EventStore
A Data Management System

Valentin Kuznetsov

with

Chris Jones, Dan Riley, Gregory Sharp

Cornell University

CLEO-c

- The CLEO-c experiment started in 2003
  - main physics topics are
    - precise studies of the D and D_s meson decays
    - Lattice QCD and search for glueballs
    - precise CKM measurements
  - Run plan
    - Phase I: $\psi(3770)$, $\mathcal{L}=3\ \text{fb}^{-1}$
      - 30 million $D\bar{D}$ events, 6 million tagged D decays
    - Phase II: $\sqrt{s} = 4140$ MeV, $\mathcal{L}=3\ \text{fb}^{-1}$
      - 1.5 million $D_s\bar{D}_s$ events, 0.3 million tagged D_s decays
    - Phase III: $\psi(3100)$, $\mathcal{L}=1\ \text{fb}^{-1}$
      - 1 billion J/Ψ events
  - we expect to collect 200TB of data
    - **data management is an issue**

Why EventStore?

- Many problems exist with the CLEO III data management system:
  - it is based on Objectivity/DB^{TM}
    - proprietary software restricts our choice of OS/compiler
    - abandoned by all other HEP experiments
  - very slow and doesn't scale
    - 2000 evt/s w/o data to memory on 500MHz Solaris
    - add 10x machines slows all jobs by 1/10
  - unnatural partitioning of our data
    - cannot run through different datasets in the same job
  - our implementation doesn't allow us to update data
    - can't just redo K_s finding in reconstruction

Requirement patterns

Data integrity:
data versioning
no corrupted data

Performance:
read 2000 evt/s
write 500 evt/s

Portability:
✓data must be portable to other sites
✓multi-platform support (Solaris,Linux)
✓should run with/without HSM and or caching systems
✓should run on laptop without full access to meta data

Security issues

Maintainability:
upgrades should be easy to install

Fault-tolerance:
handle heavy load, hardware failures

Support multiple file formats: raw, PDS, root, etc.

Reliability: short recovery time, high uptime.

