揭开支撑Oracle运行的神器Exadata的美丽面纱
原理和技术实现介绍系列（一）

刘建军，资深解决方案工程师

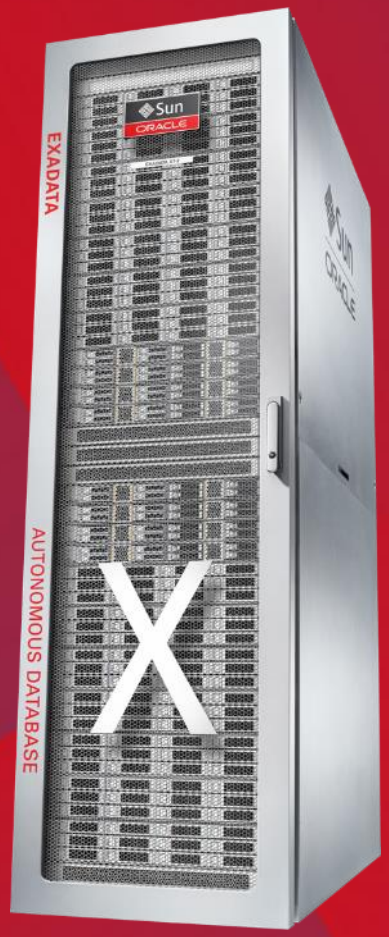
公益讲座11点准时开始，请大家先浏览云技术微信公众号技术文章
资料会在各群同步发布，已入群客户请勿重复入群！

扫码加入:
19c新特性讲座群

欢迎关注:
甲骨文云技术公众号

Exadata公开课（一）

LibCell与Share nothing MPP数据处理



Engineered Systems Solution Engineering Team

ORACLE

目录

- 为什么要研发Exadata与Exadata基本架构介绍
- Exadata IO子系统LibCell简介
- Offload-在RAC上引入Share Nothing MPP处理

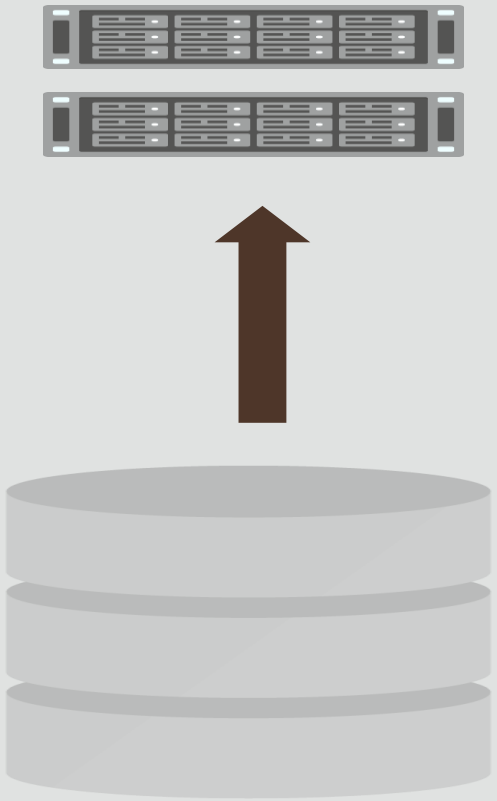
数据库系统的架构
(大约2008年左右)

DB服务器有充足的CPU

低带宽的网络

有大量磁盘的存储阵列

数据库系统的架构
(大约2008年左右)

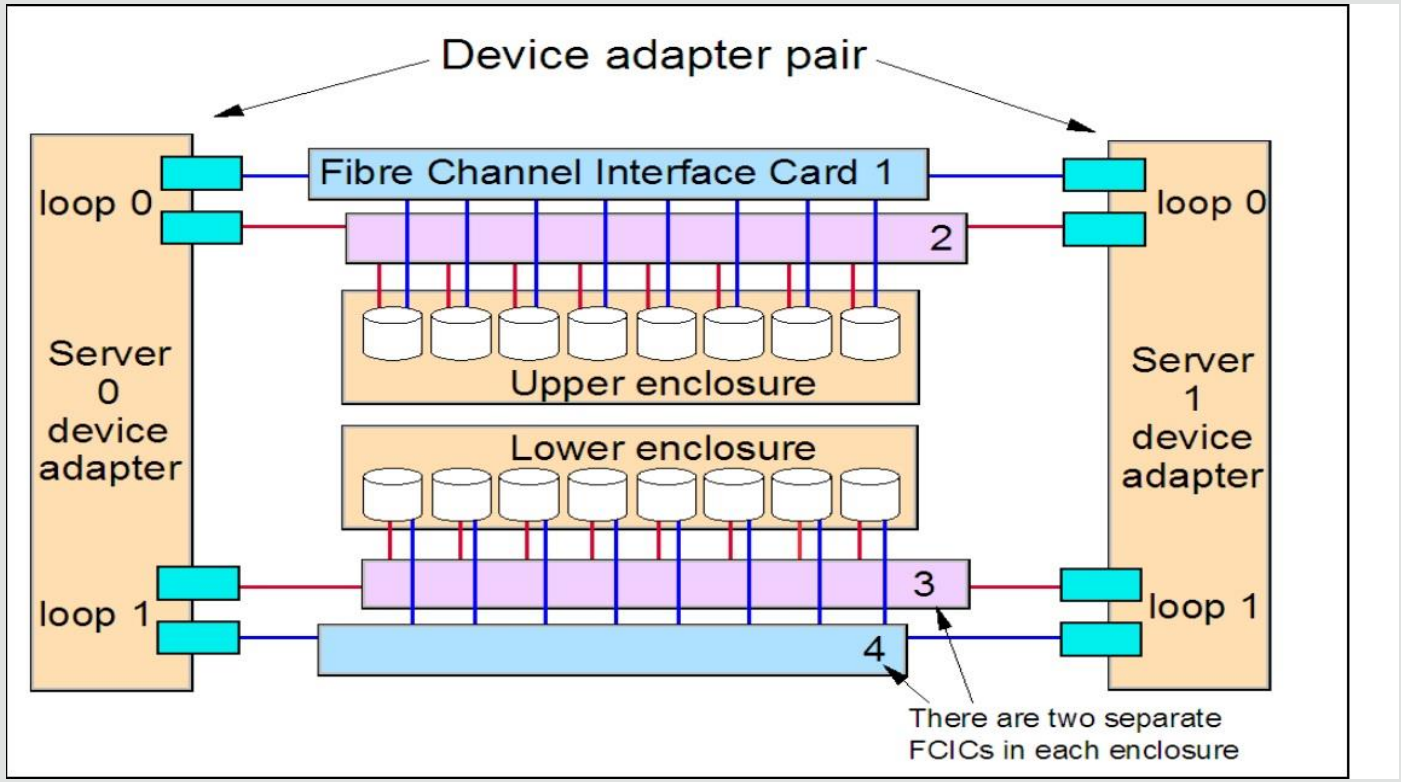


数据库运行在这里运行

网络有多缓慢?

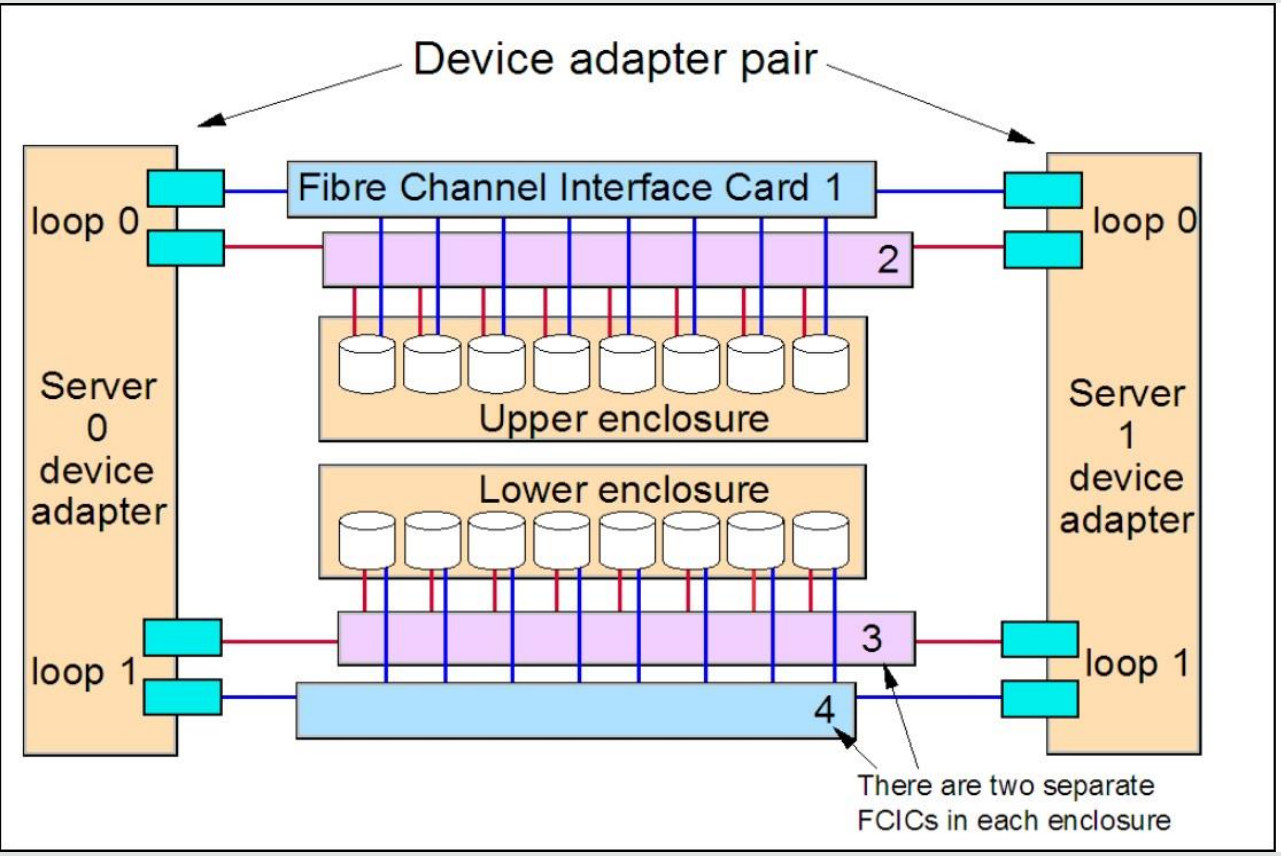
存储把数据发送到数据库服务器

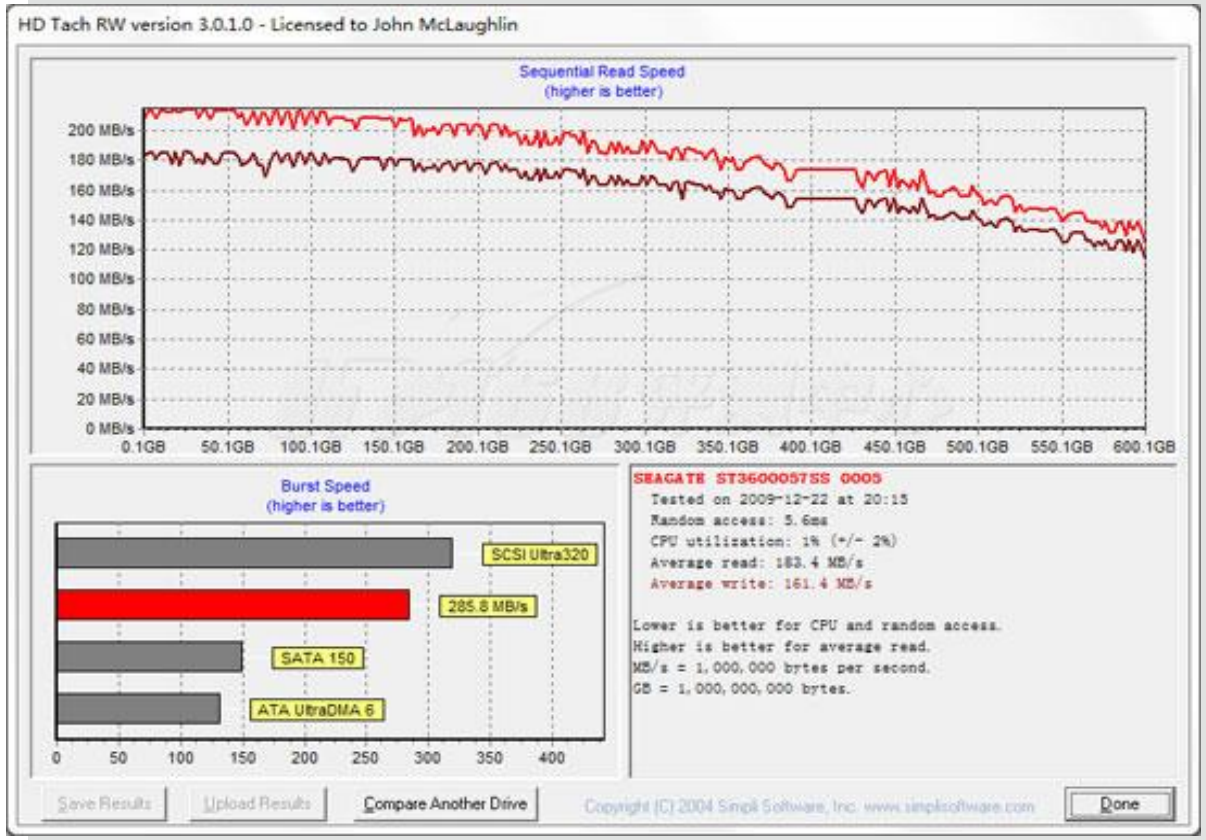
FC盘阵内部的共享后端FC DA（disk adaptor）通道



- 大部分情况下30块以上的硬盘共享一个后端FC（8Gb/16Gb）通道
- 极端情况下存在多达上百块SAS盘共享一条16Gb FC通道
- 共享的带宽也会导致读写延迟

