Engineering and Physical Sciences Research Council

ExCALIBUR 10

PAX-HPC - Modelling particles at Exascale: from atoms to galaxies

Benedict D. Rogers
University of Manchester

Phil Hasnip
University of York

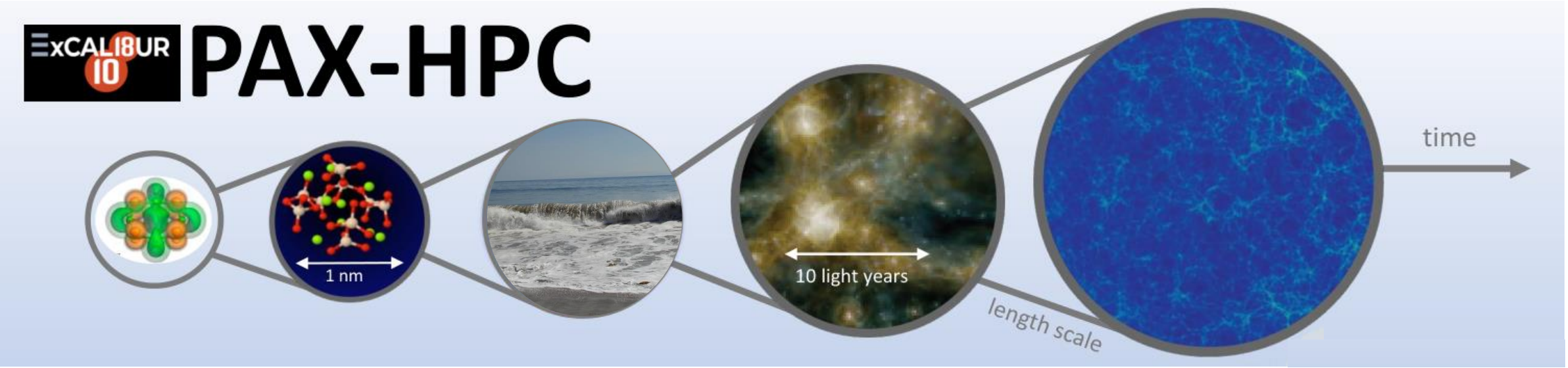
University of York · Durham University · University of Lincoln · Lancaster University · Universiteit Leiden, The Netherlands · UKRI Science and Technology Facilities Council

UCL · University of Hertfordshire UH · King's College London · Manchester 1824 · University of Southampton · epcc

Computing Insight UK 2022, 1-2nd December 2022

Contents

Particles make up the universe



Particle-based methods in Science & Engineering

- Conventional modelling has often used mesh-based methods
- Particles make up everything we see and experience
- Particle-based methods are an intuitive to simulate scientific and engineering phenomena
- Particle-based methods are **ideally suited** to massive parallelisation

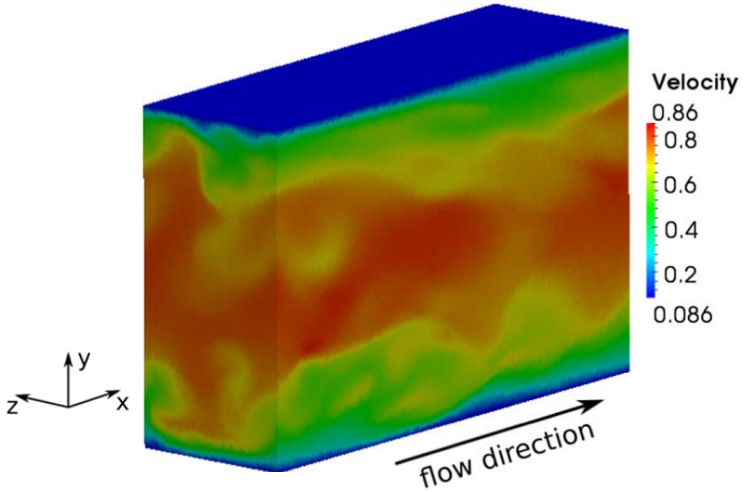
Why is Exascale Computing needed for Particles? #1

- There are three broad categories of calculations at atomic scales
- High-throughput calculations
  - Use a loosely-coupled ensemble of many simulations
  - Examples include materials discovery, using global optimisation methods (e.g. Genetic Algorithms) to find new materials with improved properties
- "Hero" calculations
  - A small number of simulations on large, complex systems
  - Examples include large-scale dynamics, e.g. liquid lithium for fusion reactor cooling, and sophisticated quantum mechanical simulations
- Complex workflows, e.g. multiscale, multiphysics
  - Coupling different methods and software together
  - Examples include embedding quantum mechanical simulations within otherwise classical simulations

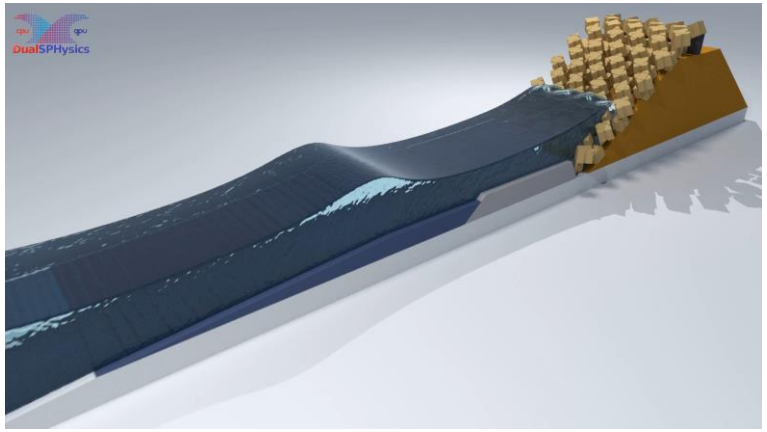
Why is Exascale Computing needed for Particles? #2

- Smoothed Particle Hydrodynamics (SPH) is a very attractive method for applications with high nonlinearity: Astronomy, planet collision, Fluid-Structure Interaction

Keggeris et al., 2019



Mayrhofer et al., 2015



Altomare et al., 2014

- Often approached by companies and industrial collaborators with '**Wicked Applications**'
- However, 'Wicked applications' in industry beyond any numerical scheme still require **engineering approximations** (with or without variable resolution)

Challenges Slide

Key Challenges for Exascale particle-based methods in Science & Engineering:

1. Long-range forces can act across the physical domain
2. Vast and rapid changes in scales, e.g. 10^9 for density
3. Enormous range of resolutions required
4. The data structures are vastly different from Mesh-based techniques

TOP SUPERCOMPUTERS IN THE WORLD June 2022

http://www.top500.org

R_{max} and R_{peak} values are in TFlops. For more details about other fields, check the TOP500 description.

