MySQL優化

– 從硬體、系統到參數的MySQL一站式優化

葉金榮

http://imysql.com，公眾號：MySQL中文網

2015.12.19 臺北

- 葉金榮
- CMUG副主席
- Oracle MySQL ACE
- 大陸最早的MySQL推廣者之一
- 2006年創辦首個MySQL專業技術中文網站 http://imysql.com
- 10餘年MySQL經驗，擅長MySQL性能優化、架構設計、故障排查

Agenda

- MySQL的特點
- 硬體、系統優化
- MySQL配置優化
- SCHEMA設計優化
- SQL優化
- 其他優化

MySQL的特點

- 不要當做ORACLE、SQL Server或PostgreSQL來用
- 不適宜的場景
  - 存儲大文本、圖片、附件等大物件
  - 複雜查詢，複雜運算，或全文檢索
- 好的應用場景
  - 快速提交的小事務
  - 不太複雜的SQL請求
  - 合適的表物件設計規則，單表壓力不要太大

MySQL的特點

- CPU資源利用特點
  - 5.1之前，多核支持差
  - 5.1，最高可用4個核
  - 5.5，最高可用24個核
  - 5.6，最高可用64個核
  - 每個query對應一個執行緒，只能用到一個邏輯CPU
  - 每個連接對應一個執行緒，只能用到一個邏輯CPU
- 建議
  - CPU主頻越高越好，核數越多越好
  - 多用新版本，少用舊版本
  - 每次請求儘快結束，少用複雜SQL，事務及時提交/回滾
  - 空閒連接主動快速斷開

MySQL的特點

- 記憶體資源利用特點
  - 類似ORACLE SGA：innodb buffer pool、key buffer、query cache等
  - 類似ORACLE PGA：sort buffer、join buffer、tmp table等
  - MyISAM只能緩存索引
  - InnoDB緩衝資料、索引
  - 無索引排序會生成記憶體臨時表
- 建議
  - 使用更多記憶體可減少物理IO，提高tps
  - 關閉沒什麼卵用的query cache
  - 所有實例分配的記憶體不超過實體記憶體的50%~70
  - 檢查避免沒索引的情況，減少臨時表記憶體消耗
  - KV等類型資料利用redis/memcached存取

MySQL的特點

- 磁片資源利用特點
  - undo log的I/O特徵：順序寫，隨機讀；
  - redo log、binlog的I/O特徵：順序寫，順序讀；
  - 數據文件的I/O特徵：隨機寫，隨機讀；
  - MyISAM是堆組織表（HOT），InnoDB是索引組織表（IOT）
  - InnoDB相比MyISAM更消耗磁片空間
- 建議
  - 加大記憶體
  - 使用更高速的I/O設備
  - data file放在高速I/O設備上，binlog、redo log放在機械盤上
  - 採用xfs檔案系統
  - kernel的io scheduler採用noop/deadline

性能瓶頸分析工具

- MySQL層面
  - slow log
  - show [global] status
  - profiling
  - show processlist
  - show engine innodb status
  - pt-ioprofile

