MySQL Database Service

吴炳锡

大纲

- MySQL Database Service 介绍
- MGR 如何实现多点写入
- MySQL HeatWave 架构
- MySQL 和 Heatwave 的结合

1 MySQL Database Service

MySQL Database Service

MySQL Database Service

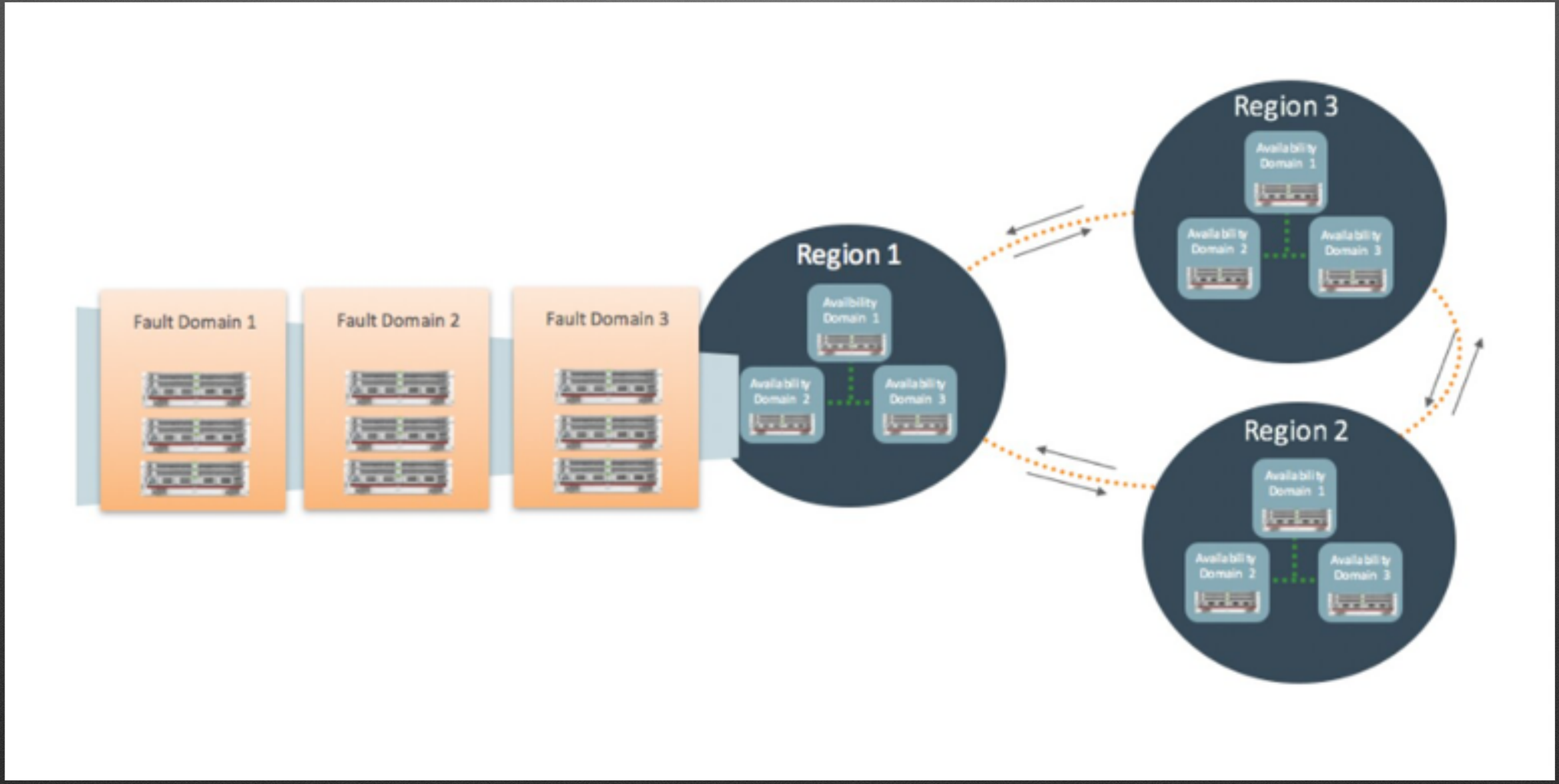
Oracle MySQL Database Service is a fully managed database service that lets developers quickly develop and deploy secure, cloud native applications using the world's most popular open source database. MySQL Database Service is the only MySQL cloud service with an integrated, high performance, in-memory query accelerator—HeatWave. It enables customers to run sophisticated analytics directly against their operational MySQL databases—eliminating the need for complex, time-consuming, and expensive data movement and integration with a separate analytics database. HeatWave accelerates MySQL performance by orders of magnitude for analytics and mixed workloads. Optimized for Oracle Cloud Infrastructure (OCI), MySQL Database Service is 100% built, managed, and supported by the OCI and MySQL engineering teams.

