Constructing New Indexing System in ATLAS Database

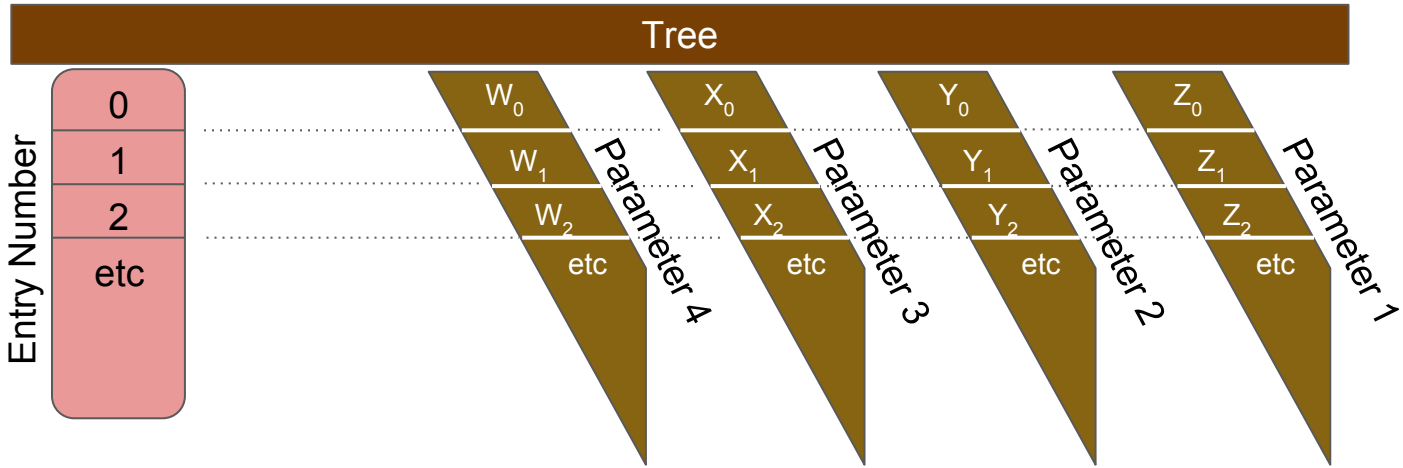
Nikita Dulin
DOE SULI Summer 2018
Mentor: Peter van Gemmeren

