Hemodynamics with lattice Boltzmann

Rupert Nash

r.nash@epcc.ed.ac.uk

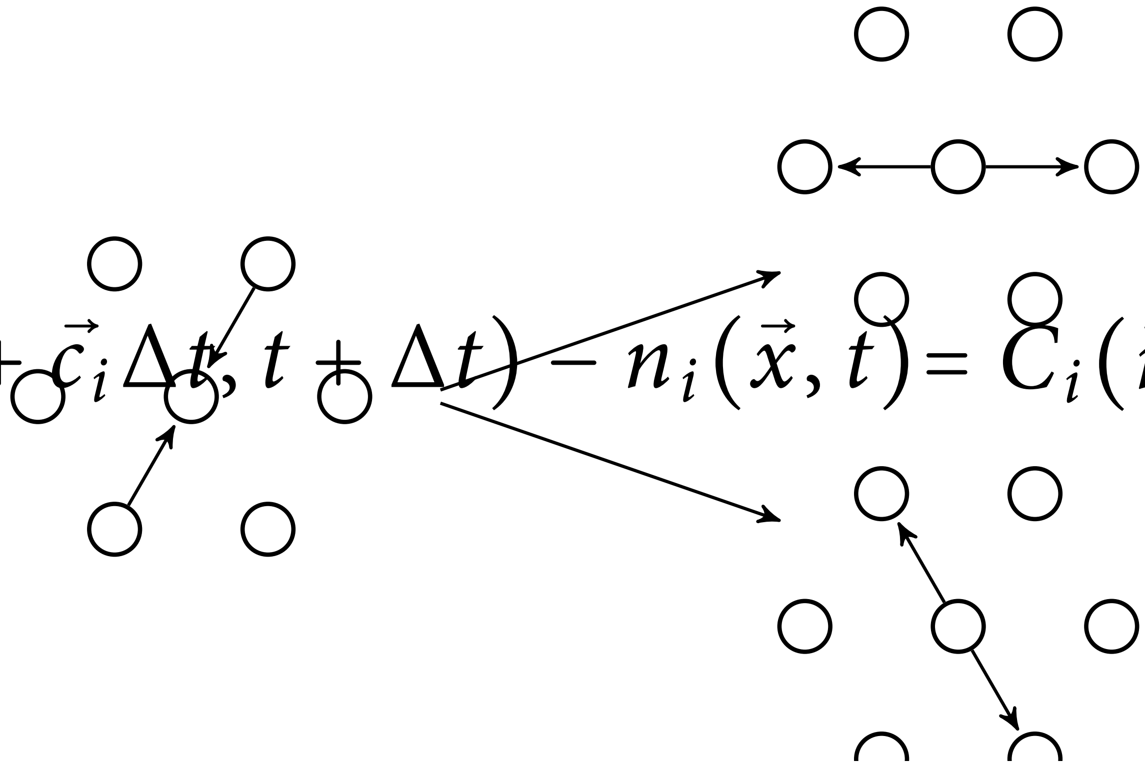
@rupe_nash

The lattice Boltzmann method

A lightning summary

LB prehistory - lattice gas cellular automata (1980's)

$$n_i(\vec{x} + \vec{c}_i \Delta t, t + \Delta t) - n_i(\vec{x}, t) = C_i(\vec{n})$$



One collision rule – choose randomly between two output states

Boltzmann BGK Equation

$$\left[\partial_t + \vec{v}\cdot\nabla_{\vec{x}} + \vec{g}\cdot\nabla_{\vec{c}}\right] f = -\frac{1}{\tau}\left(f - f^{(0)}\right)$$

$$f^{(0)} = \rho\left(\frac{m}{2\pi k_B T}\right)^{D/2} \exp\left(-\frac{m}{2k_B T}|\vec{c}-\vec{u}|^2\right)$$

$$\rho = \int f \,\mathrm{d}^3\vec{c},$$

$$\rho u_\alpha = \int f c_\alpha \,\mathrm{d}^3\vec{c},$$

$$S_{\alpha\beta} = \int f(c_\alpha c_\beta - c_s^2\delta_{\alpha\beta})\,\mathrm{d}^3\vec{c};$$

Discretise in velocity space

- Reduce continuous three dimensional space to discrete set of velocities

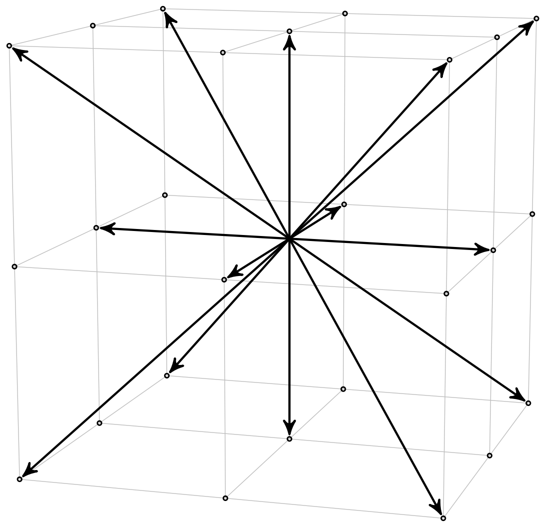
- DVBE:

$$\partial_t f_i + \vec{c}_i \cdot \nabla f_i + [\vec{g} \cdot \nabla_{\vec{c}} f]_i = -\frac{(f_i - f_i^{(0)})}{\tau}$$

Force term too long for slides!

