Tree Analysis Modules

Corey Reed, MIT

ROOT Users Workshop

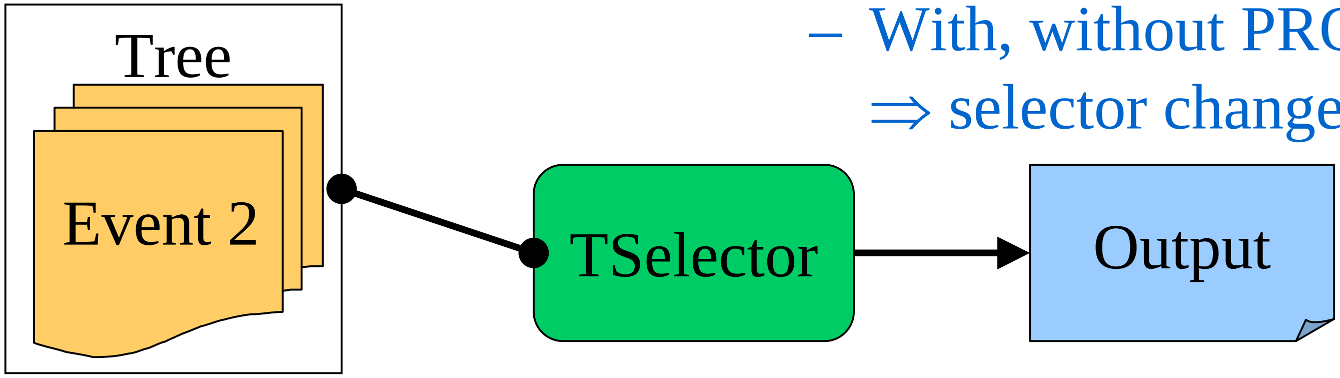
September 29, 2005

What Is TAM?

- Framework for modular analysis of trees
- Works with PROOF
  - Run with or without PROOF – no change to modules
- Handles interaction with tree
  - Efficient
  - Ensure data integrity
- Developed at MIT by Corey Reed, Maarten Ballintjin

Why TAM?

- TSelector: Strength
  - Automatic tree interaction
  - Structured analysis
  - Interface for PROOF
- TSelector: Weakness
  - Big macros
  - New analysis ⇒ new selector
  - Tree change ⇒ selector change
  - With, without PROOF ⇒ selector change

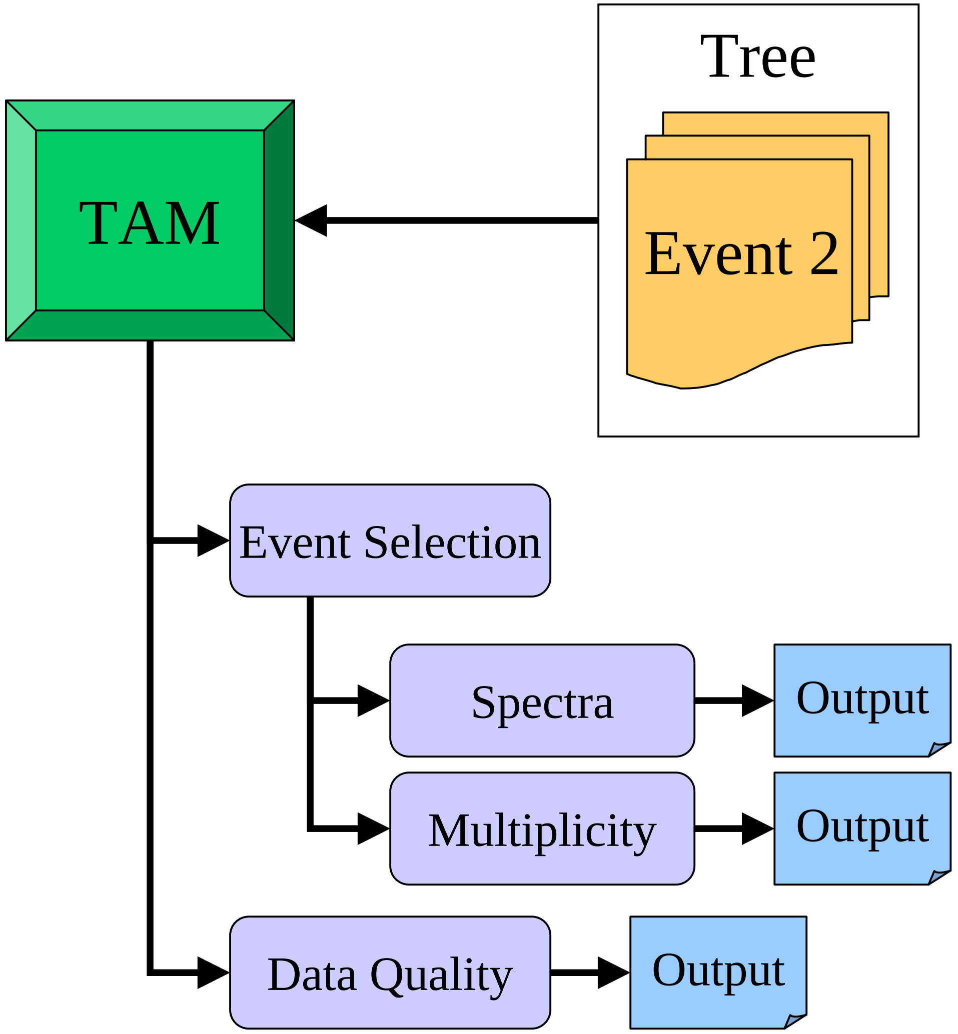


Why TAM?

