SDC
STORAGE DEVELOPER CONFERENCE
SNIA SANTA CLARA, 2014

Toward High-Performance Shadow Migration

Youngjin Nam, Aaron Dailey

ORACLE
ZFS STORAGE APPLIANCE

Talk Outline – Learning Objectives

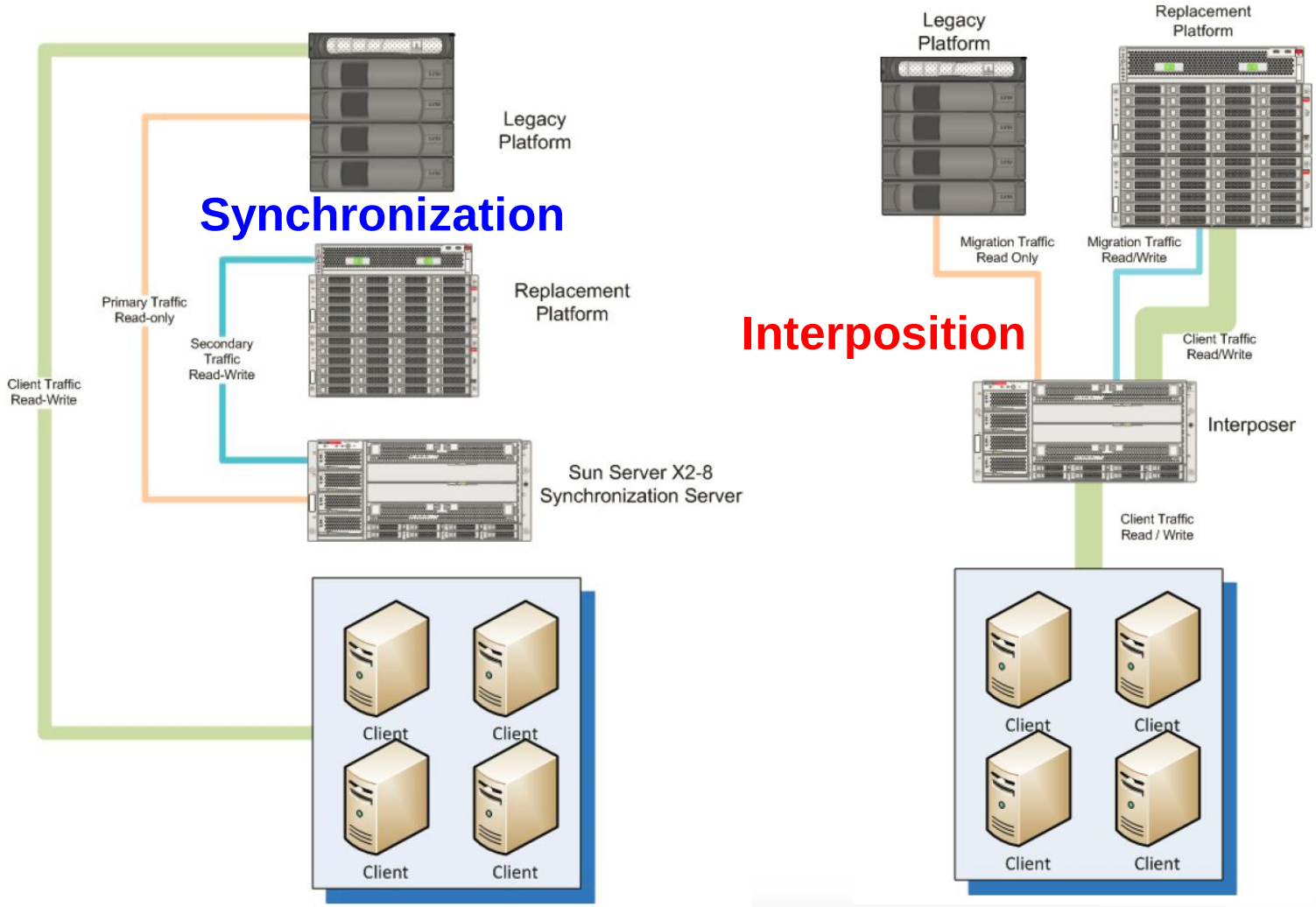
- **Use cases of shadow migration in ZFSSA**
- Shadow migration at a high level
- Optimization issues for higher performance
- Summary

Need of Data (File System) Migration

- Oracle ZFS Storage Appliance (ZFSSA)
  Application-engineered storage system designed to provide unique Oracle software integration with storage performance and efficiency

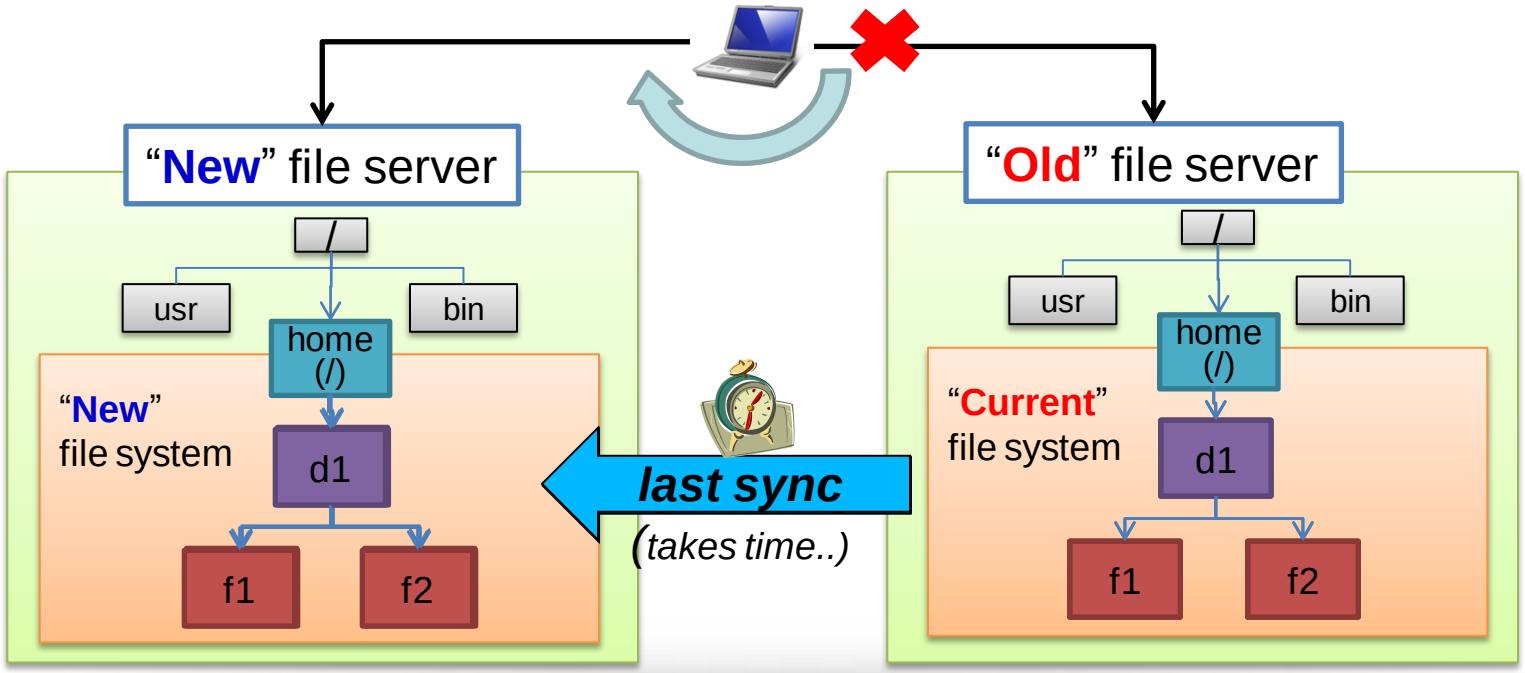
- How do users migrate their data (file systems) easily to their new storage?

