h-to-*p* efficiently: a Nektar++ update on comparisons of CG and HDG

R. M. Kirby, S. Yakovlev
Scientific Computing and Imaging Institute, University of Utah

D. Moxey, S. J. Sherwin
Department of Aeronautics, Imperial College London

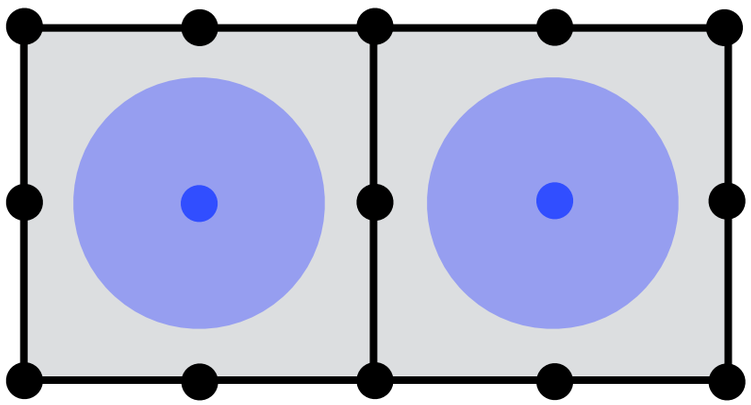
SIAM CSE 2015, Salt Lake City, Utah
14th March 2015

Outline

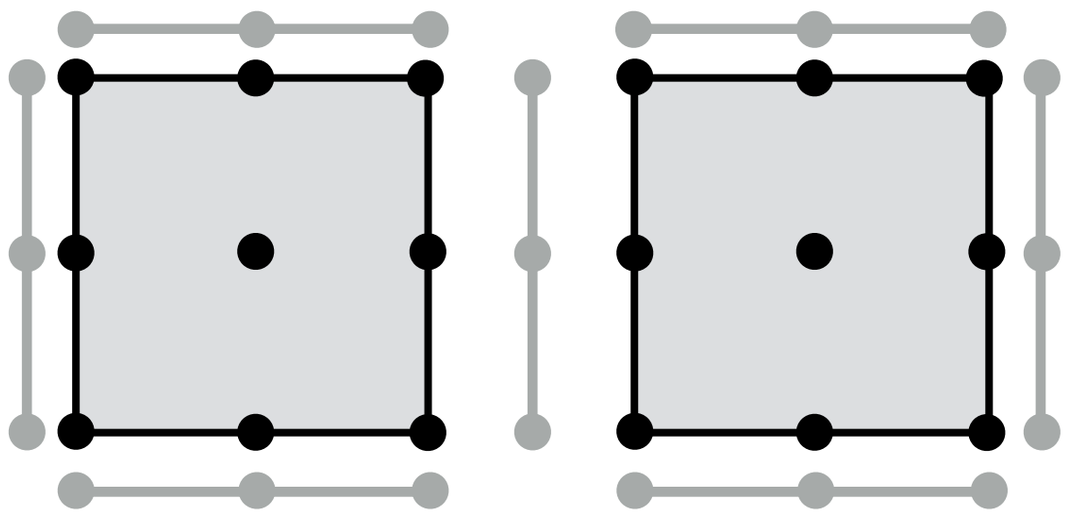
- Overview of CG and HDG formulations
- Goals
- Serial performance
- Parallel performance
- Conclusions

Formulations

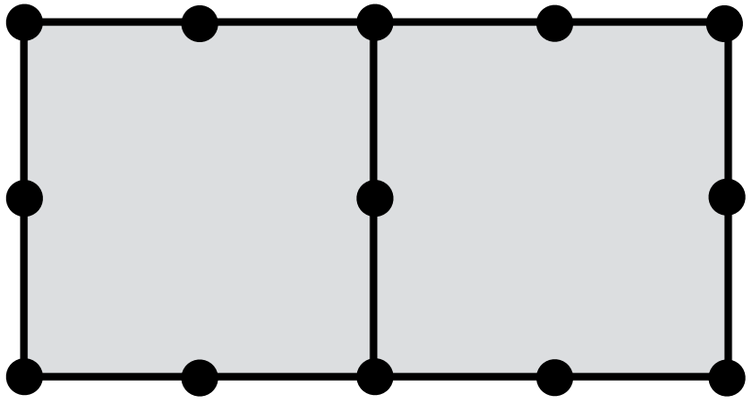
CG



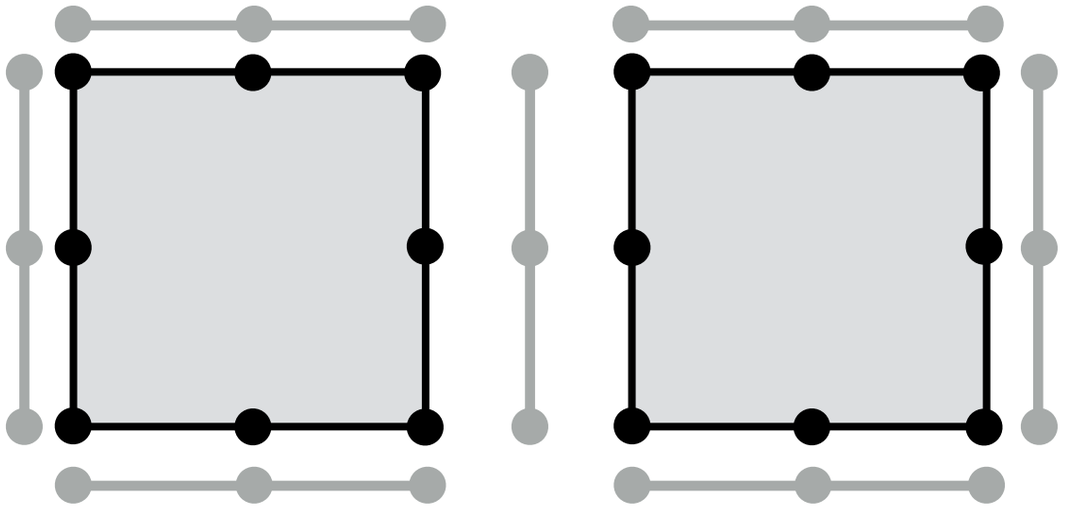
DG



static condensation



(statically condensed) CG



HDG

Formulations

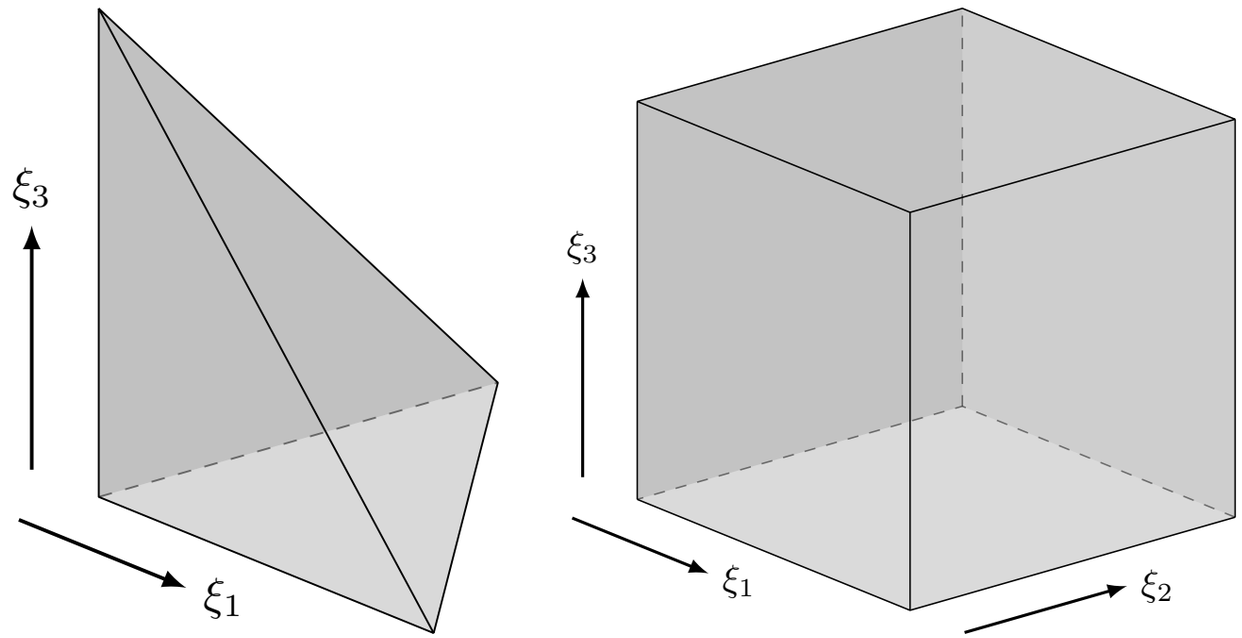
Continuous Galerkin (CG)

- Great for elliptic problems due to low degree of freedom (dof) count
- Established method: large body of knowledge

Discontinuous Galerkin (DG)

