Data Movement in HEP

Outlook for the next 5-10 years

Brian Bockelman

WARNING:

Claims here are often generalizations.
HEP is a large enough field there is always a counter-example.
All this is done with LHC-tinted glasses.

Conway's Law, redux

Before starting, think of Conway's Law:

"Any piece of software reflects the organizational structure that produced it"*

I propose a corollary:

Distributed computing infrastructure reflects the organizational structure that uses it.

Hence, I will take a minute to discuss how HEP computing organizes

*Restated: "If you have four groups working on a compiler, you'll get a 4-pass compiler.

Collaborations in HEP

- Physicists distributed across institutions; available computing resources are similarly.
  - **Data movement between resources is essential**.
  - The *biggest* computing resource may not be the most *important* resource to the physicist.
  - Ability to coordinate work is sometimes the limit, not available resources.
- Often collaborations are international:
  - Much of collaborating is negotiation: **no "common leverage" of a single funding agency**.
  - Data movement systems must **interoperate or overlay** resources that have drastically different approaches and priorities.
- Often large enough to afford specialization:
  - Computing organizations have **highly knowledgeable computing experts** they can depend on.
  - A double-edged sword: sometimes enough specialization that advanced **techniques can get "locked" into a collaboration**.

Computing in HEP

- Generate events, reconstruct simulation and detector data.
- Fixed number of workflow types run and developed.
- Work is planned centrally and (hopefully) well in advance.
- Significant CPU and data use per workflow type - cost-effective to optimize.
- **Limitation is hardware budget**.
- Output is datasets to use in analysis.

- Analyze datasets from central processing.
- Organized into “PI-plus-students-sized” groups.
- Huge variety: can have several workflow types per group.
- Work is continuous and chaotic - not planned centrally.
- Not enough experts to optimize individual workflows. Best hope is to optimize common tools and use cases.
- **Limitation is personnel time**.
- Output is “papers and science”.

A fundamental dichotomy is central processing versus analysis

Previous slides are *fundamental principles*, invariant over decade-long timescales

(Let's look at technical items)

File-based Data

- Around 15 years ago, the HEP field invested significant time and money into object databases.
    - This was widely viewed as a failure; similar levels of effort were required to dis-invest.
    - Since then, there's a social knee-jerk reaction against event-level object storage.
        - It's hard to make such databases interoperate regardless. Where they can be found, it's usually self-contained within a single system.
    - This reaction is fading as institutional memory fades; seeing more experimentation again.
- **However**, while *files* are important, POSIX-like *filesystems* are not. Medium-to-large experiments tend to keep file catalog outside the filesystem. Trends:
    - Directly using object stores for storing files. (Ceph, AWS S3)
    - Unifying disparate filesystems into a single data federation.

Custodial Data

- An important concept is *custodial data*; data a site is tasked to archive and make available for the rest of the collaboration.
    - The largest experiments sometimes keep multiple custodial copies; for the “average experiment”, there is just one.
    - Unlike additional copies, custodial data management is fairly static and deliberate. “Custodiality” is almost never changed.
- Custodial data is, without exception, kept on tape. This solution is sufficiently cost-effective, proven, and integrated into our processes that it is hard to see changes in the next 5-10 years.
- **Observation**: Use of HSM to make staging from tape transparent has widely been a failure for largest users. Most explicitly separate within their workflow management.
- **Challenges ahead**: Ratio of (disk buffer) / (total archive) is currently large enough that we can be sloppy with buffer management. May not be true in the future!
- **Challenges ahead**: The teams at the large DOE sites that manage custodial data are world-class. How can we make sure all HEP experiments have access?

Data Movement between Resources

- Data must be made available from the custodial location to other locations for processing.
    - Users and workflow systems can then utilize a variety of computational resources.
    - High level of **automation** is necessary - **we cannot afford to have a human element**.
- Important properties of solutions:
    - **When does the data move**? Streamed to job? On demand?
    - What happens when **data is unexpectedly not available**? Can we recover from an alternate source? Throw an error? Crash?
    - To what extent is data movement and **workflow management integrated**? Is there a feedback loop between the two systems? Is it a single integrated system?
    - What is the **access paradigm**? "Storage element"? Global file system? Data federation? Caches?
- There's a fine balance here - CPU efficiency gains versus cost of storage.
- Within the WLCG, we have gone through many models - started with highly static preplacement of data. (Accelerating) trend is toward dynamic placement, caching, and streaming. This **simplifies our use of disk at increased reliance on the network**.
    - In 5-10 years, 100Gbps is likely "entry-level"; this is 100Gbps to the 'site storage' and from offsite sto the compute cluster.

