Benchmarking is never optional

October 2nd, 2019

By Art van Scheppingen

October 2nd, 2019 - Percona Live Europe - Amsterdam

Agenda

1. **Introduction**
   Who am I and who is Messagebird?

2. **Primer on benchmarking**
   For those unfamiliar with it

3. **Compare products**
   Make a fact based decision

4. **Resource planning**
   Leverage benchmarks to do resource planning

5. **Predict workloads**
   What happens if you overload systems?

6. **Questions**
   In case anything is unclear

Introduction

Who am I and who is Messagebird?

Disclaimer

- Talk was submitted before I joined MessageBird in the role of Database Engineer
- Examples have no relation to MessageBird
- Examples can be applicable to MessageBird as well
- Examples might be applicable to anyone here ;)

We believe in a world where communicating with a business feels as natural as communicating with a friend.

About messagebird

MessageBird is a cloud communications platform that empowers consumers to communicate with your business in the same way they communicate with their friends - seamlessly, on their own timeline and with the context of previous conversations.

For additional information visit:
www.messagebird.com

225+ Agreements
We have 225+ direct-to-carrier agreements with operators worldwide.

15,000+ Customers
Customers in over 60+ countries, across a great variety of industries.

200+ Employees
More than 180 employees speaking over 20 languages based in the Americas, Europe & Asia.

How do we achieve that?

CONTACT CENTER
Customer Support

ON DEMAND
Ride hailing

2FA
Banking

NOTIFICATIONS
Logistics

MB AIR
Travel

MARKETING
Lead Generation

NOTIFICATIONS
Alerting

ON DEMAND
Food delivery

MessageBird

We’re hiring!

SREs, developers, infrastructure engineers and more!

https://www.messagebird.com/careers

Primer on benchmarking

For those unfamiliar with it

What is benchmarking?

“Benchmarking is the practice of comparing business processes and performance metrics to industry bests and best practices from other companies. Dimensions typically measured are quality, time and cost.

Benchmarking is used to measure performance using a specific indicator (cost per unit of measure, productivity per unit of measure, cycle time of x per unit of measure or defects per unit of measure) resulting in a metric of performance that is then compared to others.”

- Wikipedia

Benchmarking basic rules

- Reproducible
  - Define a process (choose a product: sysbench, tpc-c, ycsb, etc)
- Unit of measure
  - Choose unit(s) of measure (tps, iops, GB, etc)
- Baseline
  - Set a baseline (current server, normal workload, etc)
- Run benchmark
  - Collect data (output, iotop, prometheus, pmm, etc)
- Compare
  - Interpret the collected data against the baseline
- Reliability
  - Repeat the benchmark multiple times to obtain a reliable outcome

Benchmarking stereotype?

Image: Wikimedia

Why do we benchmark?

- Compare products / services
- (Long term) planning of resources
- Predict workloads
- Workload tuning

Why should you compare products?

Image: Youtube

Why should you do resource planning?

- Determine your limits / boundaries
- Cloud: spawn many instances, compare and choose the best one(s)
- Good indication on when should you upgrade or scale out
- Use as input for (long term) strategic planning
- Any deviation means something went bad
  - New deployments
  - Hardware revisions

Why do we want to predict workloads?

- Explore limitations to systems
- Improve systems
- Systems may not fail 100%
  - Partial failure can have devastating effects
    - Cause for inconsistency
    - Depending subsystems may become instabile
    - Cascading effects due to slow response times
- Isn’t this just load testing?
  - Compare normal workload against abnormal/unexpected workloads

Why do we need to tune for workloads?

- Every workload is different
- Read or write?
- Inserts?
    - Frequent or infrequent updates?
    - Batch processing
- Durable or improved performance?
- Rinse and repeat

Why doesn’t everybody benchmark then?

- Afterthought
- Time consuming
- Not always beneficial
- Works out of the box
- Most people will never exhaust all their resources

When not to benchmark?

- Incomparable products (apples vs oranges)
- Differentiation in (measured) units
- Truthfulness / trustworthiness

When not to benchmark?

- Incomparable products (apples vs oranges)
- Differentiation in (measured) units
- Truthfulness / trustworthiness

Compare products

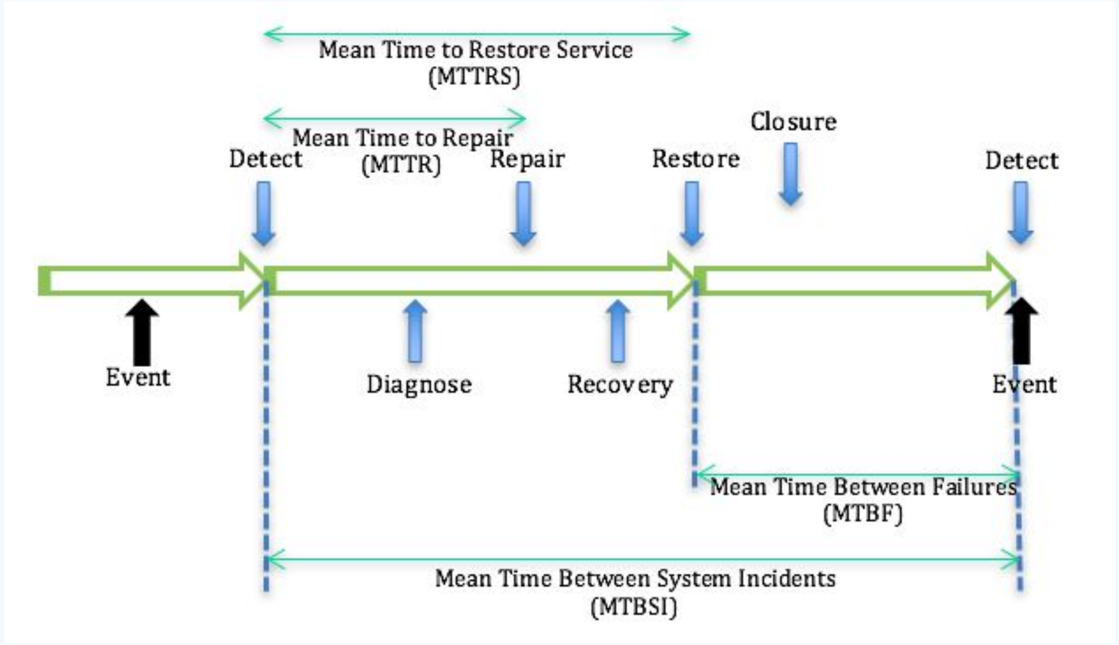
Make a fact based decision

Example: Benchmark Galera State Transfers (SST)