Everyone uses simple velocity sets

- Usually D3Q19 (or D3Q15 or D3Q27)



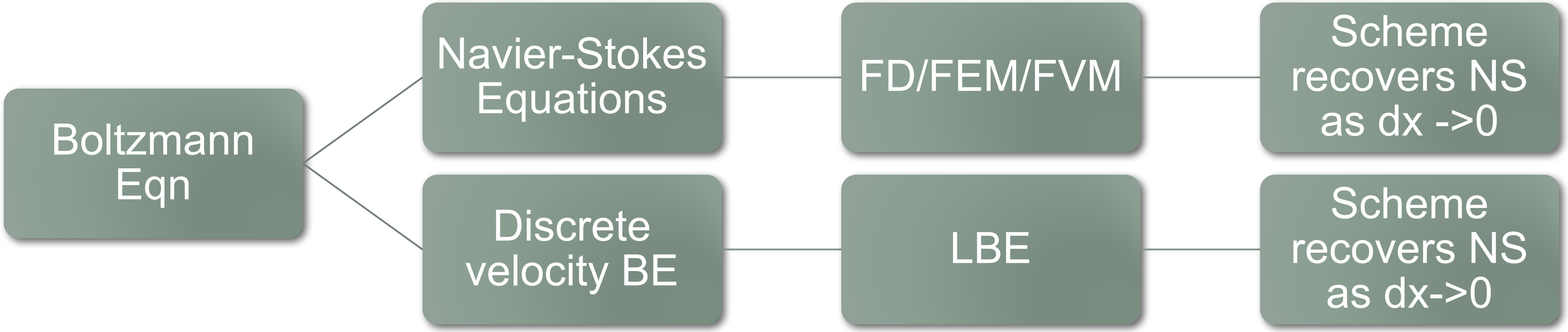
- OK as long as symmetric “enough”
- Introduces more errors but still second order in Δx
- Except sometimes for turbulence?
  - White & Chong, J. Comput. Physics (2011); S. K. Kang and Y. A. Hassan, J. Comput. Phys. 232, 100 (2012).

DVBE ⇒ LBE

- Integrate along each velocity with the trapezium rule for one timestep
- Make simple substitution get explicit equations

$$\bar{f}_i(\vec{x} + \vec{c}_i \Delta t, t + \Delta t) - \bar{f}_i(\vec{x}, t) = -\frac{\Delta t}{\tau + \Delta t/2}(\bar{f}_i(\vec{x}, t) - \bar{f}_i^0(\vec{x}, t))$$

LB vs. CFD


Boltzmann Eqn
Navier-Stokes Equations
FD/FEM/FVM
Scheme recovers NS as dx ->0
Discrete velocity BE
LBE
Scheme recovers NS as dx->0

LB is strong

- Possible to introduce more physics rigorously*
  - Particles
  - Multicomponent, multiphase fluids
  - Liquid crystals
- Meshing is much simpler than FEM (only 1 grid resolution to set)
- Parallel implementation (P<1000) easy!
- Parallel implementation (P<=250,000, efficiency >90%) possible!
- No global communications required (unlike e.g. Poisson solver)

LB is weak

$$c_s = \frac{1}{\sqrt{3}} \frac{\Delta x}{\Delta t}$$

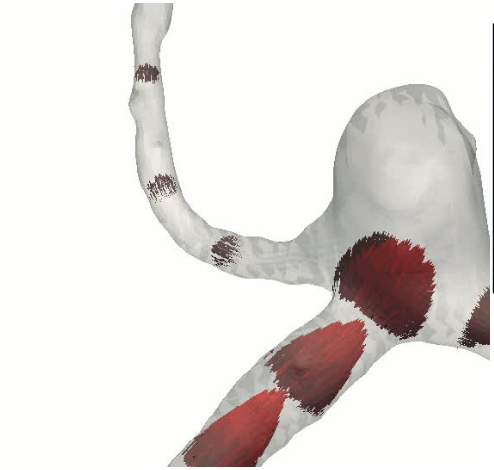
- Speed of sound fixed
  - Need to carefully chose params such that Mach number <<1
  - Mach number important because it is the parameter in the Chapman Enskog expansion
  - Sound takes many time steps to cross domain
  - Pressure takes many time steps to converge
  - Is there really an advantage over just solving the Poisson equation as in most NSE based codes?

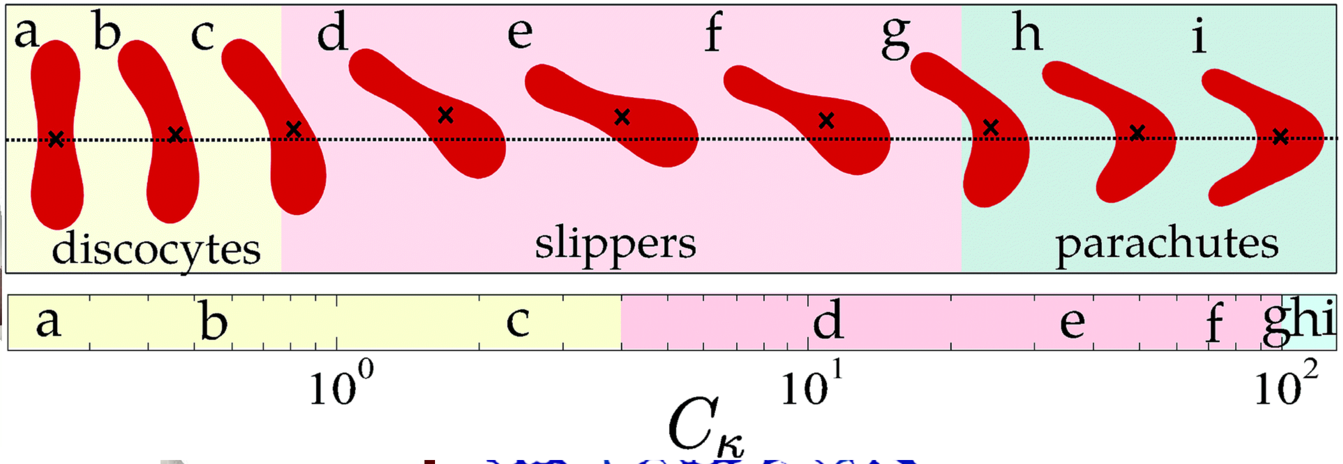
LB is horrible

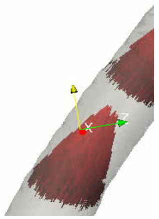
- It struggles to get to high velocities (positivity of distribution functions)
- It struggles to get to low viscosity as has poor stability
- Hence high Reynolds number requires advances techniques: cascaded LB

