MySQL at NetEase

姜承尧
杭州网易研究院

MySQL at NetEase

- MySQL版本
  - MySQL 5.0.38 5.1.47 5.5.20
  - 逐步转向MySQL 5.5
- 数据库架构
  - master-slave replication
  - DDB（distribute database）
- 应用
  - 云音乐、云阅读、微博、博客、游戏
  - 几乎95%的应用都使用MySQL

NetEase MySQL Fork

- Why？
  - quick bug fix
  - high performance
  - high availability
  - **creativity & innovation**

NetEase MySQL

- NetEase MySQL
  - InnoSQL
  - Based on MySQL 5.5
  - open source
    - Patch
    - Binary package

NetEase MySQL

- 开源
  - https://github.com/NetEase/innosql
- 文档
  - http://mysql.netease.com/doc/
- 生产环境应用
  - 网易云音乐
  - 网易云阅读
  - 网易公开课
  - 网易博客
  - 几乎所有MySQL 5.5

Main Changes in InnoSQL

- InnoDB L2 Cache
- InnoDB buffer pool fast warm up
- Virtual sync replication
- Slave batch commit
- Role table
- Resource Governor

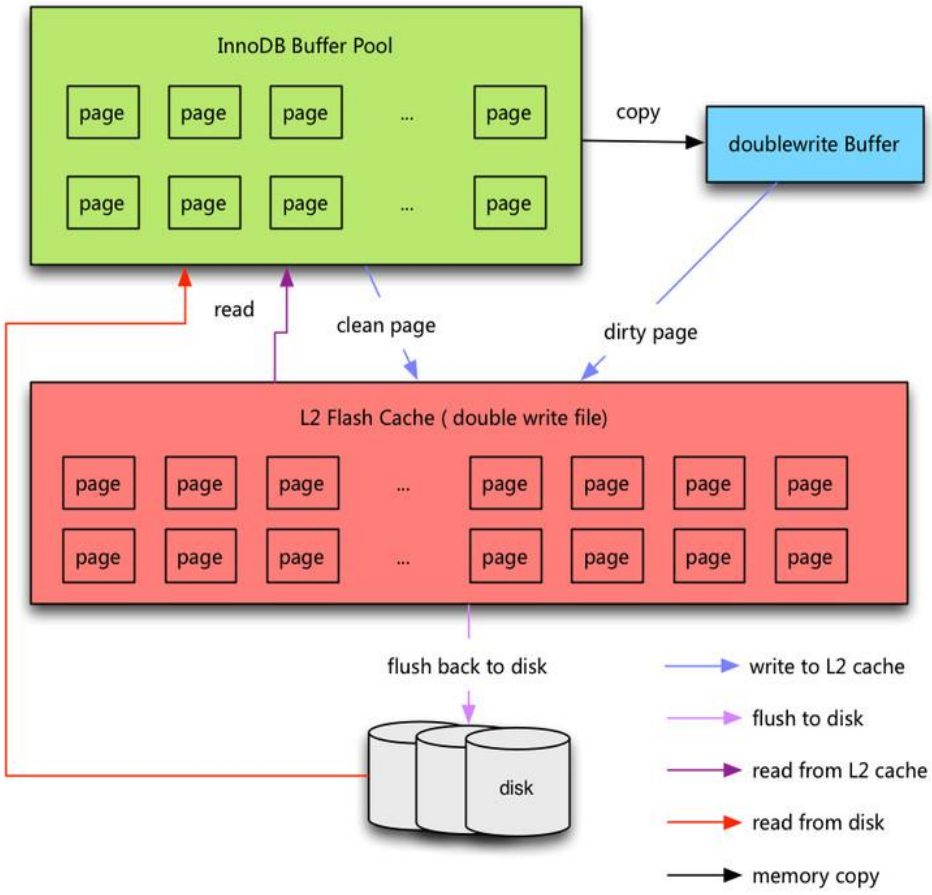
InnoDB L2 Cache

- SSD flash cache
- Support write back or write through
- Storage engine level
- Write IO on SSD is sequential
- Workload
  - Read-intensive
  - Write-intensive