參考：[MySQL FAQ]系列 — processlist中哪些狀態要引起關注

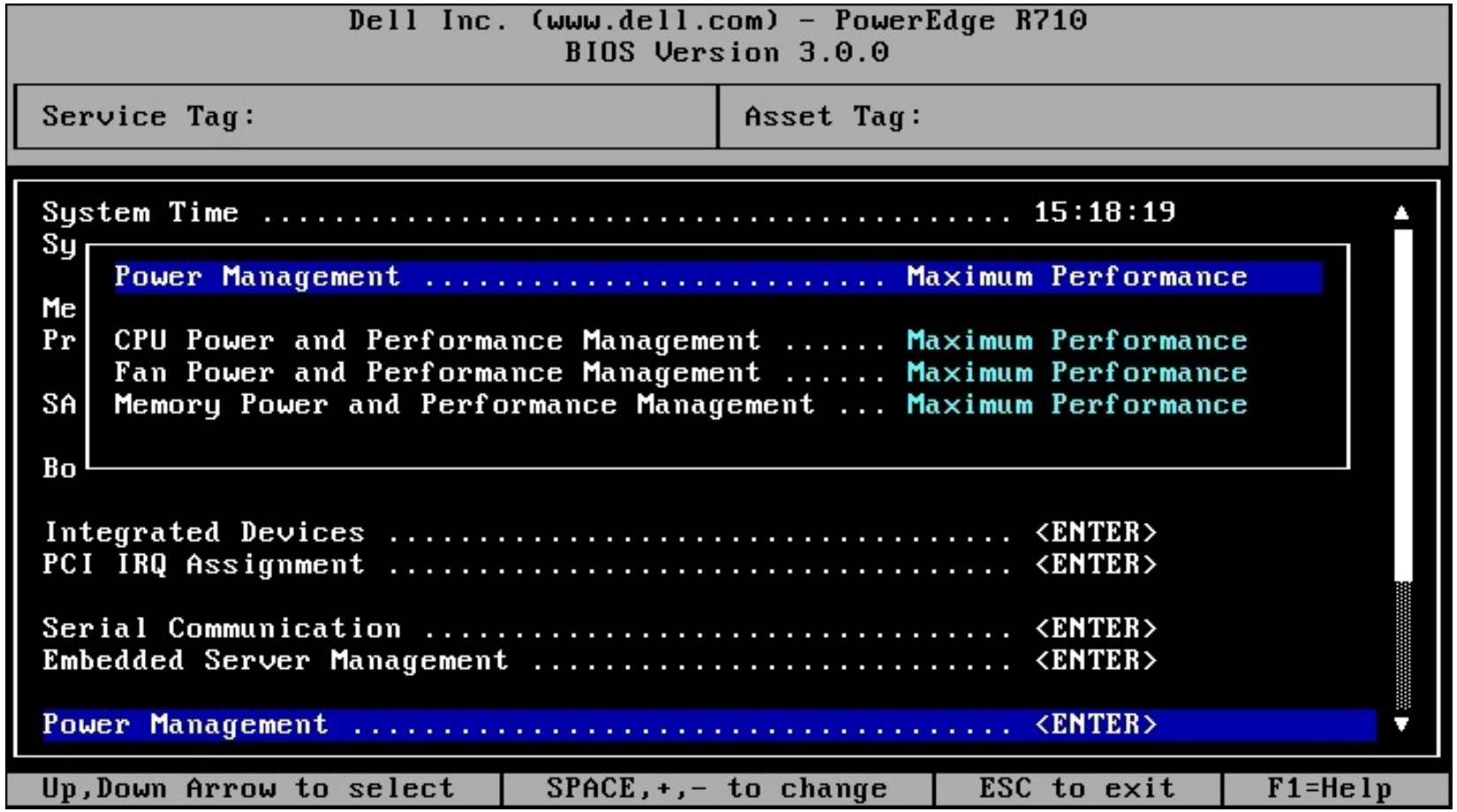
硬體優化

- BIOS配置優化
  - CPU設置Maximum Performance
  - 關閉C1E、C states
  - 關閉NUMA
- I/O優化優化
  - 使用PCIe-SSD等高速I/O設備
  - RAID-10
  - CACHE & BBU
  - WB & FORCE WB

