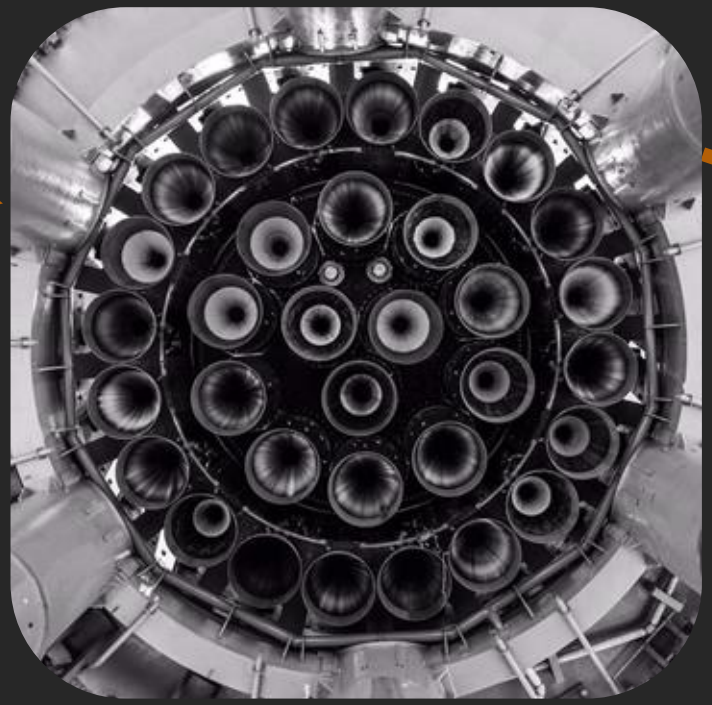
CFD at 1 quadrillion degrees of freedom via unified memory on OLCF Frontier

Find me in the SC25 Proceedings, GB Finalist

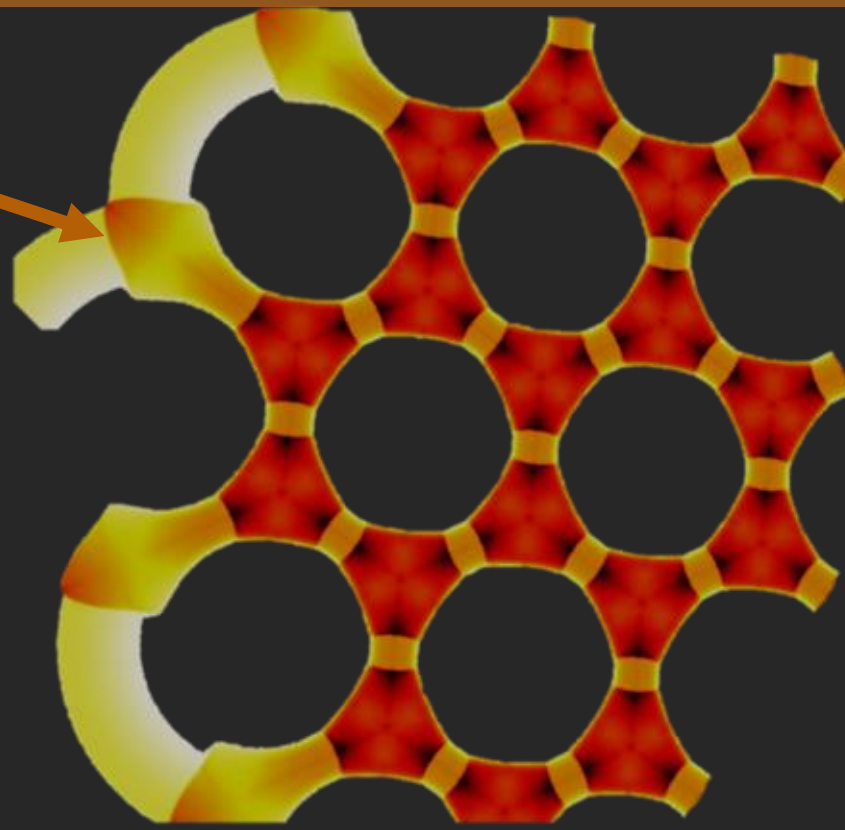
Spencer Bryngelson
Georgia Institute of Technology

OLCF User Meeting

Grand challenges in space exploration



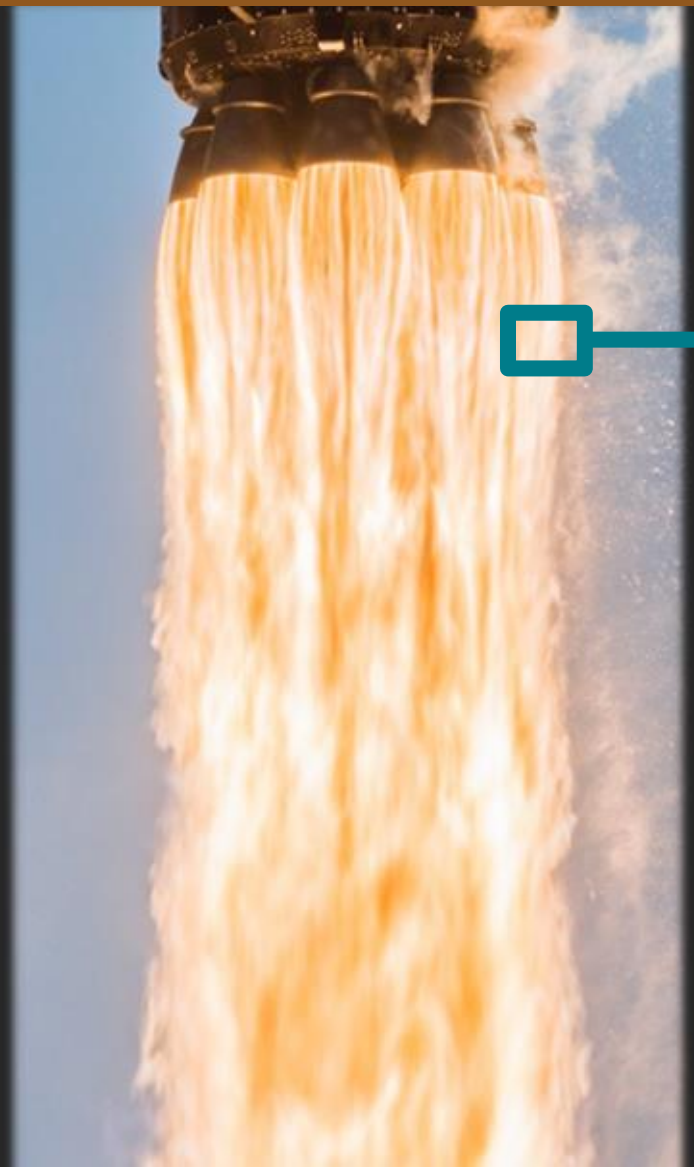
Engine configuration
(SpaceX Super Heavy)

