PALOMINODB

OPERATIONAL EXCELLENCE
FOR DATABASES

Tricks and Treats for MySQL in Amazon Cloud

Vladimir Fedorkov

NYPHP, October 28th 2013

www.palominodb.com

Let’s meet each other

- Performance geek
  - MySQL, Sphinx, RDS
  - http://astellar.com
    - Still hope to update at least once in a year
- Twitter @vfedorkov
- Now fighting with clouds

Credits for this talk

- NYPHP.org
- PalominoDB Remote DBA boutique
  - http://palominodb.com
- Ben Black, Mark Filipi, Krzysztof Książek and Palomino EU team

Day-to-day MySQL operations

- Hardware faults
- Software bugs
- Reliable backup and restore
- Monitoring
- Upgrading and replication setup
- Networking and Security

Amazon for MySQL

- It's called RDS
  - Relational database service
- New features are coming fast
- MySQL, Oracle and MSSQL
- Now using MySQL 5.6
  - Modified source code
    - You'll never know what's different

What you can do?

- Create MySQL instances
- Create replicas
- Promote replica to be master
  - Detach from master
- Force backup and point-in-time recovery

What you can't do

- Access MySQL box using SSH
- Manage replication
  - It's now better with bin-log access
  - Set up cross-region replication
    - RDS team is working on that
- Run xtrabackup
  - Mysqldump only
    - Use parallel dump to speed up

Settings

- Several regions are available
- Parameter groups
- RDS types
- Network security
- Storage

Regions and availability zones

- US
  - East 1 (Northern Virginia)
  - West 1 (Northern California)
  - West 2 (Oregon)
- EU: Ireland
- Asia Pacific: Singapore, Tokyo, Sydney
- South America: São Paulo
- More to come across the world

Availability zone

- Several AZs within region
- You can create AZ-enabled master
  - Will help in case of server crash
    - Might not help to keep replication up
- You can create replica in different AZ
  - You actually should ☺

Storage

- PIOPS and non-PIOPS disks
  - Architecture is completely different
- You are in cloud

RDS Sizes

- **Micro DB Instance**: 630 MB memory, Up to 2 ECU (for short periodic bursts), 64-bit platform, Low I/O Capacity, Provisioned IOPS Optimized: No
- **Small DB Instance:** 1.7 GB memory, 1 ECU (1 virtual core with 1 ECU), 64-bit platform, Moderate I/O Capacity, Provisioned IOPS Optimized: No
- **Medium DB Instance**: 3.75 GB memory, 2 ECU (1 virtual core with 2 ECU), 64-bit platform, Moderate I/O Capacity, Provisioned IOPS Optimized: No
- **Large DB Instance**: 7.5 GB memory, 4 ECUs (2 virtual cores with 2 ECUs each), 64-bit platform, High I/O Capacity, Provisioned IOPS Optimized: 500Mbps
- **Extra Large DB Instance**: 15 GB of memory, 8 ECUs (4 virtual cores with 2 ECUs each), 64-bit platform, High I/O Capacity, Provisioned IOPS Optimized: 1000Mbps
- **High-Memory Extra Large DB Instance** 17.1 GB memory, 6.5 ECU (2 virtual cores with 3.25 ECUs each), 64-bit platform, High I/O Capacity, Provisioned IOPS Optimized: No
- **High-Memory Double Extra Large DB Instance**: 34 GB of memory, 13 ECUs (4 virtual cores with 3,25 ECUs each), 64-bit platform, High I/O Capacity, Provisioned IOPS Optimized: No
- **High-Memory Quadruple Extra Large DB Instance**: 68 GB of memory, 26 ECUs (8 virtual cores with 3.25 ECUs each), 64-bit platform, High I/O Capacity, Provisioned IOPS Optimized: 1000Mbps

Parameter groups

- Contains all MySQL settings
- Default group needs to be tuned
  - Create custom and tune it
  - Number of parameter groups might be limited
- Some things you can’t change
- Static variables change requires restart
  - Dynamics a not
    - You can see it on the RDS console

Recent improvements

- Binary log access
- MySQL 5.6
- Instance rename
- Microsecond precision in query log
- Logs are available from RDS console
- RDS team improving it non-stop

Making MySQL faster

- Welcome to the cloud!
- Hardware might be different
  - Even each benchmark run
- IO is highly relies on network
- CPU & IO limiters are in place
- You don’t know your neighbors

First step

- MySQL configuration
  - 5% settings makes 95% performance
    - Which every query could kill
- What to tune at the first place
  - innodb_buffer_pool_size
    - Set on ¾ memory in RDS by default
  - innodb_file_per_table
  - key_buffer for MyISAM
  - server-id for EC2 and standalone

Second step

- Network
  - skip_name_resolv
    - May affect authentication
  - max_connect_errors
    - FLUSH HOSTS
  - max_allowed_packet
- IO
  - innodb_flush_log_at_trx_commit
  - innodb_log_file_size

Changes to avoid (usually)

- max_join_size
- sort_buffer_size
- tx_isolation
- Remember about 5% settings and 95% performance!

Backups

- What would you do if your MySQL has gone?
  - Replicas-Replicas-Replicas
    - … and binary logs
      - For point-in-time recovery
      - Better to another host
  - expire_logs_days
  - max_binlog_size
- RDS snapshots
- Volume snapshots on EC2
  - Works like LVM backups

Consider worst case

- Limit InnoDB IO capacity on EBS
  - Affecting replica creation time!
- Tune flush_log_at_trx_commit
- Keep tmp_table_size big enough
  - Temporary tables are BAD!
  - There are drawbacks!
- Know your queries!

Queries

- Most likely the cause of MySQL slowness
  - Affect application speed
- Fastest MySQL query is the query that did not MySQL at all
  - Cached
  - Sent to another storage (you name it)
  - Just didn’t run
    - Helps in case of overload

Bad queries: full table scans

SELECT * FROM table …

- WHERE DAY(FROM_UNIXTIME(`ts`)) = 205
- WHERE enabled != 1
- WHERE id NOT IN (1,2,3,…,10)
- WHERE url LIKE ‘%something%’
  - LIKE ‘something%’ is still okay
- ORDER BY RAND()
  - All time classic

Poor index usage

- SELECT * / COUNT(*)
  FROM users WHERE enabled=1
  - Hello half of the table
- How to fix?
  - Pre-aggregation
  - Caches!
    - Not in MySQL query cache!
  - Right time to use external storage (Mongo, Redis, Sphinx, etc)

Temporary tables

- Bad
  - Very bad if they hit the disk
    - Will hit if there are TEXT or BLOB fields
- When?
  - GROUP BY
  - Sub SELECTS (better in 5.6)
  - DISTINCT + ORDER BY

What to do?

- Tune
  - tmp_table_size
  - max_heap_table_size
- Run heavy queries on “reporting” replica
- Rewrite bad queries where possible

When load is really high

- Query cache
- Thread cache
- Table cache
- Slow query log
- wait_timeout
- connection pooling
- Sharding and replication

Keep an eye on it!

- CPU/IO overloads and spikes
- Memory usage
- Broken replicas

When to use RDS?

- When you have relatively small data set
  - Or if you can keep it sharded
- When you can't predict your load
- When you need flexible resources
- When you don't have enough hands to operate MySQL by yourself

Other ways

- Not many
  - as long as you want to keep MySQL close to your web/application servers
- Use EC2 with MySQL
  - + Tungsten
  - + Galera

Questions!

Thank you!

http://palominodb.com
http://astellar.com

Twitter: @vfedorkov