Algorithmic Design for Exaflop Machines

Rainald Löhner

Center for Computational Fluid Dynamics
College of Science, George Mason University

cfd.gmu.edu/~rlohner

Outline

- Problems Targeted
- Foreseeable Machine Architecture(s)
- Current Limitations & Corrolaries
  - Memory Limit
  - MPI Limit
- `Minimal Memory Access CFD'

If You Had An Exaflop Machine…

- Not: Potential Flow (Iphone)
- Not: Euler (Laptop)
- Not: RANS: 64-Core Workstation (SMP)

- Yes: LES, DNS
- Yes: Chemically Reacting Flow
- Yes: LES, DNS + Chem + Multiphase + ...

Optimistic Requirements

	LES		DNS	
Re	npoin	ntime	npoin	ntime
10^{6}	$10^{8.4}$	$10^{4.2}$	$10^{13.6}$	$10^{6.8}$
10^{7}	$10^{8.8}$	$10^{4.4}$	$10^{15.2}$	$10^{7.6}$
10^{8}	$10^{9.2}$	$10^{4.6}$	$10^{16.8}$	$10^{8.4}$
10^{9}	$10^{9.6}$	$10^{4.8}$	$10^{18.4}$	$10^{9.2}$

Optimistic Requirements

	LES		DNS	
Re	npoin	ntime	npoin	ntime
10^6	$10^{8.4}$	$10^{4.2}$	$10^{13.6}$	$10^{6.8}$
10^7	$10^{8.8}$	$10^{4.4}$	$10^{15.2}$	$10^{7.6}$
10^8	$10^{9.2}$	$10^{4.6}$	$10^{16.8}$	$10^{8.4}$
10^9	$10^{9.6}$	$10^{4.8}$	$10^{18.4}$	$10^{9.2}$

Consequences

- Need Many (Many !) Points ($> 10^9$)
  - Spatial Grading (Of **Isotropic** Grids) Essential
  - Why Not Space Them As Regularly As Possible ?
  - Parallel Grid Generation
  - Fast, High-Order (Space) Solvers
- Need Many Timesteps ($> 10^6$)
  - Fast, High-Order (Time) Solvers (Vortex Propagation)
- Few Will Have Access 24/7/365 to 10^6 Processors [Energy]➔
  - RANS Grids Relevant for Another 20 Years
  - Geometry (Good Body-Fitted Grids) Relevant

Worries About LES…

- Aerodynamics / Aeroacoustics: RANS → LES → DNS
  - Increase in Spatial/Temporal Order [4^{th}-6^{th} Order]
  - Increase in Nr. Of Gridpoints/DOF: $O(10^3\text{-}10^6)$ [MPI OK]
  - Increase in Nr. Of Timesteps: $O(10^3\text{-}10^5)$ [MPI Limiting]

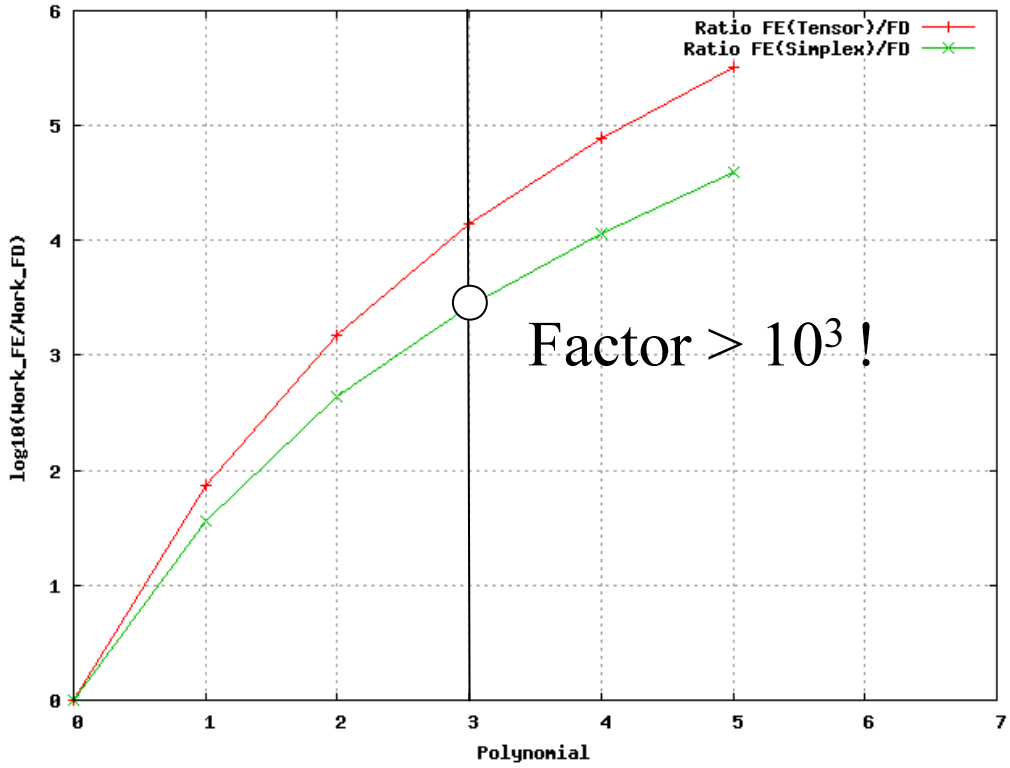
- Exascale Machines
  - Massively Parallel
  - Energy Consumption ~ Access To Memory
  - Motto: `Flops Are Free, Memory is Expensive'

- → Need **High Order Schemes With:**
  - **Minimal Memory Access**
  - **Minimal Transfer Between Domains/MPI Nodes**

Worries About Numerics…

- Higher Order Schemes: Finite Difference Methods (FDMs) Orders of Magnitude Faster Than FEM/DGMs
  - ➔ Revisit FDMs