I/O子系統一般是最大瓶頸所在，提高IOPS能力的幾種方法：

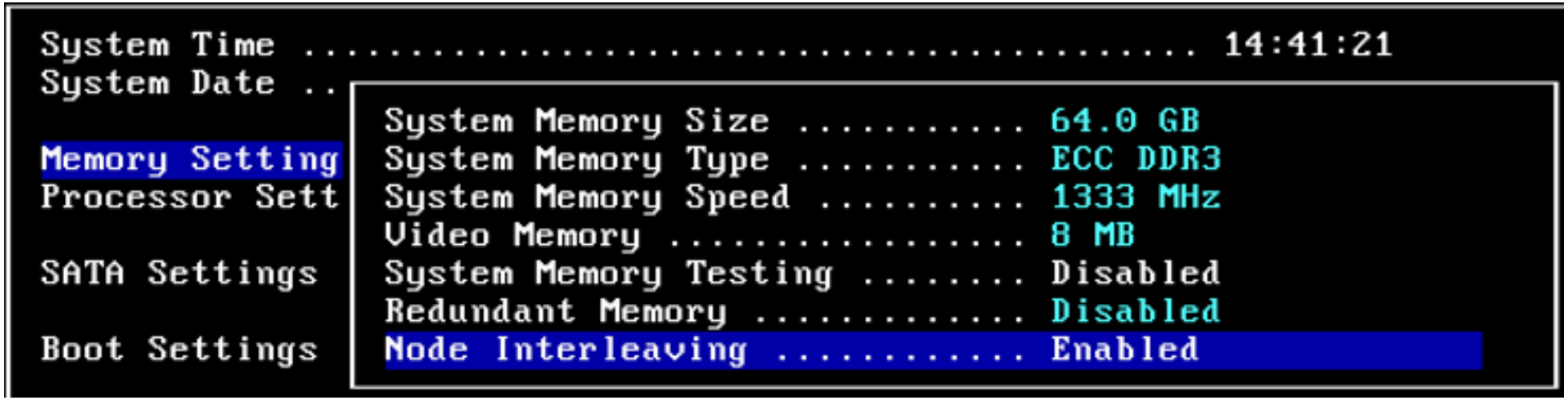
1. 換PCIe SSD（提高I/O效率，普通SAS盤2000以內的iops，而新設備可達到數萬或者數十萬iops）
2. 減少物理I/O（合併多次讀寫為一次，或者前端加記憶體CACHE；或者優化業務，消除I/O）
3. 加大記憶體，減少物理I/O;
4. 採用xfs檔案系統（相比ext3、ext4提高IOPS能力，高io負載下表現更佳）
5. 調整raid級別為raid 10（相比raid1、raid5等提高IOPS能力）
6. 調整寫cache策略為wb或force wb（利用陣列卡cache，提高iops）
7. io scheduler（優先使用deadline，如果是SSD，則使用noop）

硬體優化


Dell Inc. (www.dell.com) - PowerEdge R710
BIOS Version 3.0.0
Service Tag:
Asset Tag:
System Time 15:18:19
Power Management ..... Maximum Performance
CPU Power and Performance Management ..... Maximum Performance
Fan Power and Performance Management ..... Maximum Performance
Memory Power and Performance Management ... Maximum Performance
Integrated Devices ..... <ENTER>
PCI IRQ Assignment ..... <ENTER>
Serial Communication ..... <ENTER>
Embedded Server Management ..... <ENTER>
Power Management ..... <ENTER>
Up,Down Arrow to select
SPACE,+,- to change
ESC to exit
F1=Help