- TAM
  - Generic TSelector
    - Preserves strengths
  - Modules, not macros
    - Structured like TSelectors (Begin, Process, etc.)
    - Analysis separated from tree structure
      - Tree structure change: module unchanged
    - Code portable
  - With, without PROOF: transparent for user

Analysis With TAM

1. Module asks TAM for data
2. TAM reads tree, shows data to module
3. Module processes data
   - Applies cuts
   - Produces output
4. Run sub-modules



TAM Organization

- TAMSelector
  - Derives from TSelector
  - Runs, manages TAModules
  - Handles all interaction with tree
- TAModule
  - Base class of all modules
  - Derives from TTask
- TAMOutput
  - Stores output objects of a module
  - Merges output under PROOF

TAModule

- May contain submodules (TTask)
- Structure like TSelector
  - `Begin`, `Process`, `SlaveTerminate`, etc.
- Use `ReqBranch(name, pointer)` to request each branch the module might use
- Use `LoadBranch(name)` to load the data

A User’s Module

```
class TMyMod : public TAModule {
private:
   TPhAnTEventInfo*  fEvtInfo; // event info
   TH1F*             fEvtNumH; // event num histogram
protected:
   void SlaveBegin();
   void Process();
```

```
void TMyMod::SlaveBegin() {
   ReqBranch(“eventInfo”, fEvtInfo);
   fEvtNumH = new TH1F(“EvtNumH”,”Event Num”,10,0,10);
}
```

```
void TMyMod::Process() {
   LoadBranch(“eventInfo”);
   fEvtNumH->Fill(fEvtInfo->fEventNum);
}
```

Example Analysis

- Build module hierarchy:

```
TMyMod* myMod = new TMyMod;
TMyOtherMod* subMod = new TMyOtherMod;
myMod->Add(subMod);
```

  - No PROOF:

```
TAMSelector* mySel = new TAMSelector;
mySel->AddInput(myMod);
tree->Process(mySel);
TList* output = mySel->GetModOutput();
```

  - With PROOF:

```
dset->AddInput(myMod);
dset->Process(“TAMSelector”);
TList* output = gProof->GetOutputList();
```

Example Analysis

- Build module hierarchy:

```
TMyMod* myMod = new TMyMod;
TMyOtherMod* subMod = new TMyOtherMod;
myMod->Add(subMod);
```

  - No PROOF:

```
TAMSelector* mySel = new TAMSelector;
mySel->AddInput(myMod);
tree->Process(mySel);
TList* output = mySel->GetModO
```

Note Similarity

  - With PROOF:

```
dset->AddInput(myMod);
dset->Process("TAMSelector");
TList* output = gProof->GetOutputList();
```

Reading Data: TAM

- At ReqBranch, TAMSelector builds table:
  - The branch
  - Array of module's pointers to the branch
  - Address branch will be read into
  - Flags
    - already loaded?
    - stores a class?
    - stores fundamental types of equal size?

Reading Data: TAM

- Before Process
  - Module pointers reset to 0
- At LoadBranch, TAMSelector reads data:
  - If not already loaded
    - Call GetEntry on branch
    - Set all module pointers to point to the data

Data Integrity

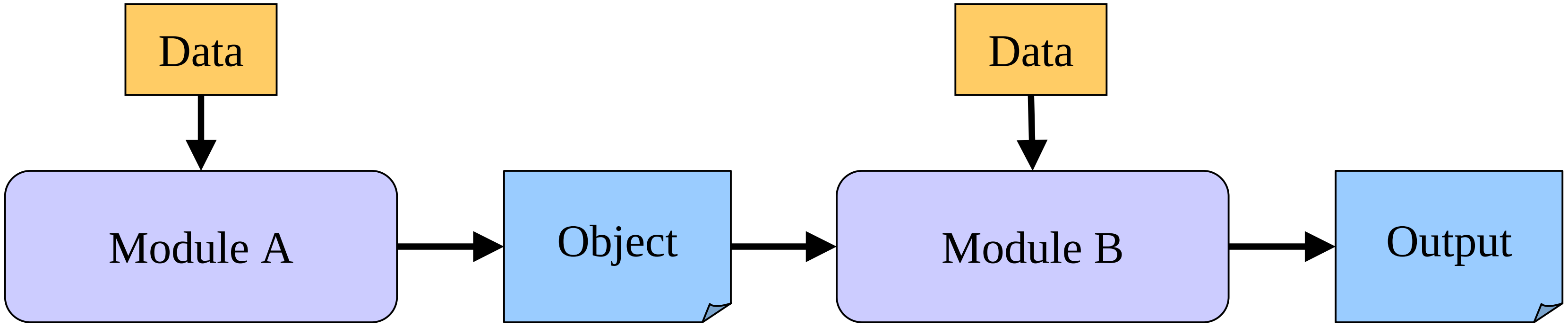
- Type checking
  - Minimal use of templates
    - Store type of pointer
  - Class
    - `type_info(pointer type)`
    - `TClass::GetTypeInfo()`