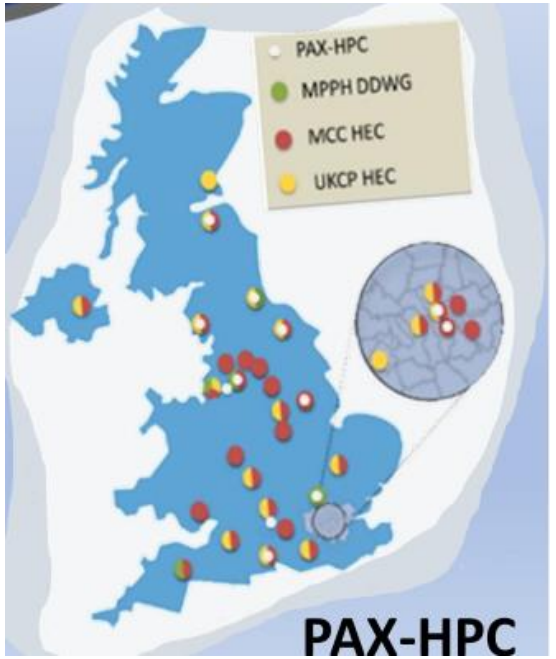
Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,730,112	1,102.00	1,685.65	21,100
2	**Supercomputer Fugaku** - Supercomputer Fugaku	7,630,848	442.01	537.21	28,899

1º Frontier (USA) 1102 petaflop/s (**consumption: 21,100 kW**)
2º Fugaku (Japan) 415 petaflop/s (**consumption: 28,335 kW**)

Frontier is **heterogeneous**: **CPUs** + **GPUs**
64-core AMD Epyc Radeon Instint MI250X

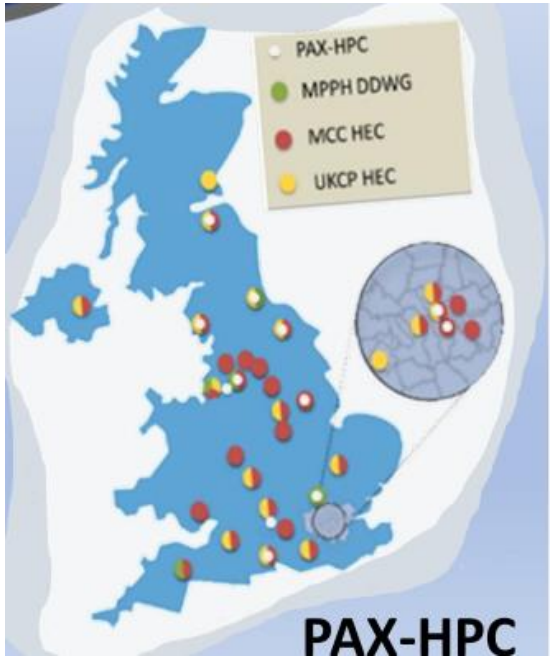
Folding@Home performed 1.22 ExaFLOPS in March 2020 using millions of Home PCs

PAX-HPC Project



- Particles-At-eXascale for High-Performance Computing (PAX-HPC), £3m ExCALIBUR project
- **Aim**: redesign high-priority particle-based codes & algorithms for exascale
- **New Methods of new hardware**
  - Performance portable programming methods (SYCL)
  - Exploit emerging network technologies (DPUs, GPU-aware communications)
- **New parallelisation strategies**
  - High-level additional parallelism (Task-farming, parallel-in-time methods)
  - Low-level task parallelism: Task graphs, task schedulers
- **Complex Workflows**
  - Coupling multiscale, multi-physics modes with FAIR data

PAX-HPC Project Team



PAX-HPC Combines Expertise:

- Massively Parallel Particle Hydrodynamics (MPPH)
- Materials and Molecular Modelling (MMM)
- Materials Chemistry Consortium (MCC)
- UK Car-Parrinello Consortium (UKCP)

- PI: Prof. Scott Woodley (UCL)

Engineering and Physical Sciences Research Council

ExCALIBUR

PAX-HPC - Modelling particles at exascale: from atoms to galaxies

Part 1: Atoms to Meso-scale

Phil Hasnip
University of York

Parallel Eigensolvers

(Marcello Puligheddu and Ian Bush)

$$H\Psi = E\Psi$$

Matrix diagonalisation to calculate eigenvalues & eigenvectors is key to many quantum mechanical simulations of molecules and materials.

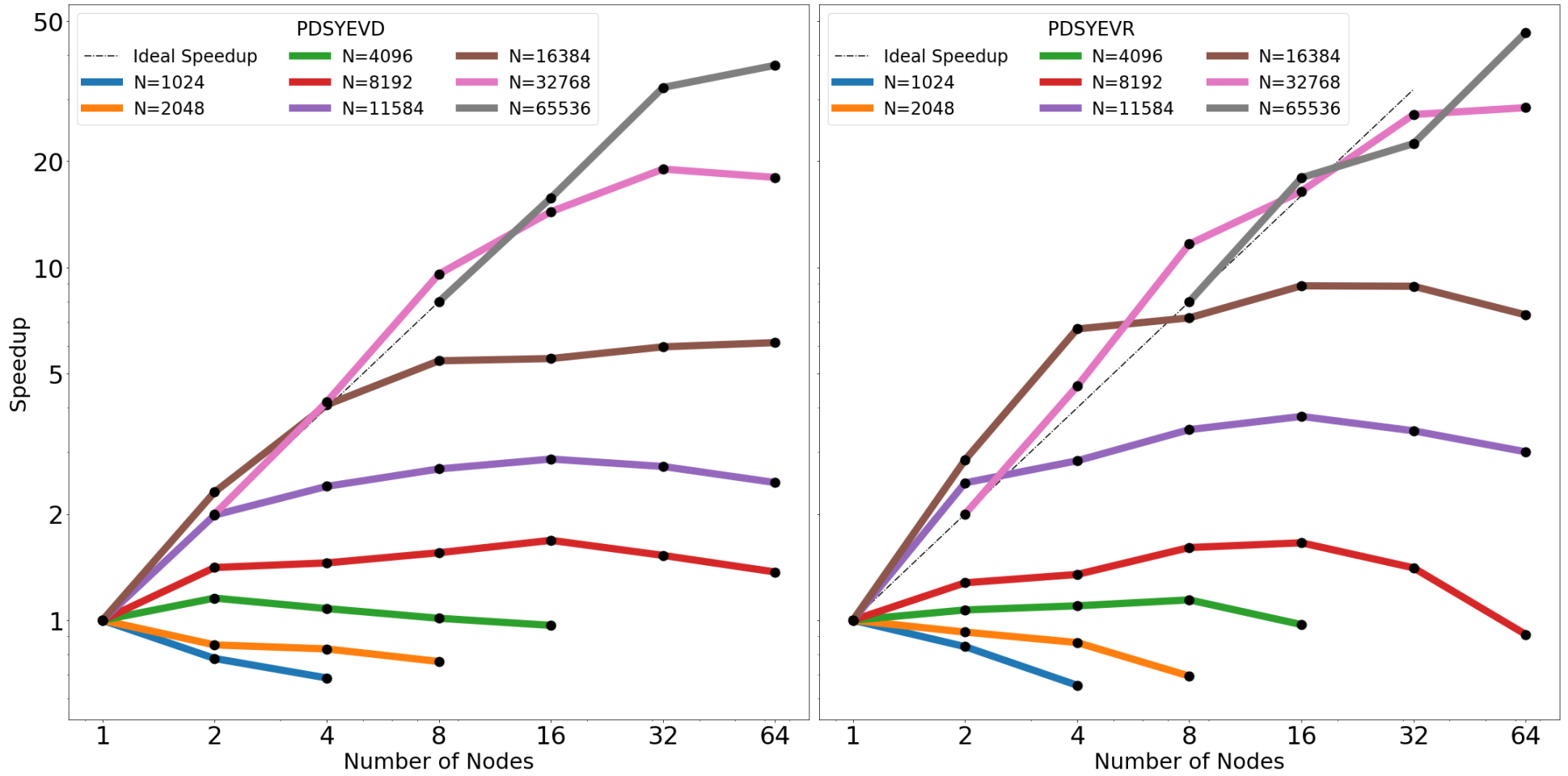
When diagonalising a N x N matrix:

- computational storage scales as N^2
- time scales as N^3.

For local basis set methods, N is the number of basis functions, but the number of electrons may be lower. We may only require 10 – 30% of the eigenvectors.

