Network level p2p

Last week - *Chord* and *The Impact of DHT Routing Geometry on Resilience and Proximity* gave us some important considerations for a p2p network

- Lookup complexity, routing geometry
- Fault tolerance, load balance
- Flexibility of node placement, route selection
- *The Impact of DHT Routing Geometry* highlighted the importance of considering physical peer proximity

Other network level considerations

- Churn: the flux of nodes arriving and leaving a network
- Iterative vs. recursive lookup
- More indirection?

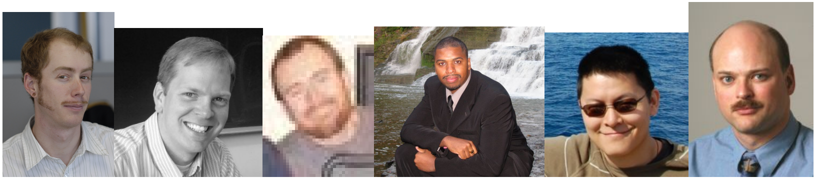
Total overview

	$\downarrow$	
Network	*Storage*	*Cloud*
Chord	*DHash++*	*Dynamo*
Can	*CFS*	*Cassandra*
Pastry	*PAST*	*GoogleFileSystem?*
Tapestry	*POND*	
Bamboo		

p2p network $\rightarrow$ p2p storage

- Peer-to-peer was HOT!
- Storage concerns:
  - Durability (vs. availability)
  - Load balance
  - Performance
  - Untrusted members
- One such p2p storage system was Pond - the Oceanstore Prototype
  - Store my baby photos for life on untrusted peer-to-peer network!

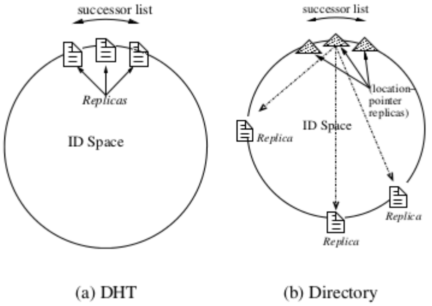
Pond Authors



Sean Rhea [Meraki], Patrick Eaton [Stackdriver], Dennis Geels [Google], Hakim Weatherspoon [Cornell], Ben Zhao [UCSB], and John Kubiatowicz [Berkeley]

Network level: Tapestry

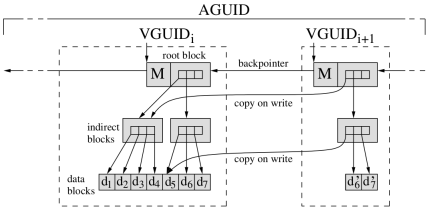
- Allowed peer-to-peer network aspects to be abstracted away.
- Resources and hosts labeled by *globally unique identifier* (GUID)



[3]

- GUID was level of indirection

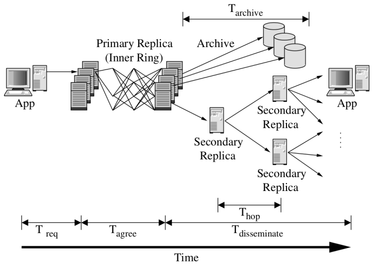
Data model



[1]

- All data (as well as nodes in the system) identified by Global Unique Identifiers (GUID)
- BGUID for data or indirect blocks, a hash of the contents
- VGUID for root block of a version of an object, hashes whole object at that version
- AGUID for a data object (all versions)
- Forms hash tree where pointers are the hash of the block
- Copy-on-write data model

Architecture



[1]

- Primary Replicas - each data object assigned to one (form inner ring)
  - Serialize writes and manage ACL restrictions
  - Map AGUID (object identifier) to most recent VGUID with heartbeat certificates
- Secondary replicas
  - Can handle read requests
  - Can cache data blocks?
- Archival storage
  - Store data encoded with Cauchy Reed-Solomon codes

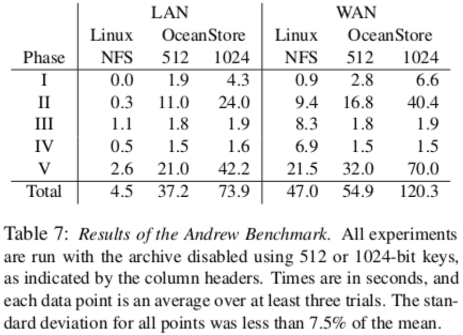
Inner Ring

- Use Byzantine agreement algorithm to coordinate assuming up to $\frac{1}{3}$ could be faulty
- Proactive threshold signatures allow inner ring churn
- Membership managed by a "responsible party" that helps initialize network
- Multicast tree to push updates to secondary replicas

Consistency

- Writes must go through primary replica to be serialized, ACID semantics
- Updates are array of potential actions guarded by a predicate tunable per-application
- Applications can also specify predicates over reads
- No support for explicit locks

Performance



	LAN			WAN		
	Linux	OceanStore		Linux	OceanStore	
Phase	NFS	512	1024	NFS	512	1024
I	0.0	1.9	4.3	0.9	2.8	6.6
II	0.3	11.0	24.0	9.4	16.8	40.4
III	1.1	1.8	1.9	8.3	1.8	1.9
IV	0.5	1.5	1.6	6.9	1.5	1.5
V	2.6	21.0	42.2	21.5	32.0	70.0
Total	4.5	37.2	73.9	47.0	54.9	120.3

Table 7: *Results of the Andrew Benchmark.* All experiments are run with the archive disabled using 512 or 1024-bit keys, as indicated by the column headers. Times are in seconds, and each data point is an average over at least three trials. The standard deviation for all points was less than 7.5% of the mean.

