CMS and Logistical Storage

Daniel Engh
Vanderbilt University

FIU Physics Workshop
Feb 8, 2007

Logistical Storage

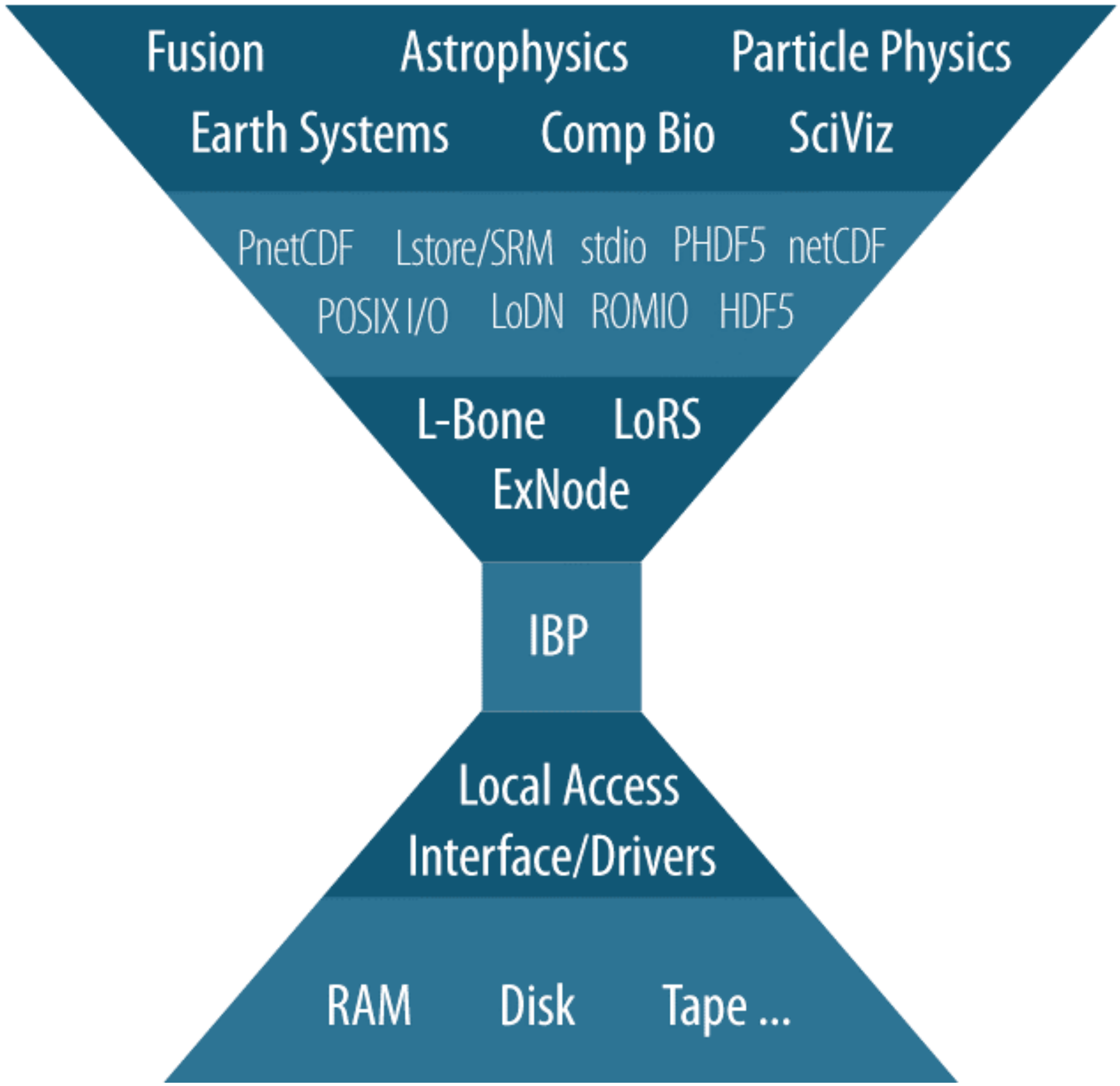
- Distributed, scaleable, secure access to data
- "Logistical"
  - In the spirit of real-world supply-line management
  - Simple, robust, commodity, scaleable, …
- L-Store "Logistical Storage" has 2 parts
  - Logistical Networking -- UT Knoxville
    - IBP (Internet Backplane Protocol)
    - LoRS file tools -- basic metadata management
  - Distributed metadata management -- Vanderbilt

What is Logistical Networking?

- A *simple, <u>limited</u>, generic storage network service* intended for *cooperative* use by members of an application community.
  - Fundamental infrastructure element is a *storage server or "depot"* running the Internet Backplane Protocol (IBP).
  - Depots are cheap, easy to install & operate
- Design of IBP is modeled on the Internet Protocol
- Goal: Design scalability:
  - Ease of new participants joining
  - Ability for interoperable community to span administrative domains