Different Approaches for Data Migration



Synchronization Approach (thru rsync)

- Copy data from old fs ('old') to new fs ('new') while data is still being accessed by clients through old
- If diff old & new gets smaller, copy the last diff & new is in service

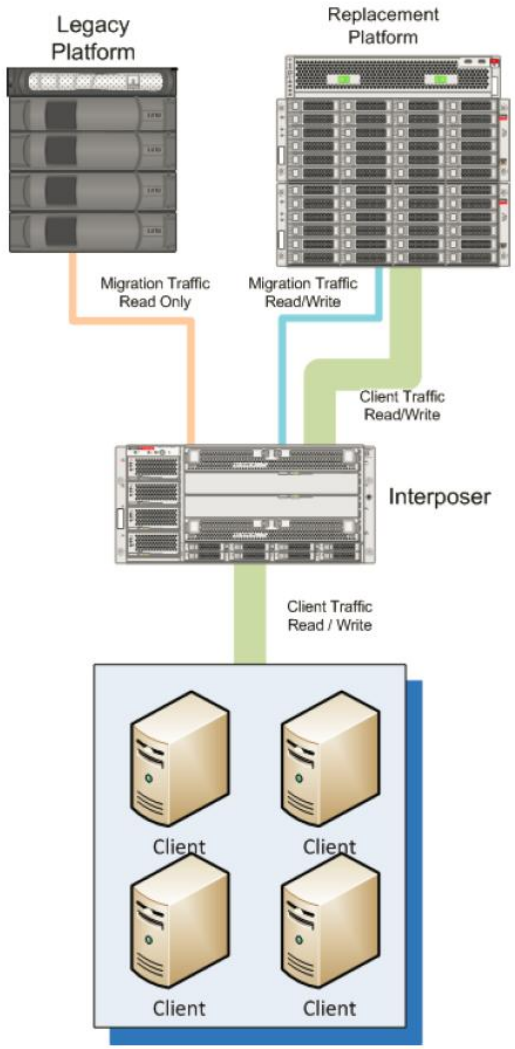


Synchronization Approach (thru rsync)

- No additional I/O delay
- Substantial downtime if a large amount of change is committed immediately before the scheduled downtime
- During migration, the new server is idle
- Coordinating across multiple file systems is burdensome, where each migration will take a different amount of time

Interposition Approach

- Device is placed in between old/new & client
- Interposing device performs background migration from old to new, transparently to the clients
- When data is not in new, it's migrated from old (on-demand migration)

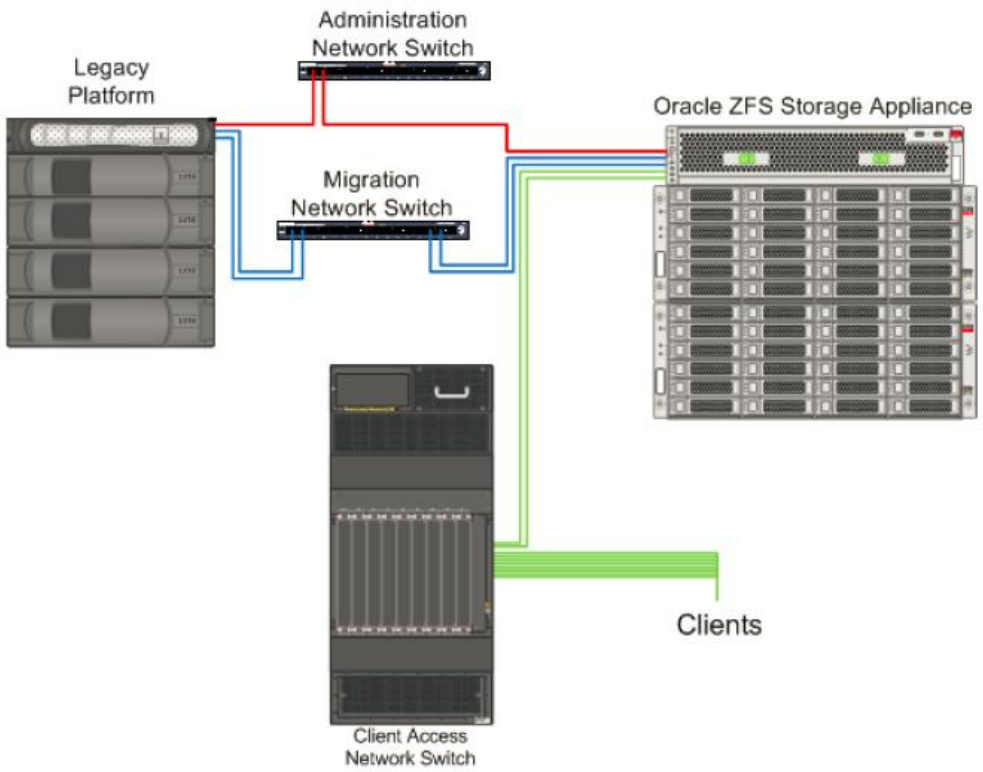


Interposition Approach

- Drastically reduced downtime
- New physical machine with additional costs & management overhead
- New point of failure within the system
- I/O latency when accessing not-yet-migrated data; the migration appliance interposes on already migrated data, incurring extra latency

