

# Vitess Topology & System Failures

Rafael Chacón

Vitess Meetup - July 2019



# Rafael Chacón

*Staff Engineer*

*Infrastructure*

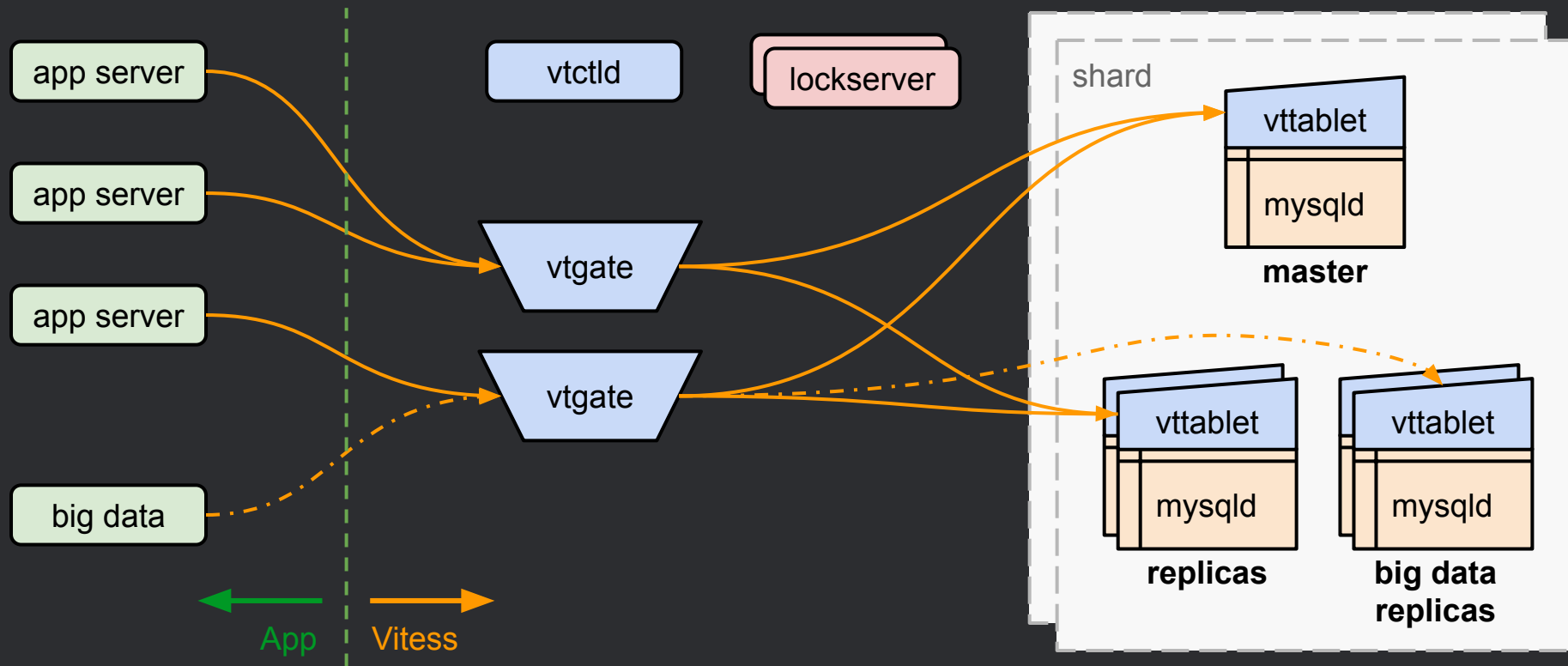


# Agenda

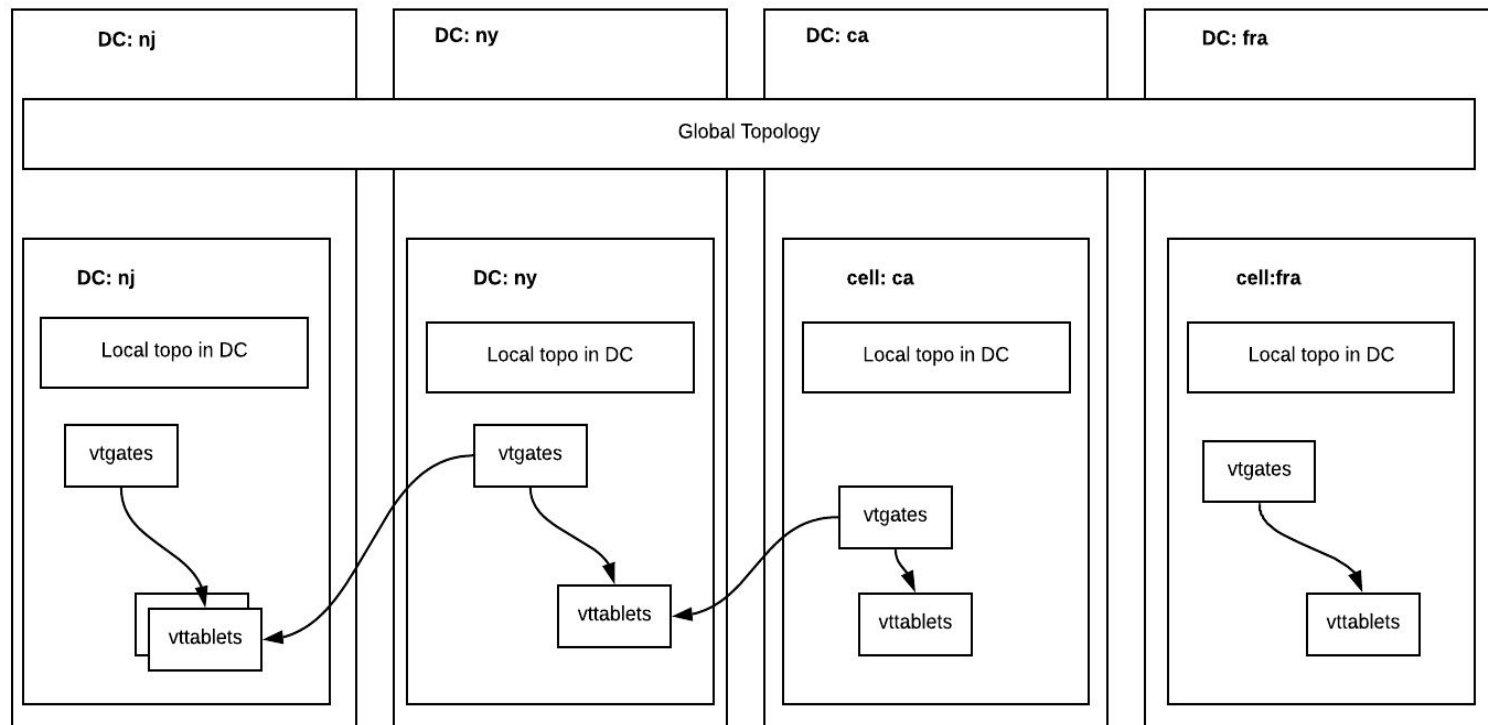
- Vitess Topology.
- Slack Vitess deployment.
- System Failures.
- Topology refactor and new deployment.
- Results.
- Future Work.



