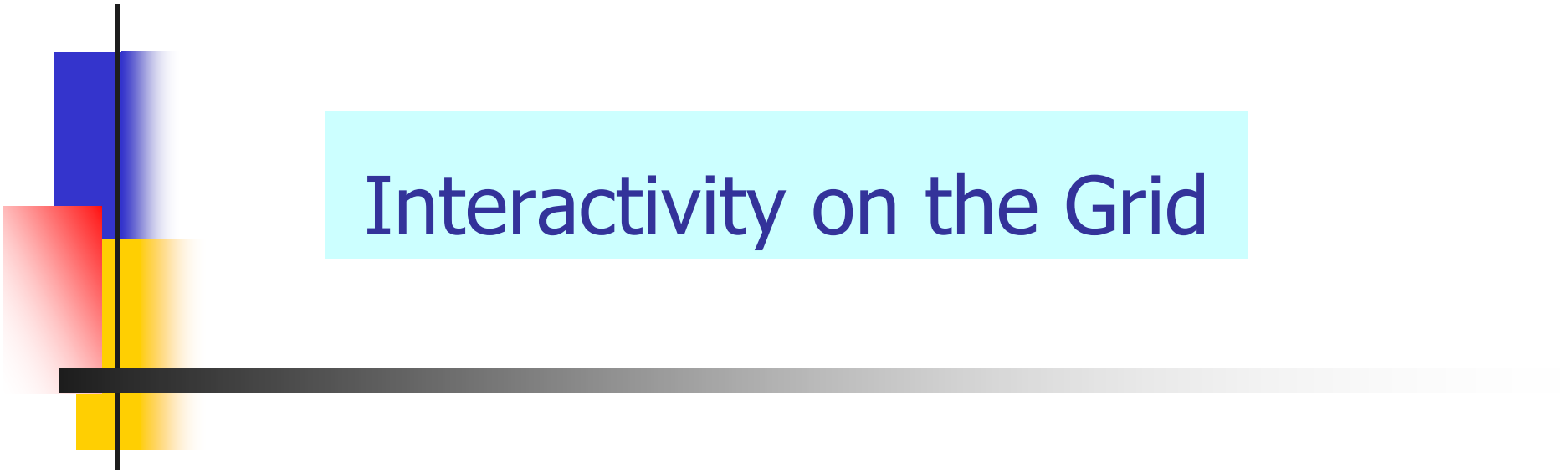


Interactivity on the Grid

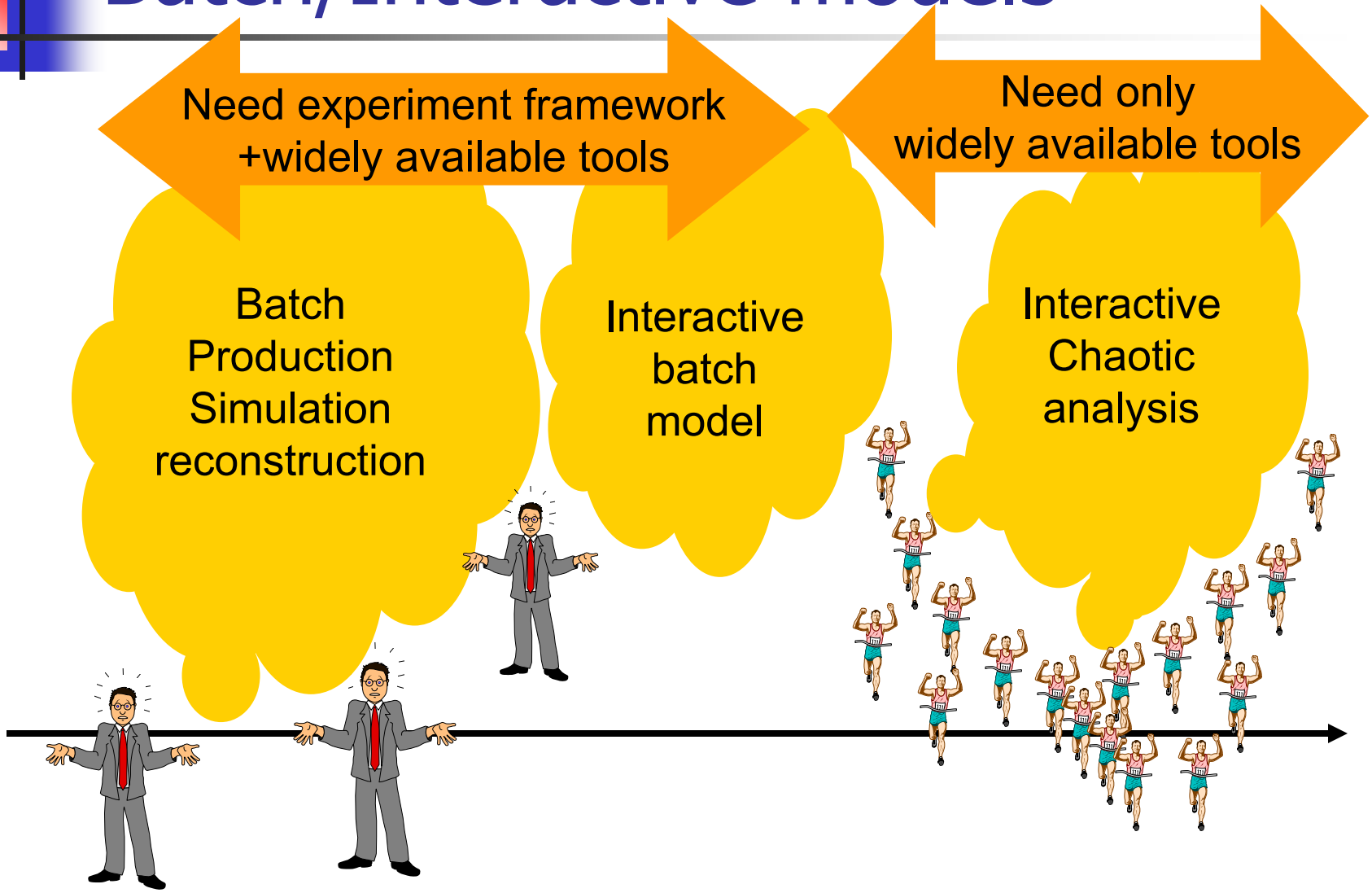
ARDA workshop

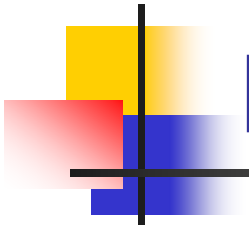
CERN 23 June 2004

René Brun

CERN

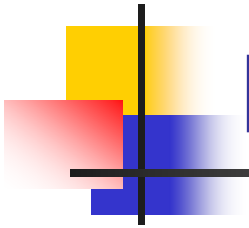
Batch/Interactive models





Batch-mode Local analysis

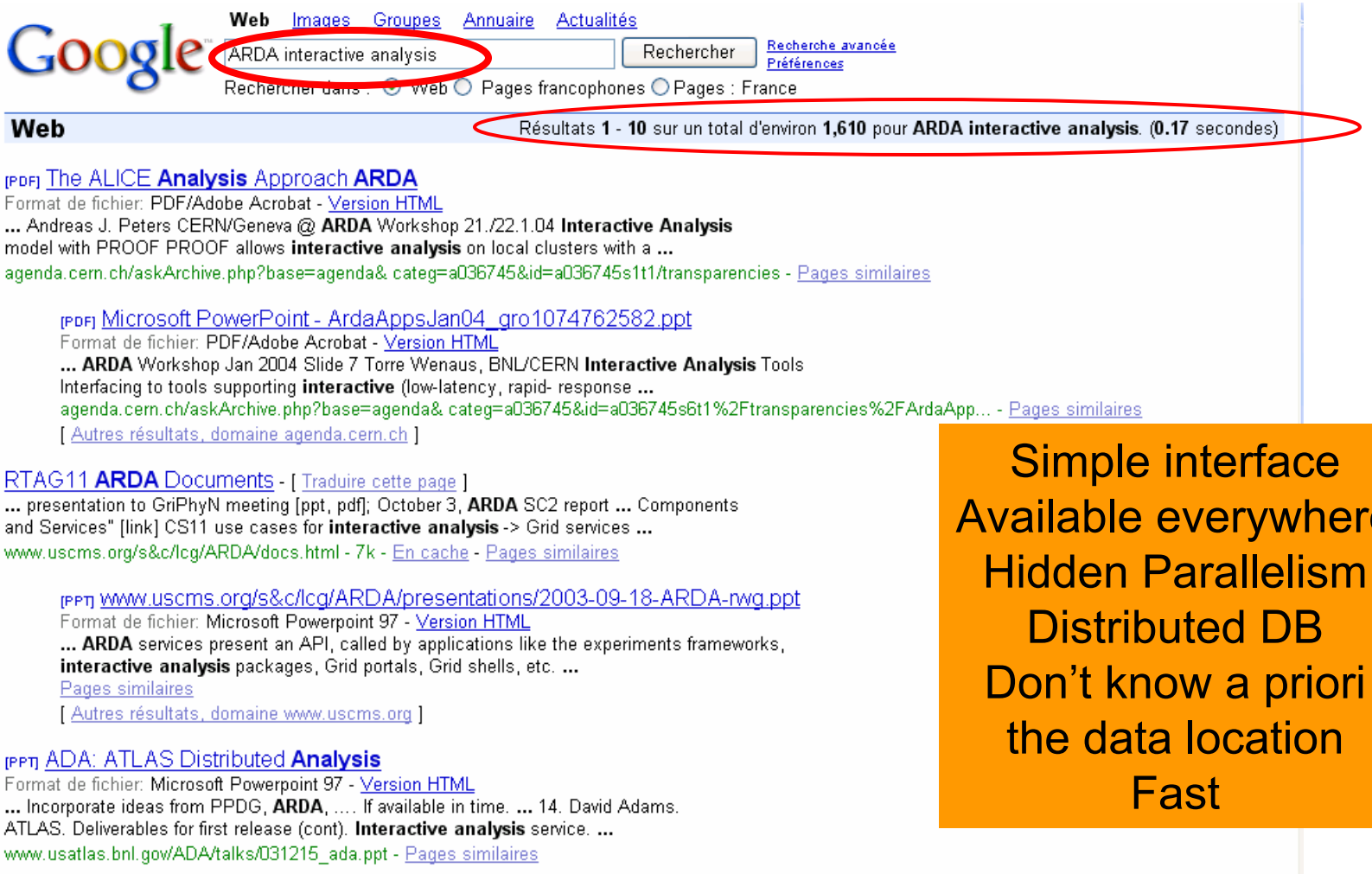
- Conventional model: The user has full control on the event loop.
- The program produces histograms, ntuples or trees.
- The selection is via user private code
- Histograms are then added (tool or in the interactive session)
