Exadata Architecture Deep dive

ORACLE®

Leon Wu
Dongxin.wu@Oracle.com
Exadata Solution Architect

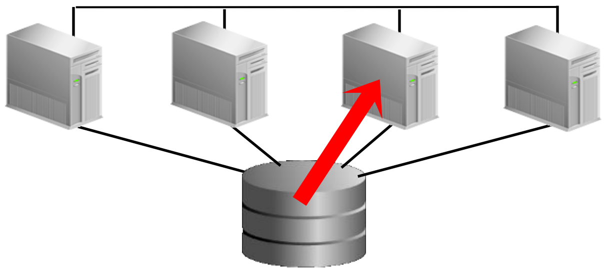
Agenda

- Exadata Platform Overview
- Exadata Component
- Exadata Architecture detail
- Smart Flash Cache Deep Dive

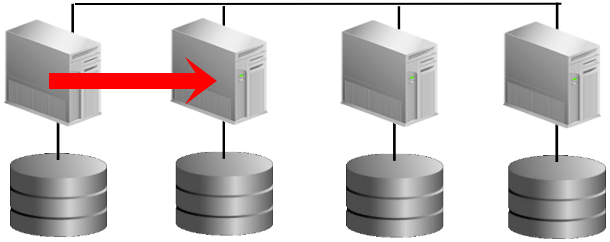
Shared Disk vs. Shared Nothing

- The Shared Disk vs. Shared Nothing debate has raged for years
- Exadata brings a fresh perspective to this debate
- Basic differences between approaches are simple

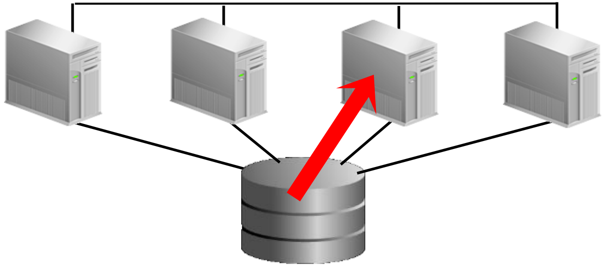
Shared disk ships data to where code is



Shared nothing ships code to where data is



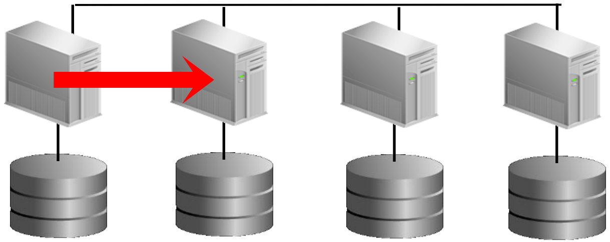
Shared Disk



- **Shared Disk wins when the amount of data shipped is relatively small**
  - Data is shipped and then cached at requesting node
  - Automatically replicates (in cache) small tables and warm blocks
    - Caching avoids future network and disk traffic
      - Rapidly and dynamically adjusts to changing workloads
    - Data transfer uses less I/O protocols
- **Shared disk is efficient and transparent for OLTP**
  - Typical query runs in single node with no inter-node communication
  - High Availability is easy to implement

X **Large IO in OLAP/DW workload may hit Storage bottle neck**

Shared Nothing



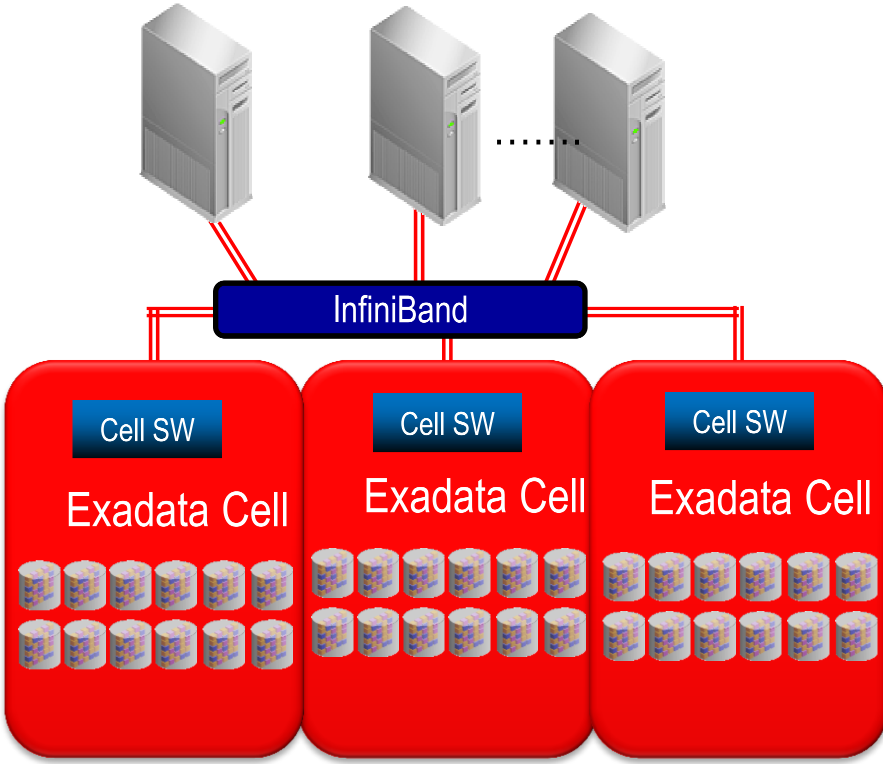
- **Shared nothing has the advantage of attaching disks directly to host nodes**
  - Bypasses potential monolithic storage and network bottlenecks
  - Enables fast scans of bulk data
  - Automatically parallel execution

X **For OLAP still cause problem**
  - Distribution key and none co-location operation
  - Non linear performance lost with node down or need cold standby nodes

X **Shared Nothing is very expensive and slow for OLTP**
  - Typical query requires execution by multiple nodes or broadcast to all nodes
  - Foolish parallel execution
  - High Availability is hard to implement and/or cause none linear performance loose

Shared Disk with Benefits of Shared Nothing

- Exadata delivers Shared Disk with the benefits of Shared Nothing
- None of the drawbacks
  - Easy HA with linear performance with nodes number and no cold standby
  - No distribution key depends
  - Controllable parallel execution on DB node
  - Automatically parallel at storage level
- Exadata ships code to data for scan operations
  - Processes bulk data directly as it comes off disk
- Exadata ships data to code for OLTP operations
  - Efficiently runs any application with no changes, including ultra-complex ones



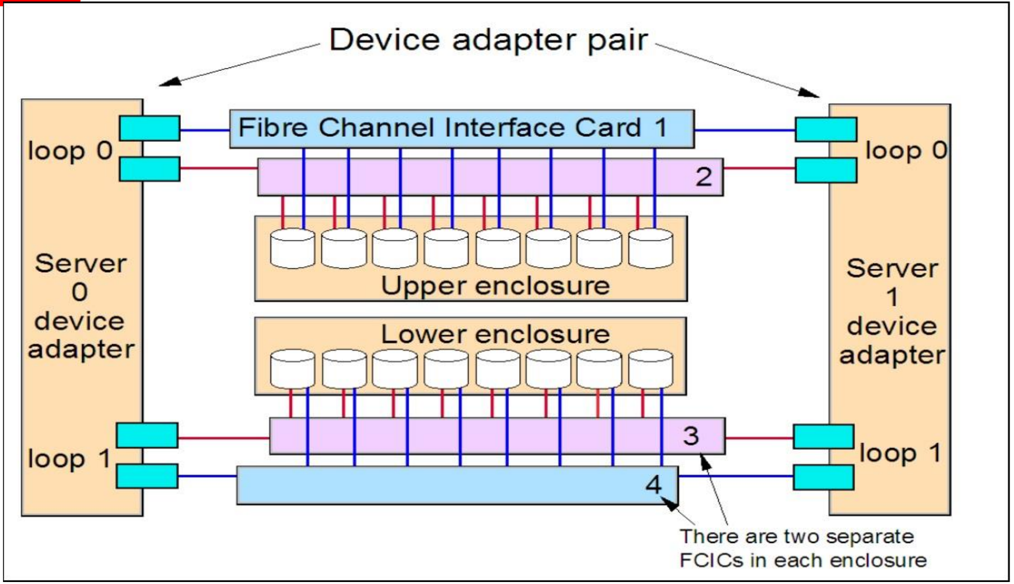
Agenda

- Exadata Platform Overview
- Exadata Component
- Exadata Architecture detail
- Smart Flash Cache Deep Dive

Hardware components overview

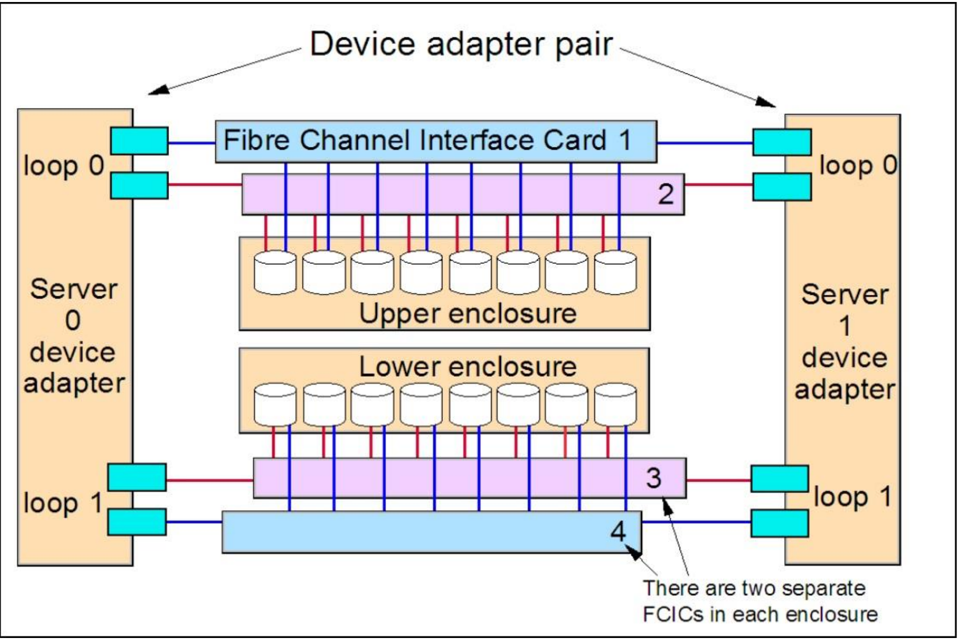
http://oracle.com.edgesuite.net/producttours/3d/exadata-x4-2/index.html

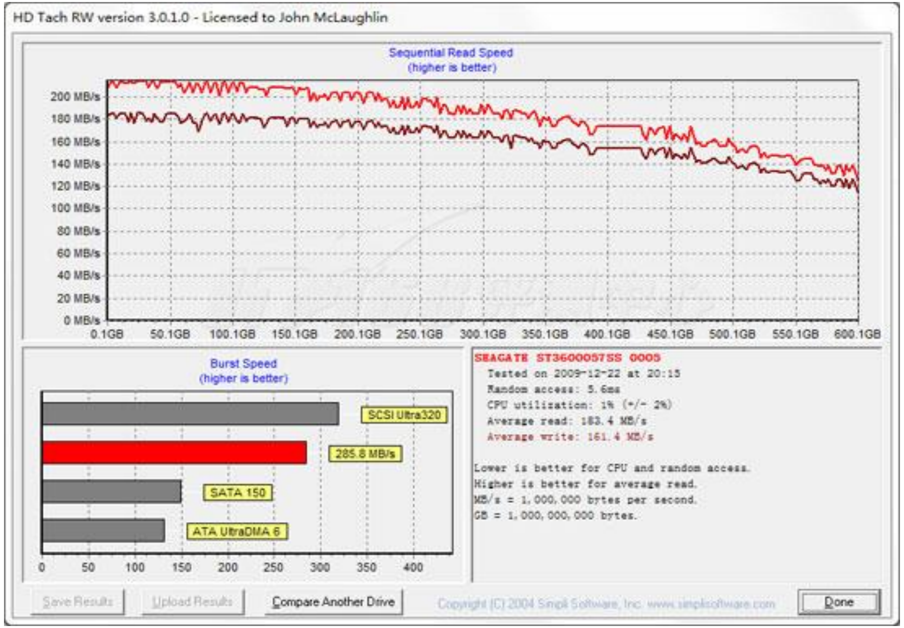
The real bottle neck: DA Channel inside the storage



