The Vera C Rubin Tape RSE for Backups

ePIC Streaming Computing Model WG

April 22, 2025

Andrew Hanushevsky, SLAC

http://xrootd.org

Produced under DOE contract DE-AC02-76SF00515 with Stanford University

The Instigator: Vera C. Rubin Observatory

- Simonyi Survey Telescope and LSST Camera (8.4 m primary, 3.2 Gpix, 6 filter bands)
- Legacy Survey of Space and Time (10 years, >18K deg^2 sky coverage, >825 visits per point)
- Finishing Construction, Pre-Operations, and Commissioning; Operations to start in mid 2025

Render

Reality (Dec 2021)

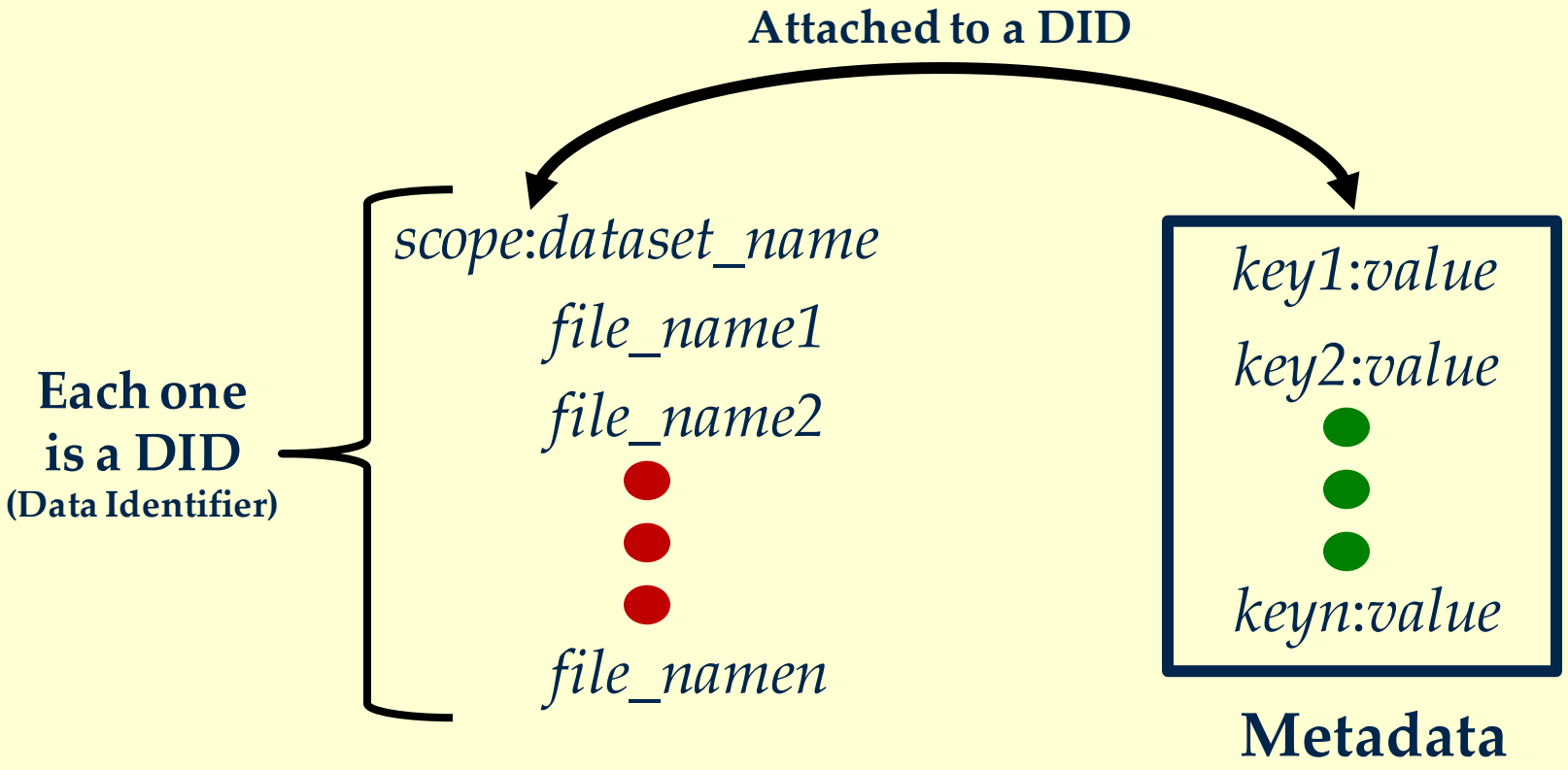
The LSST Backup Challenge

- ~15 Rucio datasets per night (4K-8K/year)
  - Up to 20K files per dataset
    - Size range from a few KB to ~1 GB
    - Average size of dataset is approximately 100 GB
  - All of which need to be backed up
    - About 8 PB/year
- Up to 12M additional dataset products/year
  - Each approximately 150GB
    - Undetermined number need backups