- Galera is an auto recovering clustered system
  - Self healing
- Incremental State Transfer (IST)
  - Copies only missing transactions from Gcache
  - Quick recovery
- State Snapshot Transfer (SST)
  - Copies state of entire node
  - Rsync, mysqldump or xtrabackup (or mariabackup)
  - Slow recovery that can take a very long time to finish

What is Mean Time To Restore Service? (MTTRS)

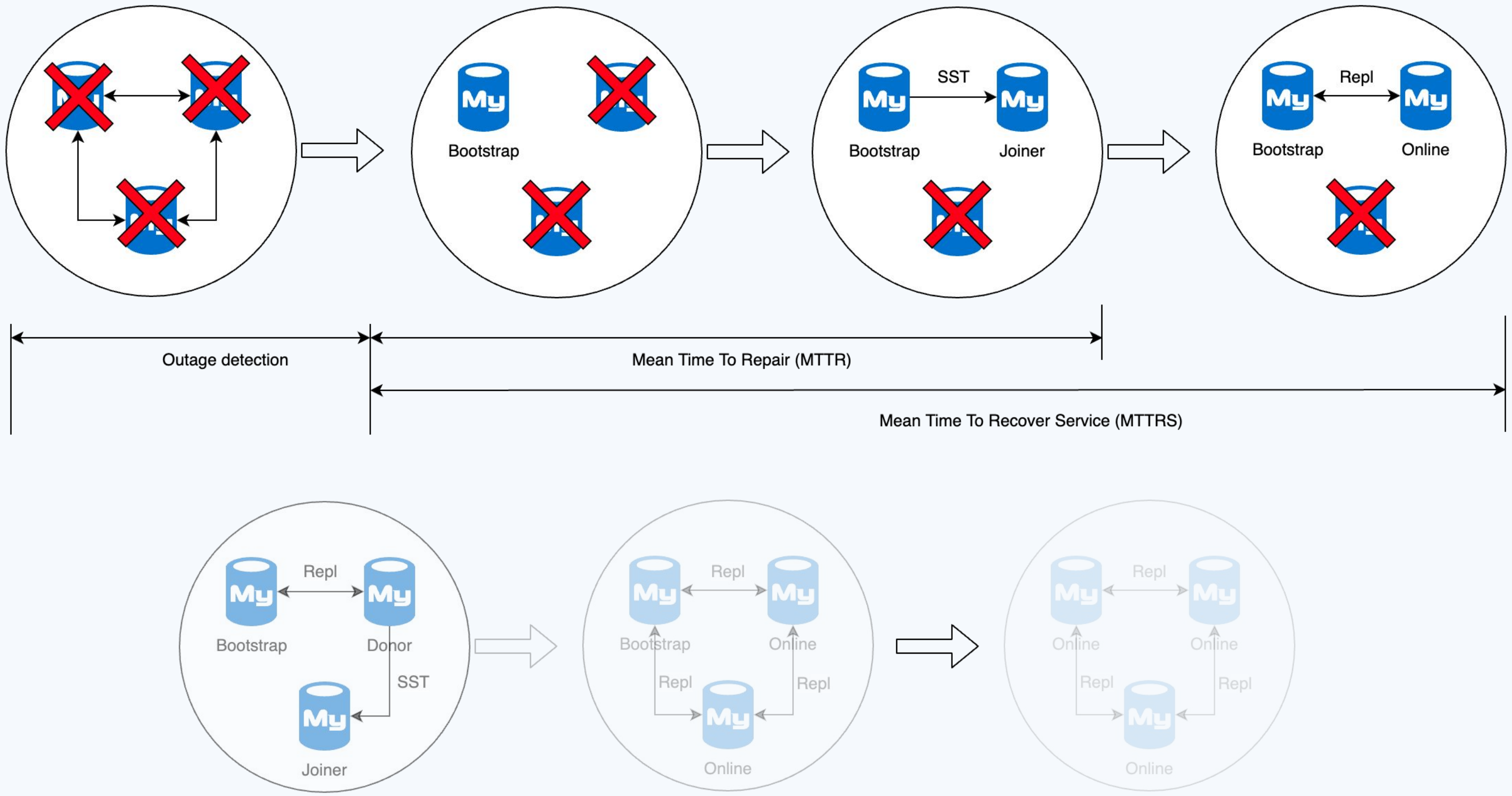
- Time between failure and restoring your service