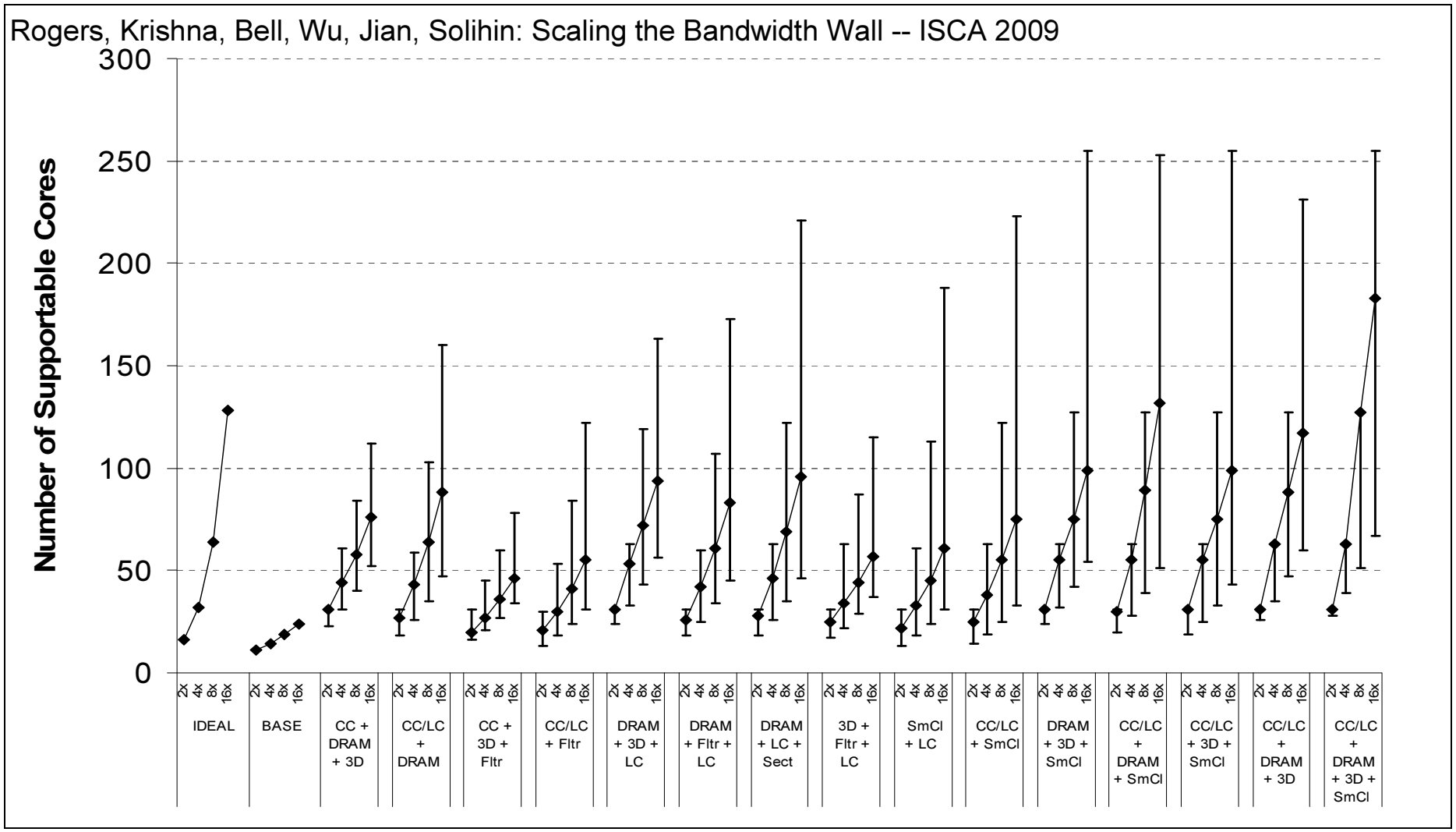
log(CPU_FEM/CPU_FDM)



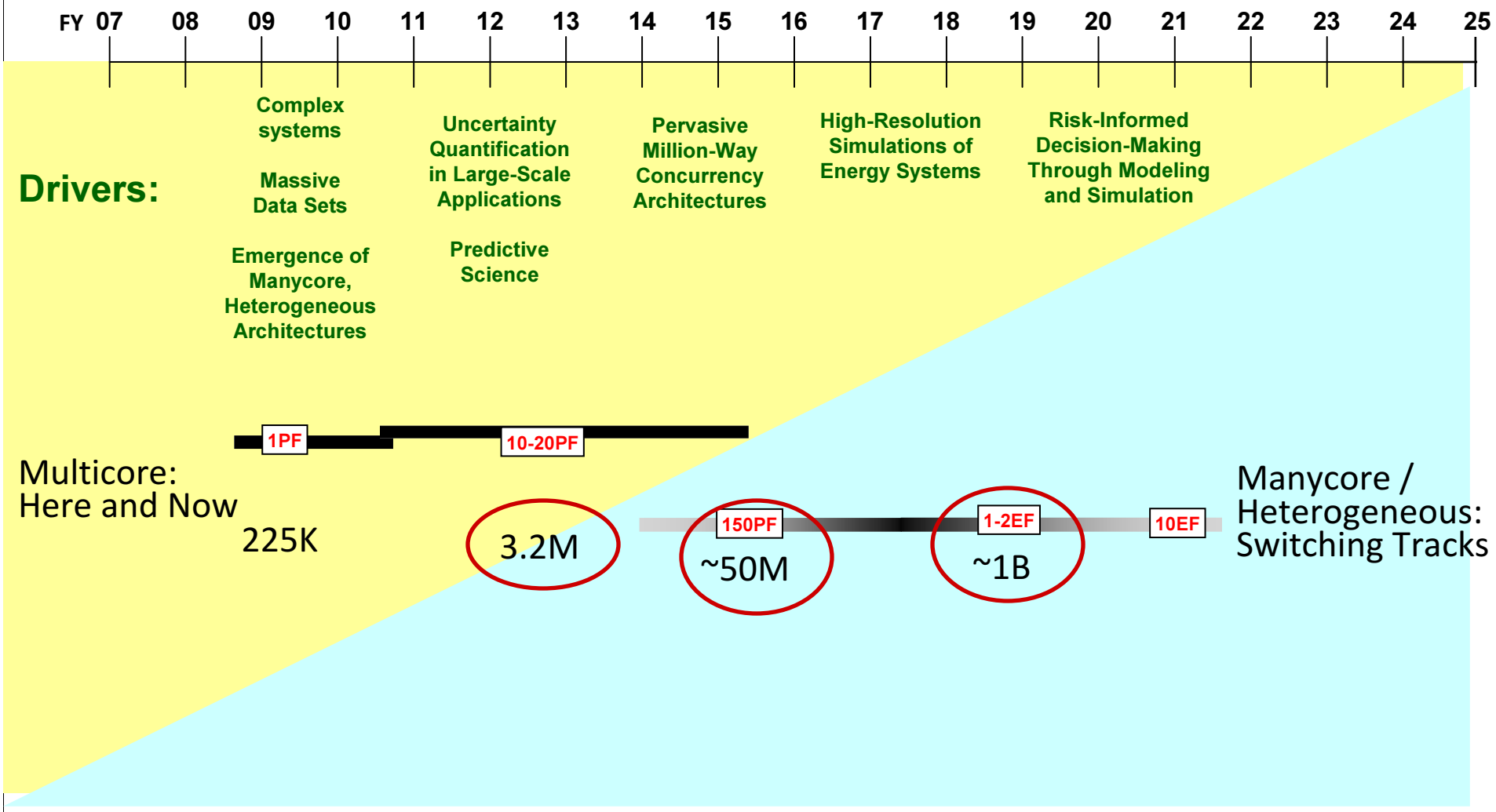
Worries About Hardware…

- Advances in Hardware Uneven
- CPUs > RAM > Network
- ➔ Red Shift ➔ Hardware Crisis
- CPUs/GPUs: Transfer Rate to Memory Limit Performance
  - Can Predict Speed By Just Counting Memory Access
  - `FLOPS Are Free'
- Even Worse for MPI/Network Transfer

How Many Cores ?



