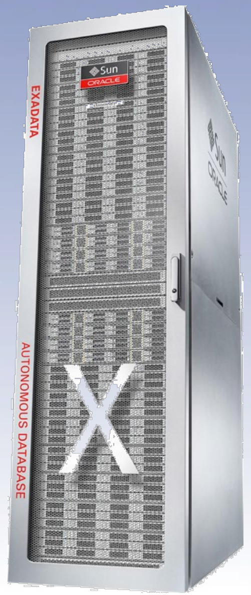
Under the Hood of an Exadata Transaction

How to harness the power of Persistent Memory?

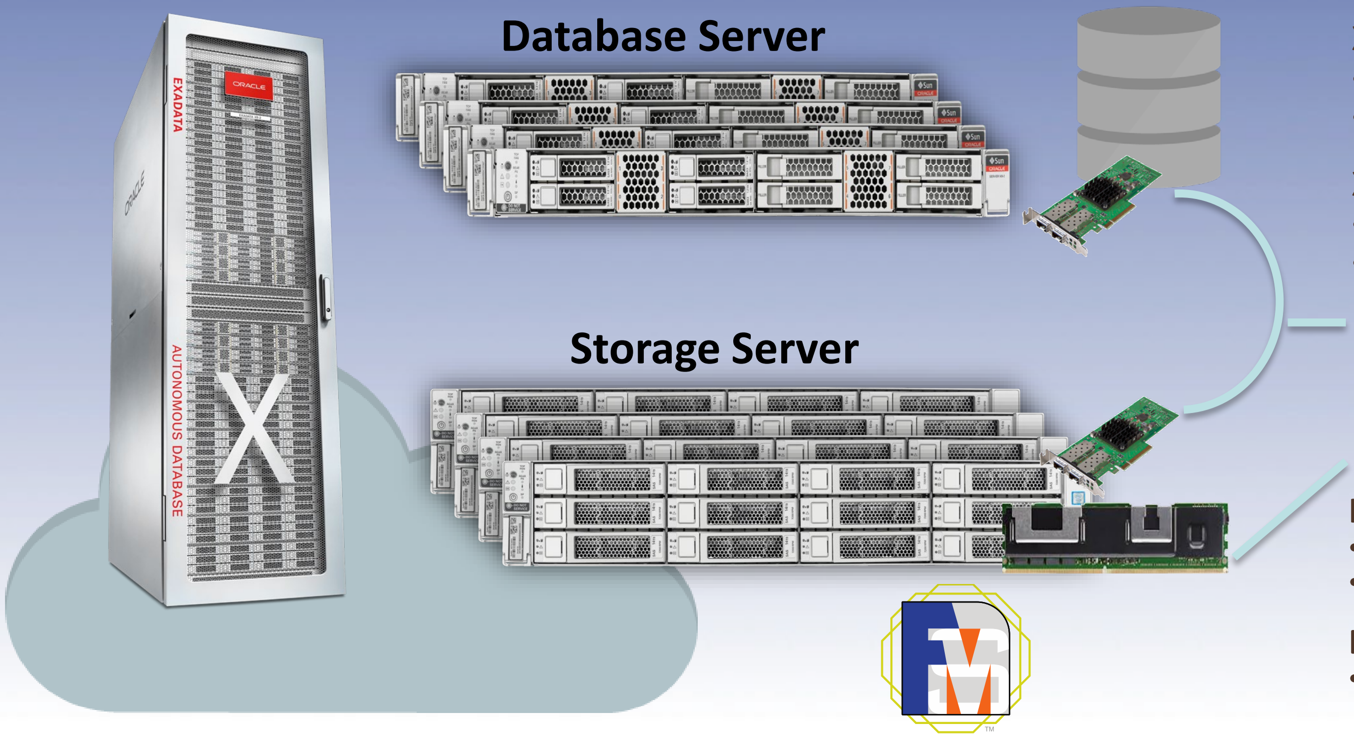
Flash Memory Summit

Jia Shi
Vice President
Exadata Development, Oracle
jia.shi@oracle.com



Meet Exadata X8M

What is the secret sauce in X8M?



