

# *Introduction to BaBar Event Store*

**Jacek Becla**



- ◆ Impossible to give a comprehensive overview in < 30 min!
- ◆ This talk:
  - Gives a general overview of main features and some implementation details
  - Highlights some pitfalls we fell into
  - Mentions some ideas to steal

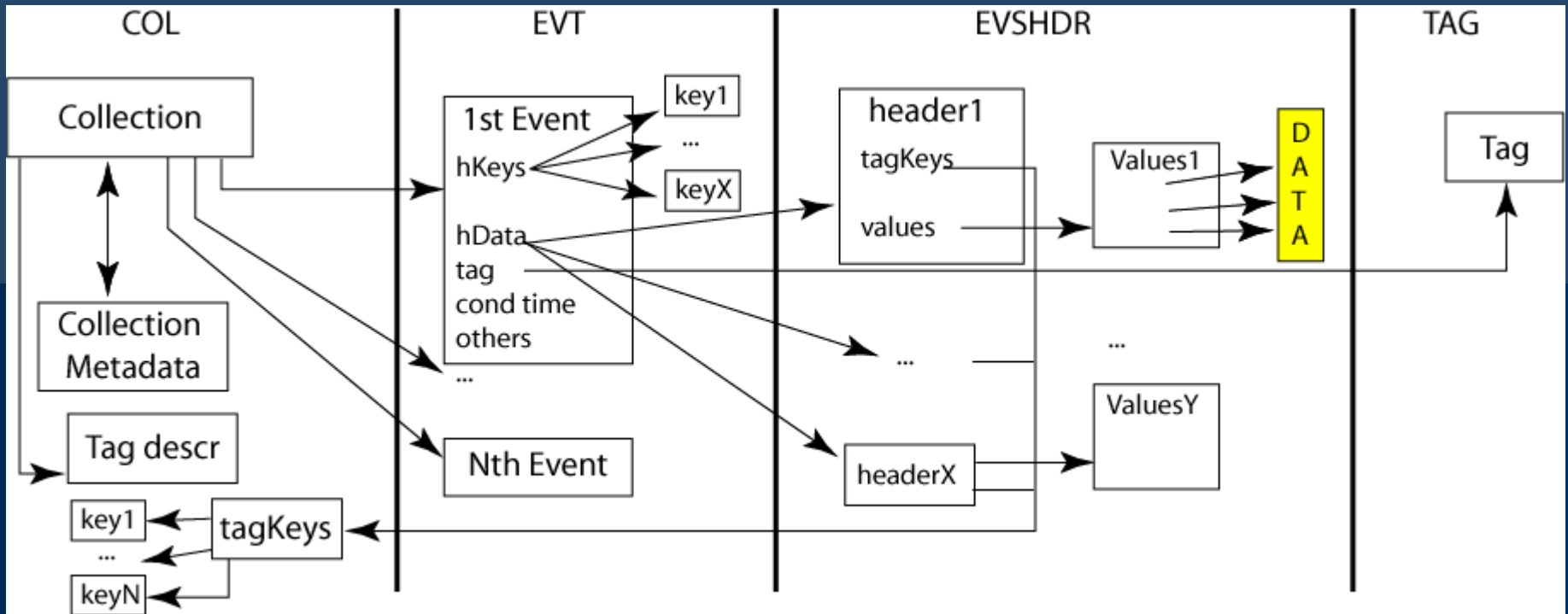
# *Event Store*

- ◆ Manages all BaBar's events
- ◆ Entry point: collection
- ◆ Main challenges:

- Billions of events
- Millions (soon billions) of collections
- Multiple writers and/or readers
- Tens of servers, hundreds of disks
- Ever increasing demands and constantly changing requirements
- Scalability
- Robustness
- Space vs. performance
- Complexity
- Long term survival

