Adaptive MPI

Performance & Application Studies

Sam White
PPL, UIUC

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

PARALLEL PROGRAMMING LAB
DEPT. OF COMPUTER SCIENCE, UNIVERSITY OF ILLINOIS
PPL UIUC

Motivation

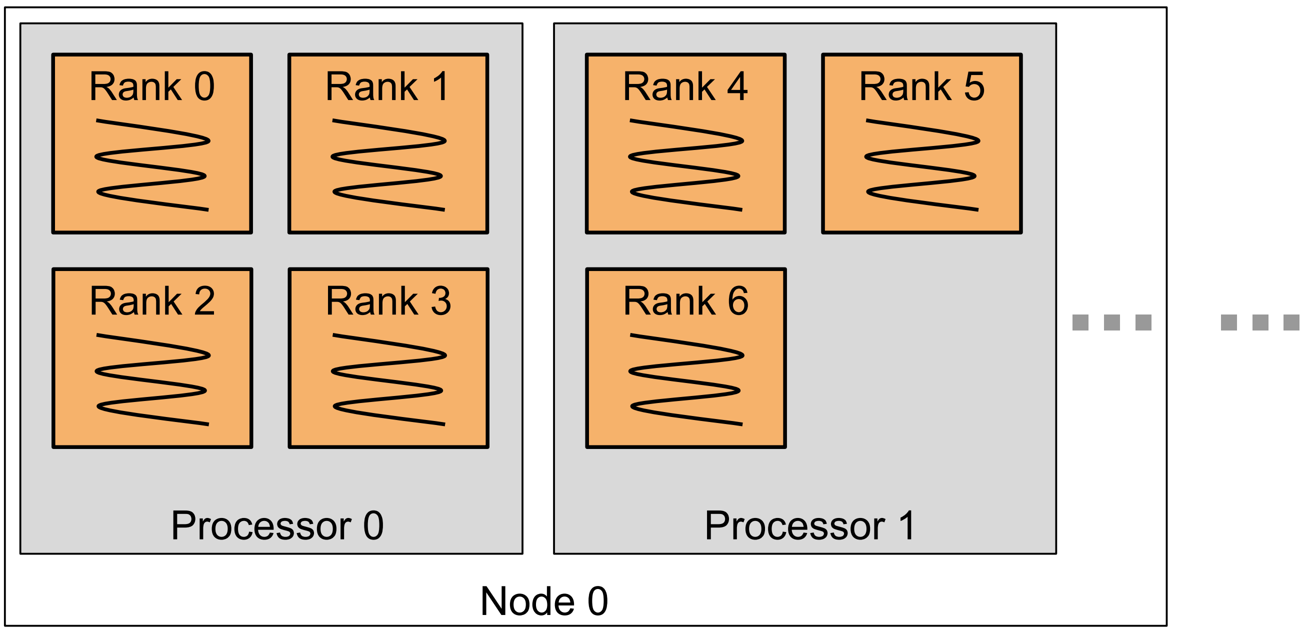
- Variability is becoming a problem for more applications
  - Software: multi-scale, multi-physics, mesh refinements, particle movements
  - Hardware: turbo-boost, power budgets, heterogeneity
- Who should be responsible for addressing it?
  - Applications? Runtimes? A new language?
  - Will something new work with existing code?

Motivation

- Q: Why MPI on top of Charm++?
- A: Application-independent features for MPI codes:
  - Most existing HPC codes/libraries are already written in MPI
  - Runtime features in familiar programming model:
    - Overdecomposition
    - Latency tolerance
    - Dynamic load balancing
    - Online fault tolerance

Adaptive MPI

- MPI implementation on top of Charm++
  - MPI ranks are lightweight, migratable user-level threads encapsulated in Charm++ objects



Overdecomposition

- MPI programmers already decompose to MPI ranks:
  - One rank per node/socket/core/…

- AMPI virtualizes MPI ranks, allowing multiple ranks to execute per core
  - Benefits:
    - Cache usage
    - Communication/computation overlap
    - Dynamic load balancing of ranks