Database Server

Storage Server

X8M-2 socket Xeon

- **48** cores per server
- **384 GB - 1.5 TB** DRAM

X8M-8 socket Xeon

- **192** cores per server
- **3-6 TB** DRAM

100 Gb/s **RoCE**
RDMA **over C**onverged **E**thernet

1.5 TB Persistent Memory

High Capacity

- **168** TB HDD
- **25.6** TB PCI NVMe Flash

Extreme Flash

- **51.2** TB PCI NVMe Flash

Let’s go under the hood of OLTP

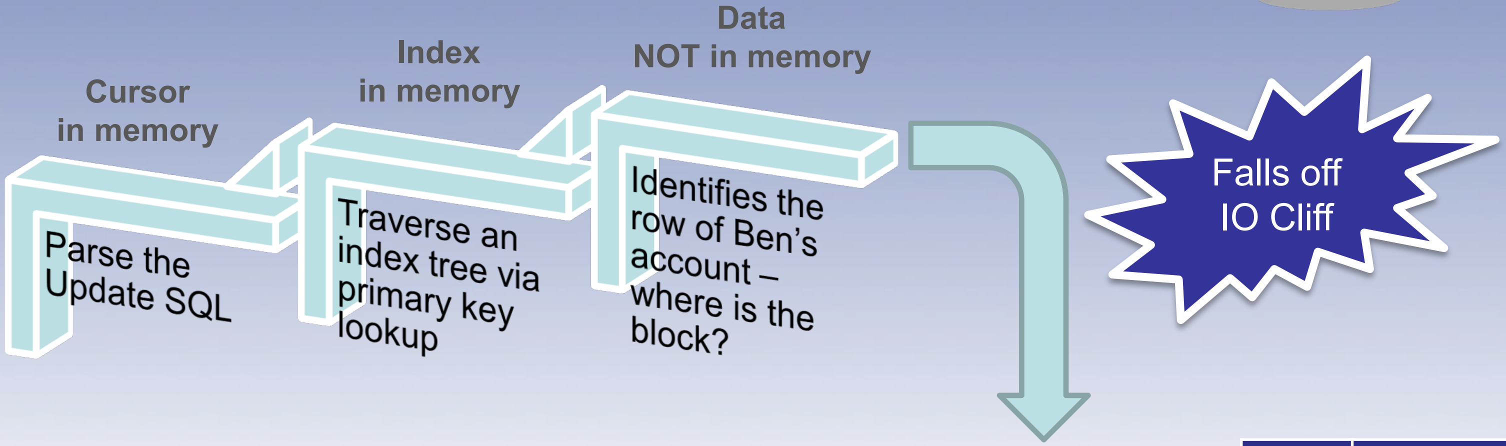
OnLine Transactional Processing

Meet Ben’s Transaction

What constitutes a database transaction?

Ben wants to deposit $1000 to his bank account.

Deposit
$1000


Cursor in memory
Index in memory
Data NOT in memory
Parse the Update SQL
Traverse an index tree via primary key lookup
Identifies the row of Ben’s account – where is the block?
Falls off IO Cliff

User	Account Balance
…	…
Ben	$2000

OLTP Challenge #1 -

What is the IO cliff for random data reads?

Challenge #1 – Random Data Read

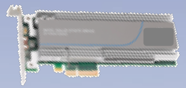
Database Server

1. Identifies the row of Ben's account – where is the block? Miss in the buffer cache!
2. Issues the data read to storage

Storage Server

1. Finds the data block cached in Flash Cache
2. Issues local read to Flash

Flash Cache



…

Flash Cache Line

Flash Cache Line

…

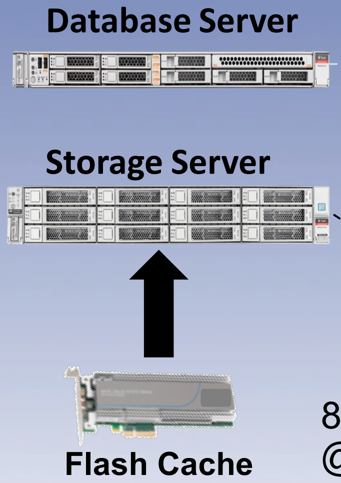
3.Sends data block to DB

Buffer Cache (DRAM)

User	Account Balance
…	…
Ben	$2000

How long did we wait for this random block read?

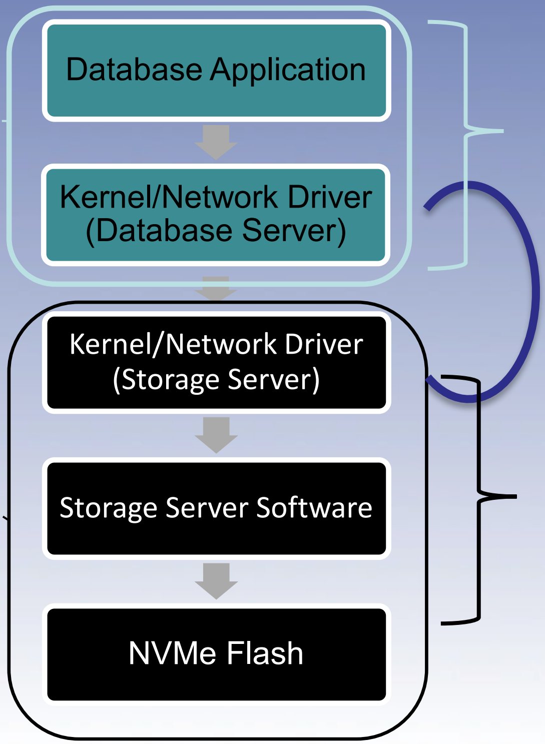
How long does an 8K random read take?



8K Local Read
@ < 100 usec

How long did we wait for this random block read?

~200 usec



- User->Kernel & Kernel-> User context switches
- Network IO interrupt processing
- IO software stack

Network Round Trip

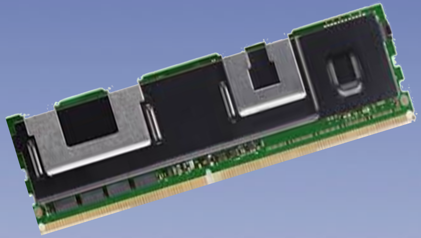
- User->Kernel & Kernel-> User context switches
- Network IO interrupt processing
- IO software stack

~100 usec

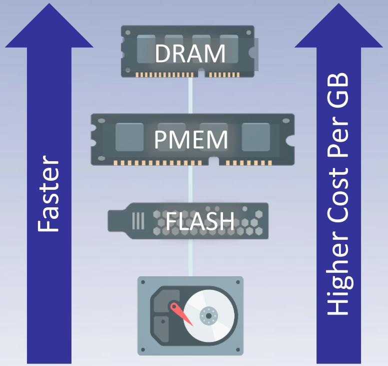
OLTP Challenge #1 -

How to conquer the random read IO Cliff?