Distributed Data Management

- From a CS point-of-view, distributed data management is one of the most intriguing problems.
    - Wide solution space.
    - Ample opportunities for modeling.
    - Rich set of theoretical results - can pull in graph theory, autonomous agents, distributed services, etc.
    - Some solutions start to look like Name-Data-Networking, a new hot topic! Some of our scales make this look like a “Big Data” problem - another hot topic!
- However, it’s also an area where pure CS research has made little impact “on the ground.” The large-scale production systems have mostly come from within the physics organization.
    - **For discussion**: Why is this? Is CS too interested in prototypes? Is HEP too insular?

Data to the Cores

- The **foundational IO layer** in almost every HEP experiment **is based on ROOT**.
    - When advances are made here, they propagate to the rest of the field over due time.
    - Improvements to this layer are often more drastic than generational improvements in hardware. Can be order-magnitude without change hardware.
- Data delivered to cores from storage tends to be **non-sequential but mostly deterministic**.
- Data rates per “Haswell core” span from **100KB/s to 10MB/s**, depending on the application.
    - Distribution of data rates can have even further outliers at 100MB/s.
    - As CPU time increases with detector complexity, data rates have remained remarkably consistent in the past decade.
- For most workflows, earlier processing stages are more CPU-bound than later stages.
    - For some setups, data must come offsite - **outgoing TCP is necessary**.
- ROOT was born around the dawn of C++. Hardware has changed significantly since then. For applications on top to scale, we need better interfaces and APIs for vectorization, parallelization, and memory efficiency.

What is Missing?

- Notably absent is “industry standard Big Data” tools.
    - Think Dremel, Hadoop, Spark, etc.
- Several research prototypes have been done.
    - Significant effort is still needed for ROOT IO to interoperate with these higher-level frameworks.
- A key issue left unsolved is how interoperability works between these tools:
    - Batch systems result in a common API: the Unix process.
    - Virtualization- and container-based systems can use a common image.
    - How does one establish a common runtime between Hadoop and Spark?
- **Unless we solve the issue of a homogeneous interface, I don’t see adoption as possible.**

Projects To Watch

Want to see the future?
Watch these projects for a hint
of where the field is going!

