Backup and its Role in Disaster Recovery

MongoDB 3.2
December 2015

mongoDB

Table of Contents

Introduction

Data is the core currency in today's digital economy. In the same way that central banks take extensive measures to safeguard this vital asset, organizations need to take every practical measure to protect their data. Yet, according to industry research, 43% of companies that experience major data loss incidents are unable to reopen business operations. [1]

Data loss incidents can take a variety of forms, but when it comes to database technology, they generally fall into two categories. The first category is catastrophic failure and the second is human error.

Catastrophic failure includes natural disasters and any other scenario that permanently destroys all the nodes in your production system. If you keep all your servers in the same data center, a fire or flood that destroys them would qualify. This is typically what we imagine when we implement a backup strategy.

While they may be less newsworthy than catastrophic failure, the reality is that human error or flawed processes account for the vast majority of IT outages.[2] Humans introduce application bugs, deliberately hack into systems or accidentally delete data. A bad code release that corrupts some or all of the production data is an unfortunate but common example. In the case of human error, the errors introduced will propagate automatically to the replicas, often within seconds.

A backup and recovery strategy is necessary to protect your mission critical data against these types of risks. With such a strategy in place, you'll gain peace of mind knowing that in the event of a failure, you're ready to restore business operations without data loss. You'll also satisfy important regulatory and compliance requirements by demonstrating that you've taken a proactive approach towards data safety and business continuity.

Taking regular backups offers other advantages as well. The backups can be used to create new environments for development, staging or QA without impacting production. This practice enables development teams to quickly and

1. TechNewsWorld
2. GigaOm

easily test new features, accelerating application development and ensuring smooth product launches.

Database systems are the most critical components to safeguard, yet often the most complex to back up and restore properly[3].

A robust database backup strategy includes a mix of technology and process to prepare for failures that might result in the complete disruption or loss of the business.

Below we'll outline considerations when preparing a backup strategy, and then specific approaches for backing up MongoDB.

Backup Considerations

All backup systems capture a snapshot of your system from a past moment in time. A critical feature of the backup is that it is frozen forever. Restoring the backup snapshot rolls back the clock on the unexpected event that caused the loss of data across your system.

In preparing a backup strategy, organizations typically start by evaluating their recovery point objective (RPO) and recovery time objective (RTO).

The RPO indicates how much data the business is willing to lose in the event of an incident, while the RTO indicates how quickly it will recover.

As not all data is created equal, this RPO and RTO should be evaluated on an application-by-application basis. For example, you'll likely have different requirements for your mission critical customer data than you will for your clickstream analytics.

Your requirements for RTO and RPO will drive the economic and performance cost of maintaining backups. For example, if you are prepared to lose up to a year's data in the event of a fire, you could take a snapshot once a year and store the backups on physical, disconnected media in a remote location.

This would be a relatively inexpensive solution. However, when disaster strikes, you will need to physically mail the media to your datacenter.

3. StorageNewsletter.com

The backup may be a year old, could take days to arrive and even longer to restore to production. On the other hand, if you never want to lose more than a minute or two of data or incur more than a few minutes of downtime, you will need a continuous backup solution with point-in-time recovery.

There is a tradeoff between achieving a better RTO and reducing the probability that a disaster also destroys your backups. Keeping your backup snapshots far from the primary database servers, both logically and physically, lowers the likelihood that they will be destroyed at the same time as your database, but increases the recovery time.

You'll also want to consider the ongoing maintenance, cost and performance impact of the backup system. While backups are crucial for the safety of your organization's mission critical data, they need to be evaluated in the context of the overall resource utilization across of production system.

Figure 1 outlines the key considerations when evaluating backup strategies.

MongoDB Backup Strategies

There are three main strategies for backing up MongoDB:

- `mongodump`, a utility bundled with the MongoDB database
- Filesystem snapshots, such as those provided by Linux LVM or AWS EBS
- MongoDB Ops Manager, a management platform that makes it easy for operations teams to provision, monitor, back up, and scale MongoDB. Ops Manager is included with MongoDB Enterprise Advanced, and provides continuous backup and point-in-time restore for MongoDB. Those interested in a managed, cloud-based backup solution should consider MongoDB Cloud Manager, which provides continuous, online backup and point-in-tume restore for MongoDB as a fully managed service.

Below we'll outline these different approaches and the benefits and drawbacks of each.

