The CDF Run II Data Handling Design

Robert D. Kennedy
FNAL Computing Division
for the CDF Data Handling Group
CHEP03 (San Diego, CA, USA)

March 27, 2003

CHEP03

Background: Evolve system in use; add network and remote access

Mass Storage System (MSS): Enstore

Network-Access Disk Caching: dCache on commodity file servers

Remote Access Framework: SAM as layer on dCache/Enstore

Future of CDF DH Design: Globally distributed data handling

Background

The CDF Data Handling System

⇒ Delivers data files from repository to clients and vice versa
⇒ Manages associated meta-data: simple conceptual view of data
⇒ Manages data access resources: tape drives, disk space, etc.
⇒ All data stored in ROOT data files, in variety of formats

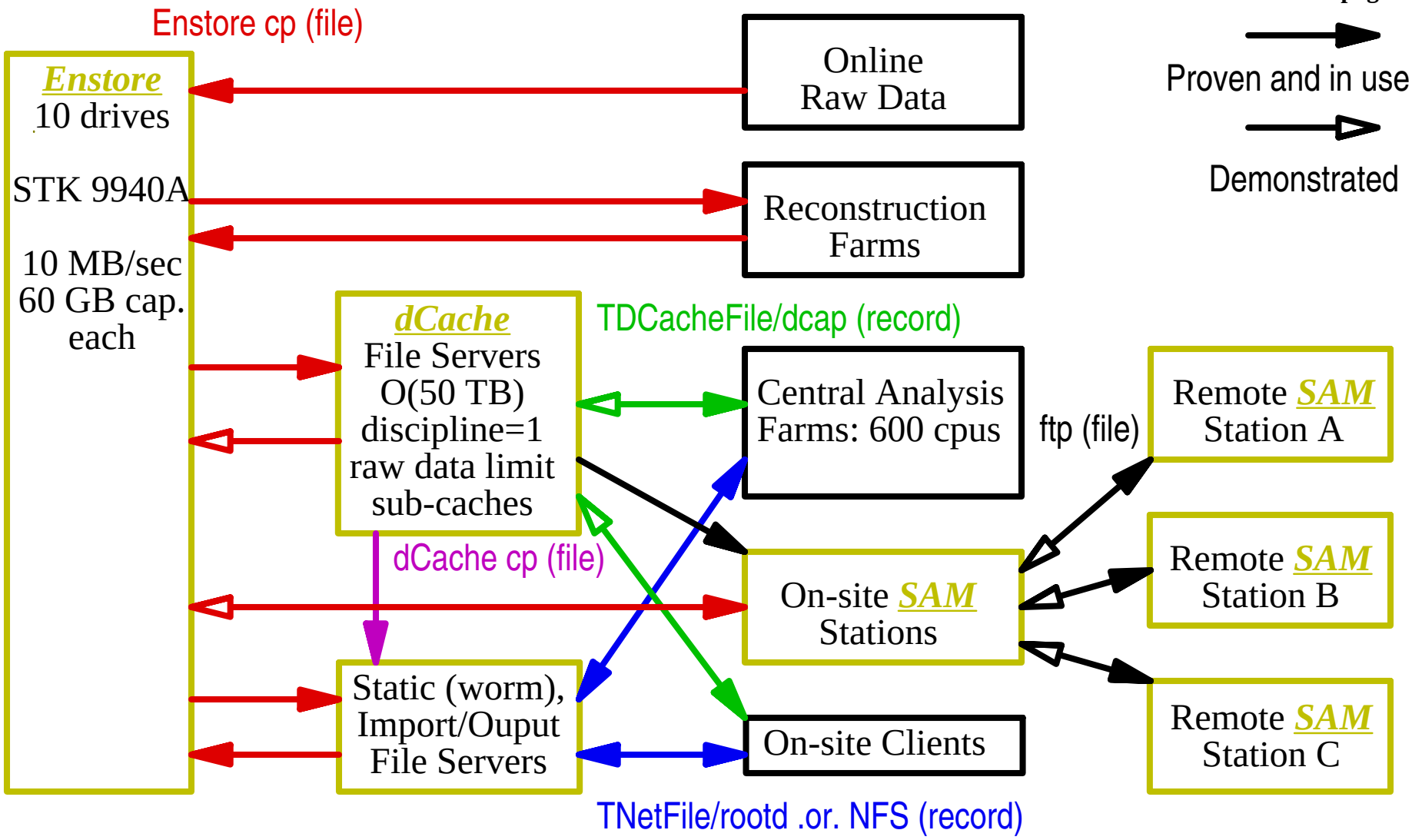
The Goals and Constraints: CDF DH in 2002

⇒ O(200 TB) of data on tape. Must store/manage O(2 PB) soon
⇒ Much meta-data to manage, and more being considered
⇒ Need reliable, "fast" data delivery over network, remote access
⇒ Extend functionality of existing DH system, *while in use.*
⇒ "Shared" components: development, maintenance, other costs
⇒ Work towards a long-term evolutionary path for CDF DH.