- ntuples/trees are combined into a chain and analyzed interactively.



Batch Analysis on the GRID

- From a user viewpoint, a simple extrapolation of the local batch analysis.
- In practice, must involve all the GRID machinery: authentication, resource brokers, sandboxes.
- Viewing the current status (histograms) must be possible.
- Advantage: Stateless, can process large data volumes.

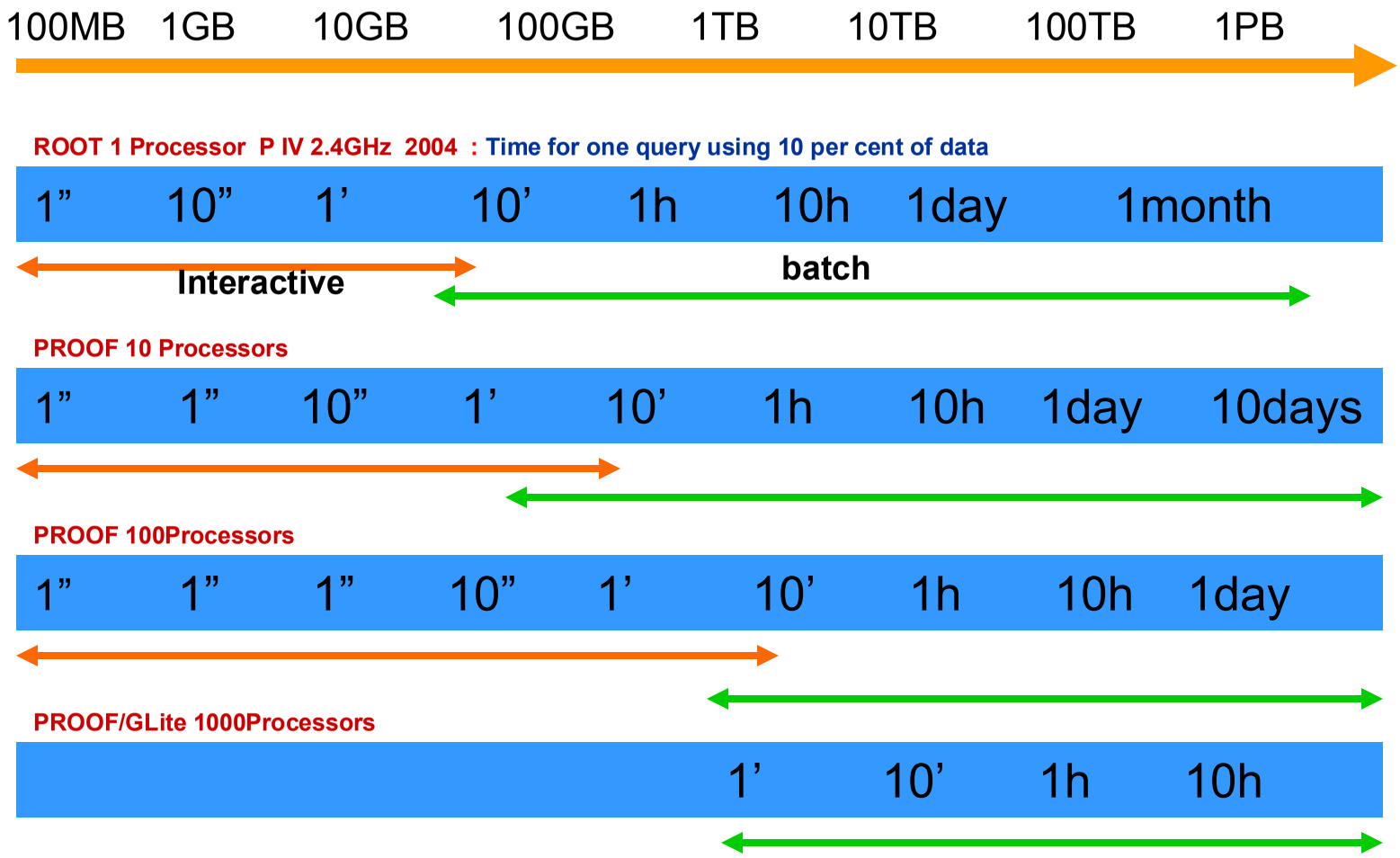
Google: a good model
Make it simple



Simple interface
Available everywhere
Hidden Parallelism
Distributed DB
Don't know a priori
the data location
Fast