Key Considerations	
Recovery Point Objective	The amount of data your business is willing to lose in the event of a disaster.
Recovery Time Objective	The amount of time it takes to recover. This includes both the time it takes to retrieve the backup and put it into production. In scenarios where restoring from a backup is necessary, it is likely that you will incur some downtime.
Isolation	Backups should be separate from the production system to ensure that any disruptions to production do not also impact the backup.
Performance Impact	Backup techniques have varying impact on the performance of the running database. Some backup solutions degrade database performance enough that you may need to schedule backups to avoid peak usage or to occur within maintenance windows. You may decide to deploy new secondary servers just to support backups.
Restore Process	A backup system is only a good as your ability to restore from it. One of the critical components of any backup strategy is not only having the data but having ability to restore it, and doing practice runs of your restores to ensure that they work in the event of a “data emergency”.
Sharding	Backing up a distributed database system, such as a MongoDB sharded cluster, presents an additional layer of complexity. To achieve a truly consistent backup, many approaches require write activity to be paused across the system.
Deployment Complexity	While backup is critical in the case of a disaster scenario, it's not your core business. Ideally you want a backup strategy that is easy to setup and maintain over time so that you can focus on your business.
Flexibility	Partial backup strategies will provide you with the flexibility to filter out data so that you aren't utilizing resources backing up non-mission critical components of your system. Similarly, during the restore process, you may want the flexibility to recover only certain components of the data to accelerate time to recovery. Finally, an incremental backup strategy will only backup the parts of the data that have changed since the last snapshot, making it a more efficient and flexible strategy than always taking complete backups.

Figure 1: Key considerations when evaluating backup strategies

Backup Strategy #1: mongodump

`mongodump` is a tool bundled with MongoDB that performs a live backup of the data in MongoDB. `mongodump` may be used to dump an entire database, collection, or result of a query. `mongodump` can produce a dump of the data that reflects a single moment in time by dumping the oplog entries created during the dump and then replaying it during `mongorestore`, a tool that imports content from BSON database dumps produced by `mongodump`.

`mongodump` is a straightforward approach and has the benefit of producing backups that can be filtered based on your specific needs. Since `mongodump` simply creates BSON, it works with all of MongoDB's supported storage engines. That is, you can use `mongodump` on deployments leveraging any supported storage engine; and you can restore to deployments running any storage engine. To learn more about MongoDB storage engines, please review the documentation.

While `mongodump` is sufficient for small deployments, it is not appropriate for larger systems. `mongodump` exerts too much load to be a truly scalable solution. It is not an incremental approach, so it requires a complete dump at each snapshot point, which is resource-intensive. As your system grows, you should evaluate lower impact solutions such as filesystem snapshots or Ops Manager.

In addition, while the complexity of deploying `mongodump` for small configurations is fairly low, the complexity of deploying `mongodump` in large sharded systems can be significant – particularly if a consistent snapshot across the cluster is required.

Backup Strategy #2: Snapshotting the Underlying Files

You can back up MongoDB by copying the underlying files that the database uses to store data. To obtain a consistent snapshot of the database, you must either stop all writes to the database (e.g., using `db.fsyncLock()`)and use standard file system copy tools, or create a snapshot of the entire file system, if your volume manager supports it.

For example, Linux LVM quickly and efficiently creates a consistent snapshot of the file system that can be copied for backup and restore purposes. If using MMAPv1, to ensure that the snapshot is logically consistent, you must have journaling enabled. Regardless of storage engine, any journal must be stored in the same file system as the data files.

Because backups are taken at the storage level, filesystem snapshots can be a more efficient approach than `mongodump` for taking full backups and restoring them. However, unlike `mongodump`, it is a more coarse approach in that you don't have the flexibility to target specific collections in your backup – though database files can be separated when using WiredTiger with `directoryPerDB` set to true. This may result in large backup files, which in turn may result in long-running backup operations. Additionally, a filesystem snapshot can only be restored to a MongoDB deployment running the same storage engine as the deployment from which the snapshot was taken.

To implement file system snapshots requires ongoing maintenance as your system evolves and becomes more complex. To coordinate backups across multiple replica sets, particularly in a sharded system, requires devops expertise to ensure consistency across the various components.

Backup Strategy #3: Ops Manager and Cloud Manager

Ops Manager is a management platform that runs in your data center and provides continuous backup and point-in-time restore for MongoDB. A lightweight agent runs within your infrastructure and connects to the configured MongoDB instances. The agent performs an initial sync to Ops Manager and then tails the oplog of a replica set's primary. After the initial sync, the agent streams encrypted and compressed MongoDB oplog data to Ops Manager so that you have a continuous backup.

By default, Ops Manager takes snapshots every 6 hours and oplog data is retained for 24 hours. Using this data, Ops Manager can create a custom, point-in-time snapshot to which you can restore in case of a catastrophic failure. The snapshot schedule, retention policy, and amount of oplog data can be configured to meet your requirements (e.g., more frequent snapshots, longer retention schedule). You also have the flexibility to exclude non-mission critical databases and collections.

