11GR2 ASM I/O规化

ASM I/O规化的对象

I/O规化的对象：

- AU
- 条带

AU有大有小，条带可有可无，什么时候用大AU、什么时候用小AU、是否有必要使用条带，等等，甚至DG中磁盘的顺序，也都有可能影响性能。这些是创建数据库前的规化阶段，必须考虑的问题。

ASM I/O规化的对象

根据条带设置，ASM有两种条带类型：

- 不可调粗粒度。
- 可调细粒度。

此处是否可调，主要指条带大小是否可调。

AU与条带 ---不可调粗粒度

直接在ASM创建DG，默认就是粗粒度不可调条带，比如，下面的语句创建DG1磁盘组，它有4块盘（或LUN），AU大为4M：

```
create diskgroup dg1 external redundancy
disk
'/dev/rdsk/c1t3d0s0',
'/dev/rdsk/c1t4d0s0',
'/dev/rdsk/c1t5d0s0',
'/dev/rdsk/c1t6d0s0'
attribute 'compatible.asm' =
'11.2','AU_SIZE'='4M';
```

AU与条带 ---不可调粗粒度

粗粒度不可调条带，相当于ASM没有条带，或者说，**AU就是条带，条带就是AU**。

条带宽度永远为1，不可改变。条带大小等于AU大小，也同样不可改变。这就是称它为“不可调”的原因。在此模式下，一个AU，其实就是一个条带。

AU与条带 ---可调细粒度（一）

1、创建DG语法同前。

2、添加细粒度模板：

```
alter diskgroup dg1 ADD TEMPLATE stp_fine1 ATTRIBUTES (UNPROTECTED fine);
```

3、创建使用细粒度的表空间

```
create tablespace tbs_data01 datafile
'+dg1(stp_fine1)/tbs_data01_01.dbf'  size 100m reuse uniform size 4m;
```

此表空间数据文件中的数据，将按可调细粒度方式分布。具体方式是，条带大小为128K ，条带宽度为8。

AU与条带 ---可调细粒度（二）

1、创建DG语法同前。

2、使用如下命令定义条带宽度、条带大小：

```
alter system set "_asm_stripewidth"=2;
alter system set "_asm_stripesize"=524288;
```

3、添加细粒度模板：

```
 alter diskgroup dg1 ADD TEMPLATE stp_fine2 ATTRIBUTES (UNPROTECTED
fine);
```

4、创建使用细粒度的表空间

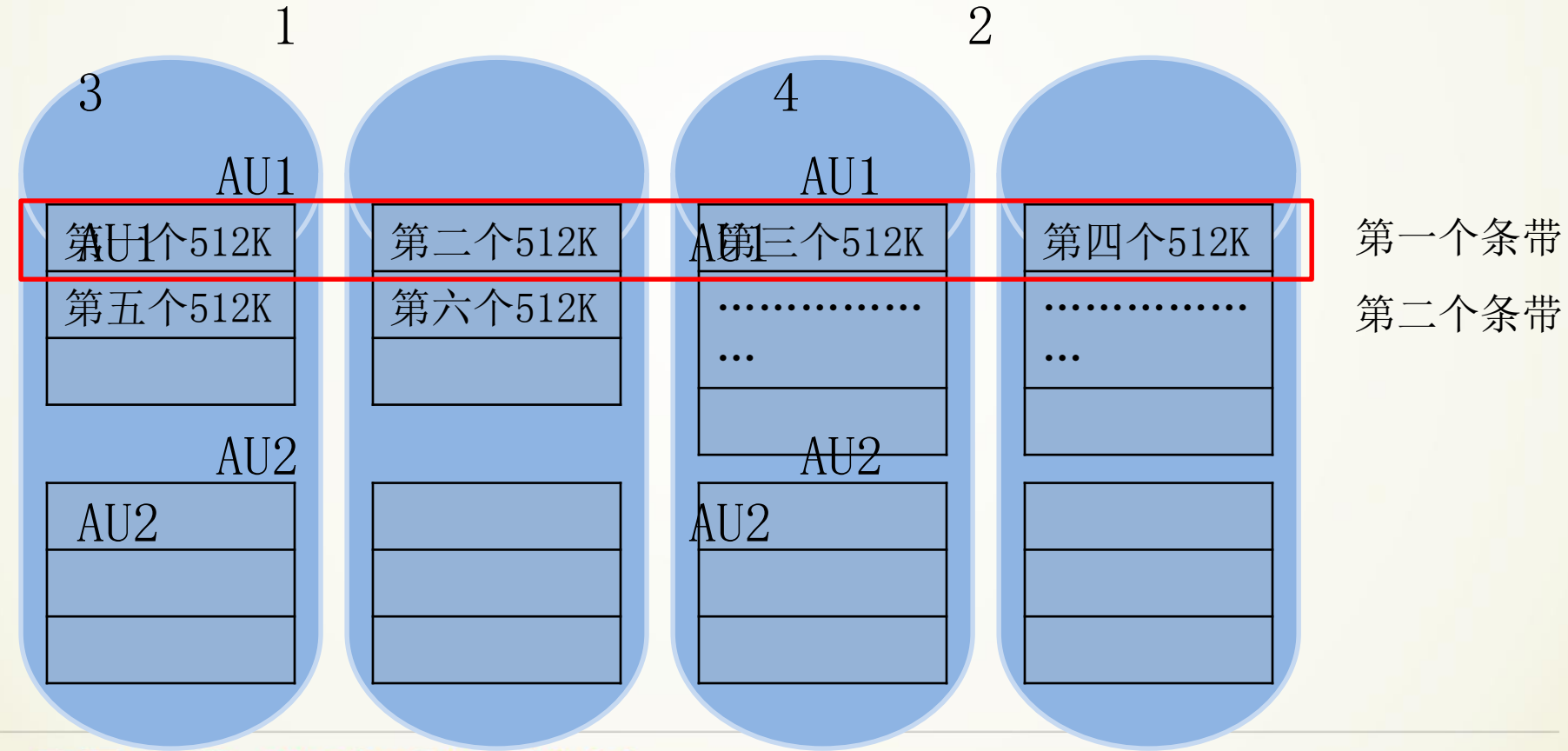
```
 create tablespace tbs_data01 datafile
 '+dg1(stp_fine2)/tbs_data01_01.dbf'  size 100m reuse uniform size 4m;
```

可以使用上面的方式，修改参数自己定义条带大小与宽度。修改参数和添加模板都必须在ASM实例中，且修改参数必须在添加模板前。此表空间的数据文件，将按512K的条带大小，条带宽度为2的方式，存储在DG1的所有磁盘中。

AU与条带 ---可调细粒度（三）

下图仍以四块盘的DG为例。假设AU大小为4M，条带大小为512K，条带宽度为4。

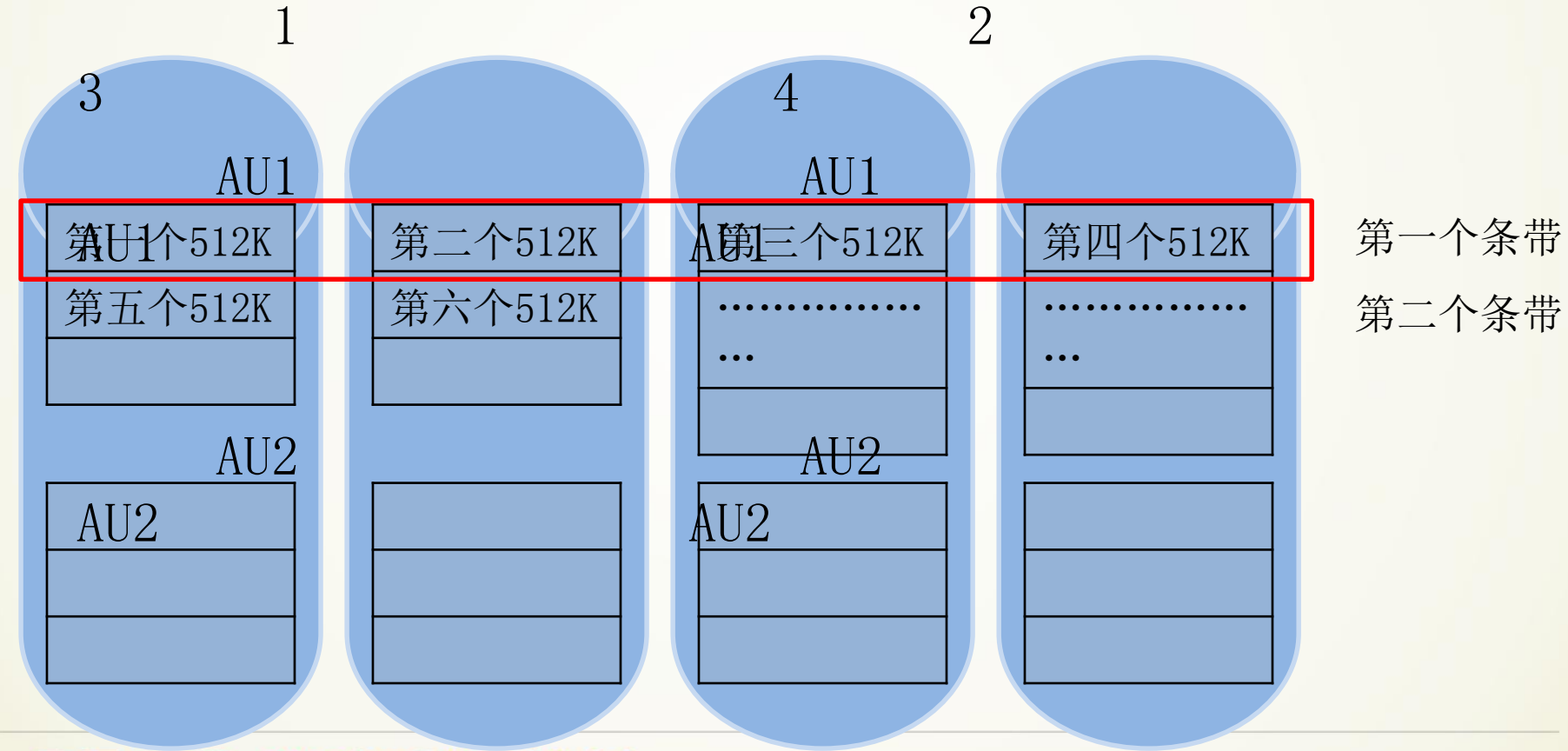
千万不要认为这样的假设在ASM中不成立。有资料说条带大小乘以条带宽度必须等于AU大小，Oracle中没有这样的限制。