- HeatWave: In-memory query accelerator
- Fully managed database service
- Costs less than Amazon RDS
- Security and compliance

引自： https://www.oracle.com/mysql/

MySQL Database Service

Realm -> Region -> Availability Domain(s) -> Fault Domains



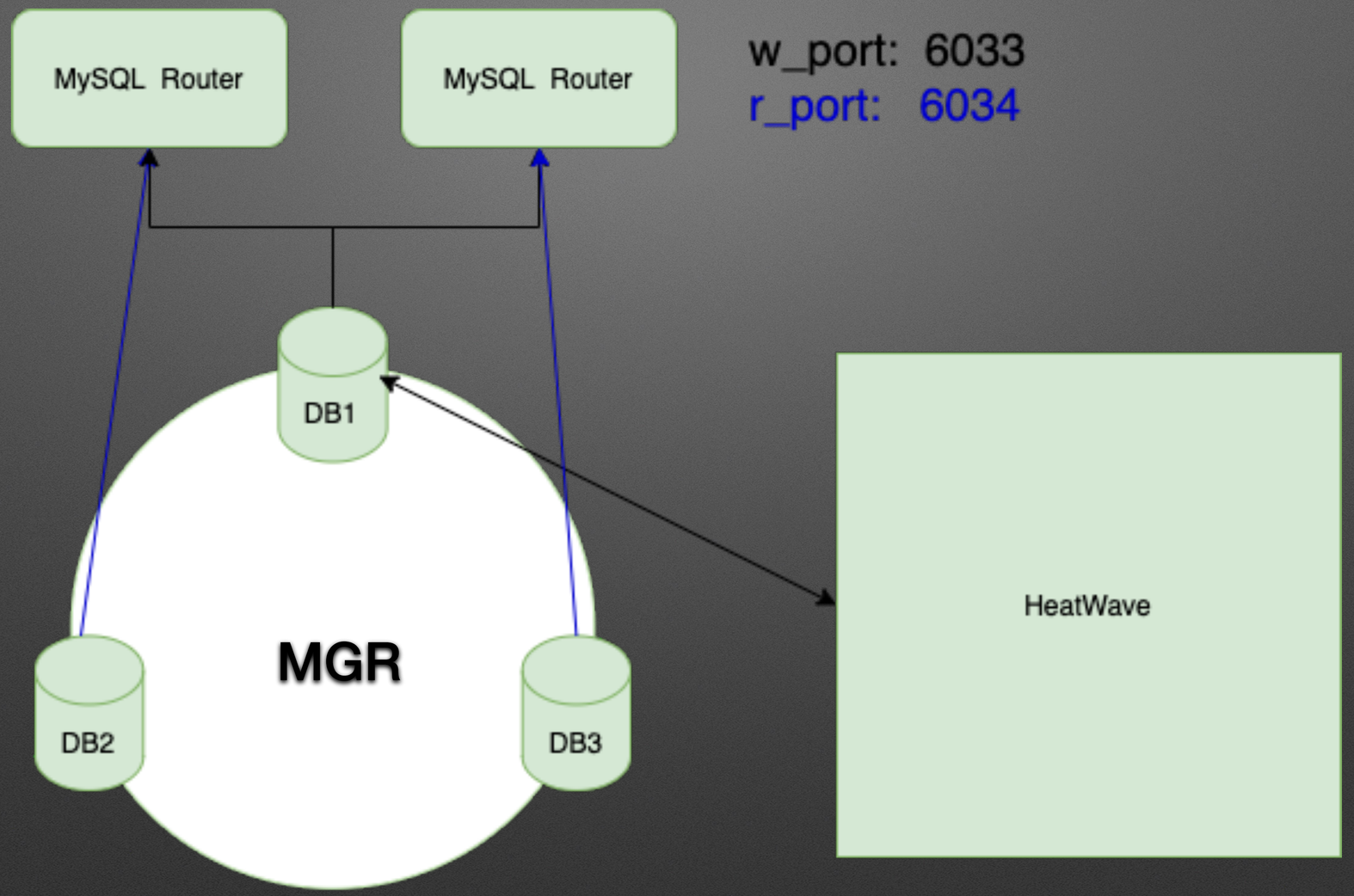
MySQL Database Service

集群节点管理

```
MySQL>SET GLOBAL clone_valid_donor_list =
'example.donor.host.com:3306';

 MySQL>CLONE INSTANCE FROM
'donor_clone_user'@'example.donor.host.com':3306 IDENTIFIED BY
'password';
```