- ◆ It is a low-level code

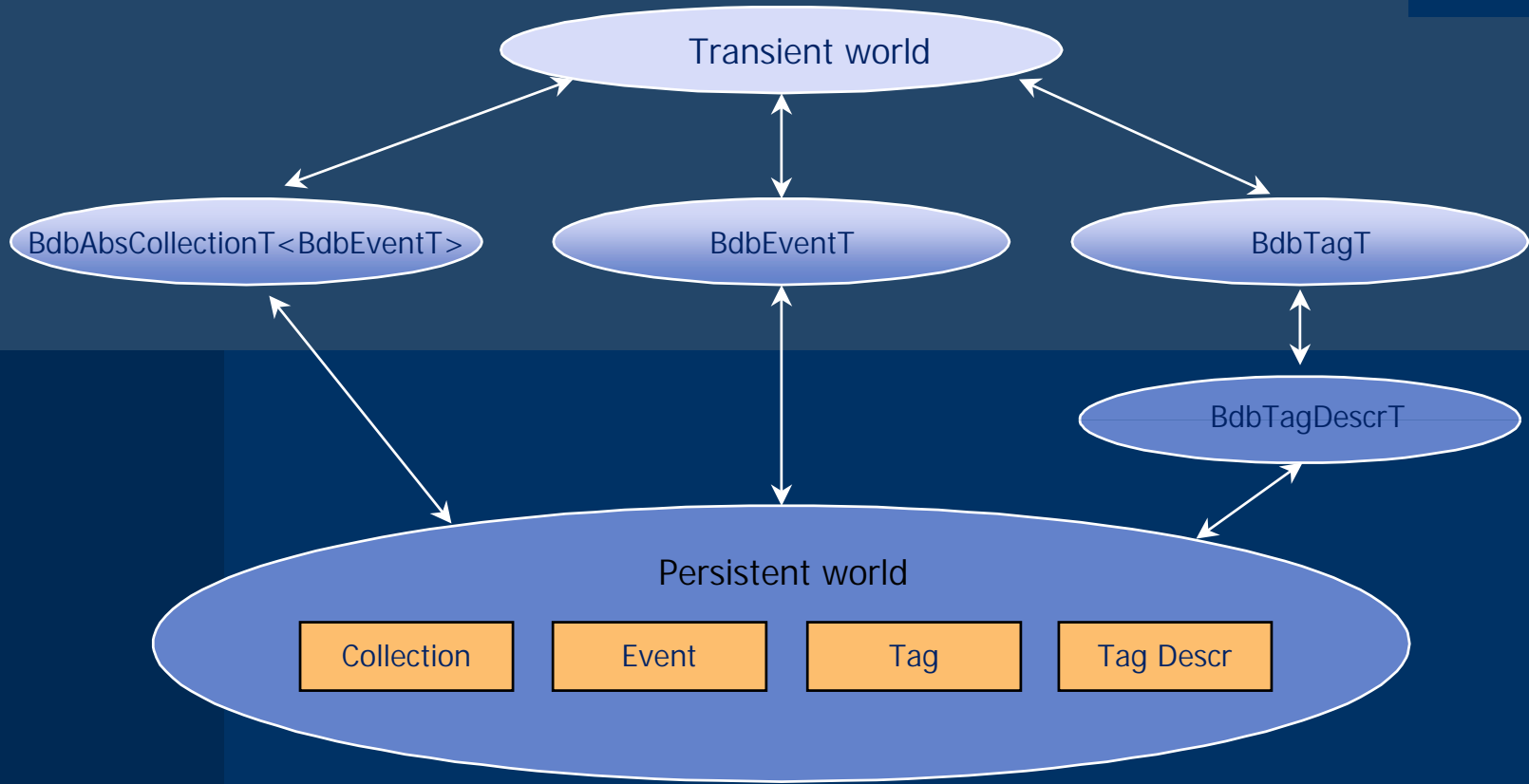
# Current EVS Design



# *Main Features*

- ◆ Scalable, hierarchical collections
- ◆ Arbitrary data types (headers)
- ◆ Logical/physical split, placement
- ◆ Transient/persistent separation
- ◆ Plus *many* small features, some listed on next slides

# *Transient-Persistent World Separation*



Recently tightened and strengthen.

