Exercise 1 (multiple choice)

=========================== Instructions

**#Excercise 1**: NoSQL Data Structures

Describe the pros & cons of four different NoSQL Data Structure families:

- key-value, column store/column, document, graph database

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute | Key value Store | Column Store | Document Store | Graph Database |
| Type | Associative Array | Records and Arrays | Documents, possibly within other documents | Data Relationships |
| Storage | Match Key with Value, similar to a dictionary | Collections of key value pairs that match a record | Data encapsulated within the document: XML, YAML, JSON, BSON | Tree like structures with nodes and edges |
|  | Each key appears once | Collections of KV pairs |  |  |
| Data Ordering and retrieval | Lexicographic, key ranges | Two dimensional arrays, arranged by key | Collections and Tables | Direct and indirect relationships, paths |
| Examples | NoSQL, Redis | Cassandra, Hbase | Apache CouchDB, MongoDB, Couchbase | Neo4j, Virtuoso, MarkLogic |
| Advantages | Easy to use, efficient, scalable | Scale and quick search | Work with deeply nested data, JSON friendly | Handle complex relational information |

**Sometimes NoSQL databases in the same family can be quite different. What are some differences between Couchbase and MongoDB.**

Couchbase vs. MongoDB (Prominent functional differences)

1. Couchbase supports key-value based storage.
2. MongoDB uses BLOB format for storage. Couchbase only supports JSON
3. Some data fields such as “time” are encoded in a special format by MongoDB
4. MongoDB supports Revision control, Couchbase does not
5. Couchbase supports memcached. MongoDB does not.
6. MongoDB supports range sharding, Couchbase does not.

**#Exercise 2**: Quorum and Dynamo Inspired Systems

[Resource] (http://www.allthingsdistributed.com/files/amazon-dynamo-sosp2007.pdf)

###Instructions

1. **Name at least three systems that implement quorum protocols.**
2. Cloudant
3. CouchDB
4. Riak
5. Cassandra

1. **Define the following:**

\* ‘W’ : Write quorum size, number of nodes that participate in a successful write operation

\* ‘R’ : Read quorum size, , number of nodes that participate in a successful read operation

\* ‘N’ : Replicas in a group

\* ‘Q’ : Quorum level, the number of nodes that must be read from or written to

1. **Why is ‘N’ generally chosen to be an odd**

**integer?**

* Even numbers of servers do not add value to the quorum algorithm
  + For example: While a grouping of 4 servers requires 3 to form a quorum, a group of 5 also requires 3 to form a quorum. But a group of 5 allows 2 servers to fail and still maintain quorum unlike a group of 4 where only one server can fail. Hence, a system with 5 servers is more fault tolerant. This argument extends to all odd numbers.

1. **What condition relating ‘W’, ‘R’, and ‘N’ must be satisfied to "yield a quorum like system"?**

If “W” is a majority of nodes ie. W > (N/2)

And if

R + W > N,

eventual consistency can be attained with probability 100% with no time delay.

1. In the paper "Probabilistically Bounded Staleness,"

Berkeley researchers derive an analytic framework for the probability of reading a "stale" version of an object in a Dynamo-like system that implements quorum.

Using [this tool] (http://pbs.cs.berkeley.edu/\#demo) (lambda=0.1 for all latencies, tolerable staleness=1 version, 15,000 iterations/point), answer the following questions:

1. **With what probability are you reading "fresh" data for n=3, w=2, r=2?**
   1. Probability is a 100% because the rules for eventual consistency (defined above) are fully met.
2. **Does it depend on time? If so, why? If so, why not?**
   1. There is no time dependency; The Read and Write acknowledgements do not have a race condition.
3. Compare the scenarios for (w,r,n) = (2,1,3) and (1,2,3).

With (2, 1, 3) there is at least a 91.52 percent chance of reading the last written version 0 ms after it commits, at least a 96.93 percent chance of reading the last written version 10 ms after it commits and at least a 99.99 percent chance of reading the last written version 100 ms after it commits.

Read Latency: Median 8.49 ms, 99.9th %ile 37.61 ms

Write Latency: Median 16.68 ms, 99.9th %ile 61.19 ms

With (1, 2, 3) there is at least a 96.24 percent chance of reading the last written version 0 ms after it commits, at least a 99.51 percent chance of reading the last written version 10 ms after it commits and at least a 99.99 percent chance of reading the last written version 100 ms after it commits.

Read Latency: Median 16.83 ms, 99.9th %ile 61.82 ms

Write Latency: Median 8.4 ms, 99.9th %ile 37.88 ms

1. Write down and explain the differences (if any) for the time dependence of P(consistent).

With a (w, r, n) 🡪 (1, 2, 3) configuration, the backend consistency requirements are better met because of the quorum of 2 reads than with the (2, 1, 3) configuration which is a minimal quorum. Because multiple reads converge to better read consistency, the (1, 2, 3) configuration reaches consistency faster than the (2, 1, 3) configuration.