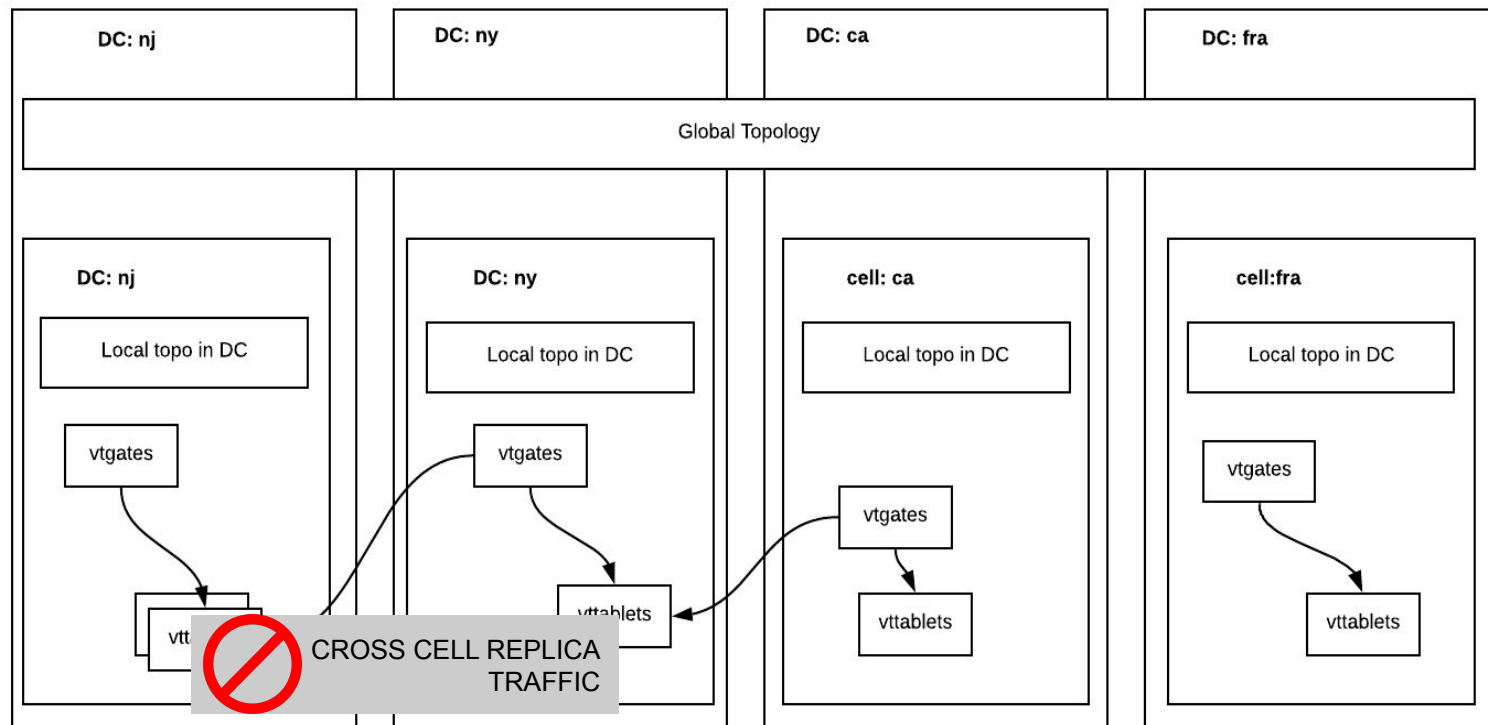
# Architecture



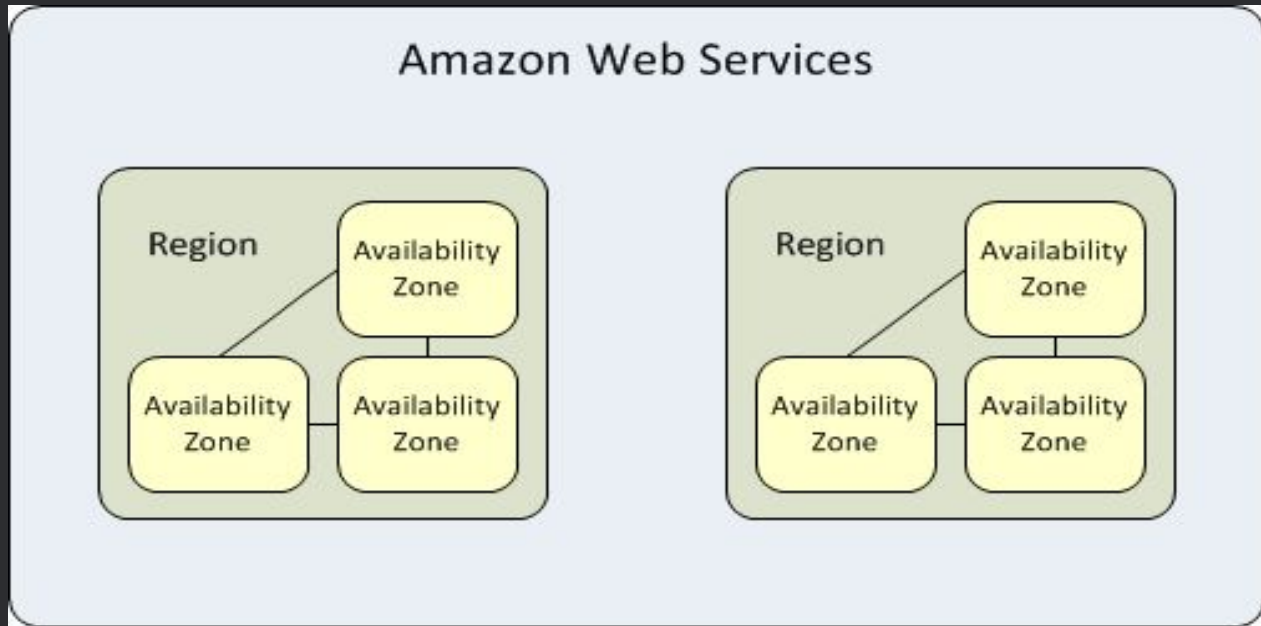
# Vitess Canonical Deployment



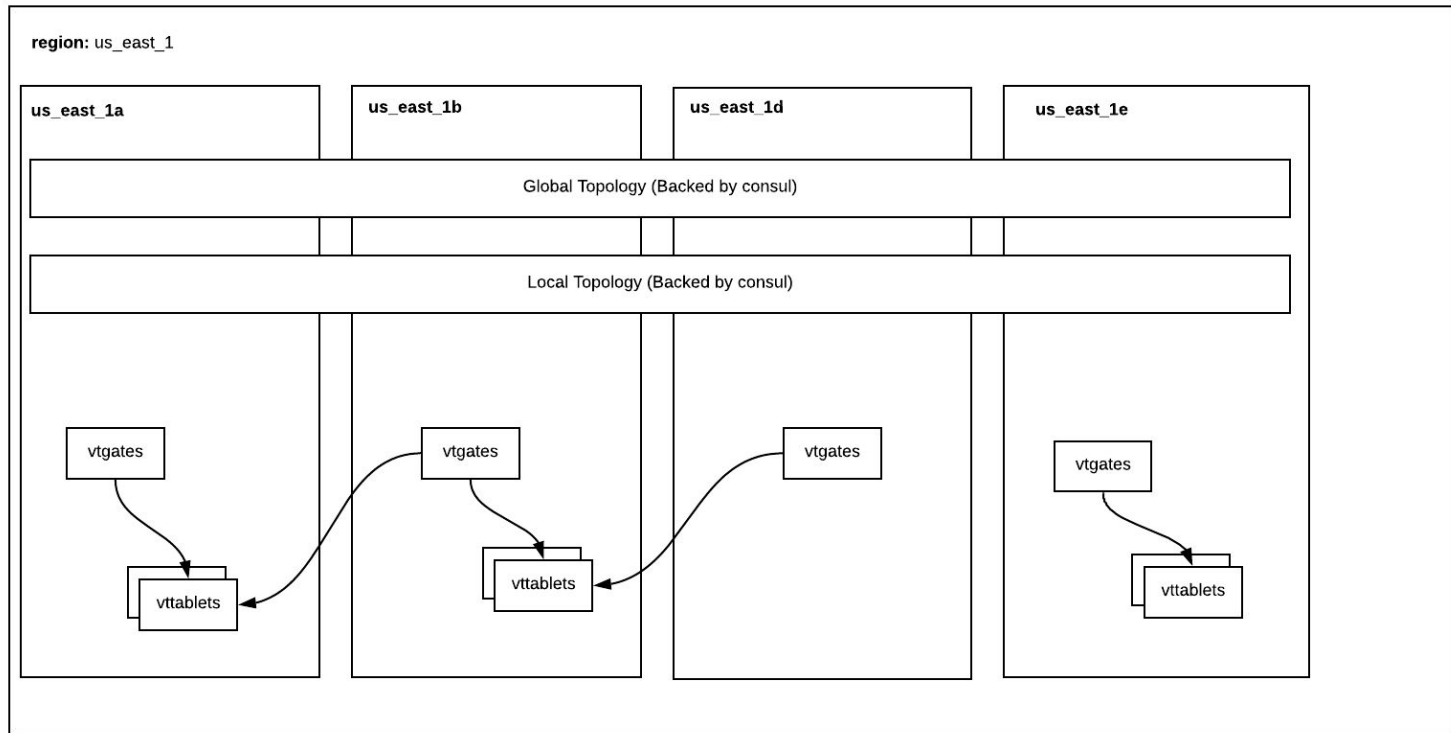
# Vitess Canonical Deployment



# Vitess Slack Deployment

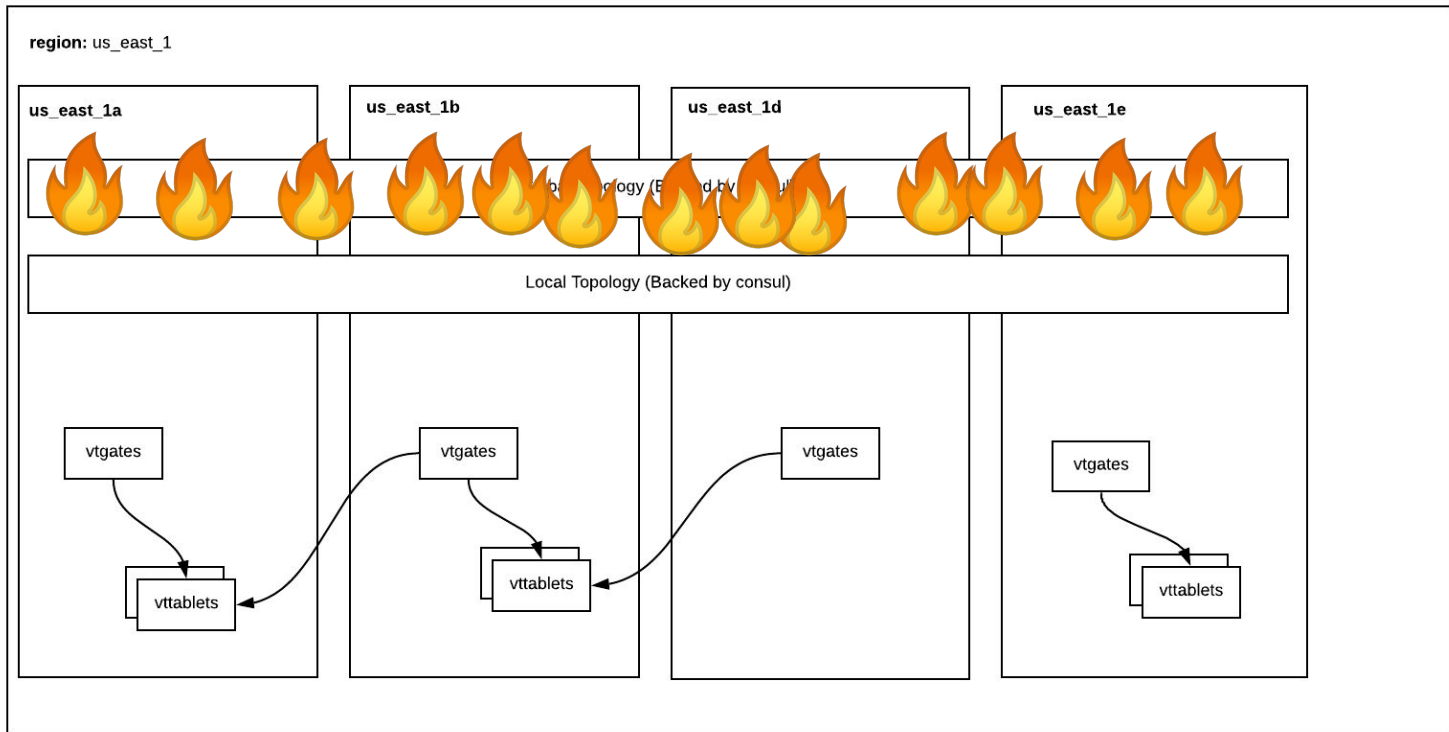


# Slack topology setup





# The unexpected happened



# The unexpected happened

```
164
165 // GetSrvKeyspaceNames returns all keyspace names for the given cell.
166 func (server *ResilientServer) GetSrvKeyspaceNames(ctx context.Context, cell string) ([]string, error) {
167     server.counts.Add(queryCategory, 1)
168
169     // find the entry in the cache, add it if not there
170     key := cell
171     server.mutex.Lock()
172     entry, ok := server.srvKeyspaceNamesCache[key]
173     if !ok {
174         entry = &srvKeyspaceNamesEntry{
175             cell: cell,
176         }
177         server.srvKeyspaceNamesCache[key] = entry
178     }
179     server.mutex.Unlock()
180
181     // Lock the entry, and do everything holding the lock. This
182     // means two concurrent requests will only issue one
183     // underlying query.
184     entry.mutex.Lock()
185     defer entry.mutex.Unlock()
186
187     // If the entry is fresh enough, return it
188     if time.Now().Sub(entry.insertionTime) < server.cacheTTL {
189         return entry.value, entry.lastError
190     }
191
