COMPARING THE EFFECTIVENESS OF PROOF WITH OTHER METHOD PARALLELIZM FOR THE EXPERIMENT DATA PROCESSING

SOLOVJEVA T., SOLOVIEV A.

The international Conference "Mathematical Modeling and Computational Phisycs"

3-7 July, 2017

Dubna, Russia

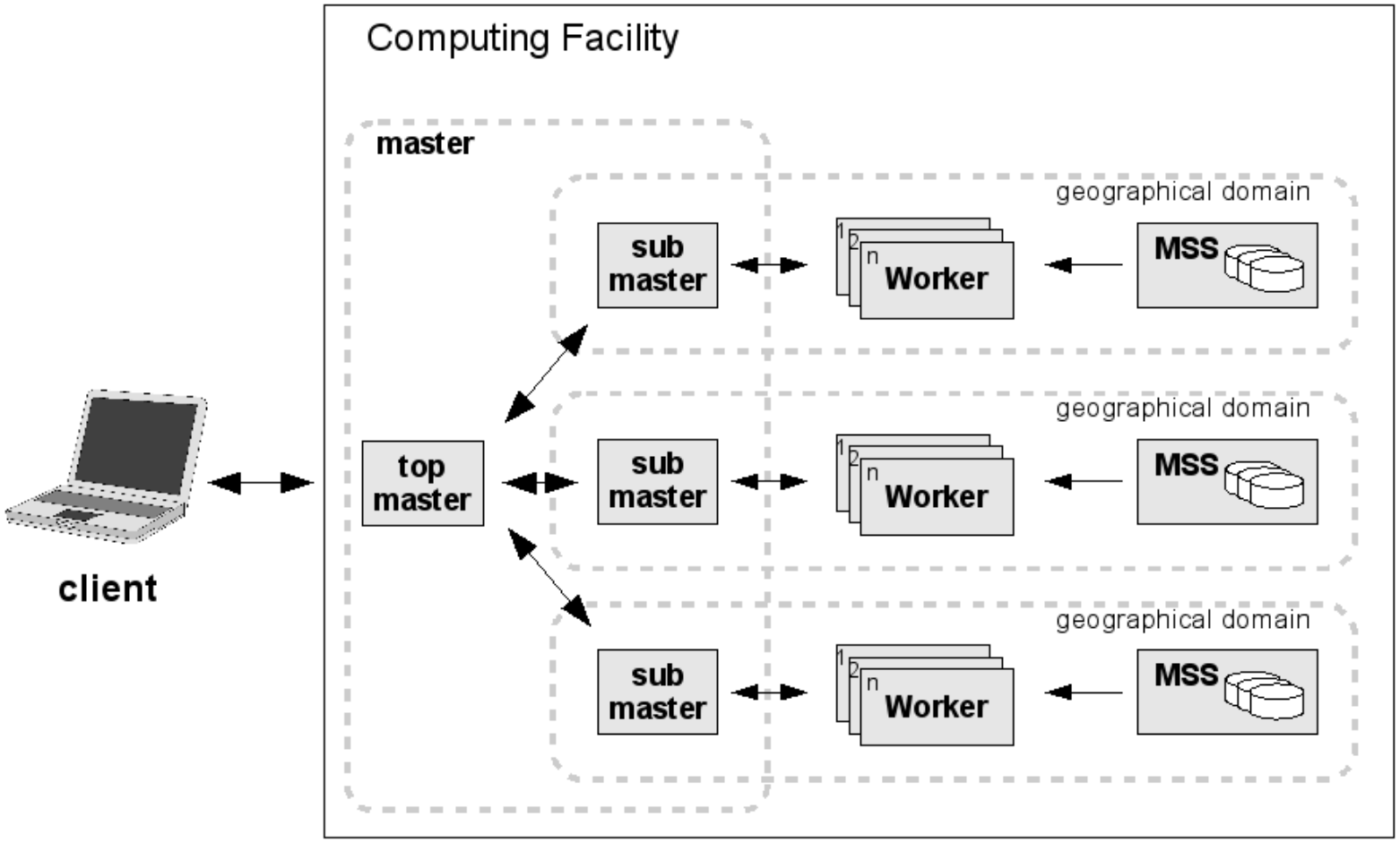
Tne Methods of Parallelism in ROOT

- PROOF – Parallel ROOT Facility - is an extension of ROOT enabling interactive analysis of large sets of ROOT files in parallel on clusters of computers or many-core machines. PROOF can parallelize tasks that can be formulated as a set of independent sub-tasks (embarrassingly or ideally parallel).

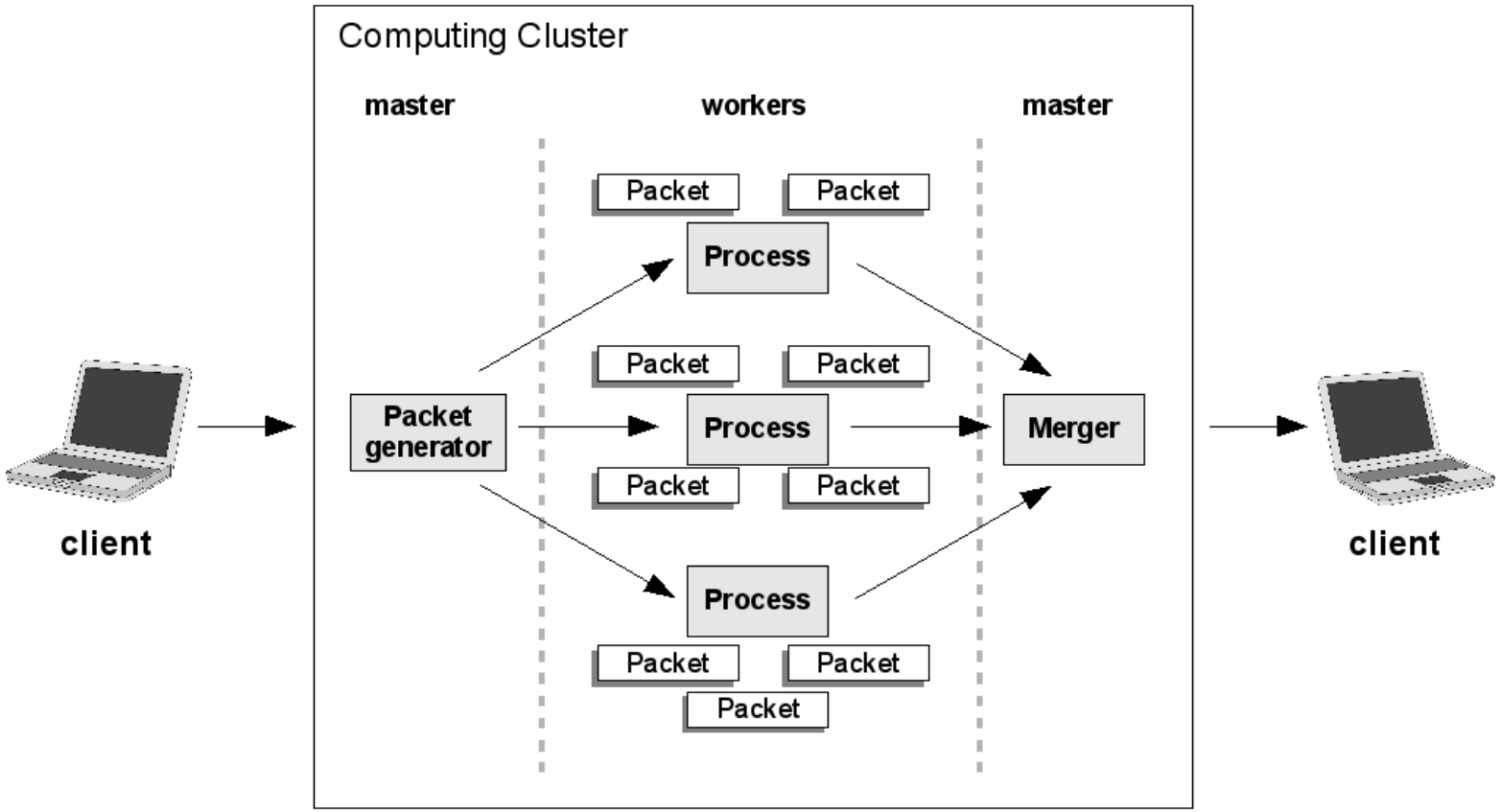
- Class Tthread - this class implements threads. A thread is an execution environment much lighter than a process. A single process can have multiple threads.

- Using Library OpenMP

MULTILEVEL ARCHITECTURE PROOF



Process PROOF



Selector Structure

- Begin() function is called at the start of the query. When running with PROOF Begin() is only called on the client.
- SlaveBegin() function is called after the Begin() function. When running with PROOF SlaveBegin() is called on each slave server.
- Process() function is called for each entry in the tree (or possibly keyed object in the case of PROOF) to be processed. The entry argument specifies which entry in the currently loaded tree is to be processed. It can be passed to GetEntry() to read either all or the required parts of the data, keyed objects is available via the fObject pointer. This function should contain the "body" of the analysis. It can contain simple or elaborate selection criteria, run algorithms on the data of the event and typically fill histograms.
- SlaveTerminate() function is called after all entries or objects have been processed. With PROOF SlaveTerminate() is called on each slave server.
- Terminate() function is the last function to be called during a query. It always runs on the client, it can be used to present the results graphically or save the results to file.

Class TThread

A thread is a sequence of instructions being executed in a program. A thread has a program counter and a private stack to keep track of local variables and return addresses. A multithreaded process is associated with one or more threads. Threads execute independently. All threads in a given process share the private address space of that process.

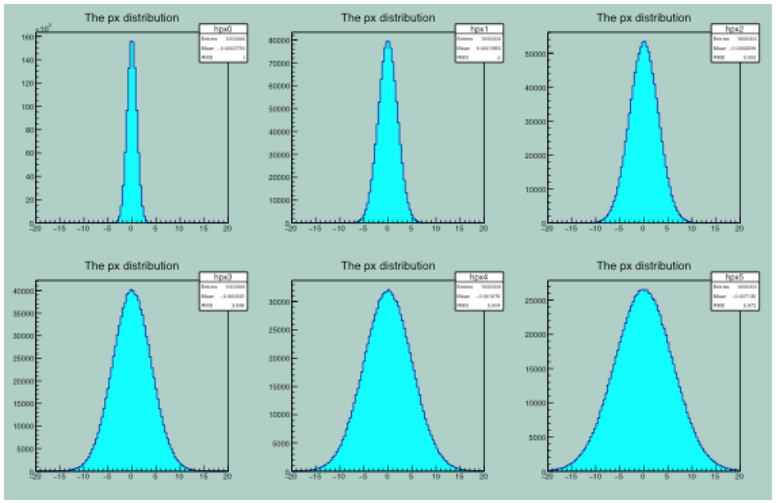
Parallelism arises when at least two threads are executing simultaneously. This requires a system with multiple processors.

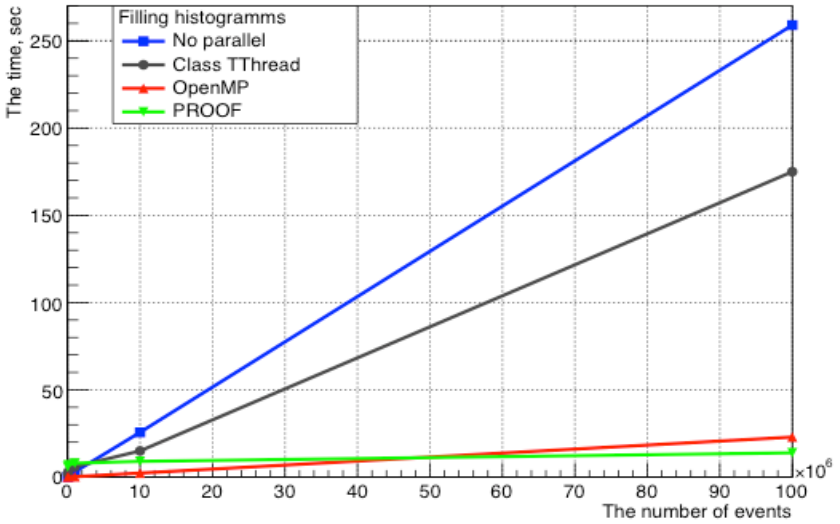
Left to their own devices, threads execute independently. Synchronization is the work that must be done when there are, in fact, interdependencies that require some form of communication among threads. Synchronization tools include mutexes (mutual exclusion lock), semaphores (mechanism of synchronization that starts out initialized to some positive value), condition variables, and other variations on locking.

First example

Events are generated according to the Gauss distribution

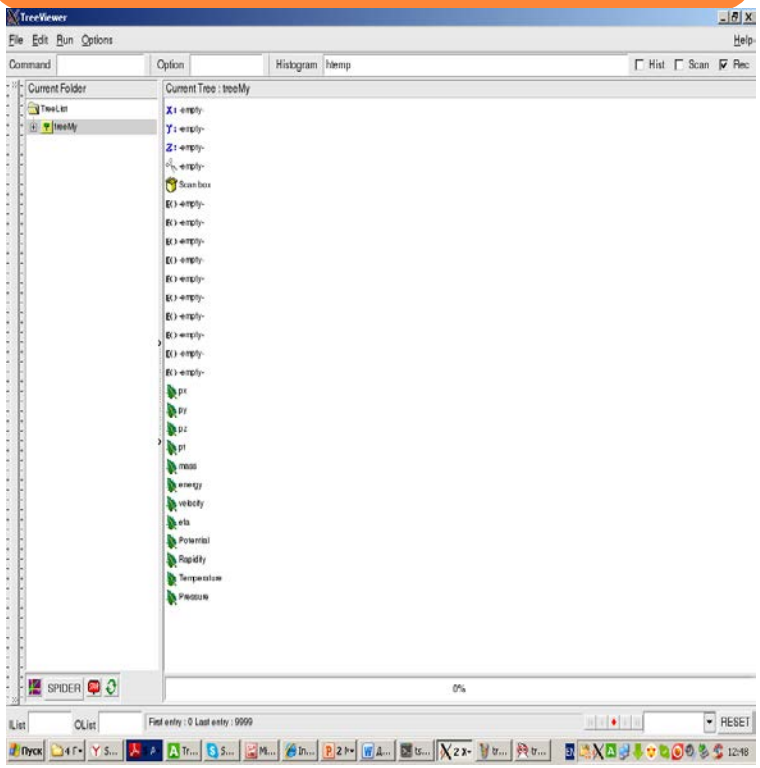
Dependence of the execution time of the macro on the number of generated events



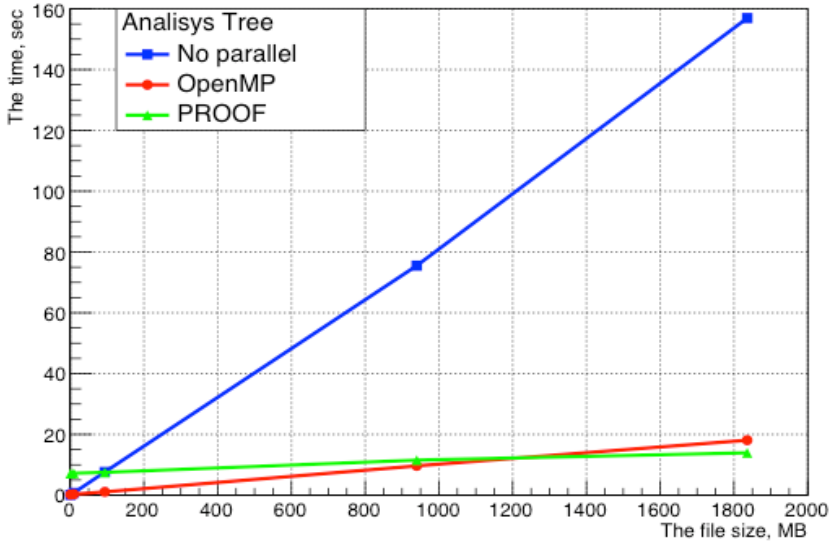


Second Example

Generation Tree with 12 branch



Dependence of the execution time of the macro on the file size



Conclusion

- As you can see from the presented graphs with small calculations or a small file size, it is advantageous to use the library OpenMP. But if you increase the amount of work, it is better to use PPOOF.
- Using a class Tthreadh gives a very small win in performance.
- Also note that PROOF has reserves for increasing productivity. In the processing of data, fittings are often used. It is executed in a program Terminate(), that is located in an unparallel section. Therefore, at this point, you can apply either the MINUIT2 program or the library RooFit. This task will be the goal of our further work.

Thank you for attention