TauSpinner update and future plans for Tauola++

Tomasz Przedziński, Jagellonian University in Kraków

Plan:

- Status of Photos++
- TauSpinner update
  - Handling top-loop-mediated processes
  - New functionality
- Future plans for Tauola++
  - Substituting existing currents
  - Adding new decay channels
- Parallelization using message queues
  - When Inter-Process Communication becomes useful?
  - Why message queues?
  - Different uses of message queues

Photos++ update

Photos++

- New version (not yet available online); fully in C/C++
- Fully within `namespace Photospp`
  - no more symbol collision with older versions of Photos
  - new libraries will now properly replace older ones
- Can be prepared for parallel computation
  - this is now trivial step but requires extensive testing
  - please let us know if there is demand/use for such setup!
- Added example of using Photos++ with `FORTRAN` code
  - interface to HEPEVT
  - introducing NLO corrections for W and Z to Fortran projects
  - will be thoroughly tested with new KKMC version mentioned by S. Jadach

Photos++

- Photos and Photos++ (Photos v3.0 – v3.52) available for LHC through GENSER project
  - /afs/cern.ch/sw/lcg/external/MCGenerators/photos/
  - /afs/cern.ch/sw/lcg/external/MCGenerators/photos++/
- Development versions available from:
  - http://www.ph.unimelb.edu.au/~ndavidson/photos/doxygen/index.html
  - http://annapurna.ifj.edu.pl/~tprzedzinski/photos/ (mirror)

Main Page | Namespaces | Data Structures | Files | Directories

C++ Interface to PHOTOS

Description of PHOTOS Interface in C++

Authors:

Nadia Davidson, Tomasz Przedzinski, Zbigniew Was

New release

The source code and documentation for release 3.52. The following files are provided for download:

- Photos_interface_design.pdf full software documentation.
- PHOTOS 3.52 source code tarball (version for LHC/LCG installation) and its revision info SVN tag, tarball creation date/time, etc. For updates with respect to release 3.0 see changelog.txt

