G4MT Performances on MIC architectures

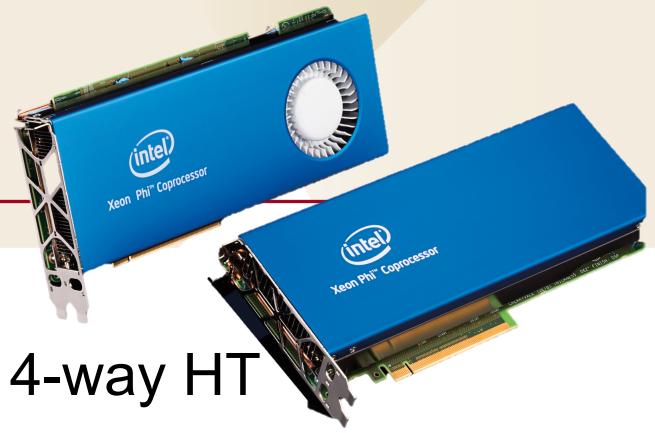
A. Dotti for the Geant4 multithreading task-force

18th Collaboration Meeting, Seville (SPAIN)

U.S. DEPARTMENT OF ENERGY
Office of Science

SLAC NATIONAL ACCELERATOR LABORATORY

Why MIC architectures?



- New HW from Intel
- Large number of x86_64 cores (60) with 4-way HT
  - 8 GB memory
  - 240 hardware threads
  - 512 bits wide registers
- PCIe card form factor
  - Up to 8 cards can be hosted in a single Xeon server
- Two ways of operating:
  - Offload: #pragma to offload functions to the card
  - Native: compile code and start it on the card (via ssh, embedded linux)
- Since it is Intel architecture no need to re-write code (e.g. no CUDA)

Notes

SLAC

- Very simple to port code to the MIC
  - Just use Intel compiler and add a switch
  - Intel provides debugging and profiling tools (V-Tune)
  - Additional HPC tools available: intelMPI
- Warnings:
  - The single core performances are much less than single core performances on Intel Xeon (shared bus, shared memory, lower frequency)
  - To fully use card need to: use maximum number of HW threads (e.g. fit 240 threads in memory budget), use vectors, improve locality of data/code (efficient use of core caches)

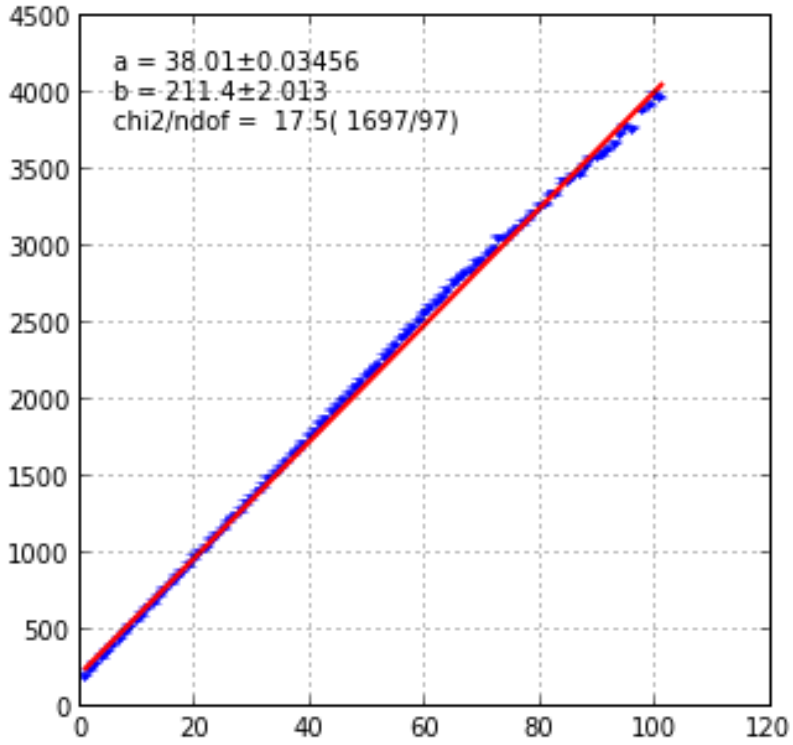
Measurements

SLAC

- Use MT version of Full CMS
  - Remove all I/O during event loop
  - Need to “massage” GDML files (XML validations file must be local, no network based validation)
- No effort in understanding vectorization yet

