Riak SmartMachine Benchmark: The Technical Details

October 31, 2010 - by **badnima**

Recently Basho and Joyent entered into a comprehensive partnership to deliver Riak Smartmachines to Joyent customers. We had been experimenting with Riak since early in 2010 and were eager to benchmark its performance on Joyent and ultimately offer a robust NO-SQL solution to our customers. Along the way, we were pleasantly surprised how well the Riak Smartmachine demonstrated a combination of performance, predictability, resilience, and linear scalability that made it relatively quick and easy run a Dynamo-class distributed data storage system.

In September, Bryan Cantrill (VP Engineering Joyent) and Justin Sheehy (CTO Basho Technology) got together for a discussion on the recent benchmark of Basho Riak on Joyent SmartMachines. This blog post is a follow up to that webinar with a technical drill down on the details summarized by Bryan and Justin.

Environment for Tests

The benchmarking studies were run on standard 4GB Riak Smartmachines running Riak 0.12.1. The tests began on a three node cluster. Nodes four and five were subsequently added, and then stepped up to an eight-node cluster which ultimately grew to fourteen nodes in two-node increments.

We used Basho_bench running on a separate Smartmachine to conduct the benchmarks. The configuration can be found in the results folders. The number of workers and object sizes varied across tests.

We used both pareto and uniformly distributed access patterns. The former more accurately represents typical web application access patterns while the latter is useful in demonstrating the graceful performance degradation of Bitcask when the key space exceeds available memory. The protocol buffers interface was used.

The Goal of the Studies

The goal of the study was to demonstrate a baseline for users to understand Riak's performance, stability, predictability, and linear scalability. The systems were not tuned for optimal performance. Instead, we chose to take standard 4 GB Riak Smartmachines and demonstrate throughput and latency for various access patterns and object sizes.

In terms of stability and predictability, we wanted to see how Riak performed both on long duration tests and on tests where the total size of the stored key-value pairs exceeded available memory. Our assumption was that Riak should demonstrate graceful and predictable throughput decline over time to a steady state value as buffers were filled, with minimal impact to GET and PUT latencies and no data loss.

For linear scalability, Riak should behave predictably under various load scenarios and scale predictably. A user should be able to calculate how many additional nodes will be needed for a given load based on the behavior of a cluster with fewer nodes. During our testing, it was fairly simple to quantify per-node-throughput as one possible unit of scale.

Test One: Comparing Riak Behavior for Various Object Size and Access

RT @DevConFive: Joyent to Speak at DevCon5 HTML5 Developers &amp; Design Conference and Judge Demo Event at WebRTC Conference &amp; Expo ...

12:56 PM - 14 Nov 12

@joyent

CONNECT WITH JOYENT

Patterns

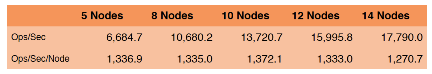
The first three test were run to illustrate the interplay of access patterns and data set size as compared to available system memory on performance.

Test 1.a - 12 hour, 5 node, 1 million keys, uniform distribution

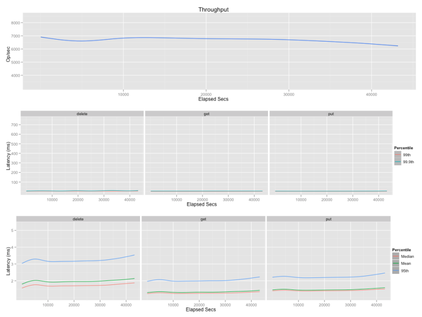
The longest running test, it demonstrates behavior when data set size does not exceed system memory. Tests 1.b and 1.c demonstrate the behavior when data set sizes exceed system memory, additionally comparing Pareto and uniform access patterns. This test demonstrates Riak performance and stability characteristics over a relatively long duration (12 hour) on a system capable of storing all key-value pairs in memory.