Strategic Planning:
DoE Applied Mathematics Research



Foreseeable Architectures

- Many (!) Low Energy Consumption Cores
- Simple CPU Architecture
- Low Number of Registers
- Access to Memory Expensive
- Access to Off-Board Memory/CPUs Really Expensive

- ➔ Minimize Memory Access, Recompute If Needed

Foreseeable Architectures

- Many (!) Low Energy Consumption Cores
- Simple CPU Architecture
- Low Number of Registers
- Access to Memory Expensive
- Access to Off-Board Memory/CPUs Really Expensive

- ➔ Minimize Memory Access, Recompute If Needed
- **New Criterion: Max Accuracy for Min Memory Access**

Memory Limit

Consequences of Memory Limit

- Motto: FLOPS Are Free, Memory Is Expensive
- Need Maximum Reuse of Variables
- Chunky `All in One' Loops
- Not Favourable to General Purpose Codes
- FEM-FCT: Factor 3 in Speedup Due to `Compacting'

Navier-Stokes + Thermal + Boussinesq

Conservation of Mass, Momentum and Energy:

$$\frac{1}{c^2} p_{,t} + \nabla \cdot \mathbf{v} = 0$$

$$\rho \mathbf{v}_{,t} + \rho \mathbf{v} \cdot \nabla \mathbf{v} + \nabla p = \nabla \mu \nabla \mathbf{v} + \rho g + \beta g \left(T - T_0 \right) + \mathrm{S_v}$$

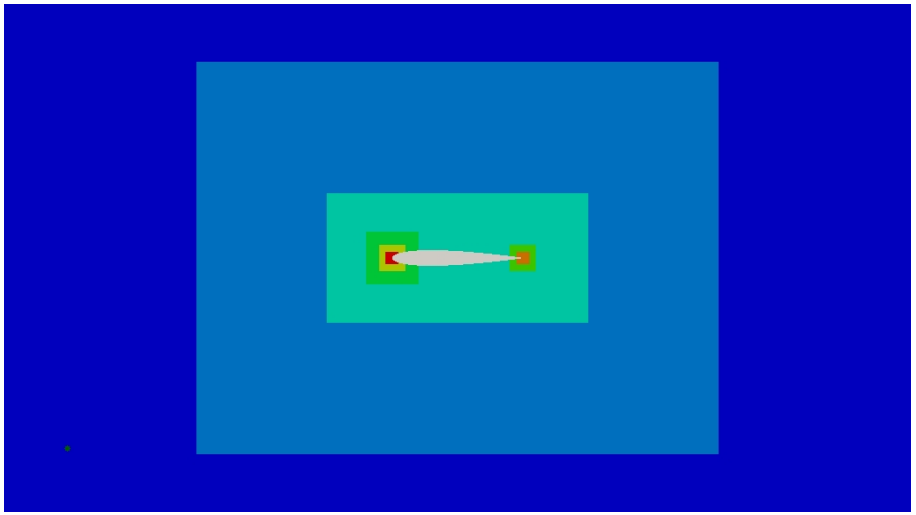
$$\rho c_p T_{,t} + \rho c_p \mathbf{v} \cdot \nabla T = \nabla \lambda \nabla T + \mathrm{S_T}$$

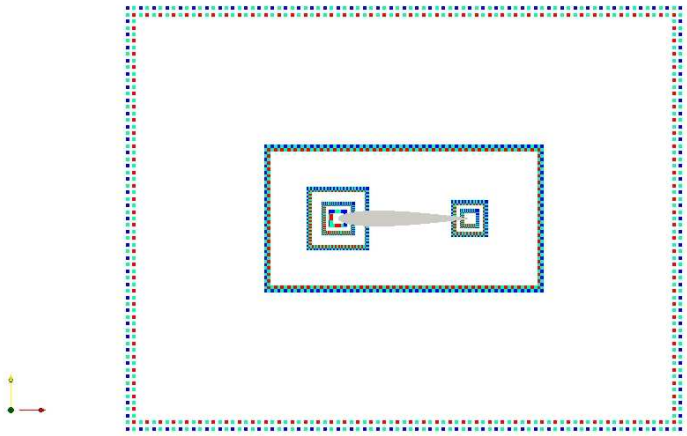
ρ: Density	**v**: Velocity	p: Pressure
μ: Viscosity	λ: Conductivity	g: Gravity
c: Speed of Sound	β: Thermal Expansion Coeff.	