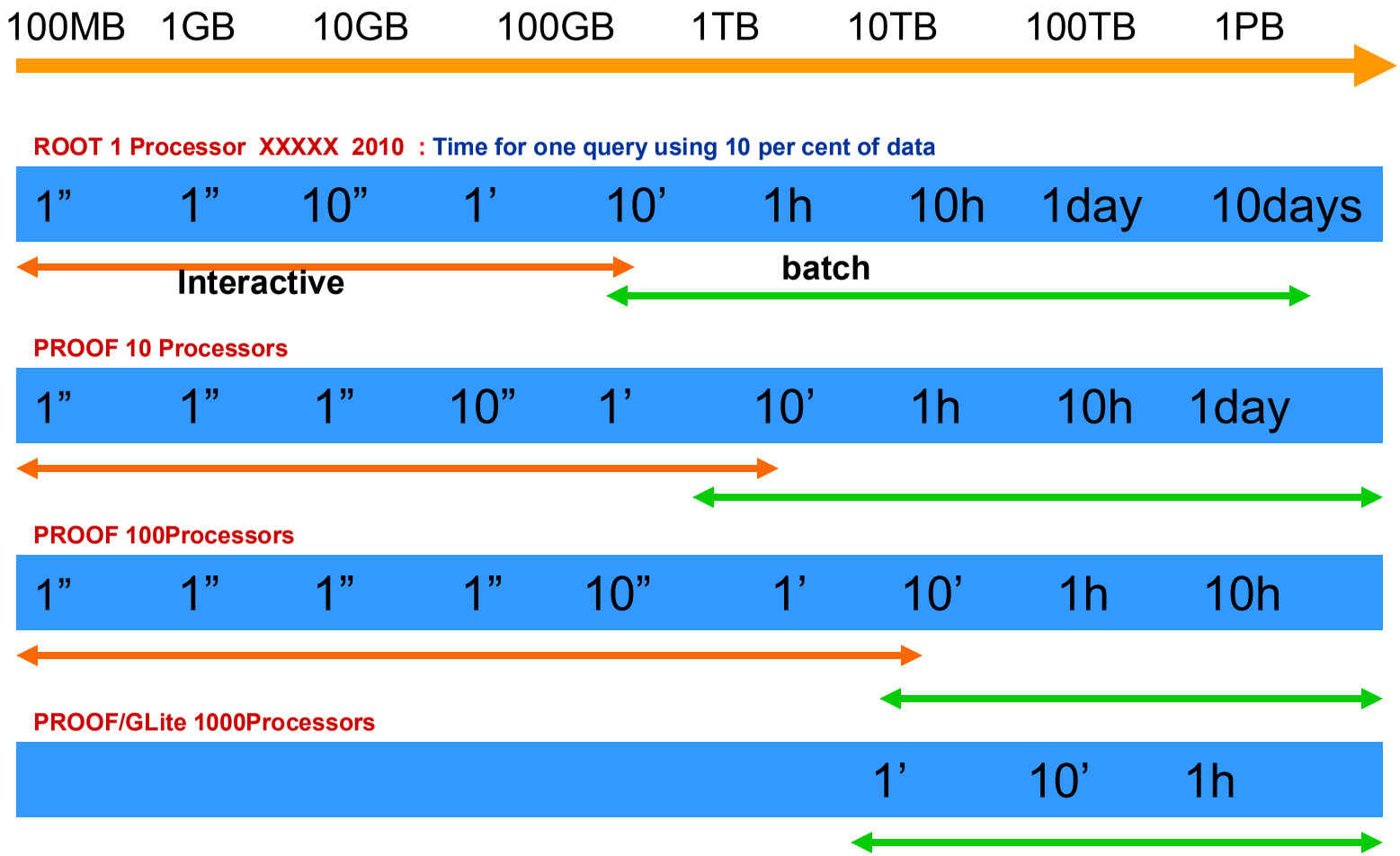
Data Volume & Processing Time

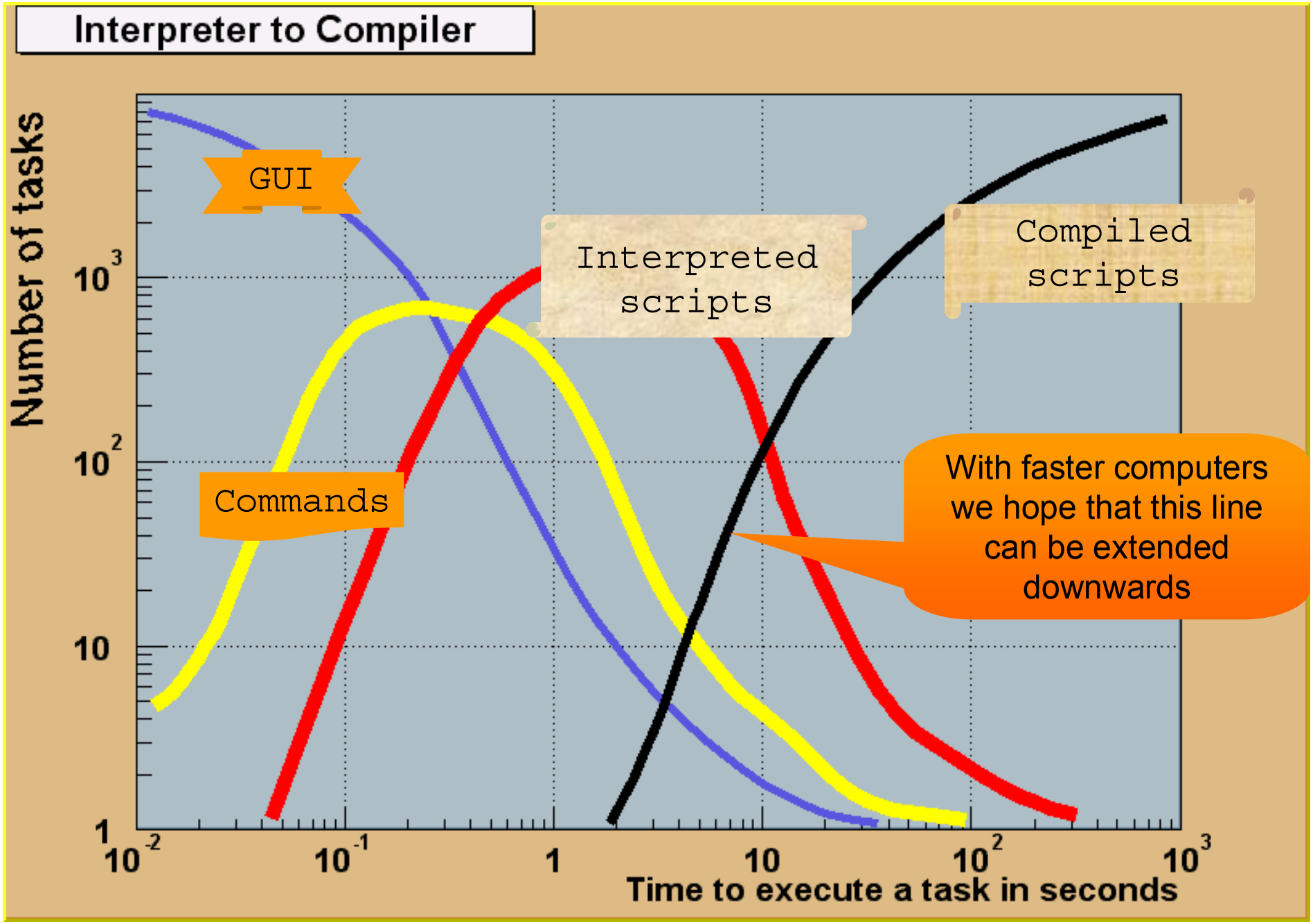
Using technology available in 2004



Data Volume & Processing Time

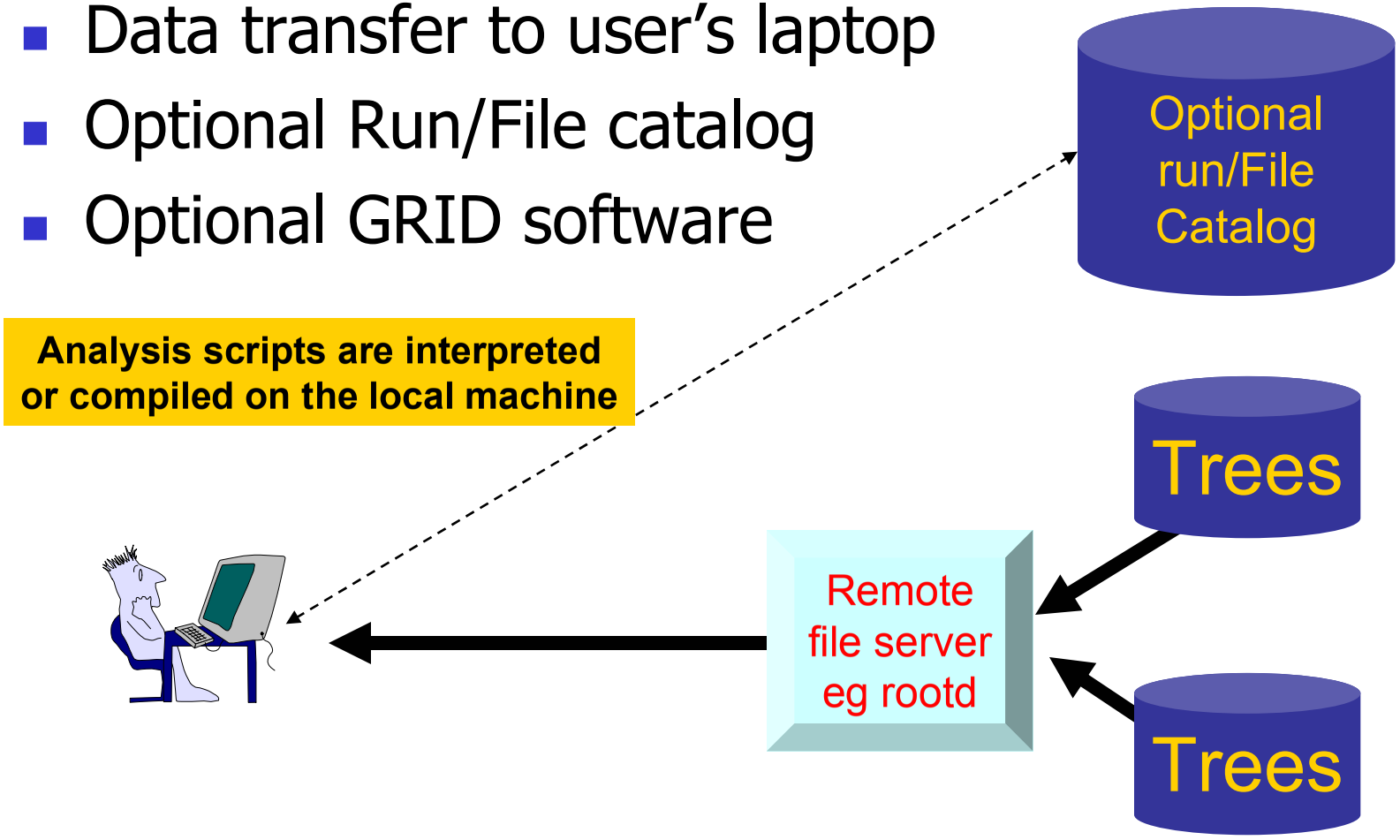
Using technology available in 2010




Interpreter to Compiler
Number of tasks
10^3
10^2
10
1
GUI
Commands
