Petascale Adaptive Computational Fluid Dynamics

PI: Kenneth Jansen, University of Colorado Boulder

Co-PIs:

- Michel Rasquin, ALCF
- Mark Shephard and Onkar Sahni, RPI

MIRA Community Conference

March 6, 2013

Argonne NATIONAL LABORATORY

University of Colorado Boulder

Rensselaer

Outline

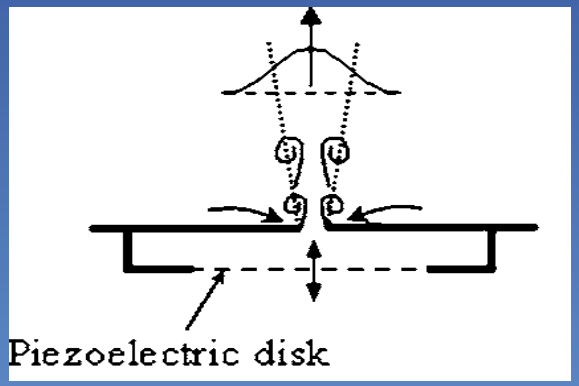
- Short description of science
- Numerical method and parallelization
- Scalability on MIRA
  - 512K MPI processes
  - Preliminary analysis
- Improvements

Aerodynamics and Active Flow Control

- Active Flow control (AFC):

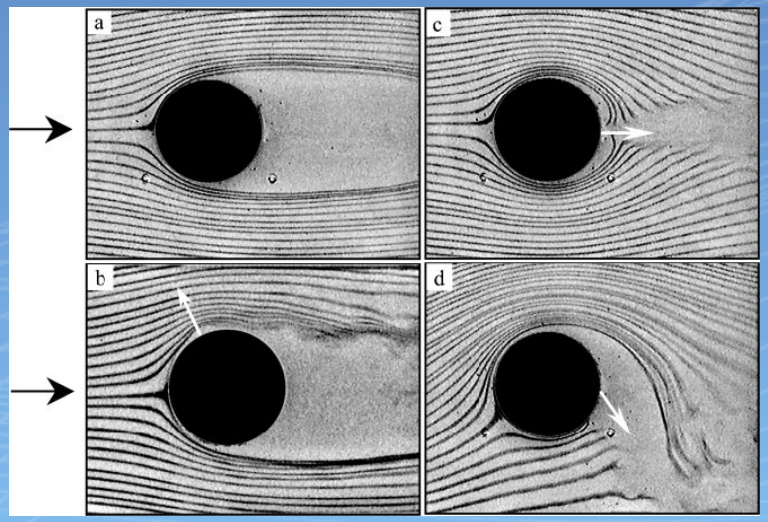
  Technology thrust aiming at altering a natural flow field into a more desirable state by placing actuators at key locations in order to improve efficiency or performance

  - Can prevent/promote flow separation
  - Can suppress/enhance turbulence



Sketch of a synthetic jet

- Potential benefits:
  - Drag reduction
  - Lift enhancement
  - Mixing augmentation
  - Noise reduction



EXP: Smoke flow viz. (baseline and actuated cases)
Courtesy *M. Amitay et. al.*

Motivations for Active Flow Control

- Airline industry:
  - Improve aerodynamic efficiency of conventional control surfaces
    - Wing + flap
    - Stabilizer + Rudder (vertical tail)
  - Reduce size and weight
  - Reduce biggest expense, i.e. fuel consumption

- Wind turbine industry:
  - Alleviate unsteady aerodynamic loading (gust)
    - Force flow reattachment on the wind turbine blades
    - Reduce maintenance cost
  - Optimize turbine performance in varying weather conditions
    - Low wind period and 3-D aero-shaping
  - Increase electricity production

PHASTA CFD Solver

- Massively parallel MPI Navier-Stokes flow solver
- Models compressible or incompressible, Laminar or turbulent, steady or unsteady flows
- 3D finite element discretization in space => Complex geometries
- fully implicit in time => Δt governed by the physics, not by the numerical scheme
- Mesh adapativity => - Grid matches physical scale
  - Anisotropic and transient
- Used to model air flow over airplane wing, blood flow, two phase flow, etc
- Scaled to 512K cores (Argonne – 2013)
  - 88% scaling factor with a 3.3B element mesh

Current approach – NS flow solver

- Implicit non-linear FEM solver with two phases of computation:
  - **Equation formation** (*Eqn. form.*) – <u>depends on *elements*</u>

PDE/strong form – $\mathscr{L}\boldsymbol{Y} = \mathscr{S}$

Stab FEM – $\int_{\Omega} (\cdot)^h \, d\Omega + \int_{\Gamma} (\cdot)^h \, d\Gamma$

Quadrature – $\sum_{GP}^{vol} c_{GP} (\cdot)^h + \sum_{GP}^{bdy} d_{GP} (\cdot)^h$

Assembly – $\mathbf{Ax} = \mathbf{b}$

  - **Equation solution** (*Eqn. sol.*) – <u>depends on *degrees-of-freedom*</u> (*dofs*):

Krylov Iterative solver (GMRES)

$\mathbf{Ax} = \mathbf{b}$

$\mathbf{p} = \mathbf{b}$

while

$\quad \mathbf{q} = \mathbf{Ap}$

$\quad$ Orthonormalize $\mathbf{q}$

...

Current approach – Parallelization

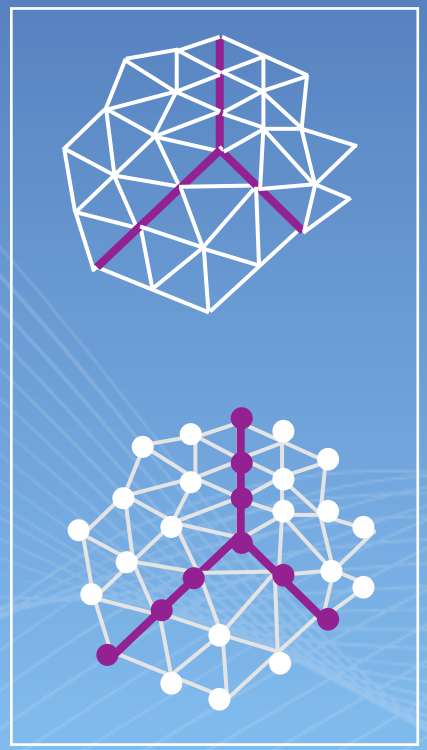
- ✓ Parallel strategy:
  - ➢ Domain decomposition approach based on the elements
  - ➢ Both compute stages operate off the same mesh partition
  - ➢ Partition defines inter-part relations (part-to-part comm.)

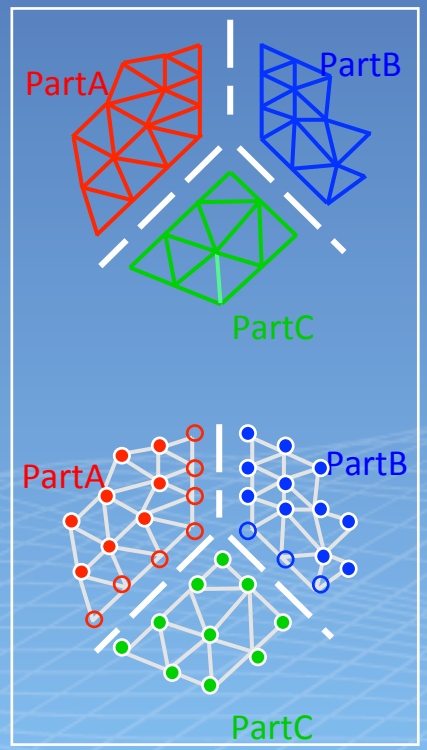
Eqn. form.

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

Eqn. sol.

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$





Locally, *incomplete* values (in ***b***, ***A***, ***q***, etc.) for shared *dofs.*

Apply communications to *complete* values/entries (in ***b***, ***q*** only)