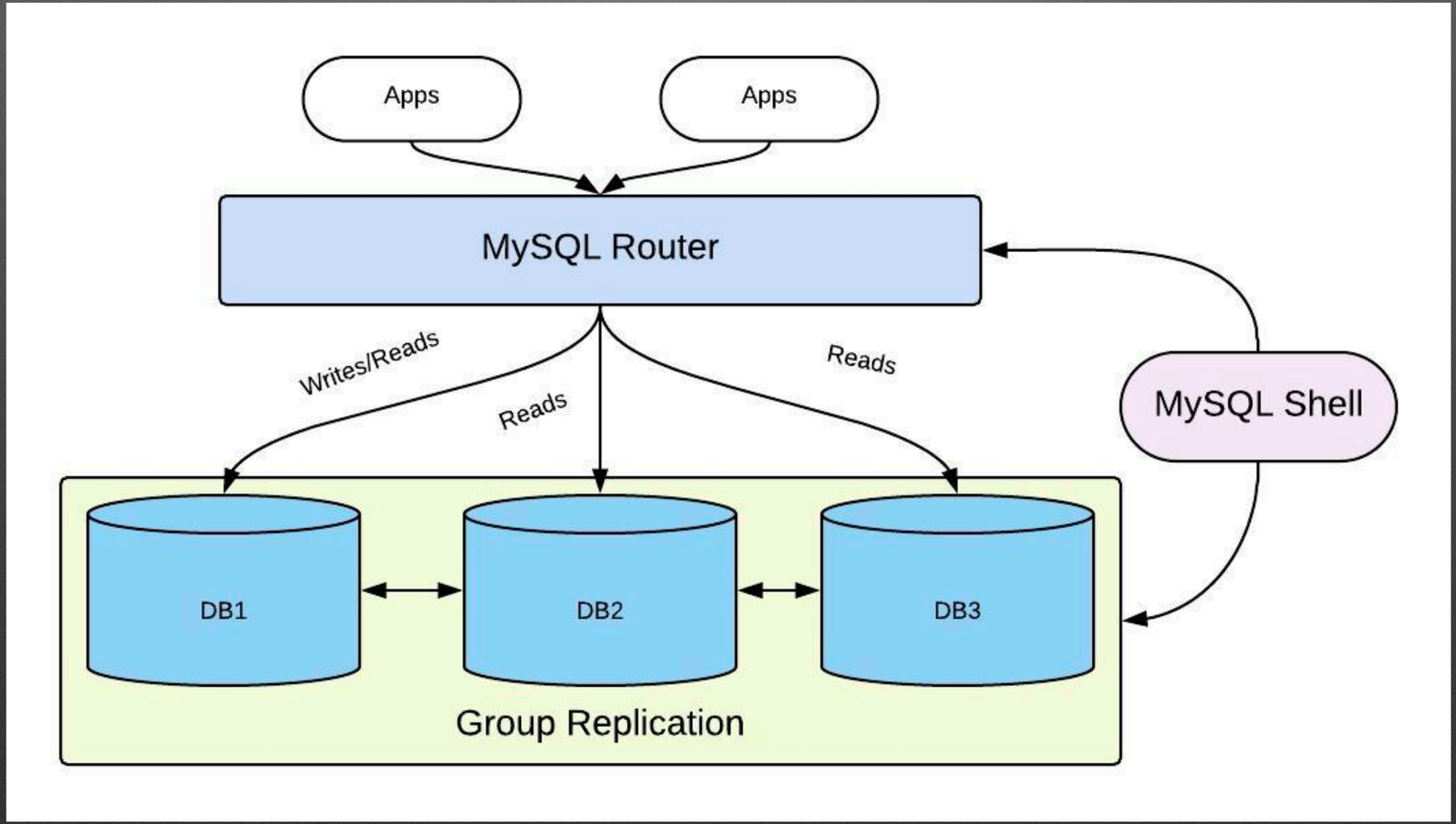
MySQL Database Service




```
MySQL>SET GLOBAL clone_valid_donor_list = 'example.donor.host.com:3306';
MySQL>CLONE INSTANCE FROM 'donor_clone_user'@'example.donor.host.com':3306 IDENTIFIED BY 'password';
```

2 MGR 如何实现多点写入

MGR 如何实现多点写入



MGR 如何实现多点写入

- 通过 KEY 约定行的唯一性， GTID当做版本号使用
  - group_replication_group_name = uuid
  - group_replication_gtid_assignment_block_size=1000000
  - binlog_transaction_dependency_tracking=WIRTESET
  - transaction_write_set_extraction=XXHASH64

