Session 4: Heterogeneous Applications:
Maximizing Benefits While Minimizing Cost

Load Balancing on Many-Core and Accelerated Systems

Cameron W. Smith, Gerrett Diamond, and Mark S. Shephard

Scientific Computation Research Center
Rensselaer Polytechnic Institute

April 25, 2019

Load Balancing, Performance, and Scalability

- Partitioning and Load Balancing - distributing work to maximize performance on a given machine
- Performance - time to solution
  - local - serial, vectorized, threaded, or data parallel procedures on many-core processor or accelerator
  - global - local operations + coordination/synchronization procedures
- Scalability - efficiency with increase in computing power
  - Weak - increase problem size with increase in computing power
  - Strong - fixed problem size as computing power increases
- High performance procedures are not necessarily scalable or portable (some parameter tuning, no algorithm changes required to get acceptable performance [1])
- Scalable procedures are not necessarily portable

[1] `https://performanceportability.org/perfport/definition/`

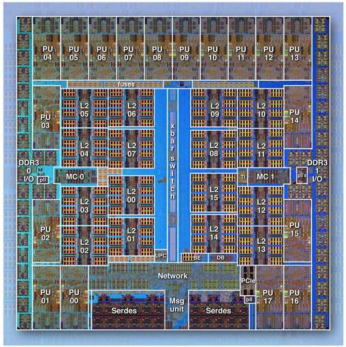
Manycore and Accelerated Systems

Manycore Node

- Many homogeneous cores provide all the computing power
- Each core can access filesystem and network
- Possibly multiple NUMA domains

Accelerated Node

- Host processor - dispatches accelerator work, facilitates communications, OS interactions
- Accelerator - provides most of the computing power
  e.g.) Summit GPUs have 95% of memory bandwidth and 98% of FLOPs



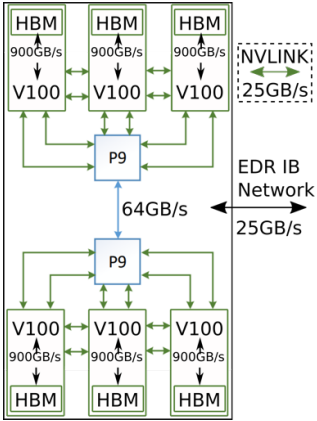
IBM A2 System-on-a-Chip
R.Haring, 2011, "The Blue Gene/Q Compute Chip"

Summit Computing Hardware

Focus on GPUs; they provide 95% of memory bandwidth and 98% of FLOPs

- 6 V100 and 2 P9 per node
- 4,608 nodes → 27,648 V100 and 9,216 P9

Processor	Double Precision TeraFLOPs	Memory Bandwidth (GB/s)
V100	7.5	900
P9	0.5	135
P9/V100	7%	15%
Full System		
V100	207,360	24,833,200
P9	4,608	1,244,160
P9/V100	2%	5%



J.Choquette. Hot Chips 2017.
"Volta: Programmability and Performance"

Summit Communication Hardware

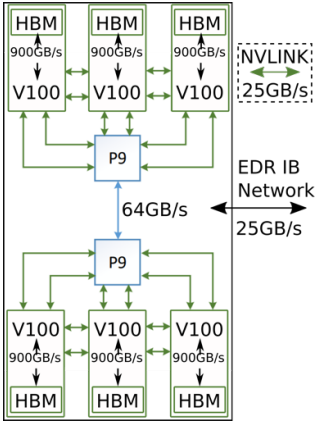
Overlap communication with computation and minimize inter-node communication: 205:1 ratio of aggregate V100 HBM bandwidth to inter-node EDR bandwidth

System	Stream Triad (GB/s)	Network Peak (GB/s)	Stream/ Network
Summit (inter-node)	5,130	25	205
Summit (intra-node)	855	50	17
Stampede2	194	12	17
Mira	27	20	1.4
Perlmutter (inter-node)[1]	5,130	100	51

Summit stream/network ratios

- inter-node - sum six V100 HBM to EDR IB bandwidth
- intra-node - V100 HBM bandwidth to NVLINK bandwidth

[1] '4x Volta Next' ?= 4x1.5xV100, Slingshot 4x 25GB/s links



J.Choquette. Hot Chips 2017.
"Volta: Programmability and Performance"

Motivation and Focus

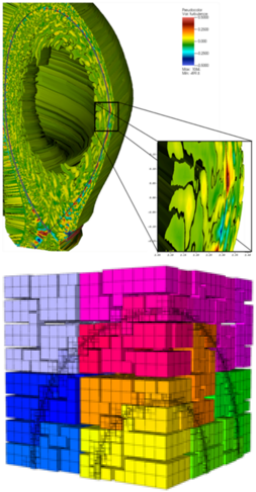
Simulations with regions of physical interest that change.