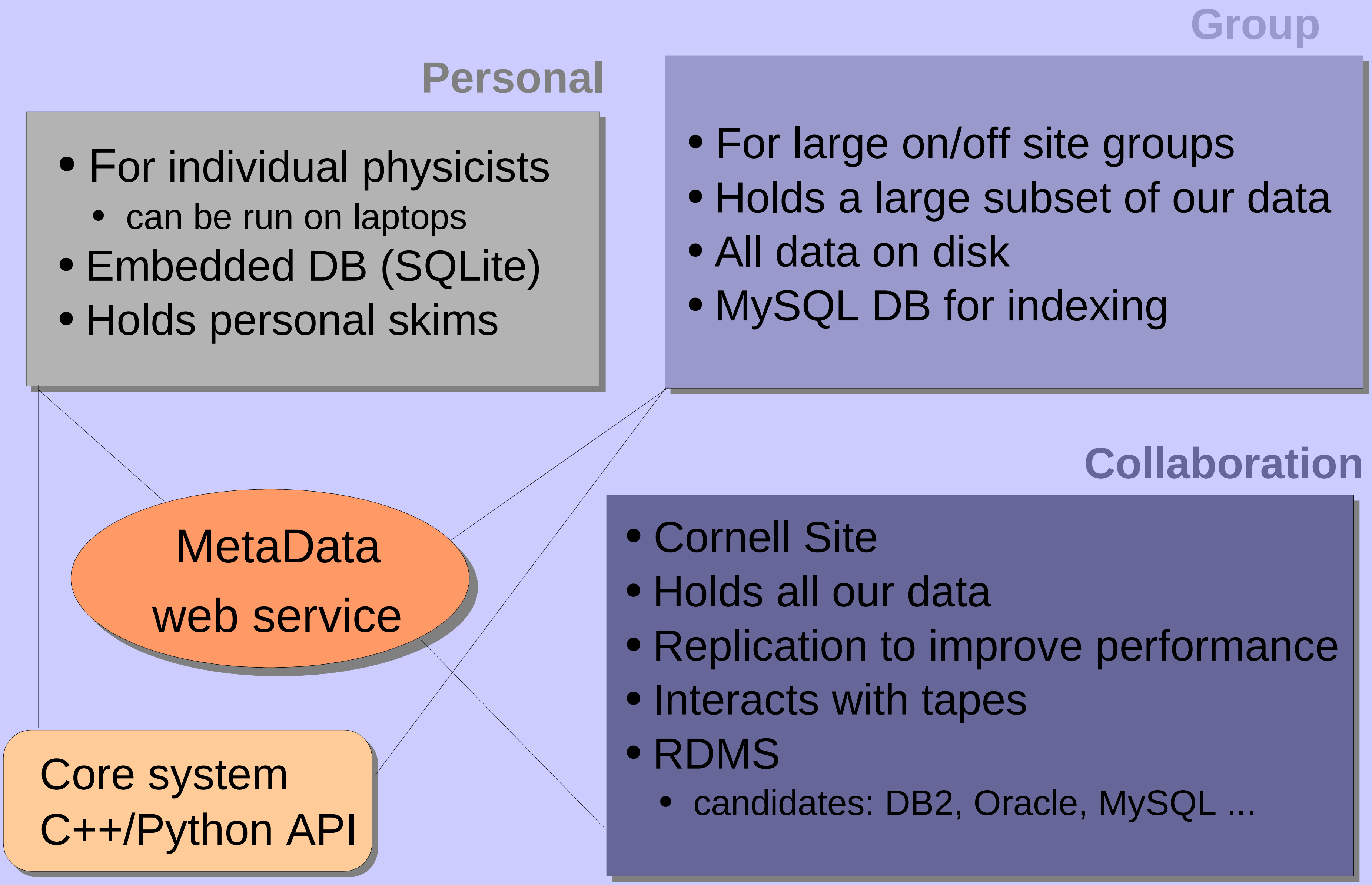
Remote data distribution:
transport part of the data to another site and make it available for users

Physics queries:
Support a variety of queries, e.g
✍ All data for runs taken at ψ(3770)
✍ request data by run/detector conditions
✍ run range 200111 to 210111
✍ datasets 31, 34

EventStore design

- We need
  - indexing: for random data access
  - versioning: for job reproducability and comparison
- Database evaluation:
  - different DB's for different use cases
  - SQL-like: for easy access/upgrade/common API
- Data relationship and organization:
  - grade: logically grouped data collections
    - e.g. raw, physics (approved for analysis)
  - view: event selection within a grade, e.g. qcd
- Necessary services:
  - location server, MetaData DB (for complex physics queries), etc.
- Prototype in Python ⇨ C++ for legacy application

EventStore “sizes”



Personal

- For individual physicists
  - can be run on laptops
- Embedded DB (SQLite)
- Holds personal skims

Group

- For large on/off site groups
- Holds a large subset of our data
- All data on disk
- MySQL DB for indexing