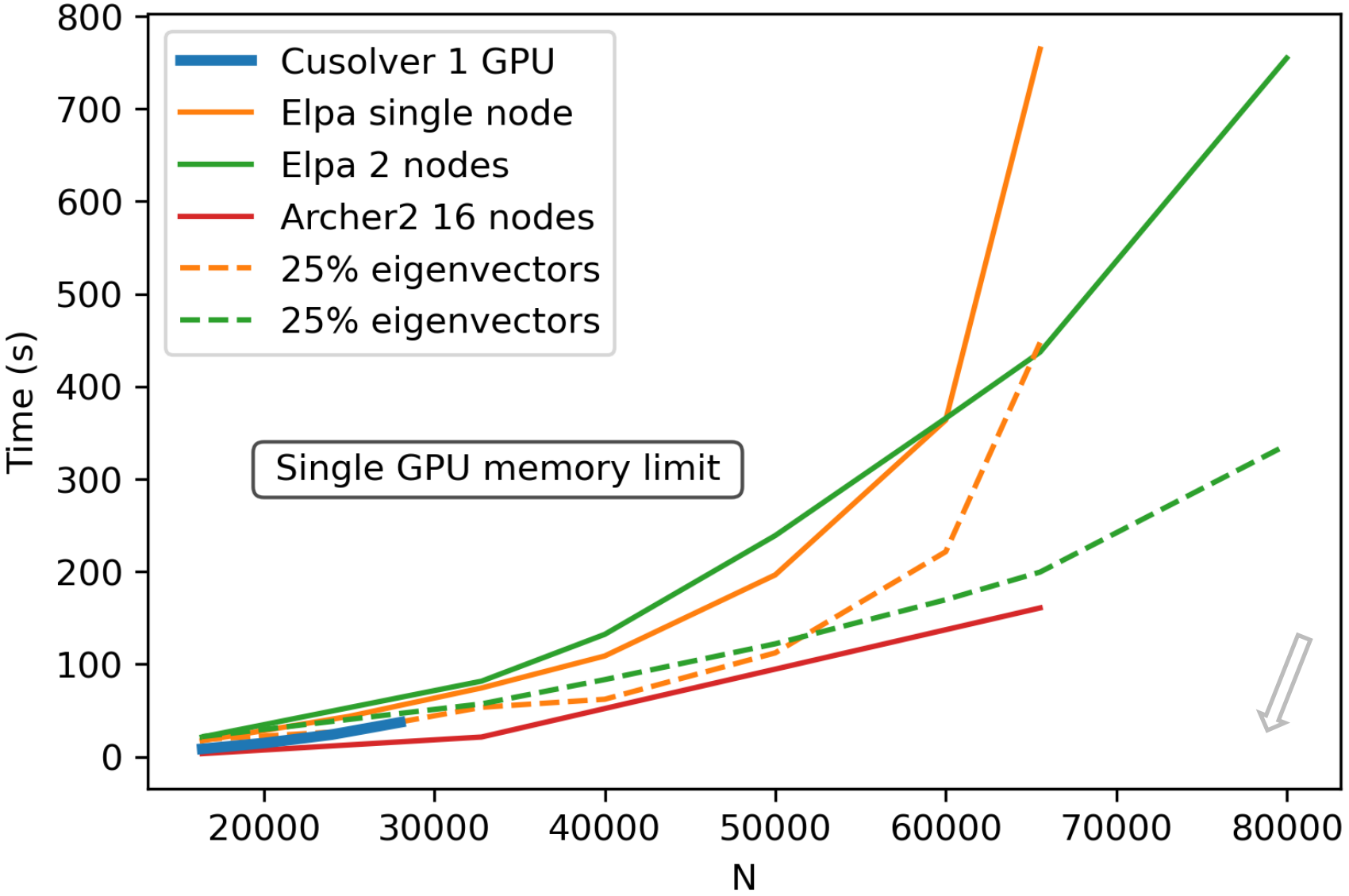
Benchmarking Eigensolvers

ScaLAPACK is a standard interface for parallel eigensolvers.



Archer2 results for hybrid OpenMP-MPI (quickest of up to 8 OpenMP threads per process)

GPU Eigensolvers (ELPA)



ELPA has a more parallelisable algorithm.

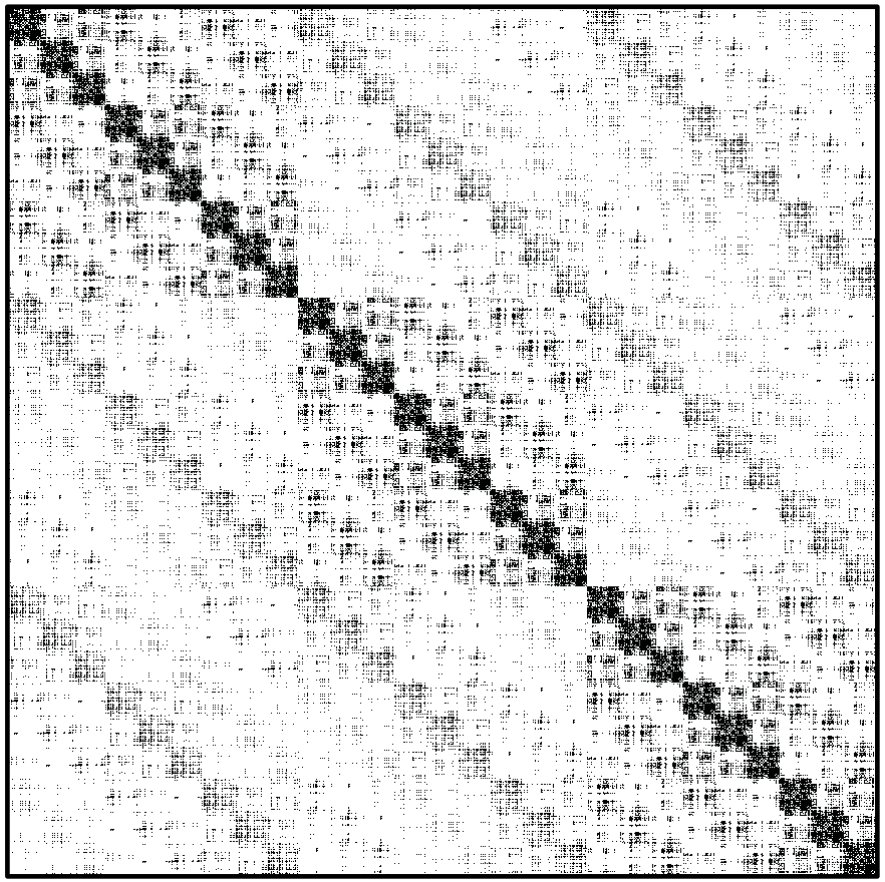
Results shown for 2-stage solver, which can also compute a partial solution.

Scarf results (2x AMD 16 cores + 4 A100 40GB per node) with the 2022.11

GPU Acceleration of the Kohn-Sham Matrix

Creating the Kohn-Sham matrix is complex process in local basis set codes (e.g. CP2K, CRYSTAL), and GPU acceleration is not trivial.

- Matrices are sparse; how can we exploit that?
- First steps:
  - Use the overlap matrix as a simple test-bed
  - Merge into CP2K and CRYSTAL
- Extend to full Kohn-Sham matrix
- 4-centre (2-electron) integrals on GPUs
- Load balancing is challenging!