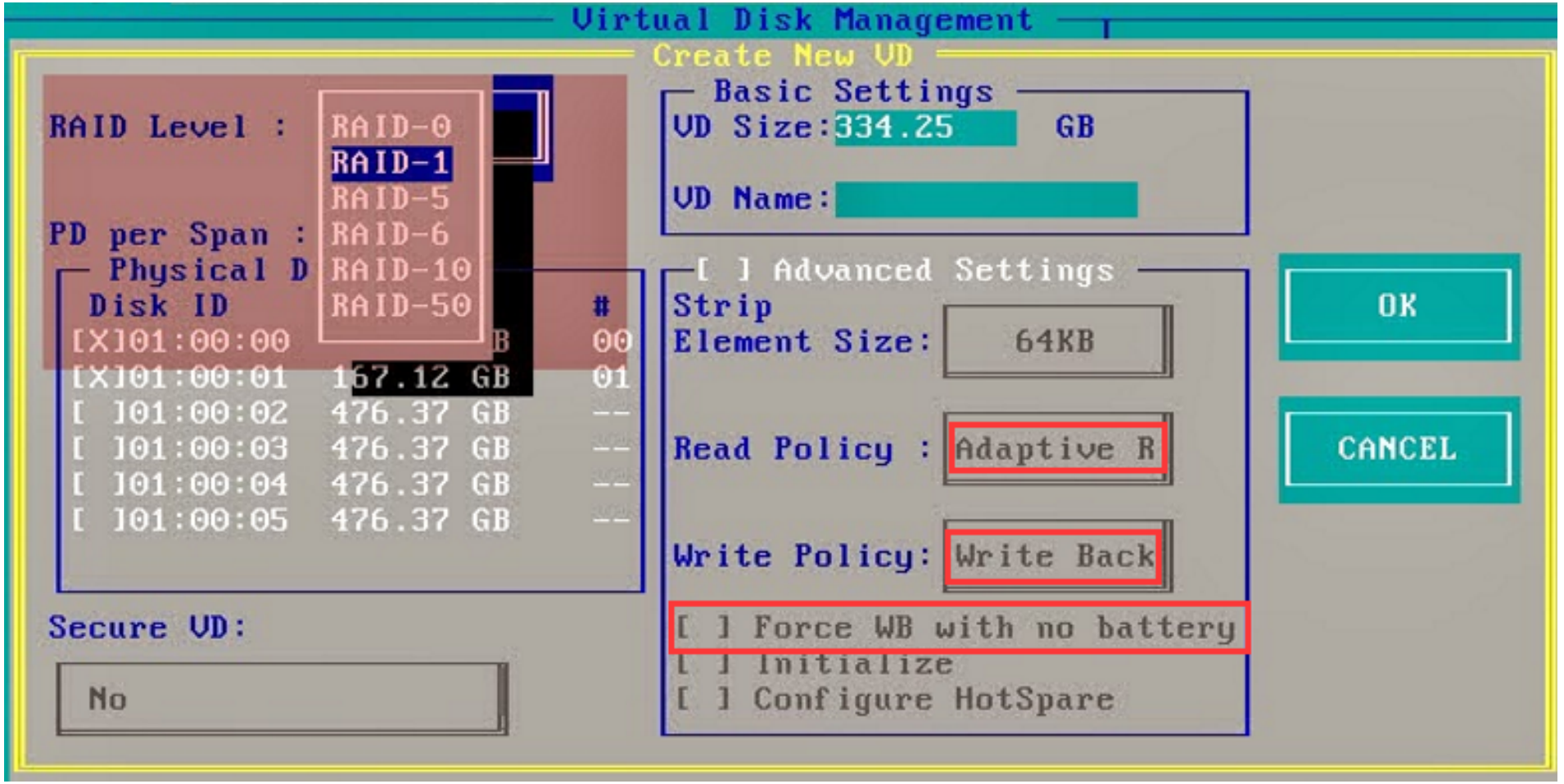
硬體優化

- 關閉numa



注：5.6.27版本開始，新增innodb_numa_interleave選項可調整numa策略

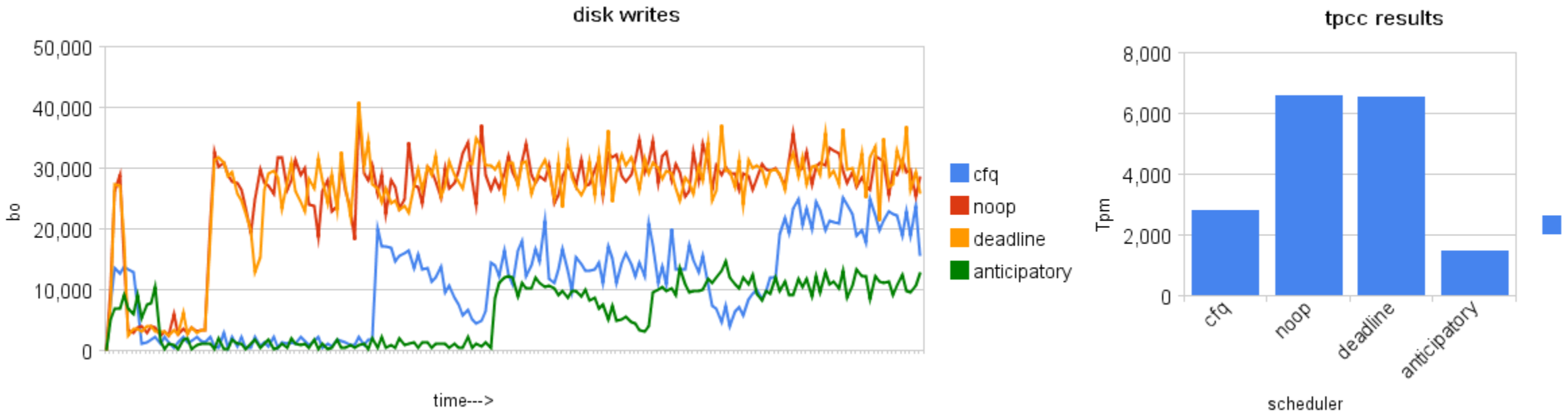
硬體優化


Virtual Disk Management
Create New VD
RAID Level :
RAID-0
RAID-1
RAID-5
RAID-6
RAID-10
RAID-50
PD per Span :
Physical D
Disk ID
[X]01:00:00
[X]01:00:01 167.12 GB 01
[ ]01:00:02 476.37 GB --
[ ]01:00:03 476.37 GB --
[ ]01:00:04 476.37 GB --
[ ]01:00:05 476.37 GB --
Secure VD:
No
Basic Settings
VD Size:334.25 GB
VD Name:
[ ] Advanced Settings
Strip
Element Size: 64KB
Read Policy : Adaptive R
Write Policy: Write Back
[ ] Force WB with no battery
[ ] Initialize
[ ] Configure HotSpare
OK
CANCEL