- We have been warned by people that used the card before: “if you do not work on using wider registers or optimize memory access patters MIC performances will be << server performances –down to 10-20% –”
  - Our expectations were not very high

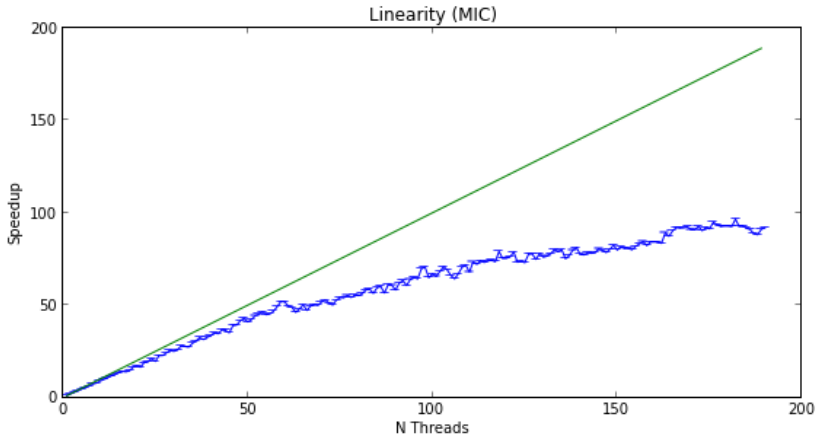
Memory consumption

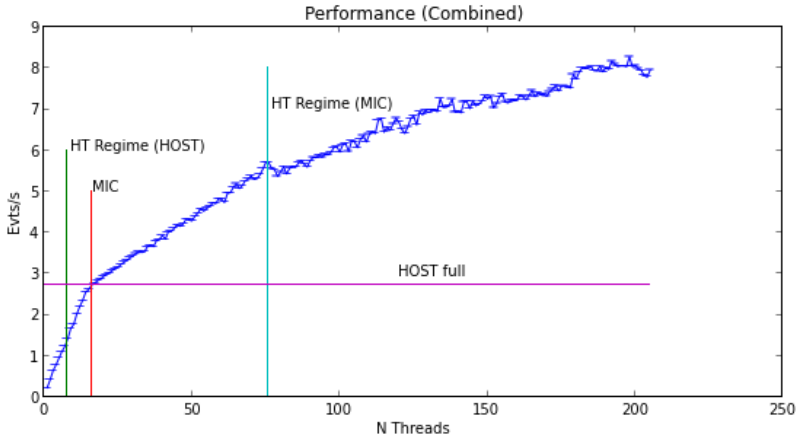


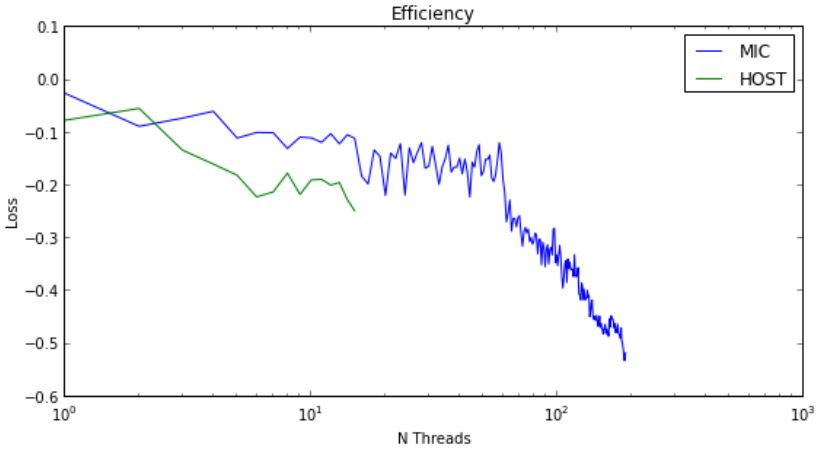
- Use “top” to see memory consumption during event loop
- 38MB/threads
  - Lower than on AMD, not yet understood

Results

- Results obtained on AMD and Intel confirmed: very good linearity
- HT still plays an important role providing more than 50% of throughput
- Max number of threads before exhausting memory: 194

