Updates of R&D activities on the parallel framework

Alessio Gianelle
on behalf of the SuperB PD Computing Group

INFN-PD

November 19th, 2012

Outline

Scheduling model

Legacy code was modified in such a way that each module:

- declares what data have to be present inside the Event to start the execution;
- declares what products it adds to the Event.

From those information we can produce a dependencies graph, a tree where each node represent an analysis module and each arc a product.

- Each module has a lock to prevent concurrent execution.

The idea was to incrementally remove this limitation.

Features of this prototype

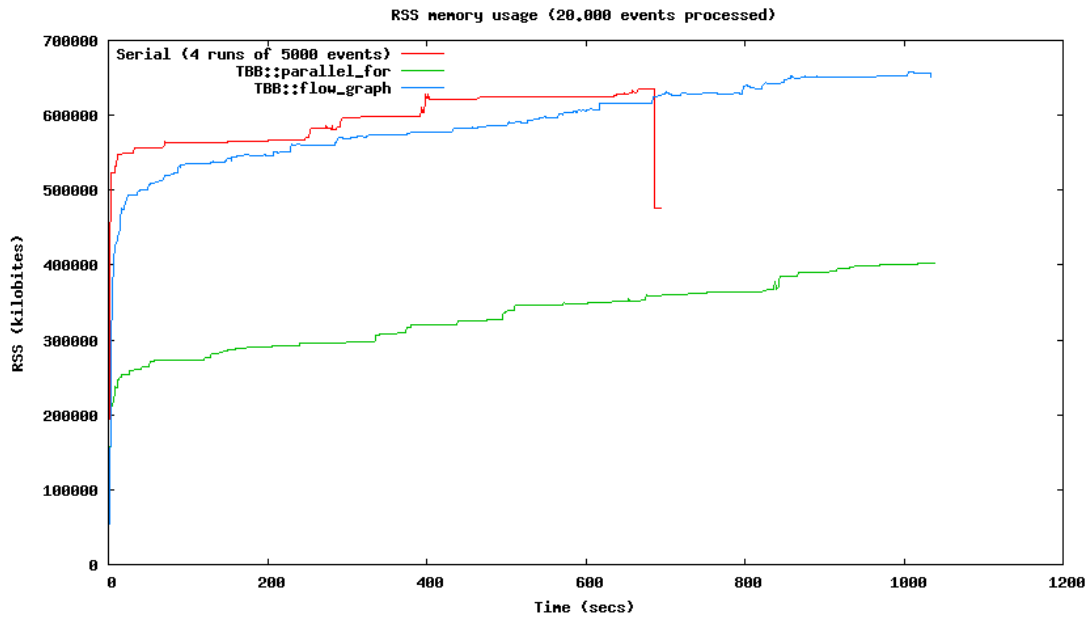
- A method has been added to the class Framework to build a Tbb flow::graph object.
- Modules have been wrapped by a functor which calls the *event* method.
- The event has been made "thread safe" substituting the list of physics products with a Tbb concurrent_hash_map.
- A mechanism to regulate the "injection" of events into the graph has been implemented.

(Old) Open Issues

- A deadlock which prevents us from injecting more than n events into the graph (where n is the threads number).
  Resolved. All initial modules, which use the same fortran common block, are locked together.
- During our test we also saw what is likely to be a memory leak.
  Resolved. See graph below.

RSS memory usage on

- System type: 1 way
- CPU: 1xIntel Veon E5630@2.53 Ghz (4 cores + HT)
- RAM: 3Gb per core
- OS: RHEL 6.3 (x86_64)



Computing model

From the prototype we have defined a computing model where:

- an analysis is defined as a set of modules;
- each module has to be independent from others;
- a module must define the products it needs and the ones it produces during its execution.

Measurements done on the prototype demostrate that:

- the model can be used to reduce the memory footprint;
- the scheduling schema may be employed to efficiently use systems with large number of cores.

Last but not least, measurements on the prototype were taken using a production setup: the prototype works!

Lessons learned

Some general software development guidelines were defined based on the framework analysis and prototype:

- fortran code has to be removed;
- widespread usage of static objects has to be avoided;
- each module as to be more OOP-compliant, in particular for what concern encapsulation;
- auxiliary data structures (Event container, etc.) have to be developed to allow concurrent access to data.

Work Plan

With the accumulate experience, we are now ready:

- to formalize specifications for analysis modules;
- to show an initial proposal design of a parallel architecture framework for experiment analysis;
- to show our idea of parallelism to the Forum on Concurrent Programming Models and Frameworks.