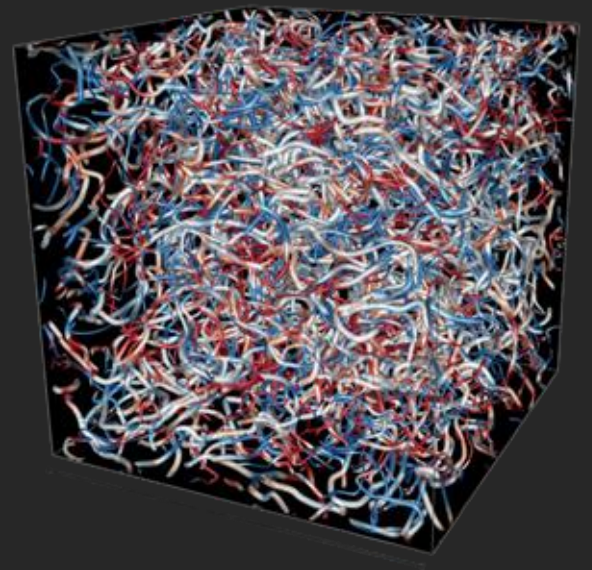


Backheat booster base
(our simulation)

Leave earth → Thrust → Heat booster base → Mission failure ✗

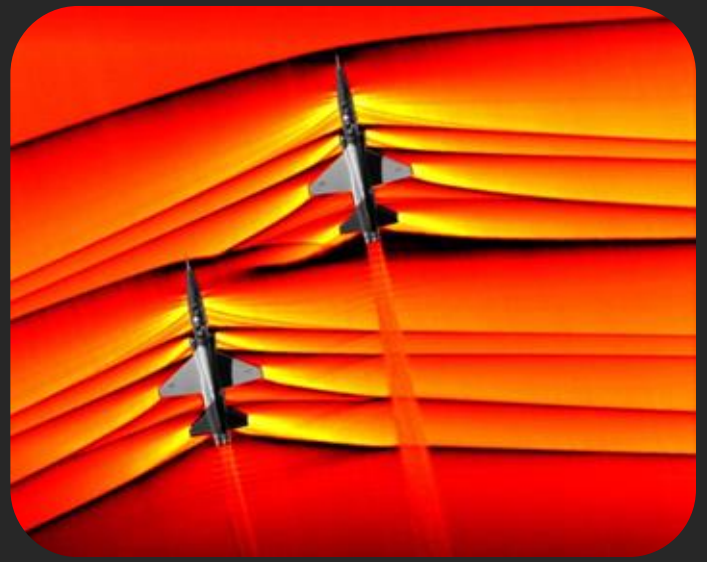
The need for scale





Turbulence

+

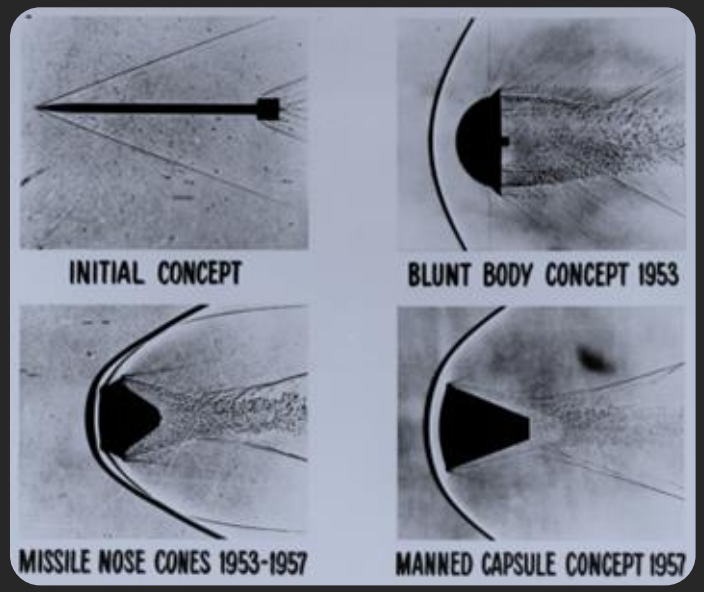


Shock interaction

Small & large scales
Multi-physics
Multi-phase
Resolution issues

Shocks in fluid dynamics

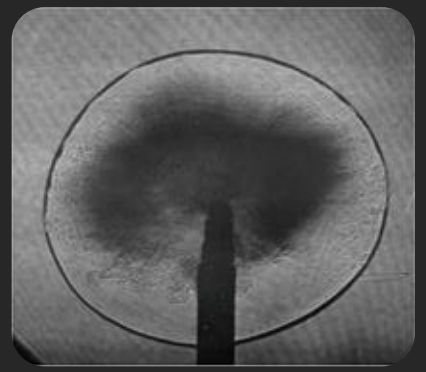
nasa.gov/image-feature/veil-nebula-supernova-remnant



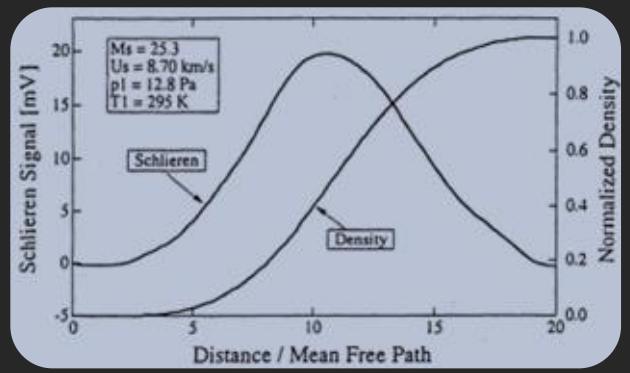
wikipedia.org/wiki/Atmospheric_entry/

Shocks

jumps in pressure, density, velocity



Takayama, 2019



Koreeda et al., 1995

Microscopically thin, so:

→ Cannot resolve in simulation

→ Expensive nonlinear numerics!