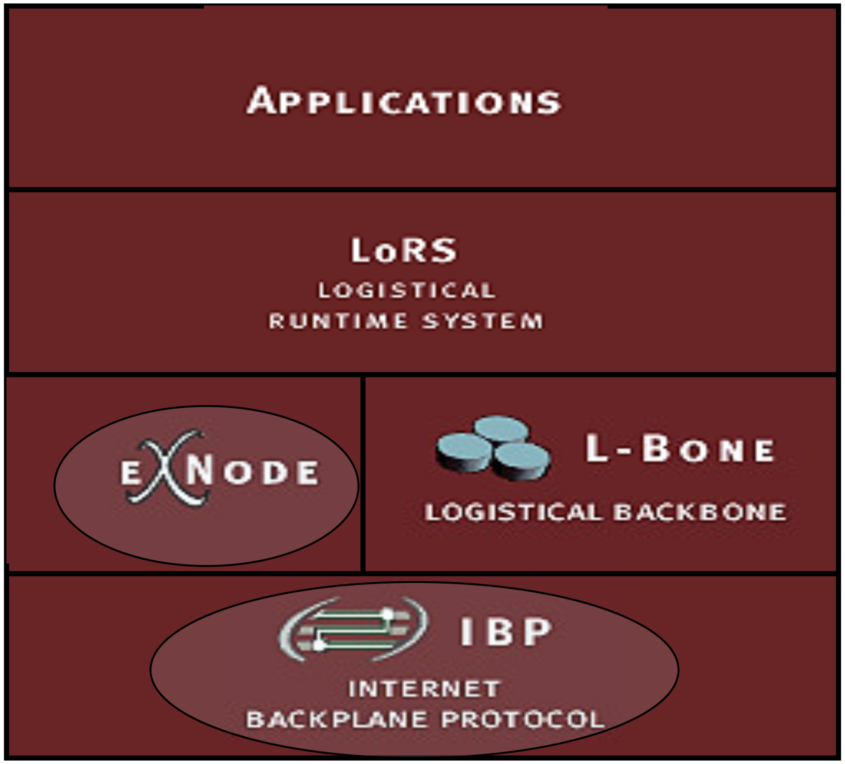
- Micah Beck, UT Knoxville

Logistical Networking Stack

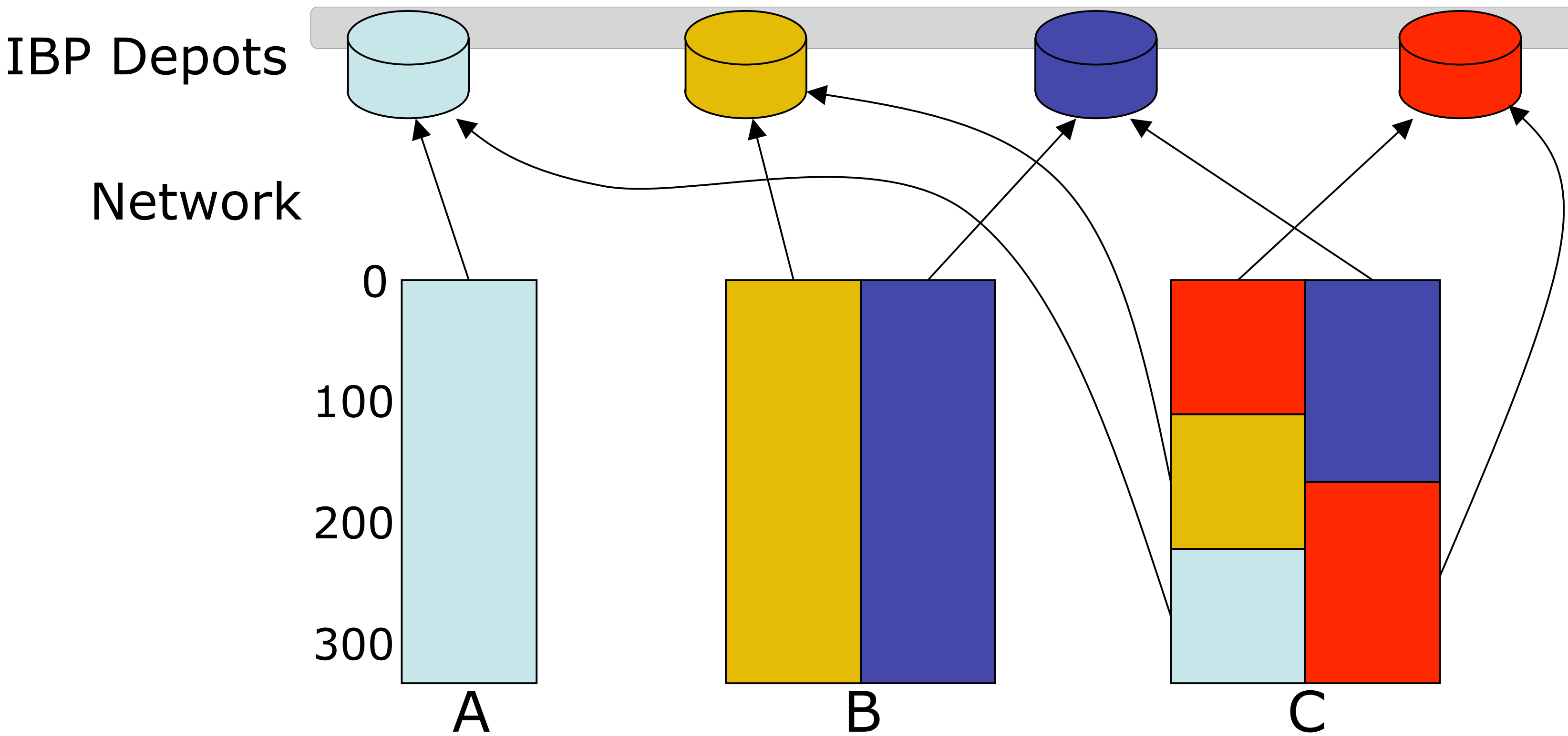


LOCI Software System

- IBP Internet Backplane Protocol
  - Middleware for managing and using remote storage
  - Allows advanced space and time reservation
  - Supports multiple threads per depot
  - User configurable block size
  - Configurable redundancy
  - Designed to support large-scale, distributed systems.



Sample exNodes



3 files uploaded to IBP depots

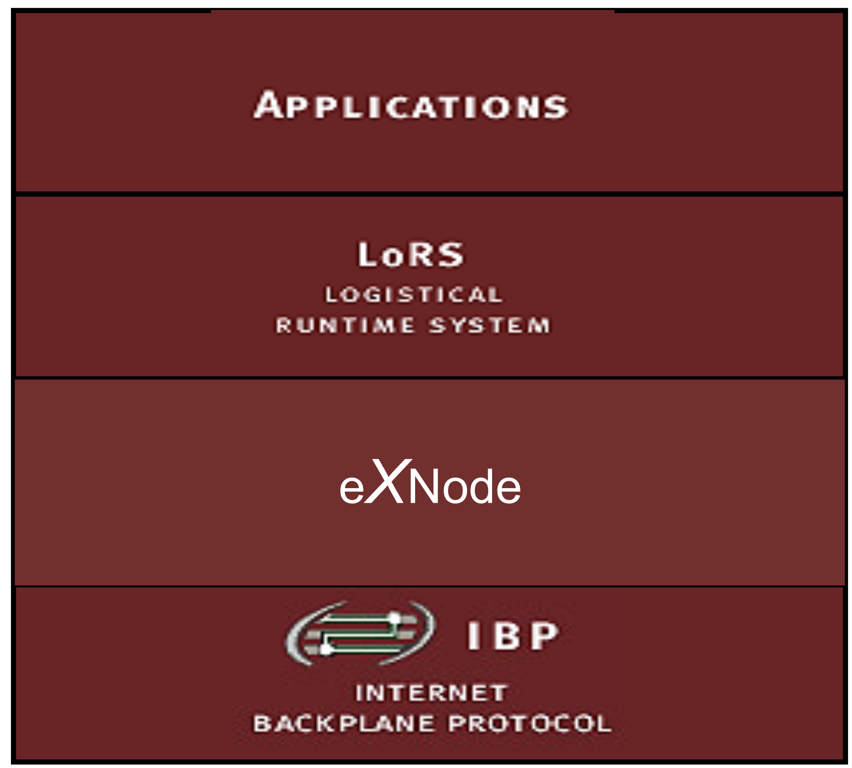
LoRS Filename Conventions

- The exNode file is an xml description of where each block/stripe of data resides.
  - File blocking/striping completely flexible
  - Redundant copies -- automatically try 2,3,…,choices
  - Adaptive striping responds to read/write speeds for each depot.
    - Currently rudimentary, sometimes inefficient.
- Files represented by local exNodes are referenced using the local name of their exNode file
  - **`lors://Filename.txt.xnd`**
  - .xnd extension can be implicit

