

PROOF Update

Maarten Ballintijn / MIT

ROOT Workshop 2004

Outline



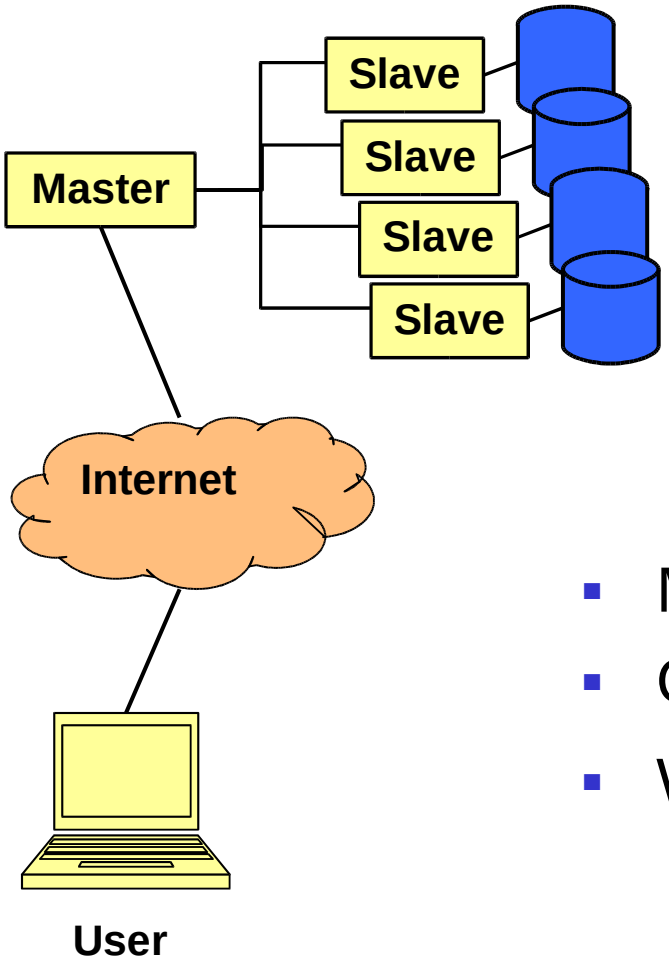
- PROOF revisited
- Improved TSelector
- Condor and “Computing on Demand”
- PROOF and COD
- Plans and ideas
- Conclusions

PROOF – Parallel ROOT Facility



- A system for the **interactive** analysis of **very large sets** of ROOT data files on a **cluster of computers**
- Speed up the query processing by employing inherent parallelism in event data
- In the GRID context, this model will be extended from a local cluster to a wide area “virtual cluster” or a cluster of clusters
- The main design goals are:
  - Transparency, scalability, adaptability
- Collaboration between core ROOT group at CERN and MIT Heavy Ion Group

PROOF, continued



- Multi Tier architecture
- Optimize for Data Locality
- WAN Ready and GRID compatible

PROOF - Architecture



- Data Access Strategies
  - Local data first, also **rootd, rfio**, SAN/NAS
- Transparency
  - Input objects copied from client
  - Output objects merged, returned to client
- Scalability and Adaptability
  - Vary packet size (specific workload, slave performance, dynamic load)
  - Heterogeneous Servers
- Migrate to multi site configurations

PROOF and Data Grids



- Many services are a good fit
  - Authentication
  - File Catalog, replication services
  - Resource brokers
  - Job schedulers
  - Monitoring

Adaptability: Use abstract interfaces

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

TDSet – The Data



- Specify a collection of TTrees or files with objects

```
root[0] TDSet *d = new TDSet(“TTree”, “tracks”, “/”);
root[1] d->Add(“root://rcrs4001/a.root”);
…
root[10] d->Print(“a”);
root[11] d->Process(“mySelector.C”, nentries, first);
```

- Returned by DB or File Catalog query etc.
- Use logical filenames (“lfn:…”)