- Most time more than 30 back-end disk share one disk adaptor(DA) channel(4/8Gbs FC or 6Gbs SAS)
- Some times more than 200 disks share on DA channel
- Shared hardware channel also cause latency

DA Channel is not enough for modern disk through-put



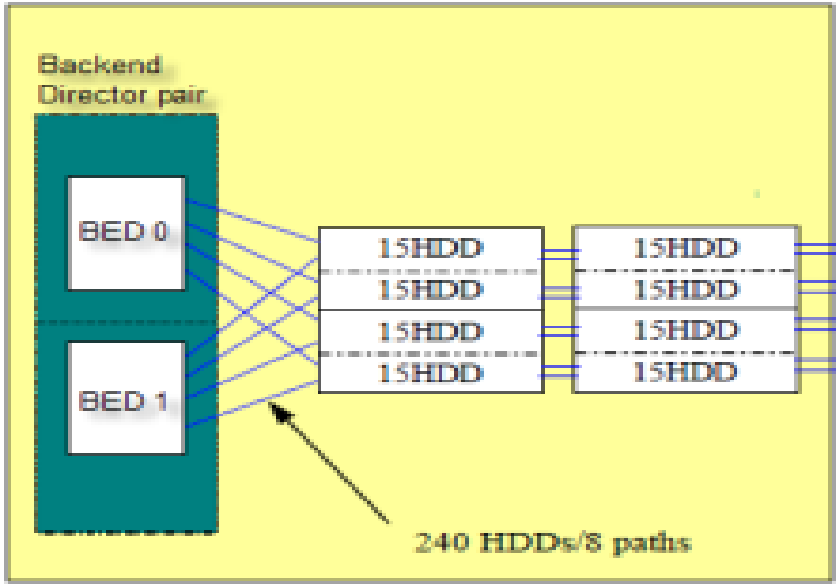


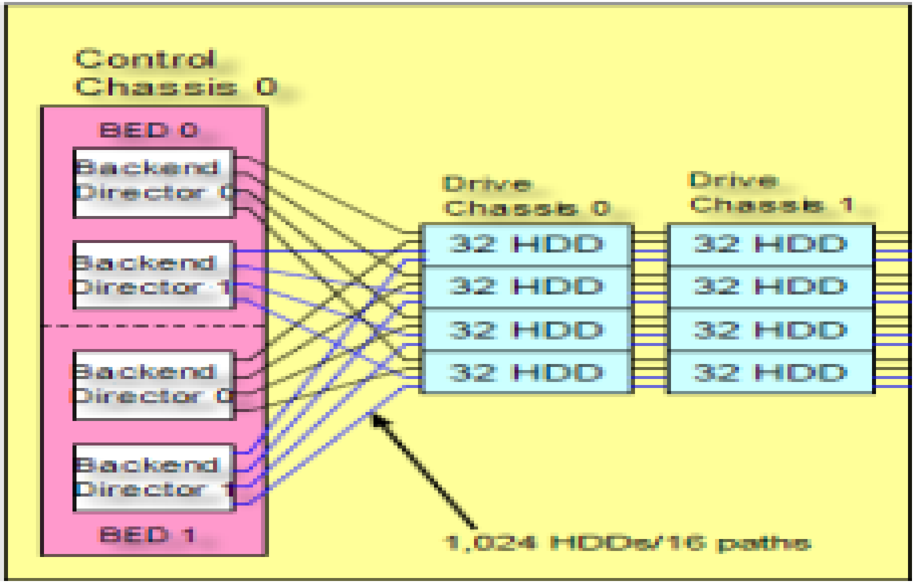
One high-end SAN storage
Bandwidth for one disk<50MB/Sec

Server level 15KRPM through-put

DA Channel is not enough for modern disk through-put

Another vender's high-end storage

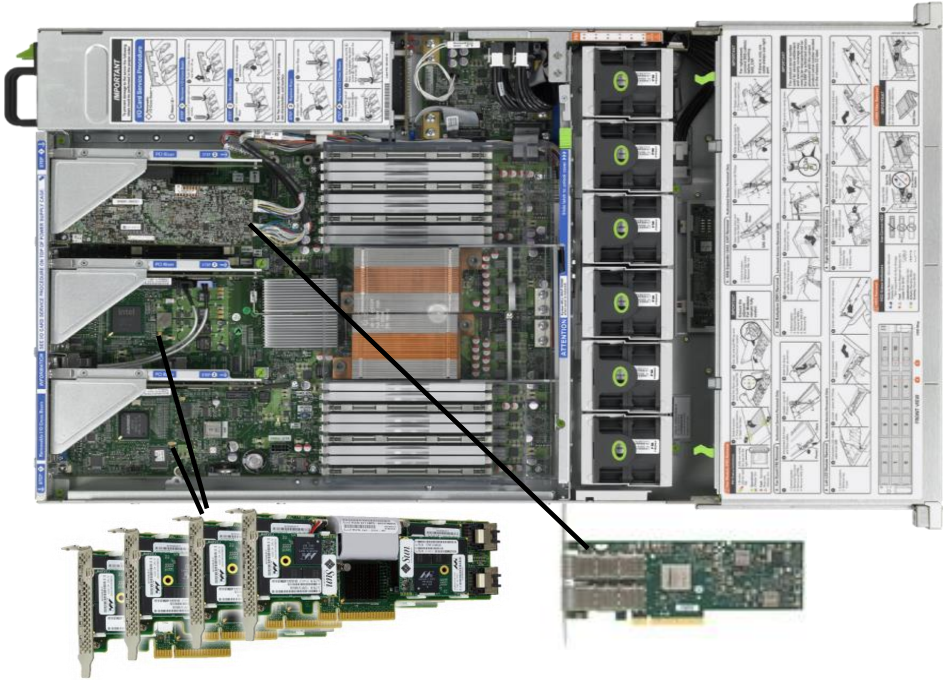




DA Bandwidth per disk<15MB/Sec

DA Bandwidth per disk<15MB/Sec

How Exadata solve DA bottle neck: CELL Architecture

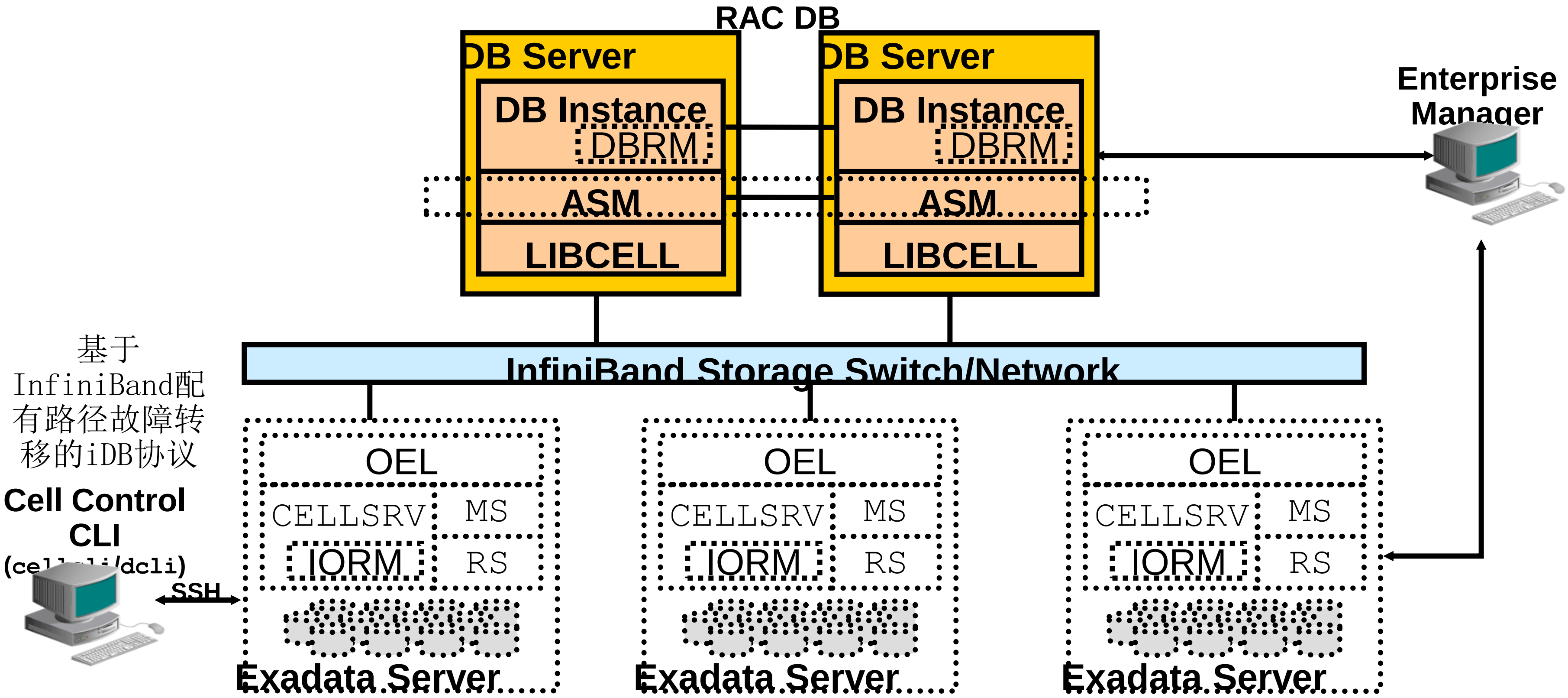


- 12 SAS disk connect to a PCIE 3.0 4X SAS controller
- Bandwidth for one disk is 160MB/s full duplex
- Point to Point connection, no SAS expander
- 4 PCIE 2.0 8X Flash card with own controller
  - No bandwidth share with disks
  - No IOPS share with DA
- Upper link is 40Gbs(x2) per cell

Agenda

- Exadata Platform Overview
- Exadata Component
- Exadata Architecture detail
- Smart Flash Cache Deep Dive

