instaclustr

Bootstrap From Backups

Reducing cluster load while adding capacity

#CassandraSummit

instaclustr.com

Who am I and what do I do?

- Ben Bromhead
- Co-founder and CTO of Instaclustr -> www.instaclustr.com
- Instaclustr provides Cassandra-as-a-Service in the cloud.
- Currently in AWS, Google Cloud in private beta with more to come.
- We currently manage 50+ nodes for various customers, who do various things with it.

Cassandra and Scaling

- Premise: We have an existing cluster and we need either more storage / better performance / higher availability.
- Normally fairly awesome, most people do the following:
  - Set seed nodes, Start Cassandra.
  - Node joins ring and take responsibility for some portion of the ring.
  - Commence the bootstrap process. The joining node streams data from other nodes for the range, builds indexes etc.
  - Specifically the node receives streamed SSTables that contain rows within the range that it is now responsible for (the data component)

Not perfect, but getting better

- Joining node can violate consistency due to range movements - Somewhat fixed in 2.1 - See CASSANDRA-2434

- Adding a replacement node with the same address/range ownership is a different workflow. replace_address workflow is still tricky for some people. - See CASSANDRA-7356

- Adding nodes to a cluster with multiple racks can also be tricky and prone to creating hotspots. This is mainly an operational issue.

A wild “fundamental issue” appears…

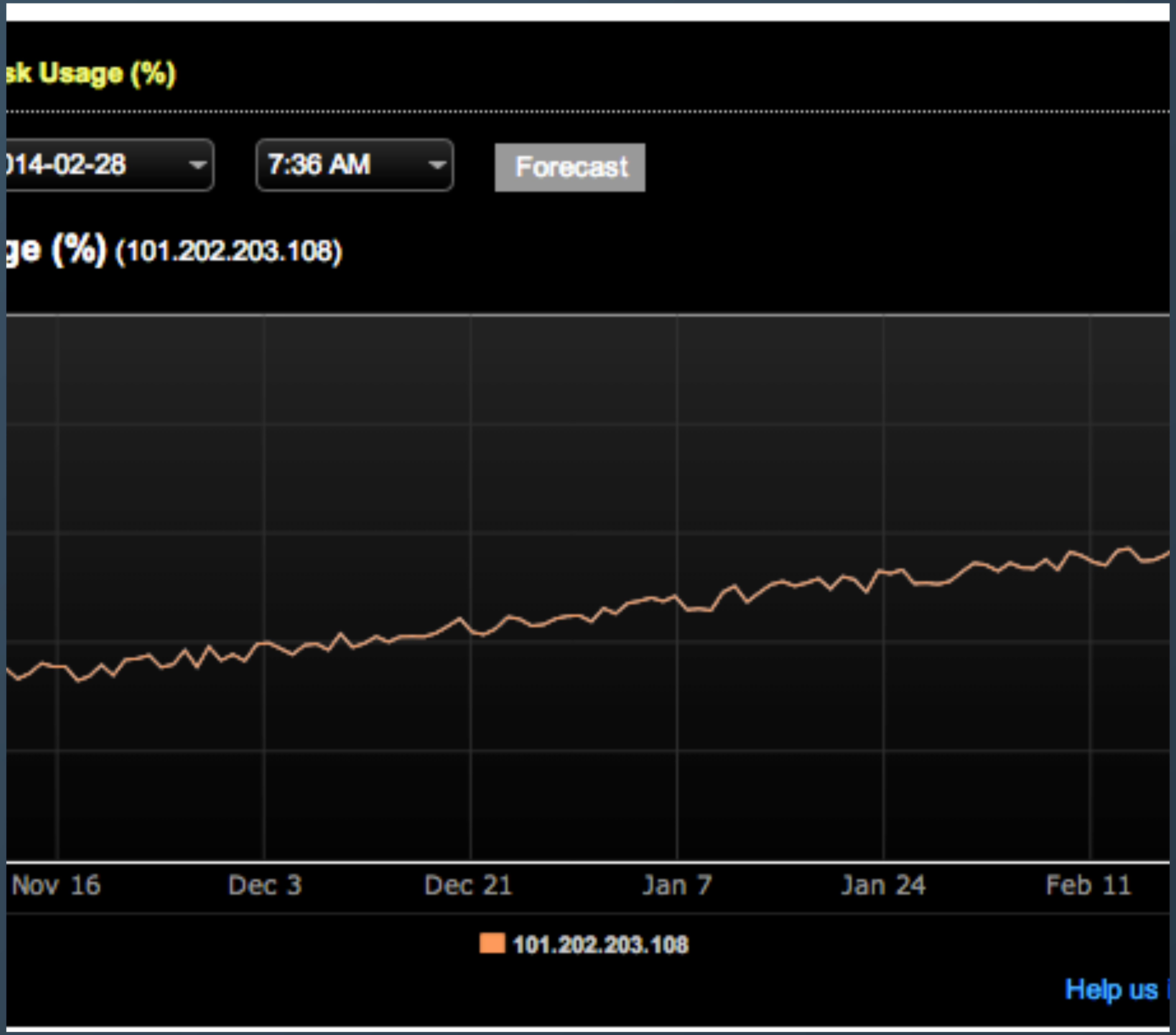
- Joining nodes add additional load on the existing nodes in the cluster.
- Joining nodes stream data from existing nodes (the node who used to be the primary for the range that is moving).
- Takes up valuable bandwidth and I/O
- Key requirement: As a managed Cassandra service, we need to make all our operations as side-effect free as possible.
- Key requirement: Our customers don’t want to worry about operation specific details.

how do we prevent this?

Solutions, part 1

Make sure your nodes never get stressed.

- Capacity planning (OpsCenter has some good tools). Traffic prediction.



