ROOT @ BABAR

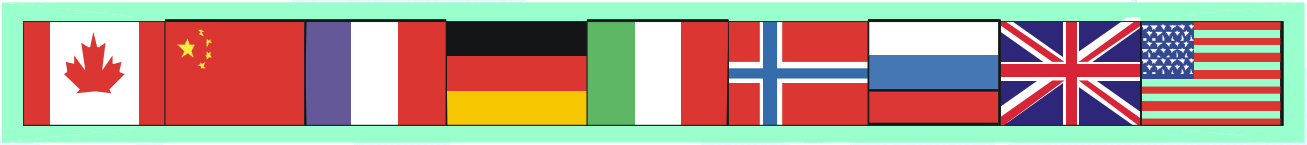
ROOT 2001 Workshop
June 14th, Fermilab

Leif Wilden for the BABAR Collaboration
TU Dresden, Institut für Kern− und Teilchenphysik

Topics

- The BABAR Experiment
- Motivations for using ROOT
- Data Storage with ROOT I/O
- Data Import and −Export
- Interactive Physics Analysis
- Histograming and Fitting
- Outlook

The BABAR Collaboration:



China [1/6]

Inst. of High Energy Physics, Beijing

Germany [3/21]

U Rostock
Ruhr U Bochum
Technische U Dresden

Italy [12/89]

INFN, Bari
INFN, Ferrara
Lab. Nazionali di Frascati dell' INFN
INFN, Genova
INFN, Milano
INFN, Napoli
INFN, Padova
INFN, Pavia
INF, Pisa
INFNN, Roma and U "La Sapienza"
INFN, Torino
INFN, Trieste

Canada [4/16]

U of British Columbia
McGill U
U de Montréal
U of Victoria

Norway [1/3]

U of Bergen

Russia [1/13]

Budker Institute, Novosibirsk

France [5/50]

LAPP, Annecy
LAL Orsay
LPNHE des Universités Paris 6/7
Ecole Polytechnique
CEA, DAPNIA, CE−Saclay

Great Britain [10/80]

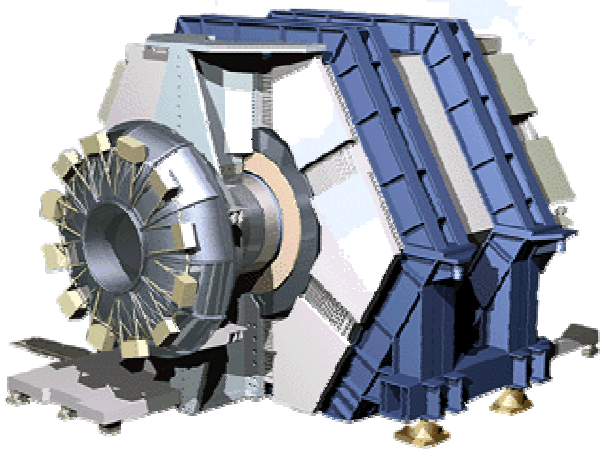
U of Birmingham
U of Bristol
Brunel University
U of Edinburgh
U of Liverpool
Imperial College
Queen Mary & Westfield College
Royal Holloway, University of London
U of Manchester
Rutherford Appleton Laboratory

USA [35/276]

California Institute of Technology
UC, Irvine
UC, Los Angeles
UC, San Diego
UC, Santa Barbara
UC, Santa Cruz
U of Cincinnati
U of Colorado
Colorado State
Florida A&M
U of Iowa
Iowa State U
LBNL
LLNL
U of Louisville
U of Maryland
U of Massachusetts Amherst
MIT
U of Mississippi
Mount Holyoke College
Northern Kentucky U
U of Notre Dame
ORNL/Y−12
U of Oregon
U of Pennsylvania
Prairie View A&M
Princeton
SLAC
U of South Carolina
Stanford U
U of Tennessee
U of Texas at Dallas
Vanderbilt
U of Wisconsin
Yale