系統優化

- io scheduler
  - deadline/noop

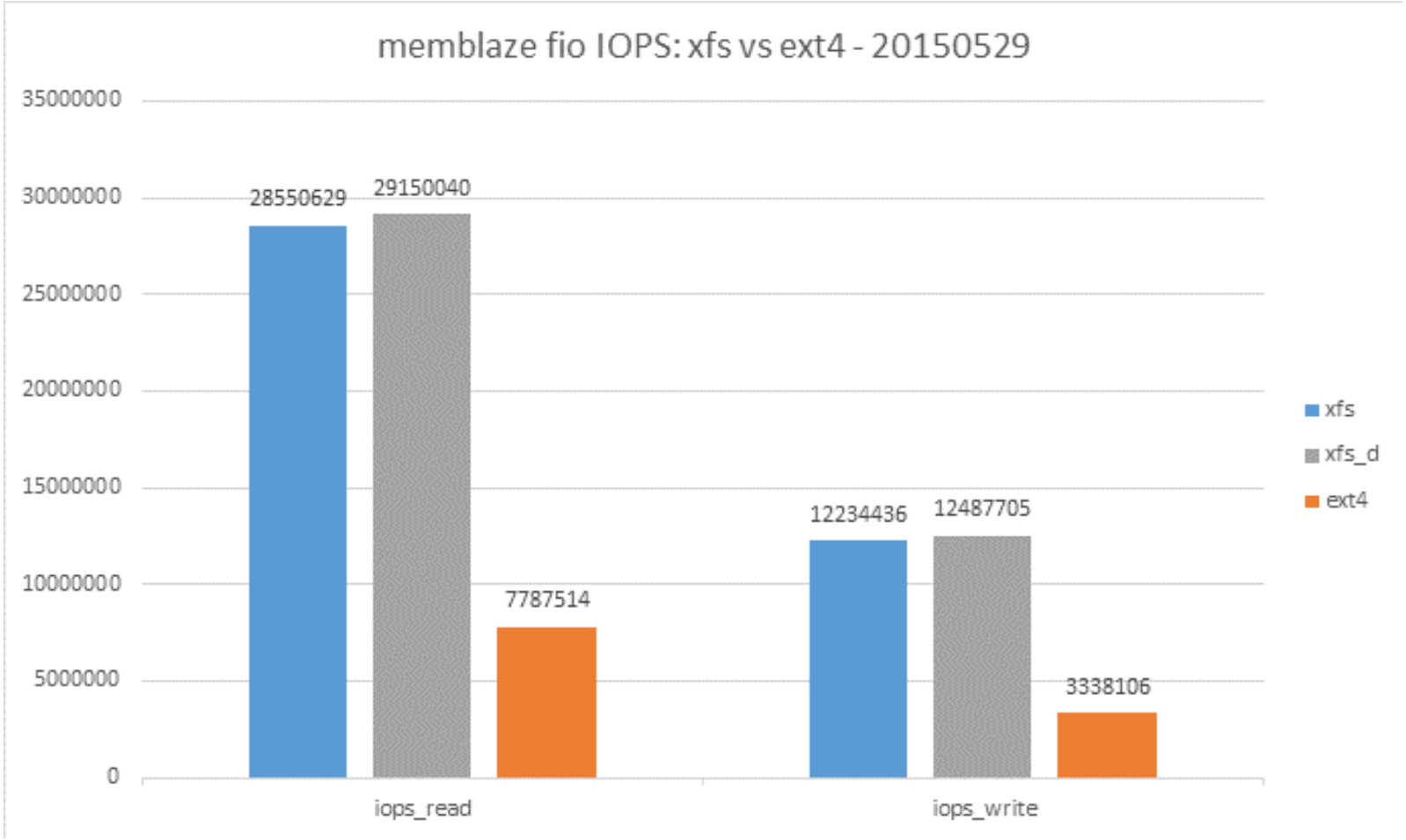
系統優化


disk writes
50,000
40,000
30,000
20,000
10,000
0
bo
time--->
cfq
noop
deadline
anticipatory
tpcc results
8,000
6,000
4,000
2,000
0
Tpm
cfq
noop
deadline
anticipatory
scheduler

系統優化

- filesystems
  - xfs
  - ext4

系統優化


memblaze fio IOPS: xfs vs ext4 - 20150529
35000000
30000000
25000000
20000000
15000000
10000000
5000000
0
28550629
29150040
7787514
12234436
12487705
3338106
iops_read
iops_write
xfs
xfs_d
ext4

系統優化

- kernel
  - vm.swappiness = 0~10，降低使用swap的幾率
  - 調低參數值避免I/O負載高時候導致系統掛起

    vm.dirty_background_ratio = 0~10

    vm.dirty_ratio = 0~20

MySQL選項優化 – 記憶體相關

- 記憶體相關參數

mysql使用总内存 = global_buffers + thread_buffers

global buffer(全局内存分配总和) =
 innodb_buffer_pool_size
+innodb_additional_mem_pool_size
+innodb_log_buffer_size
+key_buffer_size
+query_cache_size
+table_open_cahce
+table_definition_cache
+thread_cache_size

All thread buffer(会话/线程级内存分配总和) =
max_threads * (
 read_buffer_size
+read_rnd_buffer_size
+sort_buffer_size
+join_buffer_size
+binlog_cache_size
+tmp_table_size