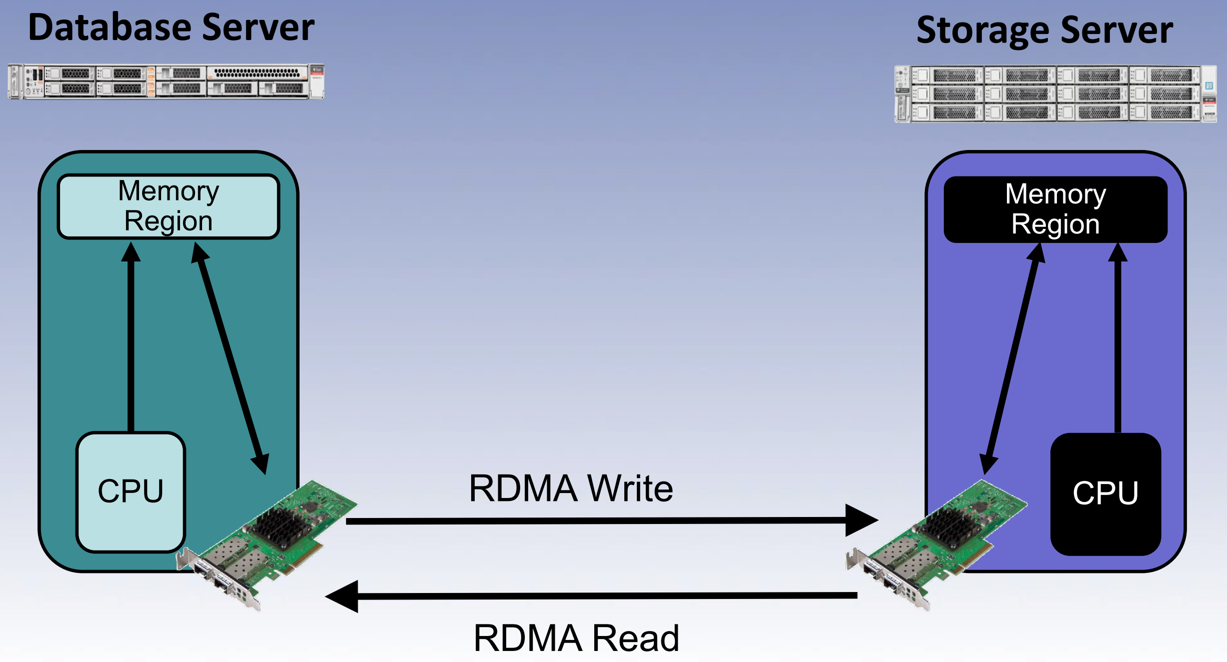
Persistent Memory (PMEM)



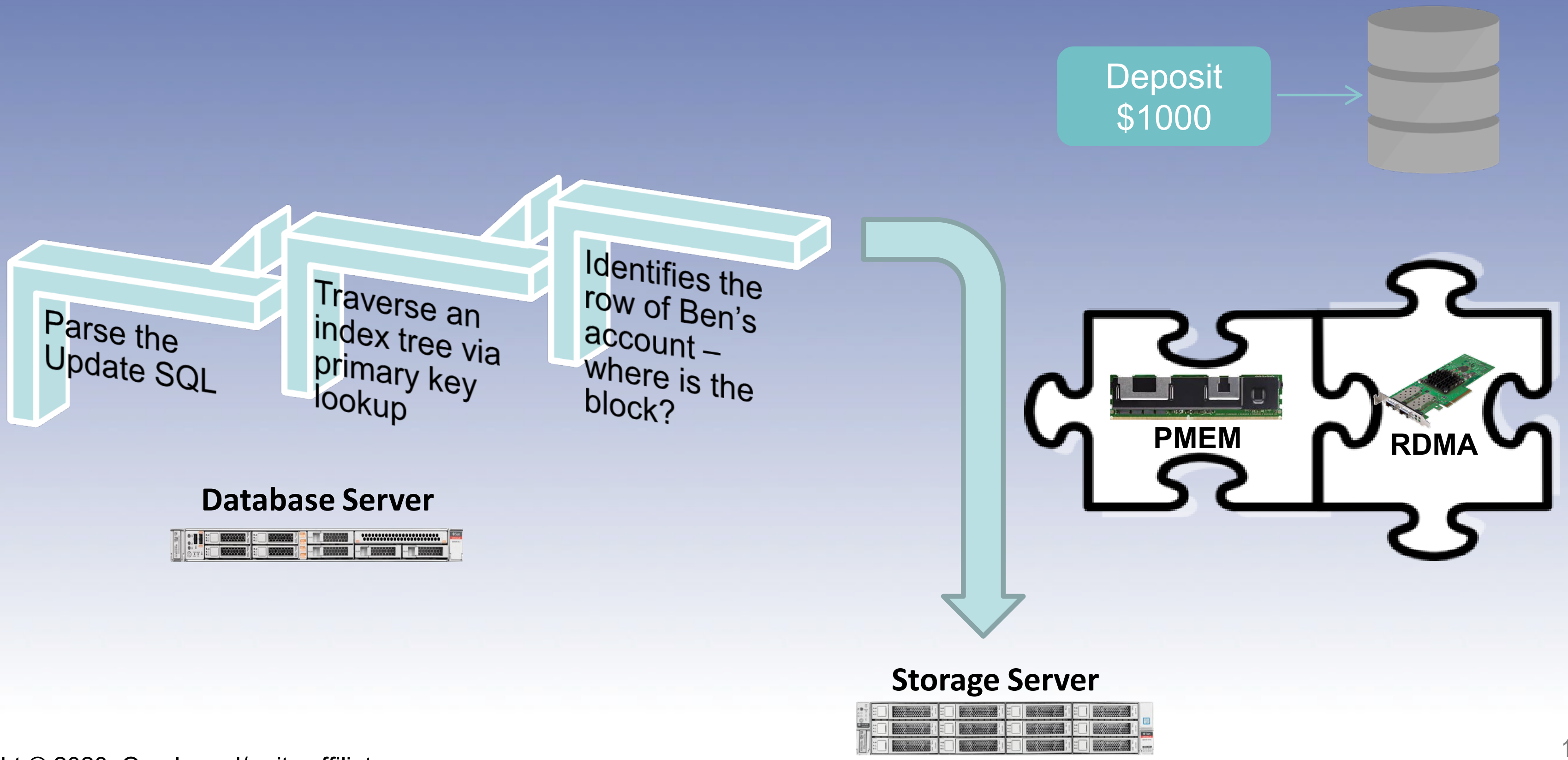
- Persistent memory is a new silicon technology
  - Capacity, performance, and price are between DRAM and flash
- Intel® Optane™ DC Persistent Memory:
  - Reads at memory speed – much faster than flash
  - Writes survive power failure unlike DRAM
- Requires *sophisticated algorithms* to maintain integrity of data on PMEM during failures
  - Call special instructions to flush data from CPU cache to PMEM
  - Complete or backout sequence of writes interrupted by a crash



Remote Memory Direct Access (RDMA)



How do they conquer the IO cliff?