传统SAN存储对于存储介质带宽造成极大浪费

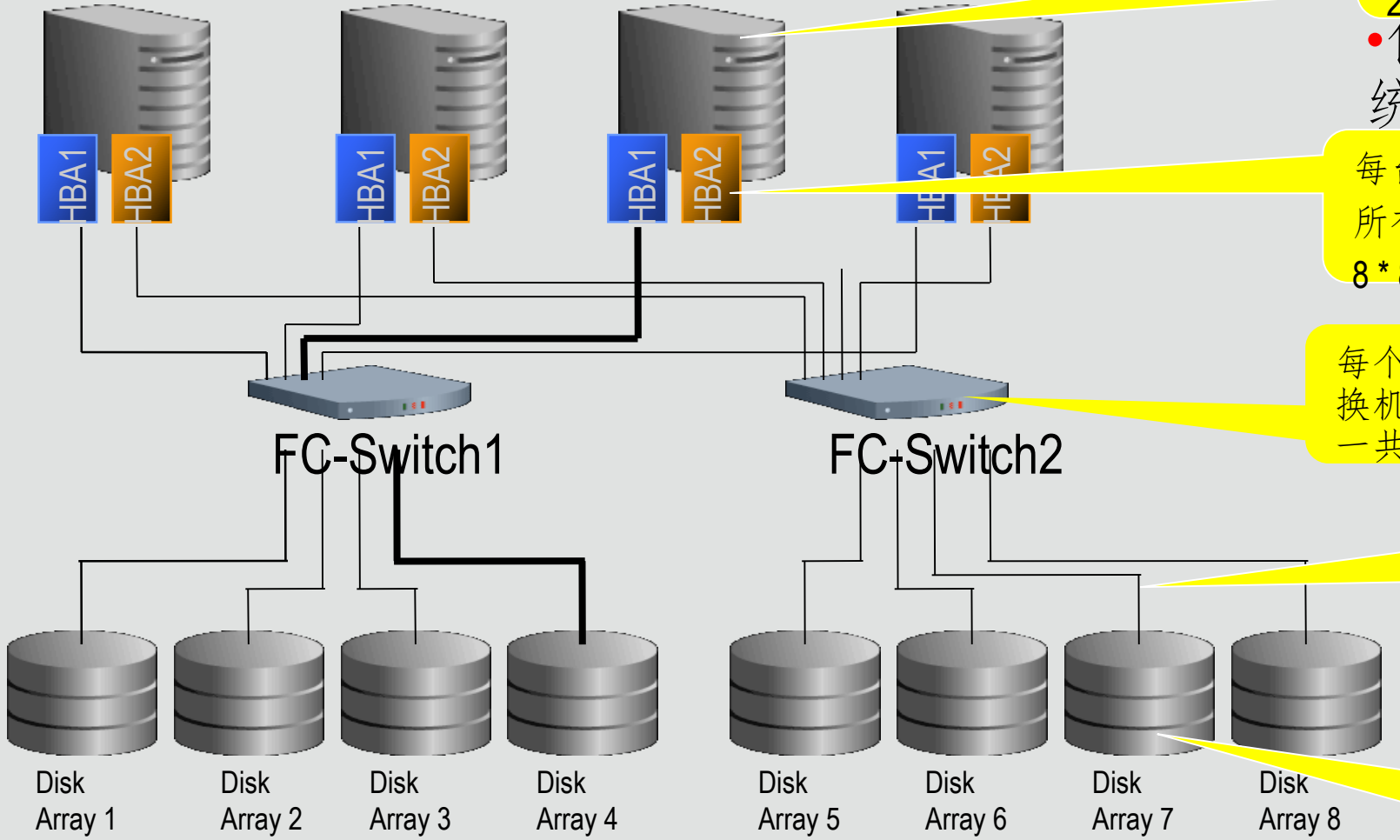




高端存储的存储瓶颈
单块硬盘吞吐<50MB/Sec

服务器级别硬盘顺序读取最低速度：120MB/Sec

现实：均衡设计几乎不可能



每台机器有16 个CPU。
所有4个服务器一共能同时处理数据：64 * 200MB/s = 12.8GB/s

- 任何部件配置不均衡会导致系统瓶颈

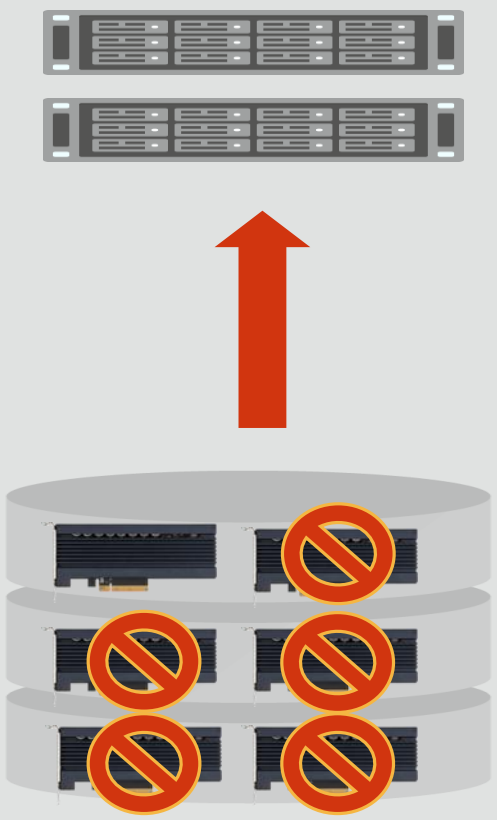
每台机器有2 个8 Gb HBA卡
所有 8个 HBA卡能处理数据：
8 * 800MB/s*2 = 12.8GB/s

每个交换机口能够支持8Gbps 的吞吐能力，每个交换机能够支持总共6.4GB/s的数据处理能力，两台一共支持12.8GB/s

每个磁盘组DA有一个8Gbps的控制器。
所有16个磁盘控制器能处理数据：16* 800MB/s = 12.8GB/s

每块磁盘的数据处理能力为100MB/s，所以一个磁盘组至少需要8块磁盘来达到800MB/s的数据处理能力

数据库系统的瓶颈 (2019年左右)



- 单张 NVMe闪存卡介质带宽即可用尽网络带宽
- 所有闪存设备的吞吐量无法充分发挥

NVMe闪存

5.8 GB/s

>

32/40Gbs 存储连接

4 GB/sec

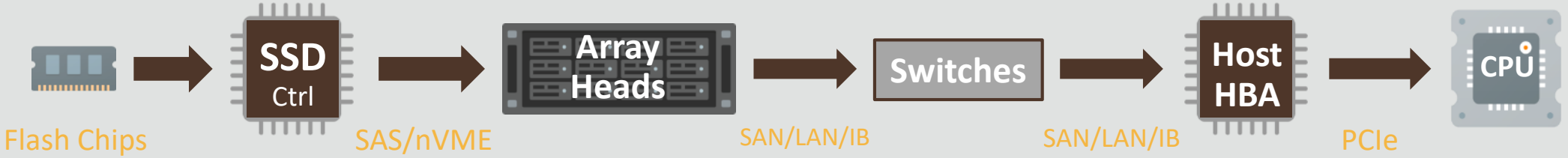
PCIE nVME解决DA带宽，但是仍然有大量瓶颈构成瓶颈

SAN/IB Link = 40Gb
4 GB/sec

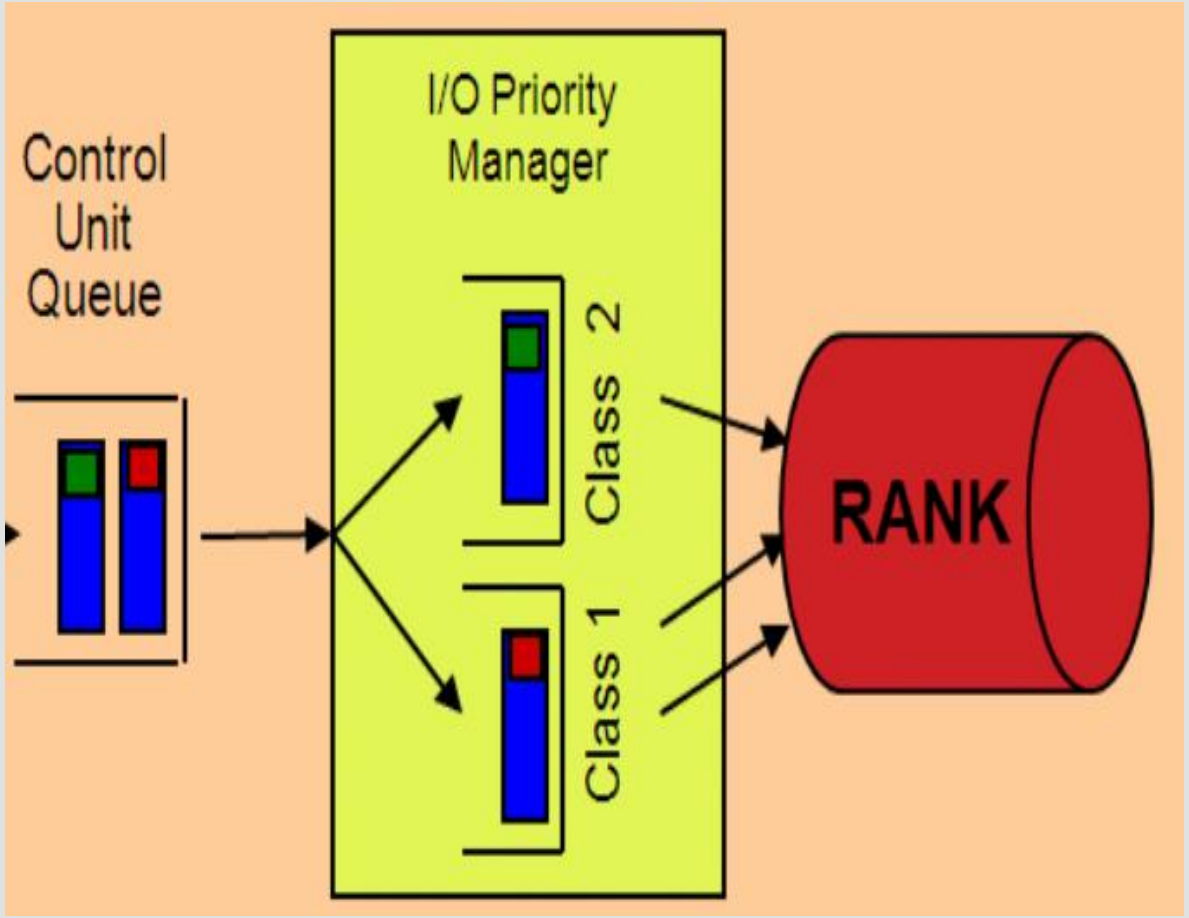
最新的 PCIe 闪存
5.8 GB/sec

- PCIE 的DA接口能为每张nVME卡提供提供8GB/S接口带宽
- **但是在共享式存储内部以及和主机连接的HA通道仍然瓶颈造成了上百倍的性能损失**
  - 单块闪存卡的性能已经与最快速的SAN或LAN/IB链路持平
  - 少量几块闪存卡的聚合性能已经快到无法有效地被从存储系统传输到服务器集群