AU与条带 ---可调细粒度（四）

数据文件的第一个512K在1号磁盘1号AU 1号条带，第二个512K在2号磁盘1号AU 1号条带，……，第5个512K在1号磁盘1号AU 2号条带，以此类推。

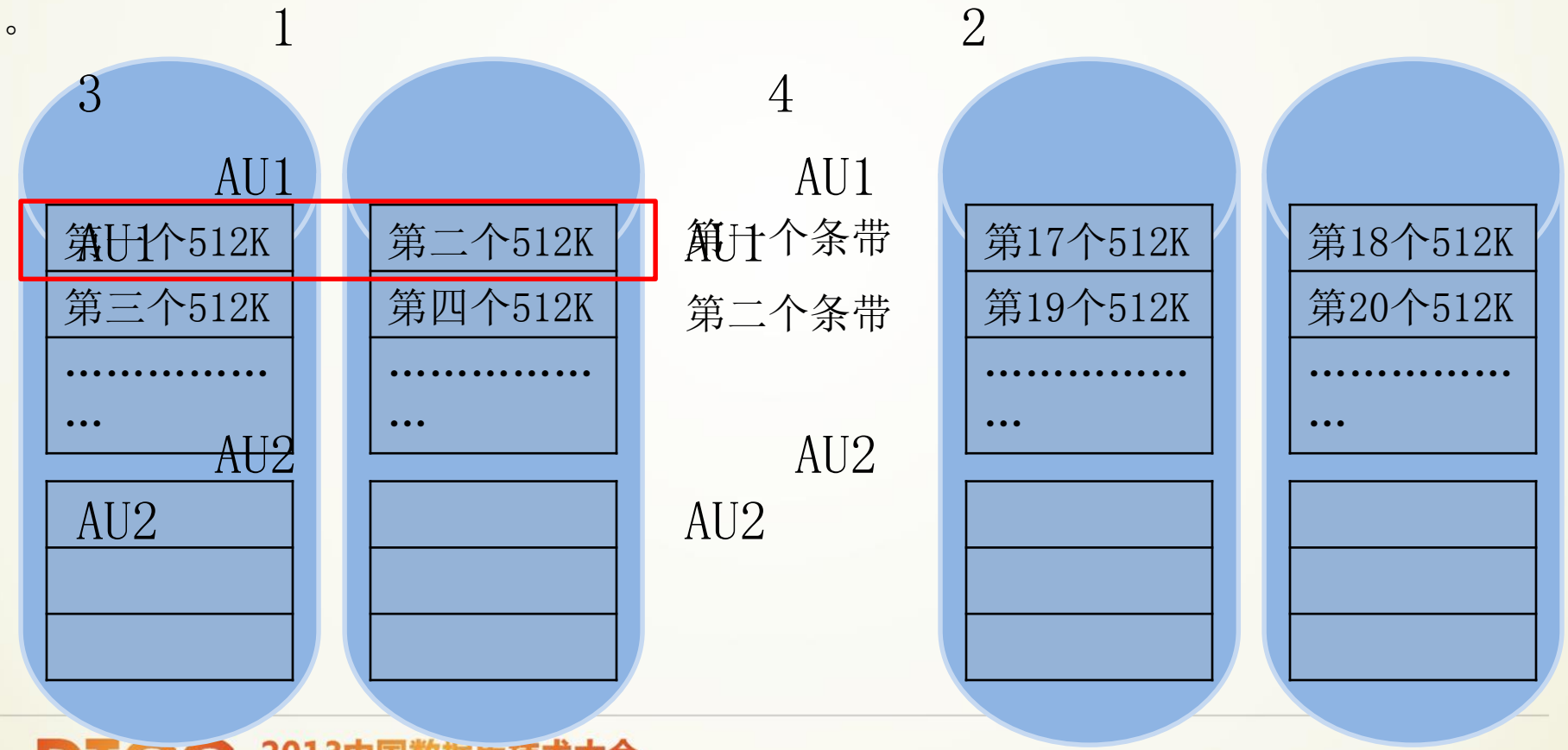
和粗粒度不可调DG相比，区别是数据以512K为单位存储。



AU与条带 ---- 条带宽度

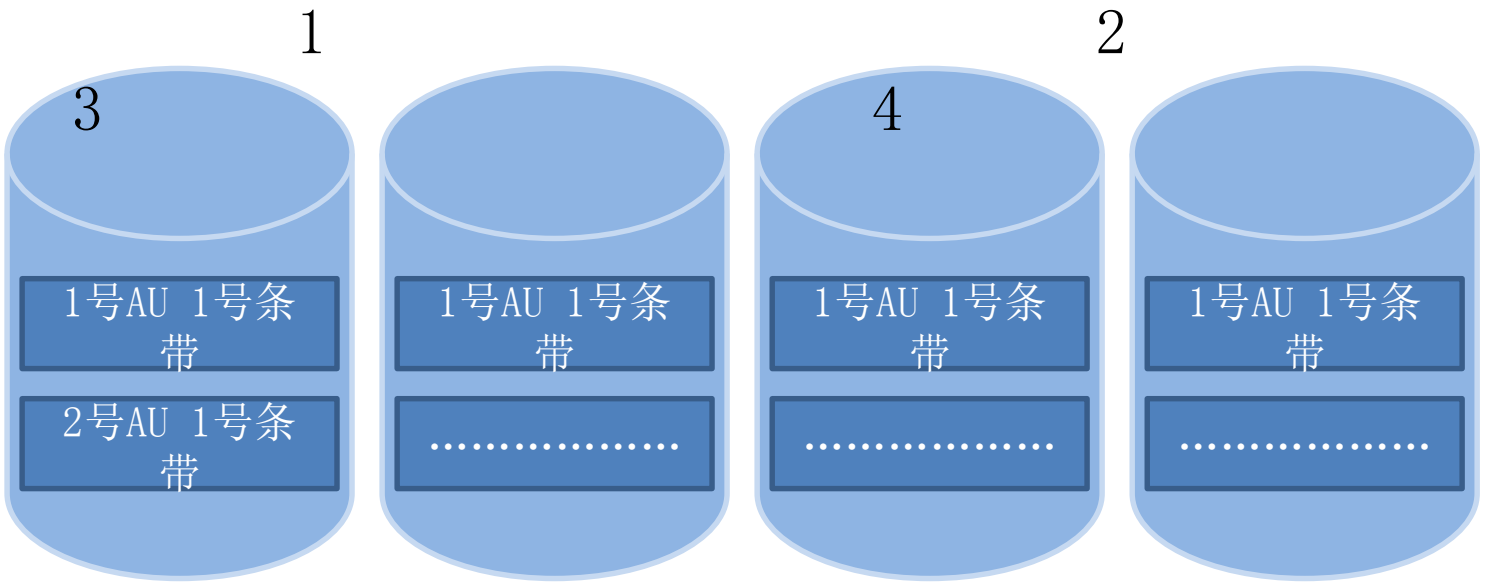
条带宽度必需等于磁盘数吗？当然不是。

假设DG中共有4块磁盘，AU大小为4M，条带大小为512K，条带宽度为2。数据的第一个512K，在1号磁盘的AU1中，第二个512K在2号磁盘AU1中，第三个512K又回到1号磁盘AU1中，因为条带宽度为2吗。直到第15、16个512K，分别在1号盘AU1、2号盘AU1中，AU1、AU2被占满了。第17个512K，在3号盘AU1中，第18个512K，在4号盘AU1中，等等，以此类推。



AU与条带 ---不可调组粒度

DG1共有4块盘，如下图，编号分别分1至4。在DG1中创建一个数据文件，文件的第一个4M在1号盘1号AU 1号条带，第二个4M在2号磁盘1号AU 1号条带，等等，第5个4M在1号磁盘2号AU 1号条带，以次类推。每个AU，只有一个条带，不存在有几号AU 2号条带这个说法。



AU与条带 ---- 分布单元

AU与条带，主要决定了分布单元的大小：

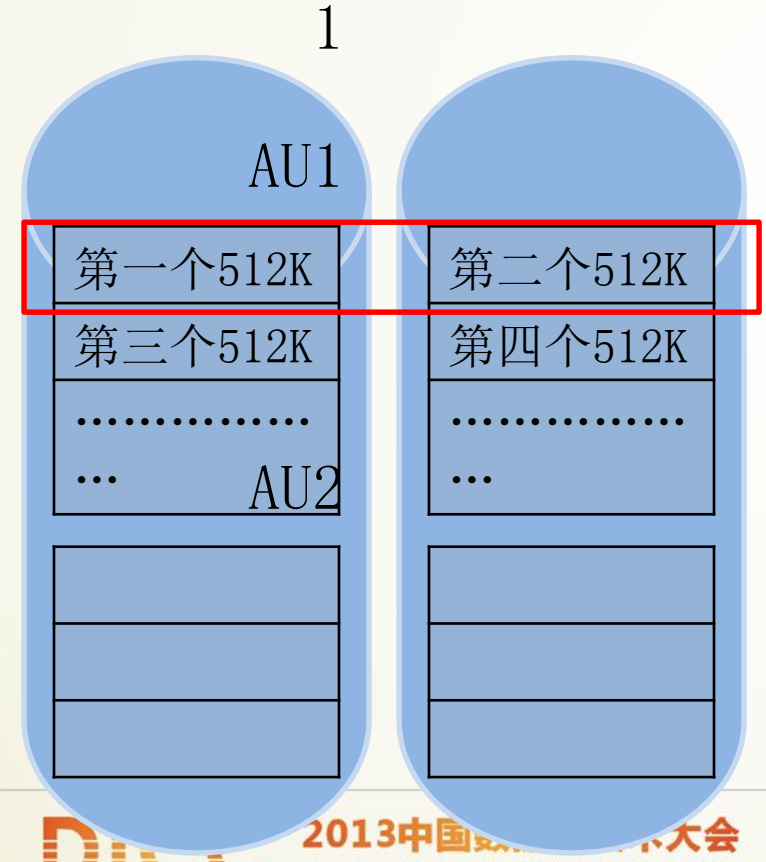
设定数据以多大的单元分布在DG的各个磁盘中，这是AU与条带的目的。前文中的两个例子：

- 4M AU大小4M条带大小，数据以4M为单位，存储在所有磁盘中。
- 4M AU大小512K条带大小，数据以512K为单位存储在所有磁盘中。

AU与条带 ---- 注意事项

➢ 并发还是要靠硬RAID：

需要注意的是，在ASM中无论条带宽为几，数据 库的单次I/O，都不会并发访问多个磁盘。比如下图，如果表有一个1M的区，正好在图中“第一个条带”中，它的前512K在磁盘1，后512K在磁盘2，当全表扫描读这1M数据时，ASM先读磁盘1中的512K，等待512K读完后，再开始读磁盘2中的512K。



