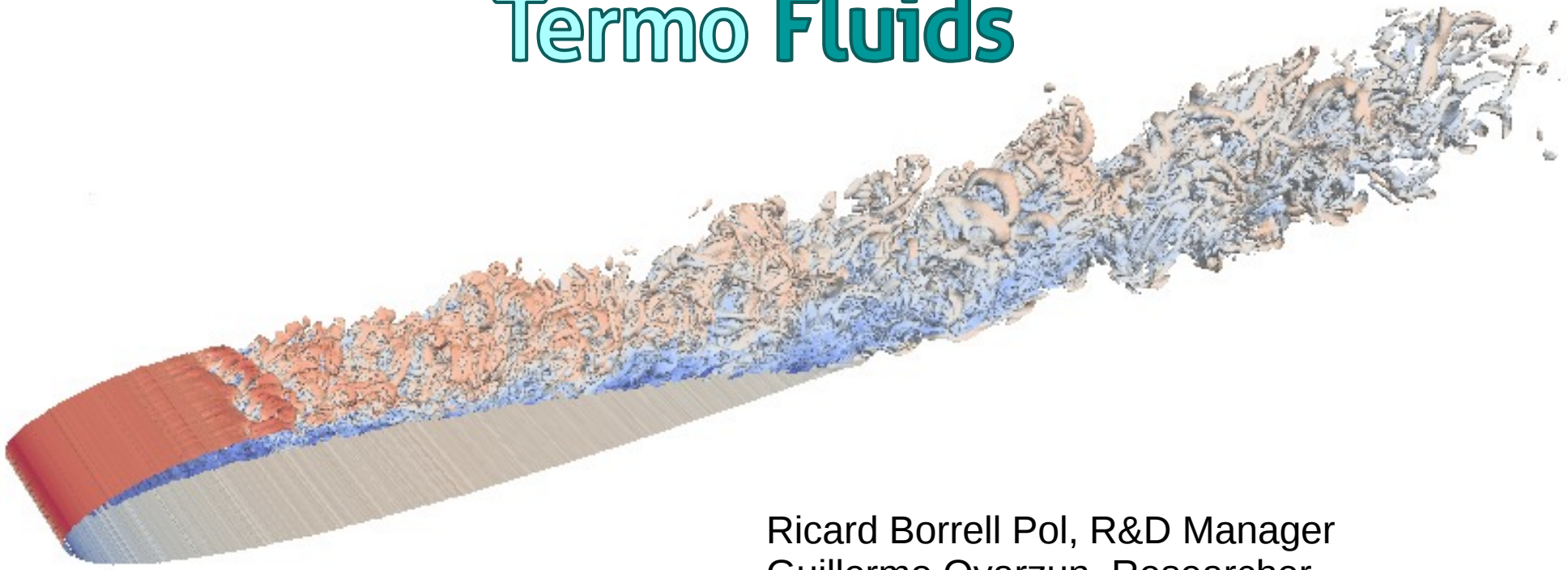
Termo Fluids



Ricard Borrell Pol, R&D Manager
Guillermo Oyarzun, Researcher

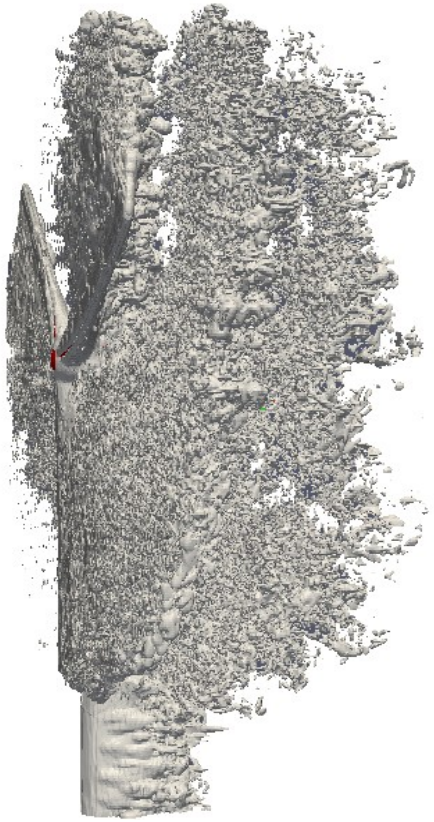
Termo Fluids S.L., Av Jaquard 97 1-E, Terrassa, Barcelona, Spain

termofluids.com

TERMO FLUIDS

Termo Fluids

- SME from Terrassa (Barcelona)
- Spin-off of the CTTC center from the Technical University of Catalonia
- We provide CFD consulting services at leading edge HPC level using advanced multi-physic models
- Our studies are based on simulations performed with our in-house unstructured CFD code
- Working for different industrial areas such as renewable power generation, thermal equipments or HVAC and refrigeration



- Network:

ABENGOA SOLAR

ETP 4 HPC

eit Knowledge & Innovation Community KIC InnoEnergy

UPC

OFFWINDTECH

CLEANSKY

EWEA THE EUROPEAN WIND ENERGY ASSOCIATION

ALSTOM

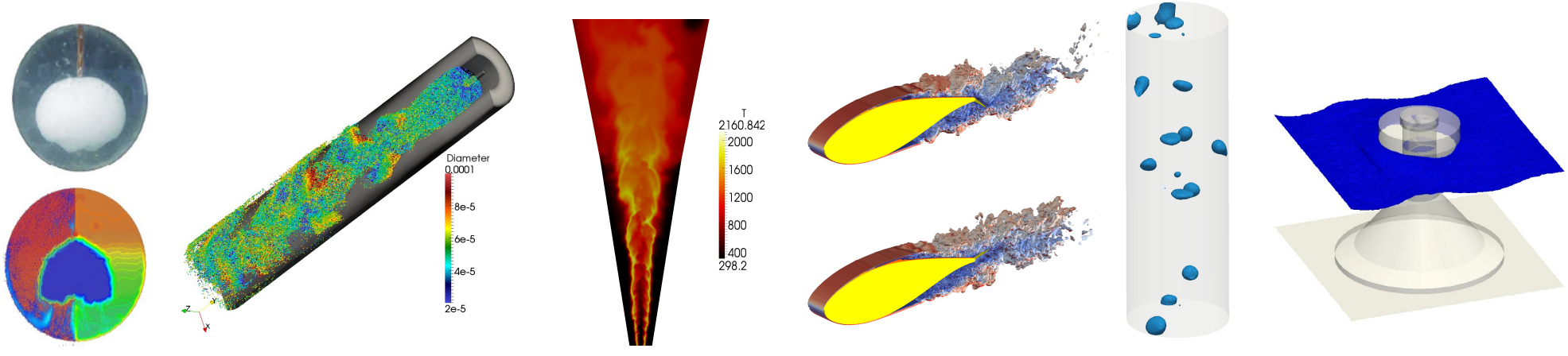
airtècnics

FAGOR

MONT BLANC

TermoFluids CODE

- **TermoFluids** currently not available as a standalone product, but is shared with different academic partners:
  - CTTC – **Technical University of Catalonia (UPC)**
  - Aerodynamics and Flight Mechanics Group **- University of Southampton**
  - Computational aeroacoustic laboratory - **Russian Academy of Sciences**
- General purpose unstructured CFD code in C++
- Finite volume symmetry-preserving discretizations on unstructured meshes
- Several LES and Regularization models for turbulent flows
- Expansion to **multi-physics simulations**: reactive flows, combustion, multi-phase flows, particles propagation, fluid structure interactions, radiation effects, dynamic meshes...



