CumuLogic

MongoDB-as-a-Service Powered by CumuLogic Platform

November 2013

Table of Contents

Introduction

CumuLogic has built is a suite of cloud services for private clouds bringing the functionality of Amazon Web Services to any cloud based on OpenStack, Apache CloudStack, Citrix CloudPlatform and VMware vCloud Director.

CumuLogic's platform provides an integrated suite of services including Relational Database-as-a-Service (MySQL-as-a-Service), NoSQL Database-as-a-Service (MongoDB-as-a-Service), Memcached-as-a-Service and many more. Cloud administrators can deploy CumuLogic Controllers and integrate with their existing private or public clouds by simply pointing to their clouds API end points.

With CumuLogic's platform, Cloud Providers can deliver high value NoSQL MongoDB Database-as-a-Service to end customers who prefer to have flexibility and control over the data tier architecture and want to optimize their architecture based on the application workload.

This paper discusses MongoDB NoSQL Database-as-a-Service for IaaS clouds detailing best practices, architectures for high availability, scalability and sharing performance benchmarks baseline numbers to expect from a given type of storage and compute resources. Please note these numbers may vary drastically based on your storage, networking and compute infrastructure.

NoSQL Databases

NoSQL databases are becoming increasingly popular with developers for mobile and web applications as they provide flexibility, scalability and performance that most next generation applications need.

There are several categories of NoSQL databases:

- Document stores such as MongoDB, Couchbase and BaseX. These types of databases allow data to be stored as JSON documents.
- Key/Value pair NoSQL database such as Riak, Redis and Cassandra, which store data as key/value pairs.
- Graph databases such as Neo4j and DEX.

CumuLogic NoSQL Database Service

CumuLogic's NoSQL database service provides a quick and easy way to access fully managed NoSQL database instances on any cloud, private or public. CumuLogic platform currently supports MongoDB database service. Users can launch a dedicated MongoDB service and optimize it for their particular workload if needed. CumuLogic platform manages the database instances from provisioning and monitoring, to replication and sharding automatically, and backup/restore. Each instance of MongoDB is optimized for a given size of the database node, so users get the highest price/performance value. MongoDB instances can be scaled on-demand by adding new replica nodes without having to shutdown the database.

CumuLogic platform users have visibility into key metrics so developers can take appropriate actions to fine tune performance. In addition the platform performs automated backups at defined frequencies and supports Point-in-Time restore for faster recovery from failures.

MongoDB Configurations

MongoDB is a highly scalable database and has built-in high availability and data redundancy. A MongoDB single instance can be used for small experimental projects or can be used as multi-node instance for data durability. For smaller applications or development experimentation, it's sufficient to use a single MongoDB instance and co-locate the application on the same instance. However, for production purposes, there are two possible architectures: Replica Sets and Sharded Cluster (with or without Replica Set), both supported by CumuLogic's platform.

1. **Single node MongoDB for Development and Experimentation**

   MongoDB can be provisioned as a single node database server for development and other environments, which are not sensitive to failures and data loss. It is possible to add replicas to an existing primary node and convert it to replica sets for redundancy purposes.

2. **Replica Set for Data Durability and Failover**

   Database replication ensures redundancy, backup, and automatic failover. Replication occurs through groups of servers known as replica sets. A Replica Set is a cluster of mongod instances (primary daemon process for MongoDB server which handles data requests, manages data formats and performs background operations). MongoDB instances replicate data amongst each other asynchronously. A Replica Set can have two or more replicas but any one of them can be a primary. In case the primary fails, the rest of the nodes in the replica set will *elect* a primary amongst themselves. MongoDB has a limit of up to 12 nodes in a given Replica Set. In case of multi-availability zone (AZ) deployments, CumuLogic platform will place most replicas in the primary availability zone to ensure a successful election of a primary mongod.

   In CumuLogic platform, replica sets are configured for strict consistency, which means all read operations are handled only by the primary node. Secondary nodes replicate the data as soon as write operations are completed by the primary.

In MongoDB, it is possible to direct read operations to secondaries, however this feature is not currently supported by CumuLogic platform.

3. Sharding for Scalability, Higher Performance, Durability and Failover

Sharding distributes data across multiple servers for scalability and data durability. A sharded cluster may consist of multiple server nodes or multiple replica sets. MongoDB uses sharding to distribute data to one or more replica sets for scalability and data durability. The major difference between replica set and sharding is that sharding distributes chunks of data to all nodes or replica sets that are part of the sharded cluster providing better database performance and scalability. In Replica Sets, all the nodes store the same data for redundancy.