192     // Not in cache or too old, get the real value. We use the context that issued
193     // the query here.
194     result, err := server.topoServer.GetSrvKeyspaceNames(ctx, cell)
195     if err != nil {
196         if entry.insertionTime.IsZero() {
197             server.counts.Add(errorCategory, 1)
198             log.Errorf("GetSrvKeyspaceNames(%v, %v) failed: %v (no cached value, caching and returning error)", ctx, cell, err)
199         } else {
200             server.counts.Add(cachedCategory, 1)
201             log.Warningf("GetSrvKeyspaceNames(%v, %v) failed: %v (returning cached value: %v %v)", ctx, cell, err, entry.value, entry.lastError)
202             return entry.value, entry.lastError
203         }
204     }
205 }
```

# The unexpected happened

```
194 // GetSrvKeyspaceNames returns all keyspace names for the given cell.
195 func (server *ResilientServer) GetSrvKeyspaceNames(ctx context.Context, cell string) ([]string, error) {
196     server.counts.Add(queryCategory, 1)
197
198     // find the entry in the cache, add it if not there
199     key := cell
200     server.mutex.Lock()
201     entry, ok := server.srvKeyspaceNamesCache[key]
202     if !ok {
203         entry = &srvKeyspaceNamesEntry{
204             cell: cell,
205         }
206         server.srvKeyspaceNamesCache[key] = entry
207     }
208     server.mutex.Unlock()
209