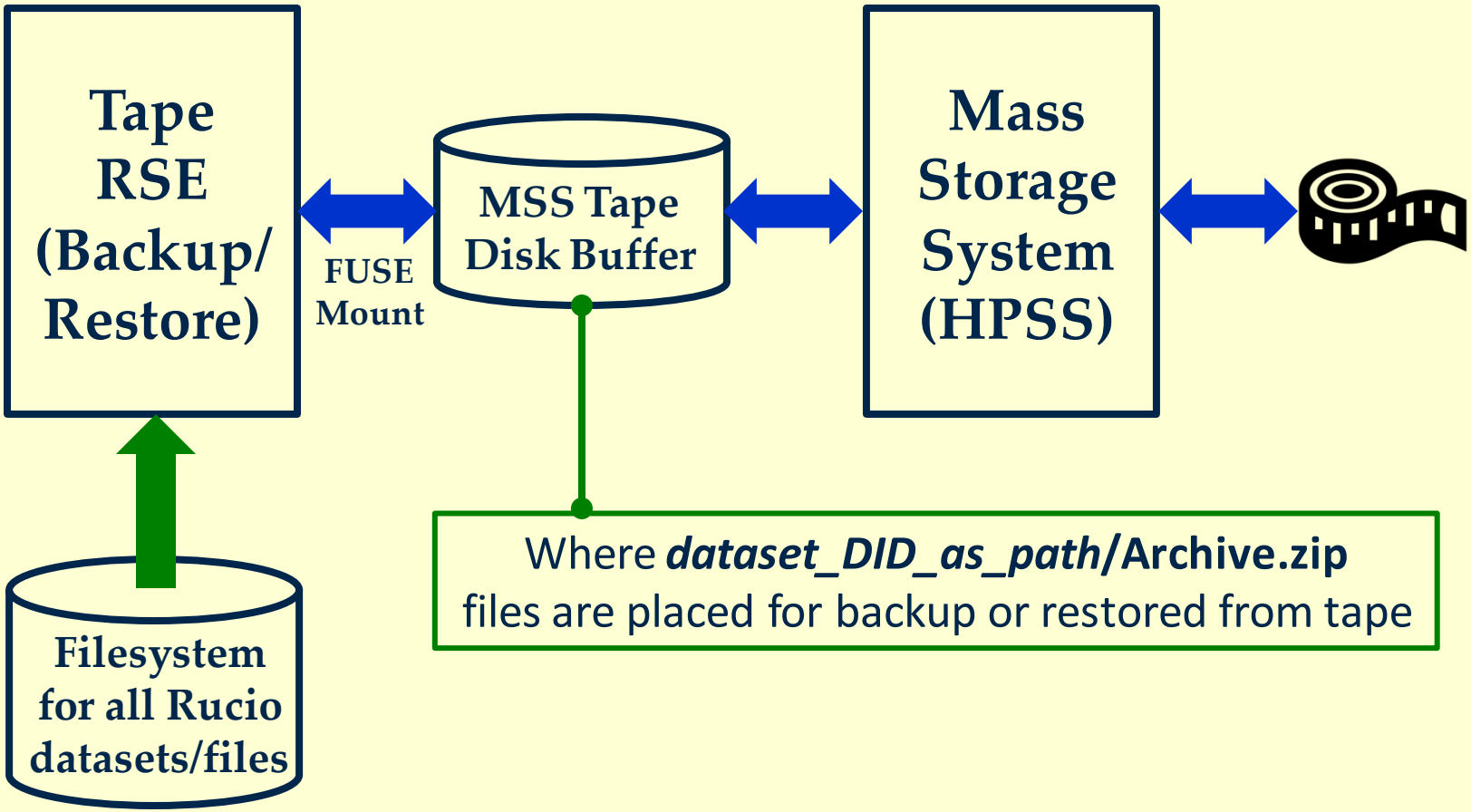
Backup/Restore Requirements

- Simple way to create a backup
  - No complicated submission interfaces
- 99+% assurance that a backup created
  - Ability to discover datasets not backed up
- Ability to restore
  - Full datasets
  - Individual files from a dataset