# Collections

- ◆ Addressing  $10^9+$  events is non-trivial
  - Tens of “streams”, over hundred “skims”, pointer collections, user collections
  - Or....bridge, slave, tree, vector collections...
- ◆ Some features
  - Hierarchical, e.g. can add collections to other collections
  - Collections can span files, servers
  - Iterators
  - Collection metadata
  - Mapping col → files (available in release 14)

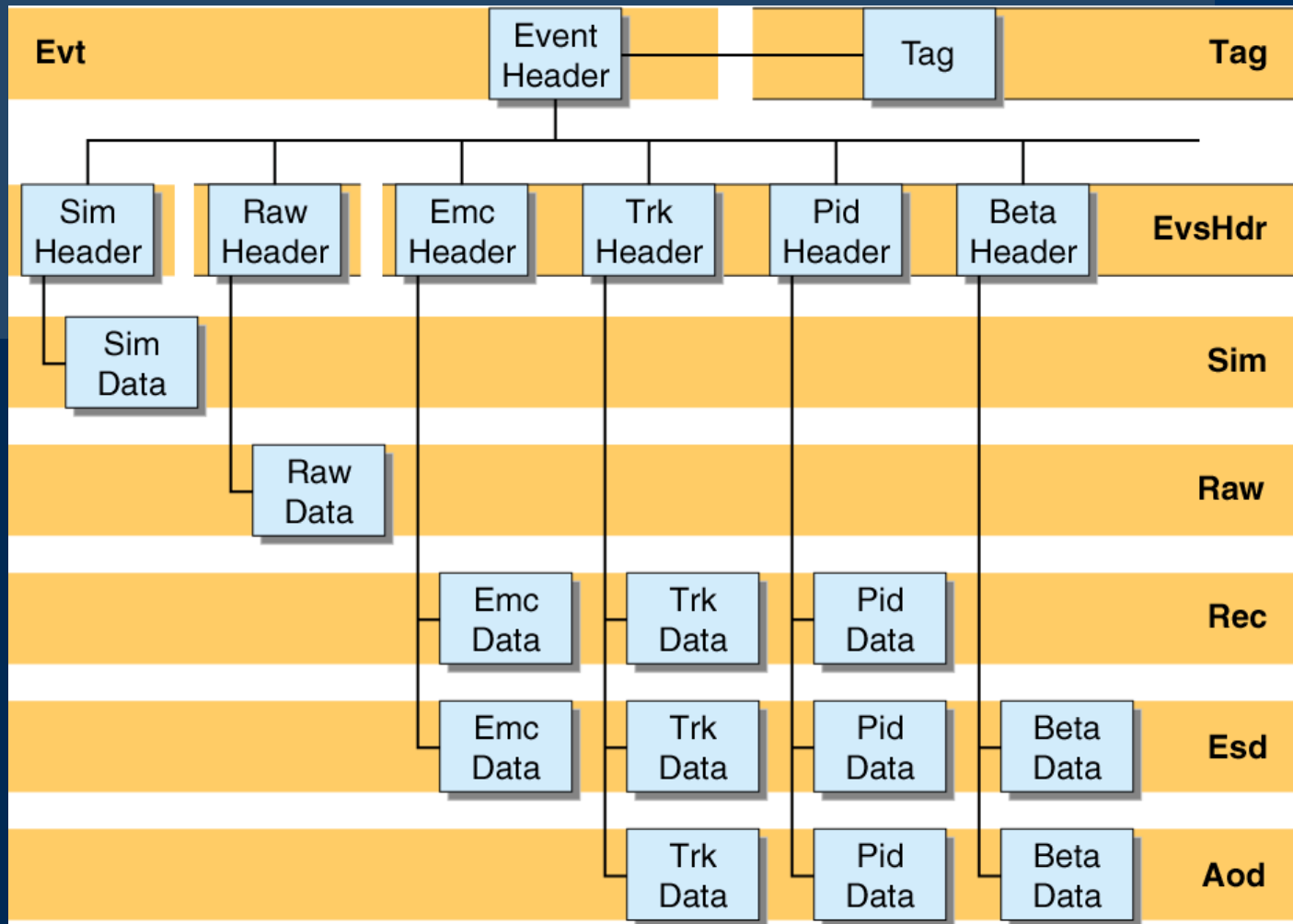
# Headers

- ◆ Provide indirection between event and event data. Allow to:
  - handle arbitrary chunks of event data transparently
  - extend event without changing persistent schema
    - saved us quite a few times: e.g. eidList, renewing tag, stateID, new mini
- ◆ Borrowing/renewing - avoid duplicating