CumuLogic platform enables sharding on replica sets for higher throughput and data durability. Sharding is enabled per database or collection basis, and users must select a shard key in the document to distribute data between the shard nodes. MongoDB distributes documents according to ranges of values in the shard key. A given shard holds documents for which the shard key falls within a specific range of values. Shard keys, such as index (in RDBMS), can be either a single field or multiple fields. Within a shard, MongoDB further partitions documents into chunks. Each chunk represents a smaller range of values within the shard's range. When a chunk grows beyond the configured chunk size (default is 64MB), MongoDB splits the chunk into smaller chunks, also based on ranges in the shard key.

To scale out MongoDB sharded cluster, it's possible to add new nodes or Replica Sets and improve performance for read and write operations.

MongoDB Internal Building Blocks

MongoDB has three processes, which manage the internal data storage, query routing and replication operations.

Mongod

Mongod is the primary MongoDB database process. It manages the internal data storage system. CumuLogic MongoDB services starts one mongod process on each node in the sharded cluster or replica set.

mongos

mongos for "MongoDB Shard," is a routing service for MongoDB shard configurations that processes queries from the application layer and determines the location of this data in the sharded cluster. Applications talk to mongos processes and not to the mongod. CumuLogic platform requires at least three nodes to enable sharding and will start mongos on each node along with the configserver. Mongos themselves don't store any data and hence loss of mongos process will not result in data loss.

configserver

Configserver process is a mongod process, which stores the cluster metadata. Mongos processes refer to configservers for the state information. CumuLogic platform uses three configservers for each sharded MongoDB cluster.

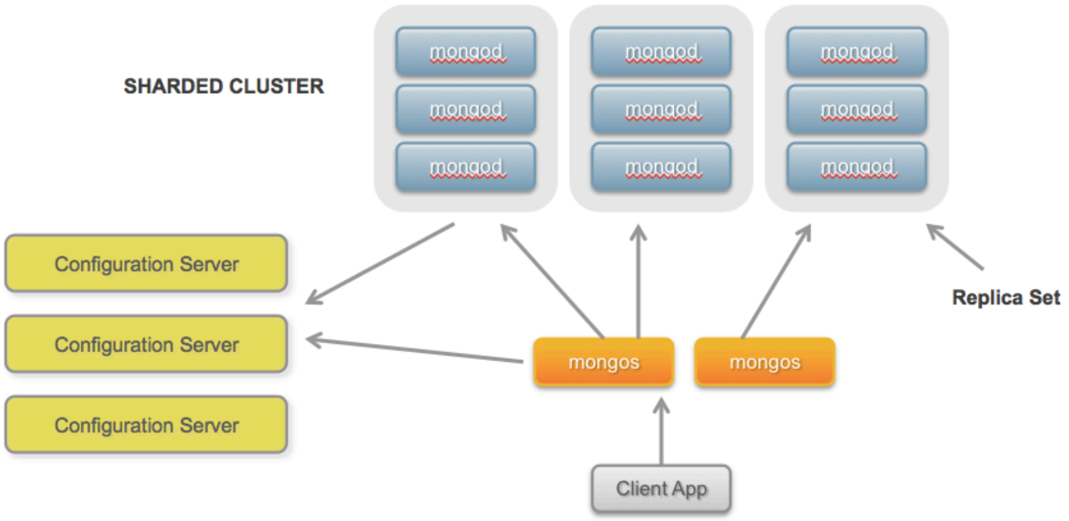


Figure 1 MongoDB Components

MongoDB Deployment Architectures

CumuLogic MongoDB service is designed to provide an easy-to-use and highly optimized MongoDB replica set or sharded cluster for variety of workloads. Users can get MongoDB up and running and ready to use in a couple of minutes on any target IaaS cloud, public (Amazon, HP Cloud or Rackspace) and private clouds (Citrix CloudPlatform, Apache CloudStack, OpenStack, Eucalyptus, VMware vCloud). They key differentiation for CumuLogic MongoDB service is the price/performance ratio on any cloud that is supported. CumuLogic platform optimizes MongoDB servers and tunes OS kernel, file system parameters and storage for optimum performance for a given size of MongoDB node. In addition, CumuLogic platform allows the end user to further fine tune the performance based on specific workload types.