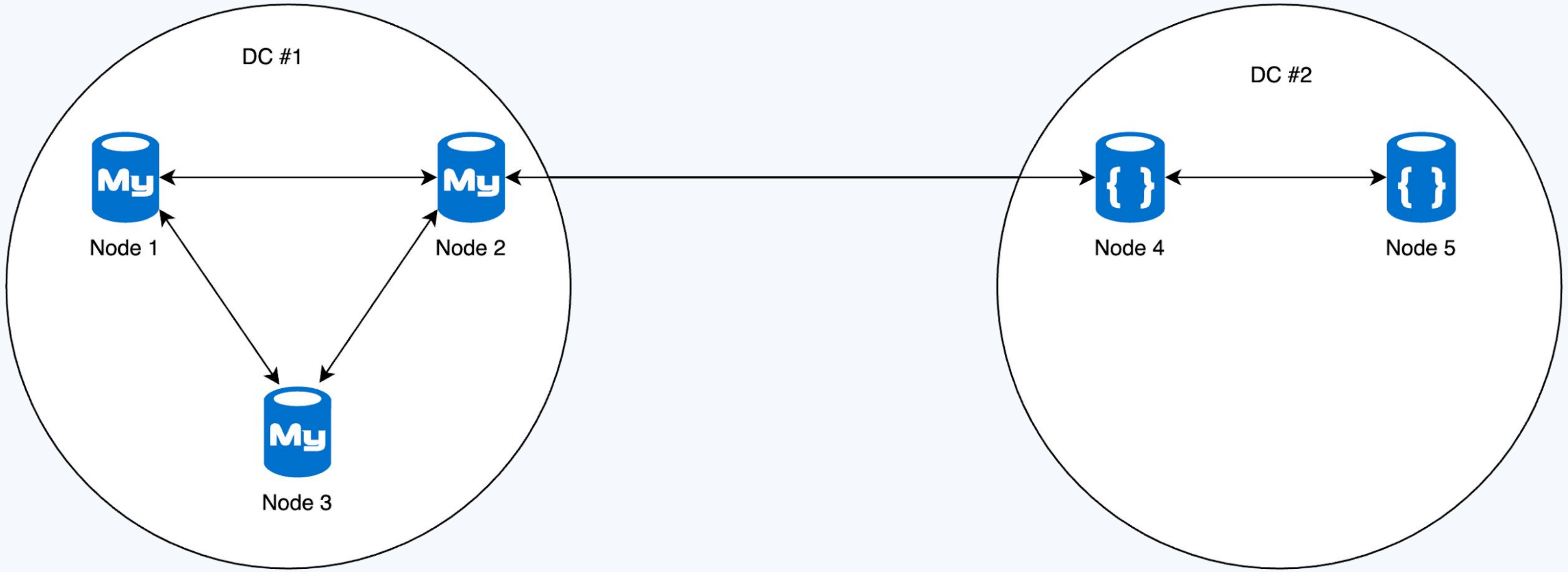
What is Mean Time To Restore Service? (MTTRS)

- Time between failure and restoring your service
- Restoring means the system functioning has been restored
- how quickly can you recover from an outage
  - The longer it takes, the more it will cost you
- Examples:
  - Restore single node (from backup)
  - MySQL master failover via Orchestrator
  - Galera cluster recovery

Example: Galera State Transfers (SST)



Problem: cross region Galera



Problem: cross region Galera

- Nodes distributed over two DCs
- Single node leaving
  - Mostly IST, but when GCache is too small this triggers SST
- Power failure in one DC causes full cluster recovery
  - Bootstrap from the most advanced node (DC #2)
  - Too small GCache, corrupt grastate.dat or unclean shutdown triggers SST
- 1TB will take 3 hours to transfer on 1Gb
  - 450GB will take 3 to 4 hours on 400Mbit
  - MTTRS (at least) > 3 hours
- How to improve this?
  - Fix automation
  - Faster network ;)
  - Make GCache bigger
  - Increase GCache durability
  - Make the transfer smaller

Problem: GCache size and durability

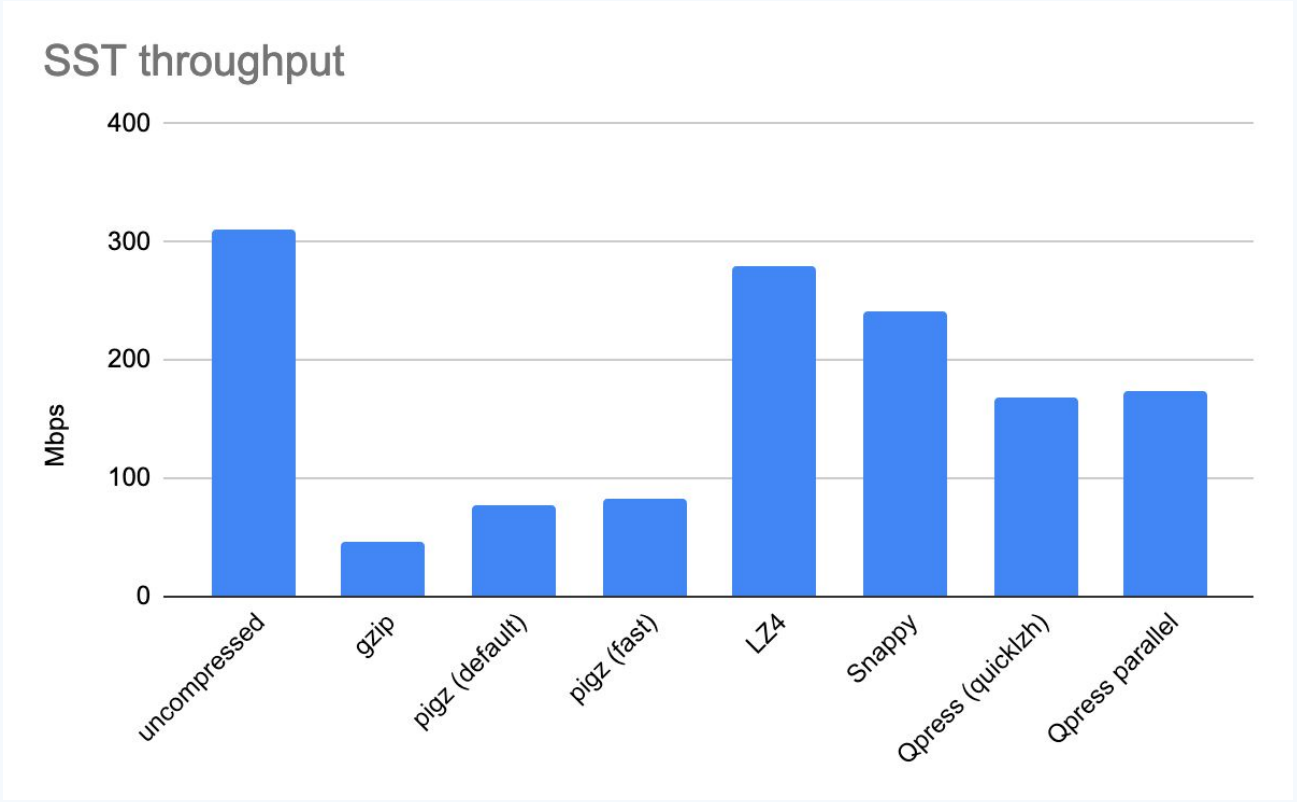
- **GCache durability ~45 minutes**
  - **binlog-row-image=full (when set to minimal ~4 to 5 days)**
  - **gcache.size=4GB**

```
CREATE TABLE product (
  pid BIGINT NOT NULL AUTO INCREMENT,
  title varchar(255) NOT NULL,
  description TEXT NOT NULL,
  specifications TEXT NOT NULL,
  stock INT NOT NULL DEFAULT 0,
  price INT NOT NULL DEFAULT 0,
  image_1 VARCHAR(255) NOT NULL,
  image_2 VARCHAR(255) NOT NULL,
  image_3 VARCHAR(255) NOT NULL,
  thumbnail_1 VARCHAR(255) NOT NULL,
  thumbnail_2 VARCHAR(255) NOT NULL,
  thumbnail_3 VARCHAR(255) NOT NULL,
  ...
  another 50+ columns
  ...
PRIMARY KEY(nid)) ENGINE=InnoDB;
```

