Roads to Zester

Ken Rawlings Shawn Slavin Tom Crowe

High Performance File Systems
Indiana University

Introduction

- Data Capacitor II
  - IU site-wide Lustre file system
  - Lustre 2.1.6, ldiskfs MDT/OST, 5PB, 1.5 billion inodes
- Reporting needs across entire file system
- Current system
  - Lester + Lustre stat()
  - Struggles since breaking 1 billion inodes
- Upcoming file system at IU
  - Lustre 2.8, ZFS MDT/OST
  - Larger than Data Capacitor II

Lester

- Lester, the Lustre lister
- Written by David Dillow at ORNL
  - https://github.com/ORNL-TechInt/lester
- Generates Lustre file list with metadata information directly from ldiskfs
- Easily parseable text file
  - path, (a,c,m)-time, mode, UID, etc.
  - 1481481220|1481481220|1481481220|1486829|601|100666|3584|175901125|||/ROOT/projects/foo/bar.txt
- No equivalent for ZFS

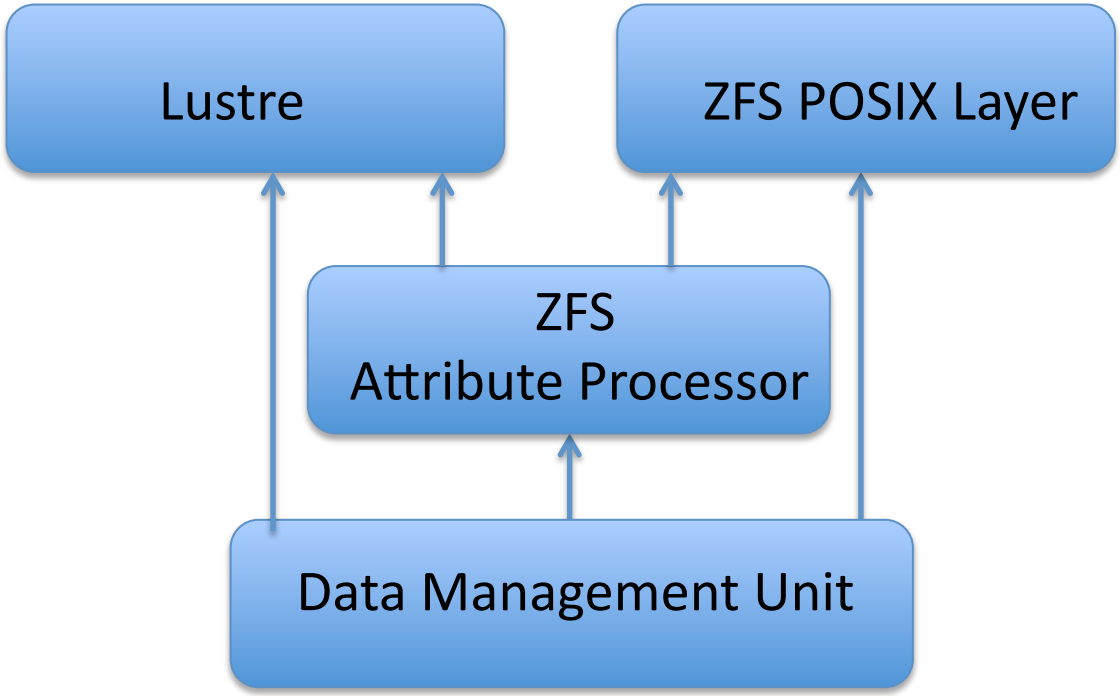
Goals

- Equivalent to current solution for ZFS
  - Without Lustre stat() if possible
- Focus
  - Regular files
  - Path, UID, GID, Mode, Timestamps, & Size
- Two Stages
  - Gather Lustre metadata from underlying ZFS layer
  - Assemble MDT/OST information and compute file sizes
- Remain mindful of scaling needs

Priorities

- Doesn't need to be perfect
  - Understood error bounds
- Faster than Lustre stat()
  - Over a billion files measured in days not weeks
- Not parasitic on filesystem
- No custom code run as root on OSS/MDS

ZFS & Lustre


Lustre
ZFS POSIX Layer
ZFS
Attribute Processor
Data Management Unit

ZDB

- Standard ZFS utility
  - Dumps information about ZFS pools and datasets
- 'zdb -dddd <dataset>' outputs dataset objects information
  - Path, Timestamps, Size, GID, UID, mode, etc.
  - Includes Extended Attributes
    - MDT: trusted.lov
    - OST: trusted.fid
- Somewhat challenging to parse

ZDB MDT Dataset

- Provides Lustre metadata information
- Need object information for OST lookup to compute files sizes
  - Requires decoding of trusted.lov

```
Object  lvl   iblk   dblk   dsize  lsize   %full  type
199      1    16K   128K  1K     128K   0.00  ZFS plain file
path    /ROOT/testfile
uid     121
gid     12
atime   Fri Oct  9 19:56:47 2015
<...>
trusted.lov = \320\013\321\013\001\000\000\...
```