+thread_stack
+net_buffer_length
+ bulk_insert_buffer_size)

MySQL選項優化 – InnoDB相關

- innodb_buffer_pool_size，最大不超過實體記憶體的50% ~ 70%
- innodb_data_file_path，ibdata1初始化時，至少1GB以上
- innodb_log_buffer_size，一般8-32MB足夠了
- innodb_log_file_size，5.5及以上可設置1~2GB，5.5以下建議256~512MB
- innodb_flush_log_at_trx_commit，0=>最快資料最不安全，1=>最慢最安全，2=>折中
- innodb_max_dirty_pages_pct，25%~50%為宜
- innodb_io_capacity，普通機械盤=>1000左右，SSD=>10000左右，PCIe SSD=>20000以上

MySQL選項優化 – 其他

- sync_binlog，0=>最快資料最不安全，系統自己決定刷新binlog的頻率；1=>最慢最安全，每個event刷新一次；N=>每N個事務刷一次binlog
- long_query_time，建議設置小於0.1秒
- open_files_limit & innodb_open_files，建議65535
- max_connections，平均突發最大連接數的80%為宜，設置過大容易導致全部卡死
- thread_handling，頻繁突發連接時啟用（企業版\Percona\MariaDB）
- query_cache_size & query_cache_type，關閉

SCHEMA設計優化

- 默認採用InnoDB，不使用~~MyISAM~~
- INT AUTO_INC Primary Key for InnoDB
- INT unsigned for IPV4，不使用~~CHAR(15)~~
- 儘量不使用~~TEXT、BLOB~~

SCHEMA設計優化

- SQL查詢基於主鍵/唯一索時性能引優於普通索引
- 複合索引比普通索引更合適，有很多時候可以避免回表
- 基數(cardinality)小的列不建獨立索引
- 長字串欄位只創建部分長度索引
  - ALTER TABLE t ADD INDEX(username(20))
- 用於檢索、排序的欄位，需要創建索引
- 多表JOIN的列類型一致，並創建合適的索引

MySQL索引限制

- 通過索引掃描的記錄數超過30%（大概值，非固定值），會變成全表掃描
- 第一個查詢條件列不是聯合索引的最左索引列無法使用該索引
- 記憶體表(HEAP 表)使用HASH索引時，使用範圍檢索或者ORDER BY
- 表關聯欄位類型不一樣（也包括長度不一樣），會發生隱式轉換
- 不支援聯合索引中多列不同排序規則
- 不支援函數索引(where date(xx) = ‘2015-12-19’)
- 不支援點陣圖索引(bitmap index)
- 不支持全模糊查詢(like ’%xxx%’)
- 注意類型隱式轉換，例如：char_col = *int_val*

SQL優化

- type

由左及右，最差到最佳

ALL	index	range	index_subquery	unique_subquery	index_merge	ref_or_null	ref	eq_ref	const	system
ALL	index	range	index_subquery	unique_subquery	index_merge	ref_or_null	ref	eq_ref	const	system

參考：[MySQL FAQ]系列 — EXPLAIN結果中哪些資訊要引起關注

SQL優化

- Extra

```
EXPLAIN SELECT * FROM trx_fee where fee > 10\G
*************************** 1. row ***********
           id: 1
  select_type: SIMPLE
        table: trx_fee
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 11
        Extra: Using where
```

通常是進行了全表/全索引掃描後再用WHERE子句完成結果過濾

SQL優化

- Extra

```
EXPLAIN SELECT * FROM trx_fee WHERE fee > 5 ORDER BY fee\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: trx_fee
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 11
        Extra: Using where; Using filesort
```

没有合適的索引用於排序，導致產生filesort

SQL優化

- Extra

```
EXPLAIN SELECT * FROM trx_fee a, trx_fee b where a.id = b.fee order by a.fee\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: b
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 11
        Extra: Using temporary; Using filesort
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: a
         type: eq_ref
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: test.b.fee
         rows: 1
        Extra: NULL
```

常見殺手級SQL

- SELECT * vs SELECT col1, col2
- ORDER BY RAND()
- WHERE func(key_col) = ?
- WHERE key_part2 =? AND key_part3 =?
- SELECT … WHERE key_col + ? = ?

參考：[MySQL優化案例]系列 — RAND()優化

常見殺手級SQL

```
SELECT a.x ...
FROM a
ORDER BY a.y LIMIT 11910298, 20;
```

採用子查詢進行優化 =>

```
SELECT a.x ...
FROM a
WHERE a.pkid > (SELECT pkid FROM a WHERE pkid >= 11910298 ORDER BY
a.y LIMIT 1) LIMIT 20;
```

參考：[MySQL優化案例]系列 — 分頁優化

慢日誌管理

- 利用pt-query-digest解析慢日誌
- 結合Box Anemometer/Query-Digest-UI集中展示慢日誌

slow log file name: slow_query.txt.vote-192.168.0.1-3306

pt-query-digest 解析日誌導入數據表中：
>> dbname=`echo ${slowlog}|awk -F '.txt.|.log.' '{print $NF}'` //得到192.168.0.1-3306
>> pt-query-digest ...--filter="...\$event->{hostname}=\"${dbname}\""

myslow_review_history數據表中：
hostname_max: 192.168.0.1-3306

慢日誌管理

```
/*
$pos = strpos($sample['hostname_max'], ':');
if ($pos === false)
{
        $conn['port'] = 3306;
        $conn['host'] = $sample['hostname_max'];
}
else
{
        $parts = preg_split("/:/", $sample['hostname_max']);
        $conn['host'] = $parts[0];
        $conn['port'] = $parts[1];
}
*/

//modified by yejr, 2013/08/22
$parts = preg_split("/-/", $sample['hostname_max']);
$parts_count = count($parts);

$conn['host'] = $parts[$parts_count - 2];
$conn['port'] = $parts[$parts_count - 1];
```

myslow_review_history數據表中：
hostname_max: 192.168.0.1-3306

解析成：
$conn[‘host’] = ‘192.168.0.1’;
$conn[‘port’] = 3306;

而後使用通用賬號密碼連接到目標DB

路在腳下 run run run