Exadata software architecture
- DB Server



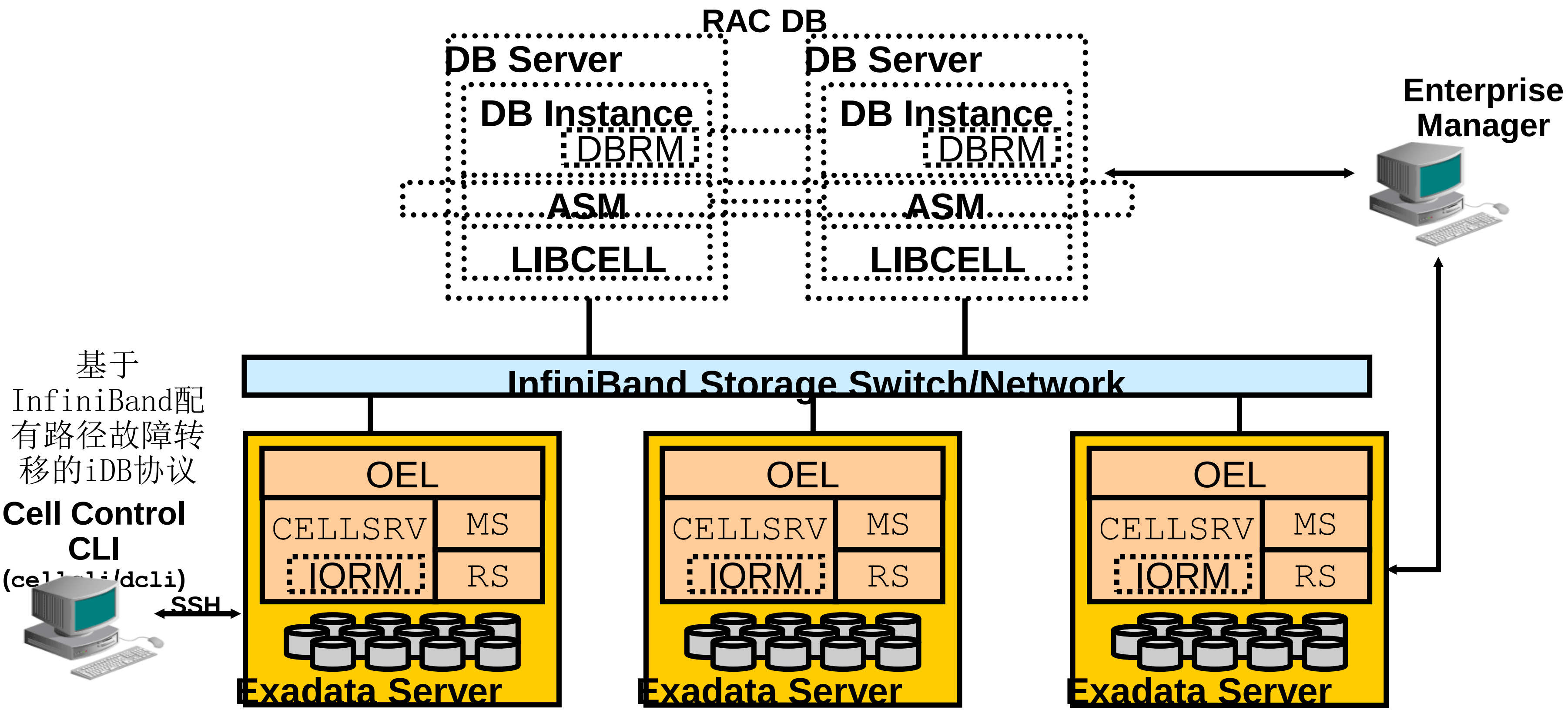
Exadata Software Architecture

– Cell server

基于InfiniBand配有路径故障转移的iDB协议

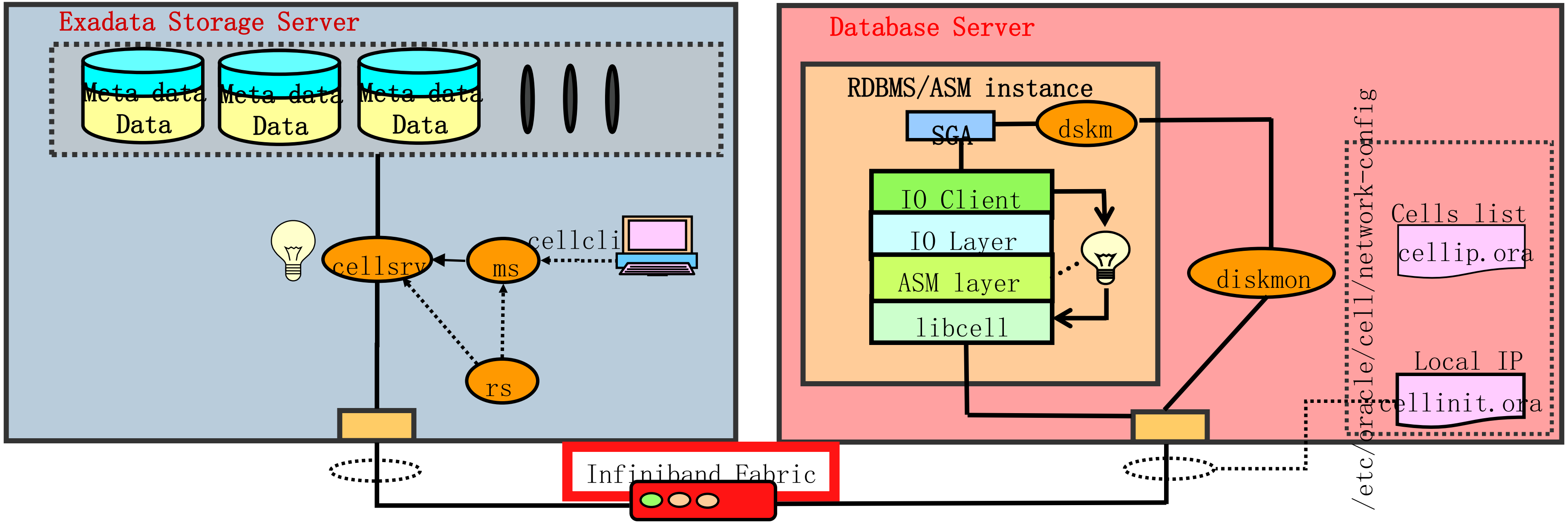
Cell Control CLI

(cellcli/dcli)



 | | Insert Information Protection Policy Classification from Slide 13

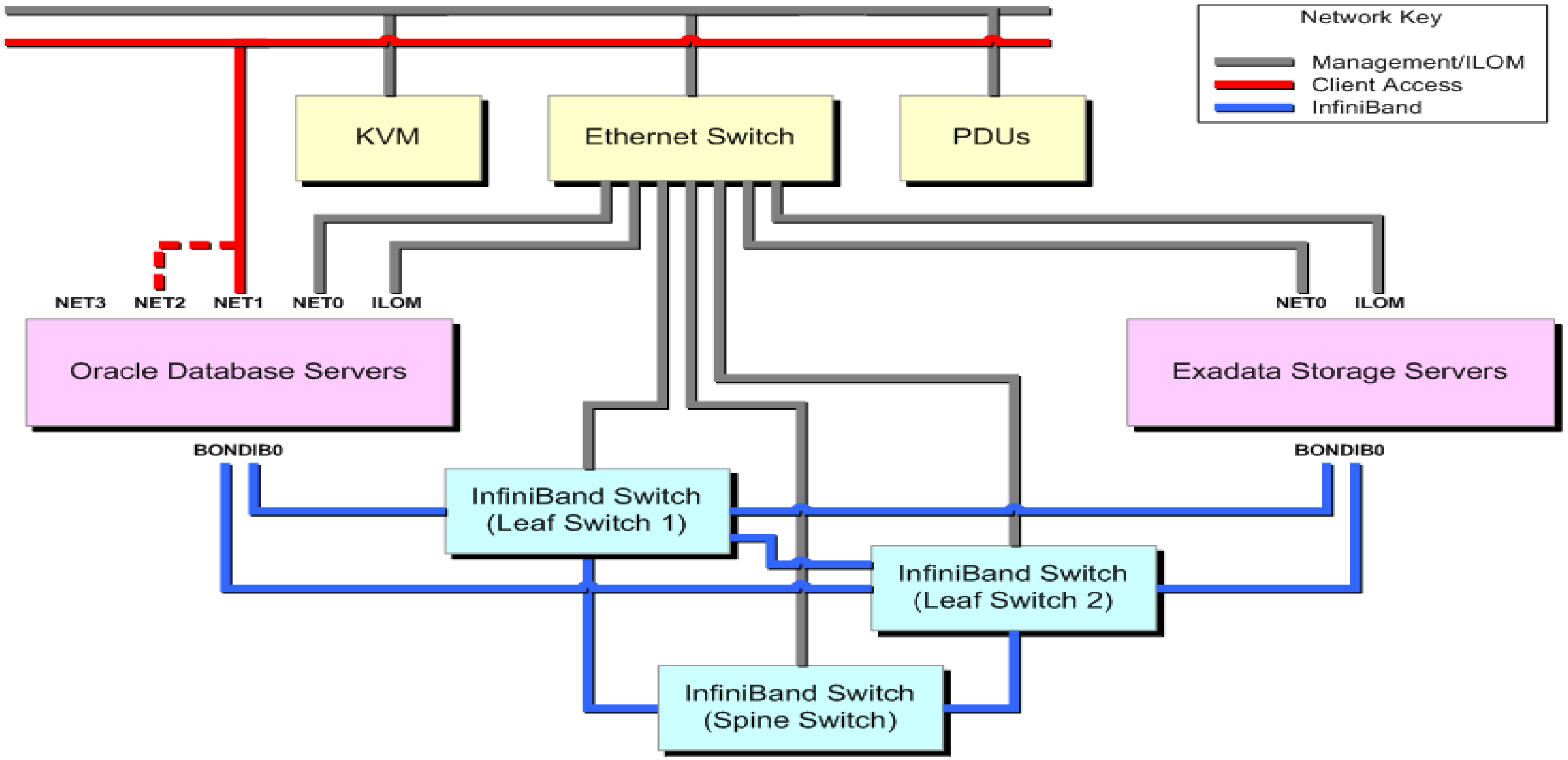
Exadata Storage Server Processes



- Cellcli 管理命令
- Management Server (MS): grid disk的创建、H/W改变、SNMP陷阱警报、email通知、阈值控制及服务器管理
- Restart Server (RS) : CELLSRV和MS的监控, 重启。

  Insert Information Protection Policy Classification from Slide 13

Exadata Network Architecture



 | |

InfiniBand Network Addresses <X4

	X3-2			X3-8	Storage Expansion X3-2		
	Full	Half	Quarter / Eighth	Full	Full	Half	Quarter
bondib0 for Database Servers	8	4	2	8	-	-	-
bondib0 for Exadata Storage Servers	14	7	3	14	18	9	4
Total	22	11	5	22	18	9	4

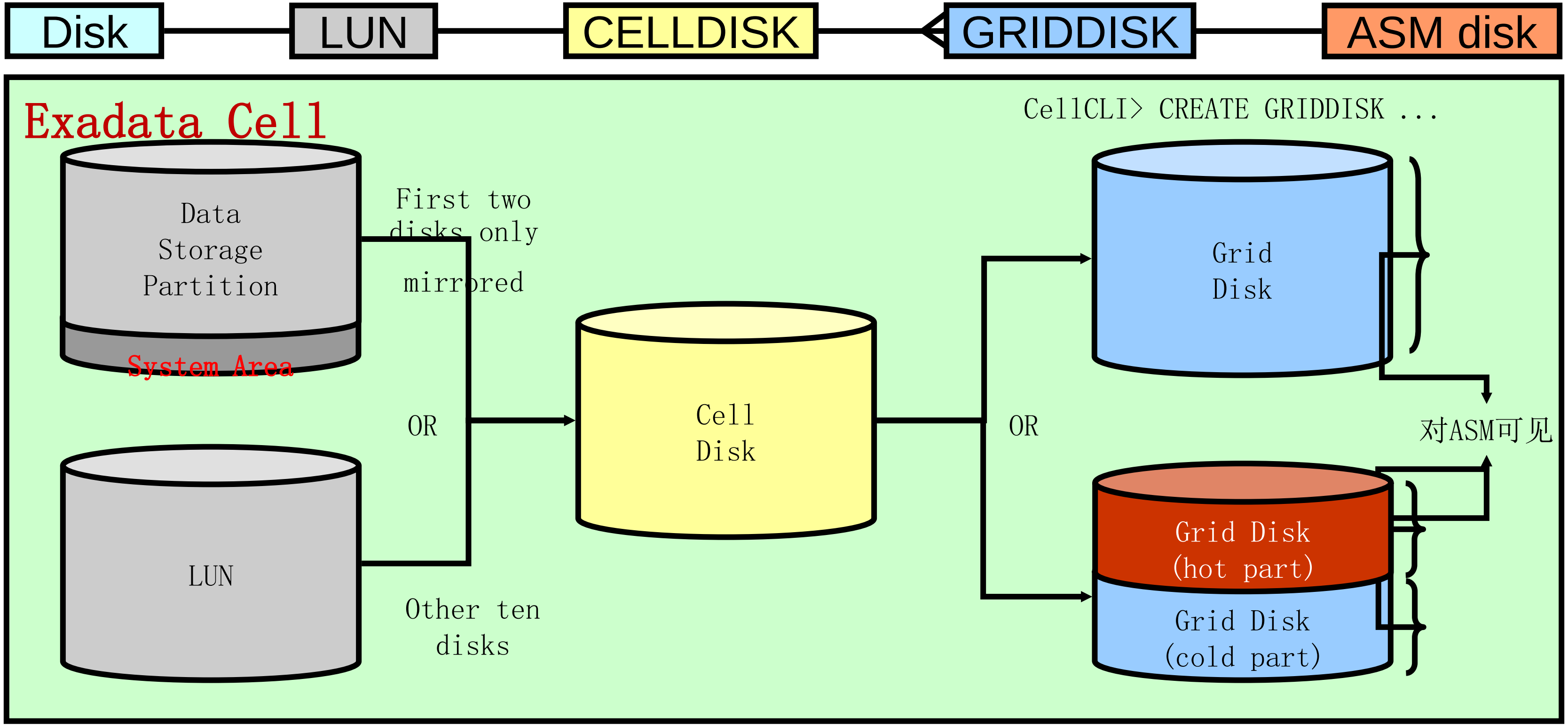
ORACLE

InfiniBand Network Addresses Begin from X4

	X4-2			X4-8	Storage Expansion X4-2		
	Full	Half	Quarter / Eighth	Full	Full	Half	Quarter
Ib0 and ib1 for Database Servers	16	8	4	16	-	-	-
Ib0 and ib1 for Exadata Storage Servers	28	14	6	28	36	18	8
Total	44	22	10	44	36	18	8

