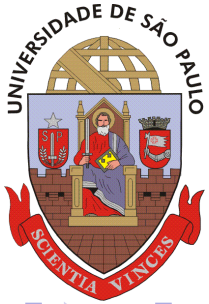
A parallel implementation of Geant4 application using OOMPI

Luís Augusto Perles, Adelaide de Almeida

Departamento de Física e Matemática
Faculdade de Filosofia Ciências e Letras de Ribeirão Preto
USP - Brazil

MCNEG 2004



Outline

Outline

Some examples

- Fast results in scientific research (everyone needs)
- Use in radiotherapy Treatment Planning Systems (TPS) in hospitals

Some examples

- Fast results in scientific research (everyone needs)
- Use in radiotherapy Treatment Planning Systems (TPS) in hospitals

Outline

When parallelize?

- When we need a lot of histories to be processed (event parallelism)
- When we have secondaries hard to track (track parallelism)
- When we have a cluster or a multiprocessor machine (of course!)

When parallelize?

- When we need a lot of histories to be processed (event parallelism)
- When we have secondaries hard to track (track parallelism)
- When we have a cluster or a multiprocessor machine (of course!)

When parallelize?

- When we need a lot of histories to be processed (event parallelism)
- When we have secondaries hard to track (track parallelism)
- When we have a cluster or a multiprocessor machine (of course!)

Outline

The main choices for OO

Para++

- Works with both MPI and PVM library
- Doesn't care about MPI Standard
- Is it still updated?

OOMPI

- Cares about MPI standard
- Frequently updated
- Very small overhead compared against MPI for C

The main choices for OO

Para++

- Works with both MPI and PVM library
- Doesn't care about MPI Standard
- Is it still updated?

OOMPI

- Cares about MPI standard
- Frequently updated
- Very small overhead compared against MPI for C

The main choices for OO

Para++

- Works with both MPI and PVM library
- Doesn't care about MPI Standard
- Is it still updated?

OOMPI

- Cares about MPI standard
- Frequently updated
- Very small overhead compared against MPI for C

The main choices for OO

Para++

- Works with both MPI and PVM library
- Doesn't care about MPI Standard
- Is it still updated?

OOMPI

- Cares about MPI standard
- Frequently updated
- Very small overhead compared against MPI for C

The main choices for OO

Para++

- Works with both MPI and PVM library
- Doesn't care about MPI Standard
- Is it still updated?

OOMPI

- Cares about MPI standard
- Frequently updated
- Very small overhead compared against MPI for C

The main choices for OO

Para++

- Works with both MPI and PVM library
- Doesn't care about MPI Standard
- Is it still updated?

OOMPI

- Cares about MPI standard
- Frequently updated
- Very small overhead compared against MPI for C

Some OOMPI features

1. Few global objects (MPI_COMM_WORLD is default)
2. Send/receive objects inherited from OOMPI_User_type
3. Each MPI function has a related class method

Object Oriented Message Passing Interface - OOMPI

Open Systems Laboratory
Pervasive Technologies Labs
Indiana University
Homepage: `http://www.osl.iu.edu/research/oompi/`

Some OOMPI features

1. Few global objects (MPI_COMM_WORLD is default)
2. Send/receive objects inherited from OOMPI_User_type
3. Each MPI function has a related class method

Object Oriented Message Passing Interface - OOMPI

Open Systems Laboratory
Pervasive Technologies Labs
Indiana University
Homepage: `http://www.osl.iu.edu/research/oompi/`

Some OOMPI features

1. Few global objects (MPI_COMM_WORLD is default)
2. Send/receive objects inherited from OOMPI_User_type
3. Each MPI function has a related class method

Object Oriented Message Passing Interface - OOMPI

Open Systems Laboratory
Pervasive Technologies Labs
Indiana University
Homepage: `http://www.osl.iu.edu/research/oompi/`

Some OOMPI features

1. Few global objects (MPI_COMM_WORLD is default)
2. Send/receive objects inherited from OOMPI_User_type
3. Each MPI function has a related class method

Object Oriented Message Passing Interface - OOMPI

Open Systems Laboratory
Pervasive Technologies Labs
Indiana University
Homepage: `http://www.osl.iu.edu/research/oompi/`

Other non-OO libraries

Other parallel libraries for C

- PVM
- MPI (LAM-MPI, MPICH, ...)
- TOP-C (MPI based)

Other non-OO libraries

Other parallel libraries for C

- PVM
- MPI (LAM-MPI, MPICH, ...)
- TOP-C (MPI based)

Other non-OO libraries

Other parallel libraries for C

- PVM
- MPI (LAM-MPI, MPICH, ...)
- TOP-C (MPI based)

TOP-C features and limitations

Features

- Task oriented abstraction
- Latency tolerance
- Works with shared and distributed memory

Limitations

- Have functions instead of classes
- Difficult to send/receive objects through MPI
- Scalability is a problem for > 100 nodes

TOP-C features and limitations

Features

- Task oriented abstraction
- Latency tolerance
- Works with shared and distributed memory

Limitations

- Have functions instead of classes
- Difficult to send/receive objects through MPI
- Scalability is a problem for $>$ 100 nodes

Outline

Important highlights

ROOT is used!

