The STAR Databases: Objectivity and Post-Objectivity

Torre Wenaus
BNL

ATLAS Computing Week
CERN
August 31, 1999

Content

I will address mainly the event store side of STAR databases

I will discuss

- Our starting point in terms of environment and requirements
- Objectivity
  - Initial decisions
  - Work and experience
  - Lessons learned
- Subsequent decisions and altered directions in light of experience
- Where we are now in STAR databases and event store
- Requirements and decision factors revisited today
- Conclusions

STAR at RHIC

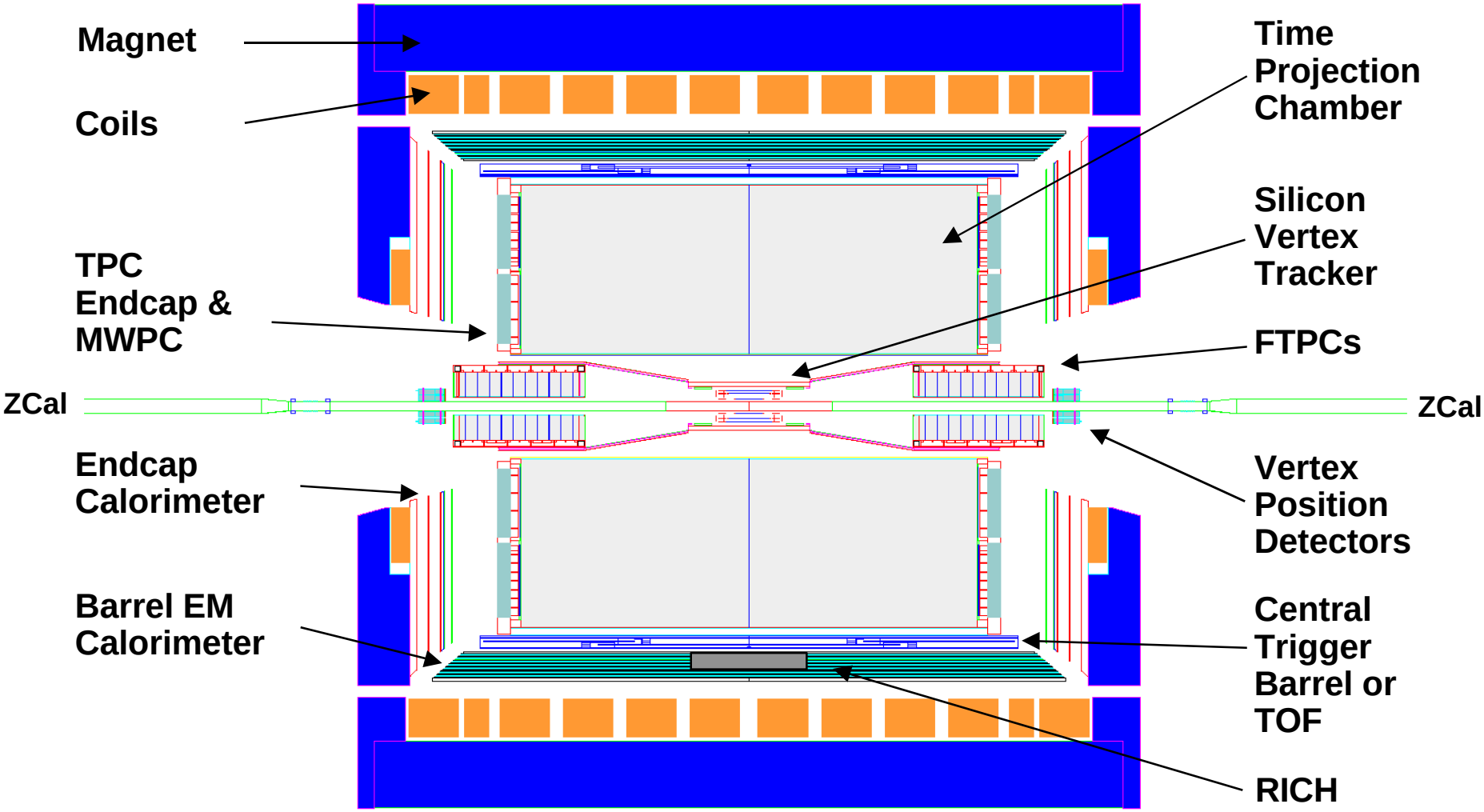
RHIC: Relativistic Heavy Ion Collider at Brookhaven National Laboratory