TSelector – The Algorithms

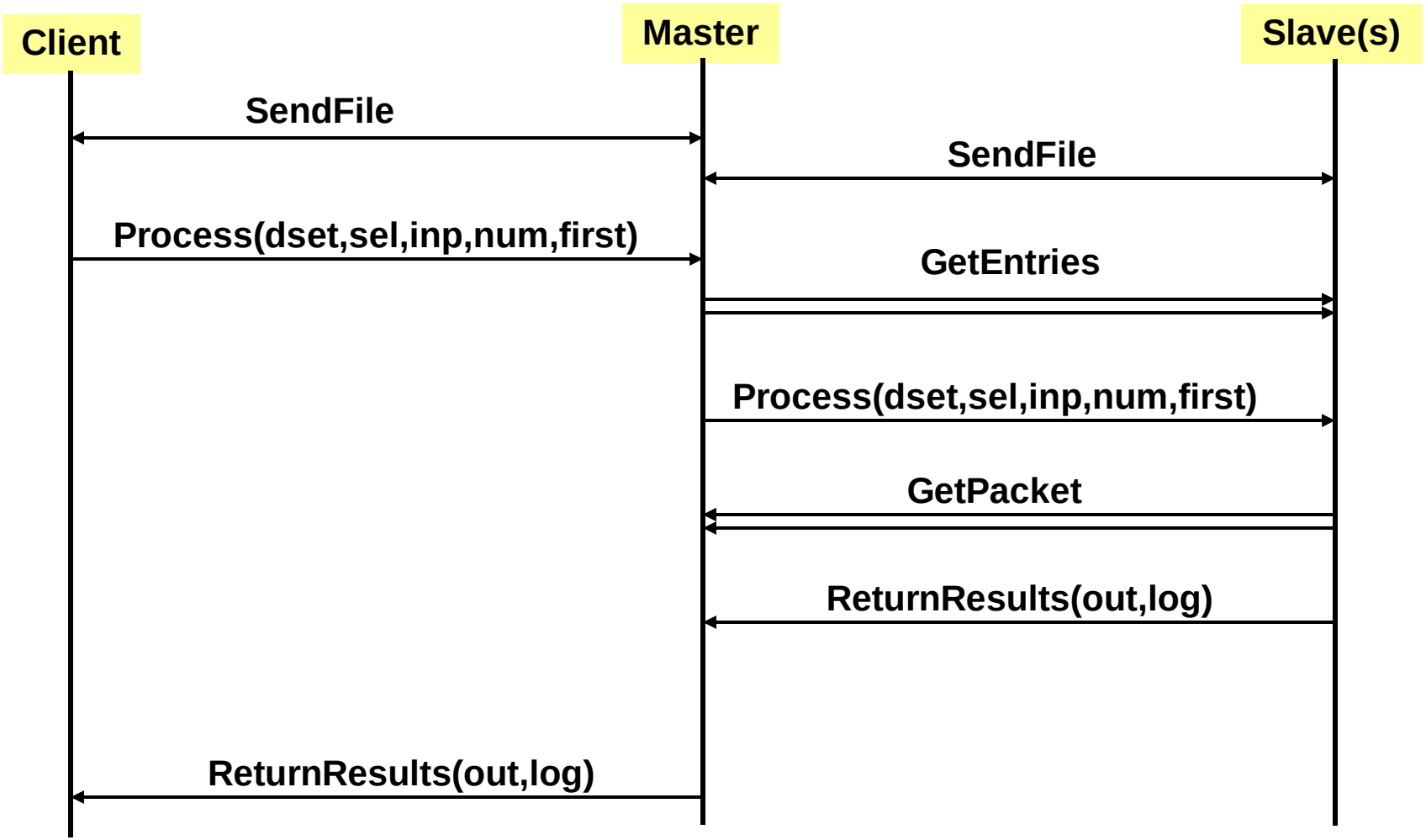


- Created via TTree::MakeSelector()

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

Simplified message flow





Outline



- PROOF revisited
- **Improved TSelector**
- Condor and “Computing on Demand”
- PROOF and COD
- Plans and ideas
- Conclusions