Data Integrity

- Type checking
  - Fundamentals
    - Use struct that is in the ROOT dictionary
    - `TDataMember::GetTypeName()`
    - `TLeaf::GetTypeName()`
    - Check if each leaf is the same size in memory
    - Variable sized leafs?
      - Use TDataMember::GetOffset()
      - Set each leaf address to address of member in struct

Inter-module Communication

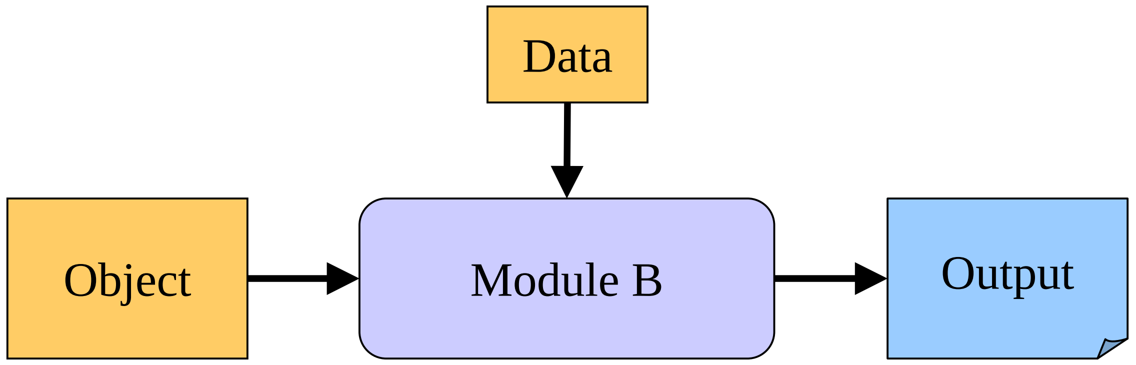
- Intermediate processing



  - Object available only during current event
    - Deleted after Process automatically
  - Available to all modules
  - Ex: tracks produced from hits in the tree

Inter-module Communication

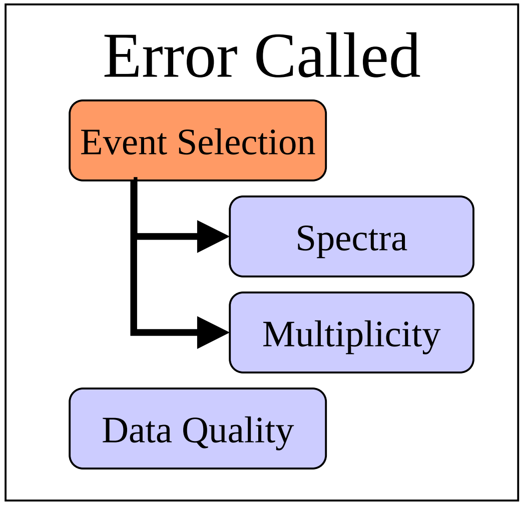
- Independent of event

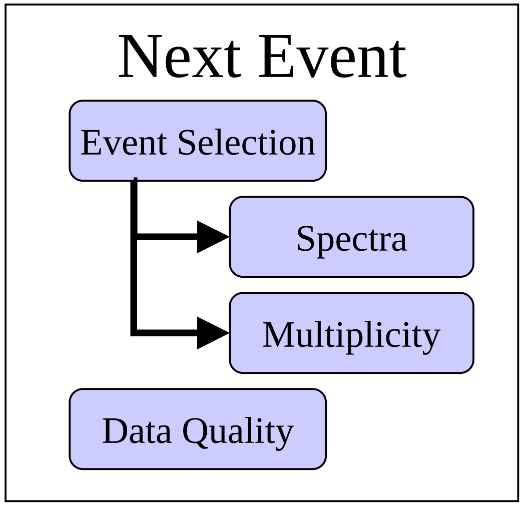


  - Object always available
  - Available to all modules
  - Ex: calibrations

Error Handling

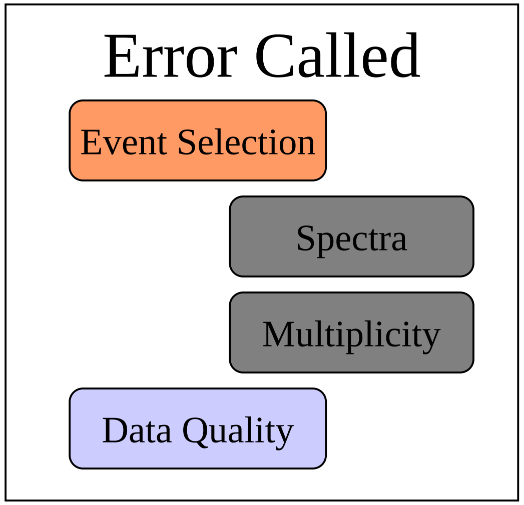
- Message + processing break
  1. No break
  2. Stop module
     - Stop sub-modules
     - Resets at next event
  3. Stop event
     - Stop all modules
     - Resets at next event
  4. Stop entire analysis