Rucio Conventions



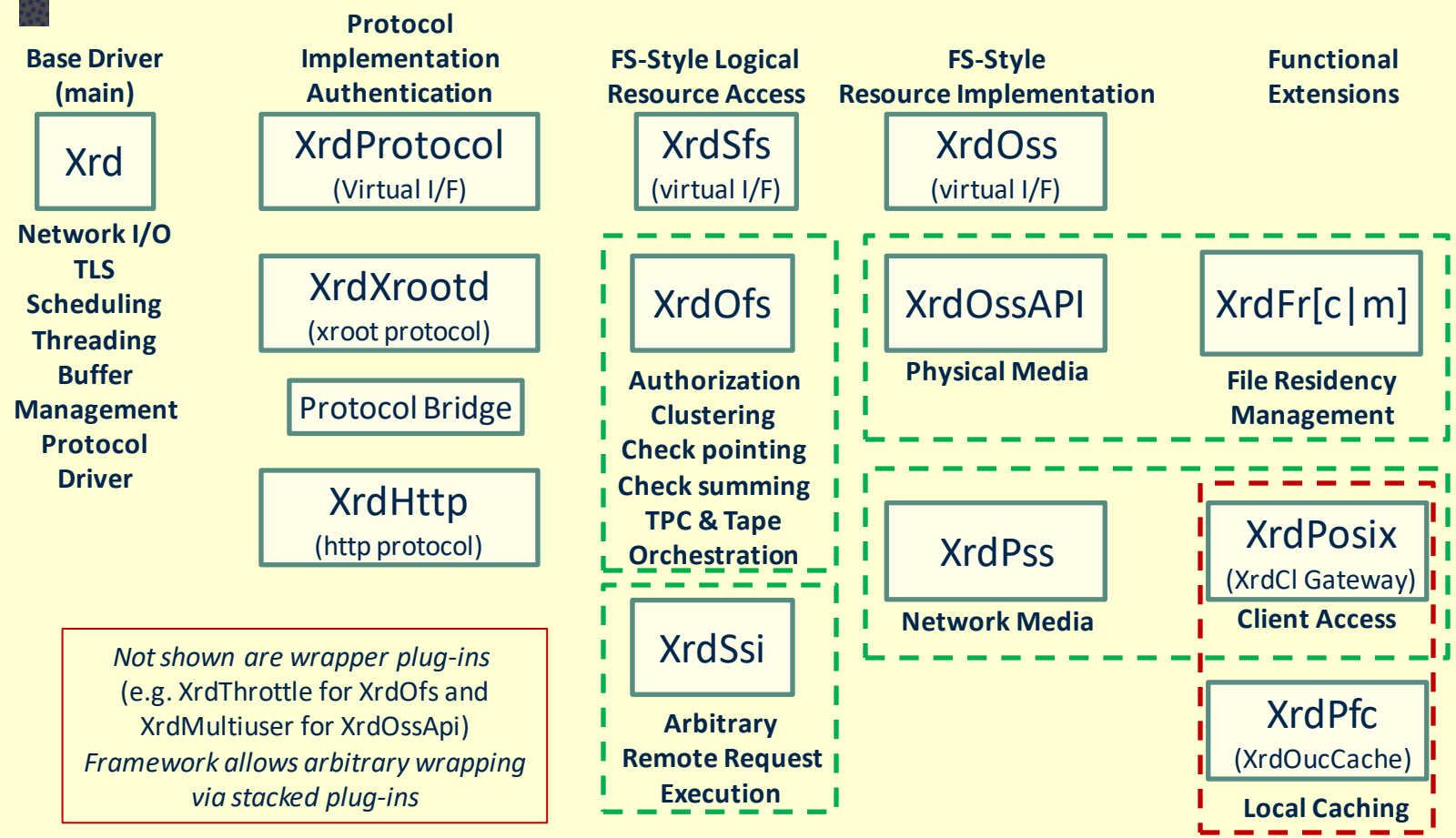
Backup/Restore Environment



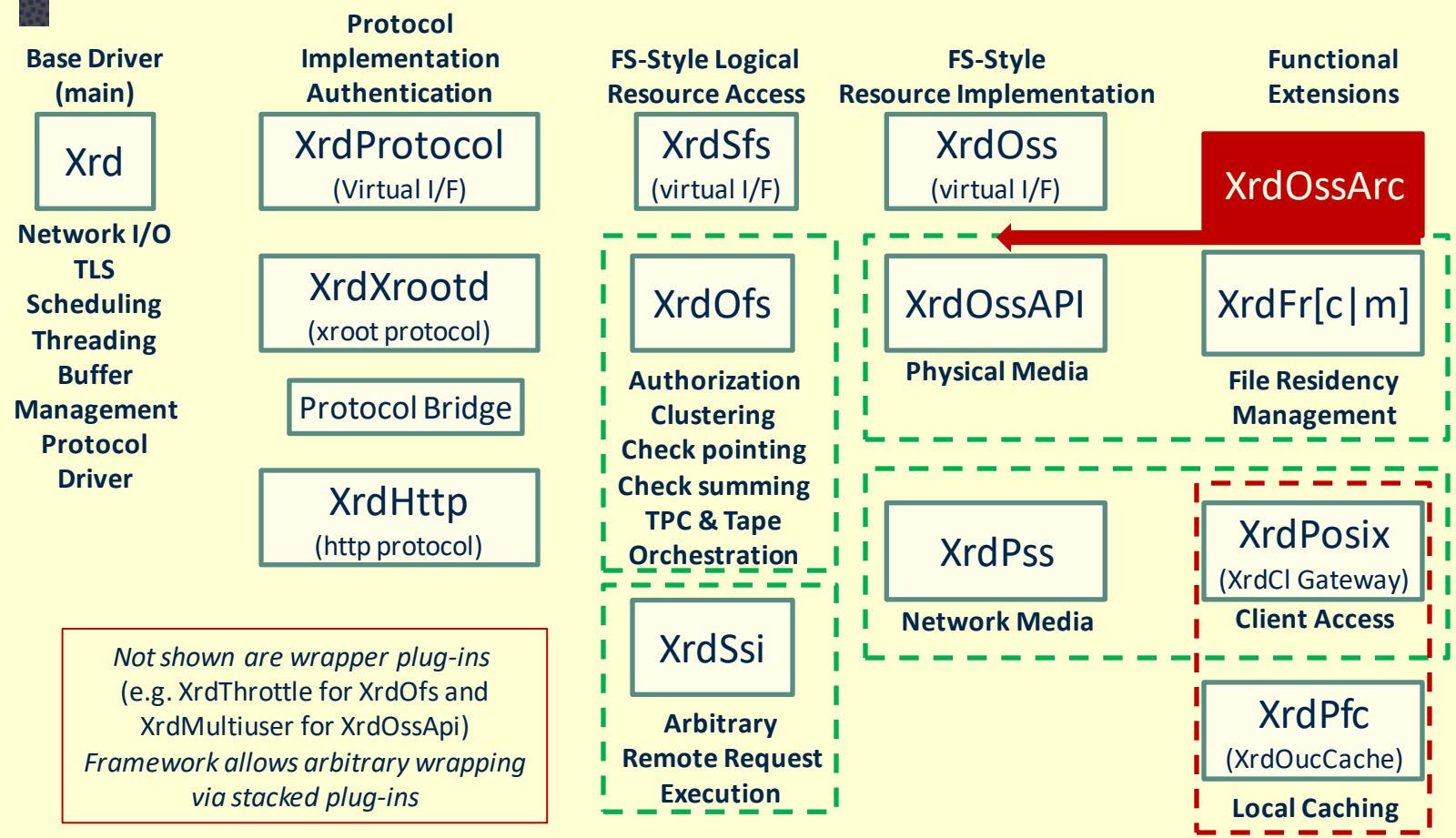
The Backup Approach

- Dataset associated w/ backup metadata
  - **arcBackup** is the metadata key
    - *taperse*:**need** - *taperse* needs to backup ds
    - *taperse*:**done** - *taperse* completed the backup
    - *taperse*:**fail** - *taperse* backup failed
- The tape RSE is a specialized xroot server
  - Standard server with extra plug-in
    - libXrdOssArc.so (configurable)
      - Orchestrates the backup and individual file restores

XRootD Plug-In Architecture



The Stacked OssArc Plug-In



Backup Orchestration

- Coordinates 3 Python scripts
  - XrdOssArc_Archiver
    - Creates zip file of dataset contents
    - Stages archive to the Mass Store System for backup
  - XrdOssArc_BkpUtils
    - Various Rucio dependent functions
      - Used to setup backup and finalize the backup
  - XrdOssArc_MssCom
    - Various MSS dependent functions

The Backup Steps

- Do Forever
  - Create list of datasets to be backed up
  - For each dataset using config scheduling
    - Setup logical dataset contents via symlinks