The BABAR Experiment

- Observe and measure CP violation in B meson system
- First collisions at May 26,1999
  Recorded 30 fb^{-1} since then
- Asymmetric machine:
  - E_{e-}= 9 GeV; E_{e+}= 3 GeV; E_{cms}=10.58 GeV
    Y(4S) Resonance
- Prompt reconstruction (data immediately available for physics analysis)
- Total integrated luminosity goals:
  - end 2001: 60 fb^{-1}
  - mid 2002: 95 fb^{-1}
- High demands on software and especially on the database

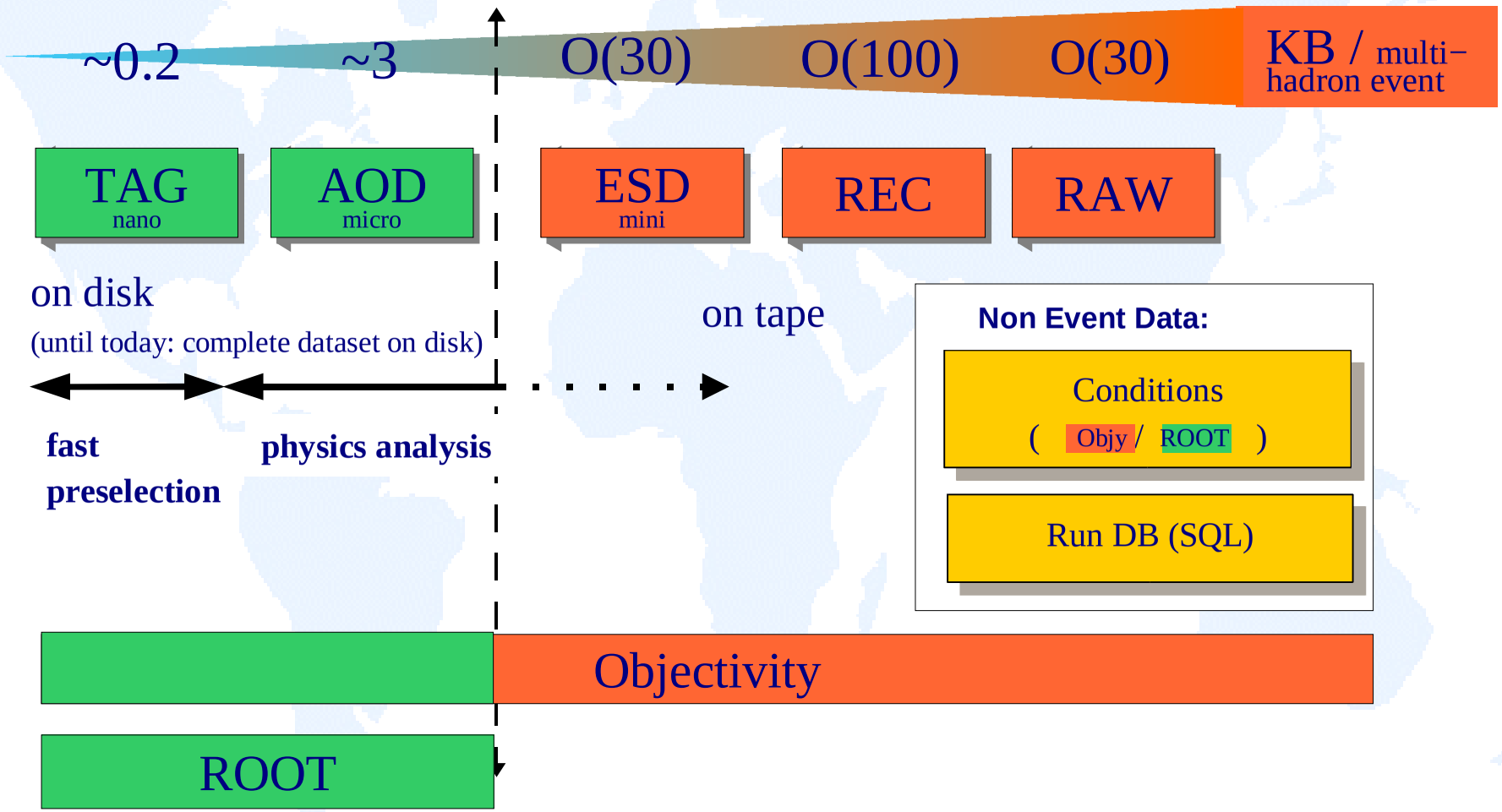


Motivation for using ROOT

- BABAR started off with Objectivity as the only database.
- Initially Objectivity could not meet all requirements:
  - The number of parallel jobs accessing the physics analysis data needed to be restricted. Jobs began to queue up.
  - Data export to remote institutions was difficult / impractical
  - Users began to express their frustrations.
- The problems are being addressed and are expected to be resolved in year 2002. Until then:
- An alternative event store ("KANGAROO" = **k**ind **an**d **g**entle **a**nalysis with **Roo**t) based on ROOT I/O has been developed in a very short time and has been in operation w/o major problems until today.

- Also many collaborators find the ROOT/cint concept and ROOT environment in general very useful. ROOT has made it's way into BABAR software:
  - today ~30 (out of ~600) packages contain ROOT specific code

