Data Management Issues

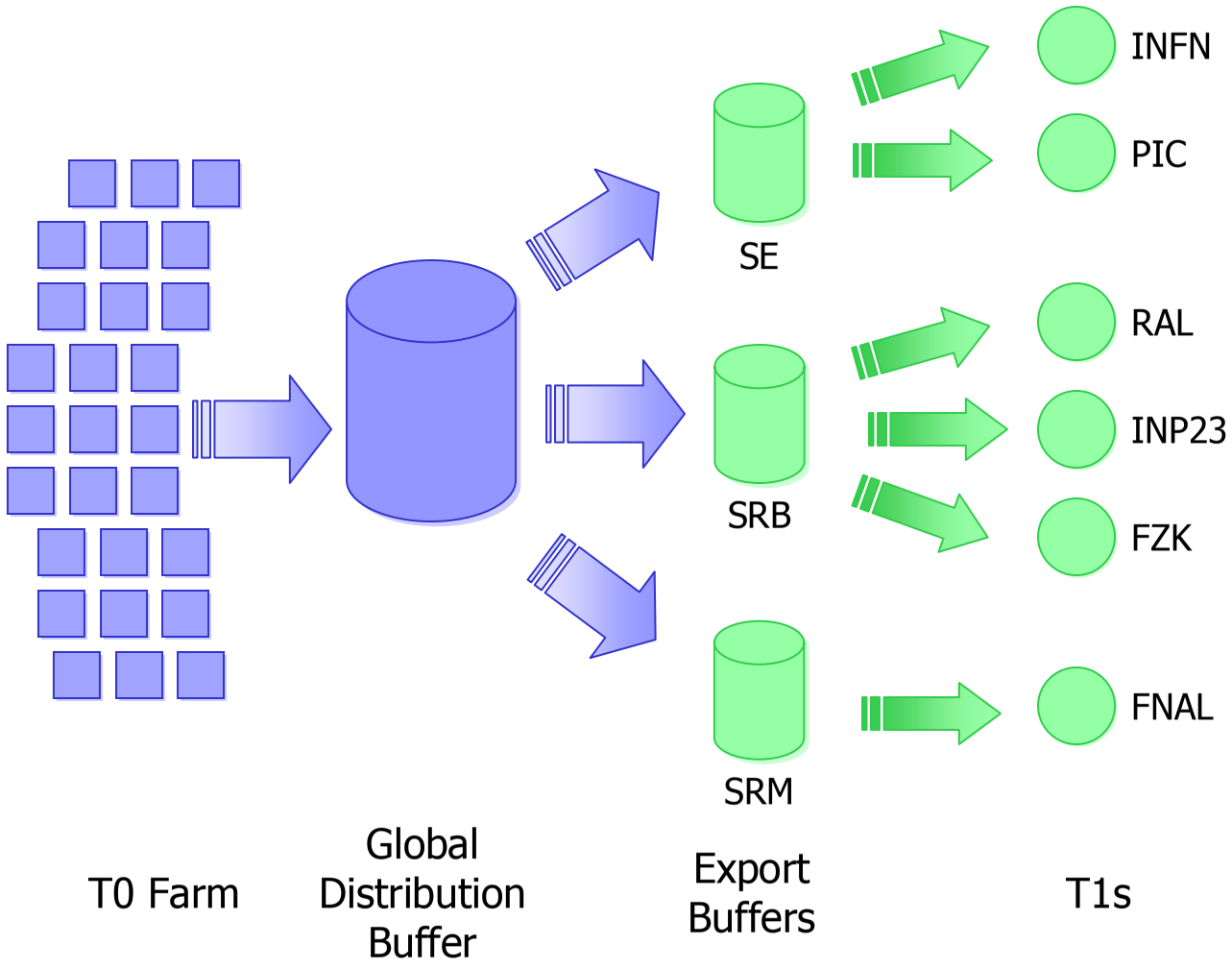
Lassi A. Tuura
Northeastern University

Part I
Data Management at DC04

DC04 Transfer System

- Data challenge started in March 2004
- No suitable data movement system existed
  - We quickly developed Transfer Management DB (TMDB)
    - In total a few weeks from design to operation
    - Emphasis on T1 choices of data transfer tools, not imposing central choice
  - State in Oracle, all file transfers made by simple software agents
  - "Free for all": agents written in C, perl, bash
- Agents
  - Tier-0 agents received "drops" from the reconstruction farm and published the information into RLS and TMDB
  - Configuration agent assigned files to destinations
    - Capability to reassign when T1 is down was not exercised
  - Transfer agents moved files to desired destinations
  - Cleaning agents evicted safely transferred files from disks
  - Also tested automatic file merging agent