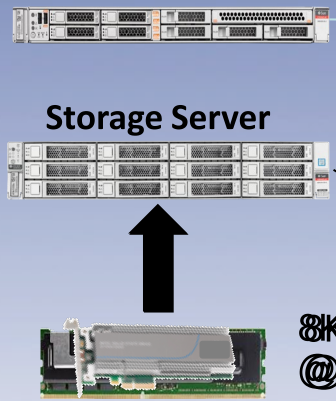
Deposit $1000

Parse the Update SQL

Traverse an index tree via primary key lookup

Identifies the row of Ben's account – where is the block?

Drop in solution: Flash -> PMEM?



PMEM Cache

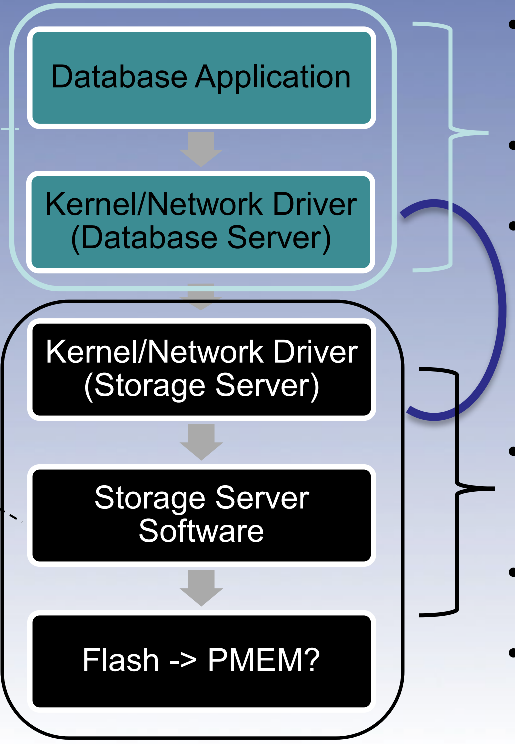
Flash Cache

8K Local Read
@ ~1.6 - 2.4 usec

8K Local Read
@ <100 usec

What happens to the IO Cliff?

~100 usec



- User->Kernel & Kernel-> User context switches
- Network IO interrupt processing
- IO software stack

Network Round Trip

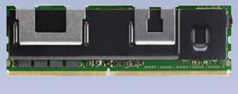
- User->Kernel & Kernel-> User context switches
- Network IO interrupt processing
- IO software stack

~100 usec

How about a radical approach: RDMA to PMEM?

Database Server

Storage Server

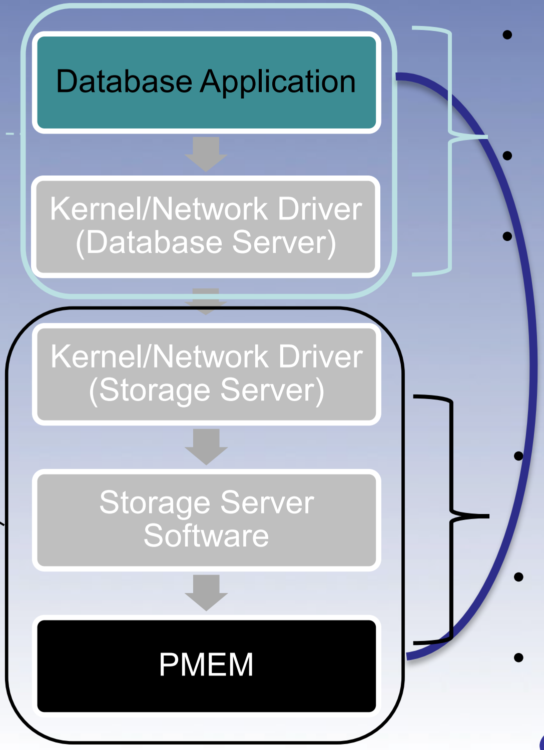


PMEM Cache

What happens to the IO Cliff?

<19 usec

10x Faster Random Read!



Database Application
Kernel/Network Driver (Database Server)
Kernel/Network Driver (Storage Server)
Storage Server Software
PMEM

- User->Kernel & Kernel-> User context switches
- Network IO interrupt processing
- IO software stack

RDMA over 100Gb/s RoCE

- User->Kernel & Kernel-> User context switches
- Network IO interrupt processing
- IO software stack

~100 usec

RDMA Read from PMEM Cache in Storage Server

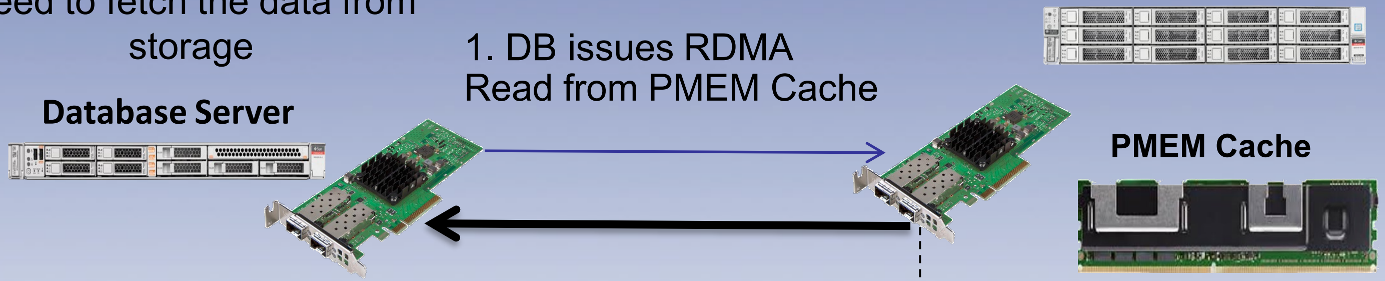
Miss in Buffer Cache:
Need to fetch the data from storage

