Orion - oracle 提供的测试 io 性能的工具

Orion 是 **oracle** 提供的测试 io 性能的工具。它可以用来模拟 Oracle 数据库 IO，也可以用来仿
真 ASM 的条带化的功能。

Orion 可以支持下列 IO 负载

1. 小的随机的 IO：OLTP 的应用主要是随机的读写，大小和数据的块大小一样（一般是 8K）。
这样的应用主要是关注的吞吐量是 IOPS 和一个请求的平均延时时间。Orion 可以仿真一个
随机 IO 负载。指定的读写百分比，指定的 IO 大小，指定的 IOs，IOs 是分布在不同的磁盘上。

2. 大的连续的 IO：数据仓库的应用，数据装载，备份，和恢复会产生连续的读写流，这些
读写是由多个 1M 的 IO 组成的。这些应用都是处理大数据量的数据，主要是关注总体的数据
吞吐量 MBPS

3. 大的随机的 IO: 一个连续的读写和其他的数据库活动同时访问磁盘。基于条带化，一个
连续的读写扩展到多个磁盘上。因此，在磁盘的这个层次上，许多的连续的读写被看作随机
的 1M 的 IO,又被称作多用户的连续 IO。

4. 混合的负载: Orion 可以同时仿真前俩种负载：小的随机的 IO，大的连续的 IO。这将使你
可以仿真，OLTP 的 8K 的随机读写的负载和 4 个连续的 1M IO 读写的备份的负载。

针对不同的 IO 负载，Orion 可以在不同的 IO 压力测试并得到性能参数：MBPS，IOPS，和 IO
延迟时间。负载是术语，代表异步的 IOs 的数目。内部本质来说，每一个负载层次，Orion
软件一直在尽快的发 I/O 请求来完成这个层次的 I/O 负载。针对随机的负载（大的和小的），
负载的层次就是 I/Os 的数目。针对大的连续的负载，负载的层次就是连续的读写流和每次
读写流的 IO 的数目。在负载层次范围内测试指定的负载将帮助用户理解性能是怎么受影响的。

测试目标：

理论上，ORION 可以用来测试任何支持异步的字符设备。ORION 已经在下列类型的设备上测试过。

1. DAS(directed_attatched)的存储：
2. SAN(storage-area network)的存储：
3. ORION 没有在 NAS(network-attached storage).

ORION 对存储设备的供应商：
供应商可以用 ORION 来理解 Oracle 是如何来在存储上执行的。也可以用 Orion 来找出适合
Oracle 最好的存储配置。

ORION 对 Oracle 管理员
Oracle 管理员可以根据期望的工作量使用 Orion 来评估和比较不同的存储阵列。他们也可以
用 Orion 来决定峰值时优化的网络连接数，存储阵列数，存储阵列控制器数，和磁盘数。附
录 A 描述了根据数据库现在的工作量来推测 IOPS 和 MBPS 需求。

开始使用 Orion
1．下载 orion：有 Linux/x86，Solaris/SPARC 和 Windows/x86 版本

2. 安装 Orion

Linux/Solaris：解压 Orion 执行文件。

gunzip orion_linux_x86-64.giz

Windows: 运行安装程序

C:\temp> orion_windows_x86-64.msi

3. 选择测试名，我们使用的名是 mytest

4. 创建文件名 mytest.lun，例如：

/dev/raw/raw1

/dev/raw/raw2

...