HPC in Termo Fluids

- HPC is an essential tool for CFD (high demand of computing and memory resources)

- **TermoFluids** code:
  - ✔ C++ object oriented
  - ✔ Mostly based on the distributed memory model (MPI) – recently developed hybrid model with GPU-coprocessors (MPI/CUDA)
  - ✔ **Performance barriers:**
    - Parallelism: inter CPU communications (point-to-point, all-reduce) → not so critical now
    - Flops: low flop per byte ratio → critical (low percentage of peak performance achievable)

Curie TGCC

MareNostrum BSC

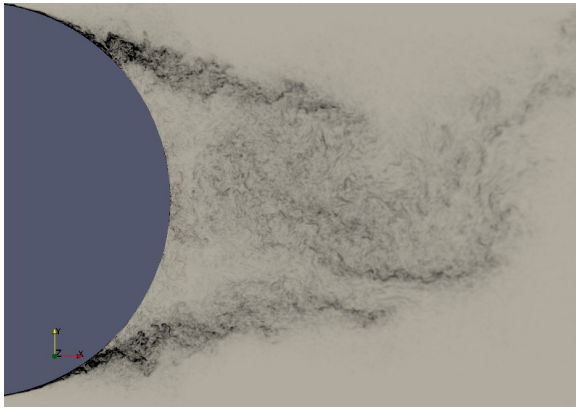
JFF CTTC

Lomonosov MSU

Mira ALCF

Recent HPC projects: DRAGON

- **TermoFluids** code recently used in **PRACE Tier0** project by CTTC (23M hours)
- Project ran in **BSC MareNostrum II** supercomputer
- **DRAGON** - Understanding the DRAG crisis: ON the flow past a circular cylinder from critical to trans-critical Reynolds numbers
  - ✔ Largest simulation with 4096 CPU-cores
  - ✔ Mesh 320M CV, Re=4e6
  - ✔ Around 10 TB of data outputs



PRACE Preparatory Access Project

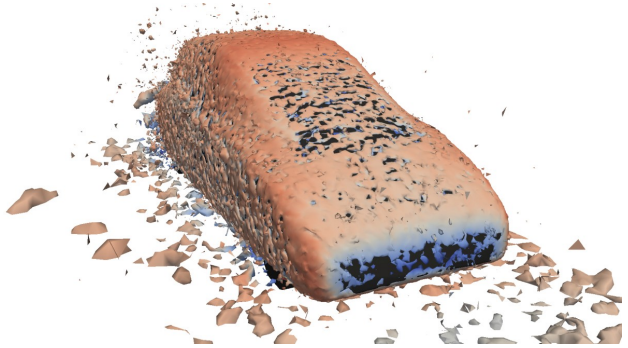
Termo Fluids

- **PRACE Preparatory access** is intended for testing and developing codes in order to prepare applications for PRACE Tier-0 systems

- **TermoFluids** was recently ported to the hybrid CPU/GPU model in the context of the **PRACE** preparatory access project:

  "Acceleration of TermoFluids code by means of GPU co-processors"

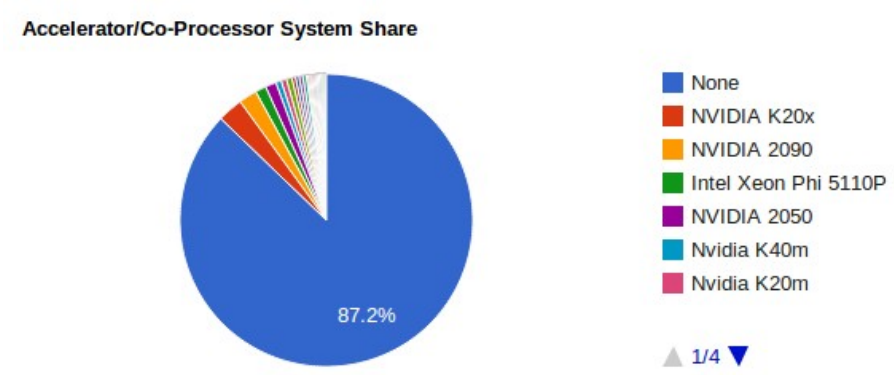
- Tests performed on the **TGCC Curie** hybrid nodes, based on:
  - 2 Intel Xeon E5640 quad-core processors
  - 2 Nvidia M2090

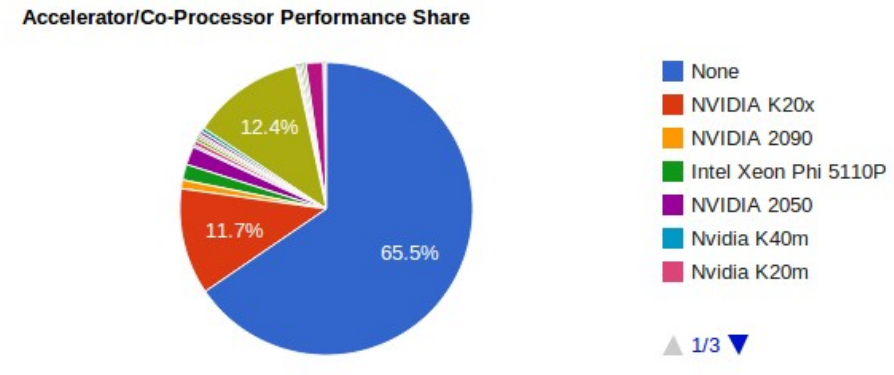


Accelerators in HPC

Termo Fluids

- Accelerators becoming increasingly popular in leading edge supercomputers
- Potential to significantly reduce space, **power consumption**, and cooling demands
- Context: Constrained power consumption target (~25MW for the entire system) → **power wall**





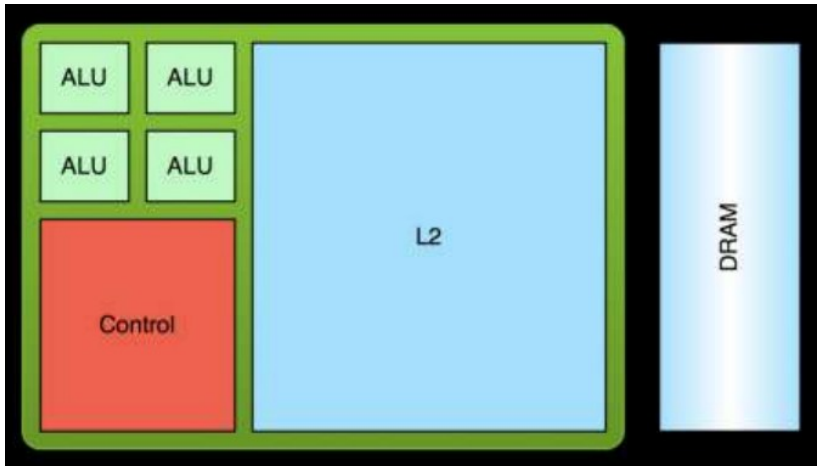
top500.org list June 2014

- **13%** of the Top500 list systems are based on hybrid nodes
- Considering the **first 15 positions** of Top500 list 8 (53%) are based in hybrid nodes
- **100%** of the **fist 15 positions** in the Green500 list are hybrid nodes with accelerators (NVIDIA)
- Those rankings are based on the High-performance LINPACK benchmark for dense linear systems but...

...in general PDEs systems are sparse!

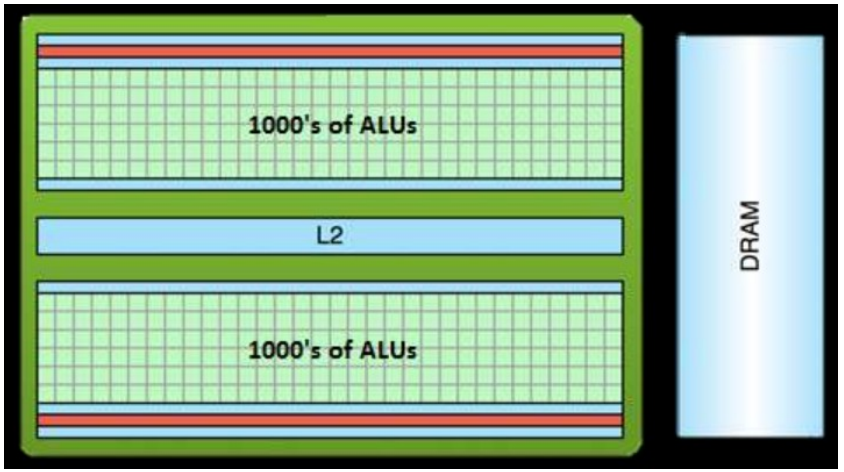
CPU vs GPU

Termo Fluids



Design goals for CPUs

- Make a single thread very fast
- Reduce latency through large caches
- Predict, speculate