Shadow Migration is Software-based Interposing Device

- No special hardware; embedded in VFS (ZFSSA/Solaris)
- A means of migrating legacy to ZFSSA (or Solaris server)
- Instant switchover
- Mitigated I/O latency



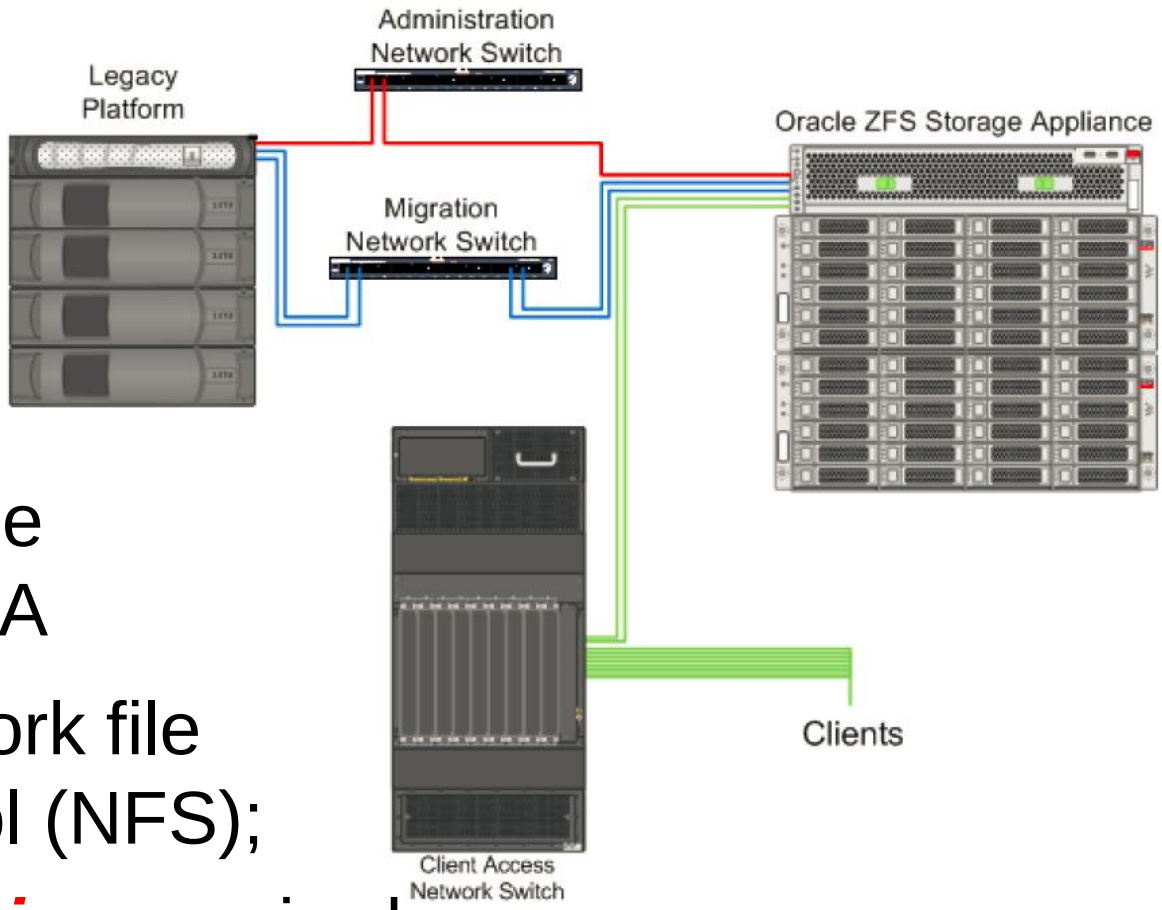
Use Cases of Data (File Systems) Migration

- Data migration from legacy platform
- Migration between different storage pools

Data Migration from Legacy System

- Access data *immediately* thru ZFSSA (***Instant switchover***)
- New data will be stored in ZFSSA
- Standard network file system protocol (NFS);

no special device required



Data Migration between Different Storage Pools

- To perform storage capacity planning, I/O workload balancing, using encrypted file system
- Different storage characteristics (device layout, data redundancy, etc) for storage pool