Information Geometric Regularization

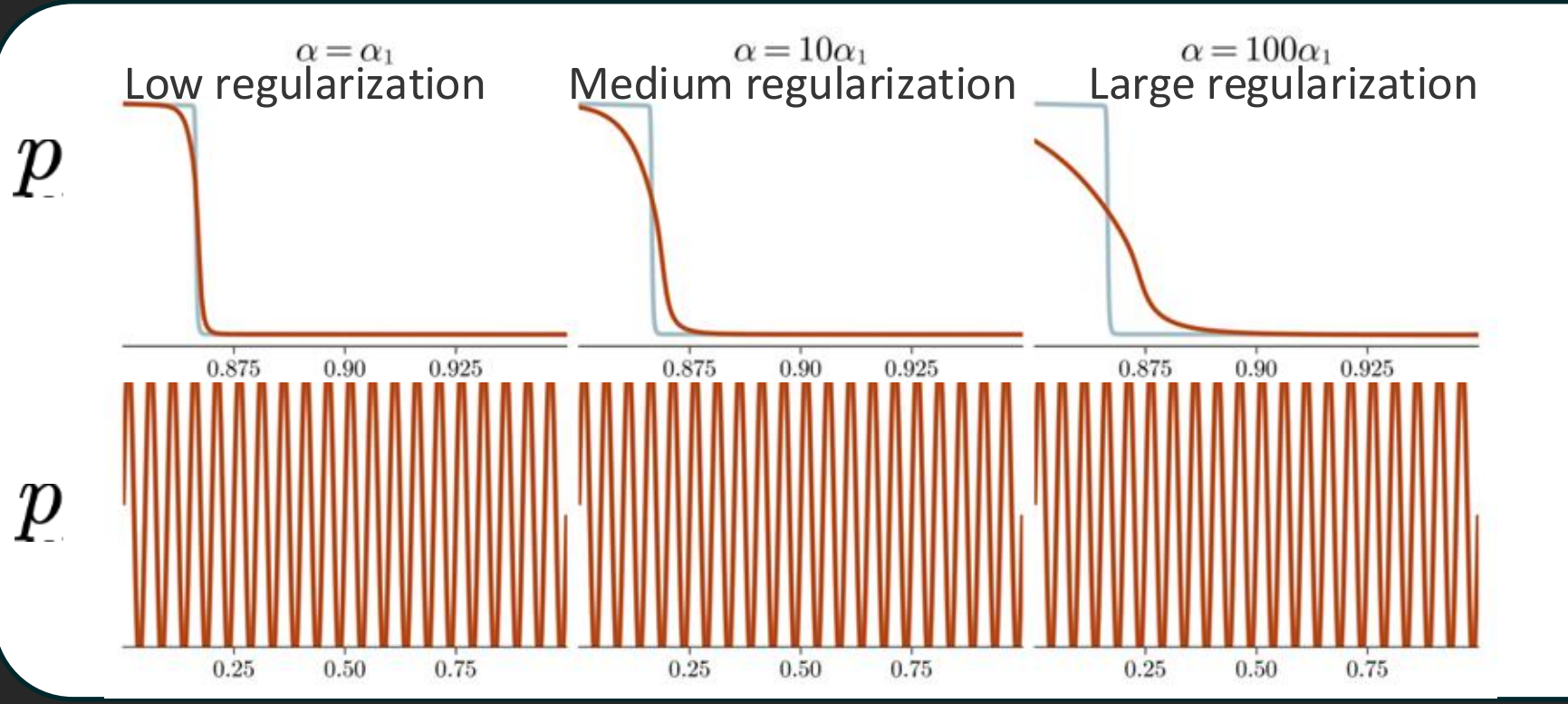
Enabling work: First *inviscid* regularization

$$\begin{cases} \partial_t \begin{pmatrix} \rho \boldsymbol{u} \\ \rho \end{pmatrix} + \operatorname{div} \begin{pmatrix} \rho \boldsymbol{u} \otimes \boldsymbol{u} + (p + \Sigma)\mathbf{I} \\ \rho \boldsymbol{u} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ 0 \end{pmatrix} \\ \rho^{-1}\Sigma - \alpha \operatorname{div}(\rho^{-1}\nabla\Sigma) = \alpha \left(\operatorname{tr}^2([\mathrm{D}\boldsymbol{u}]) + \operatorname{tr}([\mathrm{D}\boldsymbol{u}]^2) \right) \end{cases}$$

High-order smooth profile → Keeps fine-scale features!

Can use standard, linear numerics!

- Nominal solution
- Regularized solution



Cao & Schäfer, **Information geometric regularization of the barotropic Euler equation, 2023**

Information Geometric Regularization

Differential geometry allows making this notion precise

$$\ddot{\Phi} - \alpha \overset{\downarrow}{\operatorname{div}}\left(\operatorname{tr}\left([\mathrm{D}\Phi]^{-1}[\mathrm{D}\dot{\Phi}]\right)[\mathrm{D}\Phi]^{-1}\right) = -\alpha \overset{\downarrow}{\operatorname{div}}\left(\operatorname{tr}\left(\left([\mathrm{D}\Phi]^{-1}[\mathrm{D}\dot{\Phi}]\right)^2\right)[\mathrm{D}\Phi]^{-1}\right) - \alpha \overset{\downarrow}{\operatorname{div}}\left(\operatorname{tr}^2\left([\mathrm{D}\Phi]^{-1}[\mathrm{D}\dot{\Phi}]\right)[\mathrm{D}\Phi]^{-1}\right) \Rightarrow \begin{cases} \partial_t \begin{pmatrix} \rho \boldsymbol{u} \\ \rho \end{pmatrix} + \operatorname{div}\begin{pmatrix} \rho \boldsymbol{u} \otimes \boldsymbol{u} + (p + \Sigma)\mathbf{I} \\ \rho \boldsymbol{u} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ 0 \end{pmatrix} \\ \rho^{-1}\Sigma - \alpha \operatorname{div}(\rho^{-1}\nabla\Sigma) = \alpha\left(\operatorname{tr}^2\left([\mathrm{D}\boldsymbol{u}]\right) + \operatorname{tr}\left([\mathrm{D}\boldsymbol{u}]^2\right)\right) \end{cases}$$

Summarize all geometric effects into scalar Σ added to pressure

Can solve this with standard, linear numerics

IGR: So what?

