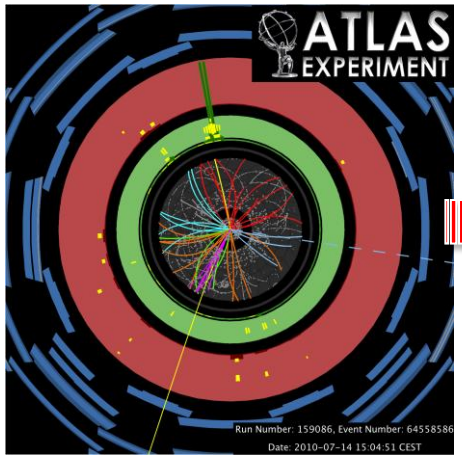
Evaluation of maximum likelihood fits on GPU devices using CUDA

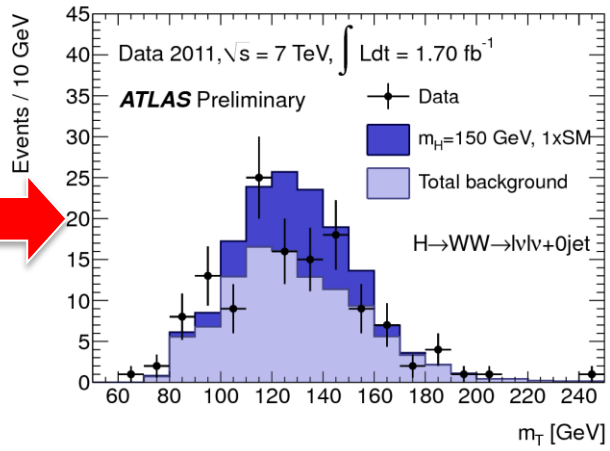
Felice Pantaleo
European Organization for Nuclear Research (CERN), Geneva, Switzerland
University of Pisa, Italy

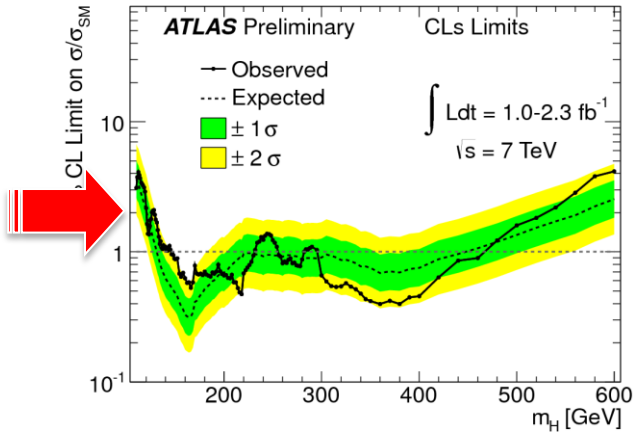
Facing the Multicore Challenge II
Karlsruhe Institute of Technology, 28-30 September 2011
Karlsruhe, Germany

Data Analysis

- Huge quantity of data collected, but most of events are due to well-know physics processes
  - New physics effects expected in a tiny fraction of the total events: few tens
- Crucial to have a good discrimination between interesting (*signal*) events and the rest (*background*)
  - Data analysis techniques play a crucial role in this "fight"







Likelihood-based Techniques

- Data are a collection of independent events
  - an event consists of the measurement of a set of *observables* (energies, masses, spatial and angular variables...) recorded in a brief span of time by the physics detectors

- Introducing the concept of probability P (= Probability Density Function, PDF) for a given event to be signal or background, we can combine this information for all events in the *likelihood function*

$$\mathcal{L} = \prod_{i=1}^{N} \mathcal{P}(\hat{x}_i|\hat{\theta})$$

N number of events
$\hat{x}_i$ set of observables for the event i
$\hat{\theta}$ set of parameters

- Several data analysis techniques requires the evaluation of L to discriminate signal versus background events
- Finding the maximum of this function is equivalent to “what is the parameter estimation that makes the data set most probable for the prediction model?”