Solutions, part 2

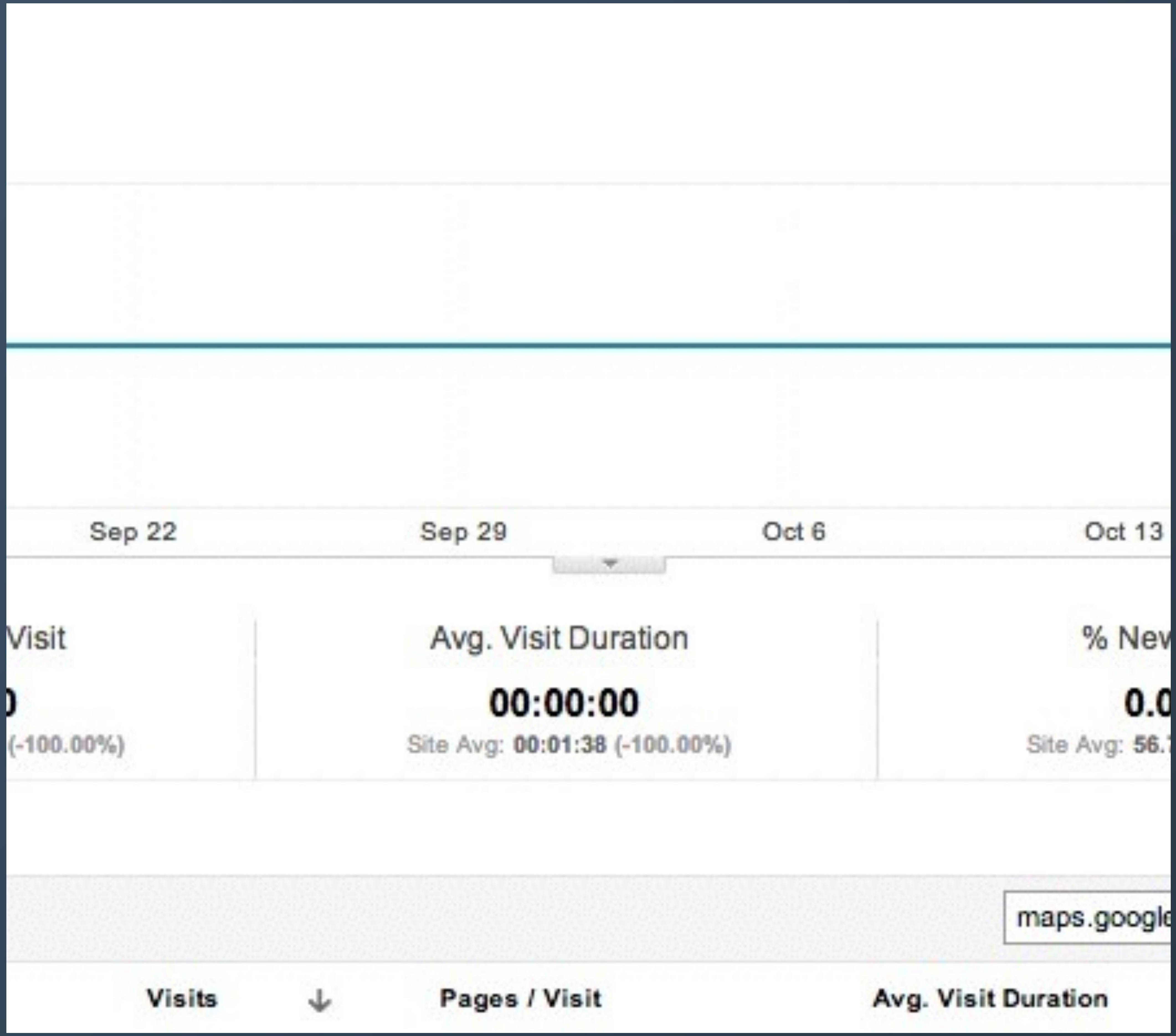
Make sure your nodes never get stressed

- Over provision.

Solutions, part 3

Make sure your nodes never get stressed.

- Ensure your startup / app / project / whatever never goes viral or gets featured in national media.



Visit
)
(-100.00%)

Avg. Visit Duration
00:00:00
Site Avg: 00:01:38 (-100.00%)

% Nev
0.0
Site Avg: 56.

maps.googl

Visits | ↓ | Pages / Visit | Avg. Visit Duration

Solutions, part 4

If your nodes are already stressed, very hard to add capacity.

- Batten down the hatches and wait for a quiet time?