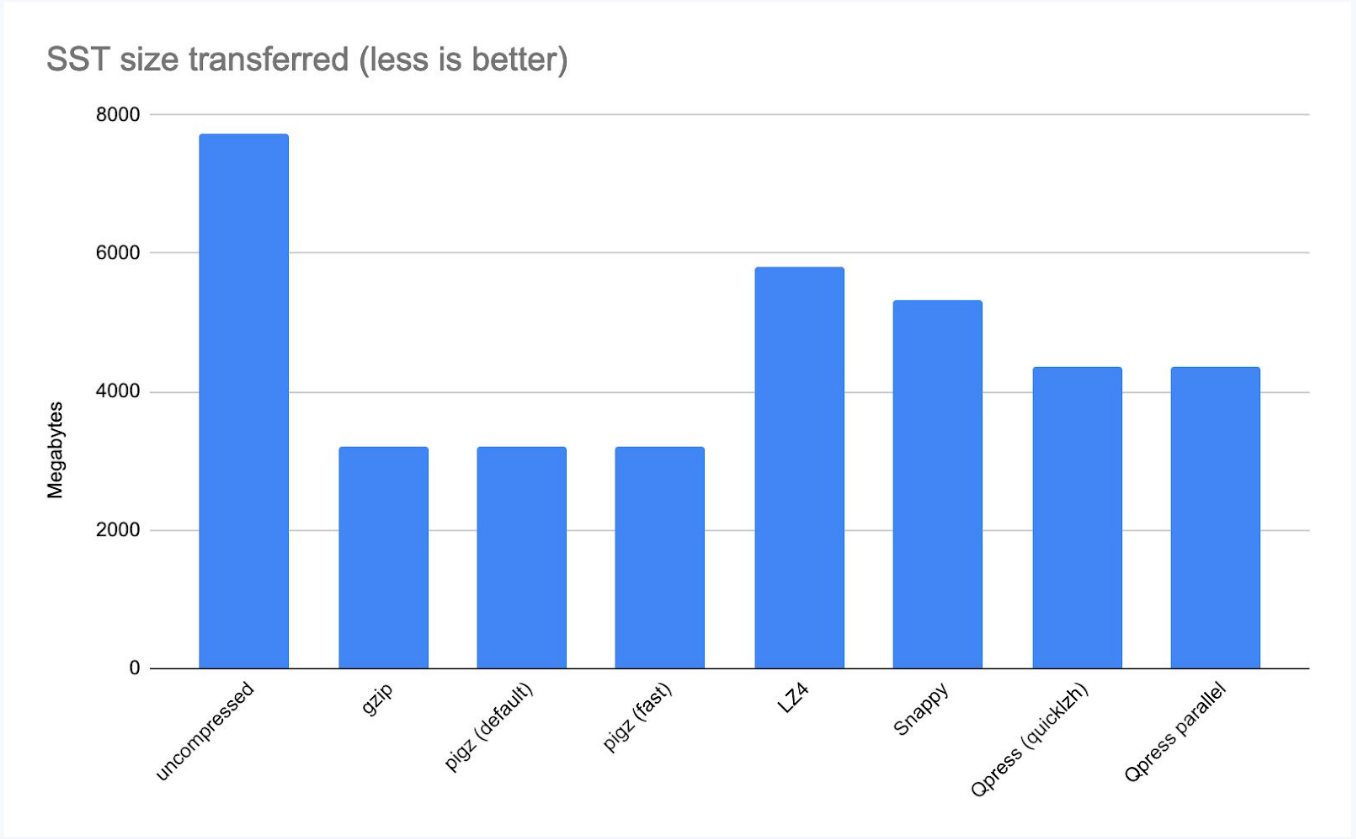
Solution: benchmark State Snapshot Transfers

- Compressors
  - Qpress: LZH based internal compressor
  - Gzip / pigz: gzip and parallel gzip
  - LZ4: multi core compressor
  - Snappy: multi core compressor
- Limit network bandwidth for consistency
- Trigger SST
- Uncompress & recover