Note: Please refer to the latest published benchmarks for IOPS performance of each size of MongoDB instance. CumuLogic platform delivers higher performance and low *latency from $100/month upwards based on the size of the MongoDB database depending on the public cloud.*

Replica Set for Data Durability

The default deployment pattern for MongoDB is a replica set. To launch a replica set, select *Launch Instance* from the NoSQL Database Service menu and select the number of replica nodes (or desired IOPS). A replica set requires at least three nodes, one of which is configured as primary; one is secondary and the third one is used as an arbiter to select a primary server in case of failures. Although, one can select any number of nodes in replica set (up to 12), it's advisable to use an odd number of nodes for easier failure recovery and primary node election. In case an even number of nodes is used, CumuLogic platform will configure one node as a non-voting member or add additional arbiter node. Non-voting members participate in data synchronization just like voting members in a replica set except that they don't participate in the process of electing a new primary node during failover.

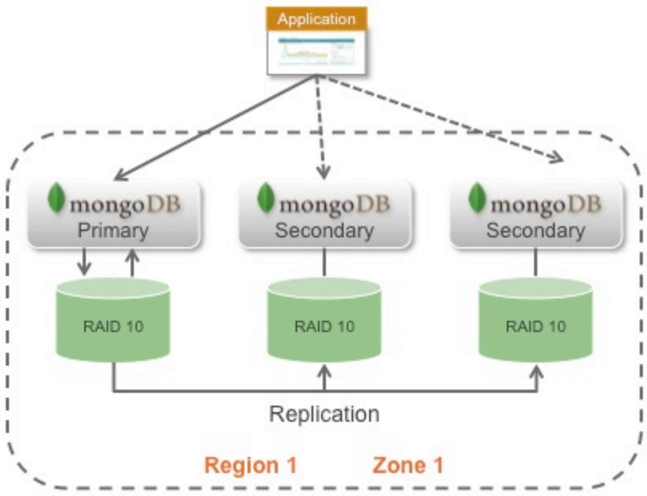


Figure 2 Replica Set in a Single Availability Zone

By default, all nodes in a replica set are provisioned in the same Availability Zone. If multiple Availability Zones are available, users must select an alternate Availability Zone to distribute nodes across the zones. Users can choose to allocate one or more nodes in either Availability Zone. CumuLogic platform will automatically provision nodes in a secondary zone as non-voting members to avoid delays in electing a primary node when in failover. An arbiter node is always provisioned in the primary zone.

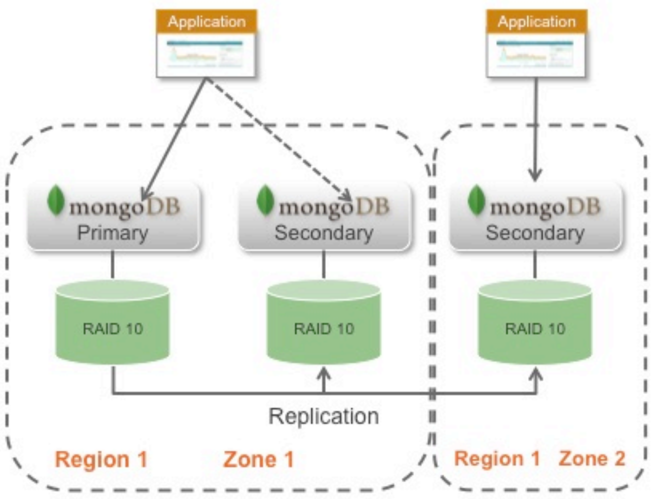


Figure 3 Replica Set in a Multiple Availability Zone

Service Summary

1. A MongoDB replica set must be used for data durability and redundancy.
2. CumuLogic platform provisions MongoDB replica set in a single availability zone (default).
3. Users must choose multi-AZ option to provision nodes in the secondary zone.
4. An odd number of nodes are advisable for replica set. In case an even number of nodes is chosen, one node is configured as a non-voting member of the set.
5. For multi-AZ deployment, any nodes in the secondary Availability Zones are configured as non-voting members of the replica set to avoid delays due to possible network latencies in choosing a primary node in case of failure recovery.
6. All replica nodes are configured for eventual consistency. This means that there may be a delay in data synchronization between the primary and the secondary nodes.

   (Important Note: All read operations are performed from primary node by default. However, it's possible to configure your mongo client drivers to set preferences to read from secondary nodes. Since, the replica sets use eventual consistency, the data read from the secondary node may be stale depending on the lag between the primary and the secondary node. If strict consistency is desired between the replica nodes, you can use "write concern" in the Mongo client. Write concern can guarantee that data is written to all nodes.)

7. In case of failure of a primary node, secondary nodes will elect a primary node in the primary zone. If one of the secondary nodes becomes unavailable, a new secondary node will be provisioned and synchronized with the primary.