Interpreted scripts
Compiled scripts
With faster computers we hope that this line can be extended downwards
10^{-2}
10^{-1}
1
10
10^2
10^3
Time to execute a task in seconds

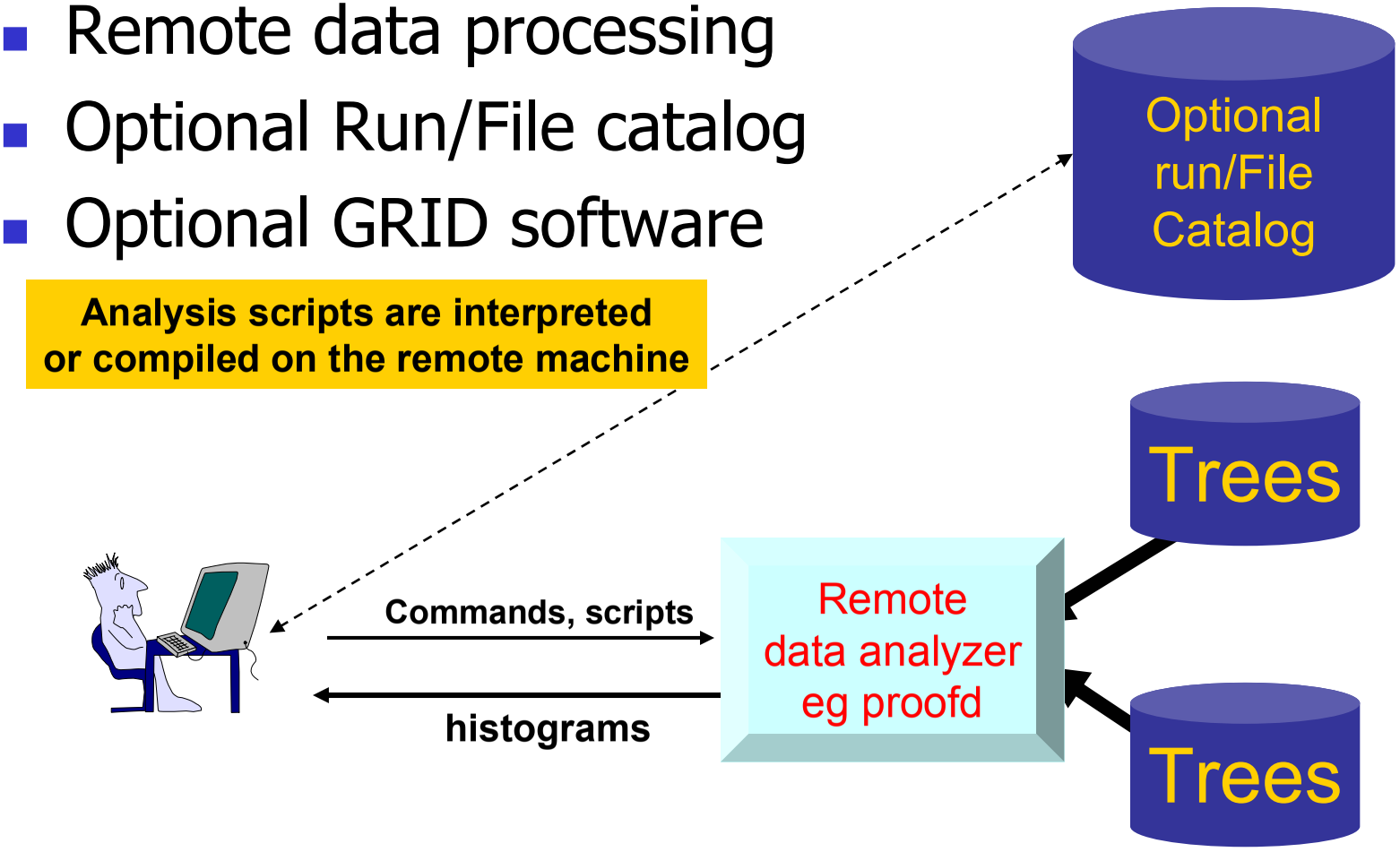
GRID: Interactive Analysis Case 1

- Data transfer to user's laptop
- Optional Run/File catalog
- Optional GRID software