- Duration: 12 hours
- Nodes: 5
- Object Size: 2 KB
- Distribution: Uniform
- Number of Keys: 1 million
- R/W/D ratio: 4/4/1

Mean Throughput: 6,684.7 ops/sec



	5 Nodes	8 Nodes	10 Nodes	12 Nodes	14 Nodes
Ops/Sec	6,684.7	10,680.2	13,720.7	15,995.8	17,790.0
Ops/Sec/Node	1,336.9	1,335.0	1,372.1	1,333.0	1,270.7



Tests 1.b and 1.c: Pareto and Uniform Access Patterns and Large Data Sets

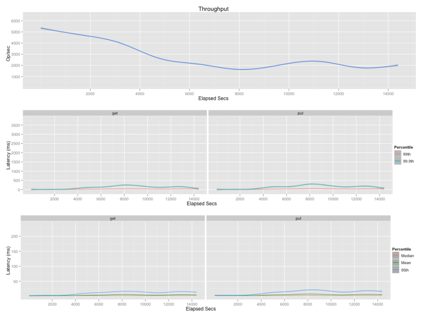
These tests demonstrate Riak performance characteristics using both uniform and Pareto distribution access patterns. These tests demonstrate what happens when the total size of the data set exceeds physical memory.

Test 1.b - Four Hours, Pareto Distribution, Five Nodes, 10

million keys

- Duration: 4 hours
- Nodes: 5
- Object Size: 2 KB min, 5 KB mean
- Distribution: Pareto
- Number of Keys: 10 million
- R/W/D ratio: 1/1

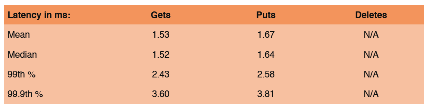
MeanThroughput: 2,762.5 ops/sec



Mean Throughput, Hour 1: 4,639.3 ops/sec

Mean Throughput, Hour 4: 2,009.4 ops/sec

Mean Throughput, Hours 2 to 4: 2,138.6 ops/sec

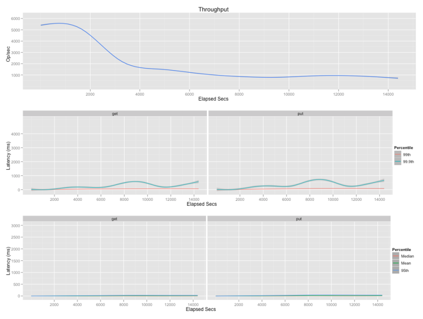


Latency in ms:	Gets	Puts	Deletes
Mean	1.53	1.67	N/A
Median	1.52	1.64	N/A
99th %	2.43	2.58	N/A
99.9th %	3.60	3.81	N/A

Test 1.c - Four Hours, Uniform Distribution, Five Nodes, 10 million keys

- Duration: 4 hours
- Nodes: 5
- Object Size: 2 KB min, 5 KB mean
- Distribution: Uniform
- Number of Keys: 10 million
- R/W/D ratio: 1/1

Mean Throughput: 1,851.9 ops/sec



Mean Throughput, Hour 1: 4,282.5 ops/sec

Mean Throughput, Hour 4: 884.3 ops/sec

Mean Throughput, Hours 2 to 4: 1,038.6 ops/sec

Latency in ms:	Gets	Puts	Deletes

Mean	1.45	1.62	N/A
Median	1.44	1.60	N/A
99th %	2.36	2.49	N/A
99.9th %	3.54	3.81	N/A

Test One: Conclusions

All the tests demonstrated Riak behaves predictably for a given access pattern and for various ratios of total data size to total cluster memory.

Test 1.a demonstrates performance of a Riak cluster provisioned with enough memory to store all active key-value pairs (in the case of uniform access patterns, all data) in memory. Throughput is markedly higher than in Tests 1.b and 1.c.

As the results indicate, Riak performance remains stable (with mean ops/sec of 6,684.7) . There were no errors reported for the period of this test.

Compare results from this test with the results in Test 1.b and Test 1.c. In Tests 1.b and 1.c, also both on a five-node clusters, we see the impact on performance when the key-value pair exceeds available operating system memory: a predictable and graceful decline in performance.

Test 1.b demonstrates a more gradual decline in throughput for tests using a Pareto distribution access pattern. Performance for the first hour, while total data was loading and did not exceed system memory, the system sustained an average of 4,639.3 ops/sec, declining to an hour four average of 2,009.4 ops/sec.