MGR 如何实现多点写入

节点	GROUP_NAME	SERVER_NAME	GTID
DB1	uuid1	uuid2	uuid1:1-1000000)
DB2	uuid1	uuid3	uuid1:1000000-2000000)
DB3	uuid1	uuid4	uuid:2000000-3000000)

假设 DB1 上写了一个事务：
DB1： uuid1:1
DB2: uuid1:1
DB3: uuid1:1

假设 DB2 上写了一个事务：
DB1： uuid1:1,1000000
DB2: uuid1:1,1000000
DB3: uuid1:1,1000000

假设 DB3 上写了一个事务：
DB1： uuid1:1,1000000,2000000
DB2: uuid1:1,1000000,2000000
DB3: uuid1:1,1000000,2000000

实际运行中的GTID： uuid1:1-100,1000000-1000100,2000000-2000100

certifier.h — mysql-server

certifier.h

plugin ▸ group_replication ▸ include ▸ certifier.h ▸ Certification_info (typedef)

```
};

/**
  This class is a core component of the database state machine
  replication protocol. It implements conflict detection based
  on a certification procedure.

  Snapshot Isolation is based on assigning logical timestamp to optimistic
  transactions, i.e. the ones which successfully meet certification and
  are good to commit on all members in the group. This timestamp is a
  monotonically increasing counter, and is same across all members in the group.

  This timestamp, which in our algorithm is the snapshot version, is further
  used to update the certification info.
  The snapshot version maps the items in a transaction to the GTID_EXECUTED
  that this transaction saw when it was executed, that is, on which version
  the transaction was executed.

  If the incoming transaction snapshot version is a subset of a
  previous certified transaction for the same write set, the current
  transaction was executed on top of outdated data, so it will be
  negatively certified. Otherwise, this transaction is marked
  certified and goes into applier.
*/
typedef std::unordered_map<std::string, Gtid_set_ref *> Certification_info;
```

```
void add_pke(TABLE *table, THD *thd, uchar *record) {
  DBUG_ENTER("add_pke");
  DBUG_ASSERT(record == table->record[0] || record == table->record[1]);
  /*
    The next section extracts the primary key equivalent of the rows that are
    changing during the current transaction.

    1. The primary key field is always stored in the key_part[0] so we can
    simply read the value from the table->s->keys.

    2. Along with primary key we also need to extract the unique key values to
       look for the places where we are breaking the unique key constraints.

    These keys (primary/unique) are prefixed with their index names.

    In MySQL, the name of a PRIMARY KEY is PRIMARY. For other indexes, if
    you do not assign a name, the index is assigned the same name as the
    first indexed column, with an optional suffix (_2, _3, ...) to make it
    unique.

    example :
       CREATE TABLE db1.t1 (i INT NOT NULL PRIMARY KEY, j INT UNIQUE KEY, k INT
                            UNIQUE KEY);

       INSERT INTO db1.t1 VALUES(1, 2, 3);

       Here the write set string will have three values and the prepared value
    before hash function is used will be :

       i -> PRIMARYdb13t1211 => PRIMARY is the index name (for primary key)

       j -> jdb13t1221       => 'j' is the index name (for first unique key)
       k -> kdb13t1231       => 'k' is the index name (for second unique key)

    Finally these value are hashed using the murmur hash function to prevent
    sending more for certification algorithm.
  */
```

xxhash64(Primarydb13t1211):uuid1:1-100,1000000-1000100,2000000-2000100

MGR 如何实现多点写入

节点	LOCAL EXECUTE GTID	节点事务	
db1	uuid1:1-110,1000000-1000100,2000000-2000100	xxhash64(Primarydb13t1211):uuid1:1-110,1000000-1000100,2000000-2000100	通过
db2	uuid1:1-100,1000000-1000100,2000000-2000100	xxhash64(Primarydb13t1211):uuid1:1-100,1000000-1000100,2000000-2000100	不通过
db3	uuid1:1-101,1000000-1000100,2000000-2000100	xxhash64(Primarydb13t1211):uuid1:1-101,1000000-1000100,2000000-2000100	不通过

Note：

在节点延迟，但当前Certification_info 不存在该行的修改记录，该写入是可以提交的。例如上图 Certification_info 中存在： xxhash64(Primarydb13t1211)：uuid1:1-110,1000000-1000100,2000000-2000100 这样的值，其它俩个节点也可以对 Primarydb13t1211 这行记录进行修改