InnoDB L2 Cache

Facebook flash cache alike	InnoDB L2 Cache
block-layer	storage engine layer
write-alloc	write when swap from buffer pool
random write	sequential write
a new device	using doublewrite as L2 cache
read-intensive workload	also write-intensive workload

InnoDB L2 Cache

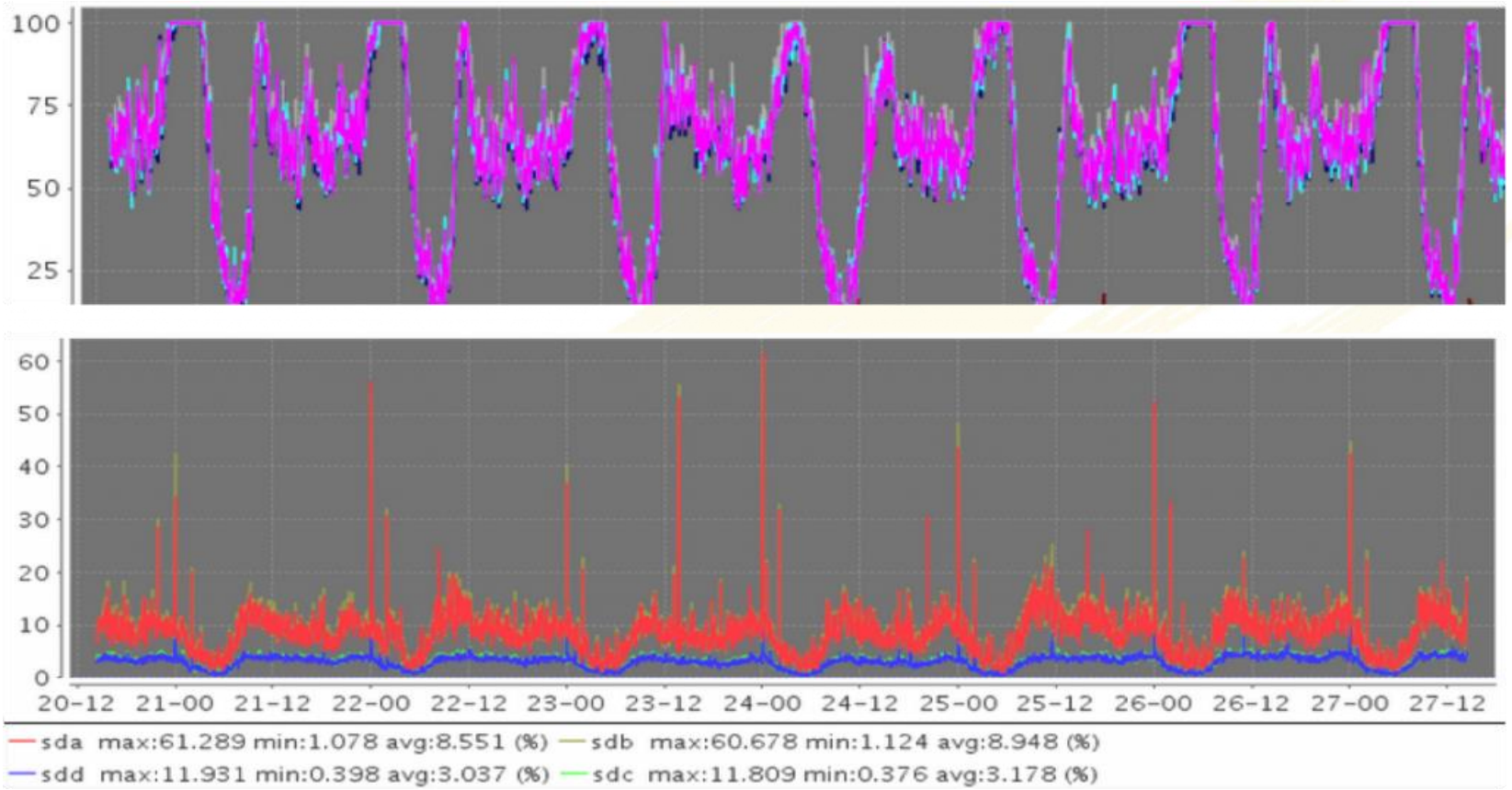


- 一个页在L2 cache中可能存在多个版本
- 回刷磁盘仅需最新版本的页
- L2 cache替代doublewrite
- 干净的页有选择性的写入cache
- cache中的页会move，确保其下次还能被命中