Design goals for CPUs

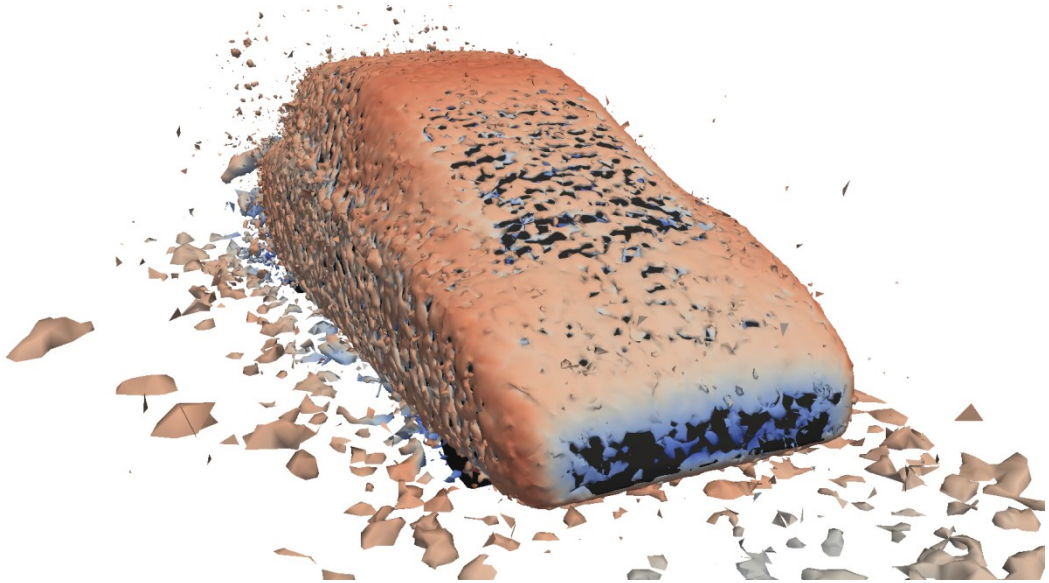
- Throughput matters and single threads do not
- More transistors dedicated to computation
- Hide memory latency through parallelism

- Remove modules to make simple instruction fast
  (out-of-order control logic, branch predictor logic, memory pre fetch unit)
- Share the cost of instruction stream across many ALUs (SIMD model)
- Multiple context per stream multiprocessor (SM) → concurrency

http://extremecomputingtraining.anl.gov/files/2014/01/Warburton-Accelerators14.pdf

PRACE Preparatory Access Project

Termo Fluids



- **Target problem**:
  - Flow around Asmo car
  - Re = 7e5
  - Mesh: 5.5 Milion CV, prismatic boundary layer and tetrahedral elements
  - Symmetry preserving finite volume discretization
  - Fractional step pressure-velocity coupling
  - Sub-grid scales: wall-adapting local-eddy viscosity (WALE)
  - Poisson solver: CG with Jacobi diagonal scaling

Flow and turbulent structures around simplified car models. D.E. Aljure, O. Lehmkuhl,, I. Rodríguez, A. Oliva. Computers & Fluids 96 (2014) 122–135

Enabling work

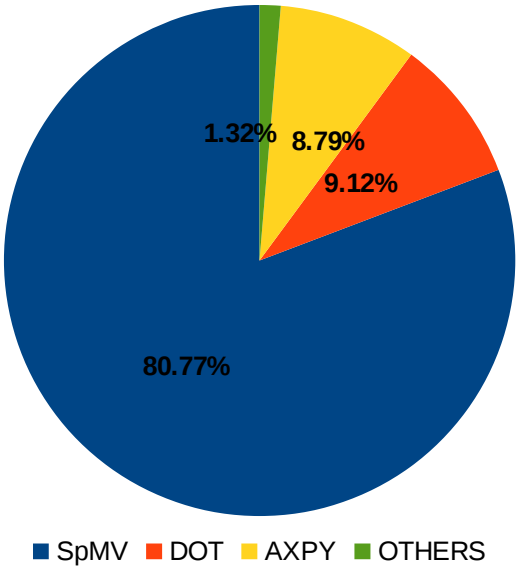
- **Code re-structuring:** conversion of **TermoFluids** momentum solver into an algebraic kernel based code
  - Loops around mesh elements assembled into sparse matrices (in a preprocessing stage)
  - Time-step integration based on algebraic kernels

- **Result:** most of time is spent in only three basic algebraic kernels:
  - SpMV: y ← A*x (A sparse matrix, x and y vectors)
  - AXPY: y ← ax+y (a scalar)
  - DOT: a ← x*y

- Easier **portability** of the time integration code, which dominates the execution costs

SpMV is the dominant kernel

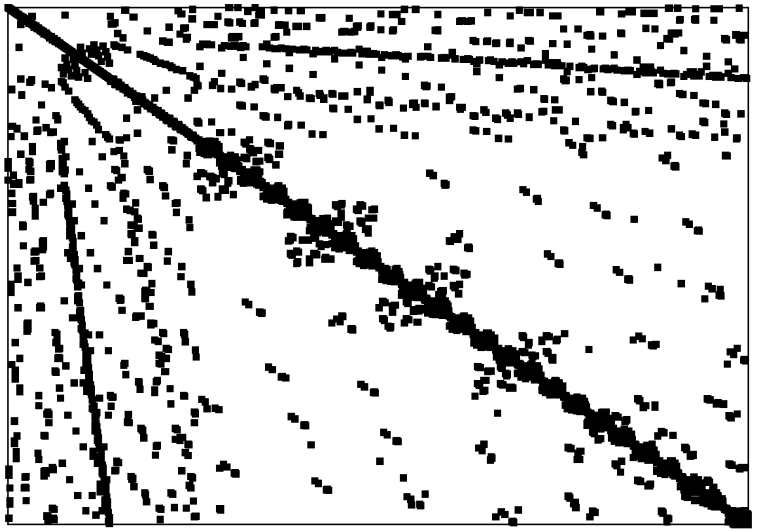
Outside CG	number
SpMV	23
AXPY	5
DOT	2
CG iteration	**number**
SpMV	2
AXPY	3
DOT	2



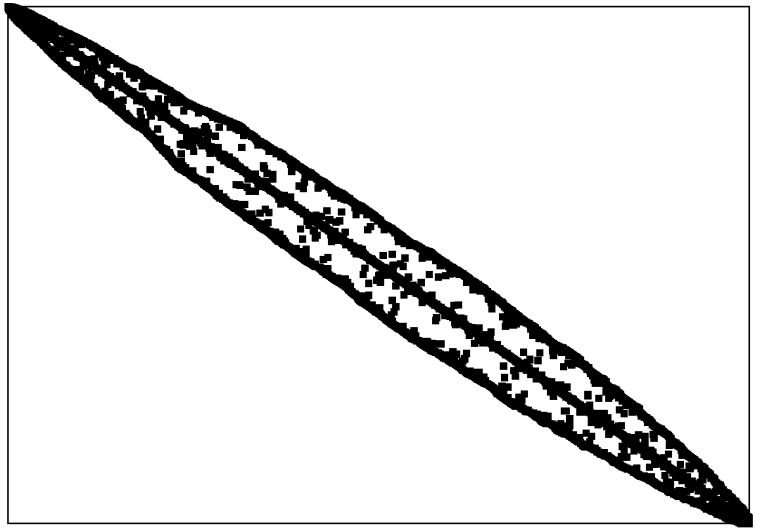
Test on target problem MPI implementation

SPMV kernel

Termo Fluids



No ordering



Cuthill - Mckee ordering

- **SpMV** kernel: **A*x=b** (A Laplacian operator)
- Tetrahedral mesh, 5 **entries per row**
- N **system size**, 5N **matrix entries**
- Storage format **ELLPACK:** 1 double (value) and 1 int (column index) per matrix entry
- **A bytes** in ELLPACK: 2*(8*5*N) = 80N
- **b bytes:** 8N
- **x bytes:** 8N (max cache reuse), 8*(5N) (no cache reuse)
- **SpMV bytes**: 2*(8*5*N) +2*(8N) = 96N – with max. cache reuse
  2*(8*5*N)+(8*5*N) + 8N = 128N – with max. cache reuse
- **SpMV flops**: 2*5N = 10N (a + and a * per matrix entry)

Performance Barrier CPU

Termo Fluids

- **Intel Xeon E5640** (4 core,Turbo Freq. 2.93 GHz, Bandwidth 25.6 GB/s)
- Sequential run of ELLPACK SpmV