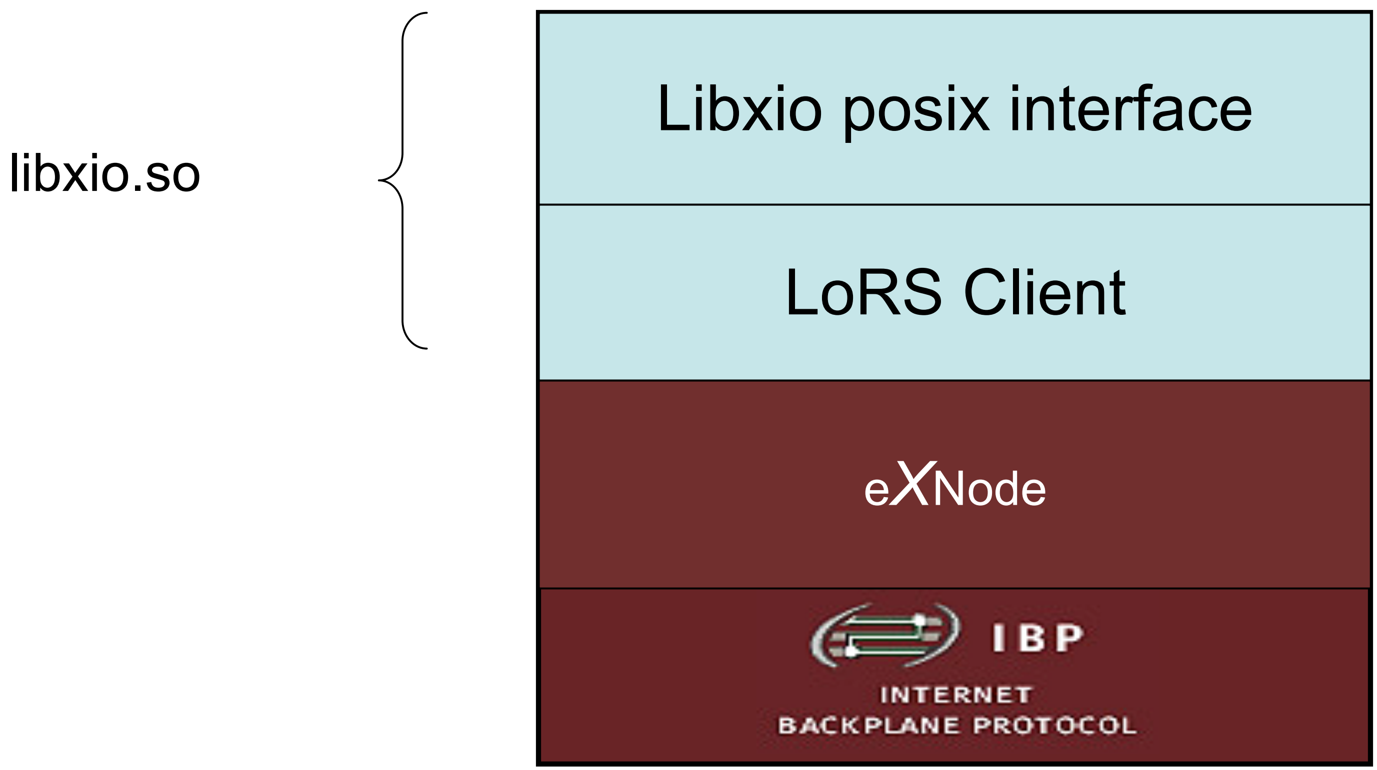
Using IBP with ROOT

- Files can be uploaded to IBP with LoRS or L-Store command-line tools
  - lors_upload …
  - lors_download …
- LoRS can retrieve data from exNode information
- Other LoRS tools not used in our case.