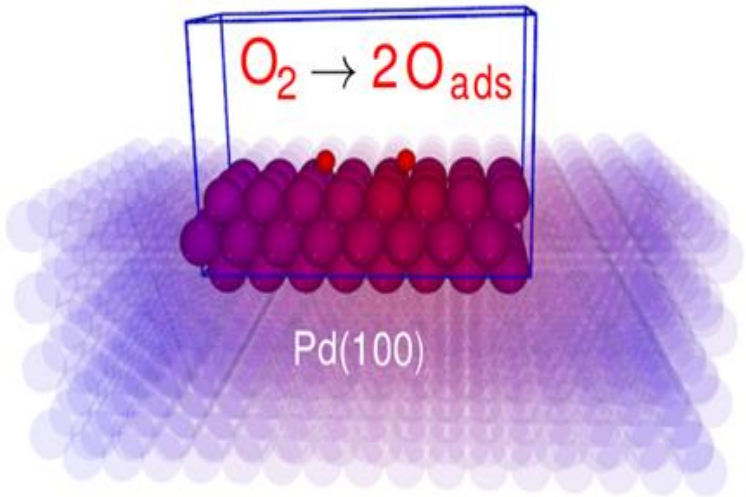


Sparsity pattern of the kinetic energy matrix for a 864 H2O molecule system

Complex materials modelling workflows

Rajany K.V., Alin Elena,
Tom Keal (STFC)

- **Initial focus on *ChemShell* QM/MM package to all PAX-HPC QM codes**
  - New prototype CASTEP interface to ChemShell
  - Work started on closely-coupled CASTEP-ChemShell interface, using a new CASTEP API
  - Setting up benchmarks for CP2K and CASTEP on STFC's SCARF cluster and ARCHER2 for periodic QM/MM of metallic systems, e.g. oxygen adsorption on palladium



- **Extend to new workflows for multiscale modelling**
- **Optimise I/O for exascale HPC (targeting multiple PAX-HPC software projects)**

Engineering and Physical Sciences Research Council

ExCALIBUR 10

PAX-HPC - Modelling particles at exascale: from atoms to galaxies

Part 2: Earthly fluids to Galaxies

Benedict D. Rogers
University of Manchester

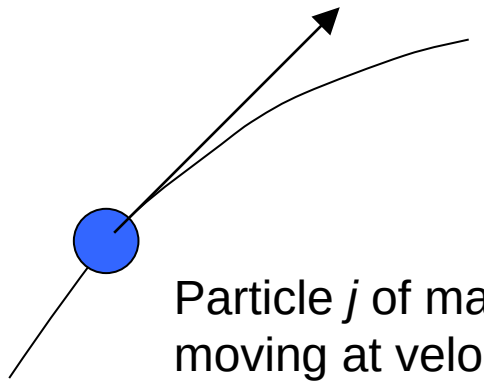
Meshless methods: Basic Idea of SPH

Meshless Our computation points are **particles** that now **move** according to governing dynamics , e.g. Navier-Stokes Equations

Particles move along a trajectory by **integrating** in time their velocity & acceleration

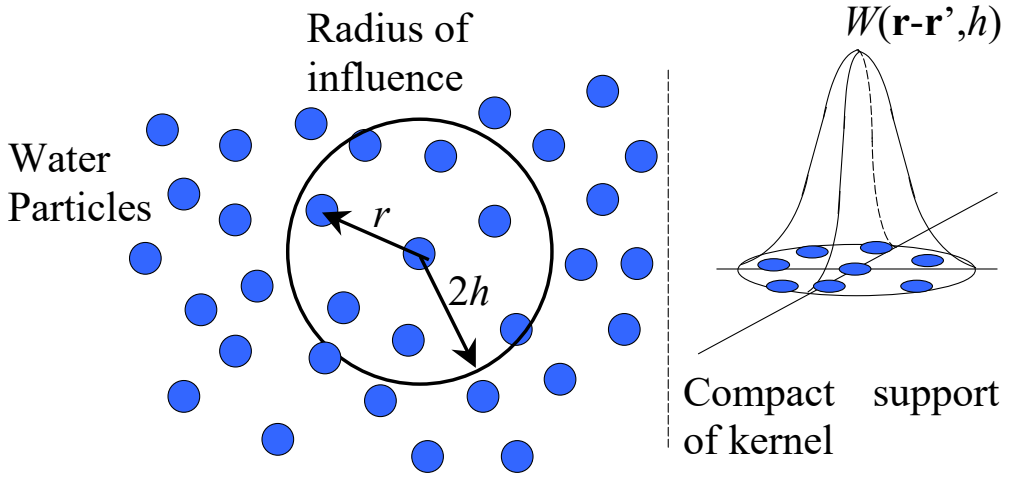
Particles possess **properties** that travel with them, e.g. density, pressure; these can change with time

Local Interpolation (summation) with a **weighting function** (kernel) around each particle to obtain fluid/solid properties



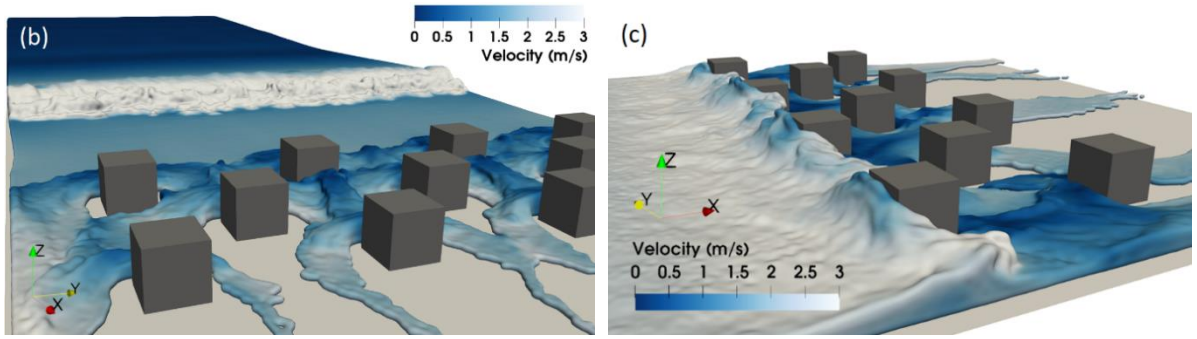
Particle j of mass m_j moving at velocity $\mathbf{v_j}$

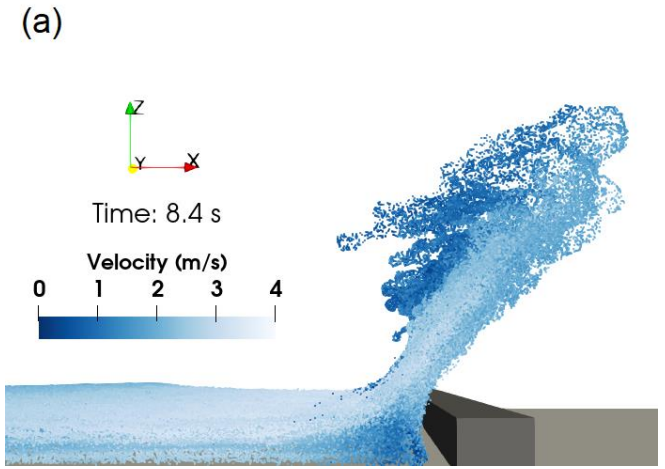
$$\langle f(\mathbf{x})\rangle = \sum_j f_j \, W(\mathbf{x}-\mathbf{x}_j)\frac{m_j}{\rho_j}$$



SPH Application: Tsunami inundation

One of the key applications of SPH is the simulation of violent wave impact on structures: **RESILIENCE + SUSTAINABILITY**





DualSPHysics (cpu / gpu)