Improved TSelector



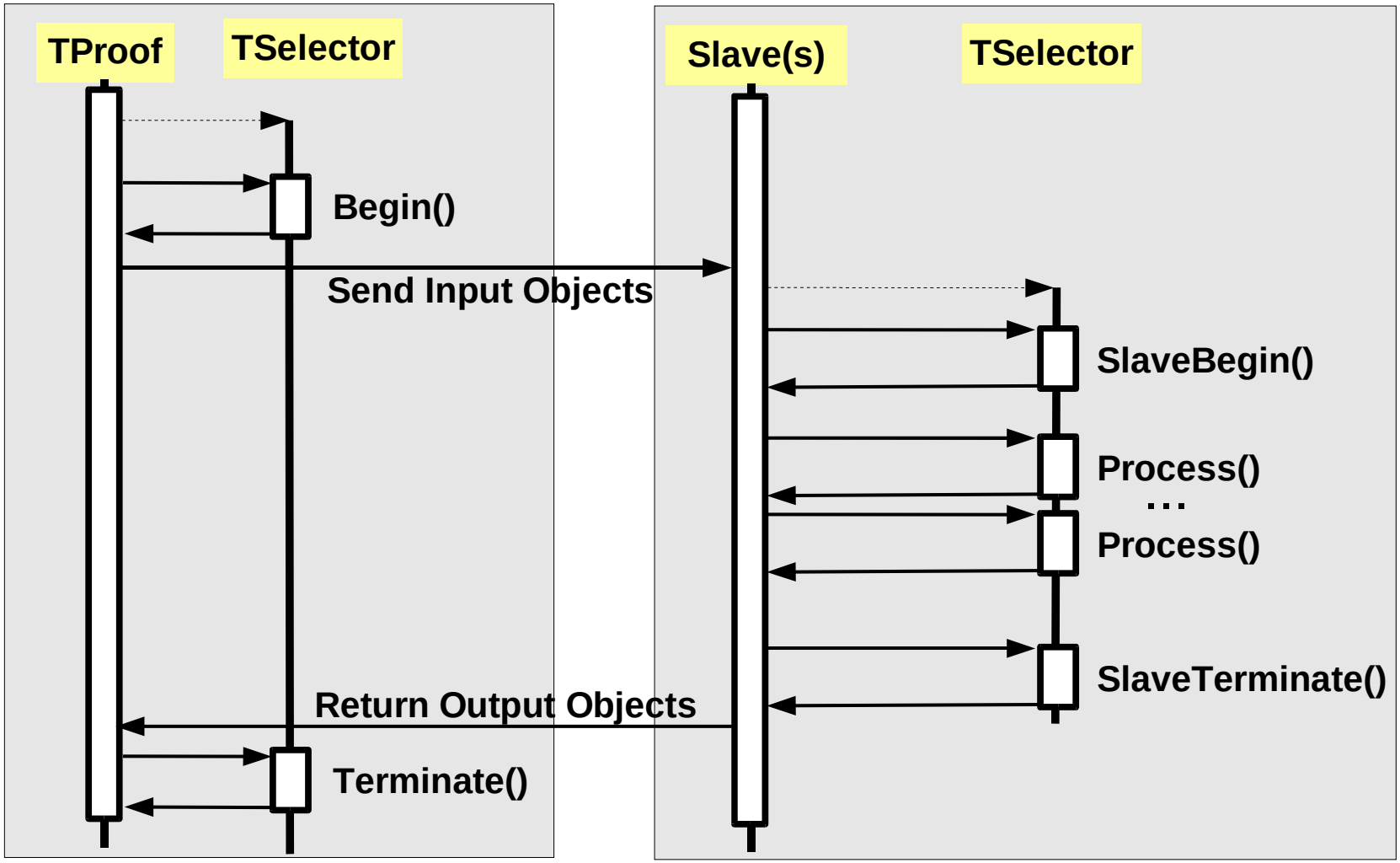
- Add SlaveBegin() and SlaveTerminate()
- Combine ProcessCut() and ProcessFill() into Process()
- Add Version() routine
- Clarify role of Init() and Notify()
  - some details being worked out

TSelector details



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

TSelector control flow



Outline



- PROOF revisited
- Improved TSelector
- **Condor and “Computing on Demand”**
- PROOF and COD
- Plans and ideas
- Conclusions