/dev/raw/raw8

5. 验证设备是不是可以访问。**Linux** 可以用下面的命令：

$ dd if=/dev/raw/raw1 f=/dev/null bs=32k count=1024

1024+0 records in

1024+0 records out

6. 验证在你的平台上已经有异步 IO 的类库。Orion 测试完全是依赖异步的 IO。在 linux 和
solaris，类库 libaio 需要安装并被访问。环境变量名是 LD_LIBRARY_PATH 或者是 LIBPATH,
window 已经安装异步 IO。

7. 第一次测试，建议使用 simple，simple 测试衡量在不同的负载下的小随机读和大的随
机读。这些结果给我一些想法，不同类型的 IO 和负载下的 IO 性能。simple 测试的命令：

./orion_linux_x86-64 -run simple -testname mytest -num_disks 4

ORION: ORacle IO Numbers -- Version 11.1.0.7.0

mytest_20101218_2205

Test will take approximately 30 minutes

Larger caches may take longer

Orion 生成的 IO 负载层次考虑了在 mytest.lun 文件中磁盘的数目。

8. 结果将被记录在输出文件中。

输出文件：

Orion 将产生几个输出文件,

1. mytest_summary.txt: 这个文件包含：

a. 输入参数

b. 针对大的随机和连续的工作量下观察到的最大的吞吐量

c. 针对小的随机的工作量的 IO 速率

d. 针对小的随机的工作量的最小的延迟时间。

[root@dbs101 software]# more mytest_20101218_2205_summary.txt

ORION VERSION 11.1.0.7.0

Commandline:

-run simple -testname mytest -num_disks 4

This maps to this test:

Test: mytest

Small IO size: 8 KB

Large IO size: 1024 KB

IO Types: Small Random IOs, Large Random IOs

Simulated Array Type: CONCAT

Write: 0%

Cache Size: Not Entered

Duration for each **Data** Point: 60 seconds

Small Columns:, 0

Large Columns:, 0, 1, 2, 3, 4, 5, 6, 7, 8

Total Data Points: 29

Name: /dev/sda5 Size: 102404703744

Name: /dev/sdb1 Size: 102404703744

Name: /dev/sdc1 Size: 102404703744

Name: /dev/sdd1 Size: 102404703744

4 FILEs found.

Maximum Large MBPS=62.80 @ Small=0 and Large=7

Maximum Small IOPS=647 @ Small=20 and Large=0

Minimum Small Latency=7.32 @ Small=1 and Large=0

2. mytest_mbps.csv 文件：这是个 csv 文件。包含大的随机或者连续的 IO 工作量。所有的 csv

输出文件有个 2 维表。行代表大的 IO 负载层次。列代表小的 IO 负载层次。simple 测试不包含

大的和小的 IO 结合。所以 MBPS 文件只有一个列，0 代表没有小的 IO。

more mytest_20101218_2205_mbps.csv

Large/Small, 0, 1, 2, 3, 4, 5, 6, 7,

8, 9, 10, 11, 12, 13, 14, 15, 16, 17,

18, 19, 20

1, 35.27

2, 49.03

3, 55.23

4, 58.20

5, 60.33

6, 60.34

7, 62.80

8, 62.44

在这个例子中，当负载层次在 5 的时候，没有小的 IO 操作，我们得到的数据吞吐量是 60.33MBPS

我们可以用 excel 图形来显示 MBPS 的速率。

test_20101218_2205_mbps.csv

Large/Small, 0, 1, 2, 3, 4, 5, 6, 7,
8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
18, 19, 20

3. mytest_iops.csv: 这是个小的随机的 IO 工作量的 IOPS 吞吐量。

4. mytest_lat.csv: 这是个小的随机的 IO 工作量下的延迟时间。

more mytest_20101218_2205_lat.csv

Large/Small, 1, 2, 3, 4, 5, 6, 7, 8,
9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
19, 20
0, 7.32, 8.63, 9.93, 11.15, 12.31, 13.38, 14.59, 15.71, 16
.83, 18.23, 19.36, 20.45, 21.77, 23.02, 24.43, 25.73, 27.16, 28.23, 29
.57, 30.87
1
2
3
4
5
6
7
8

