数据库极限性能实践

研发中心　　邵宗文

数据库极限性能实践

- 为何要测试数据库的极限性能
- 如何着手进行测试
- 通过极限性能了解，合理使用和规避风险。

为何要进行数据库性能测试

▲**低层次** —— 能知道大概要申请多少机器

▲ **一般** —— 能够合理规划分库和分表

▲**高层次** —— 知道数据库性能不足，通过架构或是别的手段去弥补

结合项目实际，了解自己的应用

- 应用类型

  读多写少（如体育项目）；读写比例差不多（如邮件）；和写多读少（如投票，统计）

- 预计数据量

  半年？一年？后续扩展？➔ 决定单表还是多表，扩展的方法

- 预计访问量

  多少读？多少写？峰值？➔ 几台服务器，主从方式

- 实时数据和非实时数据

  哪些必须实时查询？哪些可以预先准备或近似？哪些用于统计汇总？

- 时间的要求

  实时性高的项目，如财经、体育；可以允许一定时间延迟的项目，如博客圈

结合机器配置，了解自己的机器配置

- 机器型号和批次

  数据库平台主要采用dell的服务器。批次不同存在性能上的差异(如06、07、08的cpu的cache size)

- 了解机器磁盘性能

  是多磁盘做了lvm，还是做了raid，做了raid的话，要知道是raidN?

- 了解机器的内存

  比如内存的容量和插槽几个，如有需要，建议购买单条容量高的内存

- 了解网络瓶颈

  有时项目读写量达到瓶颈后，可能是网络质量造成。跨IDC大量写操作会很慢

- 了解固态硬盘ssd

  比如一些属于写少读多的项目可以考虑使用

如何进行数据库测试——步骤1

选择测试的方案

- 选择市场上常见的服务器

  选用了市面比较常规的dell服务器

- 选择数据库版本

  目前由于5.1的版本还有很多bug,因此新浪选择了使用很久的mysql5.0.41.

- mysql引擎选择

  因为互联网业务大部分是写少读多，所以myisam是不二的选择

- 测试工具

  推荐mysqlslap

- 操作系统

  CentOS5.0 x86_64

如何进行数据库测试——步骤2

具体实施测试计划

- 进行限定系统内存

  通过对grub.conf文件添加mem=1G的参数

- 选择测试表 （见后页)

  测试样本用例10个表总数据量为几十G，对十个表进行总操作为100万次的读写，以及200万次，400万次，800万次的读操作测试

- 选择不同的并发

  比如10，50，100并发执行同样的sql语句下的性能

- My.cnf的配置

  关闭query_cache，同时使用很低的参数(见后页)

如何进行数据库测试——步骤3

★ 测试用例表

```
CREATE TABLE `song1`
  `id` int(10) unsigned NOT NULL auto_increment,
  `name1` varchar(250) NOT NULL,
  `name2` varchar(250) NOT NULL,
  `name3` varchar(250) NOT NULL,
  `name4` varchar(250) NOT NULL,
  `name5` varchar(250) NOT NULL,
  `datetime` timestamp NOT NULL default
    CURRENT_TIMESTAMP,
  `rank` int(11) NOT NULL,
  PRIMARY KEY  (`id`)
```

的Song0..Song9

如何进行数据库测试——步骤4

★ My.cnf

key_buffer = 124M
max_allowed_packet = 2M
table_cache = 1024
join_buffer_size = 2M
sort_buffer_size = 2M
read_buffer_size = 2M
read_rnd_buffer_size = 2M
myisam_sort_buffer_size = 180M
query_cache_size = 0
query_cache_limit = 2M
max_tmp_tables = 128
tmp_table_size = 124M
max_heap_table_size = 124M
thread_cache = 48
thread_concurrency = 16
max_connections = 484