想要并发的从两个磁盘读，只能在存储层使用硬RAID才能实现。

ASM层的AU与条带，只是决定数据如何存储。

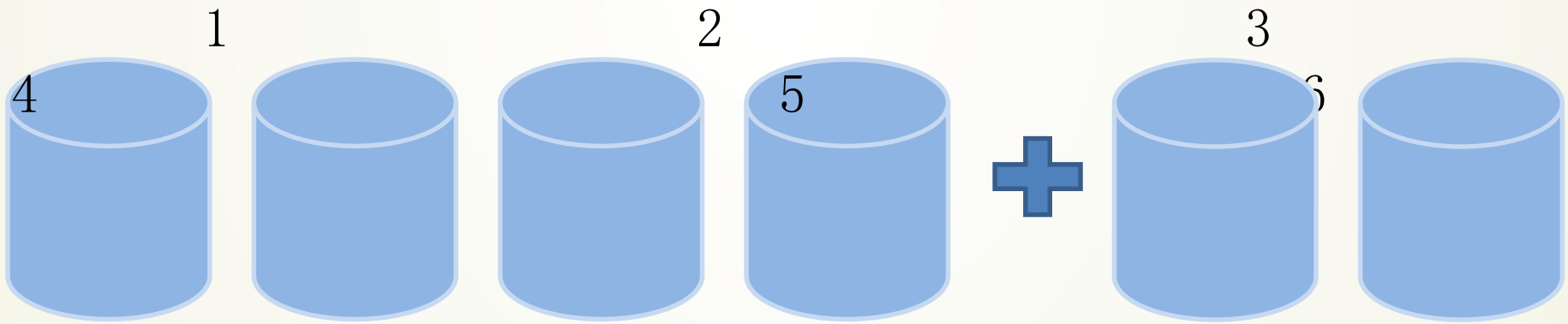
因此，想测试ASM的IO能力，如果不使用并行，是没有意义的。

AU与条带 ---- 注意事项

- **并发还是要靠硬RAID：**

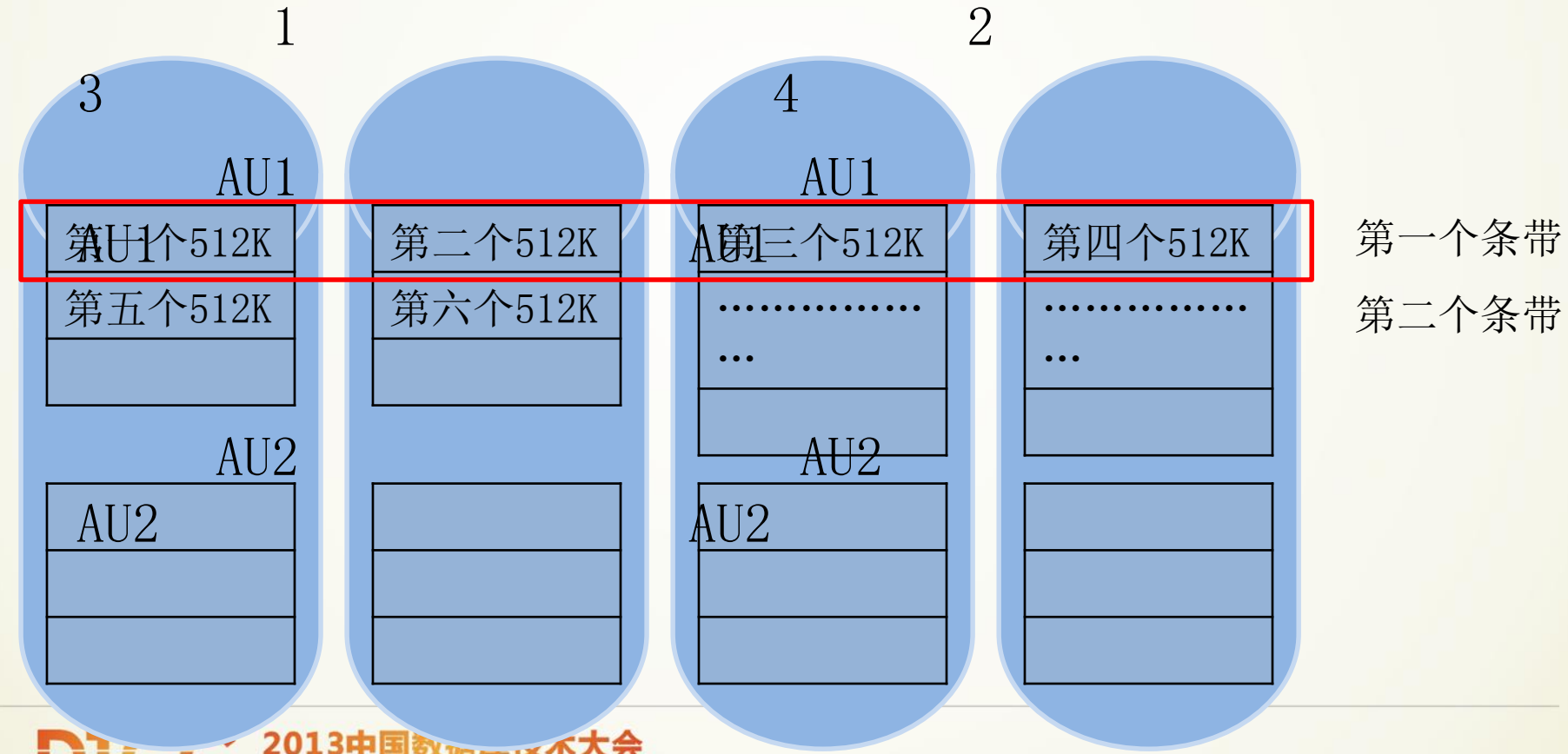
由于ASM层对于全表扫描这样的操作，不能并行读多个磁盘。因此向DG中添加磁盘，并不能提高全扫的性能。

一个4块盘的DG和一个40块盘的DG，全表扫描的性能相差无几。但是，一旦打开并行，就不一样了。



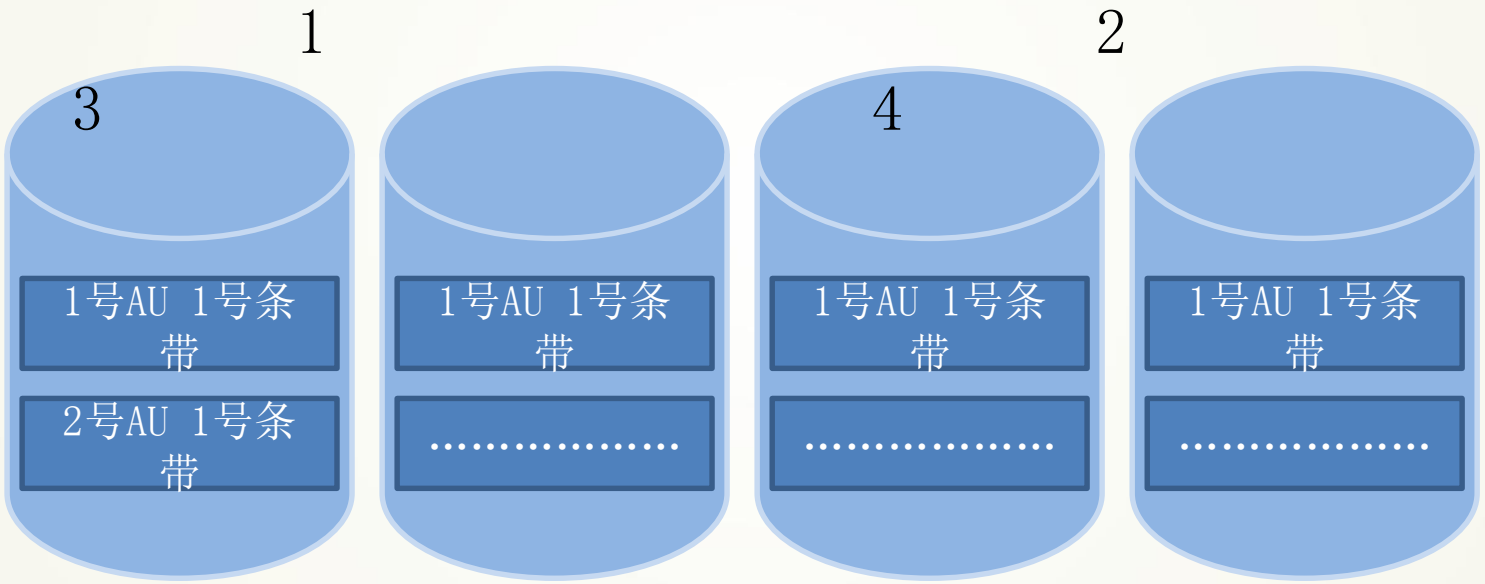
IO大小 ---- 最大IO大小

Oracle最大IO取决于连续的IO有多大，当然还有OS、和硬件的限制。比如下图中的条带下，条带大小512K，数据被以512K为单位，切割到多块磁盘。如果区大小为1M，这1M对数据库来说连续的数据，在ASM中被分别存储在两块磁盘中。此时，Oracle的最大IO大小，不会超过512K。



IO大小 ---- 最大IO大小

假设下图中AU大小、条带大小为4M（也就是不可调粗粒度条带），但OS、硬件限制最大IO为1M，Oracle的最大IO也无法超过1M。



IO大小 ---- 最小IO大小

AU、条带，包括数据库中的Extent，都只决定数据如何存储，与最小IO无关。Oracle中的最小IO大小，取决于块大小。

数据文件，最小IO是8K。

Redo文件，最小IO是512字节。

ASM元数据文件块大小是4096字节，哪 么，这些文件的最小IO就是4K。

AU、条带大小，并不影响最小IO大小。

注意：条带，或AU，并不是ASM的最小IO。

IO性能影响 ---- 数据分布

AU、条带决定了数据在存储介质中的分布，而数据的分布则可能会影响到IO性能。

想像一下，如果存储中有多个RAID组，但频繁访问的数据只被存放在某个RAID组的LUN中，这会造成什么情况。

IO压力不均衡，无法发挥存储的能力，还有可能导致缓慢的IO、影响整体性能。

就好像一个小组有4个人，每个人的工作量最好相当。如果一个人工作量多、另三个人工作量少。另三个人干完活，等着哪一个工作量最多的人。这样，总体的速度就被拖慢了。

数据平均分散到DG的多个盘中，这样压力才能均衡。

通过AU和条带，我们的目的就是要让数据合理分散。

IO性能影响 ---- 数据分布

- **数据分散的单位：**

数据平均分散，总要有个单位，不能按字节分散。AU和条带，就是用来设置这个单位的。

如果AU、条带大小都是4M，数据以4M为单位，分散到DG的所有盘中。如果AU是4M，条带是128K，数据以128K为单位，分散到DG的所有盘中。

IO性能影响 ---- 大小AU、条带总结

- **小条带：**

  优点：数据更分散，有助于分散热点。
  缺点：过小的AU、条带，连续数据不大，OLAP性能受影响很大。

- **大AU、大条带：**

  优点：数据连续存储，显著提高OLAP类操作性能。
  缺点：过大的AU、条带，可导致数据分散度不够高。

OLTP 环境IO规化案例 --- 环境介绍

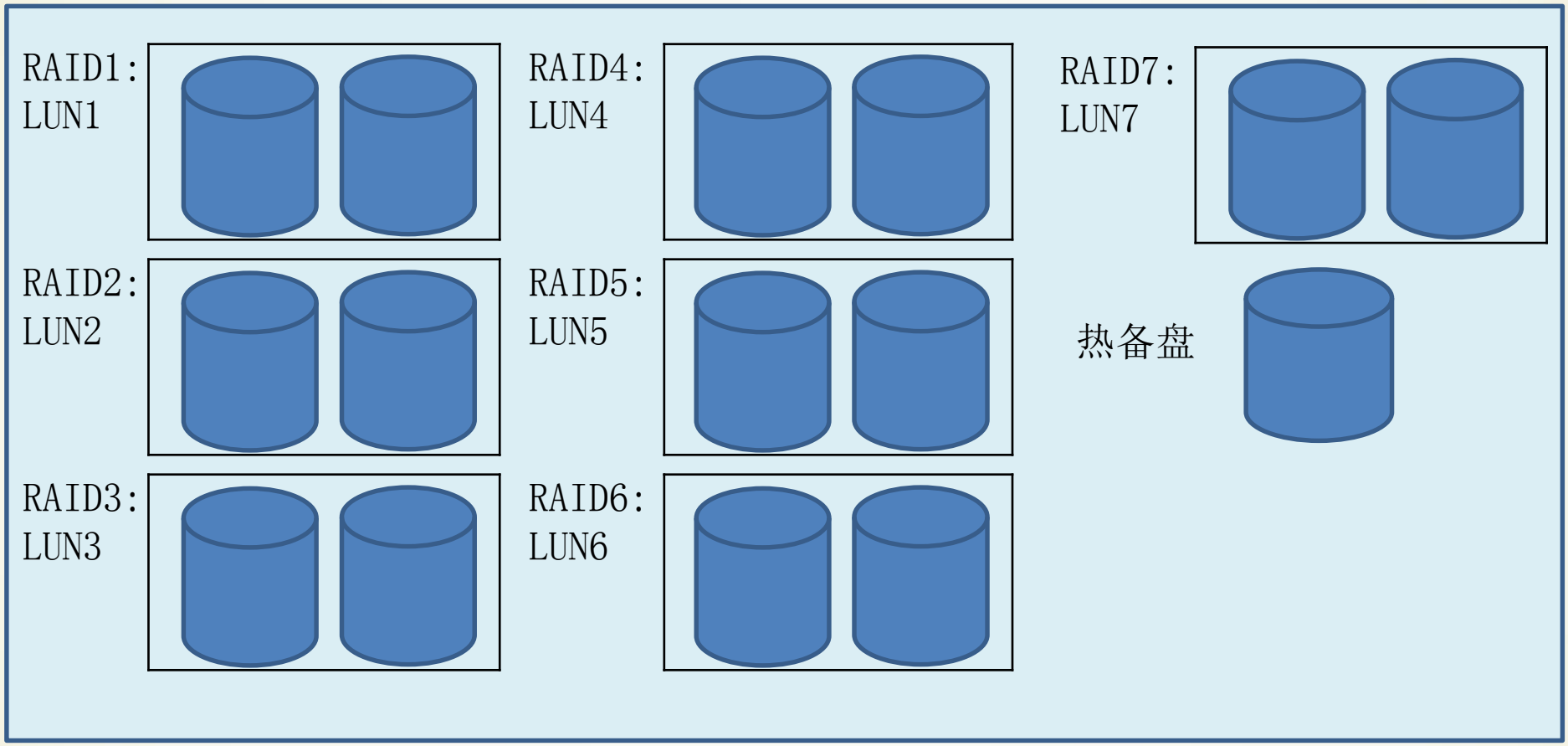
- **应用情况：**

- 日志型应用，特点大并发插入。和其他类型的应用。
- 因为有些OLAP型操作，考虑到全表扫描操作的性能，AU大小定为4M，另大部分表空间Extent（区）大小都为1M。
- 存储是盘柜，共15块盘，1块做热备，另14块做成7对RAID1。
- 每块盘600G，每个镜像的RAID1也是600G。
- 每组RAID被做为一个LUN，共7个600G的LUN。