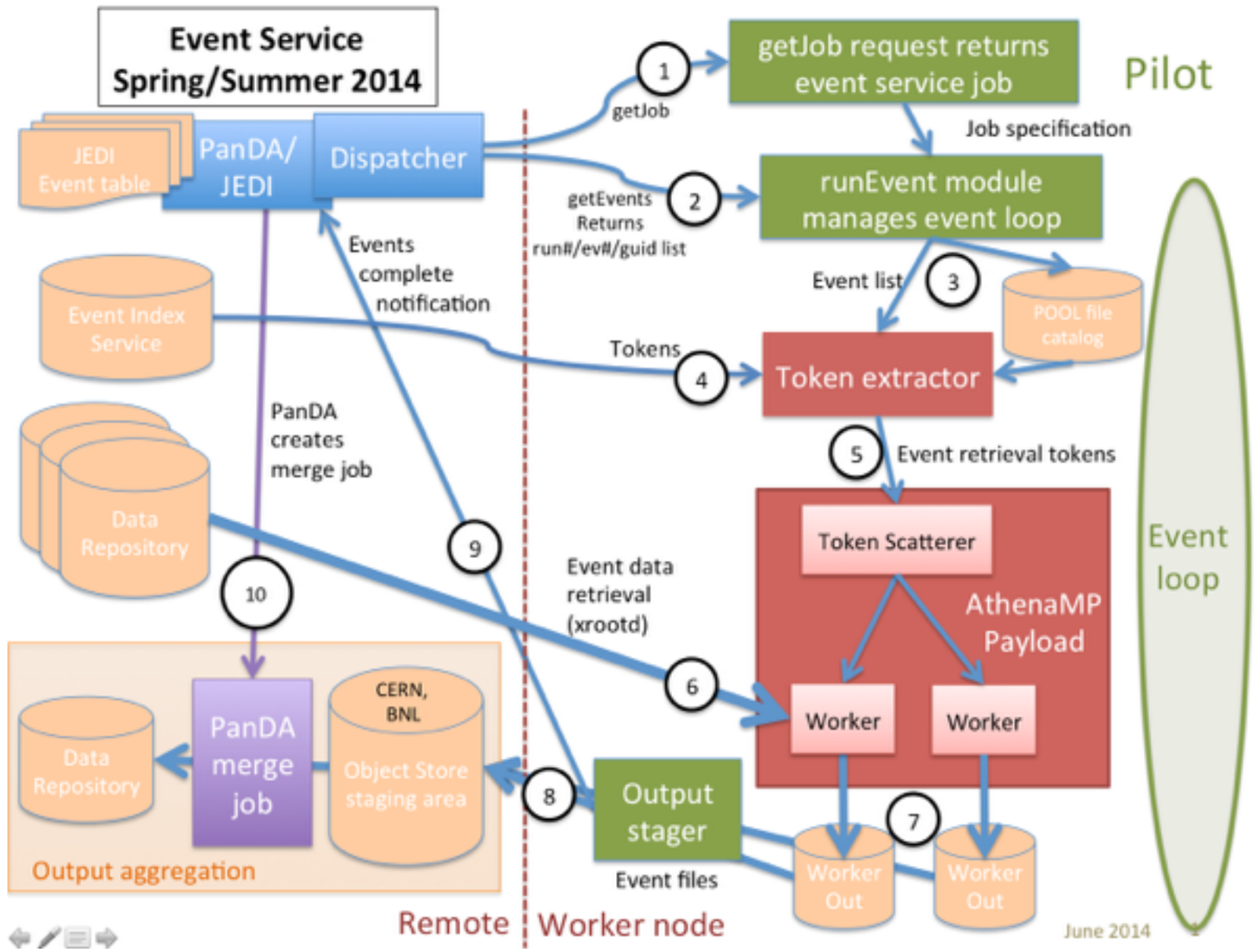
SAMGrid

- Data management system originally developed for the Tevatron Run 2 experiments.
  - Modernized and updated for FIFE (**F**abr**I**c for **F**rontier **E**xperiments).
  - Handles file locations, dataset definitions, namespace management, bookkeeping.
- Scale and complexity is smaller than LHC experiments.
  - Although there is an increased focus on dynamic dataset definition.
- **Importantly**, this is re-used by about a dozen experiments.
  - Question: What technical or organizational advantages does this software have that they have achieved wider reusability?

ATLAS Event Service

- Core idea: instead of breaking work into arbitrarily-sized jobs, have jobs work on smallest granularity possible. Gains tremendous flexibility for workflow systems.

