Lustre Replication and Migration Tool

DataDirect Networks Japan, Inc.

Shuichi Ihara (sihara@ddn.com)

2016/09/20

Background: Replication and Migration

- **Data backup is important**
  - Many backup and replication requirements
  - Lustre needs to be migrated from old to new system
  - Time to think backup for few PB Lustre system
  - Lustre Replication is not ready yet.
- **Lustre tiering is possible**
  - Data Migration from Fast HDD (or SSD) to slow disks is possible
    - Different from fully automated HSM
  - Lack of user space utilities

Lustre data replication and migration

- **Data replication (Backup)**
  - Two independent and different namespaces (file system)
  - Asynchronous file level replication
- **Data migration**
  - Migrate data from a storage device to another type of device (e.g. SSD or fast HDD to slow HDD)
  - Keep same metadata and namespace
  - Application access data transparently even after migration

Introduce two utilities (ldsync and ldmigrate) for data replication and migration in Lustre.

Challenges on Backup and Replication

- **Two major challenges on Backup and Replication at large file system**
  - **"Delta" detection between file systems**
    - Determined by file attributes (mtimes, size, checksum, etc)
    - Depends on how many files are in the file systems
  - **Data transfer time**
    - Copying many large files, as well large single shared file
    - Need efficient resource allocation and maximize utilization

Major Copy Tool : RSYNC and DCP

- **RSYNC**
  - Has been be maintained more than 10 years and packaged in Linux distribution.
  - Supports many features, but lack of parallelization
- **DCP (part of fileutils http://fileutils.io)**
  - Designed for scalability and performance
  - Started as collaboration efforts among several large US laboratories and DDN, that was involved at the beginning
  - Support MPI and any POSIX file system
  - Manage chunk of file and efficient MPI rank allocation for copy and maximize resource utilization

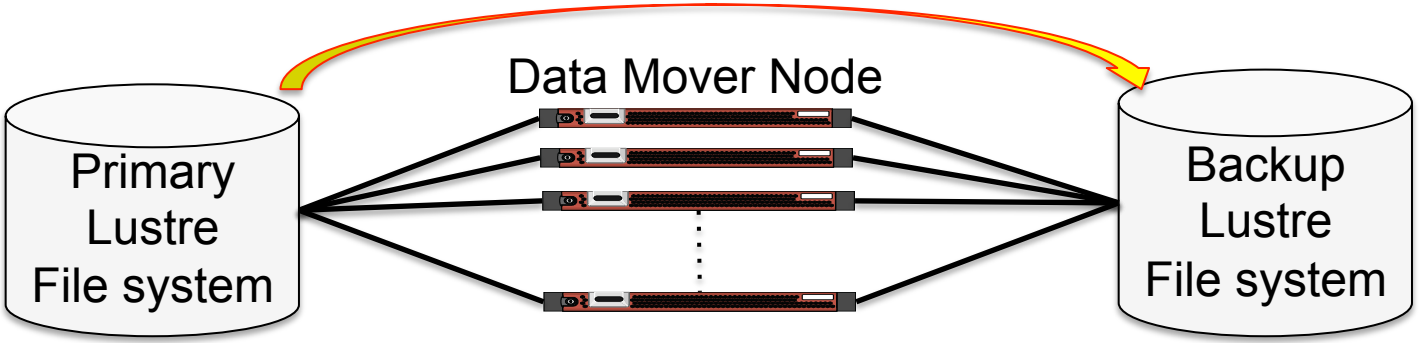
Accelerate File system's delta detection

- **"Diff" detections of two file systems**
  - RSYNC takes few hours for delta detection at tens of millions of inode in the file system
  - DCP (DCMP) in fileutils is much faster, but still consume a lot of time and metadata pressure
- **Lustre Changelog rescue**
  - Lustre Changelog records events the file system namespace or file metadata. (Timestamp, FID and operation)
  - Keep in MDT and it can fetch from Lustre client
  - No more file system scanning except initial copy!

ldsync : Parallel synchronization tool based on Lustre changelog

- **Similar tool lustre_rsync is exist, but...**
  - Single thread and still based on rsync
  - Partial changelog support
- **ldsync is a replacement of lustre_rsync**
  - A parallel synchronization framework includes Lustre changelog analyzer
  - Changelog analyzer walk through Changelog and invoke minimum stat() call to determine files to be synced
  - Flexible backend copy tool support (Use DCP for now, but any native copy tool possible)