210     // Lock the entry, and do everything holding the lock. This
211     // means two concurrent requests will only issue one
212     // underlying query.
213     entry.mutex.Lock()
214     defer entry.mutex.Unlock()
215
216     // If it is not time to check again, then return either the cached
217     // value or the cached error
218     cacheValid := entry.value != nil && time.Since(entry.insertionTime) < server.cacheTTL
219     shouldRefresh := time.Since(entry.lastQueryTime) > server.cacheRefresh
220
221     if !shouldRefresh {
222         if cacheValid {
223             return entry.value, nil
224         }
225         return nil, entry.lastError
226     }
227
228     // Not in cache or needs refresh so try to get the real value.
229     // We use the context that issued the query here.
230     result, err := server.topoServer.GetSrvKeyspaceNames(ctx, cell)
231     if err == nil {
232         // save the value we got and the current time in the cache
233         entry.insertionTime = time.Now()
234         entry.value = result
235     } else {
```

# We fixed things!

make the resilient topo cache even more resilient and informative #3641

**Merged** alainjobart merged 6 commits into `vitessio:master` from `tinspeck:resilient-topo-more-resilient-and-info`

Conversation 24 Commits 6 Checks 0 Files changed 4

demmer commented on Feb 8, 2018 Member + 👤 ...