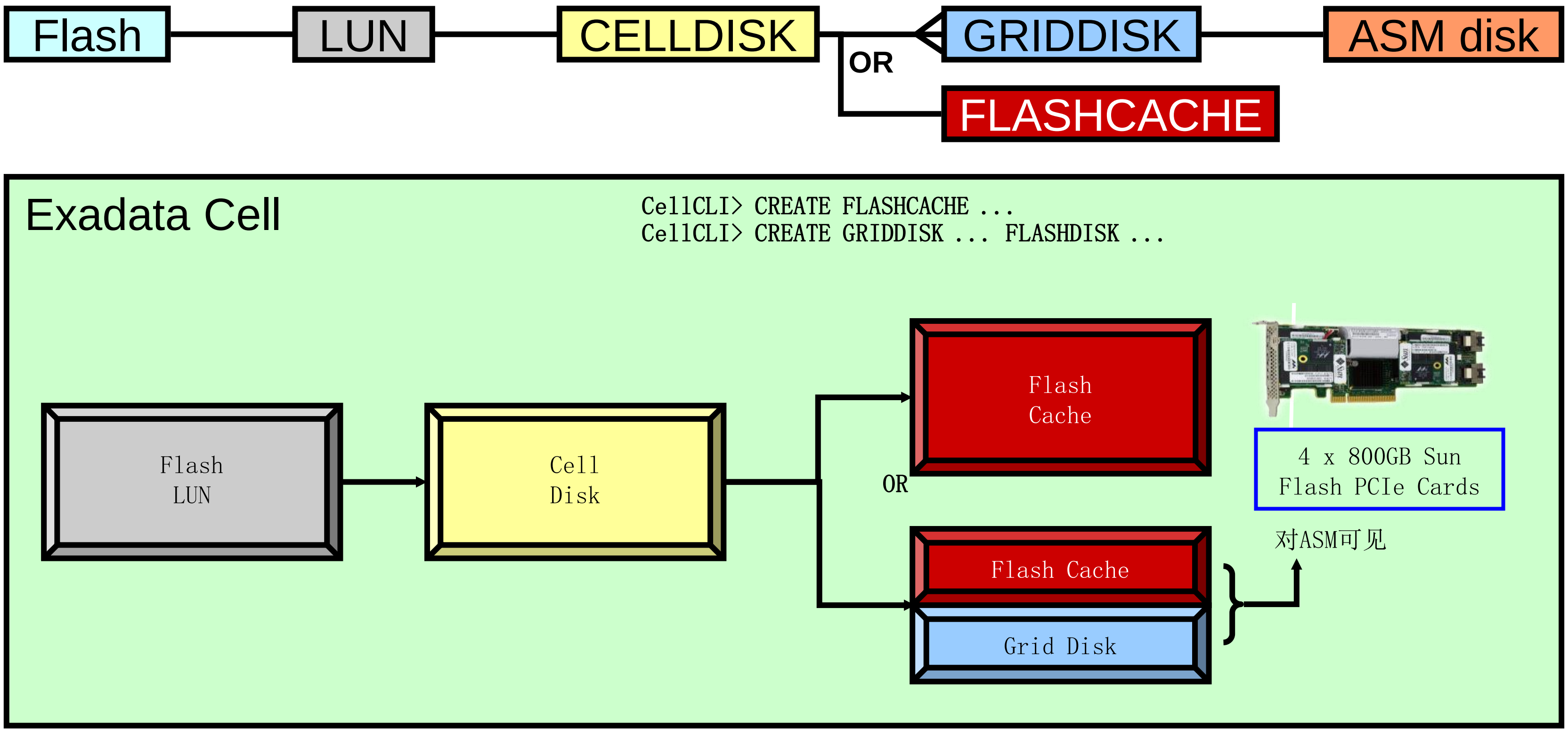
ORACLE

Hard disk layout and relation



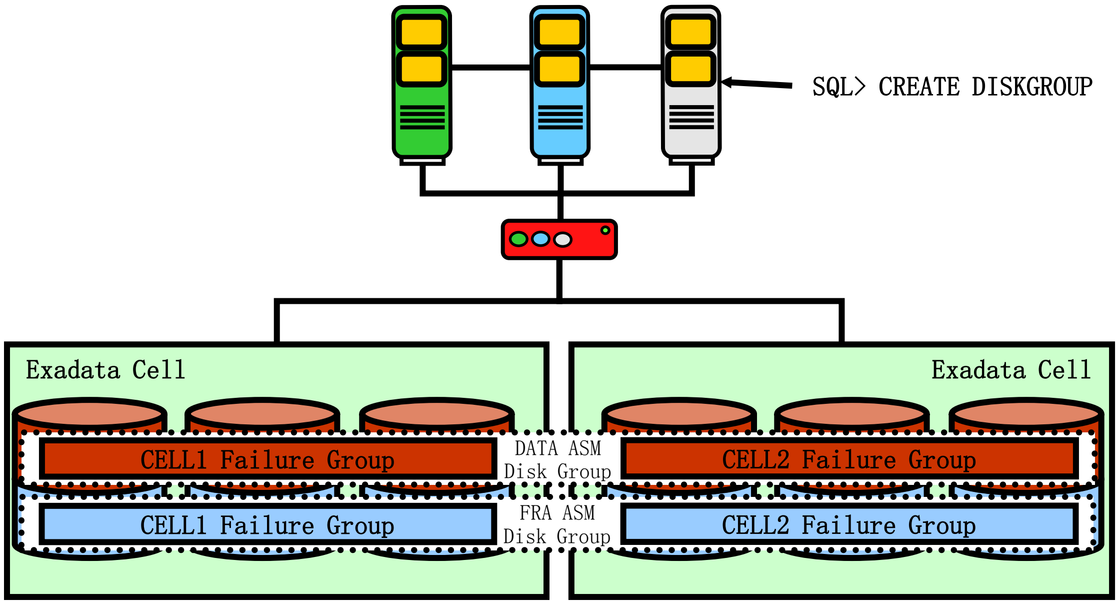
 | | Insert Information Protection Policy Classification from Slide 13

Flash Cache Layout



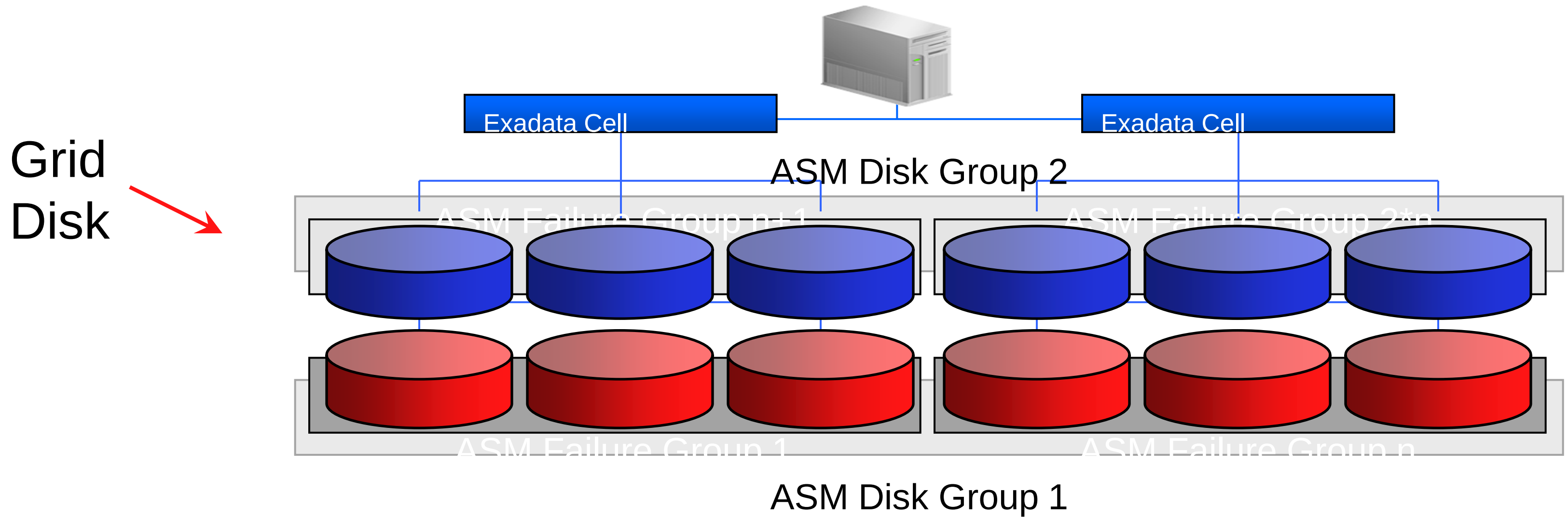
 | | Insert Information Protection Policy Classification from Slide 13

DiskGroup Architecture



- With normal redundancy, data will be distribute in 2 failure group
- Any 2 different failure group will come from 2 different storage cells
- Can support triple mirror with high redundancy

Database Machine



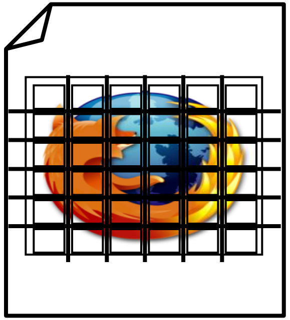
- 3 grid disks per cell disk
  - Outer disk for DATA, inner disk for RECO
  - Middle for OCR/Voting and/or DBFS
- 3 ASM disk groups across all cells

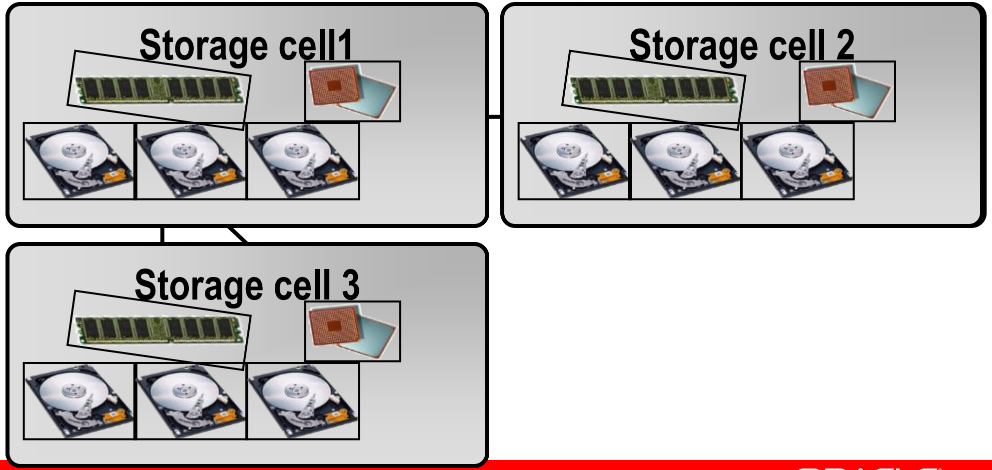
- Within each disk group
  - One failure group per cell
  - ASM mirroring / Triple Mirroring