Solutions, part 5

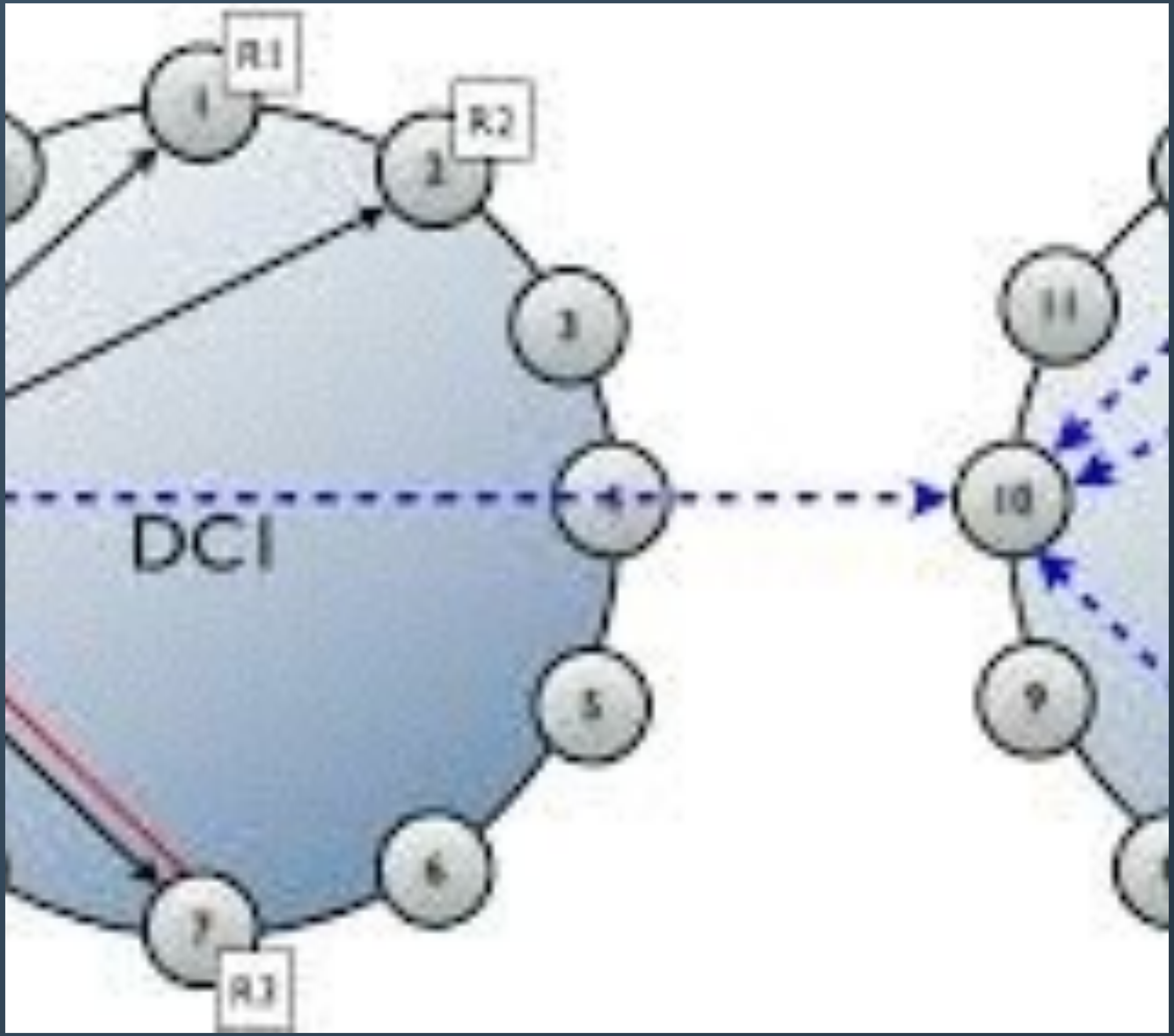
If your nodes are already stressed, very hard to add capacity.

- You are a Cassandra wizard.

Solutions, part 6

If your nodes are already stressed, very hard to add capacity.

- Rebuild from another DC.
- Add node, bootstrap = false and run nodetool rebuild --OTHER_DC



- All these solutions have various strengths and weaknesses.
- Have side-effects or a relatively costly.
- Still need to address:
  - Key requirement: As a managed Cassandra service, we need to make all our operations as side-effect free as possible.
  - Key requirement: Our customers don’t want to worry about operation specific details.

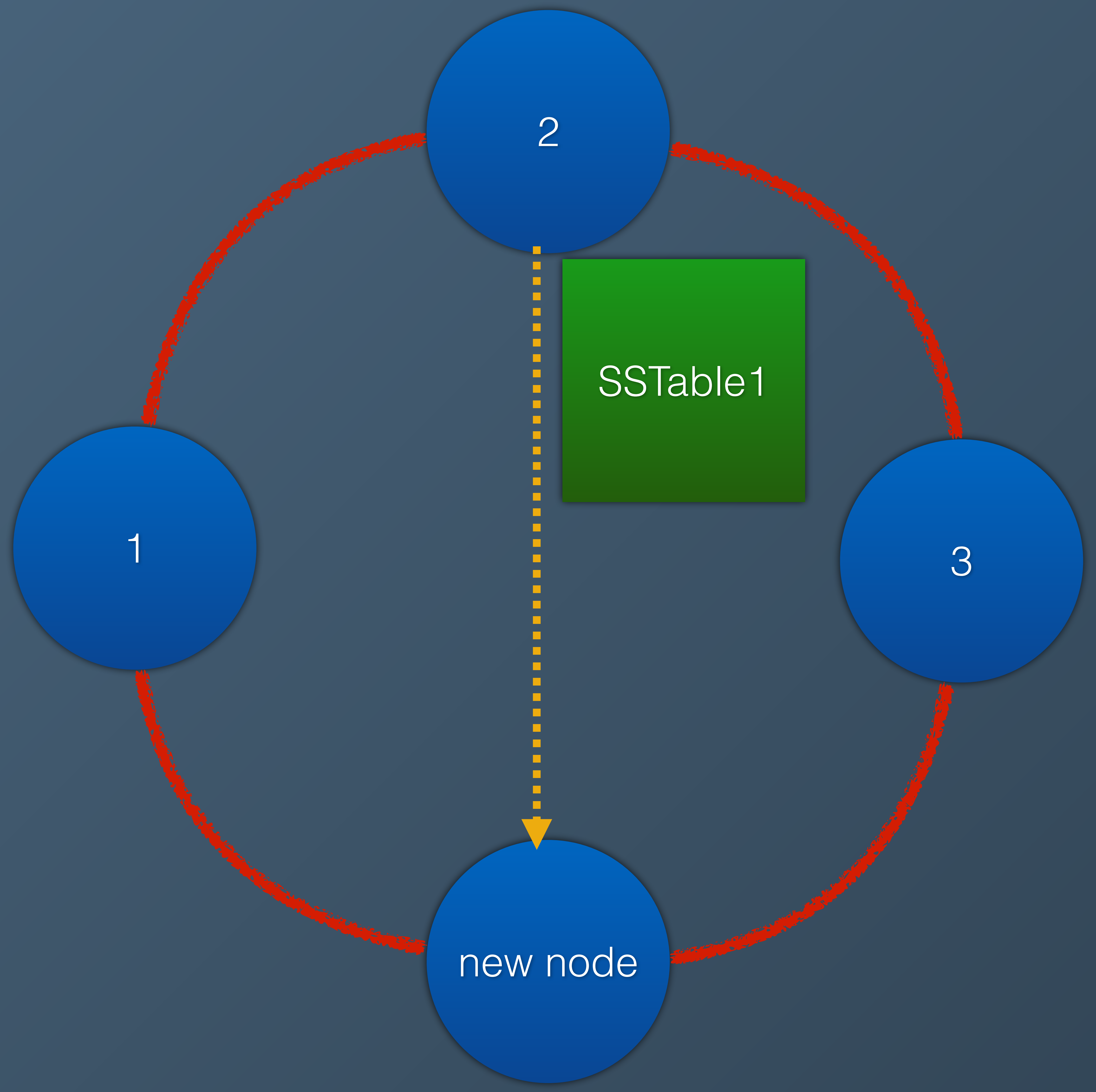
Bootstrap from Backups!