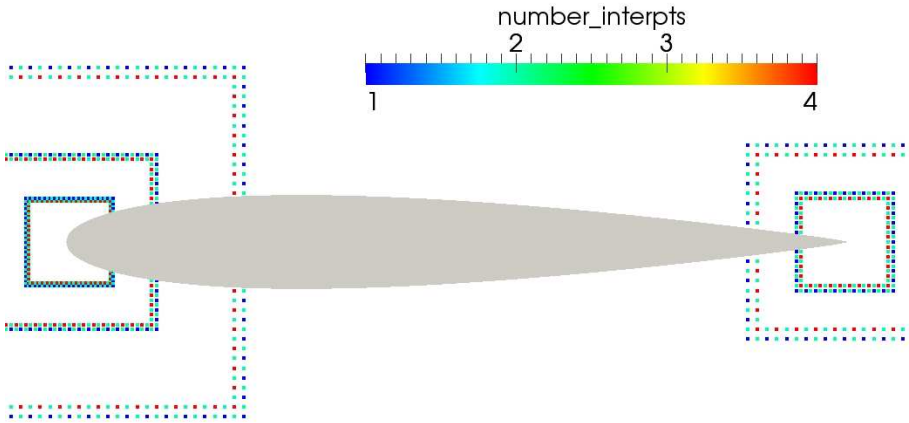
Main Loop: This is Navier-Stokes…

```
c
c     -----as we assume nhalo halos:
c
      ipoi0=npoix*npoiy*       nhalo +npoix*nhalo+nhalo+1
      ipoi1=npoix*npoiy*(npoiz-nhalo)-npoix*nhalo-nhalo
c
c     -----outer loop over the dimensions
c
      do 1000 idimn=1,ndimn
c
      c05gx=c00
      c05gy=c00
      c05gz=c00
c
c     -----jumps
c
          if(idimn.eq.1) then
             ijum2=-2
             ijum1=-1
             ijup1= 1
             ijup2= 2
             c05gx= c05gr
             npoid= npoix
      elseif(idimn.eq.2) then
             ijum2=-nx2
             ijum1=-nx
             ijup1= nx
             ijup2= nx2
             c05gy= c05gr
             npoid= npoiy
      elseif(idimn.eq.3) then
             ijum2=-nxny2
             ijum1=-nxny
             ijup1= nxny
             ijup2= nxny2
             c05gz= c05gr
             npoid= npoiz
       endif
c
c     -----advective velocity is in:
c
      iuadv=idimn+1
c
c     -----loop over the points, computing the rhs
c
      do 1200 ipoin=ipoi0,ipoi1
c
c     -----points needed
c
      ipxm2=ipoin+ijum2
      ipxm1=ipoin+ijum1
      ipxp1=ipoin+ijup1
      ipxp2=ipoin+ijup2
c
c     -----variables needed
c
      prxm2=unkno(    1,ipxm2)
      uvxm2=unkno(    2,ipxm2)
      vvxm2=unkno(    3,ipxm2)
      wvxm2=unkno(    4,ipxm2)
      texm2=unkno(    5,ipxm2)
      prxm1=unkno(    1,ipxm1)
      uvxm1=unkno(    2,ipxm1)
      vvxm1=unkno(    3,ipxm1)
      wvxm1=unkno(    4,ipxm1)
      texm1=unkno(    5,ipxm1)
      uadm1=unkno(iuadv,ipxm1)
      pr000=unkno(    1,ipoin)
      uv000=unkno(    2,ipoin)
      vv000=unkno(    3,ipoin)
      wv000=unkno(    4,ipoin)
      te000=unkno(    5,ipoin)
      visco=unkno(    6,ipoin)
      condu=unkno(    7,ipoin)
      ua000=unkno(iuadv,ipoin)
      prxp1=unkno(    1,ipxp1)
      uvxp1=unkno(    2,ipxp1)
      vvxp1=unkno(    3,ipxp1)
      wvxp1=unkno(    4,ipxp1)
      texp1=unkno(    5,ipxp1)
      uadp1=unkno(iuadv,ipxp1)
      prxp2=unkno(    1,ipxp2)
      uvxp2=unkno(    2,ipxp2)
      vvxp2=unkno(    3,ipxp2)
      wvxp2=unkno(    4,ipxp2)
      texp2=unkno(    5,ipxp2)
c
      vi0la=visco*clap1
      co0la=condu*clap1
      refac=reyf0/max(epsvi,visco)
      pefac=c_p*visco/max(epsco,condu)
c
c     -----edge 1: x_{i-1}:x_{i}
c
      uved1 = uadm1+ua000
      vmax1 = max(abs(uadm1),abs(ua000))
      dte1p = cpres/(csoun+vmax1)
      dte1v = dte1p*max(c00,min(c10,refac*vmax1-c10))
c     dte1v = cvelo*vmax1*max(c00,min(c10,refac*vmax1-c10))
c
c     -----edge 2: x_{i}:x_{i+1}
c
      uved2 = ua000+uadp1
      vmax2 = max(abs(ua000),abs(uadp1))
      dte2p = cpres/(csoun+vmax2)
      dte2v = dte2p*max(c00,min(c10,refac*vmax2-c10))
c     dte2v = cvelo*vmax2*max(c00,min(c10,refac*vmax2-c10))
c
      rhspo(1,ipoin)=rhspo(1,ipoin)
     &          +c05di*(uved2-uved1)                                  ! div
     &          +cadvp*(dte2p*(-prxp2+c30*(prxp1-pr000)+prxm1)
     &                 -dte1p*(-prxp1+c30*(pr000-prxm1)+prxm2))      ! AV
      rhspo(2,ipoin)=rhspo(2,ipoin)
     &          +cadve*(uved2*(uv000+uvxp1)-uved1*(uvxm1+uv000))      ! adv
     &          +c05gx*(prxp1-prxm1)                                  ! grdp
     &          +cadvv*(dte2v*(-uvxp2+c30*(uvxp1-uv000)+uvxm1)
     &                 -dte1v*(-uvxp1+c30*(uv000-uvxm1)+uvxm2))      ! AV
     &          +vi0la*(uvxp1+cm2*uv000+uvxm1)                        ! lapl
      rhspo(3,ipoin)=rhspo(3,ipoin)
     &          +cadve*(uved2*(vv000+vvxp1)-uved1*(vvxm1+vv000))      ! adv
     &          +c05gy*(prxp1-prxm1)                                  ! grdp
     &          +cadvv*(dte2v*(-vvxp2+c30*(vvxp1-vv000)+vvxm1)
     &                 -dte1v*(-vvxp1+c30*(vv000-vvxm1)+vvxm2))      ! AV
     &          +vi0la*(vvxp1+cm2*vv000+vvxm1)                        ! lapl
      rhspo(4,ipoin)=rhspo(4,ipoin)
     &          +cadve*(uved2*(wv000+wvxp1)-uved1*(wvxm1+wv000))      ! adv
     &          +c05gz*(prxp1-prxm1)                                  ! grdp
     &          +cadvv*(dte2v*(-wvxp2+c30*(wvxp1-wv000)+wvxm1)
     &                 -dte1v*(-wvxp1+c30*(wv000-wvxm1)+wvxm2))      ! AV
     &          +vi0la*(wvxp1+cm2*wv000+wvxm1)                        ! lapl
      rhspo(5,ipoin)=rhspo(5,ipoin)
     &          +cadte*(uved2*(te000+texp1)-uved1*(texm1+te000))      ! adv
     &     +cadvt*pefac*(dte2v*(-texp2+c30*(texp1-te000)+texm1)
     &                 -dte1v*(-texp1+c30*(te000-texm1)+texm2))      ! AV
     &          +co0la*(texp1+cm2*te000+texm1)                        ! lapl
c
 1200 continue
c
c     -----end of outer loop over the dimensions
c
 1000 continue
```

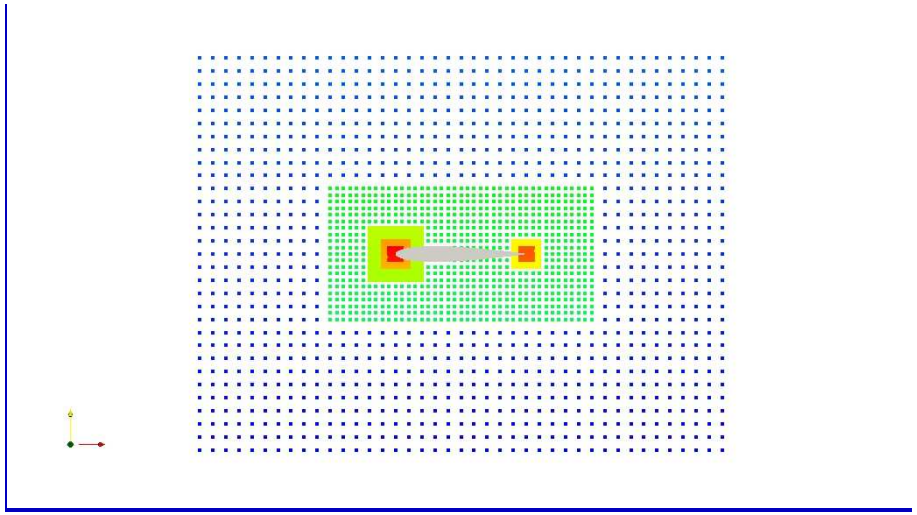
Example: NACA 0012 (1)