8. System will perform automated backups once every 24 hours. Automated backups can be scheduled at any time of the day to avoid any performance impact to the database operations during a busy period.
9. Data can be restored from the latest available full database snapshot. You can launch a new database instance from any existing snapshot.
10. Monitoring and performance data is available on the console.
11. To change the tuning parameters, users can create Parameter Groups with custom values for the parameters exposed by database service. Default optimized parameters are available for read heavy, write heavy and update heavy workloads.

Sharded Cluster for Scalability

Sharding in MongoDB provides scalability for reading and writing since it distributes the data and the database operations across multiple nodes. MongoDB partitions the data across multiple nodes and scales out by adding new nodes in a running sharded cluster. Sharding also allows MongoDB to increase the data in the working set or the data in the RAM.

Sharding can be configured on a set of single node instances or on replica sets, which means each shard can also be a replica set. Shards across replica sets provide higher scalability and reliability and must be considered for large-scale production deployments. Shards can be configured across multiple availability zones if supported on a cloud.

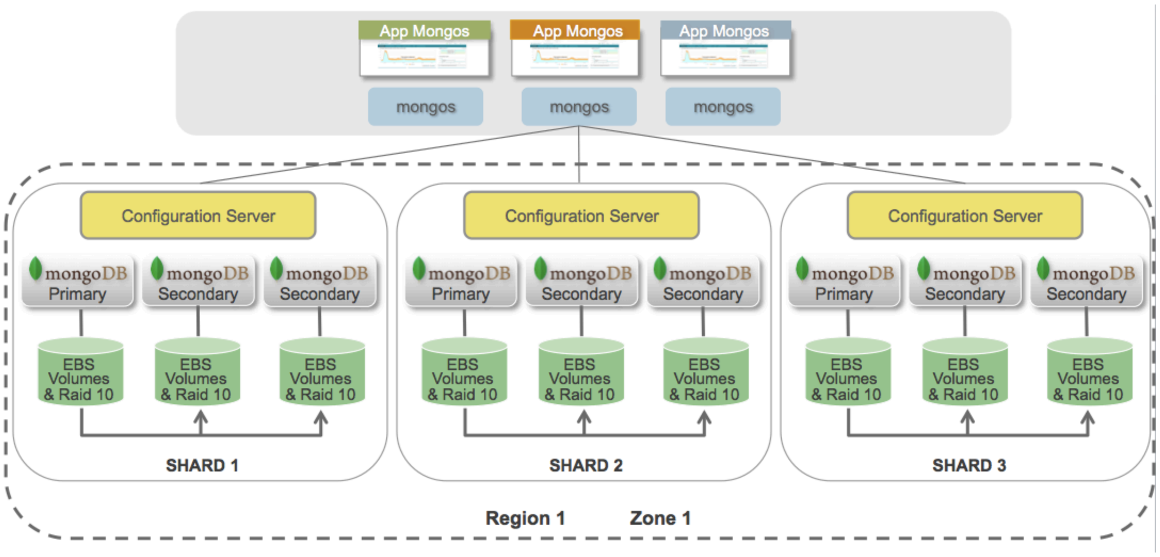


Figure 4 MongoDB Sharded Cluster

Sharding is enabled on per database basis, and once enabled, MongoDB distributes the collections across the shards. After enabling sharding, you can choose one or more collections to shard by providing a shard key.

Note: Choosing an appropriate shard key is critical to the overall performance and scalability of MongoDB shards. Here are some guidelines to choosing a shard key

Service Summary

- Use a sharded cluster for large-scale production deployments when you need scale and redundancy.
- To launch a sharded cluster, select "Create Database" from MongoDB Service menu and select "Sharded cluster" option. Select the number of nodes or replica sets in a shard. If you choose replica set, you must choose number of nodes in each replica set. When you choose this option, the system will launch replica sets and configure them as part of a sharded cluster.
- Once the cluster is provisioned, from the actions menu, choose the database to shard and provide a shard key

Guidelines for Choosing a Shard Key

The performance, capabilities and functioning of MongoDB greatly depends on the shard key. The key to use for shards depends on your database schema and the type of queries in the workload. Here are some guidelines on choosing the appropriate shard keys:

- Divisibility rule: MongoDB writes data in chunks (the default is a 64MB chunk on each node), so in order for chunks to be evenly distributed across all the shard cluster nodes, choose a key which will prevent chunk data to be split unevenly. Consider choosing the key of the field which has high cardinality (range of values across the datasets)
- Write scalability: Choosing the key with high degree of *randomness* can help increase the write scalability and hence performance of the sharded database clusters.
- Query isolation: For executing queries in the sharded MongoDB cluster, as shown in the figure 1 above, mongos process uses metadata from the configserver to route queries to appropriate mongod server instance. If the query doesn't include shard key, mongos must query the metadata from all configservers to route the query to appropriate mongod instance. This impacts the performance of the sharded cluster negatively. Choosing a shard key, which is most commonly used in each query, is key to highly scalable sharded cluster.