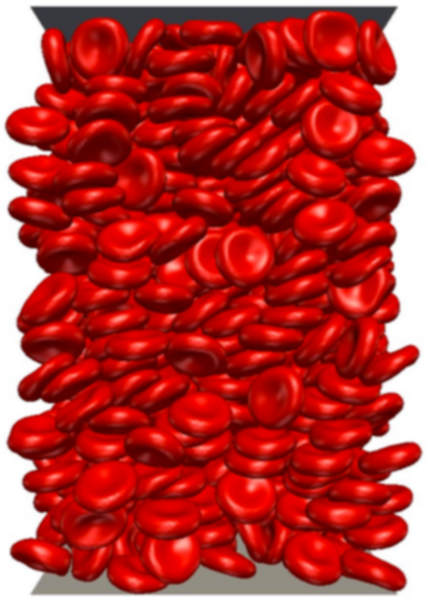
HemeLB applications

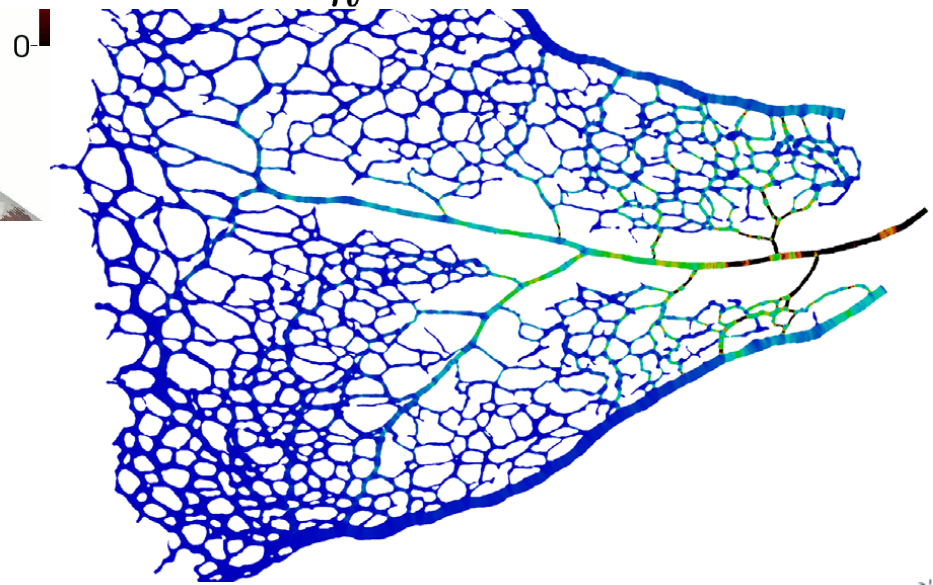
CFD (Colour for doctors)