MANCHESTER 1824

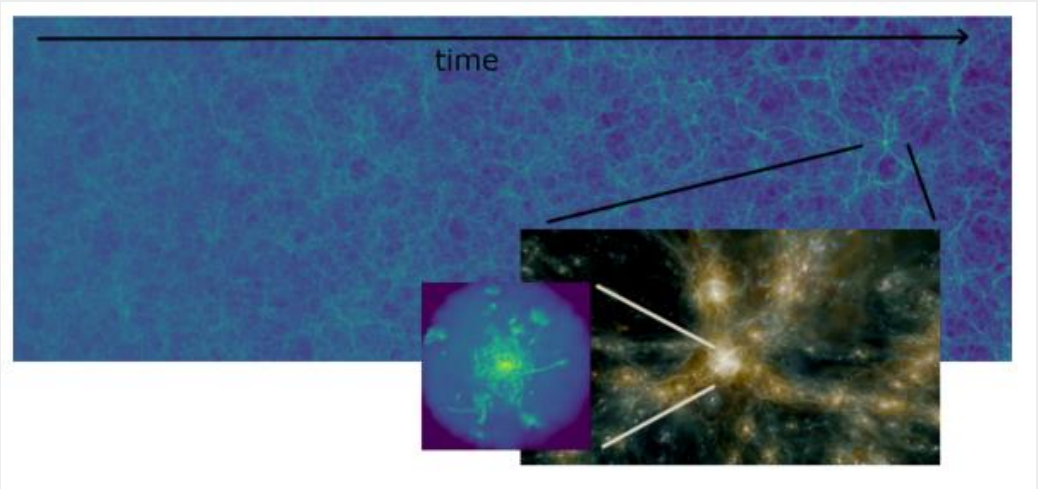
UNIVERSITY OF UDAYANA
Excellent, Self-reliance, Culture-oriented

Massively Parallel Particle Hydrodynamics (MPPH) Working Group

Astrophysics:

Durham University

Universiteit Leiden
The Netherlands




```
  ______        _________________
 / ___/ |      / /  _/ ___/_  __/
 \__ \| | /| / // // /_    / /
___/ /| |/ |/ // // __/   / /
/____/ |__/|__/___/_/     /_/
```

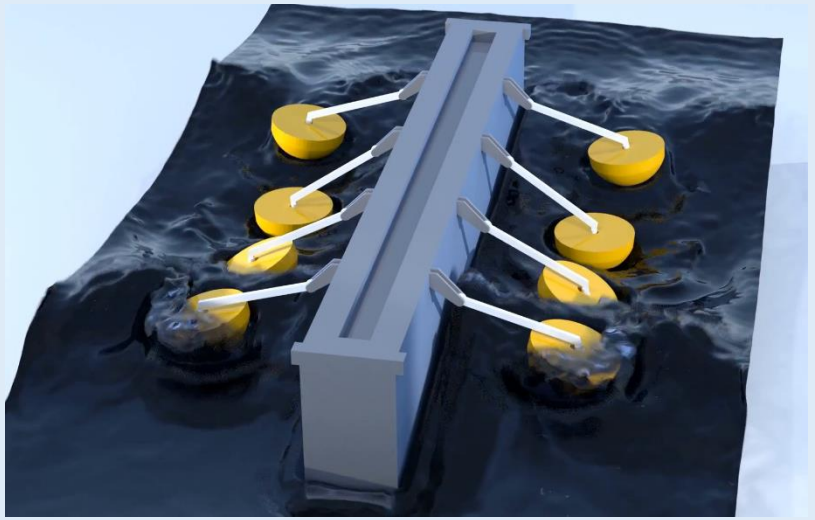
Massively-parallel **CPUs** with Task-based Parallelism

Engineering:

MANCHESTER 1824

University of Hertfordshire UH

UKRI Science and Technology Facilities Council



cpu gpu
DualSPHysics

GPU-acceleration

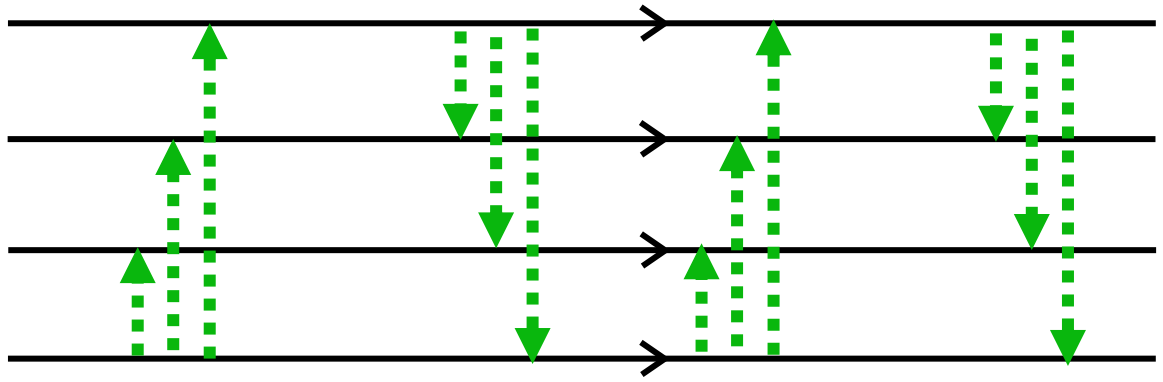
Challenges to develop Exascale-ready SPH

MPPH group has identified 4 key challenges (Bower, Rogers & Schaller, CICESE 2022):

1. Communications

Sending/Receiving data using asynchronous & overlapping communications → unpredictable delays

4 nodes



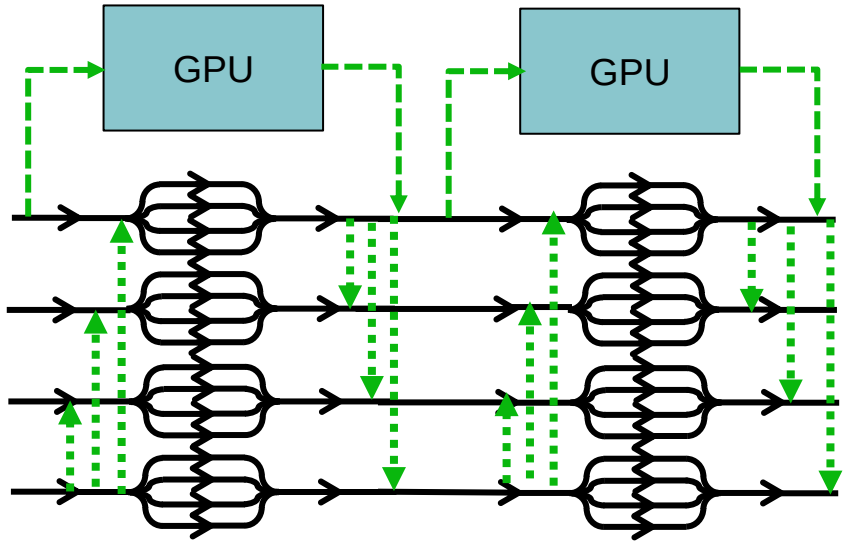
Multiple Processes on Multiple Nodes using MPI (distributed memory)

Challenges to develop Exascale-ready SPH

MPPH group has identified 4 key challenges (Bower, Rogers & Schaller, CICESE 2022):

1. Communications — Sending/Receiving data using asynchronous & overlapping communications → unpredictable delays

2. GPU-enabled code — Exascale machines will be a mix of pure CPUs and accelerators (GPUs)



4 processes x 4 threads

MPI+OpenMP +GPUs

Challenges to develop Exascale-ready SPH

MPPH group has identified 4 key challenges (Bower, Rogers & Schaller, CICESE 2022):

1. Communications — Sending/Receiving data using asynchronous & overlapping communications → unpredictable delays
2. GPU-enabled code — Exascale machines will be a mix of pure CPUs and accelerators (GPUs)
3. New algorithms — How do we exploit time-step adaptivity with variable resolution that scales over 1000k+ cores?

$$\Delta t = CFL \times \min(\Delta t_{force}, \Delta t_{visc}, \Delta t_{diff})$$

Using uniform timestep everywhere?