InnoDB L2 Cache

- 生产环境应用案例
  - 网易云阅读
  - read-intensive work load
  - 600G SAS => 120G SSD + 2T SATA

InnoDB L2 Cache



InnoDB L2 Cache

- TPC-C benchmark
  - http://www.mysqlperformanceblog.com/2012/10/25/l2-cache-for-mysql/

InnoDB BP fast warm up

- 预热
  - 快速恢复到应用状态
    - 数据库重启
    - 故障转移
- 方法
  - SELECT index
    - 加载太多无用数据
    - 浪费BP空间
- MySQL 5.6
  - dump & load BP LRU list

InnoDB BP fast warm up

- InnoSQL
  - Normal： share memory
    - fastest
  - Abnormal： dump & load BP LRU
    - dump also LRU old info
    - better warm up than MySQL 5.6
  - transfer to dump info to slave (next version)
    - for slave => master

Virtual Sync Replication

- Replication problem #1
  - slave not crash safe
    - too many 1062
    - statement-based binary log
      - update xxx set k=k+1 where ooo=???
  - lose data
    - even semi-replication

Virtual Sync Replication

- relay binlog not atomic
  - relay binlog (database)
  - update relay-log.info (file)
    - write to os cache
    - sync_relay_log_info=1
      - poor performance without BBU

Virtual Sync Replication

- MySQL 5.6
  - store relay-info in table (InnoDB)
  - atomic

```
BEGIN;
APPLY binlog;
UPDATE slave_relay_log_info
SET exec_master_log_pos=xxx, ...
COMMIT;
```

Virtual Sync Replication

- semi-replication
  - ❶ commit transaction
  - ❷ transfer binlog to slave
  - ❸ wait for slave ACK
- data inconsistency where crash at step 2 or 3

Virtual Sync Replication

- mainly for two node HA
- master commit after slave receive binary log
  - change to async mode when timeout
- no data lose when master crashed
  - using transaction-safe table
- ease of use and understanding
  - compare Galera Cluster
  - based on semi-replication