  Note: Read more about schema optimization at MongoDB Data Modeling

To launch a new sharded cluster, select "*Launch Instance*" from MongoDB Database Service menu on CumuLogic Console and provide details on number of shard nodes or replica sets.

Performance Optimization

CumuLogic MongoDB service instances are optimized for the target cloud, which means out of the box performance of each instance will be optimized to deliver the highest possible IOPS and lowest latency based on the size of the database instance.

Storage Optimization

Storage impacts the latency of read and update operations, and since on most clouds, the block storage available is mostly EBS volumes, the latency charts will vary from cloud to cloud based on optimized volumes, availability of SSD drives or guaranteed number of IOPS (example Optimized IOPS volumes on Amazon EC2). CumuLogic platform configures the persistent storage for RAID 10 configuration to provide better performance and redundancy. RAID 10 provides the best performance for larger sharded databases on replica sets.

MongoDB Optimization

MongoDB internal processes perform disk I/O and are configured for better performance. Most of these parameters can be changed to suit a particular workload by creating CumuLogic Parameter Groups. Parameter Groups are a set of configuration parameters that impact the performance of the database and must be tweaked for optimal operations. Some of the parameters, which can be configured by users, are:

Journal (Transaction log sync)

- MongoDB stores each database operation in a journal or transaction log file before committing to the disk. This flush happens every 100ms by default. This can be increased for read heavy workloads without risking loss of the data because of failures. Increasing this value by too much may negatively impact the performance when the there are excessive update operations, in which case every flush to the disk will take much longer to finish. The journal activity is available in monitoring charts for users to visualize the pattern and optimize it to suit the workload.

Filesystem and syncDelay

- CumuLogic allows users to configure another important parameter called syncDelay, a configuration parameter that determines how frequently MongoDB syncs the in-memory data to the filesystem. By default, this interval is set to 60ms and can be configured based on the type of database operations. If your database operations are mostly read-only, this parameter can be increased to lower the frequency of sync and reducing the disk I/O operations. On the other hand, if this parameter is too high for write heavy disk operations, there may be a longer sync operation that could negatively impact the overall IOPS of the database instance.

Object Size and Page Faults

- Close attention should be paid to the size of the object that you are performing operations on. If the size of the objects is bigger than the amount of free memory available on the node instance, the system will cause page faults, which require MongoDB to read the pages from the disk into the memory. Random I/O operations caused because of page faults can negatively impact performance on the database and provide low IOPS on a given size of the database instance. In order to avoid page faults, choose the instance size when launching the database instance. Typically, a small instance on most clouds is 2GB RAM, and if you encounter too many page faults, you may want to choose a larger instance size. Number of Page Faults on a given node is available in Monitoring charts.

Disable Services

- MongoDB uses internal services such as HTTP Interface Service and Object Validation Services that can be disabled to improve performance. CumuLogic disables these services by default and user can use Parameter Groups to enable them as needed. The options *noobjcheck* and *nohttpinterface* disable these services. CumuLogic platform also disables MongoDB option for pre-allocating the storage during database startup.

Slow Query Optimization

- MongoDB profiler can help identify long running queries. By default, the profiler highlights the queries that run longer than 100ms. These queries can be modified to improve the performance. The parameter *slowms* in Parameter groups can be edited to identify queries, which take longer than any interval you need to target the query's runtime.

Instance Size and IOPS

- Based on the cloud type, storage and networking, CumuLogic platform provides visibility into the approximate number of IOPS that an instance can deliver. We use ycsb benchmark to test the scalability of MongoDB on different instance size.

  Note: *These are only approximate numbers and may have a large variation between clouds. These should not be considered as final numbers to compare performance.*