Really?! Over 500,000+ cores ?

Challenges to develop Exascale-ready SPH

MPPH group has identified 4 key challenges (Bower, Rogers & Schaller, CICESE 2022):

1. Communications	Sending/Receiving data using asynchronous & overlapping communications → unpredictable delays
2. GPU-enabled code	Exascale machines will be a mix of pure CPUs and accelerators (GPUs)
3. New algorithms	How do we exploit time-step adaptivity with variable resolution that scales over 1000k+ cores?
4. Separation of Concerns	Separating SPH from the parallelisation.

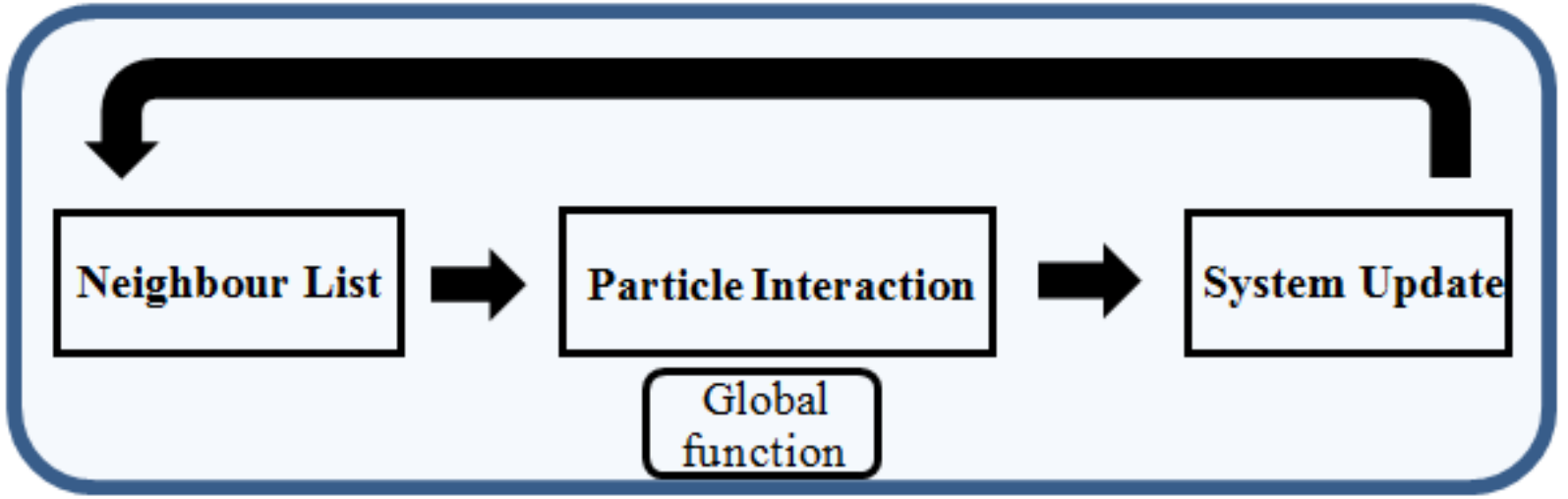
Methodologies for Exascale & Initial Results

1. Task-based parallelism
2. Asynchronous Communication
3. Task-execution on GPUs
4. Separation of Concerns

1. Task-based parallelism

Conventional SPH: **We have 3 main steps:**

1. Neighbour list (NL)
2. Particle Interaction (PI)
3. System Update (SU)



Crespo et al., 2011

SWIFT: Task-based parallelism

Tasks:

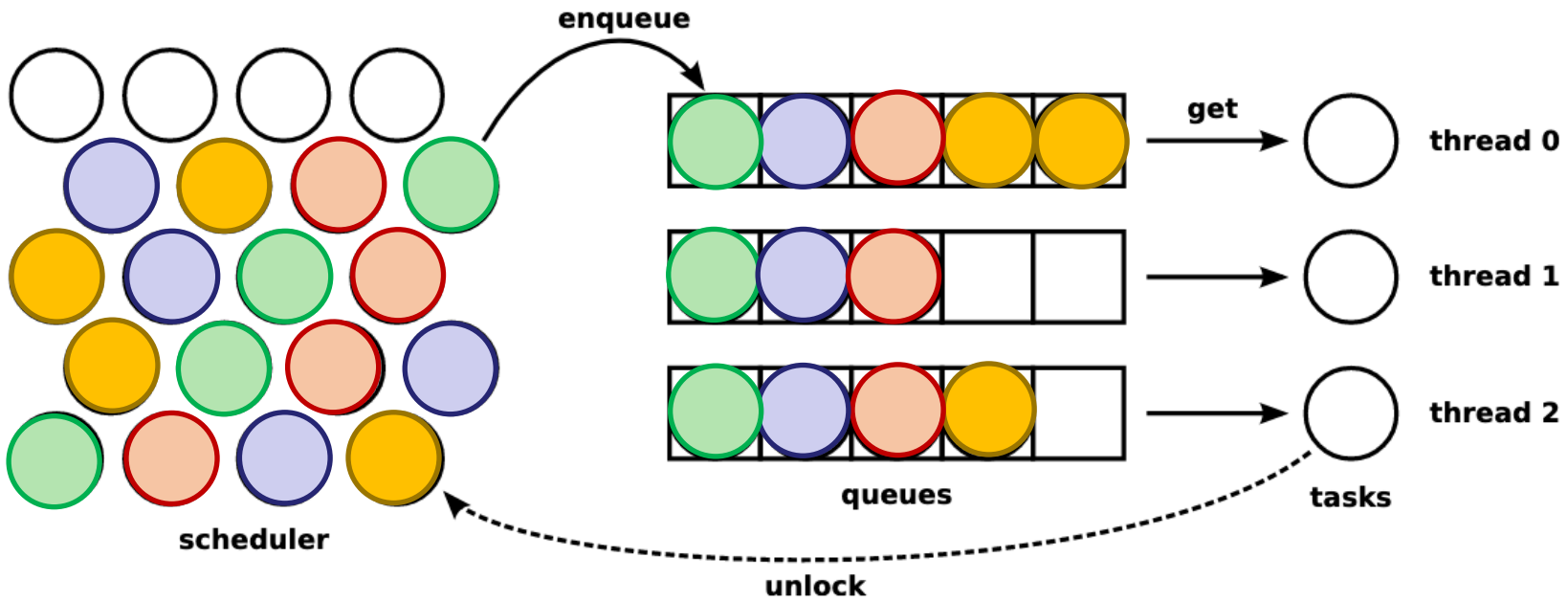
$$h_a = h_0 \left(\frac{\rho_0}{\rho_a} \right)^{\frac{1}{N_D}}$$