- SSTables are immutable.
- SSTables are also the base unit of data that nodes stream to each other.
- SSTables are what we backup.
- How about we stream the SStables from the backup location instead of the live node?

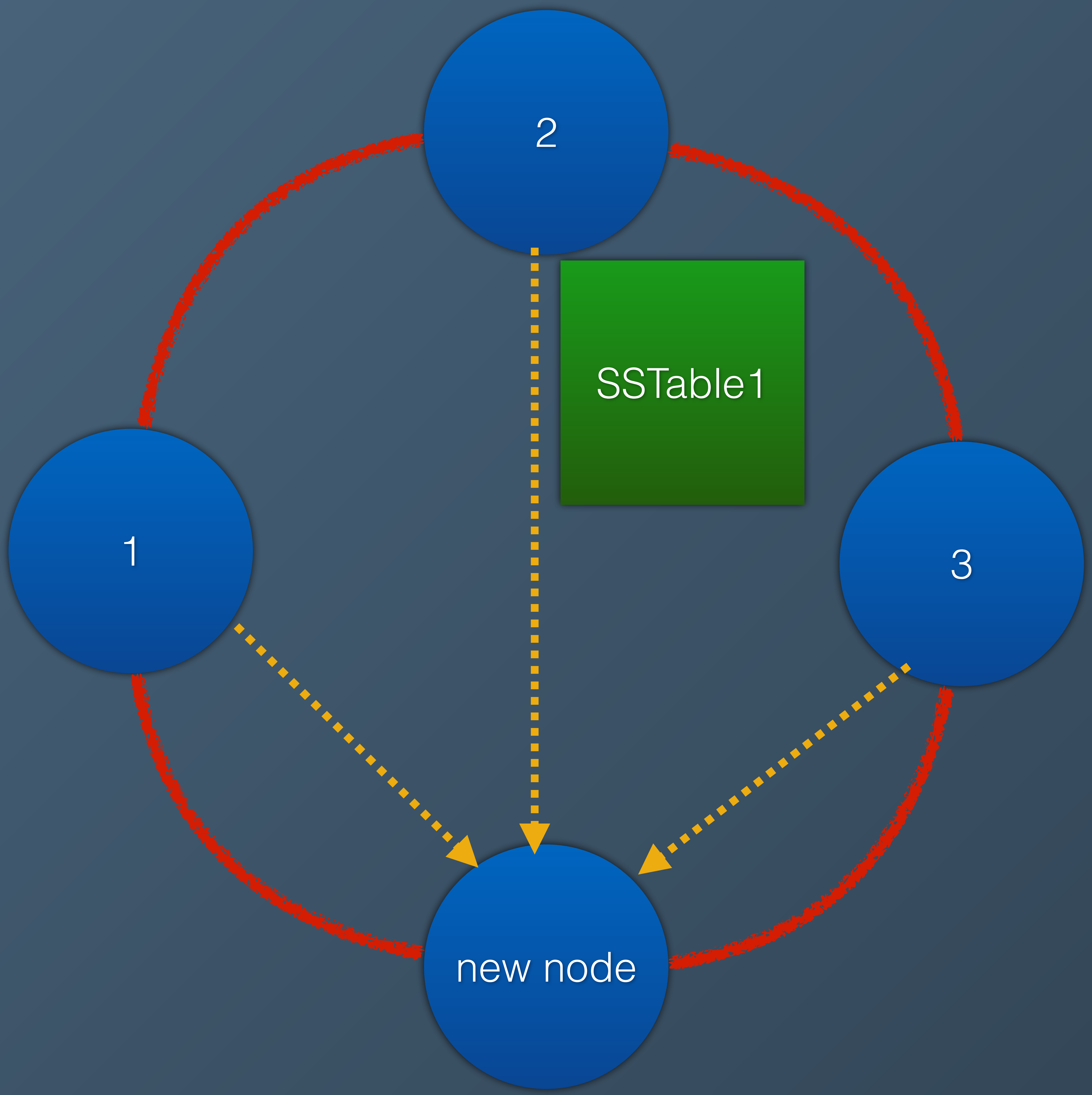
STUPIDGIFS.COM
H HD
HISTORY.COM

- Define an arbitrary command that streams the sstable to stdout.
- Cassandra will some values (broadcast address and filename) into the command to help identify which sstable to fetch.
  - e.g. cat /mnt/some-nfs-mount/%source/%filename
- Cassandra will run the command in a separate process and read the sstable from processes stdout stream.
- If the process fails, the node streams the sstable using the current streaming process. This becomes a performance optimisation rather than a replacement streaming mechanism.