全闪存存储阵列的 IO 路径: 许多的步骤, 每一步都会增加延迟并且制造新的瓶颈



传统IO实现方式的QoS对数据库来说过于原始

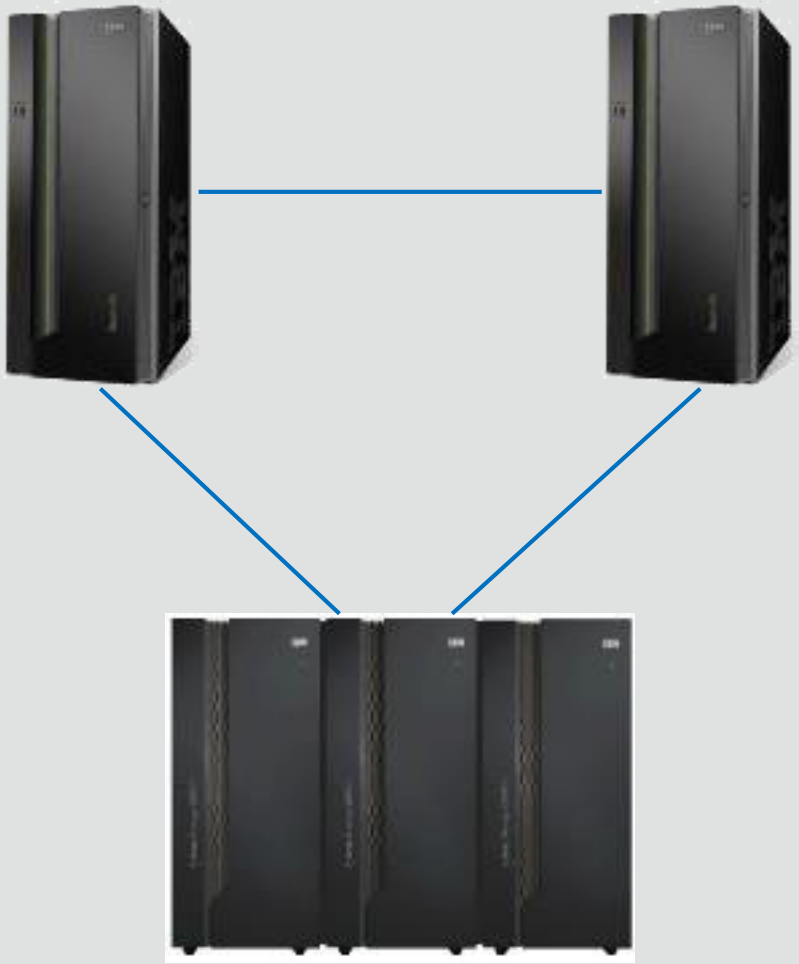


- 难以区分不同数据库的请求
  - 必须在硬件层面区分
- 根本不可能区分同一个数据库的不同工作负载
- 也无法区分相同数据库不同类型IO
  例如：redo log写，DBw写，archive log 写，rman 备份读，buffer cache读，direct path read

数据库发起的不同类型IO对的性能影响

IO类型	数据库整体性能影响	说明
Redo log写	很高	Redo写慢会导致log file sync等待事件，导致整个数据库级别commit等待;RAC环境下更加明显，redo log写还会放大集群GC相关等待事件
Archive log写	很低	只要不是online切归档慢到没有可用的redo，对数据库整体性能影响极小
Buffer cache (DBW)写	较低	对于Oracle数据库DBW写是异步后台等待，不会直接影响数据库性能，除非在DB_CACHE较小等特定情况下出现free buffer wait，或者由于DBW写的IOPS过高超过存储IO能力后出现checkpoint无法完成的情况
Buffer cache读	较高	表示在buffer cache中没有SQL需要的数据，必须从磁盘获取，如果缓慢，当前会话会等待；大面积的buffer cache读也会导致全库性能缓慢
直接路径读	中等	表示当前会话需要的数据优化器认为不应该经过buffer cache缓存，直接从磁盘获得，一般表示全表扫描或者大型SQL导致的排序、hash join/group by的中间数据读取。只影响当前会话。依赖支撑的工作负载特性确定重要性。
RMAN备份读	低	这类IO不会对性能优化有直接作用，应该放到低优先级避免影响业务

传统架构的Oracle RAC横向扩展难题



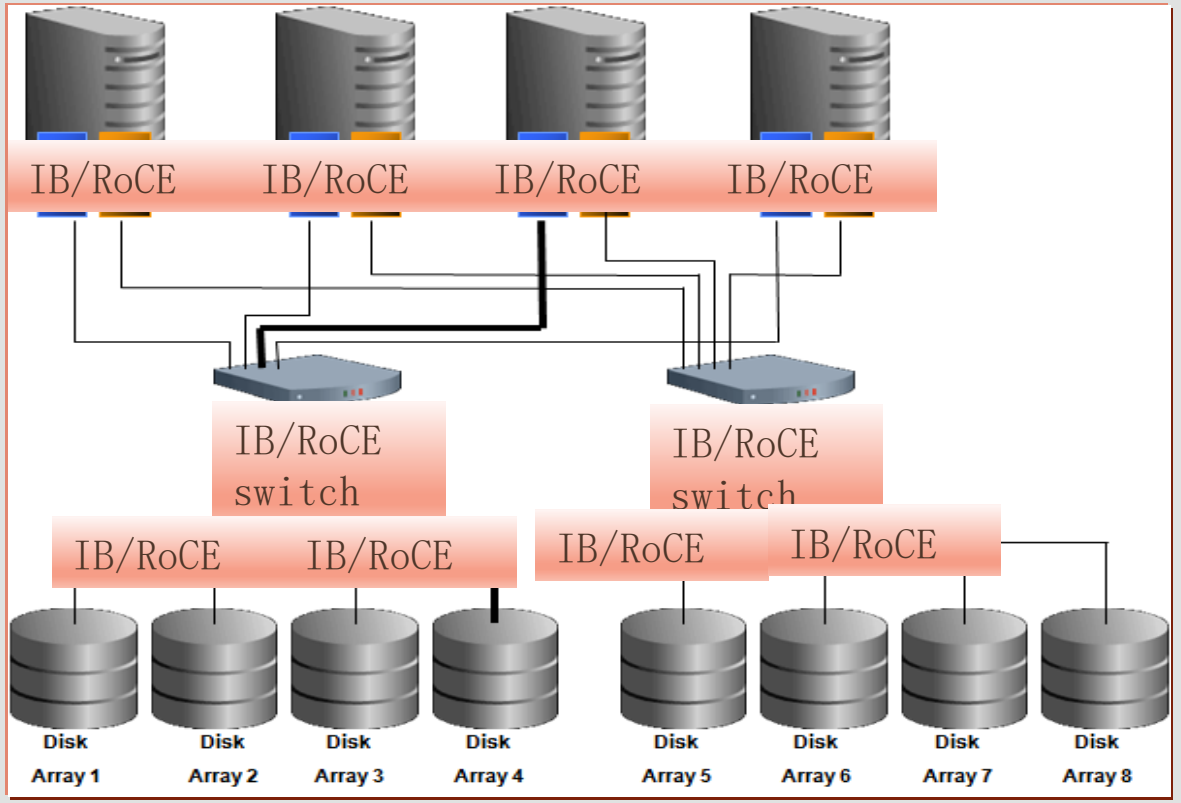
- 受限于以太网的带宽、延迟以及数据访问方式，RAC多节点扩展在传统架构上很难取得良好效果

技术	访问延迟
CPU寄存器	<1ns
CPU Cache	1-50ns
本地内存	100ns
NUMA架构非本地内存	200-300ns
以太网（UDP互联）	~1ms

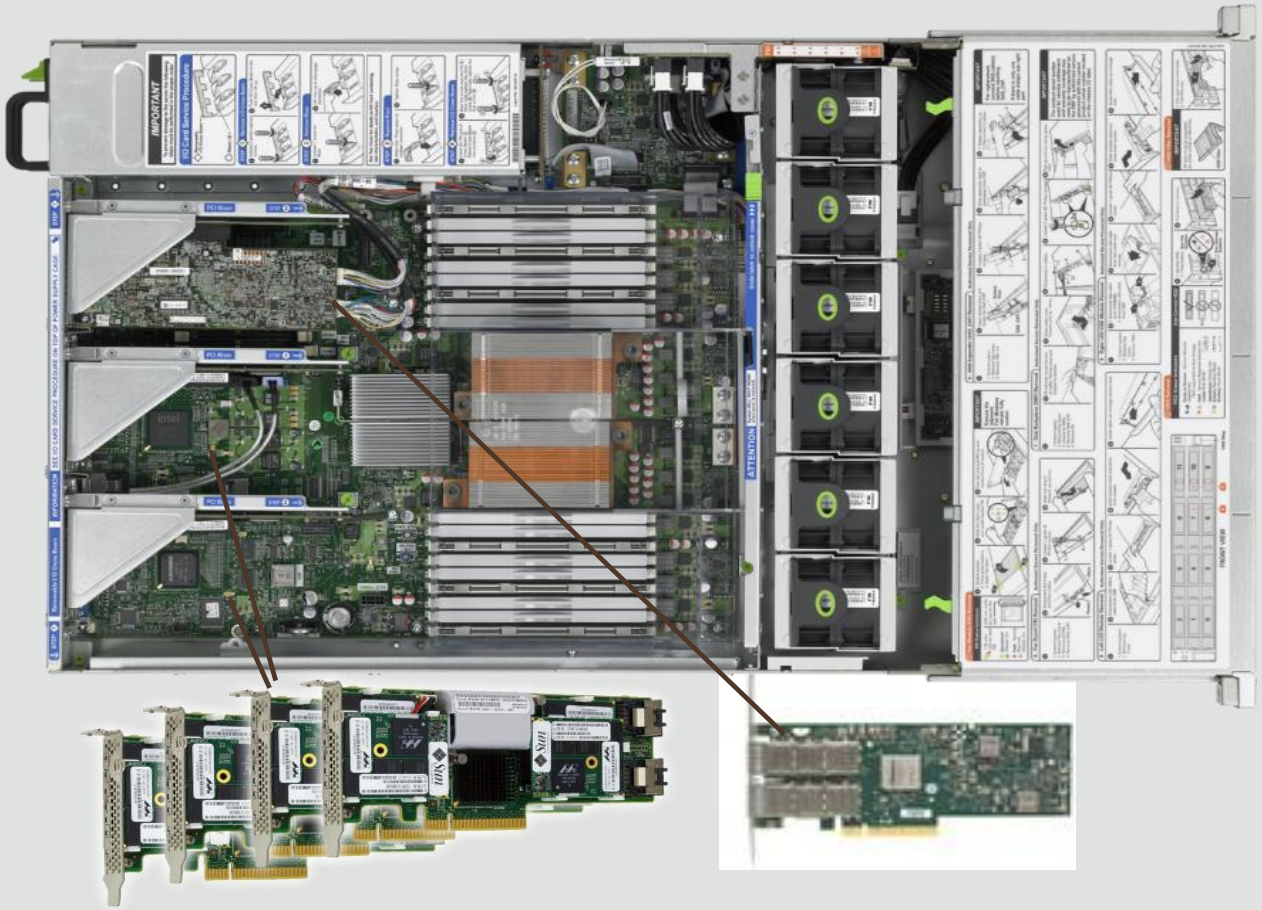
  - 在实践中容易受限于2节点，多节点需要应用分割
  - 处理能力达到上限后替换式服务器扩容
- 共享盘阵的架构受限于SAN存储的扩展能力
  - Log file sync等待放大GC相关等待
  - IO能力达到上限后铲车式替换扩容
  - 即便最高端的SAN存储也难以以驱动当代服务器的计算能力

Exadata思路：提升带宽，降低延迟，减少传输，减少软件栈调用层次中断调用

- 减少需要通过网络传输的数据量，尽可能数据就近处理
- 减少软件栈层次，甚至直接绕过OS系统调用操作硬件
- 解决硬件带宽瓶颈



Exadata的存储架构：解决DA瓶颈



- 多核心XEON CPU
- 12块磁盘共享的上端通道为PCIE 3.0 8X=8GB/s全双工
- 单块磁盘能获得640MB/S全双工带宽
- 存储节点到主机节点带宽为40Gb/s x 2(8GB/s)
- 采用4块PCIE 3.0 8X nVME Flash卡，不和磁盘争用磁盘控制器
- 每块Flash卡提供5.8GB/s带宽
- 介质带宽超过上联IB带宽？