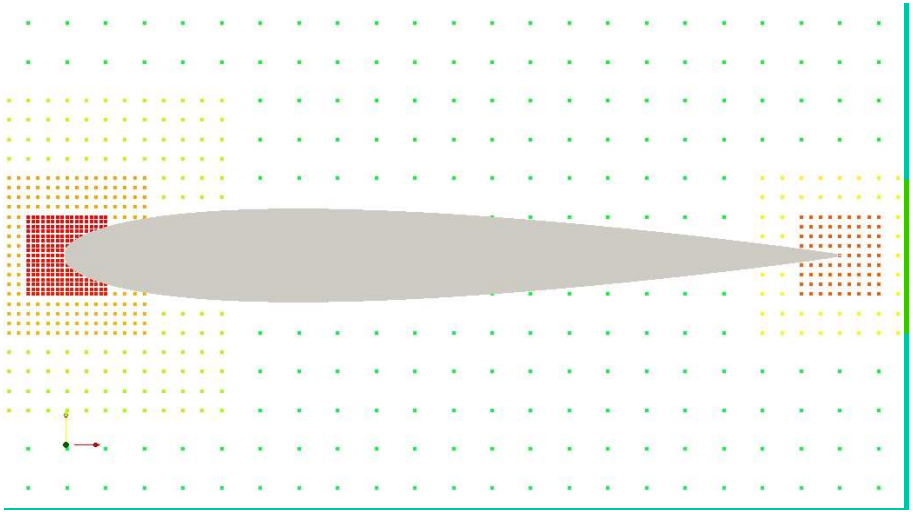


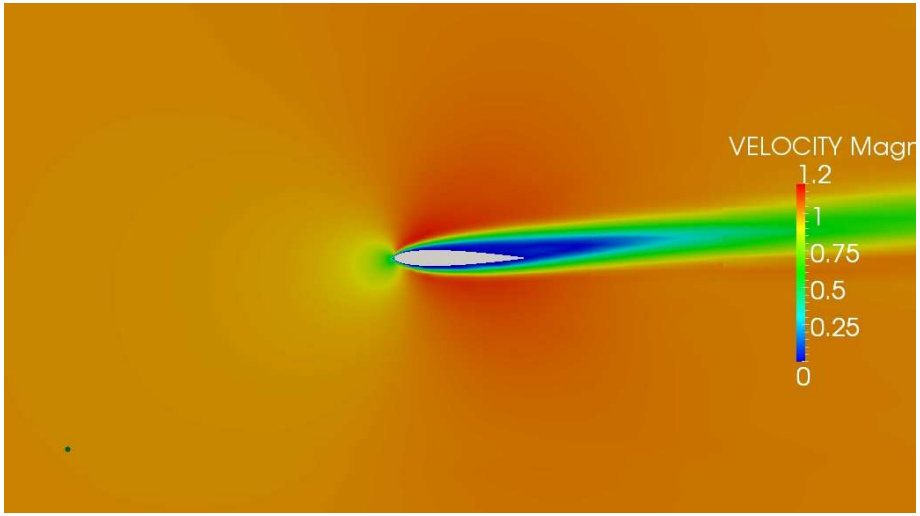


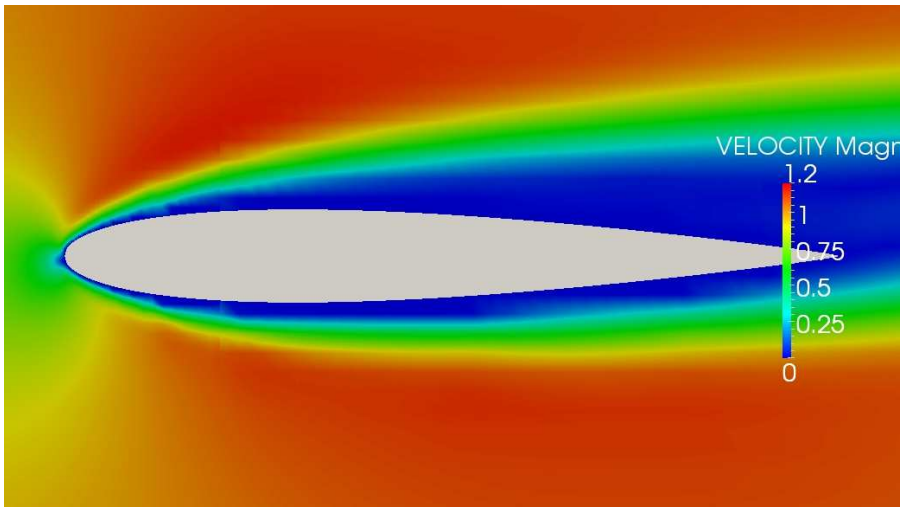


Example: NACA 0012 (2)

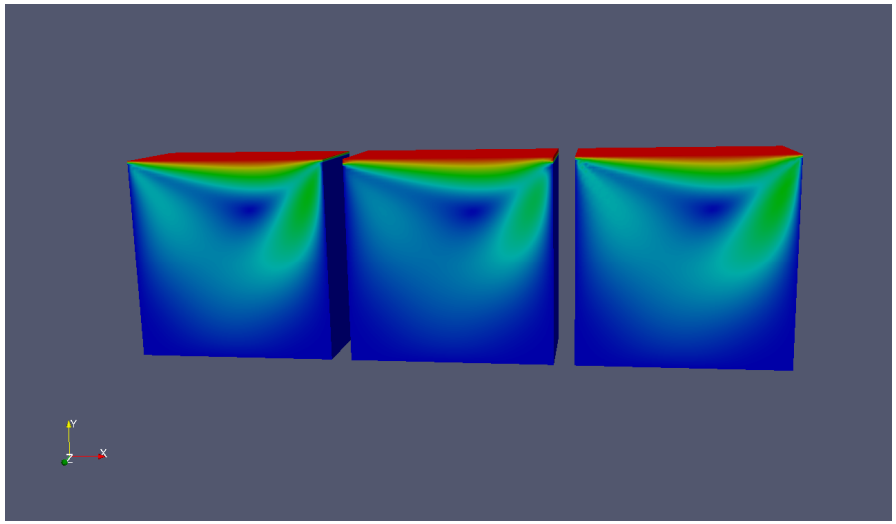


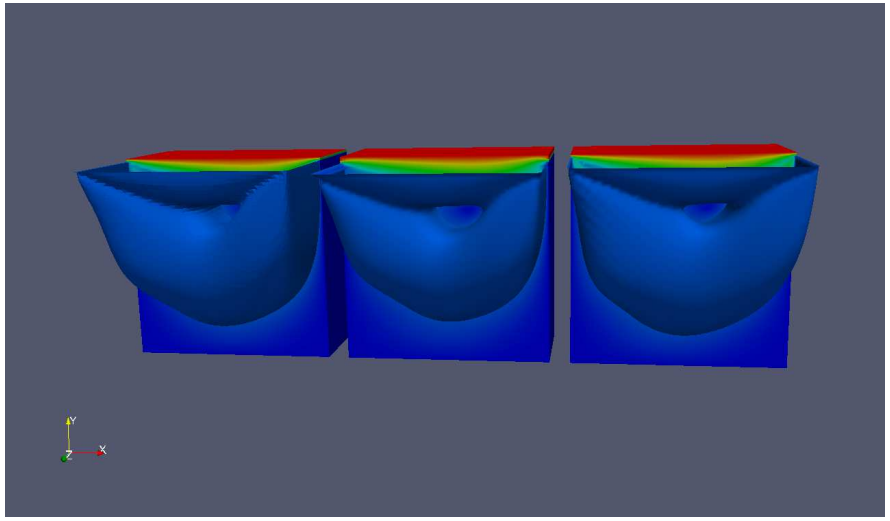






FEM / FDSOL / LBM





Lid-Driven Cavity

- From Operation Counts:
  - Total Flops : $O(7.5 \cdot 10^{10})$
  - Total RAMio : $O(2.3 \cdot 10^{10})$ R*8 $= O(1.8 \cdot 10^{11})$ Bytes