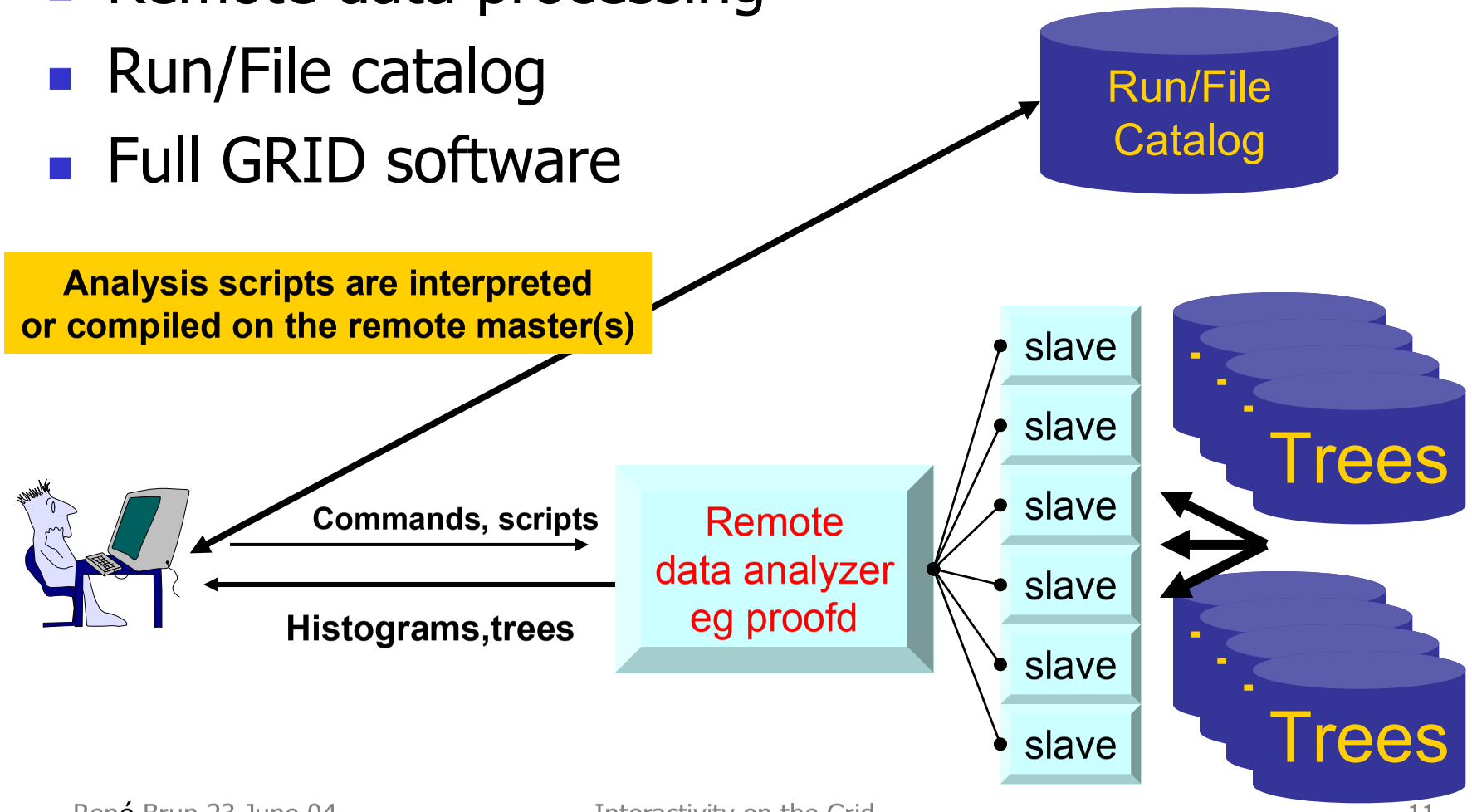
GRID: Interactive Analysis Case 2

- Remote data processing
- Optional Run/File catalog
- Optional GRID software

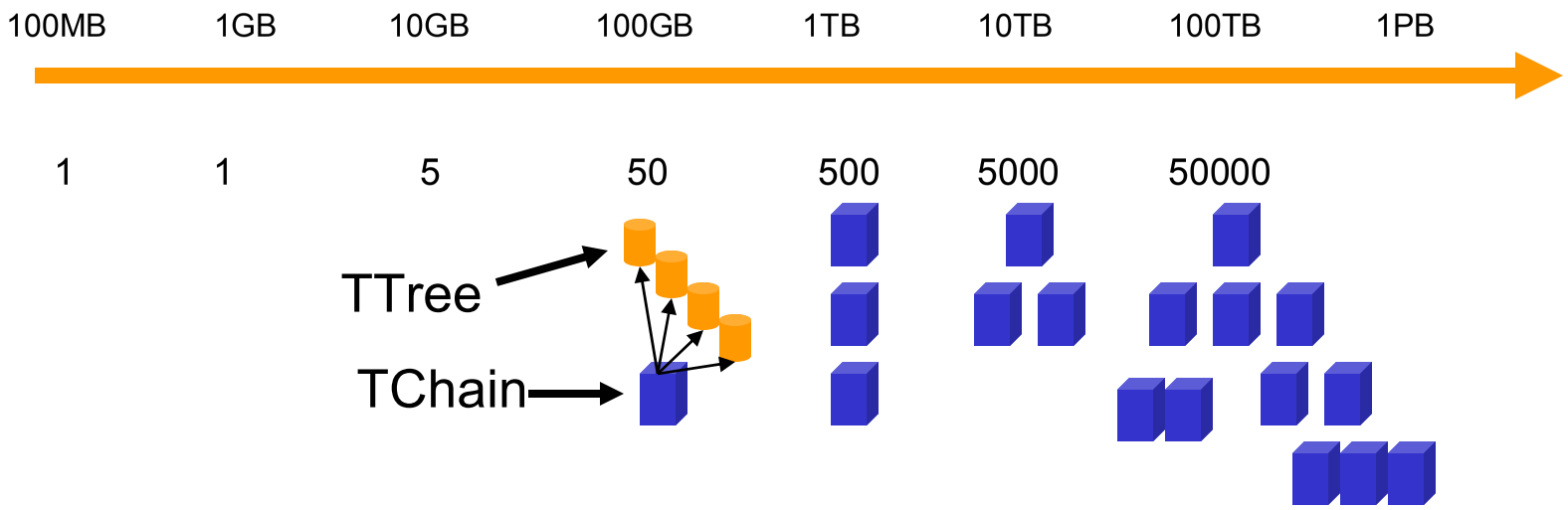


GRID: Interactive Analysis Case 3

- Remote data processing
- Run/File catalog
- Full GRID software



Data Volume & Organisation



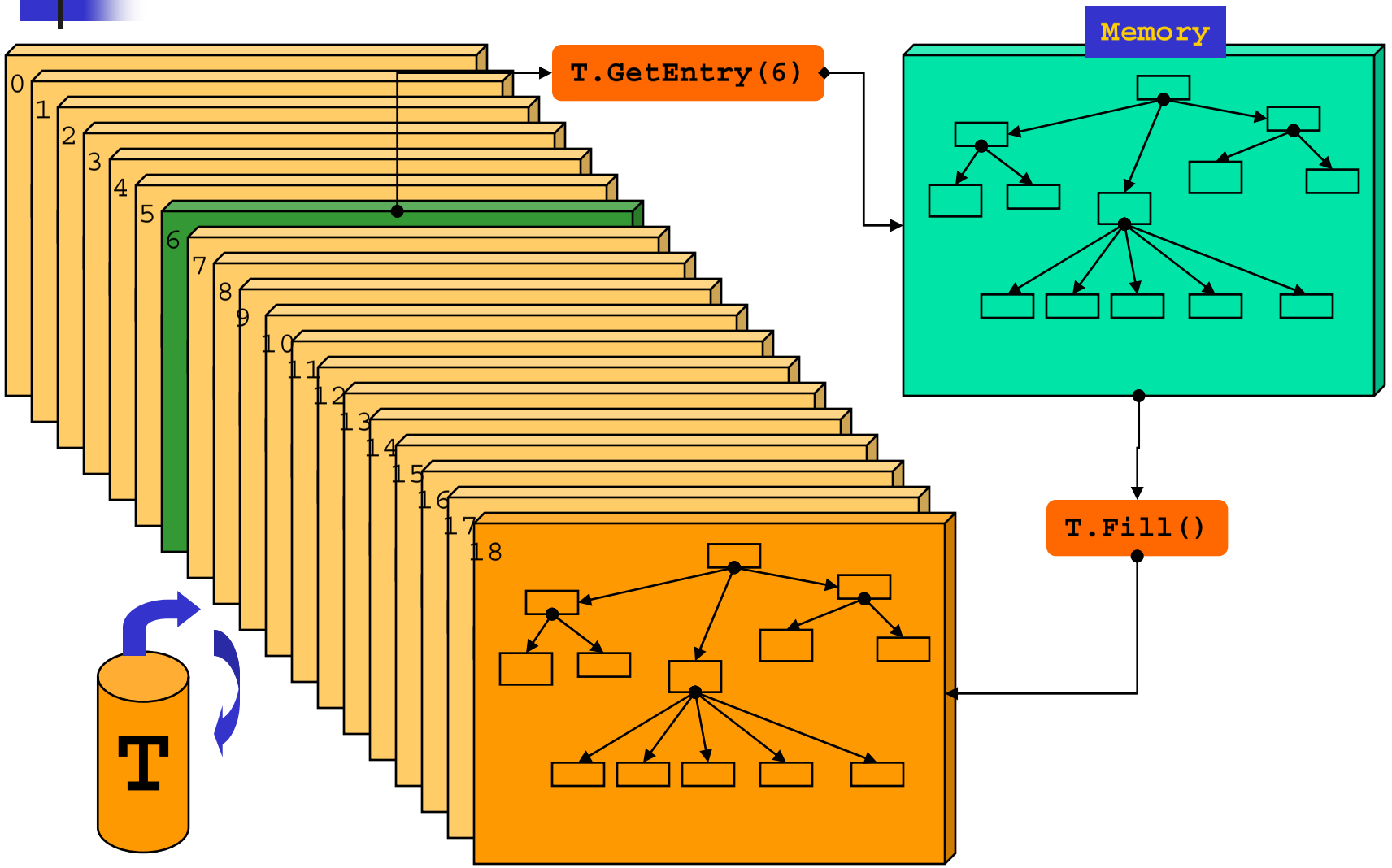
A TFile typically contains 1 TTree (or a few)

A TChain is a collection of TTrees or/and TChains

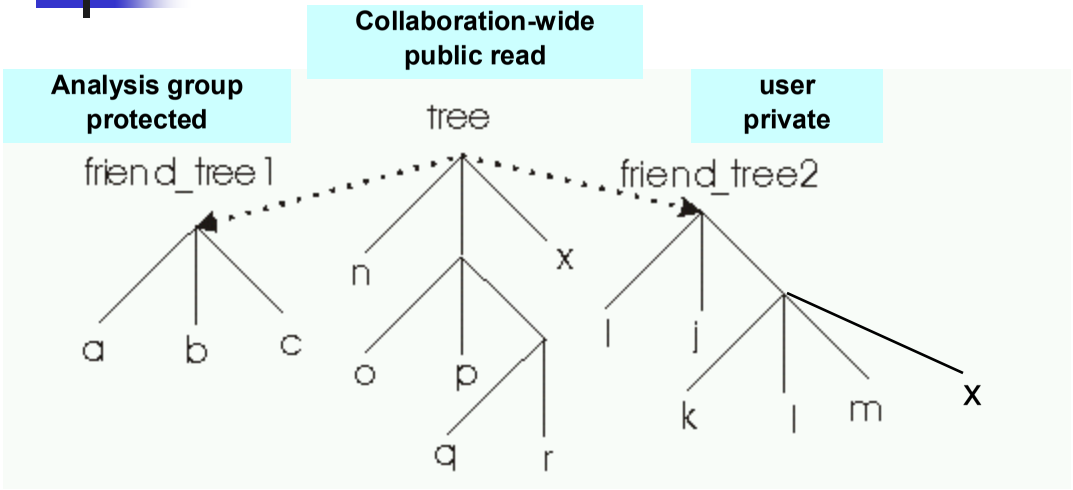
A TChain is typically the result of a query to the file catalogue

Memory <--> Tree

Each Node is a branch in the Tree



Tree Friends



Processing time independent of the number of friends unlike table joins in RDBMS

```
Root > TFile f1("tree1.root");
Root > tree.AddFriend("tree2","tree2.root")
Root > tree.AddFriend("tree3","tree3.root");
Root > tree.Draw("x:a","k<c");
Root > tree.Draw("x:tree2.x","sqrt(p)<b");
```

Chains of Trees