Exadata架构:为Oracle RAC提供独特的分布式数据库架构云平台

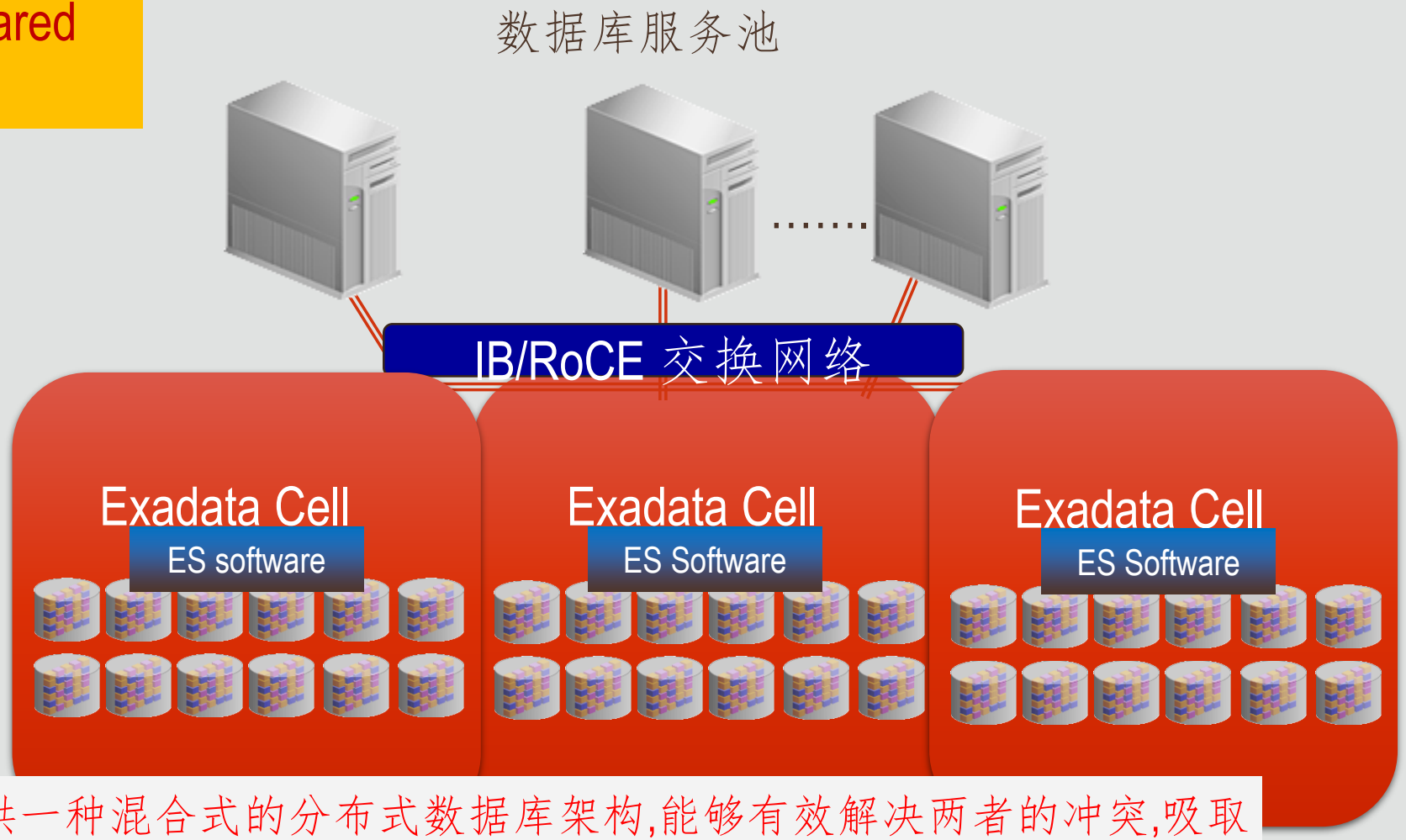
兼备Shared Nothing 、Shared Disk优点，应用透明

具备良好的普适性架构:Shared Nothing and Shared Disk

数据库处理层(Shared Disk)
数据库资源池

硬件RDMA网络层
IB/RoCE互联，低OS介入通信

智能存储层(Shared Nothing MPP)
存储资源池

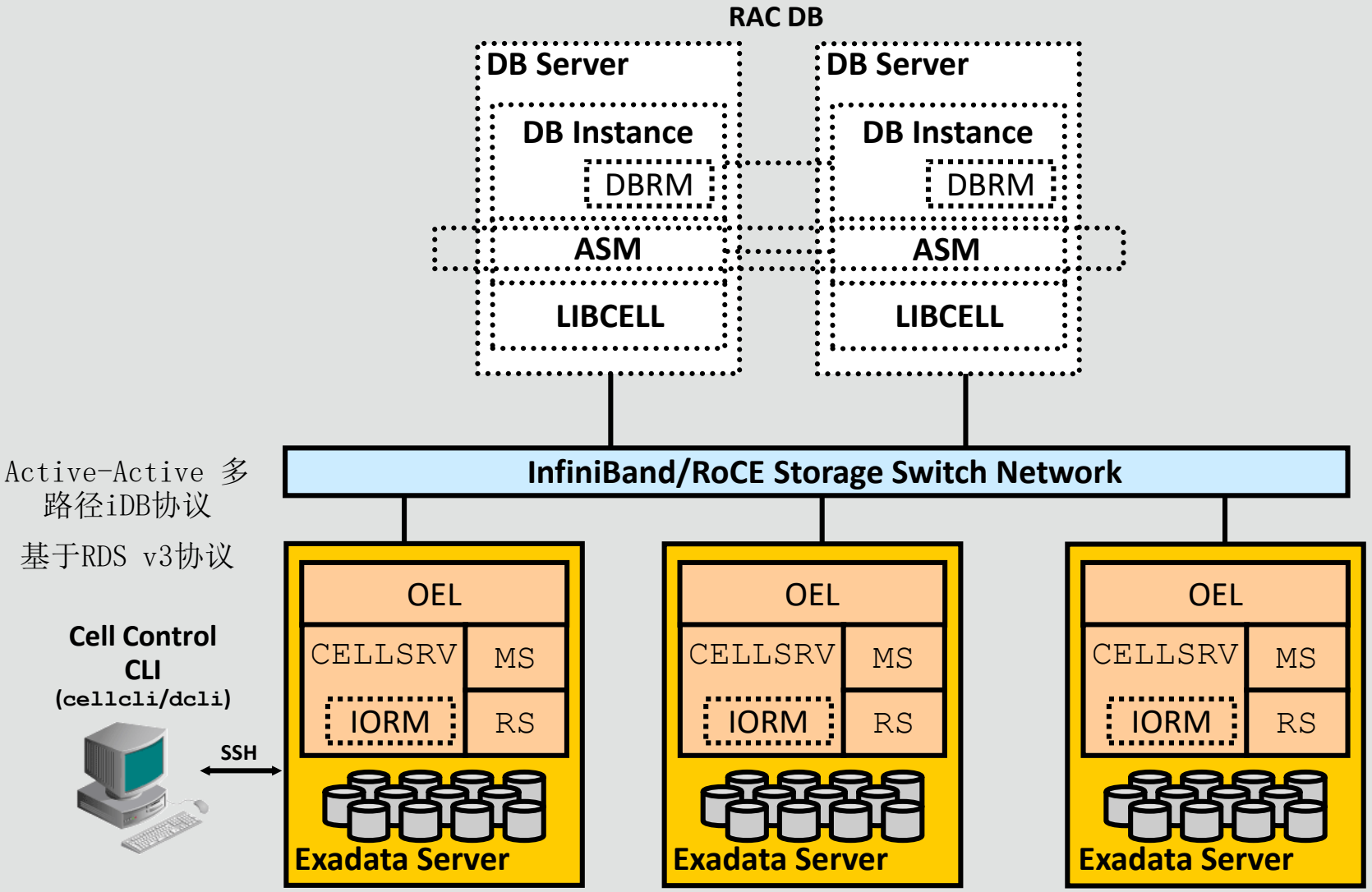


- Exadata是目前业界唯一一种提供一种混合式的分布式数据库架构,能够有效解决两者的冲突,吸取两种架构长处;既可以满足OLTP的高并发、高可用特点；又可以满足OLAP的大数据量处理要求

目录

- 为什么要研发Exadata与Exadata基本架构介绍
- Exadata IO子系统LibCell简介
- Offload-在RAC上引入Share Nothing MPP处理

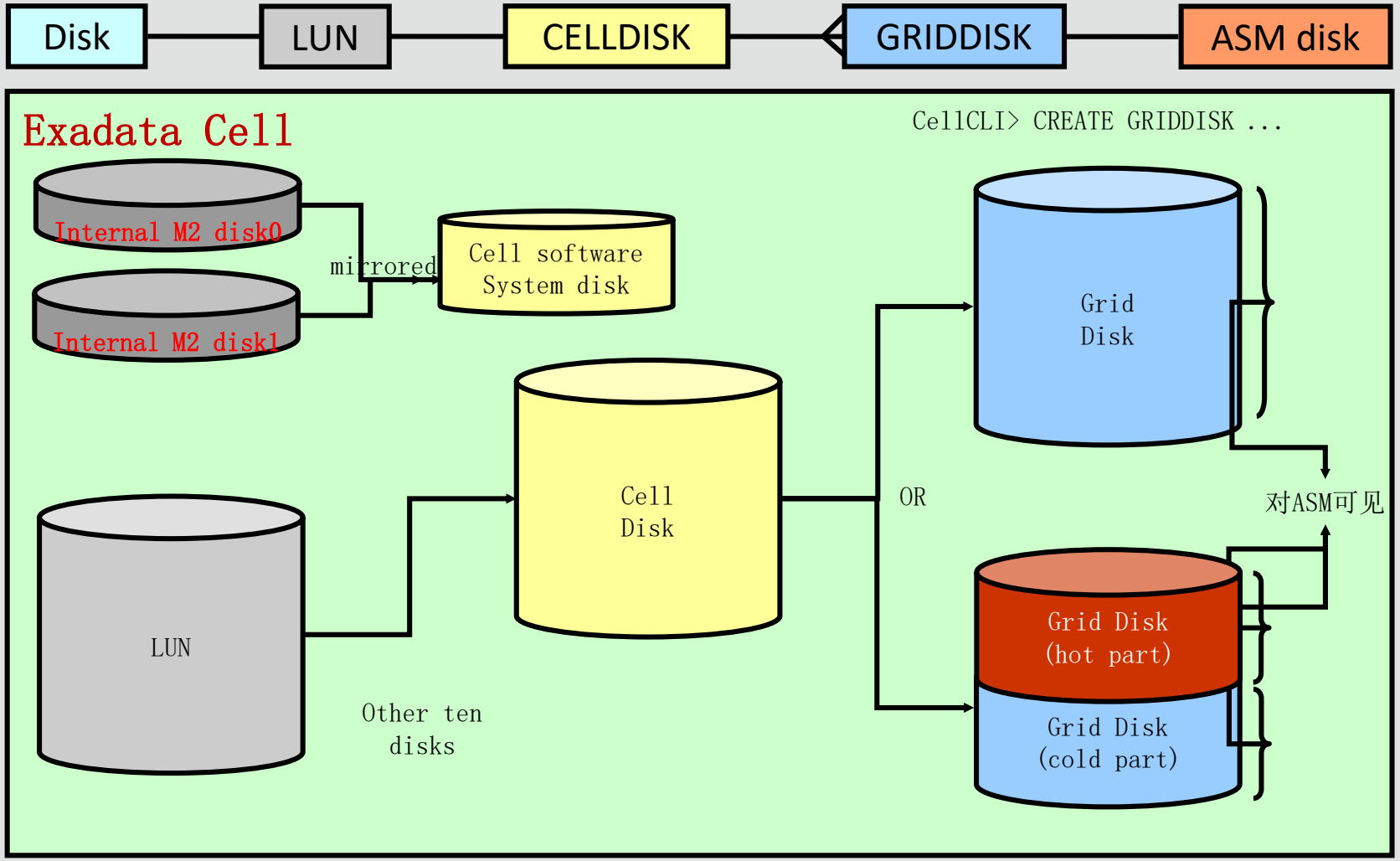
Exadata上数据库的IO架构



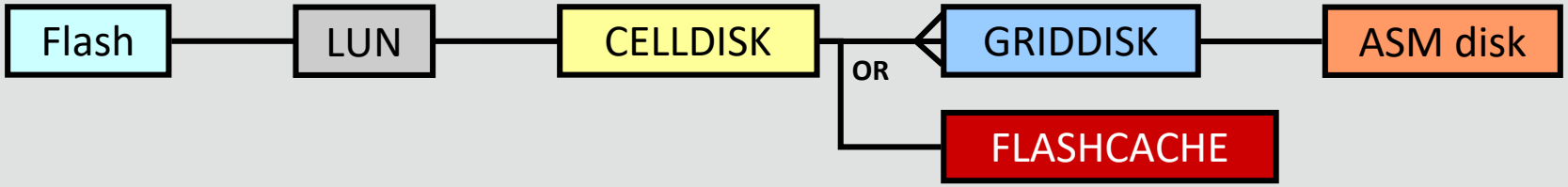
- 在数据库节点上，从OS层面看不到存储节点上的任何磁盘
- 所有存储节点的GridDisk通过libcell和相关配置文件直接由ASM管理，仅仅供Oracle数据库使用
- 如果非要给OS使用，需要使用ACFS提供共享文件系统
  - 但是并不建议将数据库文件存储在ACFS