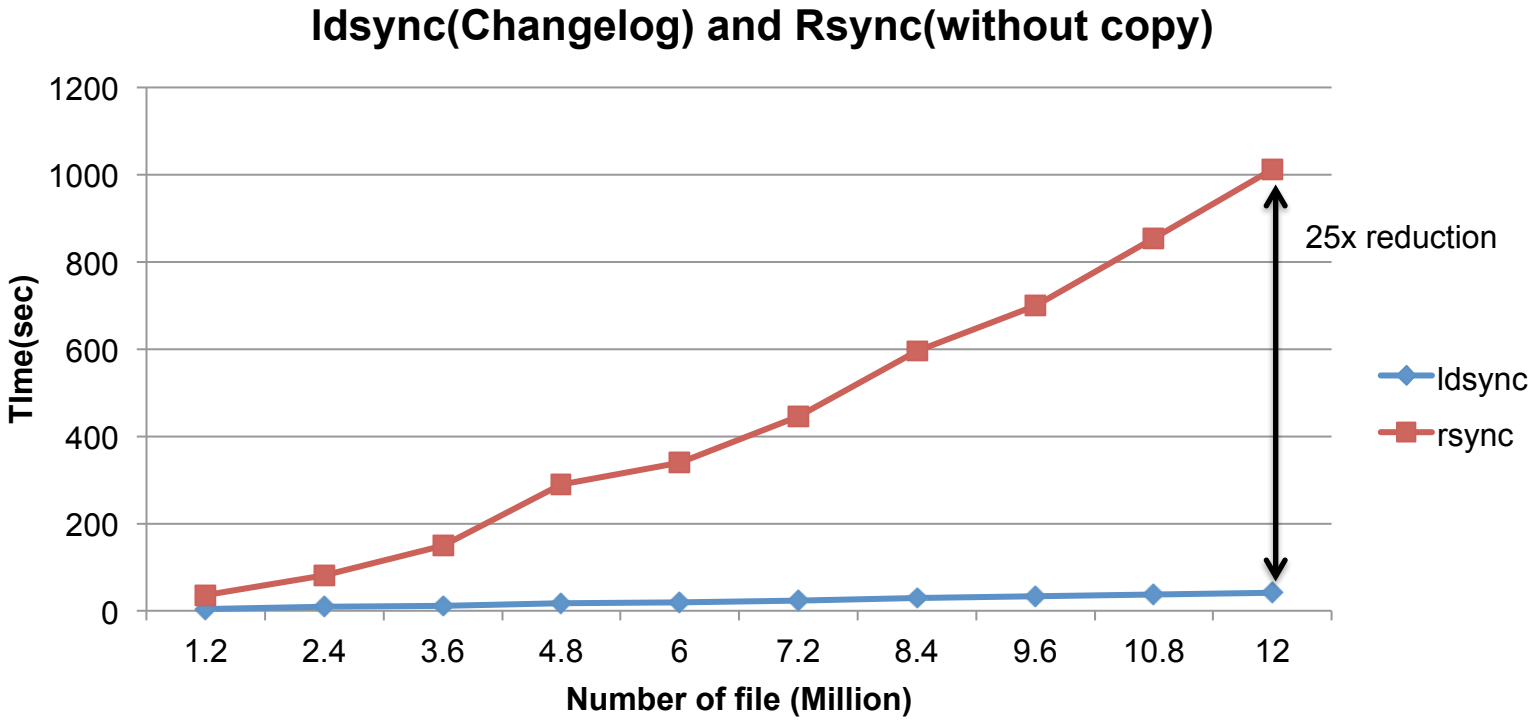
Architecture of ldsync



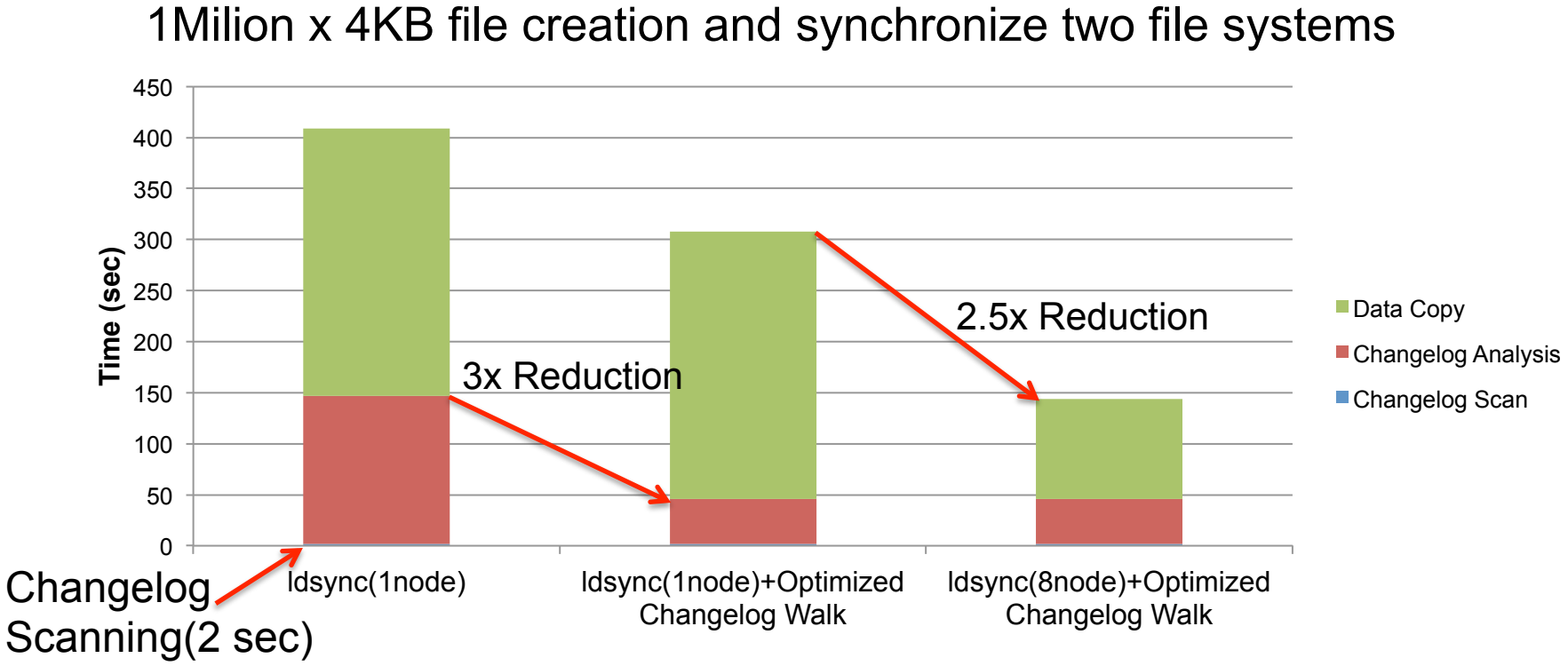
Data Mover mounts both primary(ro) and backup file system

1. Fetch Changelog from MDT and analyze
2. Minimum stat() call to MDS to get additional metadata information