Finally!

Speed ↔ Prior methods expensive w/ limiters, detectors

Little memory ↔ Can accumulate arithmetic contributions

Before

```
for each interface i+1/2:
    UL = U[i]; UR = U[i+1]

    rhoL,uL,pL = cons_to_prim(UL)
    rhoR,uR,pR = cons_to_prim(UR)

    cL = sqrt(gamma * pL / rhoL)
    cR = sqrt(gamma * pR / rhoR)

    SL = min(uL - cL, uR - cR)
    SR = max(uL + cL, uR + cR)

    FL = F(UL); FR = F(UR)

    SM = (pR - pL + rhoL*uL*(SL - uL) - &
         rhoR*uR*(SR - uR)) &
         / (rhoL*(SL - uL) - &
         rhoR*(SR - uR))
```

```
if 0 <= SL:
        Flux[i+1/2] = FL
    else if SL <= 0 and 0 <= SM:
        ULs = star_left(UL, SL, SM)
        Flux[i+1/2] = FL + SL * (ULs - UL)
    else if SM <= 0 and 0 <= SR:
        URs = star_right(UR, SR, SM)
        Flux[i+1/2] = FR + SR * (URs - UR)
    else:
        Flux[i+1/2] = FR
```

```
function star_left(UL, SL, SM):
    rhoL,uL,pL = cons_to_prim(UL)
    EL = UL[2]
    rhoLs = rhoL * (SL - uL) / (SL - SM)
    pS = pL + rhoL * (SL - uL) * (SM - uL)
    ES = ((SL - uL)*EL - pL*uL + pS*SM) &
         / (SL - SM)
    return [rhoLs, rhoLs*SM, ES]

function star_right(UR, SR, SM):
    rhoR,uR,pR = cons_to_prim(UR)
    ER = UR[2]
    rhoRs = rhoR * (SR - uR) / (SR - SM)
    pS = pR + rhoR * (SR - uR) * (SM - uR)
    ES = ((SR - uR)*ER - pR*uR + pS*SM) &
         / (SR - SM)
    return [rhoRs, rhoRs*SM, ES]
```

IGR: So what?

Finally!

Speed ↔ Prior methods expensive w/ limiters, detectors

Little memory ↔ Can accumulate arithmetic contributions

Now

```
for each interface i+1/2:
  a=max(abs(u[i])+cL,abs(u[i+1])+cR)
  UL=U[i]; UR=U[i+1]
  Flux[i+1/2]=0.5*(F(UL)+F(UR))-a*(UR - UL)/2
```

IGR: So what?

Finally!

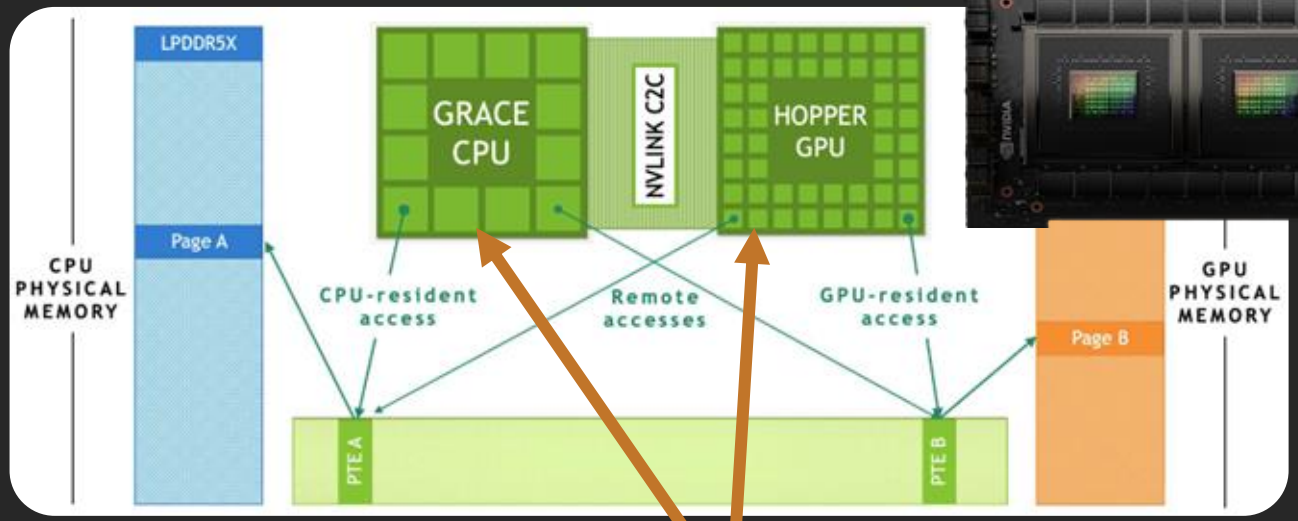
Speed ↔ Prior methods expensive w/ limiters, detectors

Little memory ↔ Can accumulate arithmetic contributions

and

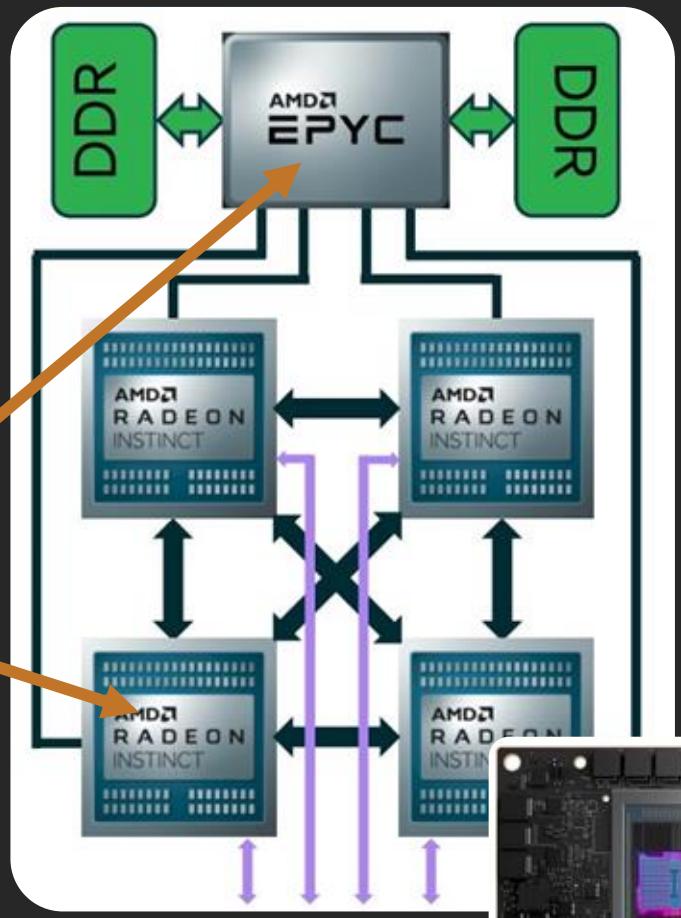
Mixed FP32/16 Precision ↔ Better conditioned numerics