The Contributors

CDF DH design based on mating of S/w, H/w systems, thanks to:
CDF Data Handlers and Computing Support, FNAL CCF dept,
DESY dCache and PNFS teams, D0 and FNAL SAM teams,
the ROOT team, and many others.

CDF DH Today (legacy systems not shown)



Underlying *Meta-data*: PNFS (Enstore/dCache), CDF Data File Catalog, SAM meta-data

Mass Storage System: Enstore

⇒ **Enstore = Mass storage management, using PNFS meta-data**

Developed and supported by FNAL CD. Enstore = network access to files on tape in robot, with request optimization layer. PNFS (from DESY) looks like a Unix file system, where the "files" are meta-data units. PNFS = meta-data distributed by NFS.
CDF Enstore: *robust and performant.* Accessible to users, esp. web-based monitoring.

⇒ **(10) STK T9940A tape drives in dual STK Powderhorn 9310s**
⇒ **Upgrading to (10) STK T9940B tape drives (same tape media)**

9940A - I/O rate: 10 MB/sec read/write, Data capacity: 60 GB/tape, up to ~10k tapes
9940B - I/O rate: 30 MB/sec read/write, Data capacity: 200 GB/tape, up to ~ 2PB

⇒ **Data outlives its media. Expect it, and plan for it.**

⇒ **Tape media cost is ultimate limitation on data-taking capacity.**

⇒ **Discipline: simultaneous transfers to any one client host**

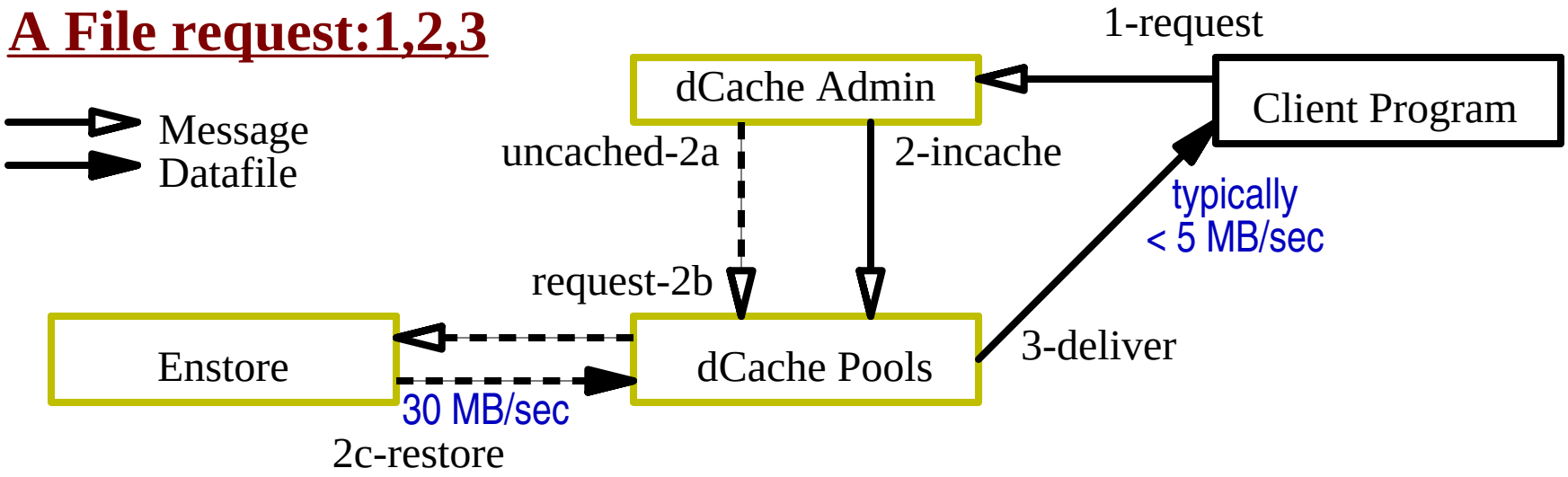
CFR: Don Petravick's talk on FNAL Data Storage Infra, Dmitry Litvintsev's talk on CDF DH Sys.