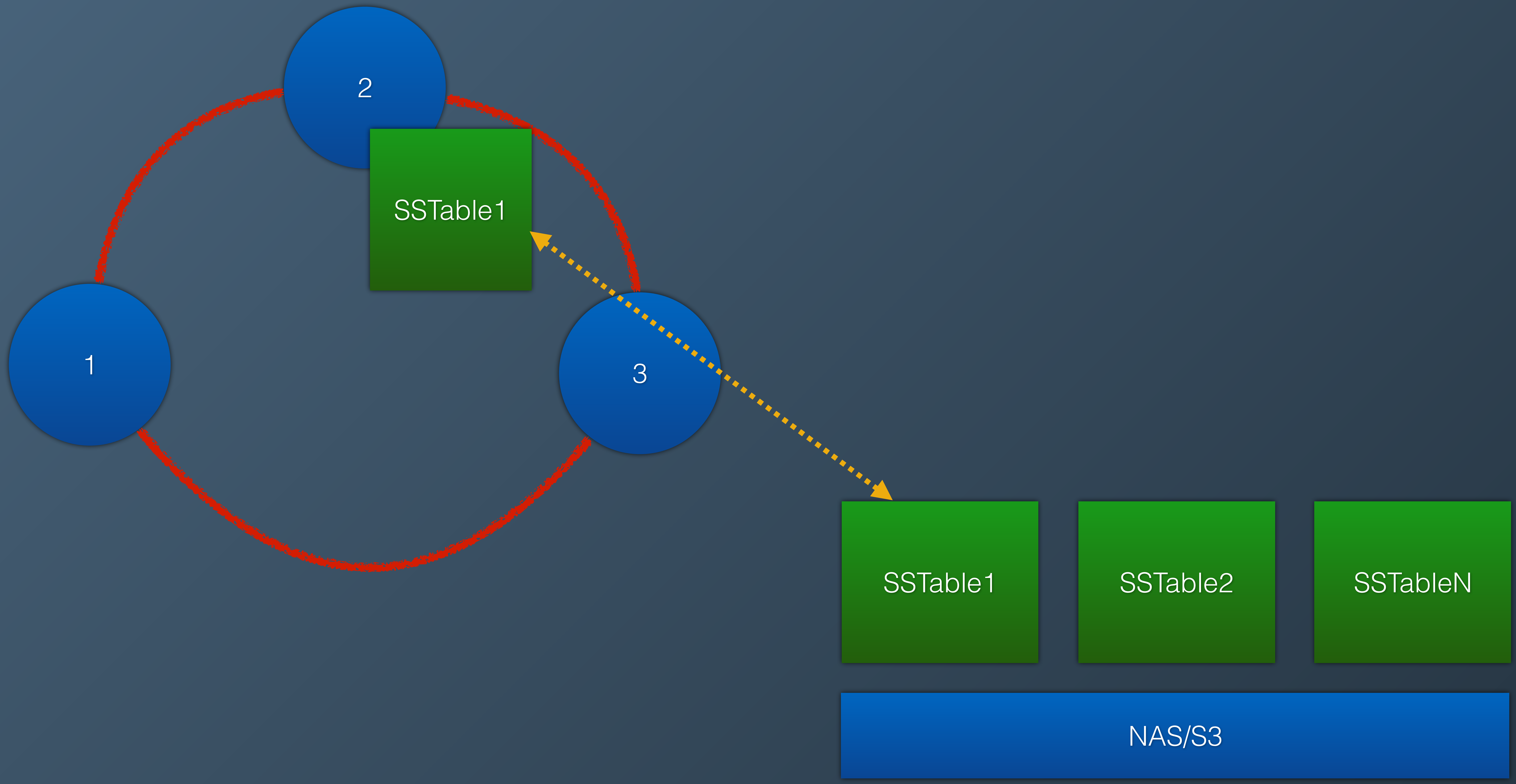
Normal Bootstrap procedure



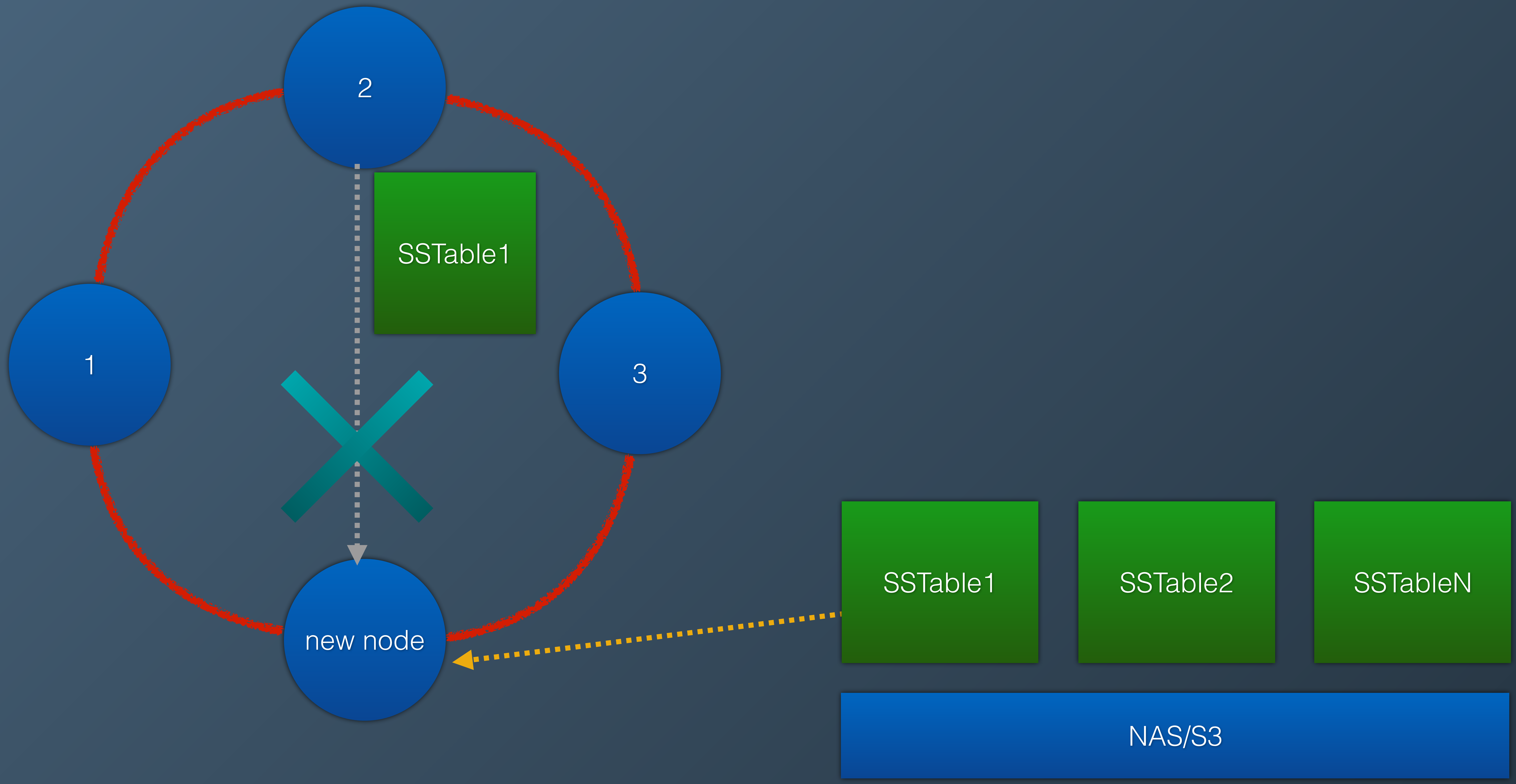
Normal Bootstrap procedure



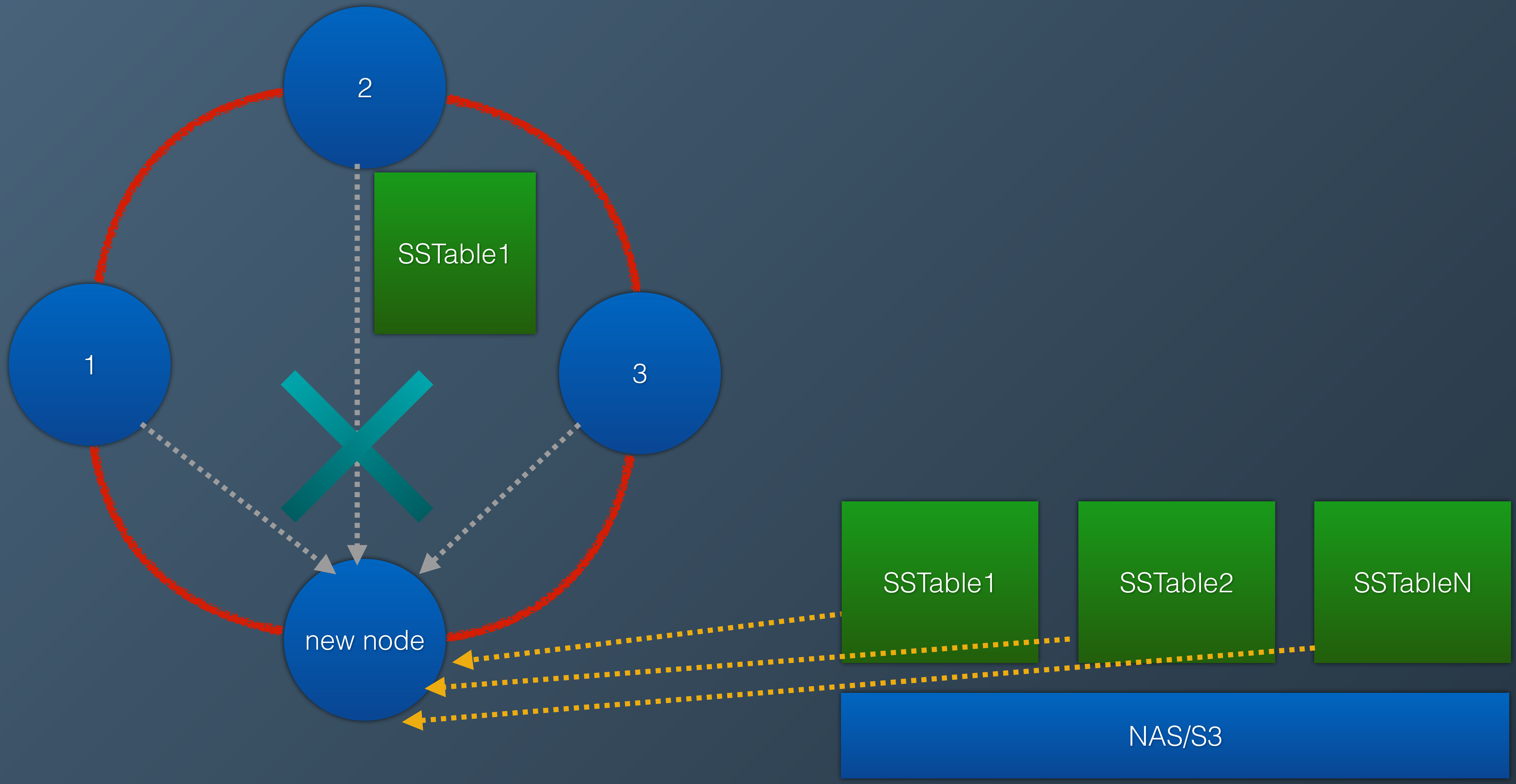
Normal Cluster with backups



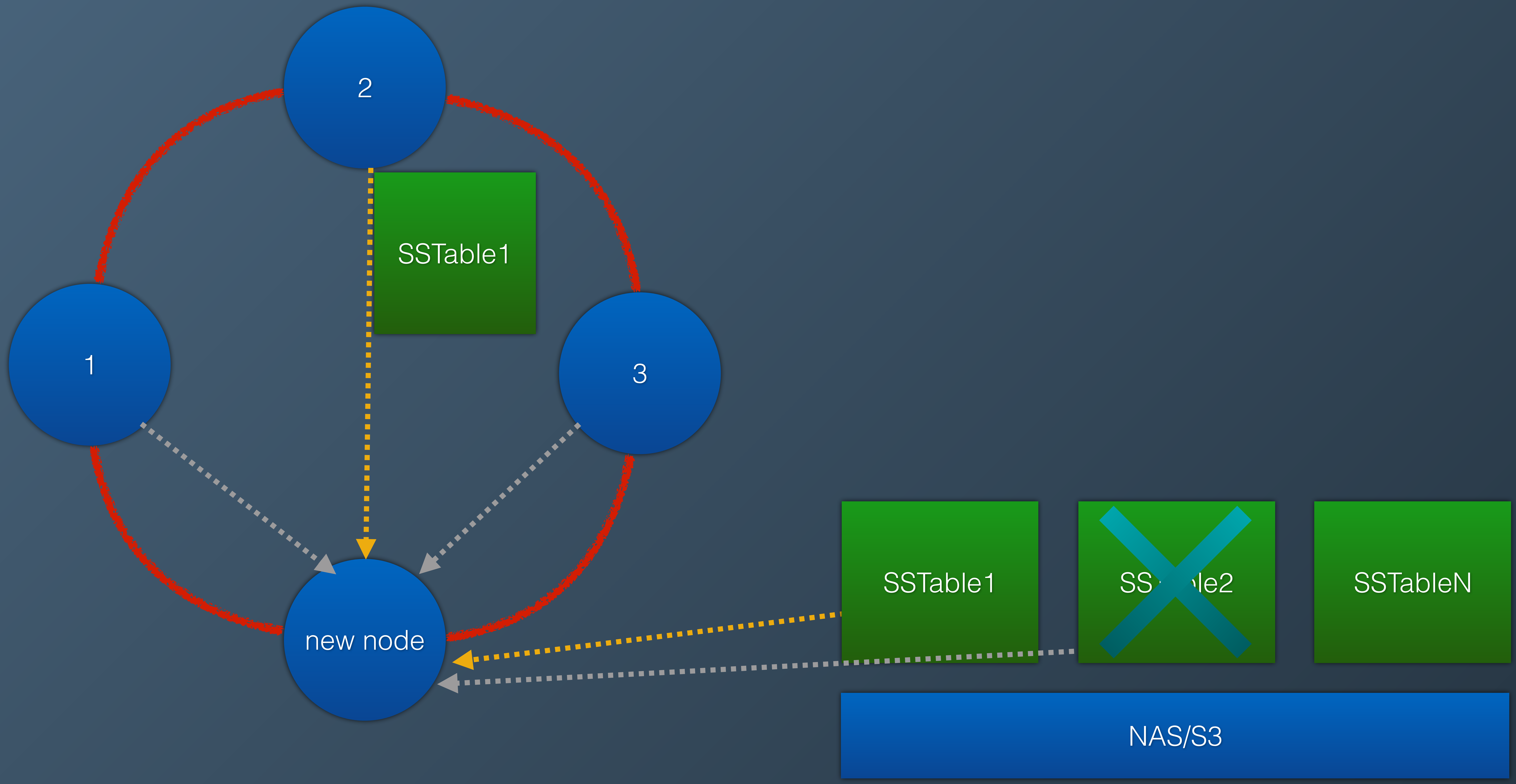
Bootstrap from backup