- This implementation uses ROOT objects to store and send/receive data through OOMPI interface!
- TH3D has been used to store dose data or TTrees to store tracks and hits since their first sequential implementations.
- TOOMPI class library was created to easily send/receive ROOT TObject's classes

So none G4THitsCollection is sent/received in this implementation!

Important highlights

ROOT is used!

- This implementation uses ROOT objects to store and send/receive data through OOMPI interface!
- TH3D has been used to store dose data or TTrees to store tracks and hits since their first sequential implementations.
- TOOMPI class library was created to easily send/receive ROOT TObject's classes

So none G4THitsCollection is sent/received in this implementation!

TOOMPI class

TOOMPI is a small class library created to serialize and send/receive ROOT TObject's through OOMPI interface.

Currently its main methods are:

- SendObject sends a TObject inherited class
- RecvObject receives a TObject inherited class
- Get_source shows from which node the current message comes

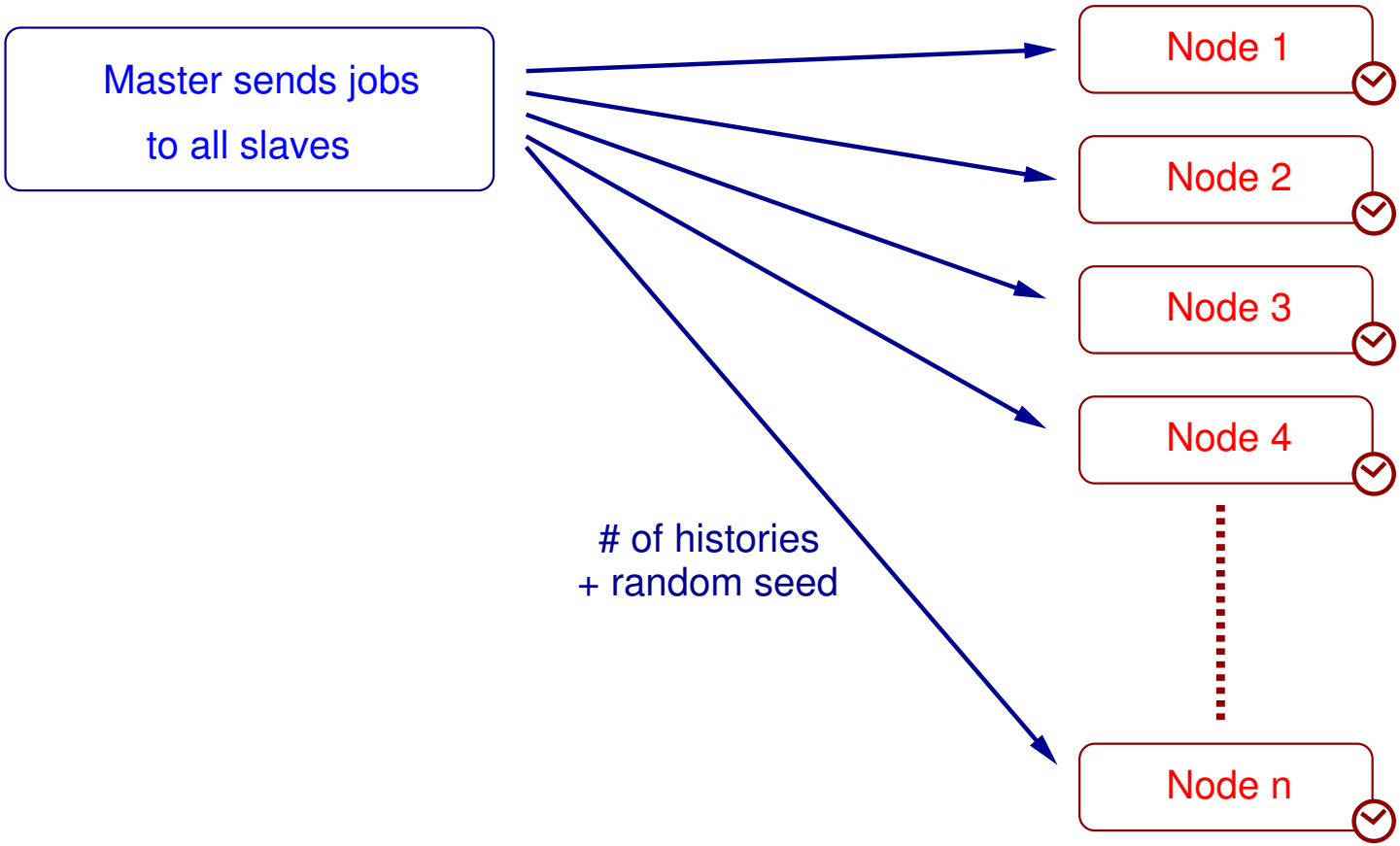
TOOMPI class

TOOMPI is a small class library created to serialize and send/receive ROOT TObject's through OOMPI interface. Currently its main methods are:

- SendObject sends a TObject inherited class
- RecvObject receives a TObject inherited class
- Get_source shows from which node the current message comes

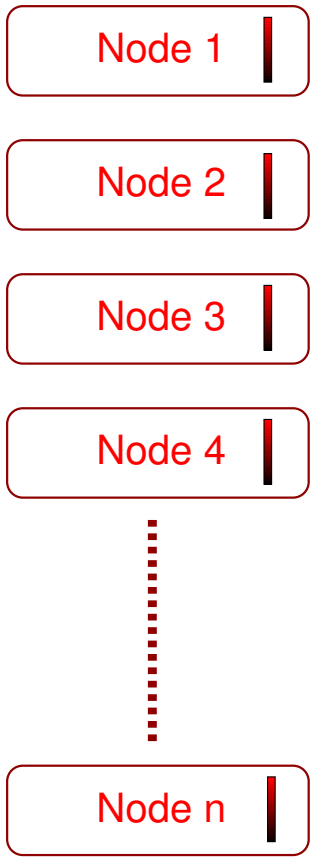
Outline

How it works - Starting

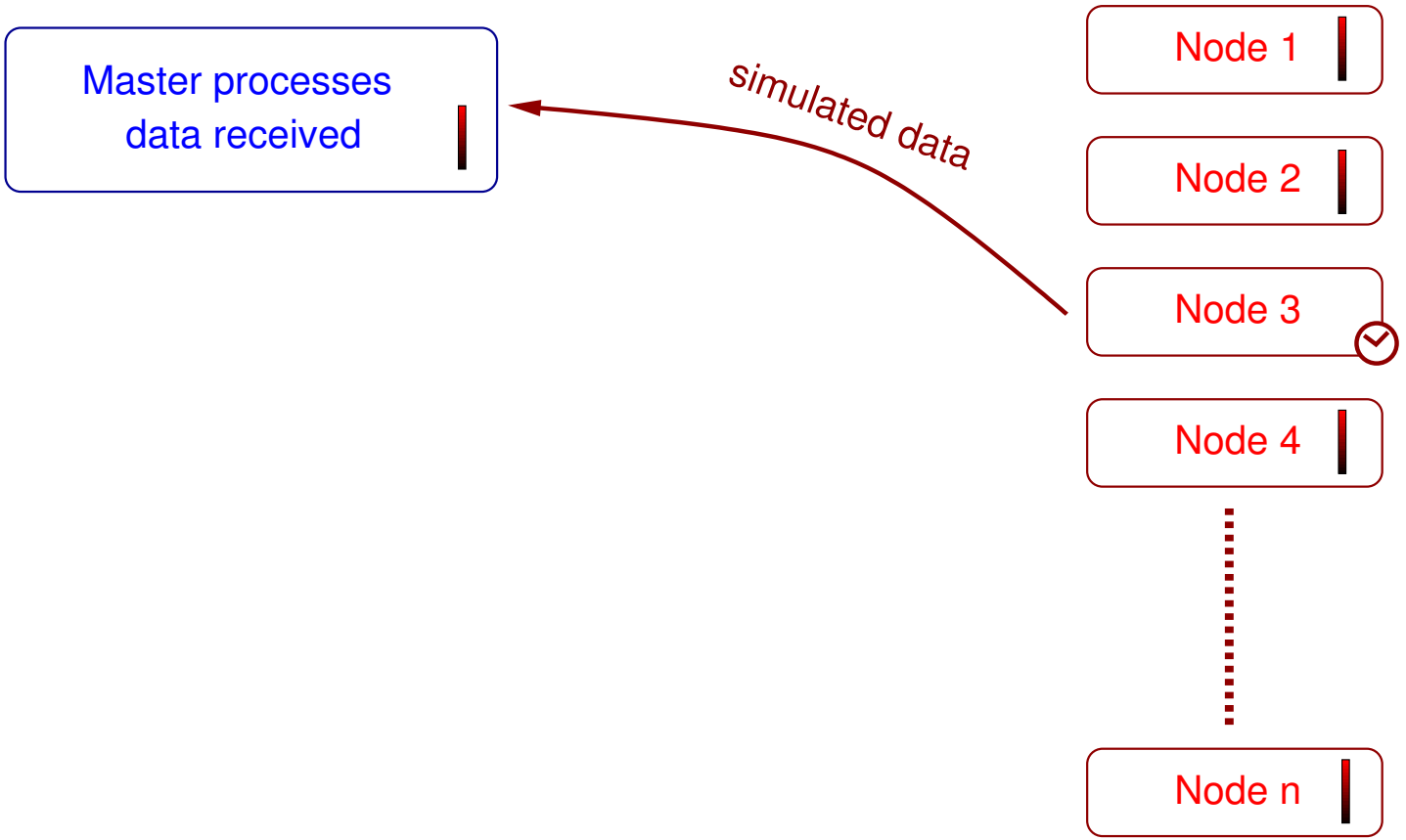


How it works - Waiting & Simulating

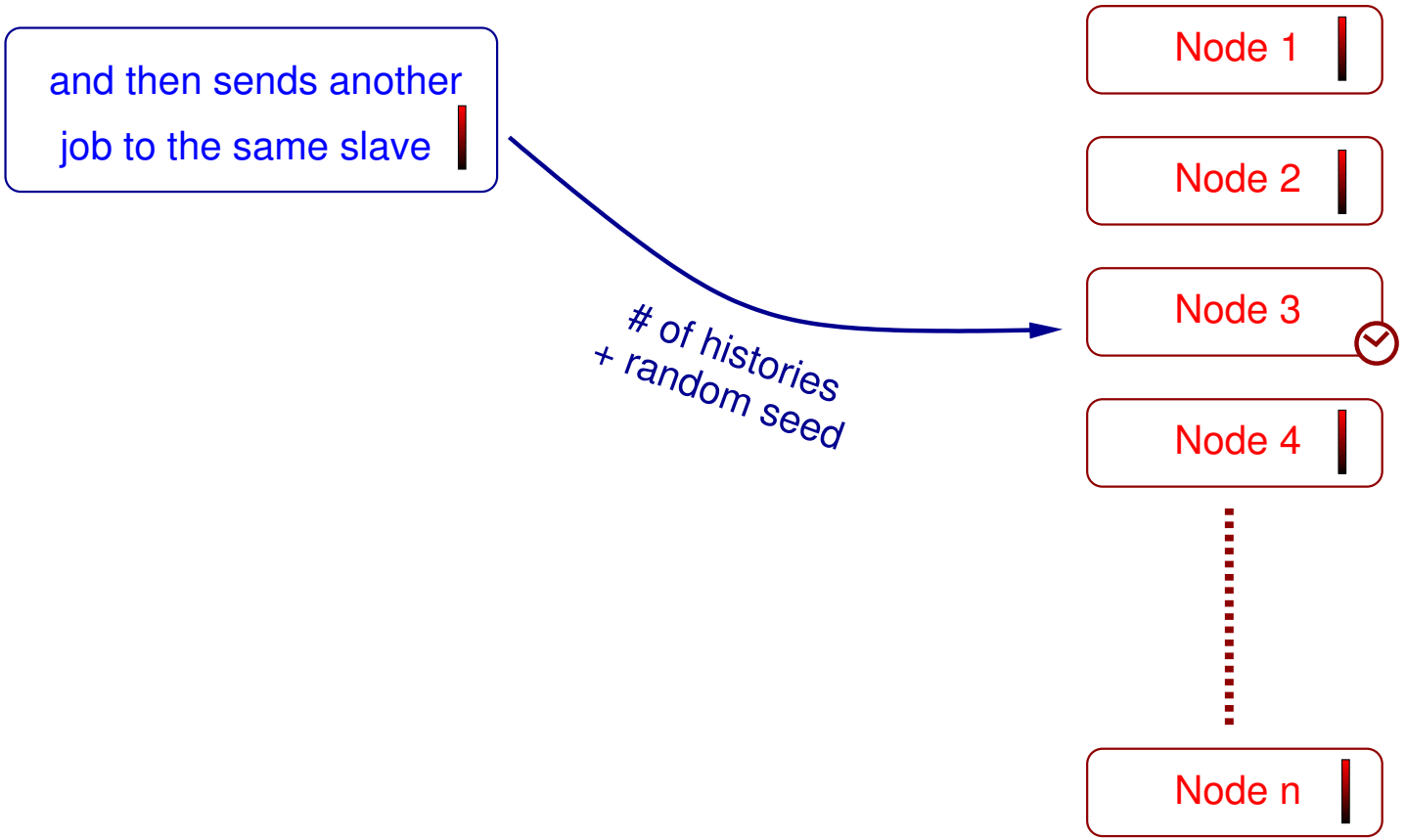
Master waits to receive data from any node