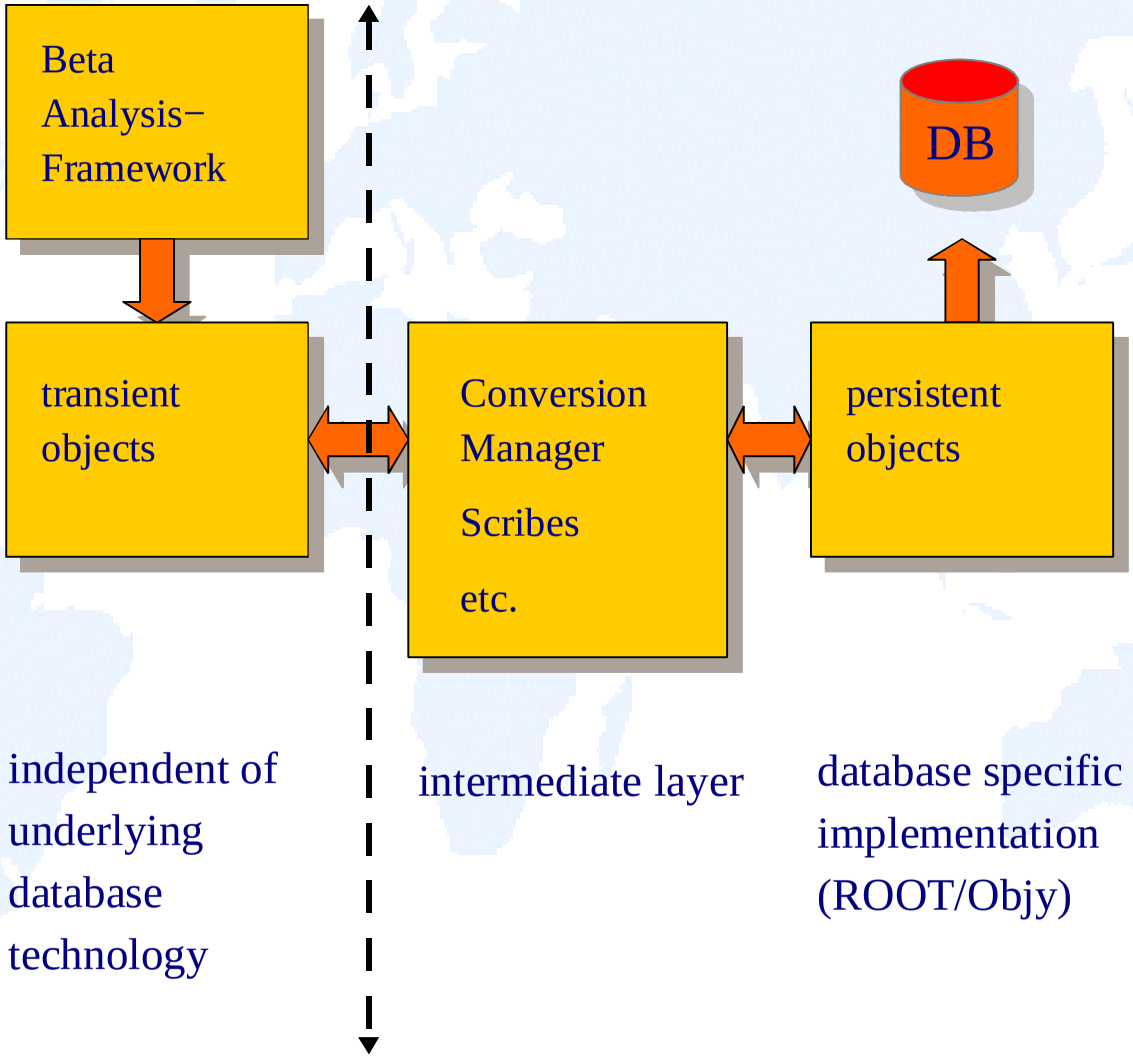
Organization of Databases



Interface to Databases

Physics tools:
- combinatorics
- vertex fits
- particle id
- tagging ...

Physics objects:
- tracks
- clusters
- hits
- ...



DB Interface Example

TRANSIENT:

```
class BtaCandidate
{
  ...
  double energy()
  HepVector p3()
  ...
}
```

PERSISTENT (ROOT):

```
class BtaCandidateR : public
  RooEvtObj<BtaCandidate>
{
   // constructor
  BtaCandidateR( BtaCandidate* )
  ...
  BtaCandidate* transient();
  ...
  ClassDef(...)
}
```

RooEvtObj derives from **TObject**

PERSISTENT (Objy):

```
class BtaCandidateP : public
  BdbEvtObj<BtaCandidate>
{
   // constructor
  BtaCandidateP( BtaCandidate* )
  ...
  BtaCandidate* transient();
  ...
}
```

BdbEvtObj derives from **ooObj**

- Objects can be registered with *scribes*. The *scribes framework* performs the reading and writing of objects, resolves pointers dependencies between objects, etc.