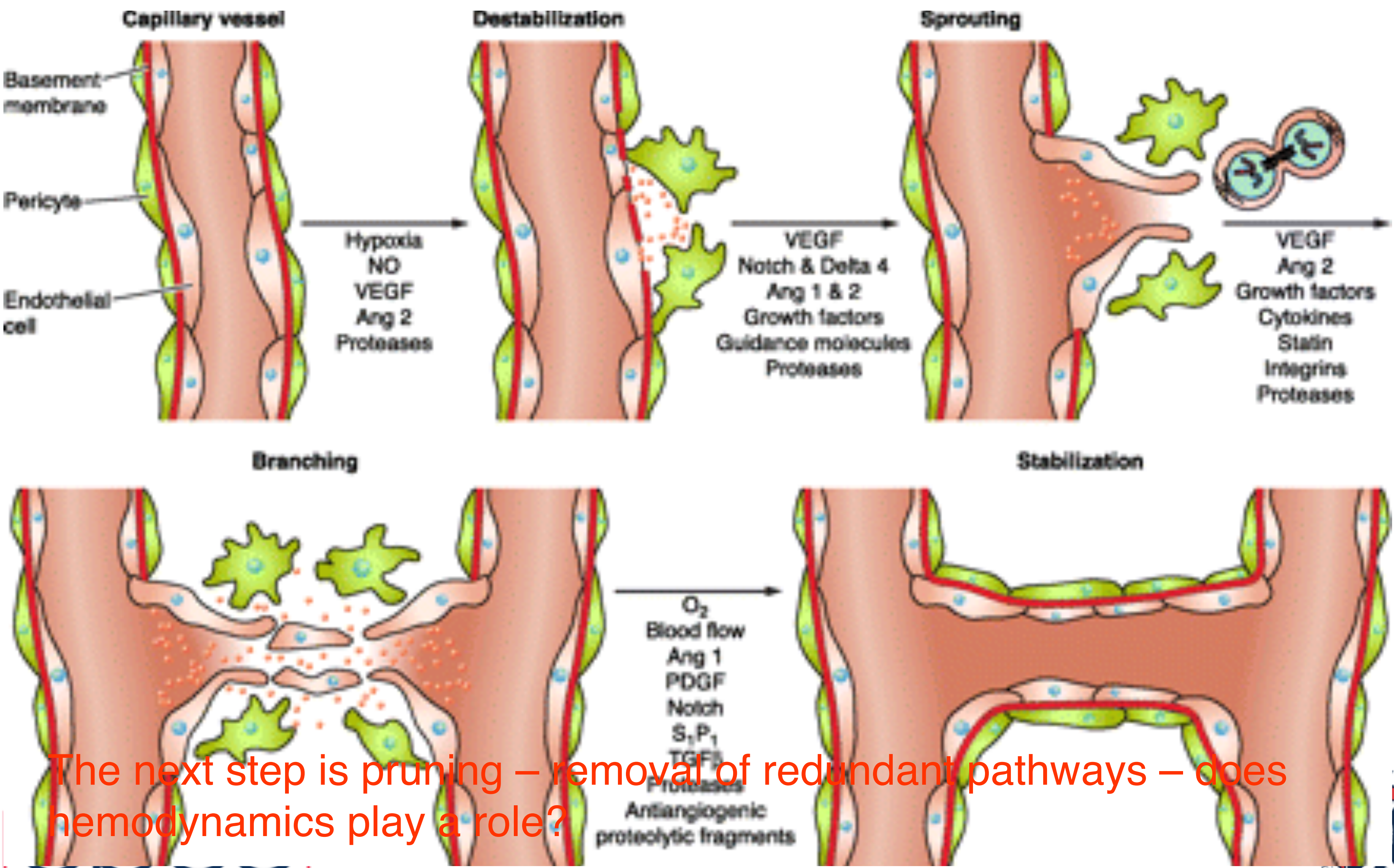






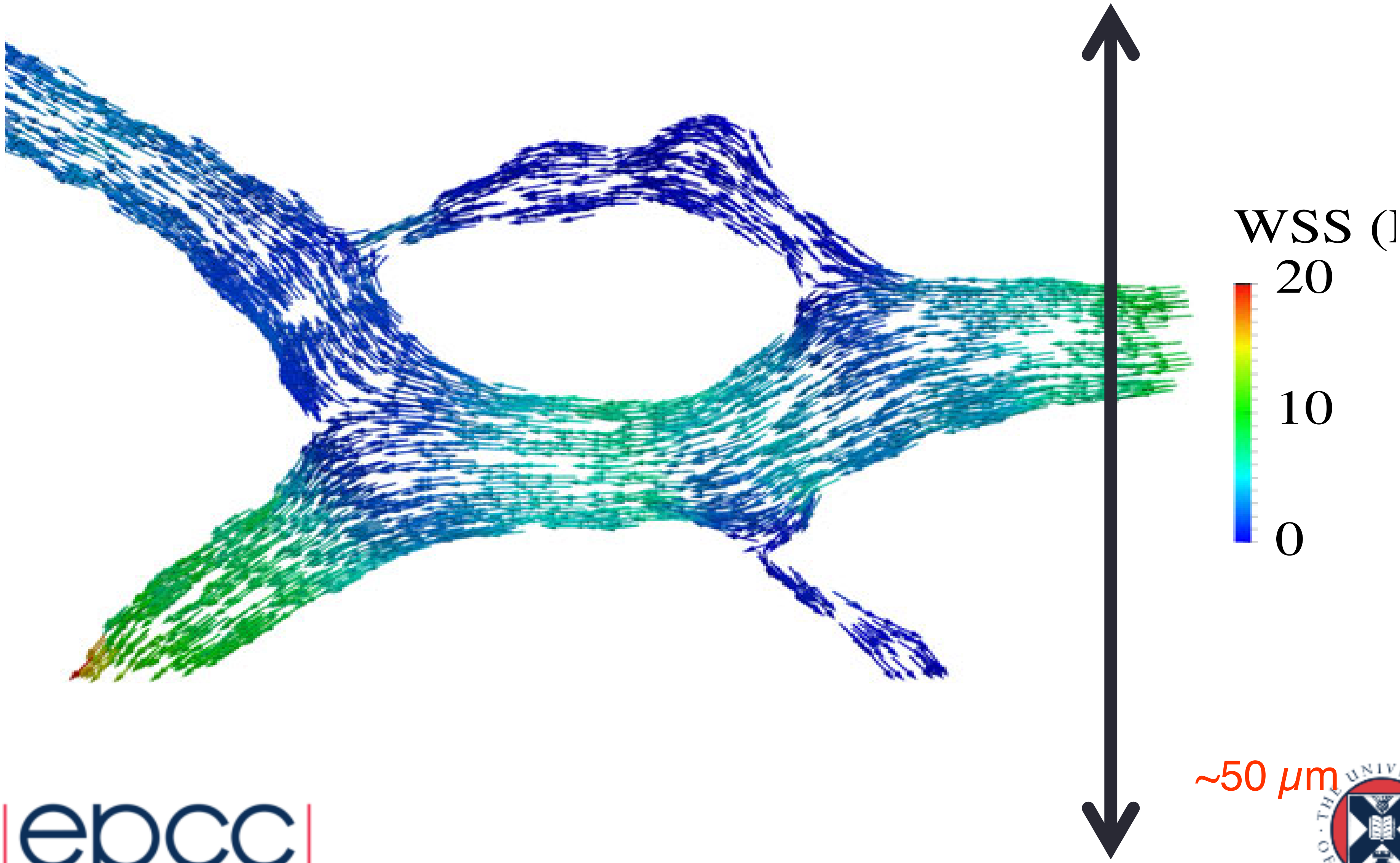


Angiogenesis



The next step is pruning – removal of redundant pathways – does hemodynamics play a role?

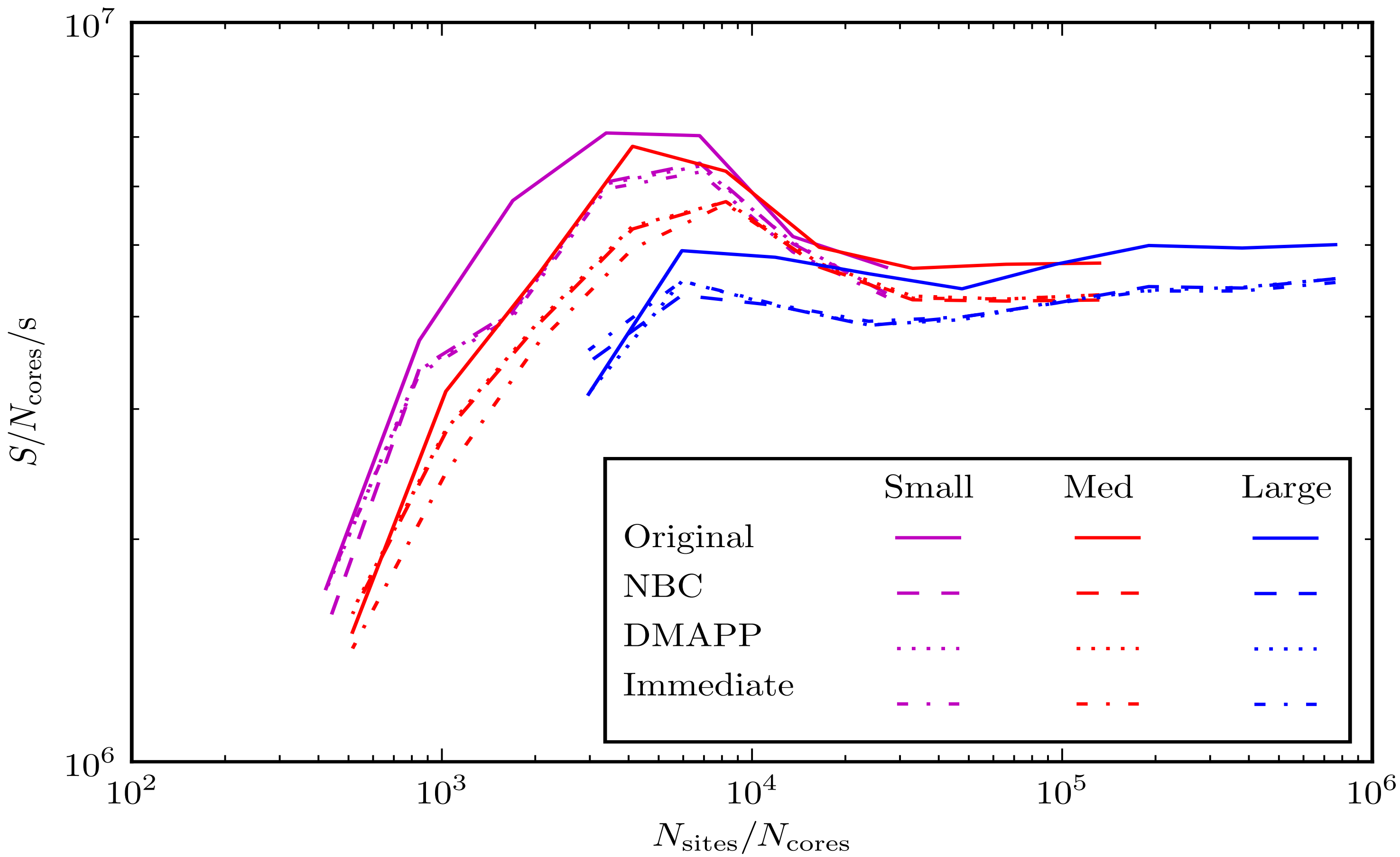
Pruning in action?