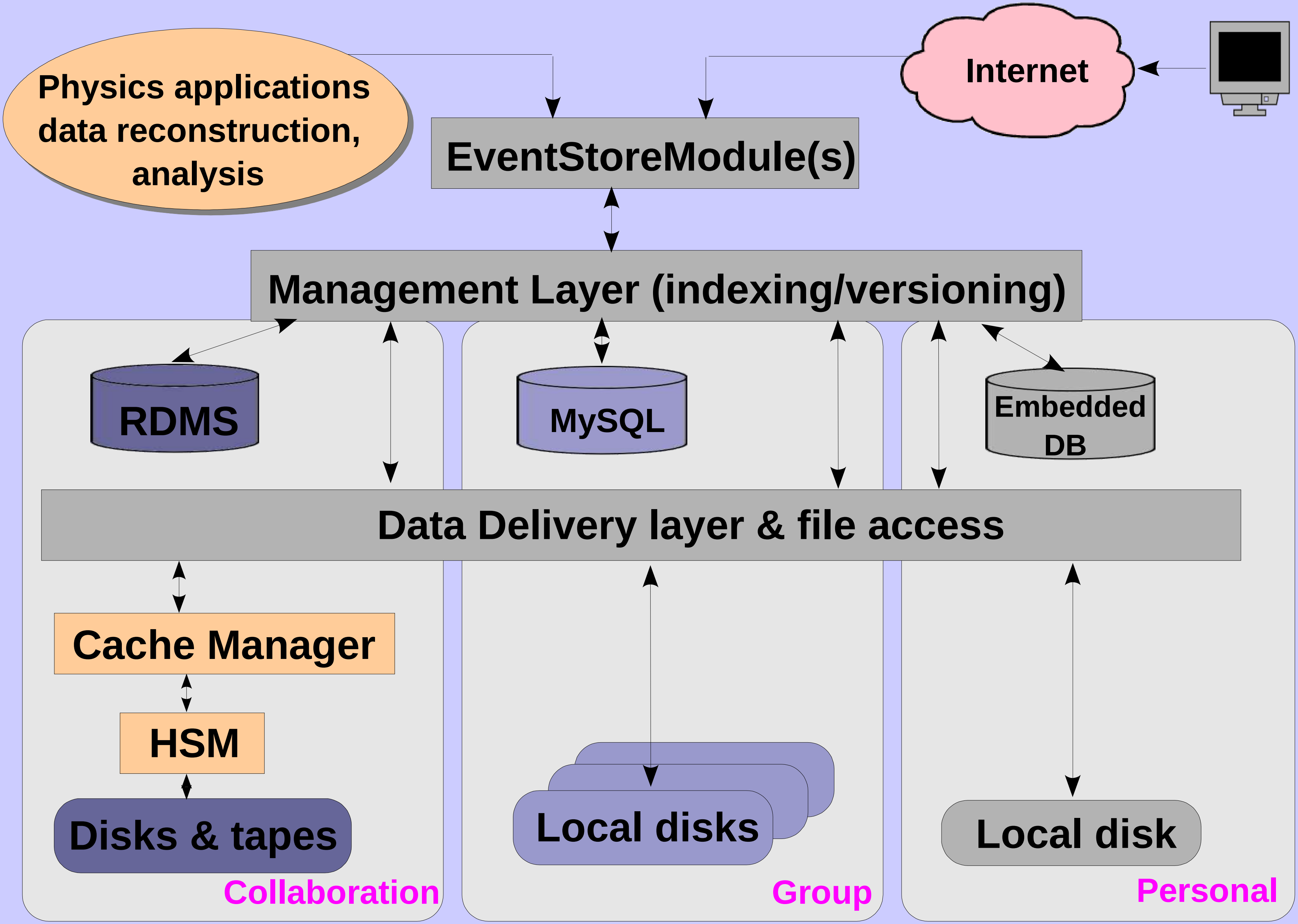
MetaData
web service

Core system
C++/Python API

Collaboration

- Cornell Site
- Holds all our data
- Replication to improve performance
- Interacts with tapes
- RDMS
  - candidates: DB2, Oracle, MySQL ...

