Parallel QTLMap

Olivier Filangi (INRA)
Pascale Leroy (INRA)
Jean Michel Elsen (INRA)
Dominique Lavenier (Symbiose/IRISA/INRIA)
Guillaume Chapuis (Symbiose/IRISA/INRIA)

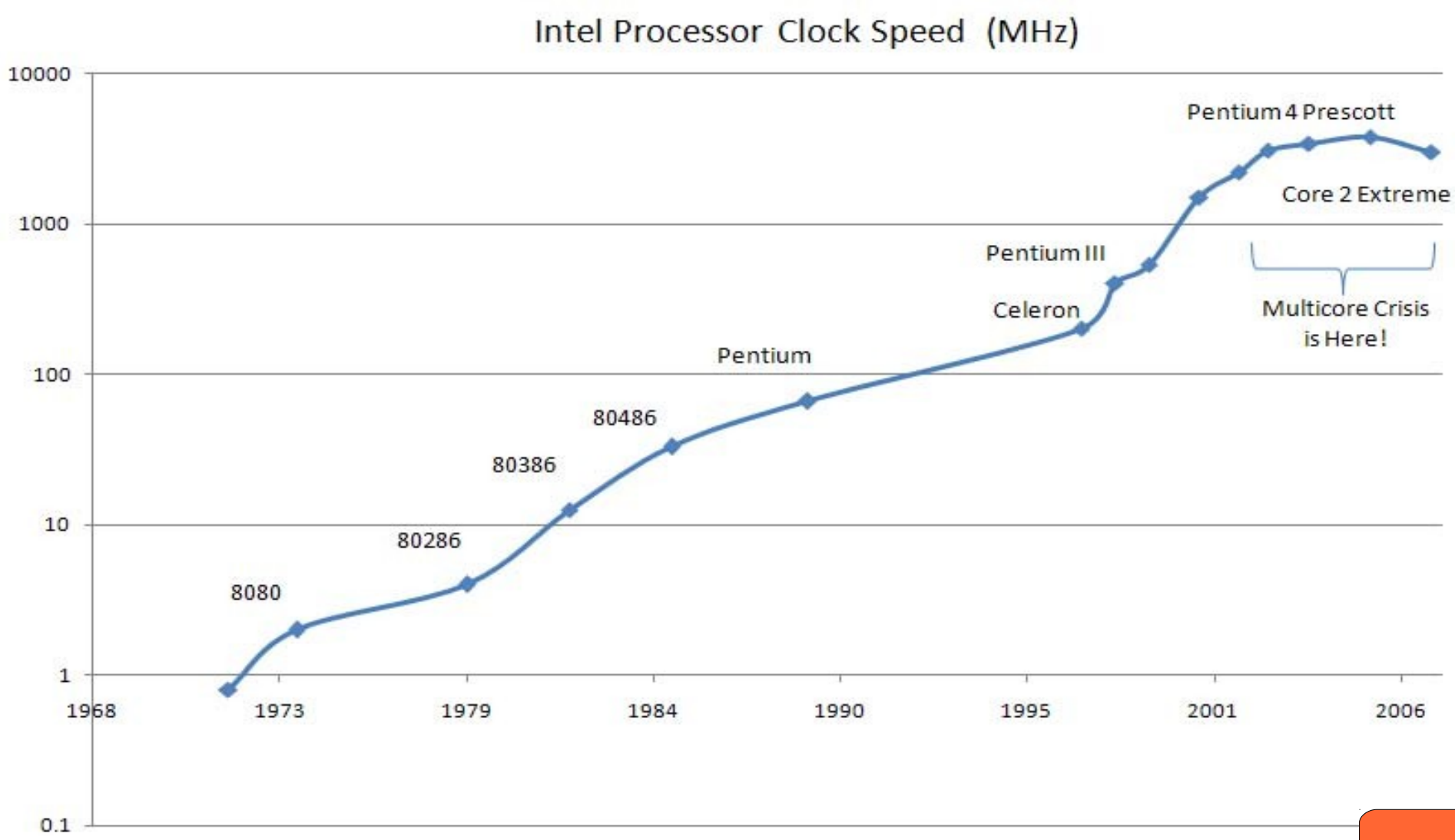
Parallel QTLMap

- Time consuming computations
  - Hours, days of computations
- Increasing amount of data
- => Speed up to keep up !
  - How ? Parallelize !
- Parallelizing = Using more than one processing unit to solve a single problem

Parallel QTLMap

- Why parallelize ?
  - Why now ?
- Where is it applicable ?
  - Only effective on some algorithms
- How ?
  - Using Graphics Cards (GPUs)
- Results

Why parallelize ?