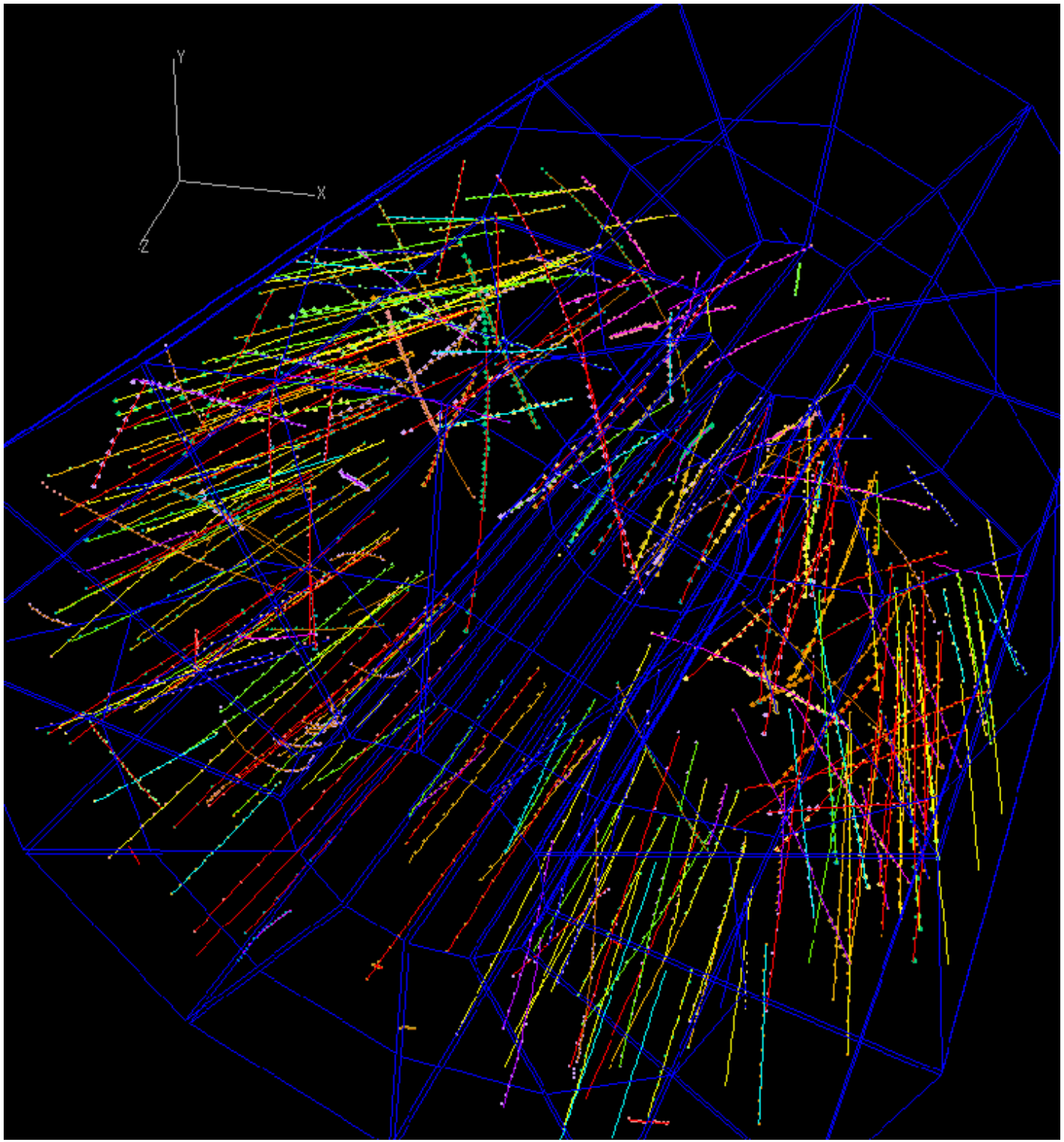
- Colliding Au - Au nuclei at 200GeV/nucleon
- Principal objective: Discovery and characterization of the Quark Gluon Plasma
- Additional spin physics program in polarized p - p
- Engineering run 6-8/99; first year physics run 1/00

STAR experiment

- One of two large 'HEP-scale' experiments at RHIC, >400 collaborators each (PHENIX is the other)
- Heart of experiment is a Time Projection Chamber (TPC) drift chamber (operational) together with Si tracker (year 2) and electromagnetic calorimeter (staged over years 1-3)
- Hadrons, jets, electrons and photons over large solid angle

STAR from the Inside - Out




Y
X
Z

The STAR Computing Task

Data recording rate of 20MB/sec; ~12MB raw data per event (~1Hz)

- ~4000+ tracks/event recorded in tracking detectors (factor of 2 uncertainty in physics generators)
- High statistics per event permit event by event measurement and correlation of QGP signals such as strangeness enhancement, J/psi attenuation, high Pt parton energy loss modifications in jets, global thermodynamic variables (eg. Pt slope correlated with temperature)
- 17M Au-Au events (equivalent) recorded in nominal year

Relatively few but highly complex events requiring large processing power

- Wide range of physics studies: ~100 concurrent analyses in ~7 physics working groups

Computing Requirements

Nominal year processing and data volume requirements:

Raw data volume: 200TB

Reconstruction: 2800 Si95 total CPU, 30TB DST data

- 10x event size reduction from raw to reco
- 1.5 reconstruction passes/event assumed

Analysis: 4000 Si95 total analysis CPU, 15TB micro-DST data

- 1-1000 Si95-sec/event per MB of DST depending on analysis
  - Wide range, from CPU-limited to I/O limited
- ~100 active analyses, 5 passes per analysis
- micro-DST volumes from .1 to several TB

Simulation: 3300 Si95 total including reconstruction, 24TB

Total nominal year data volume: 270TB

Total nominal year CPU: 10,000 Si95

Computing Facilities

Dedicated RHIC computing center at BNL, the RHIC Computing Facility

- Data archiving and processing for reconstruction and analysi
  - Simulation done offsite
- 10,000 (reco) + 7,500 (analysis) Si95 CPU
  - Primarily Linux; some Sun for I/O intensive analysis
- ~50TB disk, 270TB robotic tape, 200MB/s, managed by HPSS
- Current scale (STAR allocation, ~40% of total):
  - ~2500 Si95 CPU
  - 3TB disk

Support for (a subset of) physics analysis computing at home institutions

Offline Software Environment

Current software base a mix of Fortran (55%) and C++ (45%)

- from ~80%/20% (~95%/5% in non-infrastructure code) in 9/98
- New development, and all post-reco analysis, in C++

Framework built over ROOT adopted 11/98

- Origins in the 'Makers' of ATLFAST
- Supports legacy Fortran codes, table (IDL) based data structures developed in previous StAF framework without change
- Deployed in offline production and analysis in our 'Mock Data Challenge 2', 2-3/99

Post-reconstruction analysis: C++/OO data model 'StEvent'

- StEvent interface is 'generic C++'; analysis codes are unconstrained by ROOT and need not (but may) use it

Next step: migrate the OO data model upstream to reco

Database Development Effort

Infrastructure (including databases) development team: ~7 people

- Two less than planned minimum due to budget constraints

Database/event store development ~ 2.5 FTE-years in last 1.5 years

- Various fractions of (primarily)
  - Valery Fine, BNL
  - Yuri Fisyak, BNL
  - Victor Perevoztchikov, BNL
  - Jeff Porter, BNL
  - Sasha Vanyashin, Royal Stockholm Institute of Technology
  - Torre Wenaus, BNL

Strong reliance on leveraging community, commercial, open software

Initial RHIC DB Technology Choices

A RHIC-wide Event Store Task Force in Fall '97 addressed data management alternatives

- Requirements formulated by the four experiments
- Objectivity and ROOT were the 'contenders' put forward
- STAR and PHENIX selected Objectivity as the basis for data management
  - Concluded that only Objectivity met the requirements of their event stores
- ROOT selected by the smaller experiments and seen by all as analysis tool with great potential
- Issue for the two larger experiments:
  - Where to draw a dividing line between Objectivity and ROOT in the data model and data processing