- Expected Time from RAMio
  - Xeon : O(30.0) sec
  - Tesla : O(1.8) sec

Lid-Driven Cavity

- Timings [sec]

isolv	CPU (1)	CPU(8)	GPU
21	34.32	7.46	7.56
22	59.41	9.66	5.06
23	59.21	8.96	3.13
24	38.41	8.95	3.09
31	46.71	9.07	12.78
32	46.23	9.52	4.35

➔ Within Factor of 1:2 of RAM i/o Predicted Values !

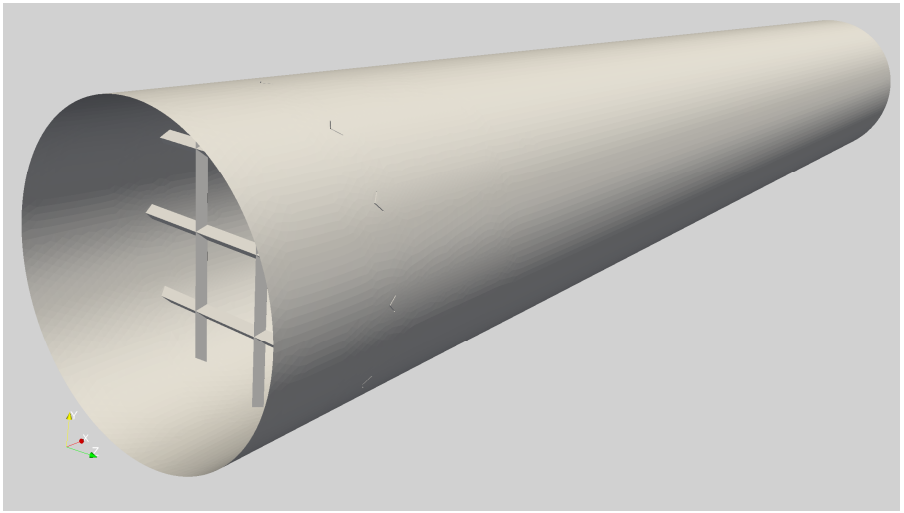
Lid-Driven Cavity

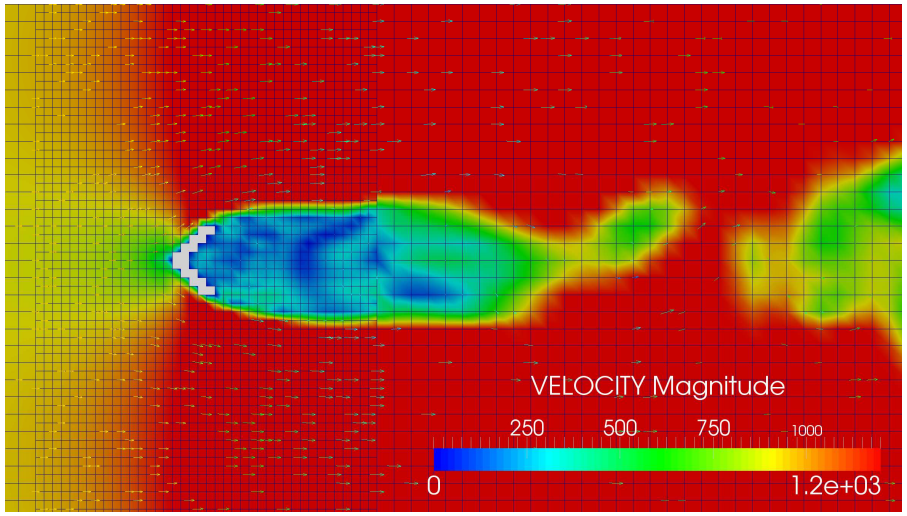
- Speed [sec/pt/step • 10^8]

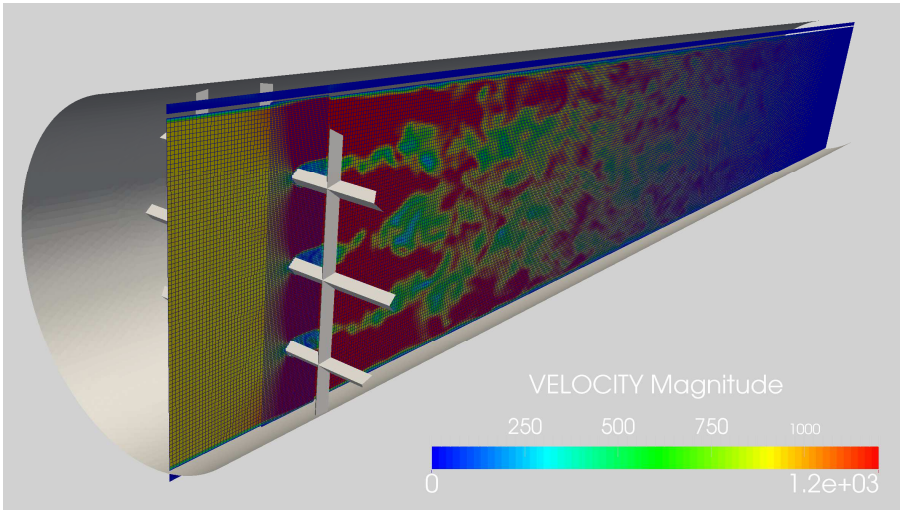
isolv	CPU(1)	CPU(8)	GPU
21	30.30	6.57	6.66
24	33.80	7.89	2.72
31	41.20	7.99	11.30
32	40.70	8.39	3.83