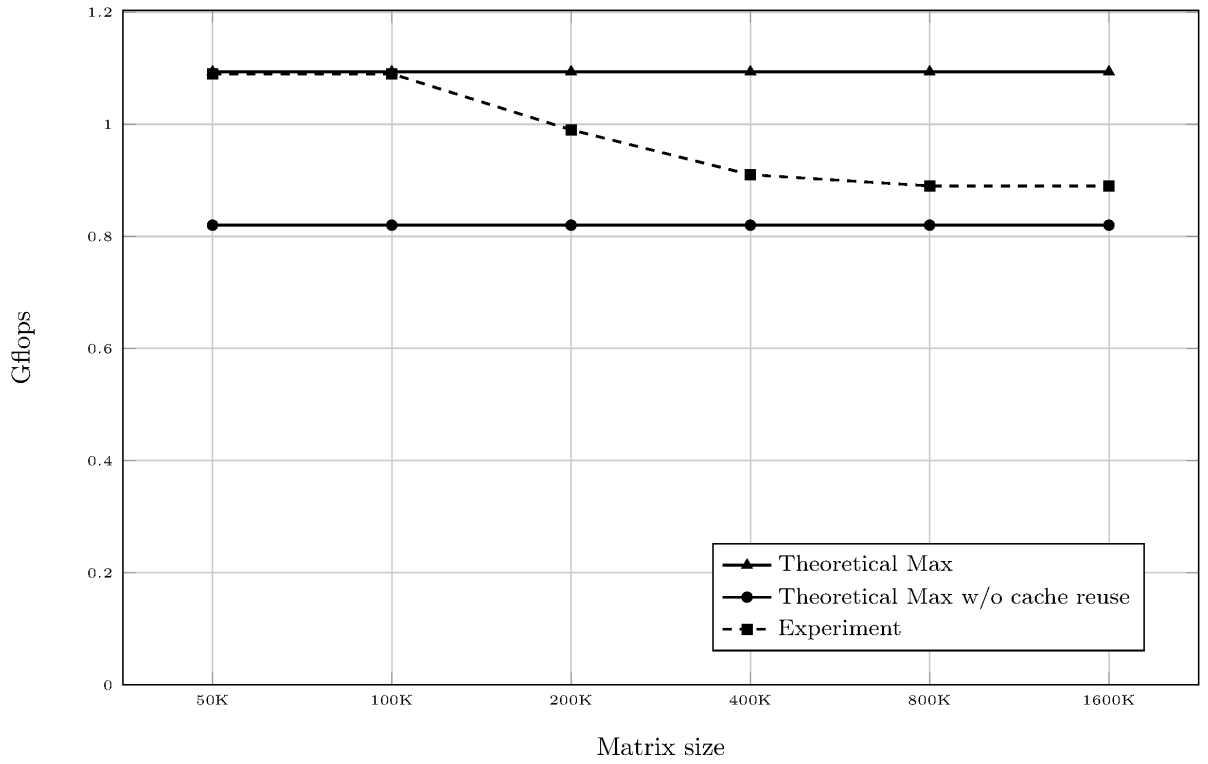
  - **Peak performance**: 4 flop/cycle x 2.93 G cycle/s = 11.72 Gflop/s (flops: 2 FMA + 2 SIMD)
  - **Peak bandwidth 1 thread**: 10.5 GB/s (STREAM test TGCC support team)
  - **Time computations:** 10N flop / 11.72 Gflop/s = 0.85N ns
  - **Time move data:** 96N bytes / 10.5 GB/s = 9.14N ns
  - **Ratio:** time_move / time_comp ≈ 10 !!
  - **Total time:** total_time ≥ max(time_move, time_comp)= 9.14N ns
  - **Achievable performance:**
    performance SpMV = 10N/ total_time ≤ 10N/9.14N = 1.09 Gflop/s

NO MORE THAN 9.3% OF CPU CORE PERFORMANCE CAN BE ACHIEVED!!

Performance Barrier CPU

Termo Fluids

- **Intel Xeon E5640** (4 core,Turbo Freq. 2.93 GHz, Bandwidth 25.6 GB/s)
- Sequential run of ELLPACK SpmV – REAL MEASURMENTS



- Maximal performance without cache reuse = 10 / 128 flops/byte * 10.5 Gbytes/s = 0.82 Gflop/s
- **Results are in agreement with the expected performance**

Performance Barrier GPU

Termo Fluids

- **NVIDIA M2090** (Tesla)
- ELLPACK SpmV on one GPU – implementation on CUDA 5.5

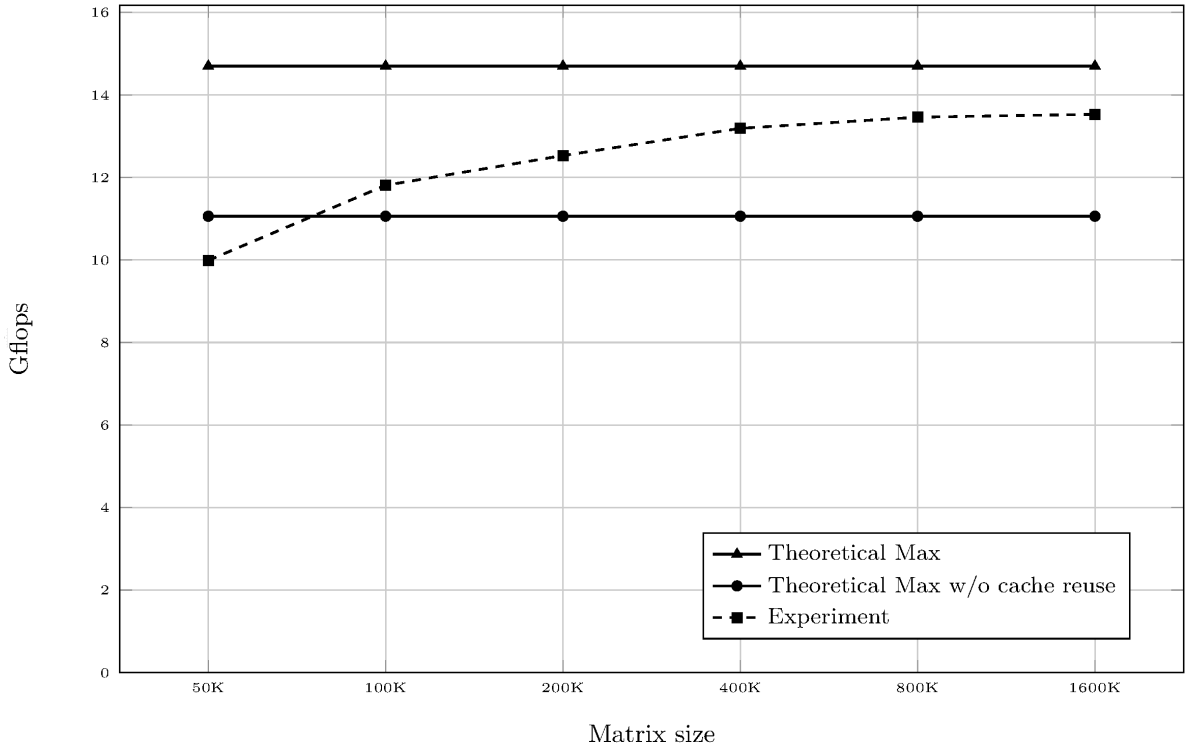
  - **Peak performance**: 666.1 Gflop/s
  - **Bandwidth**: 0.8*177 = 141.6 GB/s (20% reduction caused by ECC – NVIDIA best prac. guide)
  - **Time computations:** 10N flop / 666.1 Gflops = 0.015N ns
  - **Time move data:** 96N bytes / 141.6 GB/s = 0.68N ns
  - **Ratio:** time_move / time_comp ≈ 45!!
  - **Total time:** total_time ≥ max(time_move, time_comp)= 0.68N ns
  - **Achievable performance:**
    performance SpMV = 10N/ total_time ≤ 10N/0.68N = 14.7 Gflop/s

NO MORE THAN 2.2% OF GPU PERFORMANCE CAN BE ACHIEVED!!

Performance Barrier CPU

Termo Fluids

- **NVIDIA M2090** (Tesla)
- ELLPACK SpmV on one GPU (CUDA) – REAL MEASURMENTS

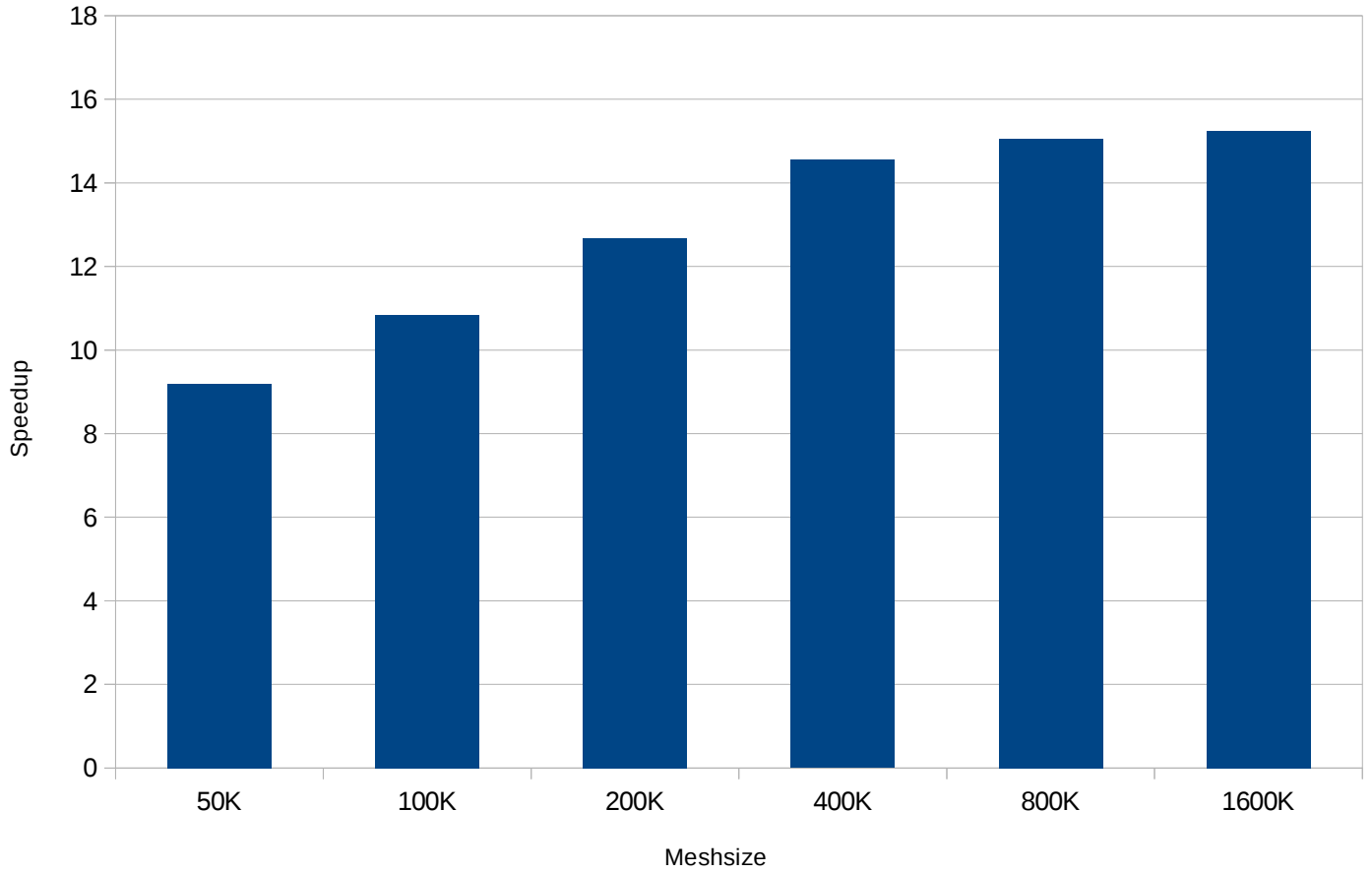


- Reaching the peak bandwidth requires certain occupancy
- **Results are in agreement with the expected performance**

GPU vs CPU sequential

Termo Fluids

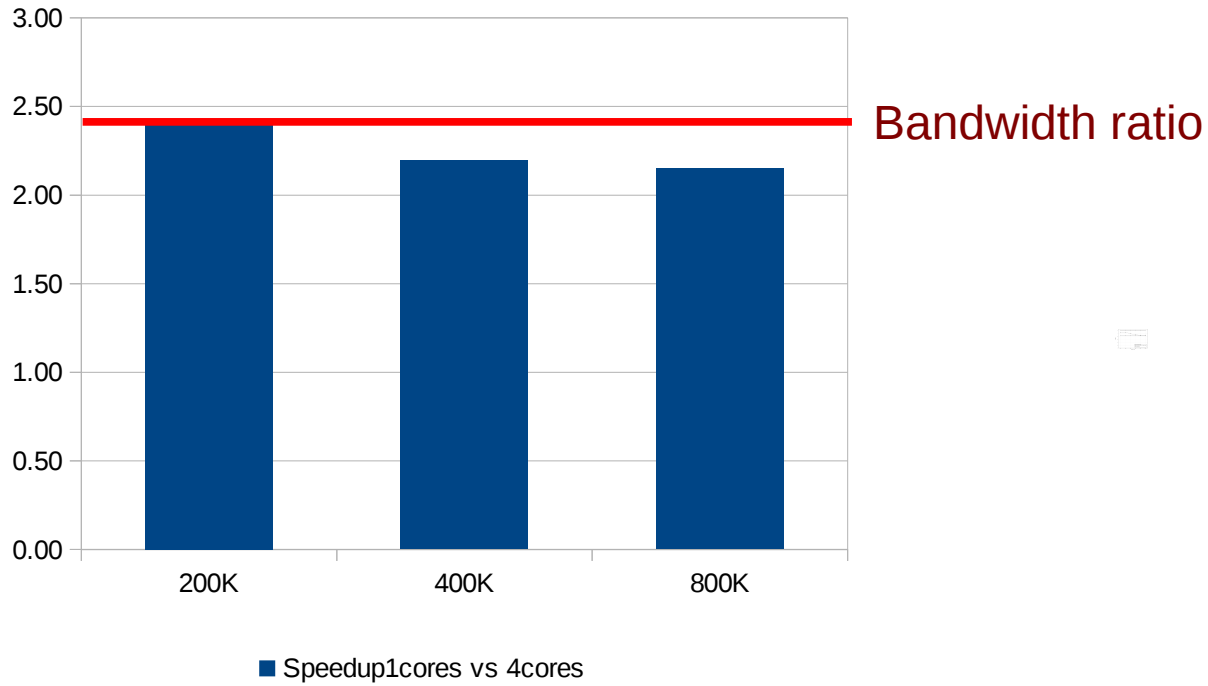
- Single core **Intel Xeon E5640** vs **NVIDIA M2090**



- Corollary of previous measurements: **speed up from 9x to 15x**

Multi-core CPU

- **Intel Xeon E5640**
- Speedup form 1 to 4 CPU-cores (inter-core communication not included)