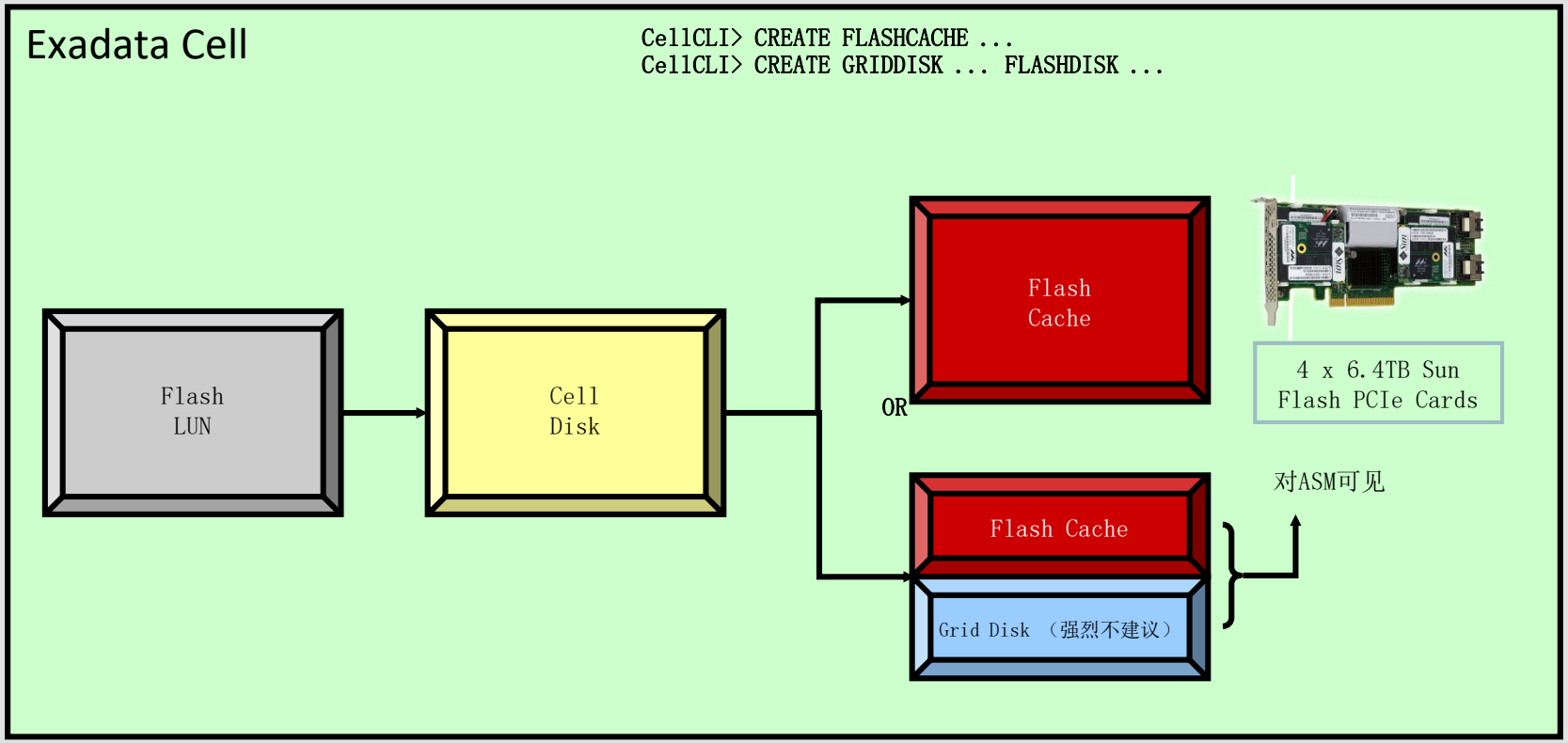
Intelligent Database protocol (iDB)

Exadata X7开始HC存储磁盘的划分

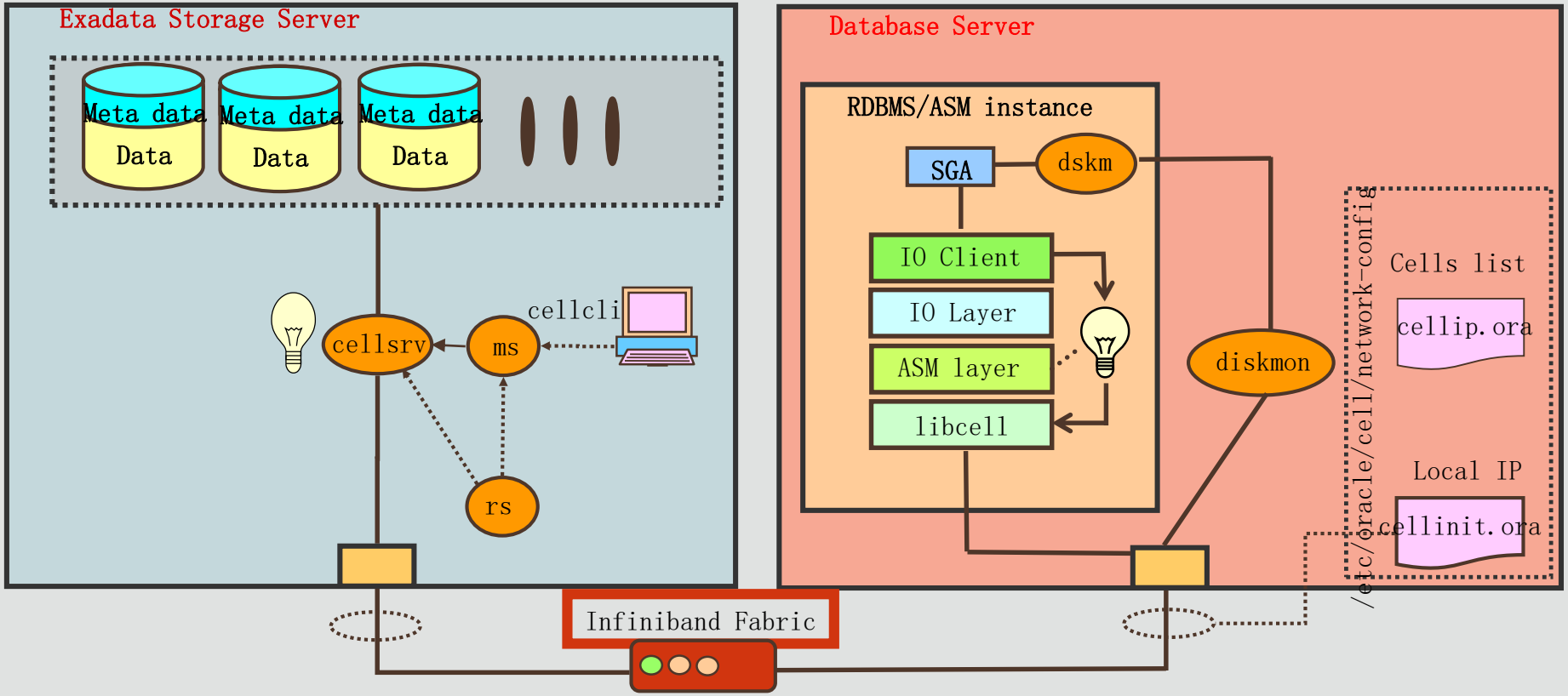


Flash Cache的布局





Exadata上数据库的IO架构（续）



- 所有读写操作由数据库进程直接调用libcell进行，不需要通过asm进程
  - 减少进程间通信
  - ASM进程仅需要和数据库进行交互维护元数据
- 绕过OS的SCSI（iSCSI NVME）内核调用，直接使用RDS V3，缩短通信路径
- 专有协议iDB，携带大量额外IO特性信息，也包括数据库SQL信息

Exadata：IO服务质量保证

带有IO资源管理器的智能存储

每个IO请求都带标记：谁发起的（CDB/pdb，Consume Group），目的和优先级

保证在复杂工作负载、多应用整合，高负荷下的性能

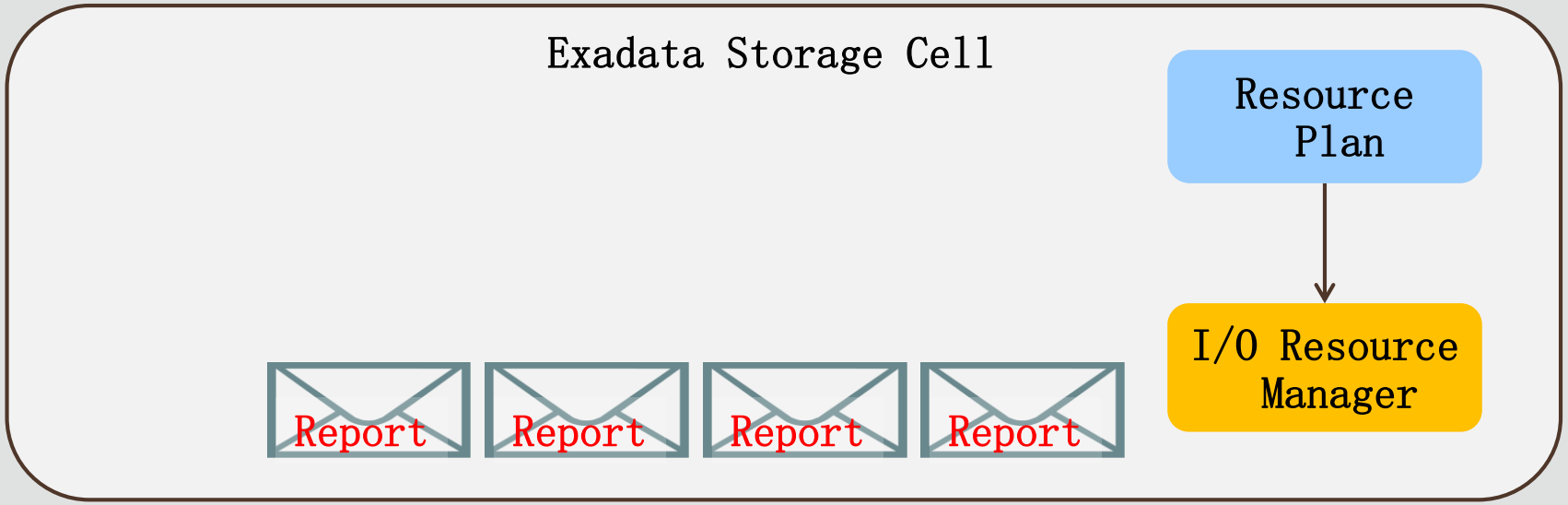
IO任务	Exadata存储软件采取的动作
关键数据仓库的表扫描	高优先级，IORM优先于其他全表扫描对其提供服务，同时从Flashcache和磁盘扫描
即席查询表扫描	低优先级，资源消耗大查询。只有flash有空闲空间时提供服务，降低磁盘或FlashCache 的IO优先级.
DBWR 写 – 没有 “free buffer wait”	不急： – 大量空闲buffer. IORM降低其IO优先级
DBWR 要解决 “free buffer wait”等待	紧急 – 用户操作被阻止. IORM优先服务此 I/O
LGWR redo 写	高优先级I/O. 通过Exadata Flash Log加速!
OLTP应用的Buffer Cache读IO	中等优先级I/O. 主要通过Flash服务，一般比其他用户IO优先级高，基于IO资源计划调整
RMAN备份读取	低优先级，比用户IO优先级都低，不会通过FlashCache缓存