- **测试并发插入，当每秒2000个插入TPS时，遇到的问题：**

- IO响应时间时快时慢。
- Db file parallel write 有时比较高。

OLTP 环境IO规化案例--存储粘朴简图

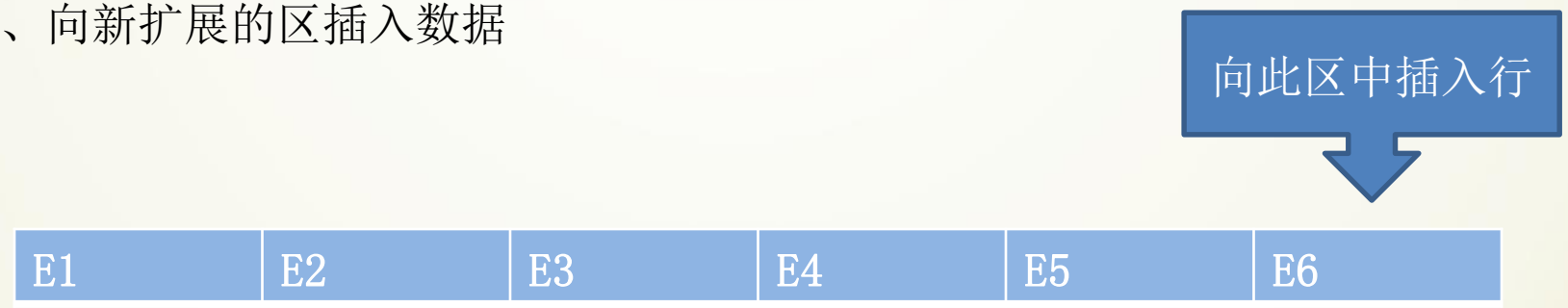


OLTP 环境IO规化案例----插入规则

1、向最后一个区插入数据

2、插满后扩展（高水点的扩展）

3、向新扩展的区插入数据



OLTP 环境IO规化案例----区

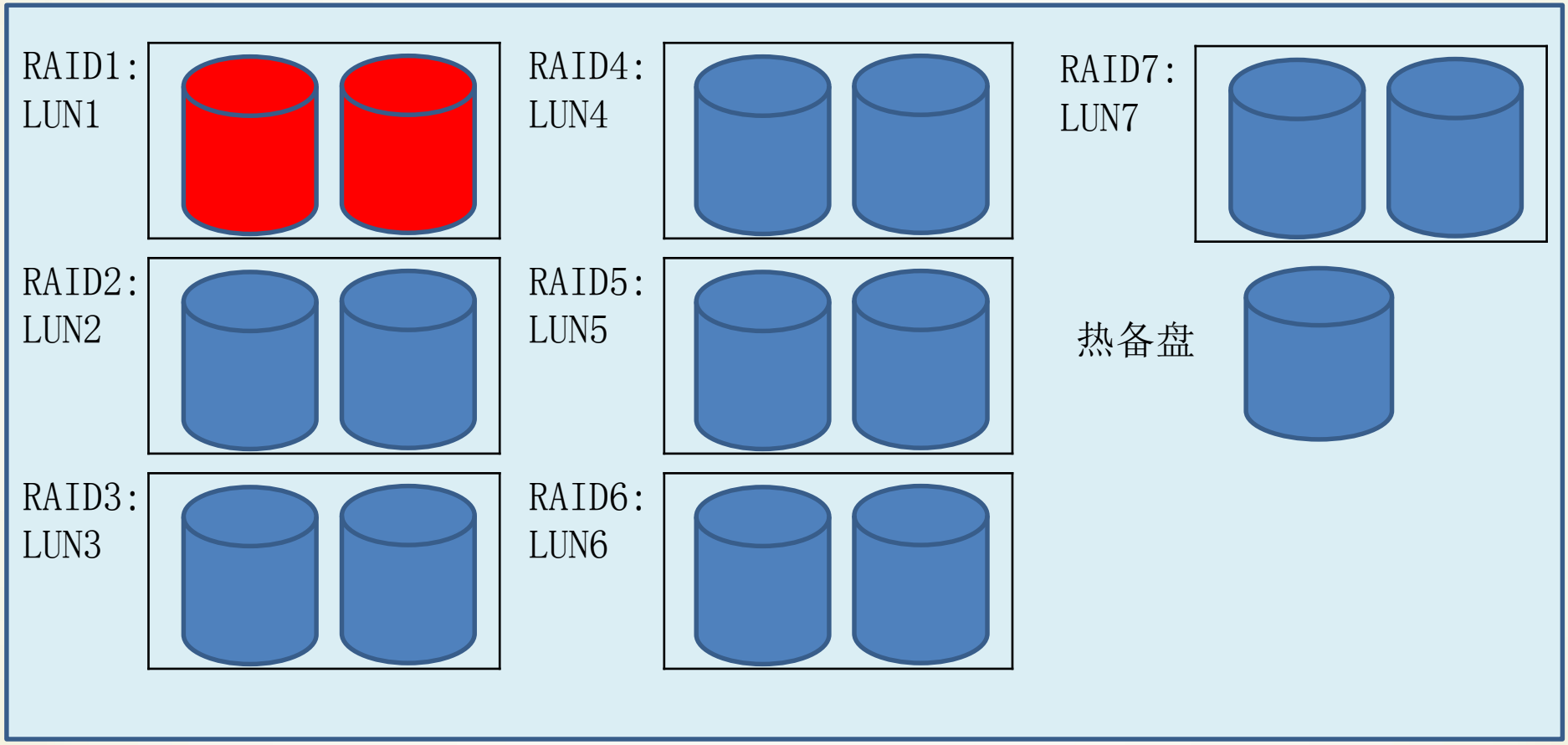
Extent大小都是1M，AU大小是4M，前4个区在第一个AU中。第一个AU在RAID1 的LUN1中。数据先被插入到LUN1，大量并发插入，使RAID1中迅速有了很多脏块，DBWR会向LUN1中写大量脏块。

第一个AU中的所有区被写满后，开始向第二个AU中的区写数据。第二个AU在RAID2的LUN2中，这又使RAID2中马上产生很多脏块，DBWR向RAID2中写脏块。