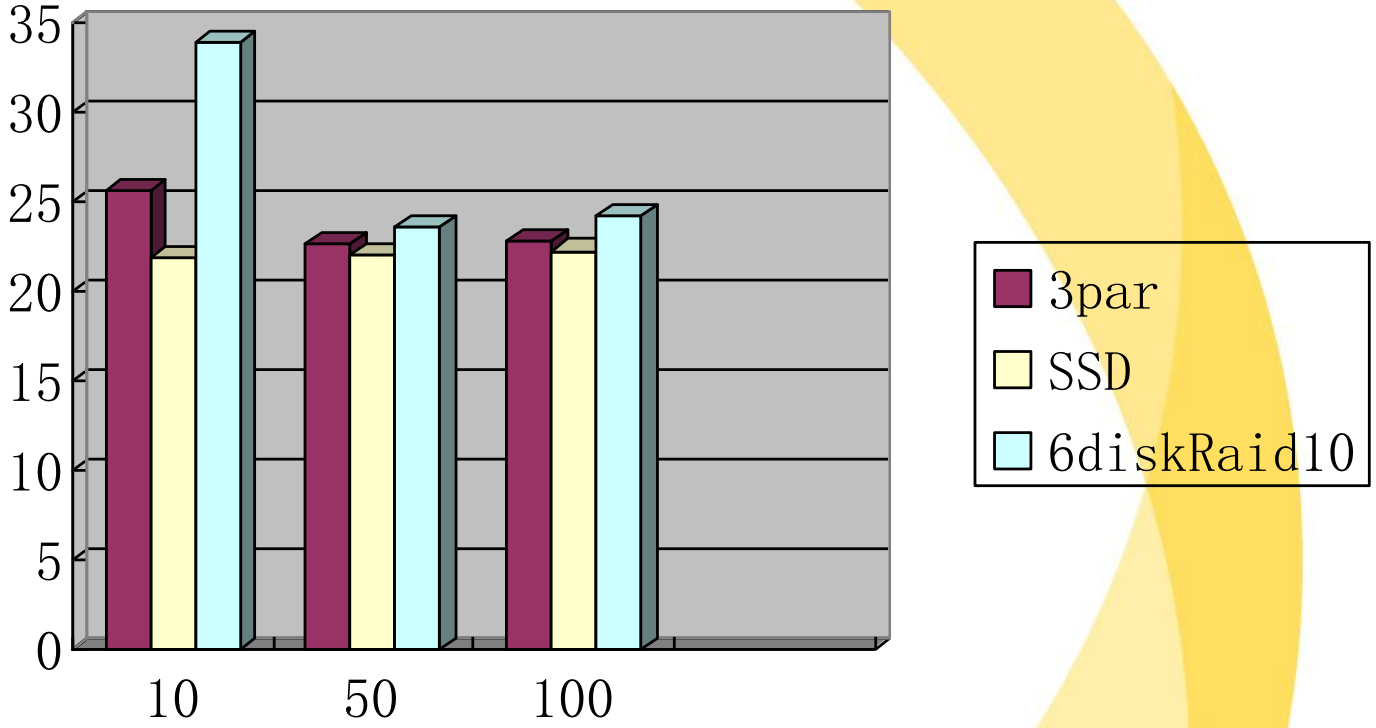
如何进行数据库测试——步骤5

.磁盘用例和目的

数据库平台的机器基本都是用raid10,或是lvm将6个盘组合成一个盘来进行服务，同时结合目前比较流行的ssd或高端商业存储3par来评测优劣，推荐选择用6磁盘做raid10的单盘来进行评测。(防止太悬殊)