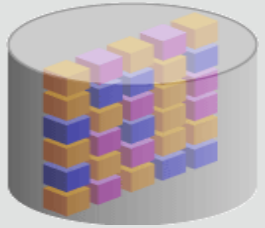
Exadata采用了IORM后的I/O 调度方式

I/O Resource Manager 控制到磁盘的IO顺序

IORM 产生足够的IO保持磁盘的效率和负荷

I/Os 根据需要在Exadata内部根据需要排序





✓ 通过使用Resource Plan来确定IO请求的顺序

✓防止由于某些业务/应用的IO请求过高导致整个数据库SLA无法保证

没有针对数据库的IO优先级全闪存也无法保障性能

Top 10 Foreground Events by Total Wait Time

Event	Waits	Total Wait Time (sec)	Wait Avg(ms)	% DB time	Wait Class
db file sequ					
log file sync					
DB CPU					
db file paral					
db file scatt					
gc buffer bu					
read by othe					
gc cr grant 2					
gc cr block					
gc cr multi b					

Top 10 Foreground Events by Total Wait Time

Event	Waits	Total Wait Time (sec)	Wait Avg(ms)	% DB time	Wait Class
db file se					
DB CPU					
log file sy					
db file pa					
gc buffer					
read by c					
db file sc					
gc cr bloc					
SQL*Net					
direct pat					

Top 10 Foreground Events by Total Wait Time

Event	Waits	Total Wait Time (sec)	Wait Avg(ms)	% DB time	Wait Class
log file sync	1,413,334	20.3K	14.39	58.7	Commit
DB CPU		11K		31.8	
db file sequential read	364,321	2439.8	6.70	7.0	User I/O
buffer busy waits	63,420	172.9	2.73	.5	Concurrency
enq: TX - index contention	14,034	128.9	9.19	.4	Concurrency
db file parallel read	12,004	110.7	9.22	.3	User I/O
enq: TX - row lock contention	11,458	106.9	9.33	.3	Application
gc current block 2-way	302,366	100.2	0.33	.3	Cluster
gc current block busy	12,306	83.7	6.80	.2	Cluster
gc cr block busy	8,277	69.6	8.40	.2	Cluster

- 在配置全闪存情况下，超过50%等待和IO有关
- 其余为GC相关等待

采用全闪存后还有高IO等待的原因：其他IO争用队列

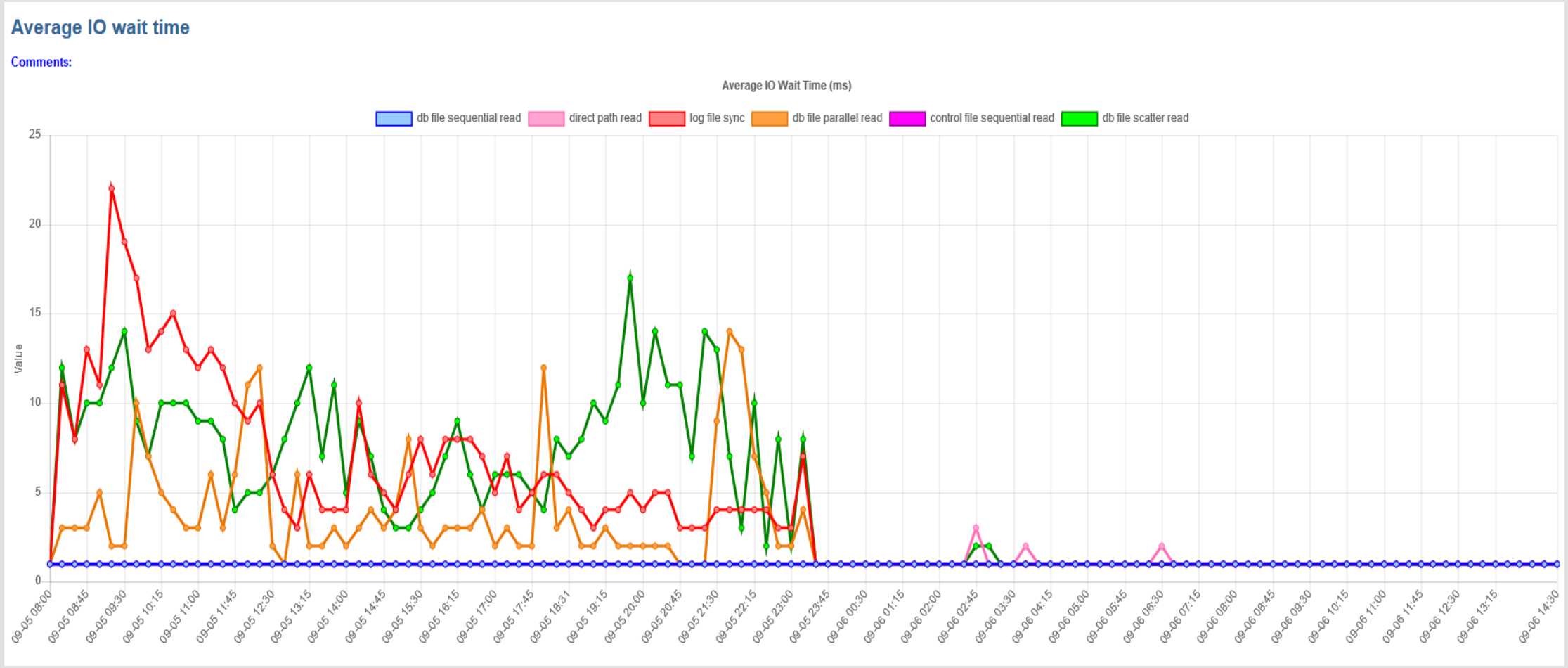
由于较多的全表扫描影响随机读

Function Name	Reads: Data	Reqs per sec	Data per sec
Direct Reads	237.5G	313.68	268.055M
Buffer Cache Reads	97G	9029.00	109.424M
Others	18.8G	33.44	21.18M
Streams AQ	18.8G	15.51	21.255M
DBWR	0M	0.00	0M
LGWR	14M	1.03	.015M
Direct Writes	0M	0.00	0M
TOTAL:	372.1G	9392.66	419.93M

由于在进行RMAN数据库备份

Function Name	Reads: Data	Reqs per sec	Data per sec
RMAN	192.7G	217.92	217.575M
Others	23.2G	47.23	26.211M
LGWR	14M	1.04	.015M
DBWR	0M	0.00	0M
Buffer Cache Reads	5.2G	745.32	5.921M
Direct Reads	307M	40.18	.338M
Direct Writes	0M	0.02	0M
TOTAL:	221.5G	1051.72	250.061M

从全闪存迁移到Exadata大容量硬盘配置后的IO表现



- 切换前后IO平均等待时间存在明显的下降，尤其是影响交易提交的log file sync等待时间下降更加明显

部署在Exadata 一个“OLTP” 环境

	Per Second
DB Time(s):	14.4
DB CPU(s):	11.0
Redo size (bytes):	4,693,067.3
Logical read (blocks):	1,242,493.7

	Per Second
DB Time(s):	6.7
DB CPU(s):	6.4
Redo size (bytes):	1,088,275.6

	Per Second
DB Time(s):	7.6
DB CPU(s):	4.8
Redo size (bytes):	1,134,026.9
Logical read (blocks):	980,320.7
Block changes:	7,632.9
Physical read (blocks):	154,365.7
Physical write (blocks):	333.8
Read IO requests:	1,810.4
Write IO requests:	156.2
Read IO (MB):	1,206.0
Write IO (MB):	2.6
Global Cache blocks received:	174.3
Global Cache blocks served:	619.9
User calls:	9,819.5
Parses (SQL):	1,660.3
Hard parses (SQL):	381.3
SQL Work Area (MB):	11.8
Logons:	2.2
Executes (SQL):	1,861.1
Rollbacks:	22.4
Transactions:	187.3

	Per Second
DB Time(s):	5.0
DB CPU(s):	3.8
Redo size (bytes):	2,210,789.0
Logical read (blocks):	254,479.2

	Per Second
DB Time(s):	7.0
DB CPU(s):	6.3
Redo size (bytes):	1,016,544.9

	Per Second
DB Time(s):	8.3
DB CPU(s):	5.3
Redo size (bytes):	3,001,222.3
Logical read (blocks):	1,110,411.2
Block changes:	22,602.3
Physical read (blocks):	77,136.2
Physical write (blocks):	2,674.9
Read IO requests:	3,115.6
Write IO requests:	201.0
Read IO (MB):	602.6
Write IO (MB):	20.9
Global Cache blocks received:	1,126.5
Global Cache blocks served:	403.6
User calls:	1,323.2
Parses (SQL):	248.8
Hard parses (SQL):	40.3
SQL Work Area (MB):	100.5
Logons:	0.9
Executes (SQL):	10,969.0
Rollbacks:	0.1
Transactions:	24.9

	Per Second
DB Time(s):	12.7
DB CPU(s):	10.0

	Per Second
DB Time(s):	6.6
DB CPU(s):	6.3
Redo size (bytes):	978,773.9

	Per Second
DB Time(s):	0.9
DB CPU(s):	0.7
Redo size (bytes):	3,414,841.7
Logical read (blocks):	86,644.1
Block changes:	15,732.3
Physical read (blocks):	20,767.5
Physical write (blocks):	346.9
Read IO requests:	370.0
Write IO requests:	185.8
Read IO (MB):	162.3
Write IO (MB):	2.7
Global Cache blocks received:	308.7
Global Cache blocks served:	585.9
User calls:	650.8
Parses (SQL):	43.5
Hard parses (SQL):	11.0
SQL Work Area (MB):	5.0
Logons:	0.5
Executes (SQL):	364.3
Rollbacks:	0.0
Transactions:	3.2

- OLTP系统同时承载接近18GB/s的全表扫描SQL带宽

部署在Exadata 一个“OLTP” IO指标

Event	Waits	Total Wait Time (sec)	Wait Avg(ms)	% DB time
DB CPU		39.6K		76.5
cell single block physical read	19,718,617	5455.3	0	10.5
SQL*Net message from dblink	130,804	1439	11	2.8
read by other session	4,030,561	1196.2	0	2.3
enq: TX - row lock contention	616	1079	1752	2.1
SQL*Net more data to client	2,310,577	778.6	0	1.5
cell multiblock physical read	360,936	619.8	2	1.2
cell smart table scan	4,612,991	479.5	0	.9

Event	Waits	Total Wait Time (sec)	Wait Avg(ms)	% DB time
DB CPU		22.9K		95.1
cell single block physical read	2,567,554	755	0	3.1
cell smart table scan	5,307,211	412.7	0	1.7
log file sync	556,154	192.9	0	.8
gc current block 2-way	2,007,121	169.5	0	.7
gc cr block 2-way	1,058,483	106.7	0	.4
gc current block 3-way	603,103	81.1	0	.3
gc cr block 3-way	439,189	63.2	0	.3
SQL*Net more data from client	1,641,920	54.9	0	.2
SQL*Net message to client	45,881,314	52.8	0	.2

Event	Waits	Total Wait Time (sec)	Wait Avg(ms)	% DB time
DB CPU		17.4K		63.7
enq: TX - row lock contention	37	8517.6	230205	31.2
log file sync	732,325	414	1	1.5
cell single block physical read	1,424,554	413.6	0	1.5
SQL*Net message from dblink	352,281	217.2	1	.8
SQL*Net more data to client	10,139,107	97.5	0	.4
SQL*Net break/reset to client	136,872	73.7	1	.3
cell smart table scan	211,609	51.1	0	.2
enq: HW - contention	24,417	43.6	2	.2
SQL*Net message to client	33,843,980	34.3	0	.1

Event	Waits	Total Wait Time (sec)	Wait Avg(ms)	% DB time
DB CPU		13.8K		76.6
SQL*Net break/reset to client	140,612	1065	8	5.9
cell single block physical read	3,078,830	895.5	0	5.0
direct path read temp	53,393	873.3	16	4.8
enq: TX - row lock contention	23	419.7	18247	2.3
SQL*Net more data from client	164,458	314.6	2	1.7
gc cr grant 2-way	1,706,289	114	0	.6
log file sync	220,049	107.4	0	.6
cell multiblock physical read	91,525	78	1	.4
gc buffer busy acquire	175,481	61.6	0	.3

Event	Waits	Total Wait Time (sec)	Wait Avg(ms)	% DB time
DB CPU		22.8K		90.1
direct path read temp	108,465	1407.3	13	5.6
cell smart table scan	7,123,319	544.1	0	2.2
cell single block physical read	1,266,337	392.7	0	1.6
log file sync	472,990	166.4	0	.7
gc current block 2-way	1,581,113	140.1	0	.6
gc cr block 2-way	878,899	88.4	0	.3
gc current block 3-way	610,334	82.4	0	.3
gc cr block 3-way	489,112	70.8	0	.3
SQL*Net more data from client	1,479,728	70.1	0	.3