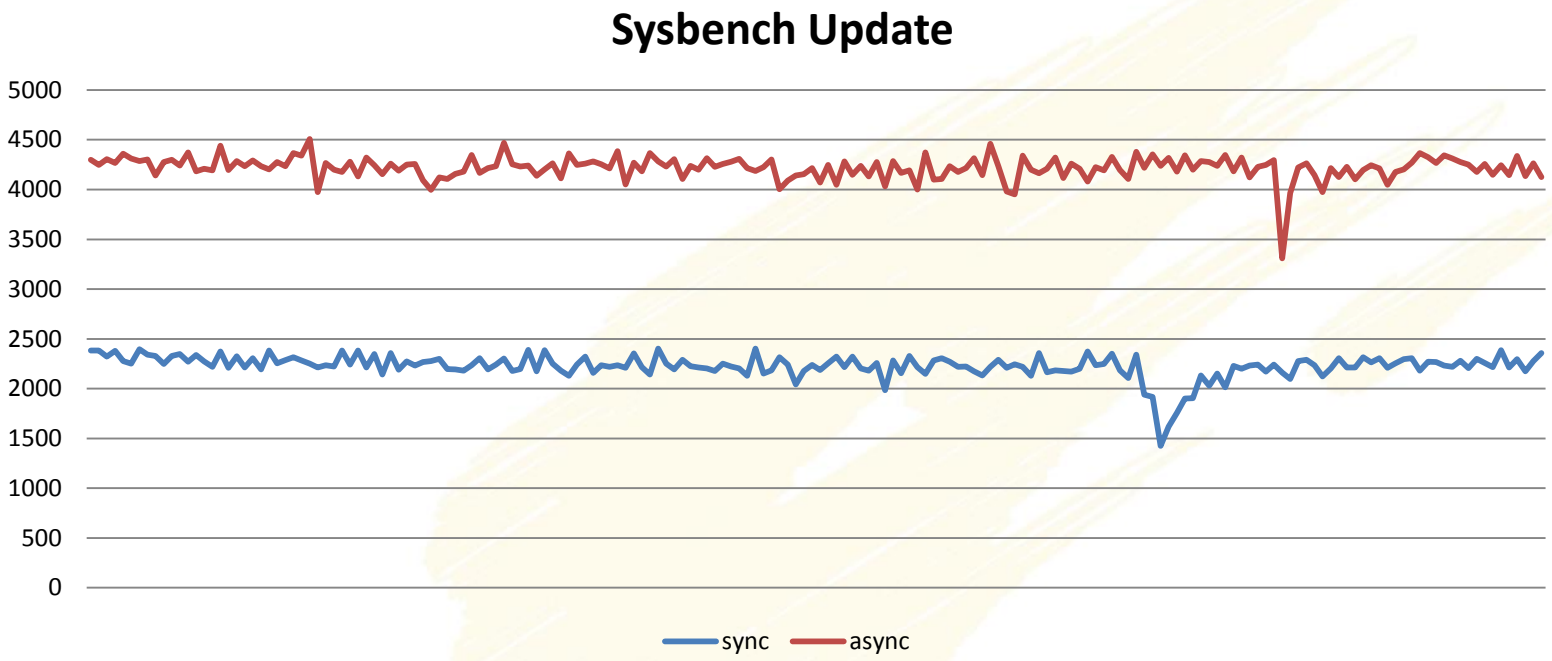
Virtual sync replication

- mainly for two node HA
- virtual sync replication
  - ❶ InnoDB generate prepare redo log
  - ❷ write binlog at master
  - ❸ transfer binlog to slave
  - ❹ wait slave ACK
  - ❺ InnoDB commit
- master commit after slave receive binary log
  - change to async mode when timeout
- no data lose when master crashed
  - using transaction-safe table

Virtual Sync Replication

- if master alive and change as slave
  - data can ben inconsistent when crash at step 3
- now should be handled by scripts
  - truncate redundant binlog at master that slave does not receive
  - also need handle partial binlog event at slave
- will be addressed this issue at internal MySQL (next version)