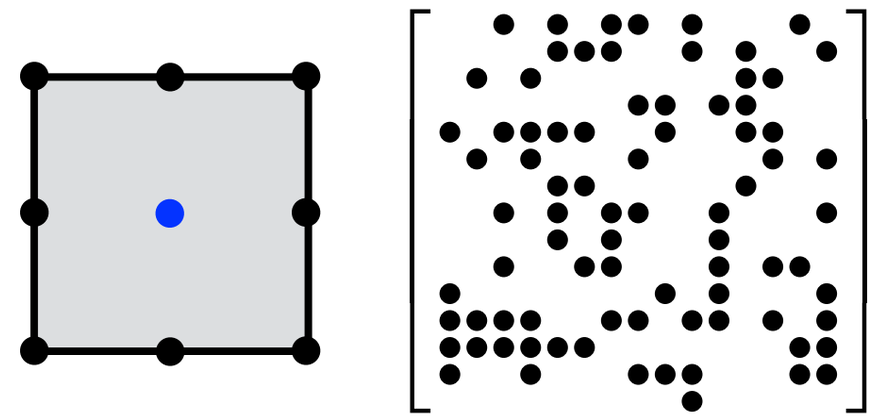
- Has attractive properties (local conservation)
- Great for hyperbolic problems
- Not as attractive for elliptic: higher DOF count

CG on an element



$\nabla^2 u$

Inner product with basis functions, integrate by parts



static cond.

$$\begin{bmatrix} L_{bb} & L_{ib} \\ L_{ib}^{\mathsf{T}} & L_{ii} \end{bmatrix}$$

Hex and tet standard elements

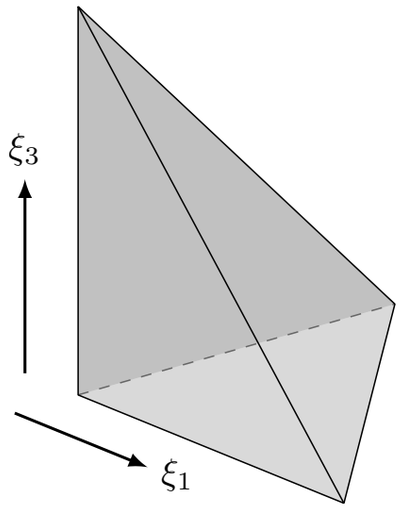
Modified orthogonal basis, order *P*

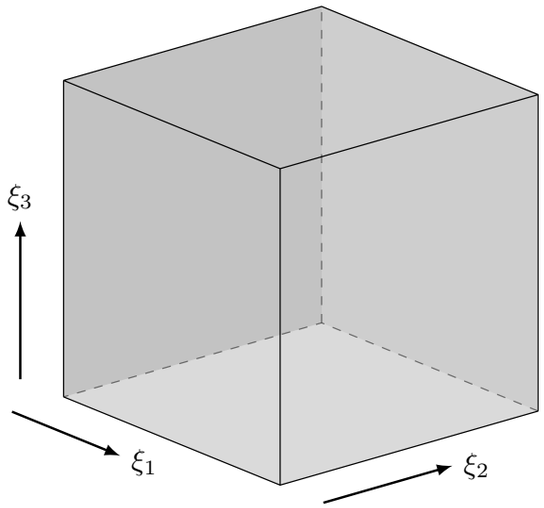
Isoparametric curvilinear elements

b = boundary

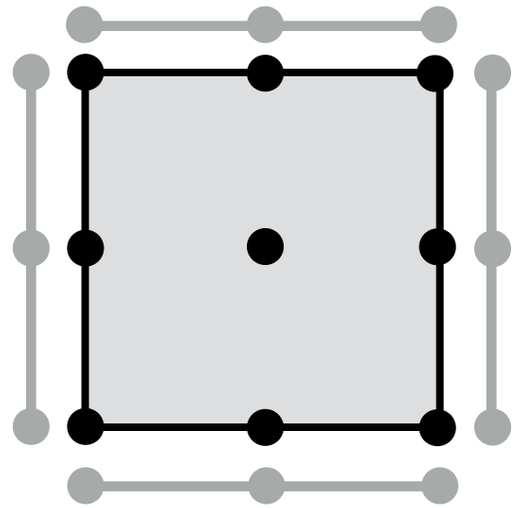
i = interior

HDG on an element

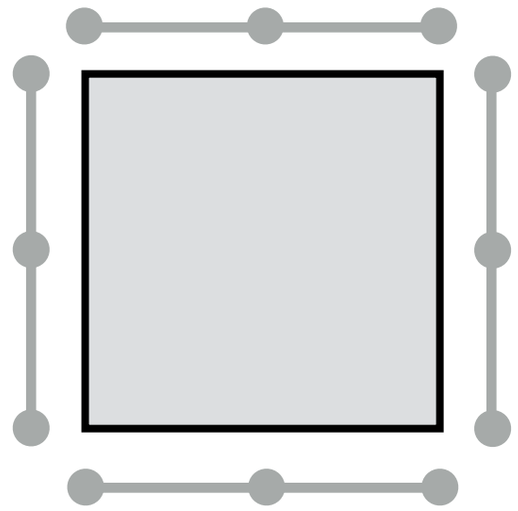




$$\boldsymbol{q} = \nabla u$$
$$u = \mathrm{div}(\boldsymbol{q}) \quad \text{on } \Omega$$



$u = \lambda$ on $\partial\Omega$
(static condensation)



A_λ local boundary system

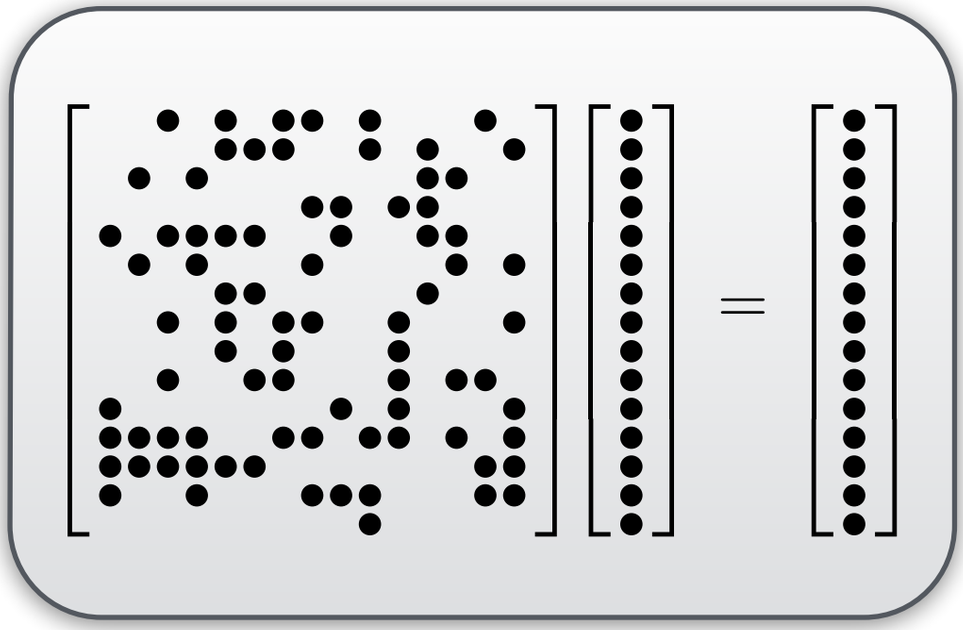
A_i interior solve

$A_{\lambda \to u}$ project λ onto u

Global matrices

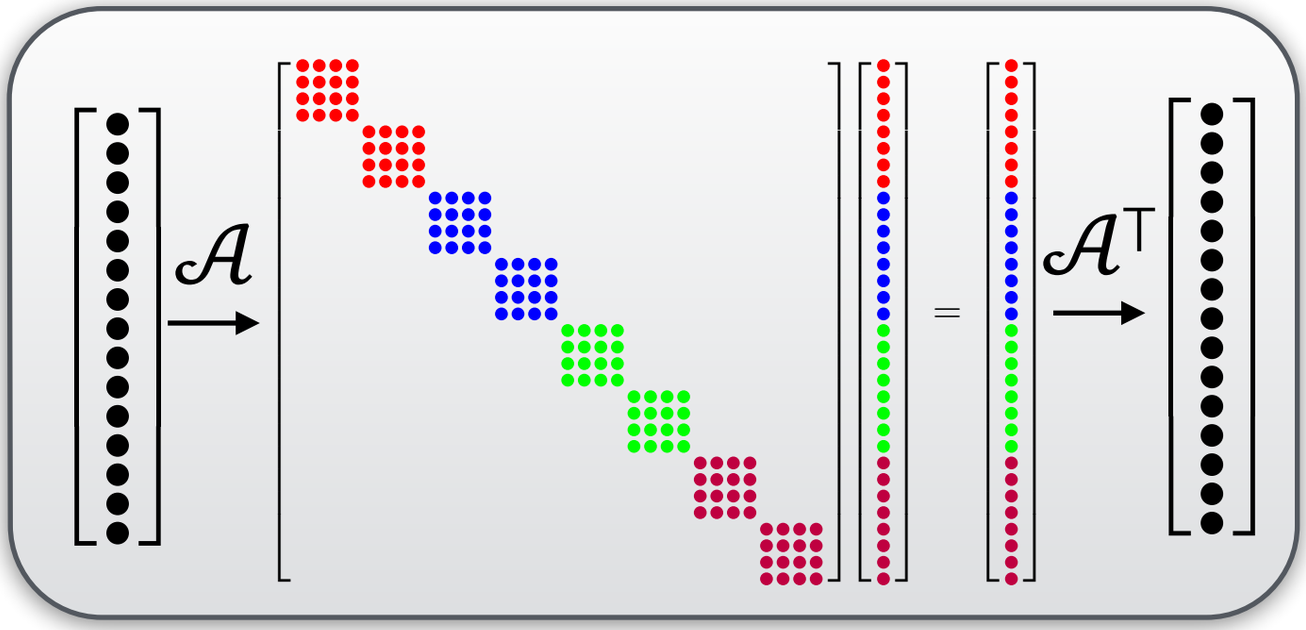
- Once we have *local* elemental matrices, need to form a *global* matrix for the entire system
- Create an *assembly map* $\mathcal{A}$ that establishes connections between elements based on mesh
- Can be seen as a global-to-local operator
- **Serial:** construct a global matrix and solve directly
- **Parallel:** form action of the global matrix using elemental matrices and assembly map

Global matrices



Global approach

serial + direct solve



Local approach
(gather-scatter)

parallel + PCG solve

Goals

- Follow on from previous study in 2D: **3D mixed elements**
- How does HDG fare against CG for:
  - Smaller problems (direct solve, **serial**)
  - Large problems (iterative solve, **parallel**)
- Performance for **elliptic** vs. **parabolic** solve
- Effect of **structured** vs. **unstructured** mesh
- In parallel, how does each method **scale**?
- How might these performance characteristics reflect in terms of complex systems?
- **Be fair** to both methods: same codebase, Nektar++

Models

$$\nabla^2 u + \lambda u = f$$

Helmholtz equation
(elliptic)

$$\frac{\partial u}{\partial t} = \nabla^2 u$$

Diffusion equation
(parabolic)

- Fundamental elliptic and parabolic equations with known non-polynomial exact solutions
- The building blocks of more complex operators
- e.g. incompressible Navier-Stokes: 1 x Poisson solve (pressure) + 3 x Helmholtz solve (velocity)

A priori expectations

- In both cases, we expect there might be a crossover point

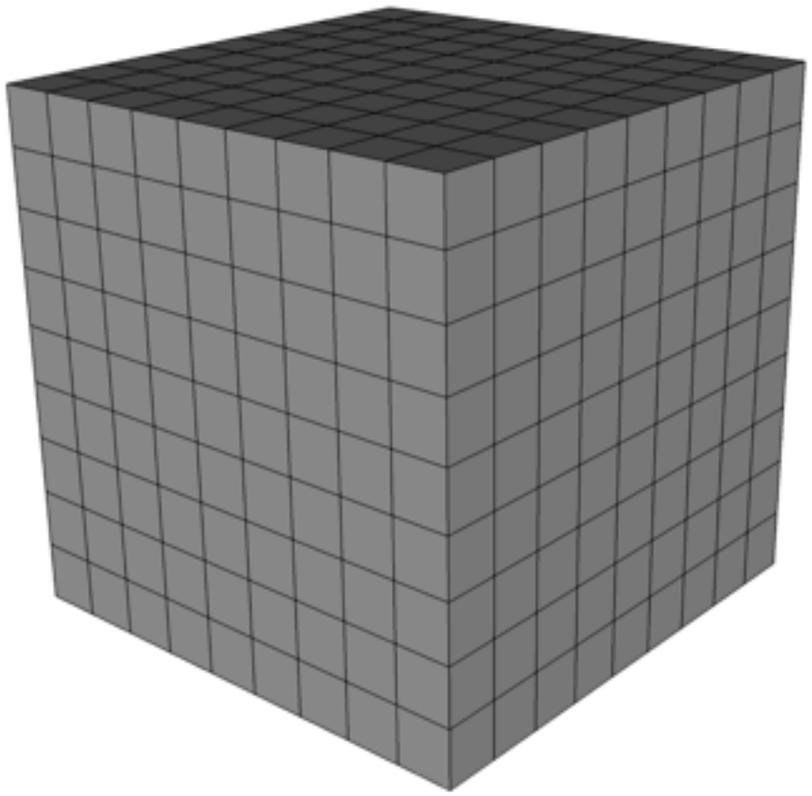
- **Serial:** factors include system bandwidth (as in 2D case), efficiency of direct solver

- **Parallel:** could depend on a number of factors: preconditioner/condition number, interconnect, mesh topology, communication patterns, ...

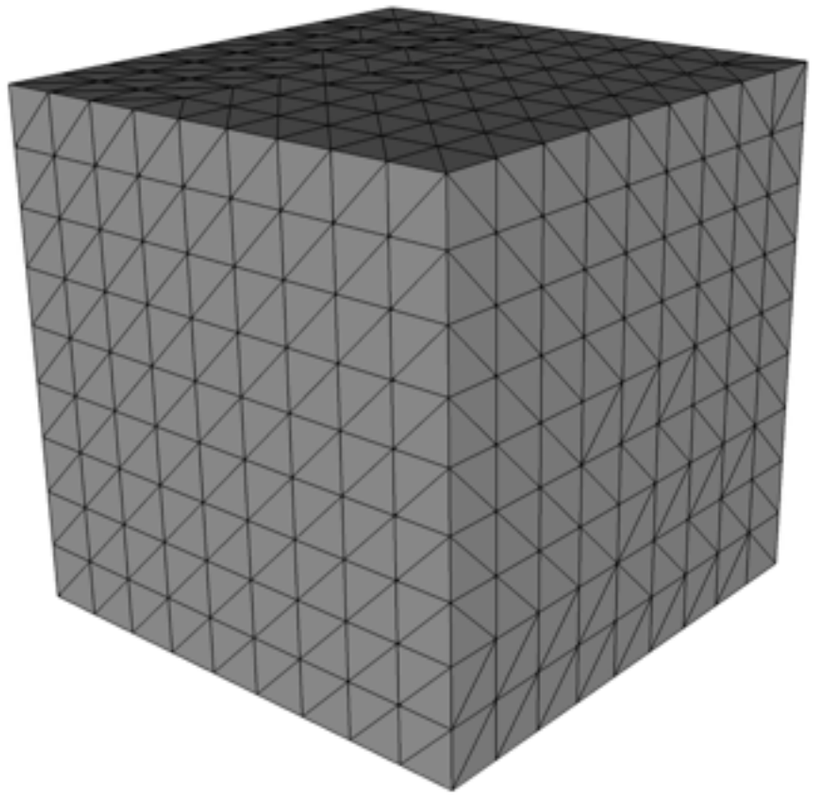
HDG postprocessing

- For some PDEs we can use an order *p* solution to construct a solution which is ***p* + 1** accurate
- Not clear whether this can hold for non-linear or time evolved systems
- However important to take into consideration when doing our timings

Serial performance



9^3 hex mesh



6 x 9^3 tet mesh

Helmholtz equation, Dirichlet boundary conditions

$\Omega = [-1,1]^3$, $u(\boldsymbol{x}) = \sin(5\pi x)\sin(5\pi y)\sin(5\pi z)$

Convergence validation

N_{CG}	Hexahedra		Tetrahedra		N_{HDG}
	L^2_{CG}	L^2_{HDG}	L^2_{CG}	L^2_{HDG}	
3	1.944e-02	8.301e-02	1.686e-01	1.323e-01	2
4	2.126e-03	6.046e-03	5.522e-02	4.367e-02	3
5	1.833e-04	3.449e-04	1.673e-02	1.373e-02	4
6	1.325e-05	2.504e-05	4.431e-03	3.981e-03	5
7	8.201e-07	1.460e-06	1.085e-03	9.973e-04	6
8	4.460e-08	7.863e-08	2.388e-04	2.283e-04	7
9	2.151e-09	3.828e-09	4.707e-05	4.562e-05	8

Serial performance

N_{CG}	DSC			DMSC			N_{HDG}
	CG	HDG	$\frac{\text{HDG}}{\text{CG}}$	CG	HDG	$\frac{\text{HDG}}{\text{CG}}$	
3	4.92	6.07	1.24	7.46	4.96	0.66	2
4	80.96	48.44	0.60	48.32	31.51	0.65	3
5	564.31	243.17	0.43	297.90	169.73	0.57	4
6	2661.04	947.12	0.36	1103.36	791.51	0.72	5
7	8765.75	2709.26	0.31	3397.99	2642.24	0.78	6
8	25450.47	6917.71	0.27	8789.20	7051.14	0.80	7
9	58228.23	16177.59	0.28	19580.75	17786.10	0.91	8

Hex cube mesh

DSC: direct static condensation

DMSC: direct **multi-level** static condensation

Serial performance

N_{CG}	DSC			DMSC			N_{HDG}
	CG	HDG	$\frac{\text{HDG}}{\text{CG}}$	CG	HDG	$\frac{\text{HDG}}{\text{CG}}$	
3	0.73	3.28	4.52	3.46	2.92	0.84	2
4	4.86	12.29	2.53	14.02	9.05	0.65	3
5	42.04	46.71	1.11	59.39	29.30	0.49	4
6	220.04	144.82	0.66	154.89	92.03	0.59	5
7	819.23	399.07	0.49	777.43	297.04	0.38	6
8	2282.90	885.17	0.39	2057.15	796.64	0.39	7
9	5747.78	1852.25	0.32	4081.03	1712.52	0.42	8

Tet cube mesh

Parallel performance

- Most operations (e.g. matrix construction) are trivially parallelisable: hard part is matrix solve

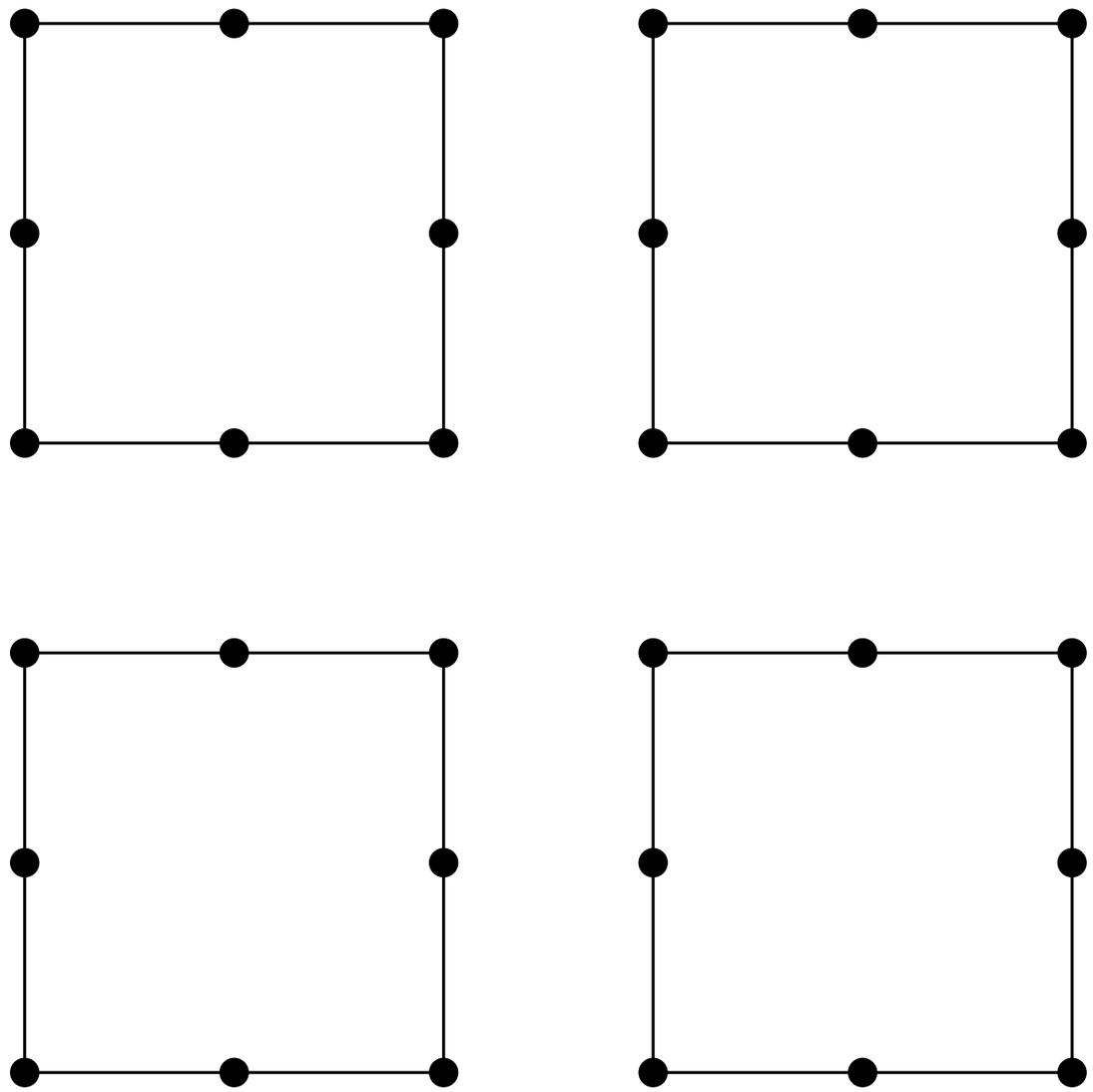
- Use preconditioned conjugate gradient (PCG)

- Most complex operation: gather-scatter required for matrix action: we use gslib (part of Nek5000)

- Measure both **weak** and **strong** scaling

- Tests on **HECToR**: 32 and 4,096 cores, **ARCHER**: 24 and 3,072 cores (1 to 128 nodes)

Communication patterns



HDG: elemental pairwise

CG: much more complex

Preconditioners and fair metrics

- CG and HDG systems will have different sizes and conditioning properties

- Therefore timings will depend (mostly) on:
  - # iterations / choice of preconditioner
  - cost of block matrix-vector multiply
  - interconnect and communication properties

- We can always build better preconditioners

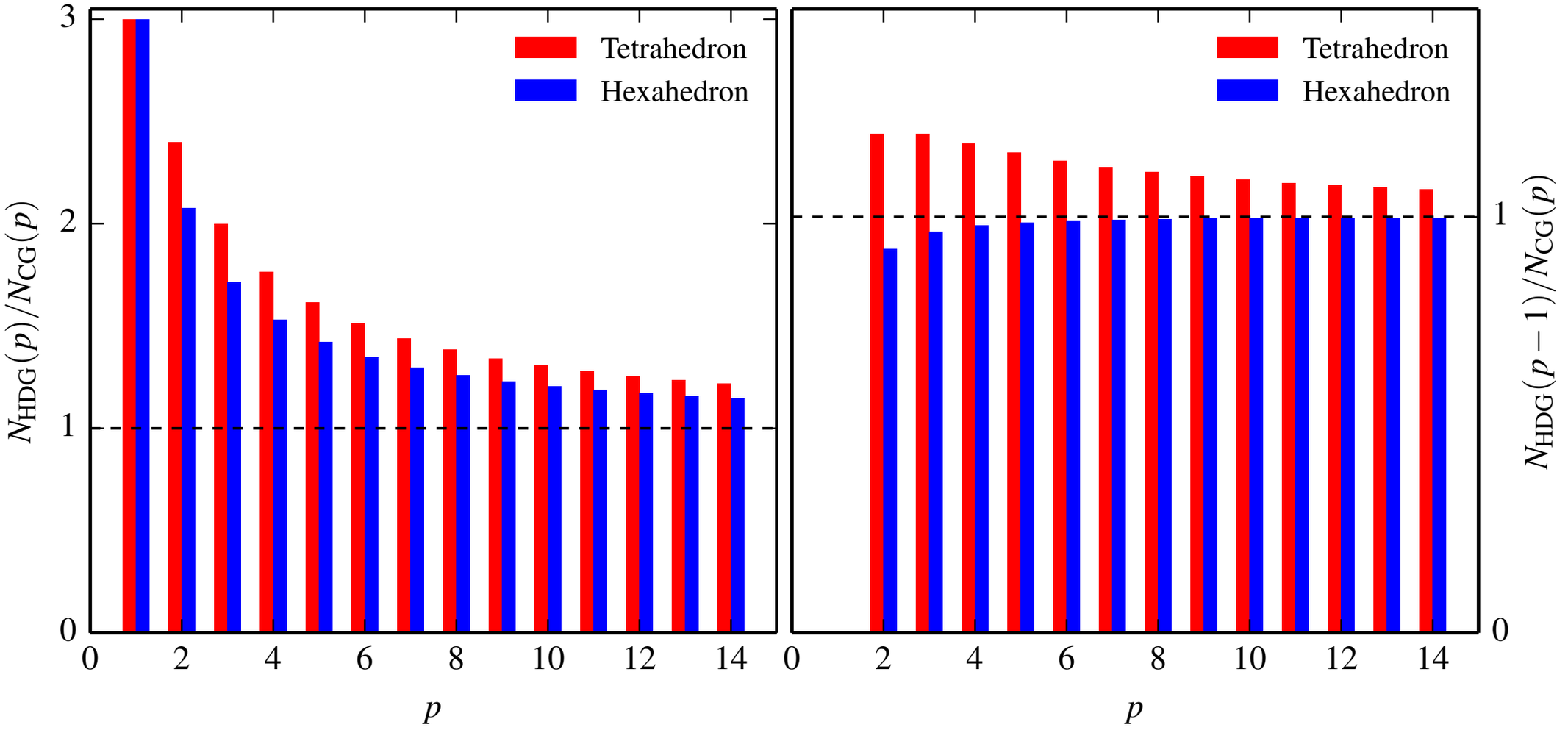
- **Fairness:** use Jacobi precon, measure time/iteration

Preconditioners: justification

N_{HDG}	HDG block	CG block	CG low energy	CG linear space	N_{CG}
2	130	51	37	29	3
3	157	98	53	35	4
4	203	150	62	38	5
5	223	204	69	40	6
6	245	252	74	41	7
7	258	295	78	42	8
8	279	345	82	44	9

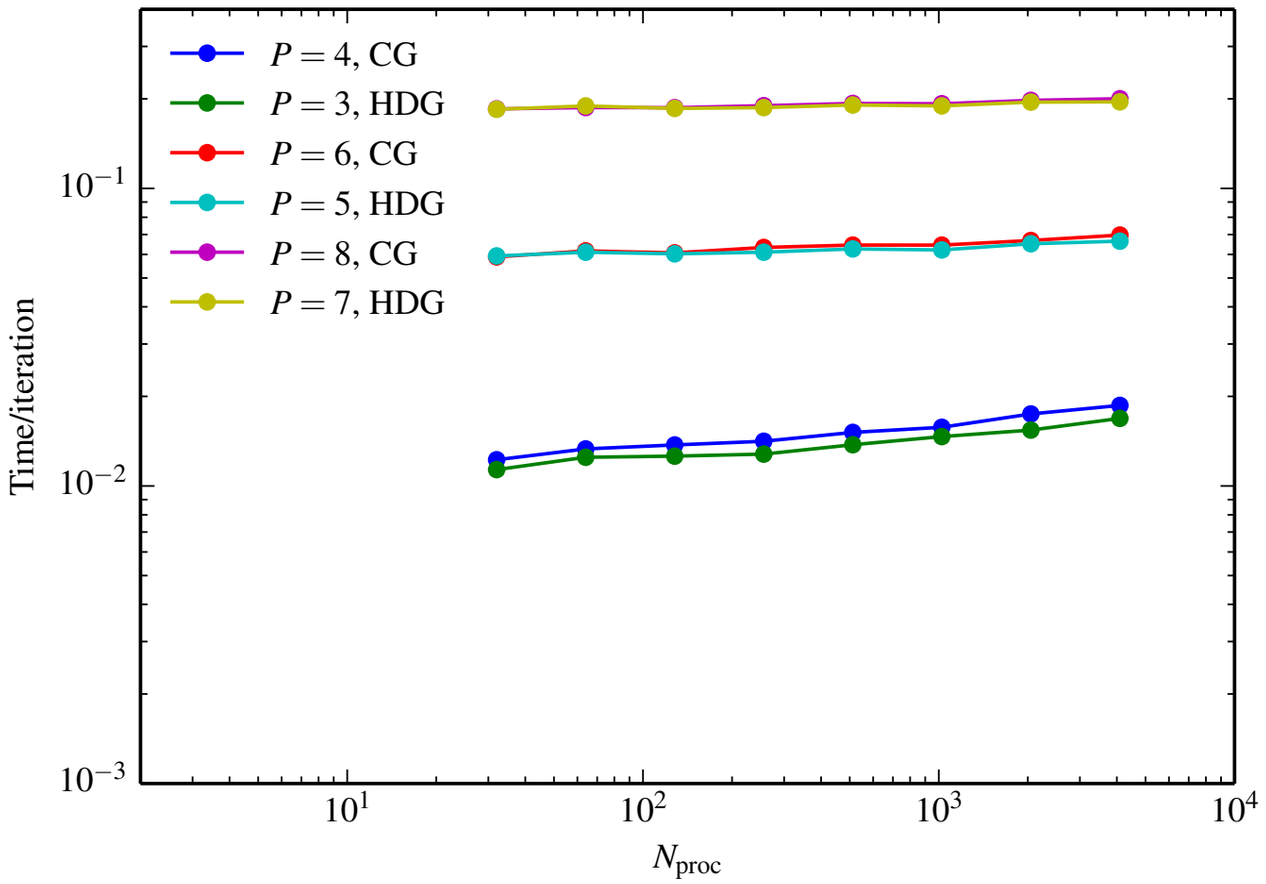
- Measure # iterations for unsteady diffusion equation
- **HDG/CG block:** invert block representing each trace face (HDG) or vertex/edge/face blocks (CG)
- **CG low energy:** basis transform
- **CG linear space:** LE + coarse space precon

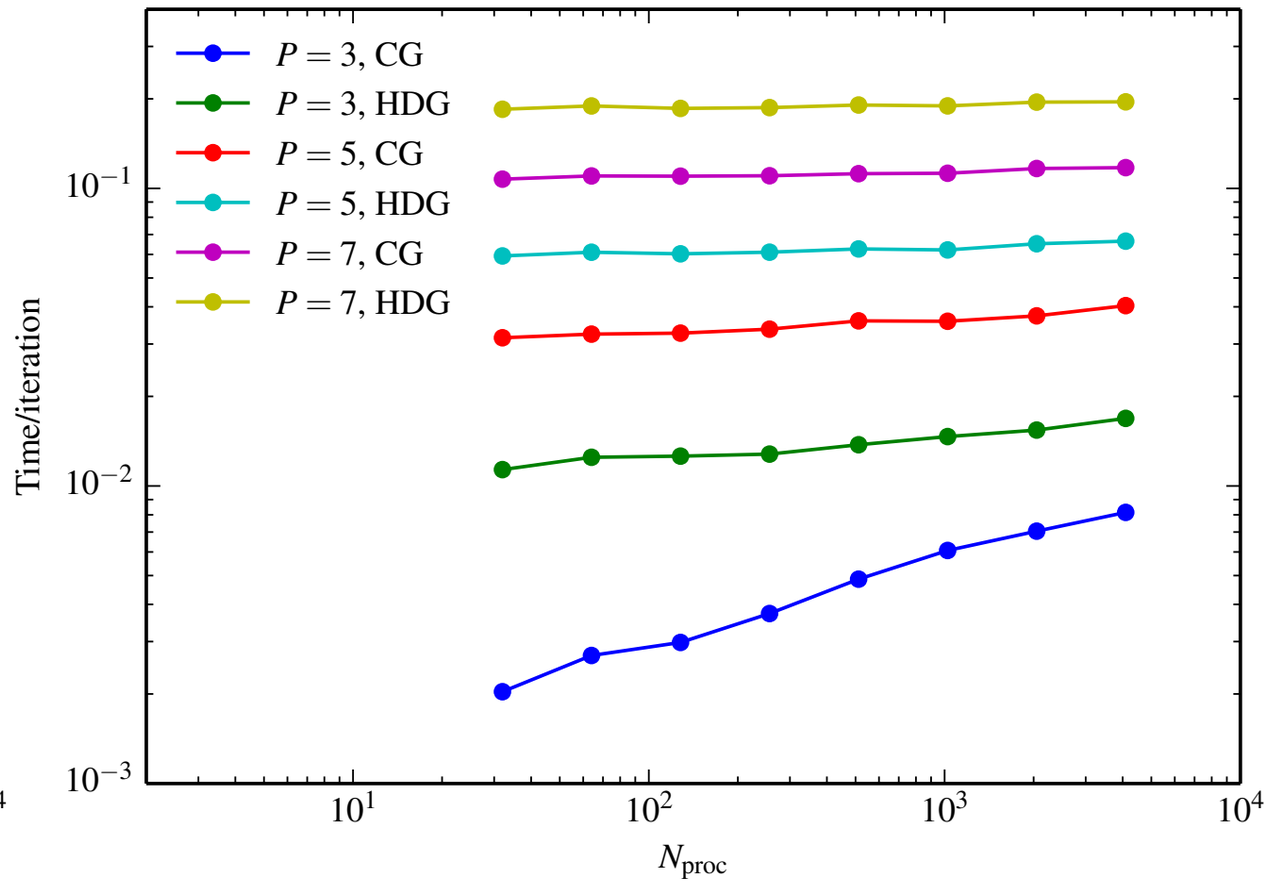
Matrix block sizes



HDG local blocks are far larger unless postprocessing taken into account

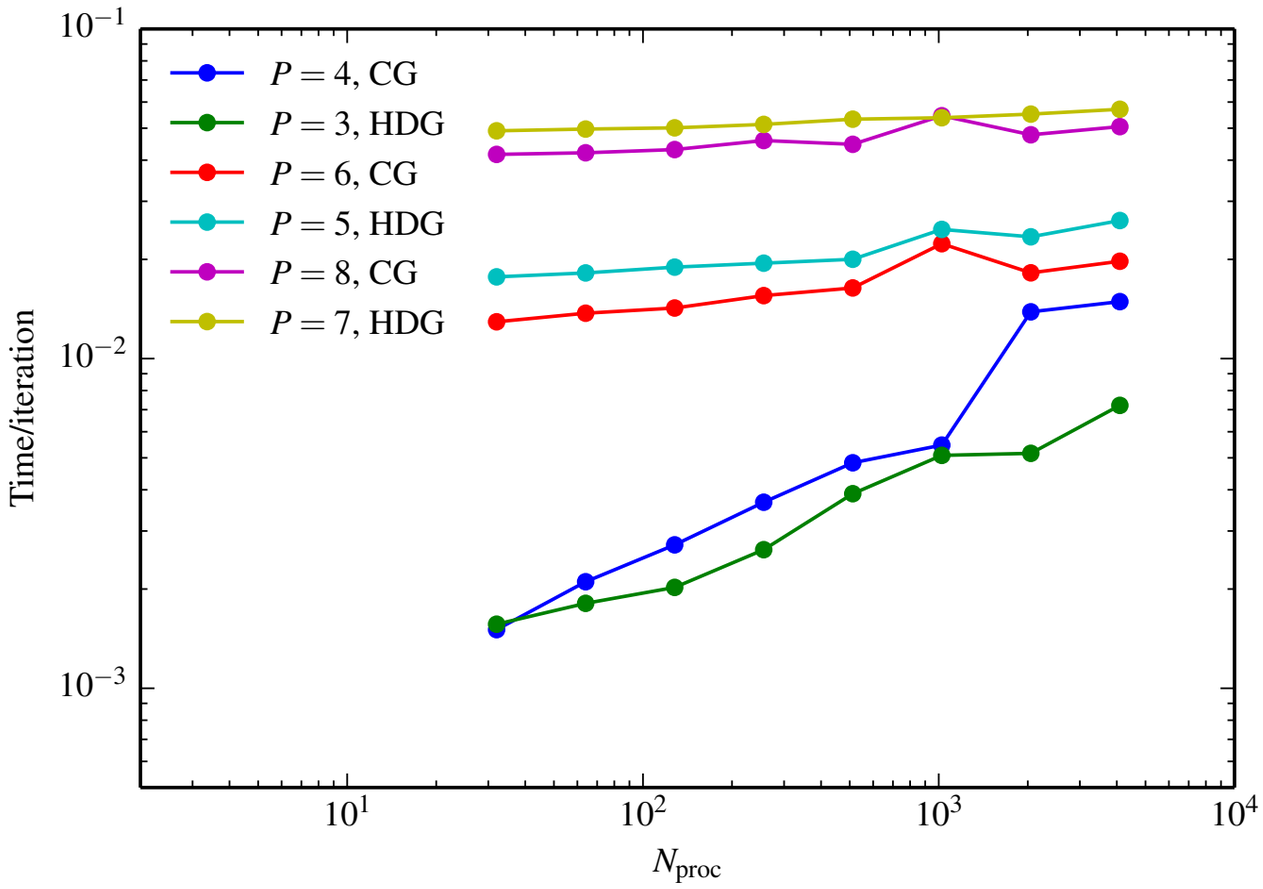
Weak scaling (hex)

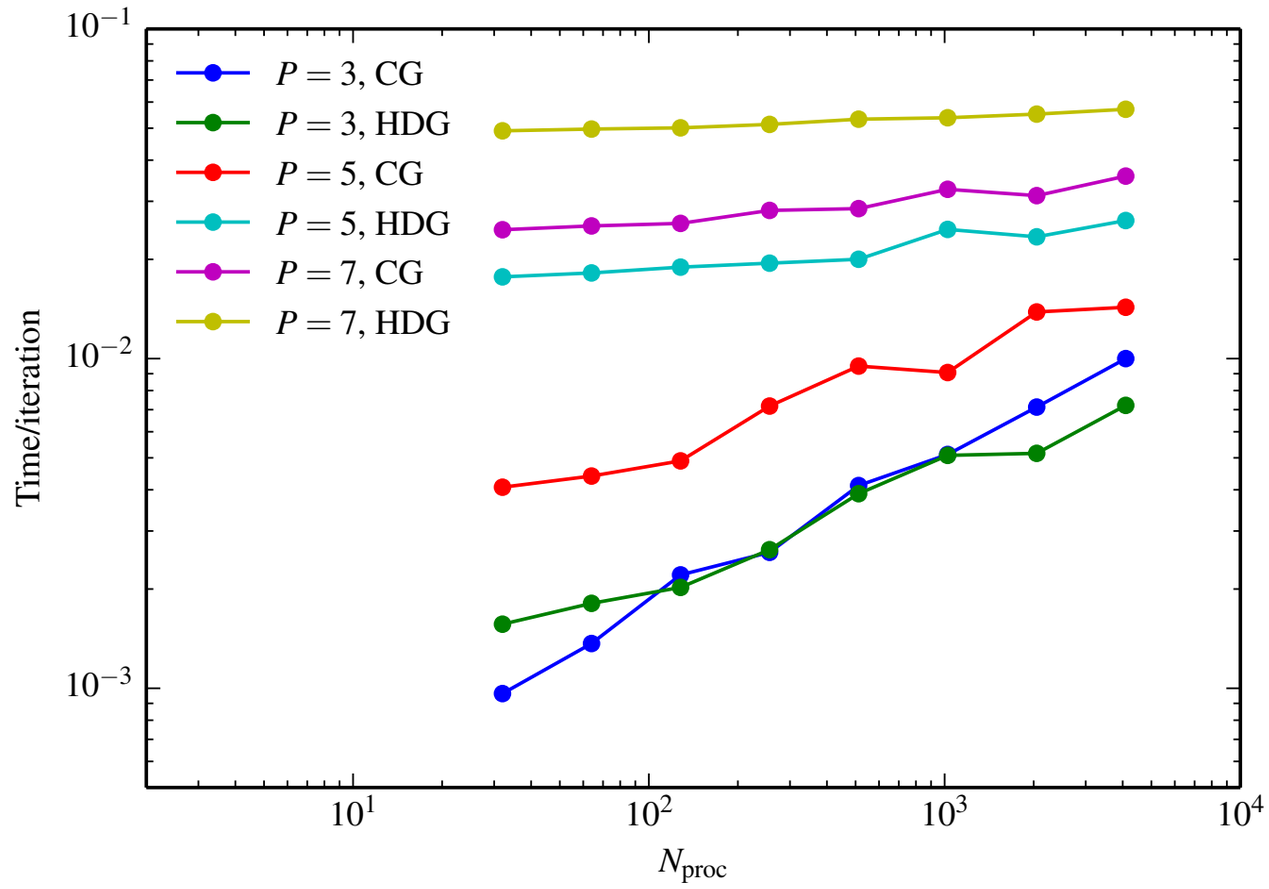




Cube geometry
3^3 hexes per core

Weak scaling (tet)

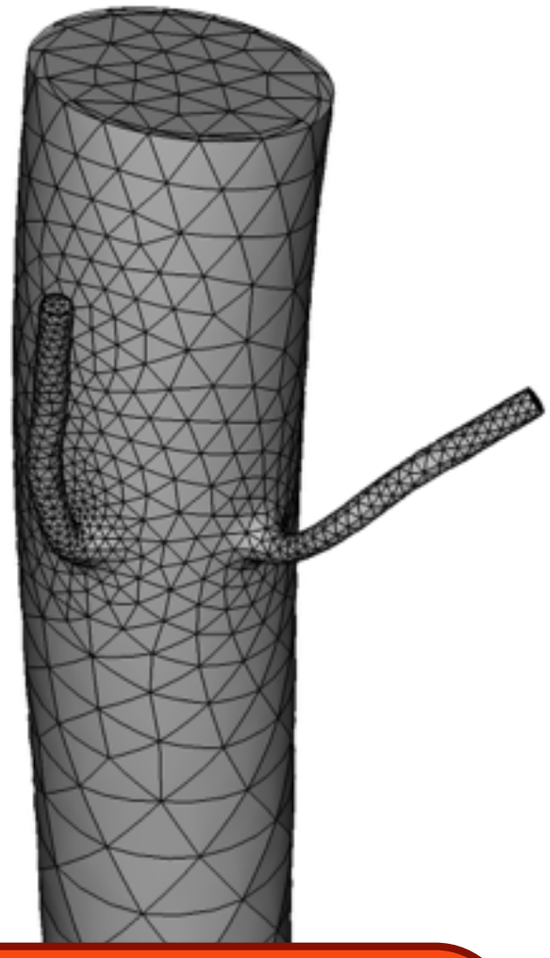


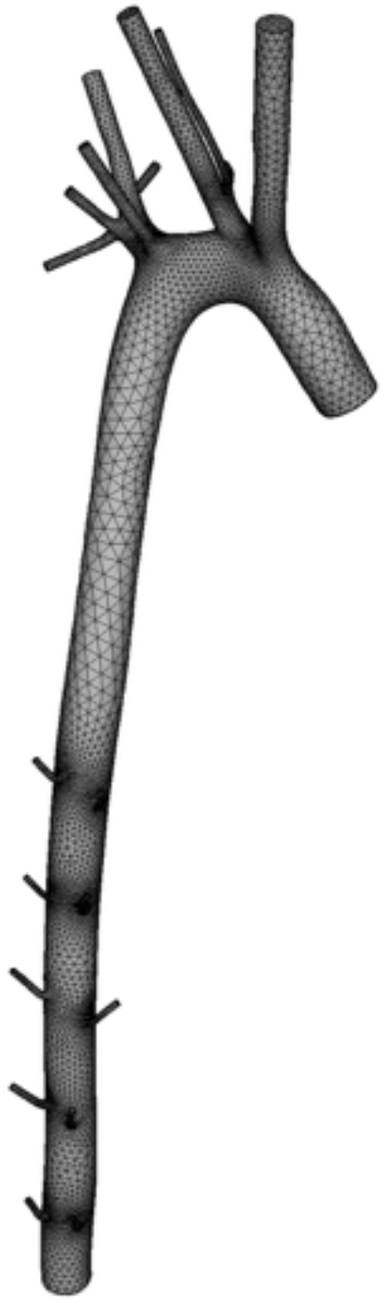


Cube geometry
6 x 2^3 tets per core

Strong scaling

intercostal pair
~8,500 tets
higher P

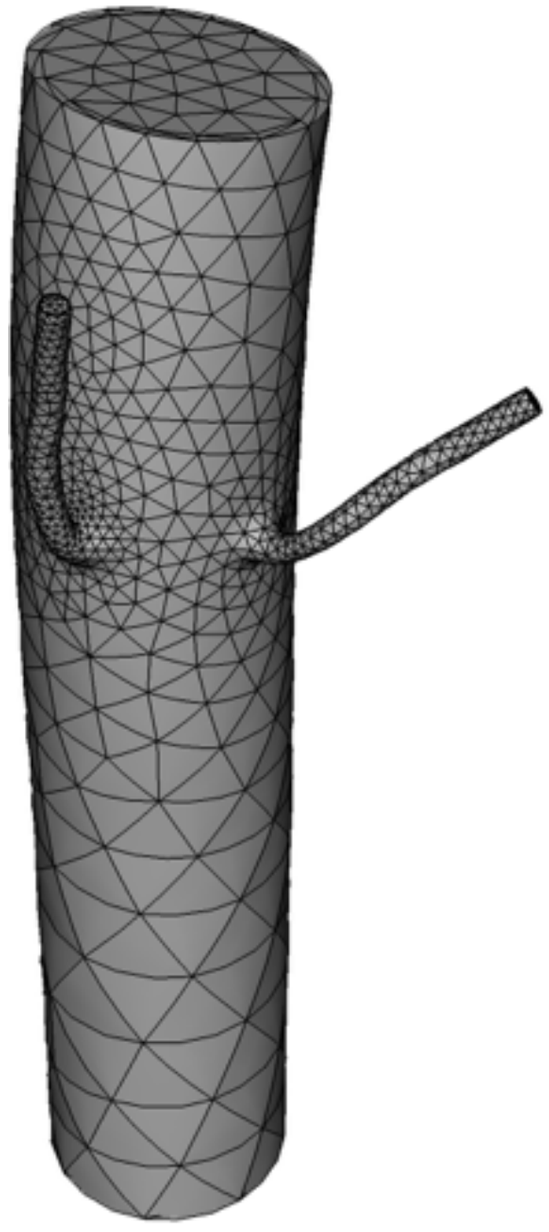


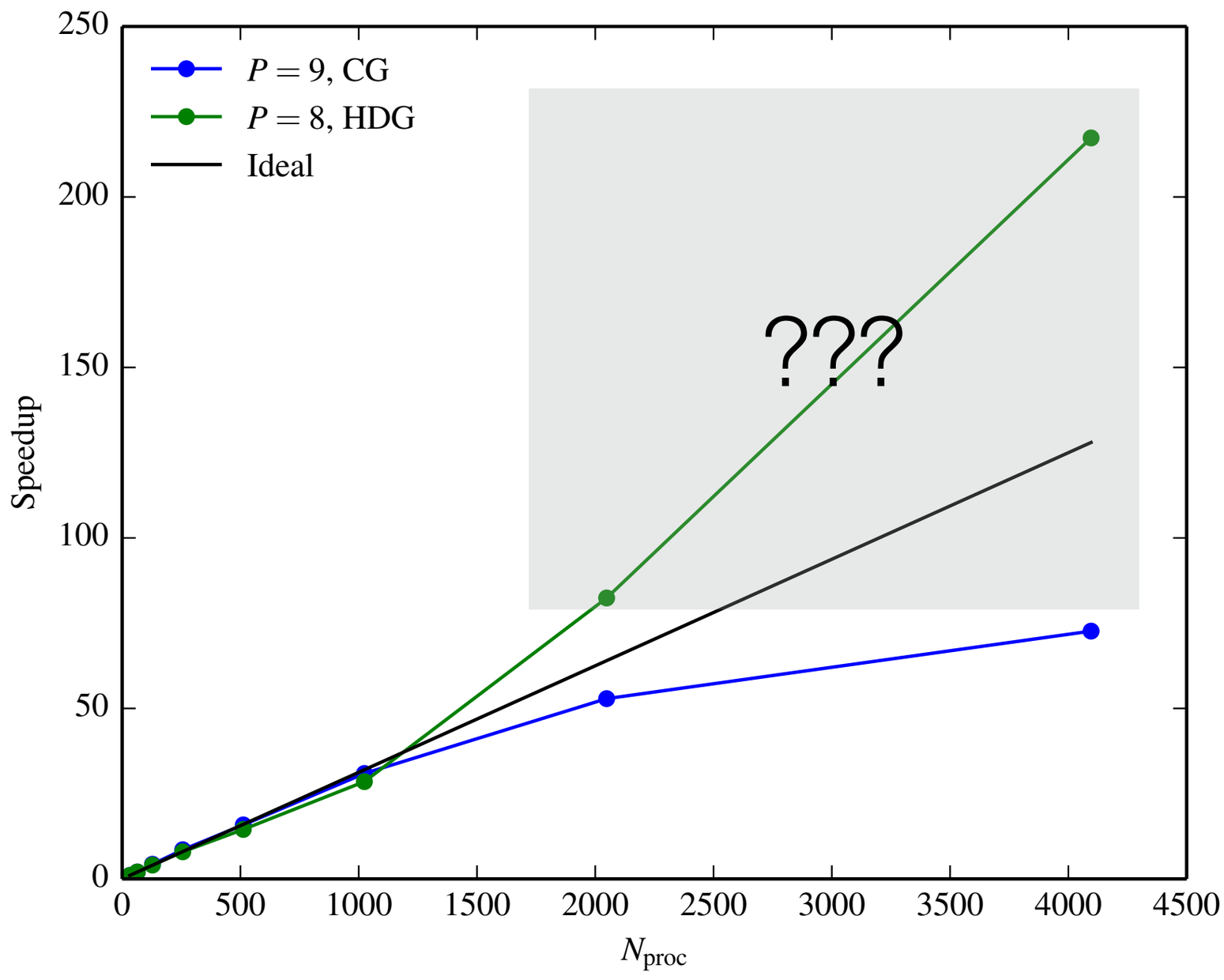


aortic arch
~150,000 tets
lower *P*

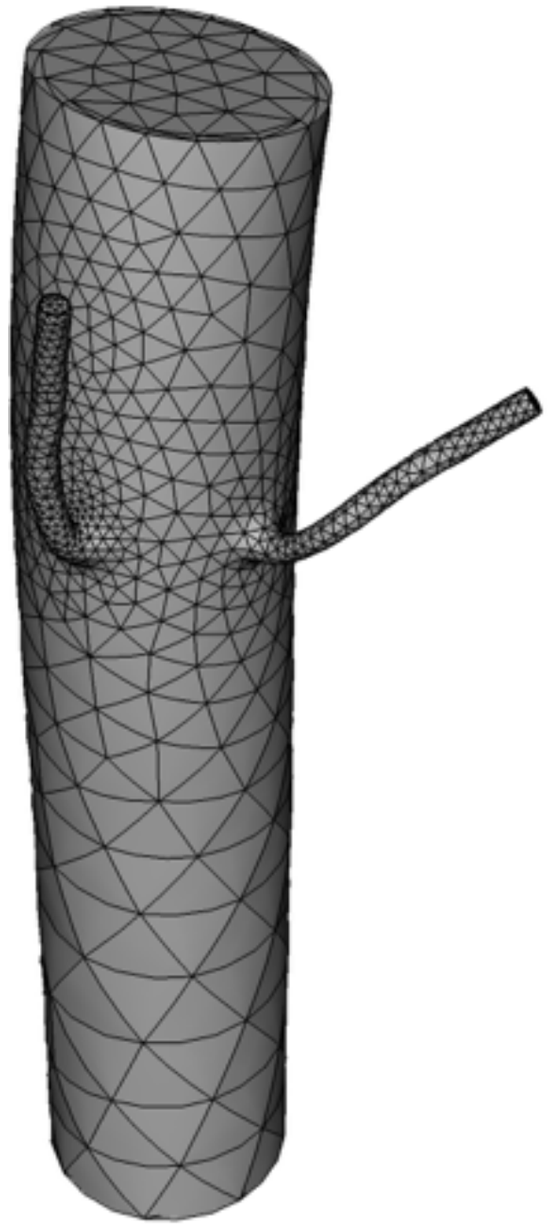
Mean vertex valency: 12
Max vertex valency: 44!