接着，以此类推，DBWR向RAID3写，向RAID4写，如下组图：

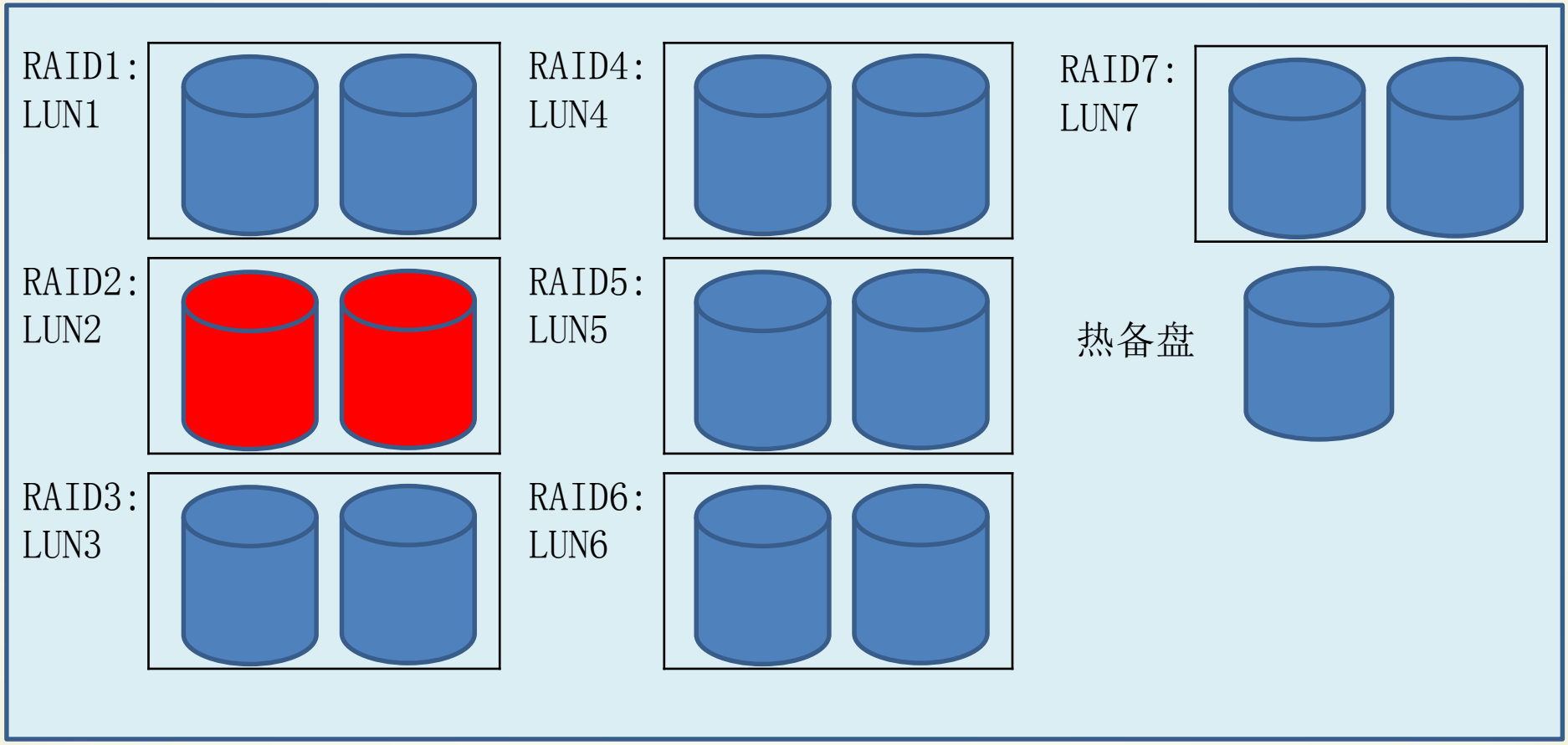
OLTP 环境IO规化案例---DBWR写

DBWR向RAID1写脏块：



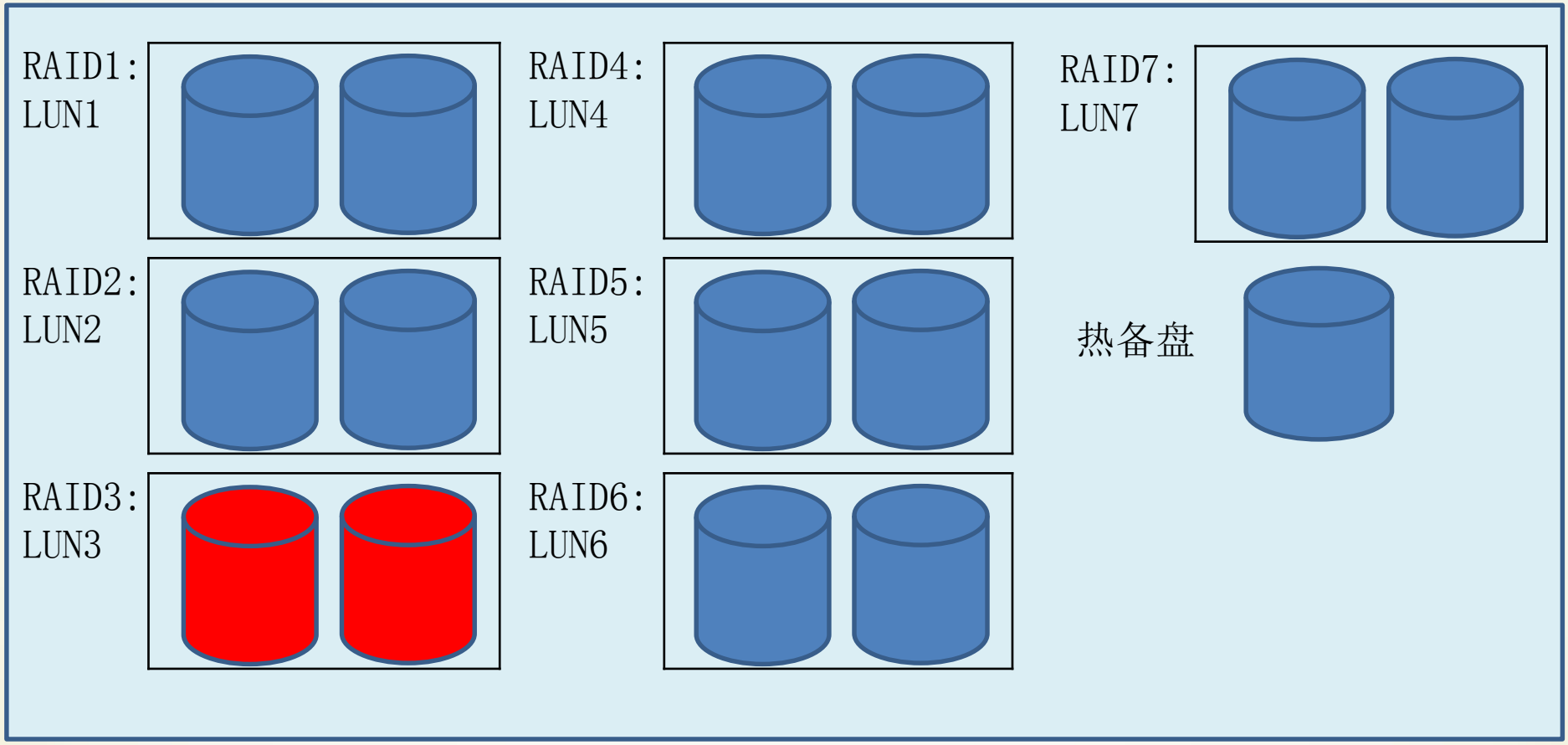
OLTP 环境IO规化案例---DBWR写

DBWR向RAID2写脏块：



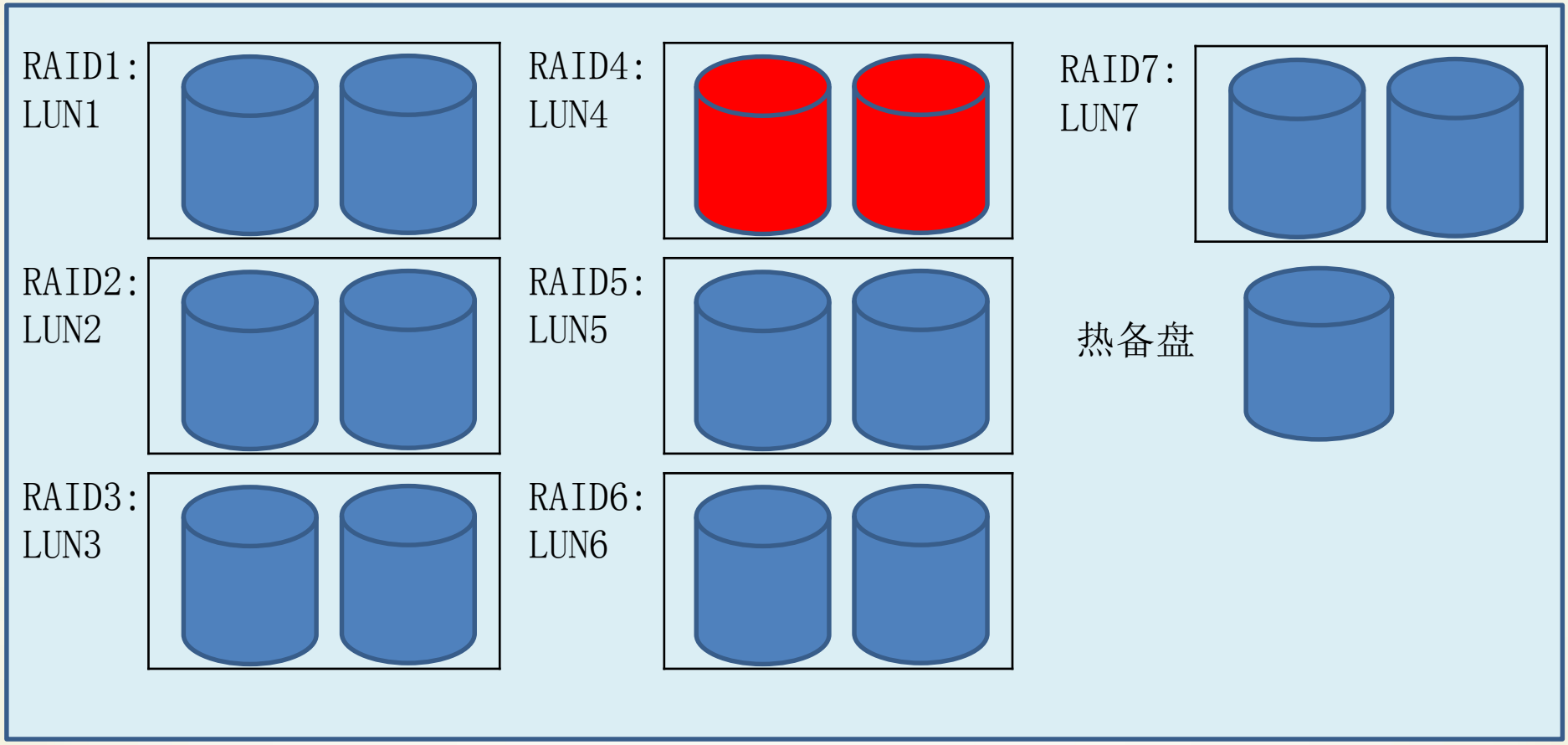
OLTP 环境IO规化案例---DBWR写

DBWR向RAID3写脏块：



OLTP 环境IO规化案例---DBWR写

DBWR向RAID4写脏块，……，等等。这就是热点了，而且是会变的热点。在系统压力比较高时，前几分钟是某些磁盘忙，后几分钟又是另外的磁盘忙。



当和其他消耗IO的应用混在一起时，会导致IO响应时间波动很大，时高时低、时快时慢。

OLTP 环境IO规化案例---解决方案

解决方案：条带：

AU大小原来是4M，数据以4M为单位分布。使用默认的细粒度条带（大小128K，宽度为8），不必修改隐藏参数。让数据以128K为单位分布。命令如下：

1、在ASM实例中为DG添加细粒度模板：

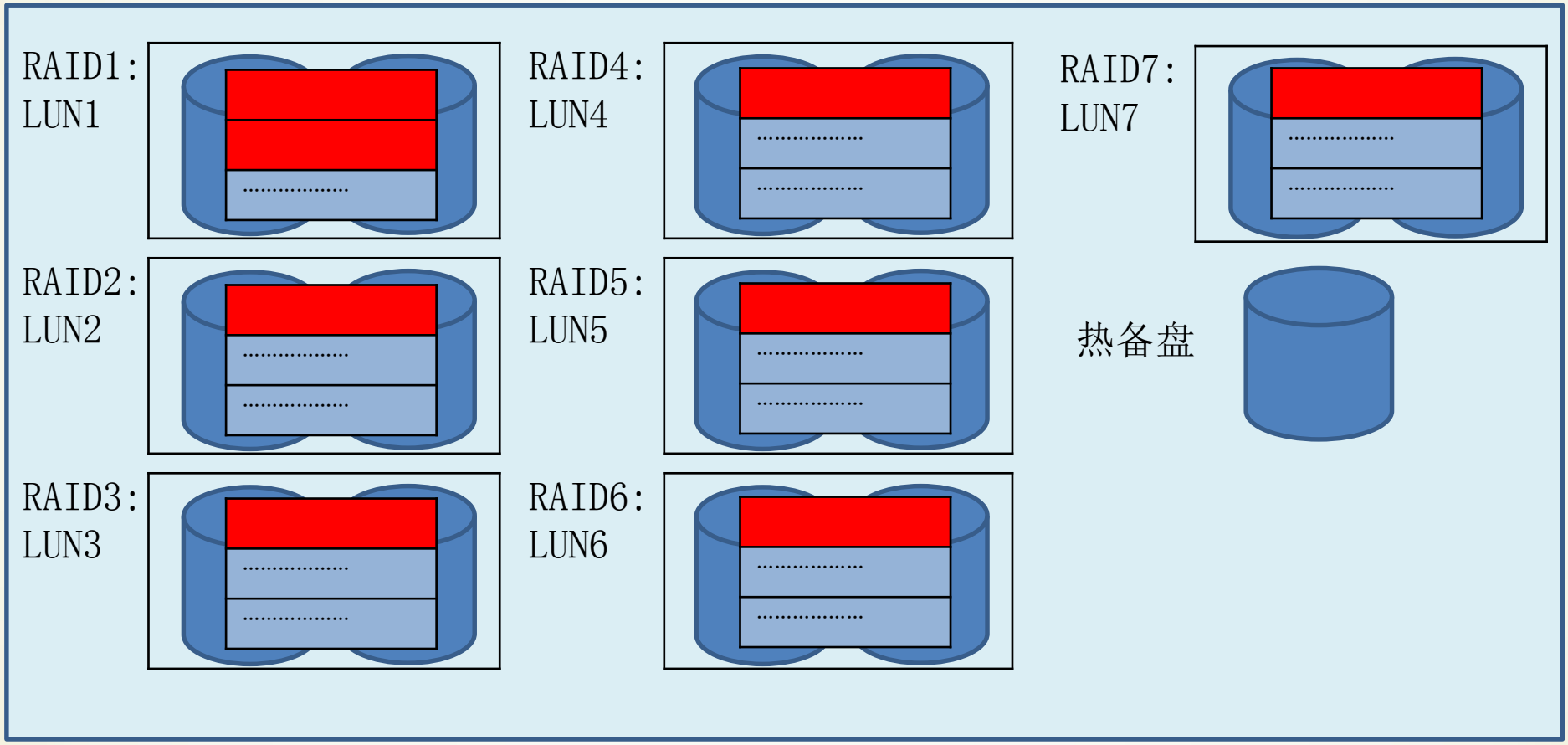
```
alter diskgroup dg1 ADD TEMPLATE stp_fine ATTRIBUTES (UNPROTECTED fine);
```

2、在数据库实例中创建使用细粒度的表空间

```
create tablespace tbs_data01 datafile
‘+dg1(stp_fine)/tbs_data01_01.dbf’  size 10240m reuse uniform size 1m;
```