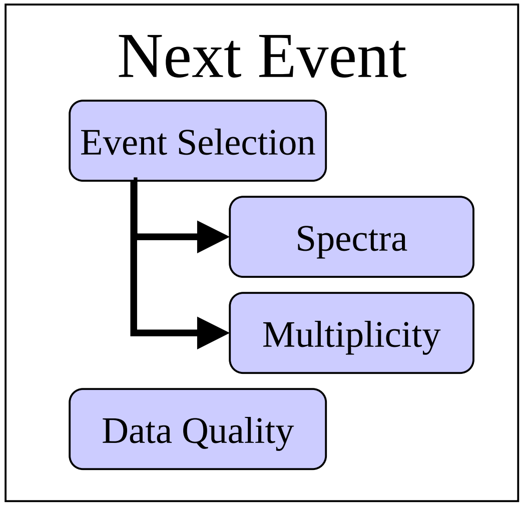




Error Handling

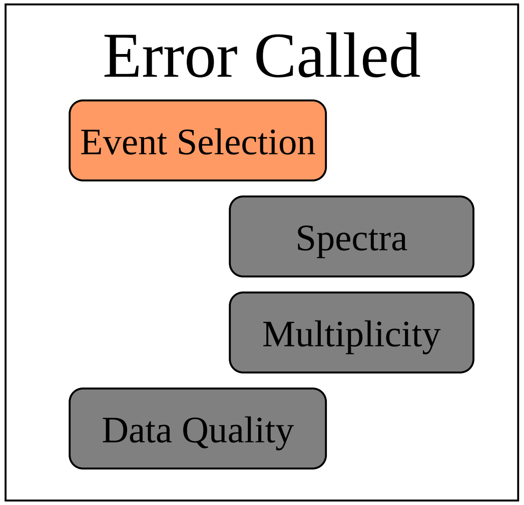
- Message + processing break
  1. No break
  2. Stop module
     - Stop sub-modules
     - Resets at next event
  3. Stop event
     - Stop all modules
     - Resets at next event
  4. Stop entire analysis

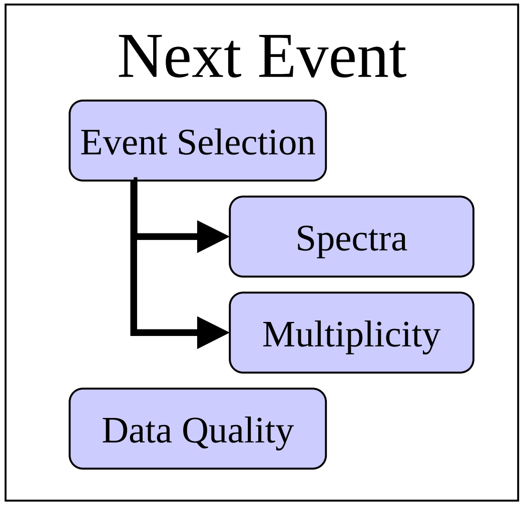




Error Handling

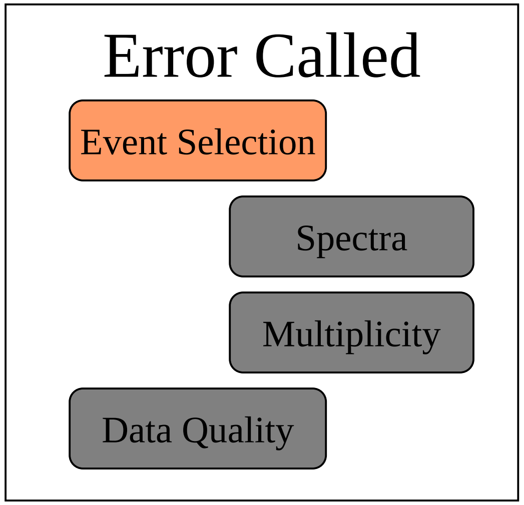
- Message + processing break
  1. No break
  2. Stop module
     - Stop sub-modules
     - Resets at next event
  3. Stop event
     - Stop all modules
     - Resets at next event
  4. Stop entire analysis

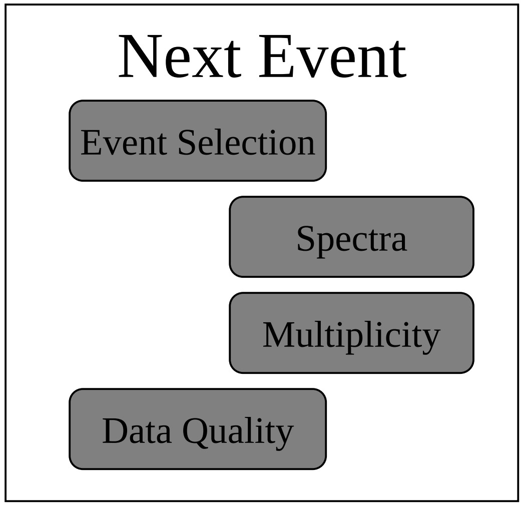




Error Handling

- Message + processing break
  1. No break
  2. Stop module
     - Stop sub-modules
     - Resets at next event
  3. Stop event
     - Stop all modules
     - Resets at next event
  4. Stop entire analysis