How it works - Getting results



How it works - Sending another job



Some features and limitations

Features

- Easy parallelizing of any G4 application
- Independent of the number of slaves (scalability)
- Master controls the slave's random seeds
- Send/Receive any kind of objects inherited from OOMPI_User_type
- Should work in any Unix like system, even in non homogeneous clusters

Some features and limitations

Features

- Easy parallelizing of any G4 application
- Independent of the number of slaves (scalability)
- Master controls the slave's random seeds
- Send/Receive any kind of objects inherited from OOMPI_User_type
- Should work in any Unix like system, even in non homogeneous clusters

Some features and limitations

Features

- Easy parallelizing of any G4 application
- Independent of the number of slaves (scalability)
- Master controls the slave's random seeds
- Send/Receive any kind of objects inherited from OOMPI_User_type
- Should work in any Unix like system, even in non homogeneous clusters

Some features and limitations

Features

- Easy parallelizing of any G4 application
- Independent of the number of slaves (scalability)
- Master controls the slave's random seeds
- Send/Receive any kind of objects inherited from OOMPI_User_type
- Should work in any Unix like system, even in non homogeneous clusters

Some features and limitations

Features

- Easy parallelizing of any G4 application
- Independent of the number of slaves (scalability)
- Master controls the slave's random seeds
- Send/Receive any kind of objects inherited from OOMPI_User_type
- Should work in any Unix like system, even in non homogeneous clusters

Some features and limitations

Current limitations

- No interactive input by terminal or graphic interface, only macrofile is accepted
- If a node is turned off or disconnected all related processes including master's will crash

Some features and limitations

Current limitations

- No interactive input by terminal or graphic interface, only macrofile is accepted
- If a node is turned off or disconnected all related processes including master's will crash

Outline

Code Changed and Added

Code Changed

- GNUMakefile
- The `main()` function
- Analysis or Tracker(Phantom)SD

Code Added

- ParRunManager
- ParRandomState

Code Changed and Added

Code Changed

- GNUMakefile
- The `main()` function
- Analysis or Tracker(Phantom)SD

Code Added

- ParRunManager
- ParRandomState

GNUMakefile

Added paths for header files and used mpic++ instead of g++

```
EXTRALIBS += -L/usr/local/lib -ltoompi -loompi
CPPFLAGS += -I/usr/local/include/oompi \
            -I/usr/local/include/TOOMPI
...

CXX = mpic++
```

GNUMakefile

Added paths for header files and used mpic++ instead of g++

```
EXTRALIBS += -L/usr/local/lib -ltoompi -loompi
CPPFLAGS += -I/usr/local/include/oompi \
            -I/usr/local/include/TOOMPI
...

CXX = mpic++
```

main()

- Included TOOMPI and OOMPI header files
- Added two code lines in order to initialize and finalize the OOMPI interface properly
- Instantiated ParRunManager instead of G4RunManager

main()

```
#define N_HIST 1000
int main (int argc, char **argv) {
  OOMPI_COMM_WORLD.Init (argc, argv);
  int rank = OOMPI_COMM_WORLD.Rank ();
  int size = OOMPI_COMM_WORLD.Size ();
  ...
  ParRunManager *runManager=new ParRunManager(N_HIST);
  ...
  OOMPI_COMM_WORLD.Finalize();
  return 0;
}
```

main()

```
#define N_HIST 1000
int main (int argc, char **argv) {
  OOMPI_COMM_WORLD.Init (argc, argv);
  int rank = OOMPI_COMM_WORLD.Rank ();
  int size = OOMPI_COMM_WORLD.Size ();
  ...
  ParRunManager *runManager=new ParRunManager(N_HIST);
  ...
  OOMPI_COMM_WORLD.Finalize();
  return 0;
}
```

main()

```
#define N_HIST 1000
int main (int argc, char **argv) {
  OOMPI_COMM_WORLD.Init (argc, argv);
  int rank = OOMPI_COMM_WORLD.Rank ();
  int size = OOMPI_COMM_WORLD.Size ();
  ...
  ParRunManager *runManager=new ParRunManager(N_HIST);
  ...
  OOMPI_COMM_WORLD.Finalize();
  return 0;
}
```

Analysis class

Two new methods were created:

GetTree to get TTree pointer (where are stored all simulated hits and tracks)

InsertNewResults to insert the received TTree object inside the main TTree object

ParRandomState

Class declaration

```
class ParRandomState:virtual public OOMPI_User_type {
public:
  ParRandomState ();
  inline void GetNextRandomStateForSlave (){...}
  inline void SetNextRandomStateForSlave (){...}

private:
  long nextSeed;
  static OOMPI_Datatype type;
};
```