### Overview

Improve the status UI for the resilient topo cache, make GetSrvKeyspaceNames fetch asynchronously, and fix a logic bug in the watch-based caching for GetSrvKeyspace.

This incorporates the ideas first proposed by @yangxuanjia in #3640 while addressing my concerns about the refresh interval and properly managing locks and state.

add resilient topo server caching for the full srv keyspace object #3610

**Merged** alainjobart merged 11 commits into `vitessio:master` from `tinspeck:resilient-topo-server-should-be-more-r`

Conversation 15 Commits 11 Checks 0 Files changed 9

demmer commented on Jan 30, 2018 • edited Member + 👤 ...

### What

Add support to the ResilientServer to cache the whole SrvKeyspace object for the configured TTL and a configurable refresh interval in addition to the TTL.

# Incomplete Approach

- Defensive programming.
- Fixing bugs.
- Multi tenant is an issue (shared consul cluster).

# Complex Systems Fail

- Complex systems are intrinsically hazardous systems.
- Complex systems are heavily and successfully defended against failure.
- Catastrophe is always just around the corner.
- Complex systems contain changing mixtures of failures latent within them.

# Complex Systems Fail

- Complex systems are intrinsically hazardous systems.
- Complex systems are heavily and successfully defended against failure.
- Catastrophe is always just around the corner.
- Complex systems contain changing mixtures of failures latent within them.

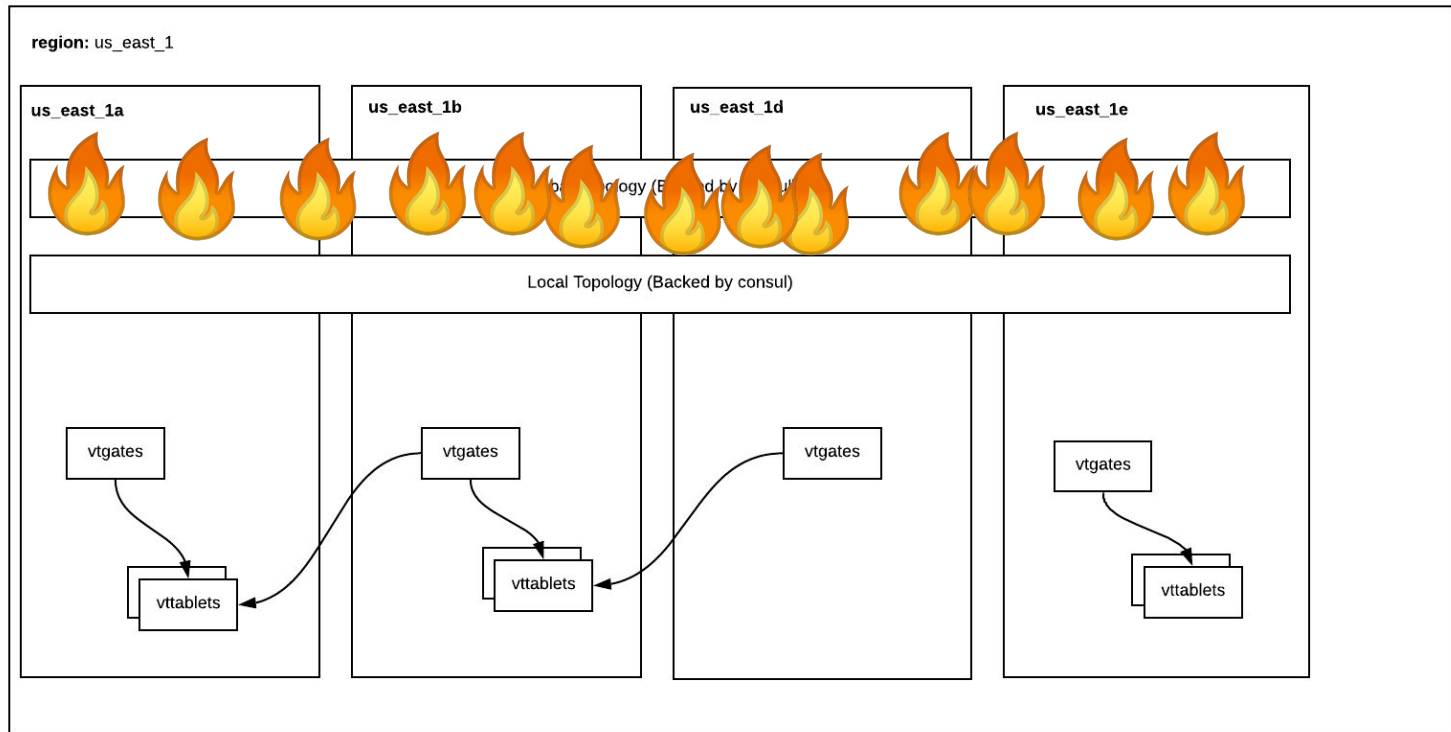
***A Short Treatise on the Nature of Failure; How Failure is Evaluated; How Failure is Attributed to Proximate Cause; and the Resulting New Understanding of Patient Safety - Richard I. Cook, MD (2000)***