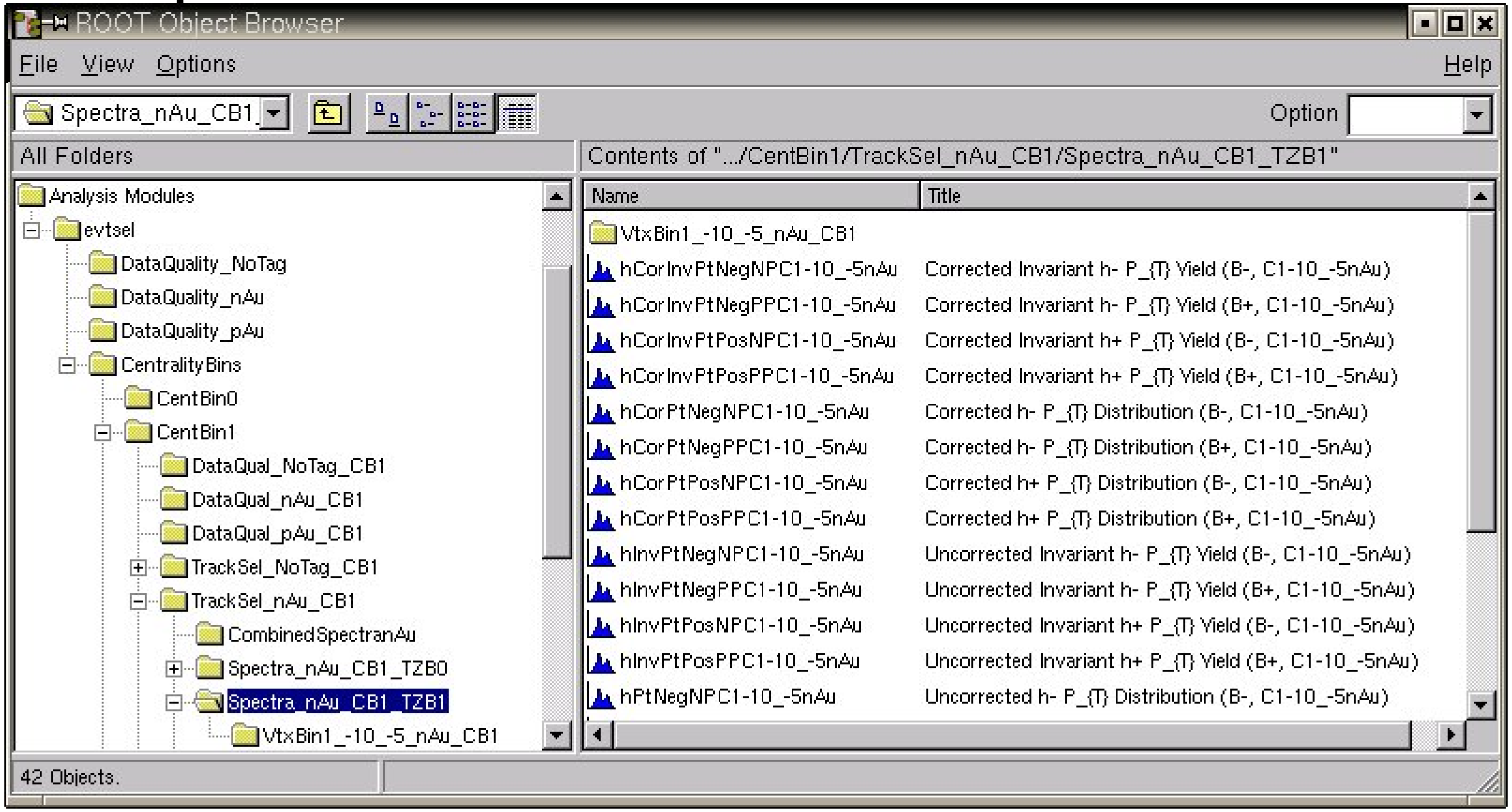


Module Output

- Output stored using `AddOutput(TObject*&)`
- Output list written to file
  - Preserving module hierarchy
  - Flattened
- Output list interface:
  - `FindOutput(obj name)`
  - `FindOutput(submodule name, obj name)`
  - `RemoveOutput(TObject*)`

