Dynamo: Amazon’s highly available key-value store

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

Presented by Patrick Owen
Basic info

• Key-value store
• Used by Amazon’s core services
• Paper is from 2007
Design considerations

• Availability valued over high levels of consistency
• “Always writeable” data store (should strictly avoid rejecting customer updates)
• Conflict resolution handled at read-time
• Incremental scalability: scale out one node at a time without affecting performance
• Symmetry: simplifies system provisioning and maintenance
• Decentralization: Avoids outages from failure of master
• Heterogeneity: Machines with different performance should work together
## Techniques overview (stolen from paper)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Technique</th>
<th>Advantage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partitioning</td>
<td>Consistent Hashing</td>
<td>Incremental Scalability</td>
</tr>
<tr>
<td>High Availability for writes</td>
<td>Vector clocks with reconciliation during reads</td>
<td>Version size is decoupled from update rates.</td>
</tr>
<tr>
<td>Handling temporary failures</td>
<td>Sloppy Quorum and hinted handoff</td>
<td>Provides high availability and durability guarantee when some of the replicas are not available.</td>
</tr>
<tr>
<td>Recovering from permanent failures</td>
<td>Anti-entropy using Merkle trees</td>
<td>Synchronizes divergent replicas in the background.</td>
</tr>
<tr>
<td>Membership and failure detection</td>
<td>Gossip-based membership protocol and failure detection</td>
<td>Preserves symmetry and avoids having a centralized registry for storing membership and node liveness information.</td>
</tr>
</tbody>
</table>
Partitioning

• Consistent hashing
• Each node has multiple virtual nodes
• Much like Chord
• Node failures evenly disperse load
• New nodes accept similar load
• Number of virtual nodes for a given node configurable
Refined partitioning scheme

Strategy 1

Strategy 2

Strategy 3
Replication

- Replicates data items at N hosts
- N is configured per-instance
- The node described previously is the "coordinator node"
- Coordinator node replicates keys to N-1 forward successors in the key space
- Skips virtual nodes for same machine
- Constructed to spread nodes across multiple data centers
- Uses a preference list
Versioning

- Vector clock attached to context counts writes handled by each node
- Conflicts are determined by two concurrent versions
- Conflicts are handled by returning all conflicting versions during a get() operation
- Clocks are truncated by a threshold to prevent unbounded clock size (unclear how this is still correct)
Read/write operation

• Handled by coordinator node (usually top of preference list)
• Involve first N healthy nodes in preference list
• Not all N nodes have to respond for a read/write to be considered successful. This is handled with a configurable R and W value.
Hinted handoff

- All read and write operations are performed on the first N healthy nodes from the preference list.
- Replicas sent to nodes outside the top N nodes in the preference list are given metadata.
- This metadata indicates the node that was down and could not receive the replica.
- When that node recovers, data hinted for that node is sent to it.
Replica synchronization

• Handles situations where hinted handoff fails
• Merkle trees are used to reduce number of comparisons required for synchronization
• Separate Merkle tree maintained for each key range, stored at every node responsible for that key range
Ring membership

- Nodes are not added or removed automatically, as failures are assumed to be transient
- Command line tool explicitly adds or removes nodes
- A node chooses its own tokens when it joins
- Membership change histories and node mappings are reconciled via a gossip-based protocol
- Old nodes transfer keys they no longer own to new nodes
External discovery

• Two nodes that join too closely together may not know about each other, creating a logical partition
• To avoid this, some nodes are seeds
• All nodes contact a seed to learn membership information, so nodes can all learn about each other this way
Failure detection

• Automatic failure detection is handled entirely locally
• Global understanding of failures is unnecessary, as permanent failures are eventually handled with manual node removal.
Overall latency

(hourly plot of latencies during our peak season in Dec. 2006)
Balance of node requests
Refined partitioning scheme review

Strategy 1

Strategy 2

Strategy 3
Comparison of partitioning strategies (N=3, 30 nodes)
Evaluation

• Dynamo seems like a useful system that serves its purpose
• Reconciling conflicts during reads instead of writes is an interesting idea
• Lack of automatic node removal mechanism could cause problems with scaling
• Omitted detail in the paper makes details harder to understand
• Could not find a description of the gossip protocol