# Designing Resilient Systems

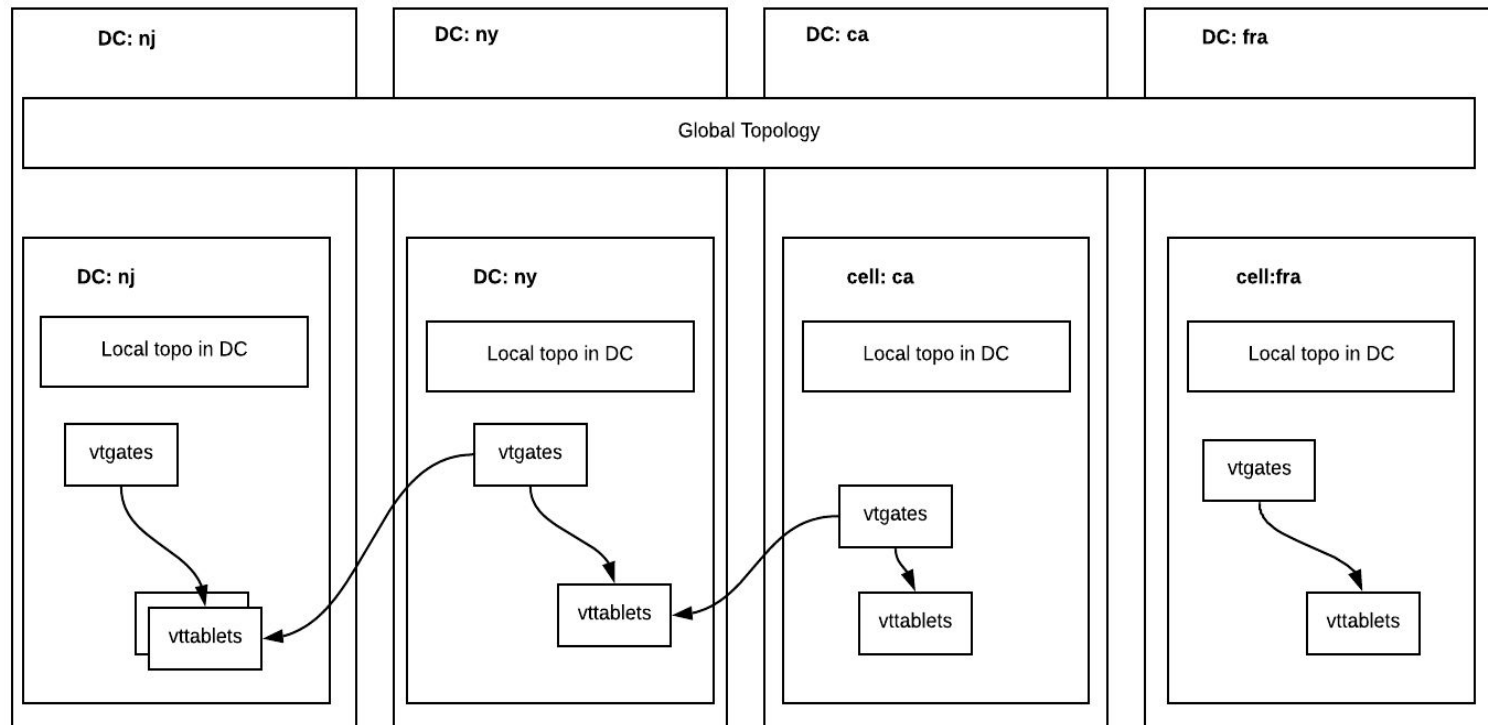
- **Isolation is key.**
  - Minimize blast radius when the unexpected happens.
  - Think about your dependencies.
  - Let it crash!



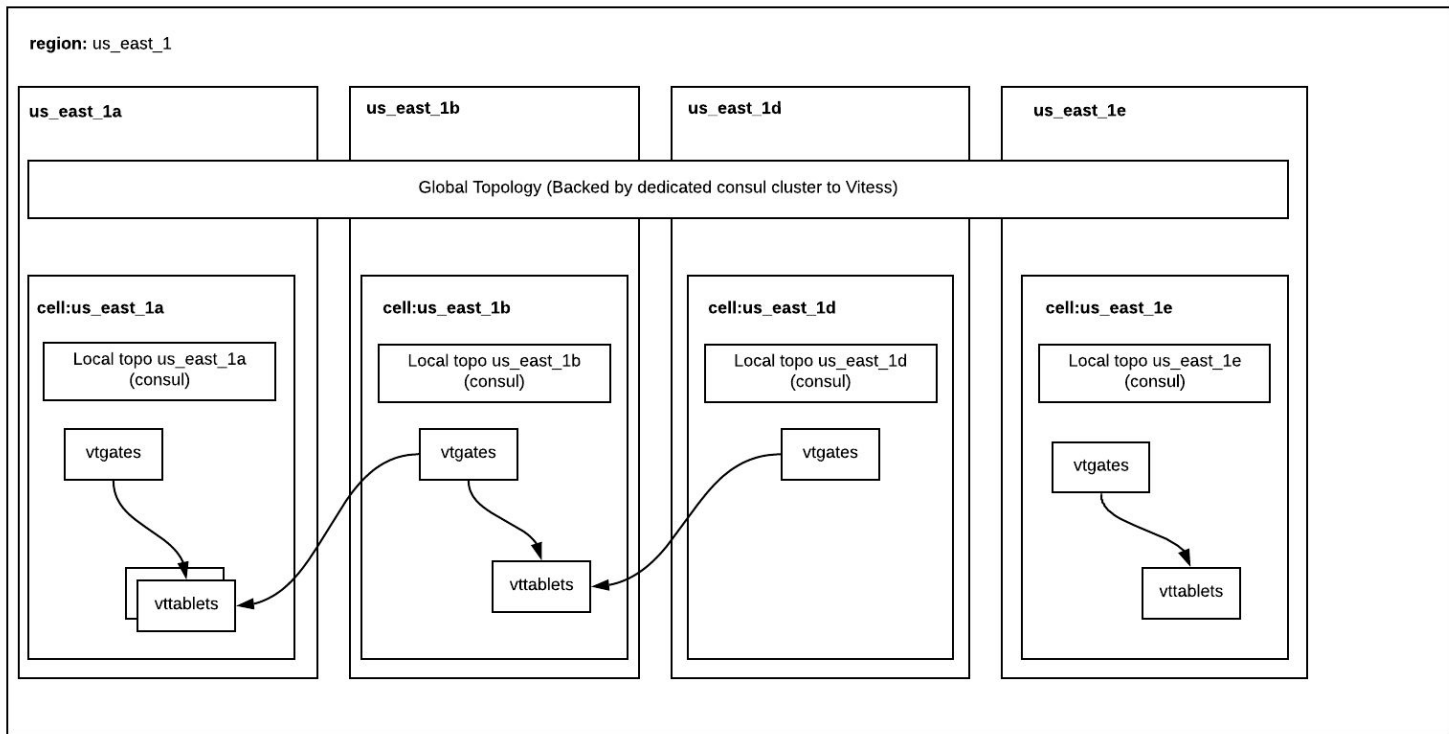
# How can we improve it?