Data Profile

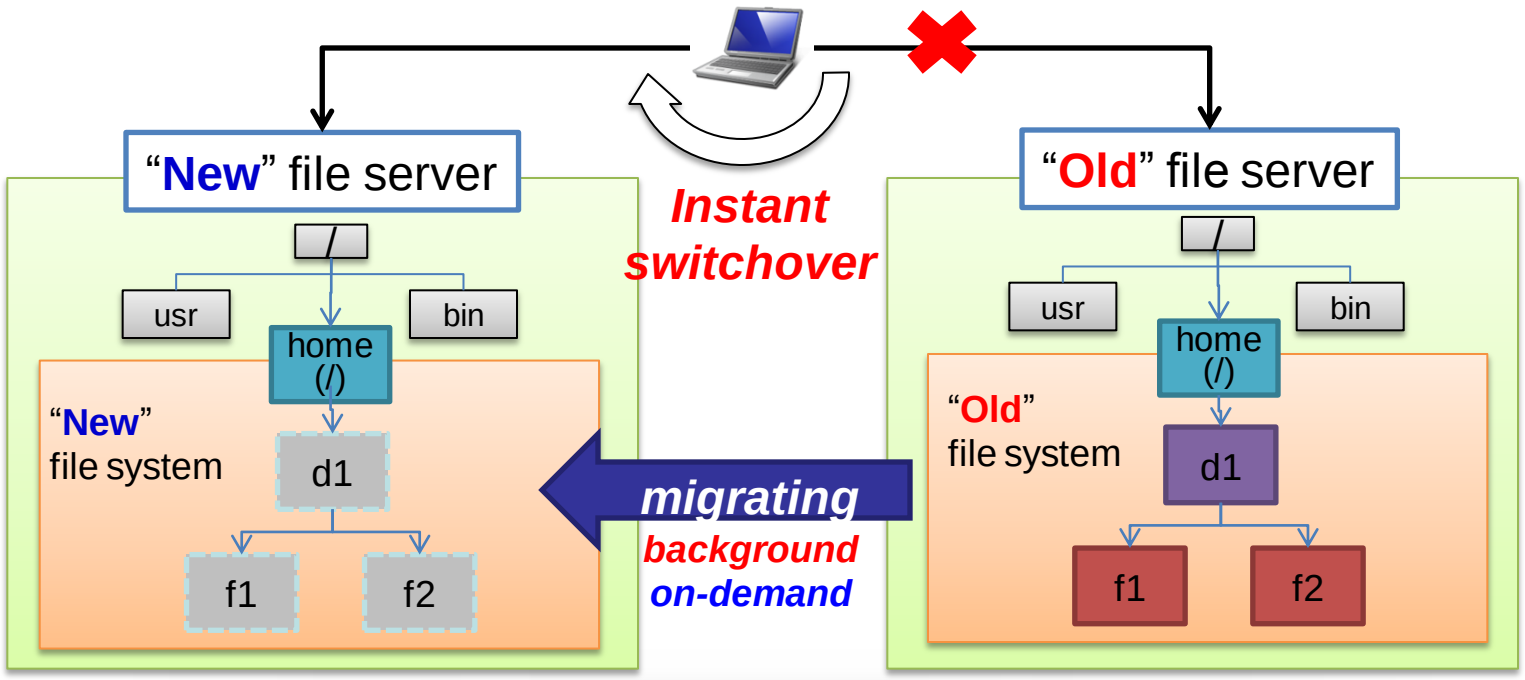
TYPE ▲	NSPF	AVAILABILITY	PERFORMANCE	CAPACITY	SIZE
Double parity	No				41.9T
Mirrored	No				25.6T
Single parity, narrow stripes	No				34.9T
Striped	No				55.9T
Triple mirrored	No				16.3T
Triple parity, wide stripes	No				46.6T

Talk Outline

- Use cases of shadow migration in ZFSSA
- **Shadow migration at a high level**
- Optimization issues for higher performance
- Summary

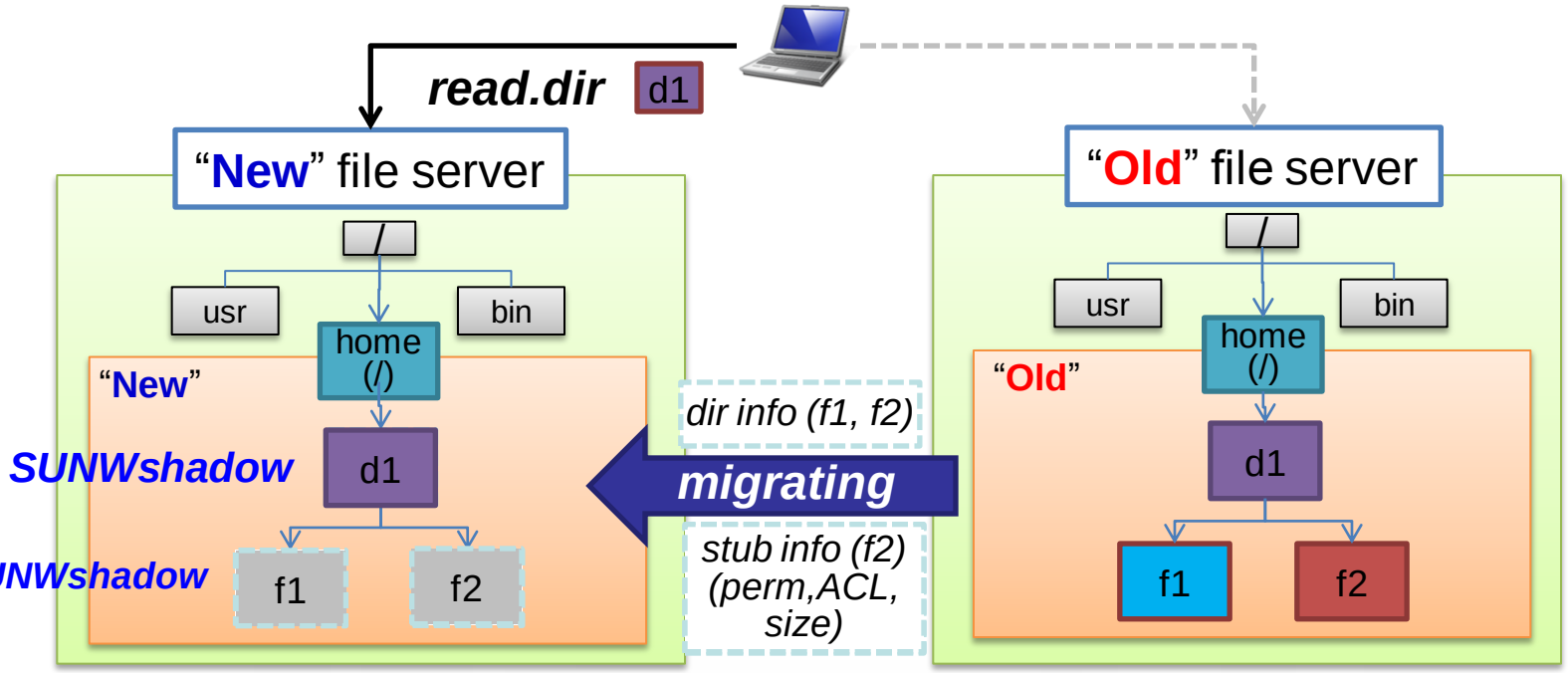
Shadow Migration

- **New** file system is **shadow** of **old** file system
- **Instant switchover**: Data would be all accessible & modifiable thru the new one once created