Schema Evolution

- extend the transient class and
- create a new persistent class

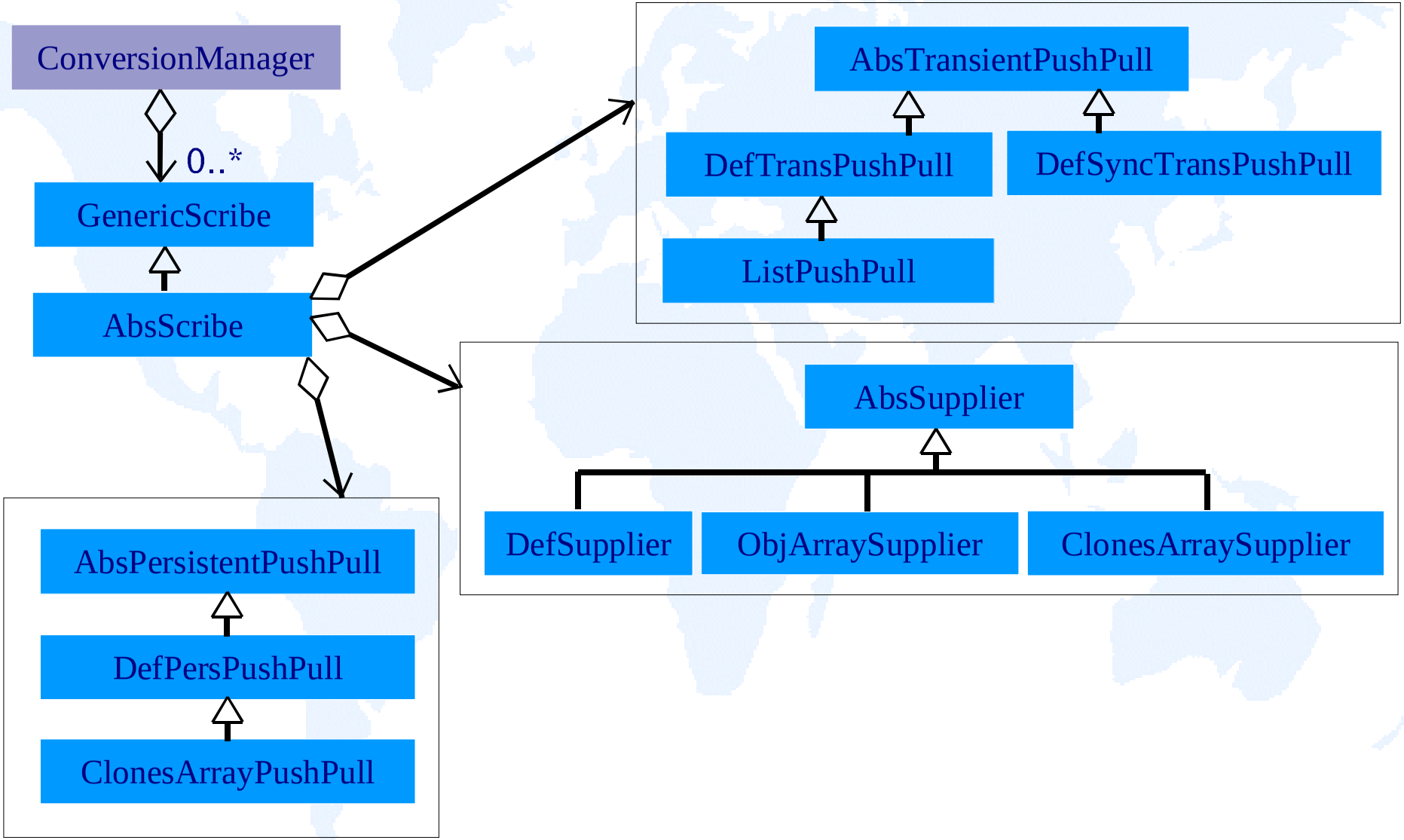
TRANSIENT:

```
class BtaCandidate
{
  ...
  double newVariable()
  ...
}
```