Decoding trusted.lov

- Tom Crowe's trusted.lov Extended Attribute decoding script
  - Provides (ostidx, objid) object pairs
- Needed EA translation from ZDB format
- zfsobj2fid utility
  - Written by Christopher Morrone at LLNL
  - trusted.fid decoding from ZDB dump
  - Available on Lustre ZFS from 2.8 forward
  - Includes general ZDB EA translation logic

ZDB OST Dataset

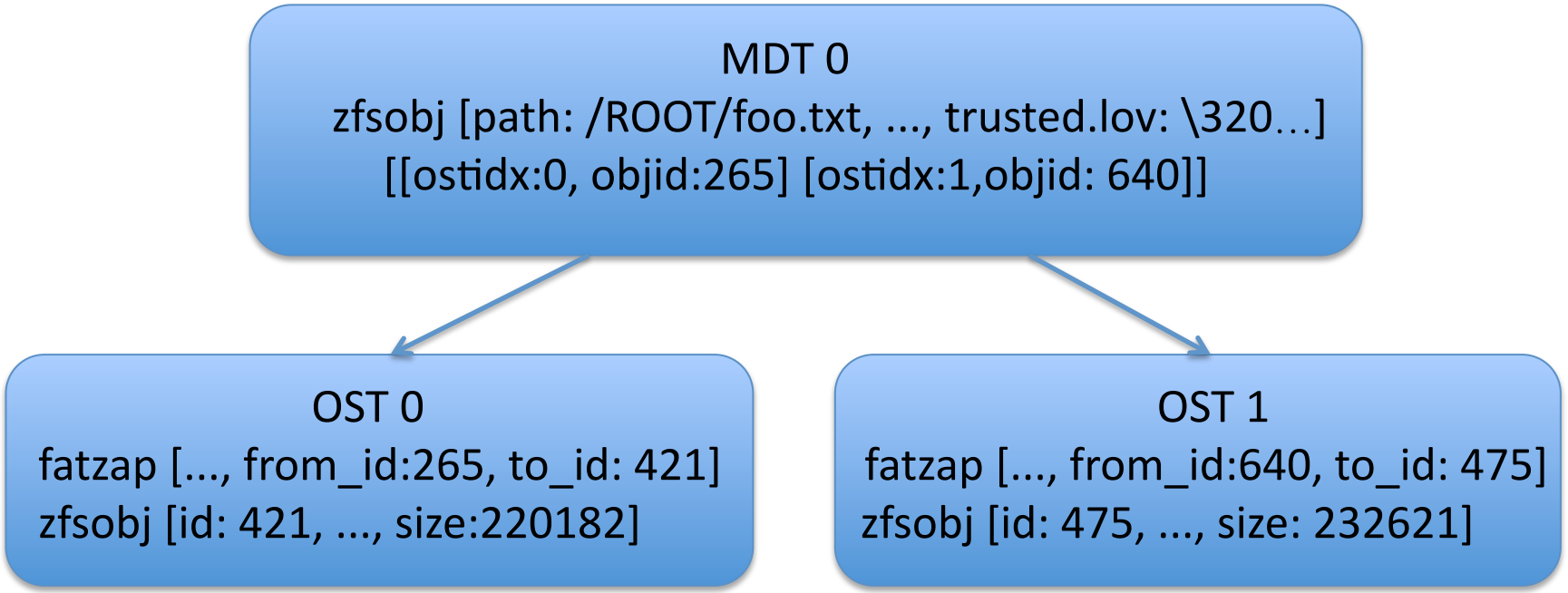
- Parent object FatZAP has key-value pair for objid lookup:

```
Object  lvl   iblk    dblk  dsize    lsize   %full    type
129       2    16K    16K  16.5K    32K  100.00  ZFS directory
<...>
Fat ZAP stats:
   265 = 421 (type: Regular File)
```

- Target ZFS object has trusted.fid EA and size

```
Object  lvl   iblk    dblk   dsize    lsize     %full      type
421     3    16K   128K  269M   269M    100.00  ZFS plain file
<...>
size 220182
trusted.fid = \000\004\000\000\002\000\000\000
```