3. Copy files by "dcp" and also unlink files from backup file system by "drm"
4. Clear old Changelog

Delta detection speed in two file systems

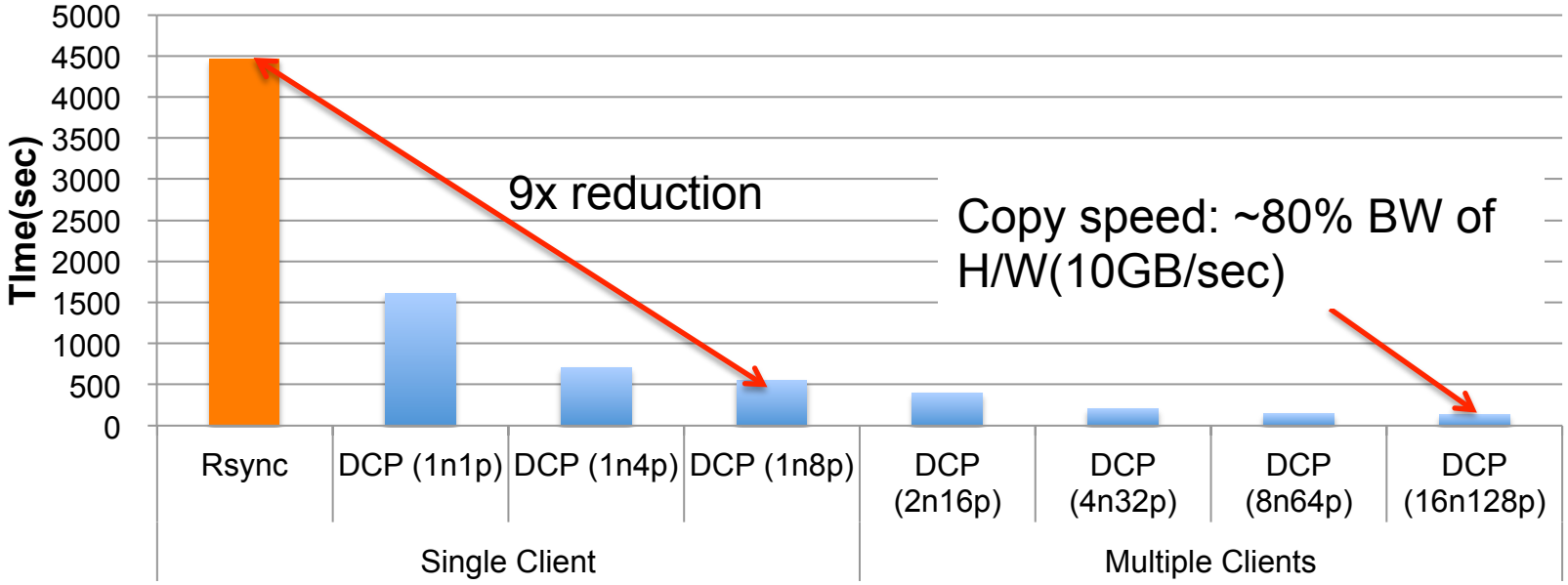


ldsync : Experimental performance (Many small files)



ldsync : Experimental performance (1TB single shared file)