Requirements -- And Fall '97 View of Fulfillment

Requirement	Objectivity	ROOT
Good C++ API	OK	OK
Scalability to RHIC data volumes	OK	No file mgmt
Adequate I/O throughput	OK	OK
HPSS compatibility	Planned	No
Integrity, availability of data	OK	No file mgmt
Recovery from permanently lost data	OK	No file mgmt
Object versioning, schema evolution	OK	Crude
Long term availability	OK?	OK?
Access control	OS	OS
Administration tools	OK	No
Backup, recovery of subsets of data	OK	No file mgmt
WAN distribution of data	OK	No file mgmt
Data locality control	OK	OS
Linux support	No	OK

Objectivity in STAR, with a BaBar Kick-Start

Objectivity selected for all STAR Computing database needs

- Event store
- Conditions and calibrations, configurations

Decision to use Objectivity would have been inconceivable were it not for BaBar leading the way with an intensive development effort to deploy Objectivity on the same timescale as RHIC and at a similar scale

- Benefit both from their code and their brush-clearing experience

Full BaBar software distribution was imported and used as an ‘external library’ providing the foundation for STAR event store and conditions database

- BaBar’s SRT and site customization features used to adapt their code without direct modification to the STAR environment

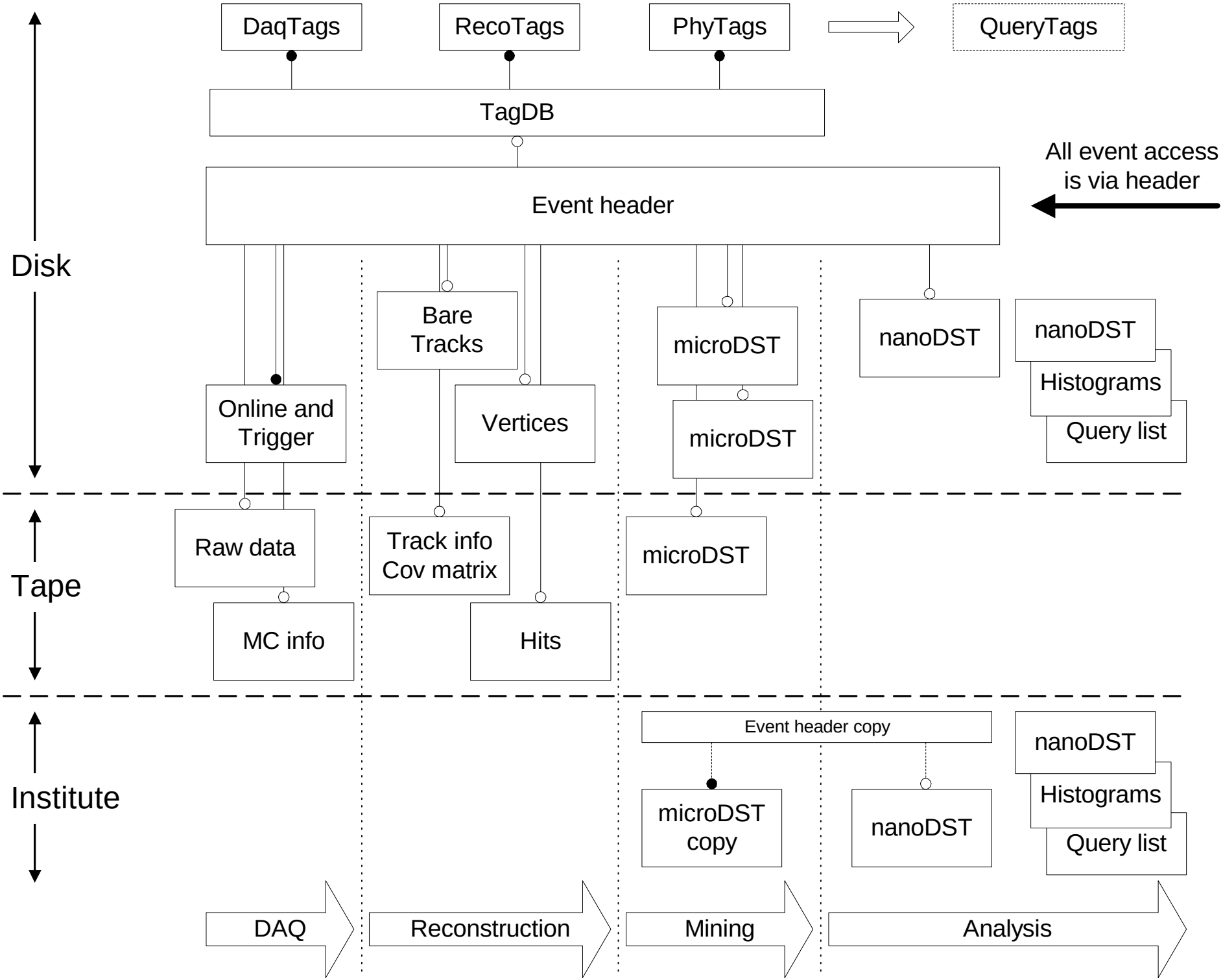
Event Store Implementation

BaBar event store software adapted to implement STAR event store down to the level of event components