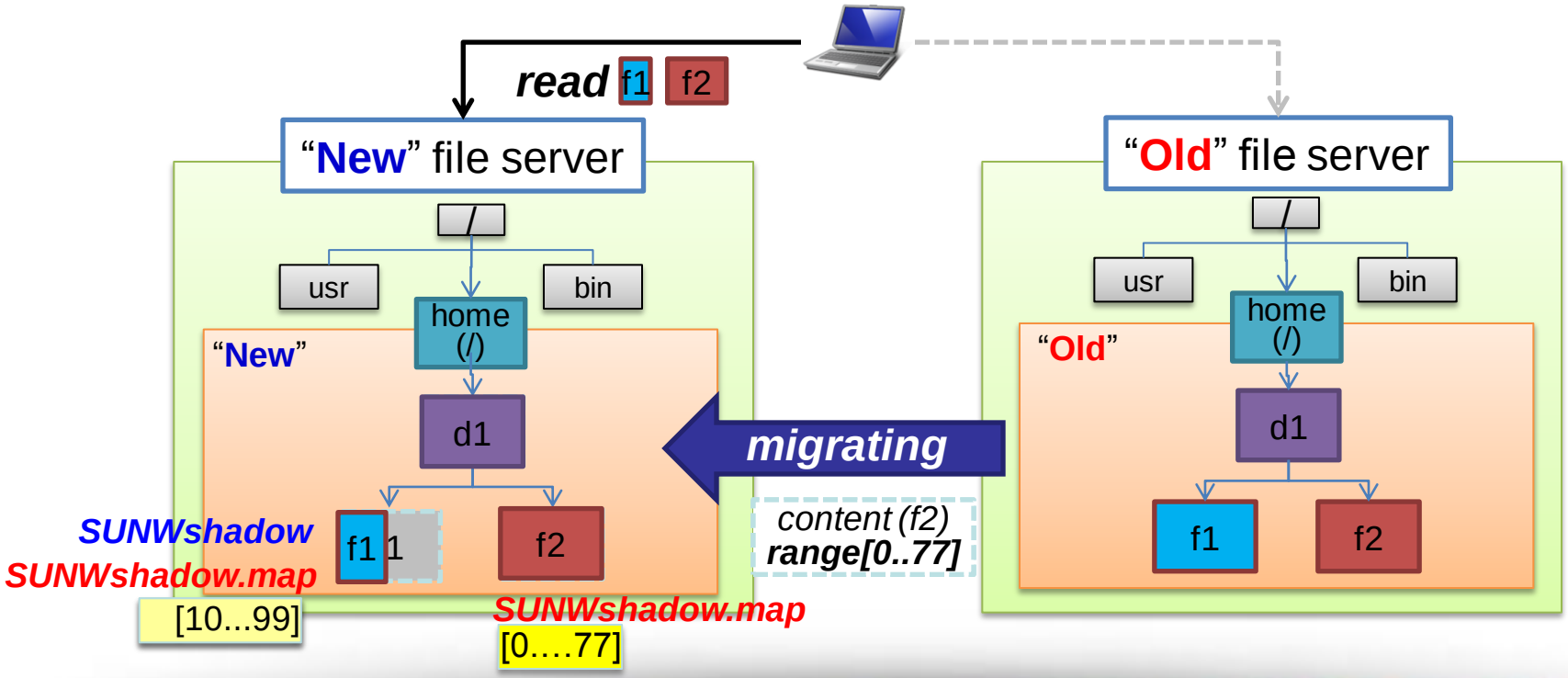
Migrating Directory (Read Dir)

- Shadow directory has ***SUNWshadow*** attribute
- Block while "*stub*" (***no content***) of all entries under the directory is migrated



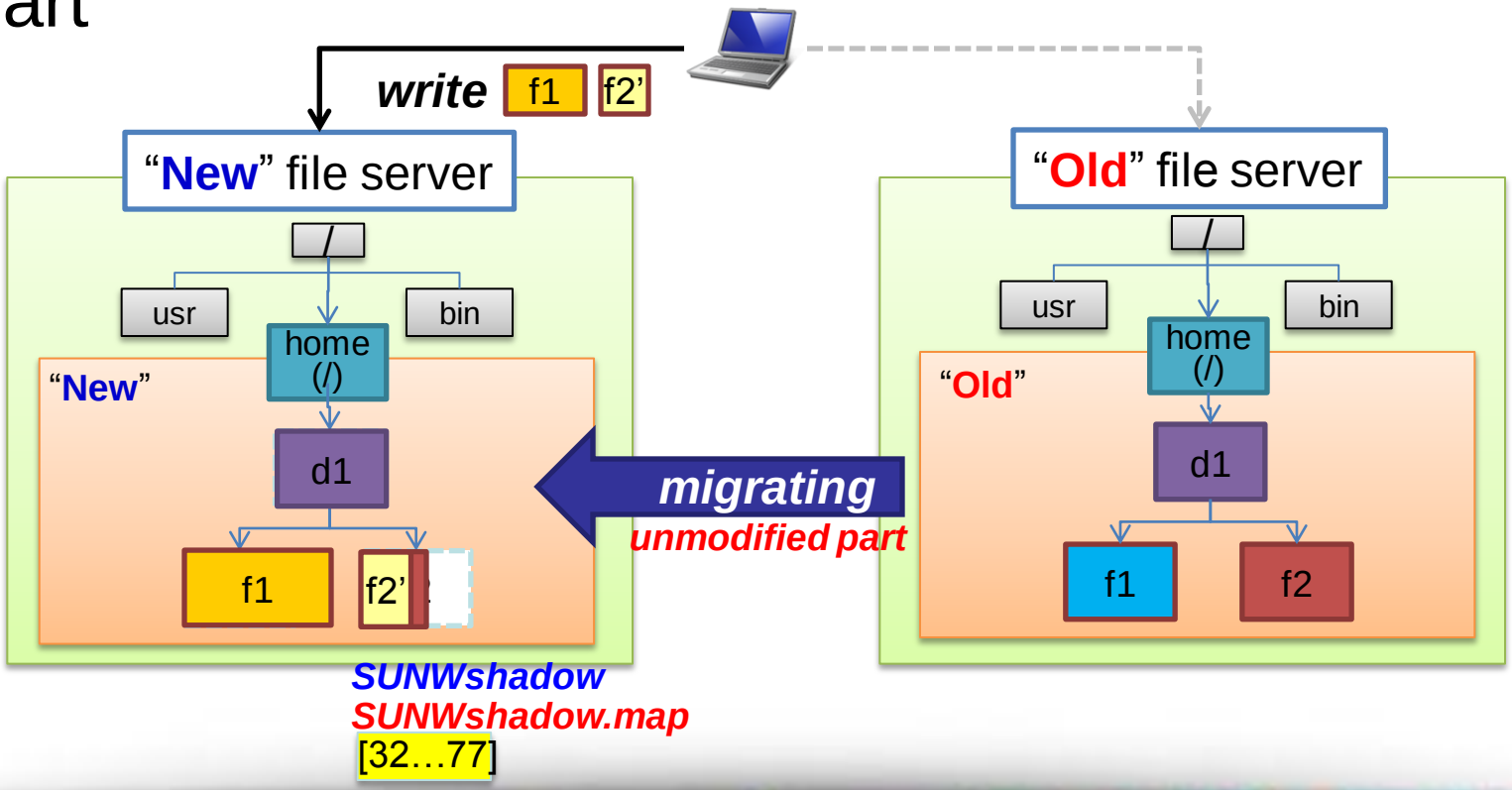
Migrating Regular File (Read File)