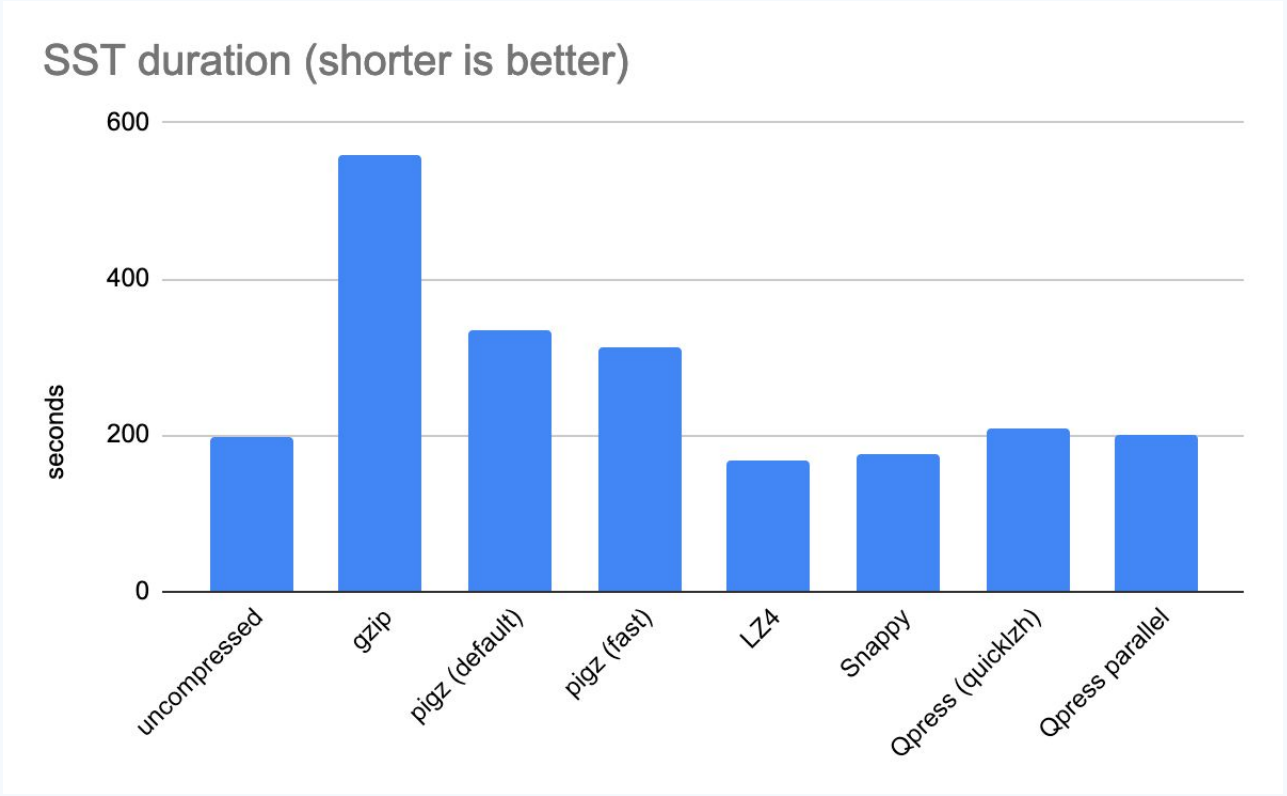
Solution: benchmark results throughput



Solution: benchmark results transfer size

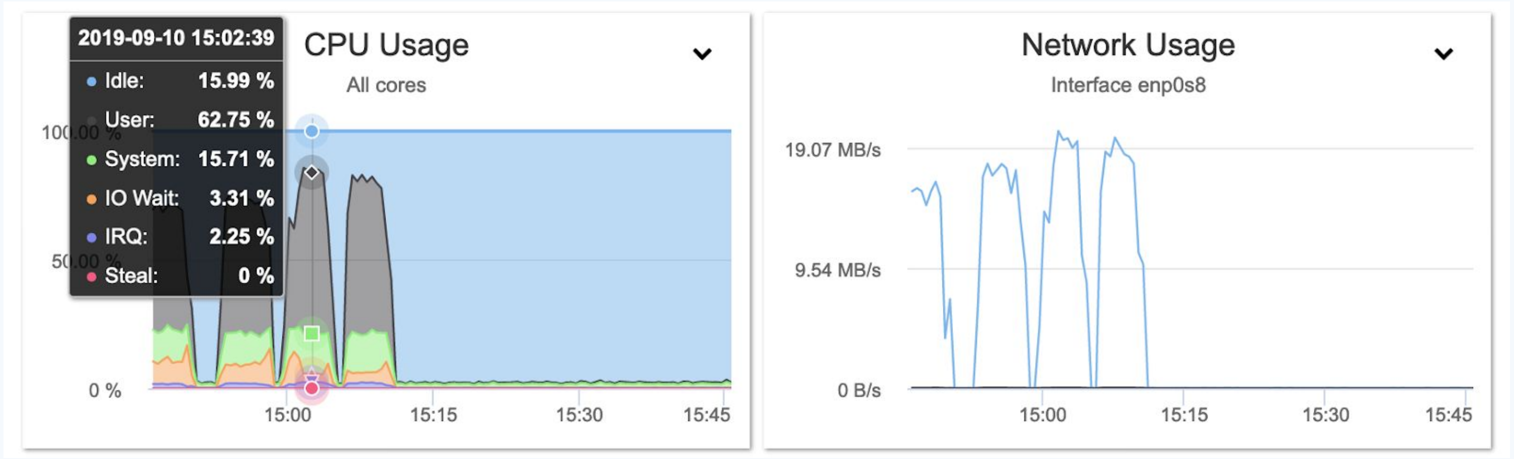


Solution: benchmark results duration



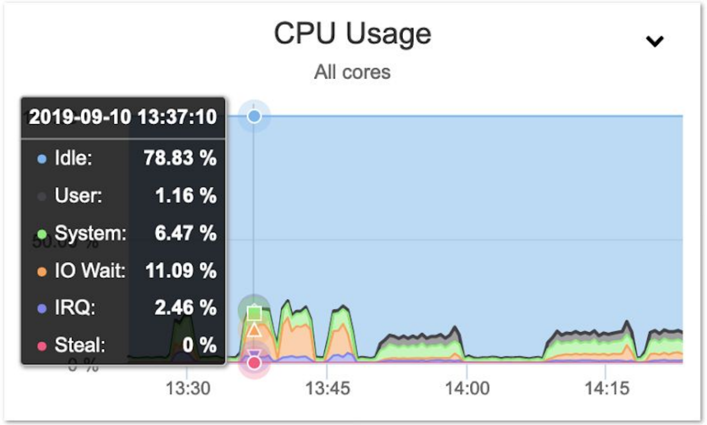
Solution: getting the data to the other end