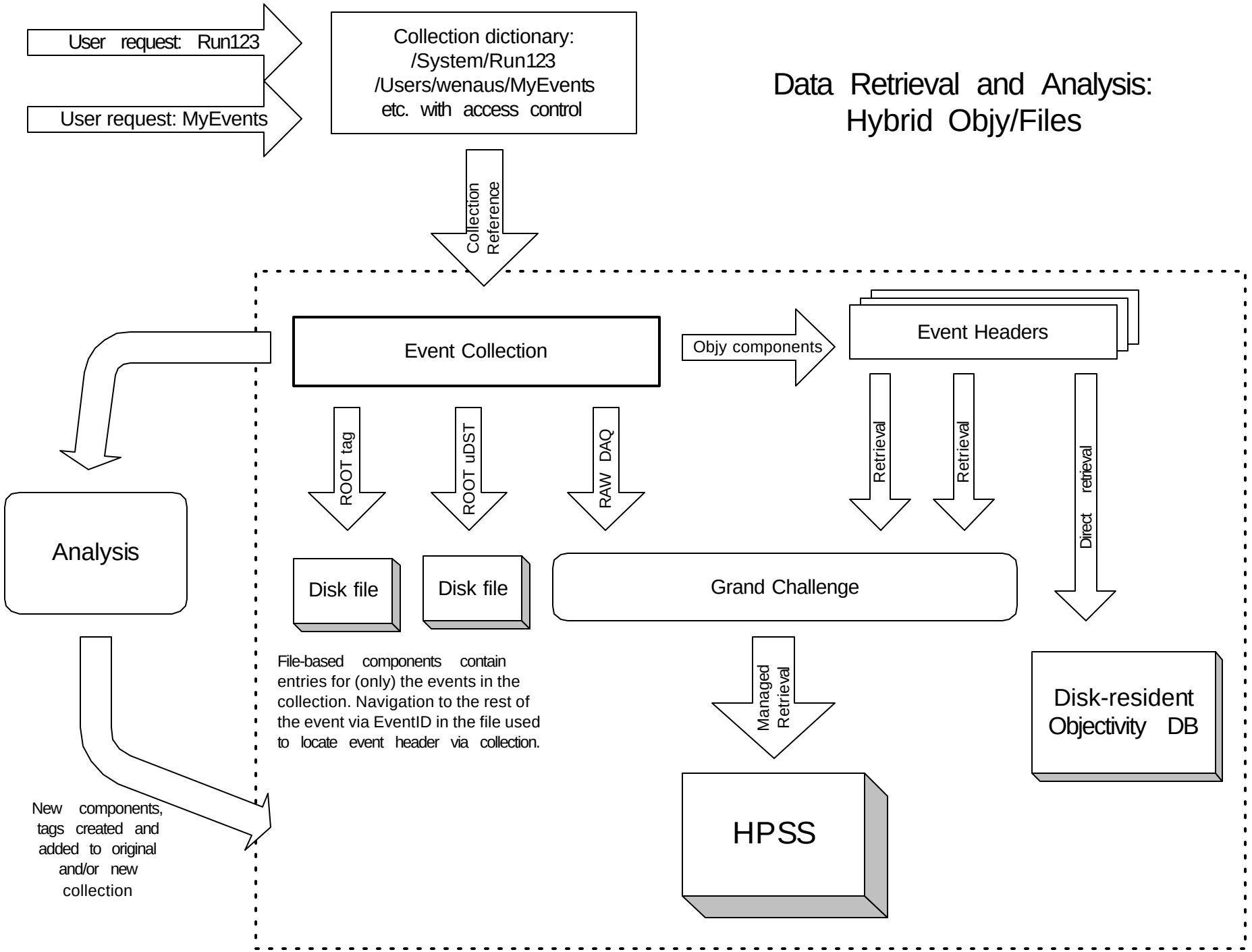
- Event collections; Event collection dictionary with organization in Unix directory tree style
- System, group, user level authorization controls
- SRT based management, build tools

Event component objects implemented in-house

- Direct mapping between objects and existing IDL-defined data structures (C structs)
- Flat structure under event header; flexible partitioning of components to different streams based on access characteristics


DaqTags
RecoTags
PhyTags
QueryTags
TagDB
All event access is via header
Event header
Disk
Bare Tracks
microDST
nanoDST
nanoDST
Histograms
Query list
Online and Trigger
Vertices
microDST
Tape
Raw data
Track info Cov matrix
microDST
MC info
Hits
Institute
Event header copy
nanoDST
microDST copy
nanoDST
Histograms
Query list
DAQ
Reconstruction
Mining
Analysis

Data Retrieval and Analysis: Hybrid Objy/Files



Alleviating Risk

Early (and easy) decisions taken to limit technology risk:

- Full separation of persistent and transient event models
  - BaBar transient-persistent mapping scheme not used
- Raw data not stored in Objectivity
  - Accommodation of file-based components in event collections
- StAF-standard XDF I/O retained as production standard and backup
- ROOT I/O pursued
  - to provide a future backup, replacing XDF, that provides a true object store
    - § remove restriction to table-based persistent data structures
  - as a competitive partial replacement of Objectivity
    - § addressing data storage, but not data management

Event Component Implementation

IDL-based persistent data model ensures XDF compatibility (fully technology-neutral; C structs with relations via indices)

- But restricts how we use Objectivity: don't even pretend it's C++

No impact on the transient event because of its decoupling from persistent

- We accept the cost of the translation; it's worth it
- At the DST level, data model is stable so code burden isn't an issue; neither is performance
- Decouples design optimization and development timescales of persistent and transient representations

Of course, in doing this decoupling, we are throwing away a good part of the advantage argued for Objectivity from the beginning

- No 'impedance mismatch'; an ODBMS can map directly onto the OO data models we want to build and work with

Regarded it as a necessity wrought of uncertainty

- can broaden our use of Objectivity later, in an environment of less uncertainty and more experience