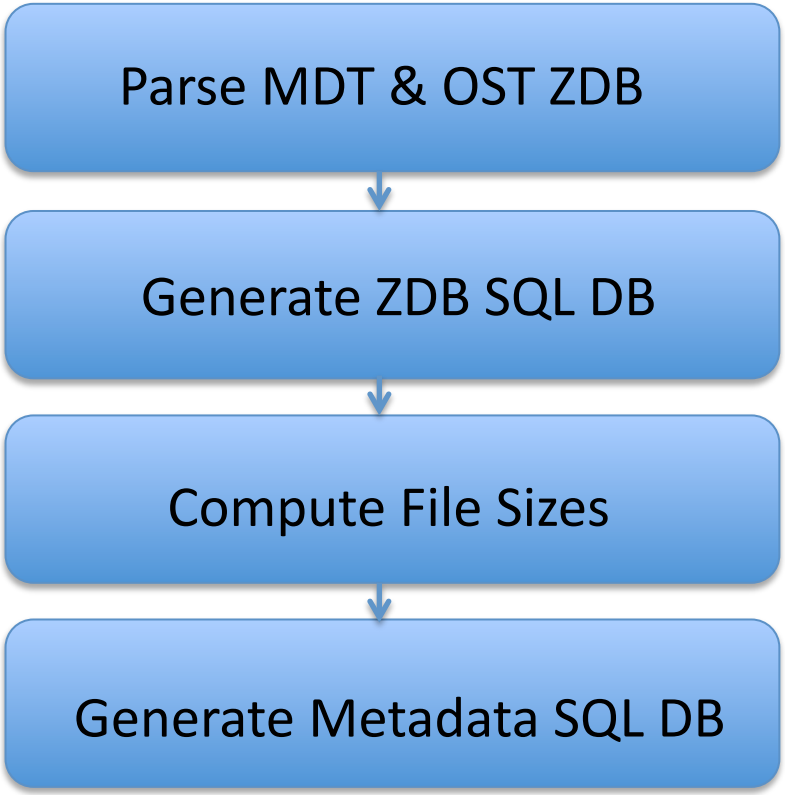
File Size Example



Implementation

- Named Zester as homage to Lester
- Written in Python 2.7
  - Evaluating future Python 3.x move
- Started with in-memory representation
  - Worked well for experimentation and initial scaling
  - Problematic past 1 million files
- Moved to SQLite as primary data representation
  - Python DB-API 2.0
  - Portable SQL

Zester Overview



ZDB Parsing

- Each MDT & OST ZDB dataset dump parsed into separate SQLite DB file
- Some parsing challenges, robust so far
- Generated DB schema
  - zfsobj [id, path, uid, gid, size, ..., trusted.fid, trusted.lov]
  - fatzap [id, from_id, to_id, file_type]
- Output
  - mdt_<idx>.zdb -> mdt_<idx>.db
  - ost_<idx>.zdb -> ost_<idx>.db

Metadata DB Generation

- Loop over all MDT ZFS objects with trusted.lov
  - Decode trusted.lov into set of (ostidx, objid) object pairs
  - For each pair
    - Translate objid to target OST ZFS object using FatZAP
    - Query target OST ZFS object for size
  - Sum object sizes
- Metadata DB
  - Represents file from Lustre viewpoint
  - metadata [path, size, mode, gid, atime, ctime, ...]
- Output
  - mdt_<idx>.db ost_<idx>.db ... -> metadata.db

Testing

- Create test files on Lustre filesystem
  - Various modes, sizes, stripes, path depths, etc.
- Lustre stat() all test files
  - Generate canonical metadata DB to test against
- Compare Zester metadata DB and canonical metadata DB
  - Currently path, mode, UID, GID, size, atime, mtime, and ctime

Current Status

- Work in progress
  - Promising Results
- No metadata errors into millions of files (including size)
  - Allow variance of 2 seconds on timestamps
    - Available timestamp precision low
- Lustre 2.8 focus
- Scale-up currently limited by testing infrastructure
  - Expect limit of testing tens of millions in reasonable time
- Will test & scale further against new file system once available

Scalability

- Mindful of billion-scale file need, no known showstoppers
- Currently processing thousands of objects per second
  - Consistent with billion objects measured in days not weeks
- Parsing parallelizable across OSTs/MDTs
- Profiling
  - ZDB parsing CPU limited by strptime()
  - Low process memory usage
- Move to DB server straightforward when/if necessary
- Python remains promising
  - C extensions where needed

Zester ZDB DBs

- Queryable DB of ZFS layer underlying MDTs & OSTs
  - ZFS Lustre "under the floorboards"
- Already proven valuable
  - Investigating alternate verification approaches
- Useful for more than just reporting
  - Filesystem forensics, etc.
- Stored information focused on project needs
  - Will add more moving forward

Future Work

- Continue scale-up and testing
- Test with additional Lustre versions
- Add other file types
- Formalized unit & integration testing
- Source code investigation
  - ZDB
  - Lustre ZFS OSD
- Adapt as Lustre changes
  - Layout Enhancement

Thank You!

- Your time and attention is appreciated
  - Feedback and suggestions: hpfs-admin@iu.edu
- Lustre community
- High Performance File Systems @ IU
  - Tom Crowe, Chris Hanna, Nathan Heald, Nathan Lavender, Ken Rawlings, Steve Simms, Shawn Slavin
- Source Code
  - https://github.com/iu-hpfs/zester
  - GPL2 licensed, collaborators welcome!
- Questions?