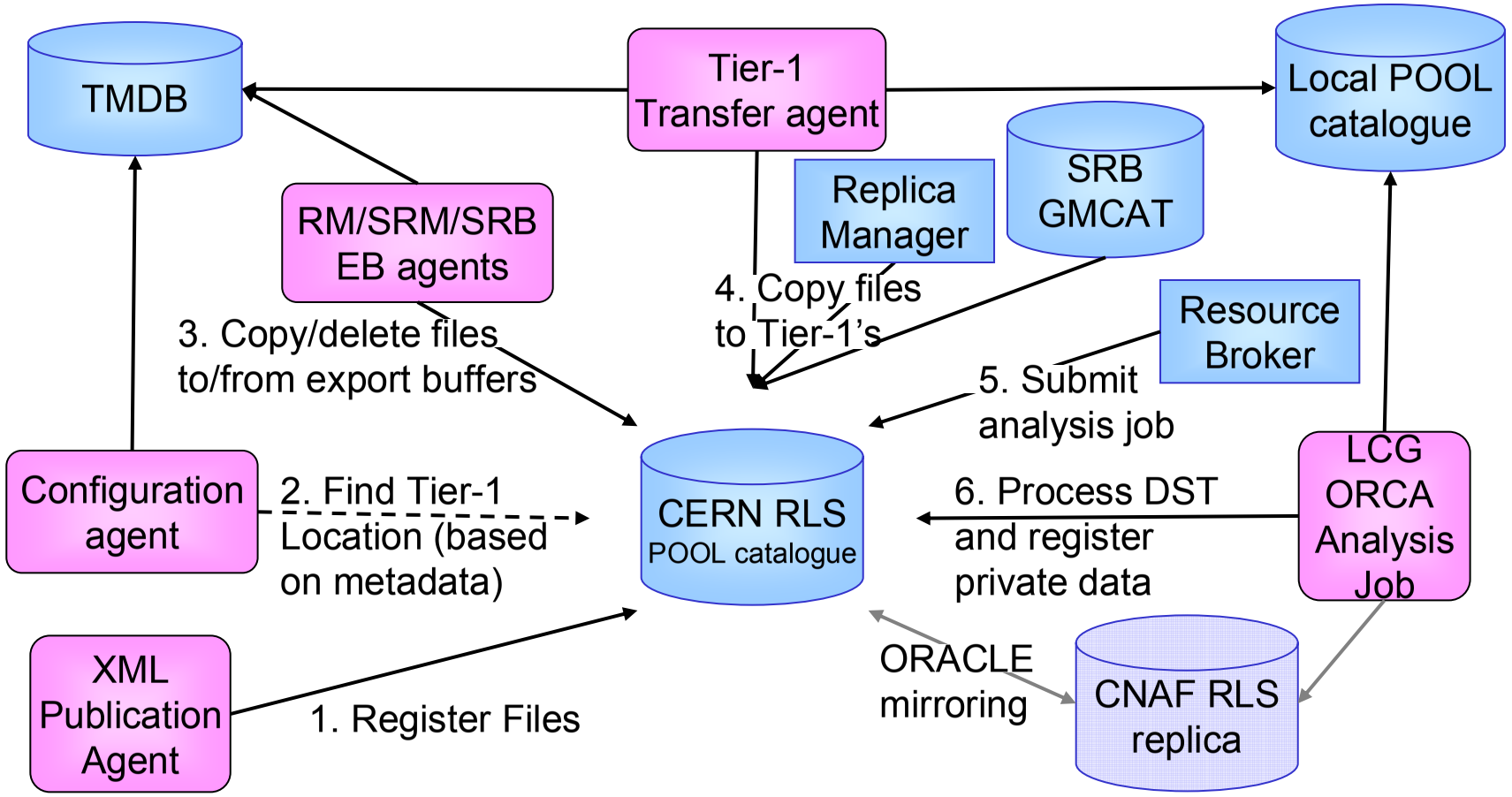
Transfer Overview



File Catalogues

- We used RLS as global, omniscient POOL file catalogue
  - Initial entries were made by Tier-0 "publishing" agent
  - Each transfer & cleaning agent maintained replica information
  - File information by GUID
    - LFN; PFNs for every replica; Meta data attributes
  - Meta data schema handled and pushed into catalogues by POOL
    - Some attributes are highly CMS-specific
  - CMS does not use a separate file catalogue for meta data
- Some sites also maintained local catalogues
  - POOL MySQL catalogue with all the files, but only local PFNs
    - On file copy, copy also the catalogue entries to the local catalogue
  - Dump the catalogue needed by an analysis job into a XML file
    - Job looked up (some?) files in the XML catalogue