1. Is the (2,1,3) state symmetric with (1,2,3)?

No, the states are not symmetric in either one of time or convergence

1. Compare both P(consistent) and the median and 99.9% latencies.
   1. For (2, 1, 3), Read Latency: Median 16.83 ms, 99.9th %ile 61.82 ms, Write Latency: Median 8.4 ms, 99.9th %ile 37.88 ms
   2. For (1, 2, 3), Read Latency: Median 8.49 ms, 99.9th %ile 37.61 ms, Write Latency: Median 16.68 ms, 99.9th %ile 61.19 ms

As surmised in earlier answers, the stricter read quorum leads to better observed-consistency

1. Provide an intuitive explanation for your results.
   1. Stricter write quorums (w ~= N) take longer time
   2. Stricter read quorum definition (r ~= N) result in better consistency
2. Do either of these states favor consistency or availability? If so, why?

Definition:

consistency (all replicas agree on the same value)

availability (you can always get an answer from a replica)

partition tolerance (the system works even if replicas can't talk),

Models that do not strictly follow the quorum system rules favor availability rather than consistency. In both (2, 1, 3) and (1, 2, 3) availability is favored.

The ratio of quorum group sizes for read and write operations should be inverse to the read/write access ratio.

1. Perform a similar comparison for the (3,1,3) and (1,3,3) states. Do either of these states favor consistency or availability? If so, why?

Models that strictly follow the quorum system rules favor consistency at the expense of availability. In both (3, 1, 3) and (1, 3, 3) consistency is favored.

1. In your opinion, assuming an n=3

system, what do you think is a reasonable choice for write heavy, read heavy, and read\~=write workloads?

1. For Write heavy – (w, r, N) 🡪 (1, 2, 3)
2. For Read heavy – (w, r, N) 🡪 (2, 1, 3)
3. For Read ~= Write - (w, r, N) 🡪 (2, 2, 3)

**#Exercise 3: Partitioning Strategies**

What is the benefit of Consistent Hashing versus other approaches (like Range Partitioning)?

Consistent Hashing’s (CH) advantage is most apparent when building distributed systems that can scale up and down. CH is especially useful when nodes are dynamically brought on line with added traffic and when nodes are taken offline with reduced traffic. The CH algorithm:

* Manages key association with servers with the least amount of remapping
* Causes the least disruption to active traffic when nodes come and go through use of a server ring where objects are automatically allocated to the next server in the ring
* Only 1/N –th of buffers are affected by a change when N is the number of servers

NOTES:

With consistent hashing, in addition to hashing the names of objects (URL, time of day etc.), the names of all the servers are also hashed to the same 32-bit range. When a node joins the cluster, it picks a random number, and that number determines the scope of data that it manages.

Range partitioning maps ranges of objects into specific storage instances. For instance, using user ID as a mapping parameter, users from ID-xx to ID-yy will go into instance I1, while users from ID-zz to ID-aa will go into instance I2. This method requires a table that maps user ID ranges to storage instances. This table needs to be managed and so is less desirable compared to consistent hashing. When a change happens, all objects on a server must be mapped to other servers randomly. This algorithm causes a lot more rebalancing, network activity, write activity and administrative activity.

**#Exercise 4: Cloudant**

\*\*NOTE:\*\* Ensure that you replace "username" and "password" in the URLs in this section with your Cloudant username and password

1. Sign up for a free Cloudant account [here] (https://cloudant.com/sign-up/).

2. Create a DB

iMac:Week8 NatarajanShankar$ curl -X PUT -H 'Content-Type: application/json' https://shankar:xxxxxxxxxx@shankar.cloudant.com/crud/

{"ok":true}

1. Run an insert using curl:

iMac:Week8 NatarajanShankar$ curl -d '{"season": "summer", "weather": "usually warm and sunny"}' -X POST https://shankar:xxxxxxxxxx@shankar.cloudant.com/crud/ -H "Content-Type: application/json"

{"ok":true,"id":"45beca4a7bf663ee360b463b2a8c38de","rev":"1-0af5e64fe24d262db237b9f14046f490"}

iMac:Week8 NatarajanShankar$ curl -d '{"season": "summer", "pleasanton [Dweather": "usually warm and sunny"}' -X POST https://shankar:xxxxxxxxxx@shankar.cloudant.com/crud/ -H 'Content-Type: application/json'

{"ok":true,"id":"c69c1ed21940fe31f880dda0234a3929","rev":"1-5c1840bd0fb5eb26407ab451ab410357"}

iMac:Week8 NatarajanShankar$

1. Ensure your new document is in the database:

curl https://username:password@username.cloudant.com/crud/\_all\_docs

iMac:Week8 NatarajanShankar$ curl https://shankar:xxxxxxxxxx@shankar.cloudant.com/crud/\_all\_docs

{"total\_rows":2,"offset":0,"rows":[

{"id":"45beca4a7bf663ee360b463b2a8c38de","key":"45beca4a7bf663ee360b463b2a8c38de","value":{"rev":"1-0af5e64fe24d262db237b9f14046f490"}},

{"id":"c69c1ed21940fe31f880dda0234a3929","key":"c69c1ed21940fe31f880dda0234a3929","value":{"rev":"1-5c1840bd0fb5eb26407ab451ab410357"}}

]}

iMac:Week8 NatarajanShankar$