ORACLE

Exadata Internal ASM file allocation

- Automatically distribute data to all cell and disk (Default)
- AU_SIZE is 4MB on Exadata
- Every Grid disk have <10 partner disk (Current is 8)





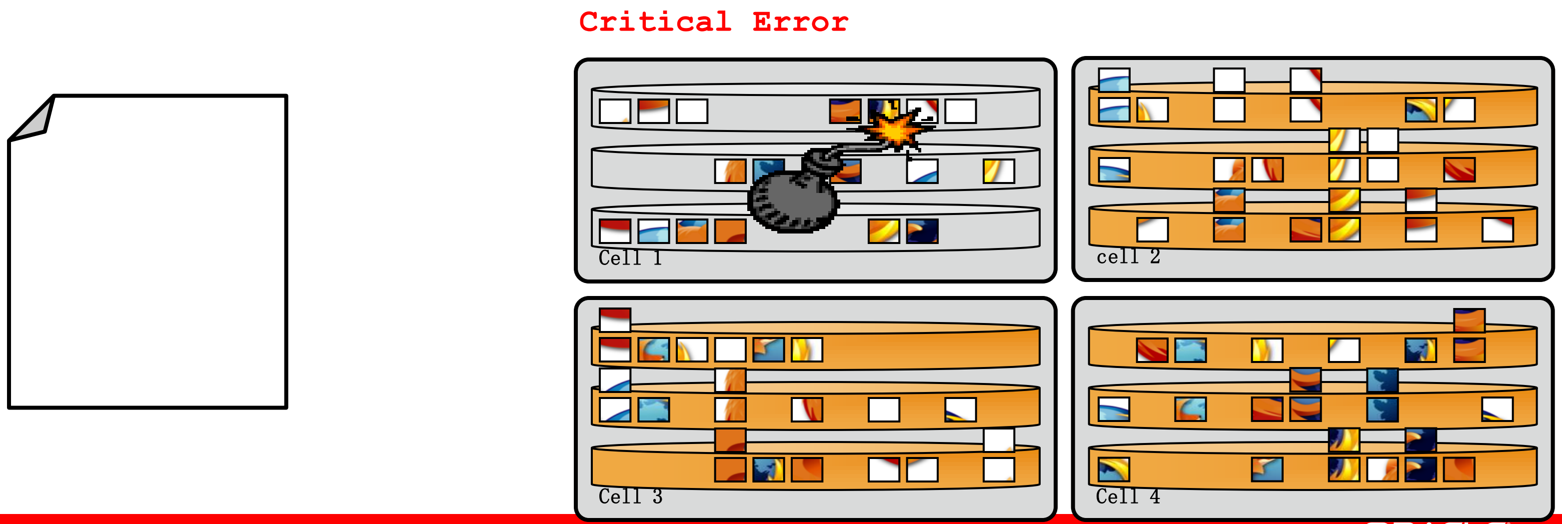
  Insert Information Protection Policy Classification from Slide 13

Single disk fail rebalance

- Using proactive Quarantine, do not wait for disk_repair_time
- One single disk have 8 partner disk
  - One disk fail may cause max 8*7=56 disk to do the rebalance theoretical
  - In real world will Around 40-50 disks to rebalance
- Very fast
- With WBFC, very little impact about SLA
  - Total throughput consume1GB-2GB/s，1/4 rack >20GB/s
  - 500-1000IOPS，1/4 configuration IOPS>100K(with WBFC)

If one cell can not be OK in disk_repair_time: Rebalance

- After one cell fail, ASM wait for disk_repair_time and do the rebalance
- It is not a single disk to disk copy, but multi to multi copy
- Only reblance the space with datafile!!! Not the whole disk

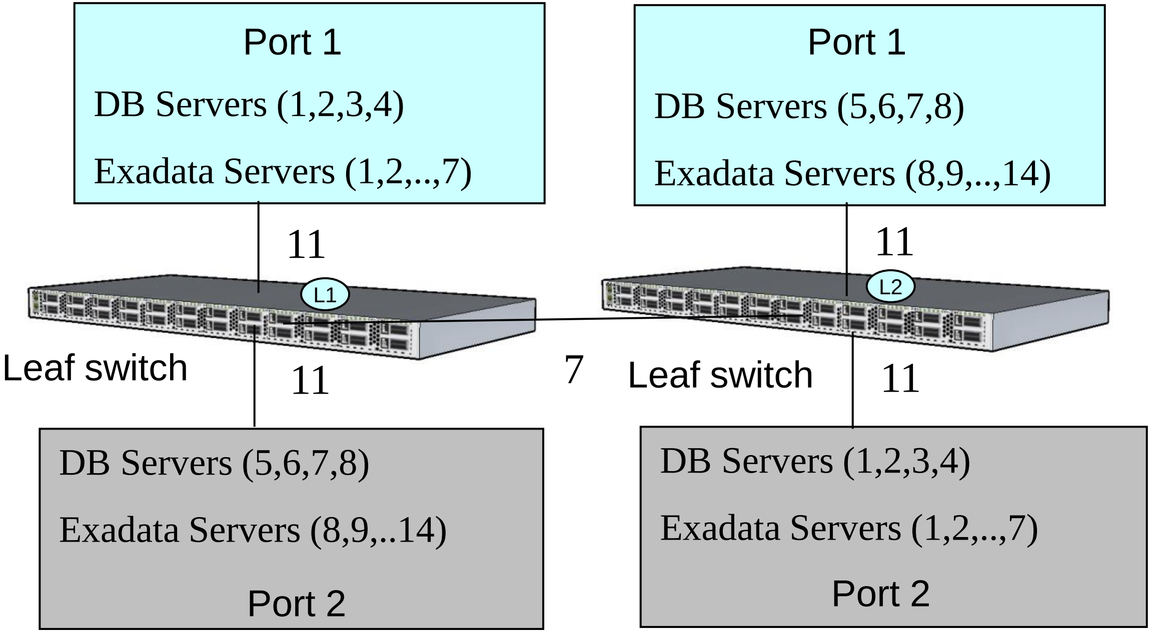


InfiniBand Network

- Unified InfiniBand Network
  - Storage Network
  - RAC Interconnect
  - External Connectivity (optional)
- High Performance, Low Latency Network
  - 80 Gb/s bandwidth per link (40 Gb/s each direction)
  - SAN-like Efficiency (Zero copy, buffer reservation)
  - Simple manageability like IP network
- Protocols
  - Zero-copy Zero-loss Datagram Protocol (ZDP RDSv3)
    - Low CPU overhead (Transfer 3 GB/s with 2% CPU usage)
  - Internet Protocol over InfiniBand (IPoIB)
    - Looks like normal Ethernet to host software (tcp/ip, udp, http, ssh,…)

ORACLE

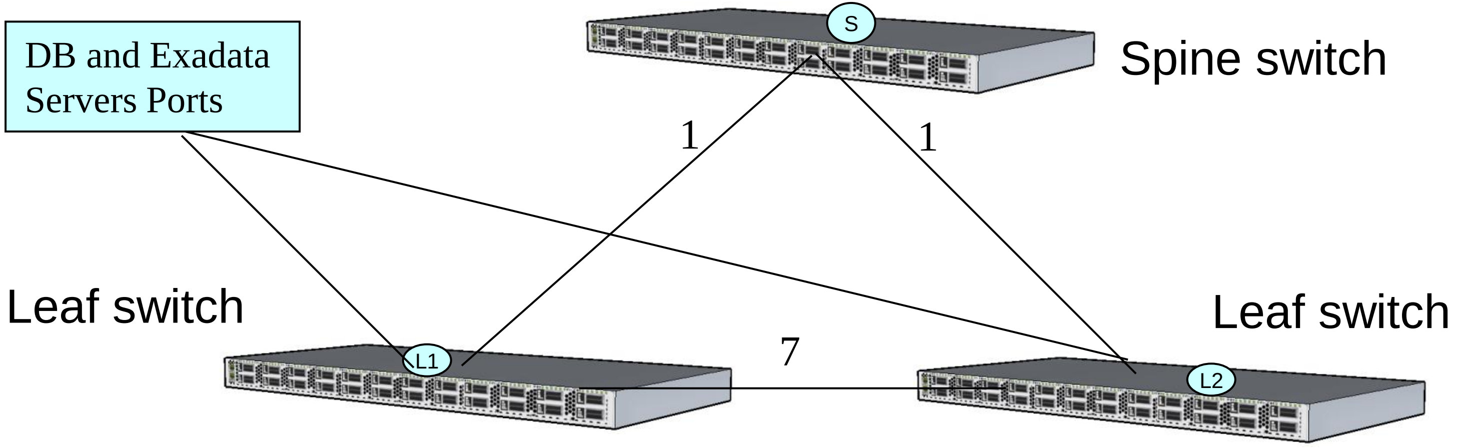
X3-2 Full Rack – Server to Leaf Switch



- Each Server HCA connects to both leaf switches for redundancy
- Active & Passive ports balanced between leaf switches
- Full Bandwidth even if switch fails
- Connections pre-wired at factory
- Half, Quarter, and Eighth rack follow same cabling methodology except they have fewer servers

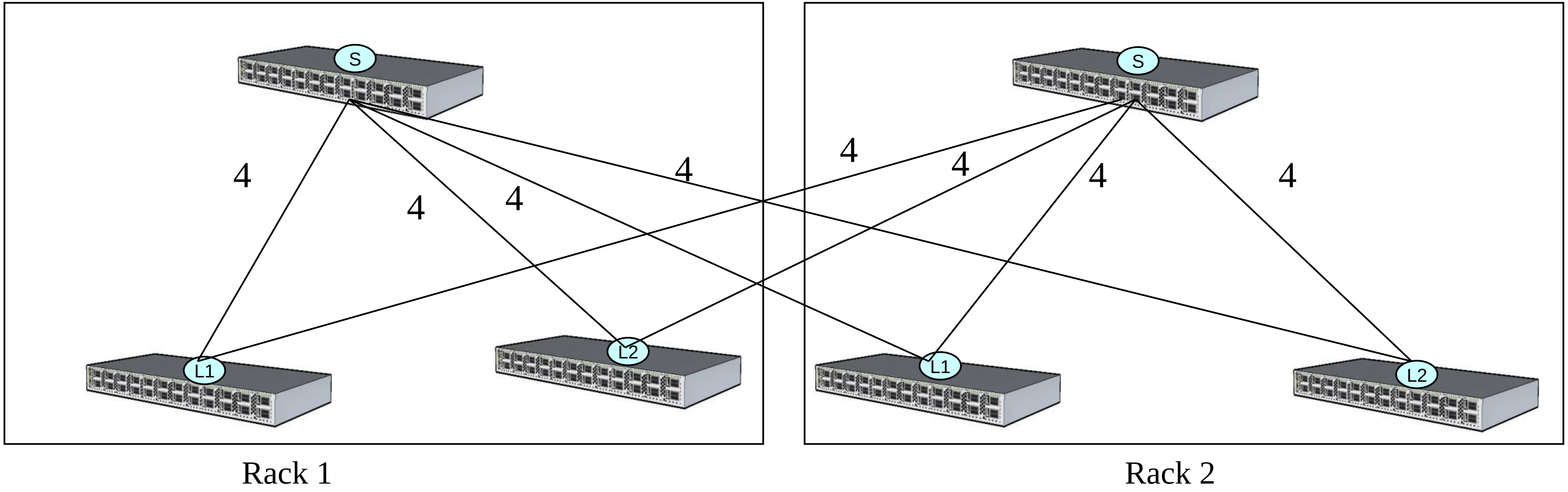
ORACLE

Spine and Leaf InfiniBand Switch



- Full and Half Racks use 3rd switch (S) as "spine" switch for expansion to multiple racks
- Connect each leaf switch to spine switch (1 links wide)
- Interconnect "leaf" switches with each other (7 links wide)
- Pre-wired at factory