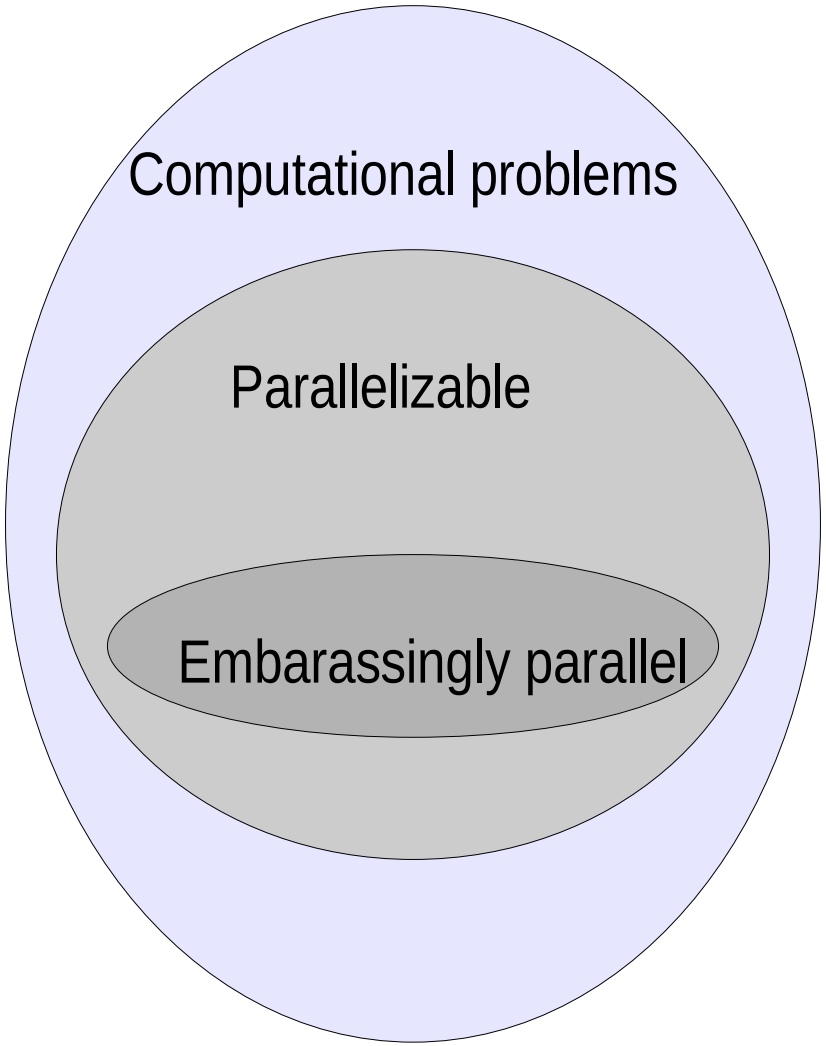


Source : Bob Warfield - A Picture of the Multicore Crisis - http://smoothspan.wordpress.com/2007/09/06/a-picture-of-the-multicore-crisis/

Why parallelize ?

- Before 2005
  - Exponentially increasing processor clock rate
  - Upgrading hardware => speed up
- Since 2005
  - Increasing number of processor cores
  - Using more than one processing unit => speed up (sometimes...)

Where ?



- Parallelizing applies to a small subset of problems
  - With independent sub tasks
  - And only a relatively small sequential part

Where ?