$$\rho_a = \sum_a m_b W(r_{ab}, h_a)$$

SEND Data

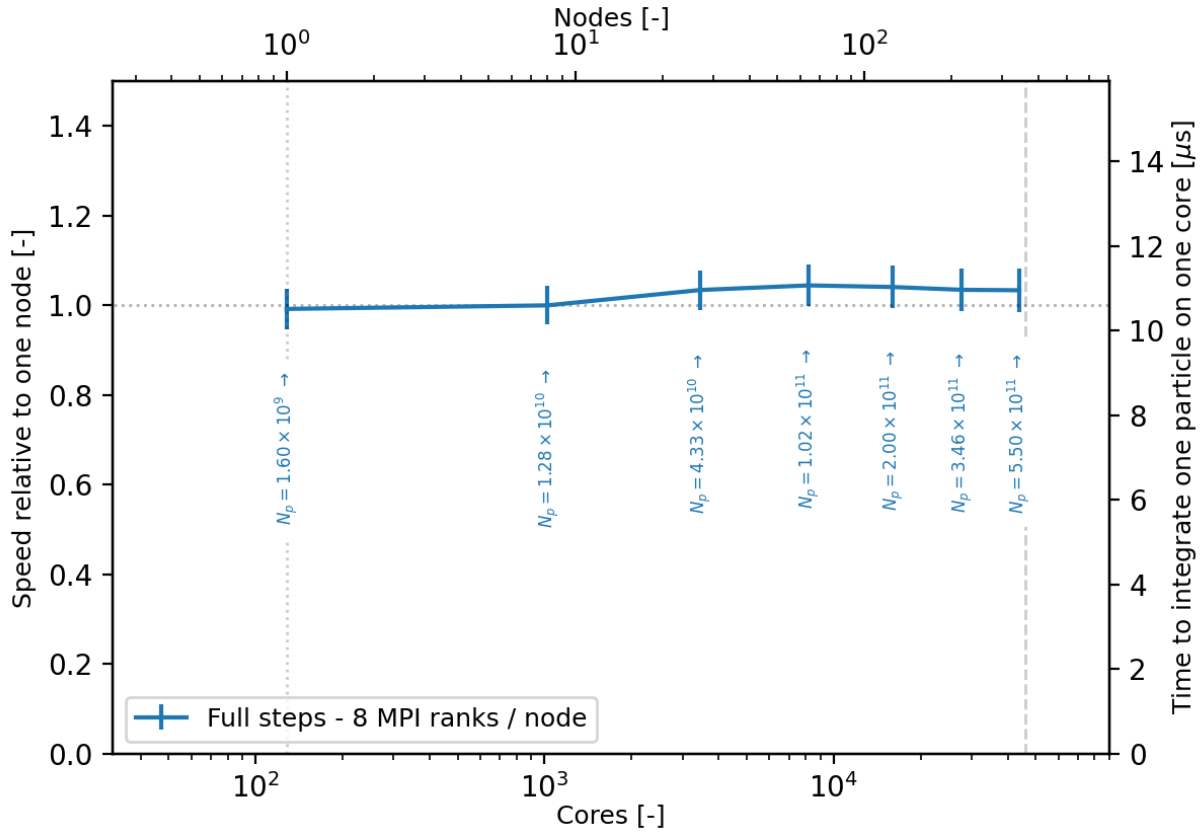
$$\frac{d\mathbf{v}_a}{dt} = -\sum_a m_b \left(\frac{P_a + P_b}{\rho_a \rho_b} \right) \nabla_a W(r_{ab}, h_a) + \mathbf{F}_v + \mathbf{F}_B$$

QUICKSCHED



Detailed full-scale testing of SWIFT

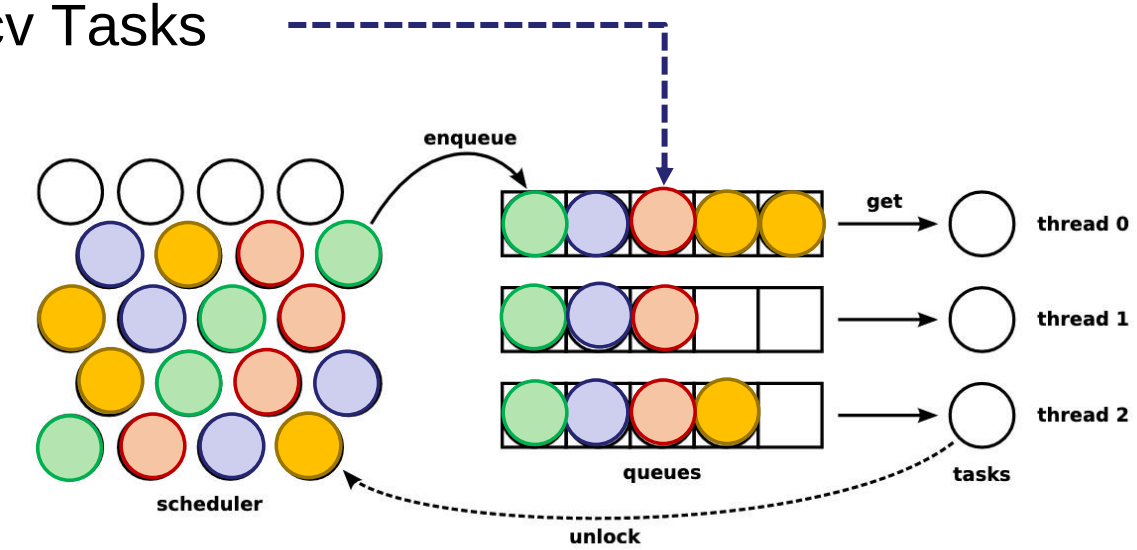
- COSMA-8 computer run by DiRAC (www.dirac.ac.uk University of Durham)
- 43,904 compute cores with 1.9 PetaFLOP peak performance
- Up to **5.5 x 10^{11} Particles** = 12.5 million particles / compute core
- Weak-scaling of 3-D periodic Kelvin-Helmholtz instability



2. Asynchronous Communication

Message Passing Interface (MPI) enables use of

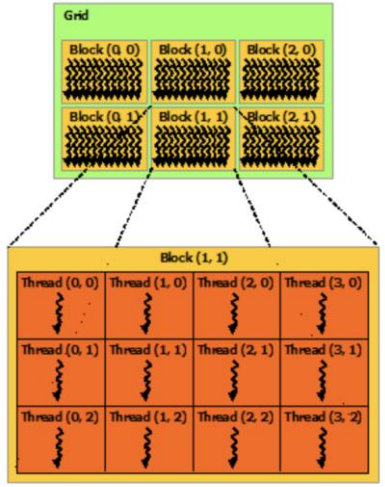
- Non-blocking data transfer → concurrent computation + data transfer
- Scheduler QUICKSCHED identifies Send-Recv Tasks



- BUT, for large simulations:
  - (i) Number of MPI communications grows
  - (ii) MPI Communications can run out of Buffer space

U-Durham using Remote Data Memory Access (RDMA) to investigate this issue.

3. Task-execution on GPUs



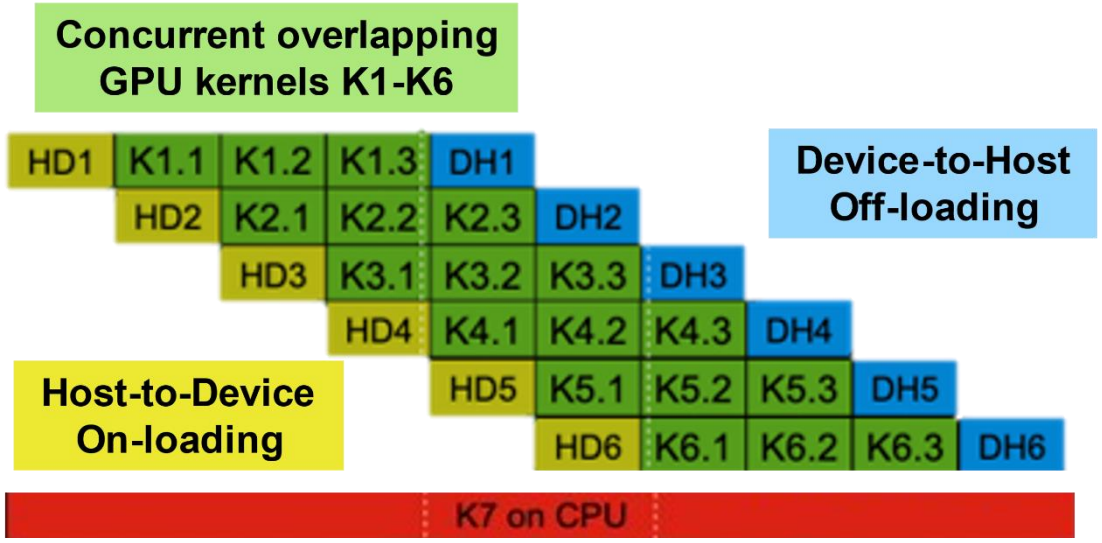
- With blocks of streaming multi-processors, GPUs are ideal for processing **large blocks of particle interactions** (e.g. DualSPHysics, GPU-SPH)

- How to use GPUs to increase concurrency of SWIFT ?