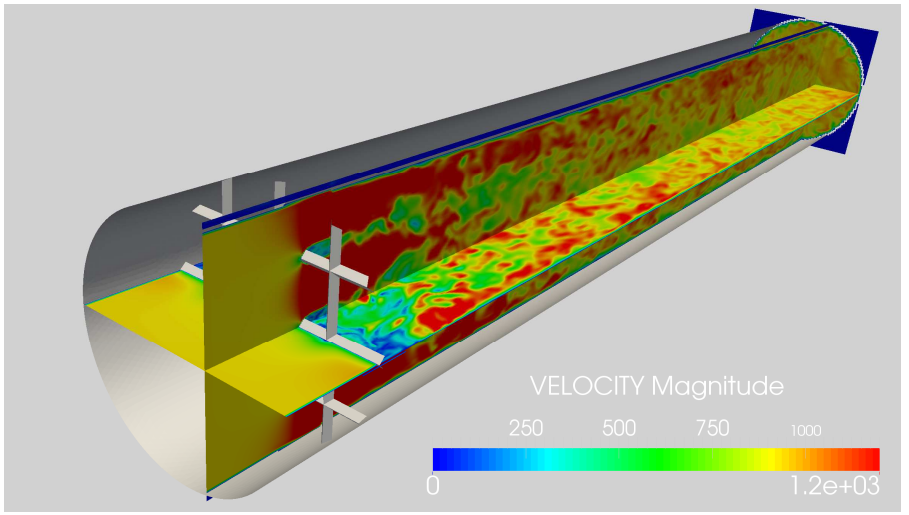
2.45 11.90 26.11 Mpts/sec/step

Flow In Small Windtunnel









Minimal Memory Access Loops

- Consider Laplacian, FDM, 1-D, 2nd Order
- Option 1: 3 F, 1 S

```
do ip=ipoi0,ipoi1
  rhs(ip)=ch2*(unk(ip-1)-2*unk(ip)+unk(ip+1))
enddo
```

- Option 2: 1 F, 1 S

```
un0=unk(ipoi0-1)
up1=unk(ipoi0 )
do ip=ipoi0,ipoi1
  um1=un0
  un0=up1
  up1=unk(ip+1)
  rhs(ip)=ch2*(um1-2*un0+up1)
enddo
```

Minimal Memory Access Loops

- General Framework For Systems (e.g. Euler)

```
   Fetch   : um1(:),uno(:),up1(:),up2(:)
   Compute: fl2(:)
 do ip=ipoi0,ipoi1
   Transcribe: um2=um1; um1=un0; un0=up1, up1=up2
   Transcribe: fl1=fl2
   Fetch      : up2(:)
   Compute    : fl2(:)
   Save       : rhs(ip)=ch*(fl2-fl1)
 enddo
```

- 1 Fetch and 1 Store for Every (Final) RHS (!)
- For Any Order of FD Stencil (!)

Minimal Memory Access Loops

- General Framework For Systems (e.g. Euler)

```
   Fetch   : um1(:),uno(:),up1(:),up2(:)
   Compute: fl2(:)
do ip=ipoi0,ipoi1
   Transcribe: um2=um1; um1=un0; un0=up1, up1=up2
   Transcribe: fl1=fl2
   Fetch      : up2(:)
   Compute    : fl2(:)
   Save       : rhs(ip)=ch*(fl2-fl1)
enddo
```

In Registers

- 1 Fetch and 1 Store for Every (Final) RHS (!)
- For Any Order of FD Stencil (!)

Minimal Memory Access Loops

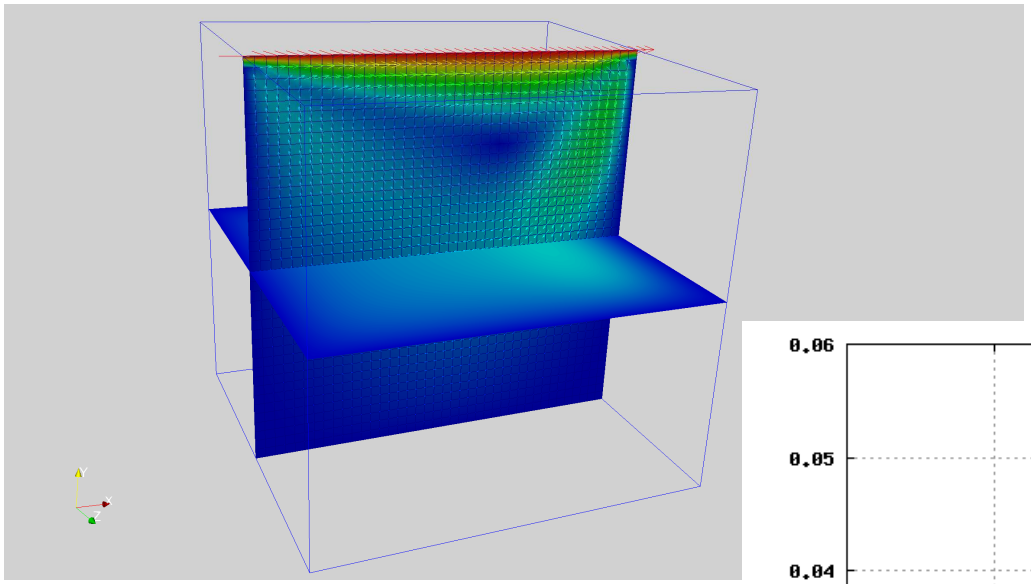
- 3-D: Need to Reorder Data ➔ Additional Fetches/Stores

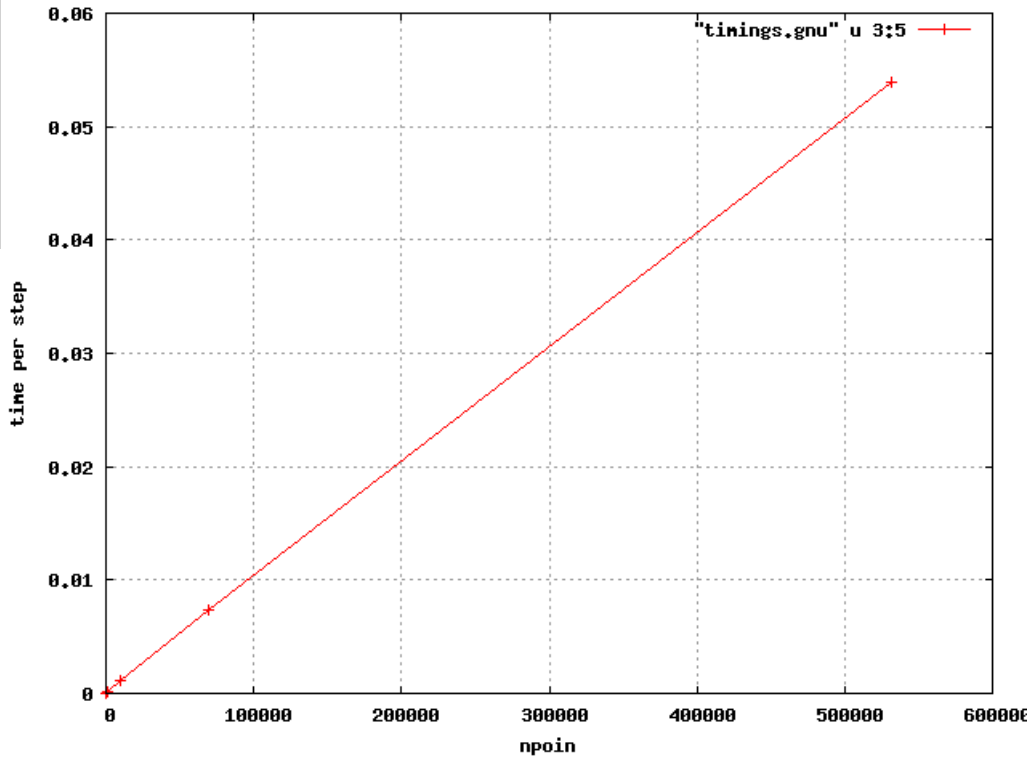
`Reorient in y,z, Store in uy,uz`	:	2 F, 2 S
`Do x`	:	1 F, 1 S
`Reorient rhs`	:	1 F, 1 S
`Do y`	:	2 F, 1 S
`Reorient rhs`	:	1 F, 1 S
`Do z`	:	2 F, 1 S
`Reorient rhs`	:	1 F, 1 S
`Total`	:	10 F, 8 S