# Vitess cells



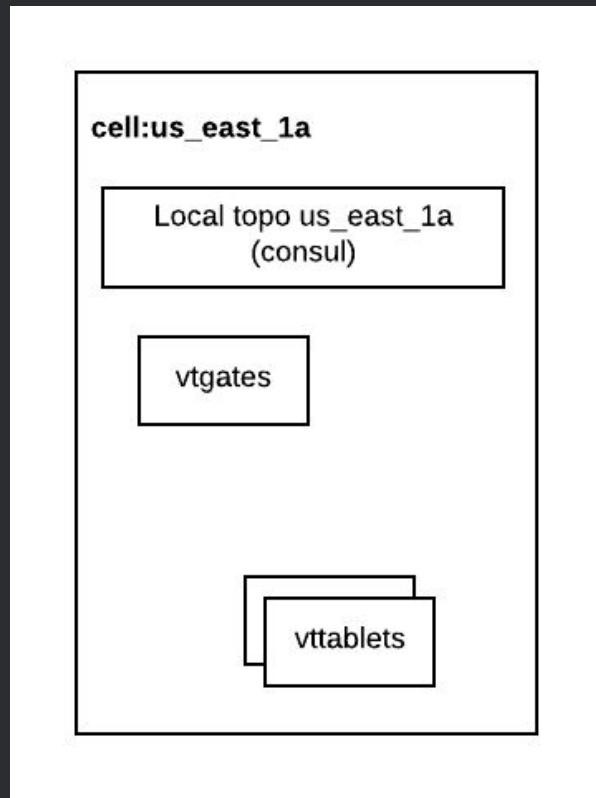
# Expanding the cell concept



# Did it work?

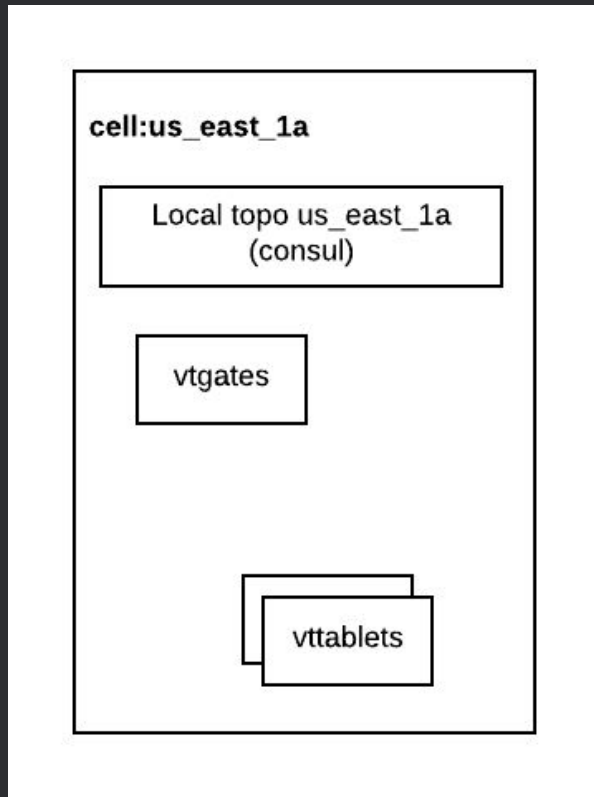
- Huge refactor of Vitess topology.
- We successfully migrated all our cluster to this new setup with no hiccups.
- More than three months.