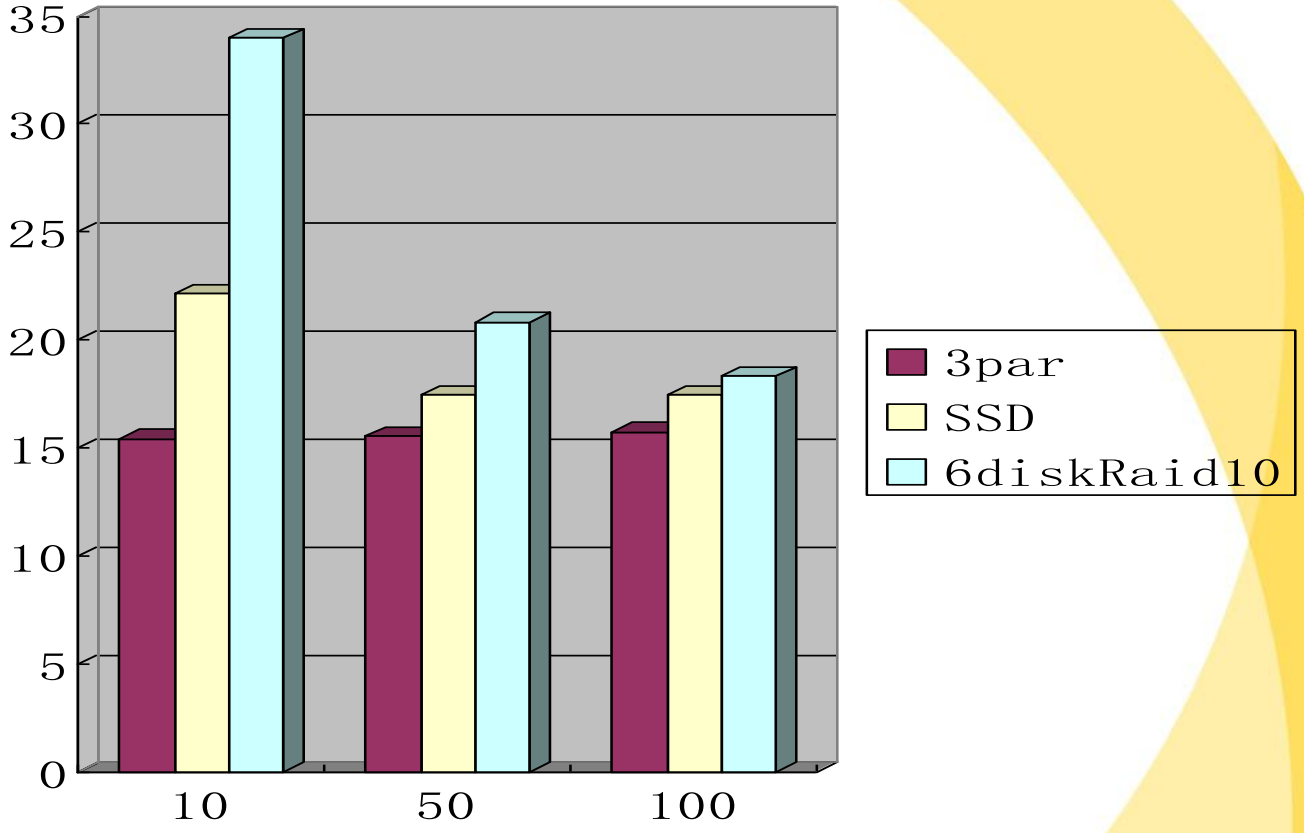
SSD 1块 机器 Dell Cpu:E5410 2.33GHz*4core*2
容量：32G SATA-2
6磁盘raid10 6块 机器 Dell Cpu:E5410 2.33GHz*4core*2
容量:780G
64磁盘3par 64块 机器 Dell Cpu:E5410 2.33GHz*4core*2
容量:152G(条带化)