Virtual sync replication



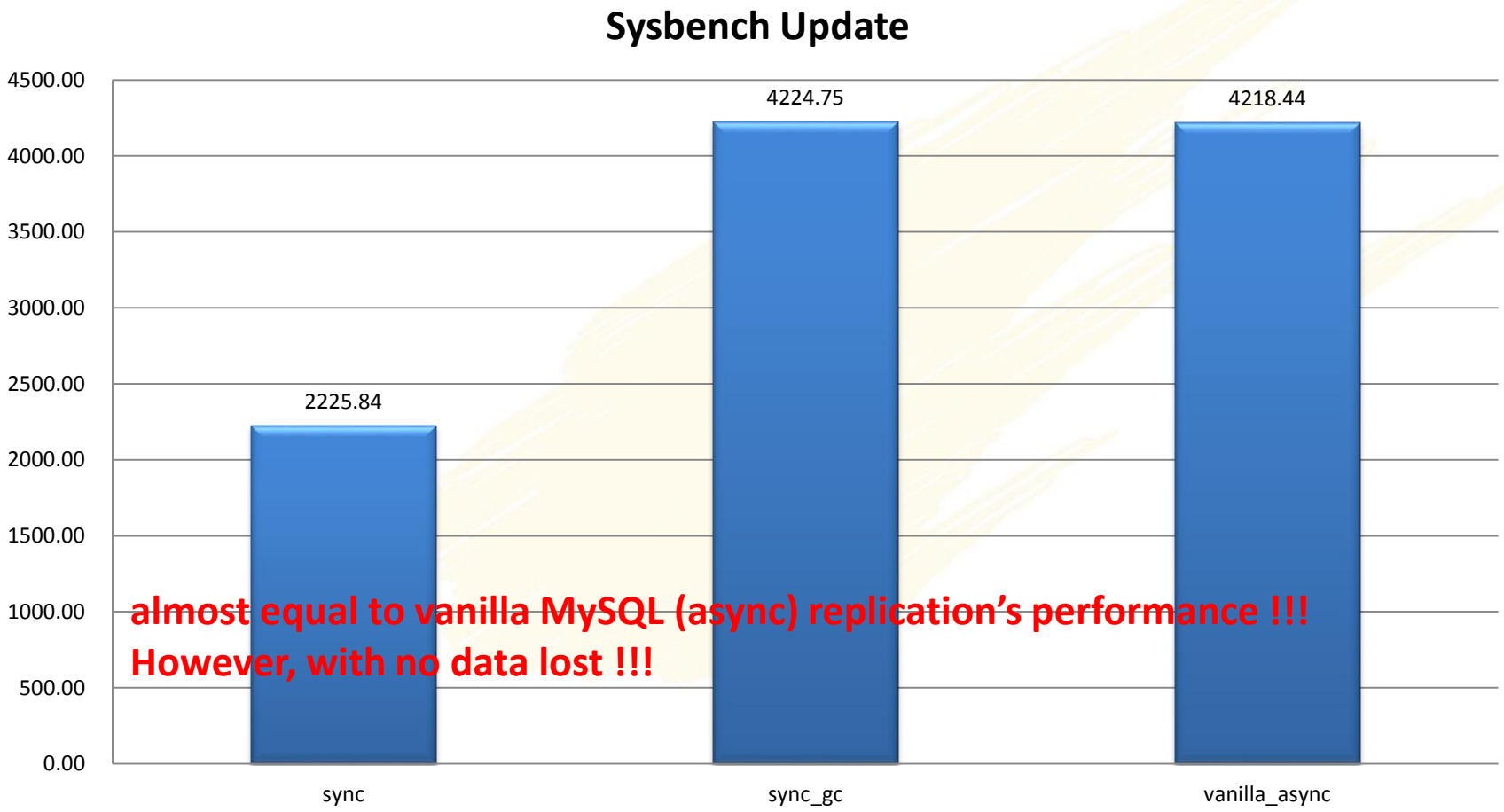
Virtual sync replication

- Original VSR performance decrease
  - 40% ~ 50% for sysbench full update benchmark
  - 15% ~ 20% for sysbench OLTP benchmark

Virtual sync replication

- VSR with group commit
- Merge MariaDB’s work
- Recover need recover group binlog
  - new binlog event

Virtual sync replication



slave batch commit

- Replication problem #2
  - single slave SQL thread
  - result in lag between master and slave
    - can be hours, even days

slave batch commit

- pre-fetch
- convert DML to SELECT
  - warm up by multi thread
- tools
  - mk-slave-prefetch
  - replication-booster-for-mysql
  - all for statement-based binlog
  - not workable for insert SQLs

slave batch commit

- parallel replication based on schema
  - MySQL 5.6
  - Tungsten Replicator
  - still single thread for one-schema database

slave batch commit

- parallel replication based on row_id
  - Taobao
  - MariaDB
  - Row-based binlog
    - Blob too big
  - Every table must have an explicit PK

slave batch commit

- InnoSQL's batch commit
  - apply binlog with one transaction
- advantage
  - reduce redo log fsync a lot
  - work for all kinds of binlog format
  - can work for insert SQLs
  - a more general solution
- disadvantage
  - still single thread

slave batch commit

BEGIN;

APPLY binlog;

UPDATE slave_relay_log_info

SET exec_master_log_pos=xxx, ...;

APPLY binlog;