Network Disk Caching: dCache

⇒ dCache = Network-accessible disk cache as front-end to MSS

Originally developed at DESY (Patrick Fuhrmann et al), now co-maintained by DESY and Fermilab CCF dept. Primary goal: rate-adapting front-end to an MSS. Oriented towards on-site client access. Expects reliable network, so no integrity check.

A File request:1,2,3



⇒ Why use dCache instead of CDF "DIM" disk cache in use?

DIM: locally mounted disks, no native network access, CDF specific product.

dCache "out-of-the-box" is more scalable, network-accessible, and feature-laden.

Adaptation was well-encapsulated - easy in software, no significant re-training of users.

CFR: Patrick Fuhrmann's talk on dcache

CDF Adaptation to dCache

⇒ **One line of TCL selects dCache or old cache in user jobs**

Abstract interface in CDF software implemented for several modes of dCache access: with PNFS mounted locally, without PNFS mounted, with a separate broker (SRM).

⇒ **TDCacheFile: ROOT TFile class for dCache client protocol**

Since our data in ROOT TFiles, we used the sub-class TDCacheFile as a URL-driven plug-in to access our ROOT datafiles. This enables any ROOT datafile format to be supported in Enstore and accessible via ROOT... a feature to be explored.

⇒ **CDF dCache: 3 admin hosts, 1 monitor host, 20-50+ pool hosts**

Admin hosts are simple dual CPU Linux PCs. Each can support different transfer and/or authentication protocols. Pool hosts are 2 TB RAID50 IDE Linux file servers, each supporting 3 "read" pools. Separate CDF test stand is maintained as well.

⇒ **Some days >15 TB read, >90% cache hit rates common**

Record-oriented transfers as well as file-oriented transfers. Use-cases growing.

⇒ **CDF dCache (read) into production state mid-April 2003**

Record-oriented transfers as well as file-oriented transfers. Expect greater rates soon.

Exploiting Affinity Feature

⇒ File Family Affinity: send data to pools based on "file family"

CDF maps "dataset id" to PNFS "file family". We control data in dCache by dataset.

⇒ Use-Case: Isolate "scrolling" data from steady-state data

Pools have been specified to have an exclusive affinity to raw datasets. All raw data goes to only these pools and only raw data goes to these pools. When pools are full, they recycle their own files to make space. This prevents steady-state datasets from being displaced by the Reconstruction Farms "scrolling" through all raw data.

Creates a independent sub-cache.

⇒ Use-Case: Provide disk space accountability

A research group can donate file servers to cache IFF they can prove to funding agency these will only be used for the supported physics: sub-cache for their dataset.

⇒ Use-Case: Guide requests to pool "closest" to client/resources

Data coming from Reconstruction farms can be directed based on the dataset id to the dCache Write pools closest on network to Reconstruction farms and Enstore.

Optimizes use of network resources.

Exploiting Pool-to-Pool Copies

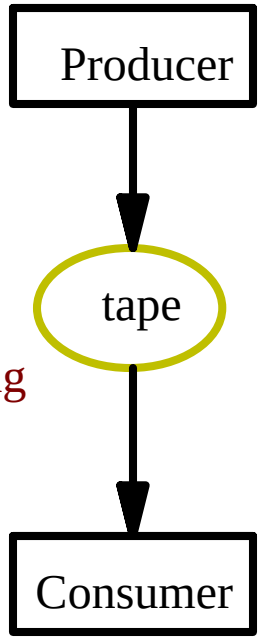
⇒ Data processing can be limited by write/read tape latency

Raw data read from tape by Reconstruction farms only after it is written to tape.
This can introduce hours, and in extreme cases days, of latency in data processing.

