Status and plans for analysis framework Bean

Yury Nefedov, Eugeny Boger

JINR Dubna

Physics & Software Workshop 2011

Motivation

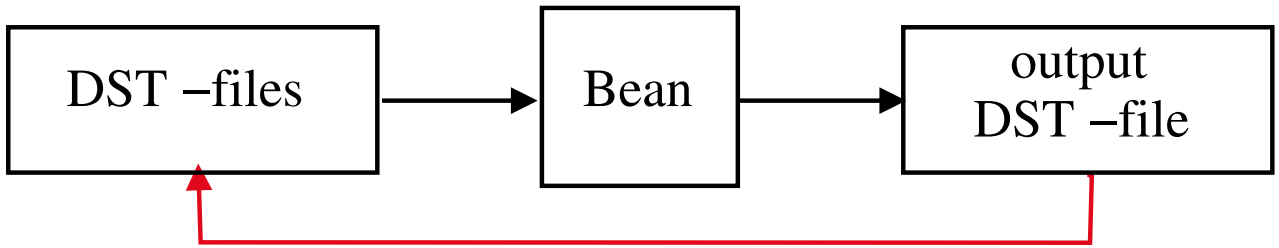
We wanted to create lightweight tools for analysis of reconstructed event's data (DST)

- It should be optimized for standalone use
- It should be fast
- It should depend on the minimum number of external libraries

The main tasks for this framework

- iterative event filtration
- analysis code development
- physical analysis

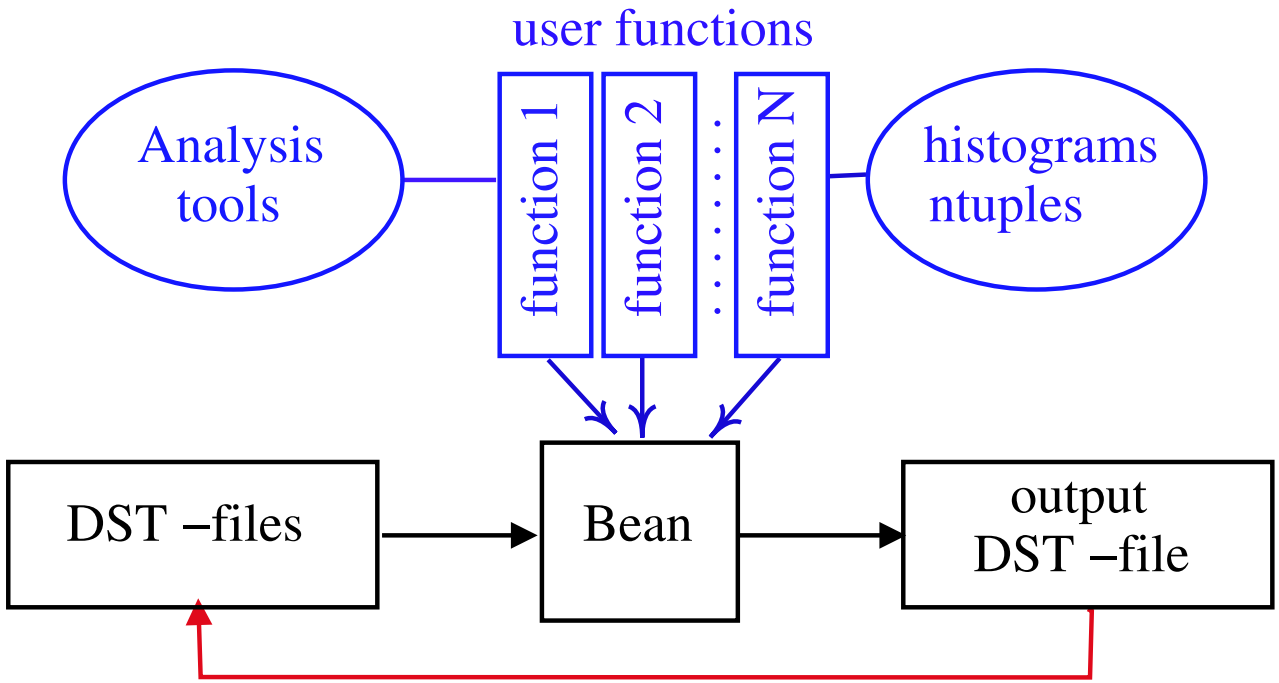
Bean — ROOT-based analysis framework



The main data flow

- RootEventData classes are used for data description
- Input format == Output format == BESIII DST
- User functions are loaded dynamically to the Bean in runtime

Bean — ROOT-based analysis framework



User functions: «flow of commands»

- The plug-in mechanism for switching on
- Several different packages of analysis adapted from the BOSS
- User histograms and ntuples are saved in output root-file.

Analysis tools

List of available packages

- Particle ID
- Kinematic Fit & Vertex Fit
  - ↪ Magnetic Field
- Database interface: (DatabaseSvc on base of SQLite)
- EventTag
- AbsCor (PhotonCor/AbsCor)

External libraries

CLHEP

- Analysis tools packages heavily use the CLHEP
- CLHEP must be installed
- We have prepared a makefile to download and install CLHEP on the local PC (see `doc/Makefile_clhep`)

Sqlite

- We have included the source code of the SQLite 3.7.5 in the Analysis/DatabaseSvc package
- If someone needs to use a different version, should change makefile in this package.

Examples of user analysis

All examples are written in the form of the user functions

List of examples in `BeanUser/`

- UserTest, User1 — are the demo examples how to write user functions
- TestPID, MagField, TestDb, TestEventTag — are the testing functions for the corresponding analysis packages
- Rhopi — is the RhopiAlg algorithm accommodated for Bean
- Bhabha — is the program for the selection of Bhabha electrons
- Etaetagamma — is the program for the selection of $\gamma\,\pi^0\pi^0$ and $\gamma\,\eta\eta$ events

Bean user interface

Synopsis

bean.exe [-option(s)] dst_file(s)

- dst_file(s) – you must specify input DST file(s)
- -u UserFunction – the name of user function

you can create several user functions and use them in a chain

bean.exe -u UserTest -u User1 . . .

- -h hst_file – file name for output histograms

bean.exe -u UserTest -h histo.root file.dst

- -o out_dst – output DST file name

bean.exe -u UserTest -h histo.root -o mydst.root file.dst

Bean user interface

bean.exe without arguments

prints out short information about available options

Short list of options:

- -N num – process first "num" events
- -D – detailed printout of content of each DST event.
- -v – set verbosity on. Causes Bean to print debugging messages about its progress.

Bean documentation

Local files

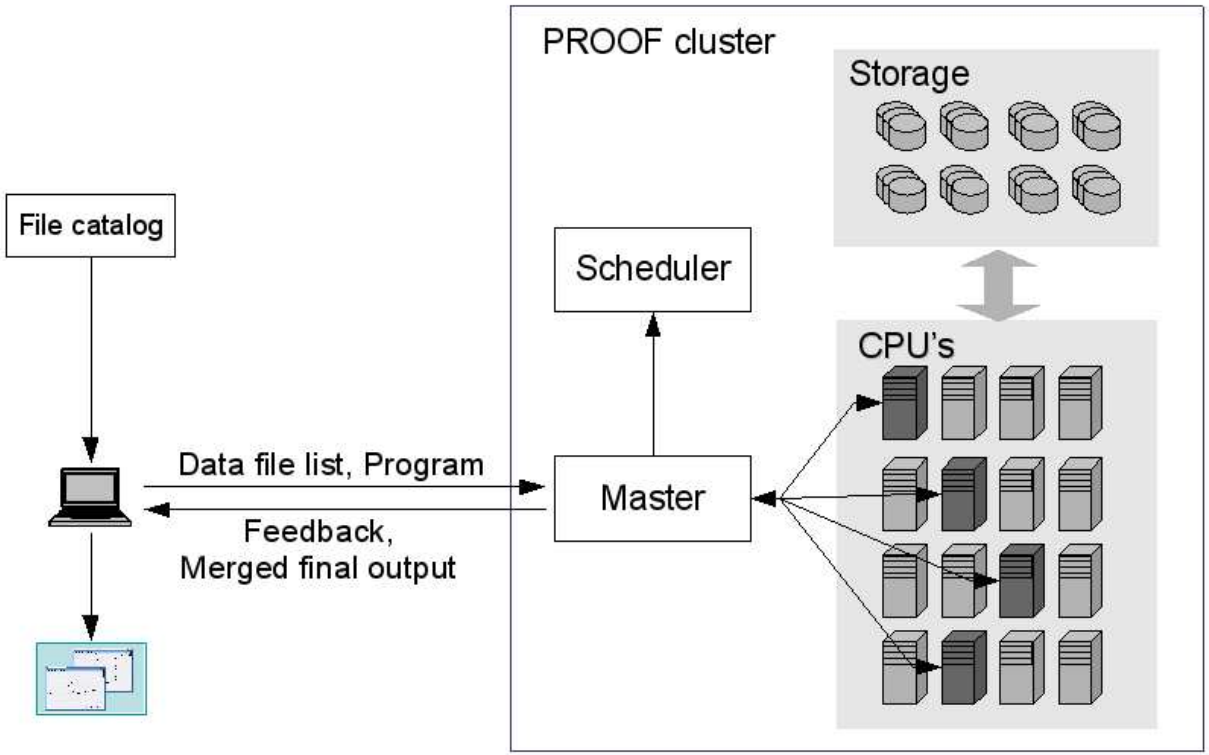
- `README, INSTALL` – short instructions on Bean installation
- `doc/` – presentations conserning Bean and some auxiliary scripts
- `BeanUser/` – examples of analysis

Wiki

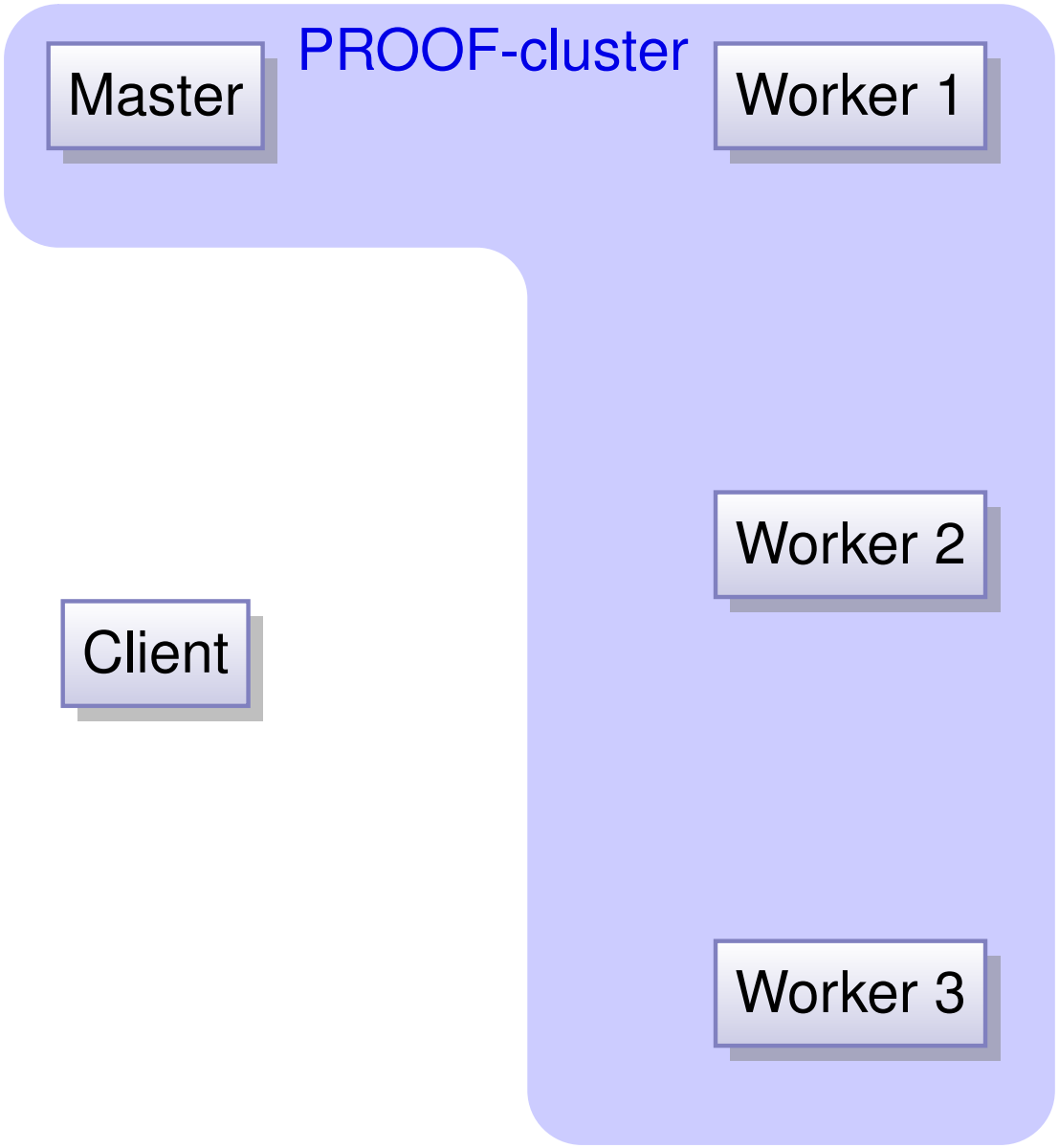
- http://docbes3.ihep.ac.cn/~offlinesoftware/index.php/BEAN
- `http://bes3.jinr.ru/bean/wiki` – is obsolete

Bean parallelization with PROOF

- The PROOF is a part of the ROOT enabling an analysis of large sets of ROOT files in parallel on clusters of computers or many-core machines
- The main idea of Bean is to run on the PROOF system with minimal changes in the user interface



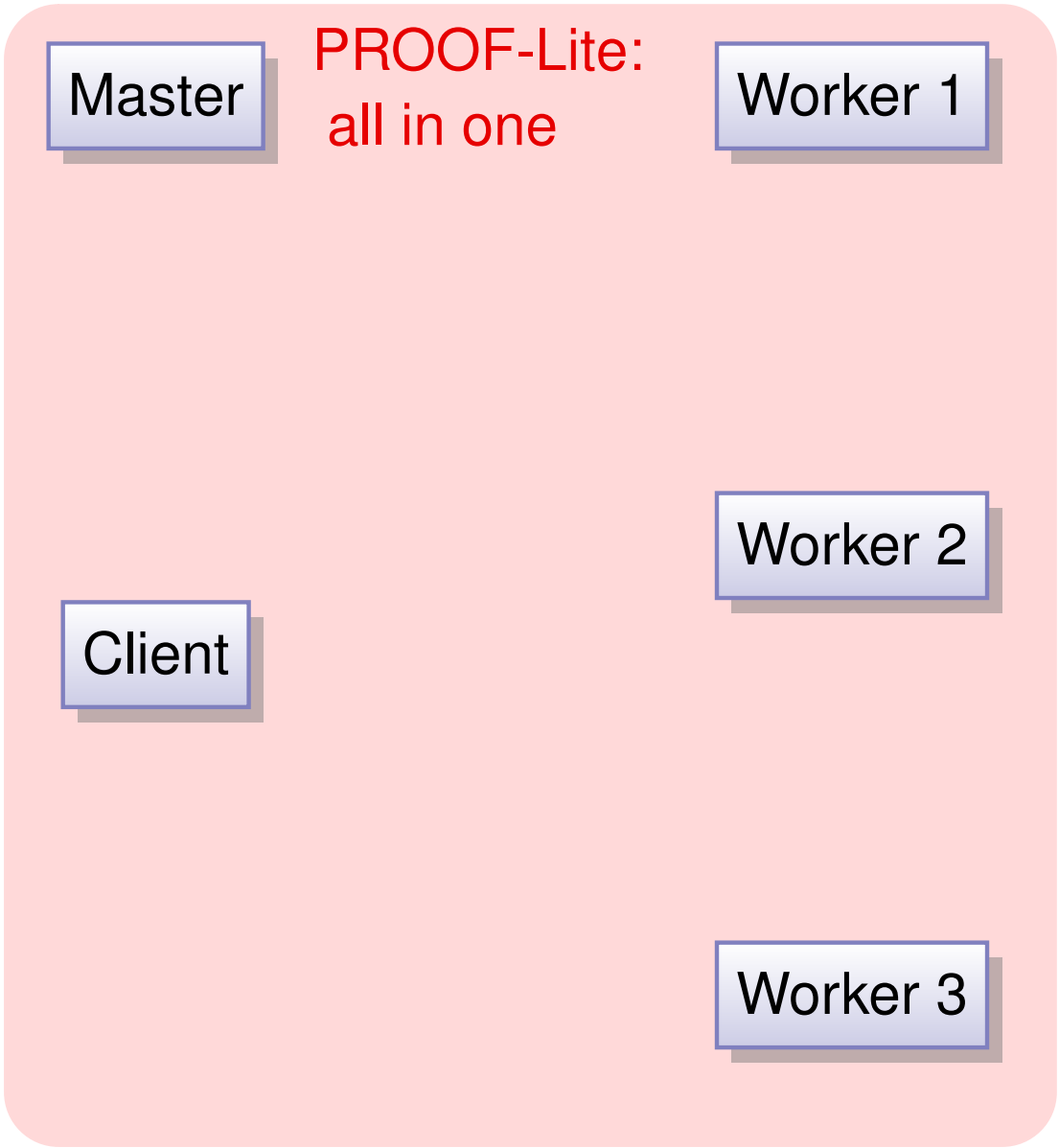
PROOF terminology



Terminology

- **Client:**
  Your machine running a ROOT session that is connected to a Master
- **Master:**
  PROOF machine coordinating work between Workers
- **Worker:**
  PROOF machine that processes data
- **PROOF-Lite:**
  Client, Master and Workers are one multicore / multiprocessor PC.

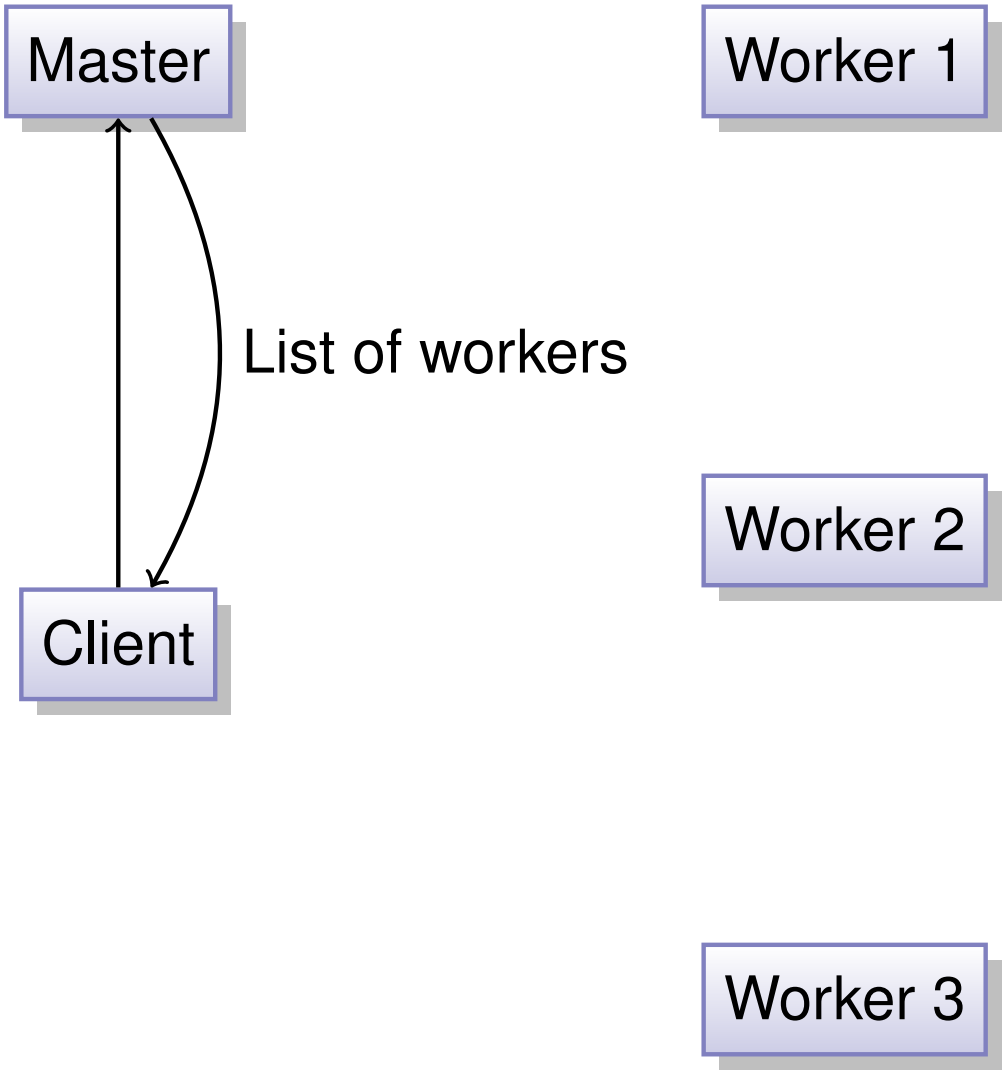
PROOF terminology



Terminology

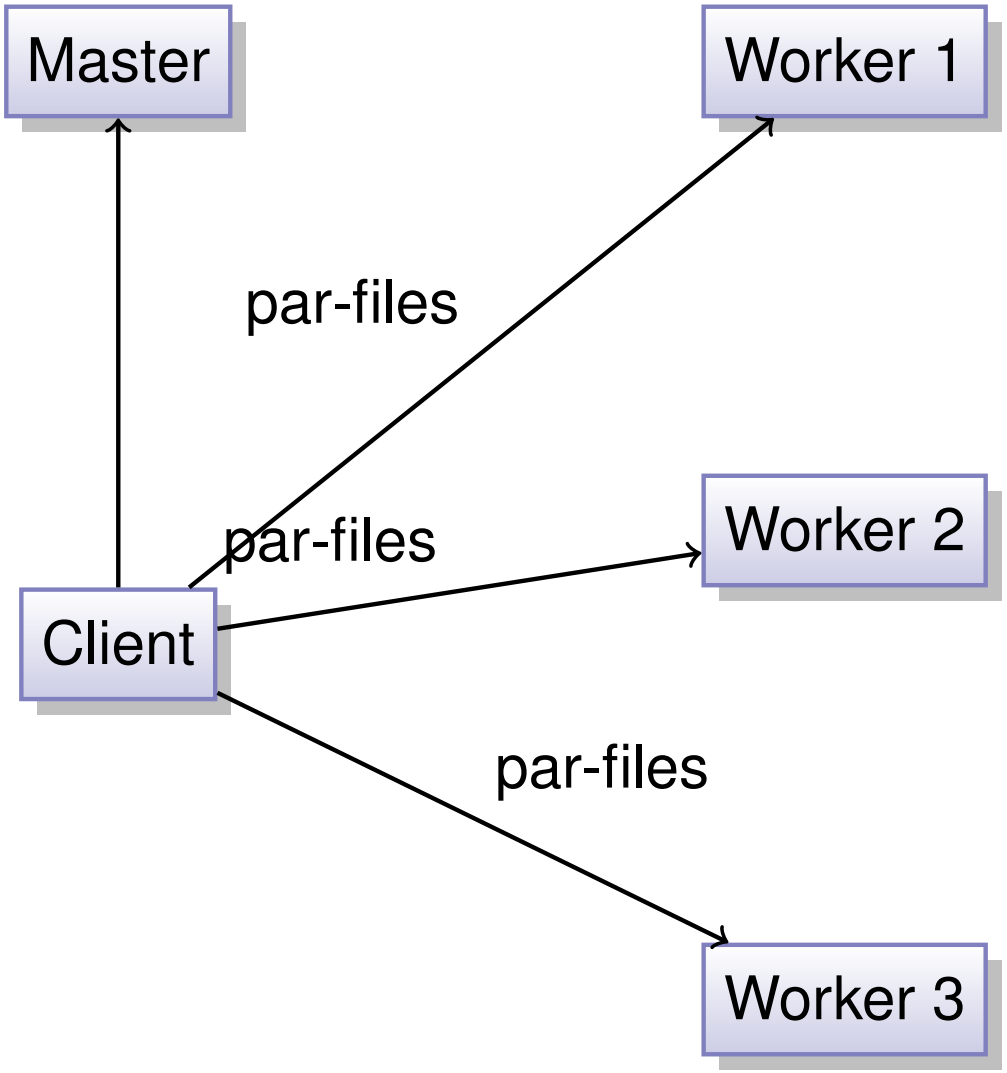
- **Client:**
  Your machine running a ROOT session that is connected to a Master
- **Master:**
  PROOF machine coordinating work between Workers
- **Worker:**
  PROOF machine that processes data
- **PROOF-Lite:**
  Client, Master and Workers are one multicore / multiprocessor PC.

PROOF animation



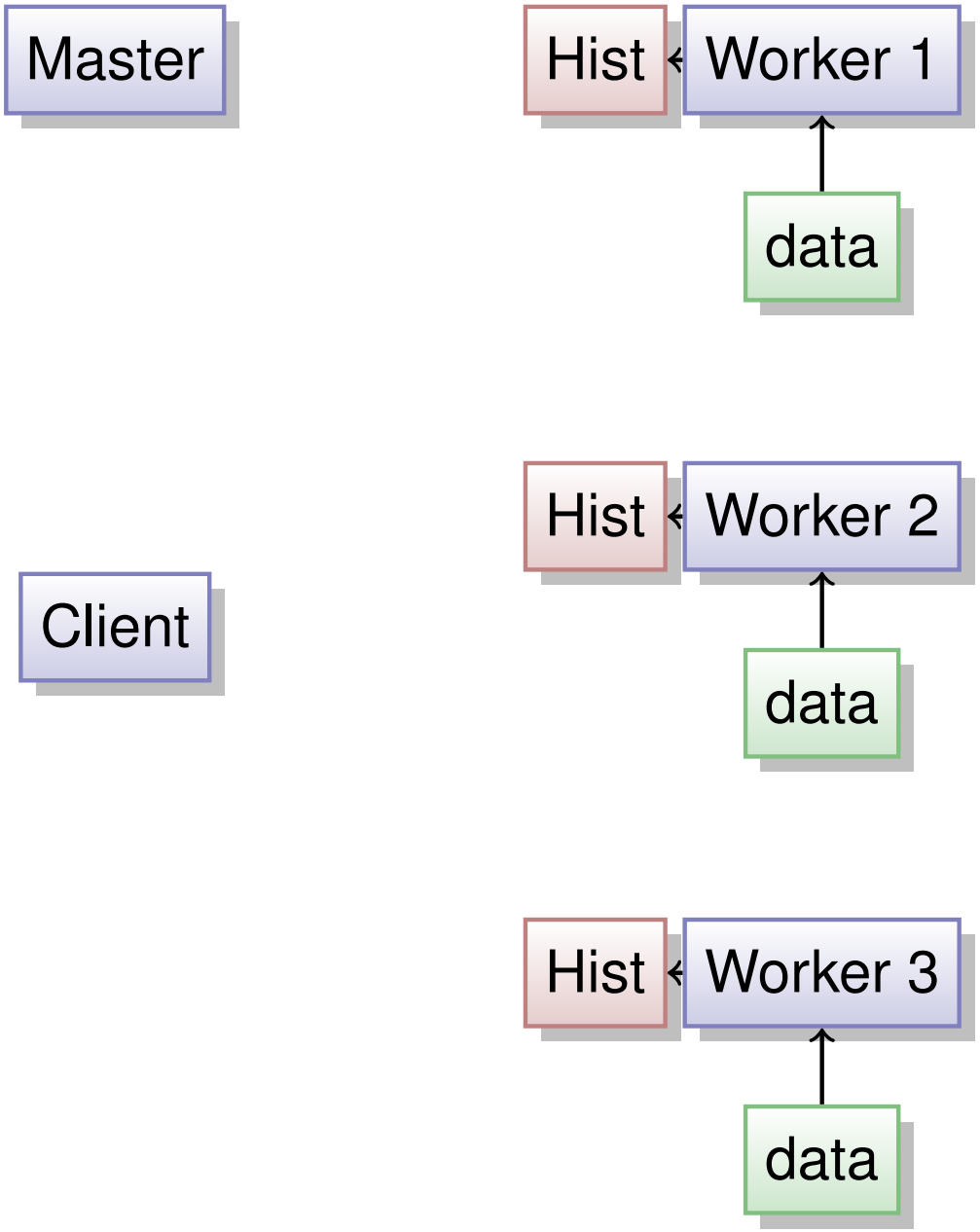
- The client submits a request for Master and gets a list of Workers
- The client sends PAR-packages to all Workers; source code are compiled on each machine
- Workers process the data and fill histograms
- Each Worker sends histograms to Master; Master merges them and sends the result to the client.

PROOF animation



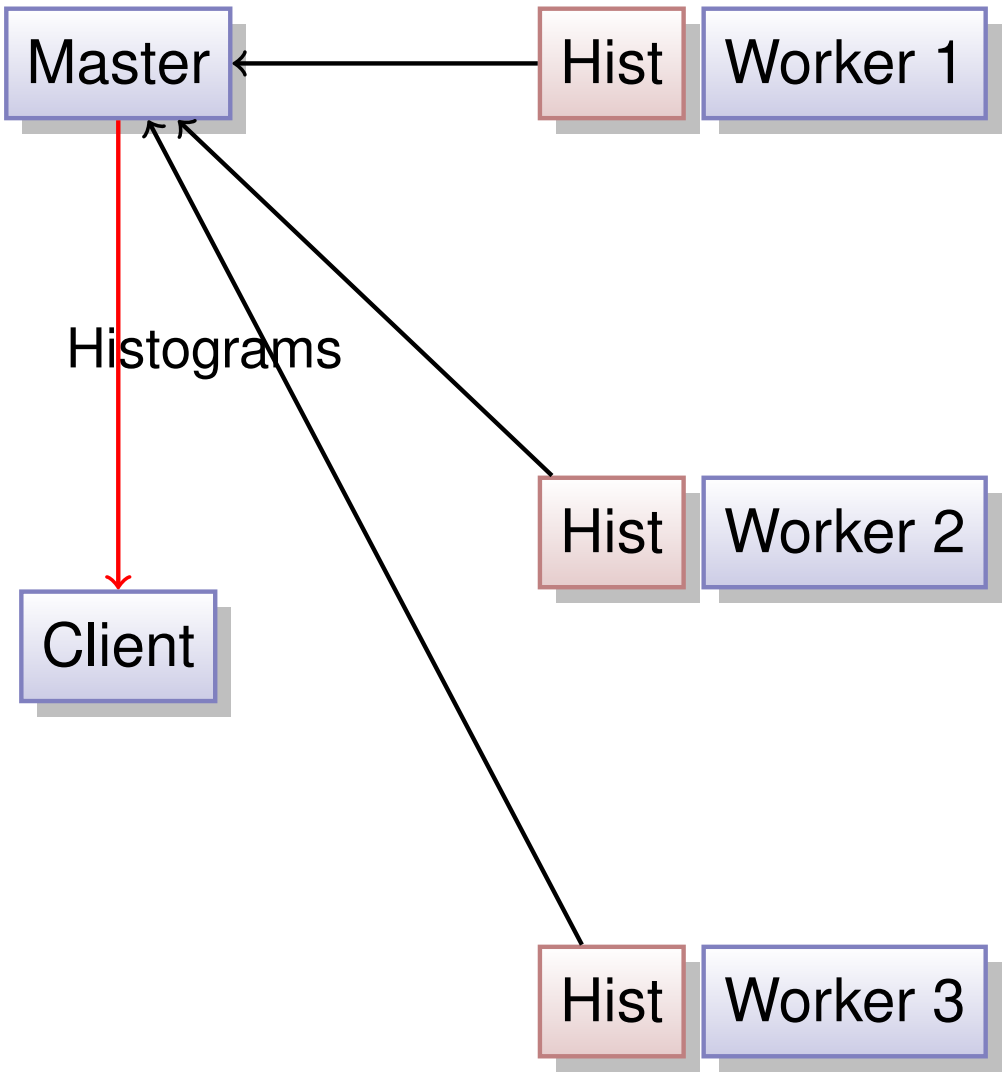
- The client submits a request for Master and gets a list of Workers
- The client sends PAR-packages to all Workers; source code are compiled on each machine
- Workers process the data and fill histograms
- Each Worker sends histograms to Master; Master merges them and sends the result to the client.

PROOF animation



- The client submits a request for Master and gets a list of Workers
- The client sends PAR-packages to all Workers; source code are compiled on each machine
- Workers process the data and fill histograms
- Each Worker sends histograms to Master; Master merges them and sends the result to the client.

PROOF animation



- The client submits a request for Master and gets a list of Workers
- The client sends PAR-packages to all Workers; source code are compiled on each machine
- Workers process the data and fill histograms
- Each Worker sends histograms to Master; Master merges them and sends the result to the client.

Working with Bean in PROOF mode

Using PROOF speeds up an analysis, but . . .

- You need to change the way of thinking about run process:
  - different parts of the program runs on different computers
  - it uses the different data sets
  - the second start may not reproduce the first
- "Error happens!" At compile time, at runtime, on master or on workers. . . How to debug?
  - **gdb** has a limited applicability
  - Reading and checking the code + "printf() debugging"
  - **valgrind** – very very slow
- The PROOF returns log-files but often they do not contain enough information. Sometimes you need to have access to log-files on workers

We recommend new users first to get some experience with Bean without PROOF, before switching to parallel data processing

How to run Bean in PROOF mode?

Bean in PROOF mode is a transparent extension of single user session

Example (**run Bean at local PC)**

```
> ./bean.exe  -u MyAnalysis   root://besdata.jinr.ru//data/bes3/run.dst \
     -h histo.root  -o selected_events.root
```

Example (**run Bean in PROOF-Lite mode)**

```
> ./bean.exe  -u MyAnalysis   root://besdata.jinr.ru//data/bes3/run.dst \
     -h histo.root  -o selected_events.root  -l
```

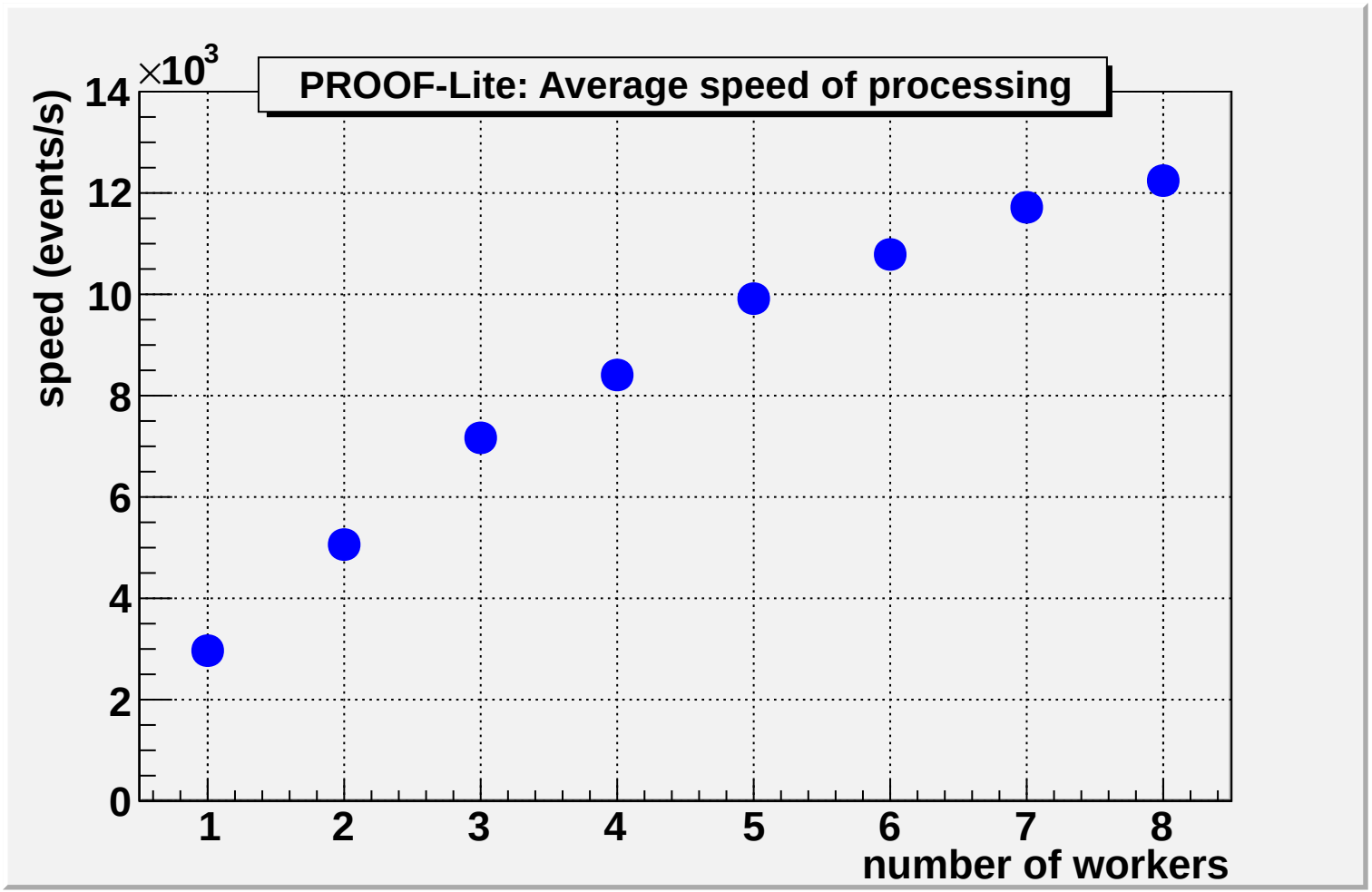
Example (**run Bean in PROOF-cluster mode)**

```
> ./bean.exe  -u MyAnalysis   root://besdata.jinr.ru//data/bes3/run.dst \
     -h histo.root  -o selected_events.root   -p “xrootd@lgdui01”
```

Bean-PROOF tests

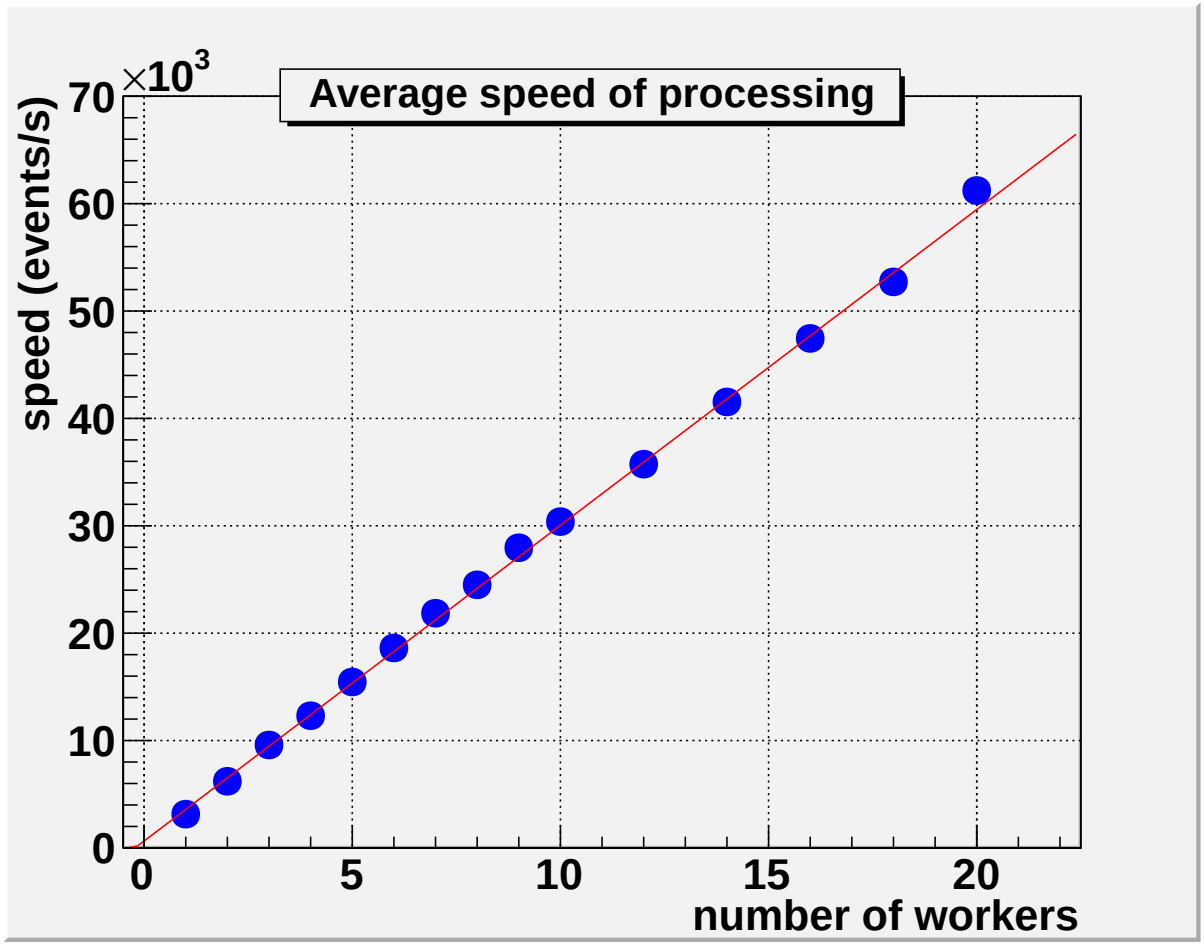
1) Bean-PROOF-Lite

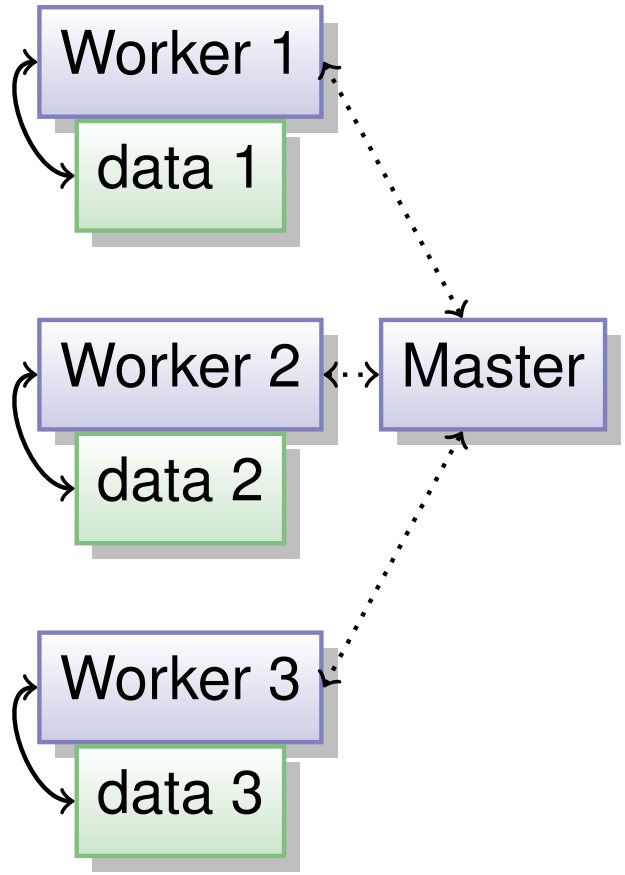
Test conditions: 2×Core 2 Quad = 8 cores; 10-files from Lustre FS



2) Bean-PROOF-cluster (Dubna)

Test conditions: 10×Dual core = 20 cores; 40 files with data were separated into 4 files on each machine





Datasets

What is datasets?

- dataset contains a list of names of DST-files and meta information about these files
- to be used in PROOF a dataset needs to be registered and verified

Using datasets with Bean

- Bean can register output DST as dataset:
  $\rightarrow$ output DST files are kept on Workers

Example (register dataset MyCuts)

```
> ./bean.exe  -u MyAnalysis   root://besdata.jinr.ru//data/bes3/run.dst \
    -h histo.root  -p “xrootd@lgdui01”  -o ds://MyCuts
```

- At the next startup, the bean can use this dataset as input:
  $\rightarrow$ Workers will use locally saved files

Example (use dataset MyCuts)

```
> ./bean.exe  -u MyAnalysis   ds://MyCuts \
    -h histo.root  -p “xrootd@lgdui01”
```

- We will get linear scalability and high rate of calculations

Summary

- This version of the Bean is ready for use (beta-version)
- Use of Bean in PROOF mode is possible
- Documentation is available and being updated regularly

TODO

- Improve the maintainability of the Bean
- What kind of functionality is missing?
  - BesDChain
  - User function parameters
- What kind of documentation would be helpful?

Credits

- We thank IHEP computer center for providing us with PROOF cluster to test our program

Thank you!