Sample IOPS Throughput chart for Heavy Updates Operations on Replica Set

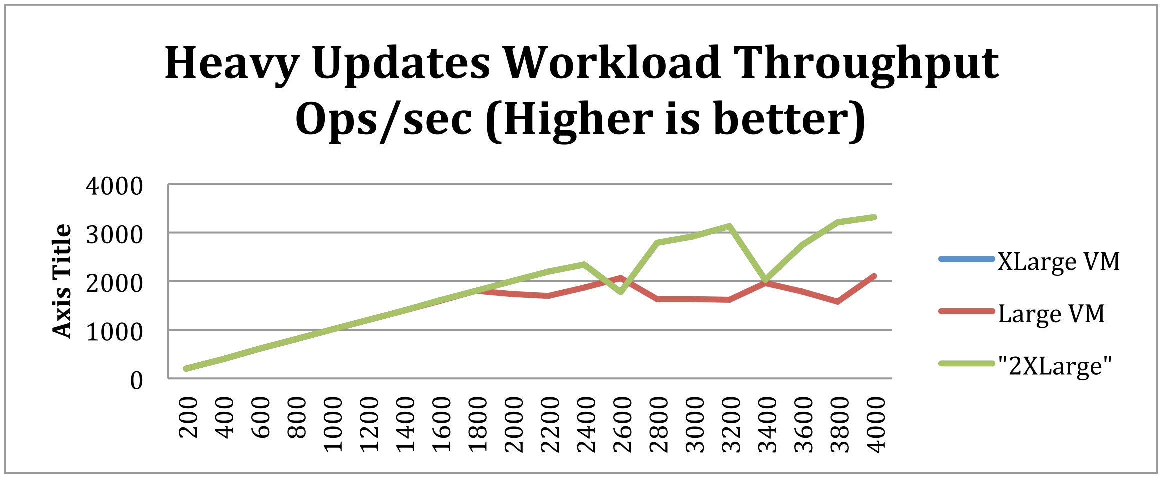


Figure 5 Example IOPS Chart for Heavy Updates

Sample IOPS Throughput chart for Heavy Read Operations on Replica Set

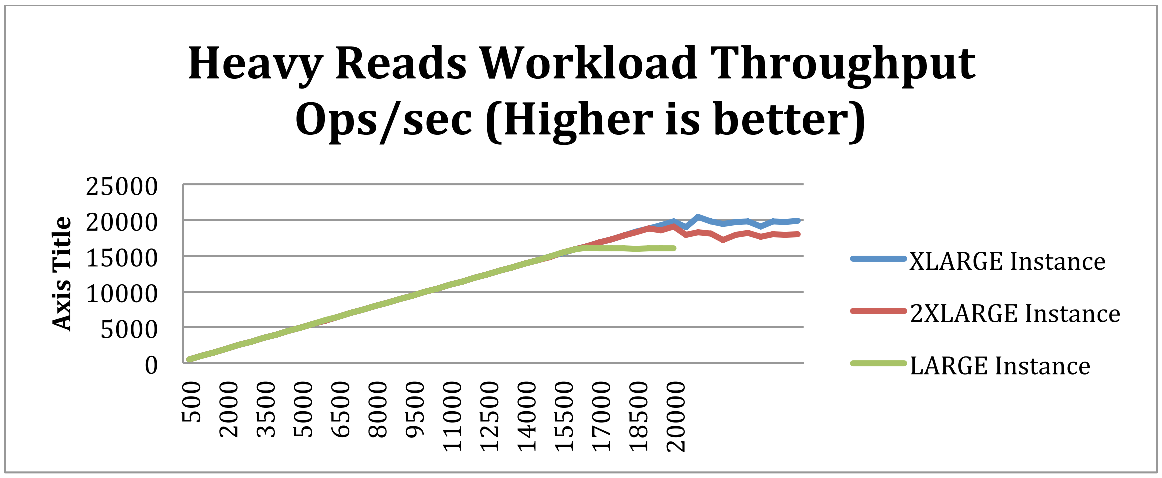


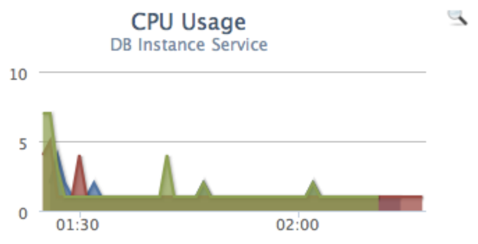
Figure 6 Example IOPS Chart for Heavy Reads