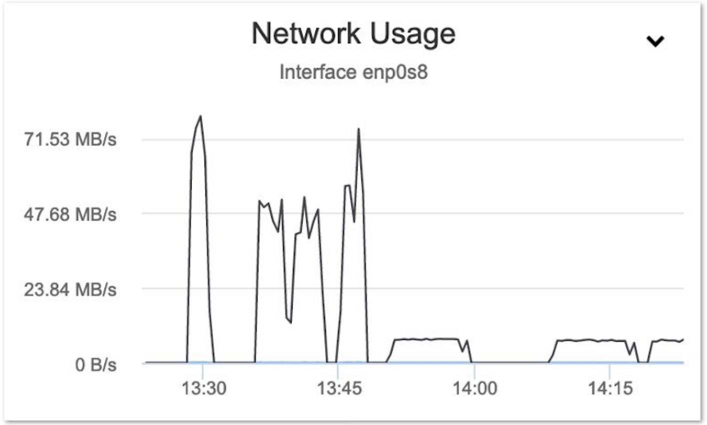
- As seen the most efficient is Snappy or LZ4
- Surprised?
    - Pigz is slower because it's heavy on CPU



Solution: getting the data to the other end

- As seen the most efficient is Snappy or LZ4
- Surprised?
  - Pigz is slower because it's heavy on CPU
  - Uncompressed is quick because there is no (de)compression phase





Solution: getting the data to the other end

- As seen the most efficient is Snappy or LZ4
- Surprised?
  - Pigz is slower because it’s heavy on CPU
  - Uncompressed is quick because there is no (de)compression phase
  - Qpress does not have the extra overhead of an external compressor but requires an additional decompression step
- In practice Snappy lowered the SST duration by 20%

Side notes

- Qpress unsupported on mariabackup (fork of xtrabackup 2.4)
  - MDEV-15069 and MDEV-17600
- Some VPNs may have compression enabled
- Automated cluster recovery can work against you
  - May recover nodes sequentially