MGR 如何实现多点写入

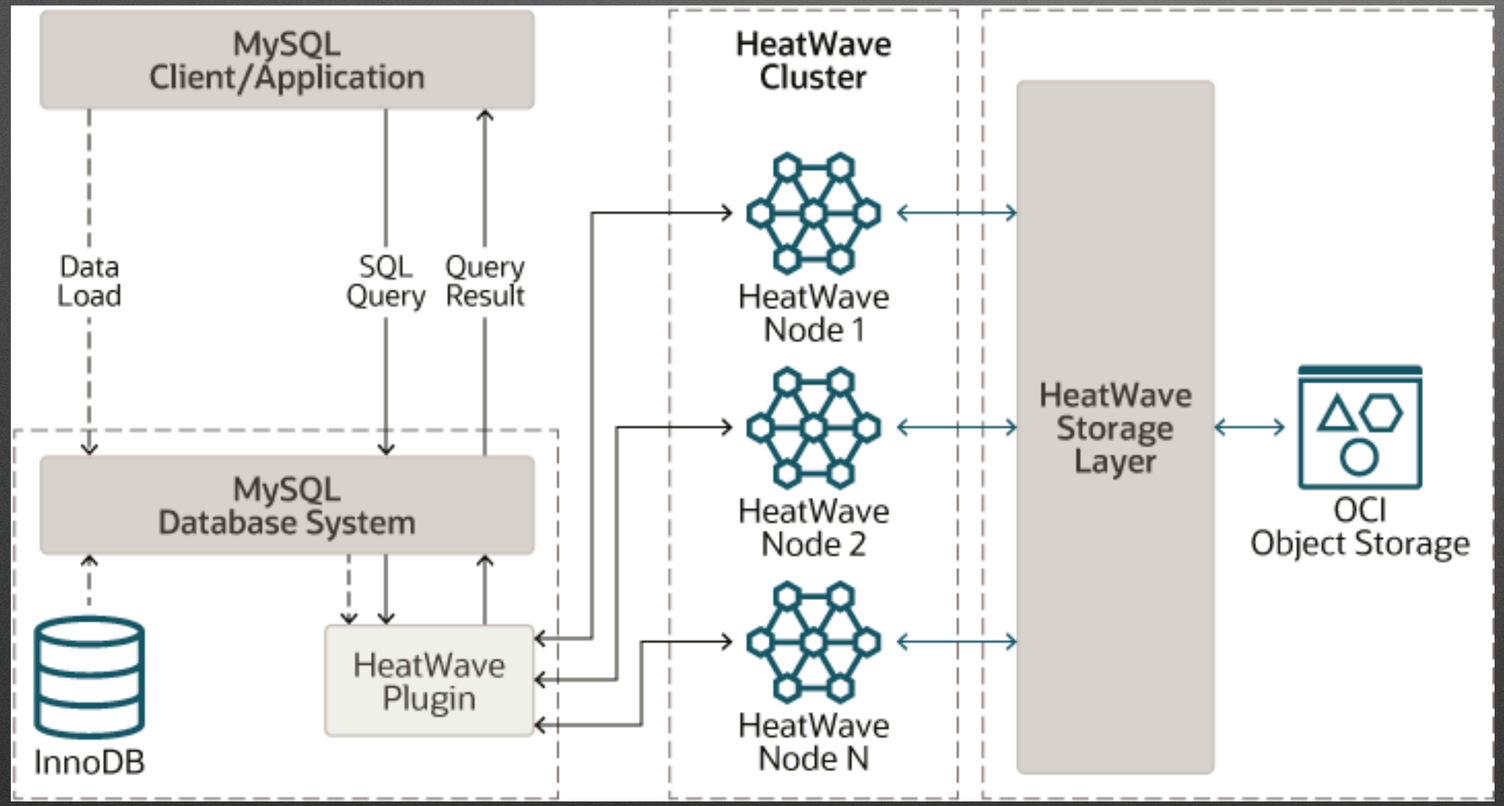
- 读写一致性控制

- group_replication_consistency

Property	Value
Command-Line Format	--group-replication-consistency=value
Introduced	8.0.14
System Variable	group_replication_consistency
Scope	Global, Session
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Enumeration
Default Value	EVENTUAL
Valid Values	EVENTUAL BEFORE_ON_PRIMARY_FAILOVER BEFORE AFTER BEFORE_AND_AFTER

3 MySQL HeatWave 架构

MySQL HeatWave 架构



1. In-Memory Hybrid-Columnar Format
2. Massively Parallel Architecture
3. Push-Based Vectorized Query Processing
4. Scale-Out Data Management
5. Native MySQL Integration