- A TChain is a collection of Trees.
- Same semantics for TChains and TTrees
  - root > .x h1chain.C
  - root > chain.Process("h1analysis.C")

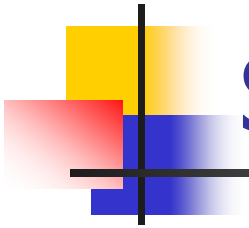
```
{
  //creates a TChain to be used by the h1analysis.C class
  //the symbol H1 must point to a directory where the H1 data sets
  //have been installed

  TChain chain("h42");
    chain.Add("$H1/dstarmb.root");
    chain.Add("$H1/dstarp1a.root");
    chain.Add("$H1/dstarp1b.root");
    chain.Add("$H1/dstarp2.root");
}
```

TSelector – The Algorithms

- Basic ROOT TSelector

```
// Abbreviated version
class TSelector : public TObject {
Protected:
    TList *fInput;
    TList *fOutput;
public
    void Init(TTree*);
    void Begin(TTree*);
    void SlaveBegin(TTree *);
    Bool_t Process(int entry);
    void SlaveTerminate();
    void Terminate();
};
```

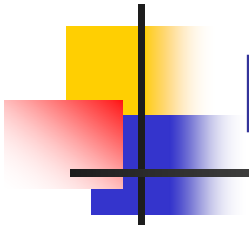
A skeleton selector
Is automatically
generated
By
TTree::MakeSelector