- Parallelizable :
  - Finding a needle in a haystack
    - Split the haystack !
- Inherently sequential :
  - Going from point A to point B
    - Sending someone else halfway won't get you to B any faster

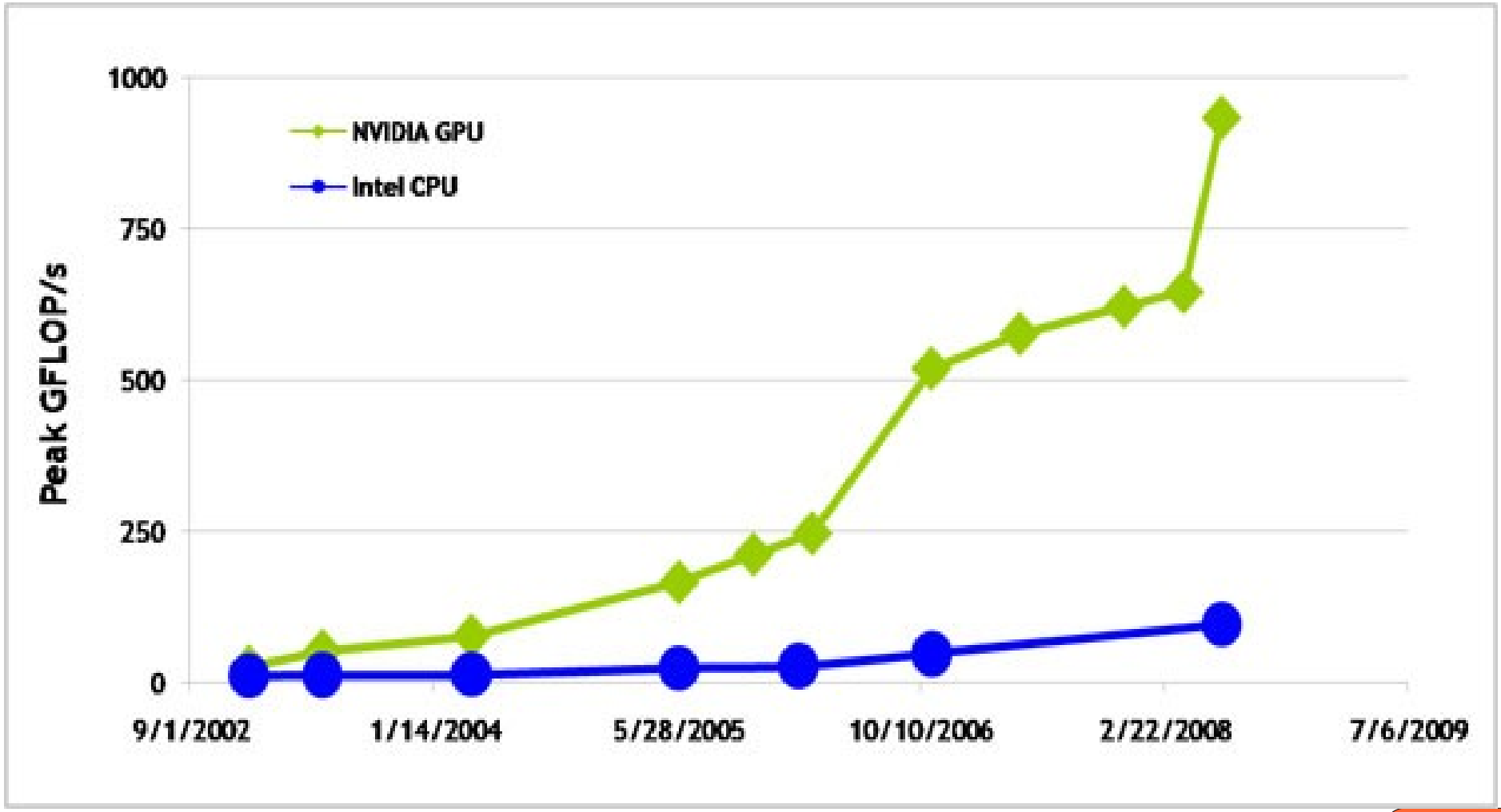
How ?