# Cell Affinity

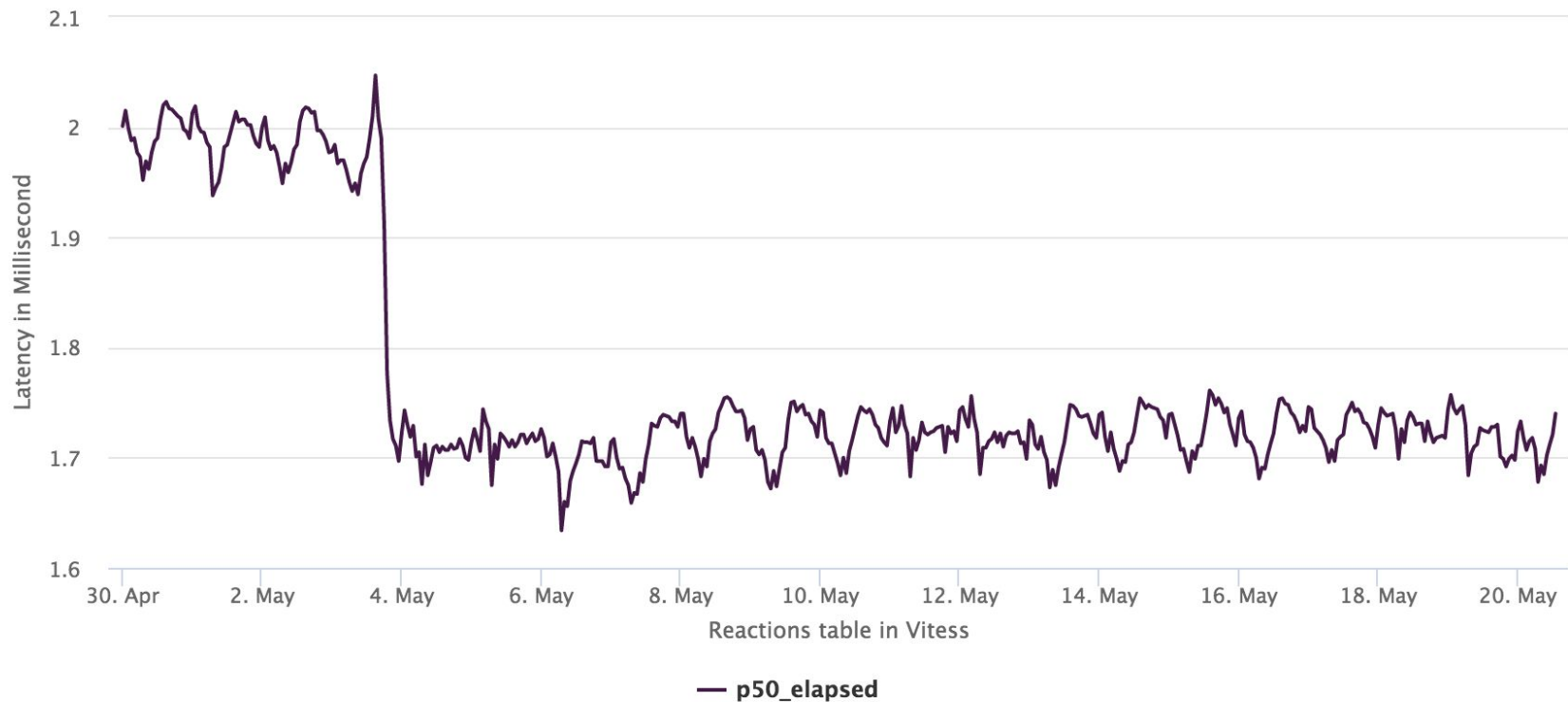


# Cell Affinity

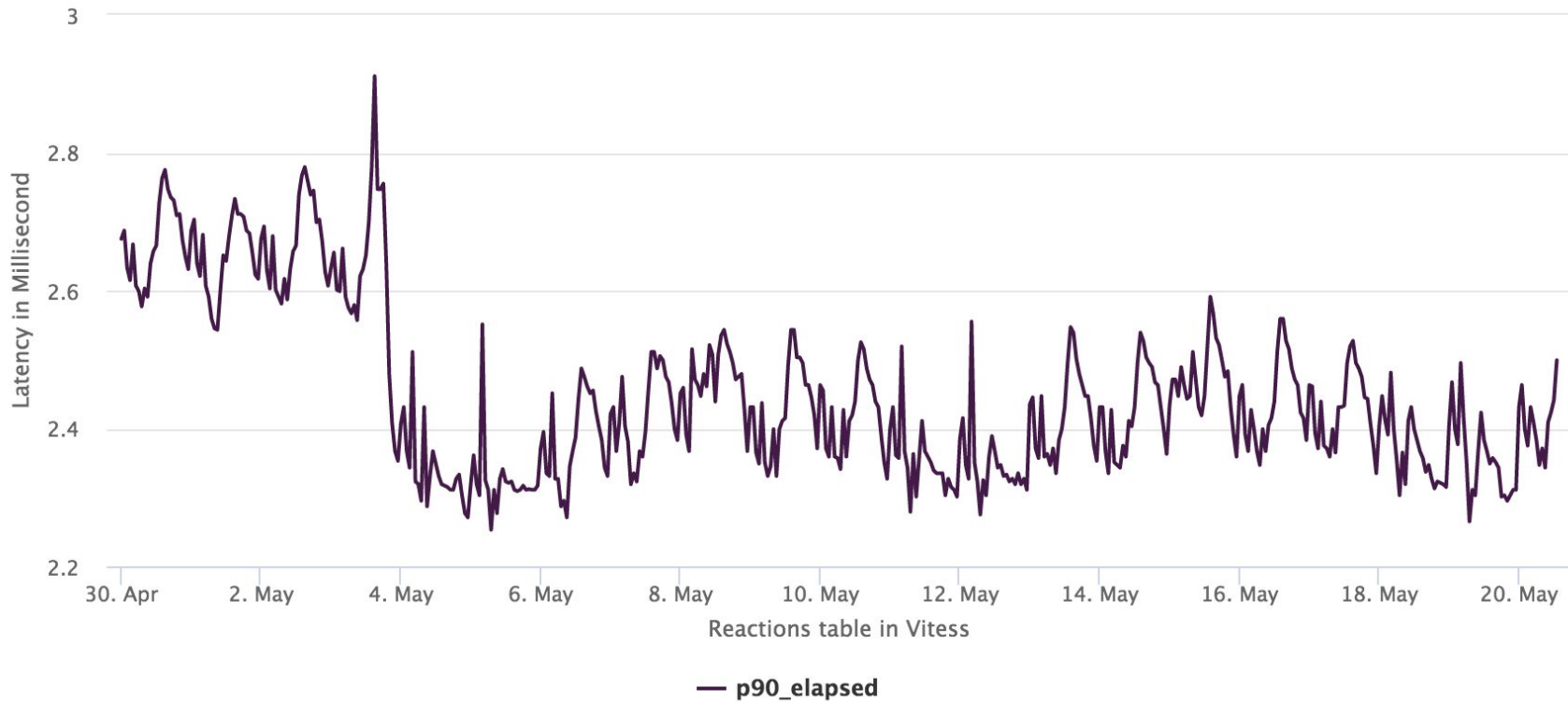
- In AWS this around 300 micros improvement in latency.
- For some of our tables, this represents at 15% improvement in the the P50.



# Cell Affinity



# Cell Affinity





# Future Work

- Enabling more replicas than zones per shard.
- Remove dependencies from topo for normal operations.
- Deployments per Cell.

# Questions?