Objectivity Deployment

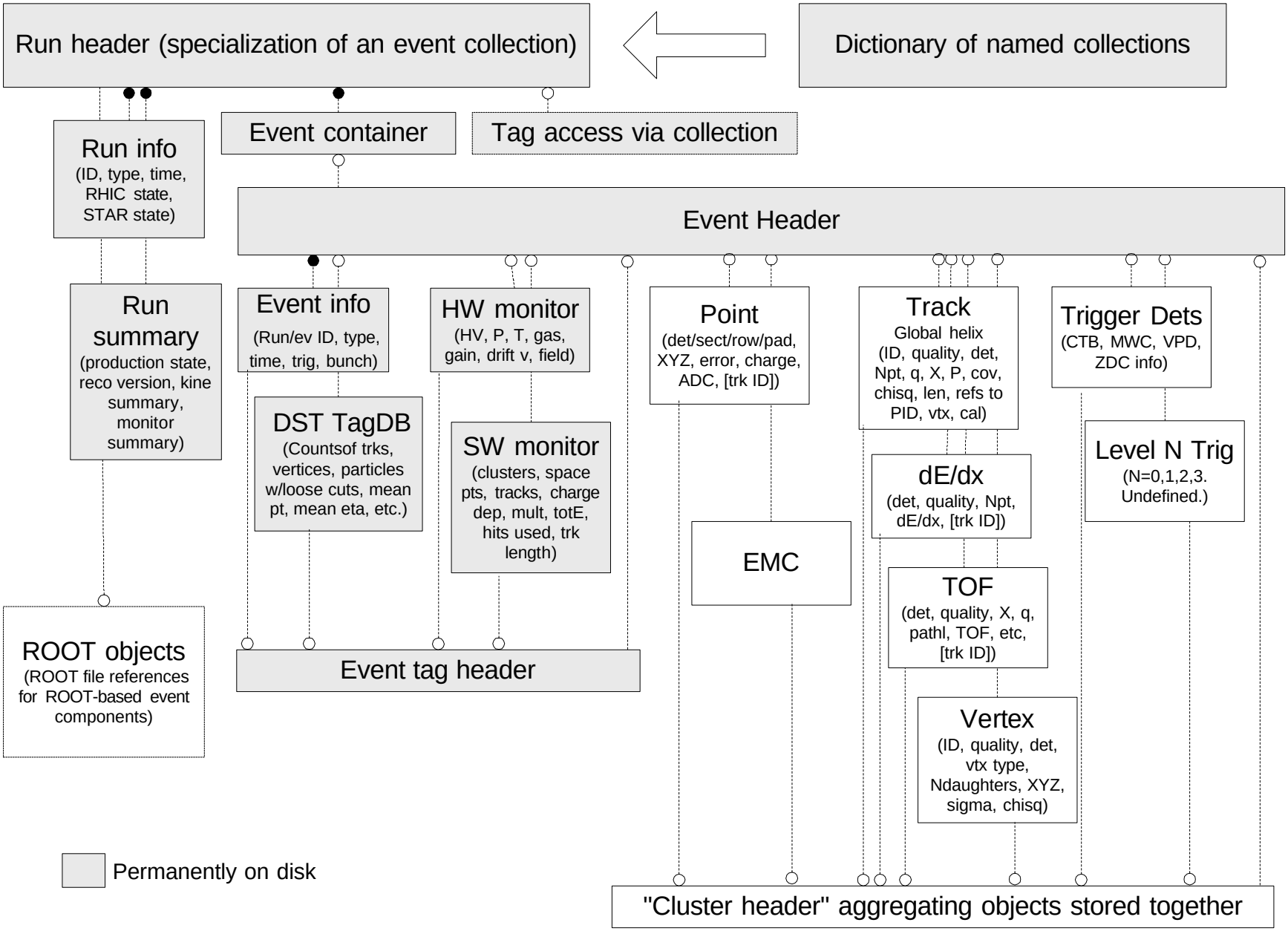
Objectivity event store deployed in Mock Data Challenge 1, Sep-Oct '98

- MDC: A RHIC-wide exercise in stress-testing RCF and experiment hardware and software in full production environment from (simulated) raw data through physics analysis

In STAR's MDC1

- 217k Au-Au events (1.7TB) simulated on offsite Cray T3Es with network transport to RCF HPSS
- 168k events reconstructed (600GB of DSTs generated)
- 50GB Objectivity DST database built (disk space limited) and used at a small scale with STAR analysis codes
  - Integrated with the offline framework with translators to/from the transient table-based DST
  - Used at a small scale with STAR analysis codes
  - Used by the Grand Challenge in testing their architecture

Objectivity-Based Persistent DST Event Model for MDC1



MDC1 Experience

Development and deployment quite smooth, but Objectivity development not a fast, efficient process

- Modify/rebuild/execute development cycle is heavy and slow, particularly if schema is changed
- It's an elegant tool but it requires *highly complex* software and software environment to develop real applications in the real world

No problems storing and working with 50GB of DSTs, in a very restricted setting

- Very small number of users
  - Access control and schema management tools not exercised in a realistic environment
- Serial usage only
- One platform (Sun/Solaris) only, and not our principal one
  - Linux port slow to appear and not keeping up with compilers
  - Big issue lurking: BaBar software not yet ported to Linux

Objectivity Burdens

The list of burdens imposed by Objectivity grew as our experience and lessons from BaBar mounted

- Management, development burden imposed by ensuring consistent schema in a single experiment-wide federation
- Schema evolution unusable if forward compatibility is desired (ability to run old executables on new data)
- Do-it-yourself access control, particularly with AMS
- Risk of major impact from platform lock-in due to porting delays; both Linux and Sun
- Scalability concerns (fall ‘98) -- lock manager performance issues in parallel usage?

A Hybrid Event Store for STAR?

No MDC1 show-stoppers, but many open issues and increasing concern over viability at our manpower levels