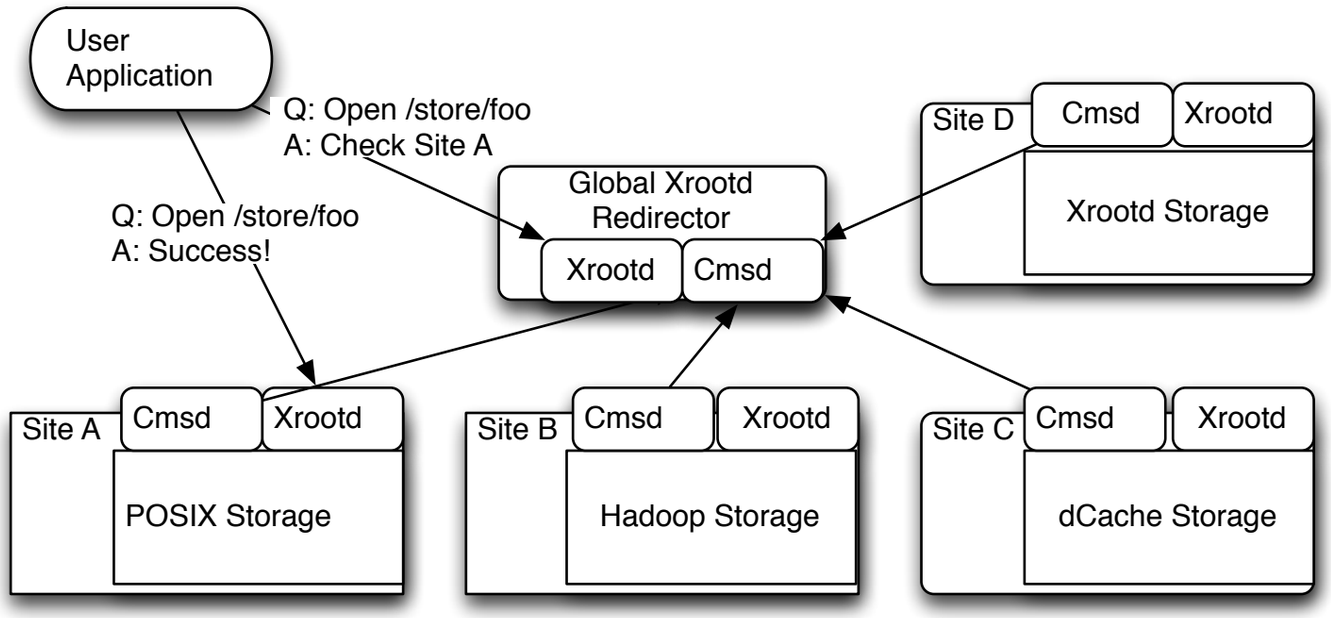
- Note use of object store for intermediate data management.



Questions? Ask next speaker, Torre!

AAA: Any Data, Any Time, Anywhere

- Goal: increase data accessibility within HEP through the use of data federations.
- Data Federation: A collection of disparate resources transparently accessible across a wide are via a common namespace.
- Simply put, the user can access any of their experiment's data from anywhere in the world without having to know details about location.
- Key aspect: **overlay** on top of existing storage system, as opposed to requiring special functionality.
- Significant R&D issues in IO provisioning, monitoring, and load-balancing a global system.
- Formal grant period wrapping up, but we've been highly successful in demonstrating the concept, operating a production service, and showing the popularity with users.



Questions?
Ask me!

ROOT IO

- In the end, all data moves through ROOT. If you want to make data movement faster / better, this is your first stop.
- ROOT IO was designed in an era of no vector units and with lower ratio of (memory latency) : (CPU cycle).
    - These APIs currently inhibit vectorization, parallelization, and make for poor memory locality.
        - With the Knight's Landing architecture around the corner, we must make targeted improvements!
    - As the common layer in HEP, changes here can help pull the field into future architectures.
- If we'd like to make progress on interoperating with other data processing stacks (Hadoop, Spark), we will need support on this layer.
- The ROOT team is aggressive and talented. For example, they just switched to a modern compiler architecture, allowing them to remove 500,000 lines of code!
    - For another case study in modernizing interfaces for parallelism, look at CMSSW's multithreading effort!

Take-aways

- Computing is a critical component of a successful HEP experiment. However, the underlying organization of these collaborations put interesting boundary conditions on how solutions are designed. These boundary conditions appear invariant on the 5-10 year scale.
    - **Limiting factor can be organization**, not necessary hardware resources!
- Managing custodial data is one of the most critical tasks, and perhaps the most mature.
- Wide area data movement is tightly coupled with larger workflow systems.
    - Trend is toward more **dynamic systems** which require **less specialized services** and **less human intervention**.
    - This is a significant, difficult optimization problem involving network, CPU, storage, and IO. Volumes of CS research in this area, but they've had little impact in this field.
    - Largest systems are built "in house" - although we're **seeing signs of greater re-use**.
- Within a site, the whole field has a common layer - ROOT IO - making it an ideal layer for investment. It will be **critical to adapt ROOT to tomorrow's architectures**.