Two Rack Case – Fat Tree Topology



- Remove inter-switch links between leaf switches (L1,L2) in each rack
- Connect each leaf switch to both the spine switches (4 links per pair)

  | Insert Information Protection Policy Classification from Slide 13

InfiniBand Network Addresses

	X3-2			X3-8	Storage Expansion X3-2		
	Full	Half	Quarter / Eighth	Full	Full	Half	Quarter
bondib0 for Database Servers	8	4	2	8	-	-	-
bondib0 for Exadata Storage Servers	14	7	3	14	18	9	4
Total	22	11	5	22	18	9	4

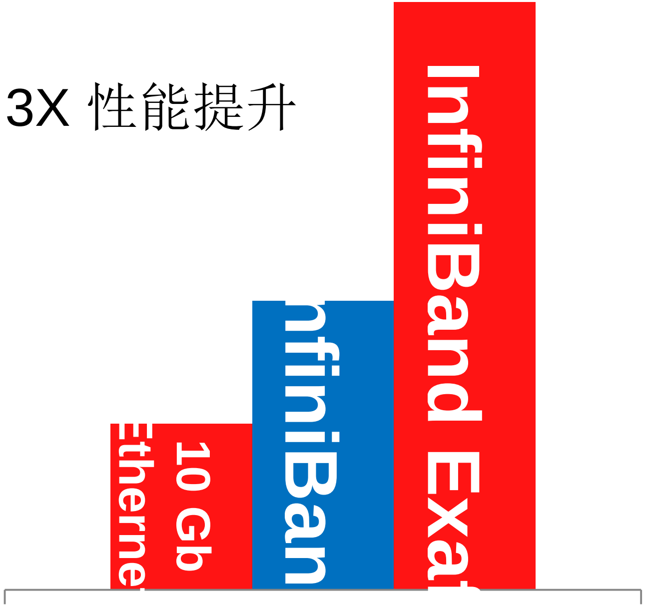
ORACLE

InfiniBand Network Addresses for X4

	X4-2			X4-8	Storage Expansion X4-2		
	Full	Half	Quarter / Eighth	Full	Full	Half	Quarter
Ib0 and ib1 for Database Servers	16	8	4	16	-	-	-
Ib0 and ib1 for Exadata Storage Servers	28	14	6	28	36	18	8
Total	44	22	10	44	36	18	8

Exafusion 直接硬件访问协议

3X 性能提升

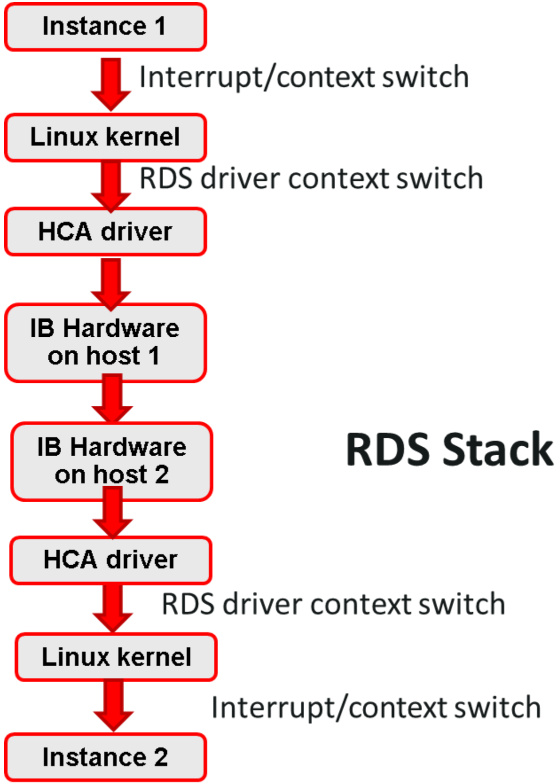


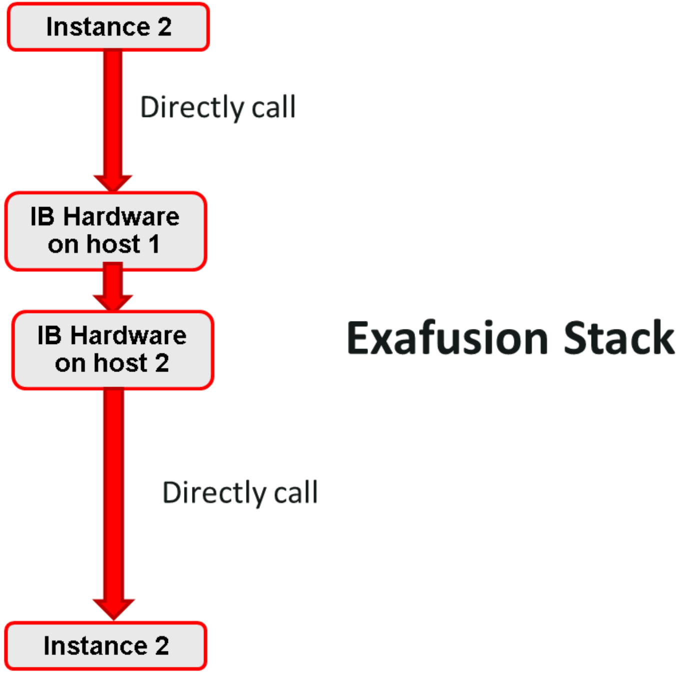
8K OLTP Block Transfers/sec

- InfiniBand 具有很高的吞吐量和低延迟
  - 但是传统的OS网络协议栈导致每个消息传输效率不高
- Exafusion 重新实现了RAC Cache Fusion
- 数据库直接调用 InfiniBand 硬件
  - 绕过OS的网络协议栈，中断，调度
- 支持除 V1 和 V2外所有硬件
- 需要数据库 12.1.0.2 BP1以上，推荐 12.1.0.2.BP13

 |

Exadata独有的Cachefusion协议栈

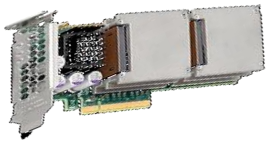




Agenda

- Exadata Platform Overview
- Exadata Component
- Exadata Architecture detail
- Smart Flash Cache Deep Dive

Flash Cache 相比Flash分层的优势



- 普通存储采用Flash分级存储
- Exadata Flash Cache 远比分级存储更快地相应工作负载变化
  - 除备份等特殊IO，每个工作负载产生的IO都会改变CACHE的内容
  - 如果有新的数据生成，马上被缓存
  - 如果采用分级存储，数据是非常缓慢根据历史趋势地进行存储迁移
  - **分级存储提升性能还是为昨天甚至上周可能上个月的数据，而不是今天的数据**
- Exadata Cache提供更精细的粒度
  - 64KB vs 1MB
  - 更加高效地捕捉热数据，提升利用率
- Exadata Cache 无需镜像所有数据
  - 读缓存部分无需镜像
- 管理方便性
  - 在数据库层面统一管理
  - 绝大部分操作无需手工操作，手工操作只有一条alter语句；相比之下请看看分级存储的实施手册

  | Insert Information Protection Policy Classification from Slide 13

Sun Flash Accelerator F40 PCIe Card

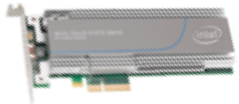
- 400GB Storage Capacity
  - 4 x 100GB Flash modules/doms
- x8 PCIe card
- No ESM need regular maintain
- Optimized for Database caching
- Measured end-to-end performance
  - 3.6GB/sec/cell
  - >100,000 IOPs/cell

 | | Insert Information Protection Policy Classification from Slide 13

Exadata X5-2 高容量 (HC) 存储服务器

更大闪存，更好性能

- 自动缓存, 横向扩展, 高可用, InfiniBand互联的智能存储
- 12 前面板安装4 TB 高容量磁盘 – 单机架672 TB
- 每台存储服务器4x 1.6TB PCIe 闪存卡
  - 最新技术 NVMe接口
  - 智能闪存缓存技术管理的闪存



单机架高容量X5-2 vs X4-2

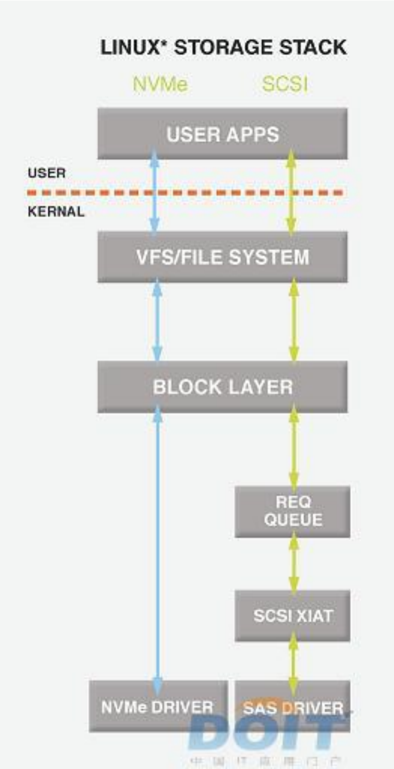
100% 更大的flash cache	89.6 TB 裸容量(无硬件压缩)
缩短25%OLTP IO响应时间	减少OLTP I/O 等待25%
增加55% OLTP读	4.14M 8K SQL读 IOPs
增加37% OLTP 闪存写	2.69M 8K SQL写
分析扫描速度加快40%	140 GB/s SQL扫描速度（未考虑压缩)
0硬件常规维护时间	磁盘控制器不再需要定期更换电池

Per standard 8 compute 14 storage DB Machine Full Rack

ORACLE

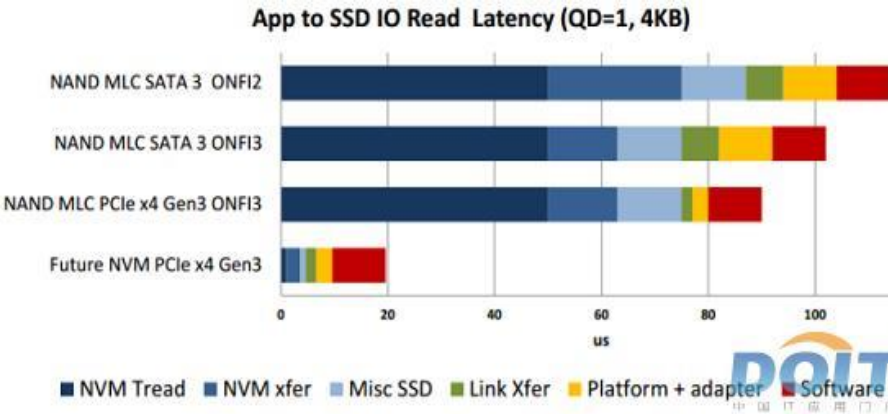
NVME vs 当前的SSD/Flash技术

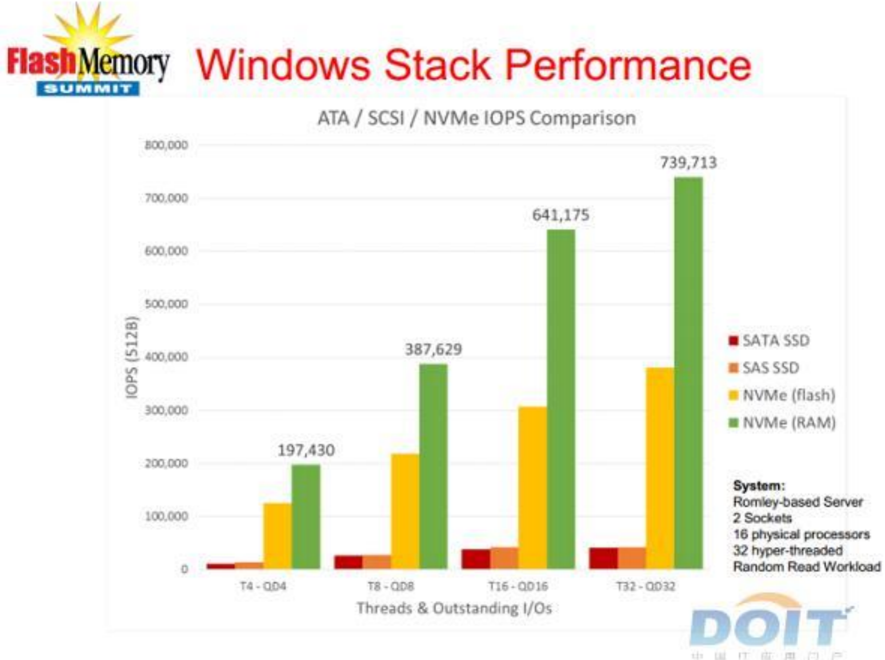
	NVMe	AHCI
Latency	2.8 µs	6.0 µs
Maximum Queue Depth	Up to 64K queues with 64K commands each	Up to 1 queue with 32 commands each
Multicore Support	Yes	Limited
4KB Efficiency	One 64B fetch	Two serialized host DRAM fetches required