测试结果100万次Insert操作



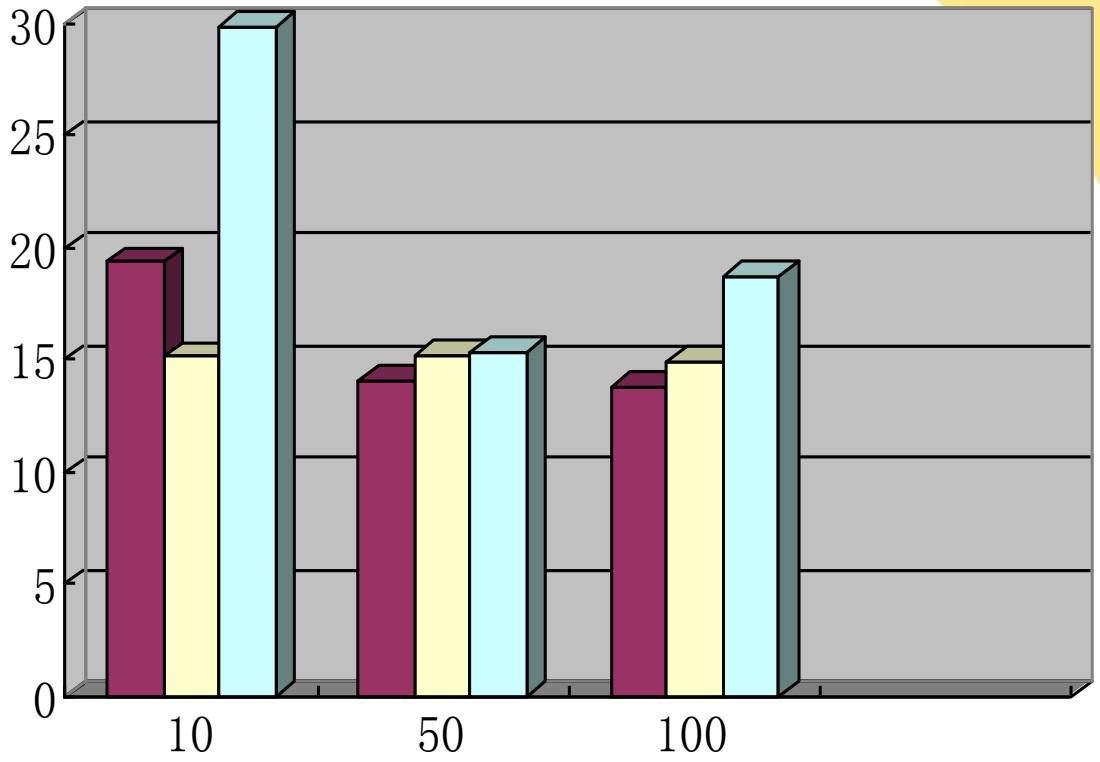
Insert	10	50	100
3par	25.617s	22.619s	22.816s
SSD	21.834 s	22.090s	22.274s
			24.196s