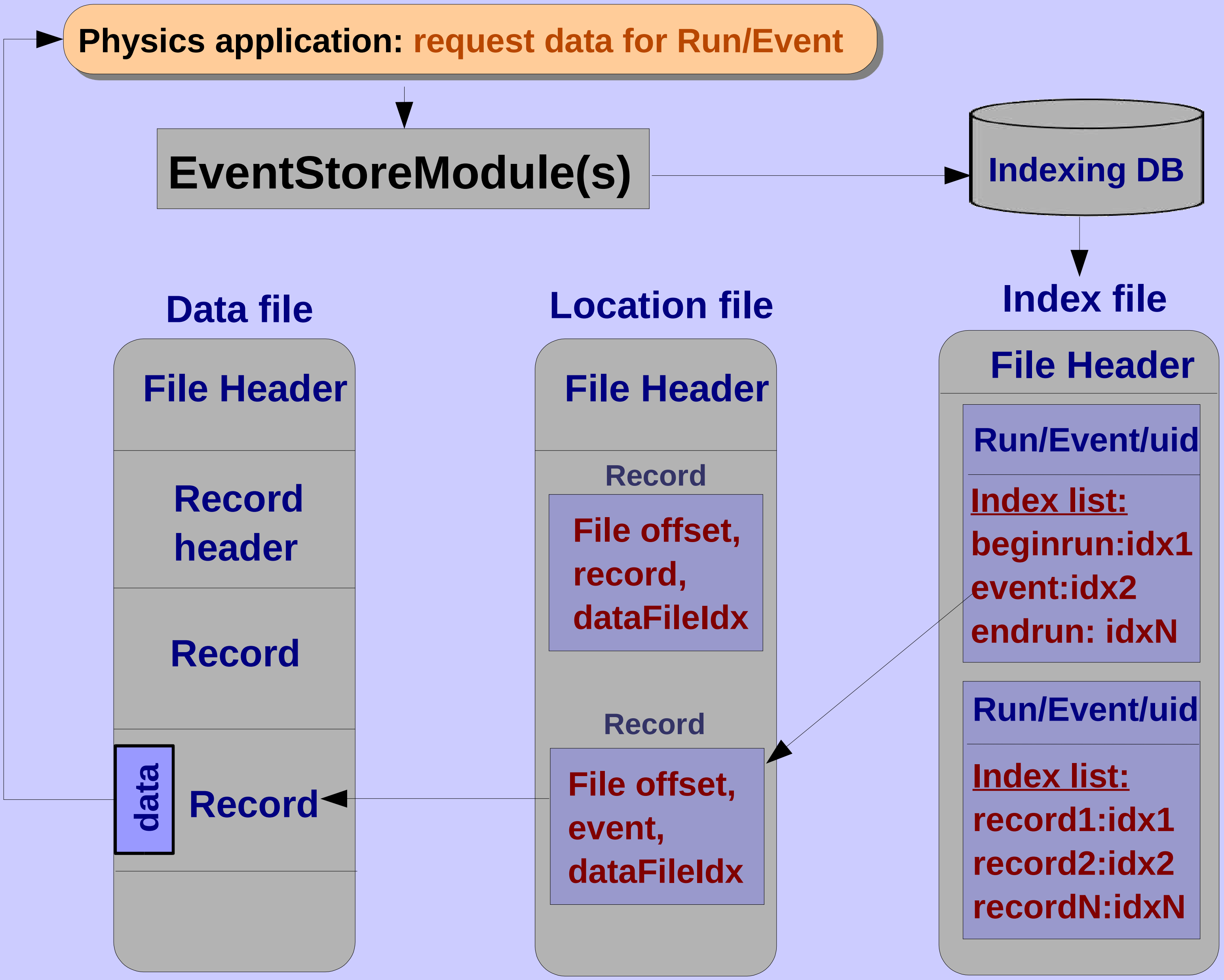
EventStore: architecture



EventStore: file formats

- EventStore supports “native” file formats
  - e.g. binary (raw data), pds (CLEO III file format)
    - Support for new file formats requires writing additional plugin; we're planning to add root and other formats
- EventStore knows location of data files which resides on disk/tape
  - Files can be moved around for load balancing
- EventStore creates auxiliary files for random data access (5% overhead to data file)
  - Format-independent “index” files for event finding
  - Format-dependent “location” files for random data access

EventStore: data lookup



Physics application: request data for Run/Event

EventStoreModule(s)

