instaclustr

COMPARISON DATASHEET

Instaclustr Managed Cassandra vs *Dynamo DB*

Introduction

Many people consider Apache Cassandra and DynamoDB as potential datastore technologies when looking to build high-scale, high-reliability services in the cloud. Both technologies are popular and well-proven to deliver at scale. However, choosing the technology most appropriate for your use case can have a significant impact on the cost of building, maintaining, and running your application.

This article considers a real-world use case, analyzes the costs of running on Instaclustr Managed Cassandra vs DynamoDB, and discusses how the features and cost models of the two technologies could impact the architecture of your solution. (All pricing presented is for the US East 1 AWS region.)

Use Case

The use case we are considering is the heart of Instaclustr's monitoring system, Instametrics. Instametrics is a time series database collecting monitoring data from the fleet of servers that we manage. The Apache Cassandra database contains both the raw (20 second) level metrics and the roll-ups of those metrics to 5 minute, 1 hour, etc. level and Cassandra's TTL features used to expire aged data. These metrics include data down to the table level in Cassandra and topic level in Kafka for an average of around 3500 metrics per monitored host. The cluster that hosts the database runs both Apache Cassandra and Apache Spark on the same nodes with Spark used to perform the analytics necessary for the rollups. Data is read from the cluster to serve the monitoring pages on the Instaclustr console, the Instaclustr monitoring API, and for other purposes such as checking cluster health as part of automated operations.

The Key Attributes of the Instametrics Cluster:

- 36 i3.2xlarge nodes (co-hosting Apache Cassandra and Apache Spark) (this cluster runs continuously with no scaling up/down for peaks).
- Each metric event written is, on average, ~100 bytes of data.
- Baseline load (raw metrics received) of 3060 batch writes per second. Each batch contains ~150 rows for a total of ~460k writes / second base load.
- Additional load when writing roll-up results in 16,200 batch writes/second. Each batch contains ~100 rows for a total of 1.6M writes/second from this load and **total peak of just over 2M writes per second**. This peak load occurs for about 1 minute out of every 5 (20% of the time).
- The baseline read load on the cluster is about 18,000 reads per second. Each read retrieves ~15 rows for a total baseline read load on the cluster of 270k rows/sec.
- Additional loads when reading data for the roll-ups is about 144,000 reads per second. These reads are actually using Cassandra functions to aggregate data before returning with each read using data from ~15 rows for 2.1M rows/sec read in total. The cluster is also at peak read load for about 20% of the time.
- The cluster currently stores around 54TB of data with a replication factor of 2.

Cost of Instaclustr Managed Cassandra Solution

The AWS infrastructure cost for this cluster (including instances, network, S3 backup, and using reserved instances) is $19,044 per month. Adding Instaclustr management fees to this cluster for a completely managed solution would, at list price, bring total cost approximately $58k per month. Instaclustr has recently released an offering based on the i3en.xl instance type which offers excellent value for money. Migrating to this instance type is expected to bring our total cost down to more like $40k per month inclusive of management fees (so we're migrating soon!).

Cost of AWS Dynamo DB Solution

AWS offers two pricing models for DynamoDB—On Demand Capacity and Provisioned Capacity. The On Demand Capacity model charges per individual read and writes, while the Provisioned Capacity model charges for an allocation of read and write throughput capacity on an hourly basis. The Provisioned Capacity model is significantly cheaper when you have a relatively consistent workload and therefore more suitable for this use case with relatively constant, predictable throughput, so we will work through that in detail.

For a single-region use case such as ours, there are four main cost components to consider:

- **Write Capacity Units (WCU):**
  Each WCU allows 1 write per second (for items up to 1kb in size). Each WCU costs $0.00065 per hour ($0.4758 per month) on demand or $0.0128 per hour per hundred units on 12 month reservation ($0.093696 per unit per month).

- **Read Capacity Units (RCU):**
  Each RCU allows 1 strongly consistent read or 2 eventually consistent reads per second (for items up to 4kb in size). Each RCU costs $0.00013 per hour ($0.09516 per month) on demand or $0.0025 per hour per hundred units on 12 month reservation ($0.0183 per unit per month).
- **Storage:**
  $0.25 per GB per month. One important note here is that DynamoDB includes a per-item storage overhead of 100 bytes to account for indexing which is significant with tiny items such as in our use case.
- **Backup:**
  $0.1 per GB per month.