5. mytest_trace.txt: 包含扩展的，未处理的测试输出。

输入参数：Orion 可以使用命令的参数来测试任意一种工作量。

强制输入的参数：

run：测试运行的层次，这个选项提供 simple，normal，advanced 的层次。如果没有指定

advanced，那么设置有些非强制的参数（-cache_size 和-verbose）将会出错。

simple：产生小的随机的 IO 和大的连续的 IO 工作量。在这个选项中，小的随机的 IO 和

大的连续的 IO 是分开测试的。这个参数对应下列的 Orion 调用：

```
./orion -run advanced -testname mytest \
-num_disks 4 \
-size_small 8 -size_large 1024 -type rand \
-simulate concat -write 0 -duragion 60 \
-matrix basic
```

normal: 除了 simple 的功能外，还会产生小的随机的 IO 和大的连续的 IO 的结合。

```
./orion -run advanced -testname mytest \
-num_disks 4 \
-size_small 8 -size_large 1024 -type rand \
-simulate concat -write 0 -duragion 60 \
-matrix detailed
```

advanced: 如果用这个选项，用户需要指定可选的参数。

testname: 输入文件必须是<testname>.lun

num_disks: 实际测试的物理磁盘的数目。

可选的输入参数：

help：帮助信息

size_small: 小的随机工作量的 IO 的大小（KB）

size_large: 大的随机的或者连续工作量的大小（KB）。

type：大的 IO 的工作量（默认是 rand）：

rand：大的随机的 IO

seq：大的连续的 IO

num_streamIO: 每个大的连续读写流的 IO 数目。只是在-type seq 下使用。

simulate：大的连续的 IO 工作量小的数据分布。

contact：串联指定的 luns 成一个虚拟的卷。在虚拟的卷上的连续的测试从某个点到一个

lun 的结束点。然后再到下一个 lun。

raid0：在指定的 luns 上条带化成一个虚拟的卷。条带的大小是 1M（和 asm 的条带大小一

致），可以通过参数-stripe 来更改。

write: 和读相比的写的百分比，这个参数在小的随机的和大的连续的 IO 工作量下适用。在大

的连续的 IO，每个读写流要么是读要么是写。这个参数是指只是写百分比。写的数据都是垃

圾数据。 写的测试将破坏的指定的 lun。

cache_size: 存储阵列的读写缓存大小（MB）。针对大的连续的 IO 工作量，Orion 将在每个测

试点之前 warm 的 cache。使用缓存大小来决定缓存操作。如果没有指定，将有个默认值。如果

是 0 的话，将没有 warm 缓存。

duration: 每个测试点的时间。（默认是 60）

matrix: 混合工作量测试的类型

basic：没有混合的工作量，小的随机的 IO 和大的连续的 IO 分开测试。

detailed：小的随机的 IO 和大的连续的 IO 结合起来测试。

point: 单个测试点，S 代表小的随机的 IO，L 代表大的随机/连续的 IO。S -num_small L -num_large

col: 大的随机/连续的 IO

row： 小的随机的 IO

max：和 detailed 一样，只是在最大的负载下测试工作量。可以用-num_small 和 -num_large 参数指定。

num_small: 小的随机的 IO 的最大数目。

num_large: 大的随机的 IO 或者一个读写流的并发数目。

verbose：打印进度和状态到控制台。

命令行例子：

为了理解你的存储性能，小的随机读和大的随机 IO 读工作量，先开始运行：

./orion -run simple -num_disks 4 -testname mytest

测试小的随机读和大的随机读的 IO 工作量，运行：

./orion -run normal -testname mytest -num_disks 4

测试 32K 和 1MB 随机读的组合

./orion -run advanced -testname mytest -num_disks 4 -size_small 32 \

-size_large 1024 -type rand -matrix detailed

测试 1MB 连续写流，仿真 1MB 的 raid-0 条带化

./orion -run advanced -testname mytest -num_disk 4 -simulate raid0 \

-stripe 1024 -write 100 -type seq -matrix col -num_small 0

常见问题：

在<testname>.lun 中的卷发生 IO 错误：

用 dd 拷贝文件命令来验证

验证操作系统支持异步 IO

在 linux 和 solaris 中， 类库 libaio 必须在类库路径中

如果使用的是 NAS

确保文件系统被加载

<testname>.lun 的文件要包含一个或多个已有文件，Orion 不和目录或加载点工作。文件要足够大，能代表你实际数据文件大小。