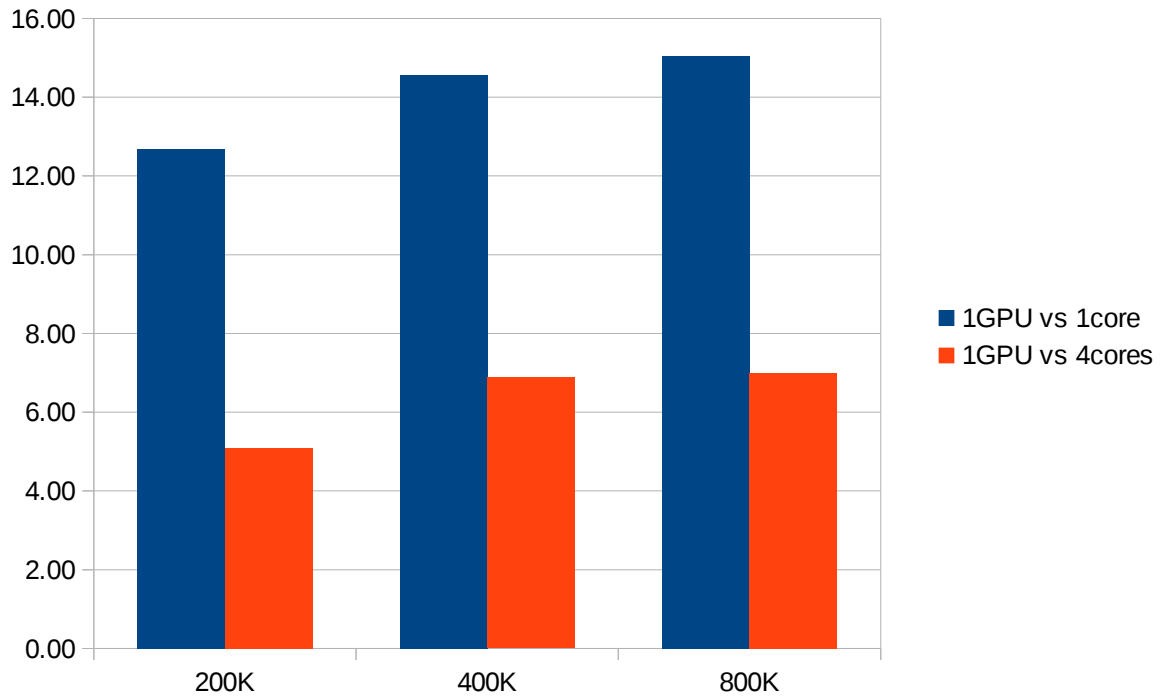


- **Potential Speedup = bandwidth increase** (only bandwidth matters!!)
- Bandwidth 1-core: 10.5 GB/s; 4-core (a CPU): 25.6 GB/s; ratio= 2.44x (<4x!!)
- The achievable performance using all 4 CPU cores reduces at ~ 6%
- **Results are in agreement with the expected performance**

GPU vs multi-core CPU

Termo Fluids

- ELLPACK SpMV **Intel Xeon E5640** vs **NVIDIA M2090**

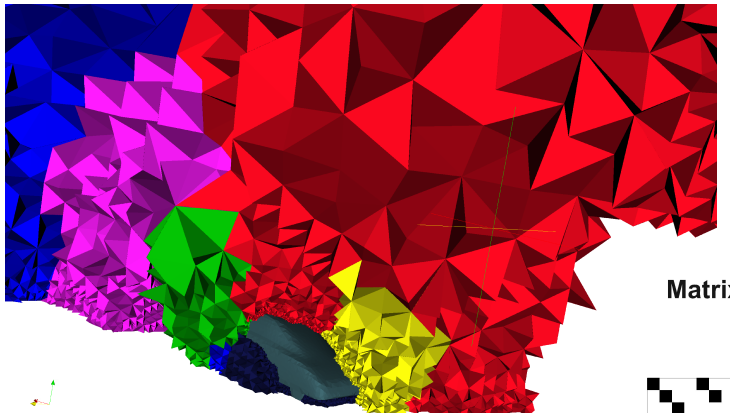


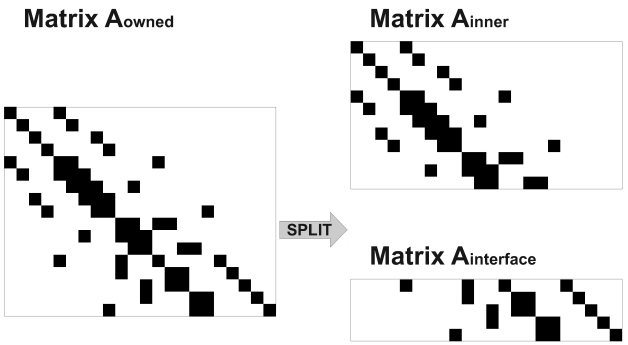
- Speedup considering all 4-cores ranges from 5x to 7x
- Ratio between peak bandwidth of both systems: 141,6 (gpu) /25.6 (cpu) = 5.5x
- GPU uses more effectively its bandwidth for higher sizes and CPU for lower ones

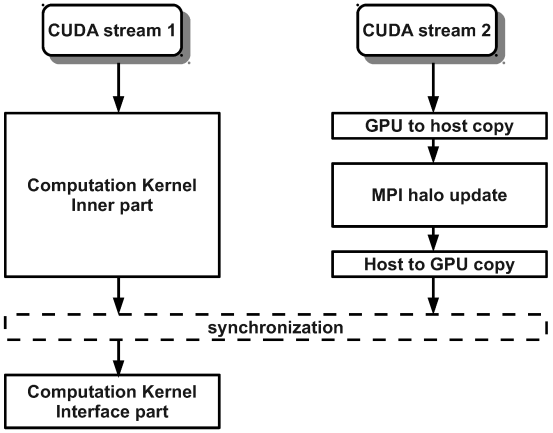
MEMORY AWARE PROGRAMMING!!

Multi GPU implementation

Termo Fluids



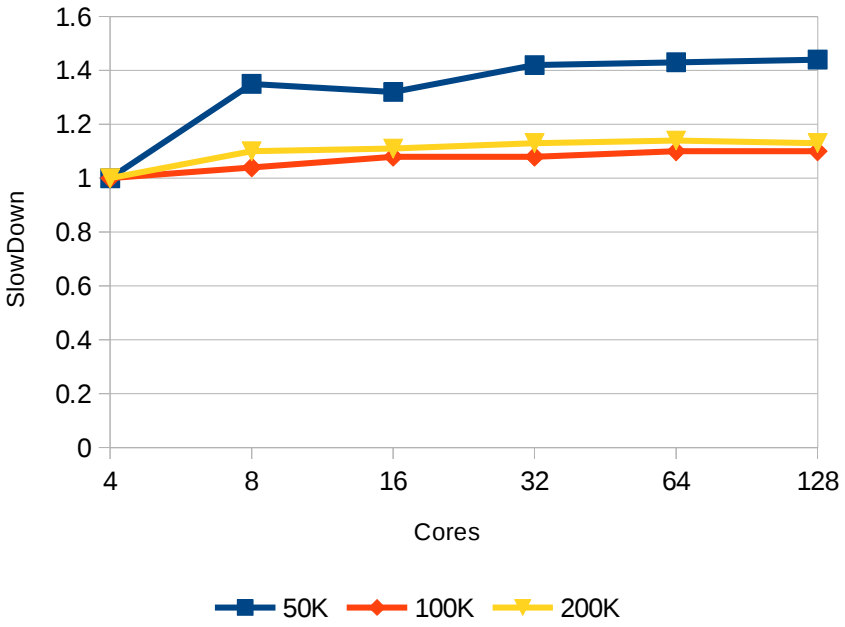


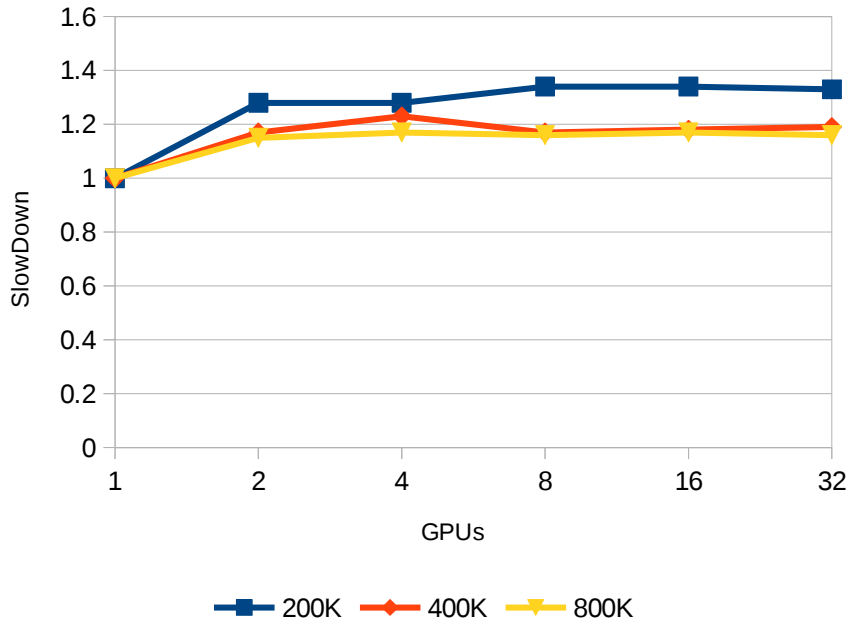


- **MPI + CUDA** implementation
- Parallelization based on a domain decomposition
- One MPI-thread per subdomain and one GPU per MPI-thread
- **Local data partition:** separate **inner** parts (do not require data form other subdomains) from **interface** parts (require external elements)
- **Local data partition + two stream model** -> **overlapping** computations on GPU with communications