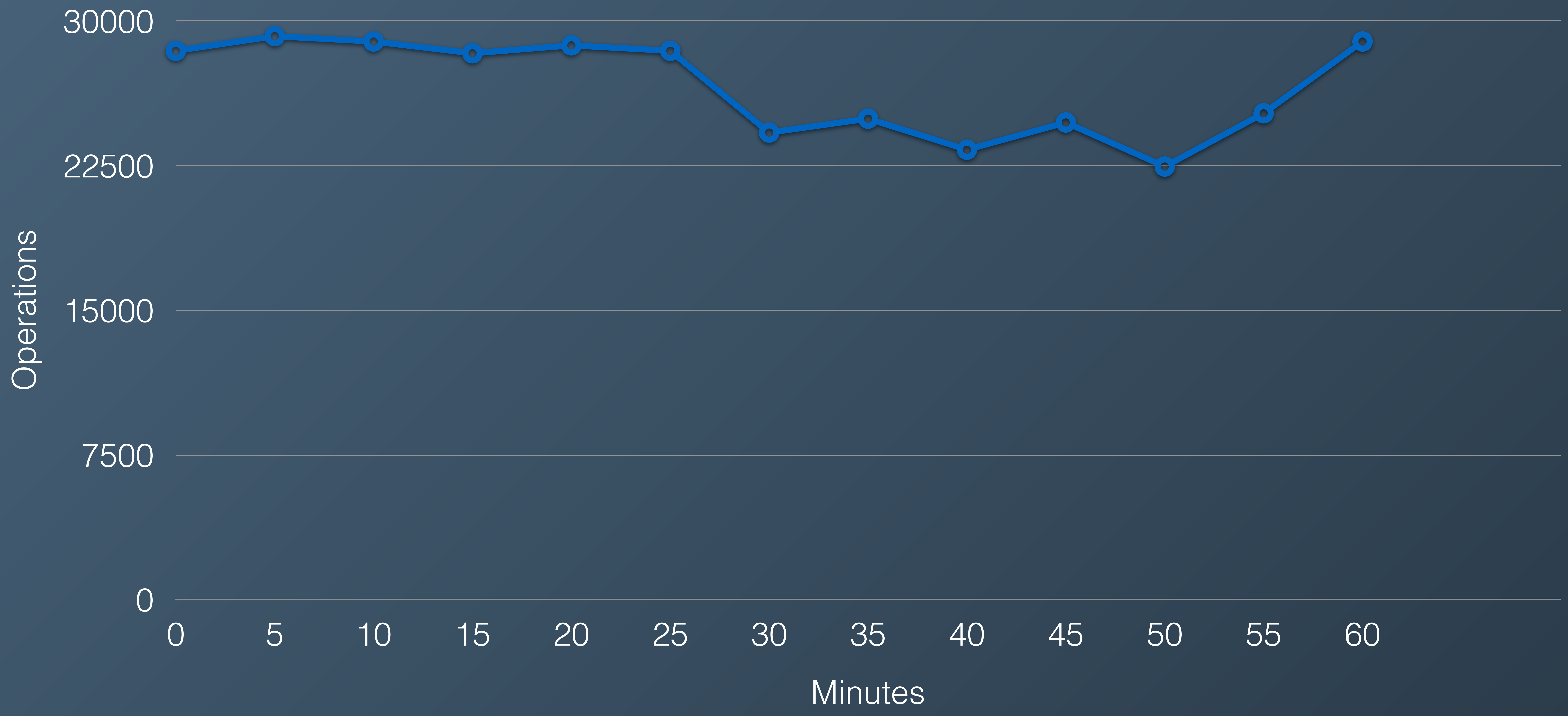
Bootstrap from backup



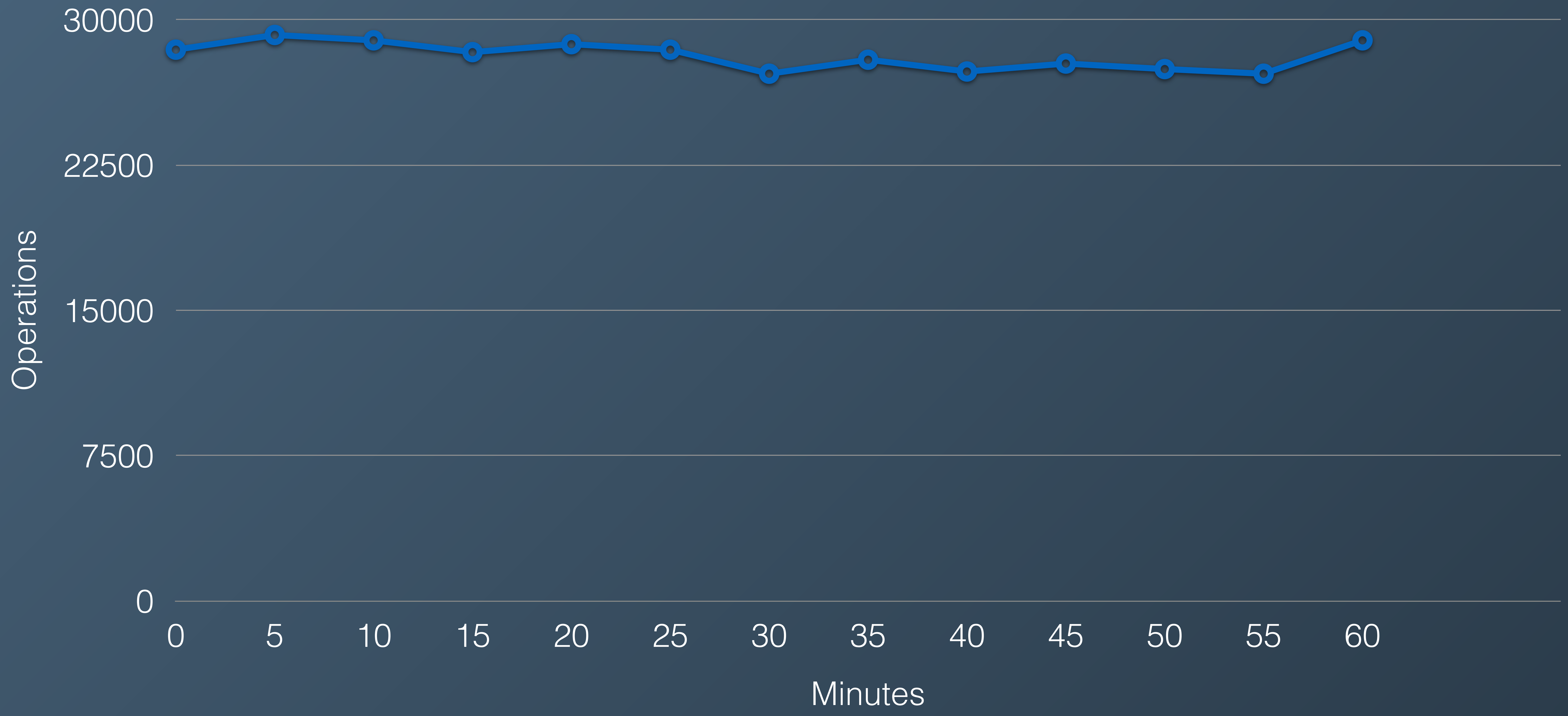
Bootstrap from backup - Catch up



How does it look in real life?



This is your cluster on regular bootstrap



This is your cluster on bootstrap from backups

Why does this matter

- Mostly side-effect free bootstrapping.
- Explore reactive scaling rather than predictive.
- Makes your cluster more cost effective to run.

When can I use this!?

- Not right now, haven’t even submitted as a patch to the C* project (we will).
- Currently running in beta with a select few of our customers.
- Not too sure how much of a good idea it is to use stdout as the stream mechanism. So far so good?
- Will probably need a refactor of the StreamMessage workflow… currently bootstrap from backups is a has that doesn't fit the current model.

Questions