Booking.com

MySQL at Booking.com

Nicolai Plum

Booking.com

Booking.com


100M
monthly active app users
232M+
verified guest reviews and
24/7
customer service in 44
languages and dialects
Since 2010, Booking.com has welcomed
4.5B+
guest arrivals
28M
total reported listings worldwide
6.6M
options in homes, apartments and other unique places to stay
140 offices in 70 countries over
5,000 employees in Amsterdam
30
different types of places to stay,
including homes, apartments, B&Bs, hostels, farm stays, bungalows, even boats, igloos and treehouses
155,000
destinations around the world
Car hire available in 140+
countries and pre-booked taxis in
over 500 cities across 120+
countries
B.

MySQL for 18+ years: what?

- Core transaction processing
- Payments & Billing
- Front-end content
- Partner and Customer Support
- Internal tools and controlplanes

Booking.com

MySQL Transaction Processing

- No transactions == no revenue
- No payments == less revenue
- Transactions: the critical MySQL use

Booking.com

MySQL analytics

- Near real-time data analytics
  - Monitoring, reputation/security
- Medium-scale (1TB) analytics on transaction data
- … anything more is for Big-data systems
- Analytics support transaction business

Booking.com

MySQL for 18+ years: why?

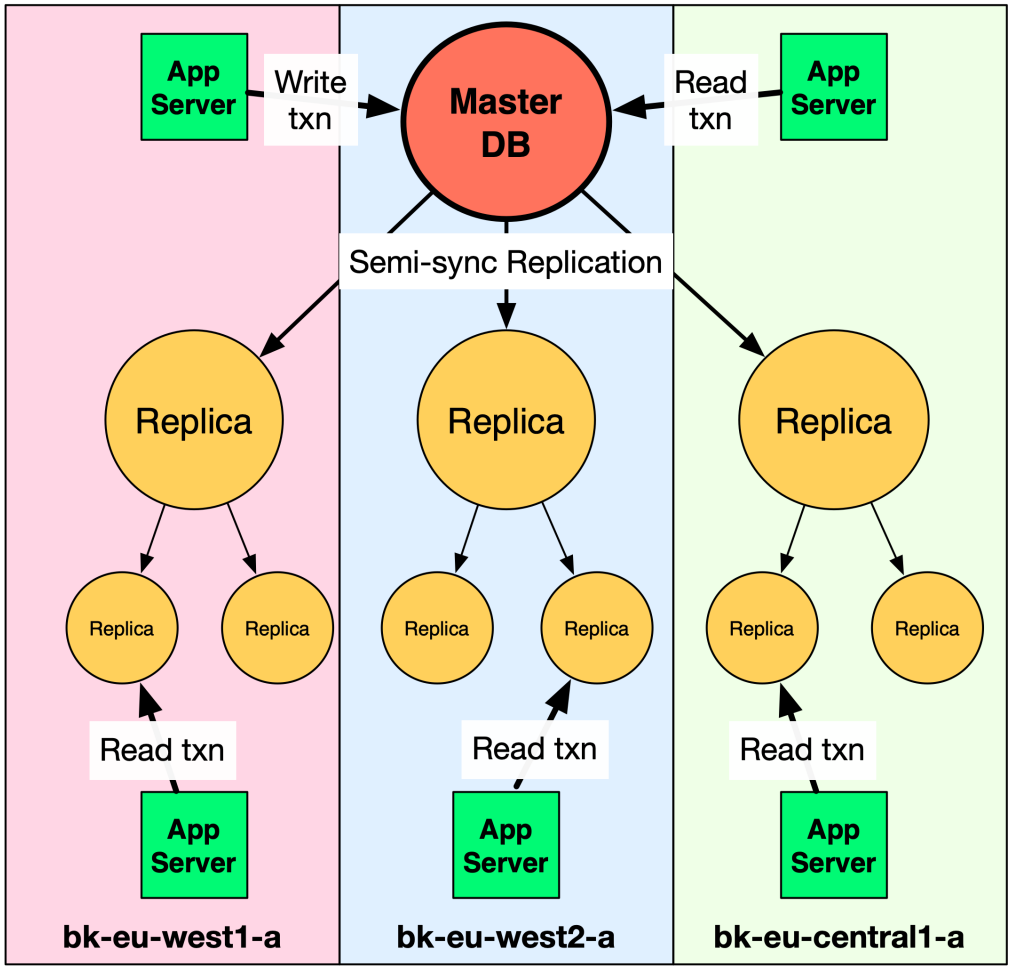
- Read scale-out
- Speed
- Good reliability
- Easy, flexible administration