Download HemeLB if you want a go

- Started at UCL
- Developers now at UCL, Edinburgh, Brunel, Clemson
- C++, object oriented, heavy use of templates
- Source code on GitHub:
  - https://github.com/UCL/hemelb
- Docker image available too:
  - https://hub.docker.com/hemelb/hemelb

Performance – Site updates per second



Parallel in Time

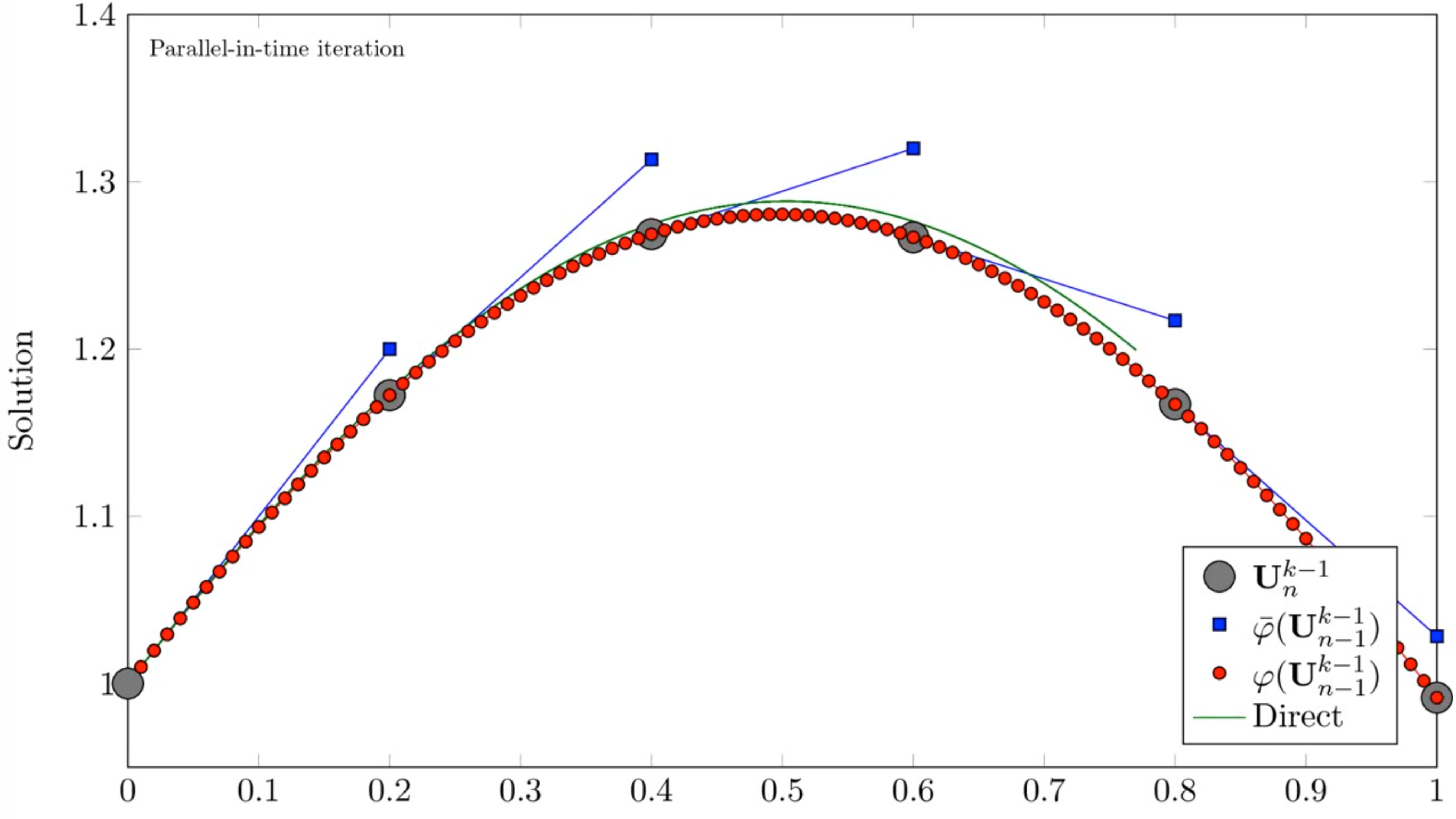
Parallelism in Time

- Parallel-in-time integration can allow the use of even more concurrency to reduce wall clock time
  - Note: will always increase CPU time
- Parareal is the most commonly used PinT method
- Has has been used with LB [Randles 2014, J Comp Phys]