Our requirements for each of these can be determined from the current cluster load:

- **Write Capacity Units:** 459,000 base load, 2,079,000 during peaks (20% of time)
- **Read Capacity Units:** we are using eventually consistent reads so 270,000/2 = 135,000 base load and 2,160,000/2 = 1,080,000 during peaks (20% of time)
- **Storage:** 54,720 GB (estimating DynamoDB's 100 byte per item overhead and Cassandra's native data compression offset the replication factor)
- **Backup:** also 54,720 GB

To determine AS cost, we firstly need to work out the cost per read/write capacity unit per month. Taking into account both the up-front component and the monthly component:

	WCU	RCU	Maths
Cost per 100 CU per hr	0.0128	0.0025	A
Yearly upfront per 100 CU	150	30	B
Total cost per 100 CU per month	21.8696	2.5	X = (A*732) + (B/12)
Total cost per CU per month	0.218696	0.025	X/100

Applying this to our base load gives us the following cost calculation using 12 month reserved capacity:

Item	Required Volume	Cost per Item per Month	Total Cost
WCU	459,000	$0.218696	$100,381
RCU	135,000	$0.025	$3,375
Storage	54,720	$0.25	$13,680
Backup	54,720	$0.1	$5,472
Total			**$122,908**

We now need to add capacity for the peak loads. As the peaks run for roughly 20% of the time, it seemed to me on first review that we had two choices: reserve capacity 100% of the time, or use on demand capacity and pay the higher per unit cost. However, reading the fine print, capacity can only be reduced once per hour (which some extra fine print that allows 27 times per days) so increasing and decreasing capacity every five minutes would not work.

So, let's calculate the cost based on purchasing reserved capacity to cater for the peak load:

Item	Required Volume	Cost per Item per Month	Total Cost
WCU	2,079,000	$0.218696	$454,669
RCU	1,080,000	$0.025	$27,000
Storage	54,720	$0.25	$13,680
Backup	54,720	$0.1	$5,472
Total			**$500,821**

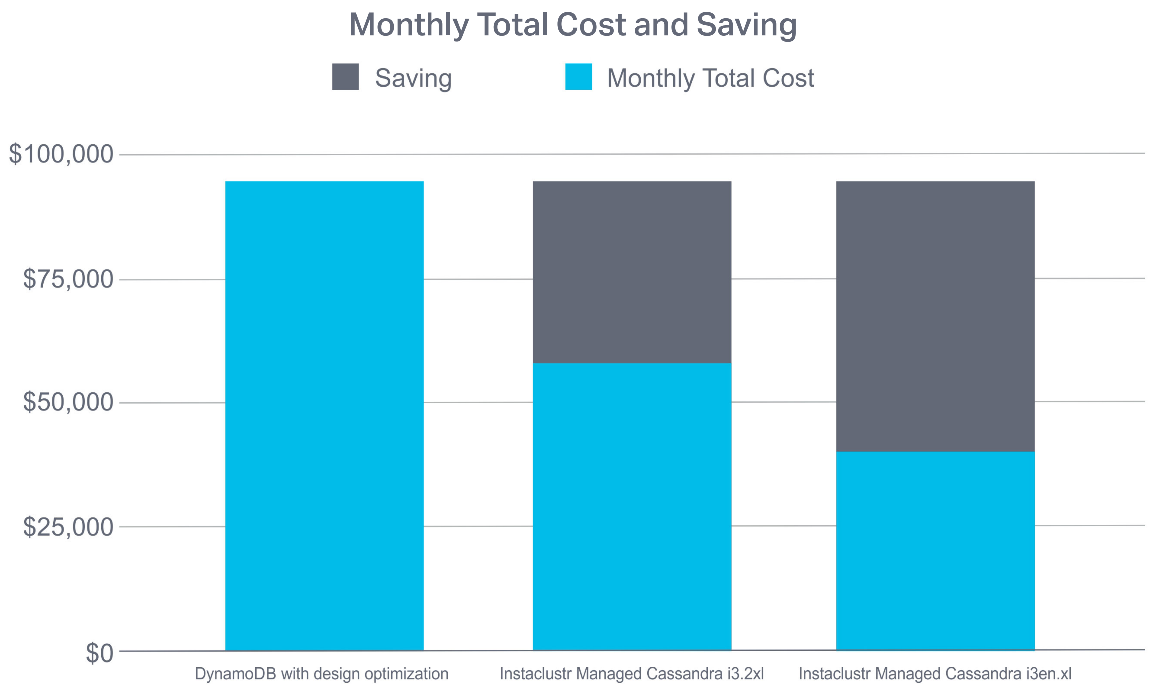
That's a big difference but we're not quite finished yet. We would still need to add support at, say, 5% extra (the actual rate depends on your overall AWS bill) and we also need to run a separate Apache Spark cluster as our Instametrics cluster is also running Spark and the Spark jobs. An estimate for AWS EMR Spark Cluster would be $2,550 per month but on top of this you would have significant self management costs as the EMR model provides minimal management. With these additions we get to a grand total AWS cost of $528,431.