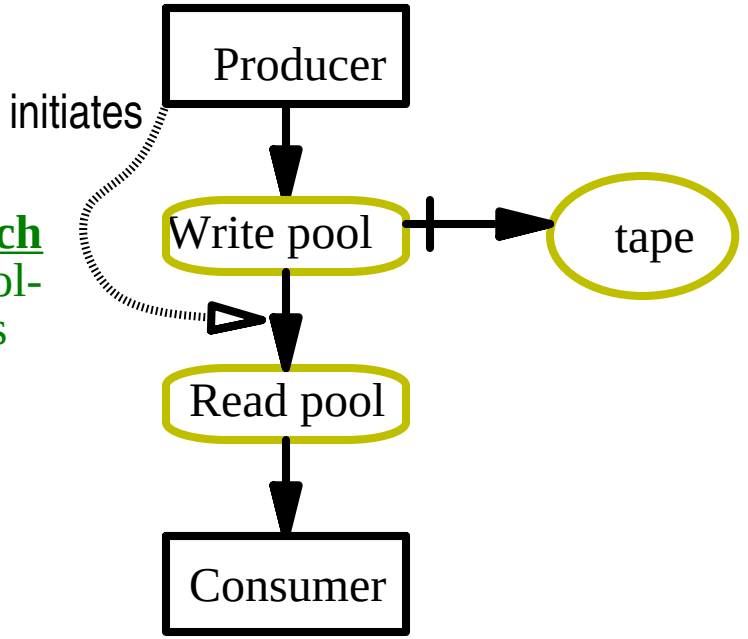
⇒ Solution: Take the tape drives out of data processing path

Producer writes data to dCache write pool, and initiates staging of that data to read pool. Clients can immediately access the data. Data written to tape asynchronously.
Data in write pool must be as "safe" as if it were on tape.

Old Approach
with tape I/O in the critical path of data processing



New Approach
exploiting pool-to-pool copies with dCache



Commodity File Servers

⇒ **Linux-based 2 TB, RAID 50, IDE fileservers - vital component**

⇒ **Volatile disk caches challenge file-system, I/O performance**

Linux kernel block I/O mechanism and ext3 file system perform poorly managing 1 GB files in a 2 TB file system in volatile disk cache. Writing large files leads to inefficient small writes to disk (eof). Recycling files soon leads to disk fragmentation, dropping read performance as well. Volatility may exacerbate other errors as well.

⇒ **Data volume processed per day > IDE disk reliability**

IDE disks quote ~1 bit error per TB. We process many TB in a day, PB in a year.

⇒ **CDF dCache: a) XFS file system, b) direct I/O writes to disk, c) check file CRC before *and after* writing it to disk.**

Both XFS, direct I/O required to maintain performance of file system, I/O. Custom Enstore cp used to write files into dCache, rechecks CRC values after write - reads file back to do so. Direct I/O prevents false positives from memory buffering in I/O.
I/O (in-memory) buffering in normal block I/O can fool naive CRC checks.
A little problem at one DH scale can become a grand challenge at the next scale up.

CFR: Frank Wuerthwein's talk on the CDF Central Analysis Farm

Remote Data Access: SAM

⇒ SAM = Data Handling framework, a "proto-DataGrid"

Originally developed by D0, FNAL CD. In use for some time at D0. "Stations" serve local disk caches to clients, talk to other Stations or a MSS to get files not in cache.

⇒ Reliable remote access, transport protocols, configurable

Datafile CRC is stored in SAM meta-data. After transport into a Station cache, CRC value is recalculated and checked. Transport protocol can be configured for situation: high-bandwidth, low bandwidth, different servers, etc. Stations can be configured to prefer files from particular Stations or an MSS when fetching.