Parareal

- Treat your simulation as an initial value problem
- Have a fine operator (F) and coarse one (G).
  - Cost of applying G must be << F
  - G can be less accurate
  - G can be on a coarser grid
- Apply G several time to advance to target time
- From intermediate points start multiple F simulations in parallel
- Correct and iterate

Parareal


Parallel-in-time iteration
Solution
1.4
1.3
1.2
1.1
1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
$\mathbf{U}_n^{k-1}$
$\bar{\varphi}(\mathbf{U}_{n-1}^{k-1})$
$\varphi(\mathbf{U}_{n-1}^{k-1})$
Direct

Parareal for HemeLB

- Collaboration with Derek Groen (Computer Science, UoB London) and Daniel Ruprecht (Mathematics, UoLeeds)
- Project began October
- Will implement the fine and coarse operators with existing simulation
- Are working on coarsening and refinement operations
  - Requires mapping between two grids in parallel with different decompositions
- Unclear how well this will work for lattice-Boltzmann

Public cloud for HPC?

Experience with HemeLB on AWS and Azure

Comparison: policy

Public cloud

- Pay provider through grant
- On demand (minutes-months)
- User defines security
  - You have root on VM
- Many services (Web, DB, serverless, ML, analytics)

HPC centre

- Apply for time through grant process
- Batch system (minutes-days)
- Centre defines security
  - You do NOT have root
- Limited services (filesystem, batch, pre-/post-processing)

Comparison: hardware

Public cloud

- Many node types
- New servers constantly being added
- Network:
  - Ethernet
  - (Some Azure nodes have Infiniband)

HPC centre

- 1-3 node types
- New hardware every 2-5 years
- Network:
  - Specialised low-latency high-bandwidth

Comparison: storage

Public cloud

- Nodes usually have some local
- Object storage
- Various databases
  - Relational
  - Columnar
  - NoSQL
  - Graph

HPC centre

- Nodes might have local storage (NVRAM?)
- Shared parallel POSIX filesystem

Comparison: build

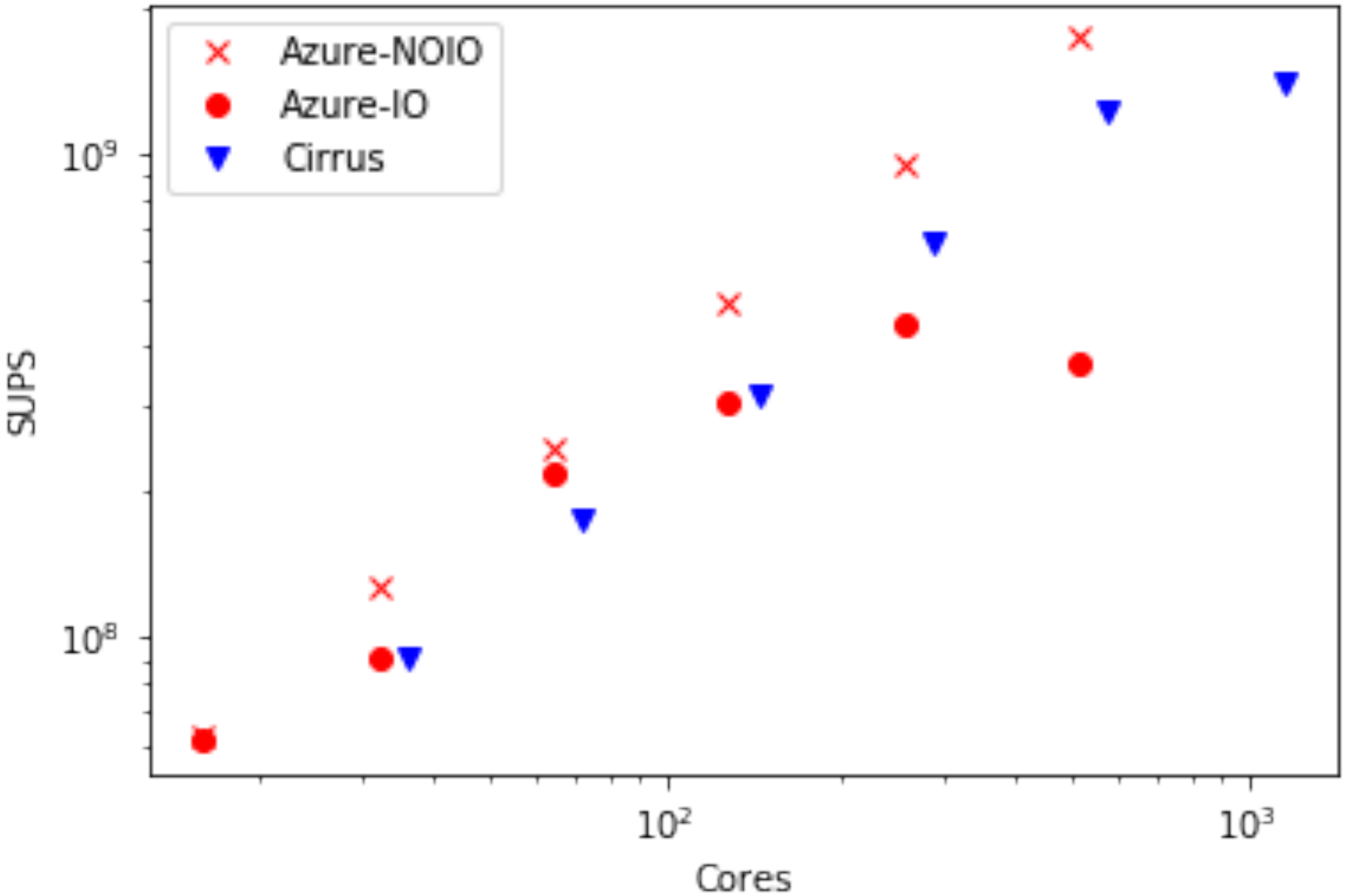
Cloud (Azure)

- yum install ...
- Configure NFS and MPI
- Compile dependencies
- cmake
- make install
- Save image
- Start pool of nodes
- Submit job
- Stop pool