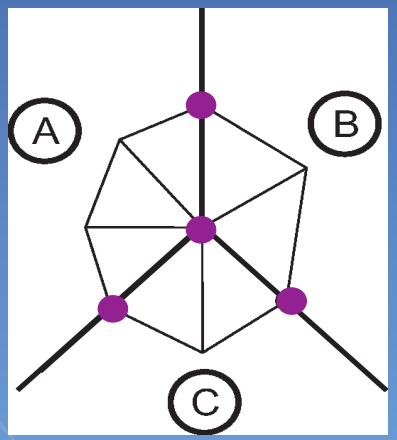
$\mathbf{b}$ during *Eqn. form.*

$\mathbf{q}$ during *Eqn. sol.*

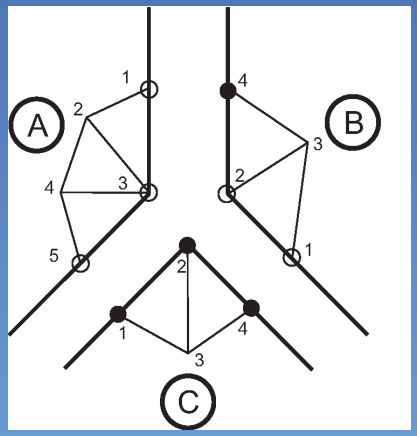
Current approach – Parallelization

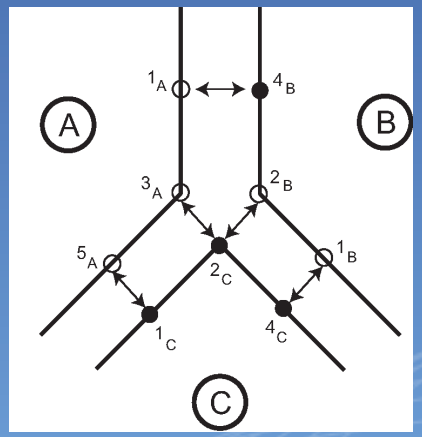
✓ Communications to *complete* values/entries and norms:

dofs are shared between parts

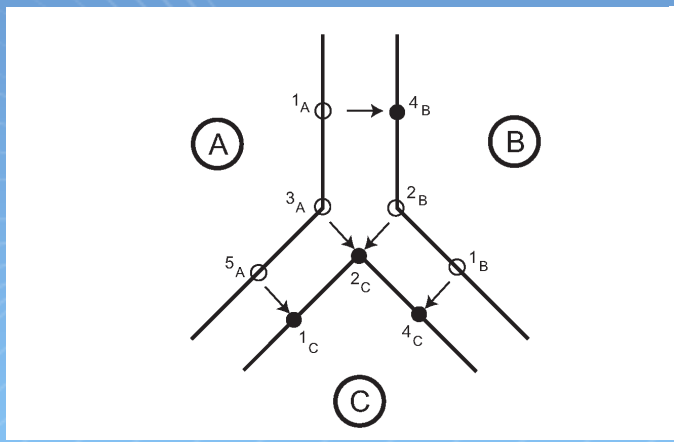


control relationship between images (solid dots indicate owner images)

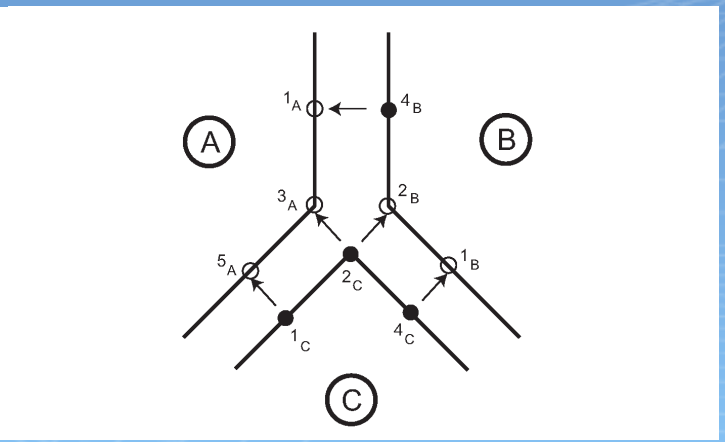




values accumulated on owners



values updated on to non-owners



*complete **b***

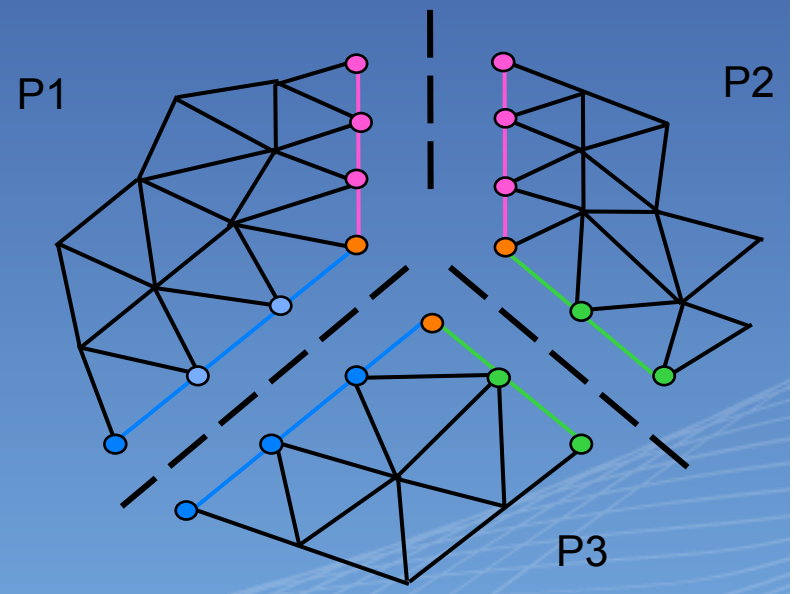
*complete **q***
*(for on-part **q**=**Ap**)*

on-part norm for q and all-reduce
(use complete q)

PHASTA Flow Solver Parallel Paradigm

- **Equation formation:**
  - O(40) peer-to-peer non-blocking comms
  - Overlapping comms with comp
  - Scales well on many machines

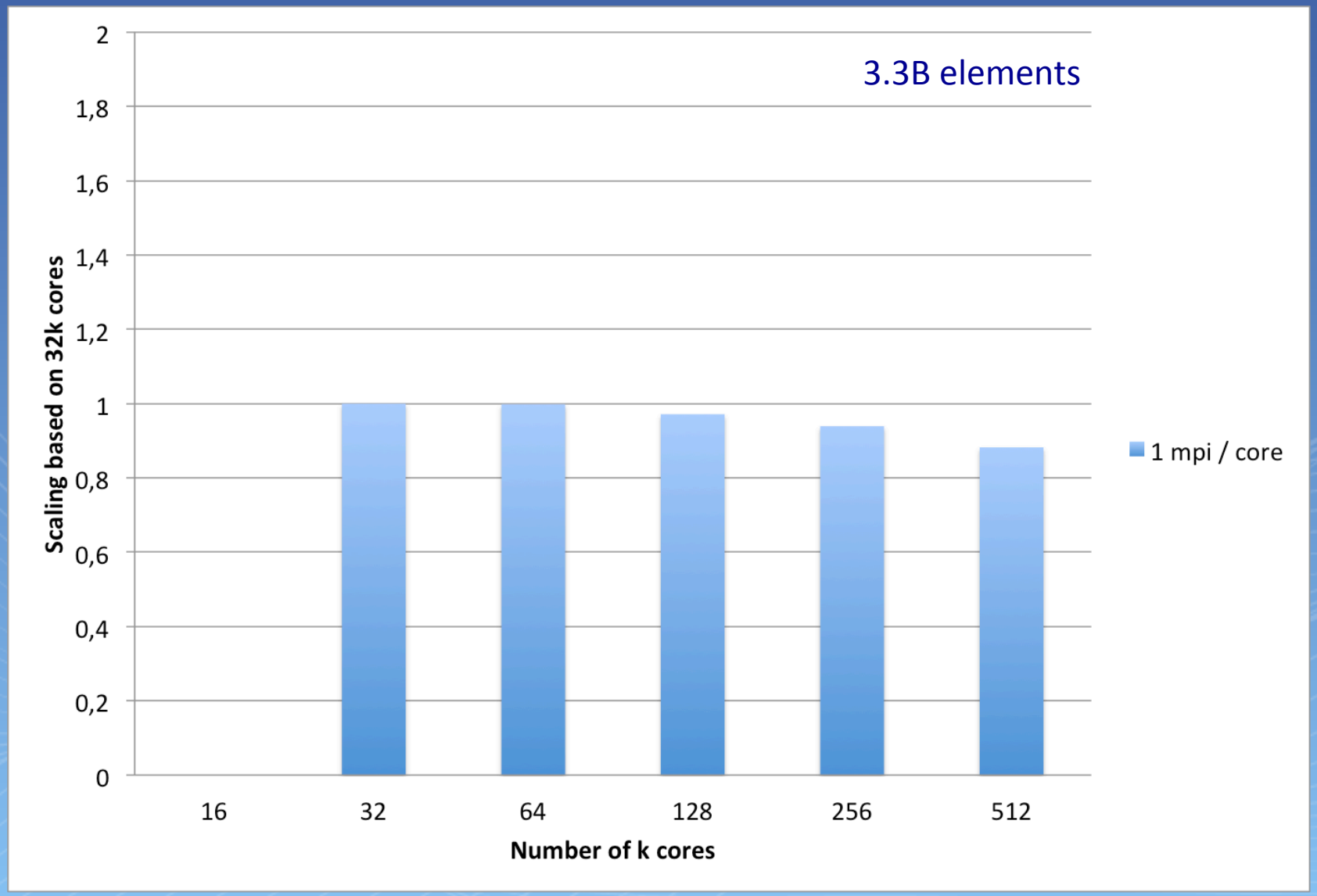
- **Implicit, iterative equation solution:**
  - Each Krylov vector is:
    - q=Ap (matrix-vector product)
    - Same peer-to-peer comm of q PLUS
    - Orthonormalize against prior vectors
    - REQUIRES NORMS => MPI_Allreduce