NFS 在 linux 下异步 IO 性能很差

如果测试的时候遇到没有初始化的或者未写过的块的时候，有些智能的 NAS 系统将产生伪造的数据，解决方法是写所有的块。

如果在 windows 下测试

在裸设备上测试的时候，要映射一个盘符

如果是运行 32 位 Orion 在 64 位 Linux 上

拷贝一份 32 位的 libaio 到 64 位机器上

如果测试的磁盘数超过 30

你应该使用 duration 参数，并为每个测试点制定一个更长的时间（120 秒）。因为 Orion 让所有的轴都运行在一个负载下。每个测试点需要加大时间。

你可能遇到下列的错误

specify a longer -duration value.

类库的错误

参考第一个常见错误

NT-ONLY：确保 oracle 的类库和 orion 在同一个目录

遇到"unbelievably good"

可能有个很大读写缓存，一般存储阵列控制器有很大的影响。找出缓存的大小，并在参数-cache_size 中为 orion 指定。

卷的大小不够，尝试关闭缓存。如果其他卷共享存储，将会看到突出的 IO 操作。

Orion 报告长时间运行

如果 num_disks 高的话，运行时间也很长。

参数 cache_size 影响运行时间，Orion 为每个测试点准备缓存需要 2 分钟。 如果你关闭了你的缓存，你可以指定参数 cache_size=0

如果指定的参数 duration 很长，运行时间是很长。

附录 A：分类数据库的 IO 负载

为了正确的配置数据库的存储设备，必须了解数据库的性能需求。

1 IO 请求主要是单个块还是多个块

数据库将发出多个 IO 请求：并行查询，查询大数据量的表扫描，直接数据装载，备份恢复。一般来说，OLTP 主要是单个 IO 请求，DSS 数据仓库是多个 IO 请求。

2 平均和峰值的 IOPS 是多少？写占多少百分比。

3 平均和峰值的 MBPS 是多少？写占多少百分比。

如果你的数据库 IO 请求主要是单个块，那就关注 IOPS，如果数据库 IO 请求主要是多个块，那就关注 MBPS。

10gR2 数据库：可以从视图 v$sysstat 得到 IO 的类型。

单个数据块的读："physical read total IO requests" - "physical read total multi block requests"

多个数据块的读："physical read total multi block requests"

读的总和："physical read total IO requests"

单个数据块的写："physical write total IO requests" - "physical write total multi block requests"

多个数据块的写："physical write total multi block requests"

写的总和："physical write total IO requests"

使用这些数据，你可以评估在一段时间范围内（包含正常时间和峰值时间）读写的 IOPS 和 MBPS，

```
select name, value
  from v$sysstat
 where name in ('physical read total IO requests',
        'physical read total multi block requests',
        'physical write total IO requests',
        'physical write total multi block requests');
```

NAME VALUE

physical read total IO requests 2068290092

physical read total multi block requests 2255995

physical write total IO requests 9968770

physical write total multi block requests 251551

单个数据块读是 98%

单个数据块写实 97%

也可以从 awr 报表中得到这些数据。

Instance Activity Stats DB/Inst: DBS108A/dbs108a Snaps: 8881-8882

-> Ordered by statistic name

Statistic	Total	per Second	per Trans
...			
physical read total IO requests	27,791	15.7	38.7
physical read total bytes	319,881,216	180,368.5	444,897.4
physical read total multi block	115	0.1	0.2
...			
physical write total IO requests	4,278	2.4	6.0
physical write total bytes	49,528,320	27,927.1	68,885.0
physical write total multi block	22	0.0	0.0

附录 B：数据仓库

在数据仓库设计和管理的时候，IO 性能是一个关键的部分，典型的数据仓库系统是 IO 集中，操作在大数据量上，数据加载，重建索引和创建物化视图。数据仓库支持的 IO 必须设计符合过度的需求。