[1]

Discussion

- How important was the distinction between the inner and outer rings?
- Why is Byzantine agreement needed to build a fault-tolerant primary replica for each data object?
- Why was this overall vision on peer-to-peer storage not realized?
- We still have big drives today! Coupled with the coming of fast fiber internet (from Google Fiber and competition they inspire) for consumers, would something like this system be viable? Maybe it was just before its time? (think: power)

Takeaways

- Designing a system for untrusted machines is hard!
- The original Oceanstore vision is possible to achieve
- Exposing consistency options to client allows flexible system
- Erasure coding for durability on unreliable systems
- Privacy complements integrity

Total overview

		↓
Network	*Storage*	*Cloud*
Chord	*DHash++*	*Dynamo*
Can	*CFS*	*Cassandra*
Pastry	*PAST*	*GoogleFileSystem?*
Tapestry	*POND*	
Bamboo		

Move to the cloud

- Peer-to-peer systems where too unreliable / bad performance, academic research moved on to new challenges
- The future is in the cloud, but many lessons and inspiration drawn from previous peer-to-peer systems
- Environment is now trusted, network bandwidth is high, machines more available

Dynamo Authors

Werner Vogels left Cornell to make $$$ at Amazon (now VP/CTO)

Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall and Werner Vogels

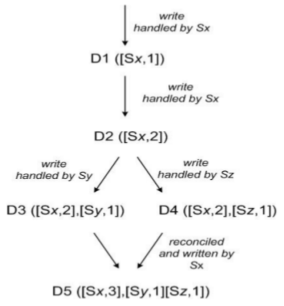
Dynamo overview

- Evolution from Chord, dhash++ to from WAN peer-to-peer to the cloud on an internal network
- Primary-key only API
- Customizable levels of replication
- SLA requirements, focus on 99.9% latencies over mean/median

Consistency

- Consistency is tunable, (N, R, W)
- Quorum if $R+W > N$, otherwise sloppy quorum
- They usually chose not to quorum, claim ACID is not sufficiently avaliable (CAP)
- System instead eventually consistent, and resolves conflicts during reads not on writes
- Writes are more important to being succesful for Amazon

Architecture



[2]

- Uses consistent hashing with virtual nodes, explored new token placement strategies
- O(1) hops with all nodes storing where all others are (scalability concern)
- Vector clocks and version trees

Handling Failures

- Hinted Handoff
- Replica synchronization
- Gossip based failure detection
- Read repair

Performance

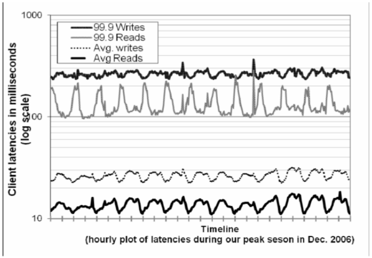


Figure 4: Average and 99.9 percentiles of latencies for read and write requests during our peak request season of December 2006. The intervals between consecutive ticks in the x-axis correspond to 12 hours. Latencies follow a diurnal pattern similar to the request rate and 99.9 percentile latencies are an order of magnitude higher than averages

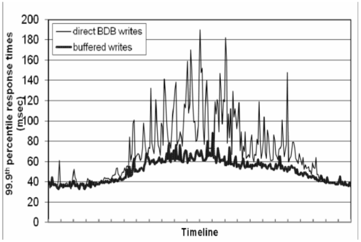


Figure 5: Comparison of performance of 99.9th percentile latencies for buffered vs. non-buffered writes over a period of 24 hours. The intervals between consecutive ticks in the x-axis correspond to one hour.

[2]

Discussion

- Isn't it a data store's job to resolve conflicts or not have them at all?
- How do you design to minimize 99.9% latencies?
- Why do they worry about rare worst case latencies but not rare divergent versions?
- Why did Dynamo use replication while Pond used erasure coding?
- Why did both systems use event based programming?

Takeaways

- Peer-to-peer techniques can be used to make highly available cloud systems.
- Choose when it is important to design for the tail
- Let applications tune consistency (N, R, W is great!)
- Let applications do conflict resolution
- Availability vs. Durability

Cited material

[1] Sean Rhea, Patrick Eaton, Dennis Geels, Hakim Weatherspoon, Ben Zhao, and John Kubiatowicz. 2003. Pond: the oceanstore prototype. In Proceedings of the 2nd USENIX conference on File and storage technologies (FAST'03). USENIX Association, Berkeley, CA, USA, 1-1.

[2] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. 2007. Dynamo: amazon's highly available key-value store. In Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles (SOSP '07). ACM, New York, NY, USA, 205-220. DOI=10.1145/1294261.1294281 http://doi.acm.org/10.1145/1294261.1294281

[3] Byung-Gon Chun, Frank Dabek, Andreas Haeberlen, Emil Sit, Hakim Weatherspoon, M. Frans Kaashoek, John Kubiatowicz, and Robert Morris. 2006. Efficient replica maintenance for distributed storage systems. In Proceedings of the 3rd conference on Networked Systems Design & Implementation - Volume 3 (NSDI'06), Vol. 3. USENIX Association, Berkeley, CA, USA, 4-4.

[4] Frank Dabek, Jinyang Li, Emil Sit, James Robertson, M. Frans Kaashoek, and Robert Morris. 2004. Designing a DHT for low latency and high throughput. In Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation - Volume 1 (NSDI'04), Vol. 1. USENIX Association, Berkeley, CA, USA, 7-7.