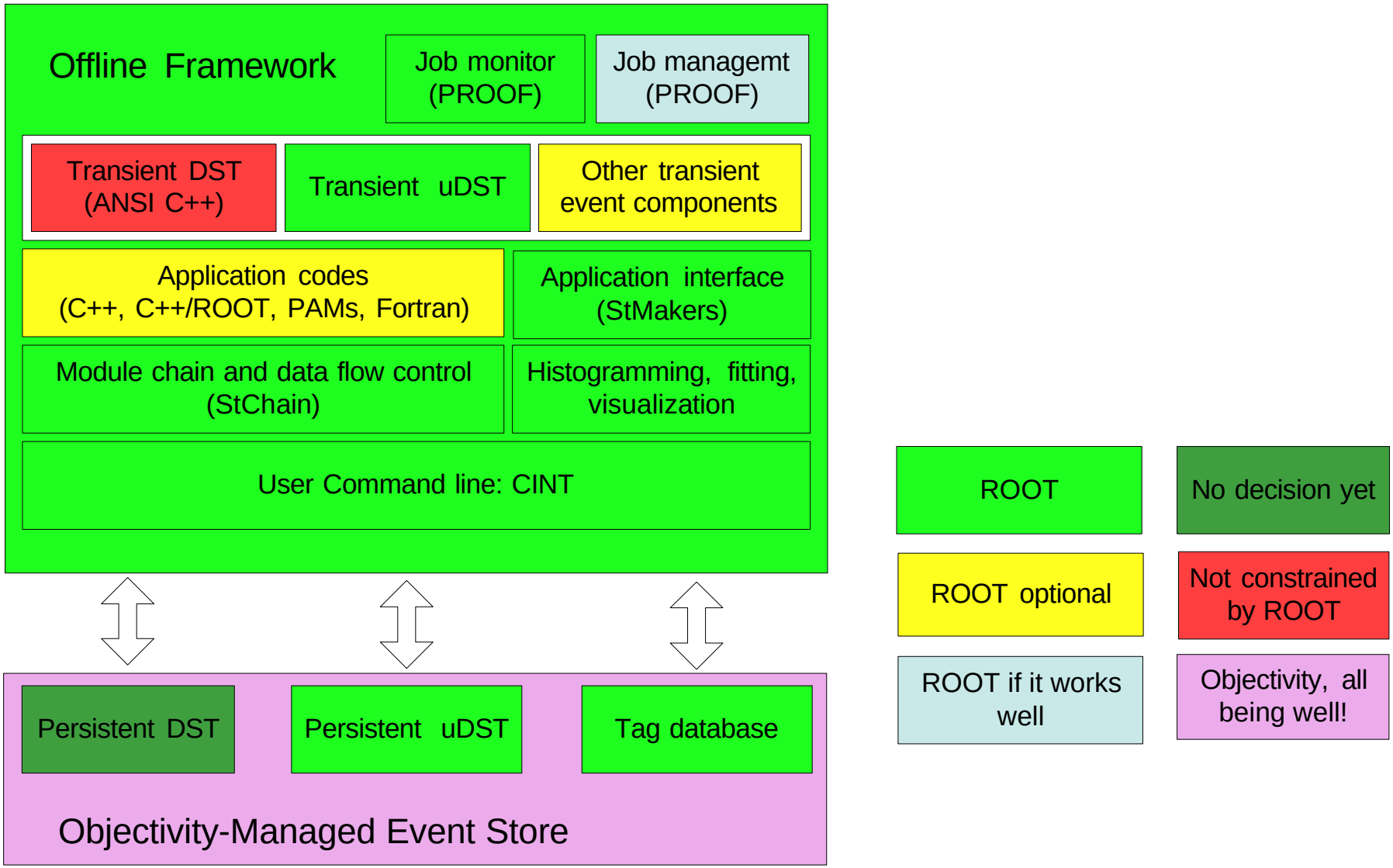
Series of review/decision workshops held in Oct/Nov '98

- How do we store our events?
- How and where do we use ROOT?
- How do we present our events to analysis codes, and what form should those codes take?

Decisions relating to data management:

- Deploy ROOT, together with Objectivity, as DST store in MDC2 (Feb-Mar '99)
  - Decide after MDC2 experience on ROOT vs. Objectivity
- Use ROOT as analysis toolkit and underlying basis for framework
  - Use ROOT for post-DST storage (micro-DSTs)
    - Reap advantages of close coupling to analysis toolkit

ROOT Decisions for MDC2 and Year 1



ROOT I/O Adopted as a Storage Solution

ROOT-based DST storage fully deployed and exercised in MDC2 production

- Revealed problems in robustness against crashes, subsequently fixed
- Meets requirements in multiple data streams for different event components

Robust I/O top priority for ROOT team (FNAL ROOT Workshop, March)

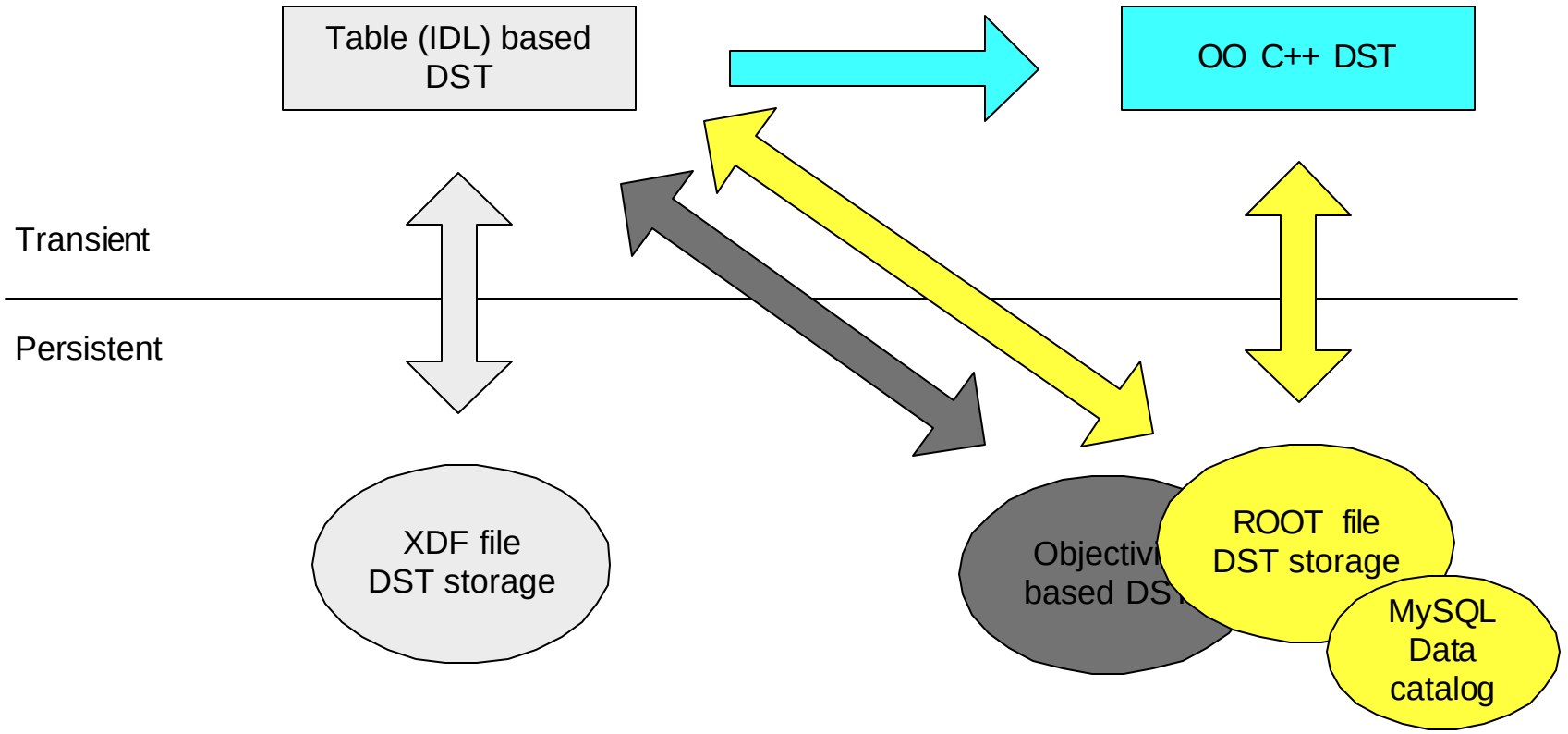
CDF's decision to use ROOT I/O in Run II also a factor