OLTP 环境IO规化案例---解决方案

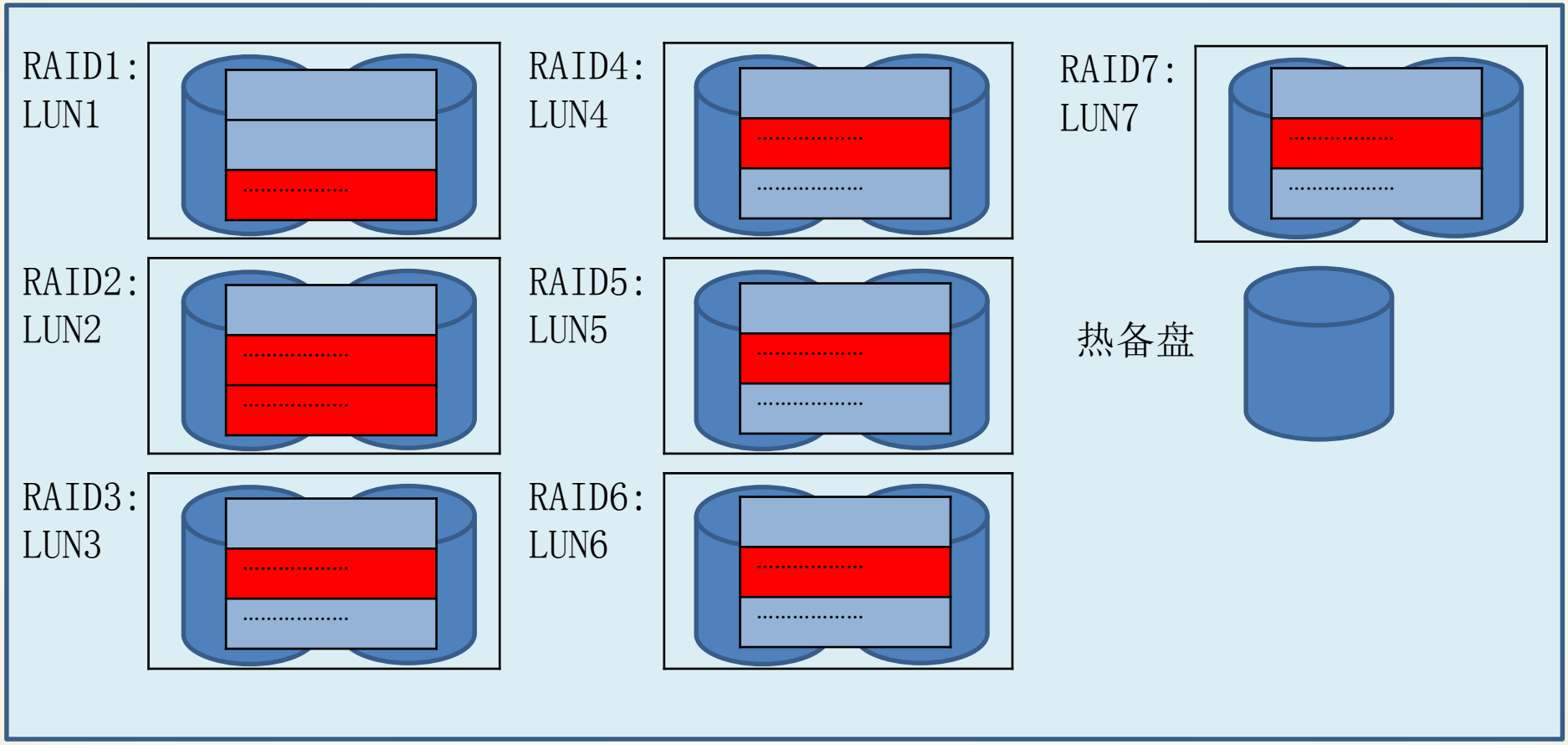
条带大小128K，1M的区，每1M数据被会以128K为单位，分散到8个盘的8个AU中。新插入的1M数据，会分别向8个盘插入。



在存储层，脏块分布在7个RAID组中。DBWR写脏块时，同时向7个RAID组写。

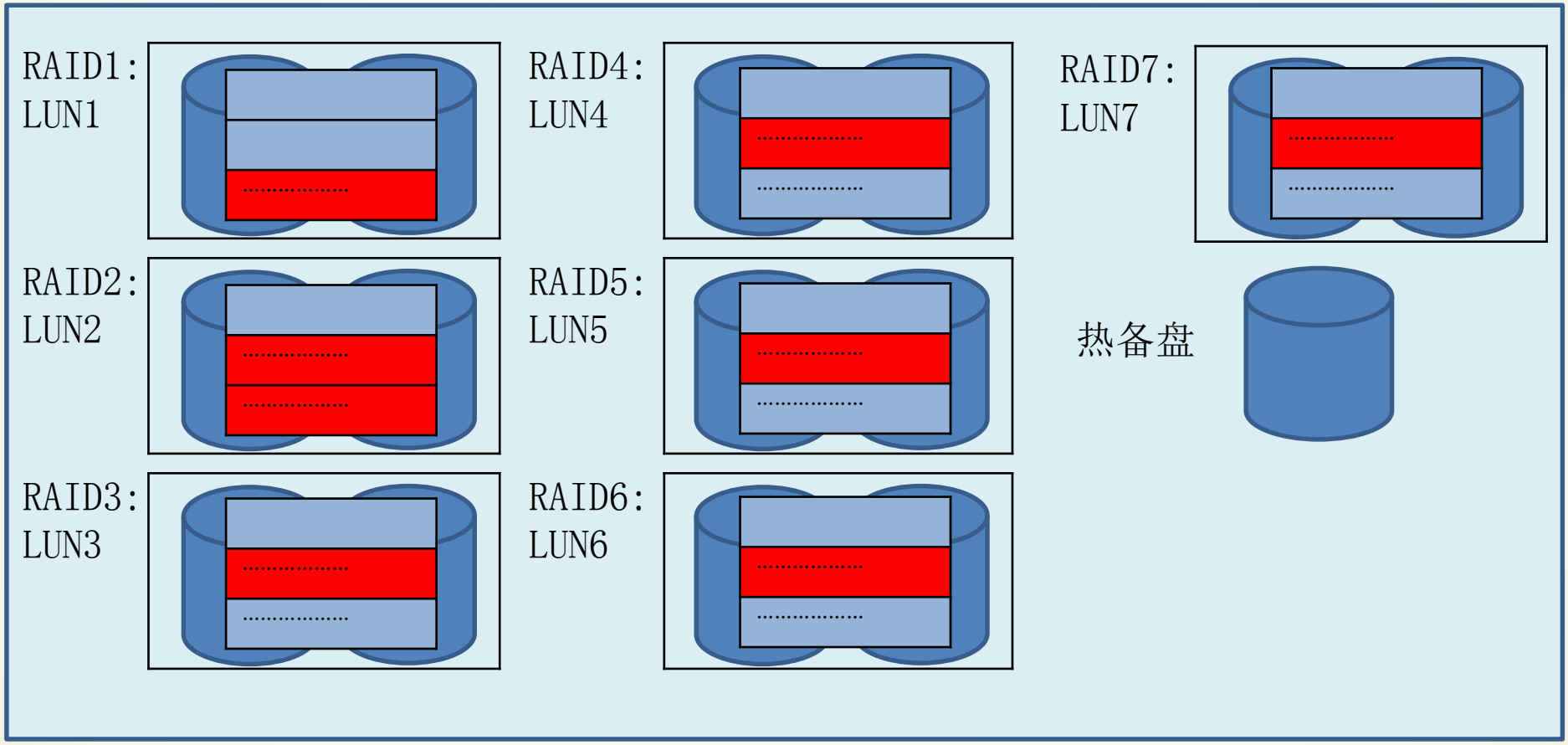
OLTP 环境IO规化案例---解决方案

一个区插满，再插另一个区。



OLTP 环境IO规化案例---解决方案

细粒度可调条带，Oracle默认的条带宽度为8。但此案例中盘柜只有7组RAID，因此，对于一个1M的区，会有一个RAID组中有两个128K。



OLTP 环境IO规化案例---总结：散

这样是不太完美，但1M数据总算分布到多个RAID中了，压力更加分散，没有了IO波动。db file parallel write、db file sequential read等IO类等待事件的等待时间更加平稳，没有了IO响应时间时快时慢的情况。

当然，如果能有8个RAID组、ASM DG中包含8个盘，这样是最好的。1M的区，128K条带，正好每个RAID组（或每个ASM盘）中一个128K的条带。当然还可以在ASM实例中调节隐藏参数，改变条带宽度。但这样做有一定风险，不建议在生产环境使用。