Description of RLS usage in DC04



Specific client tools: POOL CLI, Replica Manager CLI, C++ LRC API based programs, LRC java API tools (SRB/GMCAT), Resource Broker

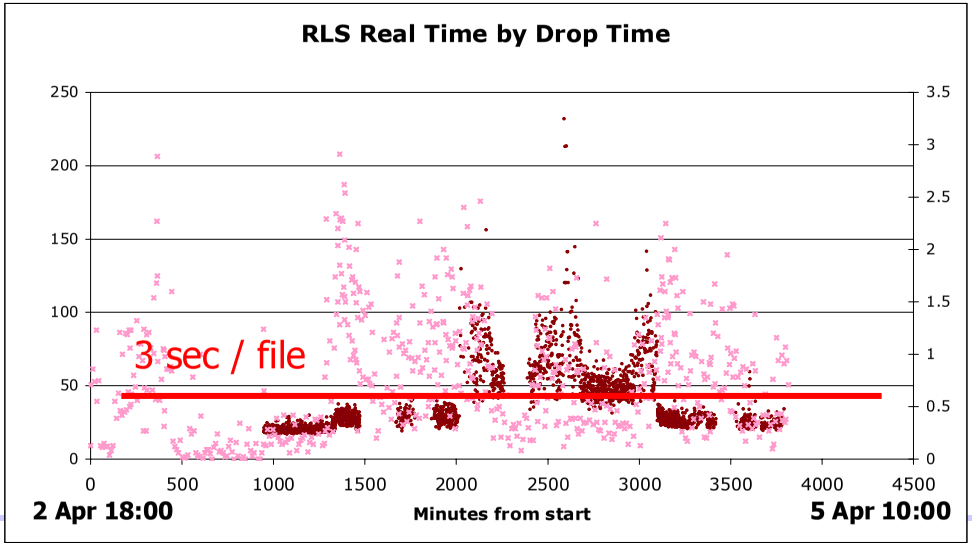
Interactions with RLS in DC04

RLS use as the global POOL catalogue:

- Publishing Agent: Register files produced at Tier-0 into RLS
  - Transfer POOL XML catalogue fragment into RLS; POOL CLI (`FCpublish`)
- Configuration Agent: Query the RLS metadata to assign files to Tier-1
  - POOL CLI (`FClistMeta`) … all data sent everywhere, not truly used
- Export Buffer Agents: Insert/delete the PFNs for the files in the buffer
  - POOL CLI (`FCaddReplica`), C++ LRC API-based programs, LRC java API by GMCAT
- Tier-1 Agents: Insert PFN for the destination location upon transfer
  - In some cases copy the RLS into local MySQL POOL catalogue
  - POOL CLI (`FCaddReplica,FCpublish`), C++ LRC-API based programs, LRC java API by GMCAT
- Analysis jobs on LCG
  - Use the RLS through the Resource Broker to submit jobs close to the data
  - Register the private output data into RLS
- Humans trying to figure what was going on (FCpublish commands)

RLS issues

- Total number of files registered in the RLS during DC04
  - About 570K LFNs each with ~5-10 PFN's and 9 metadata attributes
- General summary: we survived after several workarounds
  - PFN insertion: adequate with new tools developed in-course
  - Inserting files with their meta data attributes was slow
  - Queries woefully slow; resorted to direct Oracle access at points
    - Other POOL catalogues are radically faster
  - RLS multi-master with updates at one end tested, new tests coming



Sometimes the load on RLS increases and requires intervention on the server (i.e. log partition full, switch of server node, un-optimized queries)

⇒ able to keep up in optimal condition, so and so otherwise

Current Status

- Improvements and continuing requirements
  - Replace Java CLI with C++ API; POOL updates; index meta data
  - Bulk transfers now in RLS, promising reports from POOL, may speed up queries considerably (otherwise queries still an issue)
  - Transactions: not there, bulk transfer poor man's replacement?
  - Overhead compared to direct RDBMS catalogues: unmet
- RLS replication tests current carried out by IT-DB
  - ORACLE streams-based replication mechanism
  - If RDMBS handles it, then catalogue design needs revision?
- Not obvious RLS is the model we want to have
  - Global? Omniscient? Meta data vs. PFNs catalogue?
  - Query model may not be what we want
  - Will begin testing gLite catalogue soon