ParRandomState

Inline methods

```
inline void GetNextRandomStateForSlave ()  {
  nextSeed = (long)
    (100000000L*HepRandom::
     getTheGenerator ()->flat ());
}

inline void SetNextRandomStateForSlave ()  {
  HepRandom::setTheSeed (nextSeed);
}
```

ParRandomState

Class implementation

```
OOMPI_Datatype ParRandomState::type;

ParRandomState::ParRandomState ():
  OOMPI_User_type (type, this, 201) {
  if (!type.Built())  {
      type.Struct_start(this);
      type.Entry(nextSeed);
      type.Struct_end();
    }
}
```

ParRunManager

It manages the Geant4 run. Based in the Geant4/TOP-C examples.

This class is also responsible for sending/receiving the objects between master and slave nodes

ParRunManager

It manages the Geant4 run. Based in the Geant4/TOP-C examples.

This class is also responsible for sending/receiving the objects between master and slave nodes

ParRunManager::MasterJob() - Starting loop

Starting all slave nodes

```
do
{
  OOMPI_COMM_WORLD[node].Send(n_events);
  remaining_events -= n_events;

  randomState->GetNextRandomStateForSlave();
  OOMPI_COMM_WORLD[node].Send(*randomState);

  node++;
} while (remaining_events > 0 && node < size);
```

ParRunManager::MasterJob() - Starting loop

Starting all slave nodes

```
do
{
  OOMPI_COMM_WORLD[node].Send(n_events);
  remaining_events -= n_events;

  randomState->GetNextRandomStateForSlave();
  OOMPI_COMM_WORLD[node].Send(*randomState);

  node++;
} while (remaining_events > 0 && node < size);
```

ParRunManager::MasterJob() - Starting loop

Starting all slave nodes

```
do
{
  OOMPI_COMM_WORLD[node].Send(n_events);
  remaining_events -= n_events;

  randomState->GetNextRandomStateForSlave();
  OOMPI_COMM_WORLD[node].Send(*randomState);

  node++;
} while (remaining_events > 0 && node < size);
```

ParRunManager::MasterJob() - Starting loop

Starting all slave nodes

```
do
{
  OOMPI_COMM_WORLD[node].Send(n_events);
  remaining_events -= n_events;

  randomState->GetNextRandomStateForSlave();
  OOMPI_COMM_WORLD[node].Send(*randomState);

  node++;
} while (remaining_events > 0 && node < size);
```

ParRunManager::MasterJob() - Receiving loop

Wait for results from slave

```
msg = new TOOMPI ();
stree = (TTree *) msg->RecvObject ();
gKglAnalysisManager->InsertNewResults (stree);
delete stree;

node = (G4int) msg->Get_source();
```

ParRunManager::MasterJob() - Receiving loop

Wait for results from slave

```
msg = new TOOMPI ();
stree = (TTree *) msg->RecvObject ();
gKglAnalysisManager->InsertNewResults (stree);
delete stree;

node = (G4int) msg->Get_source();
```

ParRunManager::MasterJob() - Receiving loop

Wait for results from slave

```
msg = new TOOMPI ();
stree = (TTree *) msg->RecvObject ();
gKglAnalysisManager->InsertNewResults (stree);
delete stree;

node = (G4int) msg->Get_source();
```

ParRunManager::SlaveJob() - Process a job

It receives, processes histories and then send data to master

```
OOMPI_COMM_WORLD[0].Recv(n_events);
OOMPI_COMM_WORLD[0].Recv(*randomState);
randomState->SetNextRandomStateForSlave();

DoEvent (n_events);

stree = gKglAnalysisManager->GetTree ();
msg = new TOOMPI ();
msg->SendObject (0, stree);
```

ParRunManager::SlaveJob() - Process a job

It receives, processes histories and then send data to master

```
OOMPI_COMM_WORLD[0].Recv(n_events);
OOMPI_COMM_WORLD[0].Recv(*randomState);
randomState->SetNextRandomStateForSlave();

DoEvent (n_events);

stree = gKglAnalysisManager->GetTree ();
msg = new TOOMPI ();
msg->SendObject (0, stree);
```

ParRunManager::SlaveJob() - Process a job

It receives, processes histories and then send data to master

```
OOMPI_COMM_WORLD[0].Recv(n_events);
OOMPI_COMM_WORLD[0].Recv(*randomState);
randomState->SetNextRandomStateForSlave();

DoEvent (n_events);

stree = gKglAnalysisManager->GetTree ();
msg = new TOOMPI ();
msg->SendObject (0, stree);
```