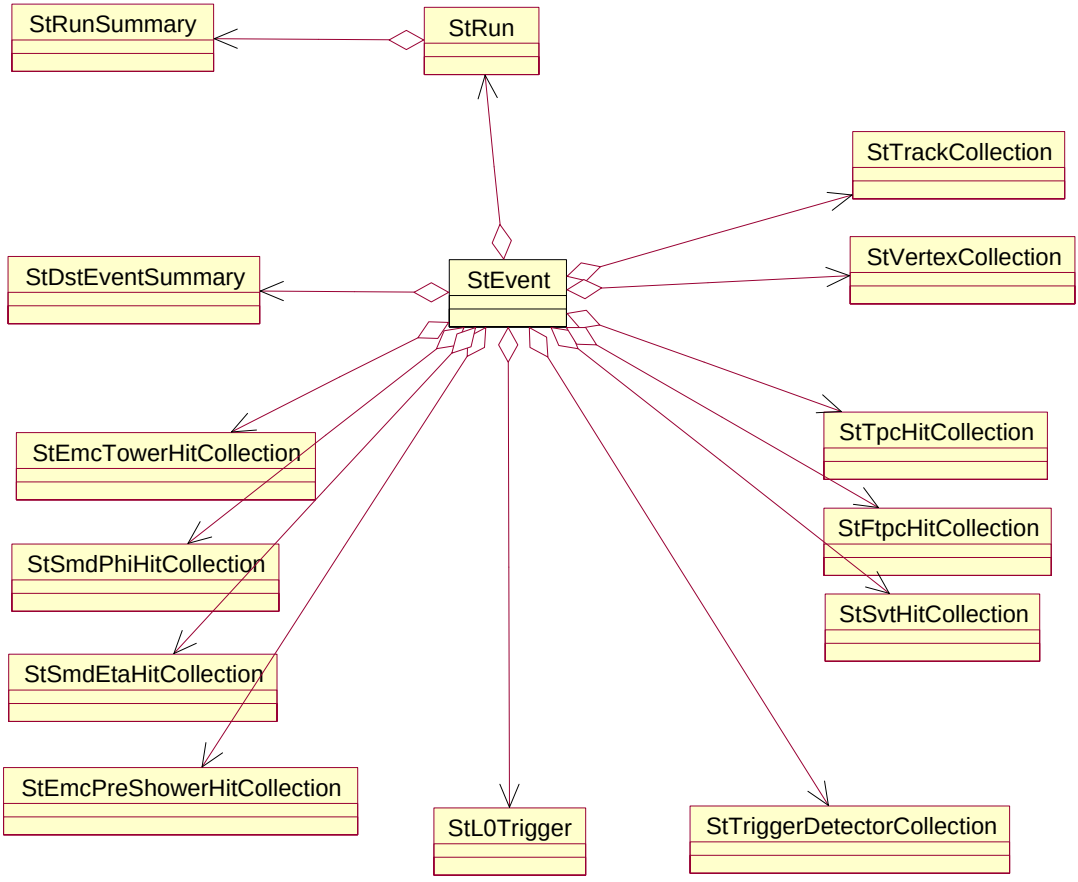
Conclusion: A functional and viable solution for an object store

- Easy decision to adopt it over Objectivity

ROOT has also been implemented as a persistent back end of our OO/C++ data model StEvent, giving us a direct object store *without* ROOT appearing in the data model interface.

Persistent and Transient DST Event Representations




StRunSummary
StRun
StTrackCollection
StVertexCollection
StDstEventSummary
StEvent
StEmcTowerHitCollection
StTpcHitCollection
StFtpcHitCollection
StSvtHitCollection
StSmdPhiHitCollection
StSmdEtaHitCollection
StEmcPreShowerHitCollection
StL0Trigger
StTriggerDetectorCollection

Requirements: STAR 8/99 View (My Version)

Requirement	Obj 97	Obj 99	ROOT 97	ROOT 99
C++ API	OK	OK	OK	OK
Scalability	OK	?	No file mgmt	MySQL
Aggregate I/O	OK	?	OK	OK
HPSS	Planned	OK?	No	OK
Integrity, availability	OK	OK	No file mgmt	MySQL
Recovery from lost data	OK	OK	No file mgmt	OK, MySQL
Versions, schema evolve	OK	Your job	Crude	Almost OK
Long term availability	OK?	???	OK?	OK
Access control	OS	Your job	OS	OS, MySQL
Admin tools	OK	Basic	No	MySQL
Recovery of subsets	OK	OK	No file mgmt	OK, MySQL
WAN distribution	OK	Hard	No file mgmt	MySQL
Data locality control	OK	OK	OS	OS, MySQL
Linux	No	OK	OK	OK

RHIC Data Management: Factors For Evaluation

My perception of changes in the STAR view from ‘97 to now are shown

Objy	Root+MySQL	Factor
þ	þ	Cost
þ	Ý	Performance and capability as data access solution
þ	Ý	Quality of technical support
ß	Ý	Ease of use, quality of doc
ß	Ý	Ease of integration with analysis
ß	Ý	Ease of maintenance, risk
ß	Ý	Commonality among experiments
ß	Ý	Extent, leverage of outside usage
ß	Ý	Affordable/manageable outside RCF
ß	Ý	Quality of data distribution mechanisms
þ	Ý	Integrity of replica copies
ß	Ý	Availability of browser tools
ß	Ý	Flexibility in controlling permanent storage location
ß	þ	Level of relevant standards compliance, eg. ODMG
þ	Ý	Java access
ß	Ý	Partitioning DB and resources among groups

Hybrid Event Store, Yes; With Objectivity, No

Adoption of ROOT I/O for the event store leaves Objectivity with one role left to cover: the true ‘database’ functions of the event store

- Navigation among event collections, runs/events, event components
- Data locality (now translates basically to file lookup)
- Management of dynamic, asynchronous updating of the event store from one end of the processing chain to the other
  - From initiation of an event collection (run) in online through addition of components in reconstruction, analysis and their iterations

But with the concerns and weight of Objectivity it is overkill for this role.