Algorithm Description

- Likelihood maximization is equivalent to minimize:

$$NLL = \sum_{j=1}^{s} n_j - \sum_{i=1}^{N} \left[\ln \sum_{j=1}^{s} \left(n_j \prod_{v=1}^{n} \mathcal{P}_j^v(x_i^v|\hat{\theta}_j) \right) \right]$$

Algorithm Description

- Likelihood maximization is equivalent to minimize:

$$NLL = \sum_{j=1}^{s} n_j - \sum_{i=1}^{N} \left[\ln \sum_{j=1}^{s} \left(n_j \prod_{v=1}^{n} \overset{①}{\mathcal{P}_j^v(x_i^v|\hat{\theta}_j)} \right) \right]$$

① Each P (Gaussian, Polynomial,…) is implemented with a corresponding class (basic PDF)

- Loop over the observables, produce a vector of results

Algorithm Description

- Likelihood maximization is equivalent to minimize:

$$NLL = \sum_{j=1}^{s} n_j - \sum_{i=1}^{N} \left[\ln \sum_{j=1}^{s} \left(n_j \textcircled{2}{\prod_{v=1}^{n} \mathcal{P}_j^v(x_i^v|\hat{\theta}_j)} \right) \right]$$

① Each P (Gaussian, Polynomial,…) is implemented with a corresponding class (basic PDF)

- Loop over the observables, produce a vector of results

② Product over all observables

Algorithm Description

- Likelihood maximization is equivalent to minimize:

③

$$NLL = \sum_{j=1}^{s} n_j - \sum_{i=1}^{N} \left[\ln \sum_{j=1}^{s} \left(n_j \prod_{v=1}^{n} \mathcal{P}_j^v(x_i^v|\hat{\theta}_j) \right) \right]$$

① Each P (Gaussian, Polynomial,…) is implemented with a corresponding class (basic PDF)
- Loop over the observables, produce a vector of results

② Product over all observables

③ Sum over all the types of events

Algorithm Description

- Likelihood maximization is equivalent to minimize:

④

$$NLL = \sum_{j=1}^{s} n_j - \sum_{i=1}^{N} \left[\ln \sum_{j=1}^{s} \left(n_j \prod_{v=1}^{n} \mathcal{P}_j^v(x_i^v|\hat{\theta}_j) \right) \right]$$

① Each P (Gaussian, Polynomial,…) is implemented with a corresponding class (basic PDF)
  - Loop over the observables, produce a vector of results

② Product over all observables

③ Sum over all the types of events

④ Reduction of all values

Parallelization Strategy

- All the calculations are done in double precision
- Data and results are organized in vectors
  - Input data read-only during the NLL evaluation
- Several loops to handle:
  - Evaluation and normalization of the basic PDFs, combinations, reduction
- Easy loop parallelism
  - Evaluation of functions over arrays of data
    - Balanced independent iterations (except the reduction)
  - Possible to exploit vectorization
    - Using Intel compiler for the auto-vectorization of the loops (using svml library by Intel)
- Reduction in parallel using block-wise algorithm, taking in account rounding problems

Implementations

I. OpenMP

- An OpenMP parallel region for each loop
  - Static partition of the iterations

II. CUDA

- Offload the loops to the GPU using CUDA kernels
  - A CUDA thread per each iteration
- Observables arrays are copied on the GPU once (synchronous)
- Results arrays are allocated on the GPU global memory once and they are deallocated at the end
- Reduction performed on the GPU, result (one double) copied to the host
- Using an heuristic approach to decide the block size:
  - 64 for kernels with transcendental functions, 128 otherwise
  - Occupancy ranges from 0.33 to 0.67, depending on the kernel

Test

$$n_a G_1^a(x) G_2^a(y) G_3^a(z) + n_b G_4^b(x) G_5^b(y) G_6^b(z) +$$
$$n_c A_1^c(x) P_1^c(y) P_2^c(z) + n_d P_3^d(x) P_4^d(y) A_2^d(z) +$$
$$n_e P_5^e(x) G_7^e(y) A_3^e(z)$$