Module Output

- Output can be browsed



ROOT Object Browser

File View Options Help

Spectra_nAu_CB1 Option

All Folders

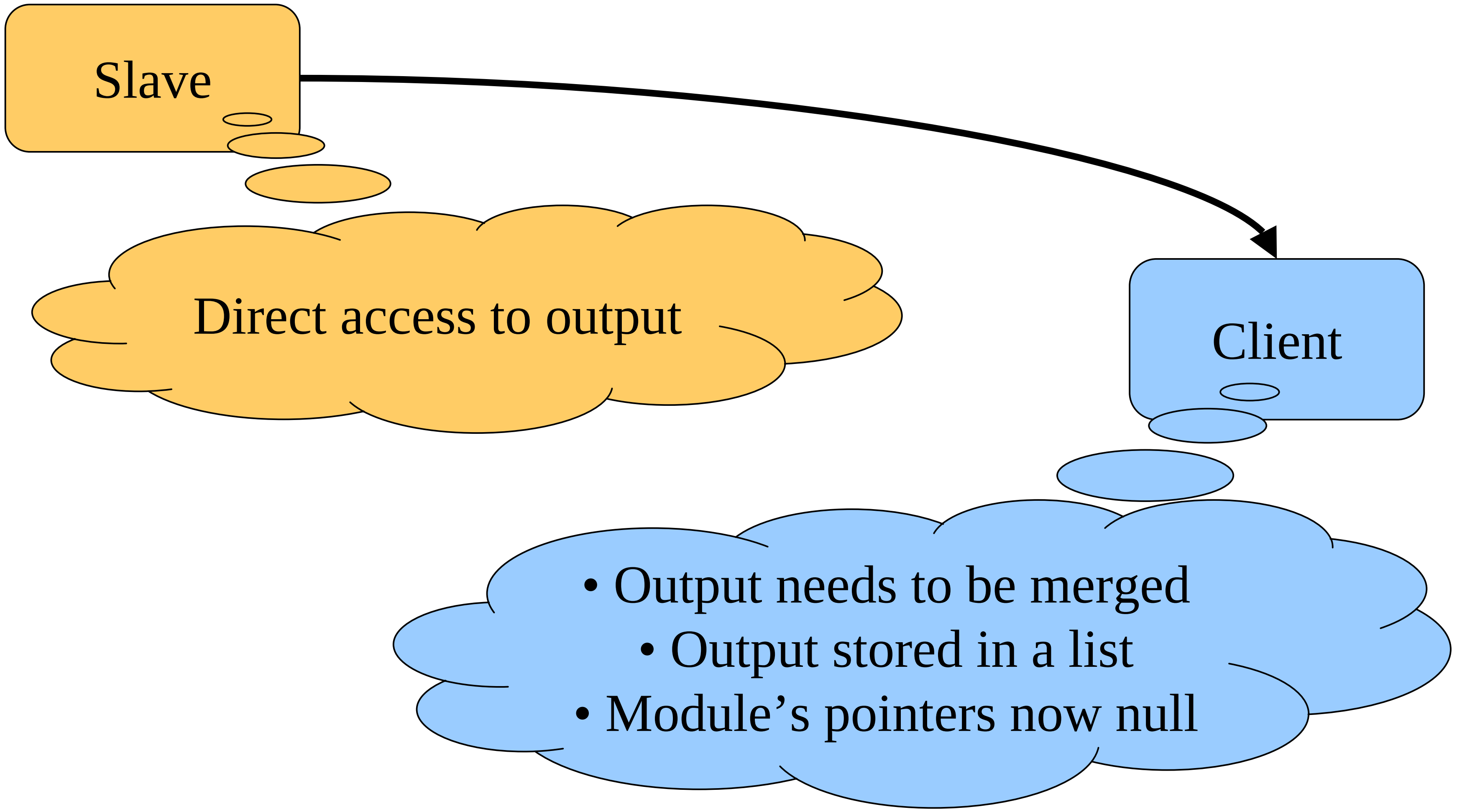
```
Analysis Modules
  evtsel
    DataQuality_NoTag
    DataQuality_nAu
    DataQuality_pAu
    CentralityBins
      CentBin0
      CentBin1
        DataQual_NoTag_CB1
        DataQual_nAu_CB1
        DataQual_pAu_CB1
        TrackSel_NoTag_CB1
        TrackSel_nAu_CB1
          CombinedSpectranAu
          Spectra_nAu_CB1_TZB0
          Spectra_nAu_CB1_TZB1
            VtxBin1_-10_-5_nAu_CB1
```

Contents of ".../CentBin1/TrackSel_nAu_CB1/Spectra_nAu_CB1_TZB1"

Name	Title
VtxBin1_-10_-5_nAu_CB1	
hCorInvPtNegNPC1-10_-5nAu	Corrected Invariant h- P_{T} Yield (B-, C1-10_-5nAu)
hCorInvPtNegPPC1-10_-5nAu	Corrected Invariant h- P_{T} Yield (B+, C1-10_-5nAu)
hCorInvPtPosNPC1-10_-5nAu	Corrected Invariant h+ P_{T} Yield (B-, C1-10_-5nAu)
hCorInvPtPosPPC1-10_-5nAu	Corrected Invariant h+ P_{T} Yield (B+, C1-10_-5nAu)
hCorPtNegNPC1-10_-5nAu	Corrected h- P_{T} Distribution (B-, C1-10_-5nAu)
hCorPtNegPPC1-10_-5nAu	Corrected h- P_{T} Distribution (B+, C1-10_-5nAu)
hCorPtPosNPC1-10_-5nAu	Corrected h+ P_{T} Distribution (B-, C1-10_-5nAu)
hCorPtPosPPC1-10_-5nAu	Corrected h+ P_{T} Distribution (B+, C1-10_-5nAu)
hInvPtNegNPC1-10_-5nAu	Uncorrected Invariant h- P_{T} Yield (B-, C1-10_-5nAu)
hInvPtNegPPC1-10_-5nAu	Uncorrected Invariant h- P_{T} Yield (B+, C1-10_-5nAu)
hInvPtPosNPC1-10_-5nAu	Uncorrected Invariant h+ P_{T} Yield (B-, C1-10_-5nAu)
hInvPtPosPPC1-10_-5nAu	Uncorrected Invariant h+ P_{T} Yield (B+, C1-10_-5nAu)
hPtNegNPC1-10_-5nAu	Uncorrected h- P_{T} Distribution (B-, C1-10_-5nAu)

42 Objects.

Module Output - PROOF


Slave
Direct access to output
Client
• Output needs to be merged
• Output stored in a list
• Module’s pointers now null

Module Output - PROOF

- Merging
  - TAMOutput in PROOF output list
  - Hierarchy preserved
  - Output objects merged by TAMOutput
- Output pulled from PROOF output list
  - User access via TAMOutput