Multi – GPU tests

- Comparison **weak speedup** multi-CPU vs multi-GPU (inter CPU/GPU comm included)

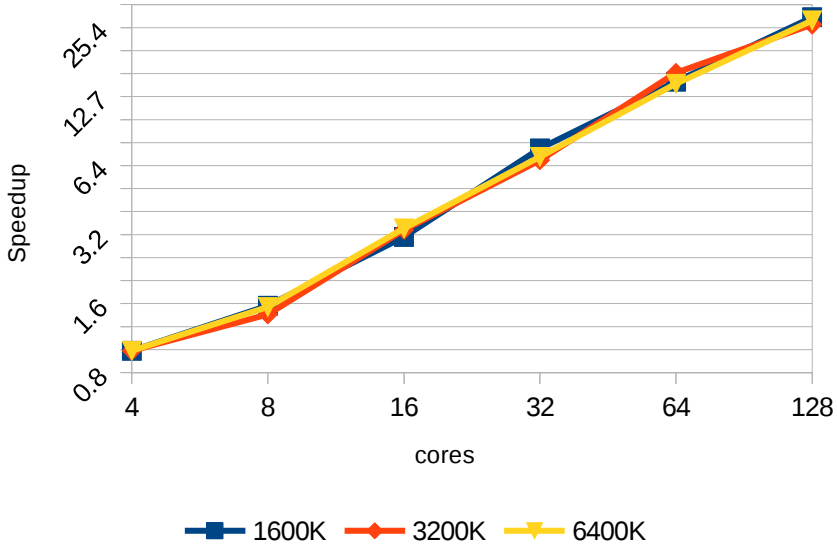


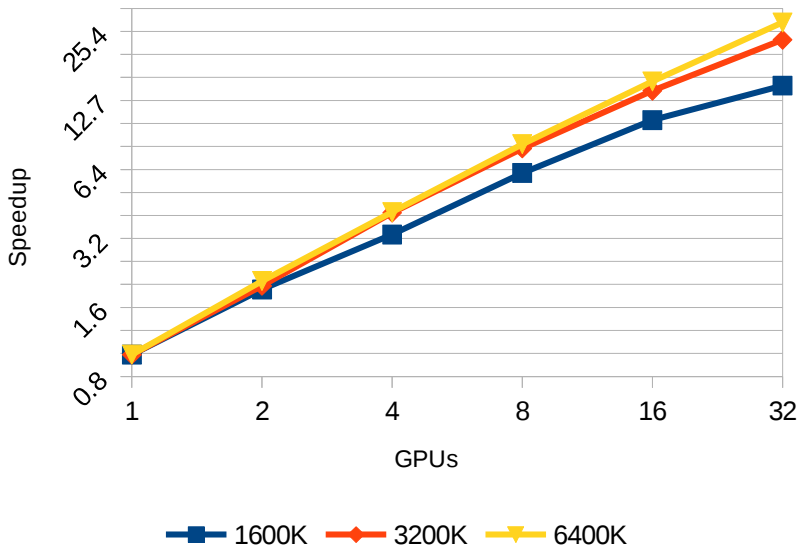


- **Similar results despite muti-GPU implementation is from 5x to 7x faster!!** causes:
  - Number of MPI-threads is 4 times lower in multi-GPU implementation → less communications overhead
  - Overlapping is more effective in multi-GPU implementation, since communication and computations are running on independent devices

Multi – GPU tests

- Comparison of **weak speedup**, multi-CPU vs multi-GPU (inter CPU/GPU comm included)

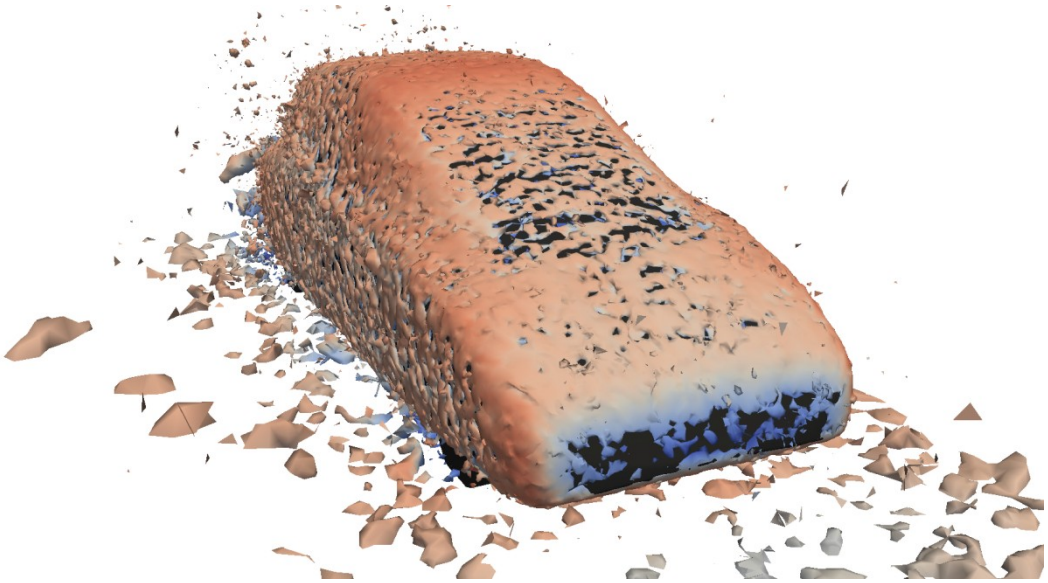


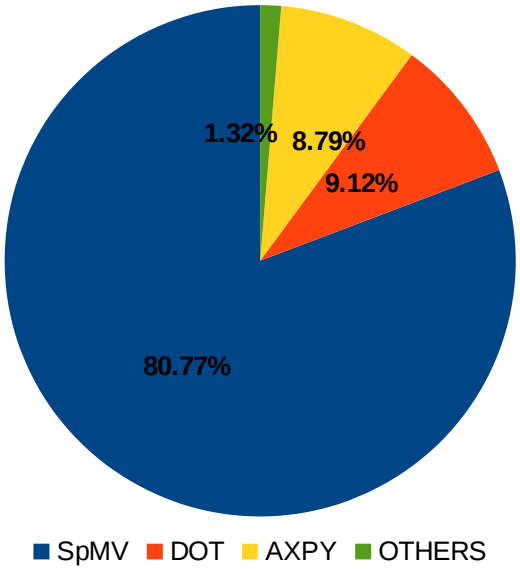


- Local size with 32 GPUs / 128 CPUs: 50K, 100K and 200K respectively
- The local problem reduction decreases the bandwidth performance on GPU (less occupancy)
- The bandwidth performance does not decrease on the CPUs and, additionally, improves the cache reuse