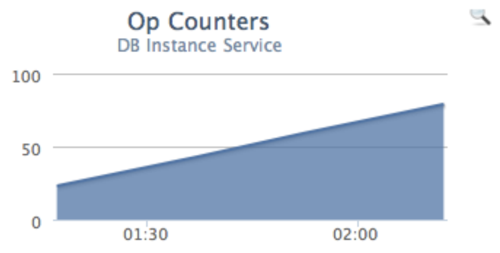
Operations

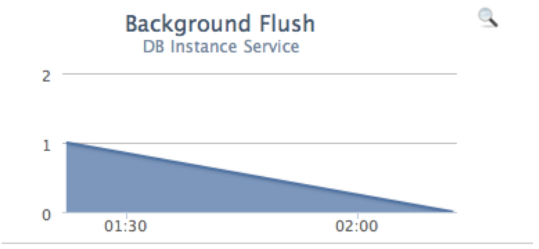
Monitoring and Self Healing

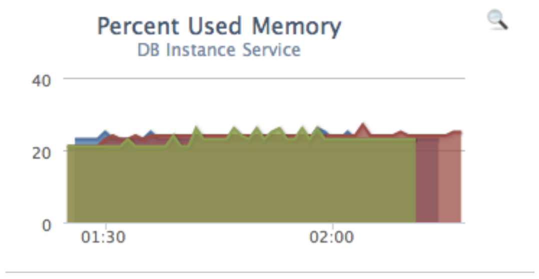
CumuLogic MongoDB Service is fully monitored and the most important metrics are available as real time charts on the user Console. No additional monitoring tools are required to monitor MongoDB instances. In case of any failures, the system will try to restore the replica set or sharded cluster. Notifications are sent to the user's email address regarding any performance degradation, failures and recovery process.

Sample Monitoring Charts









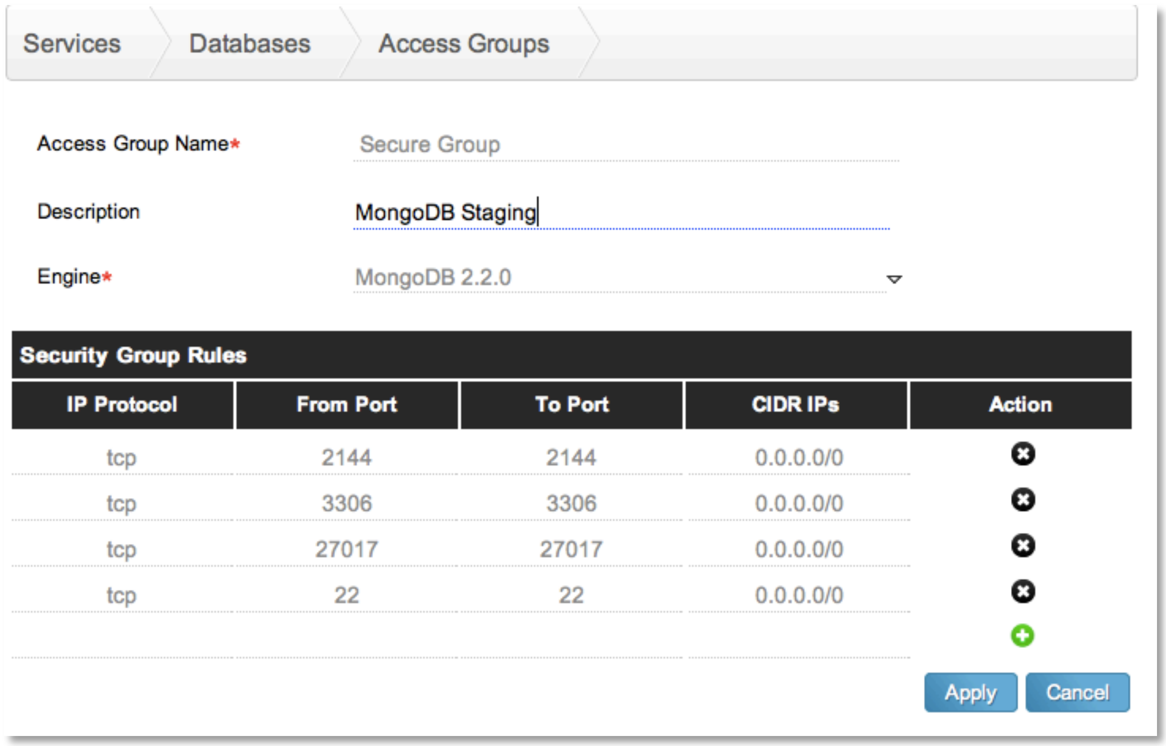
Backup and Restore

CumuLogic database service uses persistent storage for datastore and initiates a full database snapshot once every 24 hours. CumuLogic also backs up the opslog file periodically making it possible to do a point-in-time restore. In case of a single node database instance, database backup will impact database performance. For this reason, the back up cycle can be set to a desirable, low-demand time of day. In case of replica sets, a secondary node is used for backup purposes to minimize the performance impact on the primary node. The backup retention period can be changed from default one day.

To restore data, simply choose any available snapshot and restore a new database instance. Either full database can be restored or point-in-time restore can be used to recover to a given point of the day.

Security and Access Control

To access the database instance, you must first allow ingress from your application server's IP address on a defined port. To enable access, you can create *Access Group* and apply that Access Group to the running instance of MongoDB instance. Access groups' work like firewall settings to open ports for ingress IP address range.