测试结果100万次update操作



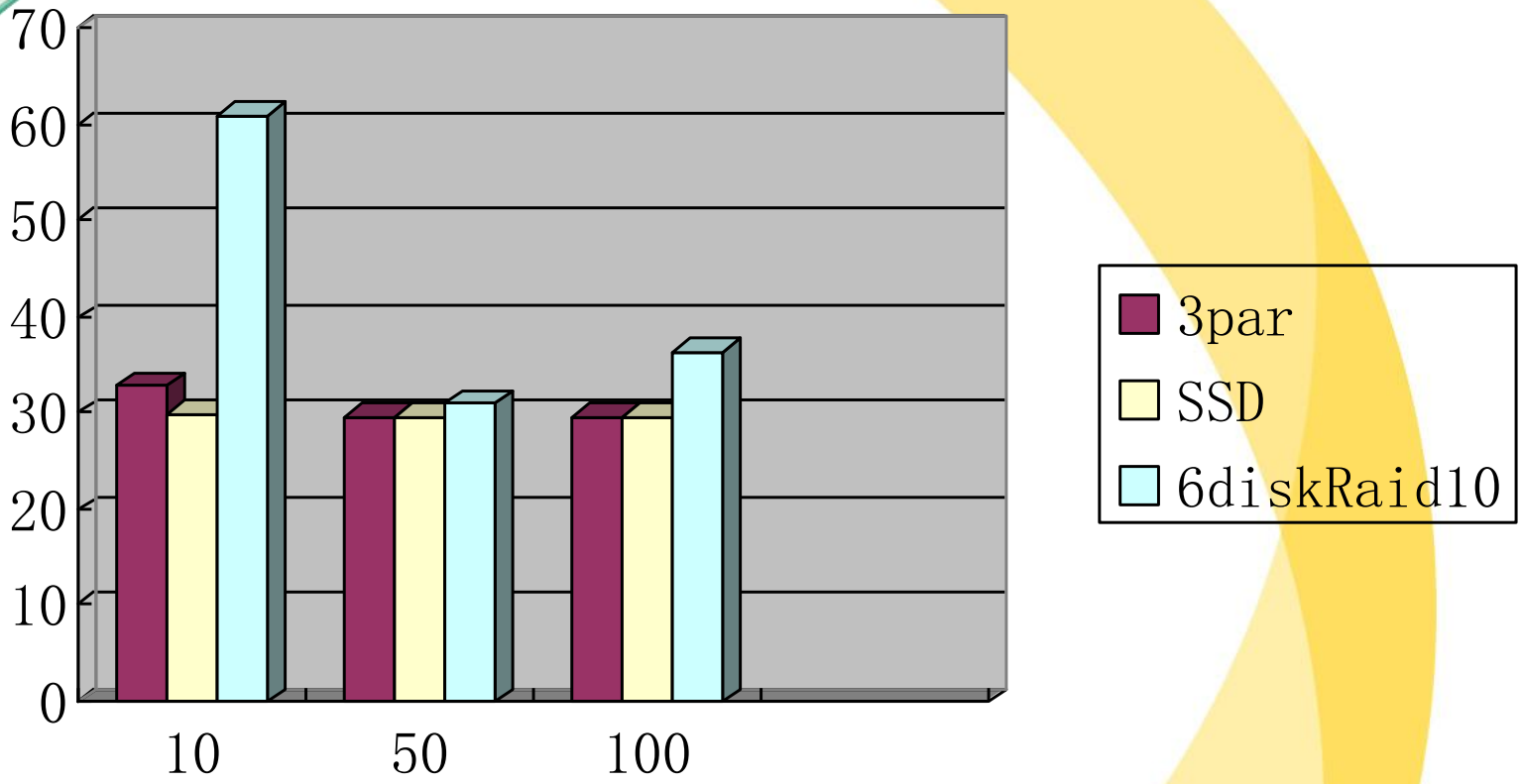
Update	10	50	100
3par	15.444s	15.549s	15.770s
SSD	22.146 s	17.472s	17.472 s
			18.345 s

测试结果100万次select操作



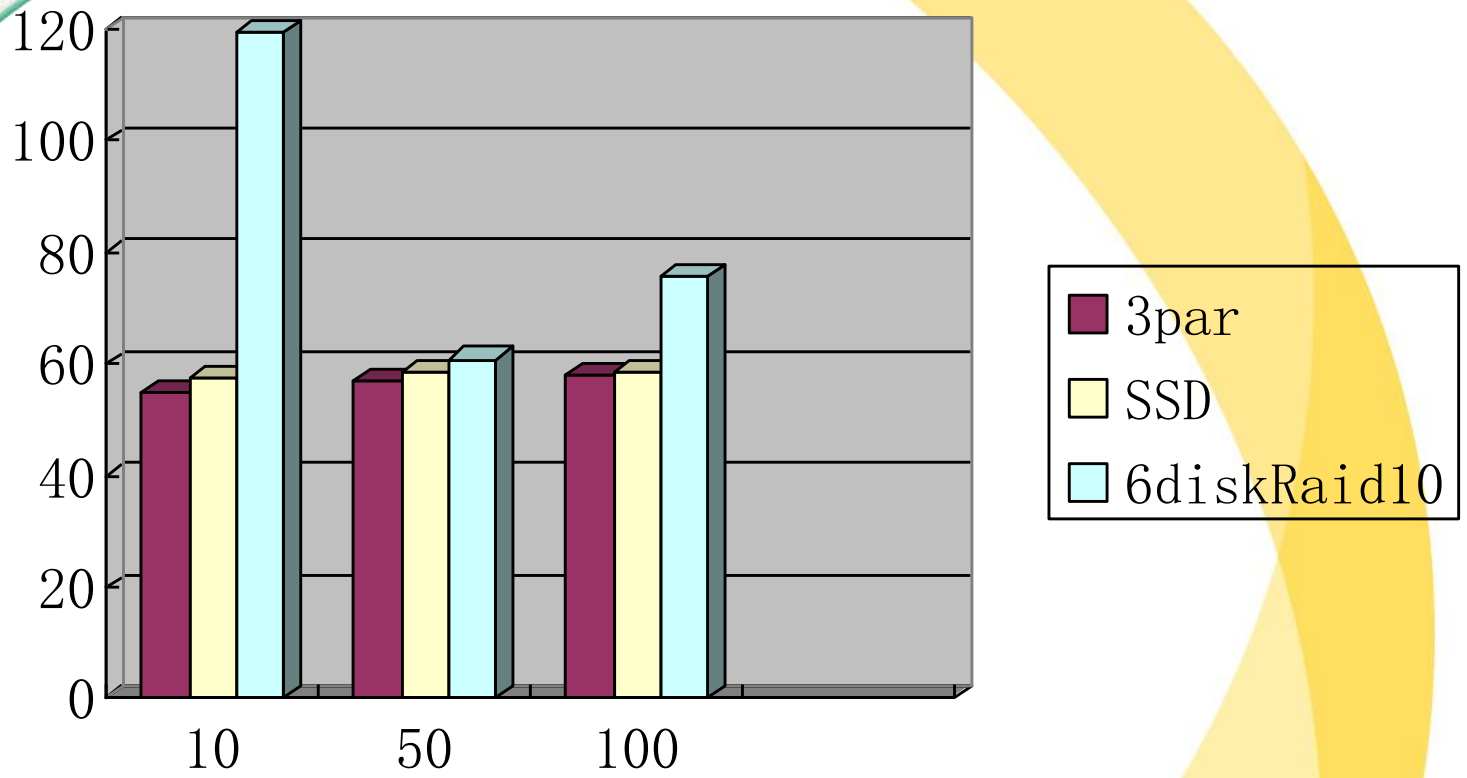
Select	10	50	100
3par	19.364s	14.042s	13.794s
SSD	15.142 s	15.188 s	14.885 s
6diskRaid10	29.892 s	15.351 s	18.674 s

测试结果200万次select操作



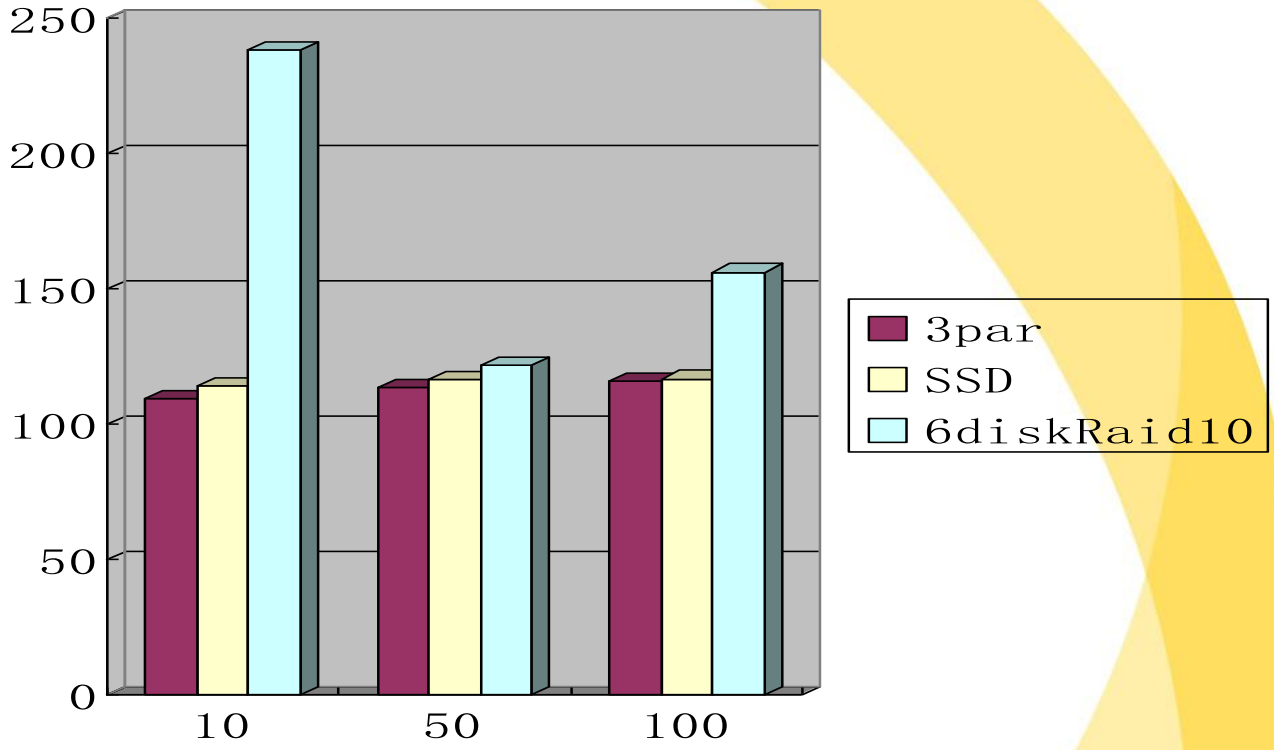
Select	10	50	100
3par	32.930	29.631	29.631
SSD	29.904 s	29.583s	29.616 s
6diskRaid10	61.026 s	31.073 s	36.360 s

