Managing FoundationDB at Scale

John Brownlee
Apple

Overview

- FDB has been in use in production for 3 years
- Hundreds of clusters with over 100K total FDB processes
- Managing this footprint requires extensive automation and operational tooling
- Today’s focus will be on our solution for Kubernetes

Overall Design

- All processes run through fdbmonitor
- One fdbmonitor / fdbserver process per pod
- Config maps provide monitor conf and cluster file
- Use anti-affinity to spread processes across hosts
- Use downward API to get host information to pods
- Sidecar process provides binaries and configuration dynamically

Design Considerations

- Use process classes to help isolate roles on different processes - storage, log, stateless
- Provision log and stateless processes based on recruited counts, with spares
- Recoveries are necessary, but avoid multiple recoveries in close succession
- Recoveries can be caused by exclusions, configuration changes, bounces, and organic failures

Core Operation Loop

- Install new binaries
- Install new config files
- Add new processes
- Reconfigure DB
- Choose processes to remove
- Exclude removed processes
- Change coordinators
- Delete removed processes
- Include removed processes
- Bounce instances

Core Operation Loop

- Install new binaries
- Install new config files
- Add new processes
- Reconfigure DB
- Choose processes to remove
- Exclude removed processes
- Change coordinators
- Delete removed processes
- Include removed processes
- Bounce instances

Installing Binaries and Conf

- Use a dynamic config update process to install new monitor conf
- Cluster files can be installed through the same mechanism
- When using config maps in Kubernetes, an indirect mechanism may be required
- Upgrades will require installing new binaries alongside the old binaries
- Sidecar container can be upgraded independently of main container

Core Operation Loop

- Install new binaries
- Install new config files
- Add new processes
- Reconfigure DB
- Choose processes to remove
- Exclude removed processes
- Change coordinators
- Delete removed processes
- Include removed processes
- Bounce instances

Add New Processes

- New processes can be bootstrapped and join the cluster with the existing cluster file
- If you are creating a new cluster, you may need to bootstrap with a dummy cluster file, or use an empty fdbmonitor conf file
- In some environments, you may need to provide IPs, machines, etc. dynamically
- NB: Cluster files must be writable by all processes, including clients

Core Operation Loop

- Install new binaries
- Install new config files
- Add new processes
- Reconfigure DB
- Choose processes to remove
- Exclude removed processes
- Change coordinators
- Delete removed processes
- Include removed processes
- Bounce instances

Change Database Configuration

- Best practice is generally to configure database after adding new processes, and before removing old processes
- Changing DCs can require multiple stages
- Most other changes can be made in a single configure command

Core Operation Loop

- Install new binaries
- Install new config files
- Add new processes
- Reconfigure DB
- Choose processes to remove
- Exclude removed processes
- Change coordinators
- Delete removed processes
- Include removed processes
- Bounce instances

Removing Processes

- Reasons for removing processes will vary based on environment: hardware failures, migrations, load issues
- Exclusions will block until data movement is complete and removal can proceed
- After shutting down processes, confirm removal in cluster status
- Once removal is confirmed by control plane and cluster status, safe to re-include

Core Operation Loop

- Install new binaries
- Install new config files
- Add new processes
- Reconfigure DB
- Choose processes to remove
- Exclude removed processes
- Change coordinators
- Delete removed processes
- Include removed processes
- Bounce instances

Change Coordinators

- When creating a new cluster, you will need to generate an initial cluster file yourself after getting initial resources
- Once cluster is running, you should recruit new coordinators whenever a coordinator is unreachable or is being removed
- If a coordinator changes its IP it will be unreachable as a coordinator
- After changing coordinators, you can update the cluster file lazily

Core Operation Loop

- Install new binaries
- Install new config files
- Add new processes
- Reconfigure DB
- Choose processes to remove
- Exclude removed processes
- Change coordinators
- Delete removed processes
- Include removed processes
- Bounce instances

Bouncing Instances

- With new monitor conf installed, bounce all fdbserver processes through the CLI
- After bounce, you can verify all instances are back up through the cluster status
- If you are bouncing as part of an upgrade, you must confirm that clients have client libraries that are compatible with both the old and new versions
- Information on client version compatibility can be found in the JSON status in cluster / clients /supported_versions /connected_clients

Bounce Strategy

- We do all of our bounces through the CLI, bouncing all processes at once
- Fast bounces give a single recovery, which improves client experience
- Fast bounces avoid different processes being on different configuration for prolonged periods, which is not well-tested
- Only need to do a rolling bounce when fdbmonitor is being restarted, e.g. after an upgrade

Wait, what? Come on.

- You may be saying “This goes against everything I know about distributed systems”
- This has proven to be a very effective strategy for us at Apple
- Confidence in new configs can be built by staging it through different clusters, e.g. QA to prod
- Canary processes within a cluster are less likely to find problems in a system as heterogeneous as FDB
- There are other changes which cannot be canaried, like configuration changes

Alternative: Rolling Bounces

- Will not work for minor / major upgrades, which are protocol incompatible
- For other kind of changes, a rolling bounce of log and stateless processes will trigger an undesirable number of recoveries
- Number of recoveries could be reduced by bringing up a brand new parallel set of log and stateless and doing an atomic cutover
- Rolling bounce of storage servers will not cause a recovery
- Heterogeneous values for knobs are not necessarily safe

Alternative: DR Cutover

- The multi-cluster DR solution could allow bringing up a new cluster, syncing data from the old cluster, and atomically cutting over
- This would give a broader set of options for validating the correctness of the new cluster
- DR does not currently work across versions, but that may be a smaller problem space to tackle
- Will be very expensive and time-consuming
- Probably not practical for simple things like knob changes or patches
- Clients have to switch to a new cluster file, which will introduce additional downtime

Conclusion

- Having a strong story around operations is key to FDB's adoption and success
- We'd like to foster a more active conversation on these areas in the coming years
- We're also eager for opportunities to share more internal tooling with the community and contribute to a stronger ecosystem