- Our solution is to overlap CPU-GPU communication and compute CUDA kernels

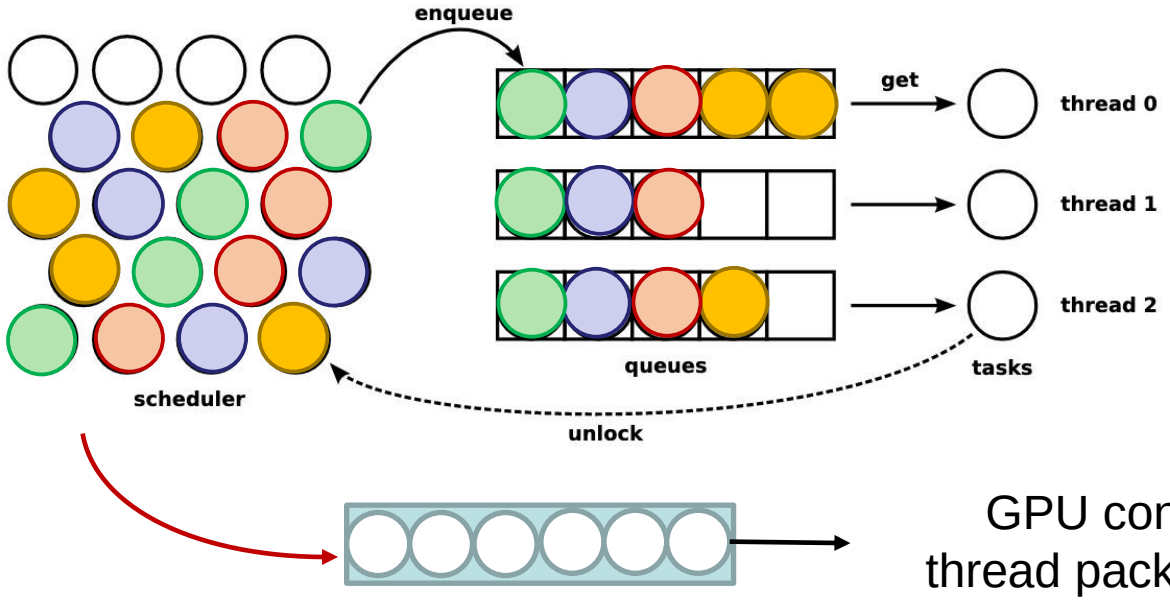
BUT

- The data structures in SWIFT (AoS) is not well suited to GPU architecture.
- How many Tasks per GPU stream do we allocate and role of pointers?

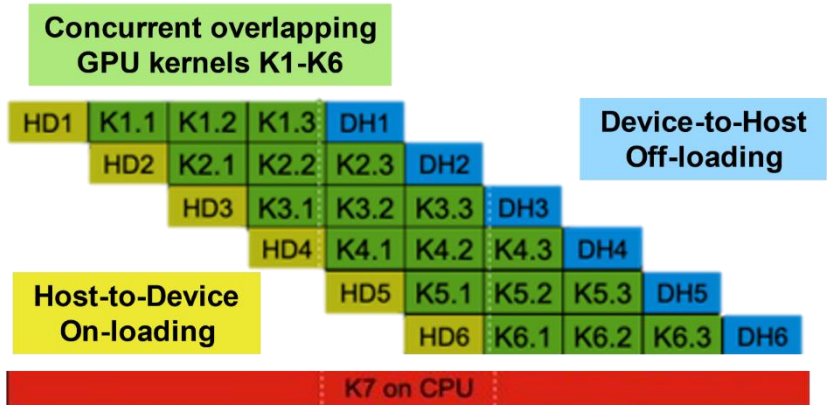


Increase concurrency: GPU on-loading/off-loading

Abouzied Nasar: Packing CPU data for GPU execution



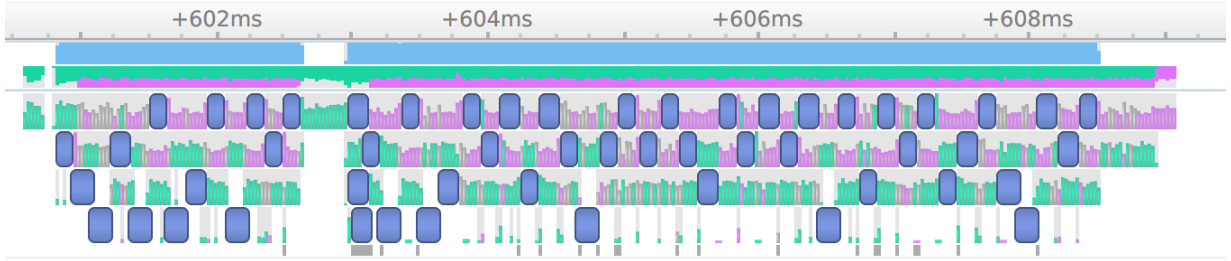
GPU control thread packs data



Optimal method still being developed.

Bundle of 8 tasks per stream: Low kernel concurrency but high memcpy/kernel concurrency

→ CPU/GPU transfer hidden



4. Separation of Concerns

SPH coder does not want to know about the parallelisation, just formulation / physics:

$$h_a = h_0 \left(\frac{\rho_0}{\rho_a} \right)^{\frac{1}{N_D}}$$

$$\rho_a = \sum_a m_b W(r_{ab}, h_a)$$

$$\frac{d\mathbf{v}_a}{dt} = -\sum_a m_b \left(\frac{P_a + P_b}{\rho_a \rho_b} \right) \nabla_a W(r_{ab}, h_a) + \mathbf{F}_v + \mathbf{F}_B$$

How to achieve this in such a massively parallel code?

SWIFT uses **Task Templates**:

- User-defined Task Graph: identifying dependencies, e.g. compute smoothing length h_i **before** computing smoothing kernel $W(r_{ij}, h_i)$
- **Challenge**: integrate with GPUs

Challenges

- Massive parallelism
  - Need new algorithms for many common kernels
  - New parallel decompositions & exploit parallelism at all levels
  - Task-based parallelism
  - How can we use network accelerators efficiently?
- GPUs
  - Complex data structures are inefficient on GPUs
  - How can multiple MPI processes share GPU data portably?
  - How can we use multiple GPUs efficiently?
- We don’t have a UK (pre-)exascale machine…

Perspectives

- Exascale computing is coming!
- Exascale machines will be heterogeneous: CPU + Accelerators (GPU) + Interconnnect
- Presents 4 Key Challenges for SPH: Communications, GPU-enabled code, New algorithms, Separation of Concerns
- Methodologies and Results: Interesting / Promising so far …