Test 1.c saw a much sharper decline in performance, as expected: 484%, dropping from an hour one average of 4,282.5 ops/sec to 884.3 ops/sec. Hour one average throughput in Test 1.c (uniform access pattern) is 92% that of average first hour throughput in Test 1.b (4,282.5 ops/sec vs. 4,639.3 ops/sec) Average throughput for hours 2 through 4 in Test 1.b is 2,138.6 ops/sec, or 206% that of Test 1.c (1038.6 ops/sec).

In both the 1.b and 1.c test cases, as Test 1.a demonstrates, increasing aggregate cluster memory by either adding more nodes or more memory to existing nodes (both simple options on the Joyent Smart platform) would decrease the number of requests going to disk and increase average throughput.

Median and 99.9th percentile latency remain at acceptable levels. As expected, 99.9th percentile latency for the uniformly distributed access pattern was greater than the Pareto distribution.

Test Two: Demonstrating Riak’s Linear Scaling Capability

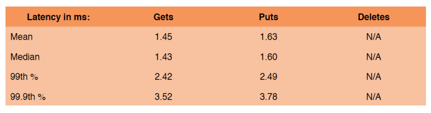
The following tests demonstrate a specific Riak behavior: linear scalability. Linear scalability is another species of predictability. When we add additional Riak Smartmachines to an existing cluster, throughput should increase by a uniform amount predicted by previous performance.

To demonstrate this behavior, we tested Riak performance on 8, 10, 12, and 14 nodes.

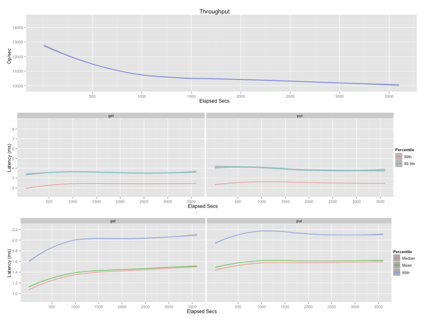
Test 2.a - 8 nodes, 1 Hour

- Duration: 1 hour
- Nodes: 8
- Object Size: 500 bytes min.
- Distribution: Pareto
- Number of Keys: 10 million
- R/W/D ratio: 4/4/1

Mean Throughput: 10,680.2 ops/sec



Latency in ms:	Gets	Puts	Deletes
Mean	1.45	1.63	N/A
Median	1.43	1.60	N/A
99th %	2.42	2.49	N/A
99.9th %	3.52	3.78	N/A



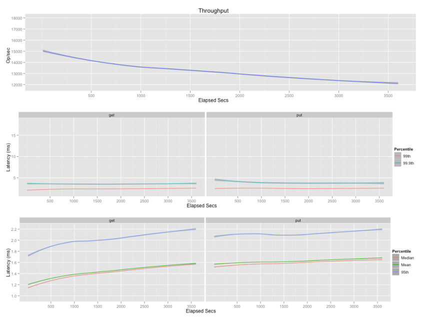
Test 2.b - 10 nodes, 1 Hour

- Duration: 1 hour
- Nodes: 10
- Object Size: 500 bytes min.
- Distribution: pareto
- Number of Keys: 10 million

- R/W/D ratio: 4/4/1

Mean Throughput: 13,720.7 ops/sec

Latency in ms:	Gets	Puts	Deletes
Mean	1.43	1.61	N/A
Median	1.41	1.58	N/A
99th %	2.43	2.51	N/A
99.9th %	3.52	3.86	N/A

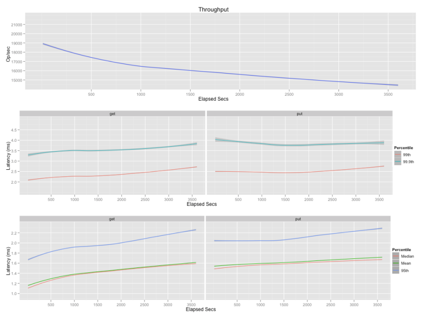


Test 2.c - 12 nodes, 1 Hour

- Duration: 1 hour
- Nodes: 12
- Object Size: 500 bytes min.
- Distribution: pareto
- Number of Keys: 10 million
- R/W/D ratio: 4/4/1