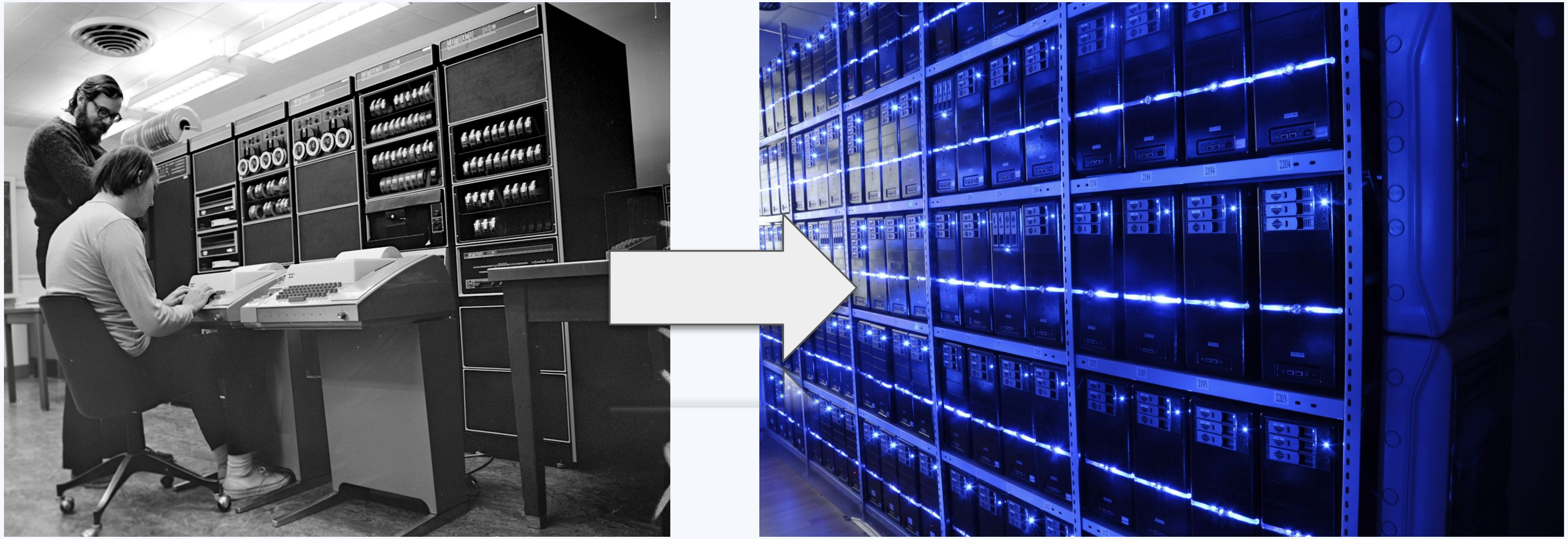
Resource planning

Leverage benchmarks to do resource planning

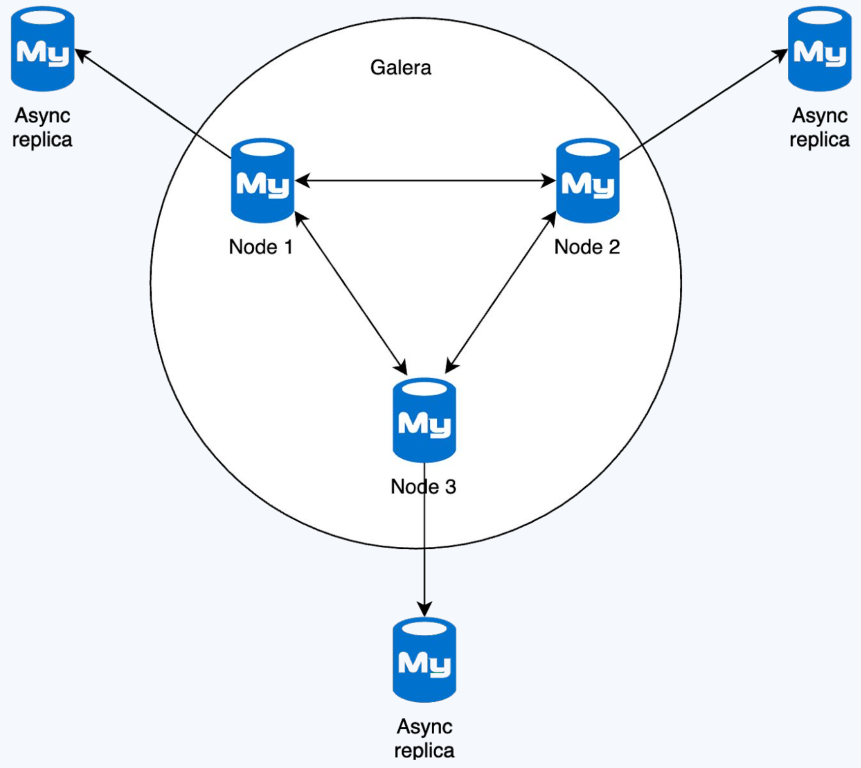
Example: hyperconverged infrastructure

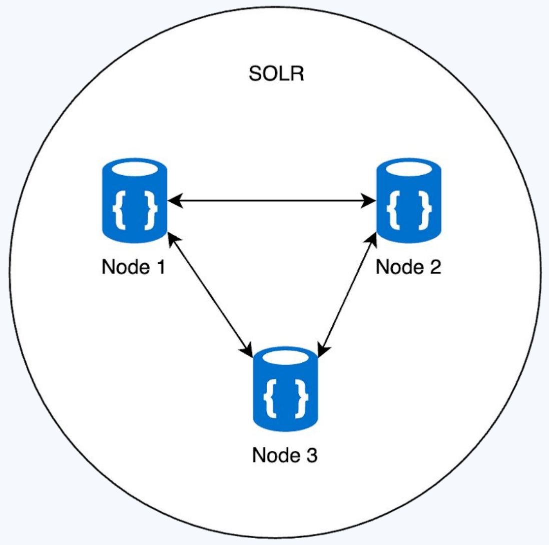
- Hardware consolidation
- New hardware bought for ~100K per cluster
- Storage is all SSD
- Hyper-converged infrastructure
- Super easy to administrate and spawn new VMs
- Distributed storage layer synchronously writes data to other hypervisors
    - Great for disk snapshots
- When a hypervisor fails, VM will be spawned on another
    - For databases recovery could take minutes
- Sounds familiar?

Example: how it was presented



Example: Galera and SOLR topology (single DC)