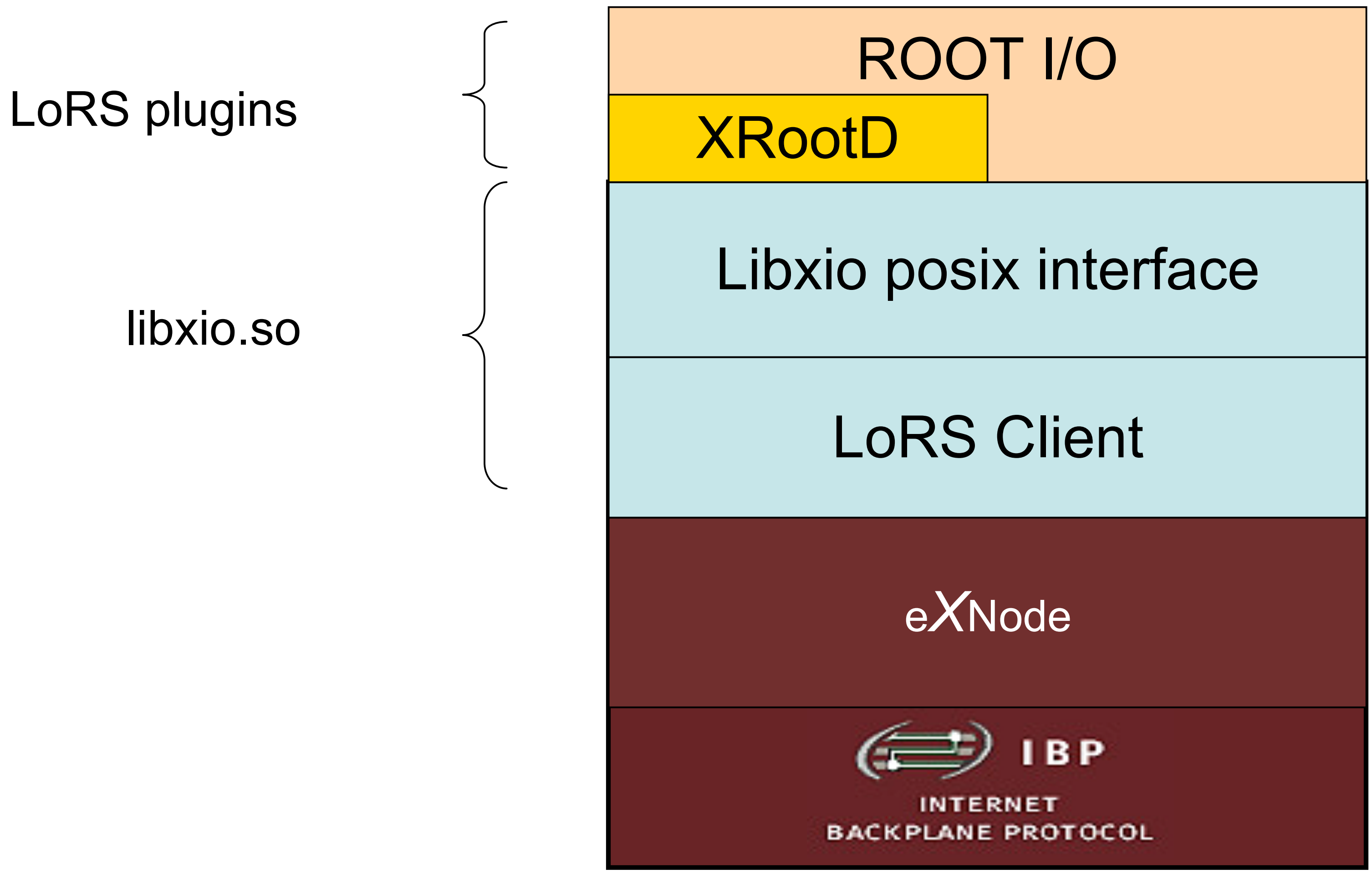
simplified software stack:



