Using AmgX to Accelerate PETSc-Based CFD Codes

Pi-Yueh Chuang

pychuang@gwu.edu

George Washington University

Our Group

- Professor Lorena A. Barba
  http://lorenabarba.com/
- Projects:
  - **PyGBe** - *Python GPU code for Boundary elements*
    https://github.com/barbagroup/pygbe
  - **PetIBM** - *A PETSc-based Immersed Boundary Method code*
    https://github.com/barbagroup/PetIBM
  - **cuIBM** - *A GPU-based Immersed Boundary Method code*
    https://github.com/barbagroup/cuIBM
  - *… and so on*
    https://github.com/barbagroup

Our story

How we painlessly enable multi-GPU computing in PetIBM

PETSc

- **P**ortable, **E**xtensible **T**oolkit for **S**cientific **C**omputation
  https://www.mcs.anl.gov/petsc/index.html
- Argonne National Laboratory, since 1991
- Intended for large-scale parallel applications
- Parallel vectors, matrices, preconditioners, linear & nonlinear solvers, grid and mesh data structure … etc
- Hides MPI from application programmers
- C/C++, Fortran, Python

PetIBM

Taira & Colonius’ method (2007):

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \frac{1}{Re} \nabla^2 \mathbf{u} + \int_s \mathbf{f}(\boldsymbol{\xi}(s,t)) \delta(\boldsymbol{\xi} - \mathbf{x}) \, \mathrm{d}s,$$

$$\nabla \cdot \mathbf{u} = 0,$$

$$\mathbf{u}(\boldsymbol{\xi}(s,t)) = \int_{\mathbf{x}} \mathbf{u}(\mathbf{x}) \delta(\mathbf{x} - \boldsymbol{\xi}) \, \mathrm{d}\mathbf{x} = \mathbf{u}_B(\boldsymbol{\xi}(s,t)),$$

$$\begin{bmatrix} A & G & E^{\mathrm{T}} \\ G^{\mathrm{T}} & 0 & 0 \\ E & 0 & 0 \end{bmatrix} \begin{pmatrix} q^{n+1} \\ \phi \\ \tilde{f} \end{pmatrix} = \begin{pmatrix} r^n \\ 0 \\ u_B^{n+1} \end{pmatrix} + \begin{pmatrix} bc_1 \\ -bc_2 \\ 0 \end{pmatrix}$$

[†] K. Taira and T. Colonius, "The immersed boundary method: A projection approach", Journal of Computational Physics, vol. 225, no. 2, pp. 2118-2137, 2007.

PetIBM

$$Q \equiv [G, \;\; E^{\mathrm{T}}], \quad \lambda \equiv \begin{pmatrix} \phi \\ \tilde{f} \end{pmatrix}, \quad r_1 \equiv r^n + bc_1, \quad r_2 \equiv \begin{pmatrix} -bc_2 \\ u_B^{n+1} \end{pmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} A & 0 \\ Q^{\mathrm{T}} & -Q^{\mathrm{T}}B^N Q \end{bmatrix} \begin{bmatrix} I & B^N Q \\ 0 & I \end{bmatrix} \begin{pmatrix} q^{n+1} \\ \lambda \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \end{pmatrix} + \begin{pmatrix} -\frac{\Delta t^N}{2^N} (LM^{-1})^N Q\lambda \\ 0 \end{pmatrix}$$

$$\Downarrow$$

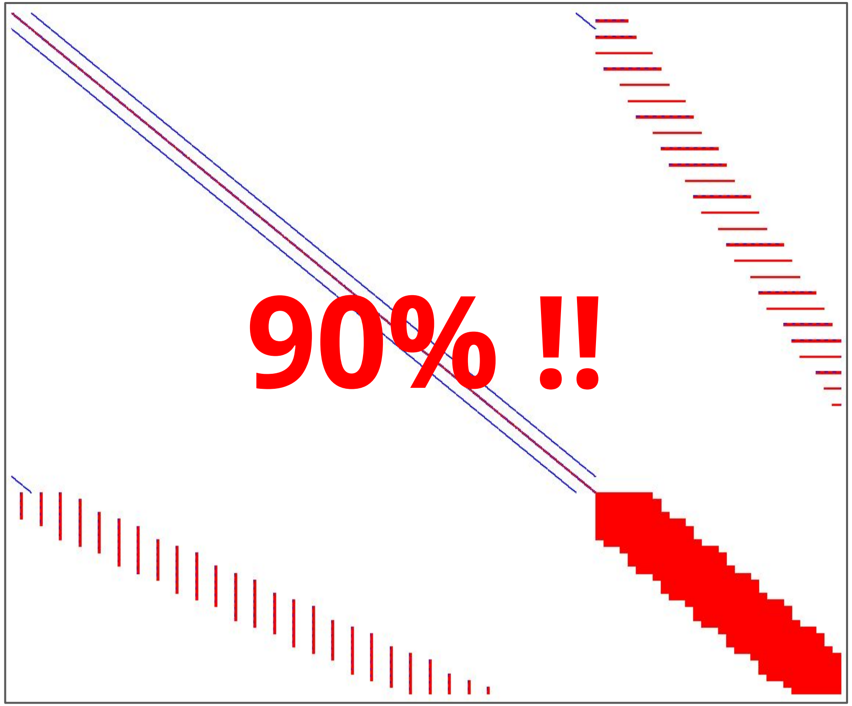
$$\begin{aligned} &Aq^* = r_1 && \text{(Solve for intermediate velocity)}, \\ &\boxed{Q^{\mathrm{T}} B^N Q\lambda = Q^{\mathrm{T}} q^* - r_2 \qquad \text{(Solve the modified Poisson equation)},} \\ &q^{n+1} = q^* - B^N Q\lambda && \text{(Projection step)}. \end{aligned}$$

Solving modified Poisson systems is tough

Possible solutions:

Rewrite the whole program for multi-GPU capability, or

<u>**Tackle the expensive part !**</u>



AmgX

- **Developed and supported by NVIDIA**
  https://developer.nvidia.com/amgx
- **Krylov methods:**
  - CG, GMRES, BiCGStab, … etc
- **Multigrid preconditioners:**
  - Classical AMG (largely based on Hypre BoomerAMG)
  - Unsmoothed aggregation AMG
- **Multiple GPUs on single node / multiple nodes:**
  - MPI (OpenMPI) / MPI Direct
  - Single MPI rank ⇔ single GPU
  - Multiple MPI ranks ⇔ single GPU

AmgX Wrapper

A wrapper for quickly coupling AmgX into existing PETSc-based software

AmgX Wrapper: Make Life Easier

Declare and initialize a solver

```
AmgXWrapper    solver;
solver.initialize(communicator & config file);
```

Bind the matrix A

```
solver.setA(A);
```

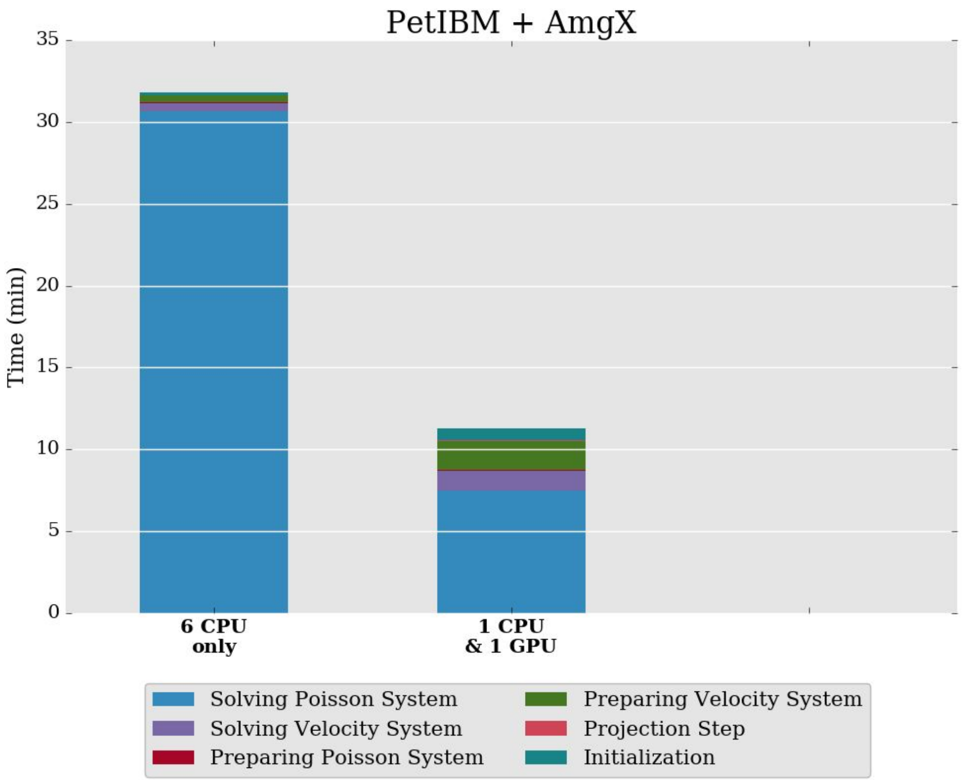
In time-marching loop

```
solver.solve(x, rhs);
```