Module Output - PROOF

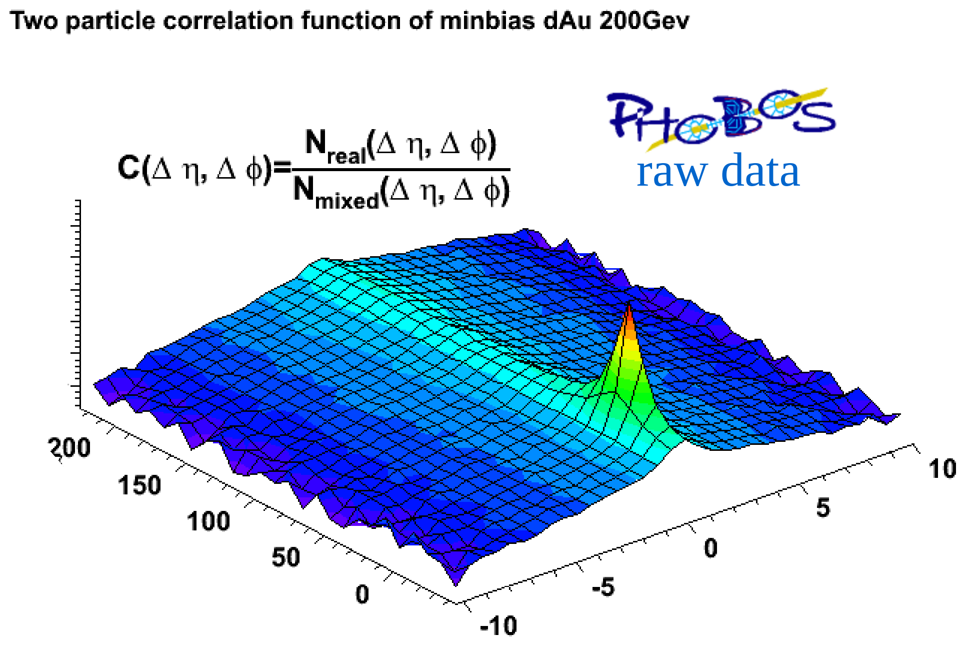
- Restoring module pointers
  - On `AddOutput(TObject*&)`
    - Store address of pointer
    - Store name of object
  - Before Terminate
    - Set module pointers to point to output objects
- To be automatic...
  - `AddOutput` called with module member
  - Module member is pointer to a class
    - Not address of instance
    - No arrays

Module Output - PROOF

- Multiple PROOF sessions
  - Must not merge more than once!
  - Two lists in TAMOutput
    - Current output objects
      - Merged
    - Stored output objects
      - Not merged
  - Before Terminate, after merge
    - Move objects from current → stored