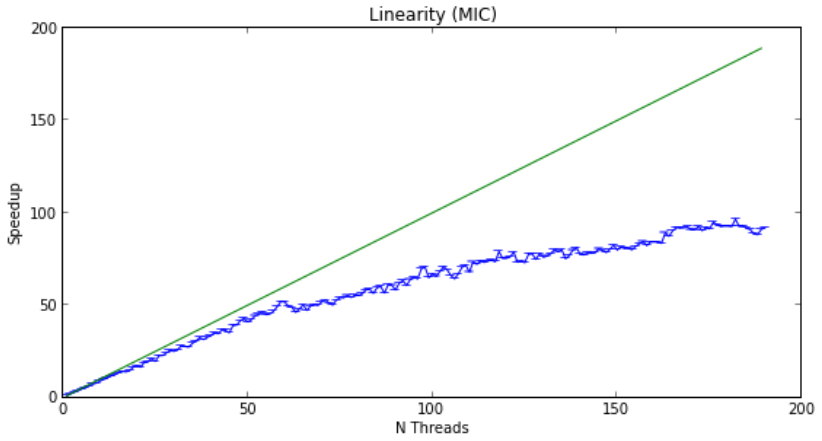


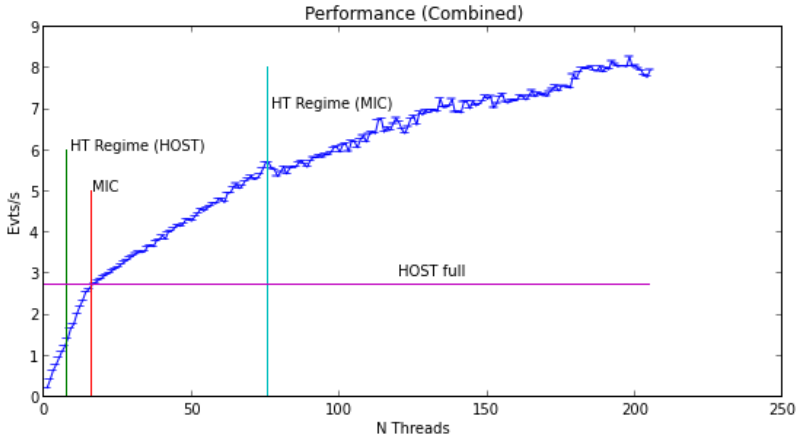


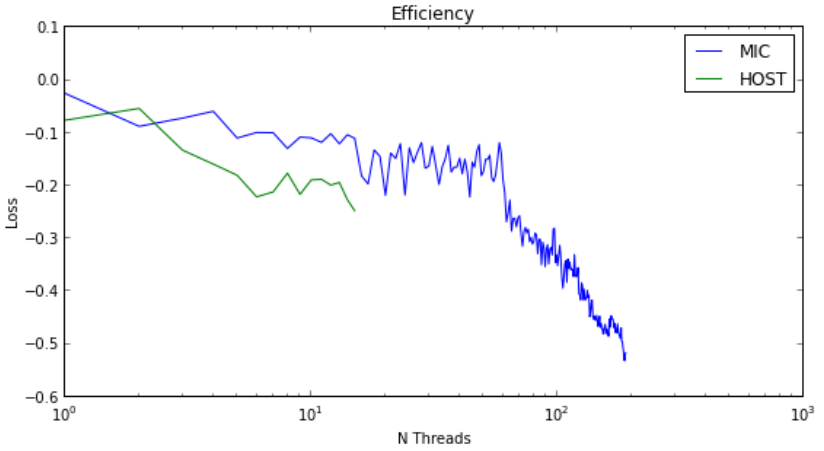


Results: the good news

- Card provides the double throughput in terms of number of events/s w.r.t. host
- Opposite of expectations!







MIC results

SLAC

- Good results can be explained if:
  - There is no I/O involved and thus no data moving from host to card during the event loop
  - We use efficiently large number of threads, compensate single thread lower performance

- In addition: split-class mechanism allows for better use of caches:
  - False sharing is reduced
  - Each core has its own L2 cache, but data can be scattered on all caches requiring synchronization
  - Split-classes ensures that data is not requested by other threads
- Just an idea for the moment to be confirmed