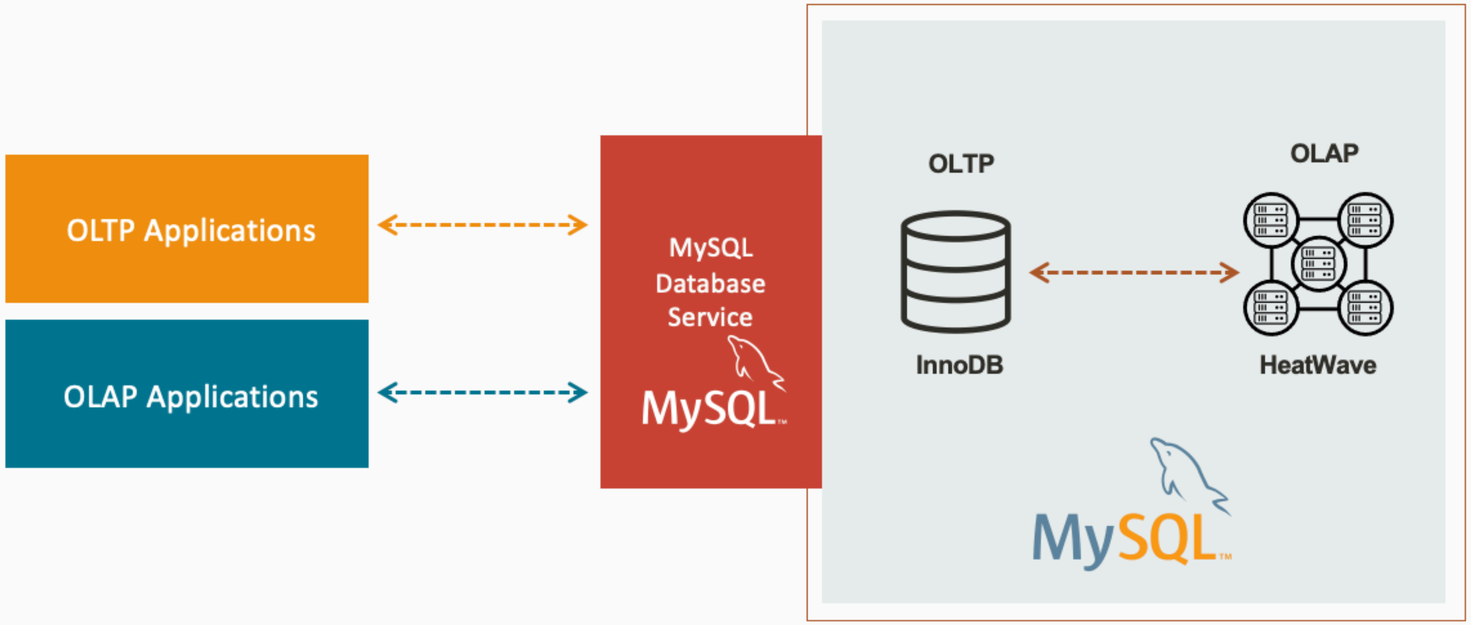
MySQL HeatWave 架构

1. In-Memory Hybrid-Columnar Format
2. Massively Parallel Architecture
3. Push-Based Vectorized Query Processing
4. Scale-Out Data Management
5. Native MySQL Integration