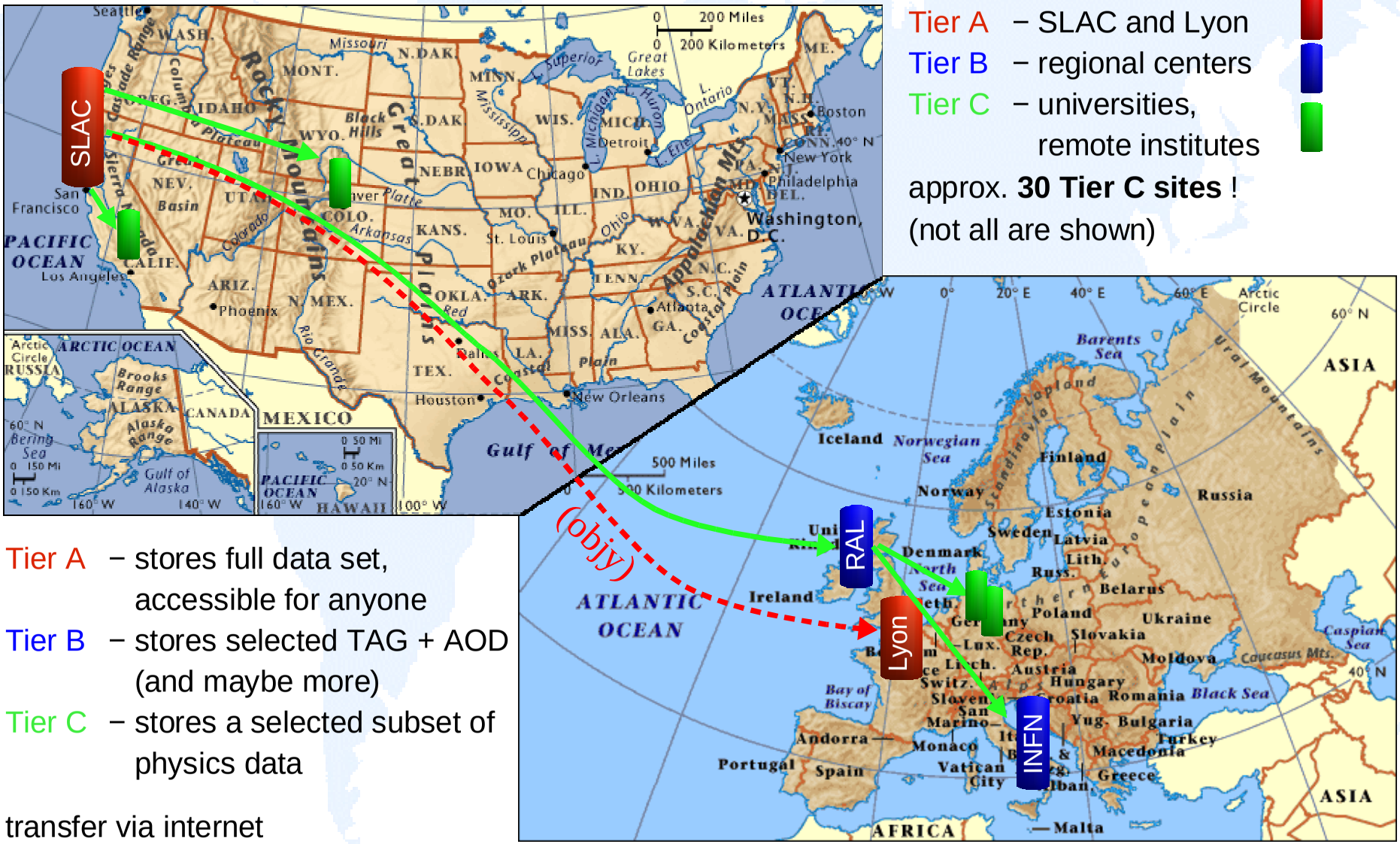
NEW PERSISTENT CLASS:

```
class BtaCandidateR_001 : public
  RooEvtObj<BtaCandidate>
{
   // constructor
  BtaCandidateR_001( BtaCandidate* )
  ...
  BtaCandidate* transient();
  ...
  Double_t fNewVariable;
  ...
  ClassDef(...)
}
```

Scribes – UML Diagram (reduced)



Data Distribution (ROOT files)



Tier A – SLAC and Lyon

Tier B – regional centers

Tier C – universities, remote institutes

approx. **30 Tier C sites** !

(not all are shown)

Tier A – stores full data set, accessible for anyone

Tier B – stores selected TAG + AOD (and maybe more)

Tier C – stores a selected subset of physics data

transfer via internet

Organization of ROOT Event Store

- Situation of year 2000:
  - Events are organized in 20 overlapping **streams** according to
    - physics criteria
    - import wishes of remote institutions
    - minimize overlaps
  - Stream = set of physics channels, e.g. events with a D*
  - One file per run and stream.
  - Organized in a complex NFS directory tree scattered over several NFS file servers (currently 4 TB, growing soon)
  - The directory structure contains run numbers, stream number, software release number etc.
- Handling of so many files requires sophisticated data distribution tools !