- Shadow file additionally has ***SUNWShadow.map***
- Block while “some” or “all” of data is transferred
- Migration completes when map gets empty



Migrating Regular File (Write File)

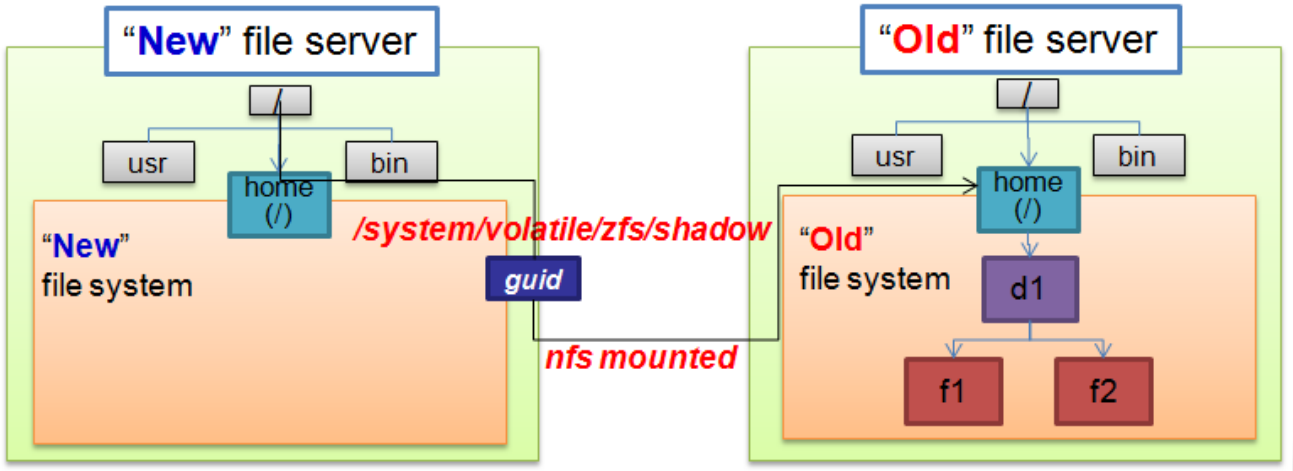
- Full replace or truncate means nothing to migrate
- Partial write requires blocking to read unmodified part



Connecting Old FS to New FS

- Original (old) file system is mounted at the same time we create or mount our ‘shadow’ (new) of it
- Shadow source (old) is established in a URI descriptor property of the target dataset at creation

```
zfs create –o shadow=nfs://old-server/export/old-fs tank/new-fs
zfs create –o shadow=file://export/old-fs tank/new-fs
```



Shadowd – Background Migration

"Reading all the entries from top (root) to bottom in background efficiently & reliably"

"while hoping to migrate entry prior to its access"

- Multiple worker threads (up to 64) to migrate data
- Trying to migrate more recently accessed one first
- Resumable from the point where interrupted
- I/O timed out when nfs source gets irresponsive
- Ensure migrating everything & cleaning up when finished

Talk Outline

- Use cases of shadow migration in ZFSSA
- Shadow migration at a high level
- **Optimization issues for higher performance**
- Summary

In General, Shadow Migration Performance "Depends on"

- Performance of the legacy device
- File size & # of files
- Access pattern of existing data during migration
- Network contention & configuration
- Destination storage pool configuration
- Log & cache configuration

Current Performance-aware Design

- Multiple worker threads for background migration
- Skipping holes (zeros) when migrating a file (becoming a sparse file at a new fs)
- Prioritized background migration (more recently accessed files/directories first)

Conversely, for higher reliability, migrated data & attributes are written to disk synchronously

Toward Higher Performance

Low Hanging Fruit

- More worker threads (64+)
- For synchronous writes,
  - Use log disks (write-back cache) in storage pool
  - Collapse multiple synchronous writes into one
- Network performance optimization
  - Use dedicated network
  - Tune TCP parameters & use different MTU size

Toward Higher Performance

High Hanging Fruit

- Speed up big directory migration (1M small entries)
- Faster big file migration (one huge entry)

  ⇒ ***Single-threaded***

  ⇒ ***no improvement with more threads***

- *Smarter prioritized background migration*
- *Pre-staged data migration (hybrid approach)*

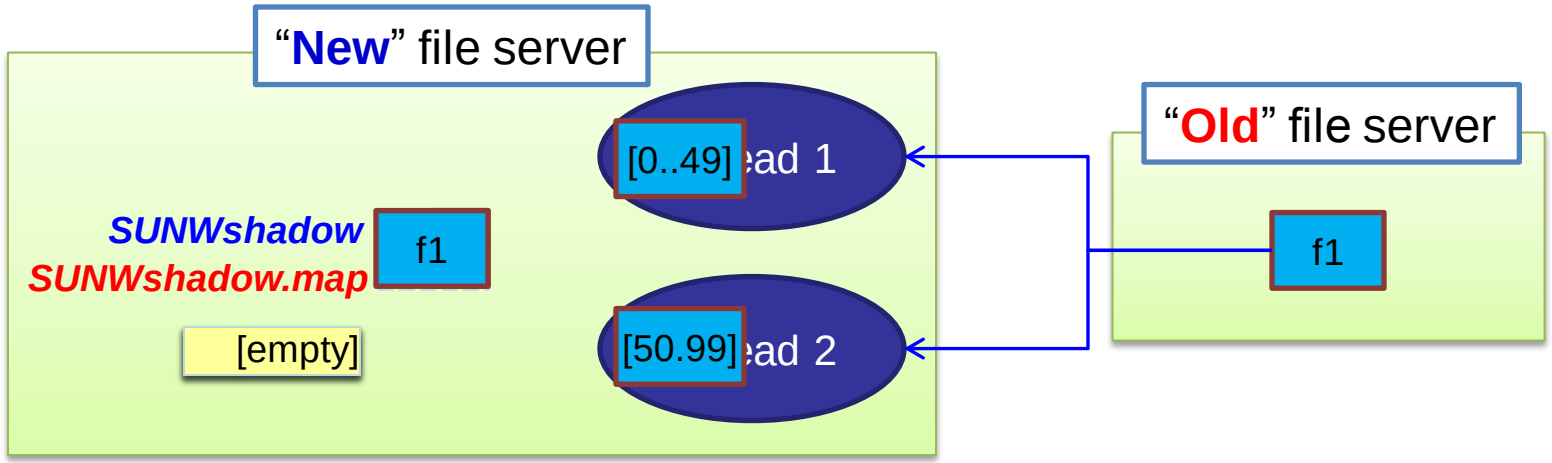
Multi-threaded Big Directory & File Migration