Event	Waits	Total Wait Time (sec)	Wait Avg(ms)	% DB time
DB CPU		19.2K		64.0
enq: TX - row lock contention	13	4972	382464	16.5
cell single block physical read	7,018,206	1891.5	0	6.3
SQL*Net more data to client	776,910	1219.9	2	4.1
direct path read temp	61,438	824.7	13	2.7
cell multiblock physical read	405,548	596.9	1	2.0
SQL*Net message from dblink	257,158	352.1	1	1.2
gc cr grant 2-way	2,687,922	172.5	0	.6
SQL*Net more data from dblink	320,845	144.5	0	.5
cell smart table scan	1,363,520	135.8	0	.5

Event	Waits	Total Wait Time (sec)	Wait Avg(ms)	% DB time
DB CPU		36K		78.9
cell single block physical read	17,141,177	5159	0	11.3
enq: TX - row lock contention	551	611.3	1109	1.3
SQL*Net more data to client	2,165,682	501.4	0	1.1
gc current block 2-way	5,968,855	465.7	0	1.0
cell smart table scan	3,777,624	408.7	0	.9
gc cr disk read	6,239,422	405	0	.9
log file sync	1,171,424	396.7	0	.9
cell multiblock physical read	222,252	368.2	2	.8
SQL*Net message from dblink	65,825	337.6	5	.7

Event	Waits	Total Wait Time (sec)	Wait Avg(ms)	% DB time
DB CPU		22.7K		95.2
cell smart table scan	9,213,297	699.2	0	2.9
cell single block physical read	1,232,025	389.3	0	1.6
log file sync	502,098	178.5	0	.7
gc current block 2-way	1,632,521	142.8	0	.6
gc cr block 2-way	1,050,415	105.8	0	.4
gc current block 3-way	558,685	74.1	0	.3
gc cr block 3-way	417,055	59.1	0	.2
SQL*Net message to client	45,380,101	49.9	0	.2
direct path read	77,503	41.5	1	.2

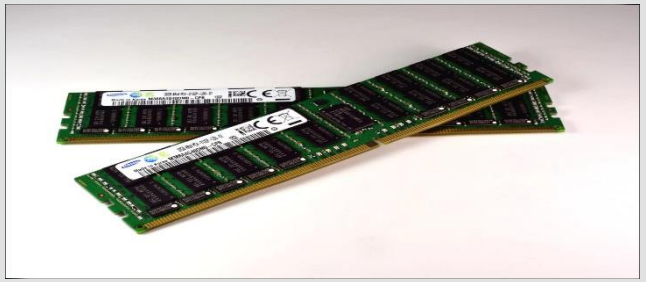
Event	Waits	Total Wait Time (sec)	Wait Avg(ms)	% DB time
DB CPU		2491.8		80.9
cell single block physical read	563,571	148.2	0	4.8
cell multiblock physical read	41,363	88.8	2	2.9
gc cr multi block request	50,555	88.5	2	2.9
SQL*Net message from dblink	49,209	34.5	1	1.1
gc current block 2-way	388,262	30	0	1.0
cell smart table scan	270,746	23	0	.7
SQL*Net more data to client	1,385,651	15.3	0	.5
gc cr grant 2-way	190,878	13.8	0	.4
direct path read	12,707	12.3	1	.4

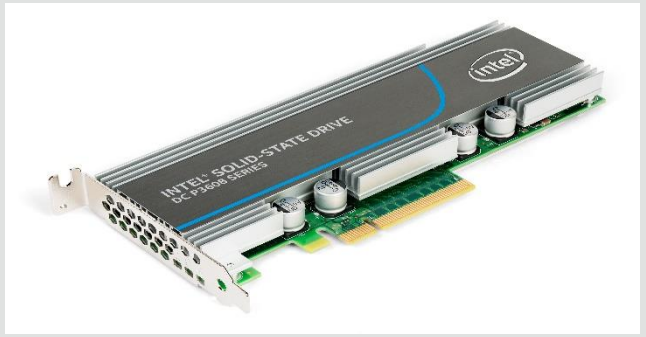
- 18GB/S持续全表扫描情况下
  - log file sync<1ms
  - Buffer cache读(cell single block read): <400us

目录

- 为什么要研发Exadata与Exadata基本架构介绍
- Exadata IO子系统LibCell简介
- Offload-在RAC上引入Share Nothing MPP处理

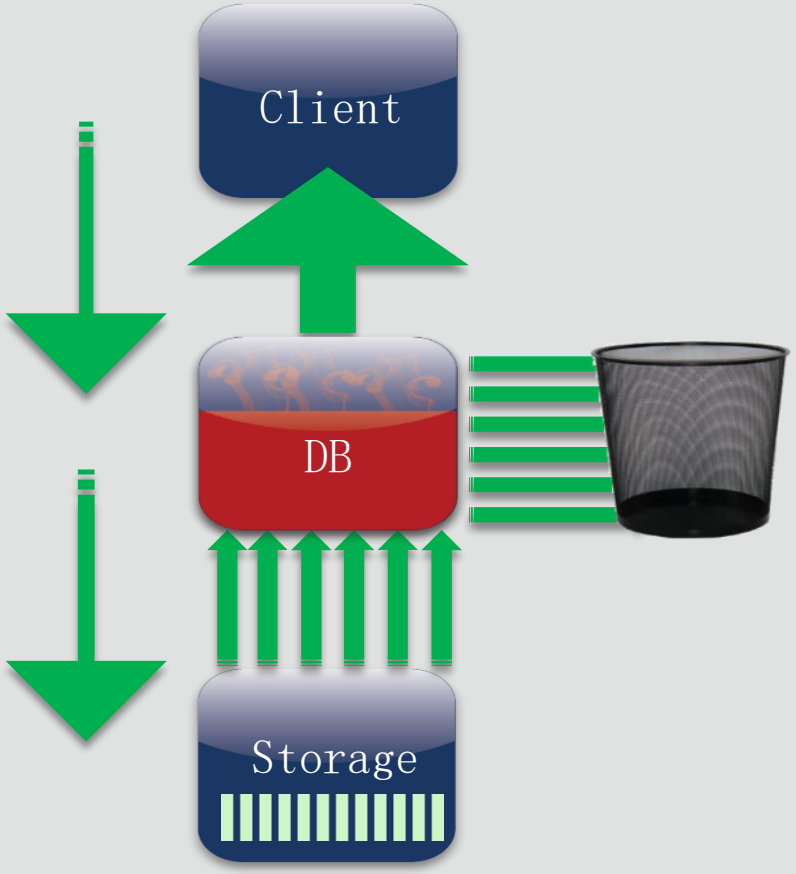
计算与数据：数据库要快速必须依赖的计算机基本原理





- 在冯.诺依曼和哈佛架构下数据离计算越近计算效率就越高
- 但是在怎样将数据离计算近却产生了两种完全不同的approach
  - 利用Cache，将数据送到离计算更近的地方
  - 将计算能力放置在离数据更近的地方
- 实际面向现实数据库计算的情况
  - OLTP有相对较小的热数据（SGA buffer cache）
  - OLAP的热数据就相对较大(Direct path read)

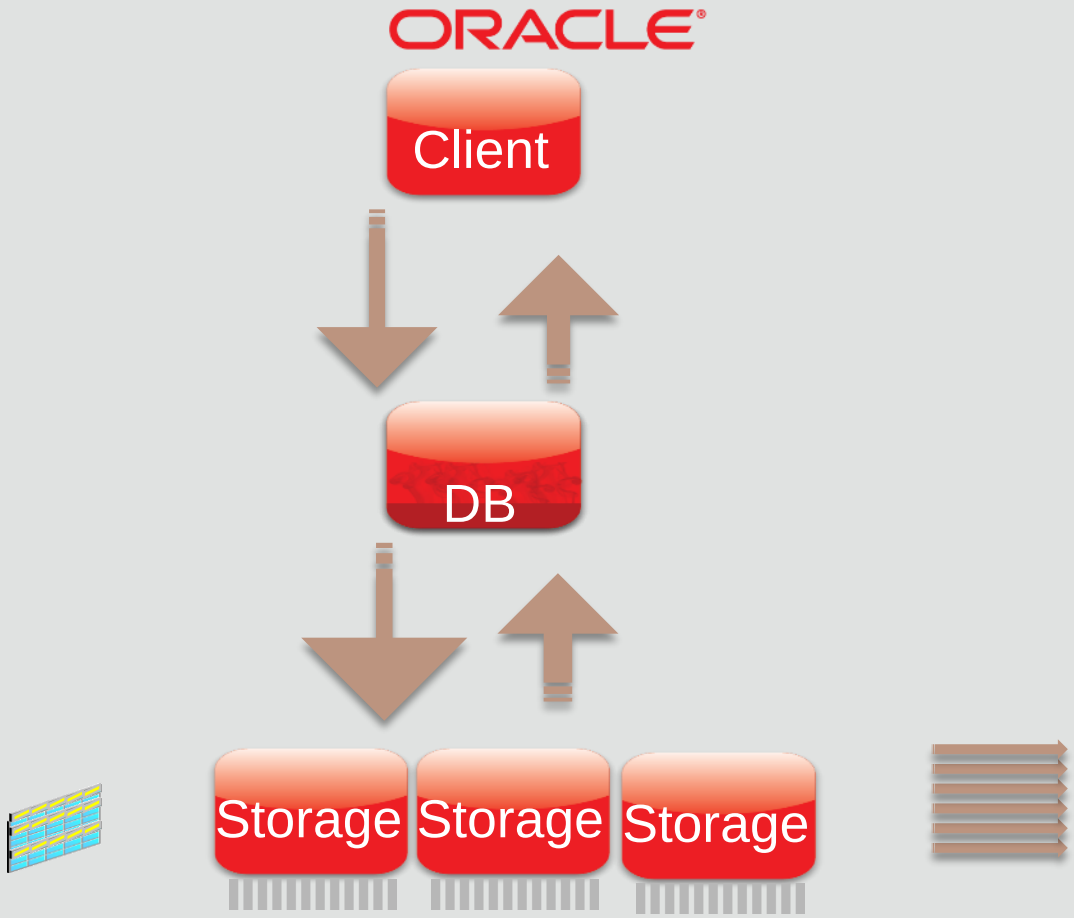
非Exadata Oracle进行全表扫描传统数据库的处理过程



- 传统存储的所有逻辑处理都是在数据库层完成，存储只做简单的IO
- 大量从存储返回到数据库主机的数据实际上是被丢弃了
- 这些被丢弃的数据消耗的宝贵的资源，而且影响了性能和工作负载

Exadata的新处理方式

Offload 到存储层进行share nothing MPP就近处理



- 只返回有效的列revenue和需要的行(where discount = 6%)
- CPU的消耗被大部分offload到存储自动并行
  - 即便在数据库层没有打开并行查询
- 让数据库服务器CPU从表扫描的压力中释放出来，并且避免了大量的无用信息
- 并不需要特殊的开发方式，只需要有正确的数据库参数和统计信息

Smart Scan的能力

所有适合share nothing执行让数据库代码接近数据的处理过程

- 全表/分区/索引快速扫描的：解压缩，标量函数
- 行过滤:
  - 只将需要的部分行返回给数据库节点，而不是扫描的所有整行.
- 列投影:
  - 只返回查询需要的列给数据库节点.
- Join 处理:
  - 大小表join在智能存储内部通过bloom filter处理
- 汇聚函数：要求19C以上数据库，部分场景
- 扫描加密数据
- 数据挖掘评分:
- 创建/扩展表空间

Exadata 存储索引(Storage Index)

	MIN	MAX
Data Chunk #1		
Order_date	03-SEP-2009	03-SEP-2009
Ship_date	05-SEP-2009	07-OCT-2009
Cust_ID	10075	20098
Prod_ID	20010	32932
Amount	10,000	20,000
Data Chunk #2		
Order_date	03-SEP-2009	03-SEP-2009
Ship_date	01-OCT-2009	03-NOV-2009
Cust_ID	10000	80300
Prod_ID	2030	30000
Amount	10,000	40,000