- With a multicore processor ?
  - Dual core, quad core
- With multiple machines ?
  - Cluster, supercomputer
- With Graphics Cards (GPUs) ?
  - Nvidia, ATI

Parallelizing on GPUs

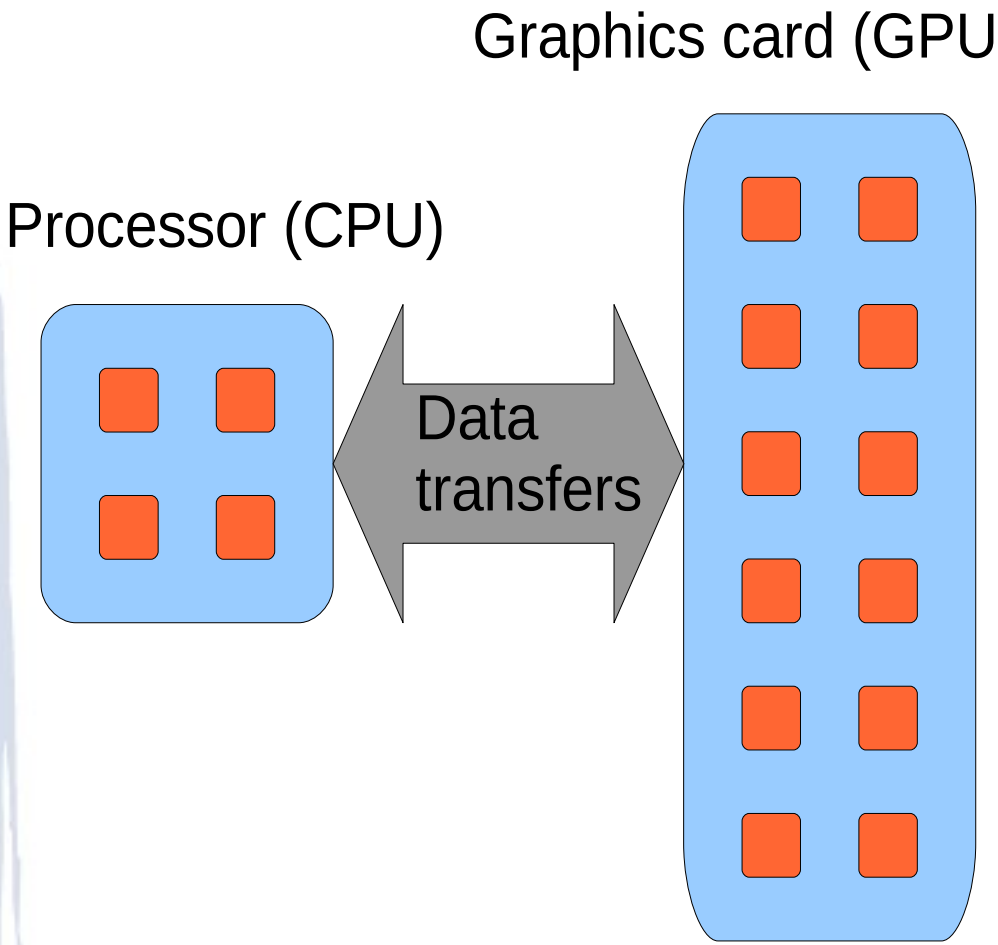
- Graphics cards are massively parallel
  - Up to 512 cores
  - All cores execute the *same task* over *different data*
- Need *many* independent and *identical* tasks !

Parallelizing on GPUs



Source : nVidia

Parallelizing on GPUs



- Computations are driven by the processor
  - Copies data to GPU memory
  - Requests execution
  - Copies results back to its memory

Parallelizing on GPUs

- Programming for a graphics card is not as straight-forward as for a processor
  - Complex memory hierarchy
  - Hard to debug
- But...
  - Standards are evolving rapidly
  - HMPP : High level programming language

Results with QTLMap

- Benchmark description
  - Data : chromosome 1 (QTLMas 2011 data)
  - # of markers : 1998
  - # of half-sib families : 20
  - Progeny with performance : 2000
  - Progeny with genotype : 3200
  - Step : 0.01 Morgan
- Source code soon available at ww.inra.fr/qtlmap

Results with QTLMap

- QTLMap single core
  - LA (1 QTL) : 7 min10 s
  - LA (2 QTL) : 13 h 50 min
- QTLMap multi core (6)
  - LA (1 QTL) : 2 min 20 s
  - LA (2 QTL) : 3 h 03 min 20 s
- QTLMap GPU
  - LA (1 QTL) : 14 s
  - LA (2 QTL) : 5 min 53 s
  - LDLA (1 QTL + 1 haplotype) : 19 s
  - LDLA (2 QTL + 2 haplotypes) : 10 min 21 s

Thank you !

Any questions ?