Since threads are shared resource:

- ***When to use multiple threads?***
  - Multiple threads might not be needed if file size or # of entries is less than threshold
- ***How many threads are used?***
  - *Using more threads does not help if the performance has been already saturated* (due to serialization)

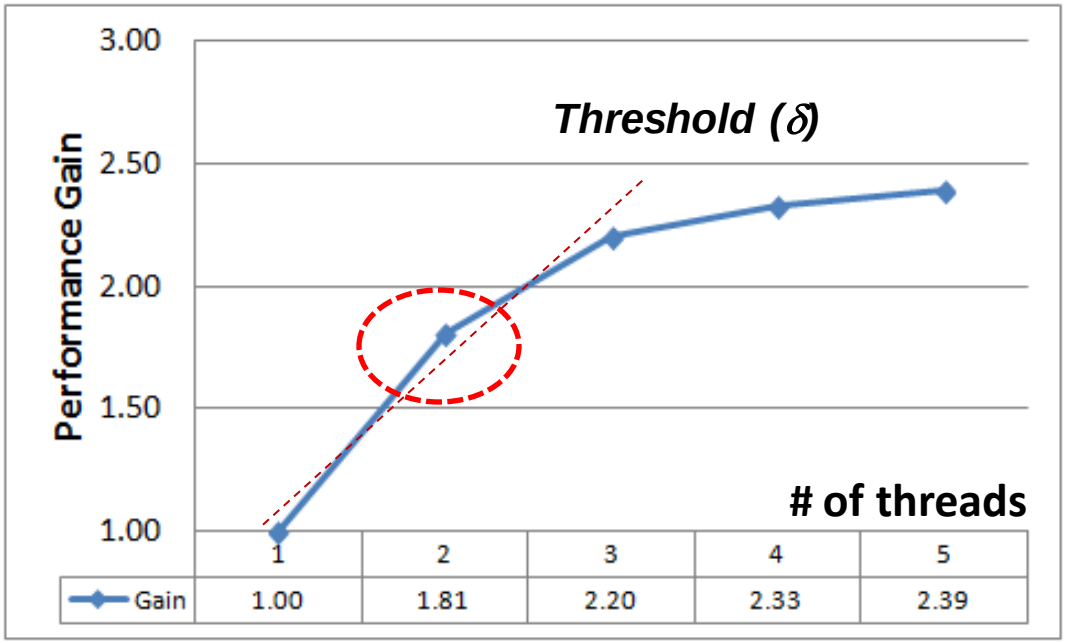
How Many Threads are Used?
Big File Migration

- Reading data from source runs in parallel
- The remaining part is ***mostly serialized***
  - Writing migrated data to a file (synchronous write)
  - Updating SUNWshadow.map (synchronous write)



Big File Migration
Performance Improvement

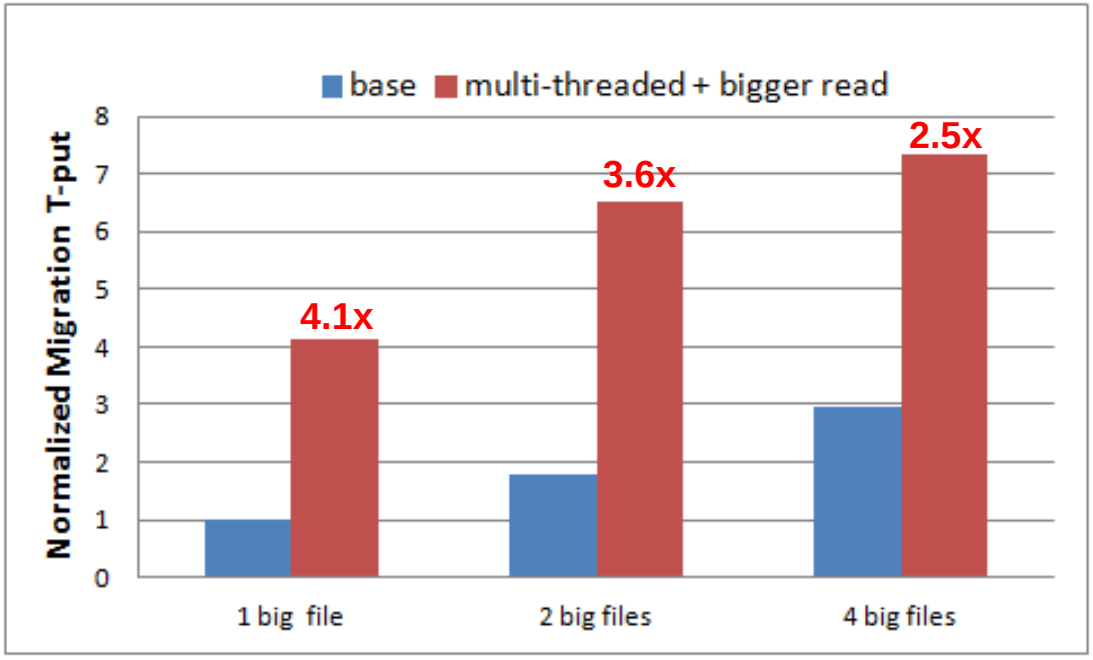
- No substantial gain observed from 3 threads
- Max. 2 threads per migration (1.8x gain)



Two ZFSSA's (ZS3-2) connected thru 10GbE Storage pool (**DE2-24P**) with log disks