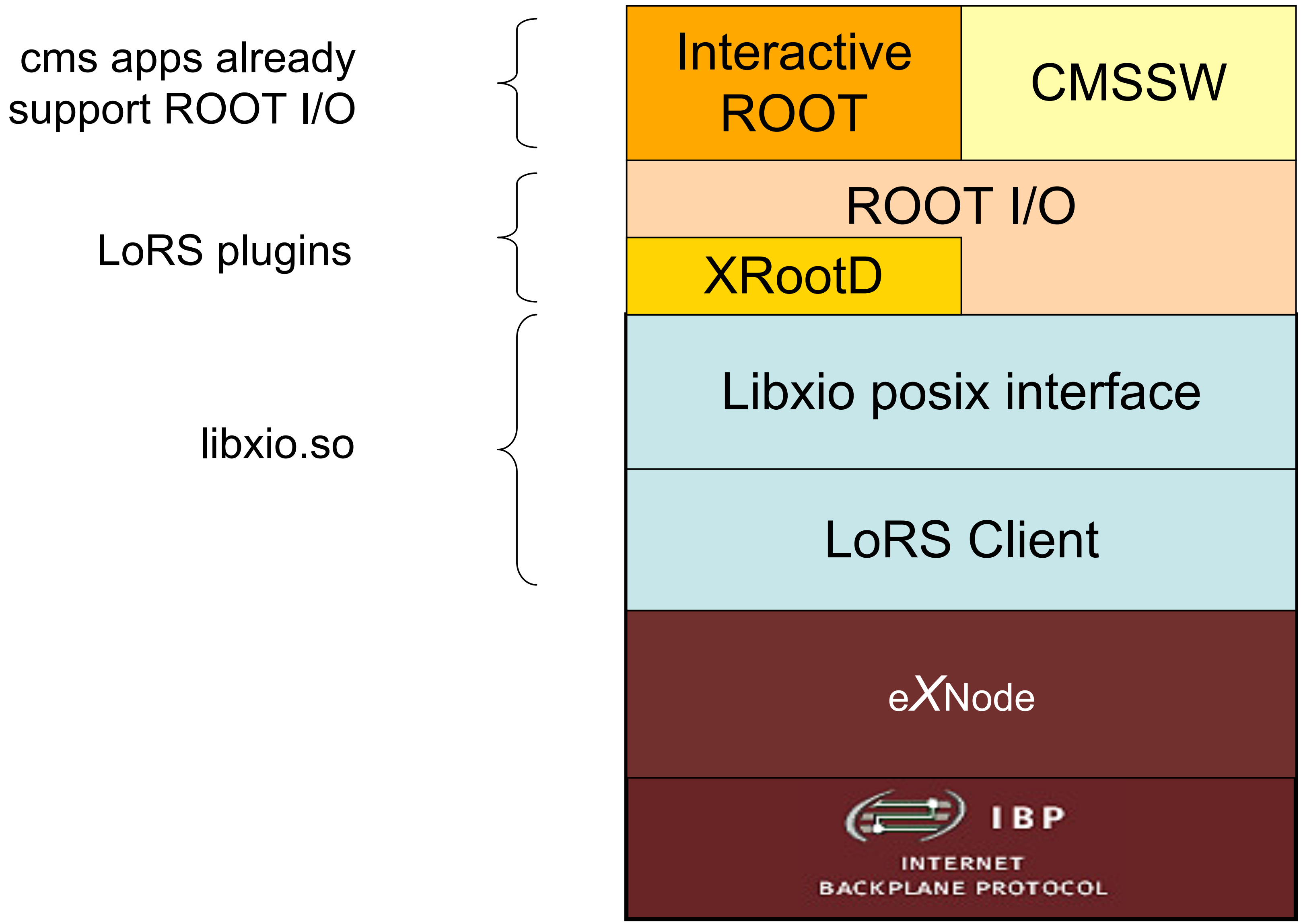
libxio.so LoRS interface



libxio.so plugin for ROOT I/O



libxio.so plugin for ROOT



CMSSW, XRootD, IBP

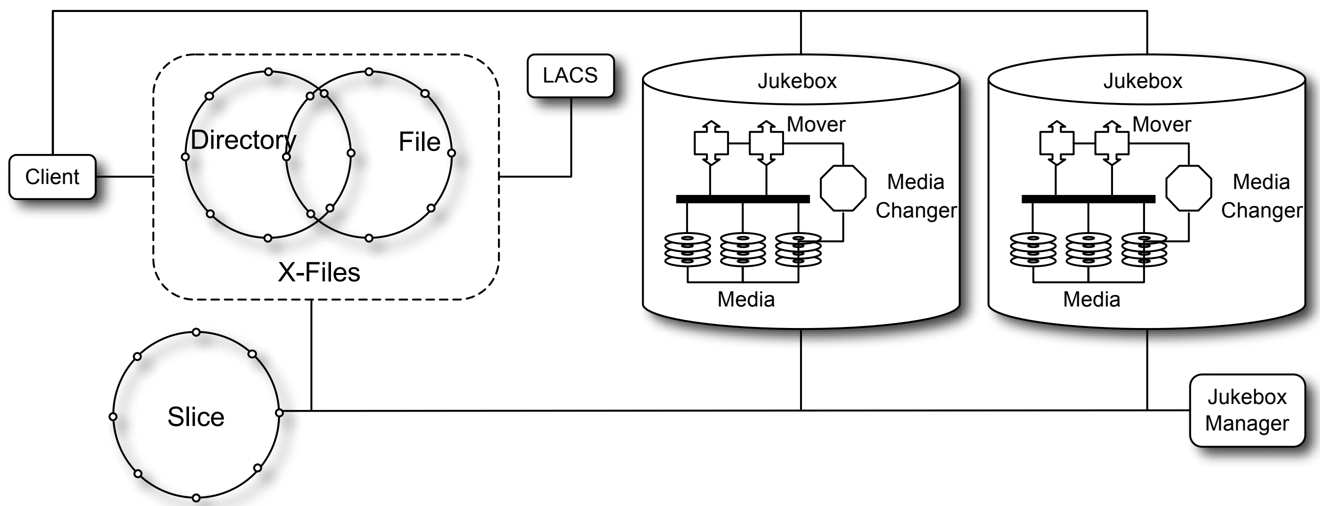
1. Upload your data file to IBP using LoRS
   - lors-upload [myfile.root] [lors flags] [depotlist]
   - exNode file created locally
2. Register your file with XRootD-IBP server
   - XRootD with LoRS/libxio plugin enabled
   - Currently copy exNode file to XRootD exnode directory
3. Stream your file from IBP directly into your ROOT I/O-application memory:
   - Interactive root session, or
   - CMSSW run
   - change file URI to “root://my_XRootD_host/filename.xnd”
4. Write out files locally
   - Stage them where ever you want afterwards

What is L-Store?

- Provides a file system interface to globally distributed storage devices ("depots").
- Parallelism for high performance and reliability.
- Uses IBP (from UTenn) for data transfer & storage service.
  - Write: break file into blocks, upload blocks simultaneously to multiple depots (reverse for reads)
  - Generic, high performance, wide area capable, storage virtualization service
- L-Store utilizes a chord based DHT implementation to provide metadata scalability and reliability
  - Multiple metadata servers increase performance and fault tolerance
  - Real time addition/deletion of metadata server nodes allowed
- L-Store supports Weaver Erasure Encoding of stored files (similar to RAID) for reliability and fault tolerance.
  - Can recover files even if multiple depots fail.

Architecture

Goal: provide a distributed name space and storage



•A directory (**Directory Ring**) is comprised of files (**File Ring**)

•Each file is comprised of several slices(**slice ring**)

•Each slice is stored on **Media** in a **Jukebox** and accessed via a **Mover**

•All directories and files have permissions (**LACS**) and extended attributes

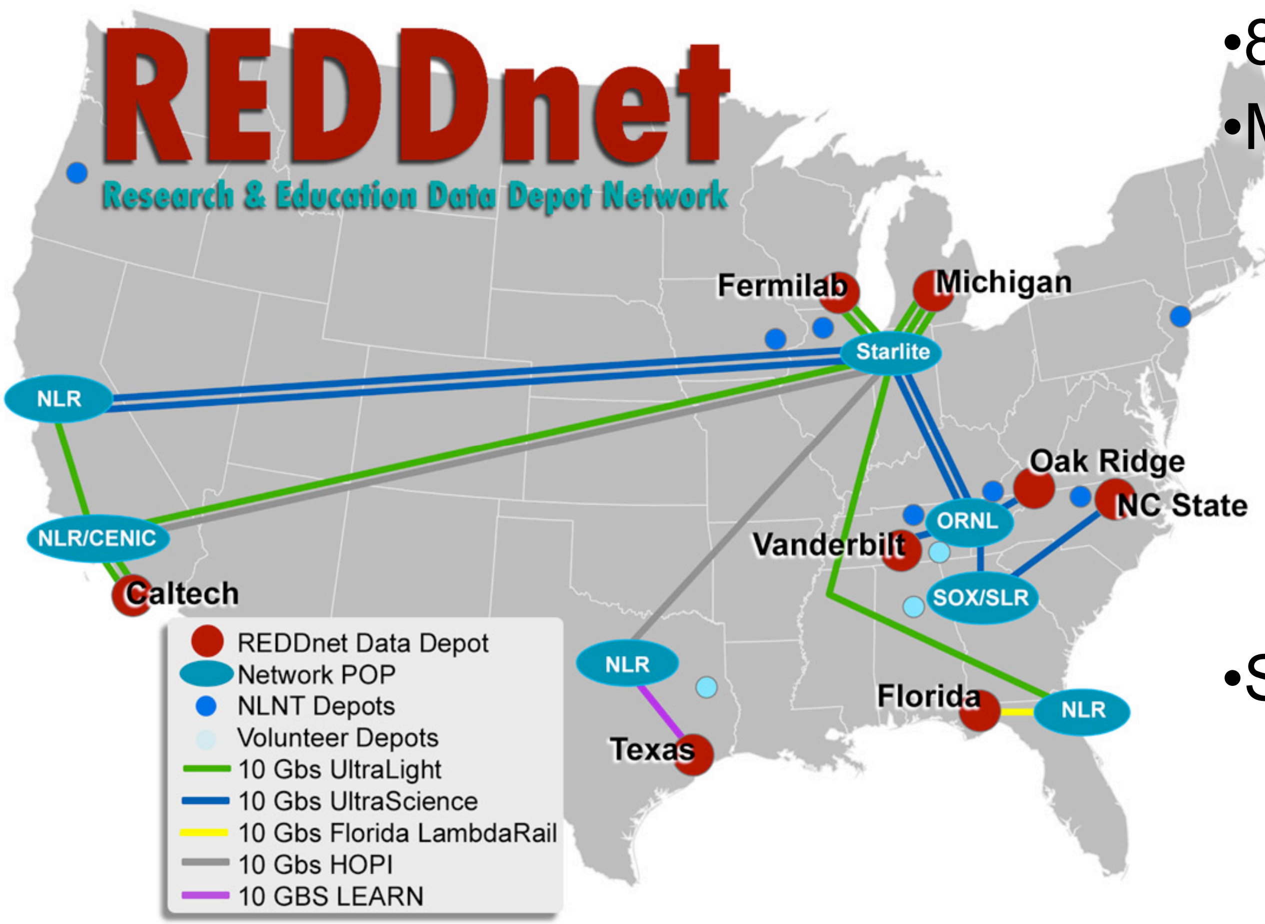
- Depot list, slice size, erasure coding, …

•Each file can have multiple representations.

- Each representation is associated with a service
- A **service** is added by providing pluggable modules for the
  - Client, X-files, and JukeBox(transfer method)

REDDnet

Research and Education Data Depot Network



- NSF funded project
- 8 initial sites
- Multiple disciplines
  - Sat imagery (AmericaView)
  - HEP
  - Terascale Supernova Initative
  - Structuraly Biology
  - Bioinformatics
- Storage
  - 500TB disk
  - 200TB tape

Future steps

- ROOT LoRS plugin -- bypass XRootD
- Host popular datasets on REDDnet
- Integrate L-Store functionality

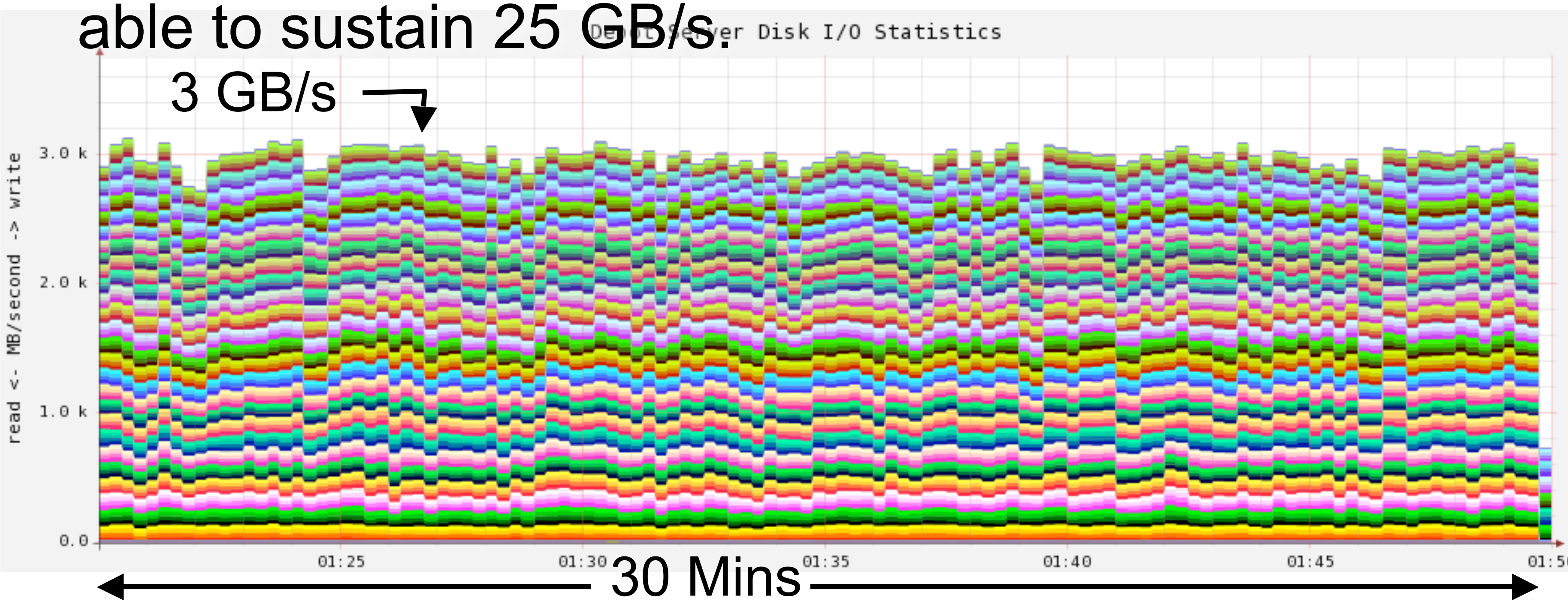
Extra Slides

What is L-Store?

- L-Store provides a distributed and scalable namespace for storing arbitrary sized data objects.
- Agnostic to the data transfer and storage mechanism. Currently only IBP is supported.
- Conceptually it is similar to a Database.
- Base functionality provides a file system interface to the data.
- Scalable in both Metadata and storage.
- Highly fault-tolerant. No single point of failure including a depot.
- Dynamic load balancing of both data and metadata

L-Store Performance

- Multiple simultaneous writes to 24 depots.
- Each depot is a 3 TB disk server in a 1U case.
- 30 clients on separate systems uploading files.
- Rate has scaled linearly as depots added.
- Planned REDDnet deployment of 167 Depots will be able to sustain 25 GB/s.