数据仓库的存储配置是根据 IO 带宽，而不是总的容量。磁盘的容量比磁盘吞吐量速

率发展快，结果少数几个磁盘可以存储大量的数据。但是大量的磁盘不能提供同样

IO 吞吐量。你可以用多个磁盘和管道来得到最大的带宽。条带化是一种方法来实现。实现一个大的条带大小（1M）来确保时间来定位磁盘和传输数据。

orion 可以仿真连续的 IO 吞吐量，例如：
白天的工作量：当终端客户，其他应用查询系统：许多单独的并发只读 IO
数据装载：终端用户可能访问数据库，写的工作量和一些可能并行读（也许是装载程序或者终端用户）
重建索引和物化视图：读写工作量
备份：只读的工作量，可能高的并行度

使用下列选项来仿真不同的数据仓库的工作量

run：使用 advanced 来仿真只读连续的 IO
large：大的连续读的 IO 大小。这个参数是 os io 大小的整数倍。
type： 使用 seq
num_streamIO: 增加这个参数来仿真并行执行操作。指定计划的并行度。一个好的开始点是 CPU 数目*一个 CPU 的线程数。
simulate：如果使用硬件条带化或者卷管理器条带化，则使用 concat。如果还没有条带化，比如 ASM，则使用 raid0。默认条带大小是 1M。
write：写占用的百分比。
matrix: 使用 point 来仿真单个连续工作量，或者使用 col 来仿真不断增加的大的连续的工作量。
num_large: 最大的大的 IOs

```
./orion -run advanced \
-testname mytest \
-matrix point \
-num_small 0 \
-num_large 4 \
-size_large 1024 \
-num_disks 4 \
-type seq \
```

-num_streamIO 8 \

-simulate raid0 \

-cache_size 0 \

-write 0

-verbose

Commandline:

-run advanced -testname mytest -matrix point -num_small 0 -num_large 4

-size_lar

ge 1024 -num_disks 4 -type seq -num_streamIO 8 -simulate raid0 -cache_size 0

-wr

ite 0

This maps to this test:

Test: mytest

Small IO size: 8 KB

Large IO size: 1024 KB

IO Types: Small Random IOs, Large Sequential Streams

Number of Concurrent IOs Per Stream: 8

Force streams to separate disks: No

Simulated Array Type: RAID 0

Stripe Depth: 1024 KB

Write: 0%

Cache Size: 0 MB

Duration for each Data Point: 60 seconds

Small Columns:, 0

Large Columns:, 4

Total Data Points: 1

Name: /dev/sda5 Size: 102404703744

Name: /dev/sdb1 Size: 102404703744

Name: /dev/sdc1 Size: 102404703744

Name: /dev/sdd1 Size: 102404703744

4 FILEs found.

Maximum Large MBPS=66.88 @ Small=0 and Large=4

从测试数据看，在这种情况下，吞吐量是 66.88M。理想的情况下：oracle 可以达到 95%。

下面的语句 4 个并发的会话。

```
select /*+ NO_MERGE(sales) */ count(*) from
 (select /*+ FULL(s) PARALLEL (s,8) */* from all_sales s) sales
```

在一个很好平衡的数据仓库配置，应该有足够的 IO 来利用 CPU。作为一个起始点，可以

使用下列规则：一个 GHZ 的 CPU 可以驱动 100M。比如说有 4 个 3G 的 CPUs，那么你的存储应

该提供 4*3*100=1200MBPS 的吞吐量。在 **RAC** 环境中，这个数量可以乘以节点的数目。

1. 1 disk

./orion_linux_x86-64 -run simple -num_disks 1 -testname mytest1

ORION VERSION 11.1.0.7.0

Commandline:

-run simple -num_disks 1 -testname mytest1

This maps to this test:

Test: mytest1

Small IO size: 8 KB

Large IO size: 1024 KB

IO Types: Small Random IOs, Large Random IOs

Simulated Array Type: CONCAT

Write: 0%

Cache Size: Not Entered

Duration for each Data Point: 60 seconds

Small Columns:, 0

Large Columns:, 0, 1, 2

Total Data Points: 8

Name: /dev/sda5 Size: 102404703744

1 FILEs found.

Maximum Large MBPS=29.20 @ Small=0 and Large=2

Maximum Small IOPS=177 @ Small=4 and Large=0

Minimum Small Latency=7.59 @ Small=1 and Large=0

从这里看到这个磁盘的极限是 177IOPS，29.20MBPS。

avg-cpu: %user %nice %sys %iowait %idle

0.10 0.00 0.13 12.38 87.39

Device: rrqm/s wrqm/s r/s w/s rsec/s wsec/s rkB/s wkB/s avgrq-sz

sda5 0.00 0.00 164.73 0.60 2635.67 9.62 1317.84 4.81 16.00

avgqu-sz await svctm %util

3.01 18.26 6.06 100.22

从 IOSTAT 的输出看到当这个磁盘的 IOPS 是 164 的时候，%util 利用率已经是 100%。等待时间是

18ms。

2. 2 disk

./orion_linux_x86-64 -run simple -num_disks 2 -testname mytest2

Commandline:

-run simple -num_disks 2 -testname mytest2

This maps to this test:

Test: mytest2

Small IO size: 8 KB

Large IO size: 1024 KB

IO Types: Small Random IOs, Large Random IOs

Simulated Array Type: CONCAT

Write: 0%

Cache Size: Not Entered

Duration for each Data Point: 60 seconds

Small Columns:, 0

Large Columns:, 0, 1, 2, 3, 4

Total Data Points: 15

Name: /dev/sda5 Size: 102404703744

Name: /dev/sdb1 Size: 102404703744

2 FILEs found.

Maximum Large MBPS=50.94 @ Small=0 and Large=4

Maximum Small IOPS=330 @ Small=10 and Large=0

Minimum Small Latency=7.41 @ Small=1 and Large=0

从 summary 文件中看到两个磁盘的极限，MBPS 是 50.94M, IPOS 是 330。

avg-cpu: %user %nice %sys %iowait %idle

0.15 0.00 0.13 12.51 87.22

Device: rrqm/s wrqm/s r/s w/s rsec/s wsec/s rkB/s wkB/s avgrq-sz avgqu-sz await svctm %util

sda5 0.00 0.00 101.61 0.60 1622.49 14.46 811.24 7.23 16.02 1.03 10.11 7.56 77.29

sdb1 0.00 0.00 99.20 0.40 1600.00 12.85 800.00 6.43 16.19 0.99 9.91 7.50 74.70

从 IOSTAT 的输出看到当这个磁盘的 IOPS 是 200 的时候，%util 利用率已经是 77%。等待时间是

10ms。

avg-cpu: %user %nice %sys %iowait %idle

0.18 0.00 0.20 12.44 87.19

Device: rrqm/s wrqm/s r/s w/s rsec/s wsec/s rkB/s wkB/s avgrq-sz avgqu-sz await svctm %util

sda5 0.00 0.00 166.73 0.80

2686.97 16.03 1343.49 8.02 16.13 5.23 31.10 5.93 99.42

sdb1 0.00 0.00 165.73 0.40

2658.12 12.83 1329.06 6.41 16.08 4.87 29.54 5.81 96.47

从 IOSTAT 的输出看到当这个磁盘的 IOPS 是 330 左右的时候，%util 利用率已经是 99%。等待时间是

30ms。

avg-cpu: %user %nice %sys %iowait %idle

0.10 0.00 0.20 12.63 87.07

Device: rrqm/s wrqm/s r/s w/s rsec/s wsec/s rkB/s wkB/s avgrq-sz avgqu-sz await svctm %util

sda5 0.00 0.00 52.10 0.60 52533.87 9.62

26266.93 4.81 996.93 3.99 76.08 17.75 93.57

sdb1 0.00 0.00 49.70 0.20 51309.02 6.41

25654.51 3.21 1028.37 3.35 66.33 17.02 84.91

从 IOSTAT 的输出看到当这个磁盘的 MBPS 是 51M 左右的时候，%util 利用率已经是 93%。等待时间是

70ms。两个磁盘的条带大小是 1M。第一块磁盘：26266.93/52.1 = 504K, 第二块磁盘：

25654.51/49.7=516K

3. 3 disk

./orion_linux_x86-64 -run simple -num_disks 3 -testname mytest3

Commandline:

-run simple -num_disks 3 -testname mytest3

This maps to this test:

Test: mytest3

Small IO size: 8 KB

Large IO size: 1024 KB

IO Types: Small Random IOs, Large Random IOs

Simulated Array Type: CONCAT

Write: 0%

Cache Size: Not Entered

Duration for each Data Point: 60 seconds

Small Columns:, 0

Large Columns:, 0, 1, 2, 3, 4, 5, 6

Total Data Points: 22

Name: /dev/sda5 Size: 102404703744

Name: /dev/sdb1 Size: 102404703744

Name: /dev/sdc1 Size: 102404703744

3 FILEs found.

Maximum Large MBPS=73.25 @ Small=0 and Large=6

Maximum Small IOPS=492 @ Small=15 and Large=0

Minimum Small Latency=7.30 @ Small=1 and Large=0

从 summary 文件中看到三个磁盘的极限，MBPS 是 73.25M, IPOS 是 492。

avg-cpu: %user %nice %sys %iowait %idle

0.13 0.00 0.20 12.41 87.27

Device: rrqm/s wrqm/s r/s w/s rsec/s wsec/s rkB/s wkB/s avgrq-sz avgqu-sz await svctm %util

sda5 0.00 0.00 108.40 0.80 1731.20 16.00 865.60 8.00 16.00 1.34 12.32 7.24 79.02

sdb1 0.00 0.00 104.60 0.40
1676.80 12.80 838.40 6.40 16.09 1.16 11.03 6.99 73.36
sdc1 0.00 0.00 100.00 0.40
1609.60 12.80 804.80 6.40 16.16 1.14 11.23 7.16 71.88

从IOSTAT的输出看到当三个磁盘的IOPS是300的时候，%util利用率已经是70%。等待时间是
10ms。

avg-cpu: %user %nice %sys %iowait %idle
0.15 0.00 0.23 12.26 87.37

Device: rrqm/s wrqm/s r/s w/s rsec/s wsec/s rkB/s wkB/s avgrq-sz avgqu-sz await svctm %util
sda5 0.00 0.00 165.13 0.60
2658.12 9.62 1329.06 4.81 16.10 4.84 29.14 5.87 97.25
sdb1 0.00 0.00 160.72 0.20
2581.16 6.41 1290.58 3.21 16.08 4.22 26.16 6.01 96.65
sdc1 0.00 0.00 160.92 0.20
2571.54 6.41 1285.77 3.21 16.00 4.00 25.01 5.95 95.79