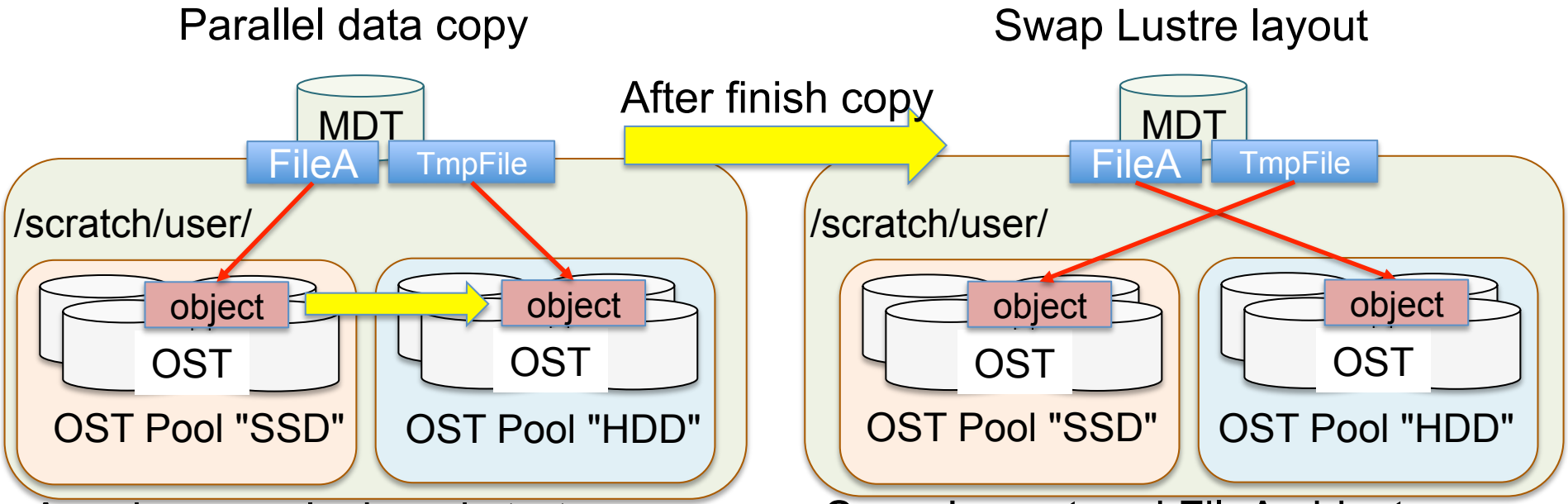
Primary System : 1 x DDN ES7K(140 x NLSAS), 2 x FDR
Backup System : 1 x DDN ES7K(140 x NLSAS), 2 x FDR



ldmigrate : Parallel data migration tool for Lustre

- **Keep metadata, but change OST object placement**
  - e.g) Migrate data from SSD OST pool to HDD
  - "lfs migrate" can do it, but limited scalability
- **ldmigrate**
  - Migrate OST objects to another OSTs (OST pool) based on Lustre data layout (determines how to place data to OSTs)
  - Parallelization and scalability
  - Integration with Job Scheduler is possible

Architecture of ldmigrate



Acquire grouplock and start copy data to tmp files by dcp

Swap Layout and FileA objects are now in OSTs of OST pool "HDD" and remove tmp files

How Setup Tiered OST Pool and Migration

- **Creating OST pool for different type of device**

  [root@mds ~]# lctl pool_new scratch.SSD
  [root@mds ~]# lctl pool_new scratch.HDD
  [root@mds ~]# lctl pool_add scratch.SSD OST[0-9]
  [root@mds ~]# lctl pool_add scratch.HDD OST[a-13]

- **Assign OST pool to directory**

  [root@client ~]# lfs setstripe -p SSD /scratch/user
  [root@client ~]# lfs getstripe -p /scratch/user/file*
  SSD

- **Copy and layout change**

  [root@client ~]# ldmigrate -g 100 -m -o SSD /scratch/user /scratch/tmp
  [root@client ~]# lfs getstripe -p /scratch/user/file*
  HDD

Conclusions

- Introduced ldsync and ldmigrate for backup and data migration.
- Demonstrated huge performance improvements compared to existing tools and techniques.
- Still investigating several performance optimization and stability. It also require more tests
- These tools still under private repository at Github, but we plan to publish as open source or push patches to fileutils.

Thank you!