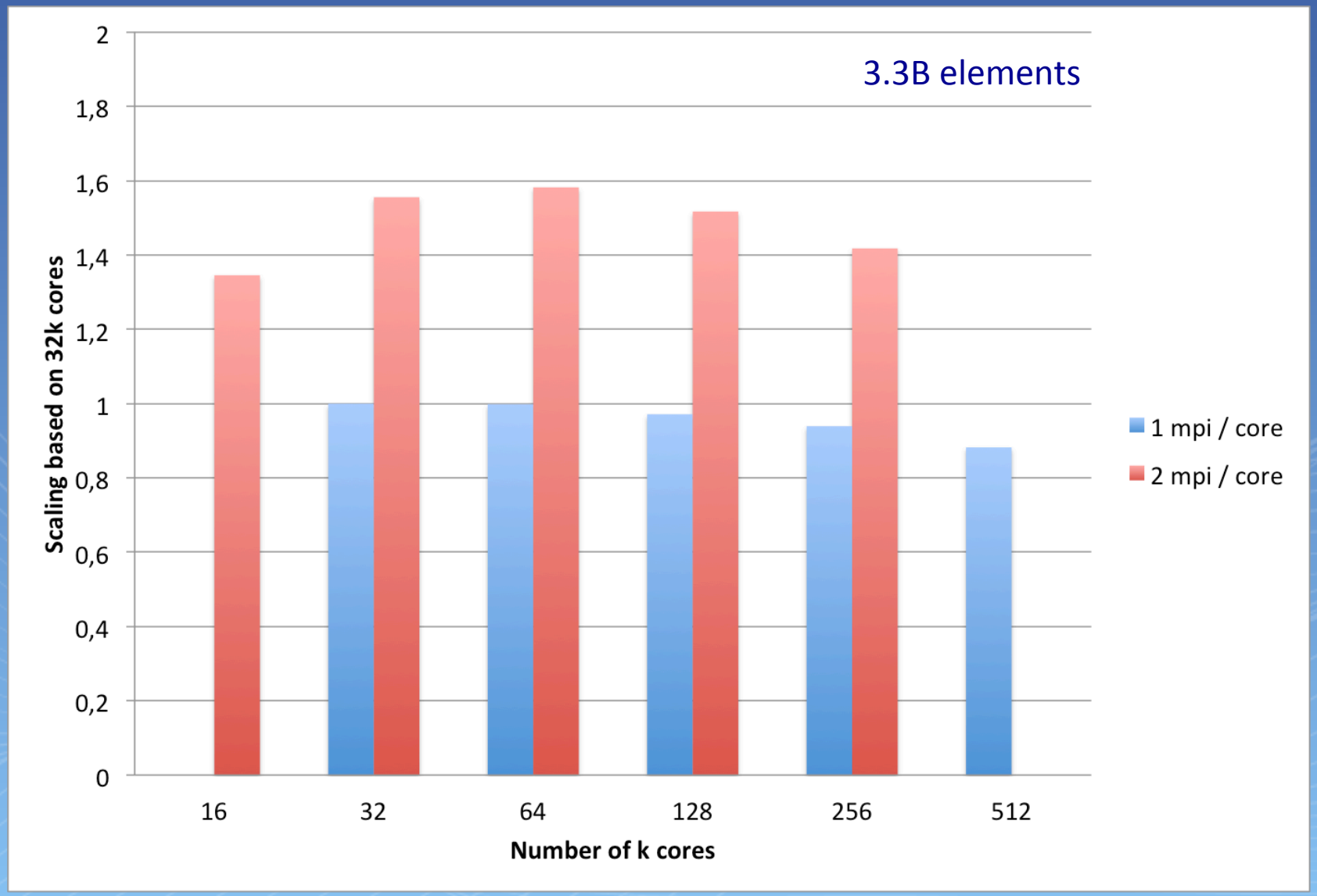
PHASTA Scaling on MIRA

K Nodes	K Cores	MPI process/core = part/core	K mesh parts = MPI process	Scaling factor (%)	Mira/Intrepid Ref: 40K nodes BG/P
2 (ref)	32	1	32	100.00	12.04
4	64	1	64	99.84	12.02
8	128	1	128	97.14	11.69
16	256	1	256	93.92	11.30
32	512	1	512	88.22	10.62
1	16	2	32	134.56	16.20
2	32	2	64	155.56	18.72
4	64	2	128	158.21	19.04
8	128	2	256	151.72	18.26
16	256	2	512	141.78	17.06
1	16	4	64	155.09	18.67
2	32	4	128	193.73	23.32