Database Server

1. DB issues RDMA Read from PMEM Cache

Storage Server

PMEM Cache



2. 8K Data Block returned to DB from PMEM
- No software involved

User	Account Balance
…	…
Ben	$2000

…

PMEM Cache Line

PMEM Cache Line

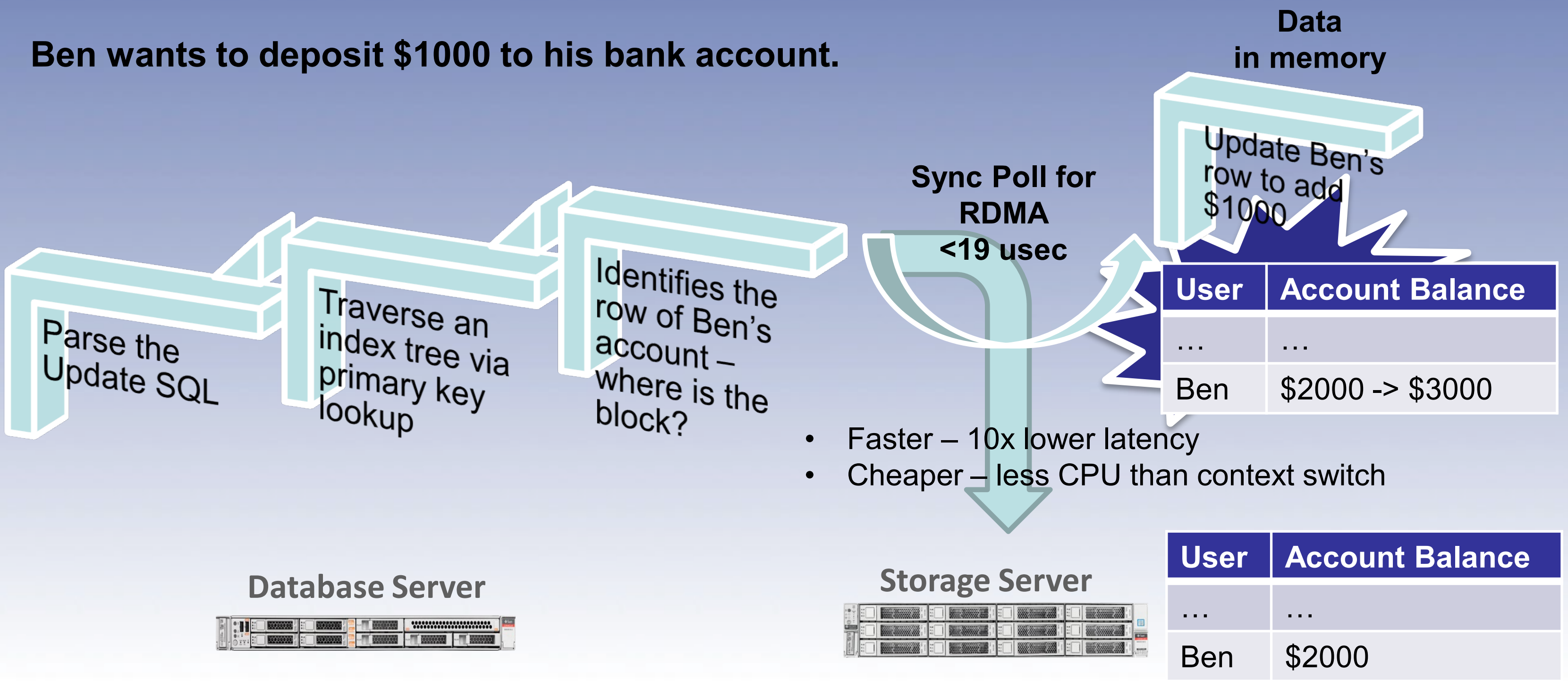
PMEM C

User	Account Balance
…	…
Ben	$2000

PMEM + RDMA

Transforming IO bound application to near memory performance

Ben wants to deposit $1000 to his bank account.



Parse the Update SQL

Traverse an index tree via primary key lookup

Identifies the row of Ben's account – where is the block?