Median Throughput: 15,995.8 ops/sec

Latency in ms:	Gets	Puts	Deletes
Mean	3.92	11.54	N/A
Median	2.19	8.36	N/A
99th %	21.64	49.95	N/A
99.9th %	42.21	181.74	N/A

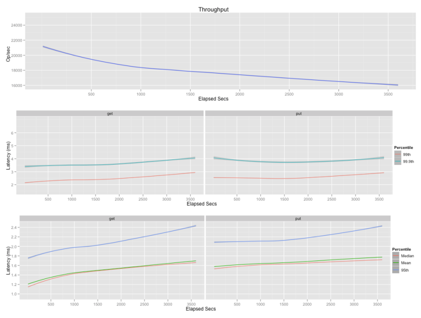


Test 2.d - 14 nodes

- Duration: 1 hour
- Nodes: 14
- Object Size: 500 bytes min.
- Distribution: pareto
- Number of Keys: 10 million
- R/W/D ratio: 4/4/1

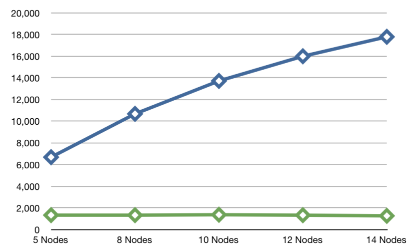
Median Throughput: 17,790.0 ops/sec

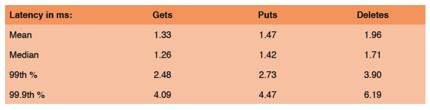
Latency in ms:	Gets	Puts	Deletes
Mean	3.92	4.86	N/A
Median	2.19	2.49	N/A
99th %	21.64	25.41	N/A
99.9th	42.21	49.59	N/A



Test Two Conclusions:

The test results confirm our previous experience - Riak throughput increases in uniform increments as nodes are added. Critically, these tests also show that latency suffers no corresponding increase.





Latency in ms:	Gets	Puts	Deletes
Mean	1.33	1.47	1.96
Median	1.26	1.42	1.71
99th %	2.48	2.73	3.90
99.9th %	4.09	4.47	6.19

The benchmarks particularly highlighted the effectiveness of running Riak on a scale-out platform like Joyent where CPU bursting and fast I/O performance are particularly good.

The results strongly indicate that Riak is particularly well-suited to scale-out platforms like Joyent: for example, when an application requires additional write/read throughput capacity, you can calculate how may more Smartmachines should be added to a clusters. Under the conditions tested, each node added (or removed) increases (or decreases) the cluster's throughput capacity by approximately 1,300 ops/sec.

Overall Conclusions:

Our benchmark tests bring us to the following conclusions:

- - **Riak behaves predictably under high loads** - depending on system resources, Riak exhibits either predictable, steady-state throughput with low errors or degrades gracefully with low errors.
- - **Riak demonstrates stability under high loads** - very few errors, no node failures under load, and behavior in line with expectations.
- - **Riak demonstrates linear scalability** - adding or removing capacity adds or subtracts a predictable amount of capacity from the cluster.

Other notes and conclusions:

- Benchmark raw data is available upon request (we may post it on this blog as a zipped file you can download)
- The benchmarking demonstrated that Riak throughput declines

gracefully and gradually under certain extreme load conditions and that these behaviors, far from unexpected, are predictable enough to be knowingly invoked. Importantly, median and 99th % latencies remained uniformly low throughout the various tests.

- The benchmarking demonstrated that Riak shows stability and resiliency under load. Errors were minimal, no node failures occurred, and no data was lost.
- When we grew the Riak cluster from 5 to 15 servers, Riak demonstrated linear properties of scaling.

POSTED IN: For Devs and Sys Admins, Open Source, Joyent Partners TAGS: basho, Big Data, database, riak, Benchmarks, load balancer

SHARE: