My Thoughts about Parallelization in HEP

Alfio Lazzaro

CERN Openlab

Multicore panel @ ACAT10, Jaipur

Use cases

- 5 main use cases:
  - Events Acquisition, online (High Level Trigger (HLT))
  - Reconstruction
  - MC Simulation
  - Data analysis: event selection and results extraction (e.g. fitting)
- Quite distinct problems for parallelization
  - Efforts should be consider differently for each use cases
  - Of course there are overlaps…

Looking more carefully (1)

- HLT requires high throughput
  - Suitable candidates for GPUs?
- Reconstruction suffers I/O bounds and memory usage
  - Efforts to reduce the memory footprint, using COW or KSM techniques
  - Still to keep in account a real parallelization (like AthenaMP or Parallel Gaudi) for data merging at the end
- MC simulation is similar to reconstruction, but it requires a distinct parallelization, i.e. Geant parallelization

Looking more carefully (2)

- Data analysis:
  - Event selection well performed in parallel using PROOF (data parallelism)
    - Still large dataset, good data parallelism
  - Fitting procedures (or similar techniques for results extractions) require a different approach: algorithm parallelism
    - Small samples, intensive CPU-time algorithms
    - Few examples on the market, still a lot to do
- In the follow I will talk about possible strategies in data analysis for results extractions

Caveats

- At the moment there is not (IMO) a clear picture about what will be data analysis at LHC (data needed!!!)
  - In the first phase it is reasonable to think that with small samples and (as usual for new experiments and in case of search for new phenomena) simple analysis will be used (events counting)
    - Reduce systematic errors estimations
  - Efforts will be concentrated to have results in a reasonable time schedule
- In other words, the customers (physics analysts') do not require complex algorithm
  - Fitting a 1D histogram it is a simple operation, do not require parallelization

But…

- Other experiments with data (BaBar, Belle, BES, CDF, Focus, D0,…) have already started to do complex analysis where parallelization can be a good (mandatory) solution to speed-up execution
  - In the Babar community we did a huge effort to
- Not forget the upgrade for LHC (sLHC) and new experiments (SuperB-factory)
  - They require a jump of factory >100 (!!) in 5 years timescale (Moore law is not enough…)
- Other communities (HPC, astrophysics, chemistry, biologist,…) have similar problems:
  - LSST has a rate of raw data of 60000 MB/s every 40 seconds (Atlas is 300 MB/s): fast analysis is mandatory to screaming the data

Other problem

- In data analysis there is not a general common framework (like reconstruction) for different analysts'
  - In general everybody wants the "power" to obtain the final results, i.e. his own version of data analysis code
- This means a "plethora" of programs
  - Not always based on the same base-code (different languages, Matlab, different algorithms…)
- Advantage: possible to make comparisons to spot bugs out
- Disadvantage: "sometimes" all the versions are not well optimized
  - Last year Babar sent a request to do a parallel version of a fitting code to ROOT people. The programs had taken about 1 hour. After few optimizations (not parallelization) now it takes 3 minutes…
    - Francois Le Diberder (current spokesperson): are we wasting resources? Shall we optimize our data analysis programs?
  - You can image the possible scenario if we move to parallel version of the code that are, by definition, more difficult to develop and debug…

My Conclusions

- Need to individuate a set of data analysis techniques and parallelize them
  - Which ones? Depends on customers (e.g. RooFit/RooStats projects)
- Individuate the most time-expensive part and write them as kernel function (plugins) that you can run in parallel inside you current program
  - Not forget optimization!
  - Not easy to run everything in parallel, unless you want to rewrite everything!
  - Run the different kernel exploring all kind of possible parallelization on the market (GPUs, multi-cores, vectors, …)
- In some cases our algorithms are old (like CERNLIB Minuit)
  - Not good algorithms to scale to many-cores
  - Need to think about how we can improve them
- Common way to procedure: same patterns, same techniques, same languages, same libraries, common framework (!!)
  - Concentrate the efforts. At the moment there is not a enormous demand for parallelization
    - AFAIK about 10 groups are working on data analysis parallelization
- Be ready for the future (which is not far away)…