Access Group Name*	Secure Group
Description	MongoDB Staging
Engine*	MongoDB 2.2.0

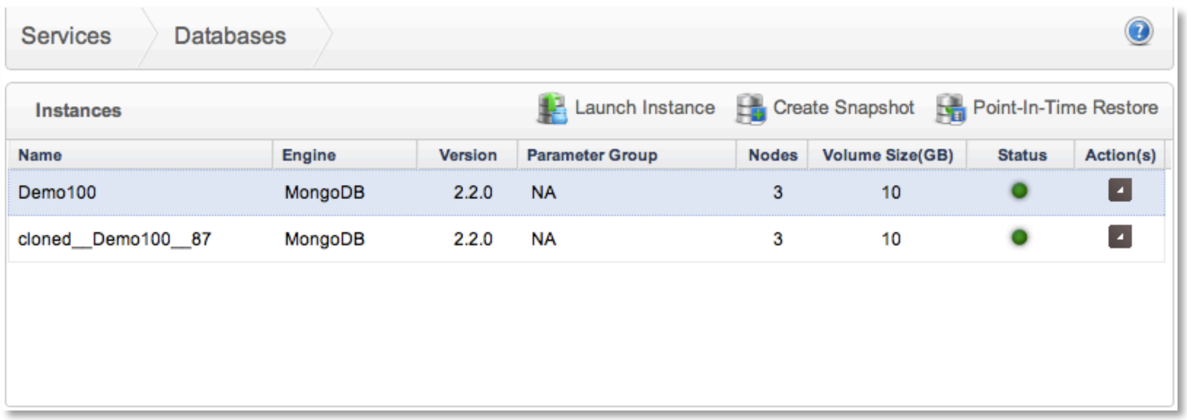
Security Group Rules

IP Protocol	From Port	To Port	CIDR IPs	Action
tcp	2144	2144	0.0.0.0/0	✖
tcp	3306	3306	0.0.0.0/0	✖
tcp	27017	27017	0.0.0.0/0	✖
tcp	22	22	0.0.0.0/0	✖
				✚

Figure 7 Configuring Access Group

Cloning

CumuLogic platform provides single-click operation to clone an existing database instance or replica set. To clone an instance, choose the *Clone* action from the instance action list. Cloning an instance creates a copy of an existing instance or replica set and restores full database to the clone.



Name	Engine	Version	Parameter Group	Nodes	Volume Size(GB)	Status	Action(s)
Demo100	MongoDB	2.2.0	NA	3	10	●	
cloned__Demo100__87	MongoDB	2.2.0	NA	3	10	●	

Figure 8 Cloning MongoDB Replica Set

API (Coming Soon)

CumuLogic MongoDB API provides full control of MongoDB instances and operations through the RESTful API.

MongoDB-as-a-Service for Private Clouds

To run MongoDB service inside your datacenter, you can download CumuLogic Installer. To run CumuLogic platform, you only need access to a single VM (or you can install it on bare metal as well) with at least 16GB RAM and 50GB disk space. CumuLogic platform integrates with Citrix CloudPlatform, Apache CloudStack, OpenStack, Eucalyptus, and VMware vSphere and vCloud environments. Performance of MongoDB and other CumuLogic services depend on the environment type and block storage available.

Summary

CumuLogic's platform simplifies provisioning, management and operations of MongoDB database clusters on any cloud – private or public. Our goal for developing the orchestration of MongoDB service is to allow users to *spin up MongoDB replica sets or sharded clusters on any preferred cloud in less than three clicks* and free up users from the complex operations of managing complex database architectures. We work closely with Cloud Providers to tune the infrastructure, storage and networking layers so they can bring you MongoDB at a very attractive price/performance.

CumuLogic can be setup to manage MongoDB infrastructure in private clouds or VMware vSphere environments.

To test drive MongoDB-as-a-Service Powered by CumuLogic, please choose a cloud provider partner from cumulogic.com/partners or download CumuLogic Cloud Services software for private clouds.

About CumuLogic

CumuLogic is a software provider that enables enterprises and cloud providers to develop and deploy applications in public, private and hybrid cloud environments. CumuLogic is redefining Platform-as-a-Service (PaaS) to include individual modular cloud services, including Database-as-a-Service, Elastic Cache, Elastic Load Balancer and more.

CumuLogic was founded by ex-Sun Microsystems employees passionate about cloud computing. James Gosling, the creator of Java, Bill Vass, former CIO of Sun Microsystems and President of Sun Federal, and Bud Albers, former CTO of The Walt Disney Company lead CumuLogic's Technical Advisory Board.