Sync Poll for RDMA <19 usec

Data in memory

Update Ben's row to add $1000

User	Account Balance
…	…
Ben	$2000 -> $3000

- Faster – 10x lower latency
- Cheaper – less CPU than context switch

Database Server

Storage Server

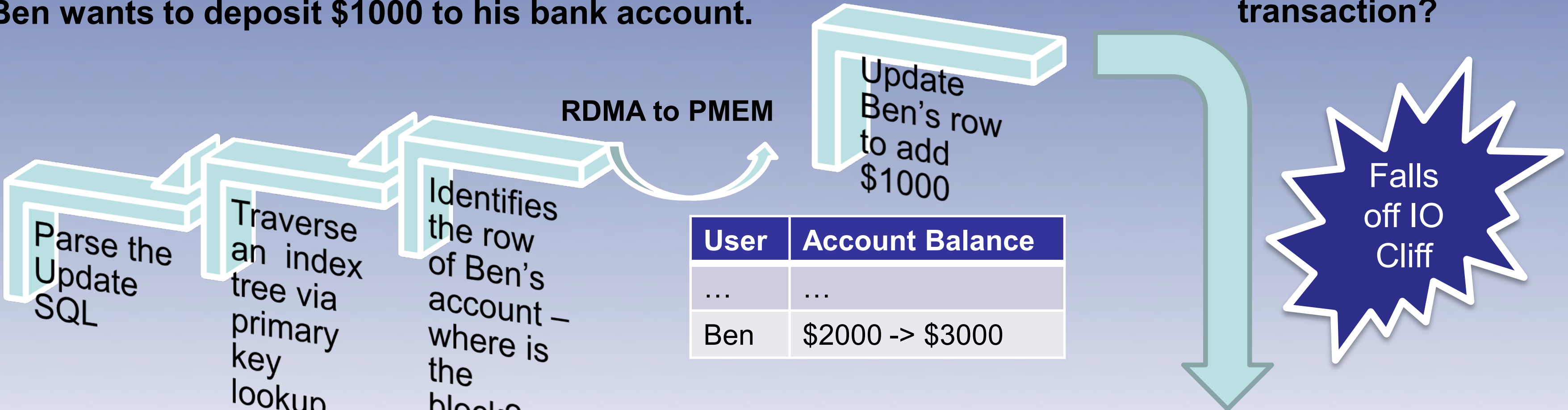
User	Account Balance
…	…
Ben	$2000

Meet Ben's Transaction – time to commit?

Putting the *D(urability)* into Database Transaction ACID Properties

Ben wants to deposit $1000 to his bank account.

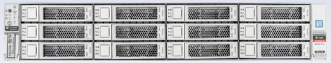
How do we commit a transaction?



Parse the Update SQL
Traverse an index tree via primary key lookup
Identifies the row of Ben's account – where is the block?
RDMA to PMEM
Update Ben's row to add $1000
Falls off IO Cliff

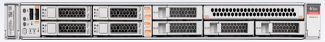
User	Account Balance
…	…
Ben	$2000 -> $3000

Storage Server



1. Update Balance for Ben +$1000
2. Commit Transaction

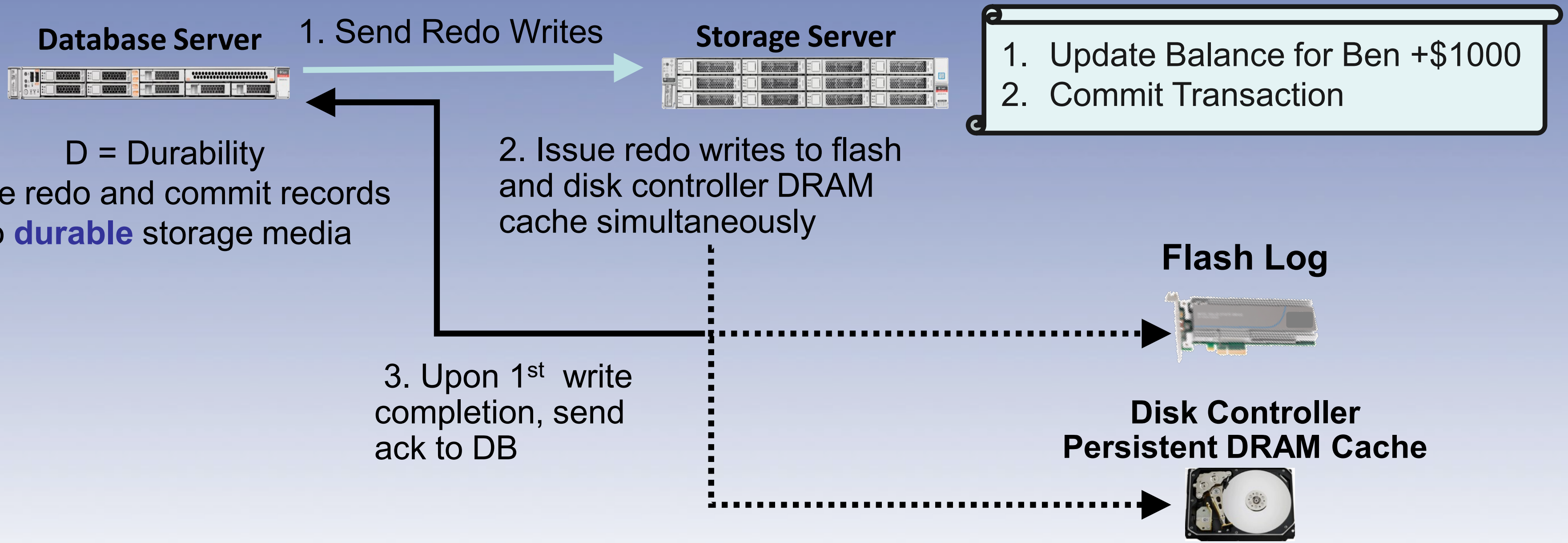
Database Server



OLTP Challenge #2 -

Can lightning strike the same place twice? What is the IO cliff for redo log writes?

Challenge #2 – Redo Log Writes



1. Update Balance for Ben +$1000
2. Commit Transaction

D = Durability
Write redo and commit records to **durable** storage media

2. Issue redo writes to flash and disk controller DRAM cache simultaneously

3. Upon 1st write completion, send ack to DB

How long did we wait for the log write?