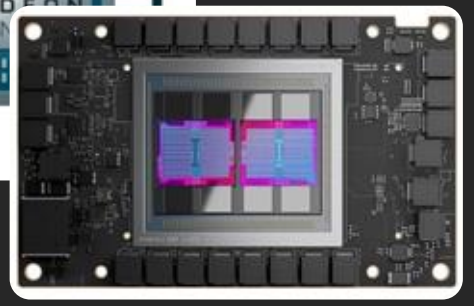
CPU–GPU Communication



GH200

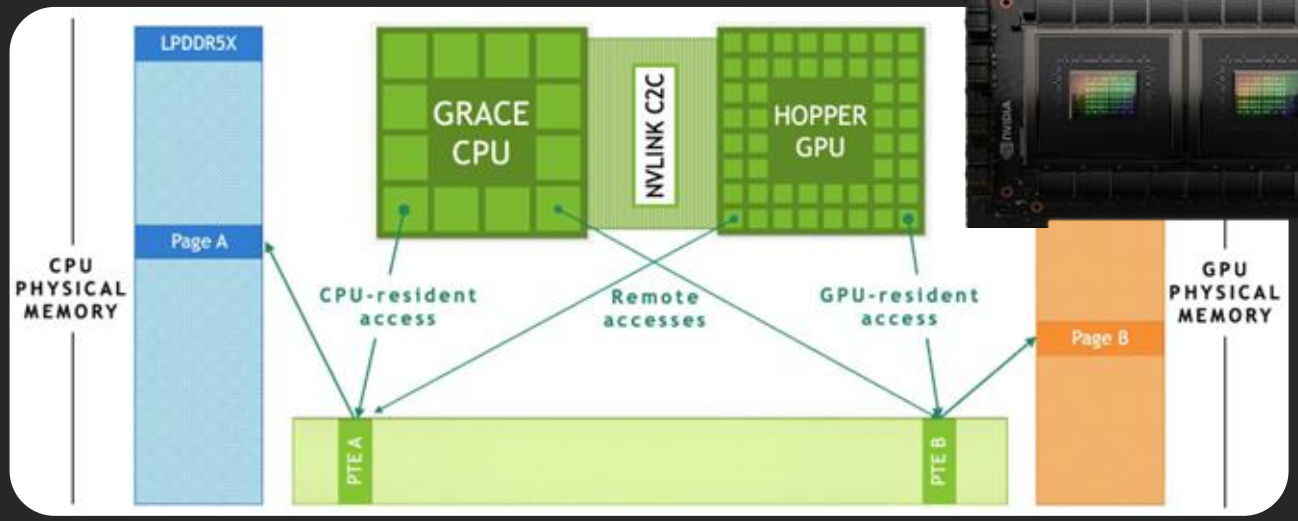
These guys
“loosely” connected at speed of computation
[we need to treat!]

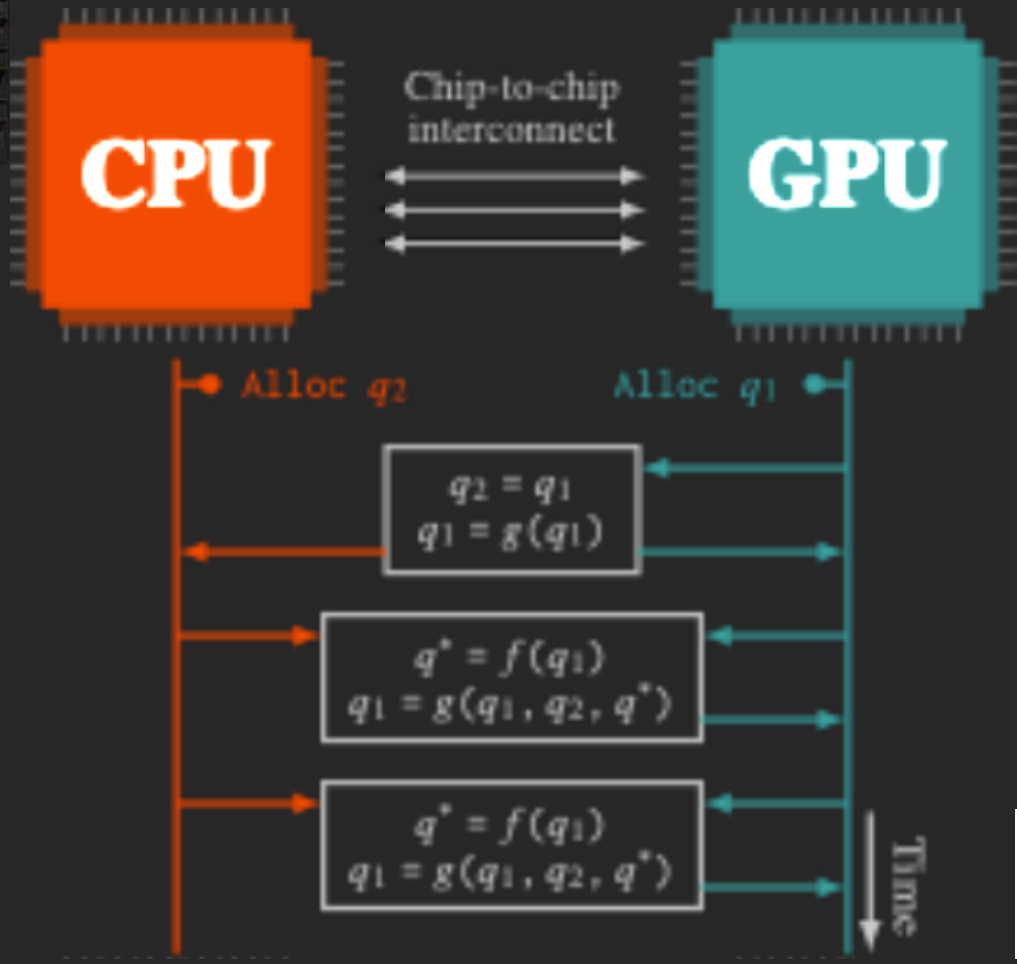




Frontier node

A solution





2560^3 grid points per Alps node
[4-way GH200]