- Regardless of Order of FD Stencil
- Conventional 4th Order FD Stencil, Split: 17 F, 3 S

MPI Limit

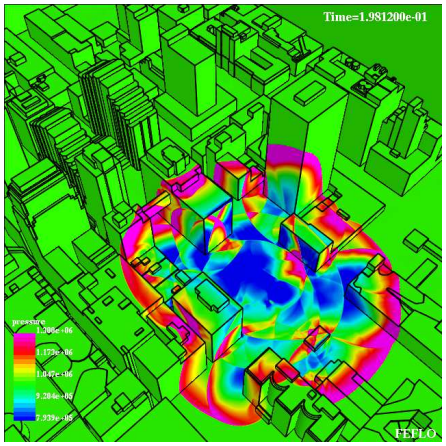
Lid-Driven Cavity, FDFLO Timings





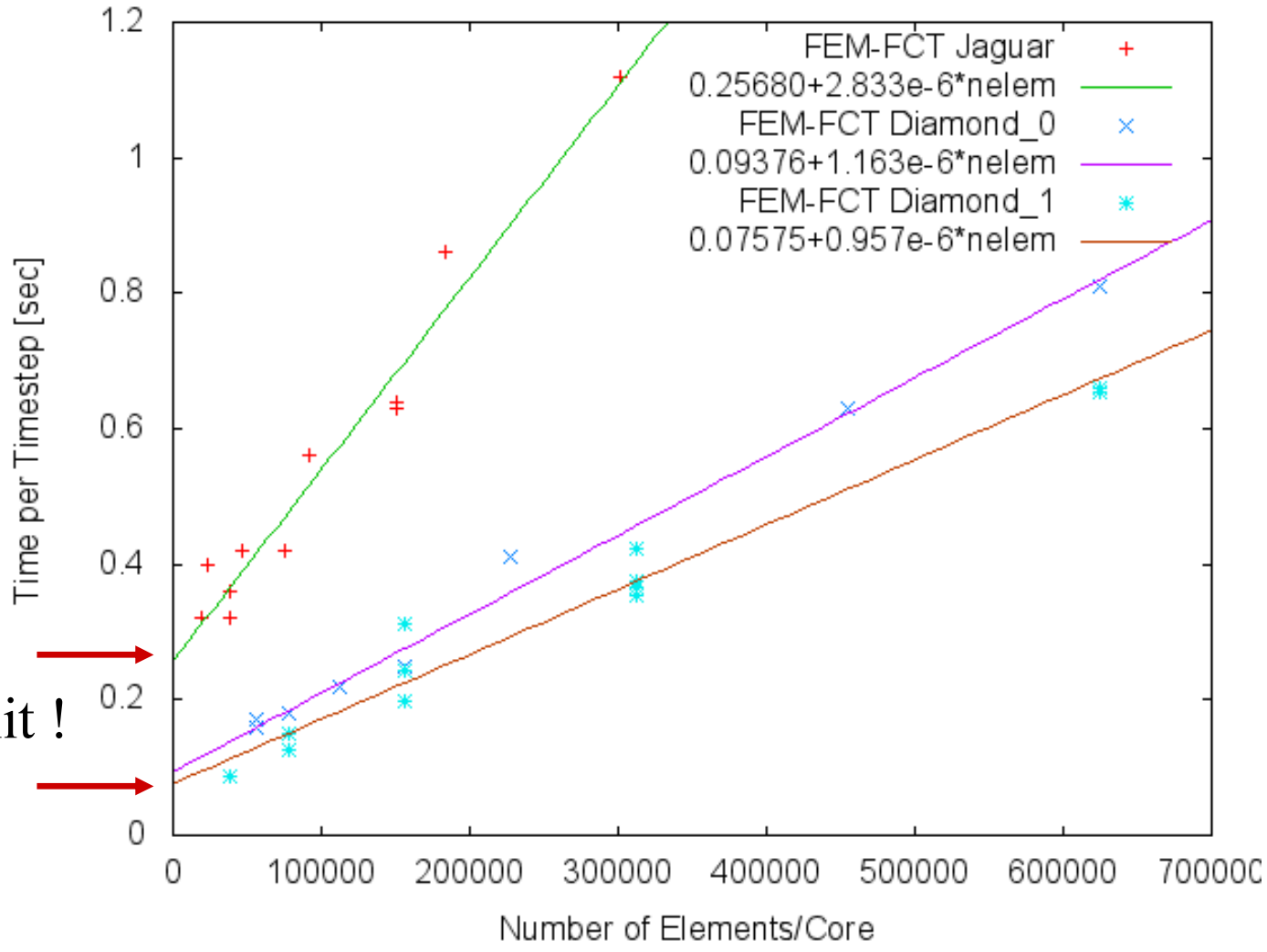
Laptop, 1 Core Running

FEM-FCT Timings



Nelem>1Bels
Nproc=O(50K)

MPI Limit !



MPI Limits (1)

- MPI Communication Overheads Limit Domain Size

- Fast Codes: O(100 Kpts)
- Disparity CPU/Communication/Rest Increasing

- ➔ Number Will Not Decrease (!)
- ➔ Limits Speed/Useful Nr. Of Processors

MPI Limits (2)

- Consequences:
- 1: Execution Time > T(Limit)
  - Time/Step/Pt Fixed If Processor Speed Not Increasing [O(0.1sec)]
- 2: Number of Useful MPI-Nodes/Cores < N(Limit)
  - Problem Size/100Kpts

- Corollaries:
- 1: LES/DNS Runs Will Remain Expensive
  - Large (10^7-10^8) Number of Timesteps to Propagate Vortices
  - 1-2 Weeks on Exaflop Machine [Wind-Tunnels: Rejoice !]
- 2: Real-Time Computing Requires Alternative Approaches
  - Pre-Compute and Interpolate/Reduce (Eigen, POD, …)

MPI Options

- Reduce Communication Between Processors [1:2]
- Compress/Decompress Data Sent [1:8]

- Improved Domain Splitting Techniques
- Aim: Minimum Communication Passes [1:2]

- Renumber Domains To Map Communication/Hardware
- Aim: Minimize/ Avoid Intermediate Hops [1:5]

- Pre-Stored Communication Patterns [1:4]

- Replace MPI by Other Libraries [1:2]