Data Distribution Tools

- ROOT file are produced centrally from Objectivity collections (skim production)
- **Tier A** Site: A sql **database** knows about all files:
  run number, reconstruction release, version of stream definition, etc.
- **Tier B/C** Site: (My)**SQL database** knows status of all existing streams:
  Status: subscribed: yes/no
  imported: yes/no
  current location : disk, tape, both
- Tool set:
  - **`skimSqlMirror:`**
    Mirrors SQL db at remote site
  - **`skimSqlSelect:`**
    subscribe for import
  - **`skimDelete:`**
    remove old, superceded, unused data
  - **`skimBackup:`**
    move file to tape / disk
  - **`skimImport:`**
    efficient data transfer using different tools:
    - **`bbftp`**
    - **`sfcp`**
      (open several connections in parallel: high throughput)
  - or custom **`rsync:`**
    good for transfer of many small files

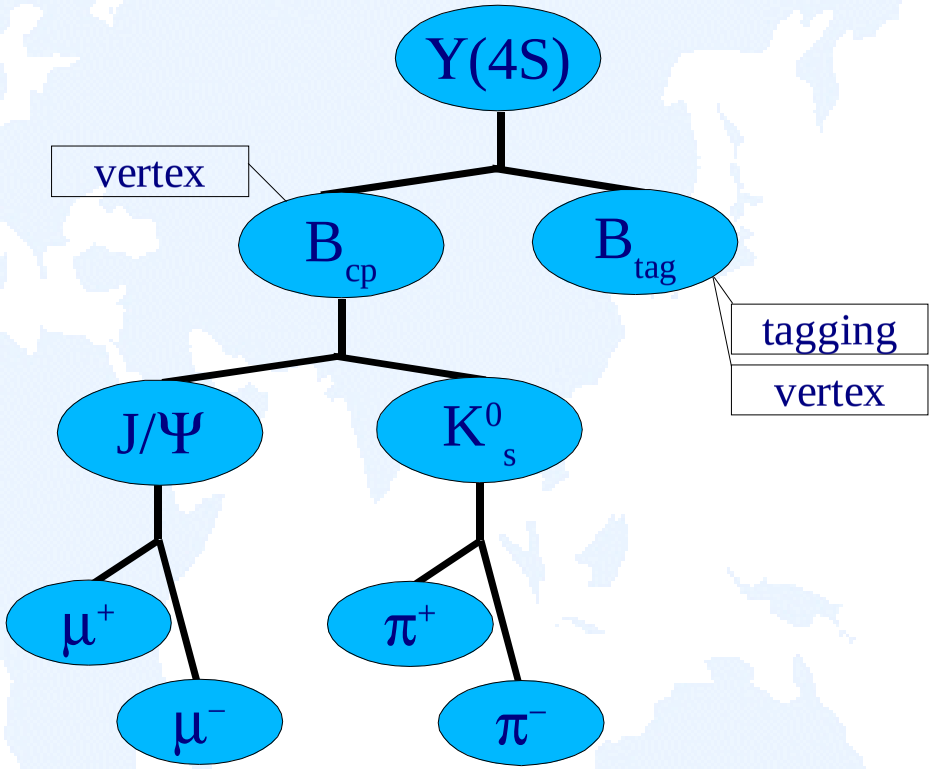
Interactive Beta

- Typical analysis pattern @ BABAR:
  - Access data with BABAR framework
  - Perform combinatorics, fits etc. with BABAR/Beta framework (run jobs in batch)
  - Write results on event basis into an n−tuple file
  - Off line analysis based on n−tuples (often with hbook/fortran...)
- Idea of Interactive Beta:
  - skip the classical n−tuple (rather persist user defined objects)
  - perform everything (interactively) within BABAR/Beta framework
  - encourage users to use our sophisticated C++ tools
- Interactive Beta is build on top of ROOT/CINT