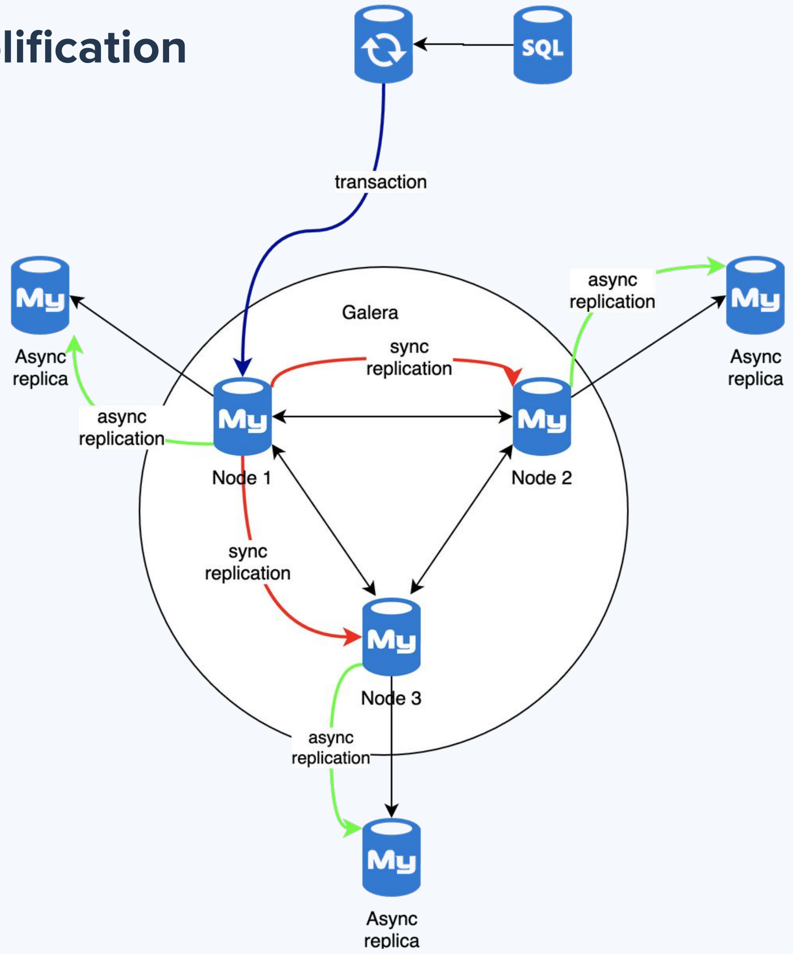




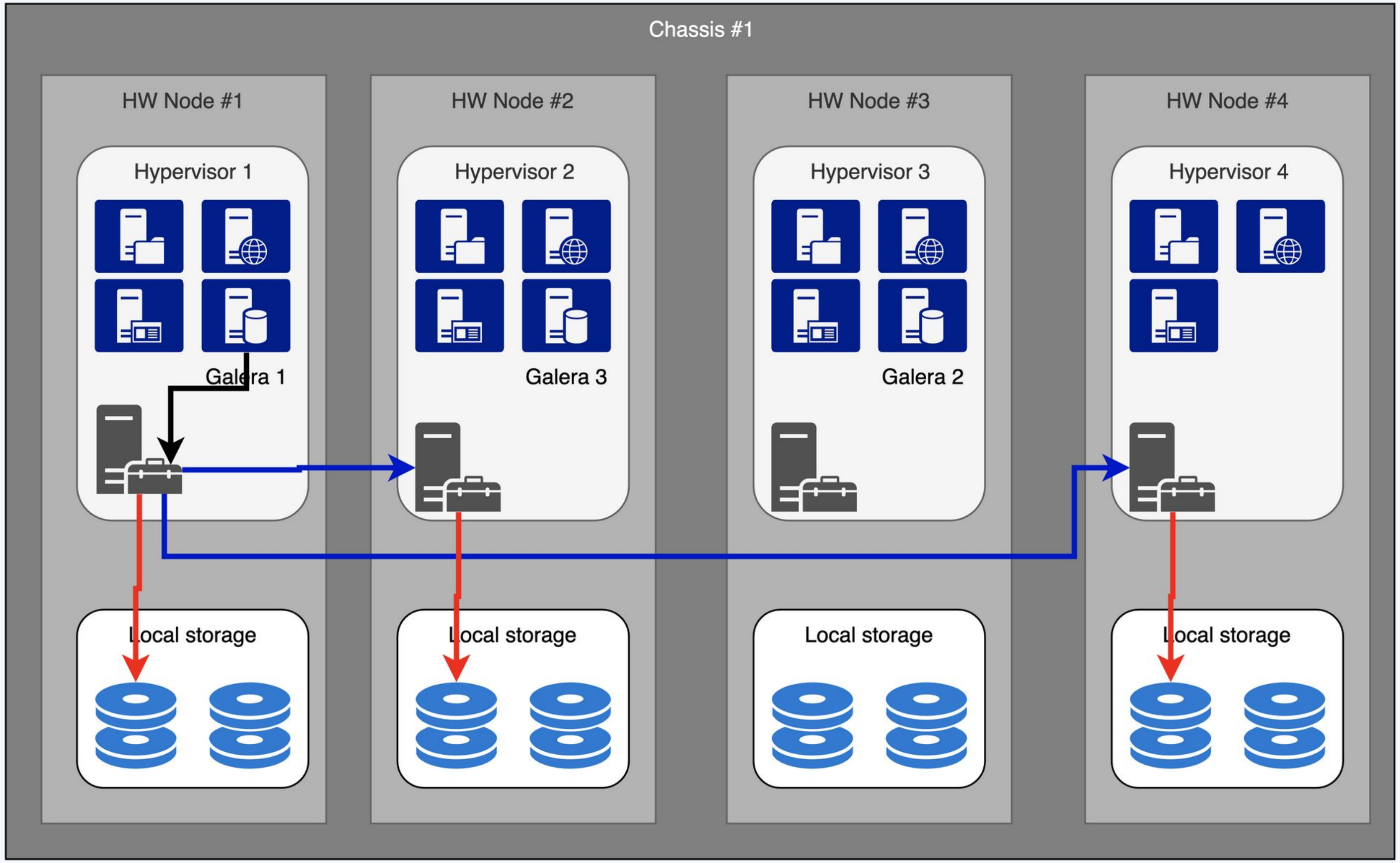
Problem: instability on Galera and SOLR

- Both databases are mostly used for read-only
- Both are clustered to allow fast failover
- Under heavy write workloads both become unstable
    - Updates on pricing / stock
    - Cart operations rolled back
- Pricing and stock update process write to both Galera and SOLR
- Many transactions are rolled back
- Application doesn’t handle rollback of cart transactions well

Suspect: write amplification



Suspect: how hyperconverged hypervisors write data



Suspect: the underlying issue

- Distributed storage (Replication factor 2)
  - Every transaction creates 3 writes to disk
  - Every byte written to any hypervisor will create two additional network writes!
  - 1 transaction = 3 nodes (3 writes locally + 6 writes over the network) = 27 writes
  - Writing large updates will saturate network
  - Replication factor can only be changed on cluster level
- Internal + external network shares the same interface
- Hyperconverged means you have mixed workloads
- Consolidation meant one cluster per DC

Proof: benchmarking the underlying issue

- Repeat the underlying issue
- Sysbench against a 3 node Galera cluster
  - 3 nodes on old hardware (VMWare) + 1 dedicated sysbench host
  - 3 nodes on hyperconverged new cluster + 1 dedicated sysbench host
- Differentiate the number of threads

Proof: benchmarking the underlying issue

- New hyperconverged infra couldn't handle more than 16 (sysbench) threads
- IO errors on sysbench fileio test > 4 threads
- Tuning hyperconverged hardware for Galera
  - Tuning the storage layer (more memory/cpu)
  - Enable jumbo frames
  - Interface bonding (20Gb instead of 10Gb)
- Maximum achievement
  - Comparable performance with old hardware
  - Slower than Google Cloud + Persistent Disks (also network disks)
- 1 transaction spawns 3^2 network operations which causes 3^3 writes
- Doubling network bandwidth will only allow you 11% increase in transactions
- Hyperconverged infrastructure + distributed databases simply don't scale

What difference would a benchmark (upfront) have made?

- Different hardware / virtualization
- Different distribution of workloads
- Faster network
- Abandon distributed databases
- Enable eventual consistency on SOLR

Predict workloads

What happens if you overload systems?

Example: SSP (Sharded Storage System)

- In 2010 sharding was tedious
  - Vitess was only offered as open source in 2015
- Created our own sharding abstraction layer
  - Shard on function
  - Shard on user
  - Cluster information by user
- Differentiate writes
  - Optimistic (asynchronous)
  - Pessimistic (synchronous)
- Any storage layer possible
  - MySQL
  - MongoDB
  - Couchbase

Example: the ingredients

- Written in Erlang
    - Great for parallelism!
- MySQL asynchronous replication (MySQL 5.6)
    - Circular replication
    - Data written interleaved within the same schema
- Both individual masters handled up to 30% workload
    - Benchmarked a single master

Problem: 30% + 30% > 100%

- Due to an outage one master has to perform both write workloads
  - Can handle 60% workload easily
- Asynchronous replication single threaded per schema
  - Operations on the master happen in parallel
  - On the slave all queries are funneled through single thread
- Update and delete queries slower than on the master:
  - Updated/deleted data is “cold”
  - Require a lookup on (spinning) disk
  - Then modify/delete data

Solution: un-funnel

- Migrating data will read + delete data
  - Slow due to cross shard transactional
- Prefetching
  - Rewrites an update / delete query to a SELECT to warm up bufferpool
  - pt-slave-prefetch single threaded
  - Bash prefetch oneliner
- Chosen solution
  - Shutdown slave (will never catch up) and “merge” shards
  - Create new shards on Galera
  - Migrate data (one by one) to new Galera shards

What difference would a benchmark (upfront) have made?

- Revealed the replication weakness
- Change design
    - Bucket per schema
        - Multi threaded replication
    - Pre-shard
        - Many (small) shards on the same hosts
- Showed necessity for shard operation tools
    - Bulk migrations
    - Split shards

Questions? Don’t Be Shy

art.vanscheppingen@messagebird.com

Thanks for your attention!