Part II
Evolution

Agents Experience

- Positive experience doing data transfers this way
  - A number of rather simple agents, reliable
  - Easy to develop and rewrite
  - Respect and support T1 choices
- Performance was "sufficient"
  - Time to starting analysis was average 20 minutes (at PIC)
  - Seem to be able to saturate networks trivially
  - Very rarely saw agents take any significant resources
    - File merging agent needs serious hardware
    - Almost everything we tried ground to halt
    - Finally 2-CPU Sun with SAN/RAID disks and Gigabit Ethernet was able to keep up

Simulated Data Movement

- Have developed statistical dummy transfer system
  - Replay authentic "drops" from DC04 through the chain again
  - Timing statistics from TMDB history and other agent logs
  - Fake out desired parts, or run only partial system
  - In theory, you can replay entire data challenges
    - See what happens when you unplug component X
    - WAN/LAN tests depending on what you want to test
    - What-if analysis with different timing profiles
- Simulation plans
  - Tests starting with RLS
  - Transfers and file merging will follow
  - Expecting to maintain near-full-scale test system in parallel to the real running data movement system

TMDB V2 Plans

- Short-term ongoing needs while EGEE delivers tools
  - CMS is in continuous production mode
    - From May/June: 1 TB/month "forever"
    - Multiple data sources and destinations
  - Parts possibly also in trigger farm environment
- Conceptual design
  - Data movement service
    - Transfer files from source A to destination(s) B
    - Provide transfer state information + safe completion (e.g. file has been copied to mass storage on all intended destinations)
    - Answer: "What is the cost and latency of moving these files there?"
    - Transient: does not remember files after move is complete!
  - Supports on-demand moves and continuous push
  - Someone else worries about what should go where
    - Now: configuration agent decides based on owner/dataset/stream and T1 data "subscriptions"

TMDB V2 Plans (II)

- Design overview
  - Files are routed like similar to internet routing protocols
    - Initially static routing only, dynamic may follow in future
  - Agents maintain routing information in the database
    - If agents or links go down, automatically adapts
  - Currently continuing to use one central database
    - Not a fundamental design feature
  - Can maintain catalogues at each site, or a global one
  - Common Perl transfer agent toolkit
- Time scale: operational in a few weeks
- Lots of room for co-operation
  - Not attached to our system—just need something that *works!*
  - Will gladly provide use cases, requirements, sit down with developers, develop, test

Other Related Components

- RefDB
  - Started out as production database
  - Evolving into three different views
    - Virtual data catalogue
    - Production bookkeeping database
    - (Logical/data set) file catalogue
  - Discussions on extension beyond production
    - Already covered by Lucia's presentation
- Job status monitoring service
  - Not covering it here

Part III
System Architecture

Defining System Architecture

- Goal: Derive system architecture from features we want
  - **Not** from services we happen to have/get/whip up
  - Target architecture doesn't need to be in place up front
    - But practical service implementation should not compromise it
- EGEE/ARDA to provide guidance and recommendations
  - Research into prior art, communicate to experiments
  - Recommendations on what to base decisions on
  - Recommendations on what **not** to do
  - Present range of possible system architectures
  - Request: system architecture overview in middleware document
- Discussion in CMS, exchange in ARDA
  - See Lucia's presentation

Getting There

- Deliver now, continuously, in real life
  - Evolution, not revolution
  - Strongly prefer functional deep slices before diversity
  - We can live with short-term limitations
    - Example: "All muon data goes to INFN"
    - Simplified job submission is ok as long as it works
  - But don't sacrifice system architecture vision
- Technical aspects and assumptions
  - Catalogues: O(millions) is a boring triviality
  - Transfer system: trivially saturate present network capacity
  - We support and will use common guaranteed effort
    - E.g. castorgrid type services
    - If the service challenge uses a real system, we will be pleased

Architecture Features

- Some key term definitions
  - Dataset is consistent set of events
- Classify architectures based on key features
  - Pick system architecture when matching features found
  - Key distinctive features
    - Open or closed?
      - Can I enumerate all resources X on the grid?
      - How about my virtual organisation?
    - Opportunistic? What mixture of local and global optimisation?
    - Global, distributed or peer-to-peer?
    - Where does policy and strategy apply?
  - Deduce service composition from architecture, not vice versa

Simple Architecture Classifications (I)

- Opportunistic
  - "Pure Condor" style: move process freely anywhere
  - In HEP: event generation, possibly simulation type jobs
  - Restrictions on the job characteristics: shared libraries, network connections etc. not welcome
- Pipeline or black board
  - Jobs read data from and write to a central service
  - In HEP: e.g. min bias handling, much of what we've done to date
- Data Grid: split and move the black board around
  - Move data, not just jobs
  - We are just really starting to experiment with this (LCG-2, TMDB)
  - Key question: what detail is specified to decide where jobs run?