The Condor Batch System



- Full-featured batch system
  - Job queuing, scheduling policy, priority scheme, resource monitoring and management
- Flexible, distributed architecture
  - Dedicated clusters and/or idle desktops
  - Transparent I/O and file transfer
- Based on 15 years of advanced research
  - Platform for ongoing CS research
  - Production quality, in use around the world, pools with 100's to 1000s of nodes.
- See: http://www.cs.wisc.edu/condor

COD - Computing On Demand



- Share batch resource with interactive use
  - Most of the time normal Condor batch use
  - Interactive job “borrows” the resource for short time
  - Integrated into Condor infrastructure
- Benefits
  - Large amount of resource for interactive burst
  - Efficient use of resources (100% use)
- Active, ongoing research and development

COD - Operations



Normal batch

Request claim

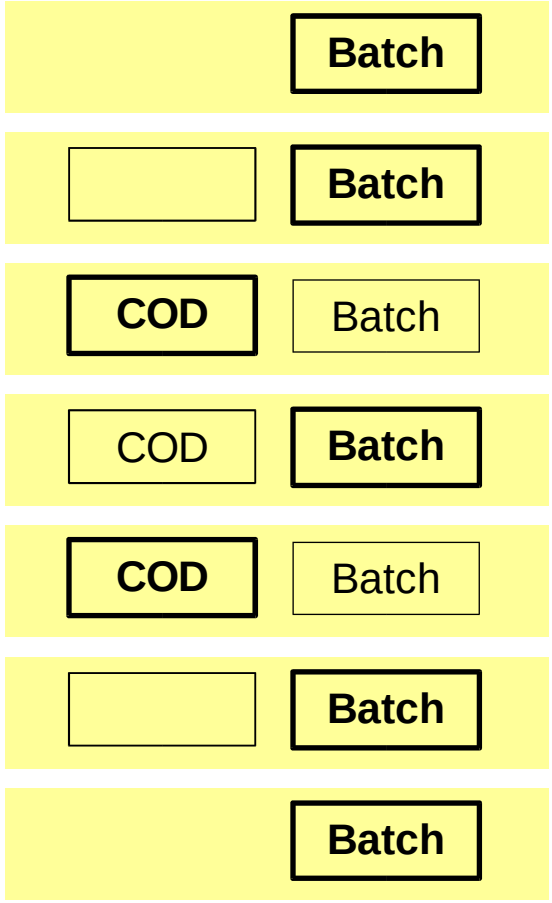
Activate claim

Suspend claim

Resume

Deactivate

Release



Outline



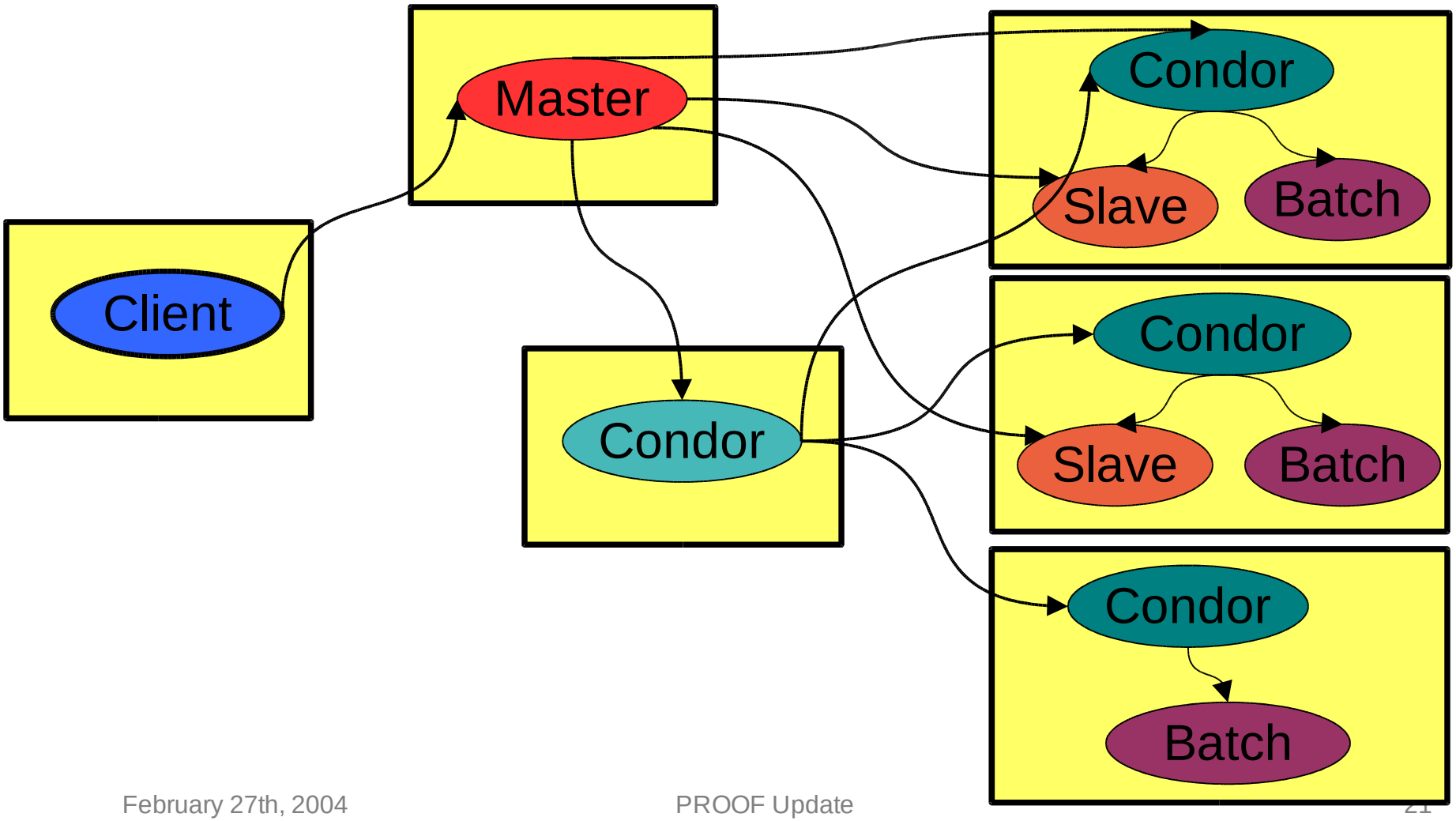
- PROOF revisited
- Improved TSelector
- Condor and “Computing on Demand”
- **PROOF and COD**
- Plans and ideas
- Conclusions

PROOF and COD



- Integrate PROOF and Condor COD
  - Great cooperation with Condor team
- Master starts slaves as COD jobs
  - Standard connection from master to slave
- Master resumes and suspends slaves as needed around queries
- Use Condor or external resource manager to allocate nodes (vm's)