4 MySQL 和 Heatwave 的结合

MySQL Database Service with HeatWave

MySQL数据库无需ETL可以轻松运行高性能分析

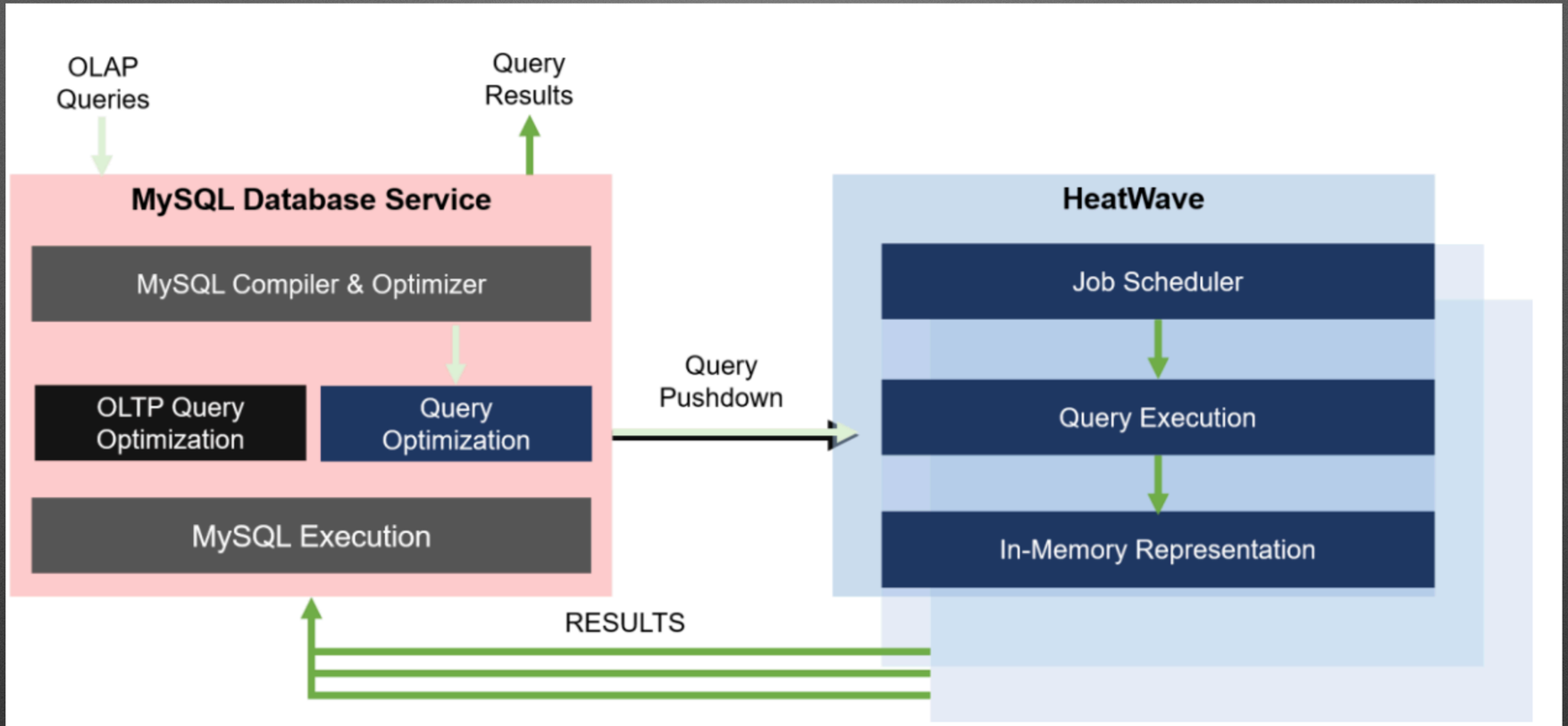


用于OLTP和分析应用程序的单一MySQL数据库

所有现有的应用程序无需任何更改即可工作

卓越的性能:比MySQL快400倍,可扩展到数千核

MySQL Database Service

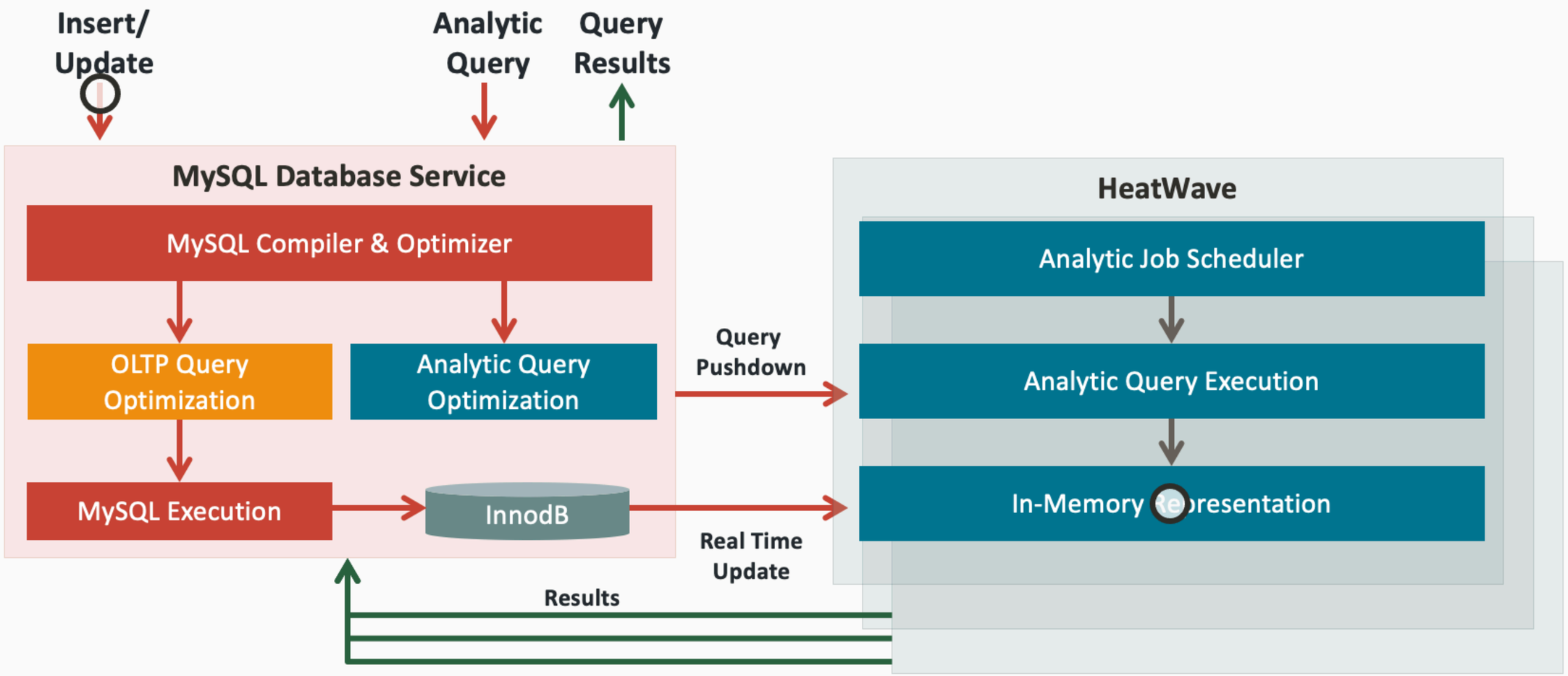


Source: Moor Insights & Strategy

https://www.oracle.com/a/ocom/docs/mysql/mysql-heatwave-engages-autopilot-report.pdf