Simple Architecture Classifications (II)

- Opportunistic Data Grid
  - Like above, but use resources opportunistically
  - Substantially different system!
- General Resource Matching: "Large Scale LSF"
  - Data is just one of the job constraints
  - Other constraints: (COBRA) meta data, calibration data, software, data files, operating system version … — the list is long and growing
  - Much of the data is dynamic
- Opportunistic Resource Matching
  - A resource manager can
    - Match against existing satisfied constraints: file sets available
    - Change the constraints: move stuff around, install and even recompile software—each of these has an associated cost and latency
    - Live with the constraints that can't be violated or changed
  - Lots of prior art in operations research and dynamic programming: especially logistics and production planning for last 50 years

Architecture Classifications (III)

- What kind of a system architecture for these?
  - Opportunistic, black board, data grid specify themselves
  - What's best for opportunistic + (data grid | general)?
    - Peer-to-peer systems?
  - Who is reviewing and presenting prior research?
    - Present to both experiments and grid projects
    - Recommendations on what to base decisions on
    - Recommendations on what **not** to base decisions on
    - Best scheduling strategies
      - E.g. IP routing protocols we looked at for TMDB expressly said "do not use this algorithm when some of cost measures are dynamic"
  - Need to maintain sufficient system architecture "vision" and design knowledge in the experiment
  - Can we hear from gurus? FedEx?

System Architecture Topology

- Some questions on intended EGEE architecture space
  - The grid/vo is open?
    - Your local file system is closed, you can enumerate every file
    - You can't enumerate DNS, you can only list what you know
    - You can partially enumerate AFS using well-known contacts
  - Is it central, distributed or peer-to-peer?
    - Can I set up my own catalogues for sharing with my peers?
      - Should CMS even want to be able to do this?
    - Which services use which model? Do some services support hybrids?
      - High-availability central database (air-line reservation model)?
      - Local (partial?) databases, possibly able to operate when links go down?
      - Appears as central but really distributed, e.g. using referrals?
      - Who owns what data? Synchronisation model? Conflict resolution?

Resource Management

- Currently the main focus for resource planning has been
  - Computing capacity
  - Which files the job will access
- Inadequate for CMS' needs
  - Too low level for our model, higher level (e.g. data set) better
    - Otherwise experiments will do the real middleware
    - Not impossible: resource manager can make decisions using entities without knowing which physical resources they represent
- Can we help develop a better resource manager?
  - TMDB V2 is actually quite powerful engine for moving data sets around, just needs someone to tell what to move where

Higher Level Resources (I)

- Imagine a directory of high-level resources
  - Data sets with appropriate matrix/slicing (AOD, DST, Hits, …)
  - Computing elements, available capacity
  - What software is installed where
  - Which calibrations exist where, which generations?
- Services provide estimates of change costs
  - How long would it take to copy this data there? What latency?
  - Available computing capacity, can it be pre-empted
  - Quality-of-service, priority and policy
- Considerably less information than file catalogue
  - If resource broker doesn't need global file catalog, the whole file catalog design can become totally different, e.g. each site has its own replica, file transfers maintain coherence

Higher Level Resources (II)

- Some resource management options
  - Global and omniscient: all information is always valid
    - Resource manager's job is easy
  - DNS model: distributed, advertise life time
    - What does resource manager do when data set expires at site X in three hours? Schedule or not? Let job fail? Reschedule if the data isn't there when the job finally runs?
  - AFS/DHCP model: resource reservation
    - Sites publish resources
    - Resource manager obtains leases
    - Job completion releases the lease
    - Site/VO managers can retire resources respecting outstanding leases

Data Management

- Current expectations on data management services
  - Not all files are the same
  - Distributed databases (Dirk has good comments)
  - Currently expecting file needs to be expressed at high level
    - Cells of data matrix (dataset/slice/collection range)
    - If job description is wrong, not unreasonable for job to fail
  - On-demand file access and movement still required
    - CMS software has plug-ins to handle file access protocols (e.g. http, ftp, zip-member, rfio, dcache, …)
    - "Contract" with resource management yet to be worked out
    - Worker nodes expected to have outbound access
  - Not yet clear on multiple processing vs. replication

Security

- Not many comments right now
  - Read the classics already?
    - Multics Security Evaluation: Vulnerability Analysis
      http://www.acsac.org/2002/papers/classic-multics-orig.pdf
    - Thirty Years Later: Lessons from the Multics Security Evaluation
      http://www.acsac.org/2002/papers/classic-multics.pdf