Model from B. Aubert *et. al.*,
Phys. Rev. D80, 112002, 2009

21 PDFs in total, 3 observables, 5 types

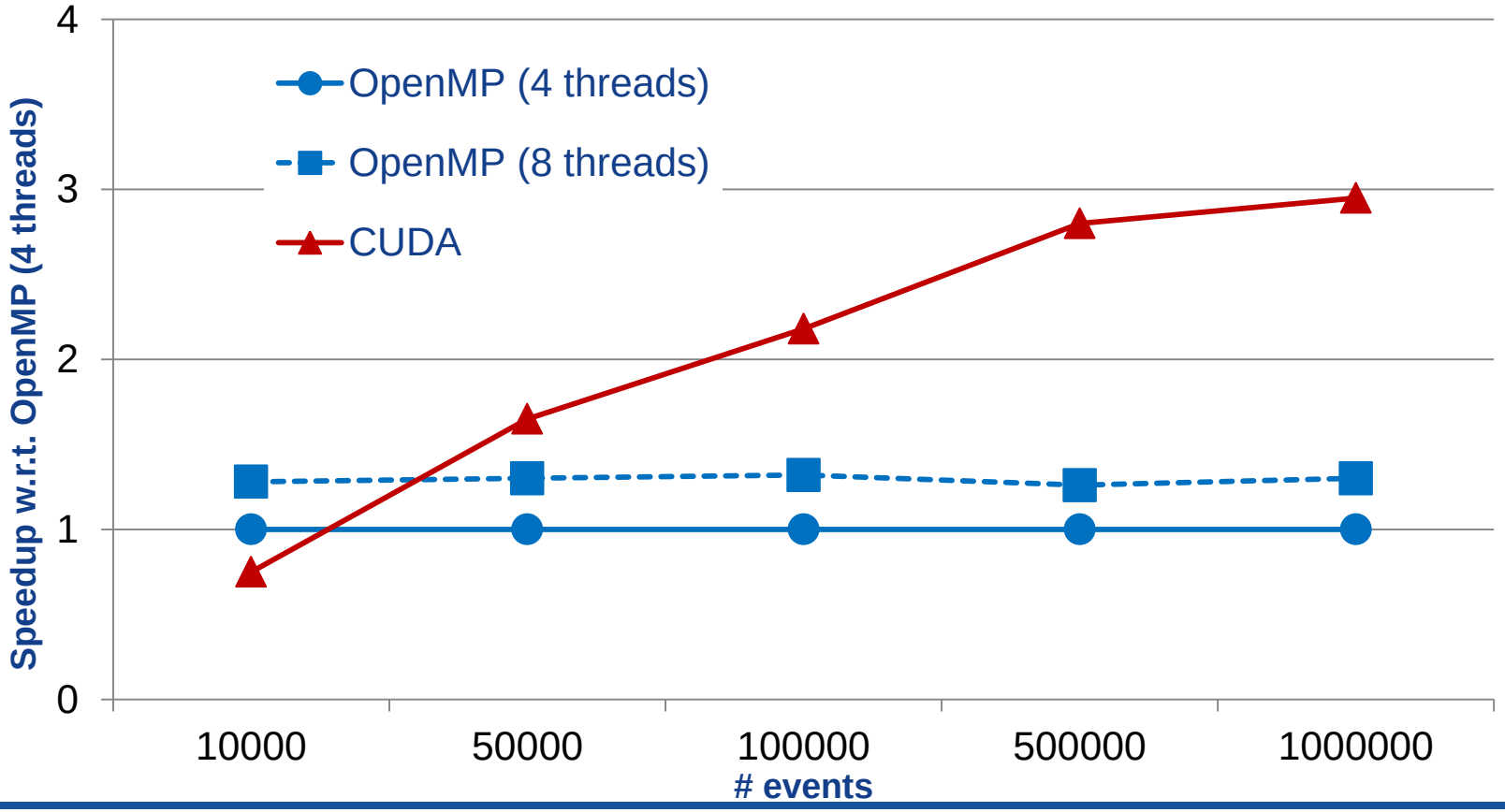
- G: Gaussian
- A: Argus function
- P: Polynomial

60% of the execution time is spent in exp's calculation

Note: all PDFs have analytical normalization integral, i.e. ~97% of the sequential portion can be parallelized

Results

- OpenMP CPU implementation on Intel 4-core “Nehalem”-based desktop system (Intel Core i7 965 @ 3.2GHz) compiled with Intel Composer XE 2011
  - Speed-up is 3.6x with 4 threads
  - Limited by sequential portion of the code
- CUDA GPU version implementation on NVIDIA GTX470, CUDA v4.0



Conclusion

- OpenMP CPU implementation
  - Good scalability and overall performance
- Successfully ported the algorithm using CUDA
  - Good performance on GPU cards
- More data will be collected by LHC experiments in the following years
  - GPU implementation will play an important role
- This is still a R&D project
  - Move to production code already started, which involves several groups in the experiments for testing
- Working on heterogeneous CPU+GPU (OpenMP + CUDA) and multiple GPUs systems.

Previous work references

- S. Jarp *et. al.*, *Parallelization of maximum likelihood fits with OpenMP and CUDA*, proceeding of "The International Conference on Computing in High Energy and Nuclear Physics", Taipei (Taiwan),18-22 October, 2010, to be published on Journal of Physics: Conference Series. CERN-IT-2011-009
- S. Jarp *et. al.*, *Evaluation of likelihood functions for data analysis on Graphics Processing Units*, ipdpsw, pp.1349-1358, 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum, 2011
- S. Jarp et. al., *Parallel Likelihood Function Evaluation on Heterogeneous Many-core Systems,* presented at International Conference on Parallel Computing, 30 August, 2011, Ghent (Belgium)

Q & A

CERN openlab