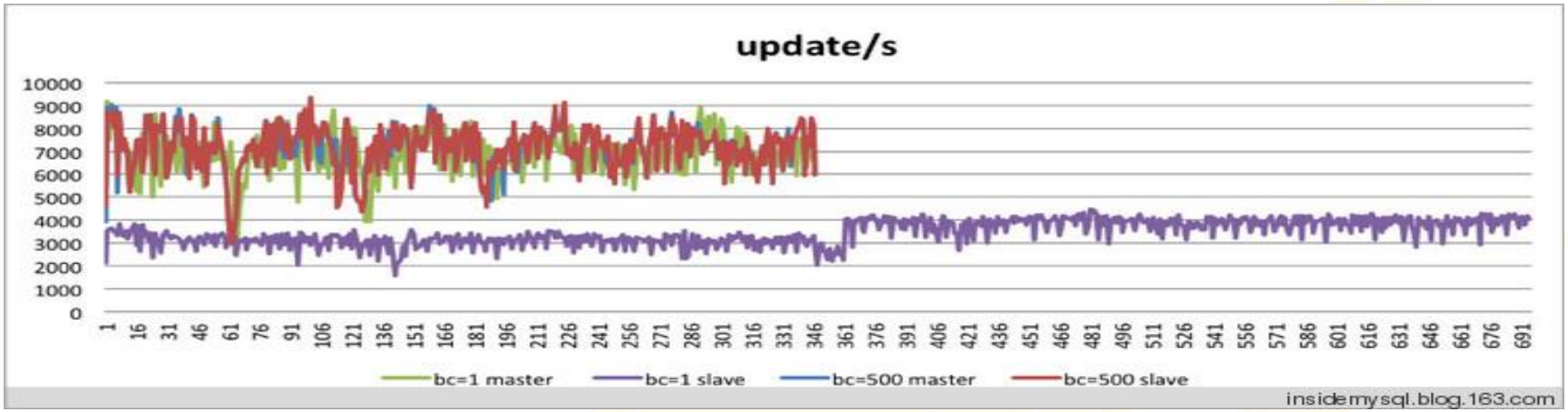
UPDATE slave_relay_log_info

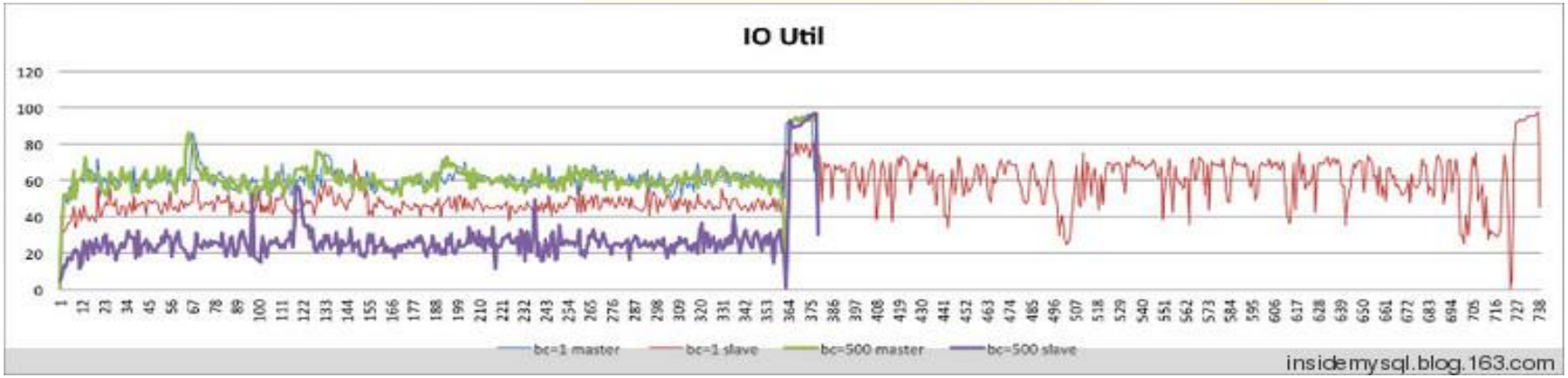
SET exec_master_log_pos=xxx, ...;

……

COMMIT;

slave batch commit





Resource Governor

- Control resource:
  - IO
  - CPU
  - row count
- mainly for cloud service
  - multi tenancy