# *Placement / Clustering*

- ◆ Correlated pieces of events stored together
  - Col, tag, evt, evshdr, aod, esd – separate files
  - Streams never share files
- ◆ Logical / physical mapping
- ◆ File dispersification
- ◆ Easily reconfigurable file parameters

# Logical / Physical Mapping



# *Transaction Management*

- ◆ Allows multiple writers and readers to coexist / share files
- ◆ Allows to recover from crashes
- ◆ Prevents from data corruption
  - Insures operations are atomic
  - ACID properties
- ◆ Most important part of each **database** system
  - It is a large and most complex part of Bdb code
    - Will have to be replaced with other techniques/strategies in the ROOT-based system
- ◆ Not specific to event store, but complexity (e.g. access to multiple fds) forced by evs

# Tags

- ◆ Sharing achieved by storing borrowed tag as a header
- ◆ Alternative - create new tag. Requires creating a new full-blown event
  - Or use specialized event “TagEvent” (in rel. 14)

# Tags descr

- ◆ Maps the name of each tag value to its type, and offset within the event tag varray
  - Slow (many lookups)
- ◆ Current limit: one per collection
  - Artificial - to avoid writing tag descr per each event.
- ◆ “Original” and “default”
  - Confusing and unnatural
- ◆ Contains lots of extra info, e.g. even if only few fields stored persistently, all possible field *names* always stored
- ◆ All above addressed by evs redesign

# *Common Objects*

- ◆ Most data same for many events
- ◆ Compression helps. Price: CPU cycles
- ◆ Better to come up with schema that well matches data
  - Common objects – objects shared by many events
  - Available in rel. 14

# Others...

- ◆ Merging events
- ◆ State ID (in rel. 12)
  - state ID vector (in rel. 14)
- ◆ Authorization
  - Prevents users from overwriting production data
- ◆ Inhibit system
  - Allows to take offline a subset of data for maintenance
- ◆ (Lack of...) deletion
  - Important or not? Decision can influence design (keeping track of owner)

# Essential / Important Features – Summary

	Available?	Covered by redesign?	Requires cross file refs?
Scalable collections	Yes	Improved	Probably
Headers	Yes		Yes
Placement	Yes		No
Recovery from crashes	Yes		No
Merging events	Yes		No
Trans/pers separation	80%	20%	No
State ID vector	1 state id	Yes	No
Authorization	Yes		No
Inhibit system	Yes		No

# Consider / Do Not Forget

	Available?	Covered by redesign?	Requires cross file refs?
Allow multiple tag descr per collection	No	Yes	No
1 tag desc per tag	No	Yes	No
Validation of coupling tag ↔ descr	No	Yes	No
Collapse headers with events on transient level	No	No	No
Store only tag desc attribute names that are used	No	Yes	No
Cache transiently event's parts, flush event to pers. store at once	No	Yes	No
"TagEvent"	No	Yes	Maybe
Come up with persistent schema that well matches data		Yes	Likely
Use common objects to store common data	No	Yes	Likely
Deletion	No	?	Likely
Identifiers clashes	Yes		No
Navigational components in RDBMS	N/A	N/A	No
Load on servers (e.g. # open files)	Yes		No
Maintain domain separation	Yes		No
Moving files between hosts/sites	Yes		No
Backwards compatilby	N/A	Yes	No

# Summary

- ◆ Objy-based event store
  - Solid design, working
  - Improved & refreshed by ongoing redesign
- ◆ Recommendations
  - Reuse code
  - Use our experience
  - Learn from our mistakes
  - Minimize major changes