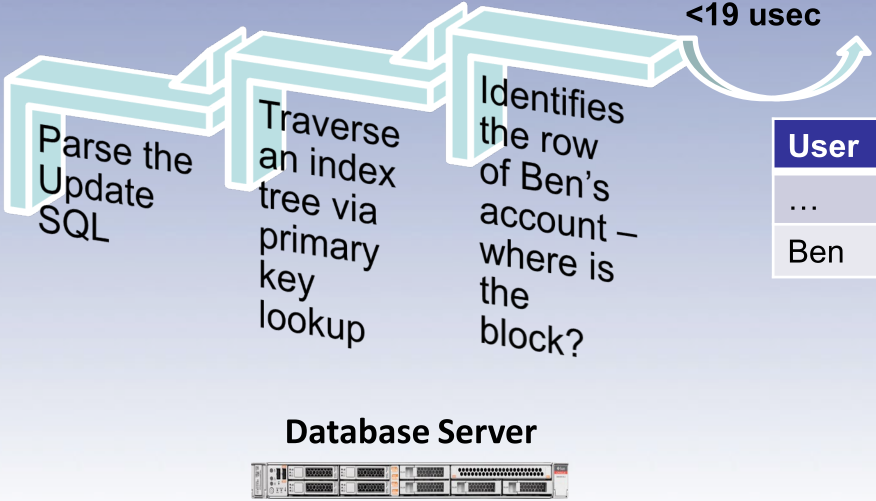
~200 usec

Can PMEM + RDMA come to the rescue again?

Lightning strikes the same place twice!

Ben wants to deposit $1000 to his bank account.

How do we not fall off the IO cliff?



Parse the Update SQL

Traverse an index tree via primary key lookup

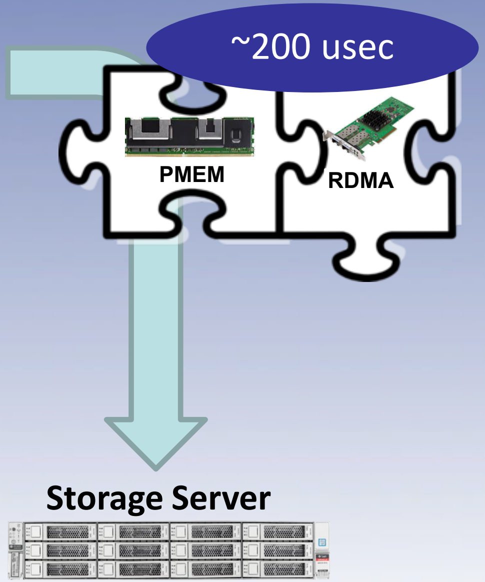
Identifies the row of Ben's account – where is the block?

RDMA to PMEM <19 usec

Database Server

Update Ben's row to add $1000

User	Account Balance
…	…
Ben	$2000 -> $3000

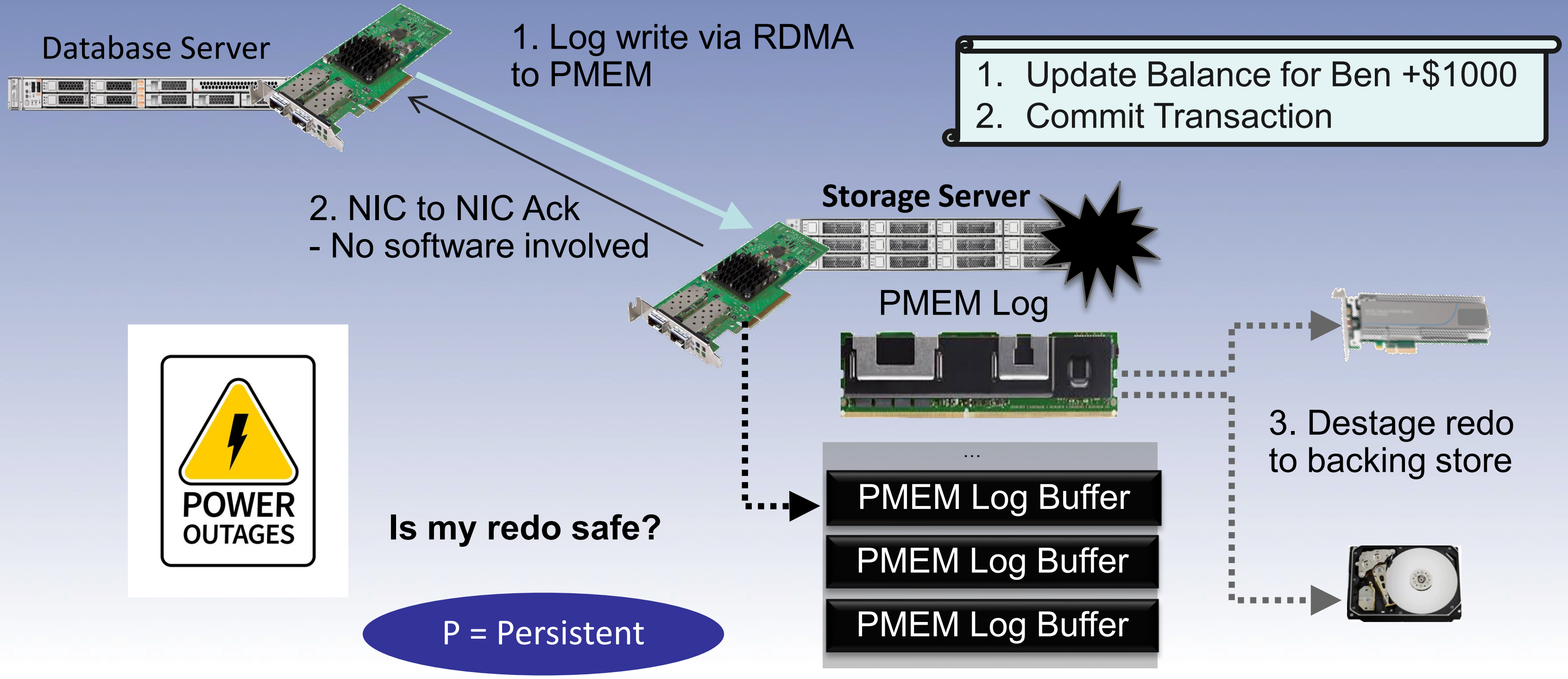


~200 usec

Storage Server

1. Update Balance for Ben +$1000
2. Commit Transaction

Redo Log RDMA write to PMEM in Storage Server



1. Update Balance for Ben +$1000
2. Commit Transaction

Is my redo safe?