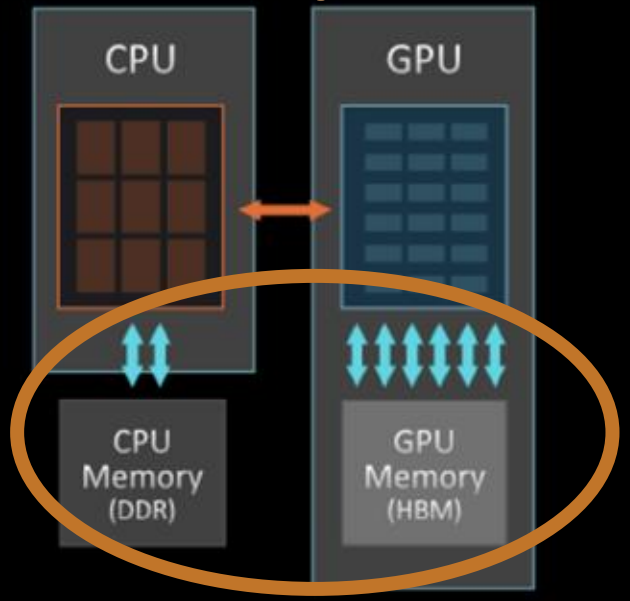
2780^3 grid points per Frontier node
[Trento+MI250X]

My comments

AMD MI300A UNIFIED MEMORY APU ARCHITECTURE BENEFITS

AMD CDNA™ 2 Coherent Memory Architecture → AMD CDNA™ 3 Unified Memory APU Architecture

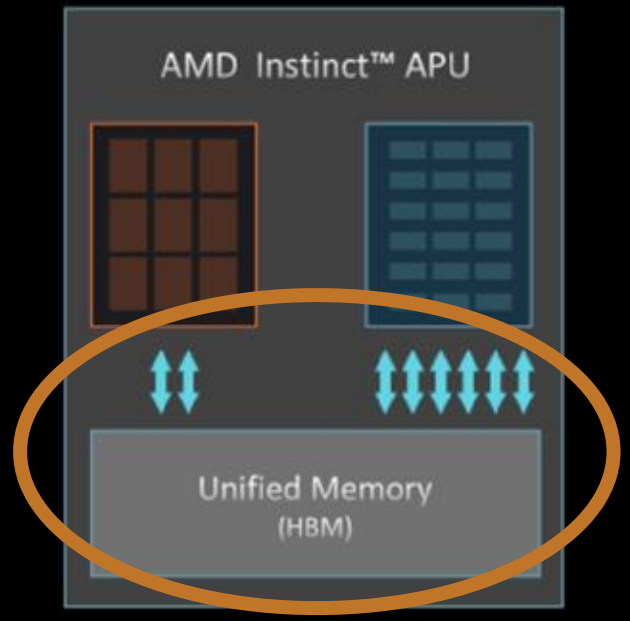
MI250X+Trento / GH200–like



Now, we are fast (w/ tricks)

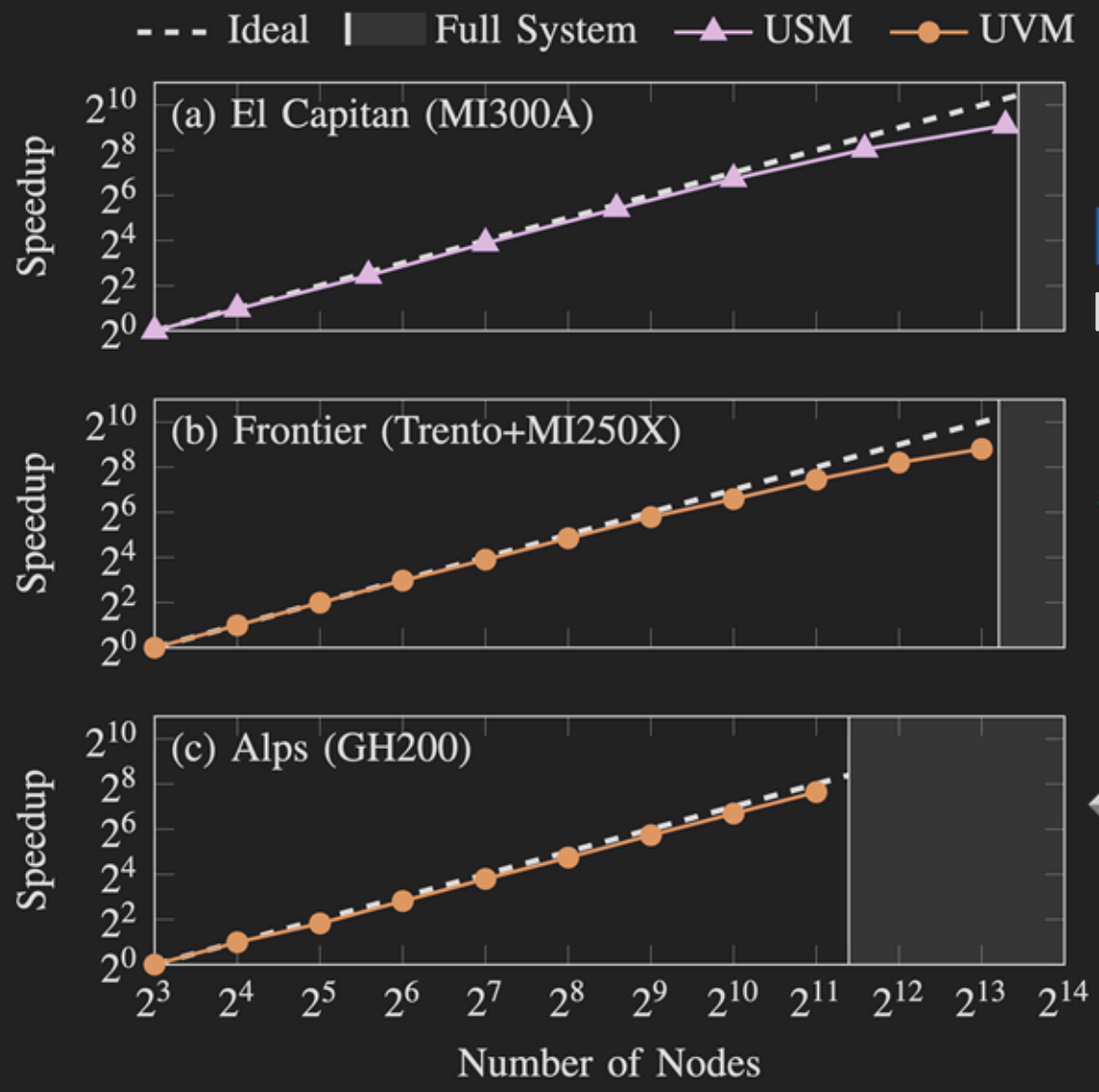
- Eliminate Redundant Memory Copies
- No programming distinction between host and device memory spaces
- High performance, fine-grained sharing between CPU and GPU processing elements
- Single process can address all memory, compute elements on a socket

MI300A-like



1 HBM pool

Performance: Strong scaling



Lawrence Livermore National Laboratory

LLNL El Capitan

OAK RIDGE National Laboratory

OLCF Frontier

CSCS

CSCS Alps

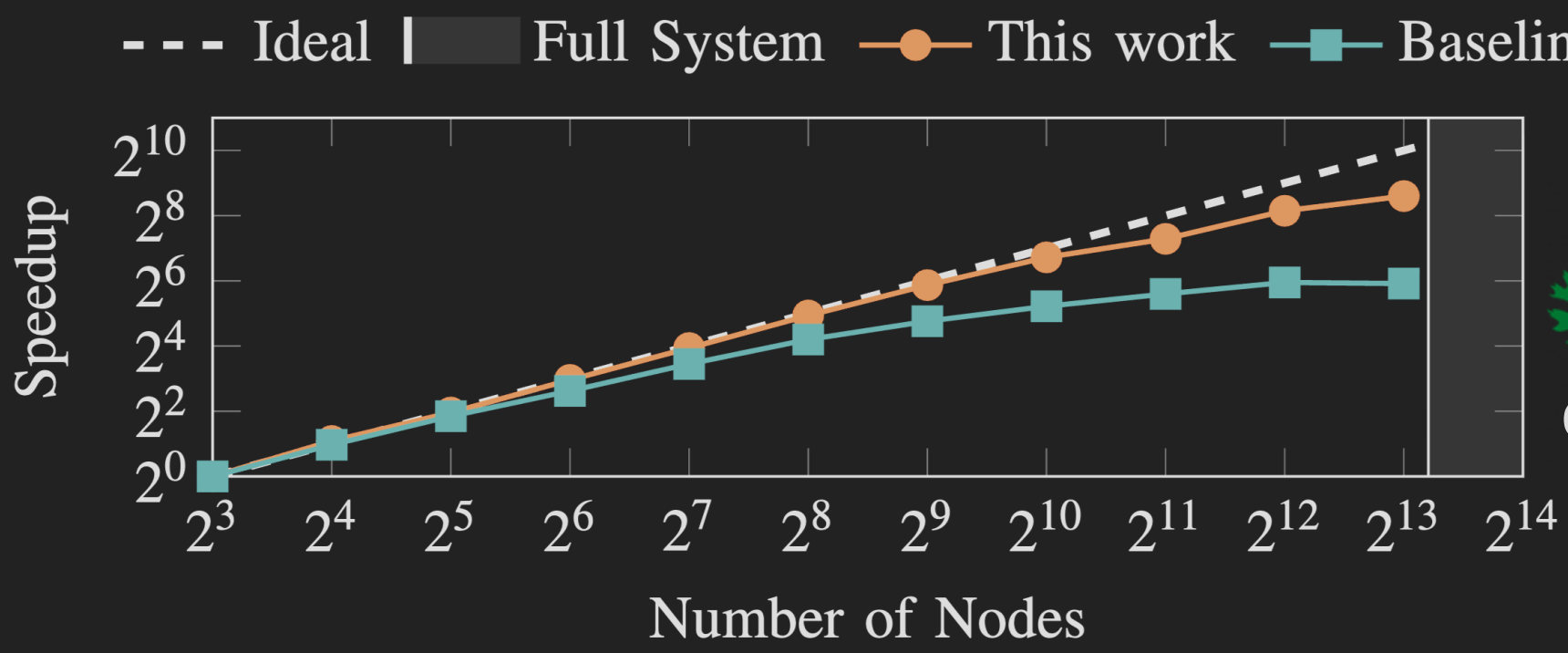
MFC

mflowcode.github.io

From 8 nodes to full system

< 50% perf. penalty

Performance: Strong scaling



mflowcode.github.io

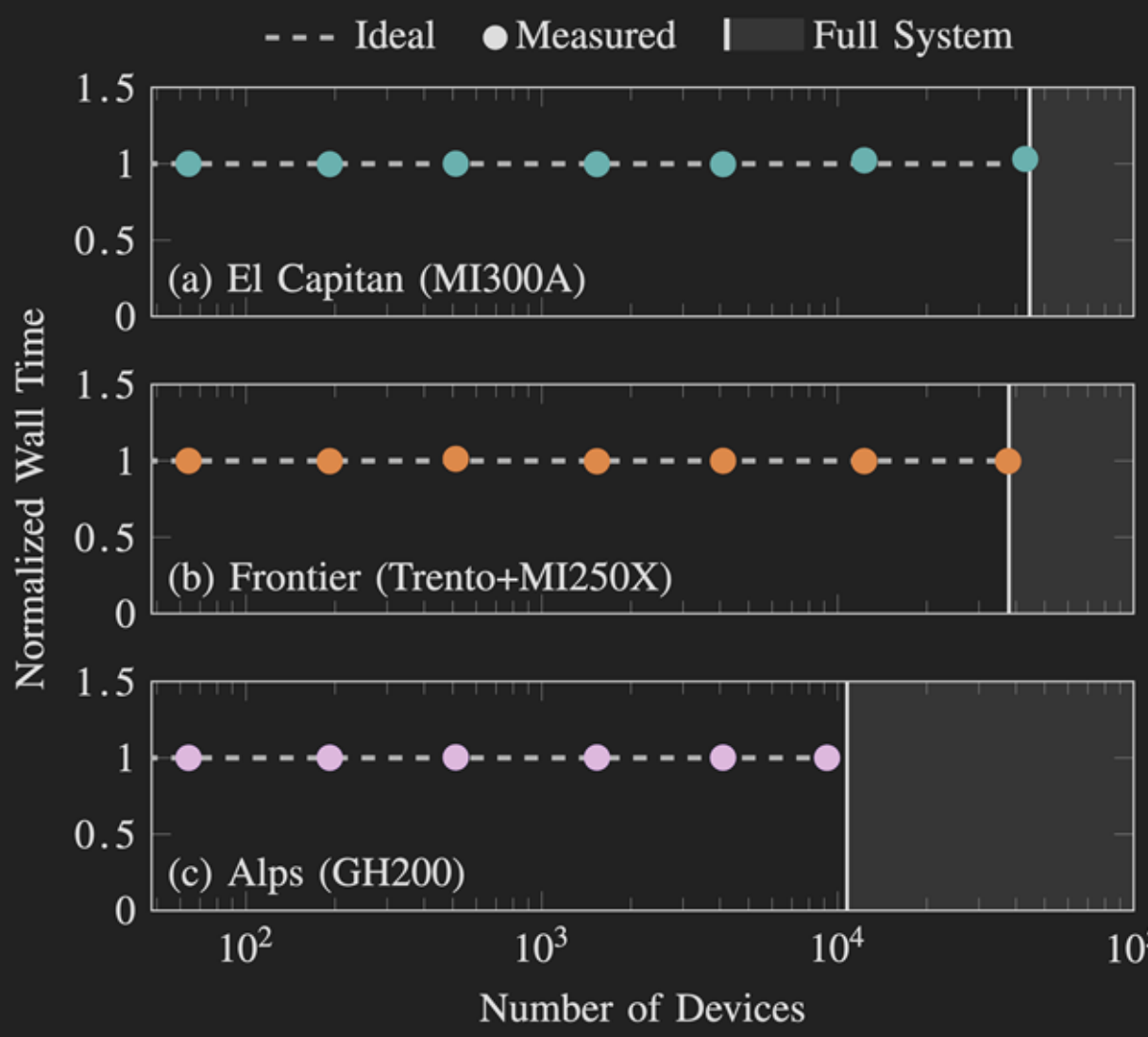
OAK RIDGE National Laboratory

OLCF Frontier

Before: Greater than **95%** perf. penalty ❌

Now: Less than **50%** perf. penalty ✅

Performance: Weak scaling



Lawrence Livermore National Laboratory
LLNL El Capitan

OAK RIDGE National Laboratory
OLCF Frontier

CSCS
CSCS Alps

MFC
mflowcode.github.io