ORACLE

NVME vs 当前的SSD/Flash技术





ORACLE

Flash Cache and Flash Grid Disks

- 1600 GB flash memory per cell is used to create 16 cell disks 100GB each (4 cards x 4 FDOMs)
- Flash-based cell disks can be used for
  - Smart Flash cache
    - Uses all available space by default
    - Managed automatically for maximum efficiency
    - Beneficial for OLTP and DW workloads
  - Flash-based grid disks (Not recommend)

ORACLE

Flash-based Devices in CellCLI

- diskType attribute for griddisk, celldisk, lun, physicaldisk
- ALL FLASHDISK and ALL HARDDISK qualifiers

```
CellCLI> LIST CELLDISK DETAIL
    name:                   FD_00_cell01
    diskType:             FlashDisk
    . . .
    name:                   CD_00_cell01
    diskType:             HardDisk
    . . .

CellCLI> CREATE GRIDDISK ALL FLASHDISK –
         PREFIX='FAST', SIZE=10G
   GridDisk FAST_FD_00_cell01 successfully created
   GridDisk FAST_FD_01_cell01 successfully created
   . . .
```


Pinning DB Objects in Cache

- DBA can enforce that an object is kept in flash cache
  - CELL_FLASH_CACHE = Keep, Default, None
- Different cache retention policy for ‘keep’ objects
  - Cached more aggressively
  - Cannot be pushed out by ‘default’ objects
- Keep blocks are automatically ‘un-pinned’ if
  - Object is dropped, shrunk, or truncated
  - Object is not accessed on the cell within 24 hours
  - Block is not accessed on the cell within 48 hours

Exadata Smart Flash Table Caching From 11.2.3.3.0

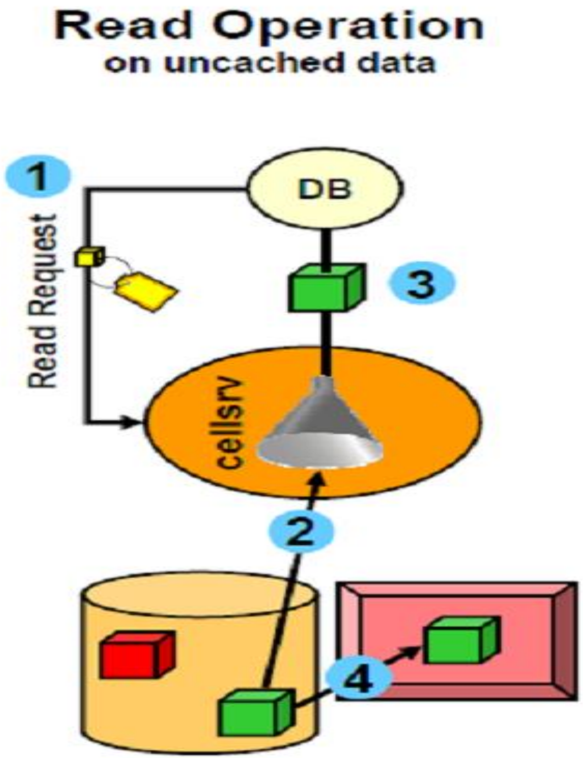
- Smarter flash caching for large table scans
  - Exadata software understands database table and partition scans and automatically caches then when it makes sense
  - Avoids thrashing flash cache when tables are too big or scanned infrequently or scanned by maintenance jobs
  - If scanned table is larger than flash, then subset of table is cached
  - No need to manually “KEEP” tables that are only scanned

SQL Storage Clause CELL_FLASH_CACHE

- DBA can assign caching priority for DB objects
  - In storage clause, during create or alter

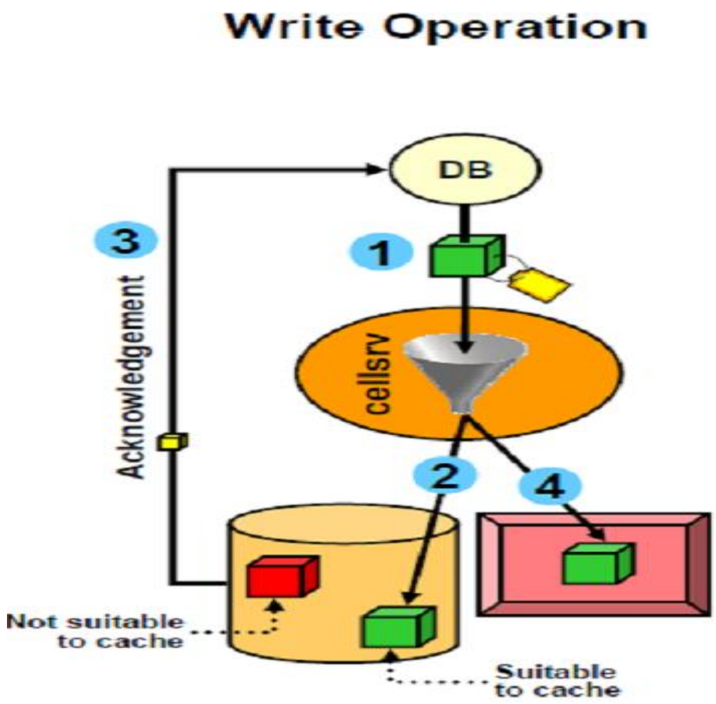
```
ALTER TABLE tkb STORAGE (CELL_FLASH_CACHE NONE);

CREATE TABLE pt (c1 number, c2 clob) TABLESPACE TBS_1
  PARTITION BY RANGE(c1)
   (PARTITION p1 VALUES LESS THAN (100) TABLESPACE
       TBS_2 STORAGE (CELL_FLASH_CACHE DEFAULT),
    PARTITION p2 VALUES LESS THAN (200) TABLESPACE
       TBS_3 STORAGE (CELL_FLASH_CACHE KEEP));
```

Write-back Read Path



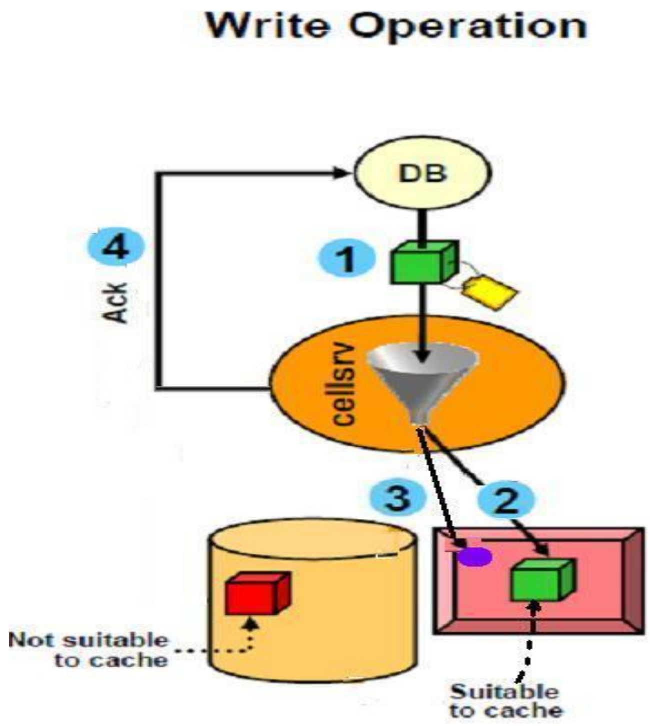
- DB Read request
  - Check in memory cache directory to see if data is cached
  - If cache hint and cache hit then read from cache. Otherwise(no cache hint or cache miss), read dirty pages from cache and clean pages from disk.
  - Send data back to client
  - If cache hint and cache miss, then populate cache
  - If any data is marked valid, then persist Metadata

Write-through Write Path



- DB Write request
  - Invalidate any valid data
  - Write data to disk
  - Acknowledge write to the DB
  - Populate cache with copy of data if write came with cache hint

Write-back Write Path



- Write with cache hint
  - Maybe do invalidation + pers
  - Write data to flash
  - set valid+dirty and persist metadata to flash unless redirty
  - Acknowledge write to the DB
- Nocache write
  - Only to disk. No populate.
    - Invalidate clean data before going to disk and dirty data after writing to disk
  - Ack client

Write path from Database perspective

- Log writes -- (LGWR)
  - Upon every transaction "commit"
  - Latency is #1 priority
  - Exadata cell behavior: Flash logging
    - Write to DISK and FLASH… ack on first completion.
- Flushing the Buffer Cache -- (DBWR)
  - Clean dirty buffers in SGA
  - IOPS bandwidth is key
  - Exadata cell behavior
    - Writes to FLASH

- Direct path writes -- (Oracle Shadow and PQ)
  - Bandwidth is typically #1 priority
  - typically LARGE IO sizes
    - Exadata cell behavior:
      - cell_flash_cache determines the path
        - Keep -> writes to FLASH
        - Default -> writes to DISK
- TEMP writes -- spills out of PGA
  - Sort, Hash Join, etc..
  - Exadata cell behavior
    - Writes go to DISK

Write path from Database perspective… cont

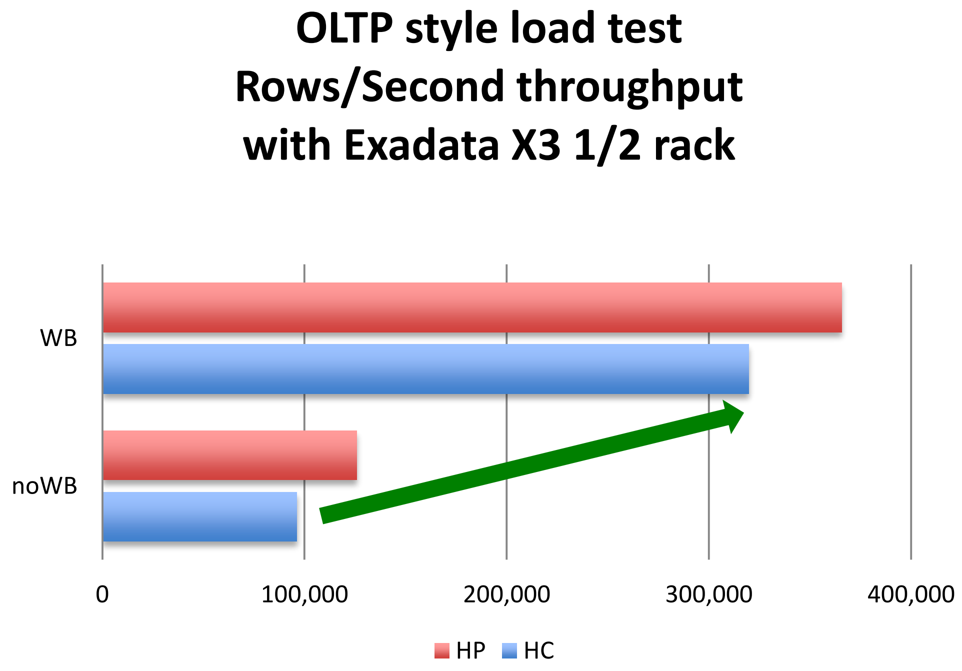
- Archive Log writes -- (ARCH)
  - After REDO logs are switched, they are written to the “RECO” space by ARCH
  - These are “LARGE” writes
  - Exadata cell behavior:
    - Write to DISK only