Big File Migration
Performance Improvement

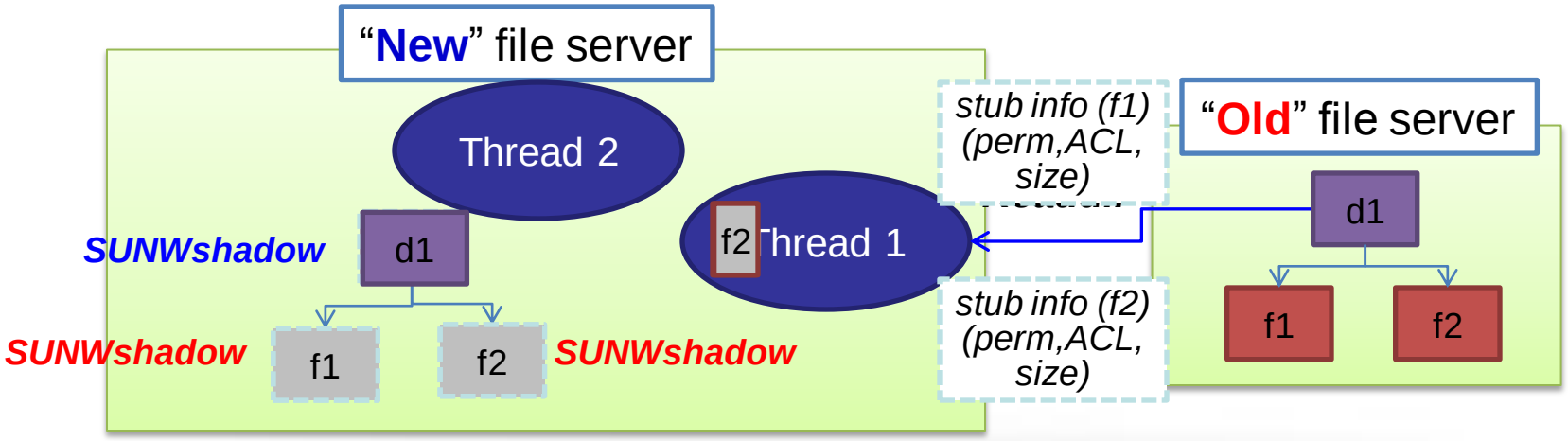
- Two threads per migration (1.8x)
- *Larger buffer for data read from source (KB → MB)*



8 threads
in shared pool
by default

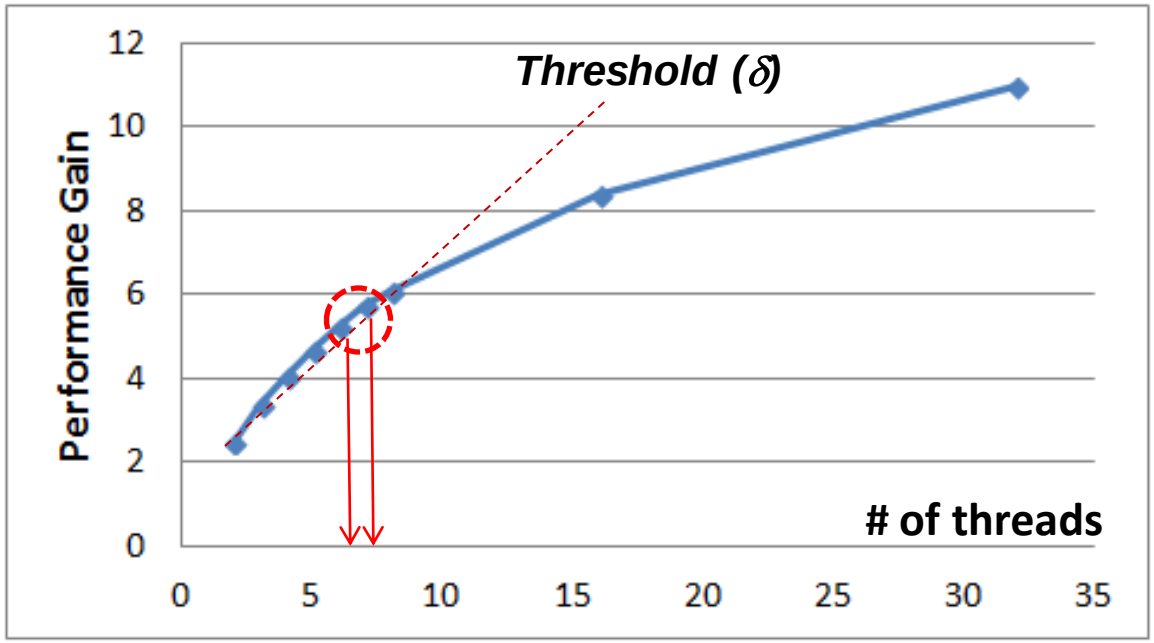
How Many Threads are Used?
Big Directory Migration

- Reading dir entries (VOP_READDIR) from the source is currently serialized
- The remaining part ***mostly runs in parallel***
  - Each directory entry can be migrated concurrently
  - Creating the entry itself & creating SUNWshadow



Big Directory Migration
Performance Improvement

- Scales better than big file migration (~6x gain)
- Using 6-7 threads is a sweet-spot



Another Approach (Verification) Determining # of Threads to be Used

- Given each file or directory migration function consisting of [s,p], where
  - s : the time that should be serialized
  - p : the remaining time that can be parallelized

- Minimum # of threads $n_{min} = max\{n \mid G'(n) > \delta\}$ to assure a given performance gain slope (δ), where
  - G(n) : expected performance gain with n threads is computed from [1+d]/[1+d/n], where d=p/s
  - G'(n) : first order derivative (slope) of G(n)

Toward Higher Performance

High Hanging Fruit

- *Speed up big directory migration (1M small entries)*
- *Faster big file migration (one big-sized entry)*

- Smarter prioritized background migration
  - Presently, more recently accessed files/directories first
- Pre-staged data migration
  - Synchronization + Interposition approaches

⇒ ***Almost "zero" additional I/O delay***

Talk Outline

- Use cases of shadow migration in ZFSSA
- Shadow migration at a high level
- Optimization issues for higher performance
- **Summary**

Shadow Migration

- ***Software-based interposing device*** (built-in feature in ZFSSA) with ***instant switchover*** for data migration
- Also available in Solaris
- Has been **evolving** with new features & higher performance

- ***Pre-staged*** data migration in research

Thank you!

Toward High-Performance Shadow Migration

Youngjin Nam, Aaron Dailey
(Special Thanks to Tim Haley)

ORACLE
ZFS STORAGE APPLIANCE