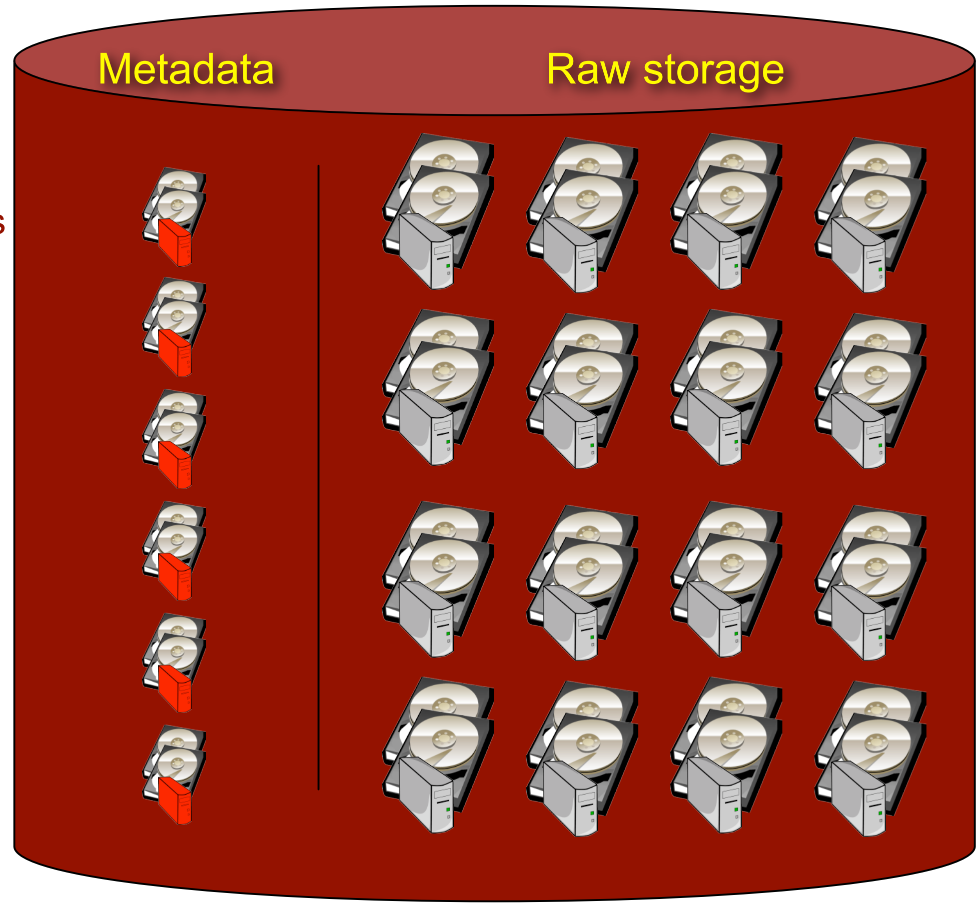


“Data explosion”

- •Focus on increasing bandwidth and raw storage
- •Assume metadata growth is minimal
  - – Works great for large files

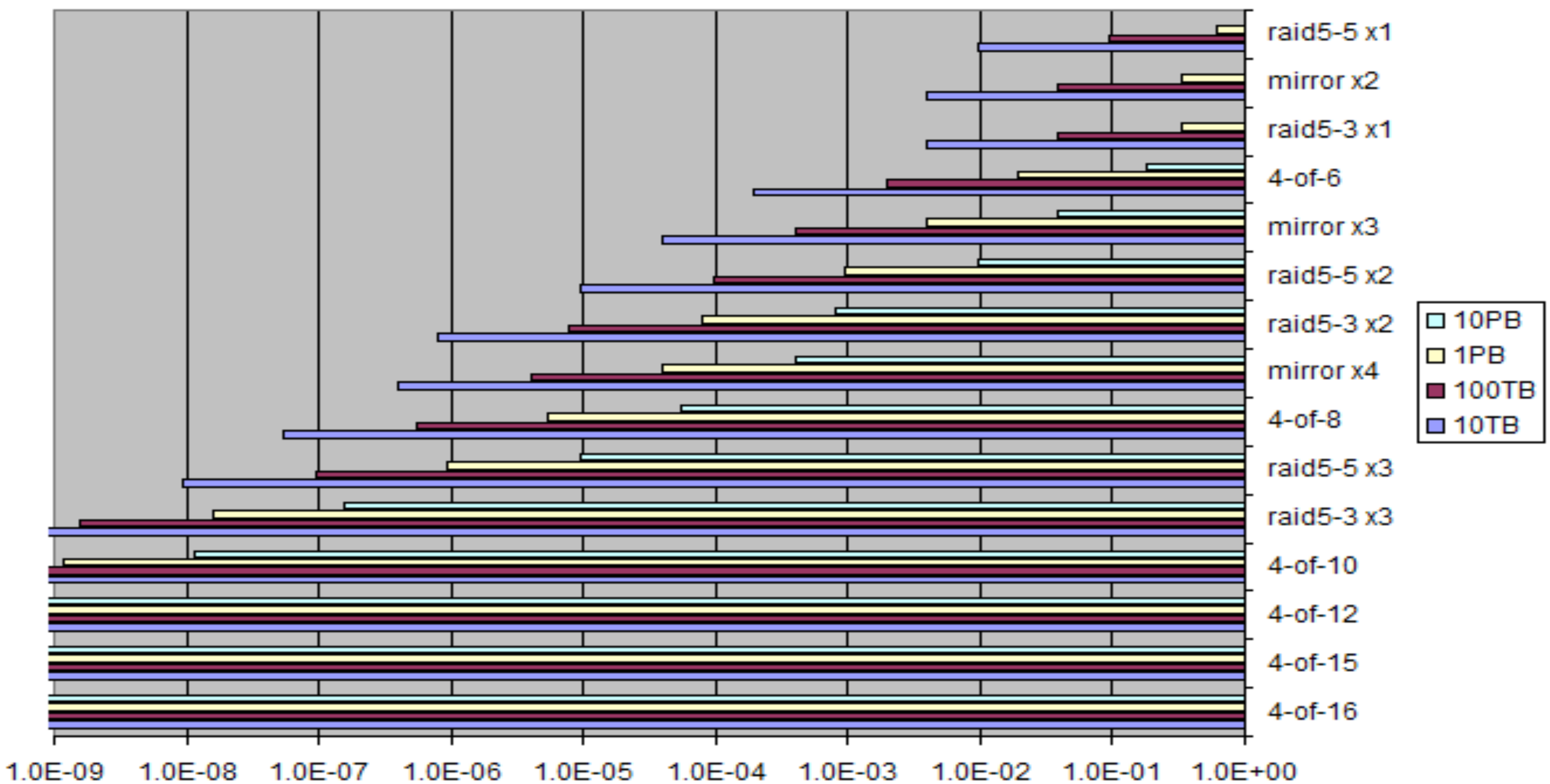
- •For collections of small files the **metadata** becomes the bottleneck
- •Need ability to scale metadata

- •ACCRE examples
  - – Proteomics: 89,000,000 files totaling 300G
  - – Genetics: 12,000,000 files totaling 50G in a single directory



Probability of data loss with 1% disk failure

(http://elib.cs.berkeley.edu/storage/psi/docs/Petabyte-20040710.ppt)