PROOF and COD



PROOF and COD Status



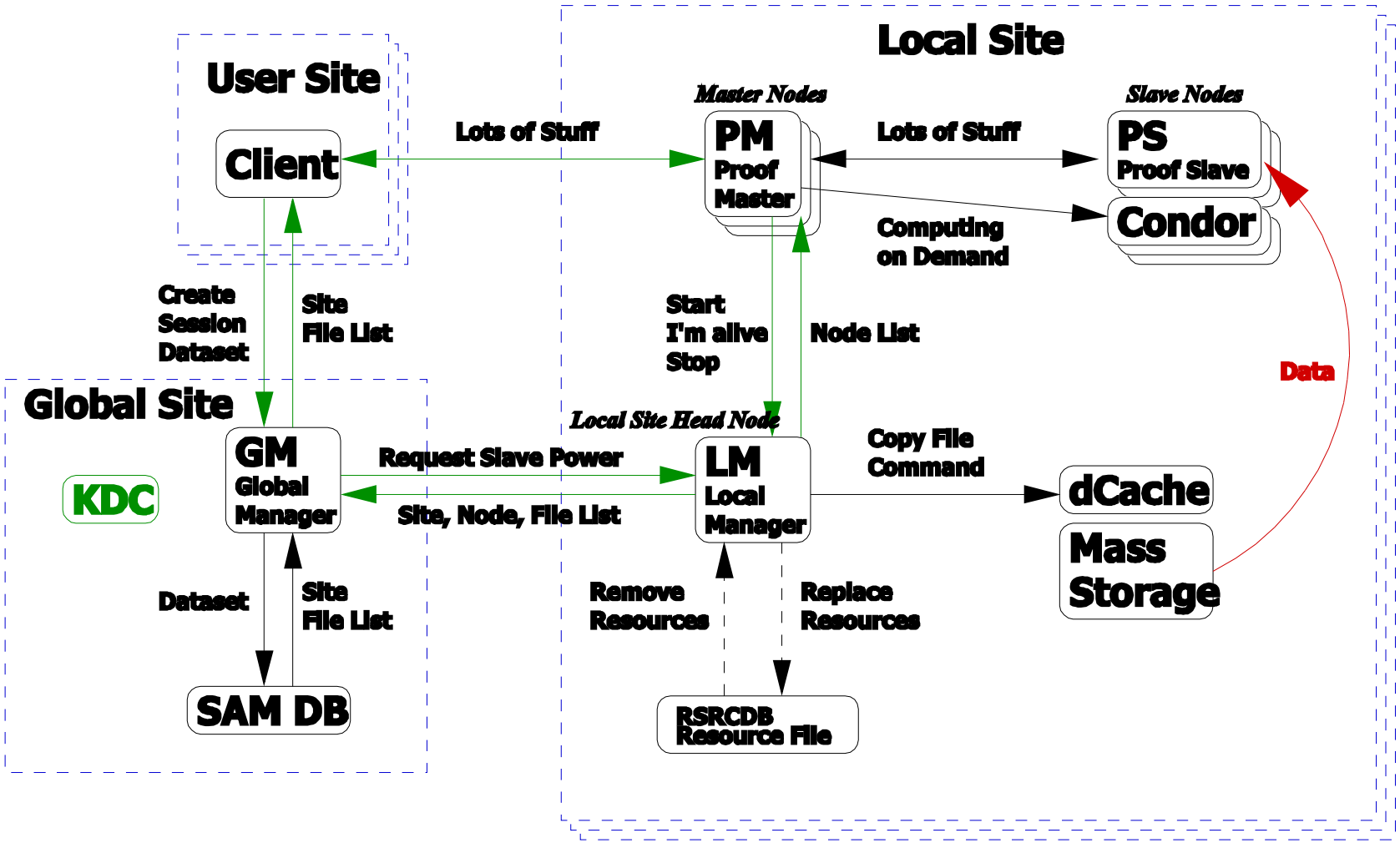
- Status
  - Basic implementation finished
  - Successfully demonstrated at SC'03 with 45 slaves as part of PEAC
- TODO
  - Further improve interface between PROOF and COD
  - Implement resource accounting

PEAC – PROOF Enabled Analysis Cluster



- Complete event analysis solution
  - Data catalog and data management
  - Resource broker
  - PROOF
- Components used: SAM catalog, dCache, new global resource broker, Condor+COD, PROOF
- Multiple computing sites with independent storage systems

PEAC System Overview



PEAC Status



- Successful demo at SC'03
  - Four sites, up to 25 nodes
  - Real CDF StNtuple based analysis
  - COD tested with 45 slaves
- Doing post mortem and plan for next design and implementation phases
  - Available manpower will determine time line
  - Plan to use 250 node cluster at FNAL
  - Other cluster at UCSD

Outline



- PROOF revisited
- Improved TSelector
- Condor and “Computing on Demand”
- PROOF and COD
- **Plans and ideas**
- Conclusions

Upcoming developments



- TProof::Draw()
- Fault tolerance and error handling
- ARDA interface
- Friend Trees and Event lists
- Improved session startup
  - Parallel slave startup
  - Improved Condor interface
  - Abstract interfaces

Conclusions



- PROOF is maturing
- Lot of interest from experiments with large data sets
- COD essential to share batch and interactive work on the same cluster
  - Maximizes resource utilization
- PROOF turns out to be a powerful application to use and show the power of Grid middle ware to its fullest extend
  - See talk by Fons Rademakers on PROOF and AliEn