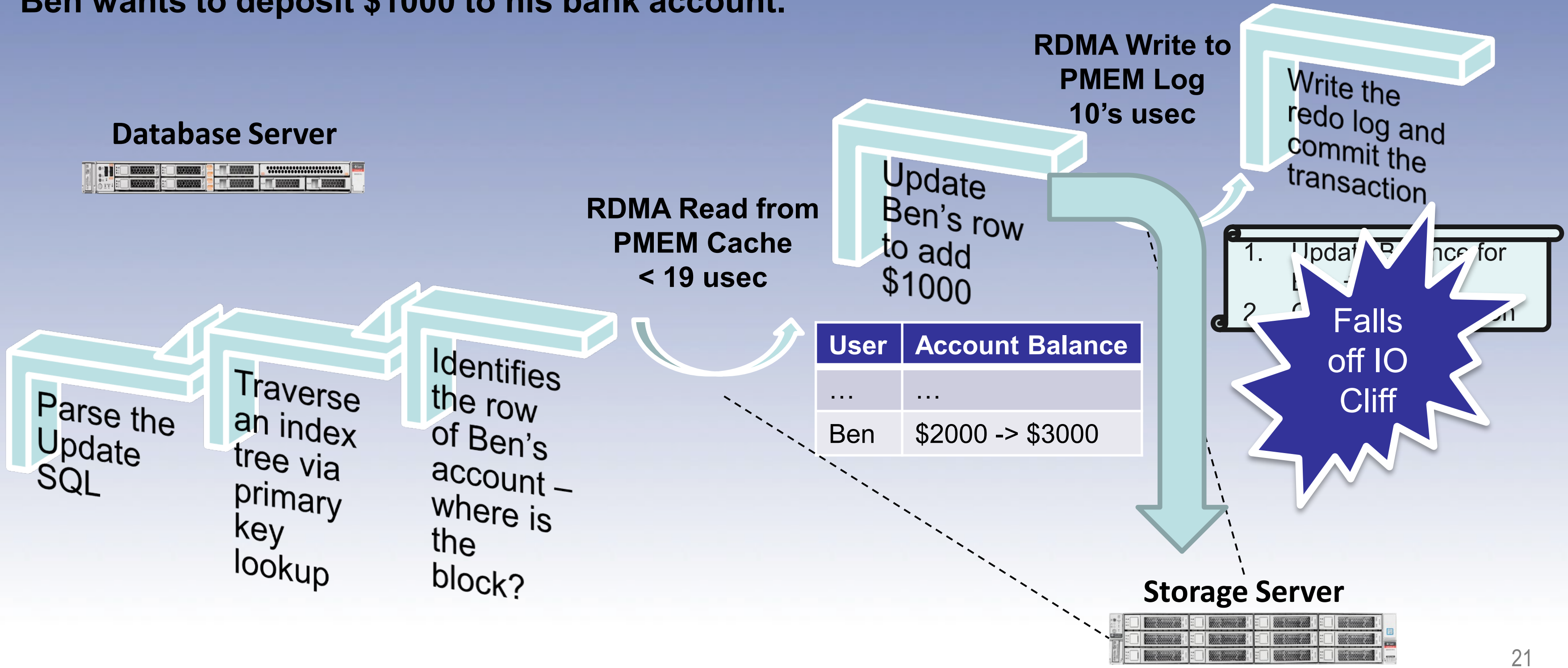
P = Persistent

Do not fall off the IO Cliff!

What happens to the IO Cliff?

8x Faster Log Writes!

Ben wants to deposit $1000 to his bank account.



Database Server

Parse the Update SQL

Traverse an index tree via primary key lookup

Identifies the row of Ben's account – where is the block?

RDMA Read from PMEM Cache < 19 usec

Update Ben's row to add $1000

User	Account Balance
…	…
Ben	$2000 -> $3000

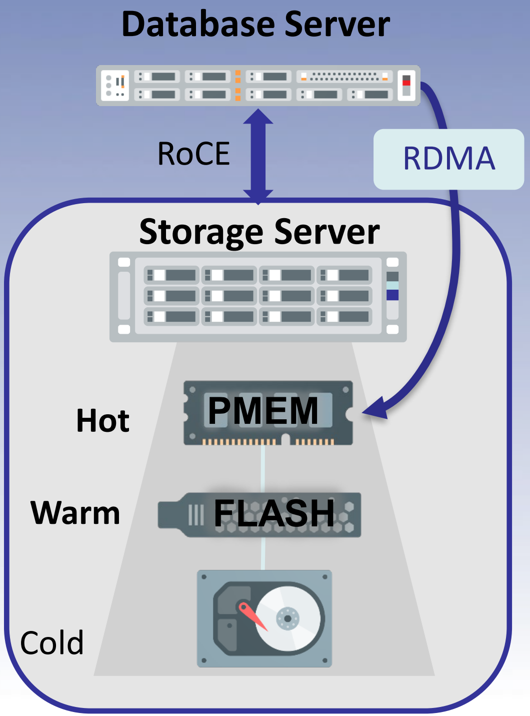
RDMA Write to PMEM Log 10's usec

Write the redo log and commit the transaction

Storage Server

OLTP on Exadata

How do we harness the power of PMEM?



How to have a cake and eat it too?

>99% of PMEM used for PMEM Cache – 1.5TB per server

<1% of PMEM used for PMEM Log – 10G per server

RDMA to PMEM in Storage

10X better transaction processing IO latency @ <19 usec

8X faster log writes for faster commit processing

16 Million read IOPS on a full rack of Exadata database machine

How do you harness the power of PMEM?

Flash Memory Summit

Drop me an email - jia.shi@oracle.com