MongoDB periodically writes markers to the oplog in all of the shards which enables consistent, cluster-wide point in time recovery. Because Ops Manager reads only the oplog, the ongoing performance impact to a MongoDB deployment is minimal, similar to that of adding an additional secondary to a replica set.

Integrating with existing storage infrastructure, MongoDB backup files created by Ops Manager can be stored on a standard network-mountable file system. DBAs can automate their database restores reliably and safely using Ops Manager and Cloud Manager. Complete development, test, and recovery clusters can now be built in a few simple clicks.

Those interested in a cloud-based backup solution should consider MongoDB Cloud Manager, which provides continuous, online backup and point-in-time restore for MongoDB as a fully managed service.

Conclusion

A backup and recovery strategy is a critical part of any database deployment. MongoDB offers several options for backing up MongoDB with various tradeoffs associated with each strategy. To learn more about these backup strategies and operational best practices, please visit the links listed below.

- MongoDB Operations Best Practices

- Backup and Restore with MongoDB Tools
- Backup and Restore with Filesystem Snapshots
- Backup & Restore using Ops Manager
- Backup & Restore using Cloud Manager
- Learn More About Ops Manager and MongoDB Enterprise Advanced

We Can Help

We are the MongoDB experts. Over 2,000 organizations rely on our commercial products, including startups and more than a third of the Fortune 100. We offer software and services to make your life easier:

MongoDB Enterprise Advanced is the best way to run MongoDB in your data center. It's a finely-tuned package of advanced software, support, certifications, and other services designed for the way you do business.

MongoDB Cloud Manager is the easiest way to run MongoDB in the cloud. It makes MongoDB the system you worry about the least and like managing the most.

MongoDB Professional helps you manage your deployment and keep it running smoothly. It includes support from MongoDB engineers, as well as access to MongoDB Cloud Manager.

Development Support helps you get up and running quickly. It gives you a complete package of software and services for the early stages of your project.

MongoDB Consulting packages get you to production faster, help you tune performance in production, help you scale, and free you up to focus on your next release.

MongoDB Training helps you become a MongoDB expert, from design to operating mission-critical systems at scale. Whether you're a developer, DBA, or architect, we can make you better at MongoDB.

Resources

For more information, please visit mongodb.com or contact us at sales@mongodb.com.

Case Studies (mongodb.com/customers)
Presentations (mongodb.com/presentations)
Free Online Training (university.mongodb.com)
Webinars and Events (mongodb.com/events)
Documentation (docs.mongodb.org)
MongoDB Enterprise Download (mongodb.com/download)

Appendix: Backup Strategies Compared

The table below outlines the key backup considerations for the three MongoDB backup strategies.

	mongodump	Filesystem Snapshots	Ops Manager
Recovery Point Objective	Limited to snapshot moments.	Limited to snapshot moments, but snapshots are lower overhead and hence often more frequent.	Point-in-time (any moment in time for replica sets; down to 15-minute granularity for entire sharded clusters).
Recovery Time Objective	Requires running `mongorestore`. Latency of `mongorestore` depends on location of the dumps and granularity of what is being recovered.	Depends on latency of bringing the snapshot closer to the production servers and expanding it into a running filesystem.	Depends on how long it takes to transfer the backup snapshot over the network. Restore time will increase if constructing and restoring to a custom, point-in-time snapshot vs. a stored snapshot.
Isolation	Depends on how far the backup snapshots are kept from production.	Depends on how far the backup snapshots are kept from production.	Depends on how far the backup snapshots are kept from production – note that Cloud Manager stores the backups in the cloud.
Performance Impact	Significant if run in online mode on the primary.	Fairly low, depending on implementation.	May add significant load on initial sync. After that, impact is similar to that of a secondary node, typically very low.
Restore Process	Use `mongorestore` to unpack the BSON files.	Typically expand the snapshot onto a new filesystem.	Transfer snapshot from Ops Manager then expand. Ops Manager restores actual database files, not BSON dumps. Ops Manager 2.0 and later includes the option to automate restoring the backups in existing MongoDB processes.
Sharding	Requires scripting across the entire cluster and synchronizing snapshots.	Requires scripting across the entire cluster and synchronizing snapshots.	Support for consistent snapshots of sharded clusters.
Deployment Complexity	Low for small systems, high for large clusters – requires scripting, storage management, and monitoring.	Low for small systems, medium for large clusters – requires scripting, storage management, and monitoring.	Low
Flexibility	Requires custom scripting.	Coarse approach – cannot target specific collections.	Flexibility to exclude non-critical collections, set custom snapshot schedule and retention policy.