Tests on TARGET CASE

Termo Fluids

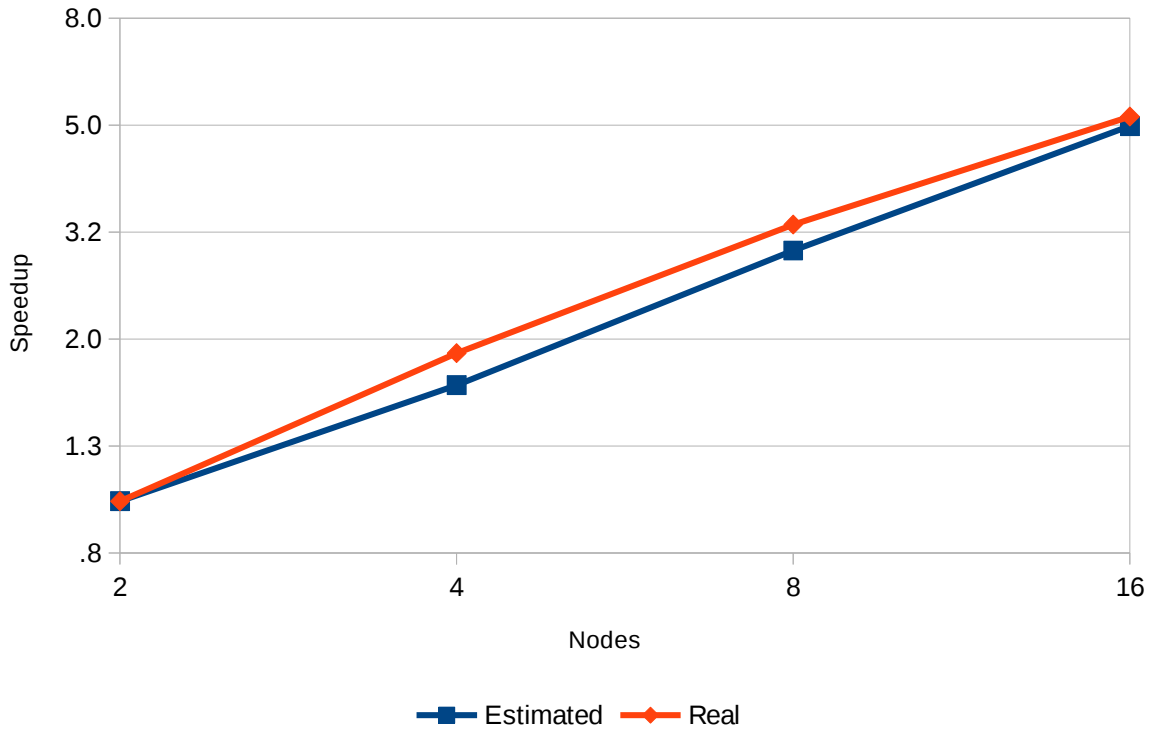


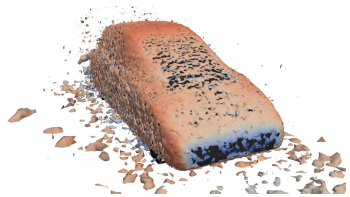


- Flow around Asmo car, Re=7e5
- Mesh 5.5 million CV
- Prismatic boundary layer → JAGGED storage format (sliced ELLPACK)
- AXPY and DOT on mkblas 14.0.3.174 and cublas 5.0
- From 1 to 16 nodes used → from 2 to 32 GPUs - from 8 to 128 CPUs

Tests on TARGET CASE

- **Strong speedup** multi-GPU for the implementation of momentum solver

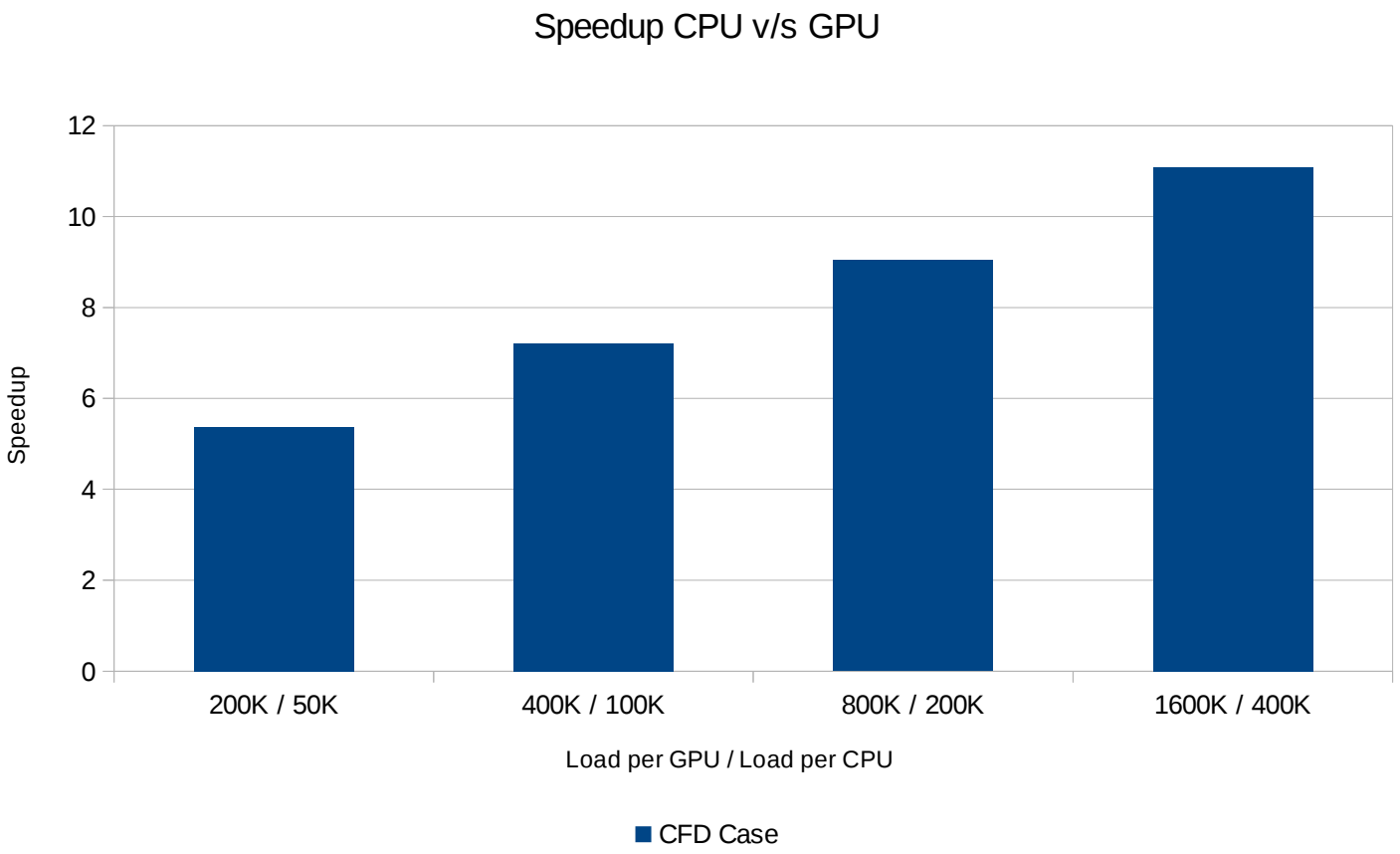


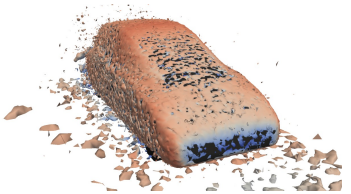


- **63% PE achieved** (with 32 GPUs load per GPU < 172K cells)
- Performance can be estimated according: i) number of repetitions of each kernel per time step, ii) parallel performance of kernels (SpMV, AXPY, DOT)
- "Good" agreement between estimation and real measurements → **performance could be estimated in any system by only studying it for the basic kernels**

Tests on TARGET CASE

- Speedup muti-GPU vs multi-CPU for the momentum solver





- **Speedup ranges between 5x and 11x**, depending on the workload

CONCLUDING REMARKS

- Initial goal was accelerating **TermoFluids** by means of GPUs but...

  - We have developed a portable version of the code based on an algebraic operational approach
  - We have seen that 98% is spent on three kernels: SpMV, AXPY, DOT
  - SpMV dominates the execution with 81% in our target application
  - The three algebraic kernels are memory bounded, performance depends exclusively on the bandwidth achieved
  - Our SpMV implementation on single CPU-core and single GPU show the expected performance
  - Similar parallel performance is achieved for both multi-CPU and multi-GPU implementations
  - Parallel performance of the overall time step execution is estimable from performance of the basic kernels
  - Speedup of multi-GPU vs multi-CPU implementation in our target problems ranges from 5x to 11x

CONCLUDING REMARCS

Termo Fluids

- **Ongoing work:**
  - Porting other physics solvers to GPUs
  - Overlapping GPU and CPU solvers in multi-physics applications
  - Looking for the trade-off between code performance and programmers performance (our engineers are more used to mesh loops rather than algebraic kernels to implement discretizations)

Acknowledgments

PRACE Preparatory Access

TGCC Curie support team

THANK YOU FOR YOUR ATTENTION!