      - Default assumption all DID's are accessible to RSE
    - Invoke archiver to create and stage-out backup
      - Create *dataset_DID*/Archive.zip
        - Zip member names are the file DIDs
      - Move archive to MSS stage-out file system buffer
      - Cleanup by removing logical dataset
    - Finalize the backup by updating DS metadata

Backup Special Processing

- What if the dataset is just too large?
  - OssArc can be configured for split archives
    - **ossarc.arcsize** *optsz* [**range** *minsz maxsz*]
      - Dataset split into pieces *minsz* <= size <= *maxsz*
        - With attempt to be as close to *optsz* as possible
    - Each archive file named Archive*m-n*.zip
      - *m* is the file sequence and *n* is the total number of files
        - 3 files: Archive1-3.zip, Archive2-3.zip, Archive3-3.zip
    - A small ordinal metadata tag is added to dataset
      - **arcIndex** which allows for quick individual file restores
        - I.e. only the zip file containing wanted file is accessed

The Restore Approach I

- Individual files
  - Copy out file from the archive RSE
    - xrdcp xroot://*rse*//backup/*dataset_DID*~*file_DID dest*
      - Can also use your favorite HTTP copy program
      - Currently, using a tilde **(~)** to separate *ds* from *fn*
        - Only unassigned special character without UTF req
          - Done by Rucio lfn to pfn plugin for the RSE
      - Considering other alternatives like CGI "arc.fn="
    - Restore typically requires a stage-in from tape
      - Does not require unpacking the zip file
        - OssArc plug-in knows how to read/extract zip files

The Restore Approach II

- Full dataset
  - Copy out the full archive zip file (fast restore)
    - xrdcp xroot://rse//backup/*dataset_DID*~Archive.zip
      - Can also use your favorite HTTP copy program
    - Unzip archive in-place
  - Potentially use Rucio replicate (by file restore)
    - Would require registering every file as replica
      - Doesn't solve the shared file restore problem

Conclusions

- The Tape RSE provides a needed service
  - Passive Rucio dataset backups
  - Active restores
    - Plan to make process more transparent
- Can be used in many environments
  - Active components are Python scripts
    - Can be easily modified or replaced
      - Accommodate different MSS models

That's All!

Questions?