Indexing DB

Index file

File Header

Run/Event/uid

Index list:
beginrun:idx1
event:idx2
endrun: idxN

Run/Event/uid

Index list:
record1:idx1
record2:idx2
recordN:idxN

Location file

File Header

Record

File offset,
record,
dataFileIdx

Record

File offset,
event,
dataFileIdx

Data file

File Header

Record
header

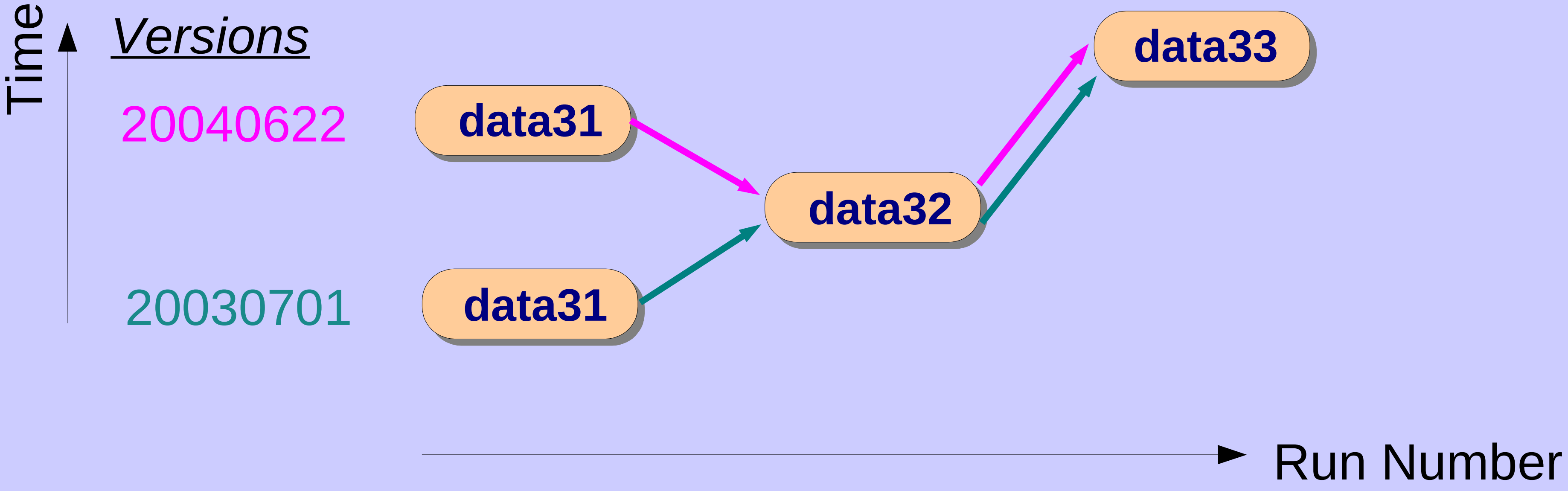
Record

data