Finalization

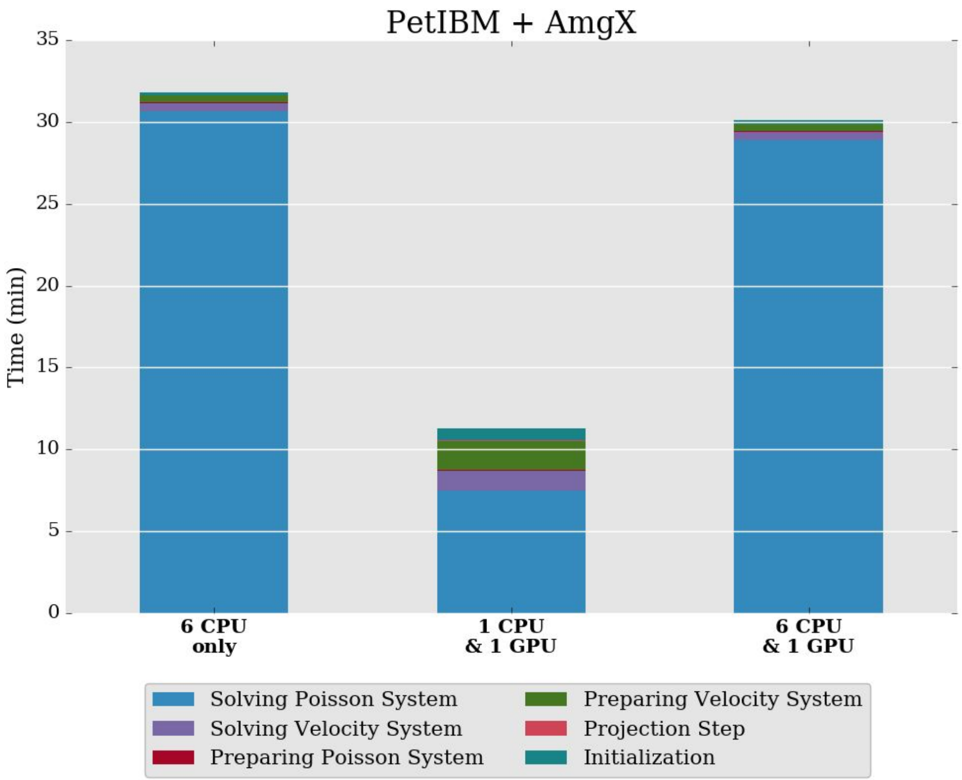
```
solver.finalize();
```

Example: 2D Cylinder Flow, Re=40



- **Mesh Size: 2.25M**
- **1 NVIDIA K40c**
- **Velocity:**
  - **PETSc KSP - CG**
  - **Block Jacobi**
- **Modified Poisson**
  - **AmgX - CG**
  - **Aggregation AMG**

Example: 2D Cylinder Flow, Re=40



- Mesh Size: 2.25M
- 1 NVIDIA K40c
- Velocity:
  - PETSc KSP - CG
  - Block Jacobi
- Modified Poisson
  - AmgX - CG
  - Aggregation AMG

Solution

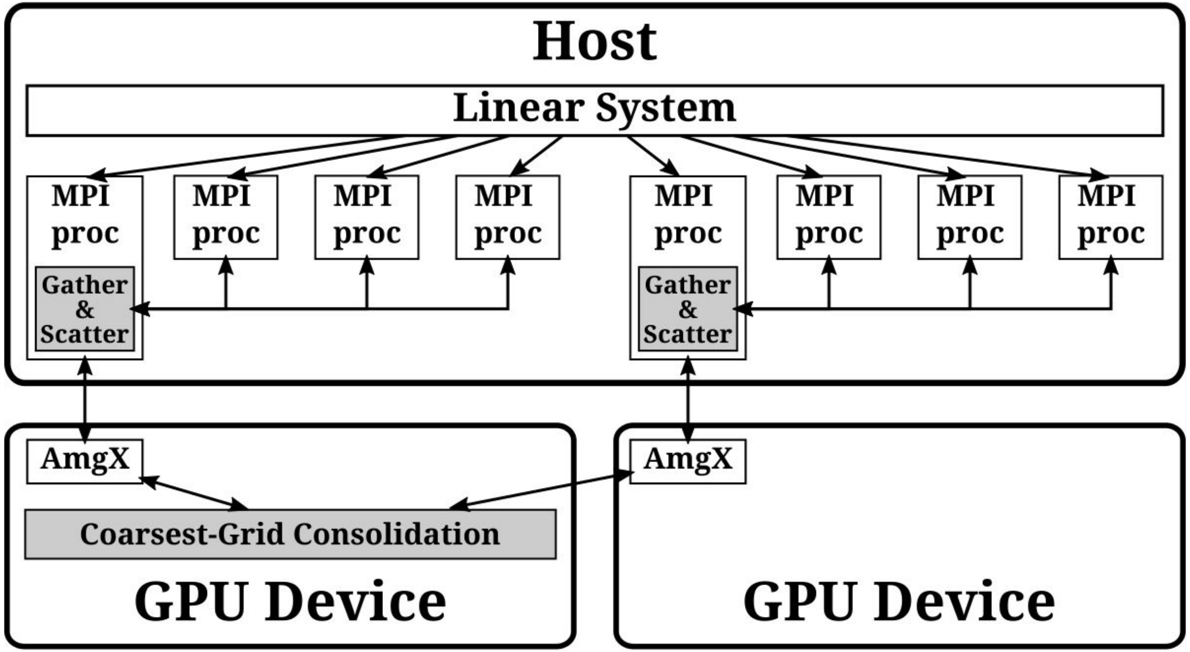
Assure there's always only one subdomain solver on every GPU

We want to make using AmgX easy

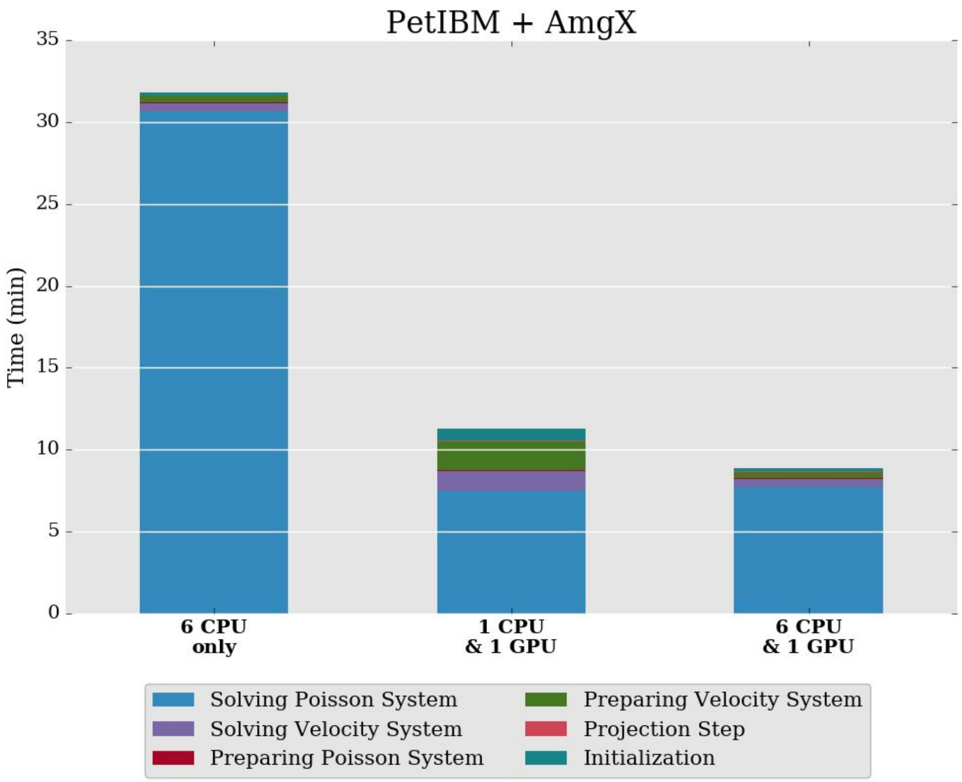
The solution should be implemented in the wrapper, not in PetIBM

The wrapper makes things easier

No need to modify original codes in PETSc-based applications

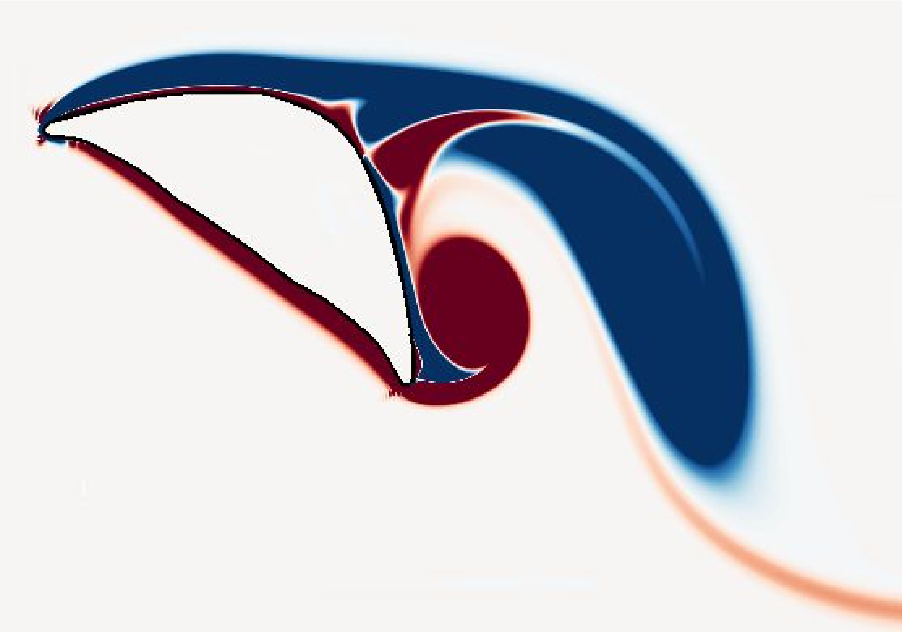


Back to Example: 2D Cylinder Flow, Re=40



- Mesh Size: 2.25M
- 1 NVIDIA K40c
- Velocity:
  - PETSc KSP - CG
  - Block Jacobi
- Modified Poisson
  - AmgX - CG
  - Aggregation AMG
- AmgX Wrapper

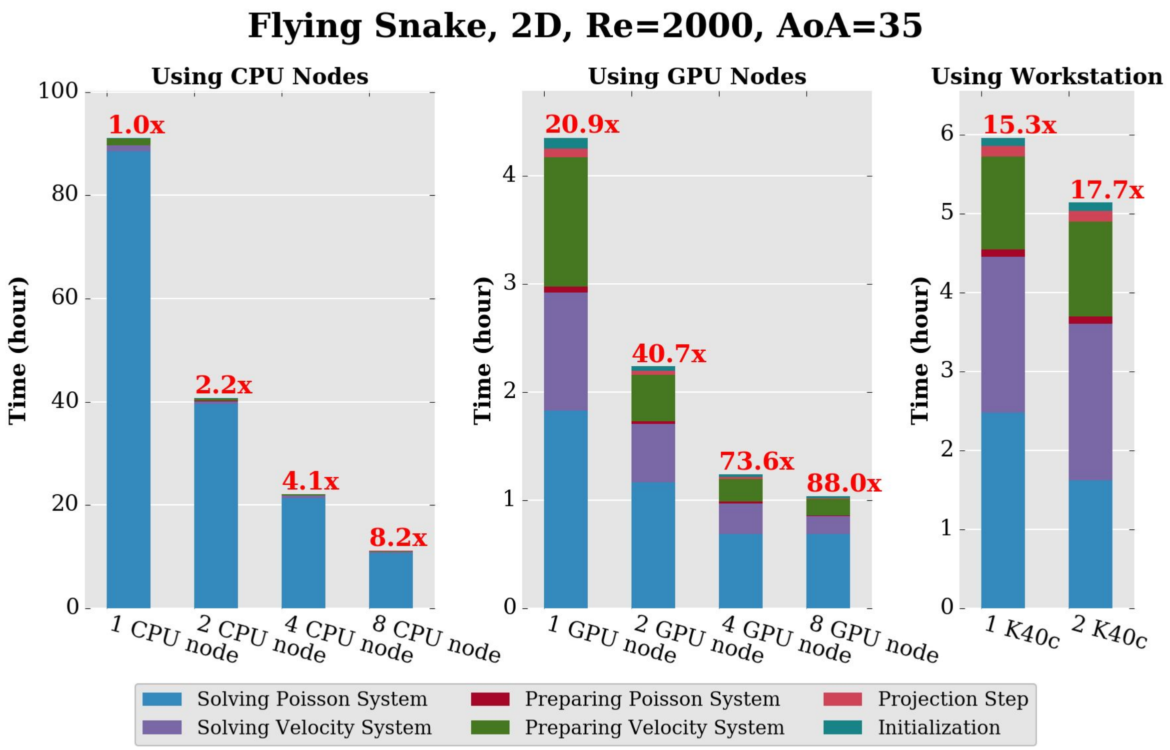
Benchmark: Flying Snakes



- **Anush Krishnan et. al. (2014)†**
  - **Re=2000**
  - **AoA=35**
  - **Mesh Size: 2.9M**

†A. Krishnan, J. Socha, P. Vlachos and L. Barba, "Lift and wakes of flying snakes", *Physics of Fluids*, vol. 26, no. 3, p. 031901, 2014.

Example: Flying Snakes



- **Per CPU node:**
  - 2 Intel E5-2620 (12 cores)
- **Per GPU node:**
  - 1 CPU node (12 cores)
  - 2 NVIDIA K20
- **Workstation:**
  - Intel i7-5930K (6 cores)
  - 1 or 2 K40c

Time is money

Potential Savings and Benefits: Hardware

For our application, enabling multi-GPU computing reduces

- costs on extra hardware,
  - motherboards, memory, hard drives, cooling systems, power supplies, Infiniband switches, physical space… etc.
- works and human resources on managing clusters,
- socket to socket communications
- potential runtime crash due to single node failure or network failure, and
- time spent on queue at any HPC centers

Potential saving on cloud HPC service

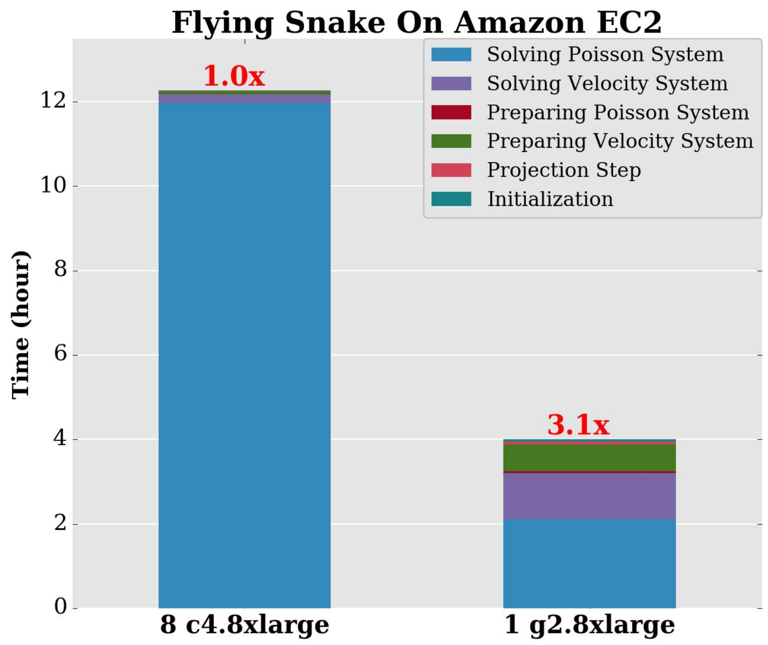
Running GPU-enabled CFD applications with cloud HPC service may save a lot

Potential Saving and Benefits: Cloud HPC Service

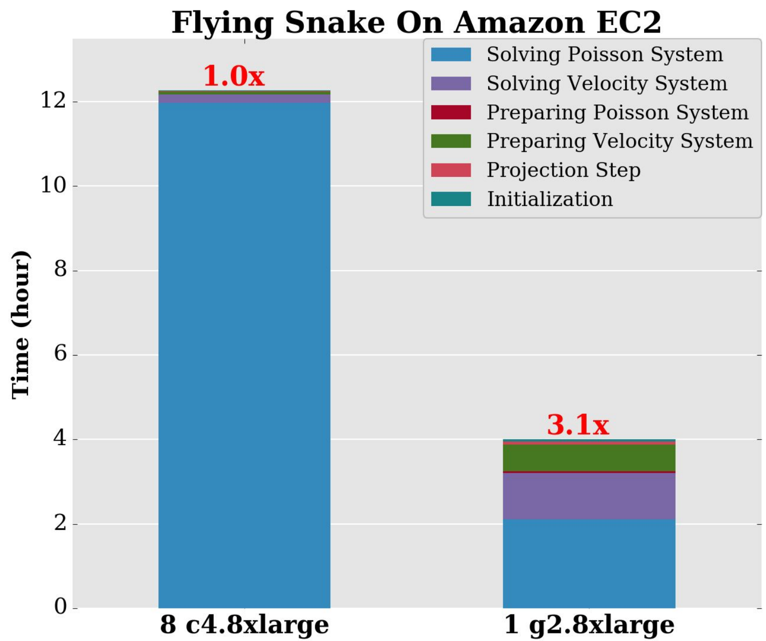
Reduce execution time and needed nodes. For example, on Amazon EC2:

- **GPU nodes - g2.8xlarge:**
  - 32 vCPU (Intel E5-2670) + 4 GPUs (Kepler GK104)
  - Official Price: $2.6 / hr
  - Possible Lower Price (Spot Instances): < $0.75 / hr
- **CPU nodes - c4.8xlarge**
  - 36 vCPU (Intel E5-2663)
  - Official Price: $1.675 / hr
  - Possible Lower Price (Spot Instances): < $0.6 / hr

Potential Saving and Benefits: Cloud HPC Service



Potential Saving and Benefits: Cloud HPC Service



- **CPU:**

  12.5 hr × $1.675 / hr × 8 nodes = **$167.5**

- **GPU:**

  4 hr × $2.6 / hr × 1 node = **$10.4**

Conclusion

- **AmgX and our wrapper**
  - https://developer.nvidia.com/amgx
  - https://github.com/barbagroup/AmgXWrapper
- **PetIBM with AmgX enabled:**
  - https://github.com/barbagroup/PetIBM/tree/AmgXSolvers
- **Speed up in a real application: flying snake**
- **Time is money**
- **Complete technical paper:**
  - http://goo.gl/0DM1Vw

Thanks!

Acknowledgement:

Dr. Joe Eaton, NVIDIA

Technical paper:

http://goo.gl/0DM1Vw

Contact us:

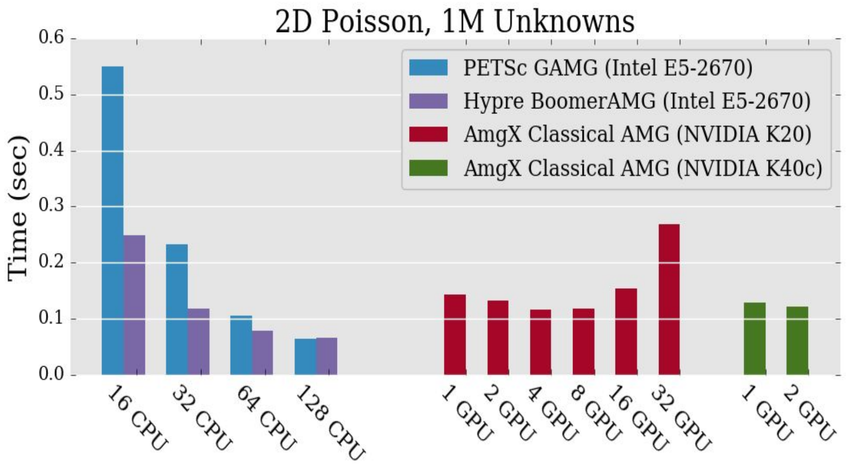
Website: http://lorenabarba.com/

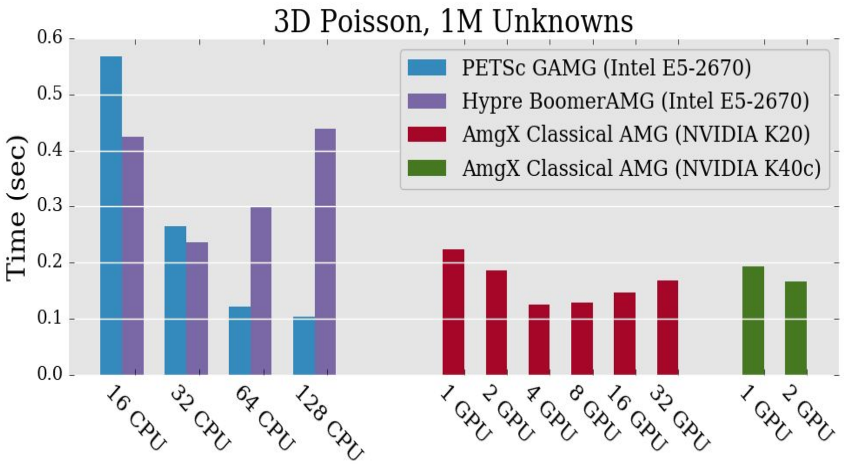
GitHub: https://github.com/barbagroup/

Q & A

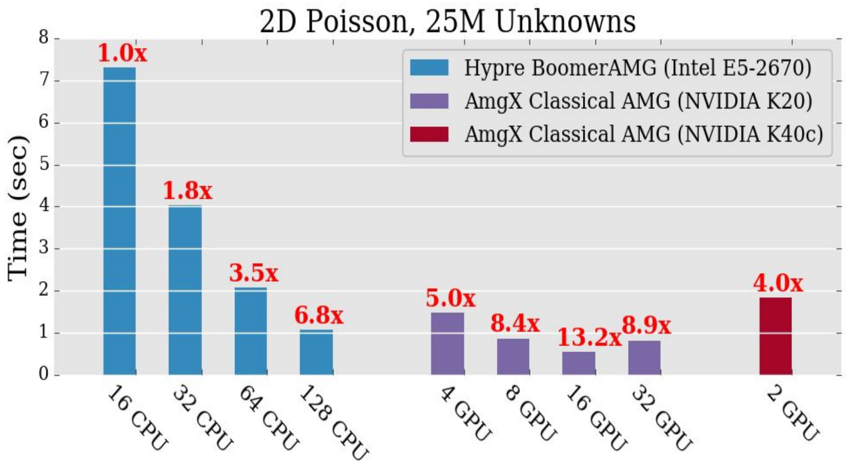
Extra Slides

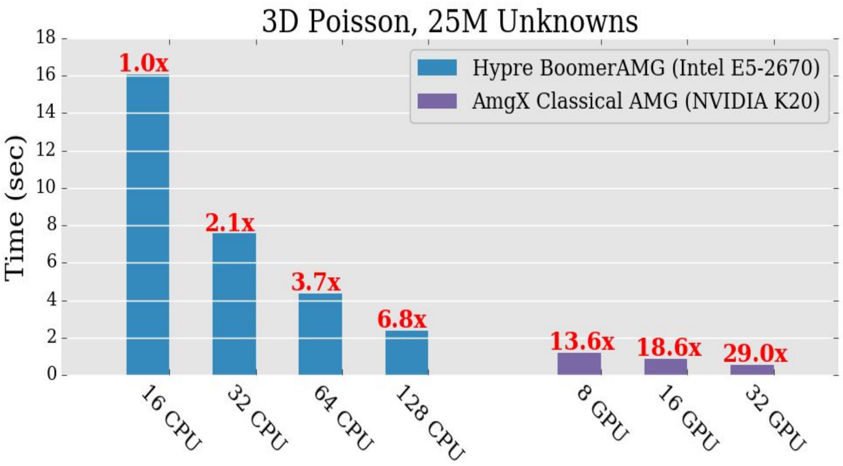
Example: Small-Size Problems



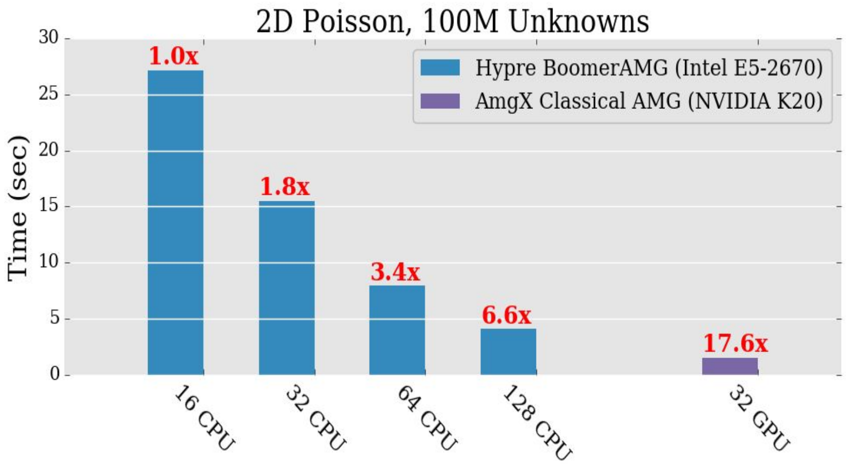


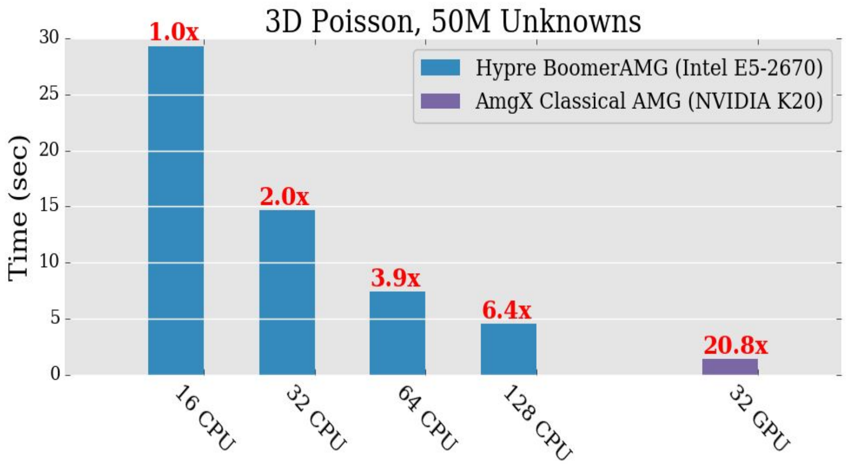
Example: Medium-Size Problems



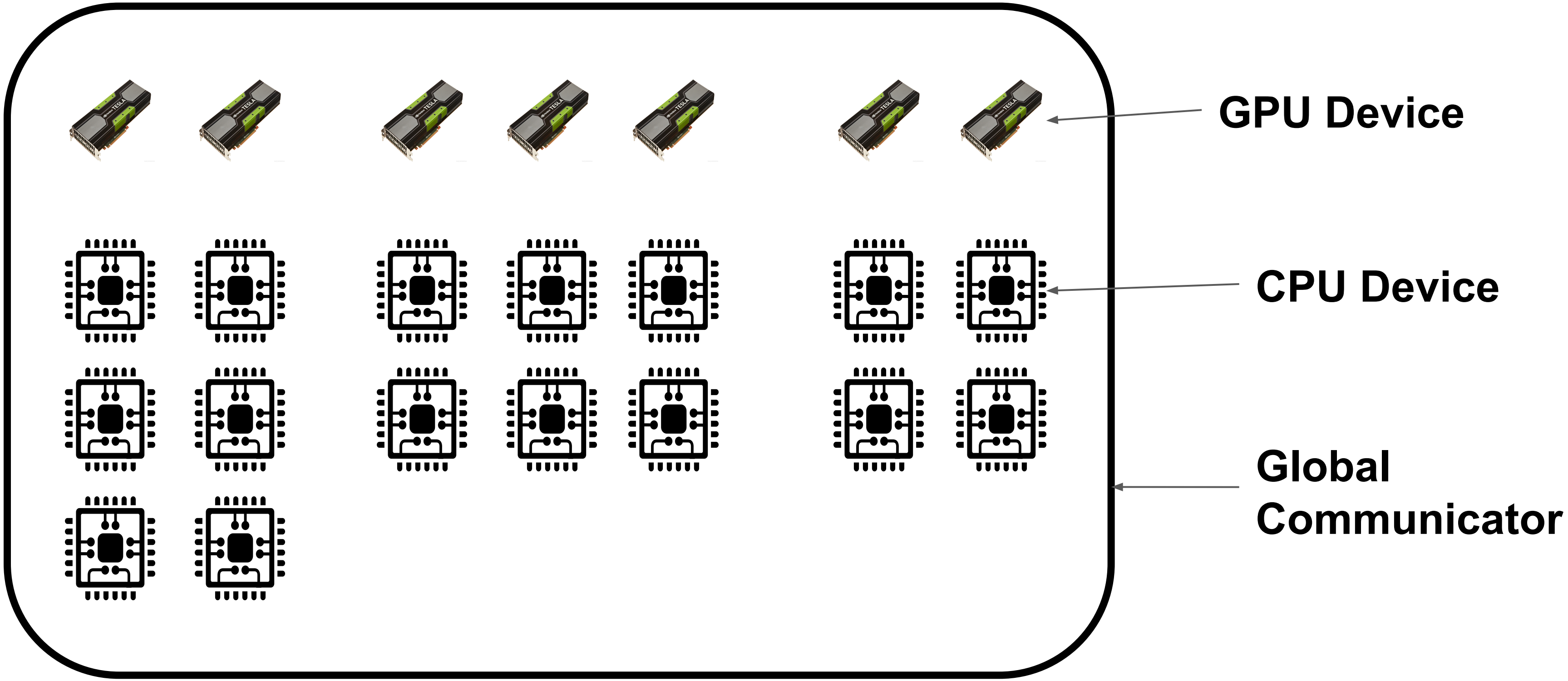


Example: Large-Size Problems

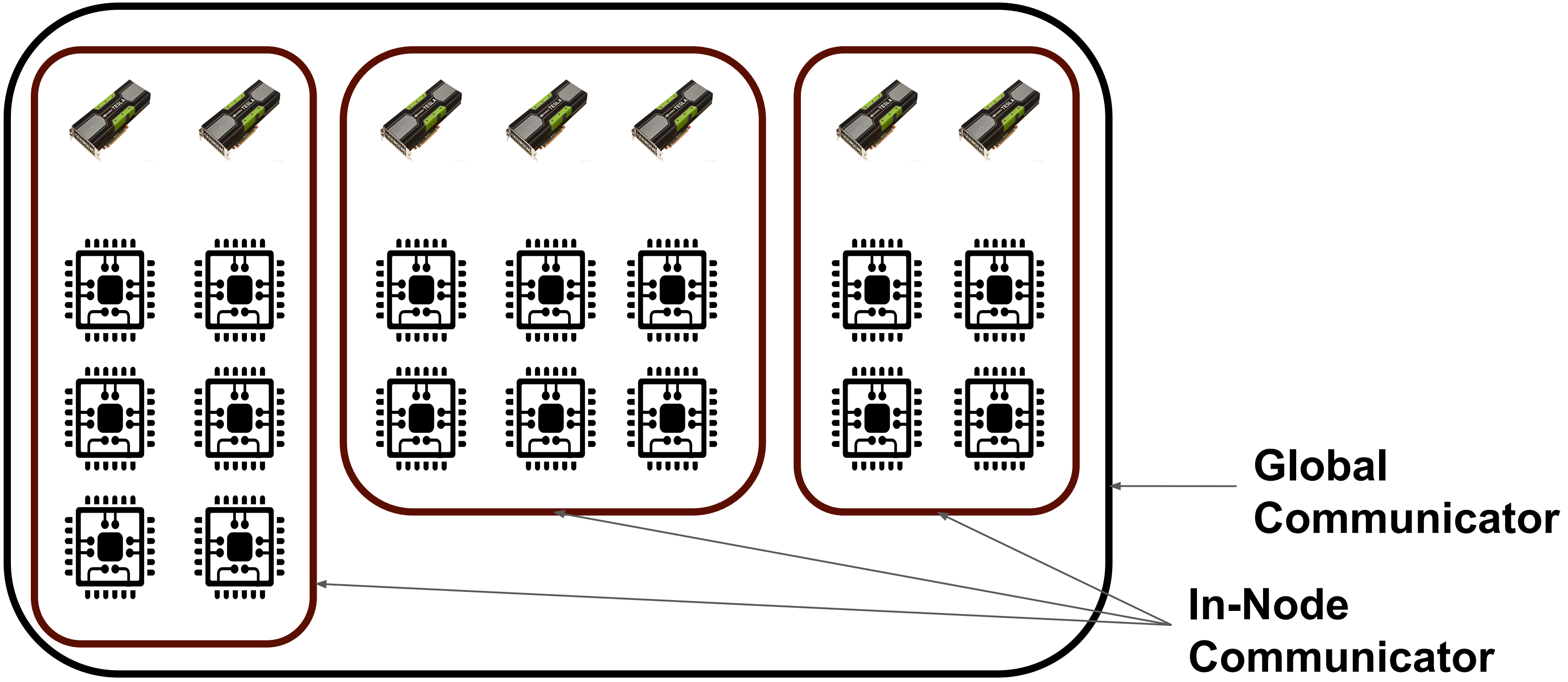




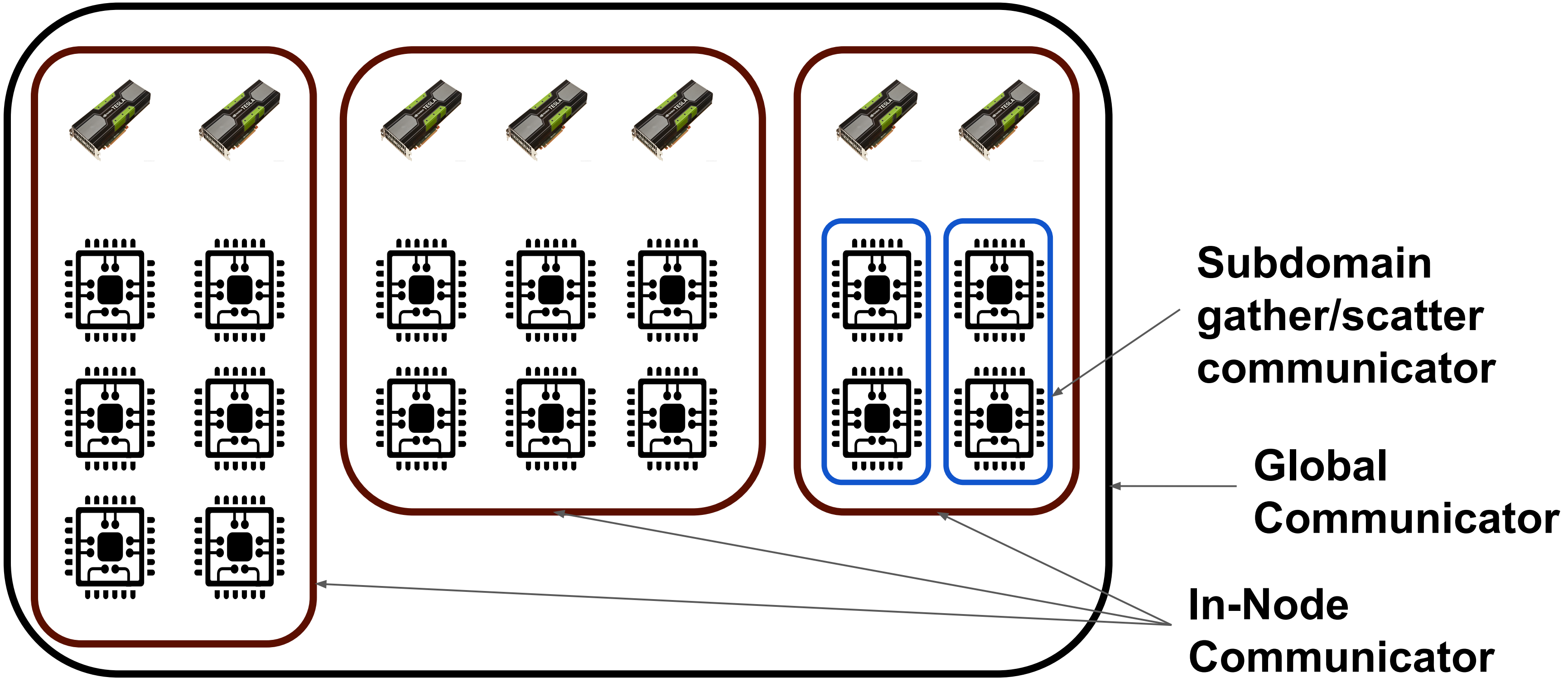
Our AmgX Wrapper handle this case !



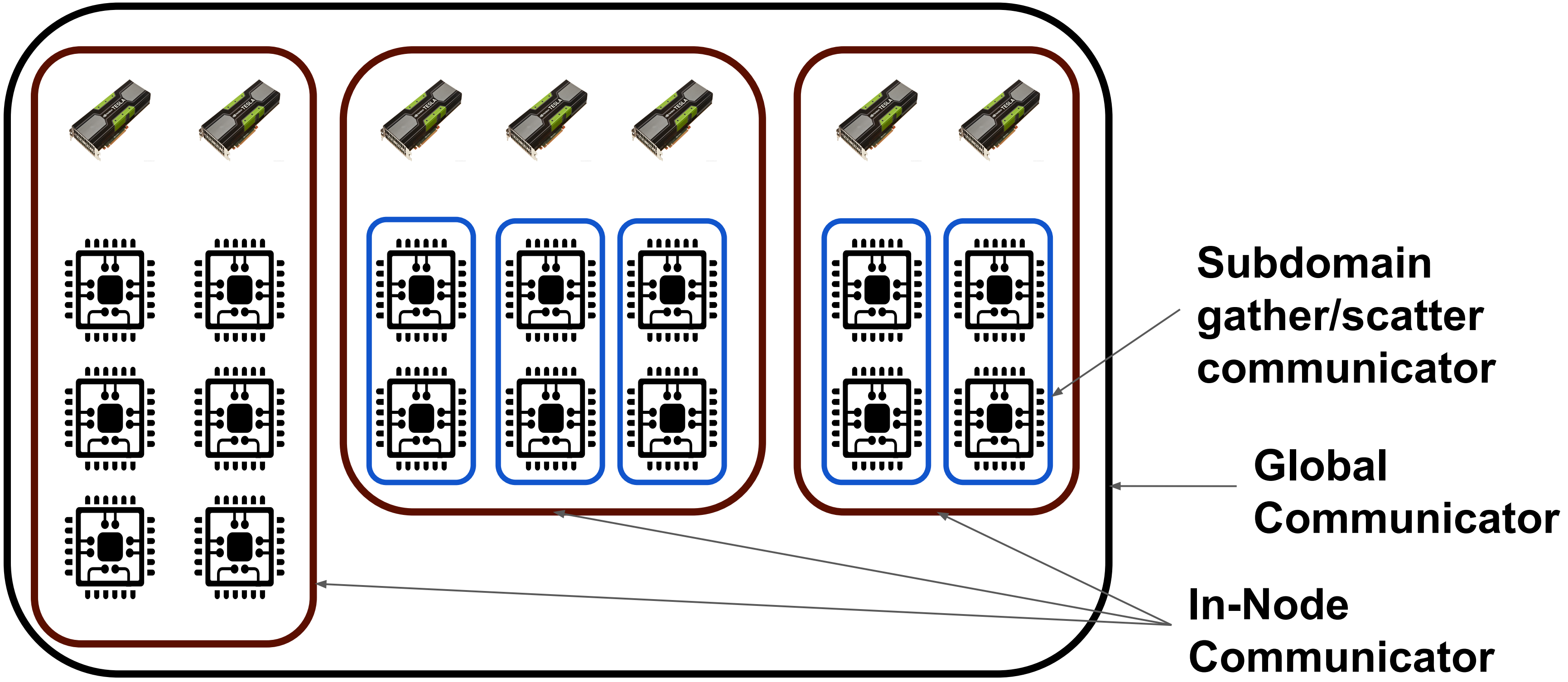
Our AmgX Wrapper handle this case !



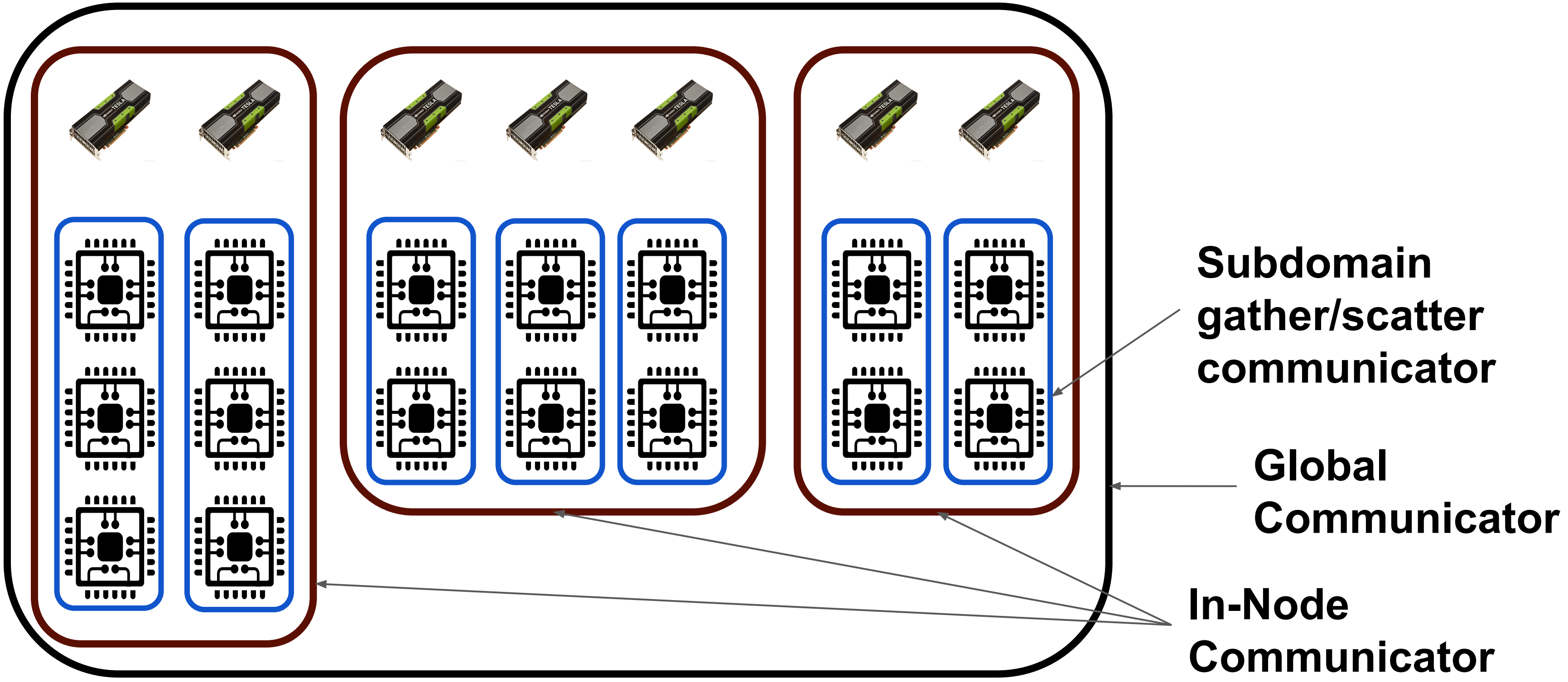
Our AmgX Wrapper handle this case !



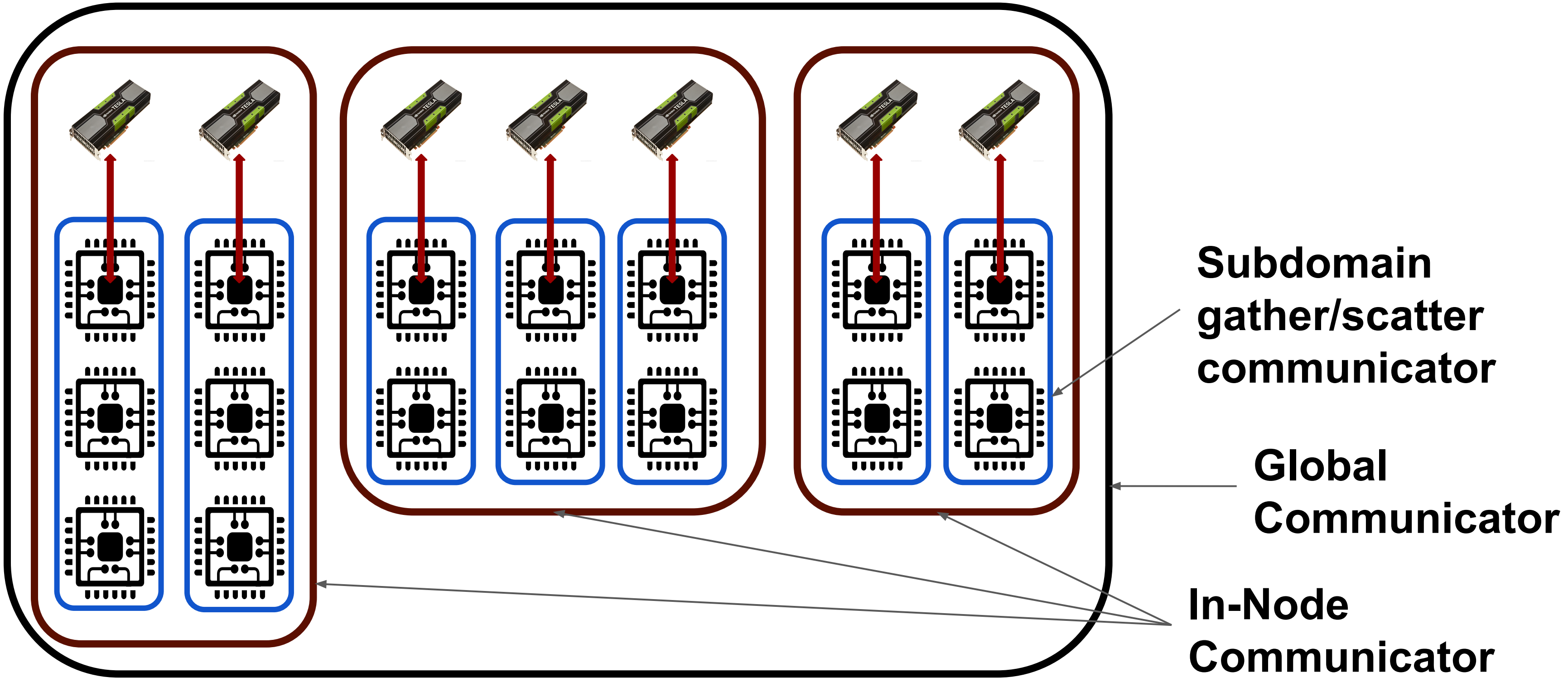
Our AmgX Wrapper handle this case !



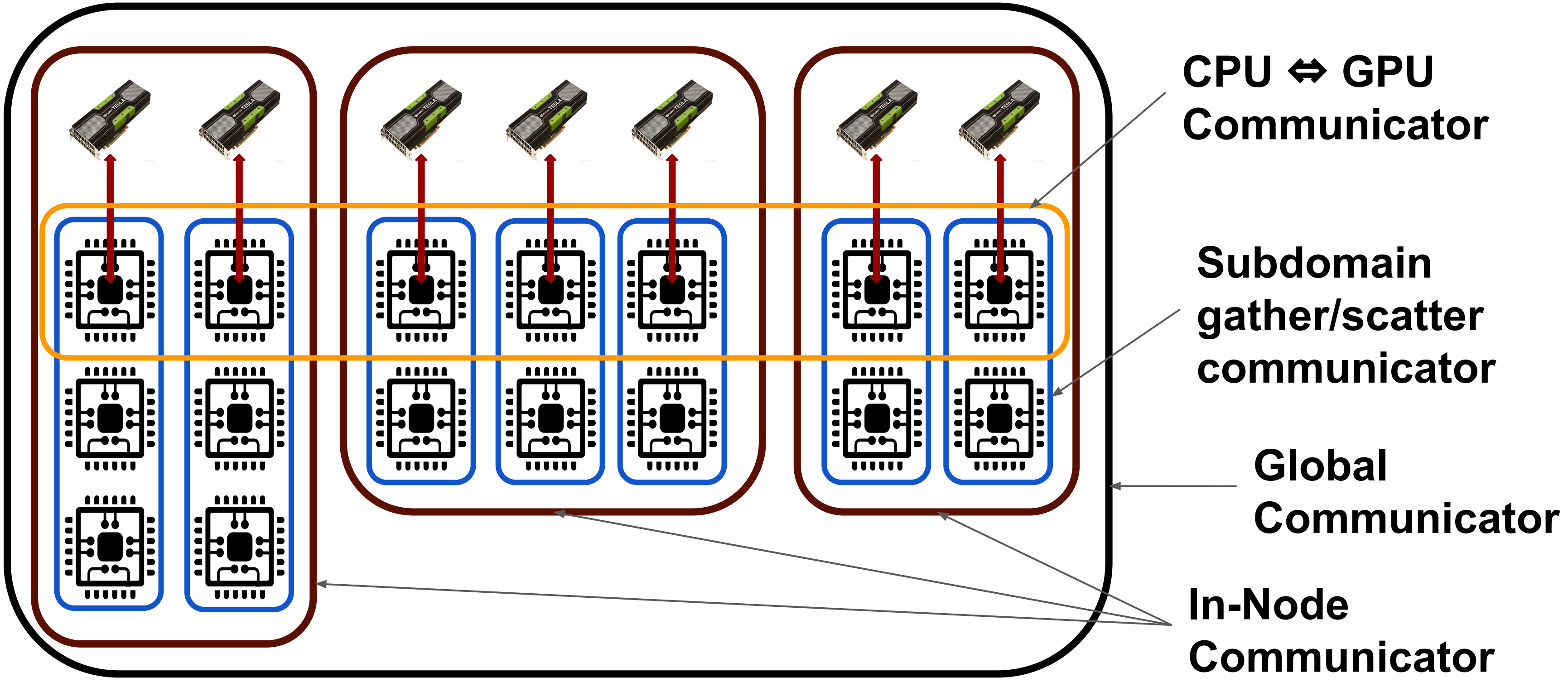
Our AmgX Wrapper handle this case !



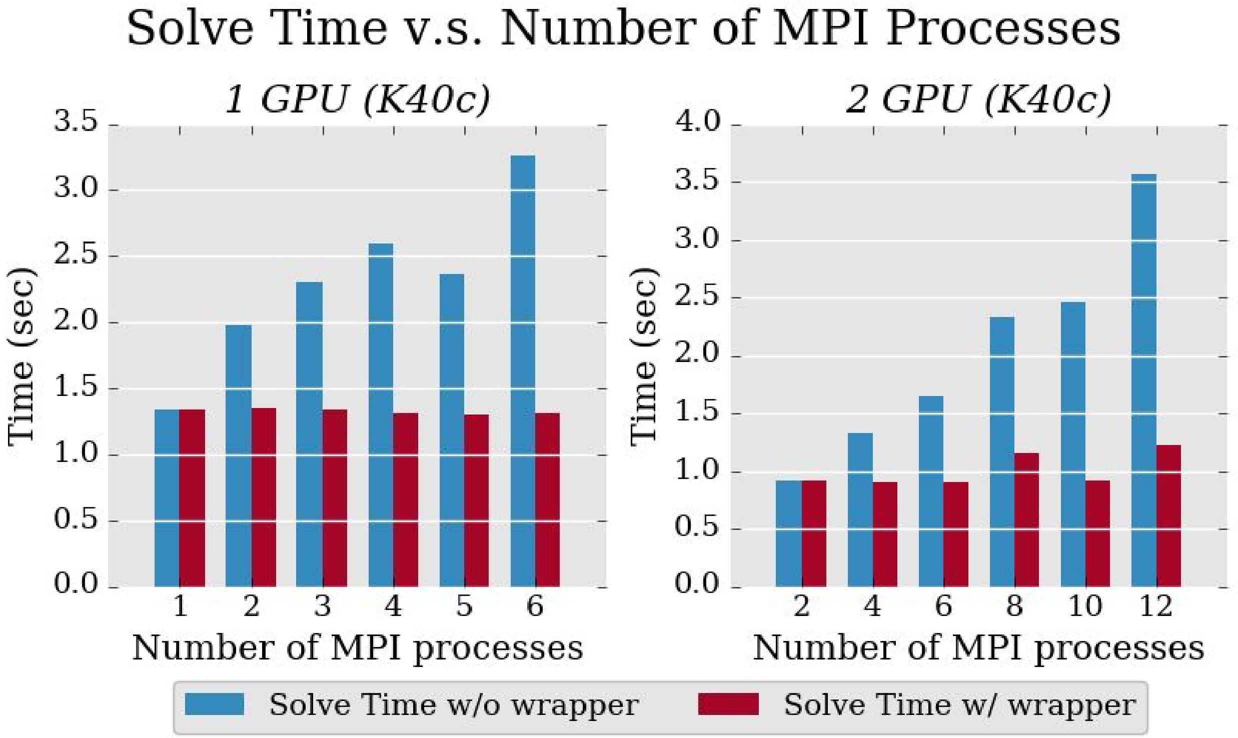
Our AmgX Wrapper handle this case !



Our AmgX Wrapper handle this case !

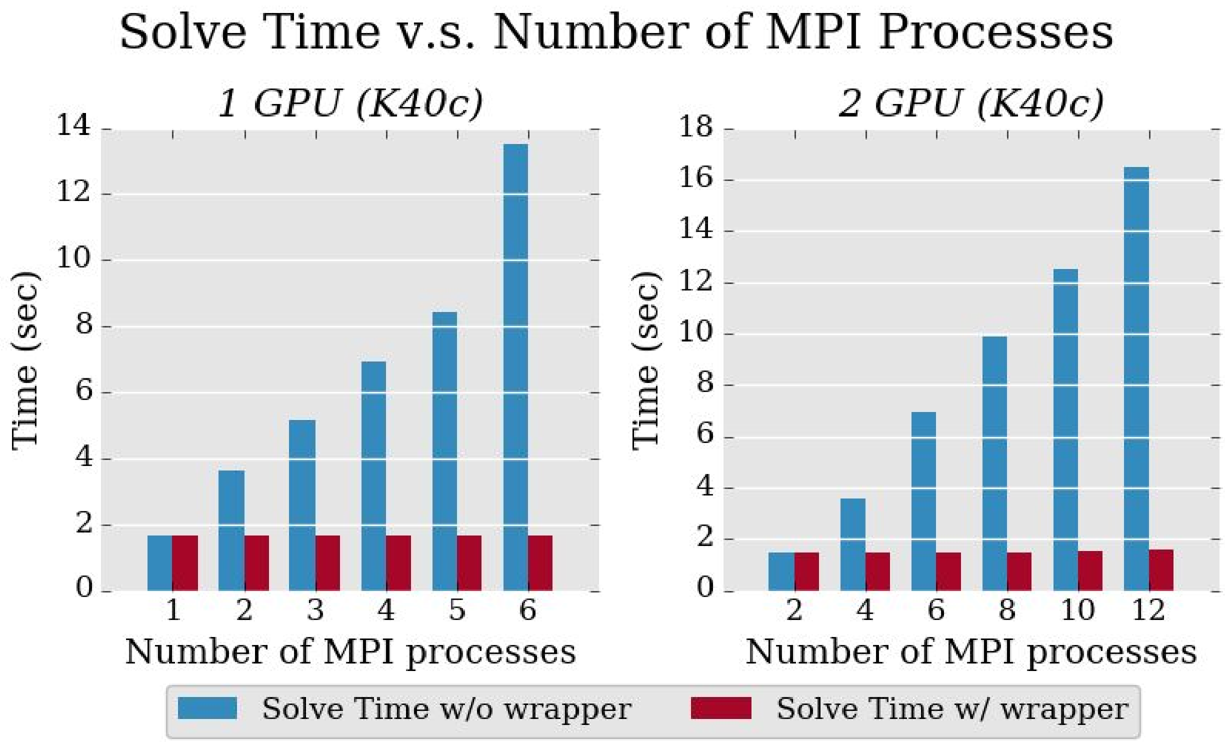


Check: 3D Poisson



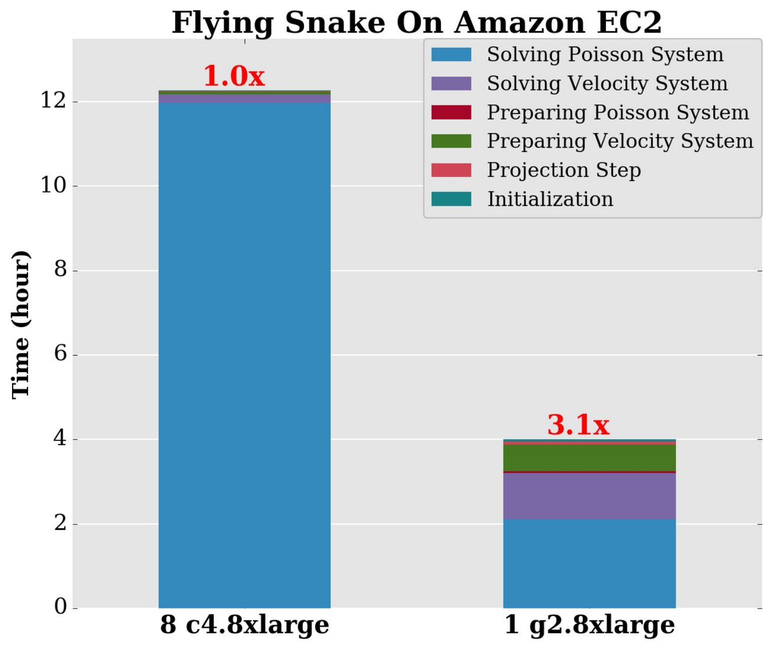
- 6M unknowns
- Solver:
  - CG
  - Classical AMG

Check: Modified Poisson Equation



- **2D Cylinder, Re 40**
- **2.25M unknowns**
- **Solver:**
  - **CG**
  - **Aggregation AMG**

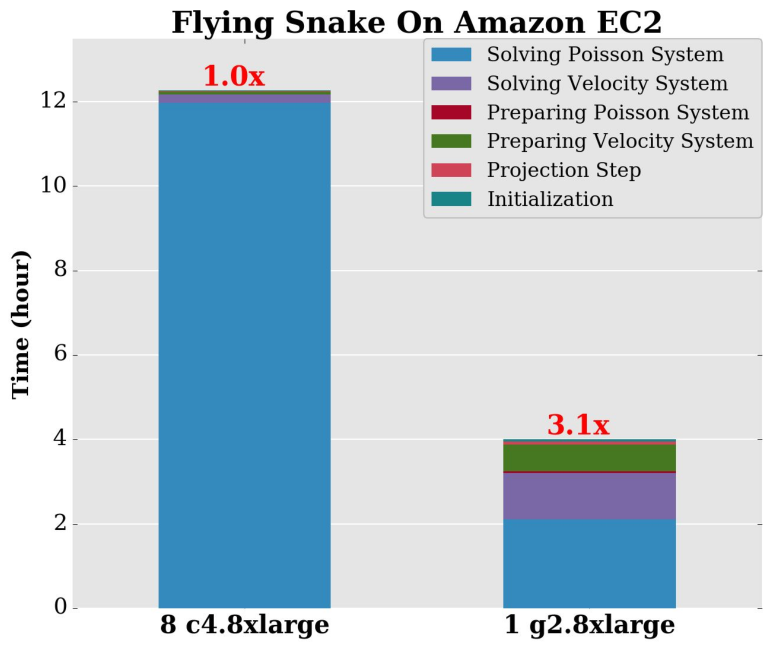
Potential Saving and Benefits: Cloud HPC Service



- **Using Spot Instances**
  - **CPU:**
    12.5 hr × \$0.5† / hr × 8 nodes = **\$50.0**

†This is the prices of the spot instances we used at that time.

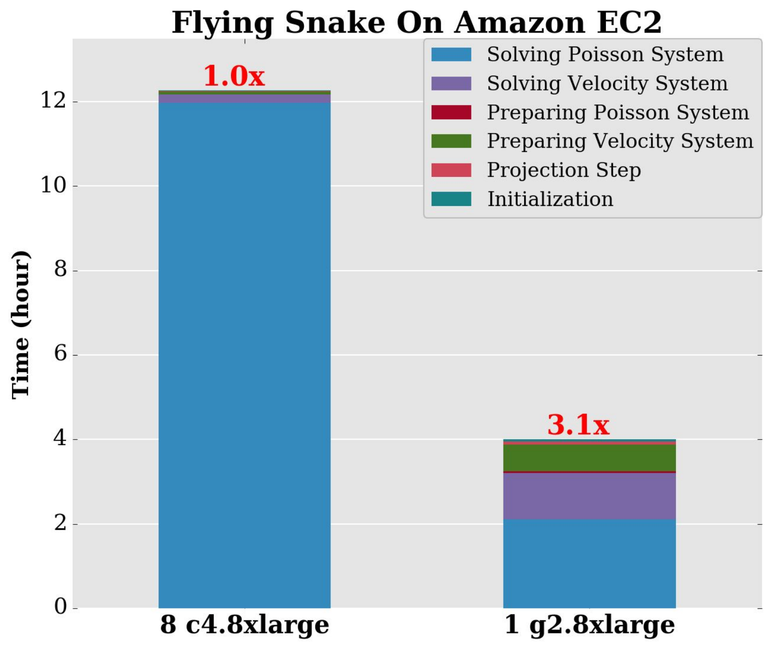
Potential Saving and Benefits: Cloud HPC Service



- **Using Spot Instances**
  - **CPU:**
    12.5 hr × \$0.5† / hr × 8 nodes = **\$50.0**
  - **GPU:**
    4 hr × \$0.5† / hr × 1 node = **\$2.0**
- **Using Official Price:**
  - **CPU:**
    12.5 hr × \$1.675 / hr × 8 nodes = **\$167.5**

†This is the prices of the spot instances we used at that time.

Potential Saving and Benefits: Cloud HPC Service



- **Using Spot Instances**
  - **CPU:**
    12.5 hr × \$0.5† / hr × 8 nodes = **\$50.0**
  - **GPU:**
    4 hr × \$0.5† / hr × 1 node = **\$2.0**
- **Using Official Price:**
  - **CPU:**
    12.5 hr × \$1.675 / hr × 8 nodes = **\$167.5**
  - **GPU:**
    4 hr × \$2.6 / hr × 1 node = **\$10.4**

†This is the prices of the spot instances we used at that time.

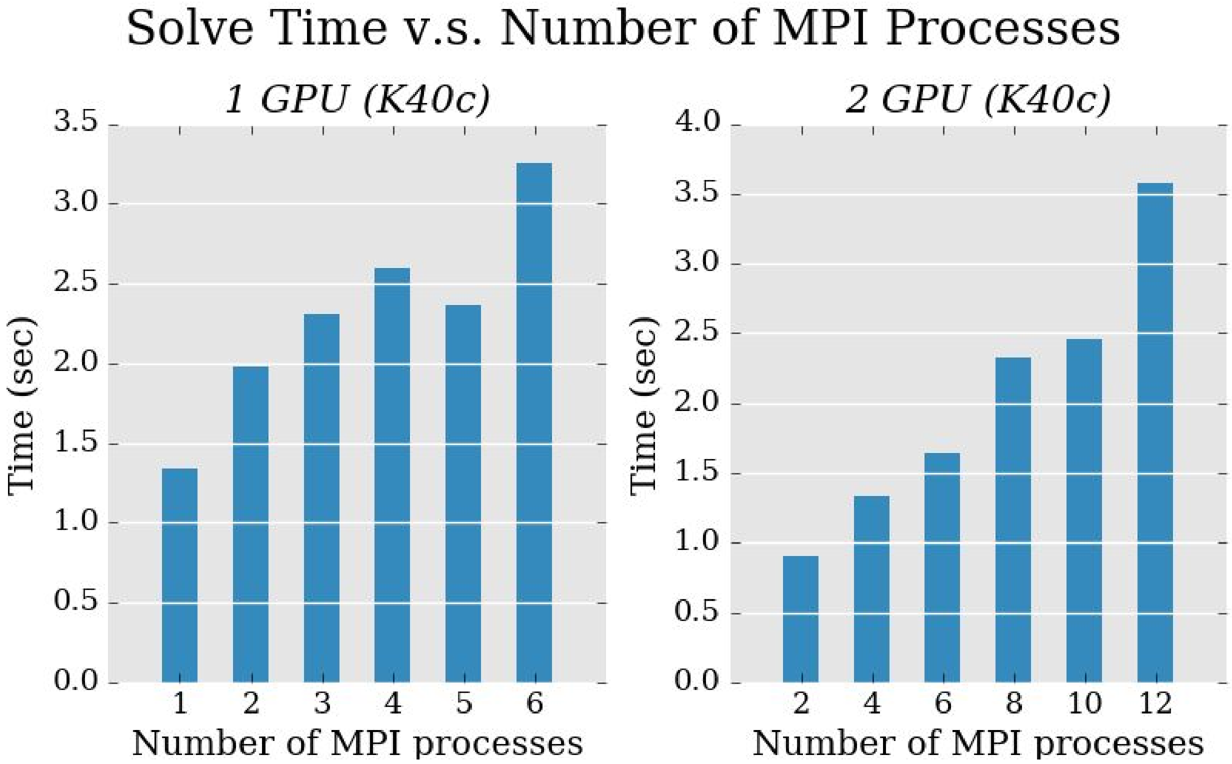
PetIBM

Solving Poisson systems in CFD solvers is already tough, but ...

AmgX

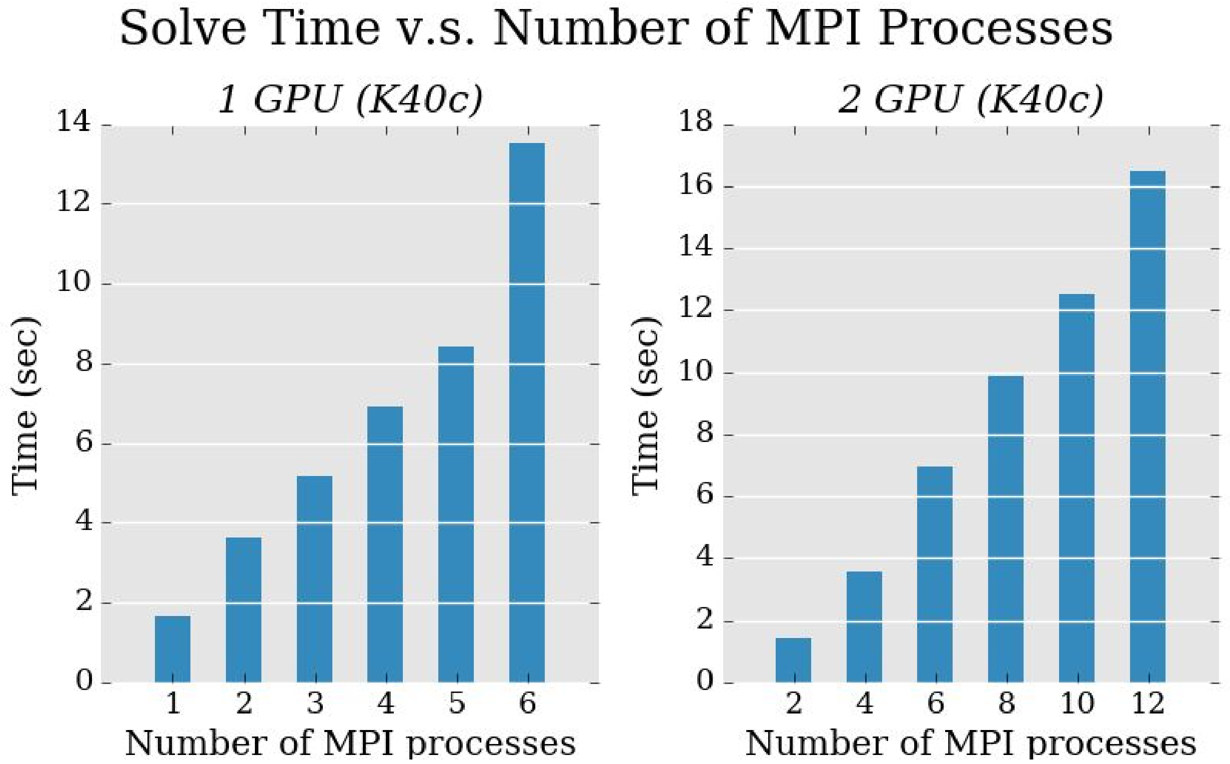
-
- **C API**
- **Unified Virtual Addressing**
- **Smoothers:**
    - Block-Jacobi, Gauss-Seidel, incomplete LU, Polynomial, dense LU … etc
- **Cycles:**
    - V, W, F, CG, CGF

Tests: 3D Poisson



- 6M unknowns
- Solver:
    - CG
    - Classical AMG

Tests: Modified Poisson Equation



- 2D Cylinder, Re 40
- 2.25M unknowns
- Solver:
  - CG
  - Aggregation AMG