Developement version

TauSpinner update

TauSpinner

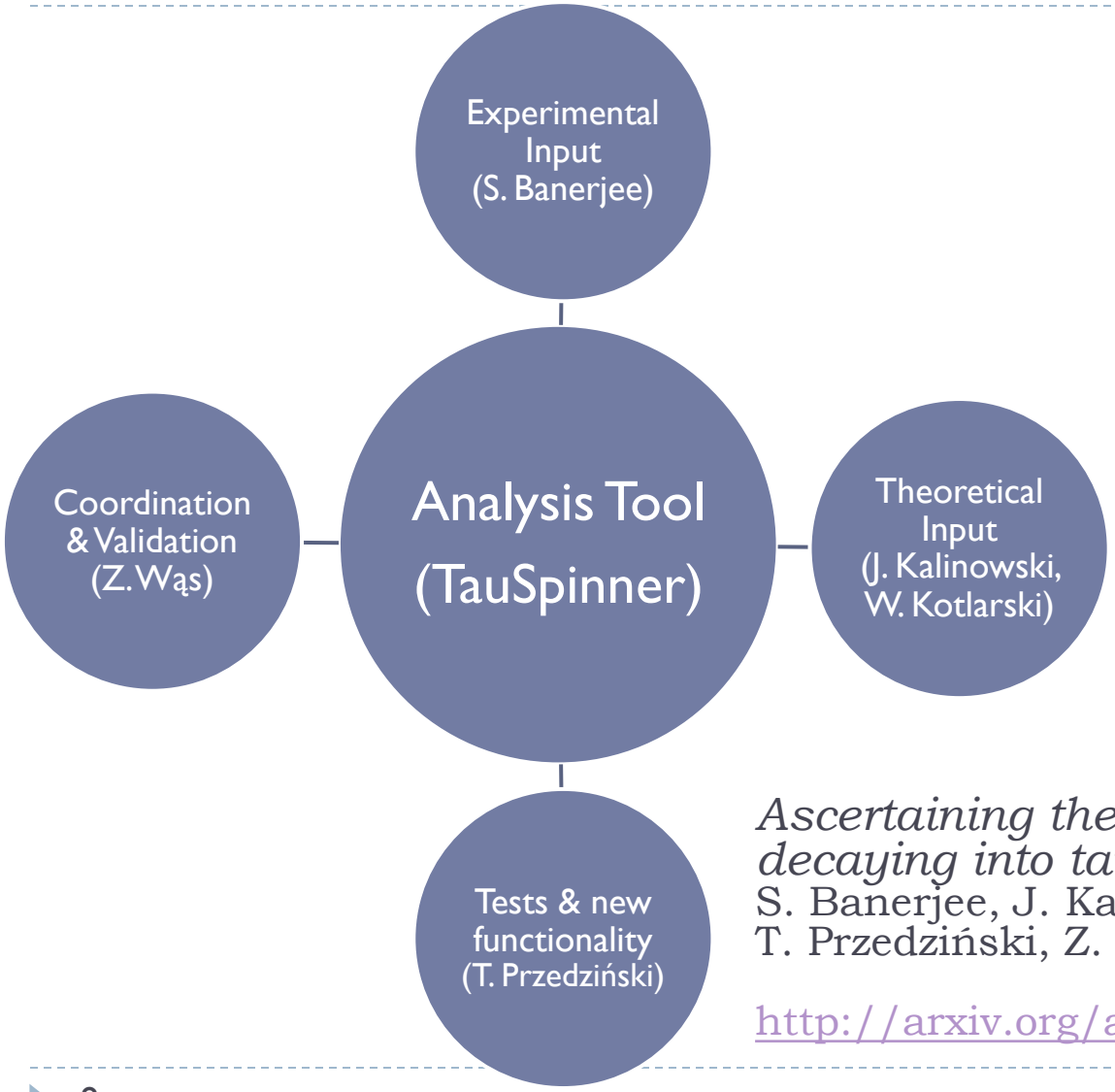
```
int main() {

  // Initialize Tauola
  Tauola::initialize();

  // Initialize TauSpinner
  initialize_spinner(Ipp, Ipol, CMSENE);

  // Begin event loop
  for(...) {

    // SimpleParticle consist of 4 momentum and PDGid
    SimpleParticle X, tau, tau2;
    vector<SimpleParticle> tau_daughters, tau_daughters2;

    // Read event from input_file
    readParticlesFromHepMC( ... );

    // Calculate weight
    if( abs(X.pdgid())==24 || abs(X.pdgid())==37 )
    {
      WT = calculateWeightFromParticlesWorHpn( ... );
    }
    else if( X.pdgid()==25 || X.pdgid()==36 ||
             X.pdgid()==22 || X.pdgid()==23 )
    {
      WT = calculateWeightFromParticlesH( ... );
    }
}
```

Highlighted function fills tau decay information from HepMC::IO_GenEvent data file.

Similar function can be written for any other source

Details on the webpage and in example distributed with the code.

TauSpinner



Ascertaining the spin for new resonances decaying into tau+ tau- at Hadron Colliders
S. Banerjee, J. Kalinowski, W. Kotlarski, T. Przedziński, Z. Wąs; December 2012

http://arxiv.org/abs/1212.2873

TauSpinner update

- **The purpose and the algorithm described by Z. Wąs during his previous talks.** Here only few technical details of the update

- Two new initialization options - `nonSM2` and `nonSMN`
  - `nonSM2` - turns on non-Standard Model weight calculation
  - `nonSMN` - combined with `nonSM2`, allows for calculation of corrections to shape only

- New function for calculating weights for new models in top-loop-mediated processes ($gg \rightarrow \tau^+\tau^-$)

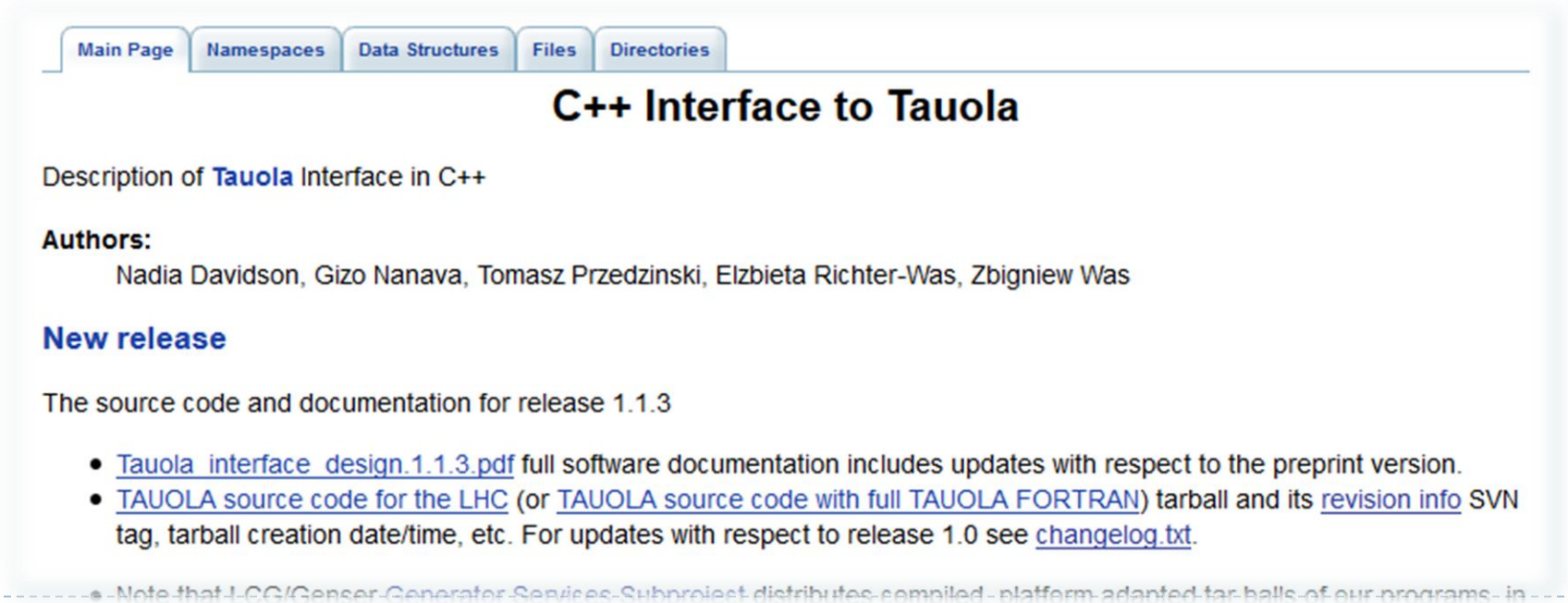
```
...

// Set functions for user-defined born
// including new physics
set_nonSM_born ( nonSM_adopt  );
set_nonSM_bornH( nonSM_adoptH );

...
```

TauSpinner update

- New functions:
  - `set_nonSM_bornH( double (*fun)(...) )`
    Sets function for user-defined Higgs born, including new physics. The parameters of this new function are described in example program, as well as in `include/TauSpinner/nonSM.h`
  - `void setRelWTnonSM(int relWTnonSM)`
    flag for calculating relative(NONSM-SM)/absolute weight for cross-section calculated as by product in longitudinal polarization method
  - `void setHiggsParameters(int jak, double mass, double width, double normalization)`
    Activates simple formula for Higgs calculation and sets Higgs mass, width and normalization for Higgs born (BW) function
  - `double getWtamplitP()`
    `double getWtamplitM()`
    Returns amplitude weight for τ^+ and τ^- respectively

Tauola++ with TauSpinner

- Since Tauola++ v1.1.0, TauSpinner is part of Tauola++ distribution
  - Compiled as separate library (needs `LHAPDF` path during configuration)
  - LHAPDF can be substituted by user PDF function
    See `TauSpinner/src/tau_reweight_lib.cxx`, function:
    `double f(double x, int ID, double SS, double cmsene)`



Main Page | Namespaces | Data Structures | Files | Directories

C++ Interface to Tauola

Description of **Tauola** Interface in C++

Authors:

Nadia Davidson, Gizo Nanava, Tomasz Przedzinski, Elzbieta Richter-Was, Zbigniew Was

New release

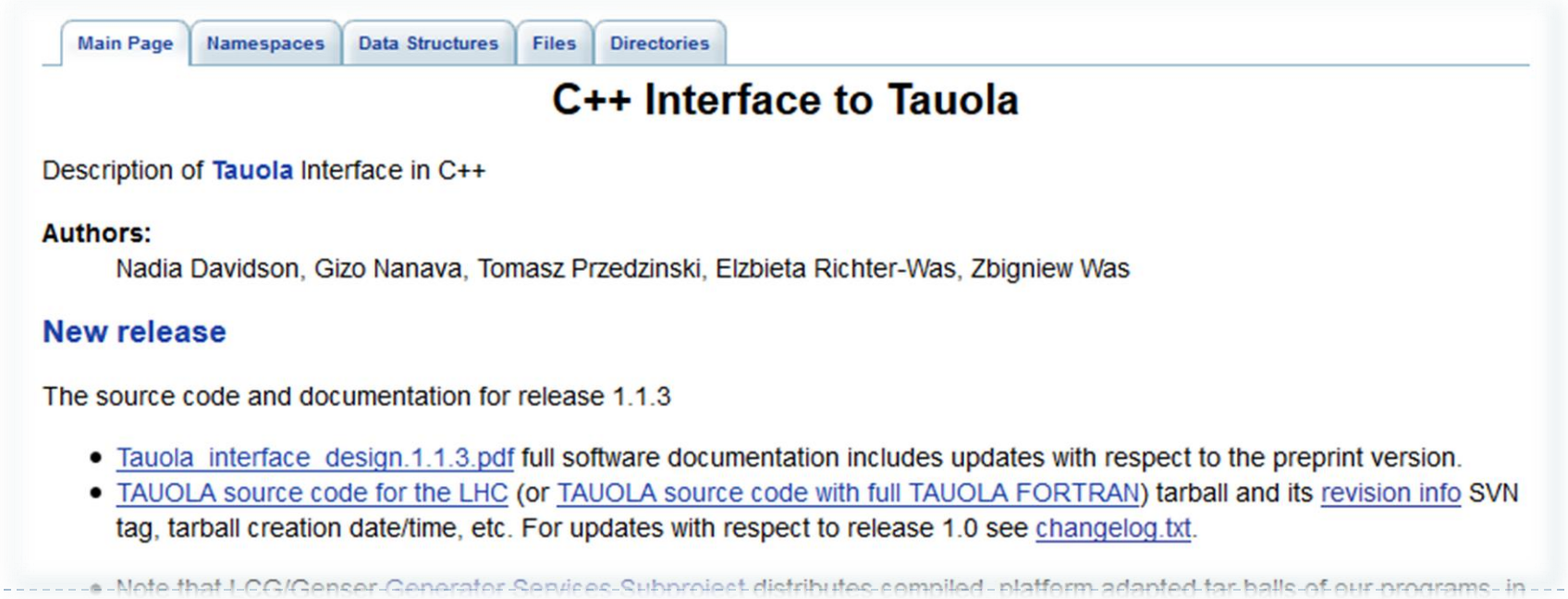
The source code and documentation for release 1.1.3

- Tauola_interface_design.1.1.3.pdf full software documentation includes updates with respect to the preprint version.
- TAUOLA source code for the LHC (or TAUOLA source code with full TAUOLA FORTRAN) tarball and its revision info SVN tag, tarball creation date/time, etc. For updates with respect to release 1.0 see changelog.txt.

- Note that I-CG/Genser Generator Services Subproject distributes compiled - platform adapted tar balls of our programs, in

Tauola++ with TauSpinner

- Available for LHC through GENSER project
  - /afs/cern.ch/sw/lcg/external/MCGenerators/tauola++/
- Development version and stable releases available from:
  - http://www.ph.unimelb.edu.au/~ndavidson/tauola/doxygen/index.html
  - http://annapurna.ifj.edu.pl/~tprzedzinski/tauola/index.html (mirror)
  - Doxygen documentation serves as quick reference for looking up functions and parameters



Future plans for Tauola++

Future plans for Tauola++

- In near future, from technical point of view, we would like to address few requests made by our users:
  - problem with symbol collision between Tauola++ and `TAUOLA-FORTRAN`; problems when using pythia6 and Tauola++ simultaneously
  - ability to change currents for existing decay modes
  - option of adding new decay channels (see talk of Olga, Pablo etc.)
  - option of swapping and comparing different models of tau decays

- We would like to use this workshop as an opportunity to ask you for new ideas on how to improve Tauola++
  - what should we focus on?
  - what other functionality Tauola++ is missing?
  - would Tauola++ benefit from fully rewriting it to C++?

Future plans for Tauola++

Example of new functionality

```
class rchl3piCurrent: public Tauola::DecayMode
{
public:
  New3piCurrent() { ... }
public:
  // Implementation of Tauola::DecayMode

  // Formfactors
  complex<double> f1( ... );
  complex<double> f2( ... );
  complex<double> f3( ... );
  complex<double> f4( ... );
  complex<double> f5( ... );

  // Decay products PDG IDs
  vector<int> getDecayProducts();

  // Default branching ratio
  double getBranchingRatio();

  // Initialization
  void initialize();
private:
  ...
};
```

- User will be able to define new decay modes as separate classes
- Then, from the main program:

```
int main()
{
  ...
  Tauola::DecayMode *dm =
        new rchl3PiCurrent();
  Tauola::setDecayMode(5,dm);

  // -- OR --

  Tauola::addDecayMode(dm);

  Tauola::initialize();

  ...
}
```

Future plans for Tauola++

Example of new functionality

```
...
Tauola::DecayModeConfiguration config1;

config1.Add(dm1,5); // substitute a1 channel
config1.Add(dm2,4); // substitute rho channel
config1.Add(dm3)    // add new decay channel
...
```

- Comparing different models
  - creating decay mode configurations
  - switching configurations during generation

```
for(...)
{
  // Decay using 1st configuration
  Tauola::setDecayModeConfiguration(config1);

  evt.undecayTaus();
  evt.decayTaus();

  Analyze(evt);

  // Decay using 2nd configuration
  Tauola::setDecayModeConfiguration(config2);

  evt.undecayTaus();
  evt.decayTaus();

  Analyze(evt);
}
```

Future plans for Tauola++

Example of new functionality

- Using this approach, both C++ and FORTRAN currents can be used

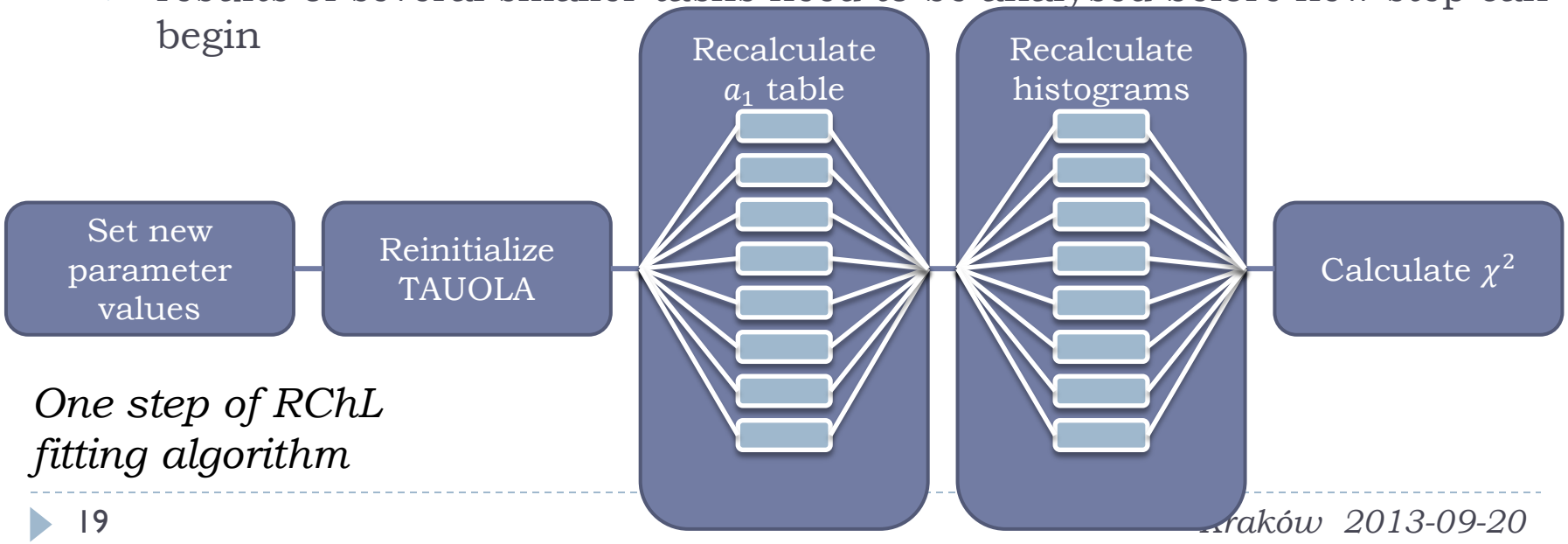
```
// Fortran function declaration
extern "C" f3pi_rcht_(...);

complex<double> rchl3piCurrent::f1(...)
{
  return f3pi_rcht_(1,...);
}
```

- All currents available so far as an option set at compile time of TAUOLA-FORTRAN will be made available through this mechanism
- Manpower problems on collaboration with Belle, BaBar (also F77 issues)
- We welcome any new ideas for other useful extensions!