Many simulations have one or more of the following characteristics that make scaling and achieving performance difficult.

- Complex relational structures.
- Irregular forms of computational and communication costs.
- Evolving imbalance of work characterized by multiple criteria.

Evolving simulations we focus on have:

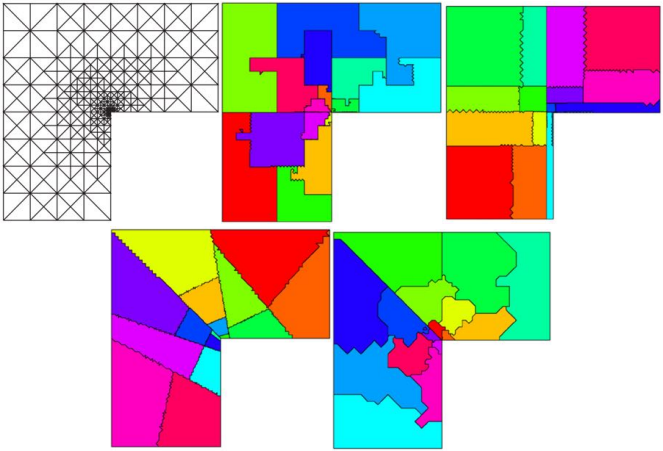
- A single executable built from library APIs - in-memory interactions
- Bulk synchronous parallel execution model
  (1) work (2) communicate (3) goto 1



XGC fusion plasma physics (top) and MFEM Laghos Sedov blast (bottom)

Common Methods for Partitioning and Load Balancing

- Multilevel Graph Methods
  - ParMETIS
  - Zoltan
- Geometric Methods
  - Recursive Coordinate(Inertial) Bisection
  - Multi-Jagged
- Diffusive Methods
  - Label Propagation
  - ParMA and EnGPar
- ... Not within a SIMT GPU
  - same work for each thread
  - e.g.) loop collapsing graph traversal - operate on edges instead of nodes



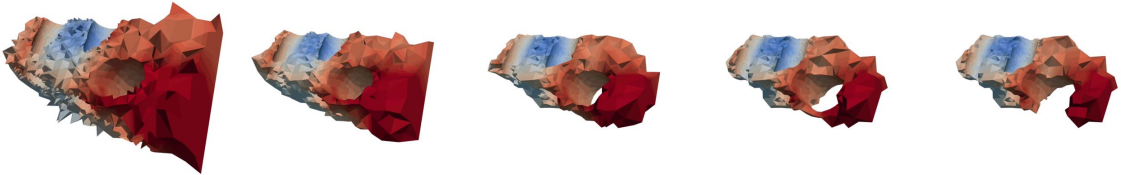
Top to bottom, left to right: mesh, space filling curve, coordinate bisection, inertial bisection, multi-level k-way.

"A refinement-tree based partitioning method for dynamic load balancing with adaptively refined grids", William F. Mitchell

EnGPar: One Approach to Partition Control

What is EnGPar?

- Provides a diffusive load balancing algorithm for partition improvement and supports multi-criteria partitioning.
- Complements existing multi-level and geometric methods.
- Utilizes a weighted multi(hyper)graph structure to represent data dependencies.
- Implemented to support efficient data parallel operations on accelerators



Multiple diffusive iterations (left to right) biased to migrate entities in order of descending distance (red to blue) from the topological center of the part (blue).

Unstructured Mesh Partition Improvement

Problem setup

- Billion element mesh of vertical tail structure.
- Run on the Mira BG/Q with one process per hardware thread.
- Target imbalance of 1.05.
- The imbalance of a given type (vtx, edge, face, or region) is defined as the max part weight divided by the avg part weight.

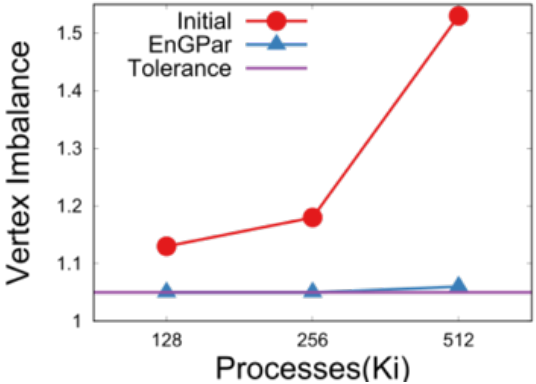
Initial partitions are built using:

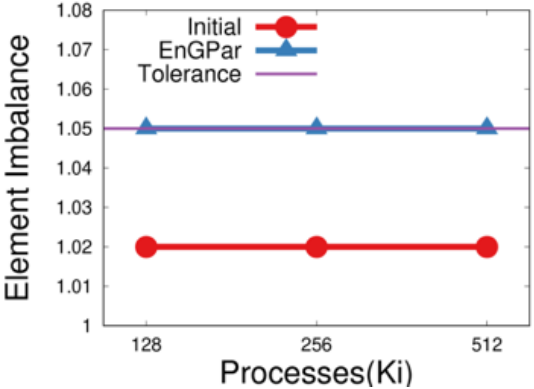
- Global ParMETIS part k-way to 8Ki($8 * 2^{10}$) parts.
- Local ParMETIS part k-way from 8Ki to 128Ki, 256Ki, and 512Ki parts.

The partitions before using EnGPar:

Number of Parts	128Ki	256Ki	512Ki
Elements per part	9,836	4,918	2,459
Vertex imbalance	1.13	1.18	1.53
Element imbalance	1.02	1.02	1.02

Element Partition: Mesh Vertex and Element Imbalance





Mesh vertex imbalances is reduced from 13% to 5% for 128Ki, 18% to 5% for 256Ki, and 53% to 6% for 512Ki. Edge cut is increased by 1%.

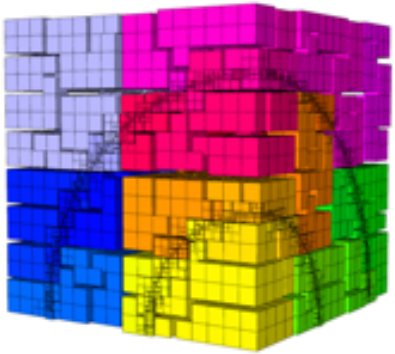
Balancing High-order Adaptive Finite Elements

MFEM Laghos MiniApp - High-order Compressible Gas Dynamics

Tzanio Kolev and Vladimir Tomov, Lawrence Livermore National Laboratory
DOE Exascale Computing Project: The Center for Efficient Exascale Discretizations (CEED)

Goals

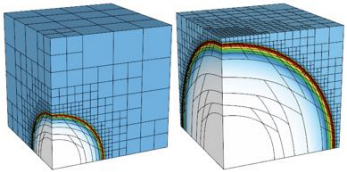
- Help applications leverage future architectures by providing them high order unstructured mesh methods.
- Collaborate with hardware vendors and software projects to guide the design of upcoming exascale systems.



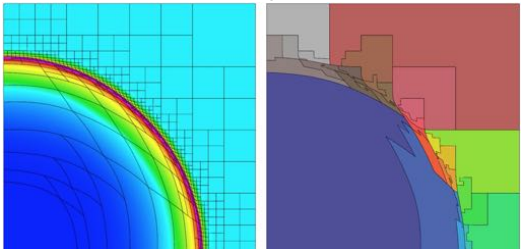
The Laghos miniapp's Sedov blast wave problem. Laghos solves the time-dependent Euler equation of compressible gas dynamics in a moving Lagrangian frame using high-order finite element spatial discretization and explicit high-order time-stepping.

MFEM Laghos MiniApp - Partitioning Approach (T. Kolev, V. Tomov)

- One MPI process per GPU
- Initial partition with METIS
- Non-conforming adaptive mesh refinement (AMR) version using space filling curves
- GPU porting of computationally dominant functions in AMR version is underway



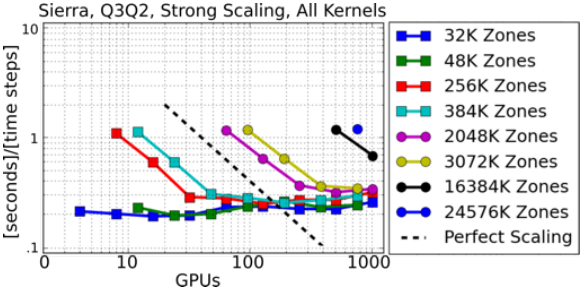
Sedov AMR at steps 900 and 2463.



Sedov AMR mesh (left) and partition (right) with space filling curves.

MFEM Laghos MiniApp - Early Sierra Results (T. Kolev, V. Tomov)

- 3rd order velocity and position (continuous kinematic), 2nd order internal energy (discontinuous thermodynamic) finite elements (Q3Q2)
- Mesh vertex repositioning without AMR
- Four GPUs/node with one MPI process/GPU
- More than 256K (256*1000) elements ('zones') required for scaling to 30 GPUs, 16M elements required for 1024 GPUs



Strong scaling for Laghos on Sierra of pure CUDA version. A zone is a mesh element.

MFEM Laghos MiniApp - Next Steps for Load Balancing

Task Mapping

- Ensure proceses that are assigned to the same node share domain data
- Zoltan2 (Karen Devine, SNL) geometric methods support the mapping of the network partition to the domain partition

EnGPar

- Integrate EnGPar w/MFEM
- Predictive balancing prior to mesh adaptation to (1) avoid memory exhaustion and (2) create a post adaptation partition w/ acceptable imbalance
- Tune post adapt partition to (1) minimize induced communications between nodes and (2) reduce work imbalance
- Complete acceleration of critical EnGPar procedures - initial work completed with Kokkos

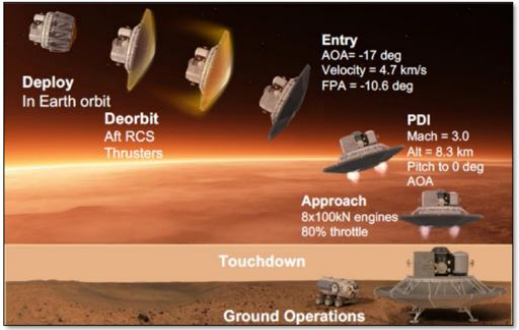
Balancing Finite Volume CFD

FUN3D - Finite Volume CFD

Eric Nielsen and Aaron Walden at NASA Langley
Summit ESP: "Enabling Human Exploration of the Red Planet"

Goals

- Science: Better understanding of retropropulsion flow physics during Mars entry of human-scale lander
- Computational: Demonstrate production readiness and efficiency advantages of GPU implementation at scale



FUN3D - Early Summit Results (E. Nielsen and A. Walden)

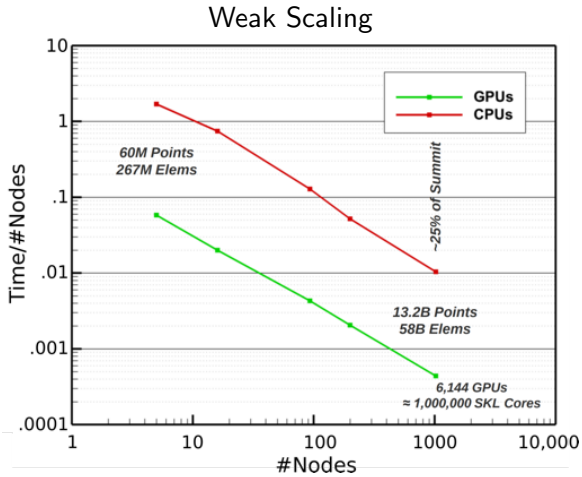
Partitioning Approach

- ParMETIS: mesh vertices → graph nodes, mesh edges → graph edges
- Mesh vertices hold DOFs
- EnGPar is integrated and being tested

Weak Scaling - nearly linear

- 13.2B ($13.2*10^9$) vertices, 58B elements, at 1024 nodes.
- GPU: MPI+CUDA, 3 ranks/socket, MPI via GPUDirect
- CPU: MPI+OpenMP, ranks/socket, 168 threads per node (SMT4)

GPU node-level performance is 23-37 times faster at scale than CPUs



Balancing Unstructured Mesh Particle-in-Cell

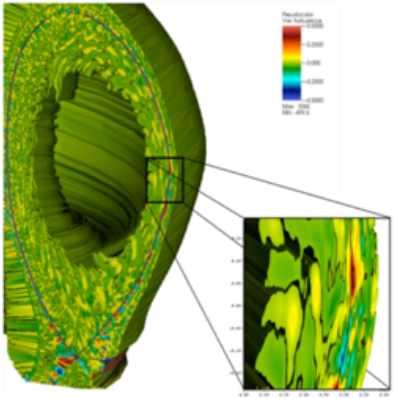
XGC - Fusion Plasma Physics

CS Chang, Princeton Plasma Physics Laboratory

Summit ESP: "Using XGC to predict ITERs boundary plasma performance and its impact on fusion efficiency"

Goals

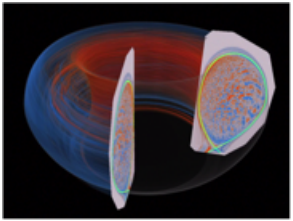
- Science: Study tokamak plasma performance near boundary
- Computational: Develop platform for multiscale plasma physics simulations on leadership systems



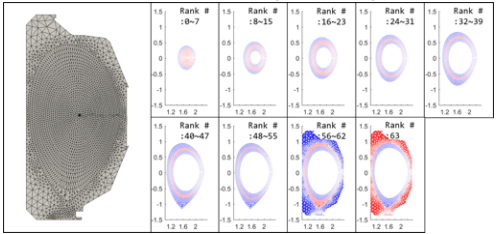
DIII-D Geometry: Boundary turbulence saturates in < 0.1ms, core turbulence in a few ms.