QoS Requirements

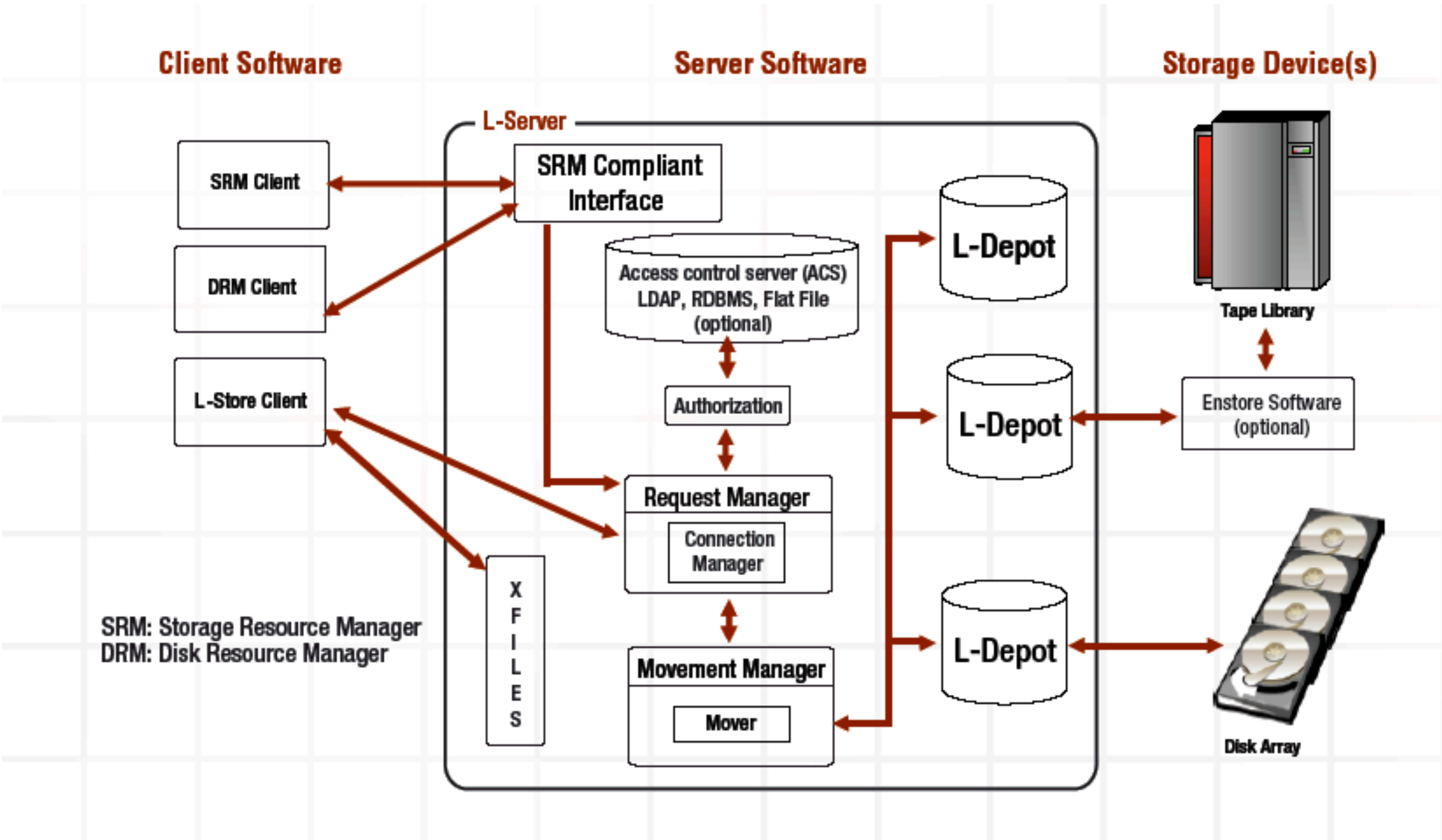
- **Availability** - Should survive a partial network outage
- **Data and Metadata Integrity**
  - End-to-end conditioning is a must!
- **Performance**
  - Metadata(transactions/s)
  - Data(MB/s)
- **Security**
- **Fault Tolerace**
  - Metadata - mirroring
  - Data - multiple complete device failures

WEAVER erasure codes

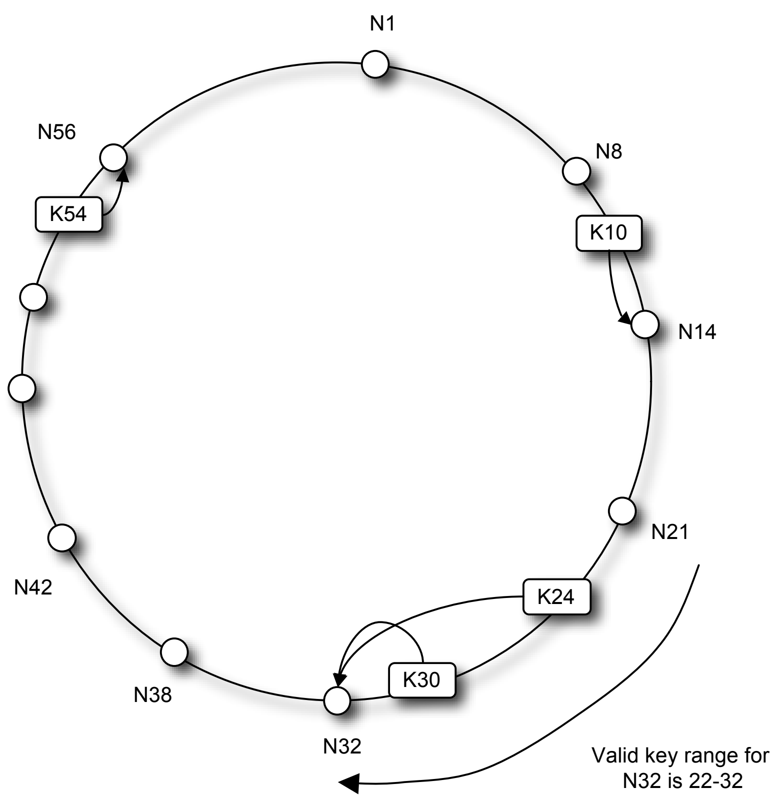
- Tables supporting up to 10 failures
  - Satisfies data fault tolerance requirements
- Vertical erasure code (parity and data stored on same resource)
- Encoding and Reconstruction times are O(t) where t = fault tolerance
  - Encoding and reconstruction done directly on the depot.
- No decoding time since vertical code

J. L. Hafner, "WEAVER Codes: Highly Fault Tolerant Erasure Codes for Storage Systems," *FAST-2005: 4th Usenix Conference on File and Storage Technologies*, December, 2005, http://www.usenix.org/events/fast05.

Simple L-Store architecture based on a single metadata server



Chord Ring
Distributed Hash Table



- Key (K##) -hash(name)
- Nodes (N##) are distributed around the ring and are responsible for the keys “behind” them.