Selectors

- A Selector script can be run
  - In batch
  - Interactive ROOT
  - Interactive ROOT + PROOF
  - Interactive or batch ROOT + PROOF + GLITE
- A Selector script can be
  - Interpreted tree.Process("myselector.C")
  - Or compiled tree.Process("myselector.C++")

Smooth transition between batch and interactive sessions

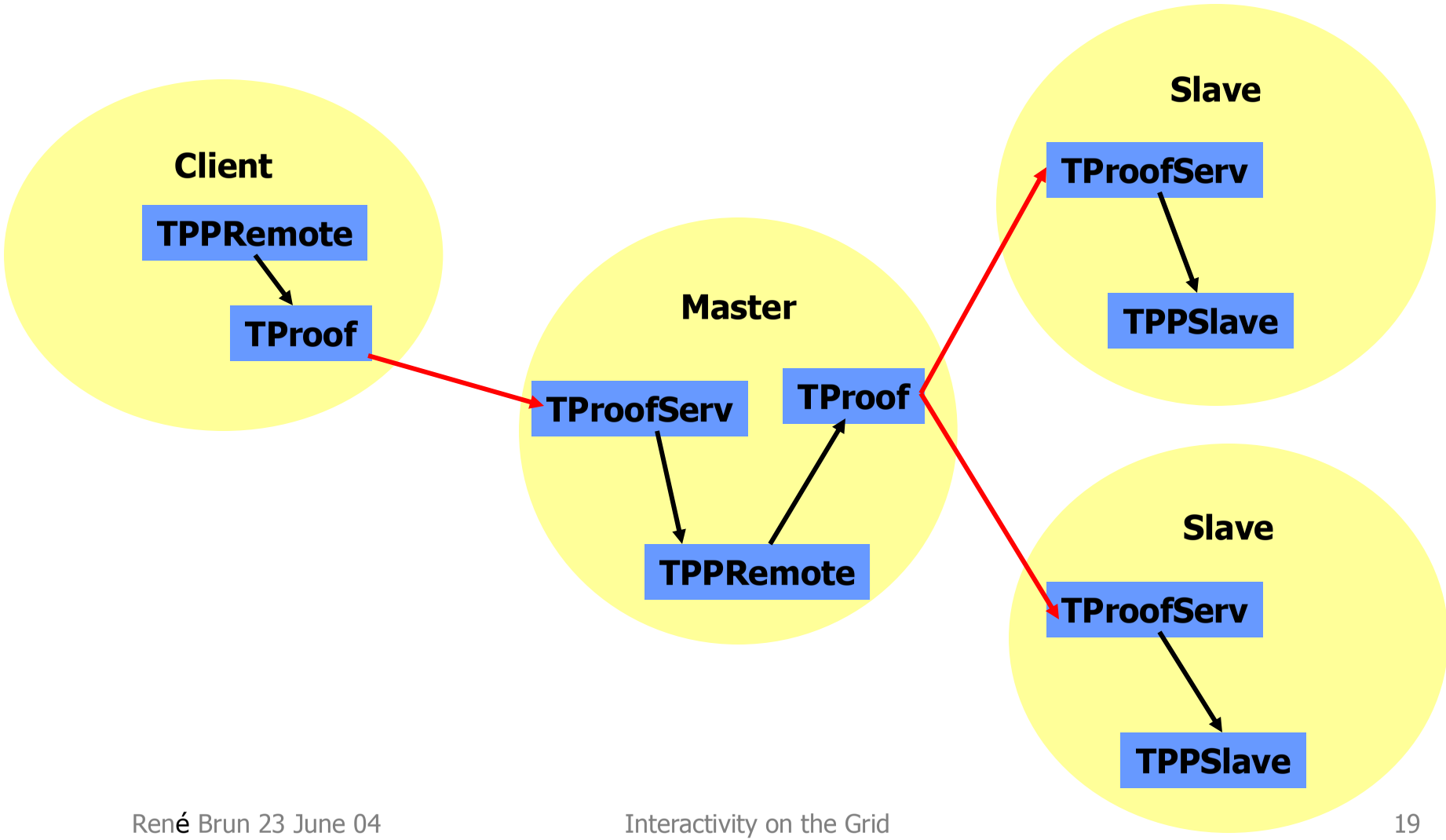


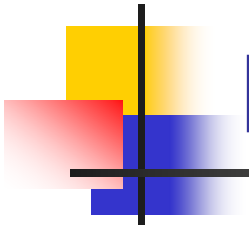
Parallel ROOT Facility

- The PROOF system allows:
  - Parallel analysis of trees in a set of files
  - Parallel analysis of objects in a set of files
  - Parallel execution of scripts

  on clusters of heterogeneous machines
- Its design goals are:
  - Transparency, scalability, adaptability

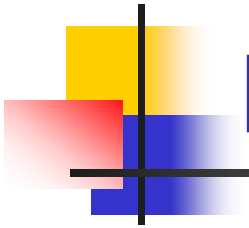
TProofPlayer





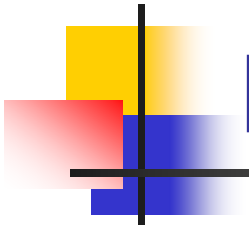
Dynamic Histogram Binning

- Implemented using THLimitsFinder class
- Avoid synchronization between slaves
- Keep score-board in master
  - Use histogram name as key
  - First slave posts limits
  - Master determines best bin size
  - Others use these values



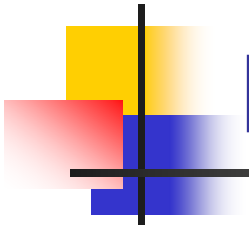
Merge API

- Collect output lists in master server
- Objects are identified by name
- Combine partial results
- Member function: Merge(TCollection *)
  - Executed via CINT, no inheritance required
- Standard implementation for histograms and (in memory) trees
- Otherwise return the individual objects



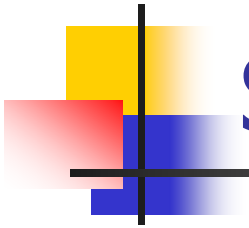
PROOF Authentication

- PROOF supports secure and un-secure authentication mechanisms
- Same as for rootd
  - UsrPwd
  - SRP
  - Kerberos
  - Globus
  - SSH
  - UidGid



PROOF Error Handling

- Handling death of PROOF servers
  - Death of master
    - Fatal, need to reconnect
  - Death of slave
    - Master can resubmit packets of death slave to other slaves
- Handling of ctrl-c
  - OOB message is send to master, and forwarded to slaves, causing soft/hard interrupt