- FLASHBACK database – (RVWR)
  - Uses it’s own format, but volume is similar to “redo” writes
  - Typically written to RECO
  - Exadata cell behavior
    - Write to DISK only

OLTP style insert test

- Simple PL*SQL loop
  - Simple insert one row at a time
  - Commit every 100 rows
  - Simple table structure with unique primary “key” index
- Multiple sessions
  - Each session connects as a “different” database user “a1, a2, …”
  - Each “user” inserts into their own schema
  - 32 sessions are started per DB node… two per core… one per “thread”
  - ½ rack HP and HC configurations were tested
    - 128 total sessions inserting with NO think time

Overall Results - OLTP style insert



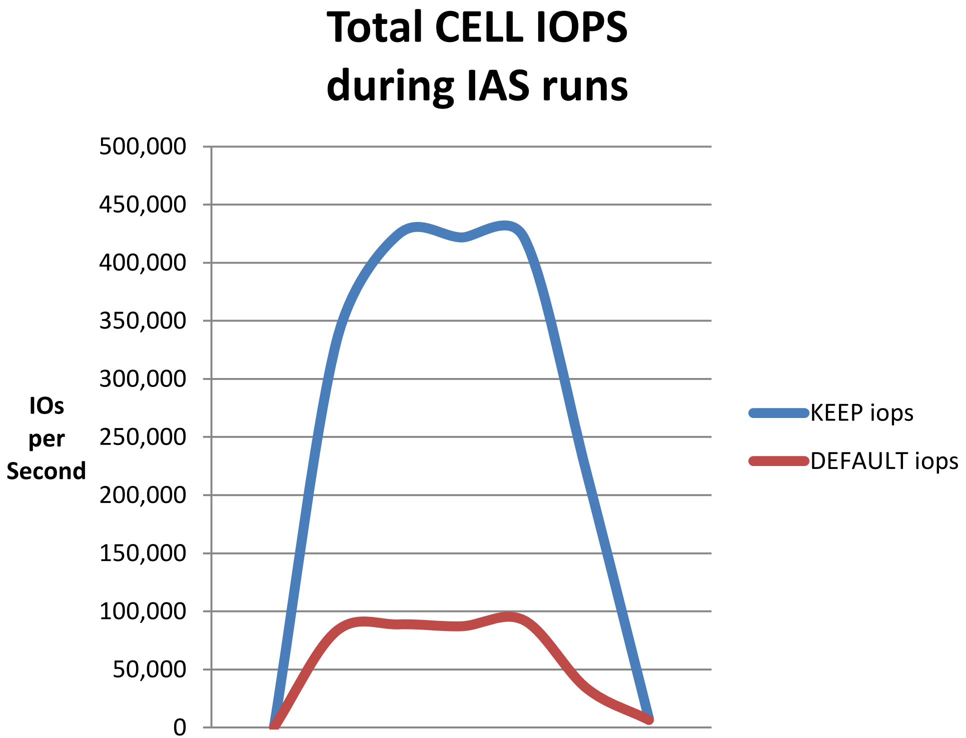
- WriteBack cache works!!
  - Over 300K rows per second insert regardless of disk type!!
  - 330% improvement in performance with HC drives & WriteBack
- HP vs HC differences?
  - JBOD -- With NO WriteBack Cache
    - HP disks provide 23% better throughput
  - WriteBack Cache enabled
    - HP drives provides only **13%** better throughput

Insert-as-Select details

- Insert-as-Select
  - /*+APPEND */ hint triggers Direct path writes
  - “cell_flash_cache” storage parameter determines the IO path
    - “DEFAULT”… Writes directly to DISK
    - “KEEP”…. Uses the “WriteBack” FLASH

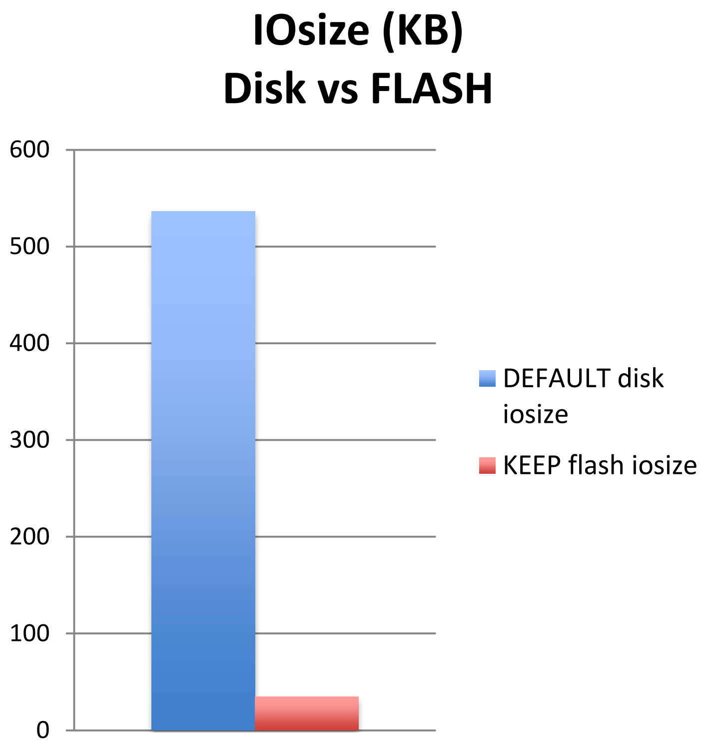
- Which is better for Direct path writes?
  - Disk can only do around 50K IOPS on a full rack
  - Flash can deliver 1,500,000 IOPS

Flash IOPS clearly higher than DISK



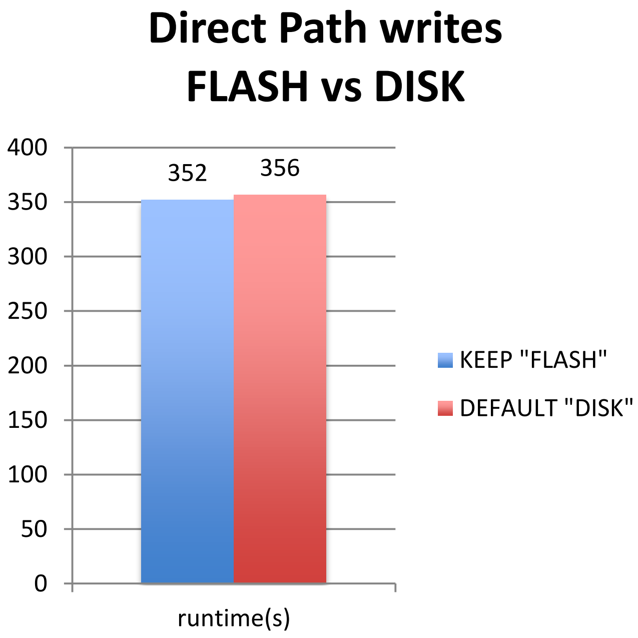
- Sum of ALL IO operations within the cell… both to DISK and FLASH.
- KEEP drastically increases the IOPS
- Storage teams are cheering… Using KEEP seems to be a clear winner… Look at those IOPS!!

What about the IO size?



- IOSTAT within the cell shows
  - Disk iosize peaks at 512K
  - Flash iosize peaks at 32K for this workload
- Large writes to DISK efficient and don't stress IOPS
- Flash writes are capped at 128k
- So, Flash IOPS are clearly higher, but the payload is smaller
- Are you ready to declare a winner?

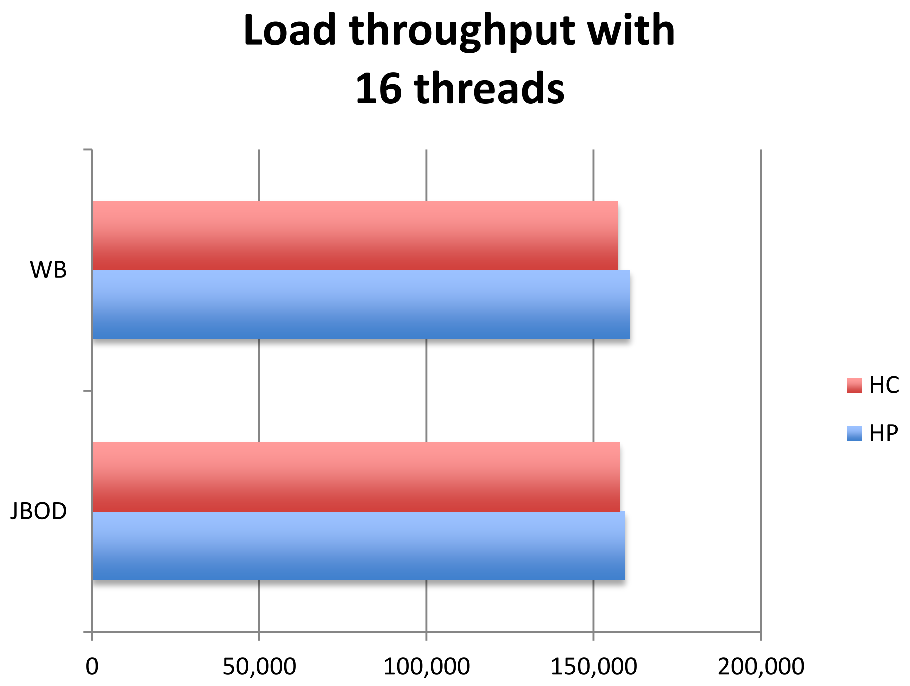
… and the winner is???



- The runtimes are ALMOST identical!!
- Using “KEEP” to write to flash improved the import times by 1.3%

- If you are importing data that will be worked on immediately by OLTP style operations, then maybe KEEP makes sense.
- If you are bulk loading large amounts of data for Parallel Query, you probably should not pollute the flash cache with this data.

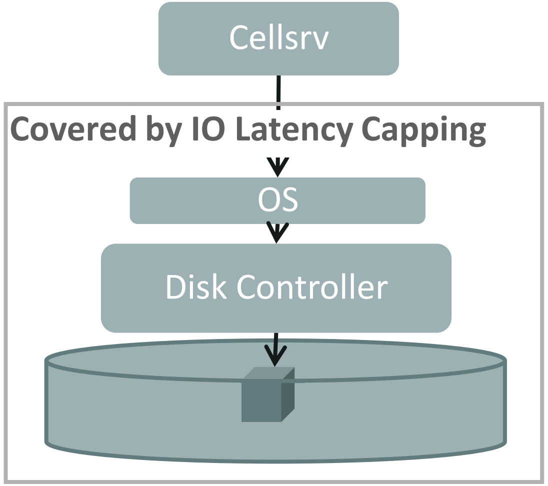
Throughput with only 16 threads?



- What if you don’t push it as hard?
  - Which is faster? (HP or HC)?
- HP and HC are about the SAME
- Write Back cache has basically the same performance as JBOD
- When a transaction commits
  - Log response time matters
  - Writes to the data file is handled offline by DBWR… as long as it keeps up it doesn’t slow you down

IO Latency capping

- IOs are "cancelled" (returned early with error) if
  - Disk/Flash is slow ("hair trigger")
  - Disk/Flash fails
  - DB will retry to other another mirror for read
  - DB will redirect write to another position
- DB Requires
  - 11.2.0.4 BP8 or 12.1.0.2 RELEASE
- Cell Requires
  - 11.2.3.3.1 (read cancellation only)
  - 12.1.1.1.1 (read cancellation only)
  - 12.1.2.1.0 (Includes FC logger, enables write cancellation)



ORACLE