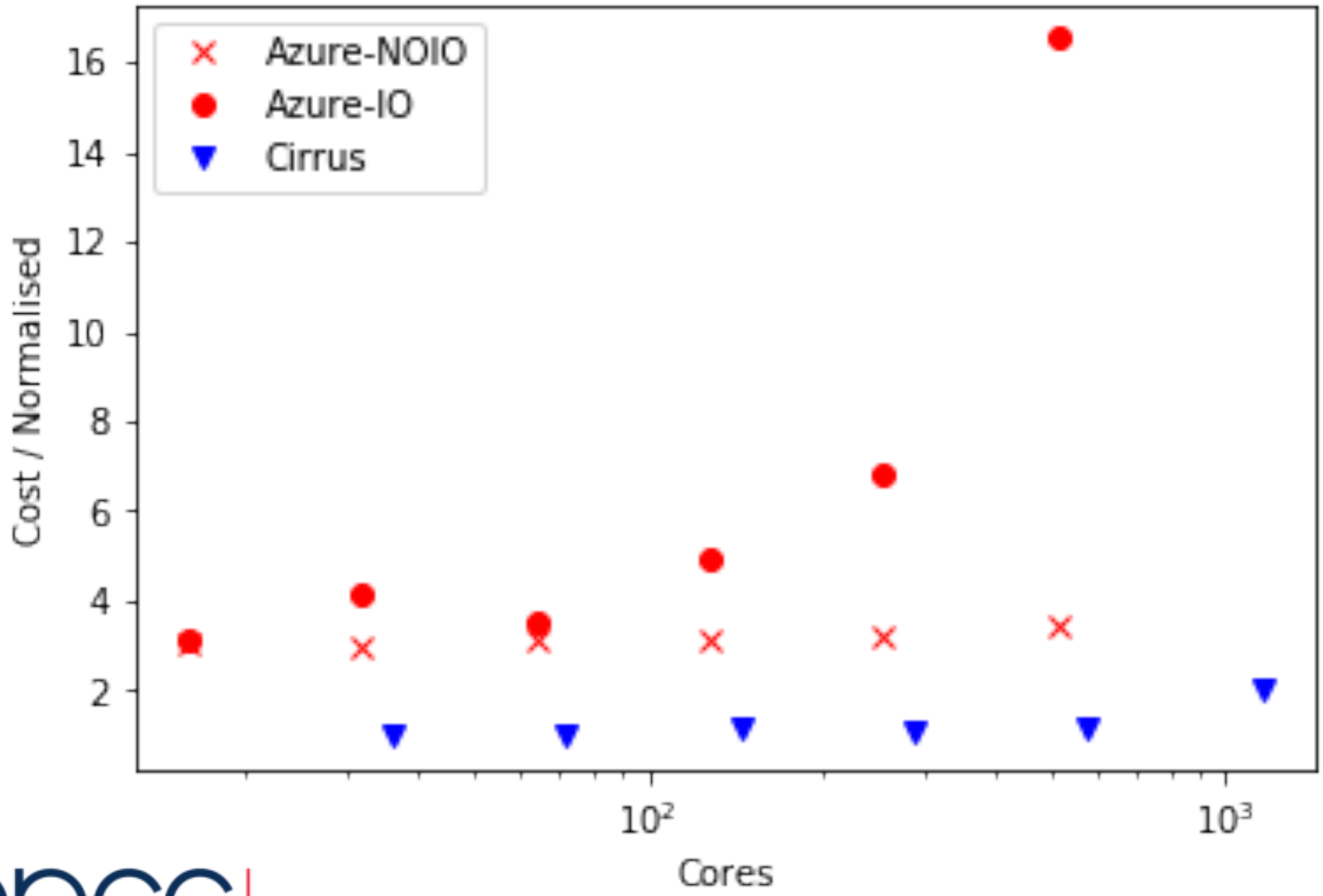
HPC (Archer)