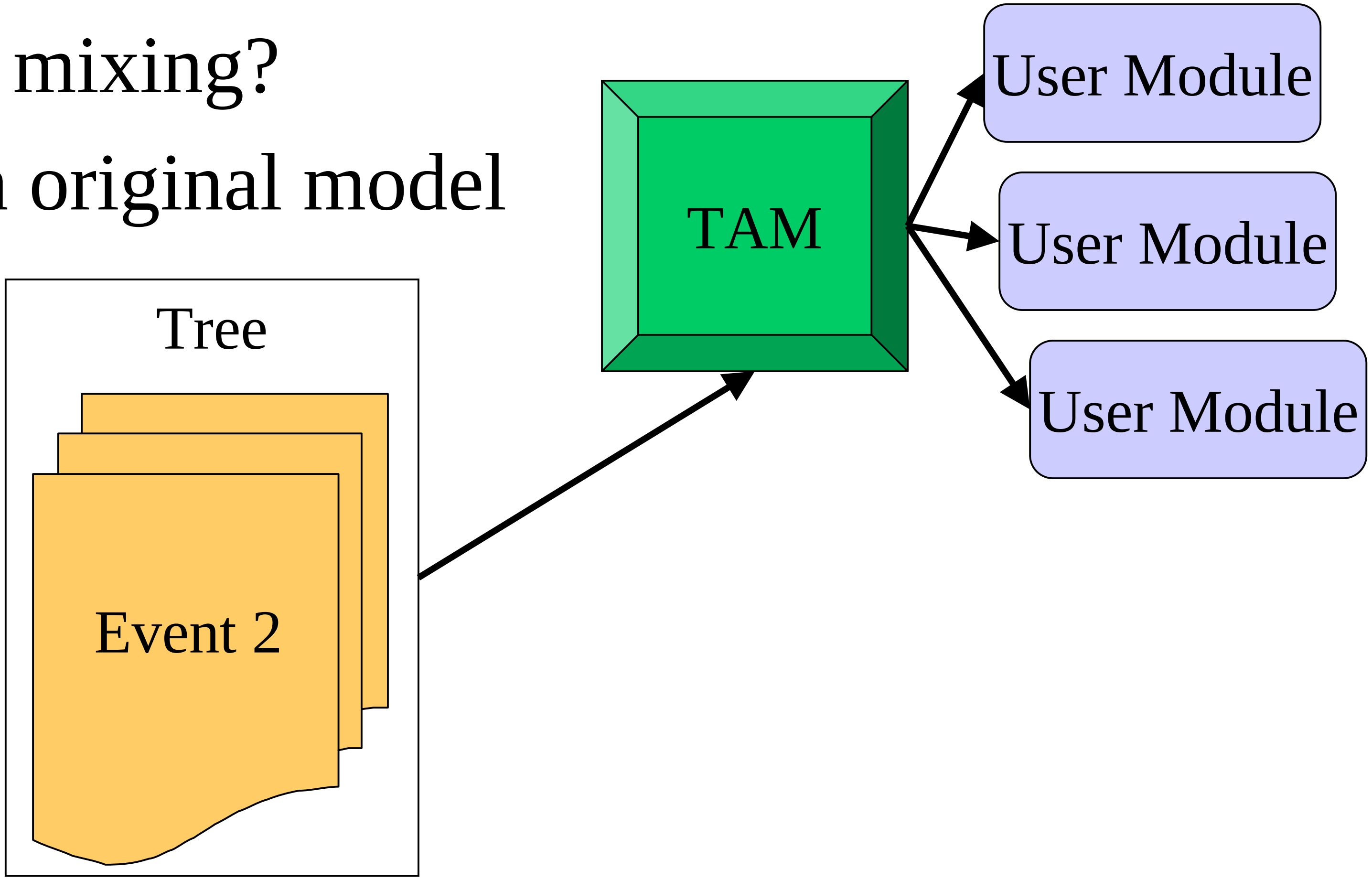
TAM in Phobos

- Phobos results shown at Quark Matter 05
  - Rare event search
  - 2-particle correlations
  - Hadron p_T spectra



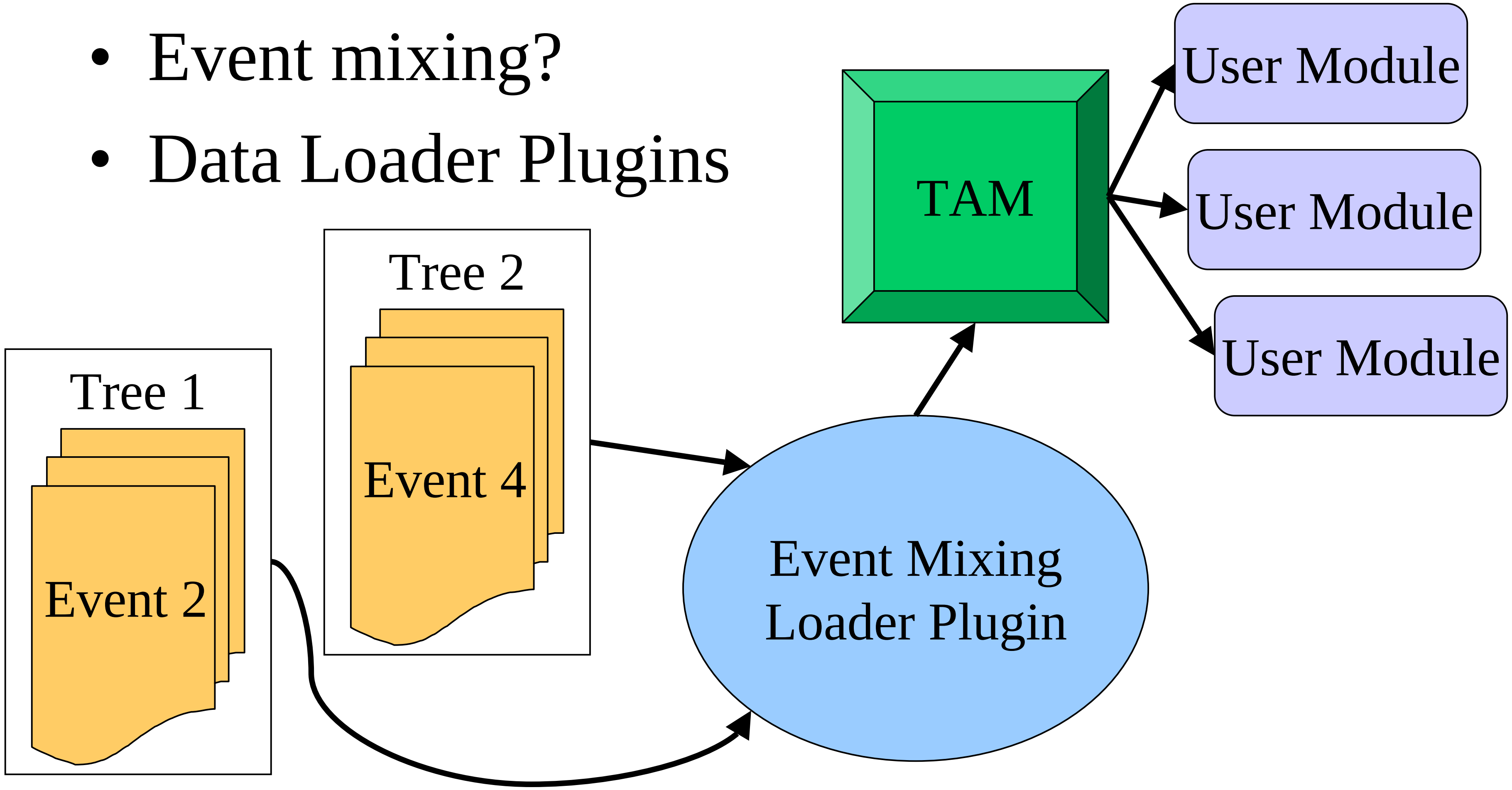
Into The Future

- Event mixing?
- Not in original model



Into The Future

- Event mixing?
- Data Loader Plugins



Summary

- TAM
  - Framework for module-based analysis
  - Works with PROOF
  - Ensures efficiency, data integrity
- Available at `http://higweb.lns.mit.edu/tam/`
  - Source code
  - Documentation