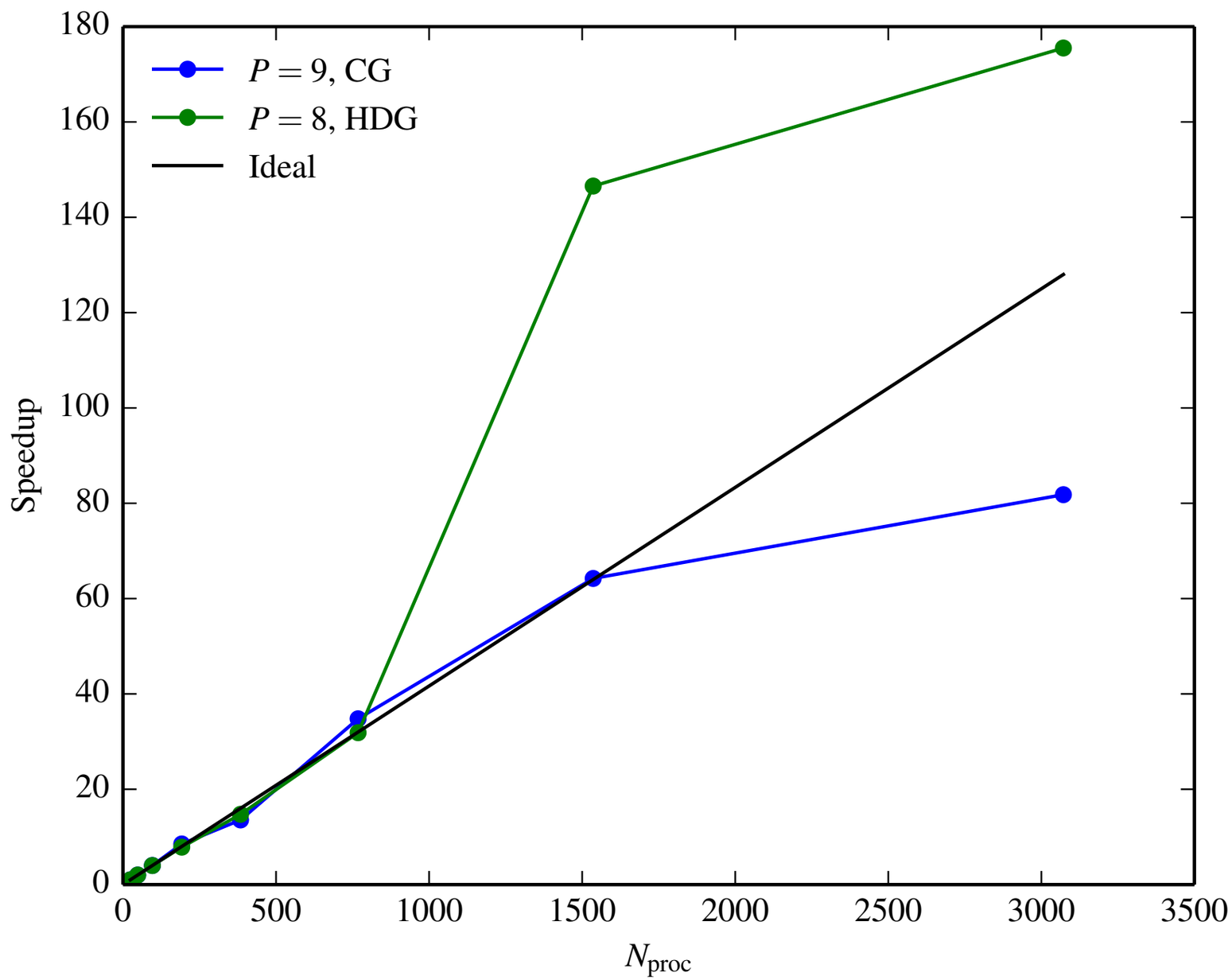
Strong scaling



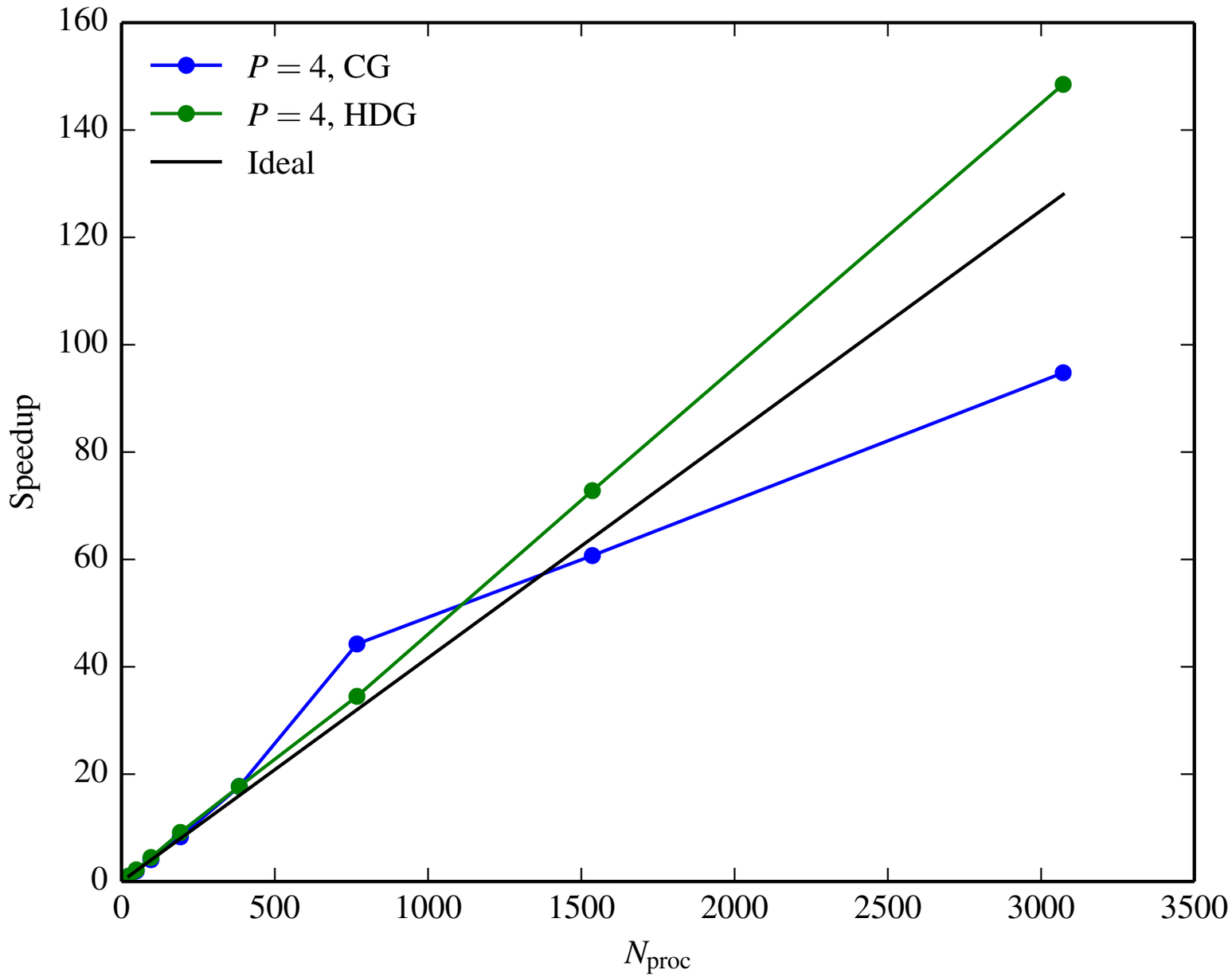


Strong scaling





Strong scaling



Summary

Direct solver + postprocessing

- HDG outperforms CG to various degrees depending on mesh + solver for elliptic problems
- Less pronounced for tetrahedral mesh

Iterative solver

- w/postprocessing: HDG can (barely) match CG
- Still need effective preconditioning strategy

Parallel

- HDG can outperform CG *on per-iteration basis* when communication cost is high
- However scaling is still pretty good for both methods

Nektar++ high-order framework

Framework for spectral(/hp) element method:

- Dimension independent, supports CG/DG/HDG
- Mixed elements (quads/tris, hexes, prisms, tets, pyramids) using hierarchical modal and classical nodal formulations
- Solvers for (in)compressible Navier-Stokes, advection-diffusion-reaction, shallow water equations, ...
- Parallelised with MPI, tested scaling up to ~10k cores

http://www.nektar.info/
nektar-users@imperial.ac.uk

Thanks for listening!

@davidmoxey

d.moxey@imperial.ac.uk