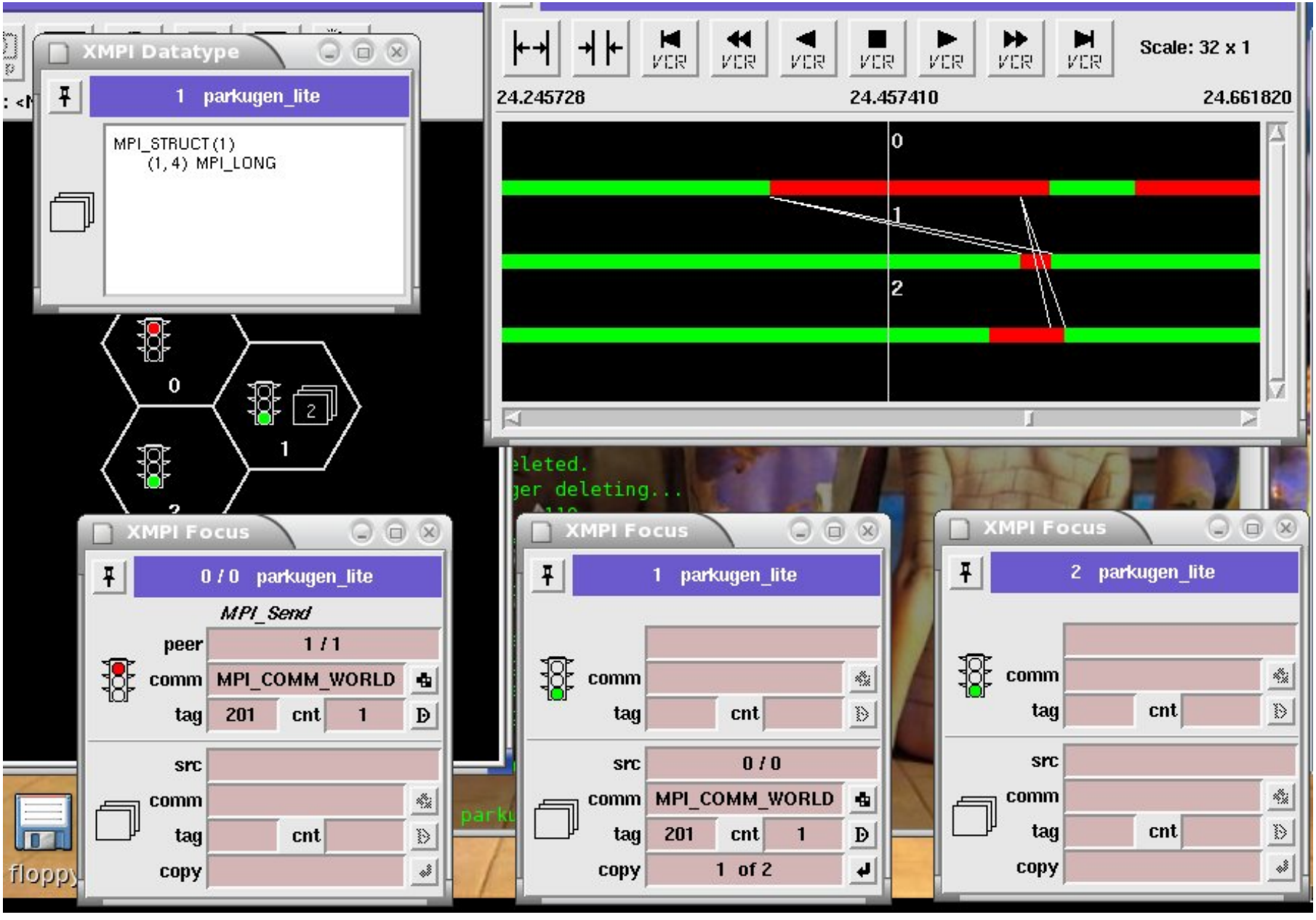
ParRunManager::SlaveJob() - Process a job