Booking.com

MySQL – Read scale-out

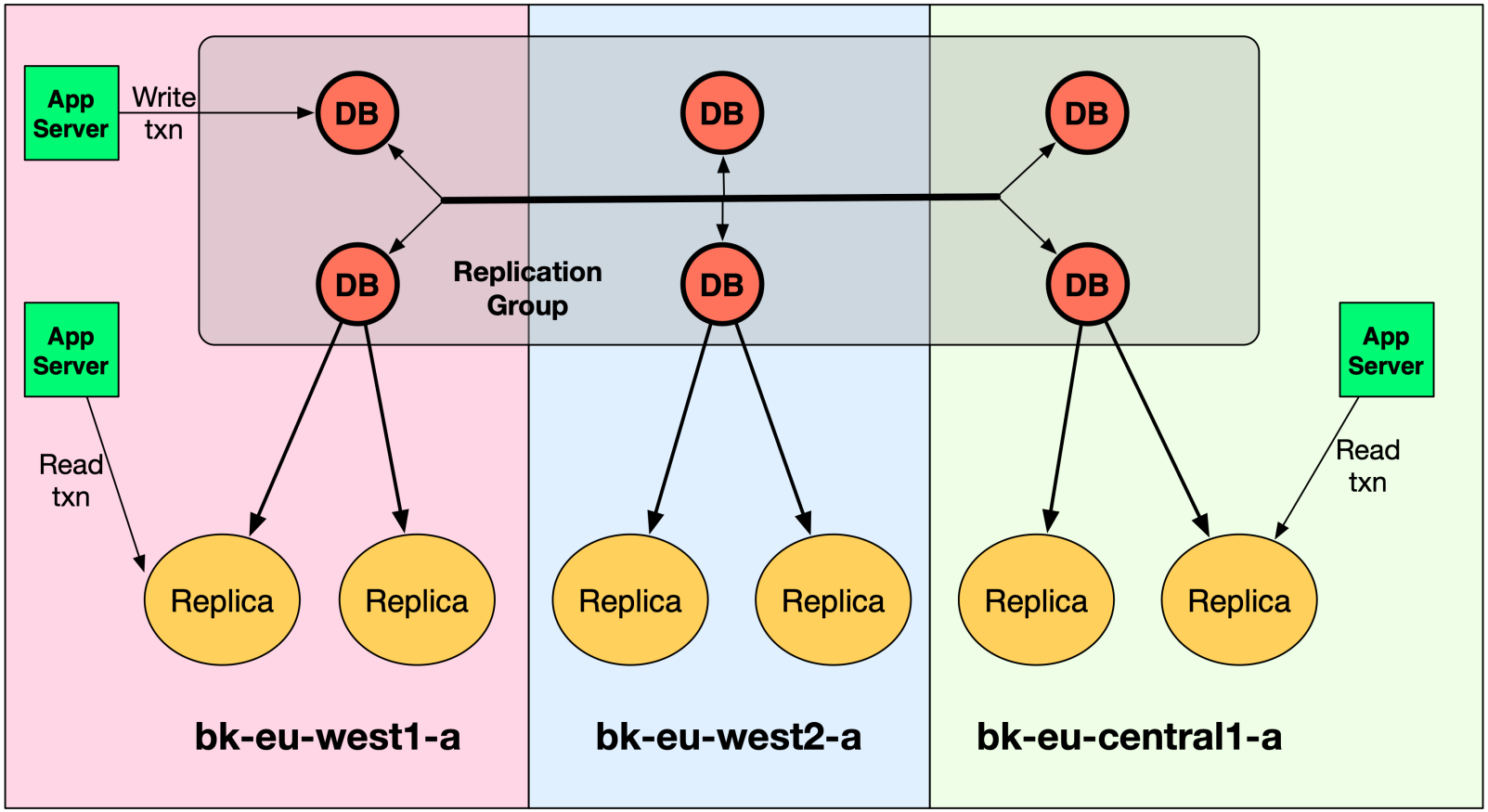
- Cross-AZ (AWS/OCI equivalent: Cross-region)
- >100 replicas in some chains chain
- Service discovery based on Zookeeper
- Online Travel is **very read-intensive**