ROOT Data Storage and I/O

Data stored column-wise in TTrees and TBranches

Different branch for each parameter - different values for each entry

Basically n-tuples, each entry number referring to a single n-tuple

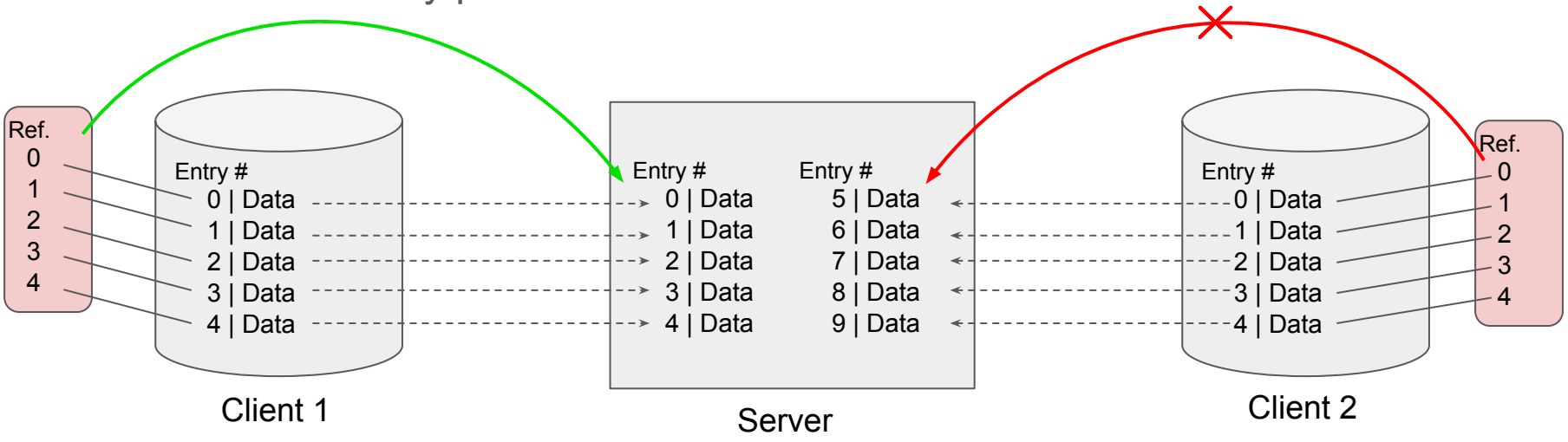


Introduction

ATLAS uses entry number for referencing

In-memory merging from multiple clients resets entry numbers

Difficult to identify particular events

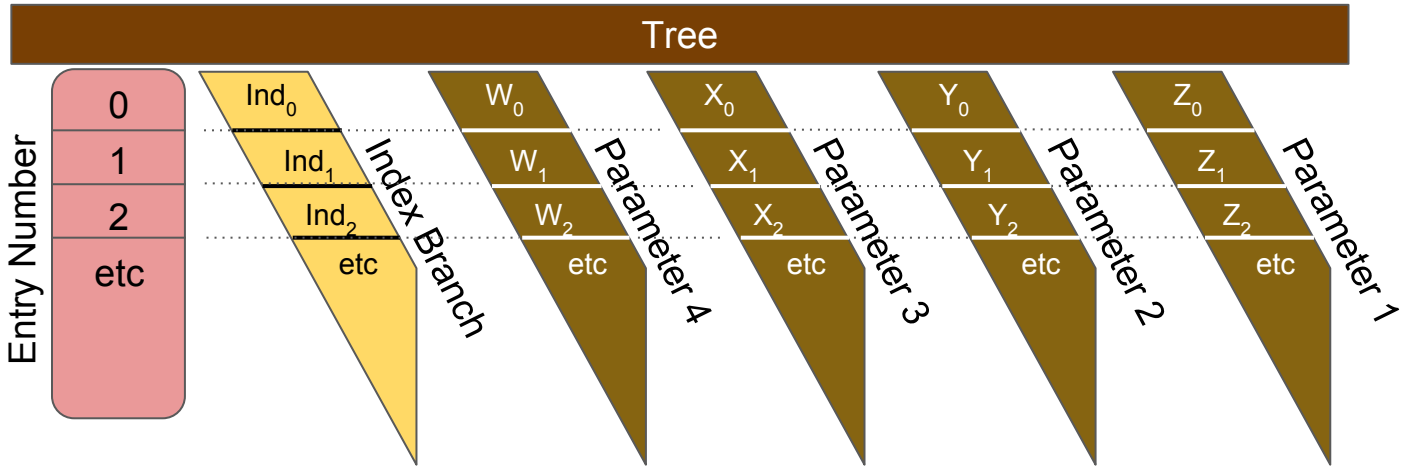


Building Index

Create separate index branch with unique identifiers for each entry/event

Added at the same time an event is recorded

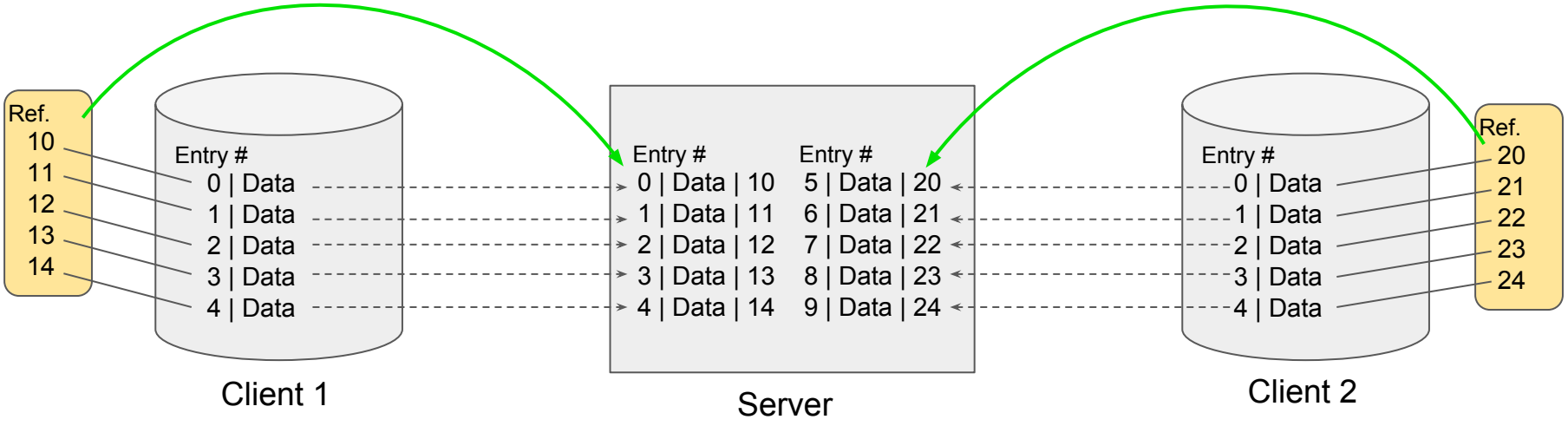
Assign branch as index (TTreeIndex class)