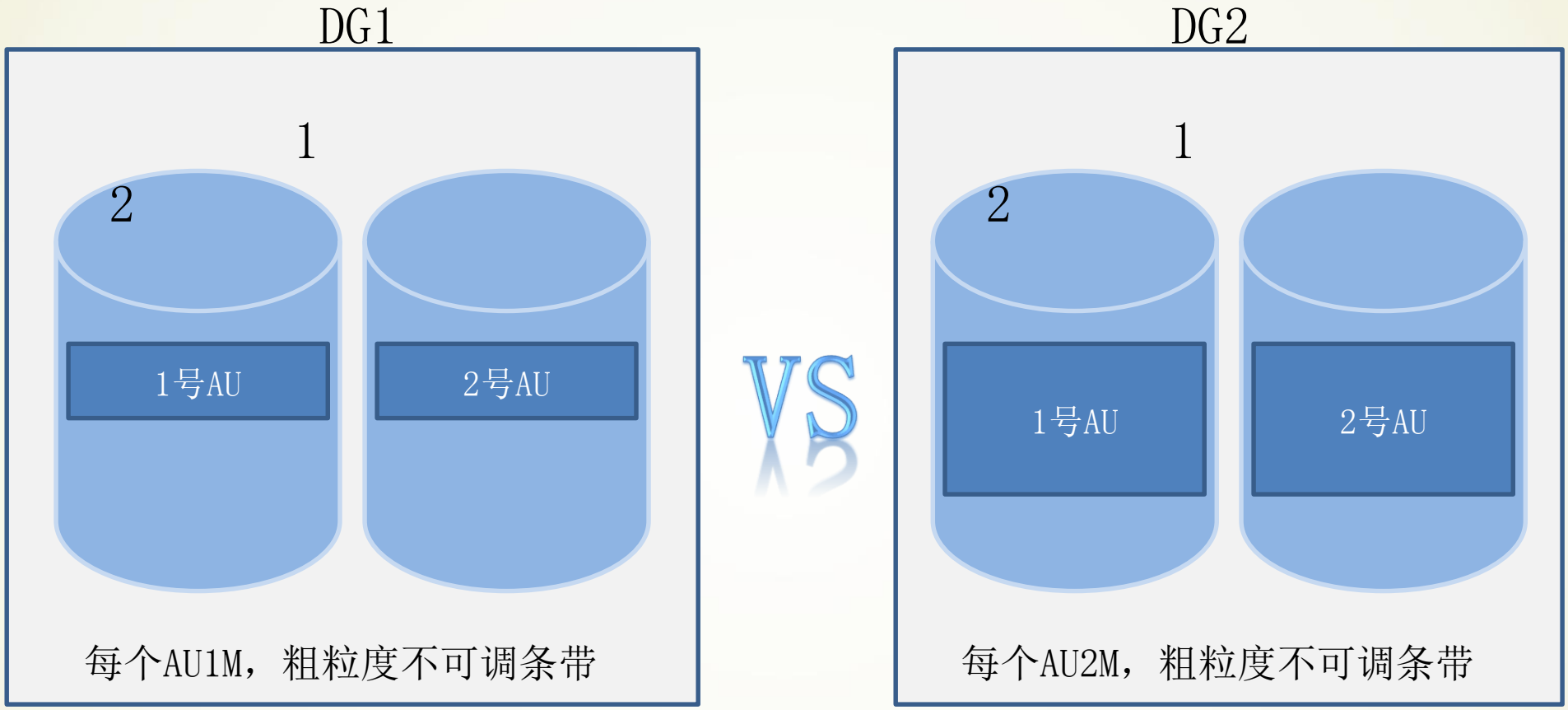
OLAP 环境IO规化注意事项 --- 连

- **OLAP的关键：连**

不同于OLTP，OLAP环境下，全扫描操作较多，这时，“连”是关键。连续IO越多，IO性能越佳。

为了让数据尽可能“连”续存储，要尽量使用大AU。

IO性能影响 ----AU大小对OLAP的影响



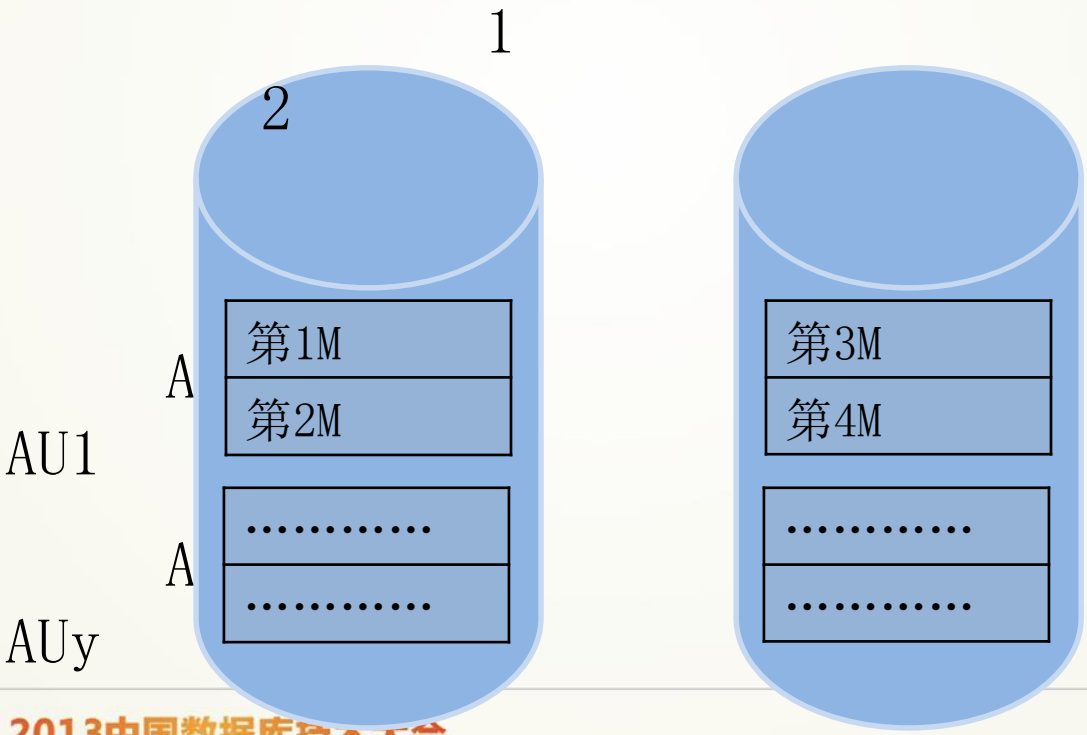
IO性能影响 ----AU大小对OLAP的影响

表大小120M，全表扫描时间对比：
1、1M AU 1M的区大小：
Elapsed: 00:00:03.58
Elapsed: 00:00:03.50
Elapsed: 00:00:03.49
Elapsed: 00:00:03.48

2、2M AU 1M区：
Elapsed: 00:00:03.02
Elapsed: 00:00:03.00
Elapsed: 00:00:03.04
Elapsed: 00:00:03.03

IO性能影响 ----AU大小对OLAP的影响

即使超过了OS、硬件对IO大小的限制，为什么大AU仍然会快。原因很简单，比如OS、硬件限制单次最大IO为1M，AU大小为2M。由于用满一个AU，才会用下一个AU。如下图，第1M和第2M数据是连续存储的。进程虽需分两次IO，才能完成读这两M数据，但两次IO是连续IO，性能肯定会比离散的快。



灵活的条带设置

AU是针对DG的。条带是针对文件的。

一个DG，只能有一种大小的AU，但可以有多个条带大小不一的文件。

如下的创建命令使用细粒度可调条带：

```
create tablespace tbs_data01 datafile
‘+dg1(stp_fine2)/tbs_data01_01.dbf’  size 100m reuse uniform size 4m;
```

如下的创建命令使用粗粒度不可调条带：

```
create tablespace tbs_data02 datafile  ‘+dg1/tbs_data01_01.dbf’ size 100m reuse uniform size 4m;
```

两个表空间，都在DG1中，一个使用细粒度可调条带，一个使用粗粒度不可调条带。

这使我们的配置很灵活，如果某个表全表扫描比较多，可以将它放在粗粒度条带表空间中。如果像前文所述并发插入比较多，可以将它放在细粒度条带表空间中。

ASM IO规化与架构总结

➢OLAP：

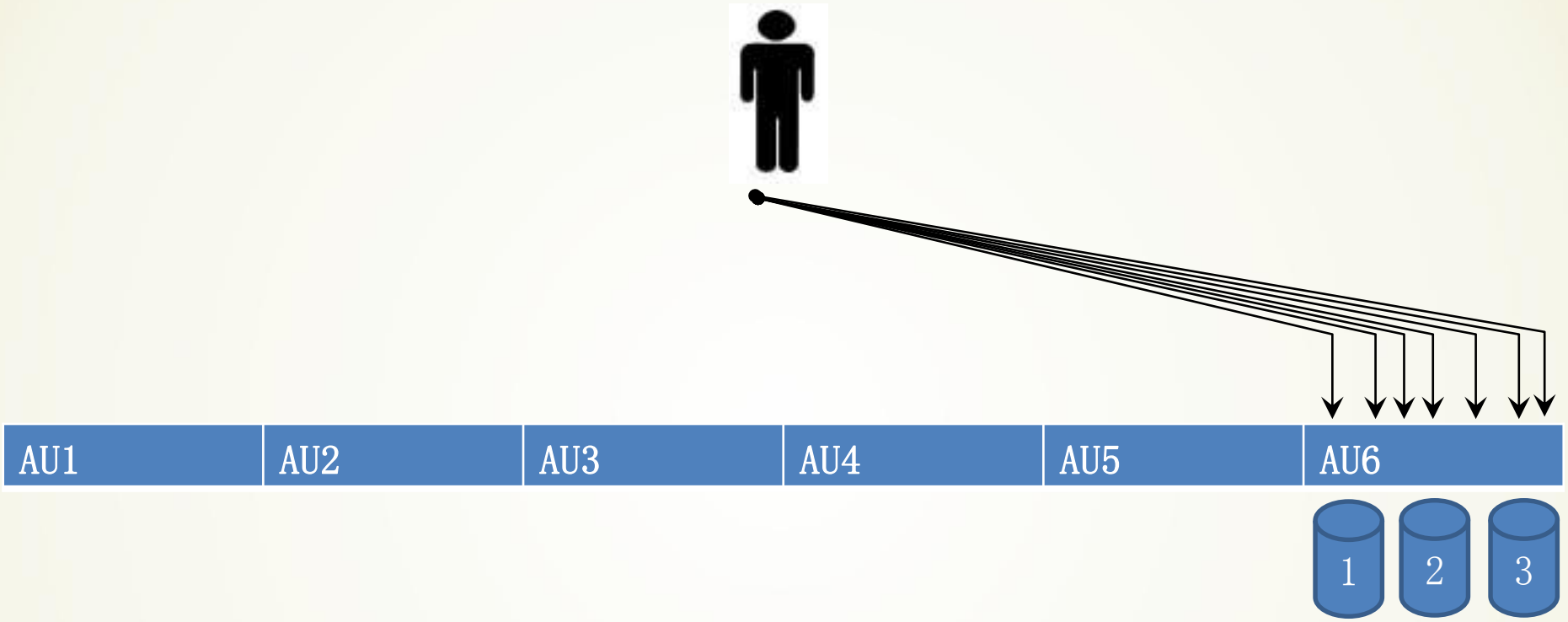
不用考虑条带，用大AU，4M或8M为宜，甚至可以考虑更大。大AU会浪费些空间。如果AU大小是8M，不可能所有文件大小都是8M的倍数。如果某个文件大小是10M，它将占两个AU，大小就是16M。这会导致空间有一点点浪费。这是我能想到的大AU的唯一缺点。

➢OLTP：

根据应用情况选择不同方案。比如前文提到的并发插入的例子，可以使用条带进一步分隔数据。

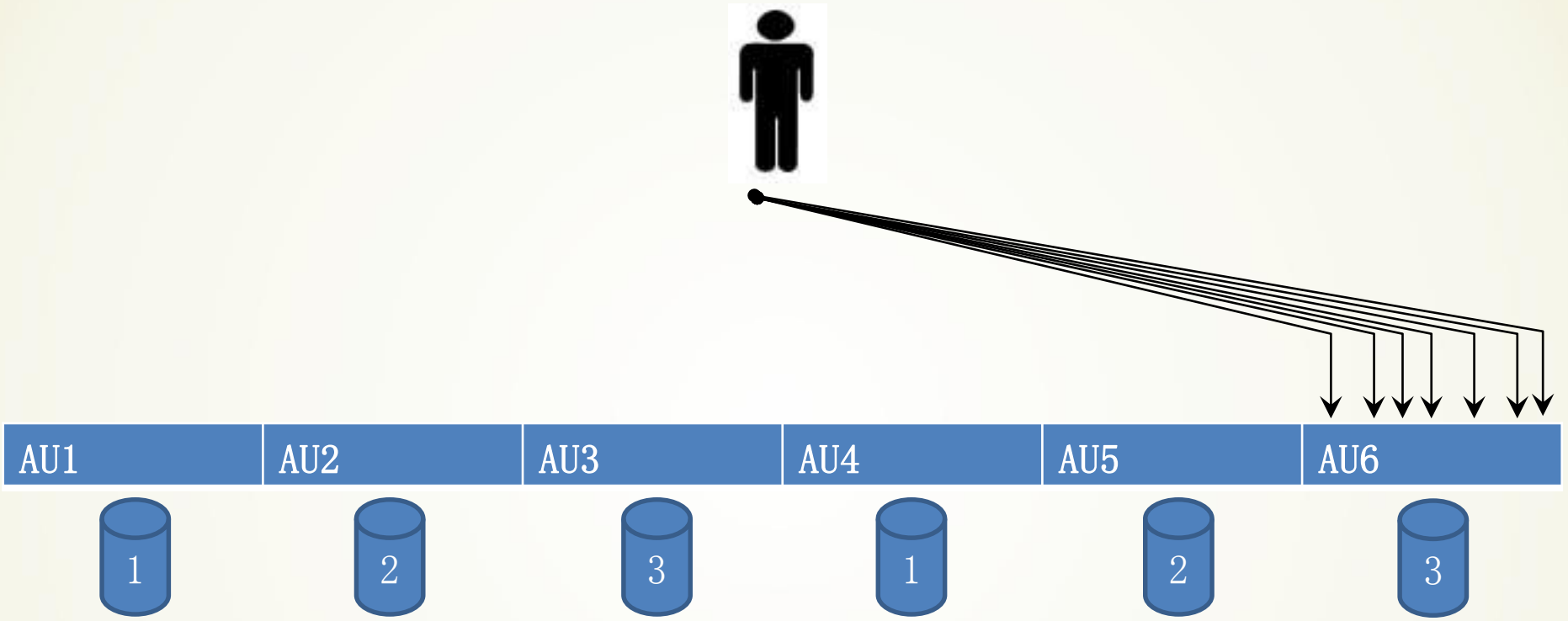
如果不是并发插入，而是随机的并发访问，大AU并不会减慢性能。对比下页图，可以看到此点。