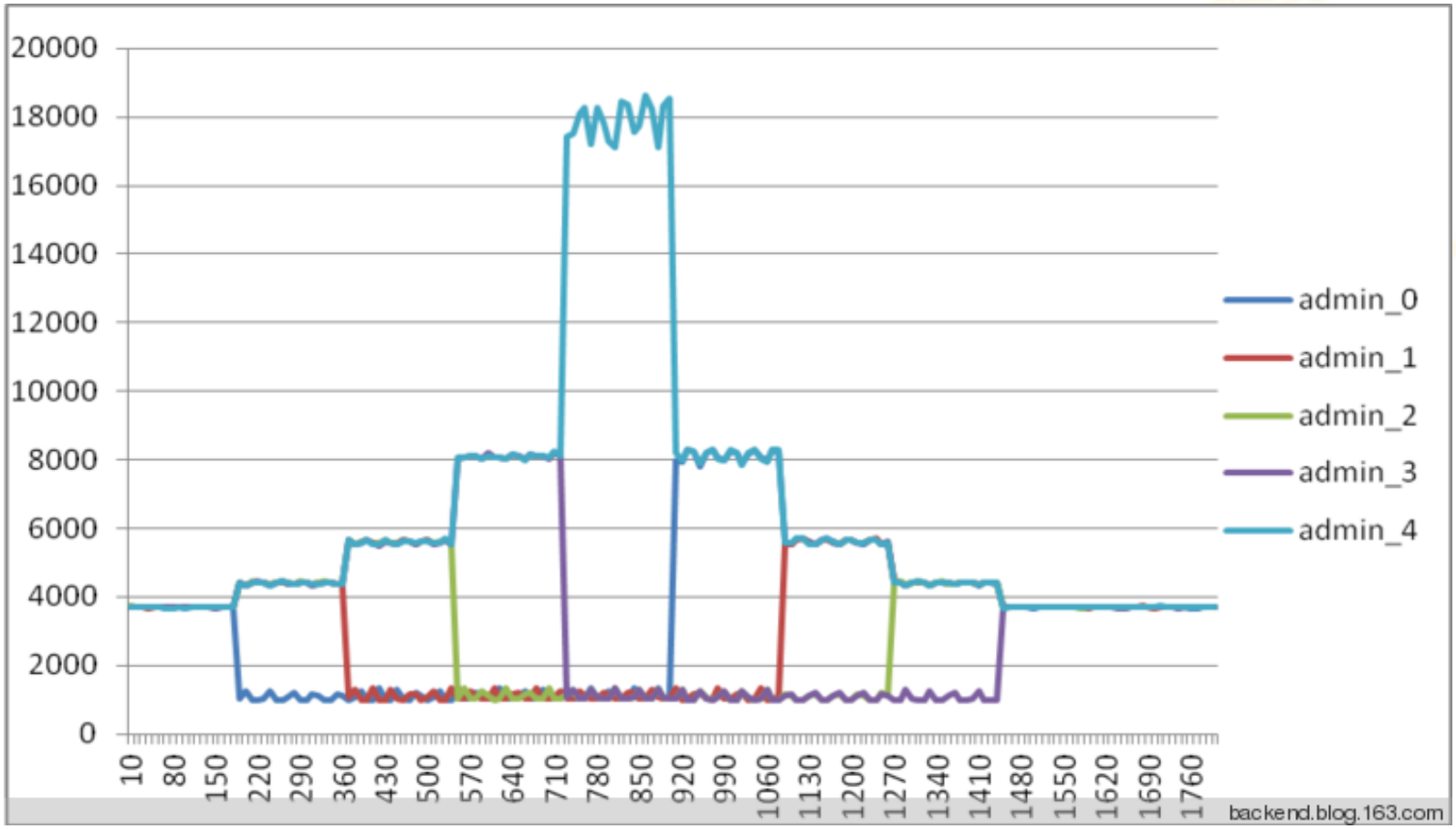
Resource Governor

- Proxy vs RG
  - Performance
    - Overhead for Proxy
  - SQL
    - RG support all kinds of SQL
  - Resource Control
    - RG more dynamic
      - CPU、IO、row_count

Resource Governor

- Oracle Profiler
  - Rollback
  - Only for InnoDB
- Microsoft SQL Server Resource Governor
  - InnoSQL use this design
  - Storage engine handler
  - Support all kinds of engine

Resource Governor



Q & A