Building Index

Create separate index branch with unique identifiers for each entry/event

Added at the same time an event is recorded

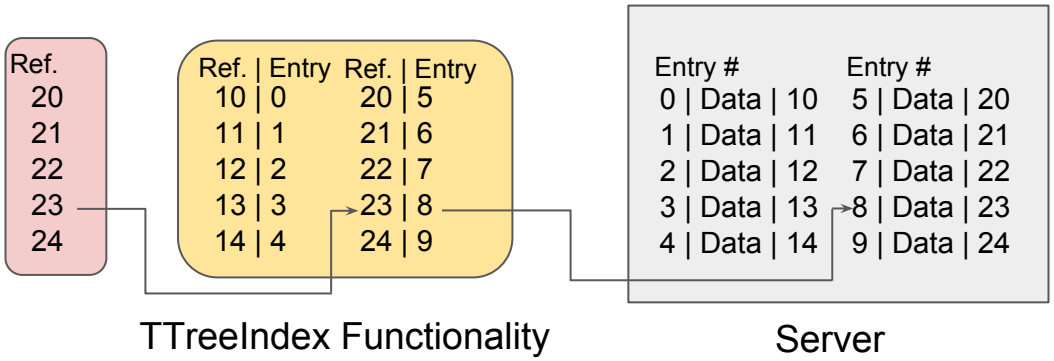


Reading from Index

Tree at server can map index number to entry number

Use index number to get entry number

Call using entry number as before



Athena POOL / APR

ATLAS uses POOL / APR

- Abstraction layer allowing for special technologies

RootTreeContainer - existing technology

- Facilitates creation/filling/writing of trees and branches

Created new technology: RootTreeIndexContainer

- Specialized RootTreeContainer to facilitate indexing

Transact

POOL has transactional model we can use

Modify RootTreeContainer version

- Called each time event added
- Called at write time

Create index branch after other branches are created (per tree)

Fill index branch each time other branches filled

```
DbStatus RootTreeIndexContainer::transAct(Transaction::Action  action){
  if (action == Transaction::TRANSACT_COMMIT) {
    if (m_tree->GetName()[0]  != '#') {
      if (m_tree == nullptr) return Error;
      if (m_index_ref == nullptr && m_tree->GetBranch("index_ref")  == nullptr) {
        m_index_ref = (TBranch*)m_tree->Branch("index_ref",  m_index);
      }
      if (m_index_ref != nullptr) {
        if (RootTreeContainer::size()  > m_index_ref->GetEntries()) {
          *m_index = this->size();
          m_index_ref->SetAddress(m_index);
          if (!m_treeFillMode)  m_index_ref->Fill();
        }
      }
    }
  }
  …
```

nextRecordId

Creates and assigns index number

Concatenation of process id and entry number

Bit shifted for uniqueness

(since process id resets after some time)

```
long long int RootTreeIndexContainer::nextRecordId()    {
  long long int s = m_index_multi;
  s = s << 32;
  if (m_tree != nullptr) {
    if (m_index_foreign != nullptr) {
      s += m_index_foreign->GetEntries();
    } else {
      m_index_foreign =
(TBranch*)m_tree->GetBranch("index_ref");
      if (m_index_foreign != nullptr) {
        s += m_index_foreign->GetEntries();
      }
    }
  }
  return s;
}
```

Building the Index

TTreeIndex::BuildIndex()

Called at final transAct call

Checks to see if the index is already built

Can be called at read, but not preferred

Can be called by client or server

in transAct:

```
…
DbStatus status = RootTreeContainer::transAct(action);
  if (action == Transaction::TRANSACT_FLUSH) {
    if (m_tree->GetName()[0]  != '#') {
      if (m_tree->GetEntryNumberWithIndex(size())   == -1) {
        m_tree->BuildIndex("index_ref");
      }
    }
  }
  return Status;
}
```

Reading

Get index number from input

Retrieve entry number

Call original function using entry number

(as things are currently done)

```
DbStatus
RootTreeIndexContainer::loadObject(DataCallBack*  call,
                                   Token::OID_t& oid, DbAccessMode mode) {
  if ((oid.second >> 32) > 0) {
    long long int evt_id = m_tree->GetEntryNumberWithIndex(oid.second);
    if (evt_id == -1) {
      m_tree->BuildIndex("index_ref");
      evt_id = m_tree->GetEntryNumberWithIndex(oid.second);
    }
    if (evt_id >= 0) {
      oid.second = evt_id;
    }
  }
  return RootTreeContainer::loadObject(call, oid, mode);
}
```

Conclusion

Everything is working

Code has been uploaded but has not been set as default

- More tests to run