MySQL (Classical) Replication



Booking.com

MySQL Group Replication



Booking.com

MySQL - Speed

- <1ms point query read
- Range queries are fast
- Working dataset in memory
  - Memcache-like speed
  - Rich, individualised content is not cacheable
- ~2000 write qps (non-Group-Replication)

MySQL On Any Platform

- Managed managed service
- Managed cloud instances
- Managed containers
- Hybrid multi-cloud
- Same reliability, user interface, controls

Booking.com

Managing MySQL at scale: Relentless automation

- New databases
- Replacing broken instances
- Patching and upgrades
- Autoscaling pools according to user needs
- … across thousands of instances.
- On every platform or managed service

Booking.com

Replace **Break/Fix**
with
Preventative Maintenance

Booking.com

Managing MySQL at scale: Technology for On-Premise

- Provisioning infrastructure
- Puppet, Ansible, bespoke management controlplane
- Authentication, Authorisation, Audit tools & policies
- Self-healing systems & topology (Orchestrator)
- Monitoring, Alerting, SLO reporting, Incident management processes

Booking.com

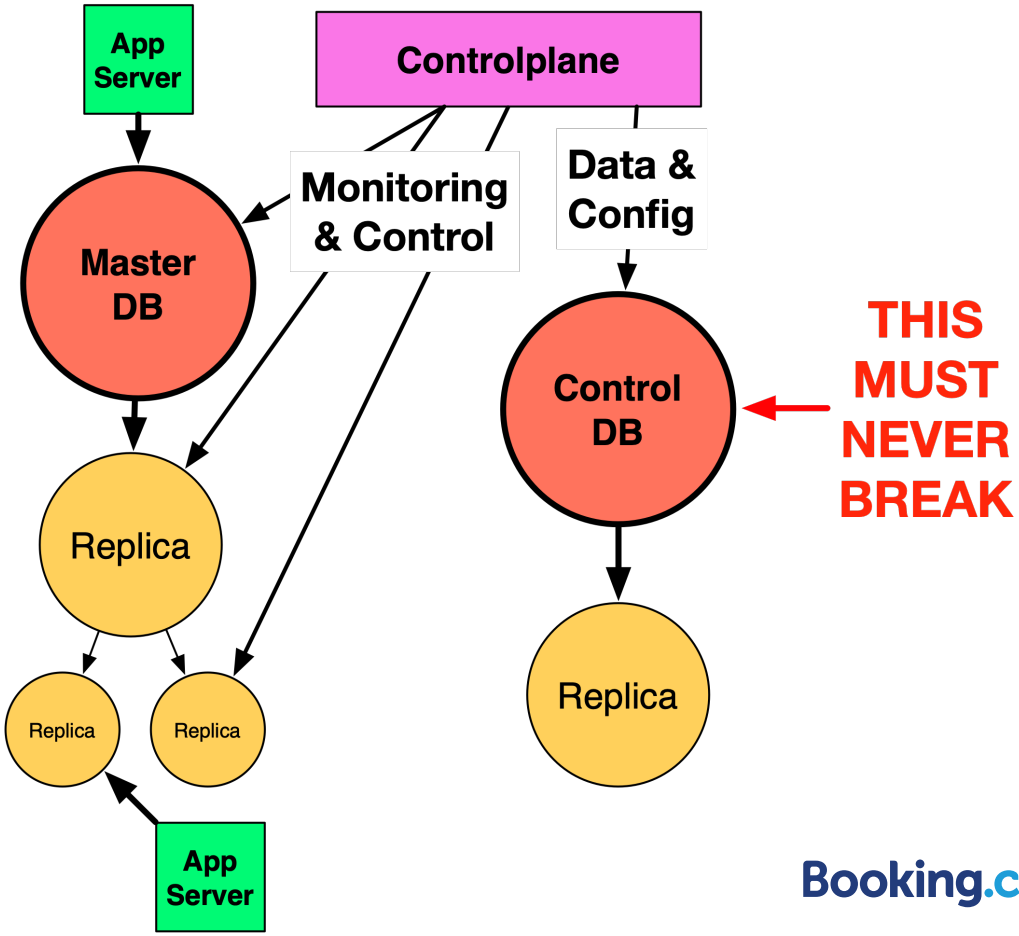
Managing MySQL at scale: Technology for Cloud