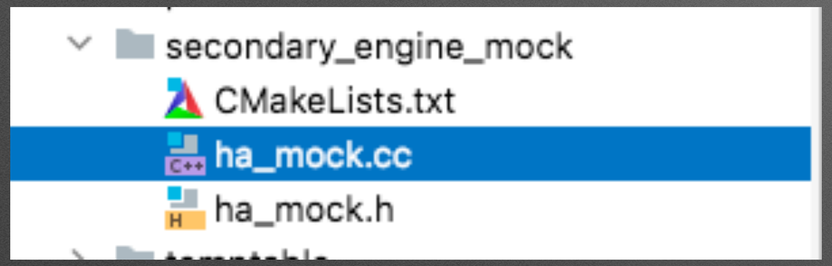
HeatWave: 实时分析

插入MySQL的数据可以在一秒内进行分析查询



MySQL 和 Heatwave 的结合

http://mysql.taobao.org/monthly/2020/11/04/

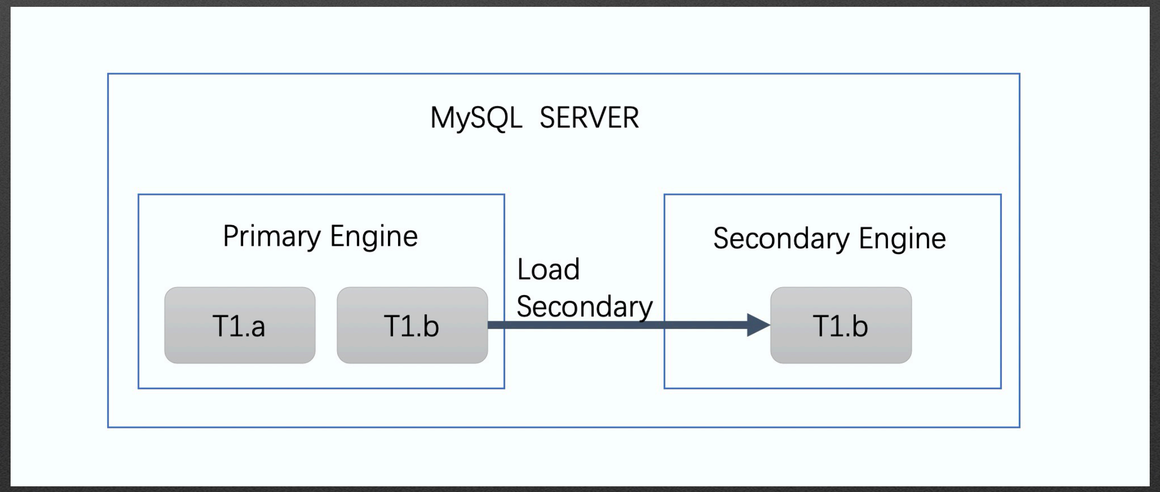


INSTALL PLUGIN mock SONAME "ha_mock.so";

CREATE TABLE t1 (a INT NOT SECONDARY, b INT)
SECONDARY_ENGINE MOCK;

ALTER TABLE T1 SECONDARY_LOAD;

ALTER TABLE T1 SECONDARY_UNLOAD;



QA环节