从IOSTAT的输出看到当三个磁盘的IOPS是485的时候，%util利用率已经是97%。等待时间是
29ms。

3. 3 disk
./orion_linux_x86-64 -run simple -num_disks 4 -testname mytest4

ORION VERSION 11.1.0.7.0

Commandline:
-run simple -num_disks 4 -testname mytest4

This maps to this test:
Test: mytest4

Small IO size: 8 KB

Large IO size: 1024 KB

IO Types: Small Random IOs, Large Random IOs

Simulated Array Type: CONCAT

Write: 0%

Cache Size: Not Entered

Duration for each Data Point: 60 seconds

Small Columns:, 0

Large Columns:, 0, 1, 2, 3, 4, 5, 6, 7, 8

Total Data Points: 29

Name: /dev/sda5 Size: 102404703744

Name: /dev/sdb1 Size: 102404703744

Name: /dev/sdc1 Size: 102404703744

Name: /dev/sdd1 Size: 102404703744

4 FILEs found.

Maximum Large MBPS=63.24 @ Small=0 and Large=8

Maximum Small IOPS=649 @ Small=20 and Large=0

Minimum Small Latency=7.27 @ Small=1 and Large=0

从 summary 文件中看到四个磁盘的极限，MBPS 是 63.24M, IPOS 是 649。

avg-cpu: %user %nice %sys %iowait %idle

0.35 0.00 0.20 12.31 87.14

Device: rrqm/s wrqm/s r/s w/s rsec/s wsec/s rkB/s wkB/s avgrq-sz avgqu-sz await svctm %util

sda5 0.00 0.00 92.77 0.60 1487.55 14.46 743.78 7.23 16.09 1.05 11.28 7.22 67.43

sdb1 0.00 0.00 93.57 0.40 1500.40 12.85 750.20 6.43 16.10 1.14 12.11 7.42 69.72

sdc1 0.00 0.00 86.14 0.40

1375.10 12.85 687.55 6.43 16.04 0.96 11.08 7.29 63.13
sdd1 0.00 0.00 86.95 0.00
1394.38 0.00 697.19 0.00 16.04 0.88 10.09 7.21 62.67

从IOSTAT的输出看到当三个磁盘的IOPS是357的时候，%util利用率已经是70%。等待时间是
10ms。

avg-cpu: %user %nice %sys %iowait %idle
0.15 0.00 0.33 12.18 87.34

Device: rrqm/s wrqm/s r/s w/s rsec/s wsec/s rkB/s wkB/s avgrq-sz avgqu-sz await svctm %util
sda5 0.00 0.00 155.51 0.80
2459.32 9.62 1229.66 4.81 15.79 5.40 34.95 6.15 96.11
sdb1 0.20 0.00 163.33 0.40
2619.64 6.41 1309.82 3.21 16.04 5.35 32.73 5.84 95.67
sdc1 0.00 0.00 163.33 0.40
2606.81 6.41 1303.41 3.21 15.96 4.27 26.08 5.88 96.31
sdd1 0.00 0.00 157.92 0.00
2561.92 0.00 1280.96 0.00 16.22 5.09 31.12 5.99 94.67

从IOSTAT的输出看到当三个磁盘的IOPS是638的时候，%util利用率已经是96%。等待时间是
30ms。

响应时间+IOPS 就说明了系统的当前io状况啊,响应时间在 10ms 左右能达到的最大iops 能力，
是系统io能力的一个最重要指标。

衡量系统io能力，我们也是关注的高峰期一段时间内稳定运行的状况

简单来说，如果一个database在peak time显示出来的db file sequential/scattered read

average wait time 的指标明显低于 7 ms, 那么，我们不应该把焦点集中在 Storage 上。虽然这个
指标不一定说明 Storage 性能很好，但是至少表明系统的瓶颈并不在 Storage 上 1，改善 Storage
性能未必能获得很高回报。反之，如果一个 database 在 peak time 显示出来的 db file sequential/
scattered read average wait time 的指标明显高于 10 ms,我们可能需要关注 Storage 的性能是否
存在瓶颈，当然，同时也可以从降低 Total IO 的角度去处理。

一个 random IO 的理论时间是：
7ms = 4-5ms(磁盘平均寻道时间）+ 2ms (传输时间) + 一些其它的消耗
如果不考虑 file system cache hit(raw device/direct IO) 以及 storage cache hit，同时没
有磁盘竞争，那么，db file sequntial read 的时间就会是 7ms 左右.

而由于 file system cache hit 和 storage cache hit 的情况下，没有磁盘竞争的系统 db file
sequntial read 会小于 7ms 如果有磁盘竞争，而且竞争产生的延迟> file system cache hit 和
storage cache hit 的好处，就会大于 7ms .10ms 可以说是一个经验值，就是磁盘竞争产生的延迟
比较高了