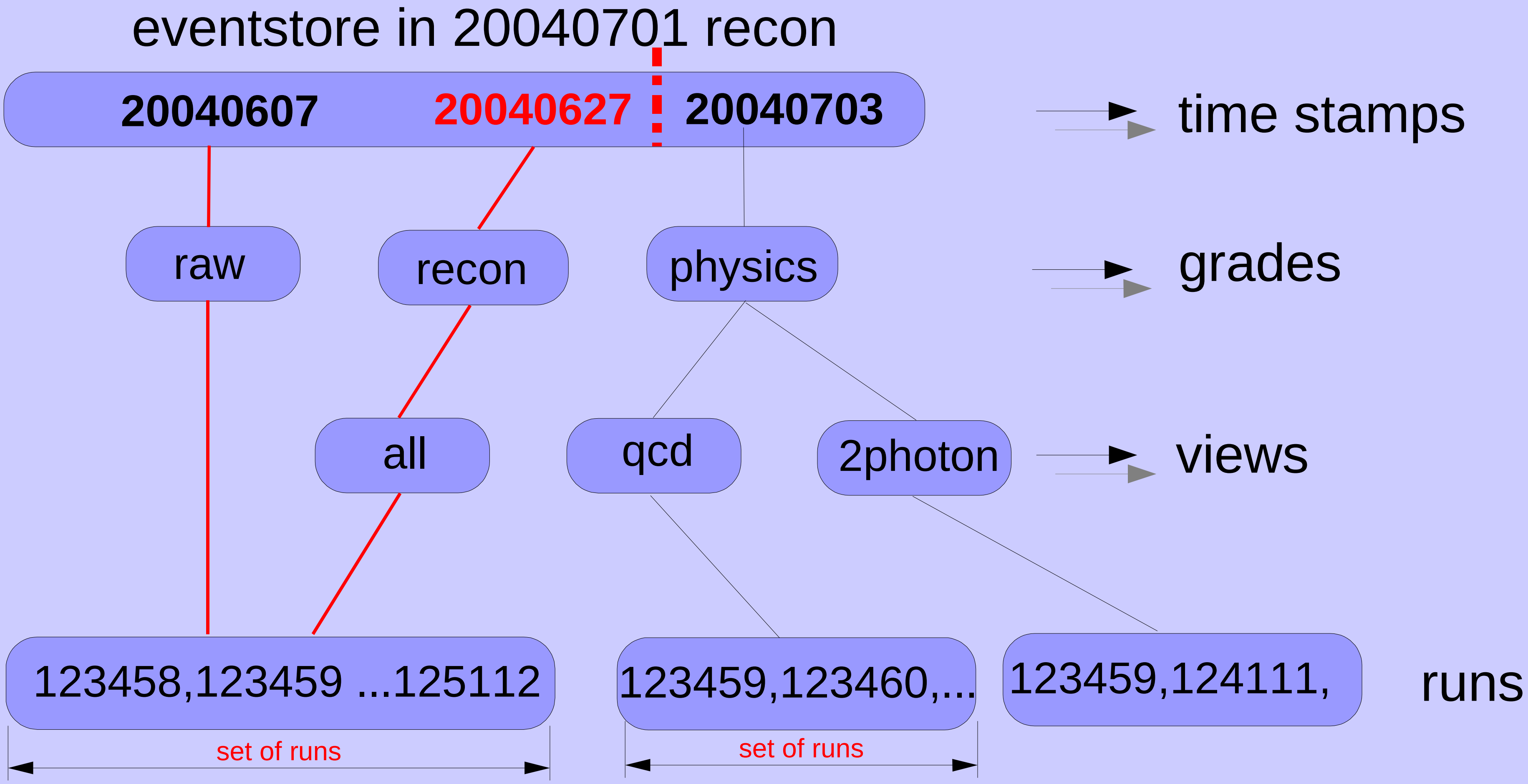
Record

EventStore: data versioning

- Users access data specifying date-stamp
  - EventStore finds the closest version before that date
- EventStore remembers version "evolution"
  - users always get consistent set of data
- If data reprocessed, assign new date-stamp
- When new dataset or skim is added, czar can append it to any date-stamp