⇒ Extend CDF DH model - adapt to (join) SAM project

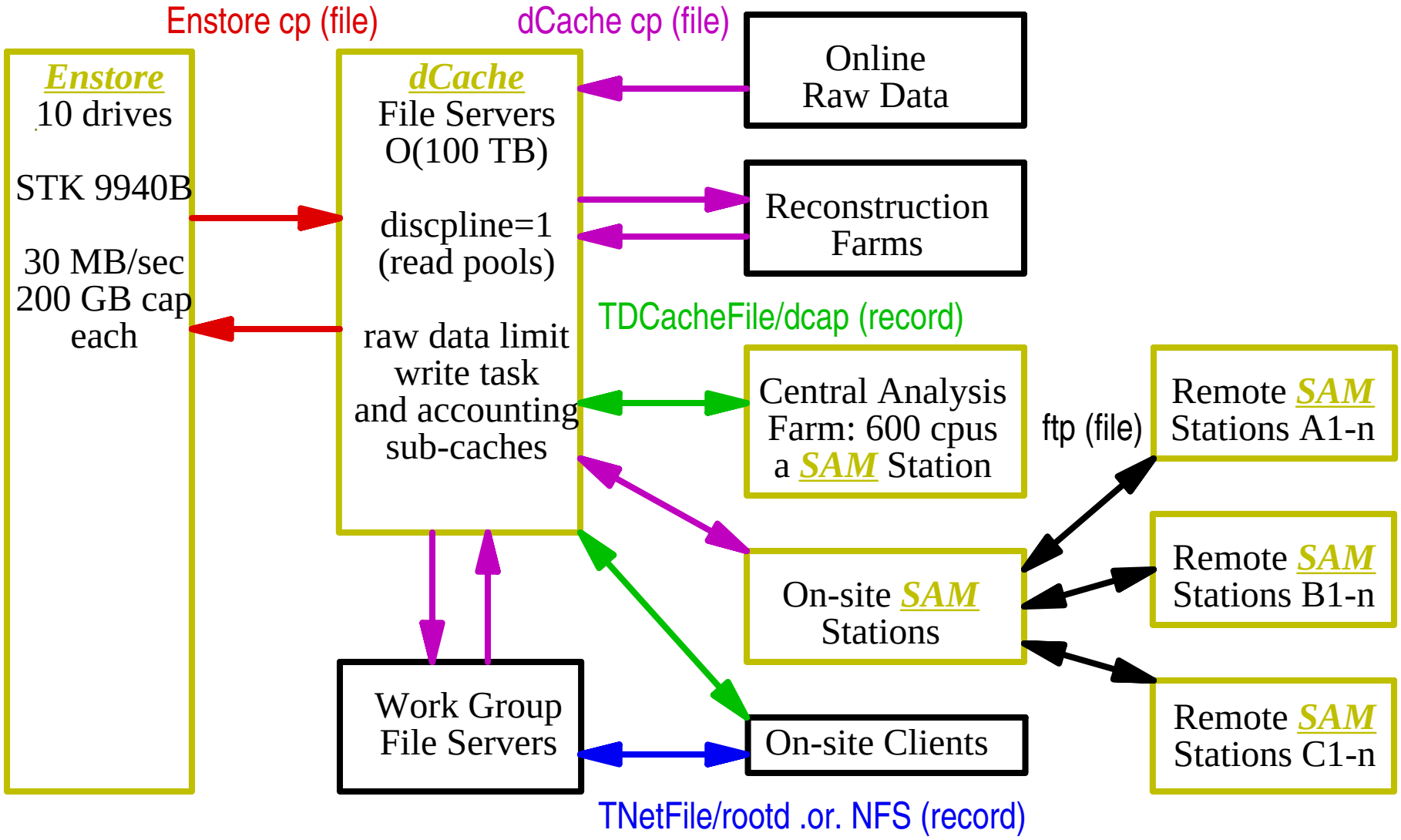
Existing CDF DH emphasized reliable local data access. Need to extend to serve remote institutions (initial motivation), also to integrate more functionality in shared solution.

⇒ Full SAM adaptation very involved, but SAM *is* in use at CDF

Must adapt independently evolved CDF framework to SAM approach.
Common meta-data schema is being defined, an intrusive and risky process.

CFR: Lee Lueking's talk on SAM-D0 Experience, Gabriele Garzoglio's talk on SAM-CDF Adapt.

CDF DH Baseline Goal



Underlying *Meta-data*: PNFS (Enstore/dCache) and SAM meta-data

Summary, Future of CDF DH

⇒ Enstore is CDF MSS. Robust, performant and in production

⇒ dCache is CDF network disk cache, about to be in production
⇒ dCache is successfully run on commodity Linux file servers

⇒ SAM for CDF remote access; CDF evolving to full use of SAM

⇒ Institutions have local data/computing resources, *want* to share

Institutions have significant resources, and want to become CDF resource providers, not just consumers. Remote SAM stations still dependent on FNAL for MSS, meta-data.

⇒ Globally distributed DH - SAM is the *first* step for CDF

CDF is involved in GRID and GRID-related efforts, and will be more so over time.
We have one MSS-backed "node" in future CDF datagrid at CDF, can expand on this.
Global distributed DH: multiple MSS and databases, not just central ones at FNAL.
Several paths: marriage of SAM and CDF distributed "Central" Analysis Farms.

CFR: Stephan Stonjek's talk on SAMGrid, Fedor Ratnikov's talk on SAM/GRID Monitoring, Frank Wuerthwein's talk on CDF Central Analysis Farms for "decentralized" CAFs,...