ASM IO规化与架构总结



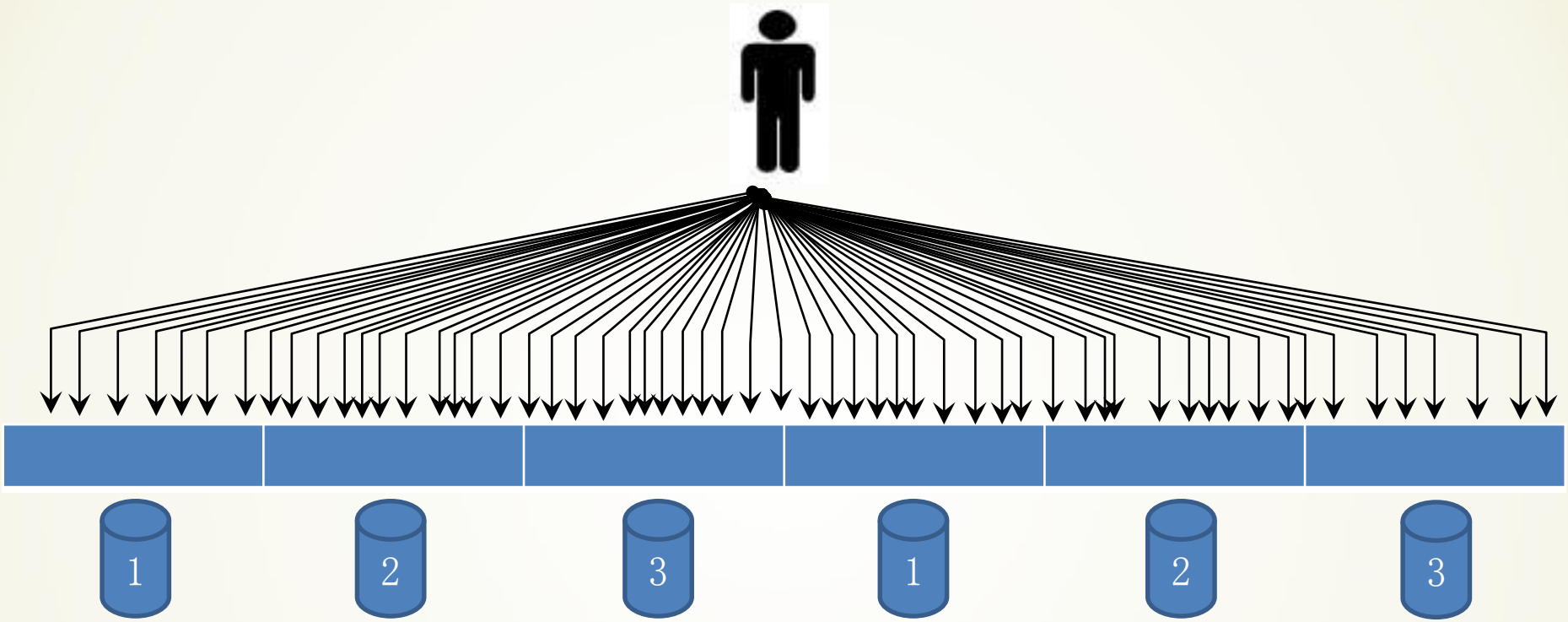
先回顾一下前文所讲的插入的例子，每次都只向一个区中插入，因此最好让一个区的空间对应多块磁盘。

ASM IO规化与架构总结



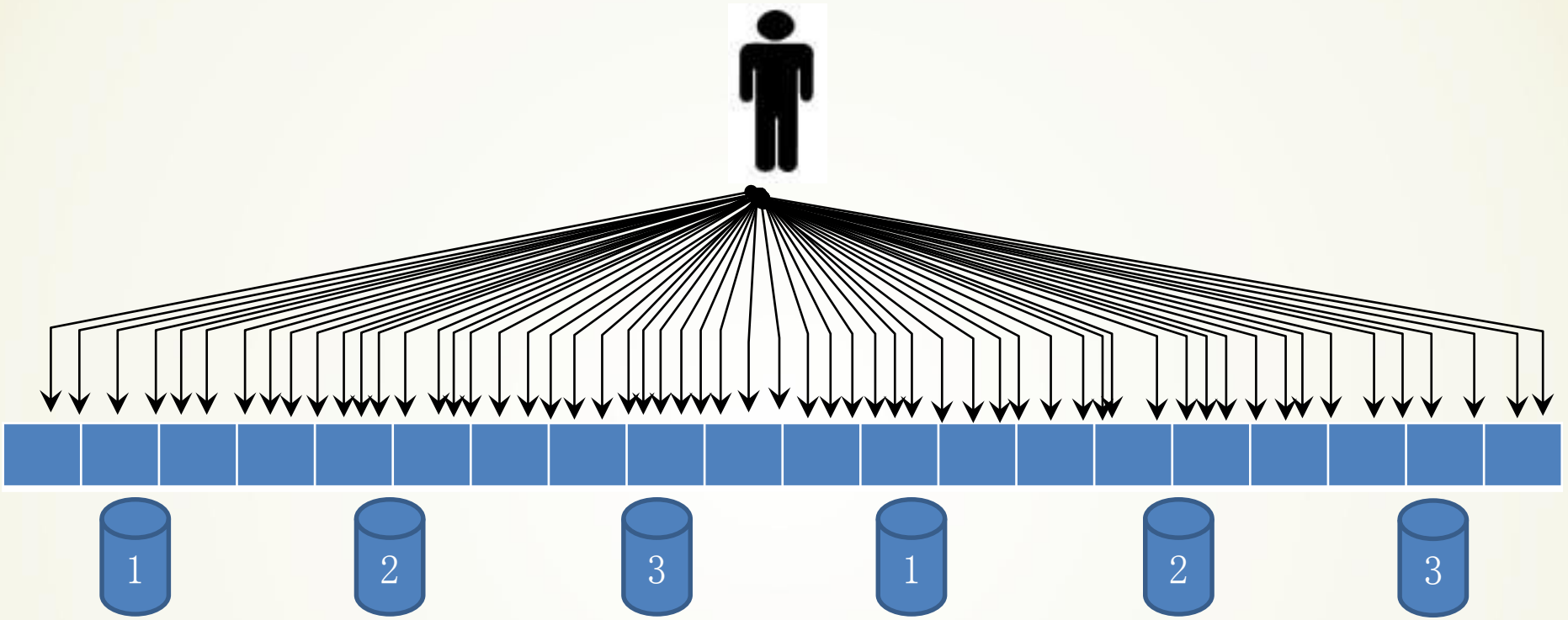
如果一个区的数据，只分布在一块磁盘中，会造成热点。

ASM IO规化与架构总结



如果不是并发插入，而是随机的并发访问，如上图，表的每个区只对应一块盘，但所有区每时每刻都在有进程访问，所有磁盘IO压力大致相似，没有热点。没有别要让一个区分布到多块磁盘上了。

ASM IO规化与架构总结



这种均匀的随机IO情况下，就算用小AU、小条带，让数据在各个磁盘上分布的更碎，对IO性能提升没有帮助。由于数据更碎，对连续、大IO访问，性能反而会下降。

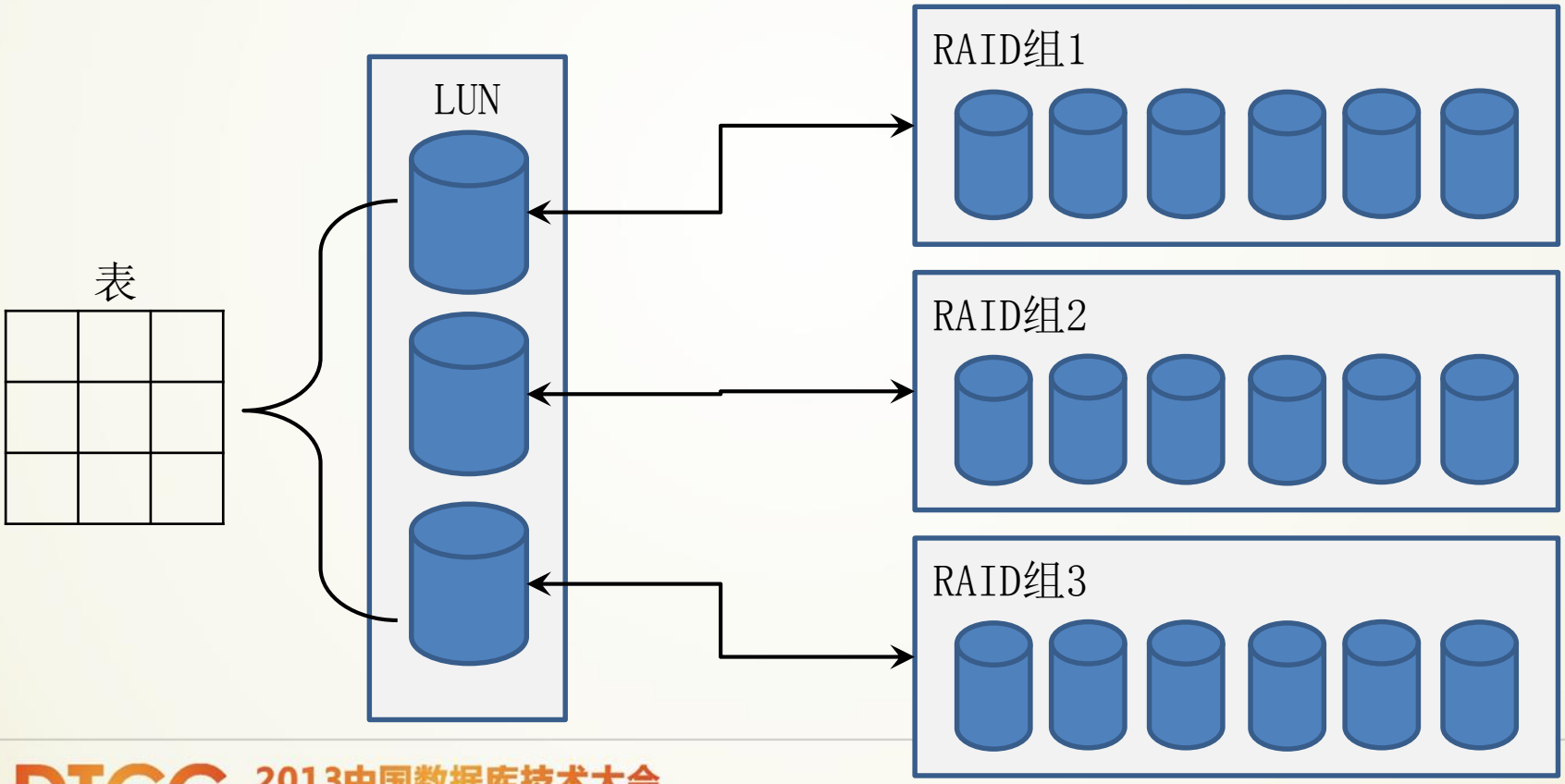
ASM IO规化与架构总结

- OLTP下，非大并发插入，大AU无条带更适合

总结，IO规化要结合应用，对于非大并发插入的应用，无论OLTP、OLAP，大AU无条带（或默认条带设置），对性能更有帮助。

ASM IO规化与架构总结

大并发插入的话，如果主机看到的盘（LUN），在存储层已经对应多块物理磁盘，可以不使用条带。数据依次向RAID组1、RAID组2、RAID组3插入。每个RAID组已经对应多块物理磁盘，IO已经分散。当然，如果想让IO更加分散，可以考虑使用条带，让一个AU跨多个RAID组。但更碎的数据分布，会影响大IO的性能，需要DBA根据情况进行取舍。



欢迎讨论

本文所有结论，不是来自Oracle内部资料，而是Dtrace、gdb/mdb跟踪分析得到的结果。欢迎更多对原理有兴趣的朋友，一起研究讨论。

可在ITPUB上搜索用户名：VAGE，查看我的相关文章。
BLOG：www.mythdata.com
微博：@文本时代_VAGE

Oracle证书+技能培训，可以关注杭州博学：www.boxue.com

欢迎莅临

2013中国数据库技术大会