✓

X

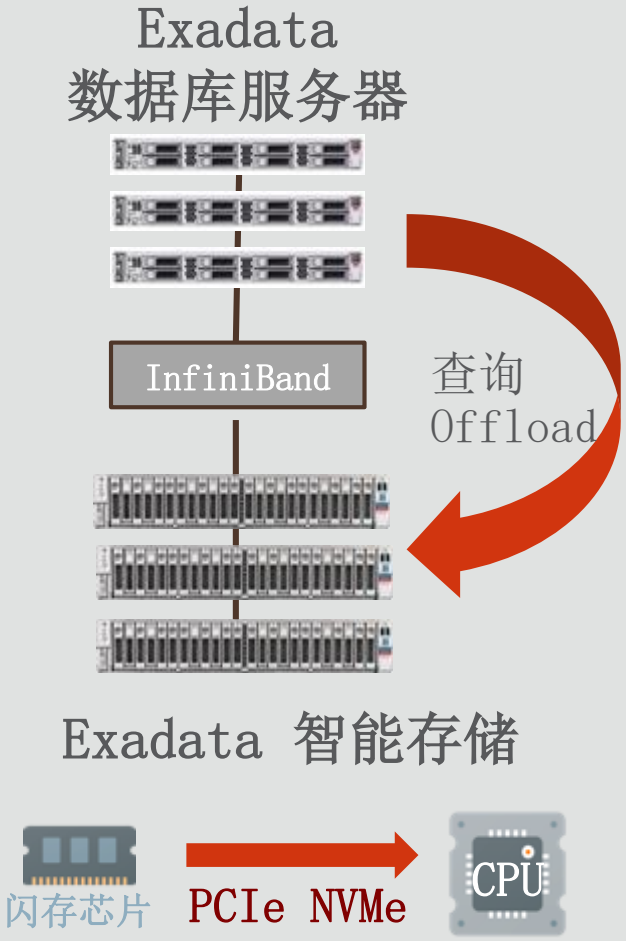
WHERE ship_date
between '01-SEP-2009'
and '31-SEP-2009'

排除 data chunk #2

Order_date	Ship_date	Cust_ID	Prod_ID	Amount
03-SEP-2009	19-SEP-2009	10075	32932	10,000.00
03-SEP-2009	05-SEP-2009	20098	20098	20,000.00
03-SEP-2009	07-OCT-2009	10089	20010	15,000.00
03-SEP-2009	01-OCT-2009	20100	10000	35,000.00
03-SEP-2009	19-OCT-2009	80300	30000	10,000.00
03-SEP-2009	03-NOV-2009	10000	2030	40,000.00

- 存储索引将无关data chunk忽略掉，提供类似 分区修剪的功能
- 11.2单表支持8列，12.2支持24列而且支持集合元素修剪

Exadata通过共享闪存获取了内存级别的性能



- Exadata X8为任何单台数据库服务器提供 560GB/s的闪存扫描带宽
  - 接近所有8台数据库服务器800GB/s 聚合DRAM带宽
- 将绝大部分计算转移到接近数据的地方获得完整的闪存潜力
  - 需要控制整个软件栈，无法单独靠存储解决
- 基本上来说，存储阵列只能共享闪存的容量而不是性能
  - 即便下一代横向扩展，PCIe网络或者NVMe over fabric
- 具备内存带宽级别的共享存储是业界的革命性典范
  - 获得DRAM级别的吞吐量，同时保持共享闪存的容量

真实生产的表现：数据仓库

IOStat by File Type (per Second)

- Total Reads includes all Filetypes: Data File, Temp File, Archive Log, Backups
- Total Writes includes all Filetypes: Data File, Temp File, Log File, Archive Log,

	Reads MB/sec			Writes MB/sec			
I#	Total	Data File	Temp File	Total	Data File	Temp File	Log File
1	101.83	100.95	0.53	5.76	4.82	0.64	0.29
2	79.02	78.48	0.22	11.77	11.08	0.29	0.38
3	2,297.15	2,260.51	36.14	44.13	4.64	36.43	3.05
4	92,121.10	92,114.47	6.58	14.97	7.86	6.94	0.16
5	124.94	112.09	12.52	44.38	18.56	21.97	3.84
6	429.20	428.81	0.06	6.29	5.54	0.28	0.45
7	1,421.06	1,355.95	64.78	102.93	11.26	90.39	1.27
8	1,297.02	1,288.87	7.77	23.49	14.67	8.47	0.34
9	868.18	864.32	3.53	14.07	8.10	5.56	0.40
10	377.53	375.37	1.84	19.05	15.23	2.87	0.94
11	760.11	754.76	5.02	20.58	12.34	5.20	3.01
12	293.61	292.94	0.34	13.99	13.26	0.45	0.26
13	290.93	229.06	61.54	109.18	14.16	94.63	0.37
14	438.63	430.51	7.79	40.91	31.95	7.28	1.67
Sum	100,900.31	100,687.08	208.65	471.51	173.48	281.38	16.45
Avg	7,207.16	7,191.93	14.90	33.68	12.39	20.10	1.17

	Reads MB/sec			Writes MB/sec			
I#	Total	Data File	Temp File	Total	Data File	Temp File	Log File
1	18,896.60	17,787.23	902.40	1,543.97	197.69	782.45	373.87
2	15,658.25	14,990.62	562.30	1,032.27	169.61	602.59	177.79
3	21,373.11	20,513.77	767.43	1,243.02	198.66	771.23	186.56
4	6,337.58	6,219.90	60.49	345.34	123.30	59.03	111.72
5	15,987.37	15,335.62	573.08	1,180.85	103.66	853.72	151.01
6	17,555.61	16,802.92	669.61	1,101.08	127.34	732.63	164.50
7	5,409.36	5,400.44	3.32	31.17	19.84	3.34	5.27
8	6,200.36	6,173.69	20.23	41.92	6.29	21.08	11.49
Sum	107,418.23	103,224.20	3,558.85	6,519.61	946.38	3,826.07	1,182.21
Avg	13,427.28	12,903.02	444.86	814.95	118.30	478.26	147.78

- 实际上由于数据启用HCC压缩，真正的数据扫描有效带宽是物理带宽*压缩比
- 实际数据扫描带宽在500GB/s以上
- 任何不具备offload到存储进行share nothing处理的Oracle数据库绝不可能获得这样的性能！！！！

数据仓库负载中Offload的效果

IOStat by Function (per Second)

- Total Reads includes all Functions: Buffer Cache, Direct Reads, ARC
- Total Writes includes all Functions: DBWR, Direct Writes, LGWR, AR

	Reads MB/sec			Writes MB/sec		
I#	Total	Buffer Cache	Direct Reads	Total	DBWR	Direct W
1	2,144.08	389.30	1,510.84	1,557.61	79.66	7
2	809.47	156.19	530.38	1,018.92	143.15	5
3	1,560.66	302.50	1,120.79	1,257.75	148.02	7
4	361.08	139.72	153.27	347.00	113.17	
5	1,071.04	297.59	649.95	1,158.09	76.66	7
6	1,235.88	172.95	928.87	1,102.52	73.96	6
7	33.19	13.81	12.49	31.21	1.93	
8	125.70	29.79	86.38	41.85	3.41	
Sum	7,341.10	1,501.86	4,992.95	6,514.96	639.97	3,5
Avg	917.64	187.73	624.12	814.37	80.00	4

Back to I/O Statistics
Back to Top

IOStat by File Type (per Second)

- Total Reads includes all Filetypes: Data File, Temp File, Archive Log,
- Total Writes includes all Filetypes: Data File, Temp File, Log File, Arc

	Reads MB/sec			Writes MB/sec		
I#	Total	Data File	Temp File	Total	Data File	Temp File
1	18,896.60	17,787.23	902.40	1,543.97	197.69	782.45
2	15,658.25	14,990.62	562.30	1,032.27	169.61	602.59
3	21,373.11	20,513.77	767.43	1,243.02	198.66	771.23
4	6,337.58	6,219.90	60.49	345.34	123.30	59.03
5	15,987.37	15,335.62	573.08	1,180.85	103.66	853.72
6	17,555.61	16,802.92	669.61	1,101.08	127.34	732.63
7	5,409.36	5,400.44	3.32	31.17	19.84	3.34
8	6,200.36	6,173.69	20.23	41.92	6.29	21.08
Sum	107,418.23	103,224.20	3,558.85	6,519.61	946.38	3,826.07
Avg	13,427.28	12,903.02	444.86	814.95	118.30	478.26

cell physical IO bytes eligible for predicate offload	373,514,175,053,824
cell physical IO bytes saved by storage index	59,690,933,190,656

- Smartscan减少93%存储网络流量
- Storage Index减少16%物理介质IO

OLTP应用能否利用这个offload特性？
让我们看看一些真实AWR

	Per Second
DB Time(s):	93.7
DB CPU(s):	19.3
Redo size (bytes):	9,803,097.2
Logical read (blocks):	4,846,931.9
Block changes:	55,708.7
Physical read (blocks):	52,115.4
Physical write (blocks):	2,705.5
Read IO requests:	11,167.6
Write IO requests:	1,453.4
Read IO (MB):	407.2
Write IO (MB):	21.1
Global Cache blocks received:	1,215.4
Global Cache blocks served:	1,083.8
User calls:	109,759.7
Parses (SQL):	3,481.8
Hard parses (SQL):	3.3
SQL Work Area (MB):	136.5
Logons:	4.0
Executes (SQL):	28,786.0
Rollbacks:	695.8
Transactions:	3,174.8

IOStat by Function summary

- 'Data' columns suffixed with M,G,T,P are in multiples of 1024
- ordered by (Data Read + Write) desc

Function Name	Reads: Data	Reqs per sec	Data per sec
Direct Reads	1.7T	295.10	247.987
Buffer Cache Reads	1.1T	10825.22	158.311
Others	176.2G	73.94	25.202M
LGWR	23M	0.26	.003M
DBWR	0M	0.00	0M
RMAN	93G	28.88	13.302M
Direct Writes	2G	0.30	.284M
TOTAL:	3T	11223.69	445.088

init.ora Parameters

Parameter Name	
_PX_use_large_pool	TRUE
_add_col_optim_enabled	FALSE
_b_tree_bitmap_plans	FALSE
_bloom_filter_enabled	FALSE
_cgs_health_check_in_reconfig	FALSE
_cleanup_rollback_entries	2000
_clusterwide_global_transactions	FALSE
_cursor_obsolete_threshold	200
_datafile_write_errors_crash_instance	FALSE
_deferred_log_dest_is_valid	TRUE
_gc_defer_time	0
_gc_policy_time	0
_gc_read_mostly_locking	FALSE
_gc_undo_affinity	FALSE
_ktb_debug_flags	8
_lm_lms_priority_dynamic	FALSE
_lm_sync_timeout	1200
_lm_tickets	5000
_memory_imm_mode_without_autosga	FALSE
_mv_refresh_use_stats	FALSE
_optimizer_join_elimination_enabled	TRUE
_optimizer_mjc_enabled	FALSE
_optimizer_squ_bottomup	TRUE
_optimizer_use_feedback	FALSE
_pred_move_around	FALSE
_resource_manager_always_off	TRUE
_resource_manager_always_on	FALSE
_serial_direct_read	NEVER

- 理论上OLTP会用不到，如果应用写的足够好都是合理的index range/unique SCAN，而且不提供报表
- 测试中也可以，因为测试案例往往只是少量几种交易
- 但是实际生产中很难，实际上的应用大部分都是混合负载HTAP(Translytic)
- 左图，运营商核心OLTP，利用隐含参数关闭非并行查询direct path read情况下，仍然direct path read数据量大于buffer cache读数据量

小结

- 在Exadata上基于LibCell的IO实现解决了传统OS下读写协议不够智能和不够高效的问题：
  - 减少了软件调用层次并让面向数据库应用层的IO优先级成为现实
  - 结合Exadata上Cell Disk->Grid Disk->ASM Disk+iDB实现，也让实现存储智能的share nothing MPP数据处理成为可能
- Exadata上运行的Oracle RAC在传统Share disk架构上增加了Share Nothing MPP的处理，兼具两种架构优点
  - 通过Smart Scan、Storage index实现将数据库代码发送到离数据近的地方就近执行
  - Smart Scan在实际的生产中不仅在DW/OLAP应用中有不可替代的作用，对于实际的OLTP由于大部分存在混合负载场景，也能实现性能的加速

揭开支撑Oracle运行的神器Exadata的美丽面纱
原理和技术实现介绍系列（一）

刘建军，资深解决方案工程师

公益讲座11点准时开始，请大家先浏览云技术微信公众号技术文章
资料会在各群同步发布，已入群客户请勿重复入群！

扫码加入:
19c新特性讲座群

欢迎关注:
甲骨文云技术公众号