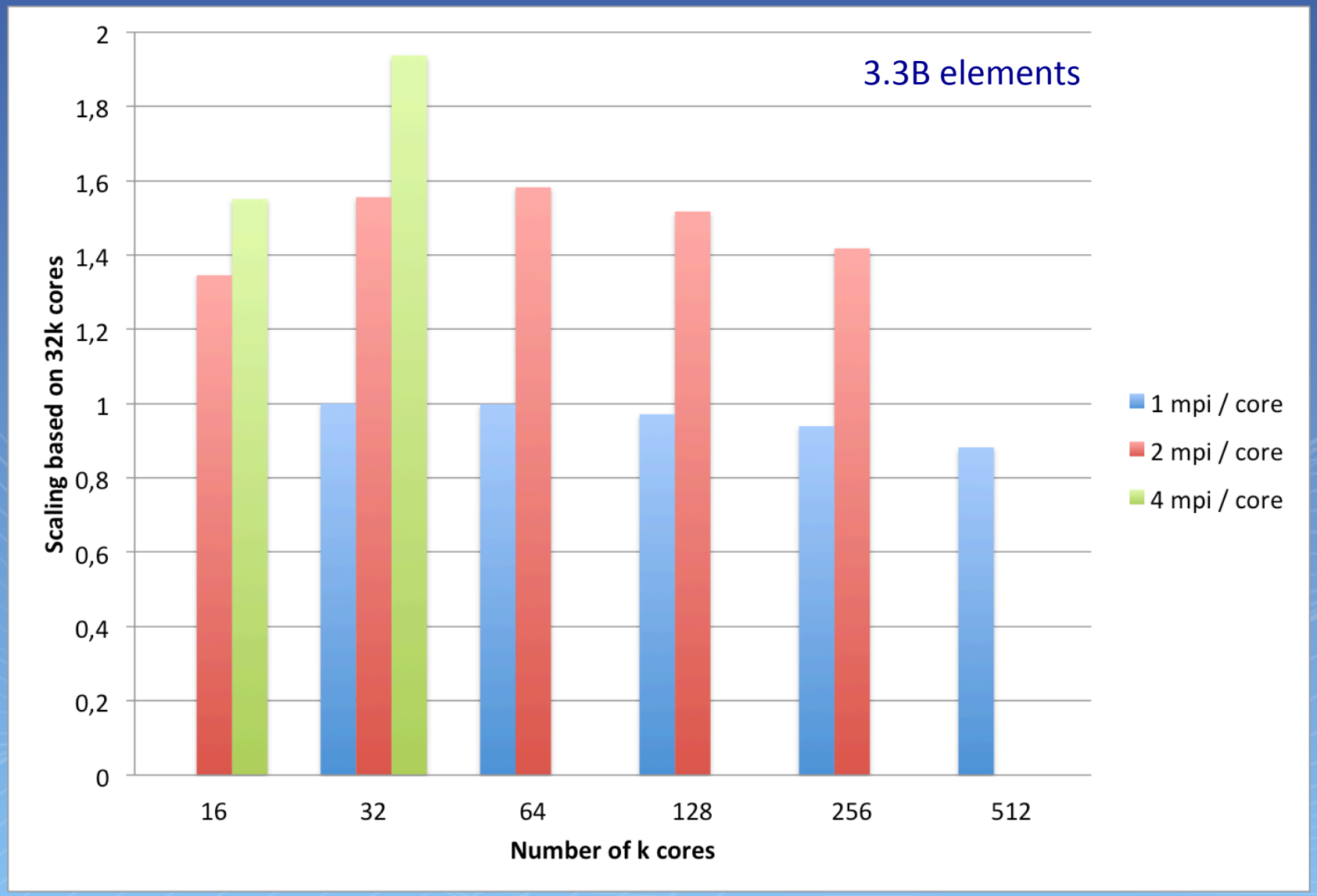
PHASTA Scaling on MIRA



PHASTA Scaling on MIRA



PHASTA Scaling on MIRA



Mesh partitioning

3.3 billion elements mesh

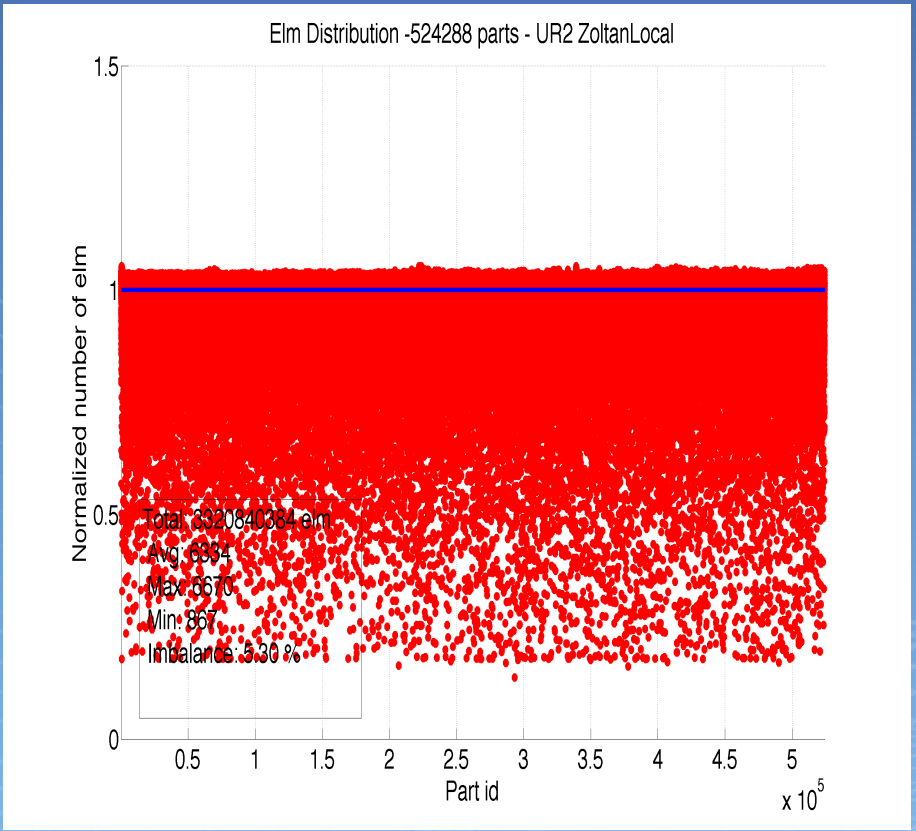
Parts	32K		64K		128K	
	Elements	Vertices	Elements	Vertices	Elements	Vertices
Total	3.3 billion	628,518,185	cst	658,364,797	cst	690,969,857
Target avg	101,344	19,181	50,672	10,046	25,336	5,272
Max in part	107,087	20,910	53,362	11,053	26,681	5,940
% imbalance	**5.67**	**9.01**	**5.31**	**10.02**	**5.31**	**12.67**
Min in part	56,203	10,790	28,427	5,824	11,993	2,610

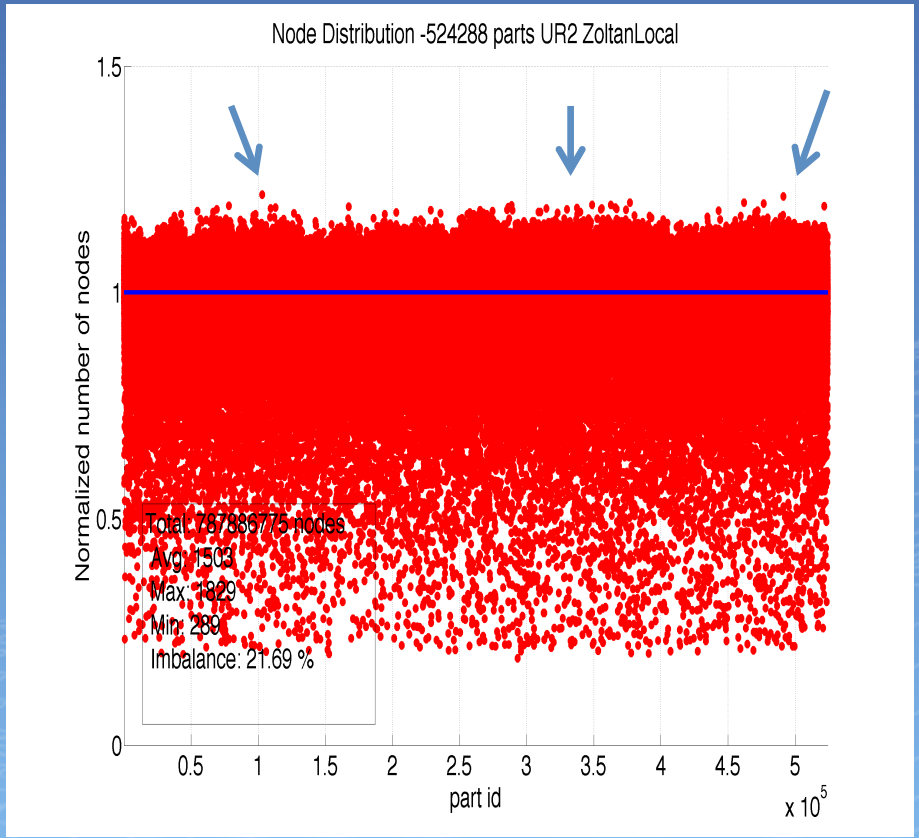
Parts	256K		512K		1536K	
	Elements	Vertices	Elements	Vertices	Elements	Vertices
Total	cst	733,192,244	cst	787,886,775	cst	905,248,408
Target avg	12,668	2,797	6,334	1,503	2,111	576
Max in a part	13,340	3,267	6,670	1,829	2,231	875
% imbalance	**5.30**	**16.80**	**5.30**	**21.69**	**5.68**	**51.91**
Min in a part	5,251	1,170	867	289	0	0

- Pure element-based partitioners lead significant vertex imbalance
- More and more shared vertices increases communication cost

Mesh partitioning

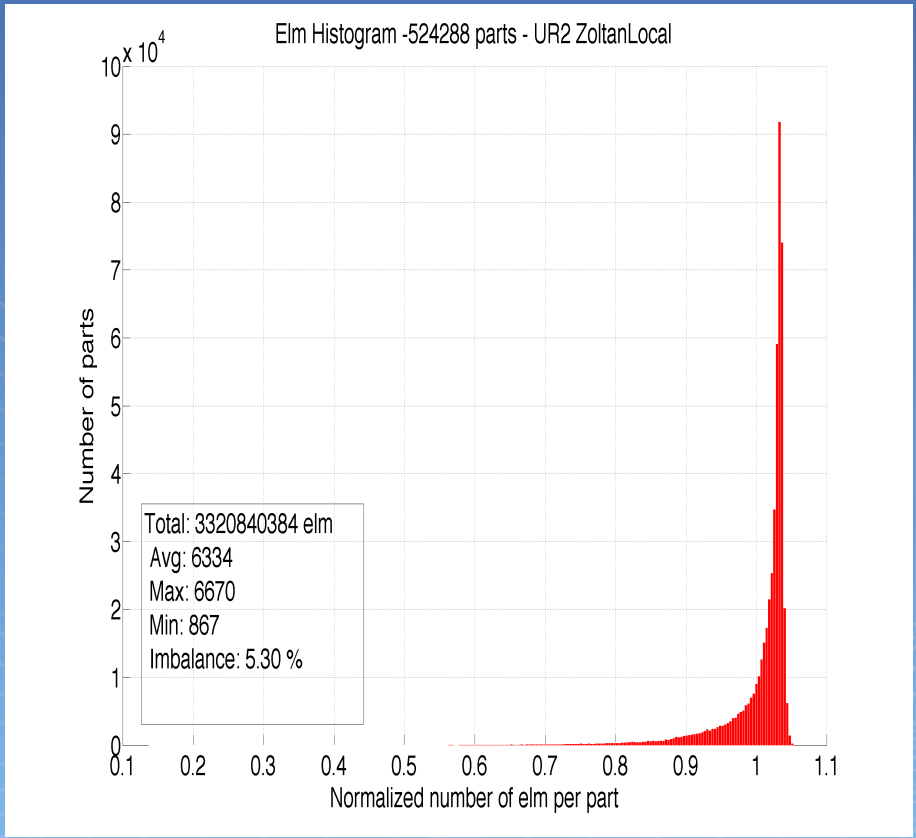
3.3 billion elements partitioned in 512K parts

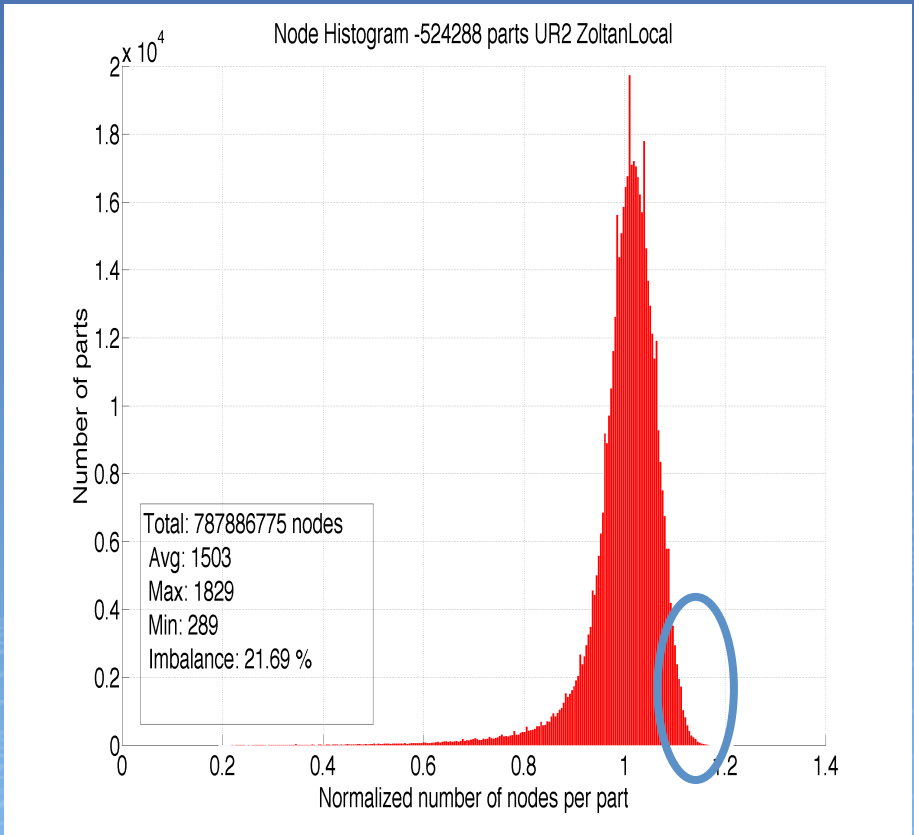




Mesh partitioning

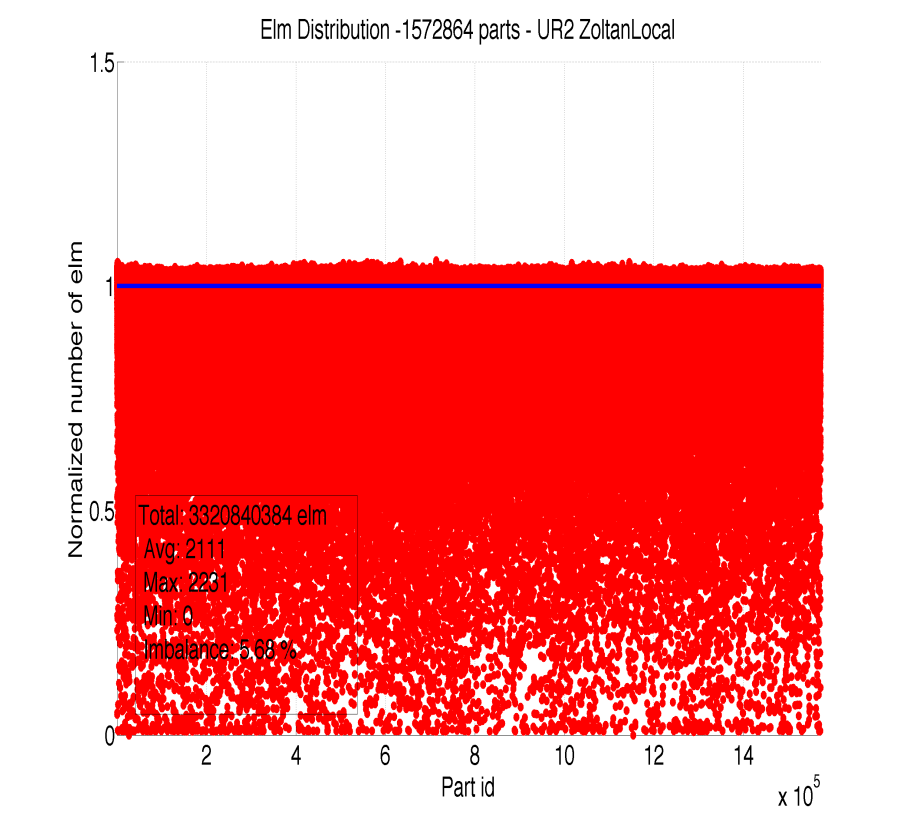
3.3 billion elements partitioned in 512K parts

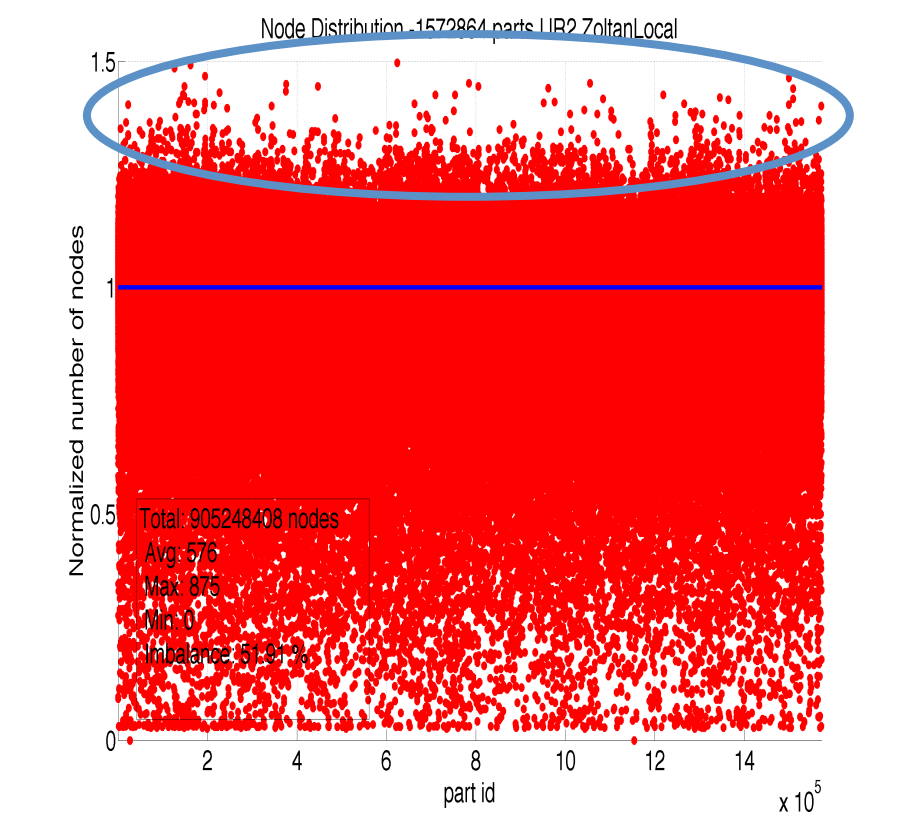




Mesh partitioning

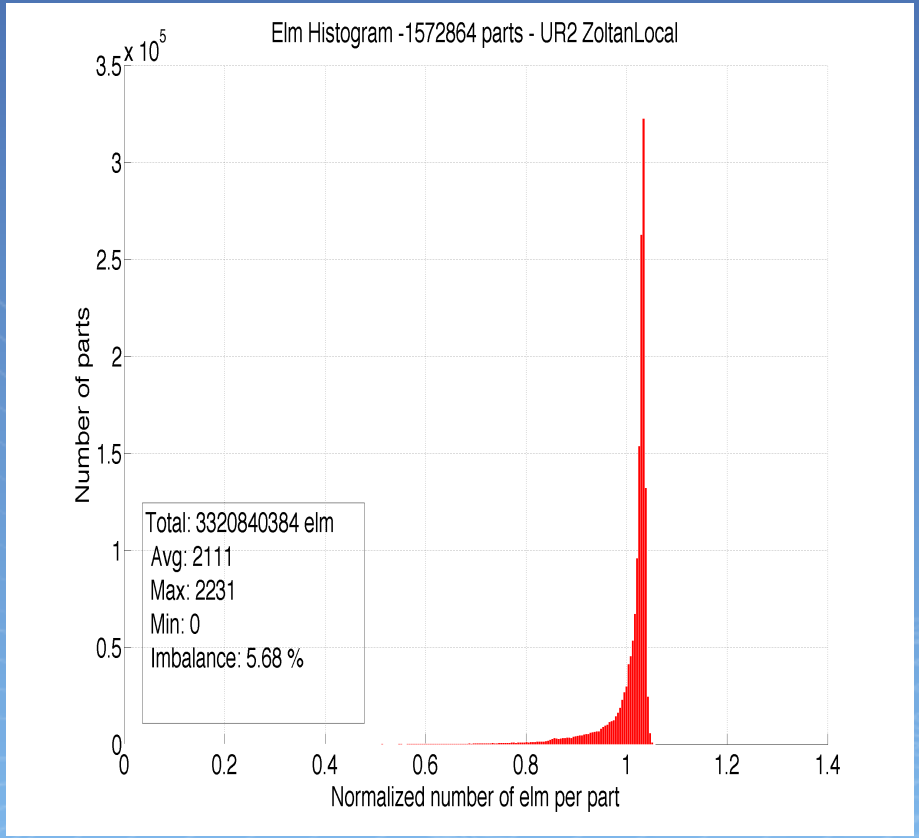
3.3 billion elements partitioned in 1536K parts

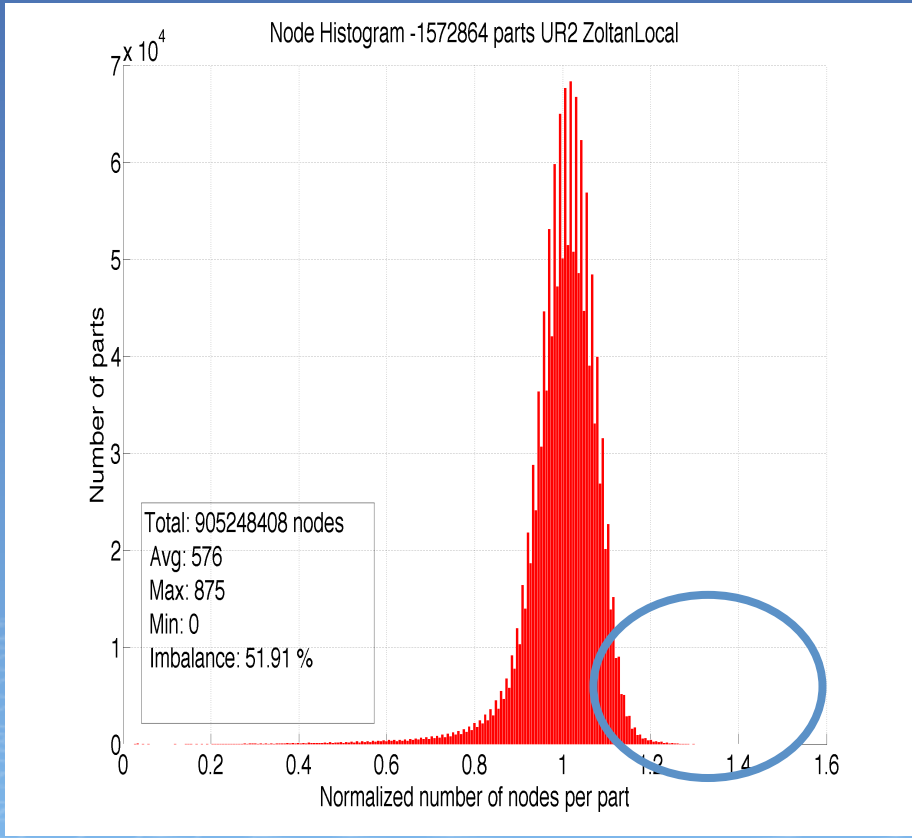




Mesh partitioning

3.3 billion elements partitioned in 1526K parts





Mesh partitioning Improvement

- ParMA Library (Partitioning using Mesh Adjacency) developed at the Rensselaer Polytechnic Institute
- Improve the vertex balance
- At the price of some element imbalance
- Decrease the number of shared vertices on part boundaries
- Decrease the communication cost

=> **Parts are better balance (both in terms of element AND vertex)**

Mesh partitioning Improvement

- ParMA: 2 algorithms applied successively
  - ParMA Diffusive:
    - Migrate a few elements **locally** from heavily loaded parts to lightly loaded **neighboring** parts
  - ParMA Heavy Part Splitting: (see later)

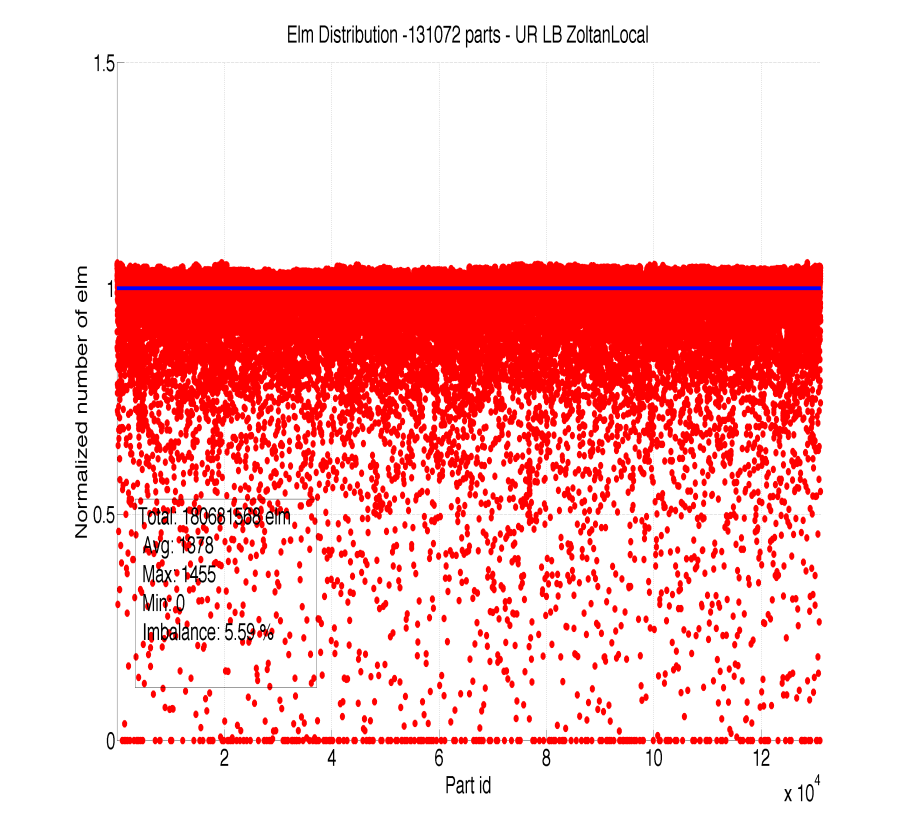
Mesh partitioning Improvement

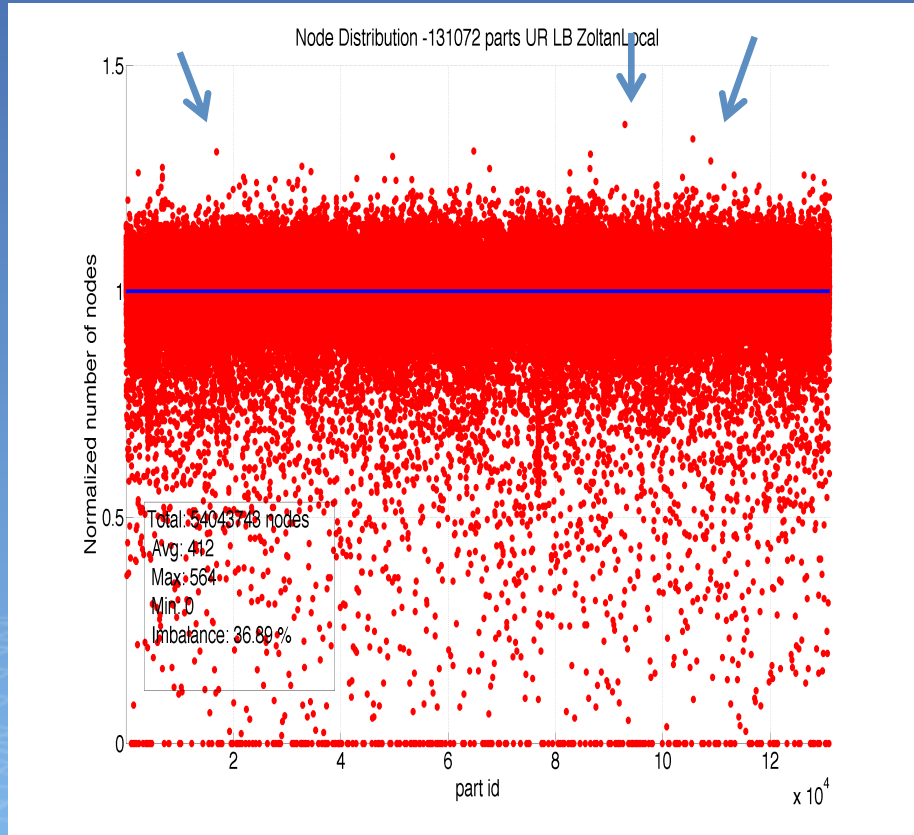
- Parma Diffusive with an extreme case
- 180 million elements mesh partitioned in 128K parts

Parts	128K (no Parma)		128K (with Parma)	
	Elements	Vertices	Elements	Vertices
Total	180 million	54,043,396	cst	52,723,643
Target avg	1,378	412	1,378	402
Max in a part	1,455	564	1,604	462
% imbalance	**5.59**	**36.89**	**16.40**	**14.93**
Min in a part	0	0	1	4

Mesh partitioning Improvement

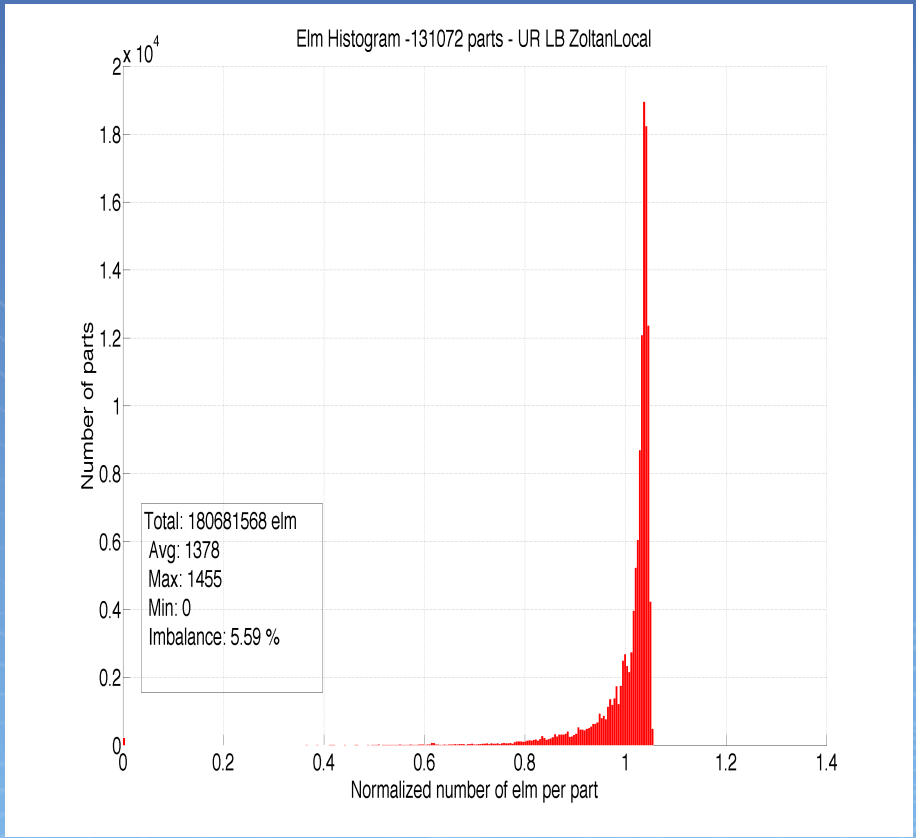
180 million elements partitioned in 128K parts (no Parma)

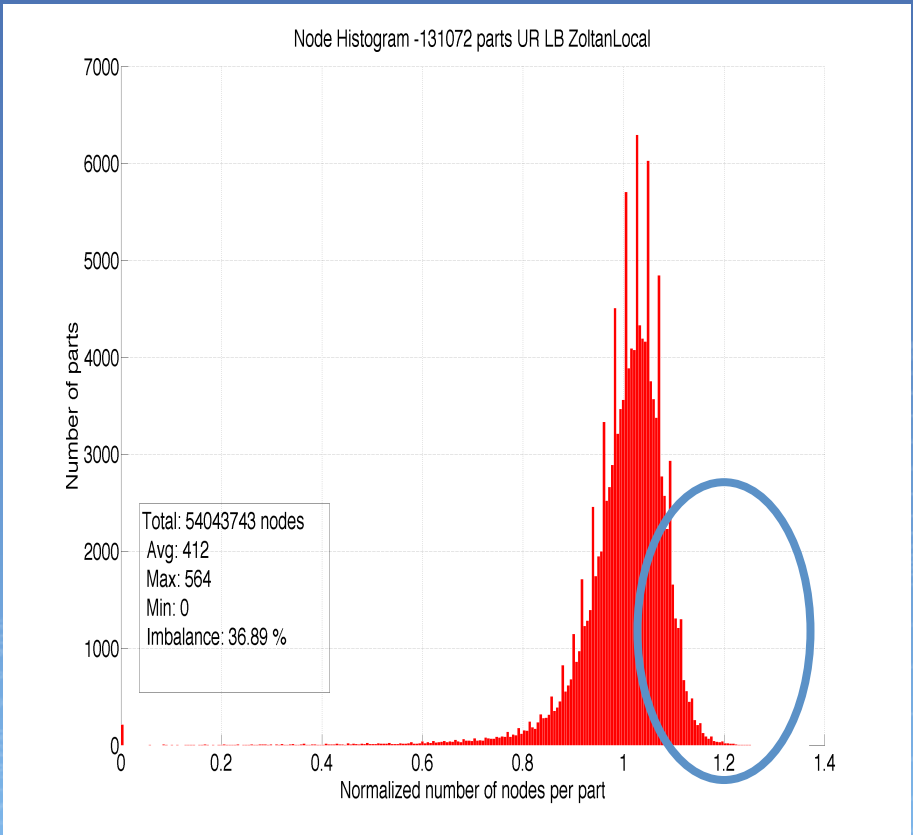




Mesh partitioning Improvement

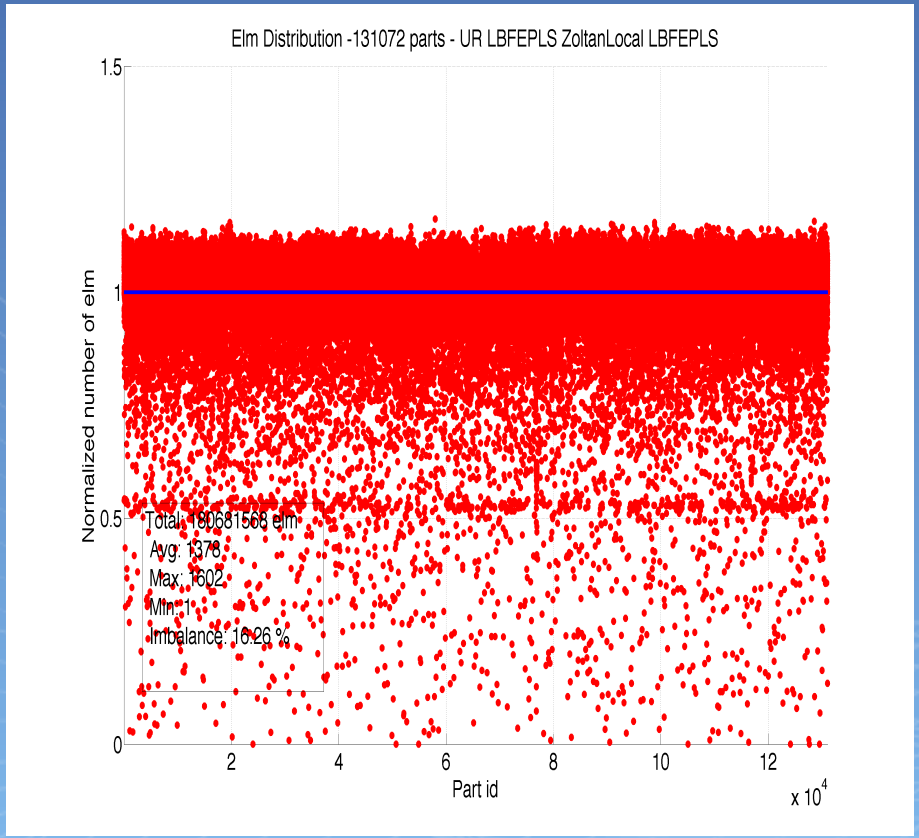
180 million elements partitioned in 128K parts (no Parma)