Analysis Model I

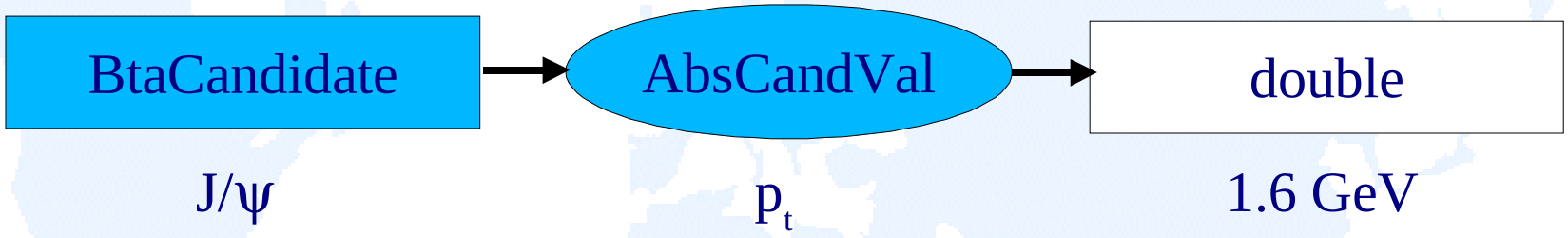
A ROOT file contains:

- tag database
  (bits, ints, and floats)
- AOD database
  (compressed analysis data, tracks, clusters etc.)
- lists of candidates
  (e.g. fully recoed B candidates with genealogy)
- additional information for these candidates
  - selection information
    (e.g. invariant mass, helicities etc.)
  - vertexing data
  - tagging information

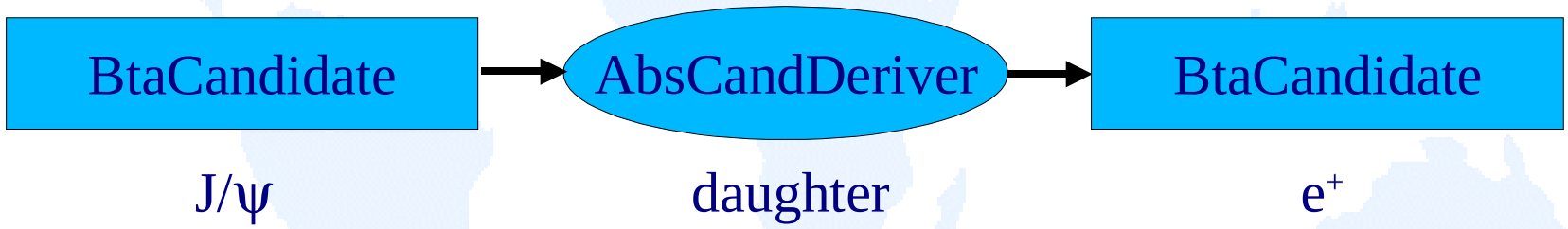


Analysis Model II

Map candidates to numbers...



...or candidates to candidates:



Interactive Beta Example

```
root[1] RooEventManager evt("my.tcl");
This will make an object that will iterate over all the events in the
root files specified in "my.tcl".

root[2] RooCandAccessor a("Output",PdtLund::B0);
This will make an object that will get all the B0's on the list called
"Output" from an event.

root[3] RooKangaCandIter itr(&evt,&a);
This will make a object that will use first two objects you made to
iterate over all the B0's on the list called "Output".

root[4] BtaCandidate* aCand = itr.next();
This will get the first B0 and set aCand to point to it.

Then you can play around with the candidate:
root[5] aCand->print(cout);
root[6] aCand->p3().x()
```

Histograming and Fitting

- BABAR developed the toolkit **RooFitTools** for ...
  - unbinned likelihood fits
  - toy MC studies
  - displaying fit results
- RooFitTools is build on top of ROOT...
  - mostly independent of BABAR software
  - can possibly be contributed to the ROOT project
  - see dedicated talk by Wouter Verkerke !

Conclusion

- A ROOT event store (currently 4 TB, 20^{-1} fb data plus MC) has been in operation far more than 1.5 years without major show stoppers.
- ROOT is widely being used for histograming and fitting within BABAR.
- An interactive analysis framework based on ROOT/cint has been developed. People are now starting to use it.
- Some feedback I got
  - Most people that I asked seemed rather happy with ROOT (−:
  - Interfaces are sometimes obscure (e.g. SetOptFit, SetOptStat, color indices etc.)
  - Documentation of member functions in code is often missing or not clear enough
  - BABAR stores large amounts of data in ROOT files:
    - ROOT data format must always stay backward compatible!