测试结果400万次select操作



Select	10	50	100
3par	54.837s	56.832s	57.994s
SSD	57.598 s	58.360s	58.575 s
6diskRaid10	119.367s	60.740 s	75.637 s

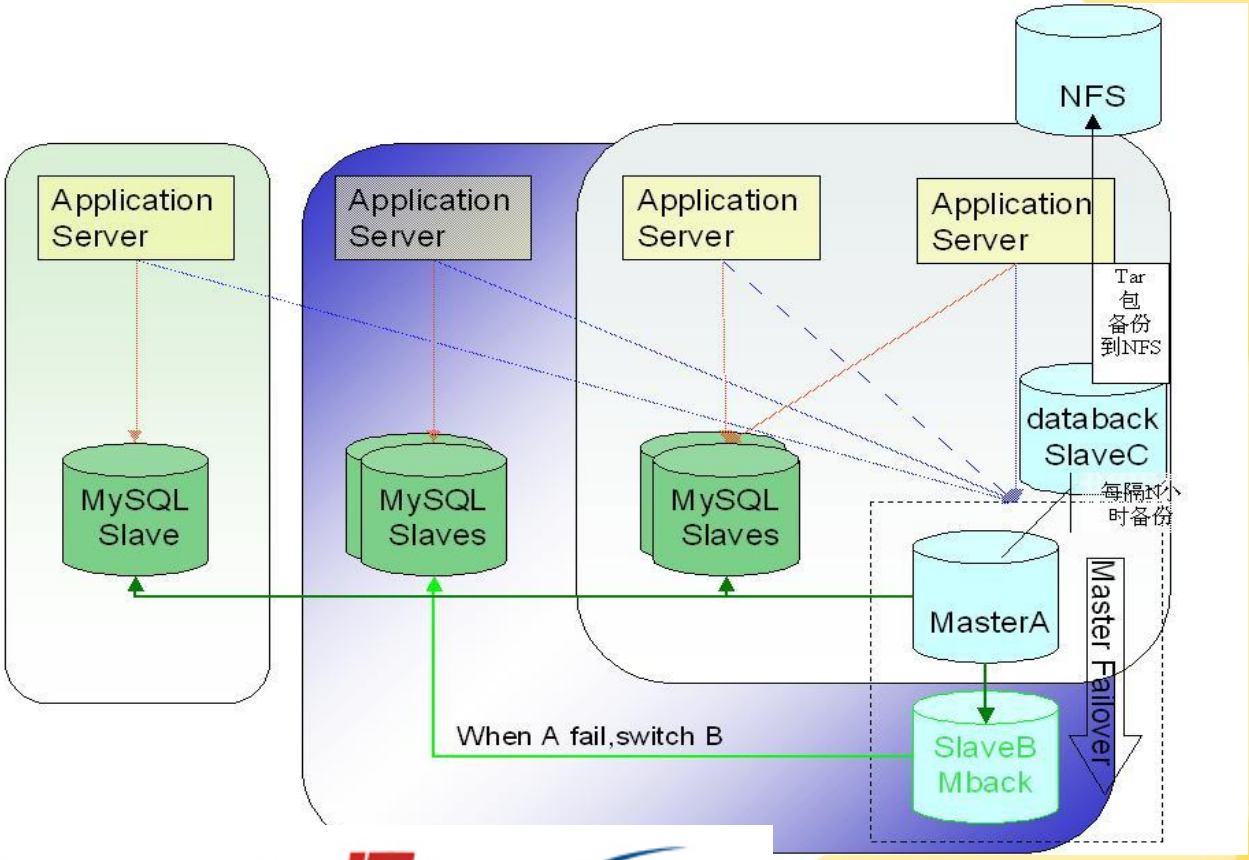
测试结果800万次Select操作



Select	10	50	100
3par	109.454	113.642	116.042
SSD	114.198s	116.557s	116.741 s
6diskRaid10	238.281s	121.584 s	156.120 s

如何通过架构来弥补

1. 通过简单读写分离



2. 通过分多个主库,便于未来可扩展

top - 17:48:02 up 497 days, 22:31, 3 users, load average: 0.36, 0.65, 0.70
Tasks: 145 total,   1 running, 144 sleeping,   0 stopped,   0 zombie
Cpu(s): 2.1%us, 8.3%sy, 0.0%ni, 88.2%id, 0.2%wa, 0.1%hi, 1.1%si, 0.0%st
Mem: 16627888k total, 16209572k used,   418316k free,   155352k buffers
Swap: 4289312k total,   300k used, 4289012k free, 12527292k cached

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
24655	my3310	15	0	980m	809m	4428	S	18	5.0	19791:13	mysqld
32746	my3311	15	0	970m	873m	4680	S	9	5.4	14362:23	mysqld
16334	my3309	15	0	761m	704m	4720	S	8	4.3	33603:30	mysqld
14612	my3308	15	0	851m	802m	4784	S	2	4.9	14745:18	mysqld

3. 通过多IDC提升数据库平台99.999%稳定性

如何通过其他手段来弥补

1. 通过简单的key-value模式数据库来处理简单逻辑业务

如berkeley DB, LightCloud, Tokyo Tyrant

2 .通过Memcache来缓冲如投票等频繁update的数据库

比如通过设定阀值500次才往数据库做一次写操作，或是间隔30分钟往数据库写一次。

3 .通过使用如ebay公司开发的heap补丁来解决一些如session业务

比如跑一些数据总大小不会很大，但是update特别频繁的，比如用户状态值，补丁的好处是省内存。

4 .通过使用replicate_do_db(table)来解决从库追主库延迟时间较长的问题

由于mysql的从库只能单进程追，而通过上述方式，就能形成多进程追不同库来减少延迟，缺点是管理成本会很大。

5 .通过使用insert批量的方式来提升主库的写速度

比如财经各股的交易数据，还有博文入博客圈的时候，因为同一篇博文能进不同的圈，通过批量values模式都能提升主库写性能。

6.采取从库不同索引的模式来提升性能

比如有些项目，有很多不同的排序需求，需要建立很多索引，但是如果都加必然导致性能下降，所以采取不同功能使用对应索引的从库来解决。

7．可以选择merge引擎来提升代码开发速度

1.比如有些项目，需要定期存用户离线消息，可以采取程序只访问对应的merge表，然后merge表对应7个子表(比如周一到周日)。

2.比如统计项目，可能分表策略是每个月一个表，然后要做如一季度，二季度的统计，为了方便开发，可以采取程序只访问对应merge表，然后自由结合1，2，3，4，N表作为merge表的子表。

结 语

“结合实际情况不断优化”

“让数据库多做它擅长的工作”

谢 谢 参 与!