Sandbox – The Environment

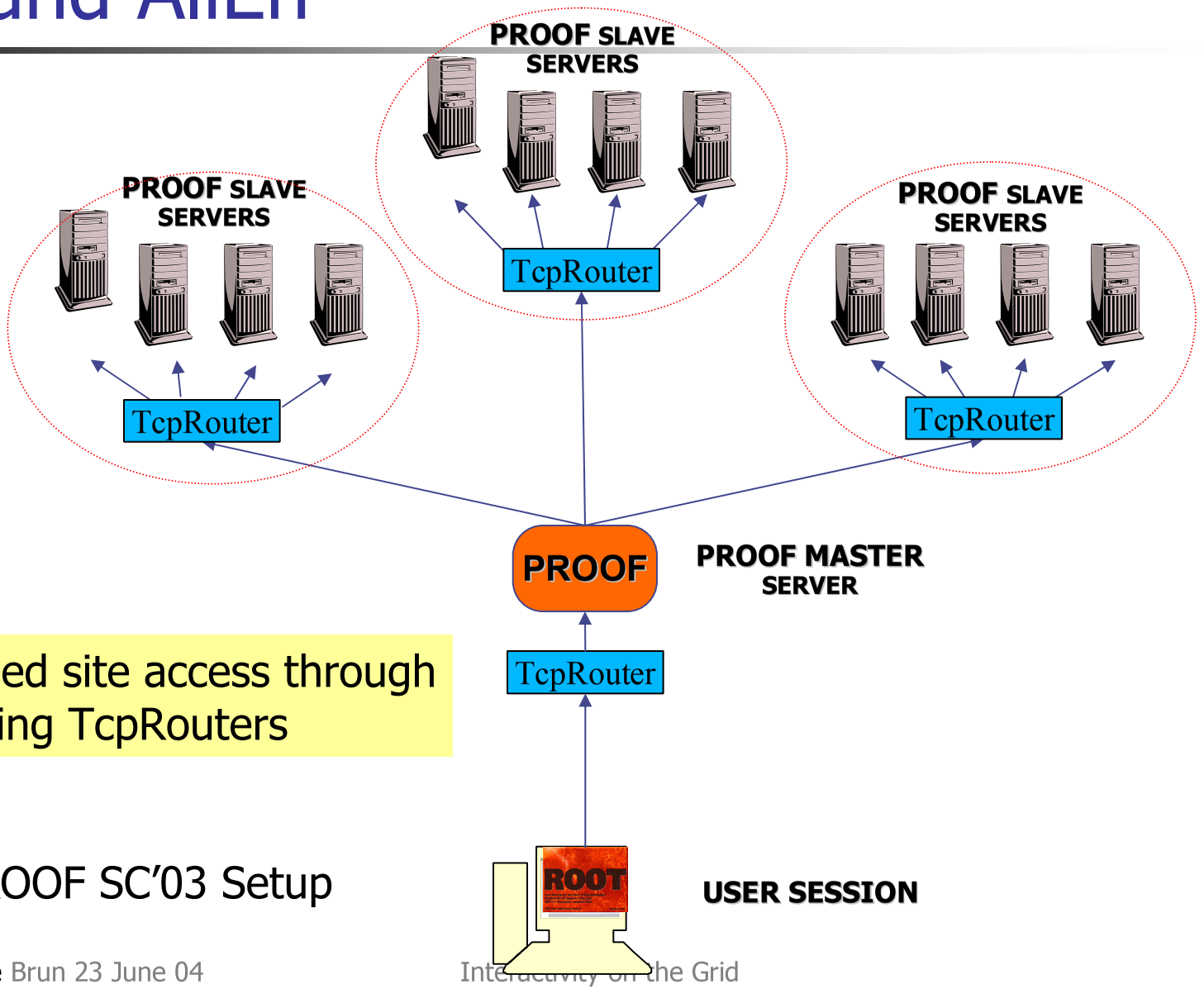
- Each slave runs in its own sandbox
  - Identical, but independent
- Multiple file spaces in a PROOF setup
  - Shared via NFS, AFS, shared nothing
- File transfers are minimized
  - Cache
  - Packages

Running PROOF Using AliEn

```
TGrid *alien = TGrid::Connect("alien");

TGridResult *res;
res = alien->Query("lfn:///alice/simulation/2001-04/V0.6*.root");

TDSet *treeset = new TDSet("TTree", "AOD");
treeset->Add(res);

gROOT->Proof(res);   // use files in result set to find remote nodes
treeset->Process("myselector.C");

// plot/save objects produced in myselector.C
. . .
```

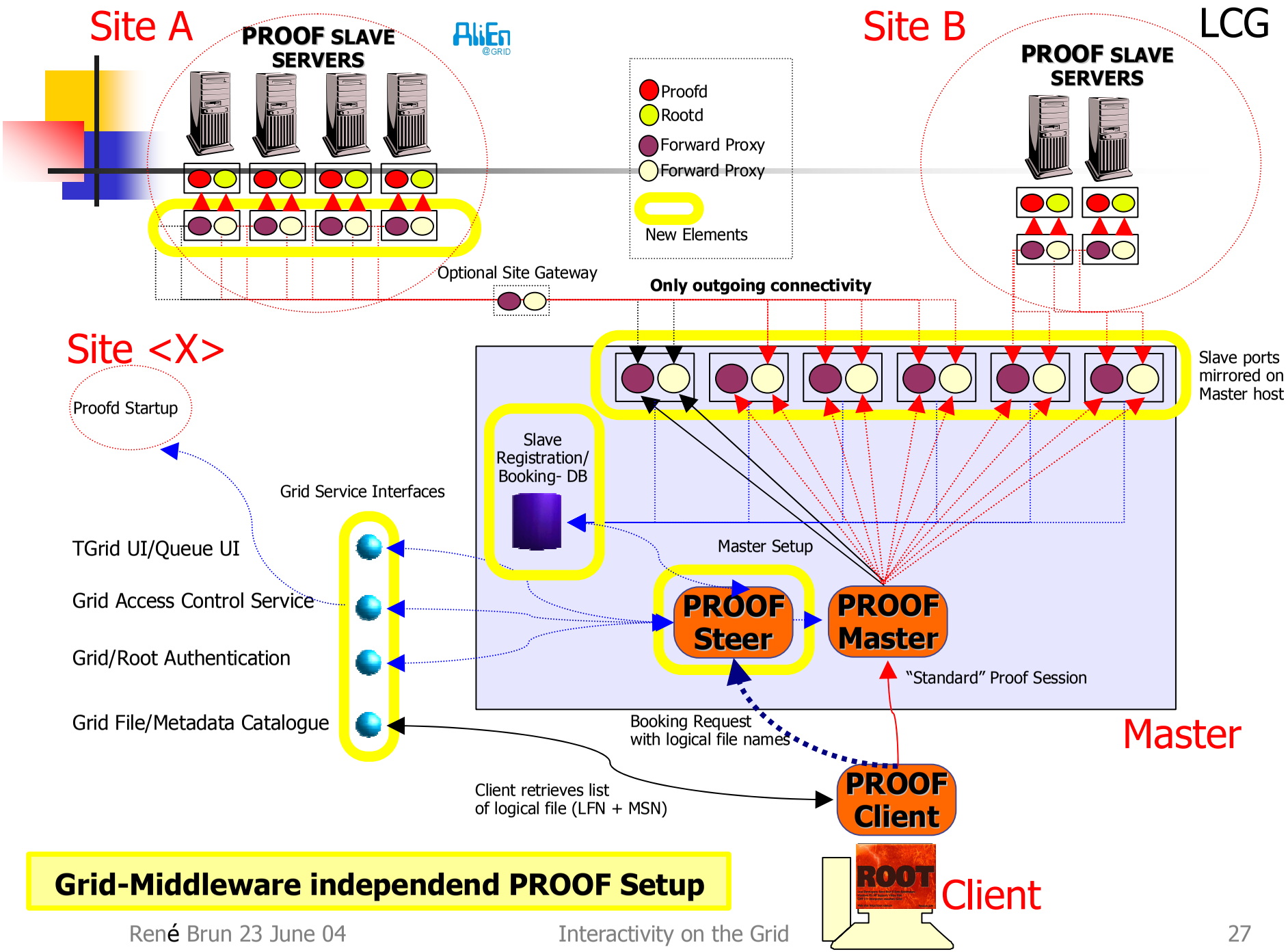
This scenario was demonstrated by ALICE at SC'03 in Phoenix

Interactive Analysis with PROOF and AliEn



Guaranteed site access through Multiplexing TcpRouters

AliEn/PROOF SC'03 Setup



Grid-Middleware independend PROOF Setup

Interactive Demo