Ideal scaling

~1.5 EFLOPS

200T points [Frontier]
(1 quadrillion DoF)

25x SoA

Performance: Speed and energy

mflowcode.github.io

Wall-time performance

Device	Baseline	IGR (in-core)	IGR (unified)
GH200	21.73	5.41	5.65
MI250X GCD	76.25	18.23	25.90

~5x faster

Energy use

Energy (μJ)	Frontier	Alps
Baseline	165.76	49.88
IGR	48.08	8.80

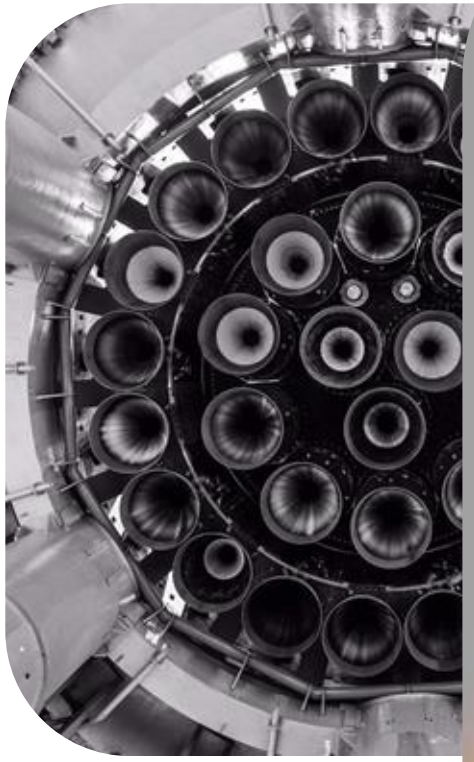
~4x less energy

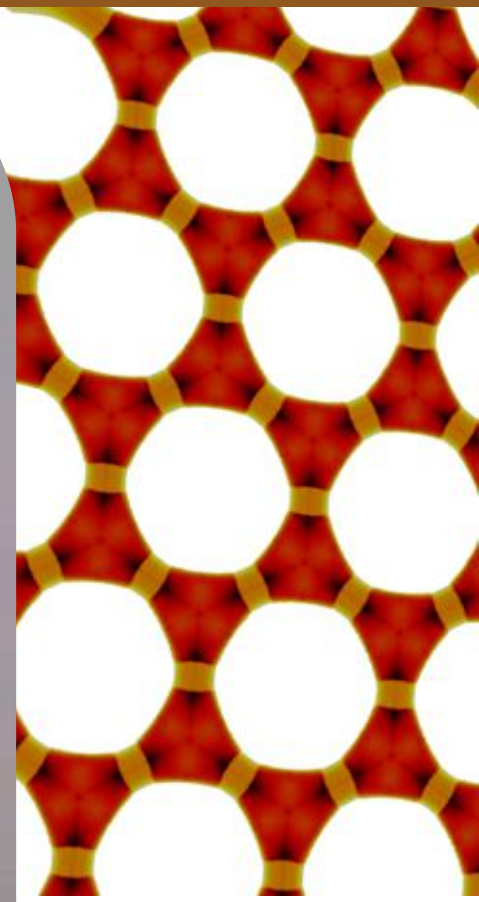
…all compared to optimized baseline in same code

Demonstration

2-species (gas/gas) Mach 10 mock exhaust

Example: Rocket engines





SpaceX Super ster 🧨

Take away

Efficient CPU–GPU use

Change equations

Fast, scalable simulation