- module load ...
- cmake
- make
- qsub

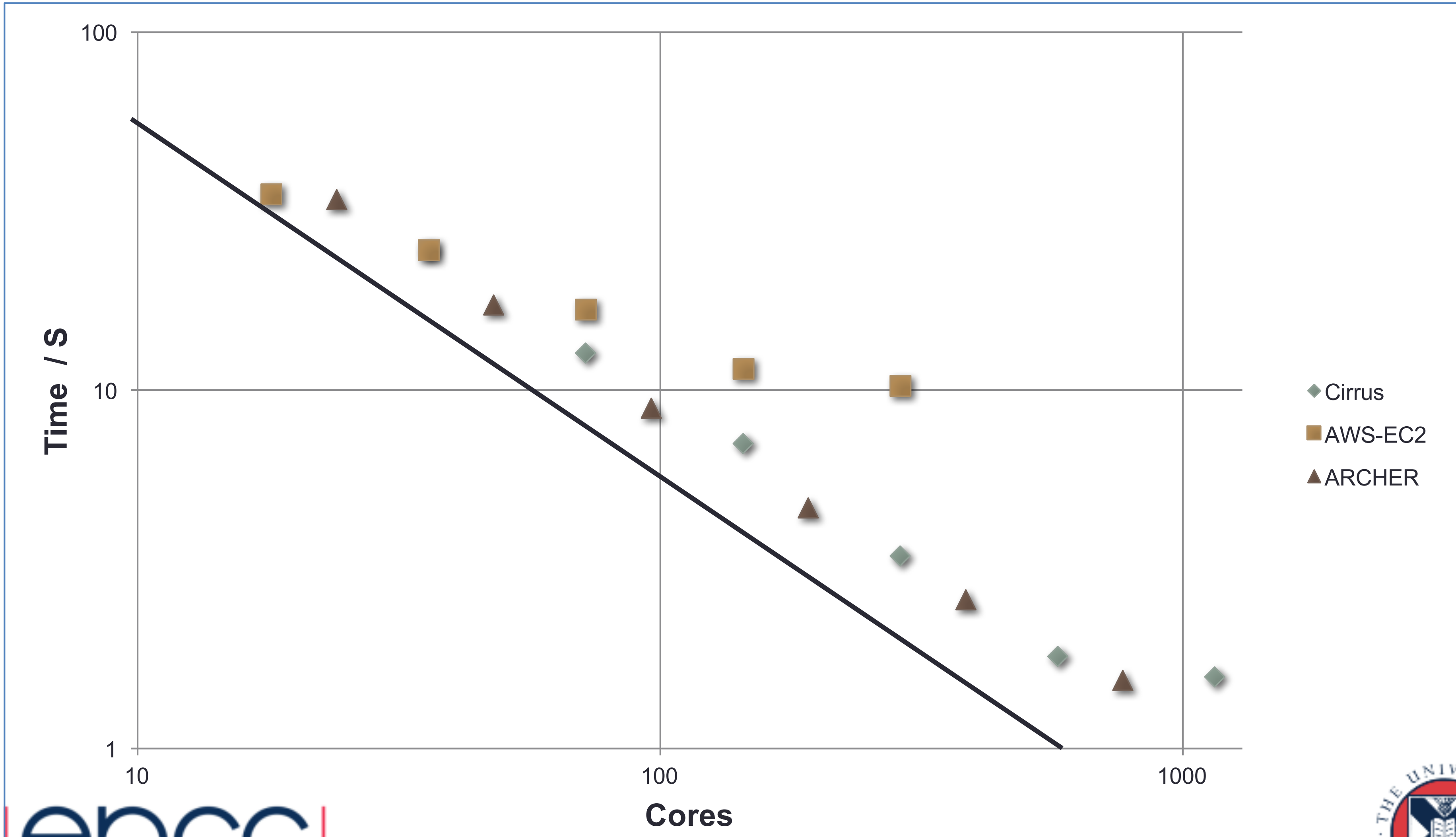
HemeLB performance



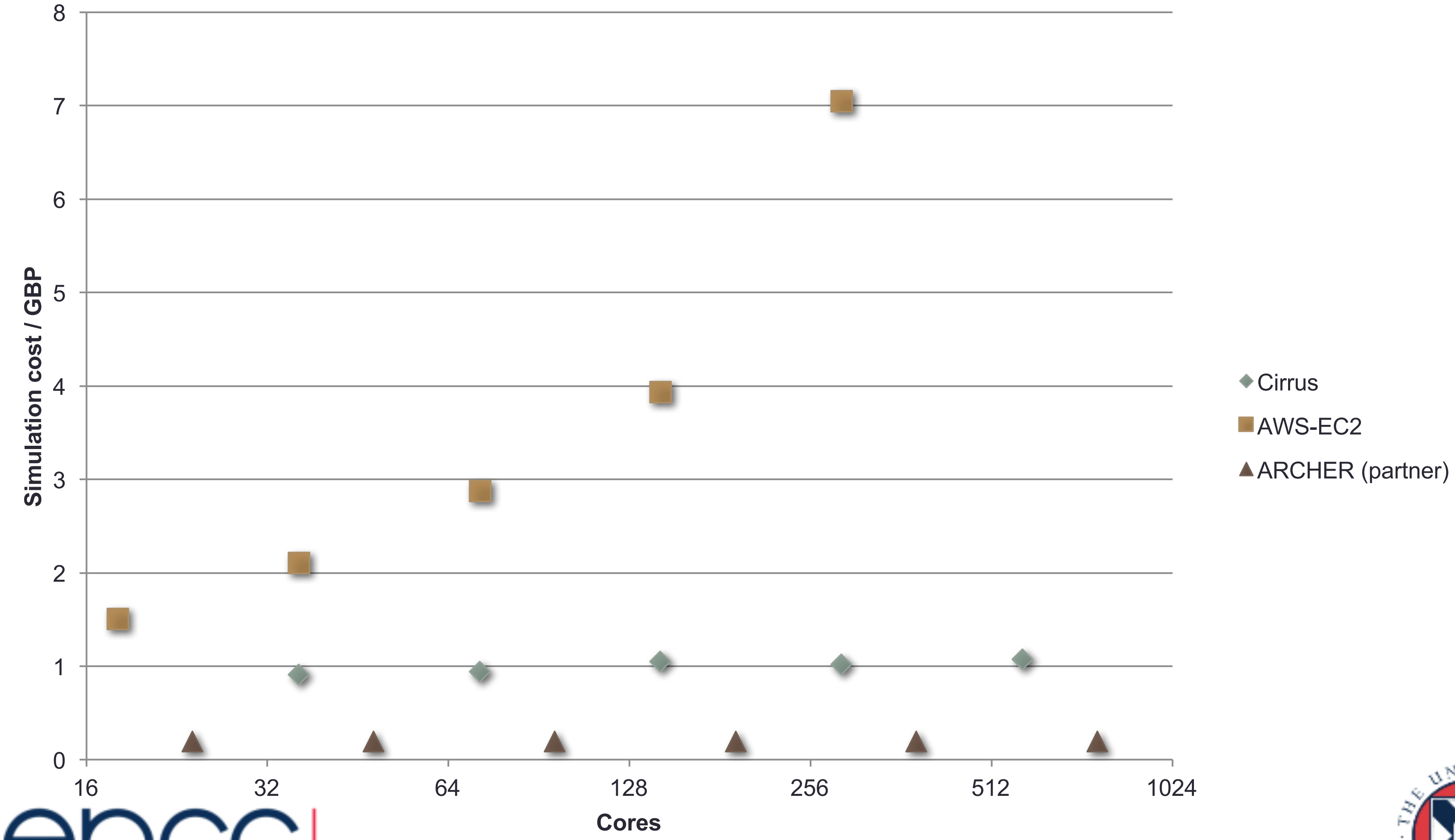
Simulation cost



AWS Performance



AWS Cost



8
7
6
5
4
3
2
1
0
Simulation cost / GBP
16
32
64
128
256
512
1024
Cores
Cirrus
AWS-EC2
ARCHER (partner)

Acknowledgements

- HemeLB was started at UCL (Peter Coveney)
- Major code contributors:
  - RWN, Miguel Bernabeu, Derek Groen, Sebastian Schmieschek, Hywel Carver, Jens Nielsen, Mayeul d'Avezac, James Hetherington, Marco Mazzeo, Steven Manos
- AWS study by Steven Steven
- Azure credit provided by MS Research
- All errors are mine