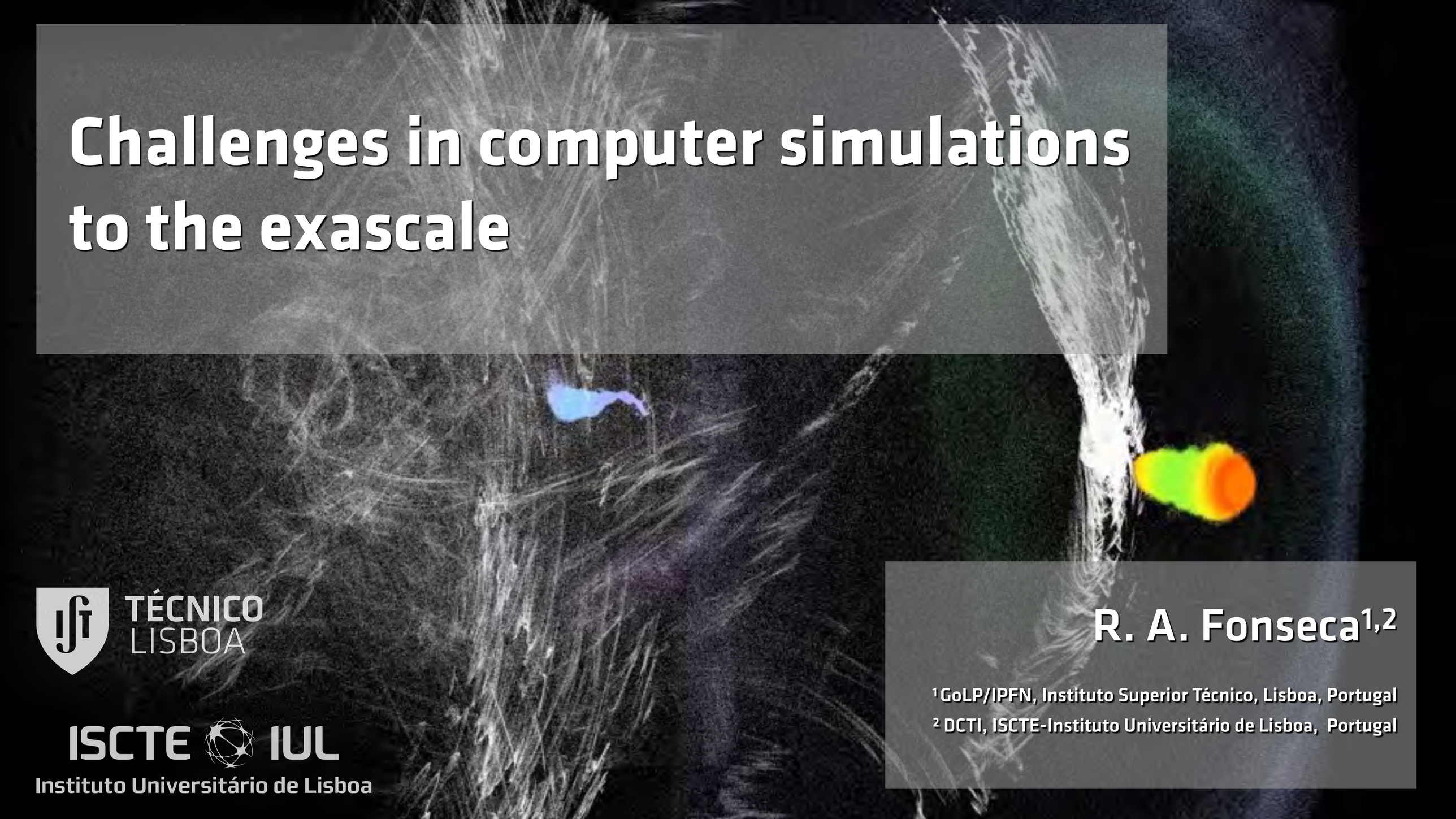
Challenges in computer simulations to the exascale



TÉCNICO LISBOA

ISCTE IUL
Instituto Universitário de Lisboa

R. A. Fonseca[1,2]

[1] GoLP/IPFN, Instituto Superior Técnico, Lisboa, Portugal
[2] DCTI, ISCTE-Instituto Universitário de Lisboa, Portugal

Acknowledgements

IST

- J. L. Martins, T. Grismayer, J. Vieira, P. Ratinho, K. Schoeffler, M. Vranic, U. Sinha, T. Mehrling, A. Helm, L. O. Silva

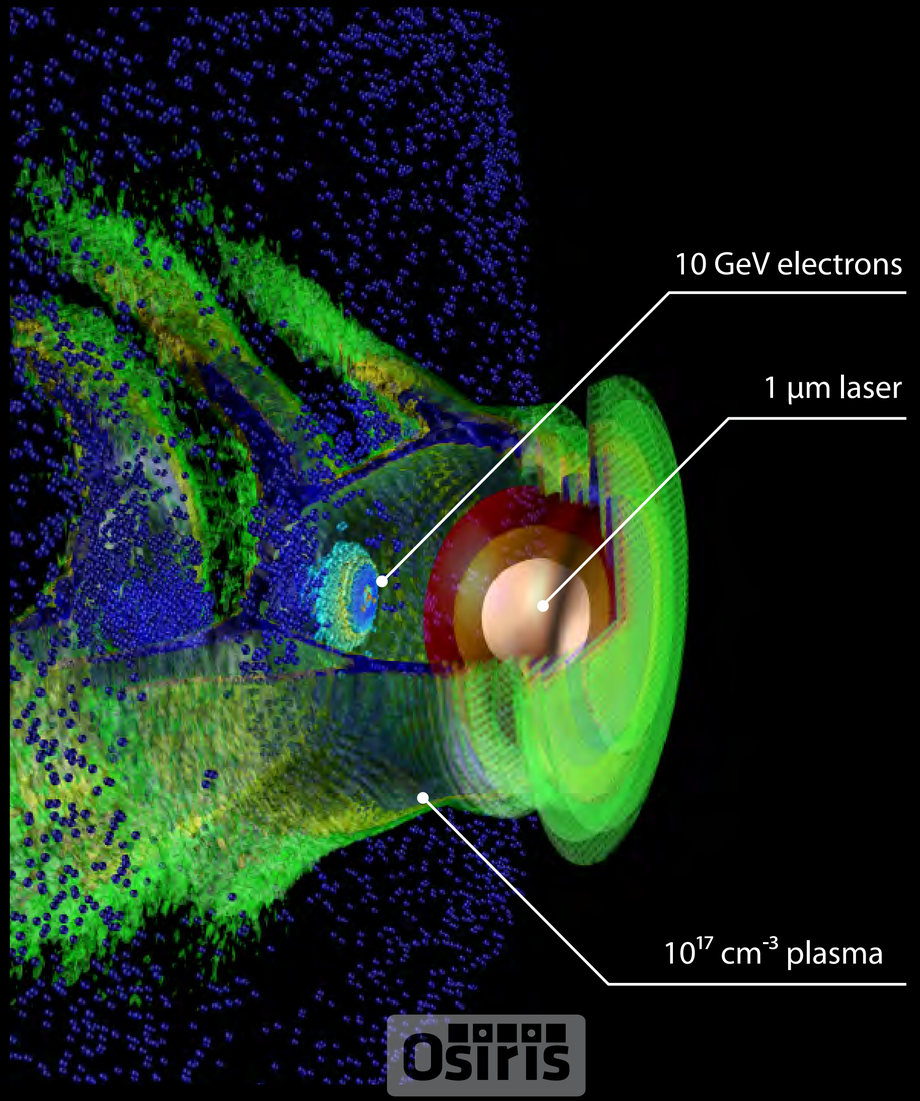
UCLA

- A. Tableman, A. Davidson, P. Yu, T. Dalichaouch, F. Tsung, V.K.Decyk, W. B. Mori, C. Joshi

Simulation results

- Accelerates Clusters (IST), Dawson/Hoffman Clusters (UCLA), Jugene/Juqueen (FZ Jülich), Jaguar (ORNL), SuperMuc (LRZ), BlueWaters (NCSA), Sequoia (LLNL)

Why Exascale computing?



High-Intensity Laser-Plasma Interaction
- Particle Acceleration
- Radiation sources

Multi-scale problems
- Large disparity of spatial/temporal scales

Sample problem: 10 GeV LWFA stage
- $\lambda_0 \sim 1\ \mu m$
- $L \sim 0.5\ m$

Computational Requirements
- ~ 10^9 grid cells
- ~ 10^{10} particles
- Iterations ~ 10^6 - 10^7
- Memory ~ 1 - 10 TB
- Operations ~ 10^{18} - 10^{19}

Exascale performance
- Simulation time ~ 10s

Community of Particle-in-cell codes
- ALaDyn
- Calder
- EPOCH
- *HiPACE*
- INF&RNO
- OSIRIS
- PICADOR
- PIConGPU
- PSC
- *QuickPIC*
- SMILEI
- turboWAVE
- UPIC-EMMA
- VLPL
- Vorpal
- VPIC
- WARP
- ...

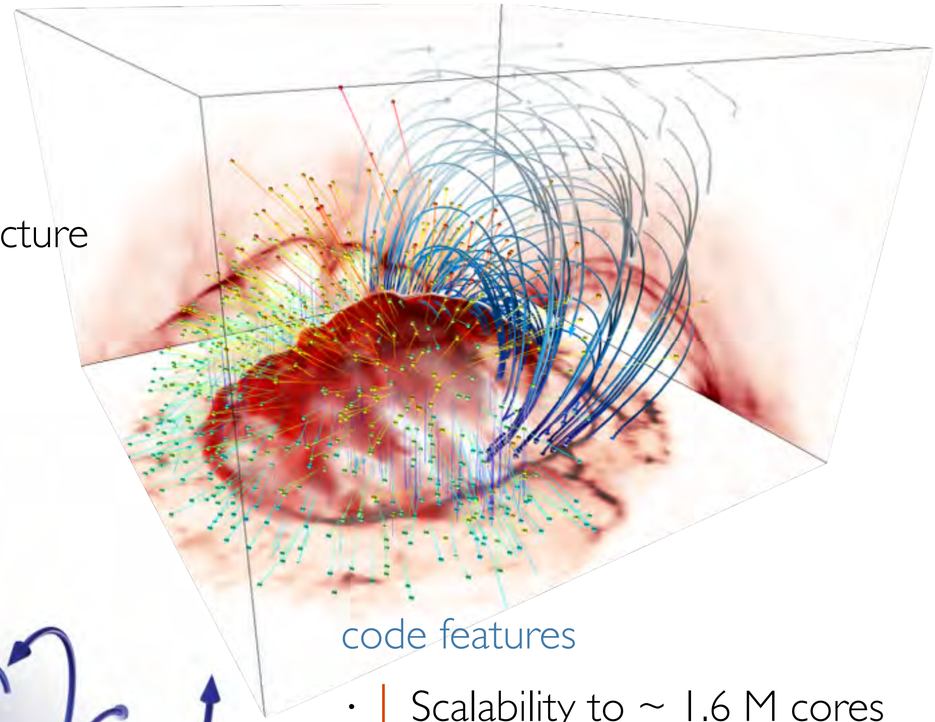
osiris 3.0

Osiris 3.0

osiris framework

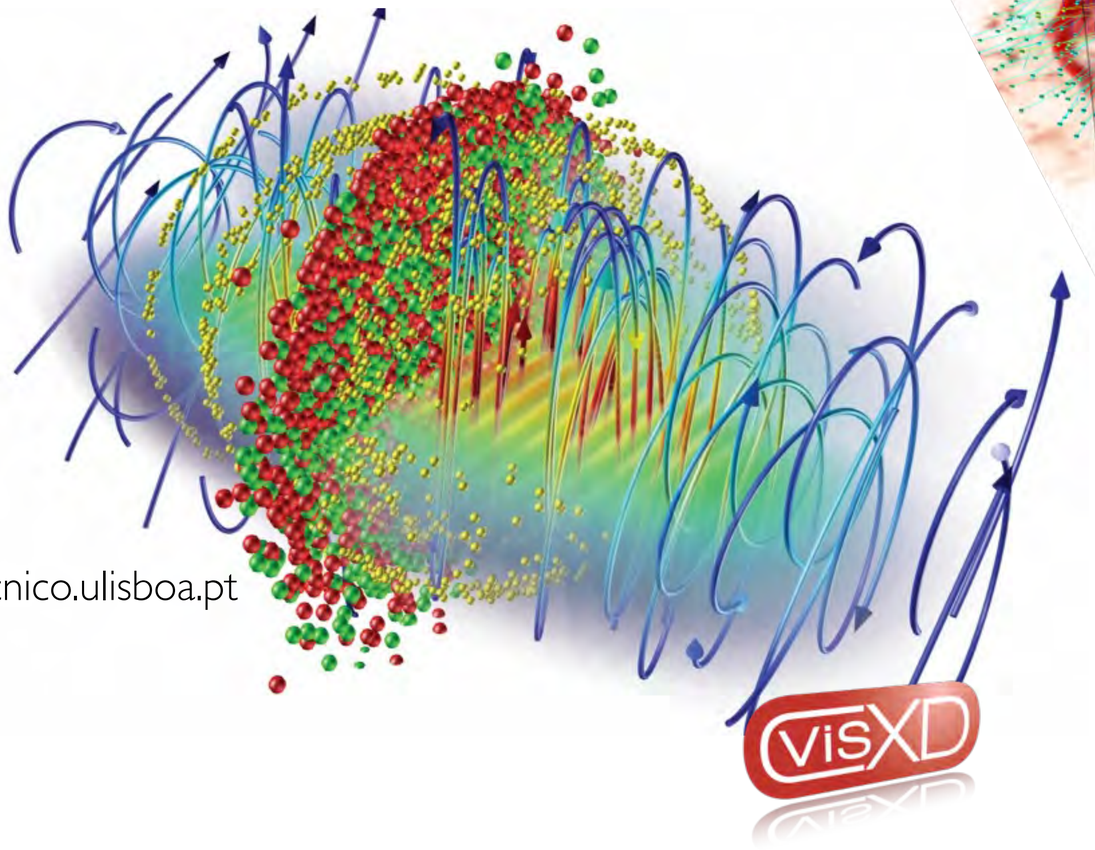
- Massivelly Parallel, Fully Relativistic Particle-in-Cell (PIC) Code
- Visualization and Data Analysis Infrastructure
- Developed by the osiris.consortium

  ⇒ UCLA + IST



TÉCNICO LISBOA

UCLA



code features

- Scalability to ~ 1.6 M cores
- **SIMD hardware optimized**
- Parallel I/O
- **Dynamic Load Balancing**
- QED module
- Particle merging
- **GPGPU support**
- **Xeon Phi support**

Ricardo Fonseca: ricardo.fonseca@tecnico.ulisboa.pt

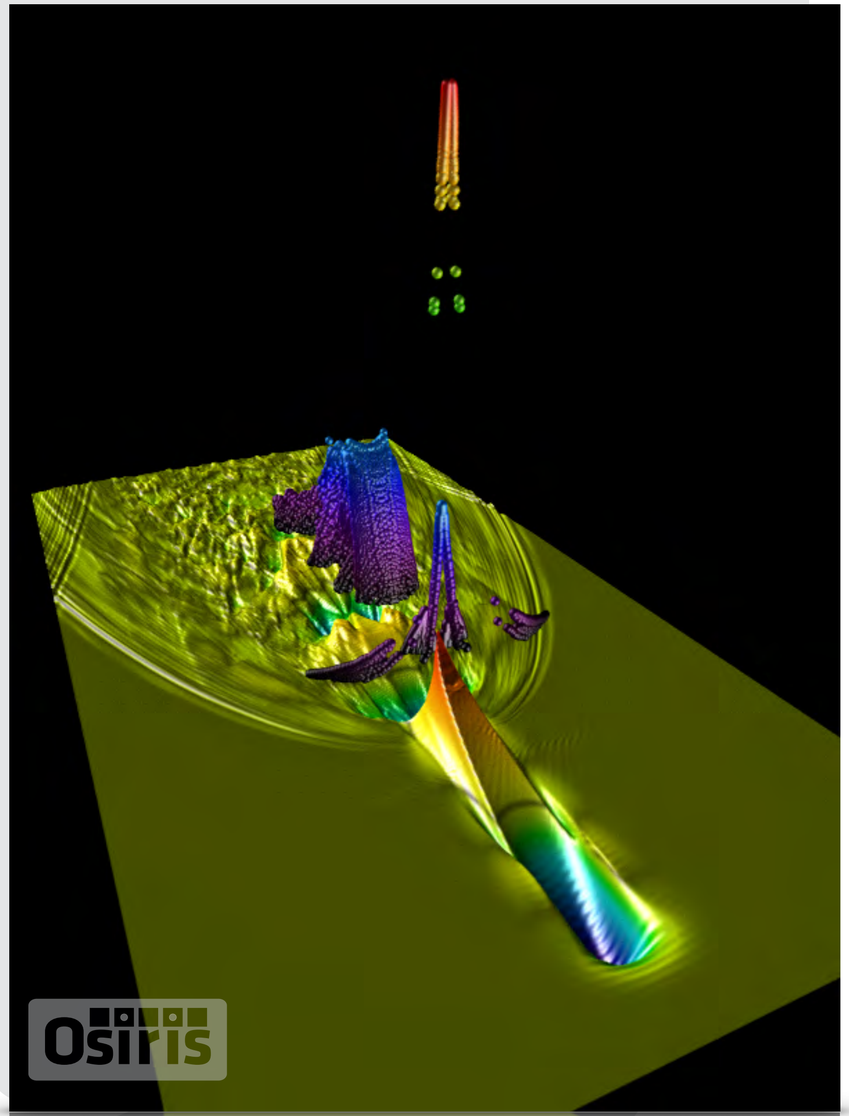
Frank Tsung: tsung@physics.ucla.edu

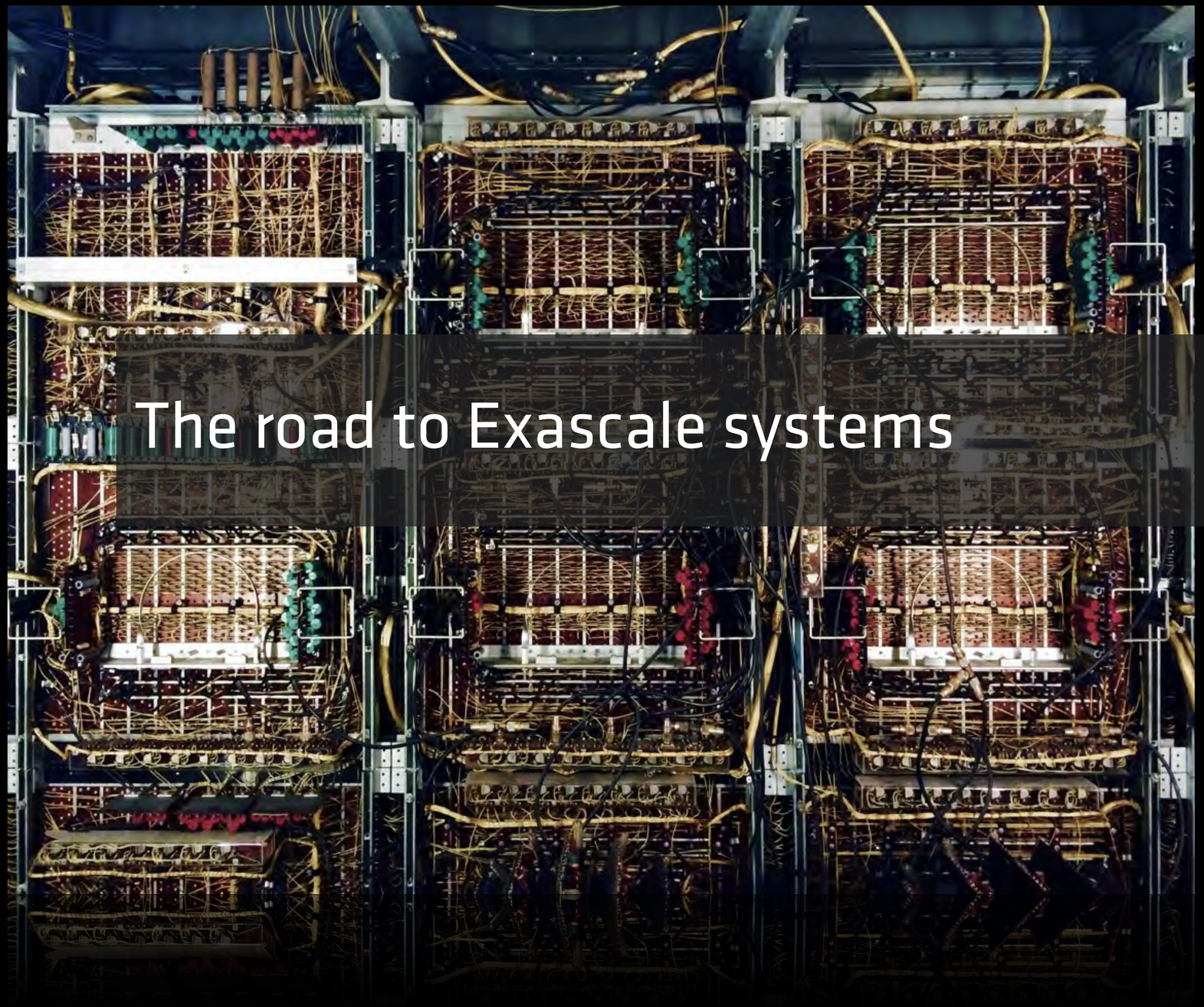
http://epp.tecnico.ulisboa.pt/

http://plasmasim.physics.ucla.edu/

Outline

- The road to Exascale Systems
  - HPC system evolution
  - Current trends
  - Multi scale parallelism
- Deploying the on large scale HPC systems
  - Parallelization strategies
  - Dealing with load imbalance
- Recent and future architectures
  - General purpose GPUs
  - Intel manycore (MIC)
- Overview





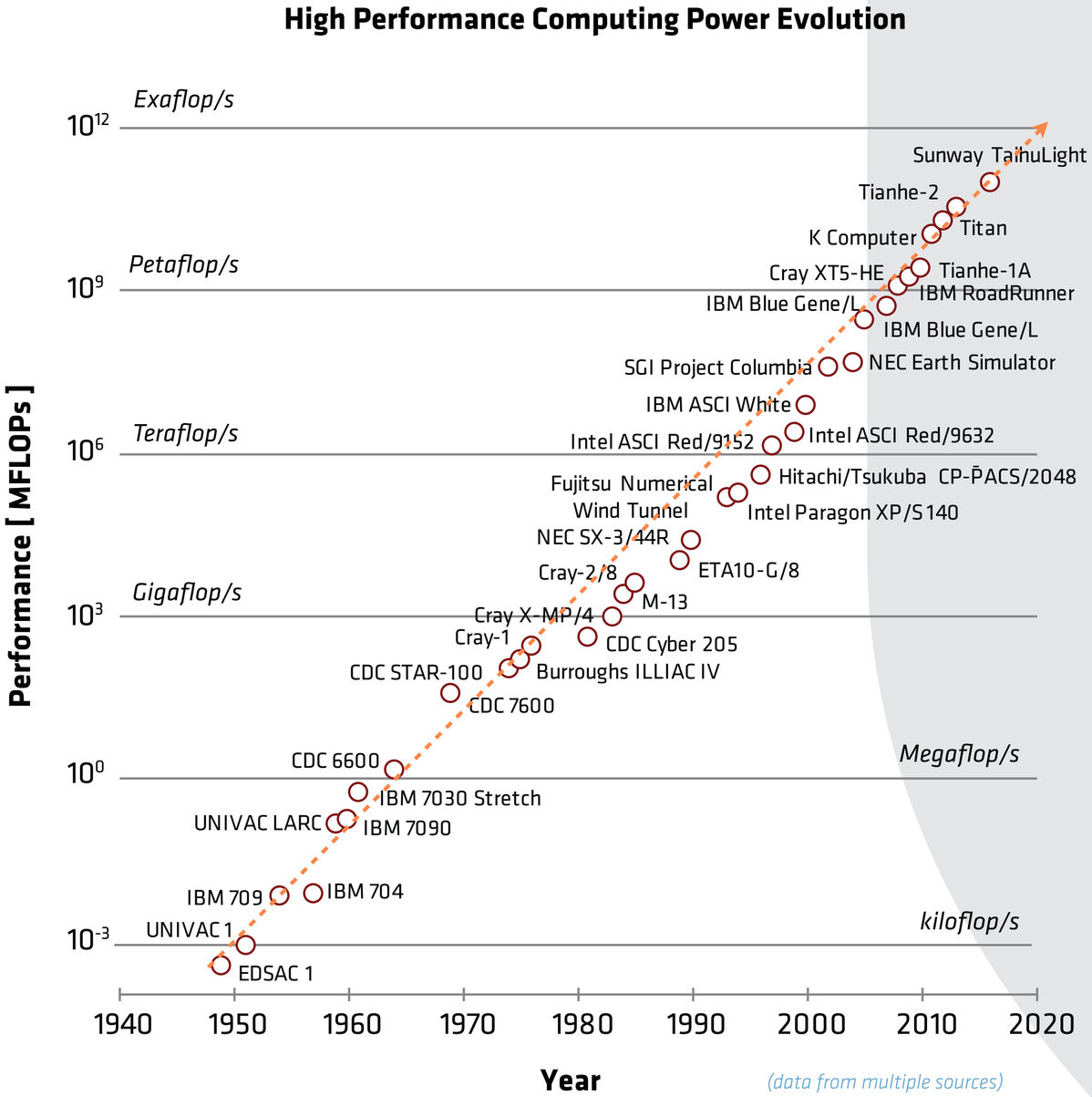
The road to Exascale systems

TÉCNICO LISBOA

UNIVAC 1 - 1951

Internal view

The Road to Exascale Computing



Sunway Taihulight
NRCPC, China
#1 - TOP500 Jun/16

Sunway Taihulight

- 40 960 compute nodes

Node Configuration

- 1× SW26010 manycore processor
  - 4×(64+1) cores @ 1.45 GHz
- 4× 8 GB DDR3

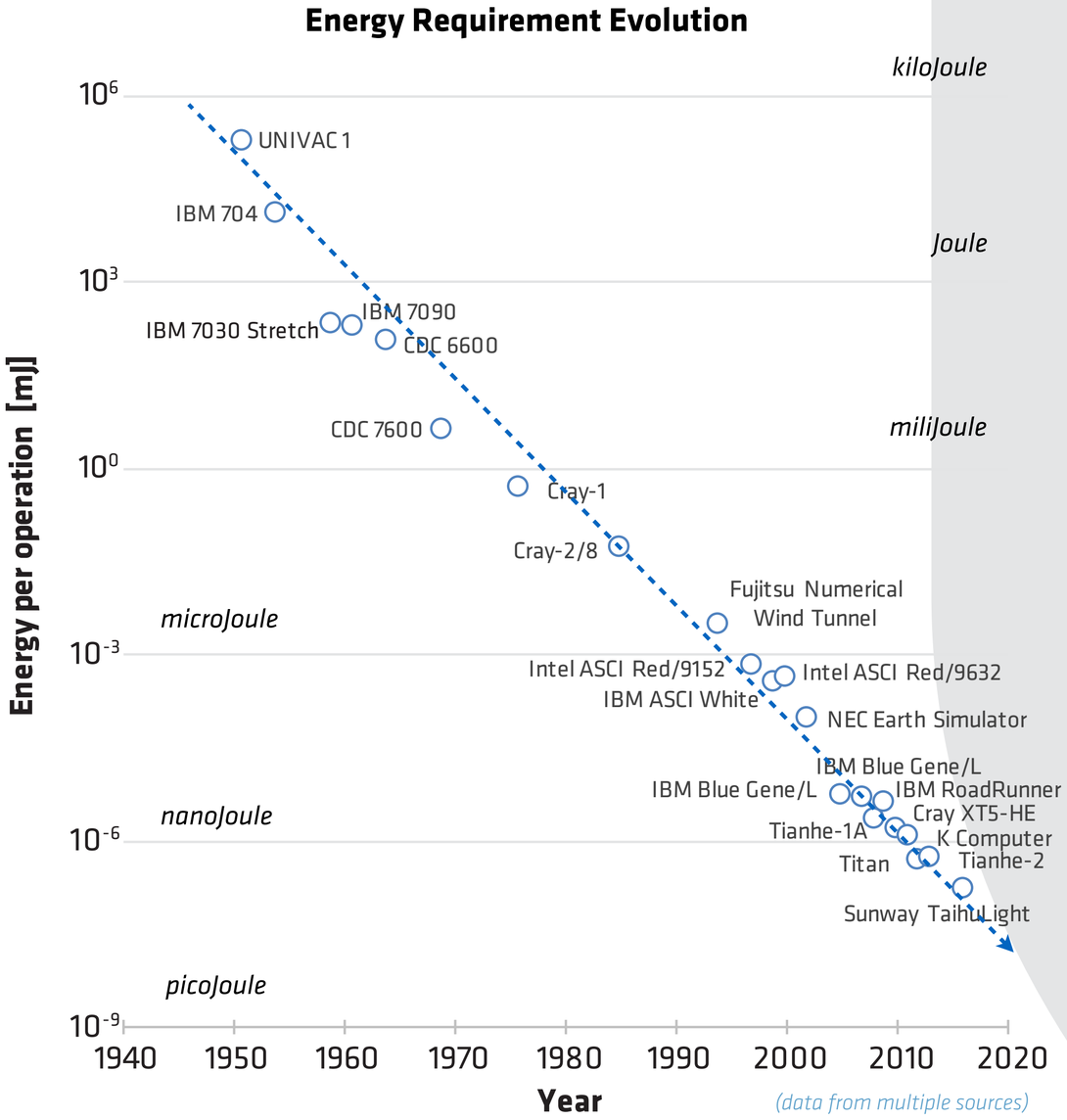
Total system

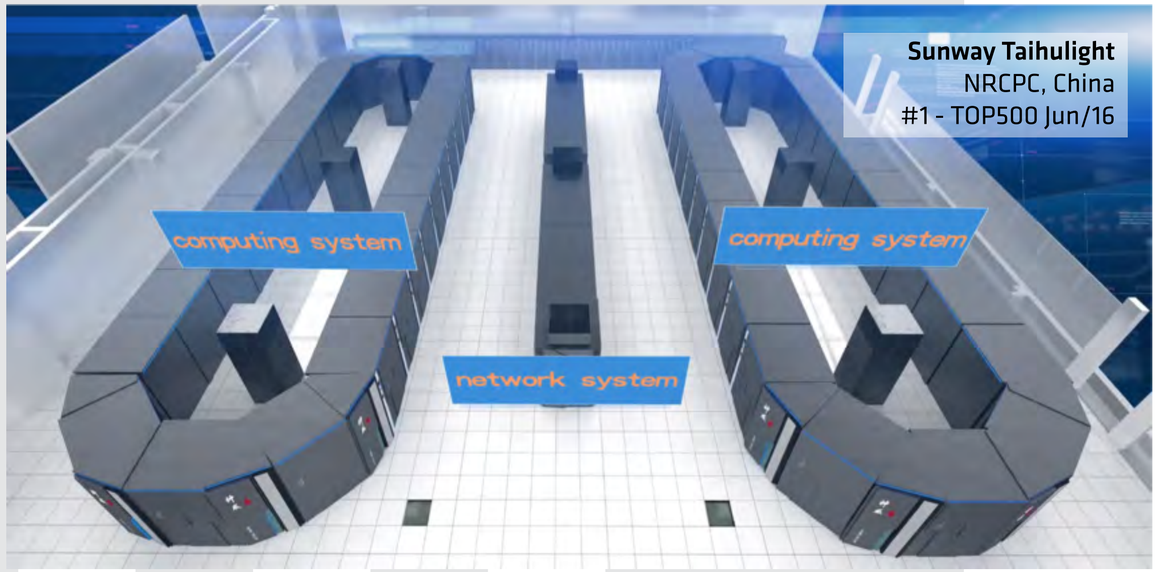
- 10 649 600 cores
- 1.31 PB RAM

Performance

- R_{peak} 125.4 Pflop/s
- R_{max} 93.0 Pflop/s

The Road to Power Efficient Computing





Sunway Taihulight
NRCPC, China
#1 - TOP500 Jun/16

Sunway Taihulight

- Manycore architecture
- Peak performance 93 PFlop/s
- Total power 15.3 MW
  - 6.07 Gflop/W
  - 165 pJ / flop

Petaflop systems firmly established

The drive towards Exaflop

- Steady progress for over 60 years
  - 95 systems above 1 PFlop/s
- Supported by many computing paradigm evolutions
- Trend indicates Exaflop systems by next decade
- Electric power is one of the limiting factors
  - Target < 20 MW
  - Top system achieves ~ 6 Gflop/W
    - ~ 0.2 GW for 1 Exaflop
    - Factor of 10× improvement still required
  - Best energy efficiency
    - 7.0 Gflop/W
    - PEZY-SC accelerator

Multicore systems

- Maintain complex cores and replicate
- 4 systems in the top 10 are based on multicore CPUs
  - 1× Fujitsu SPARK
  - 3× Intel Xeon E5

Manicore

- Use many (simpler) low power cores
- IBM BlueGene/Q Architecture has 2 systems in the top 6
  - *Seem to be the last of their kind*
- #1 (Sunway Taihulight) and future Intel Knights Landing systems

Accelerator/co-processor technology

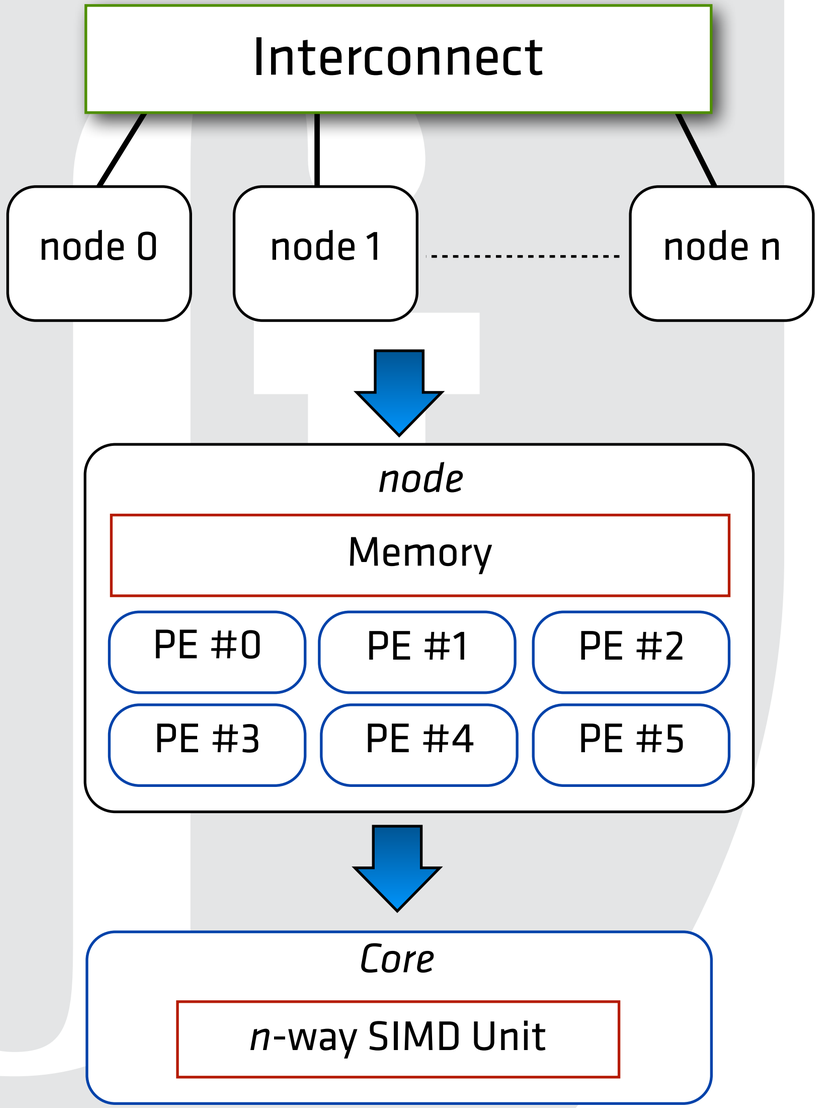
- 93 systems on top500 (jun 2016) use accelerator hardware
  - down from 104 in previous list (nov 2015)
  - 66 use NVIDIA GPUs, 27 use Intel MIC, 3 use ATI Radeon and 2 use PEZI-SC
- 3 systems in top 10
  - #3 (Titan) and #8 (Piz Daint) use NVIDIA GPUs
  - #2 (Tianhe-2) uses Intel MIC

Simple hardware abstraction for HPC systems

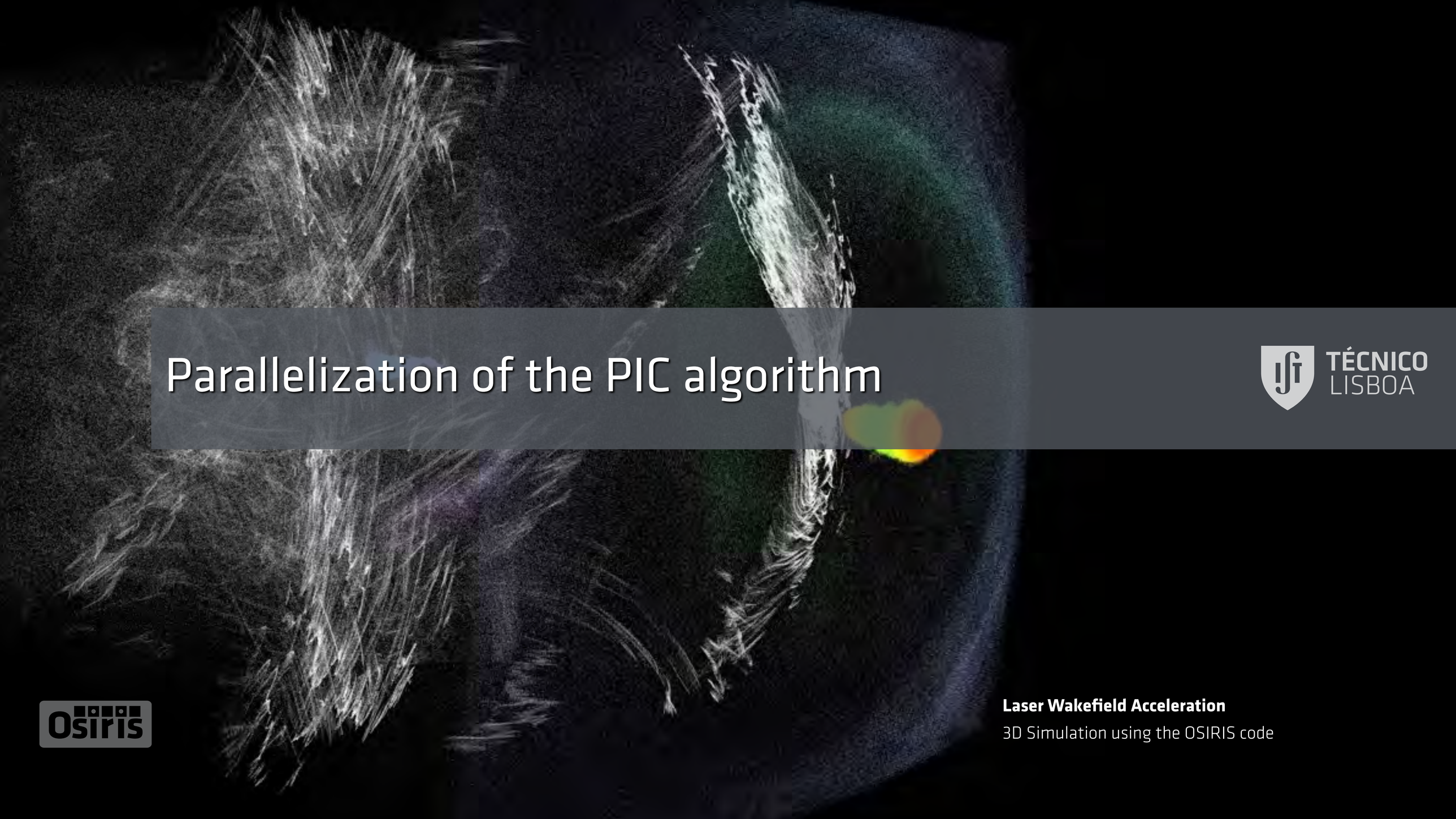
Multiscale Parallelism

- Modern HPC systems present a hierarchy of parallelism
  - At the highest level they are a network of computing nodes
  - Each node is a set of CPUs / cores (+ GPUs/ MICs) sharing memory inside the node
  - Most processing cores have a vector SIMD unit (Intel, PowerPC, Fujitsu)

- **Efficient HPC system use requires taking advantage of all these levels of parallelism**



Parallelization of the PIC algorithm



TÉCNICO LISBOA

OSIRIS

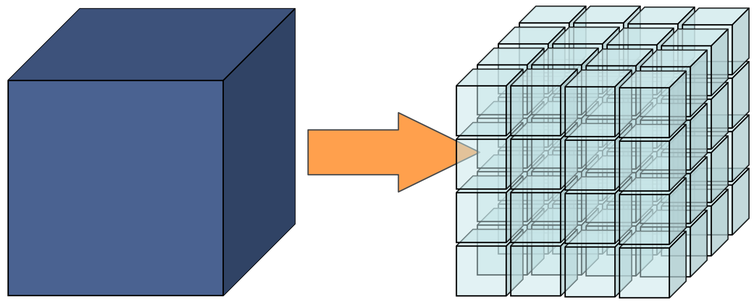
Laser Wakefield Acceleration

3D Simulation using the OSIRIS code

Node level parallelism

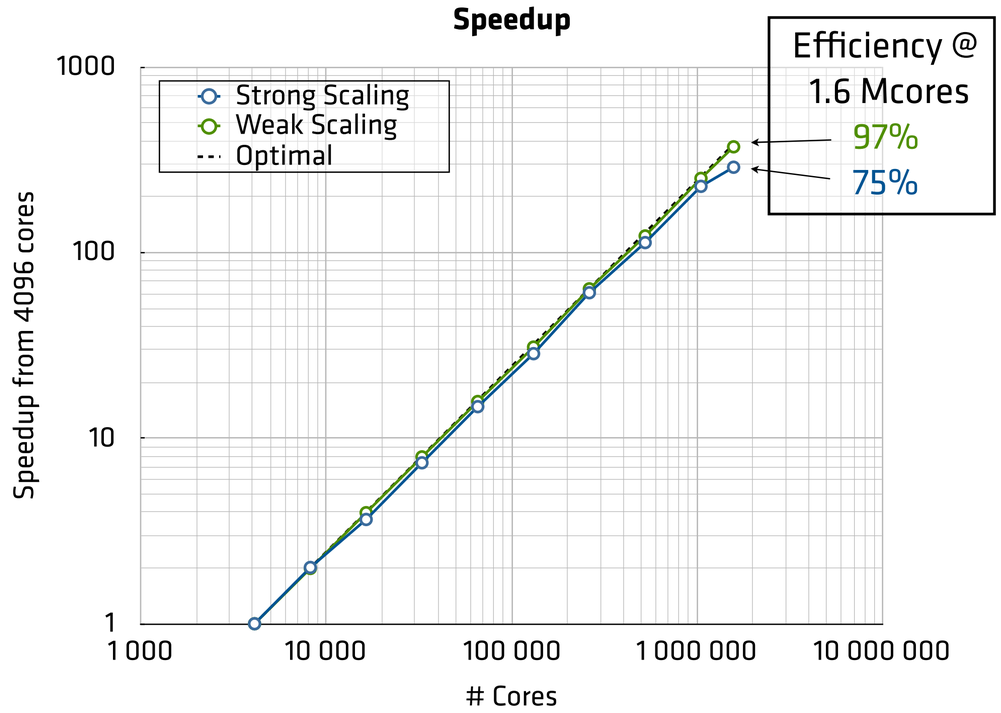
Spatial domain decomposition

- Each process cannot directly access memory on another node:
  - Information is exchanged between nodes using network messages (MPI)
- Standard parallelization uses a spatial decomposition:
  - Each node handles a specific region of simulation space
- Works very well also on multi-core nodes
  - Benefits from shared memory
  - Message passing inside a node is very efficient
- Very efficient for uniform plasmas



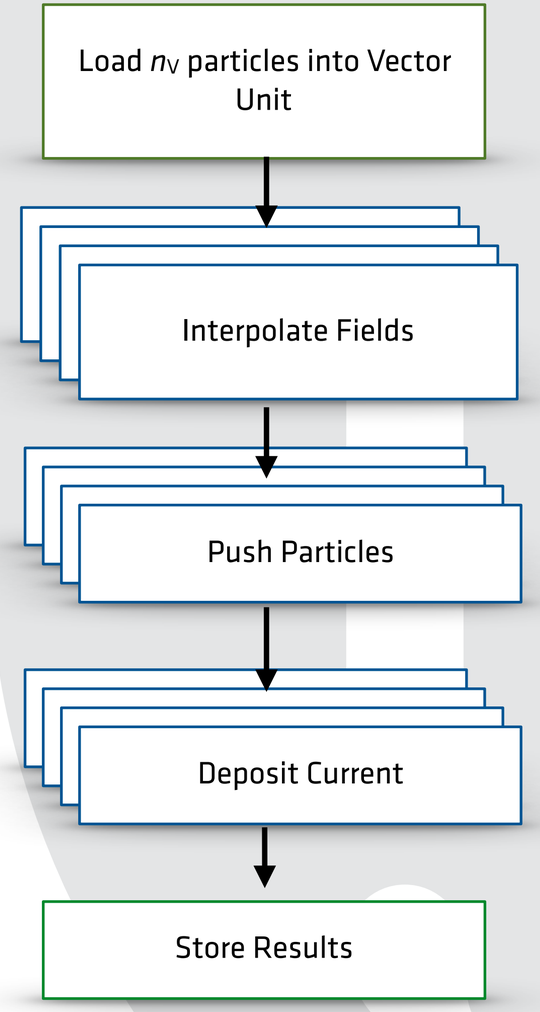
Sim. Volume *Parallel Domain*

Liewer and Decyk, JCP **85** 302 (1989);Fonseca et al., PPCF **55** 124011 (2013)



Vectorization of the PIC algorithm

- **PIC codes are good candidates for optimization**
  - Operations on each particle independent from each other…
  - except for current deposition
  - For most cases work well in single precision
- **Process n_V (vector width) particles at a time**
- **Field interpolation requires a gather operation**
  - Field grid may be altered to avoid this
- **Current deposition may cause memory collisions**
  - Serialize memory accumulation
  - Change grid structure
  - Transpose vectors (vectorize by current line)



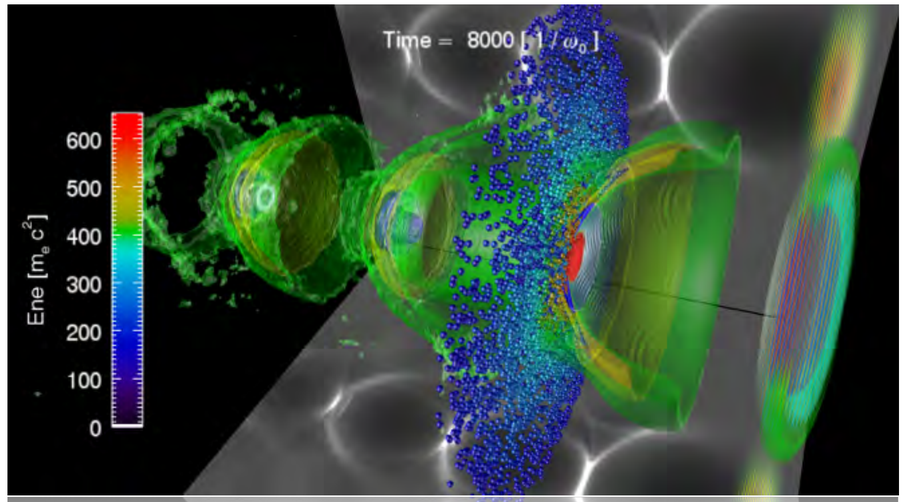
BlueWaters CPU tests

- XE Partition
  - 772 480 AMD6276 cores
- Warm plasma tests
  - Quadratic interpolation
  - u_{th} = 0.1 c
- 3D Problem size
  - cells = 38624 × 1024 × 640 (~2.5×10^{10})
  - 400 particles/cell (~ 10^{13})
- Computations
  - **2.2 PFlop/s performance**
  - **31% of R_{peak}**

BlueWaters
NSCA, U.S.A.

Bowers et al., PoP **15** 055703 (2008); Fonseca et al., PPCF **55** 124011 (2013); Vincenti et al., arXiv:1601.02056 [physics.comp-ph] (2016)

Maintaining parallel load balance is crucial



LWFA Simulation

Parallel Partition

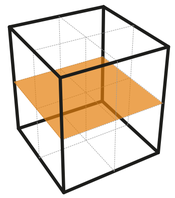
- 94×24×24 = 55k cores

Load Imbalance (max/avg load)

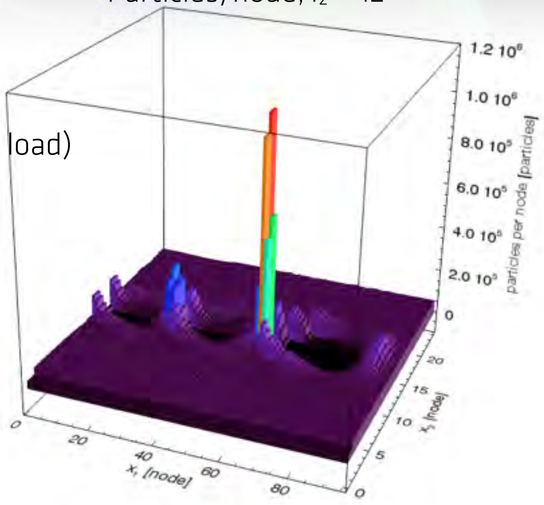
- 9.04×

Average Performance

- ~12% peak

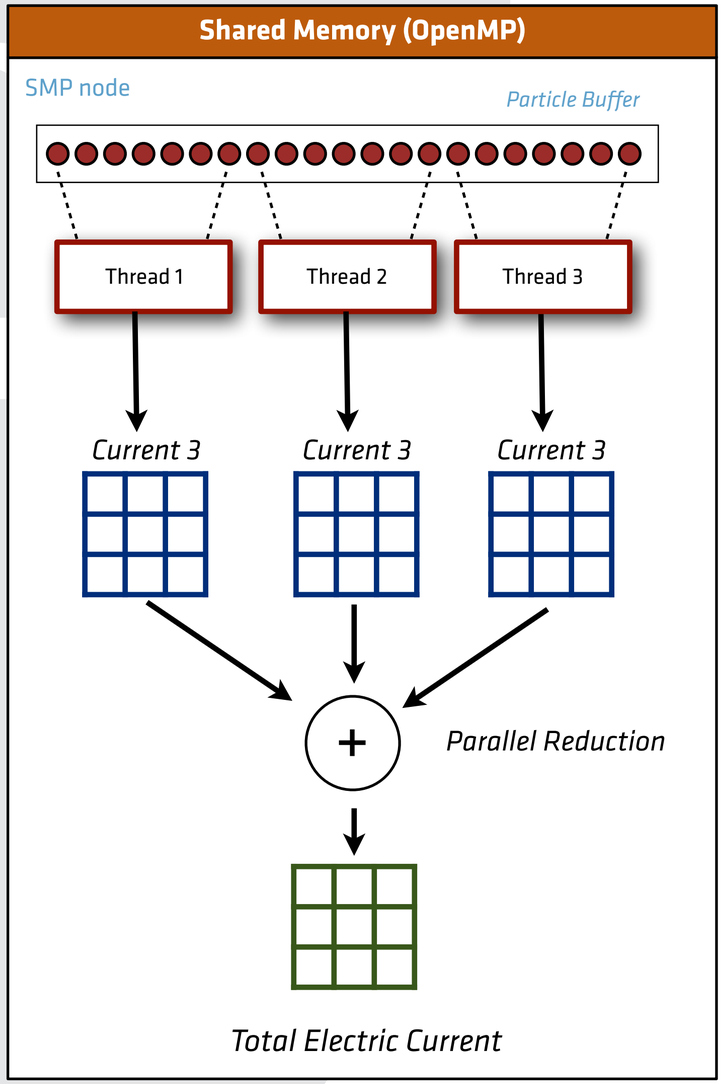


Particles/node, i_z = 12



R A Fonseca et al., PPCF **55** 124011 (2013)

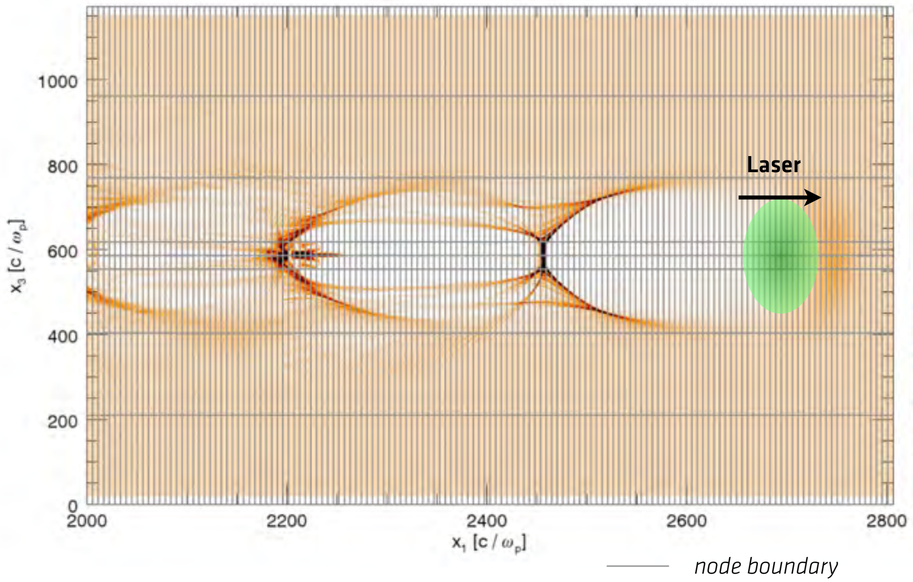
- For large core counts the simulation volume inside each node is very small
  - Fluctuations on the plasma density lead to load inbalance
- Shared memory parallelism can help
  - Use a "particle domain" decomposition inside shared memory region
  - Smear out localized computational load peaks
- Spawns n_T threads to process the particles:
  - Use n_T copies of the current grid
  - Divide particles evenly across threads
  - Each thread deposits current in only 1 of the grid copies
- Accumulate all current grids in a single current grid
  - Divide this work also over n_T threads

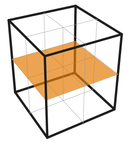


Adjust processor load dynamically

Redistribute computational load between nodes

- The code can change node boundaries dynamically to attempt to maintain a even load across nodes:
  - Determine best possible partition from current particle distribution
  - Rearrange parallel partition



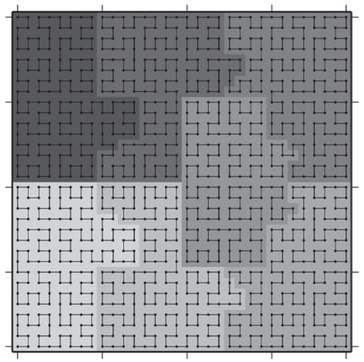


x_1-x_2 slice at box center
similar partition along x_3
> 30% improvement in inbalance

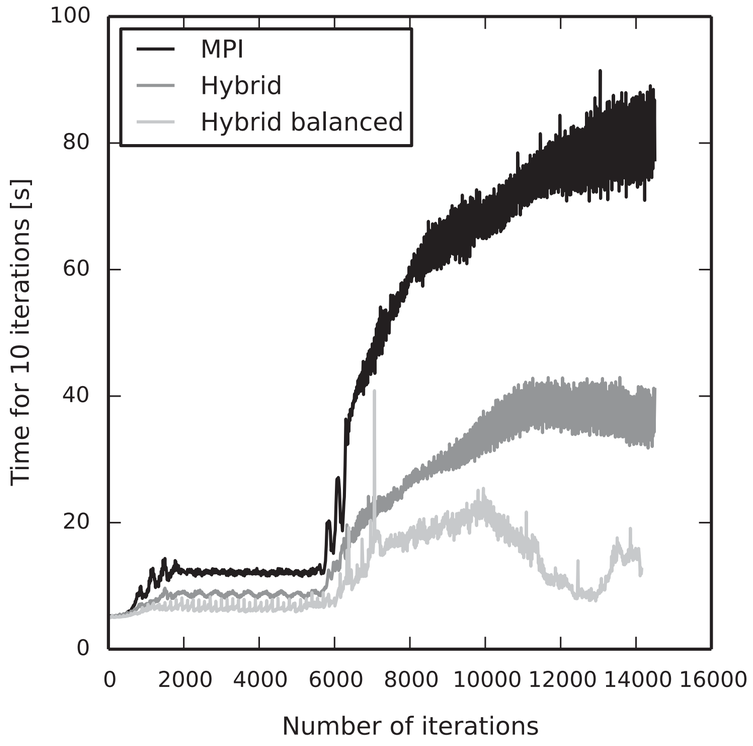
Patch based load balance

- Partition the space into (10-100x) more domains (patches) than processing elements (PE)
- Dynamically assign patches to PE
  - Assign similar load to PEs
  - Attempt to maintain neighboring patches in the same PE

Patch assignment

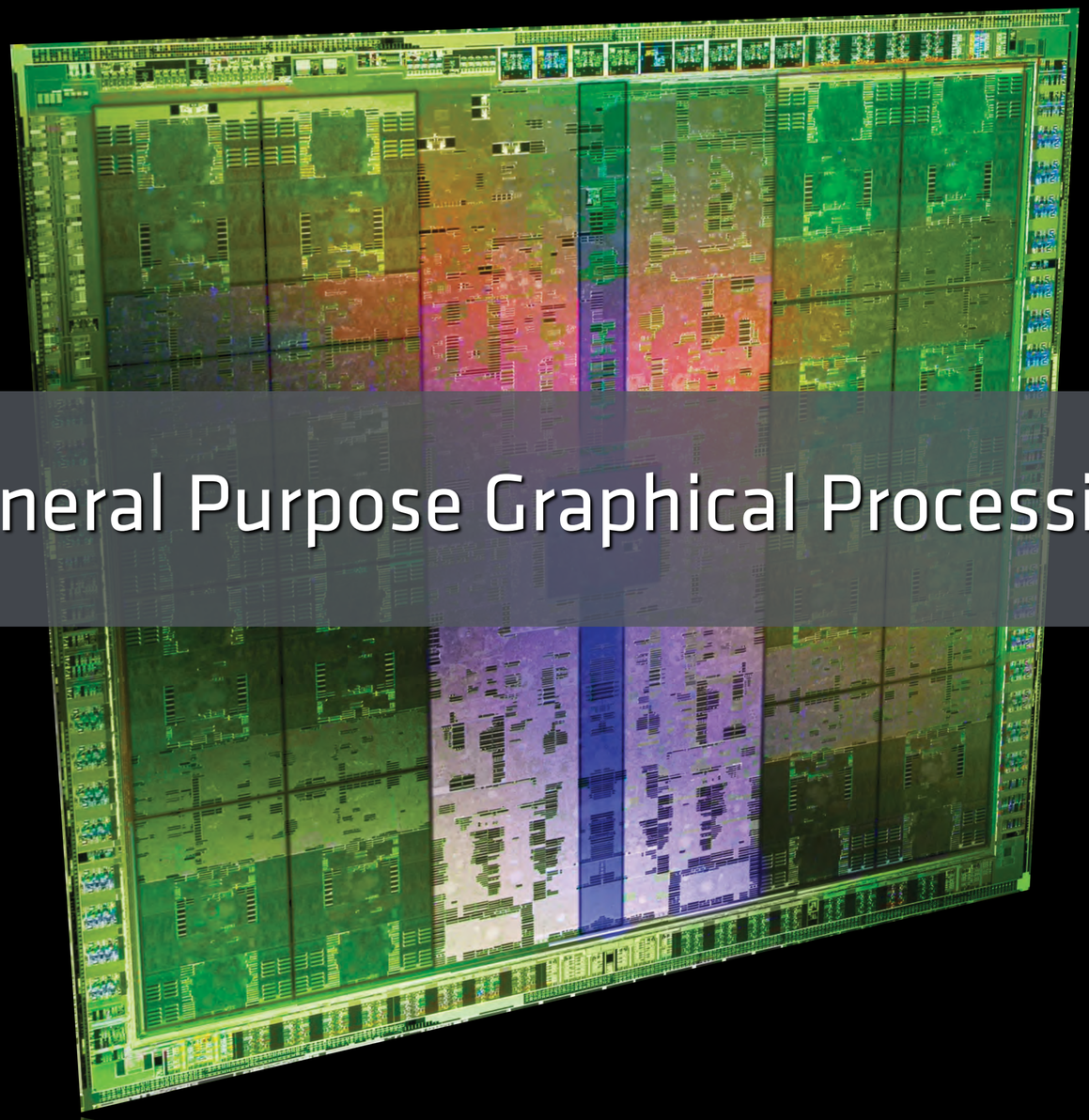


7 PE example



Fonseca et al., PPCF **55** 124011 (2013); Germaschewski et al., JCP **318** 305 (2016); Beck et al., NIMPR-A **829** 418 (2016)

General Purpose Graphical Processing Units



NIVIDIA Fermi K20x die

TÉCNICO LISBOA

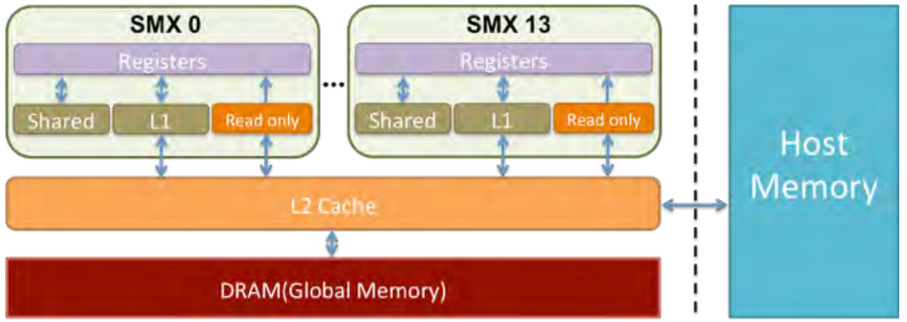
General Purpose Graphical Processing Unit Accelerators

ORNL Titan

- Cray XK7
  - 18 688 Compute Nodes
  - 8.2 MW
- Interconnect
  - Cray Gemini interconnect
- Node configuration
  - 1× AMD Opteron 6274 @ 2.2 GHz (16 cores)
  - 1× NVIDIA Tesla K20x
  - memory 32 GB (host) + 6 GB (GPU)
- Total system
  - 299 008 host cores + 18 688 GPUs
  - 0.6 PB host RAM + 0.1 PB GPU RAM
- Performance
  - R_{MAX} = 17.2 PFlop/s
  - R_{PEAK} = 27.1 PFlop/s

NVIDIA Tesla K20X (Kepler) accelerator

- K20X Accelerator
  - 14 SMX streaming multiprocessors @ 732 MHz
  - 6 GB GDDR5
  - 1.5 MB L2 Cache
- SMX streaming multiprocessors
  - 192 CUDA cores for float/int
  - 64 double precision cores
  - 64 KB shared memory / L1 cache
  - 64 K registers
  - up to 2048 threads
    - Fast switching between threads
  - executes 32 threads at a time (warp)
    - SIMD like operation
- Peak performance
  - 3.95 TFlops / 1.31 Tflops peak (single/double precision)
  - DRAM Bandwidth 250 GB/sec

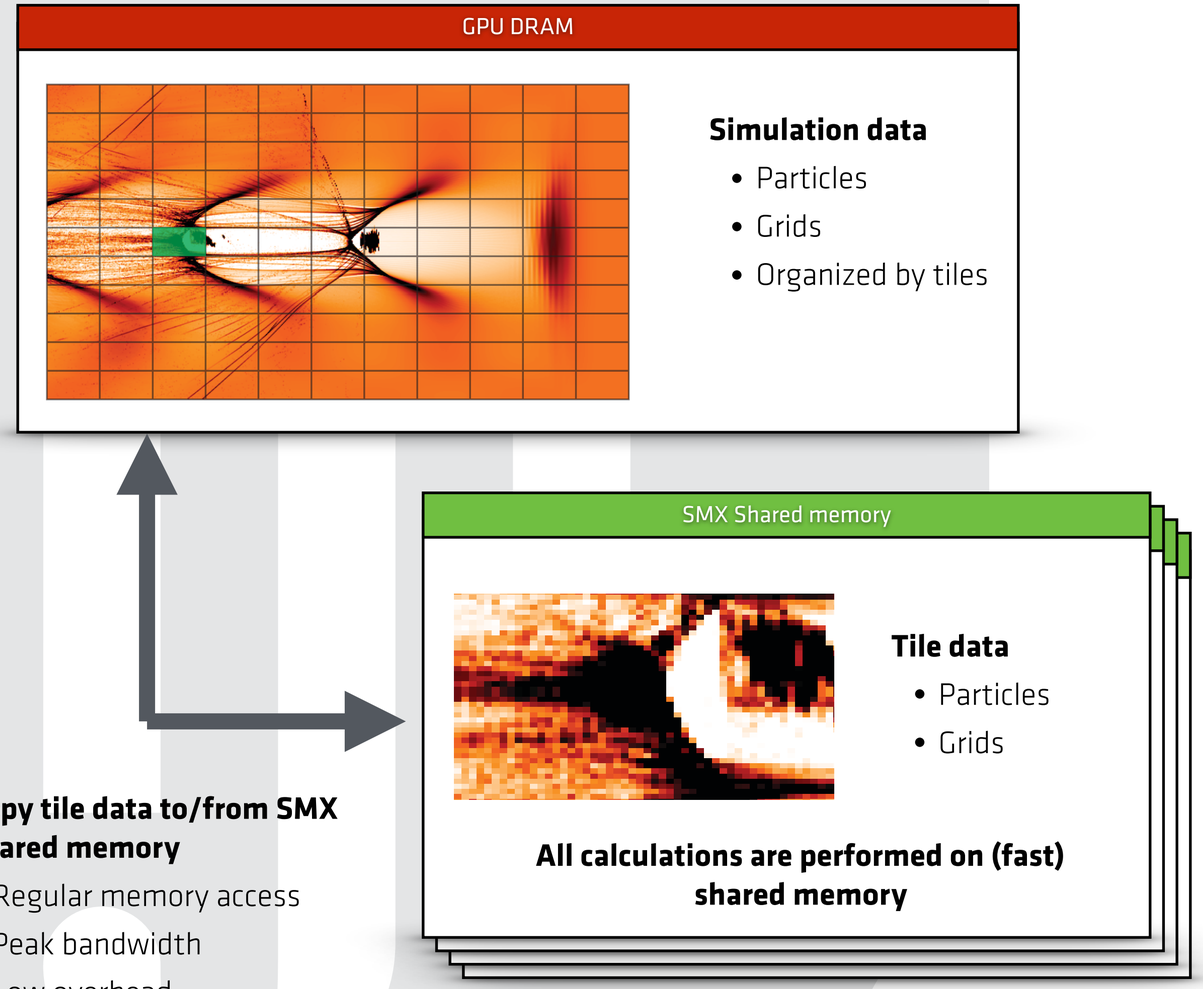


PIC algorithms on GPU architectures

- Most important bottleneck is memory access
  - PIC codes have low computational intensity (few flops/memory access)
  - Memory access is irregular (gather/scatter)
- Memory access can be optimized with a streaming algorithm
  - Global data read/write only once
  - Regular (coalesced) memory access

- PIC codes can implement a streaming algorithm by keeping particles ordered by tiles
  - Minimizes global memory access since field elements need to be read only once
  - Global gather/scatter is avoided.
  - Deposit and particles update have optimal memory access.

- **Challenge:** optimizing particle reordering



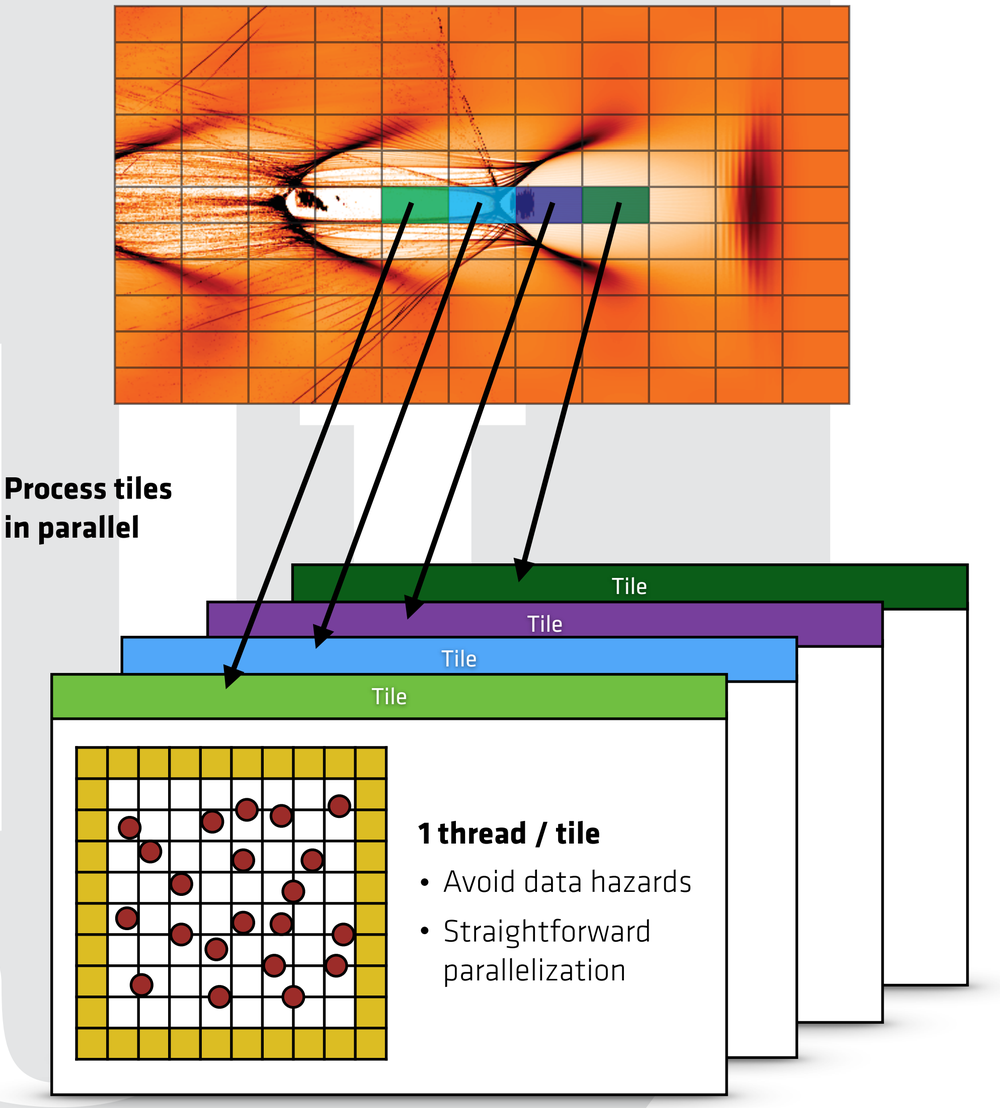
Decyk and Singh, CPC **185** 708 (2014); Zenker et al., arXiv:1606.02862 [cs.DC] (2016)

Advancing particles / deposit current

- Within a tile, all particles read or write the same block of fields.
  - Before pushing particles, copy fields to fast memory
  - After depositing current to fast memory, write to global memory
  - Different tiles can be done in parallel

- Each tile contains data for the grids in the tile, plus guard cells
  - Similar to MPI code, but with tiny partitions

- Parallelization of particle advance trivial
  - Each particle is independent of others, no data hazards
- Current deposit is also easy if each tile is controlled by one thread
  - This avoids data collisions where two threads try to update the same memory
- However, if each tile is controlled by a vector of threads, data collisions are possible.
  - Atomic updates
  - "Checkerboard" tile access
  - Line current deposition



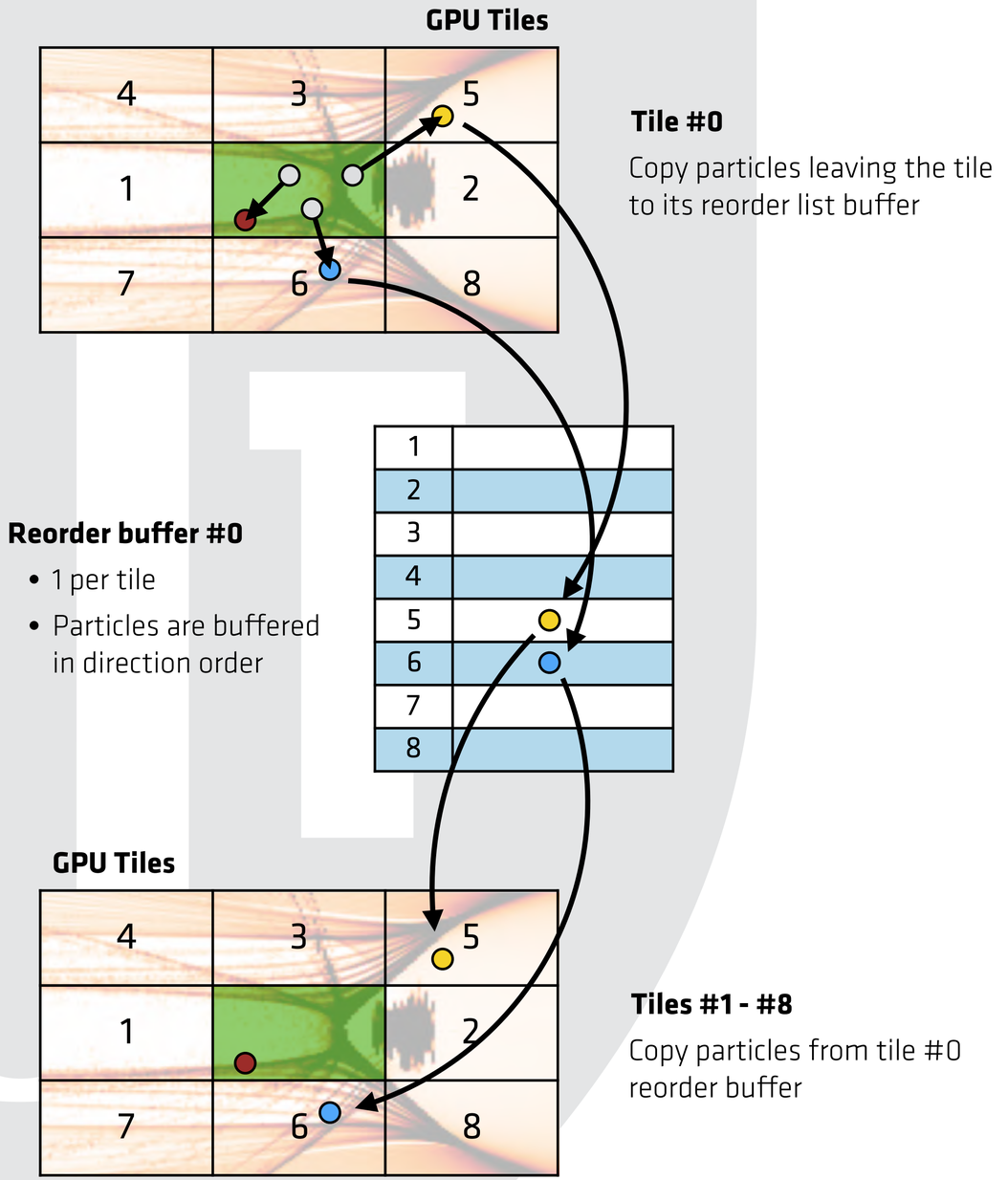
Decyk and Singh, CPC **185** 708 (2014); Zenker et al., arXiv:1606.02862 [cs.DC] (2016)

Efficient particle reordering between tiles

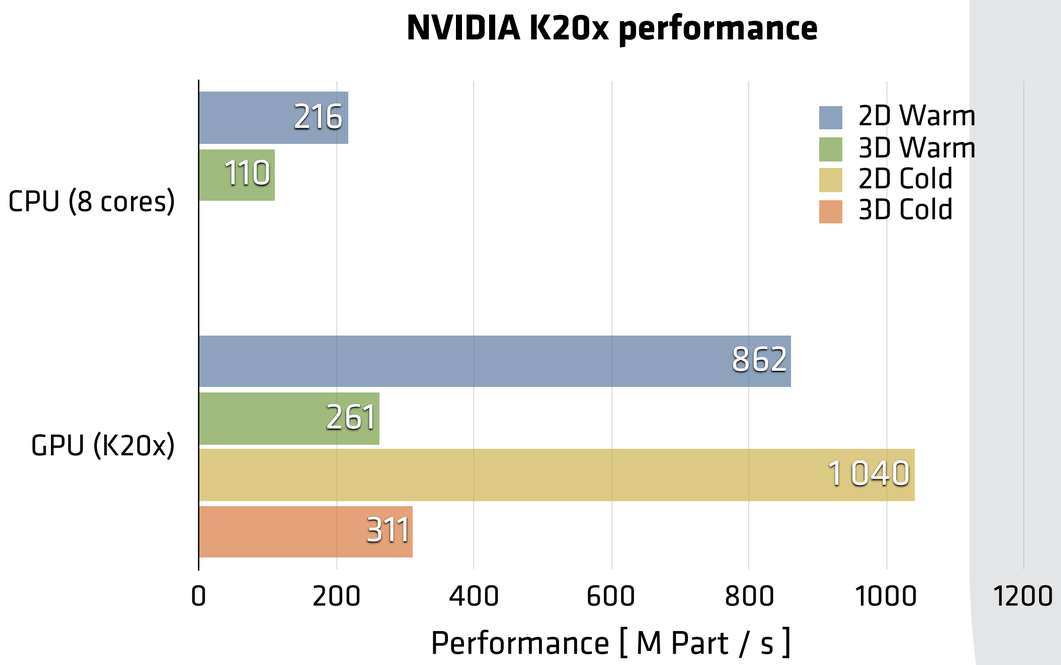
- Three steps:
  1. Create a list of particles which are leaving a tile, and where they are going
  2. Using list, each thread places outgoing particles into an ordered buffer it controls
  3. Using lists, each tile copies incoming particles from buffers into particle array
- Less than a full sort, low overhead if particles already in correct tile
  - Can be done in parallel
  - Essentially message-passing, except buffer contains multiple destinations
- Same algorithm works well for shared memory CPU/ OpenMP

- Extend to multiple boards using MPI
  - Copy from GPU buffer to/from MPI buffers
  - Pack multiple tiles into single message

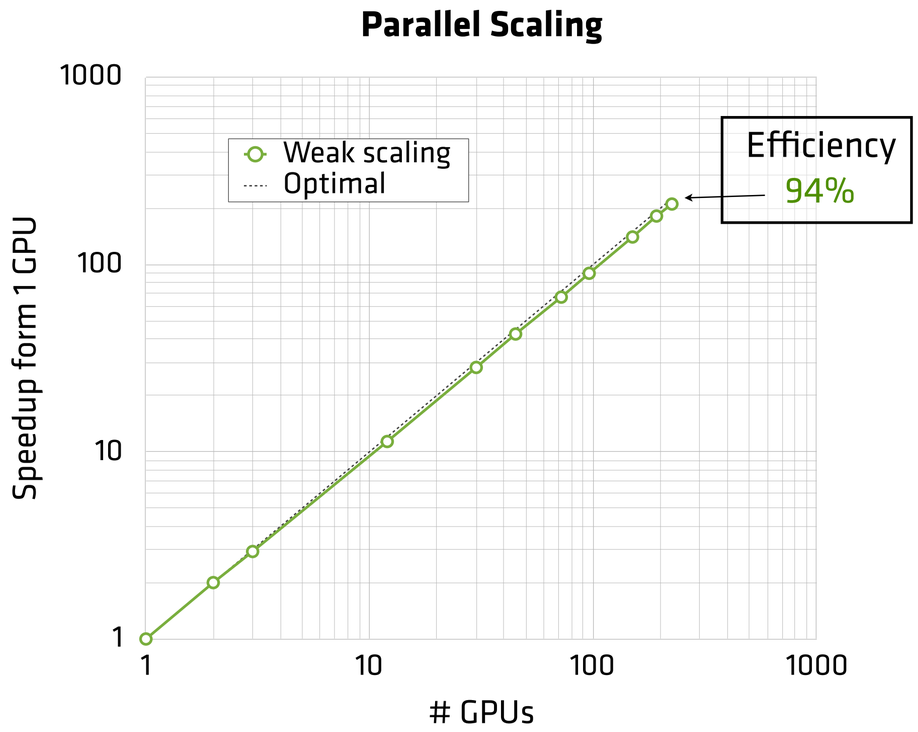
Decyk and Singh, CPC **185** 708 (2014)



OSIRIS CUDA performance



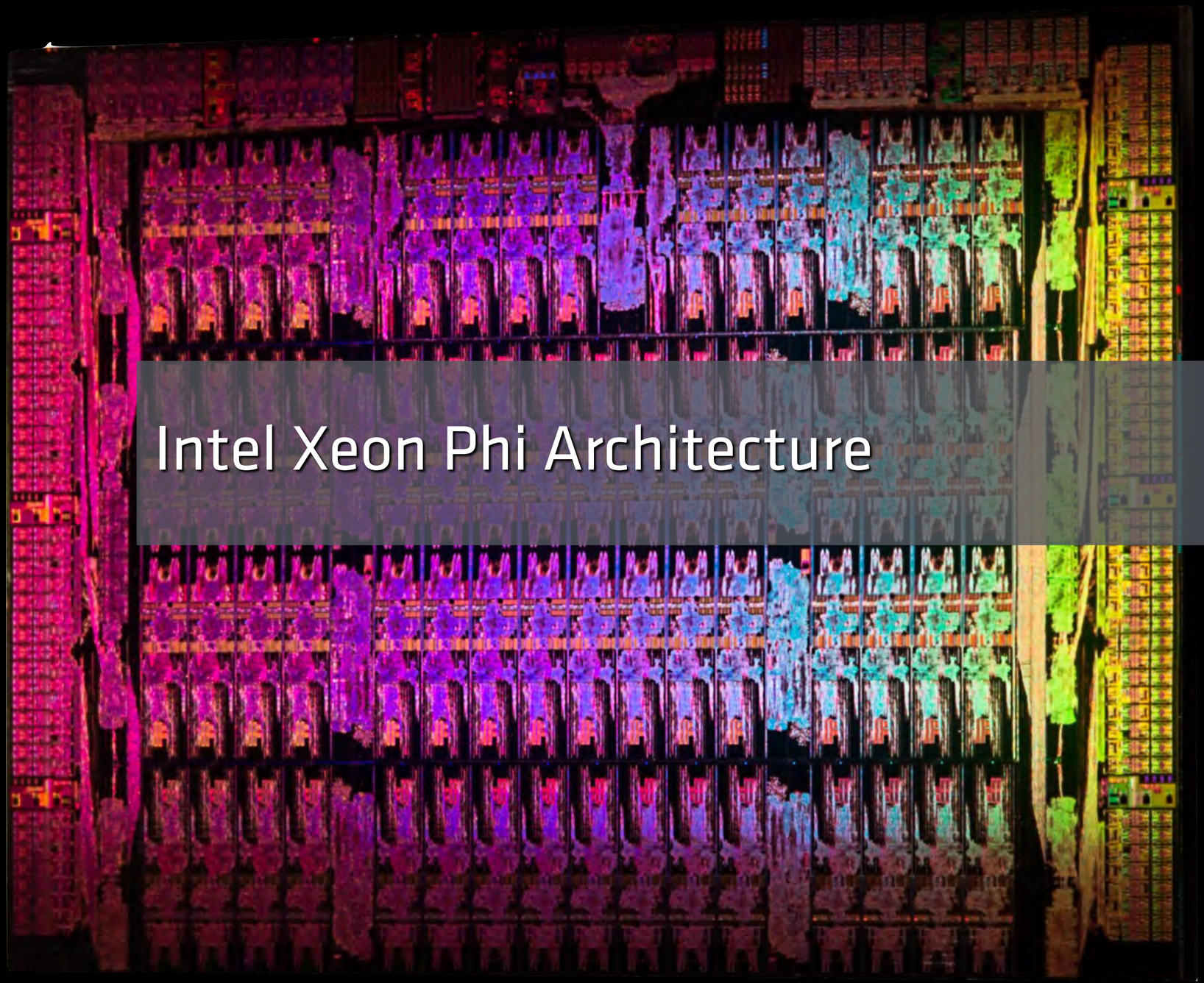
- Single board tests
- Up to 4× speedup from 1 cpu (8 cores)
- Cold plasma tests shows performance > 1 G part/s
  - Impact of tile reordering ~ 17%



- **Weak scaling tests**
  - Start from 1 GPU board
  - Scale problem size linearly with number of boards
  - Increase number of boards

- **Near perfect scaling up to 225 boards**
  - Parallel efficiency 94%

Intel Xeon Phi Architecture



Intel Xeon Phi 5110p die

TÉCNICO LISBOA

Intel Xeon Phi architectures

NUDT Tianhe-2a

- TH-IVB-FEP Cluster
  - 16 000 Compute Nodes
  - 17.8 MW
- Interconnect
  - TH Express 2
- Node configuration
  - 2× Intel® Xeon® CPU E5-2692 @ 2.20GHz (12 cores)
  - 3× Intel® Xeon® Phi 31S1P
  - 64 GB (host) + 2x8 GB (MIC)
- Total system
  - 512 000 host cores + 48 000 MICs
  - 1 PB host RAM + 0.26 PB MIC RAM
- Performance
  - R_{MAX} = 33.9 PFlop/s
  - R_{PEAK} = 54.9 PFlop/s

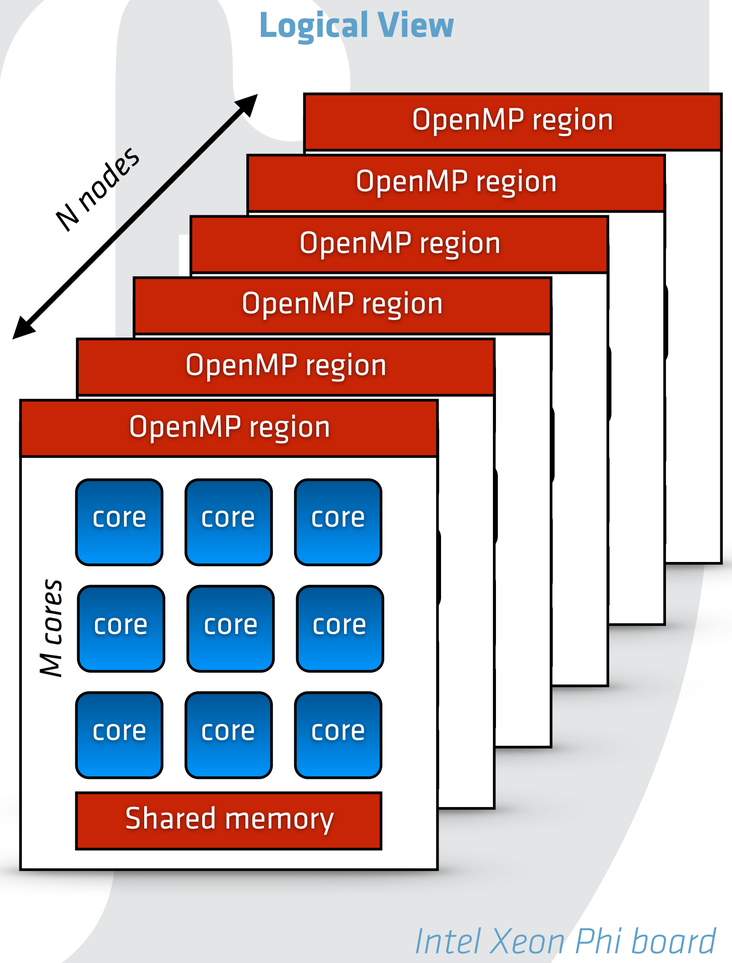
Intel Xeon Phi (MIC)

- **Knights Corner Architecture**
  - Also known as “Many Integrated Core” or MIC
- **Processing**
  - 60 x86_64 cores @ 1.053 GHz
    - 4 threads / core
    - 512 kB L2 + 32 kB L1I + 32 kB L1D cache
  - Scalar unit + 512 bit vector unit
    - Up to 32 flops / cycle /core
  - 2.02 TFlops single precision
  - 1.01 TFlops double precision
- **Memory**
  - Shared 8 GB GDDR5 RAM
  - Up to 320 GB/s
- **System Interface**
  - PCIe x16 connection
- **Offload execution**
  - CPU code offloads heavy sections to the MIC
- **Native execution**
  - Board runs 64bit Linux
  - Network connection to other boards/nodes (MPI)
  - **Run all code inside the board(s)**

Logical view of a Xeon Phi board

Intel Xeon Phi Boards

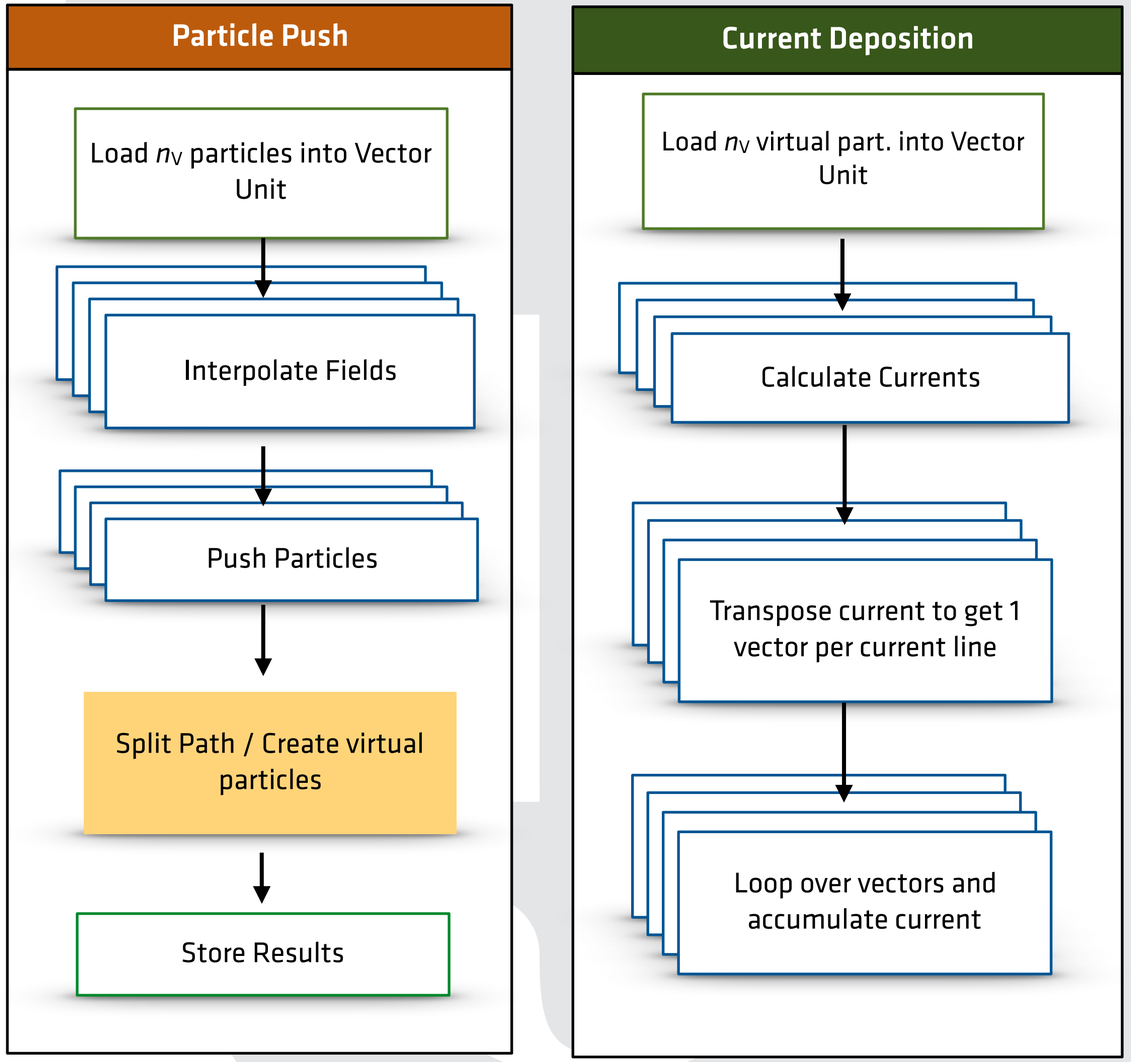
- Use the board in native execution mode
  - All code runs inside the board
- Each MIC board can be viewed as small computer cluster with ***N* SMP nodes** with ***M* cores** per node
  - Use a distributed memory algorithm (**MPI**) for parallelizing across nodes
  - Use a shared memory algorithm (**OpenMP**) for parallelizing inside each node
- The exact same code used in "standard" HPC systems can run on the MIC board
- Extending to multiple boards/nodes straightforward using MPI



Intel Xeon Phi board

EM-PIC codes on Xeon Phi clusters

- **Same strategy for standard HPC clusters works well**
  - Exploit all levels of parallelism available: distributed memory, shared memory and vector units
- **Paralellization of PIC algorithm**
  - Spatial decomposition across nodes: each node handles a specific region of simulation space
    - Communicate between nodes/boards using MPI
  - Split particles over cores inside node
    - Use OpenMP for parallelism
  - Multiple nodes can fit inside a single board
- **Vectorization of PIC algorithm**
  - The Xeon Phi has a wide vector unit
    - Process 16 particles at a time
    - Explicit vectorization yields the best result



Vectorized by particle

- Vectors contain same quantity for all particles

Transpose vectors

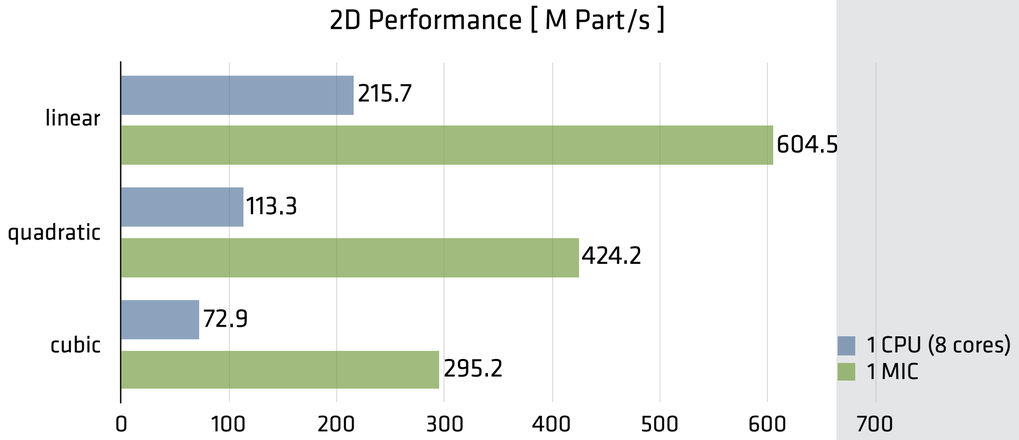
- Done through vector operations

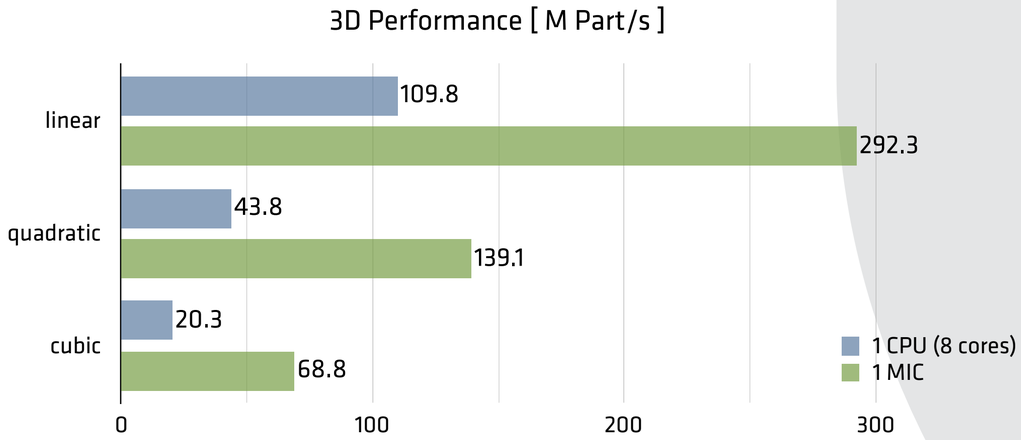
Vectorized by current line

- Vectors contain 1 line of current for 1 particle

Fonseca et al., PPCF **55** 124011 (2013); Surmin et al., CPC **202** 204 (2016); Vincenti et al., arXiv:1601.02056 [physics.comp-ph] (2016)

Single Board performance (warm plasma)





CPU Intel Xeon E5-2660 @ 2.20GHz (8 cores)

Intel icc 13.0.1:

[cpu] icc -O3 -xHost -no-prec-div -ipo -align array32byte

[mic] icc -mmic -O3 -no-prec-div -align array64byte

Speedup [MIC / 1 CPU]

	2D	3D
linear	2.80	2.66
quadratic	3.74	3.18
cubic	4.05	3.38

manual vectorization on both MIC and CPU

Speedup MIC manual / auto vectorization

	2D	3D
linear	7.12	4.12
quadratic	6.18	3.52
cubic	5.86	3.29

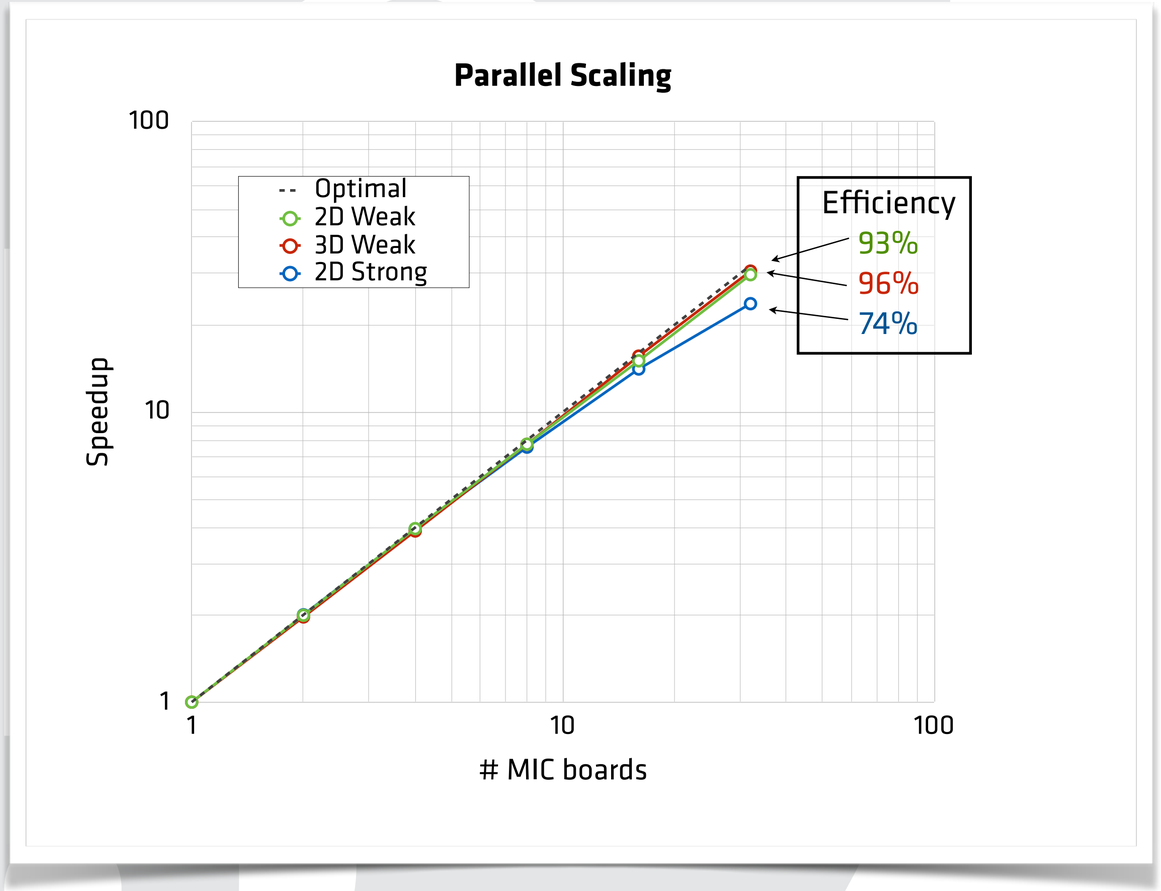
- **Only particle advance/deposit was vectorized**
  - Standard Fortran 03 code used for the remainder of the code
- **Manual vectorization also plays a key role in CPU code**
  - Explicit AVX vectorization used
- **Up to 4× speedup from 1 cpu (8 cores)**
- **Manual vectorization at least 3.2× faster than auto vectorization**

Parallel Scalability

- **Use existing MPI parallelization for distributed memory systems**
  - MPI universe spawns multiple boards / nodes
  - No changes required to the single MIC code

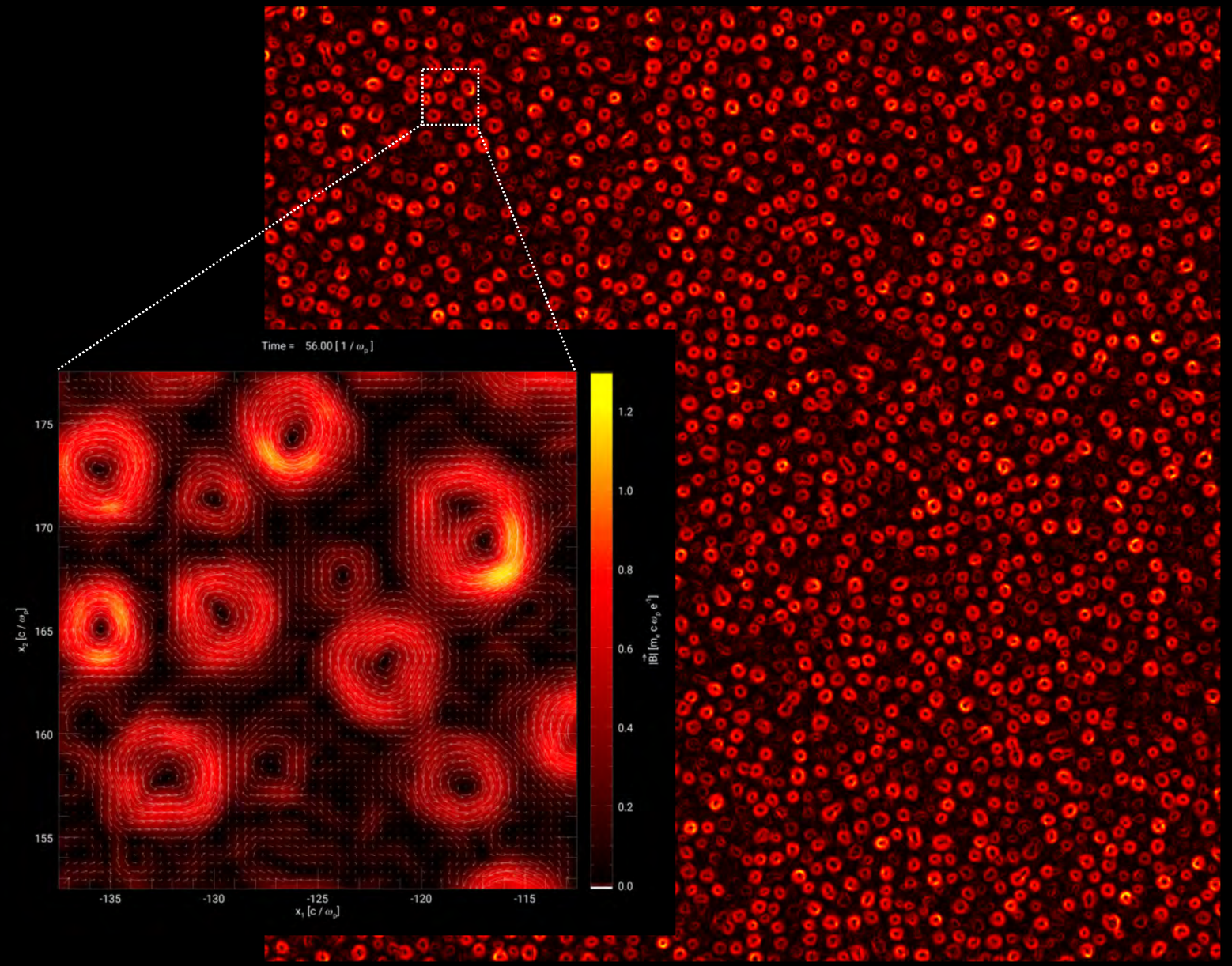
- **Weak scaling tests**
  - Start from 1 MIC board
  - Scale problem size linearly with number of boards
  - Increase number of boards
- **Near perfect scaling up to 32 boards**
  - Parallel efficiency 93% (2D) and 96% (3D)
  - Final problem size
    - ~ 10^9 particles

- **Strong scaling tests**
  - Start from 2 MIC boards
  - Keep problem size constant
  - Increase number of boards
- **Good scaling up to 32 boards**
  - Parallel efficiency 74%

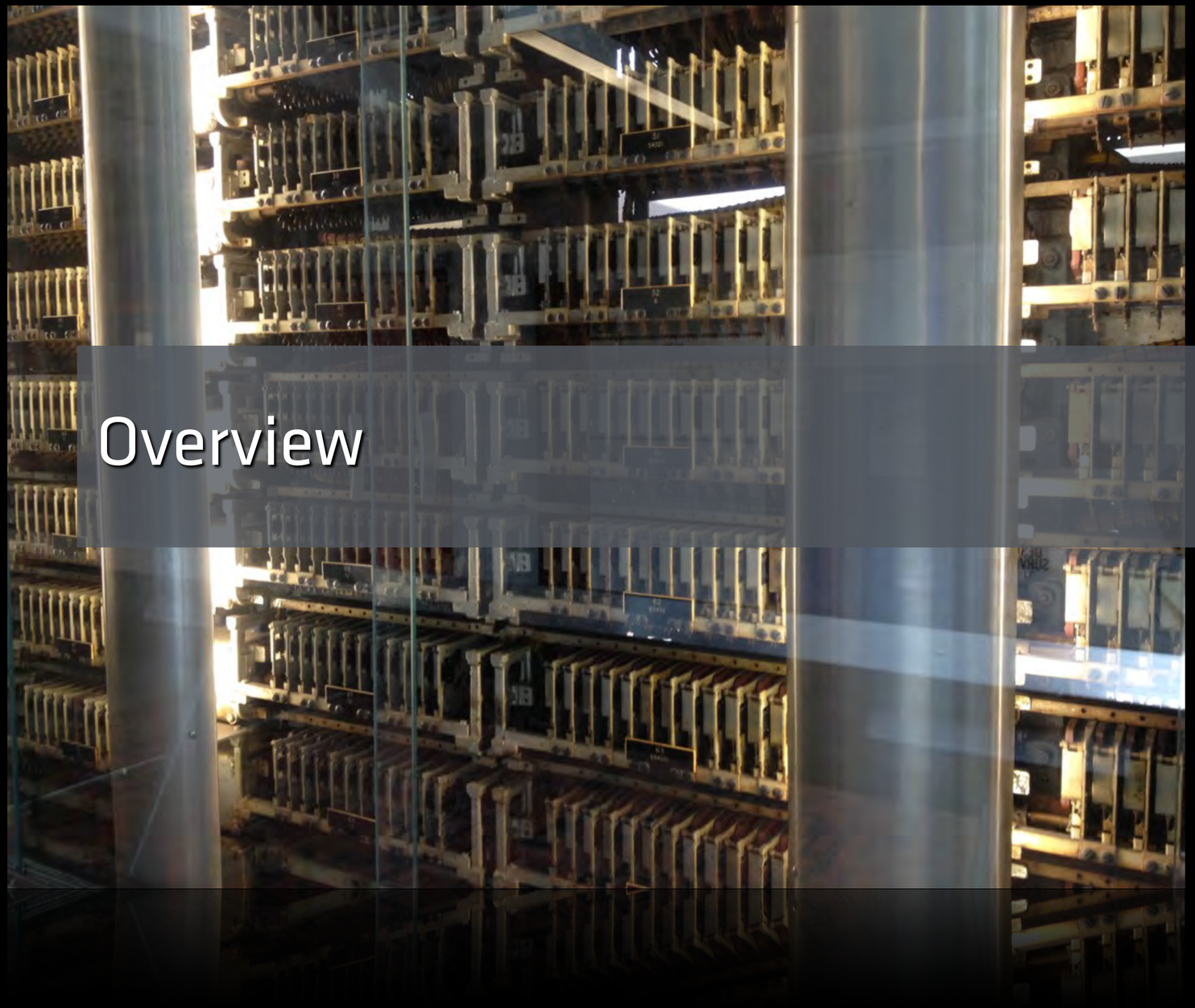


Production runs

- Simulation setup
  - Collision of an electron and a positron plasma cloud
  - Physics dominated by the Weibel instability
  - 2D simulation in the perpendicular plane
- Parameters
  - 4096 × 4096 cells
  - 2^2 particles / cell / species
  - $\gamma\ v_{fl} = \pm\ 0.6\ c$
  - $\gamma\ v_{th} = 0.1\ c$
  - Run on 16 MIC boards
- Very good performance for calculations



Overview



TÉCNICO LISBOA

Harvard Mark I - 1944

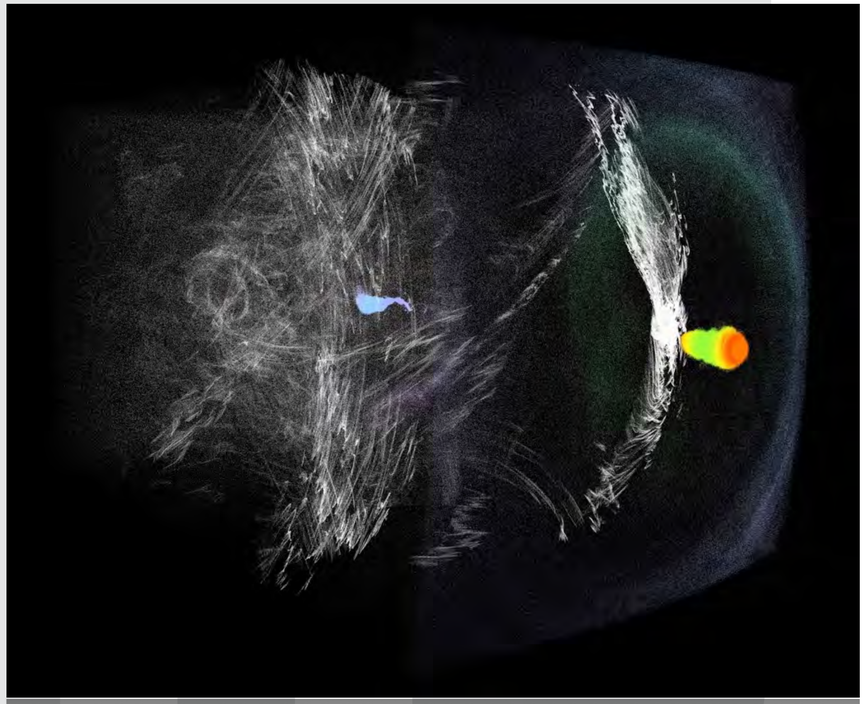
Rear view of Computing Section

Overview

- **Outstanding progress in computational power since 1950s**
  - Present systems can reach performances of 0.1 EFlop/s
  - Energy cost for calculations has gone down by 14 orders of magnitude
  - Continuous evolution of architectures and computing paradigms

- **Exascale simulations are within reach**
  - Present simulations can already track > 10^{13} particles for millions of time steps
  - Increasing quality and quantitative fidelity of simulations
  - Continuously evolve algorithms and codes to efficiently use new generations of computing hardware

- **This is a community effort among experts in large scale plasma simulation**
  - This evolution presents a formidable challenge for computational physicists
  - Useful to have an ecosystem of codes where ideas are shared.
  - The community needs sustainable support for exascale software development



PICKSC

http://picksc.idre.ucla.edu/