Parallelization using Message Queues

Parallelization – when IPC becomes useful

- High Energy Physics simulations are easily parallelized
  Especially MC simulations
  - In most cases, each event can be treated independently; batches of events can be generated or analysed simultaneously as separate jobs
  - Usually, there is no need to parallelize computation of a single event/job
- When parallelization of a single job becomes useful?
  - time-consuming computation of a single step of the analysis
  - results of several smaller tasks need to be analysed before new step can begin



One step of RChL fitting algorithm

Parallelization using OpenMP

- OpenMP
  - Can easily parallelize small pieces of code
  - Supports C/C++ (through `#pragma` directives) and Fortran (through comments)
  - Very efficient on a single machine
  - Can be incrementally implemented into existing projects
- Disadvantages (seen from the point of view of physics simulations)
  - Only few basic concepts are easy to learn. Rest is significantly harder
  - Requires at least basic knowledge about synchronization and thread-safe code
  - Advanced functionality requires extensive knowledge about memory sharing and how to deal with synchronization problems
  - Requires external libraries and compilers compatible with OpenMP

```
int main()
{
  const int N = 100000;
  int i, a[N];

  #pragma omp parallel for
  for(i=0;i<N;i++)
    a[i]=i*i;

  return 0;
}
```

Parallelization using message queues

- Inter-Process Communication (IPC) using message queues
  - Full functionality through three system functions: `msgctl`, `msgsnd`, `msgget`
  - Can be used for synchronous or asynchronous communication
  - Can be used for communication between different programs (especially useful where threads cannot be used)
  - Managed by operating system
  - Handled by all POSIX-compatible systems (most Linux distributions, OS X, FreeBSD, Solaris, etc.) without the need for external libraries, compilers or compiler flags
  - In most cases, limited to a single machine
  - Not as efficient as other methods
  - Use in Fortran through C wrappers

```
void code_for_child_process()
{
  message m;

  while(true)
  {
    // Grab first message
    // from the queue
    mq.receive(m);

    // Calculate...
    m.y = m.x*m.x;

    // Send result
    mq.send(m);
  }
}
```

Parallelization using message queues

- Simple example:

```
void code_for_child_process()
{
  while(true)
  {
    get_message()
    compute()
    send_result()
  }
}
```

```
int main()
{
  create_message_queue()
  create_computing_nodes()

  while(true)
  {
    send_jobs_to_queue()
    gather_results()
    analyze_results()
  }

  remove_computing_nodes()
  remove_message_queue()
}
```

- Main program creates computing nodes and sends jobs to message queue
- Computing nodes receive messages and send back the results
- The results are then analysed by the main program and the next iteration can be run

Parallelization using message queues

- Keep in mind:
  - In this example, messages are received by first node that is free. It can be easily changed to send specific jobs to specific nodes and create nodes for different tasks
  - The results are returned in order of the job competition. If needed, the main program can be easily modified to receive messages in specific order
  - This example uses synchronous operations (nodes wait for new message to arrive). It can be easily changed to perform asynchronous operations (nodes check if there are new messages and if not – compute something else)

```
Sending: 1
Sending: 2
...
Sending: 99
Sending: 100

Received: 1
Received: 2
...
Received: 56
Received: 57
Received: 3
Received: 4
Received: 58
Received: 59
...
```

Parallelization using message queues

- Is it worth learning?
  - that really depends on the task at hand. There is usually no need to parallelize MC simulations. However, for most of the other analysis it saves a lot of time
  - in case of fitting RChL currents to the data, by using 8-core machines we gain a factor of 7-7.5
  - fits using `minuit` are very hard to parallelize using separate jobs
  - in chain, where results of next step depend on the analysis of the previous one, this approach allows to set up several iterations at once

- For more details on how to use message queues, see example attached to the conference materials
  - Example contains two useful classes for easy handling of message queues and creating computing nodes

Thank you for listening!