In case you're wondering if the other pricing models would be a better fit, crunching the numbers gives a monthly cost of $1.1M for using Provisioned Capacity on demand and $3.9M if you use the full on demand model—a massive penalty if you pick the wrong pricing model for your use case.

Now, clearly before you got to a DynamoDB bill of $500k per month you would be having a hard think about how to change your application design to reduce costs. An obvious way to do this would be to group up several individual metrics into a single DynamoDB item thus reducing our number of writes. In doing this we have to be aware of the limit of 1KB per write which equates to 10 of our 100 byte items. It seems unlikely we can come up with a reasonable design that both groups at write time and groups in a way that reduces the number of reads we need to do so we'll leave the number of reads consistent. Our storage requirements will also be reduced due to less item indexing overhead giving us the following cost estimate:

Item	Required Volume	Cost per Item per Month	Total Cost
WCU	207,900	$0.218696	$45,467
RCU	1,080,000	$0.025	$27,000
Storage	29,848	$0.25	$13,680
Backup	29,848	$0.1	$2,985
Total			**$89,132**
Support			$2,795
Spark Cluster			$2,550
Grand Total			**$94,477**

So, with some re-architecting effort we've brought our cost down to the ~50% more than our i3.2xl Managed Cassandra cluster and still more than double our estimated cost using i3en.xl instances. This is summarized in the following chart:



It should be fairly clear from the costing above that the high volume of writes in our use case made it a particularly good fit for Cassandra but there is still plenty of room for other use cases to save cost using Managed Cassandra. The 50% cost advantage of the i3en.xl scenario significantly broadens the use case where Managed Cassandra will save you money.

Of course, cost is only one aspect to consider. Other advantages of Cassandra over DynamoDB include:

- Cassandra Apache Foundation Open Source, guaranteeing you no vendor lock-in and also providing compatibility with a wide range of open source tools
- Cassandra provides tunable consistency not just within the local data center but across multiple active-active regions.
- DynamoDB's capacity is limited by partition with a maximum of 1,000 write capacity units and 3,000 read capacity units per partition. Cassandra capacity is distributed per node. In this example above we are processing up to 55,000 writes per second per node all of which could happily be directed to a small number of partitions at any one time.
- Cassandra's CQL query language provides a simple learning curve for developers familiar with SQL.
- DynamoDB only allows single value partition and sort (called clustering in Cassandra) keys while Cassandra support multi-part keys. A minor difference but another way Cassandra reduces application complexity.
- Cassandra supports aggregate functions which in some use cases such as this one can provide significant efficiencies.

If you'd like to compare the cost of running your application on Cassandra vs DynamoDB then please get in touch with Instaclustr and we'll be happy to work with you to optimize a cost model specific to your requirements.

About Instaclustr

Instaclustr delivers reliability at scale through our integrated data platform of open source technologies such as **Apache Cassandra®**, **Apache Kafka®**, **Apache Spark™**, **Elasticsearch**, and **Redis**. Our expertise stems from delivering more than 70 million node hours under management, allowing us to run the world's most powerful data technologies effortlessly.

We provide a range of managed, consulting, and support services to help our customers develop and deploy solutions around open source technologies. Our integrated data platform, built on open source technologies, powers mission critical, highly available applications for our customers and help them achieve scalability, reliability, and performance for their applications.

© Instaclustr 2020
This document is provided for information purposes only, and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor is it subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. Instaclustr specifically disclaims any liability with respect to this document, and no contractual obligations are formed either directly or indirectly by this document.

Apache Cassandra®, Apache Spark™, Apache Kafka®, Apache Lucene Core®, Apache Zeppelin™ are trademarkS of the Apache Software Foundation in the United States and/or other countries. Elasticsearch and Kibana are trademarks for Elasticsearch BV, registered in U.S. and in other countries.