- Provisioning & Service API Interfaces
- IAM, audit, policies
- Self-healing system & topology
- Monitoring, Alerting, SLO reporting, Incident management processes

Booking.com

Managing MySQL at scale: It's Databases all the way down!

- Controlplane uses MySQL?
- … which must never break?
- … or we have a circular repair dependency?



Booking.com

Unbreakable database?

- Must be **Non-Stop** in the short term
- **Clustered databases** keep working after single instance failures (without intervention)
- Guarantee that automation databases will be up as long as automation maintains them in the background

Experience (general MySQL)

- Single instance rarely crashes
- InnoDB data durability is bulletproof
- Replication can break, can *usually* be fixed
  - Sometimes requires heroics
- Replication delay requires full-stack management
- Grant management commands scale badly
- Downgrade is impossible

Experience (Group Replication)

- When it works, it's great – Non-Stop MySQL
- When it breaks, it is complex to fix
- MySQL Shell auto-repair is excellent
- Quite a few bugs, some serious (cluster-wide failures)
- Upgrade is awkward. Downgrade is impossible.
- **Cluster member changes disturb whole cluster**
- Protocol and some config changes not possible online
- Latency increase, QPS decrease (compared to single master)

Cluster & Topology Reliability

- Service discovery for clients
  - Writer and readers
- Auto repair
  - Rejoin failed members, or replace missing members
- Upgrades & patches
  - Switch to new master/primary, add new, remove old.
- Topology management: re-attach replicas

System Reliability Engineering

- What SLOs do you need?
  - How about: Write 99.95% Read 99.99%
- Topology changes are **Frequent**
- GR member/replica attach/detach must be low impact
- Connection counts can be very high, >10k
  - Not every stack has connection pool
  - External connection pooling: ProxySQL sidecar
- MySQL Community version allows code inspection and commonality across platforms
  - Plugins for auditing and custom behaviour

SRE Agility

- Automated progressive upgrades
- Error & performance observability
- Testing methods to bound the risk of new deployments

Booking.com

Observability – tracing

- Objective: full-stack tracing
- In place…
- … except Database
- No Open Tracing support
- Need to trace all database activity from an example business action
- Need hooks to inject tracing ID from application, and send tracing data to Observability systems

Data operations online

- Online (not Instant) DDL blocks replication
- Large tables need table rewrite to reclaim space
  - No background table free space compaction and disc space reclamation
- InnoDB compression is limited
  - Can’t compress data in background
  - Efficiency could be better
  - Filesystem holes cause management problems
- Need LSM storage for 5x space efficiency, still has to be fast on small select/update

MySQL Binary Logging

- Binlog is essential, but a cost overhead
- Binlog compression is excellent
- GTID greatly improves replication reliability
- No standard binlog consumer library means irregular support for any of this in external applications

Dataset scaling

- No future path for dataset bigger than one instance
- Constant compute power and growing data is unsustainable
- A long-term pain point
- Strong reason to use other databases

No very good data scale-out products

- MySQL NDB Cluster
  - Limited SQL, complex, we crashed the cluster
- MySQL Heatwave
  - Full SQL, limited resilience & deployment options
- Vitess
  - Very limited SQL, complex operations
- TiDB
  - Less limited SQL, complex, full-featured, maturing
- CockroachDB
  - Less limited SQL, quite promising

?

nicolai.plum@booking.com