XGC - Partitioning Approach (CS Chang)

- Each process has entire copy of 2D poloidal mesh, mesh repeated N times in the toroidal direction
- Radial 1D partition of poloidal mesh vertices used for push and collision
- Each plane has a set of processes working on it, each process is assigned a different part
- Particles are owned by the part/process they reside in
- Load balancer monitors wall clock time of push and collision steps and uses a Golden Section Search to minimize the time as a function of the max particle count per process



3D tokamak simulation. 2 poloidal planes shown.



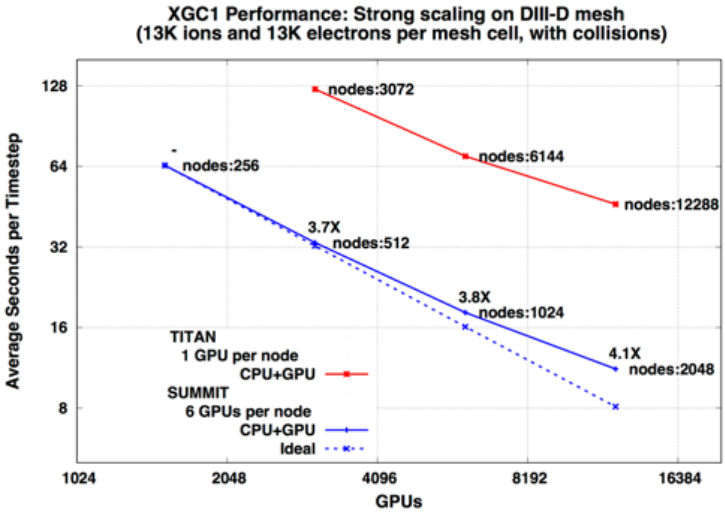
2D mesh and process groups.

XGC - Early Summit Results (CS Chang)

Strong Scaling vs Titan

- Scaling runs using up to 44% of Summit, 2048 nodes
- MPI + CUDA (electron push) + OpenACC (collision) + OpenMP
- 25 times faster than Titan

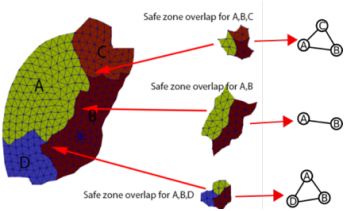
CPU+GPU performance is 11 times faster than CPU only at 12,288 GPUs (2Ki nodes)



XGC - Ongoing Developments

Developing tools for unstructured mesh PIC

- Distributed mesh with 'safe' zones to avoid communication during push
- Bulk communication protocol that accounts for 'safe' zones and can quickly update field data already on the GPU - Omega_h [1] for GPU mesh
- Particle load balancing with EnGPar
  - Construct subgraphs connecting processes for each overlapping safe zone.
  - Set the weights of vertices to be the number of particlesin the elements for the overlapping safe zone.
  - Diffusively migrate weight (# of particles) in each subgraph until processes are balanced.



Subgraphs created from core parts and safe zone parts.

[1] github.com/SNLcomputation/omega_h

Closing Remarks

Summit GPUs provide >90% of the system performance

- Technically heterogeneous...
- Cost to thread, optimize and load balance for the CPUs is not worth the person-hours

Focus on providing a 'decent' load imbalance to GPUs with minimized induced inter-GPU data movement

- Placing processes near those they share domain data with
- Optimizing partitions for communication, possibly sacrificing load balance
- Predictively load balance to reduce calls to balancer

Many more challenges if we had a processor/node with multiple specialized accelerators (e.g., compression, fft, spmv, graph traversal, encryption, etc...)

- Will current programming models/tools work effectively? Major code rewrite?
- Many post exascale technologies in the pipeline: `www.crnch.gatech.edu/sites/default/files/02-siamcse-2019-shalf.pdf`

Thank You

Questions?

Acknowledgements:

- NSF SI2-SSE: Fast Dynamic Load Balancing Tools for Extreme Scale Systems
- DOE FASTMath SCIDAC Institute
- CEED ECP