It receives, processes histories and then send data to master

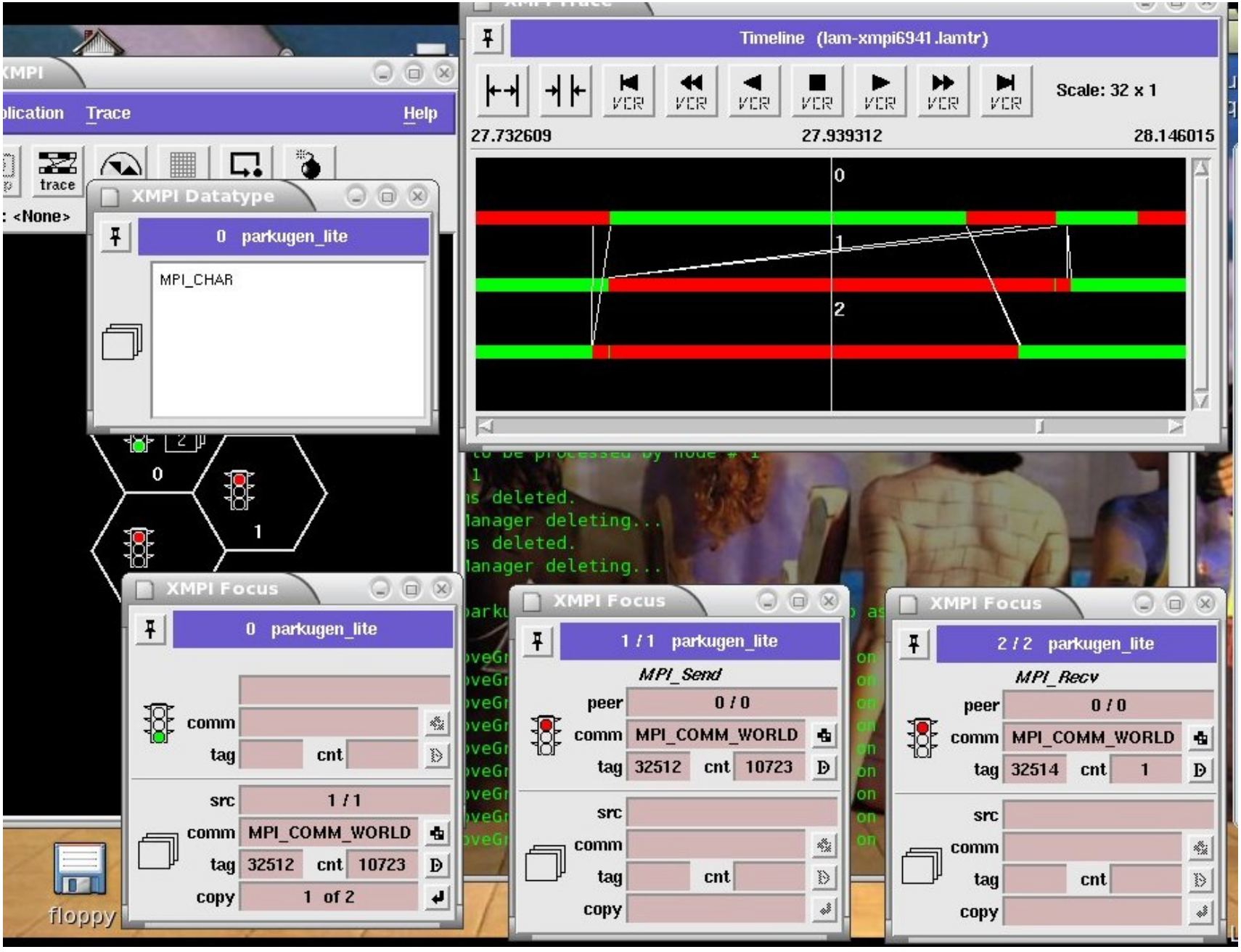
```
OOMPI_COMM_WORLD[0].Recv(n_events);
OOMPI_COMM_WORLD[0].Recv(*randomState);
randomState->SetNextRandomStateForSlave();

DoEvent (n_events);

stree = gKglAnalysisManager->GetTree ();
msg = new TOOMPI ();
msg->SendObject (0, stree);
```

Outline

XMPI view: sending # of histories and random seed



XMPI view: slaves sending simulated data to master



Outline

Useful links

- Geant4: `http://geant4.web.cern.ch/geant4/`
- ROOT: `http://root.cern.ch`
- LAM-MPI: `http://www.lam-mpi.org/`
- OOMPI: `http://www.osl.iu.edu/research/oompi/`
- TOOMPI: `http://dfm.ffclrp.usp.br/perles`

Future improvements

- Re-write it
- Make an easy send/receive G4THitsCollection possible
- Interactive terminal/graphics mode
- Load profile of every node in order to reduce the latency

Future improvements

- Re-write it
- Make an easy send/receive G4THitsCollection possible
- Interactive terminal/graphics mode
- Load profile of every node in order to reduce the latency

Future improvements

- Re-write it
- Make an easy send/receive G4THitsCollection possible
- Interactive terminal/graphics mode
- Load profile of every node in order to reduce the latency

Future improvements

- Re-write it
- Make an easy send/receive G4THitsCollection possible
- Interactive terminal/graphics mode
- Load profile of every node in order to reduce the latency

Acknowledgements

- CNPq by scholarship and support
- MCNEG and NPL people by the travel bursary
- My tutor Prof. Dr. Adelaide de Almeida

Contact

Luís Augusto Perles
e-mail: perles@dfm.ffclrp.usp.br
homepage: http://dfm.ffclrp.usp.br/perles