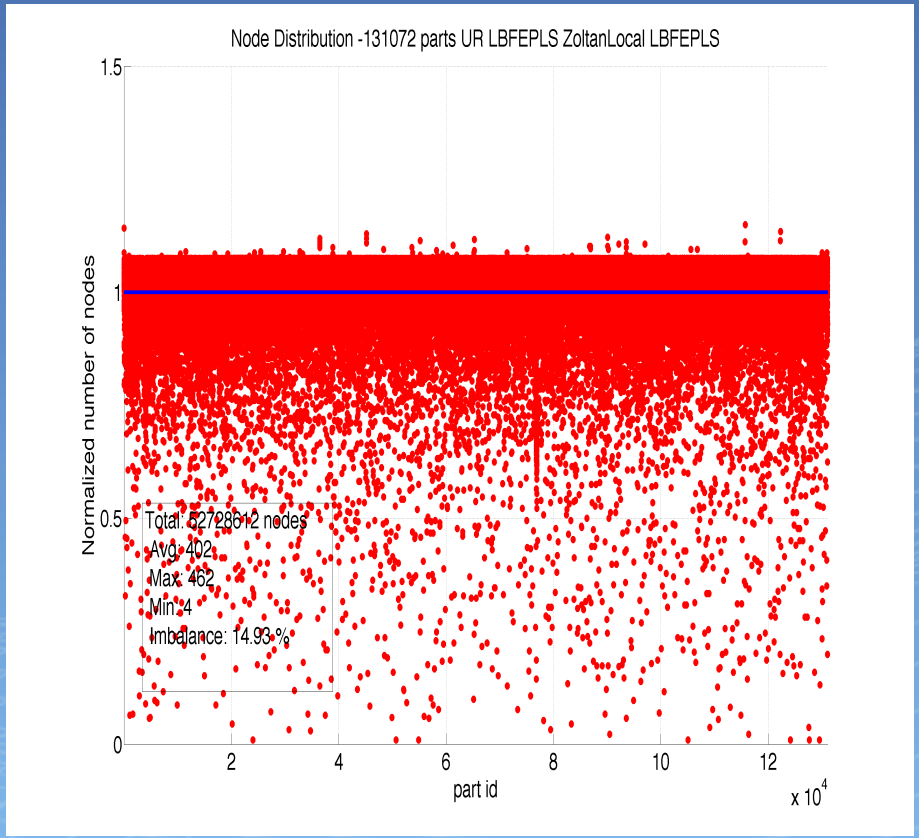




Mesh partitioning Improvement

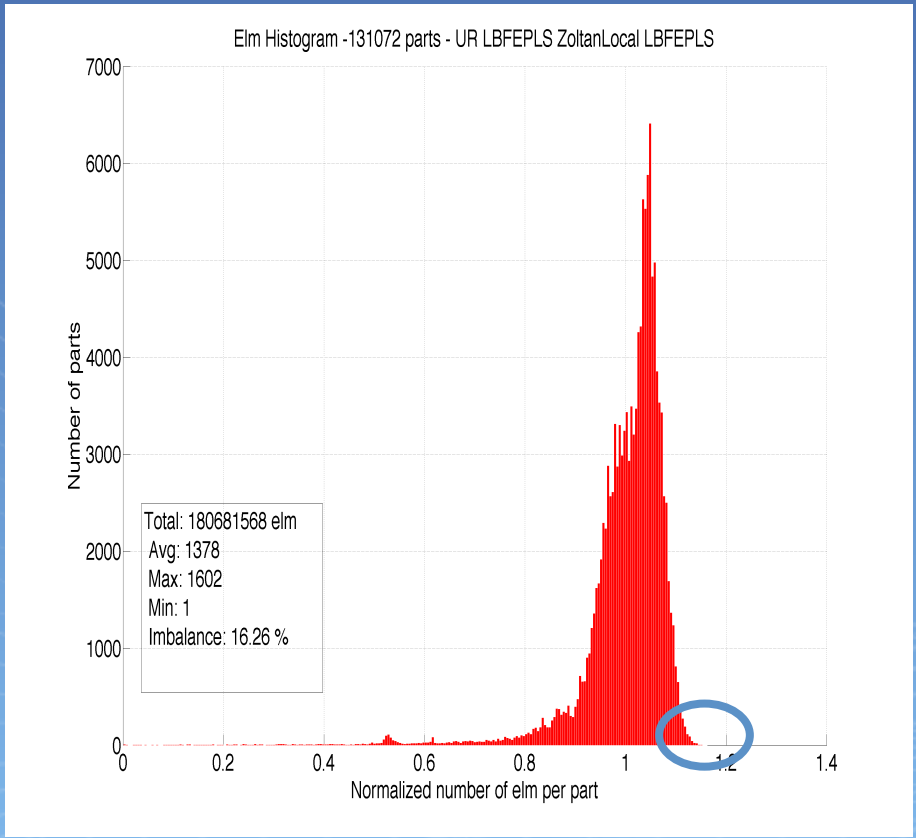
180 million elements partitioned in 128K parts (with Parma)

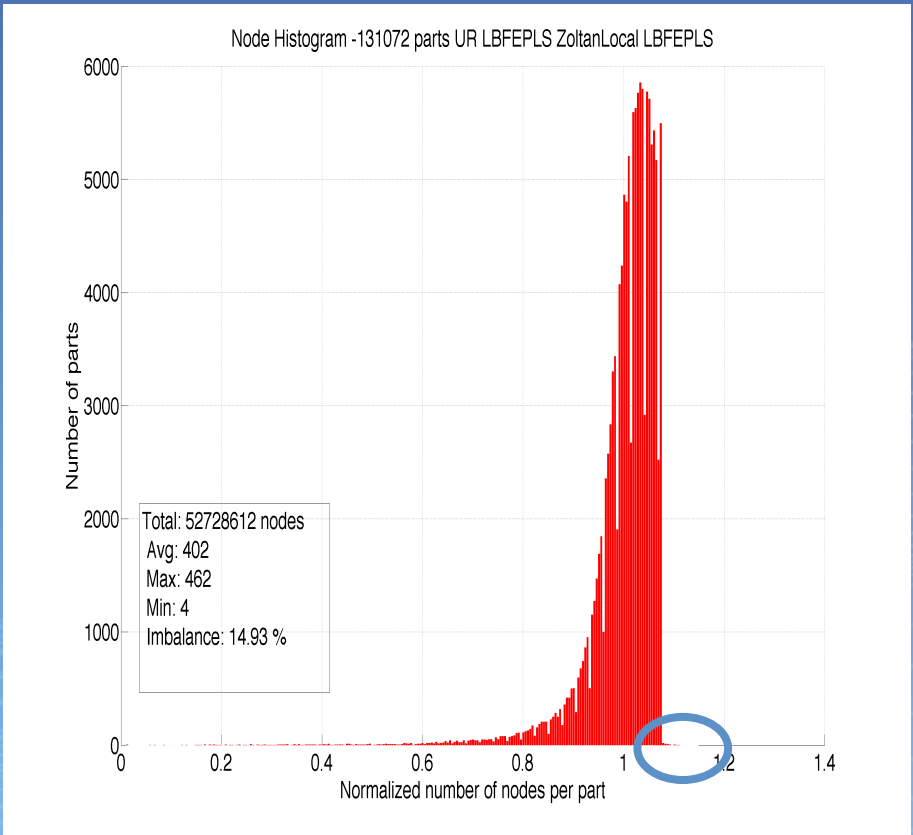




Mesh partitioning Improvement

180 million elements partitioned in 128K parts (with Parma)



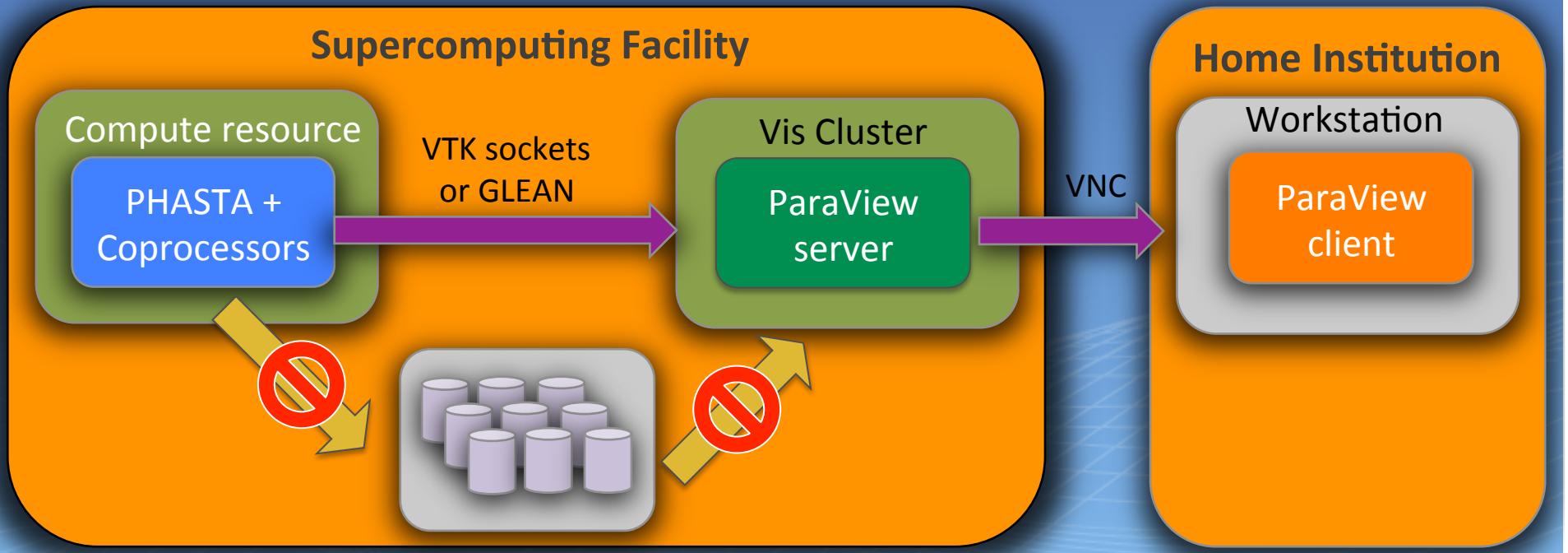


Mesh partitioning Improvement

- ParMA: 2 algorithms applied successively
  - Parma Diffusive:
    - Migrate a few elements **locally** from heavily loaded parts to lightly loaded **neighboring** parts
  - Parma Heavy Part Splitting:
    - Estimate the number of heavy loaded parts in the mesh (elm and vtx)
    - Generate as many empty parts as needed by grouping lightly loaded parts together
    - Split the heavy loaded parts in two or more parts and migrate these new split parts to the empty parts

Live Data Co-Visualization

- Massively parallel solvers create data at ever increasing rates
- Our ability to write, store, and process the data has not kept pace
- Co-visualization seeks to resolve this dilemma
- Collaboration with Kitware (ParaView)



- PHASTA simulation running on 160K cores of Intrepid BG/P
- ParaView server running on up to 100 Eureka nodes and 800 cores

Conclusions

- Lessons learned:
  - MPI works well up to O(1 million) processes (and beyond?)
  - PHASTA (and fully implicit finite element methods) scale very well
  - Unstructured meshes partitioning: element AND node balance (ParMA @ RPI)
  - MPI-IO based library to load O(1 million) parts and write the solution for every rank

- Ongoing and future work:
  - Scaling improvement through mesh partitioning
    - ParMA: heavy part splitting for element and node balance
  - PHASTA: global performance improvement
  - Live Data Co-Visualization on Mira/Tukey