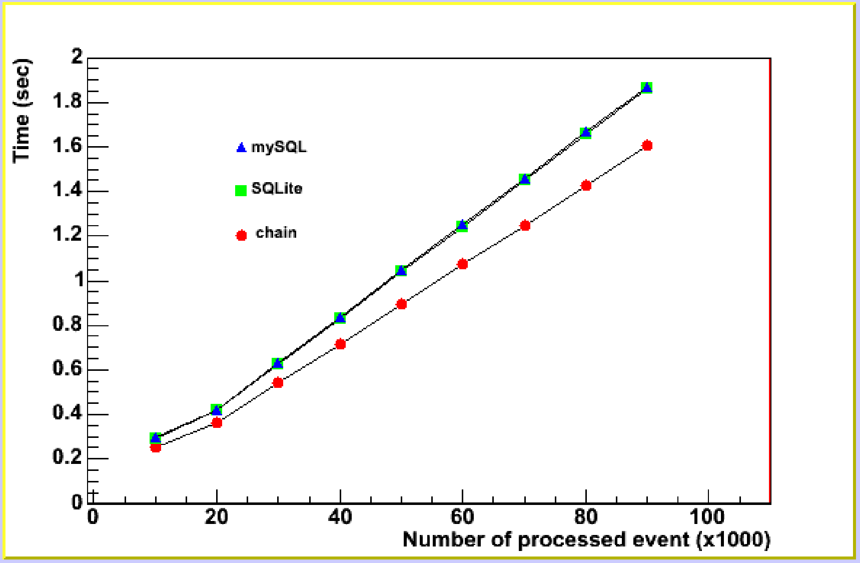


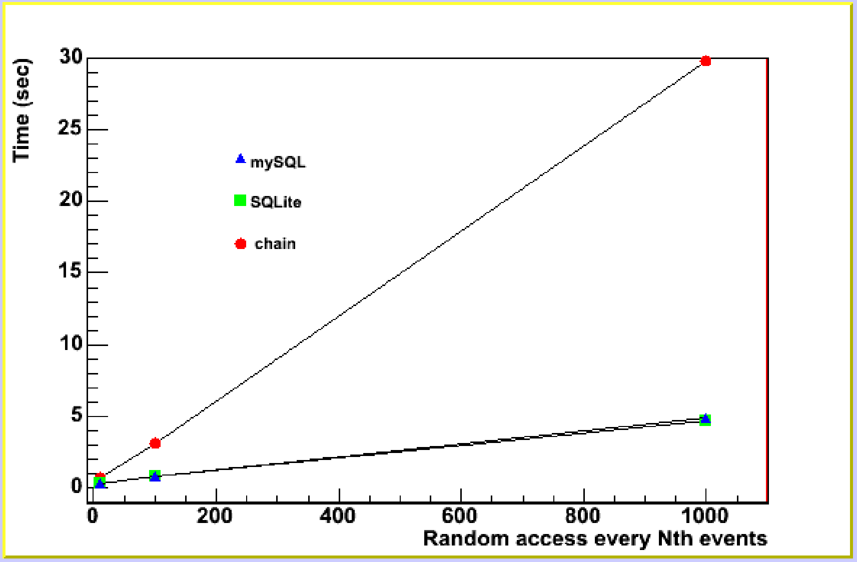
EventStore: granularity



Upon user request, the closest date stamp is chosen and all available data for this date are available

EventStore: performance





- Sequential data access is very compatible with a chain of files
  - Using Linux w/ data in memory on Intel/1GHz/256Mb:
    - 50K evt/sec on local IDE disk
- Random data access has tremendous improvements over chain of files
  - factor of 6 to access every 1000th event
- MySQL/SQLite show identical results

EventStore: current status

- EventStore 5+ times faster Objectivity/DB
  - Objectivity w/o data to memory 2000 evt/s
  - EventStore w/ data to memory 11000 evt/s

500 MHz Solaris

- Data stored in native format
- Can manage data and MC
- Data versioning:
  - can always get back data you used before
- Support simple physics queries:
  - dates, run ranges
- Run over multiple skims in same job
- Easy add/remove data to/from DB
- Random access to data

Summary

- EventStore has been released for users
  - very fast and robust
  - uses embedded SQLite DB (“personal size”)
    - “Group” EventStore under construction
  - Users are switching to EventStore:
    - all CLEO-C data presented:
      - 1800 runs, 120M events, 21K files
    - no file management and bookkeeping
    - no long scripts to chain files
    - simple interface
    - can do many new things not present in current system
      - random data access; run over multiple datasets; update data, etc.
- New data taken this fall will be in EventStore
  - Objectivity DB will be used for calibration only