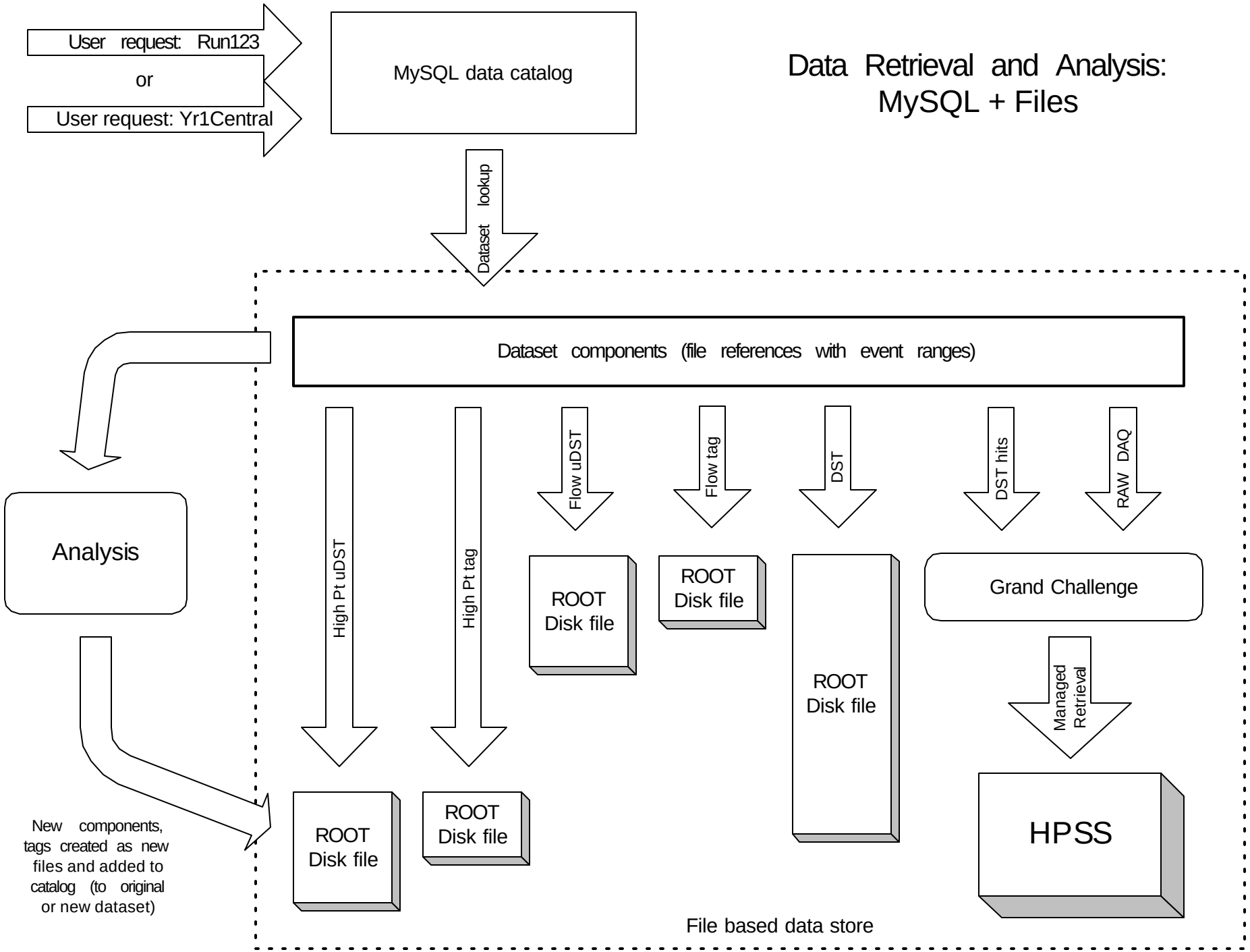
So we went shopping…

- looking to leverage the world around us, as always
- and eyeing particularly the rising wave of Internet-driven tools and open software

and came up with MySQL in May.

MySQL as the STAR Database

- Relational DB, open software, very fast, widely used on the web
- Not a full featured heavyweight like Oracle (hence simple and fast)
  - No transactions, journalling
- Development pace is *very* fast with a wide range of tools to use (good interfaces to Perl,C/C++, Java; easy and powerful web interfacing)
- Wonderful experience so far. Like a quick protyping tool that is also production-capable for the right applications
- In production use since June for file cataloguing and run logging
  - ~tables of ~20k rows cataloguing ~9TB of data
- Under test as the basis for compact event tags (and to test scalability)
  - No problem so far with 2M row tables of 100bytes/row
  - Query response in a few seconds
- Multiple servers, databases can be used as needed to address scalability, access and locking characteristics
- MySQL based conditions (calibrations, geometry, parameters, conditions) implemented
- Developing generic tools for STAR and outside use.


User request: Run123
or
User request: Yr1Central
MySQL data catalog
Data Retrieval and Analysis:
MySQL + Files
Dataset lookup
Dataset components (file references with event ranges)
Analysis
High Pt uDST
High Pt tag
Flow uDST
Flow tag
DST
DST hits
RAW DAQ
ROOT Disk file
ROOT Disk file
ROOT Disk file
Grand Challenge
Managed Retrieval
HPSS
ROOT Disk file
ROOT Disk file
New components, tags created as new files and added to catalog (to original or new dataset)
File based data store

Conclusions

The circumstances of STAR

- Startup this year
- Slow start in addressing event store implementation, C++ migration
- Large base of legacy software
- Extremely limited manpower and computing resources

drive us to very practical and pragmatic data management choices

- Beg, steal and borrow from the community
- Deploy community and industry standard technologies
- Isolate implementation choices behind standard interfaces, to revisit and re-optimize in the future

which leverage existing STAR strengths

- Component and standards-based software greatly eases integration of new technologies
  - preserving compatibility with existing tools for selective and fall-back use
  - while efficiently migrating legacy software and legacy physicists

After some course corrections, we have a capable data management architecture for startup that scales to STAR's data volumes

… but Objectivity is no longer in the picture.