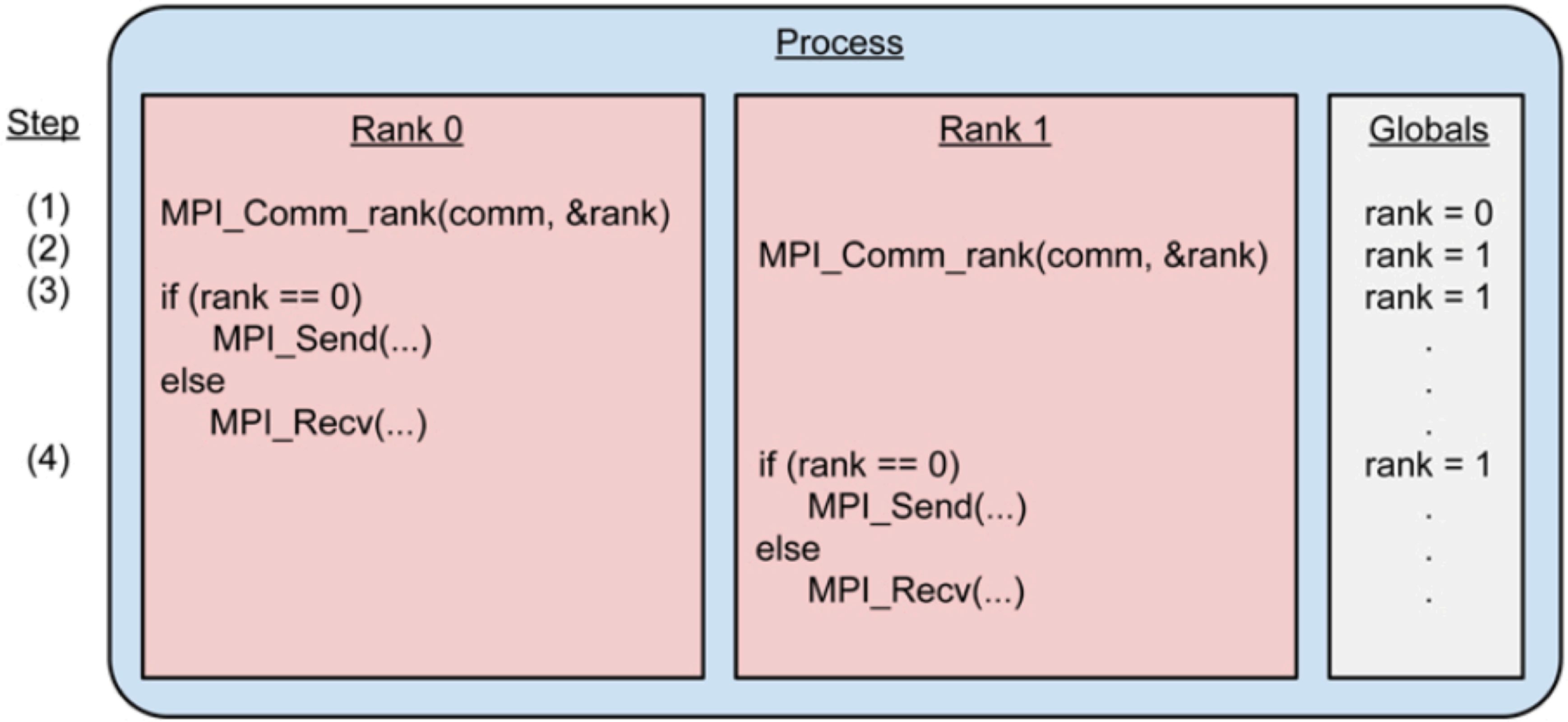
Thread Safety

- AMPI virtualizes ranks as threads
  - Is this safe?

```
int rank, size;
int main(int argc, char *argv[]) {

    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);


    if (rank==0) MPI_Send(…);
    else MPI_Recv(…);

    MPI_Finalize();
}
```

Thread Safety

- AMPI virtualizes ranks as threads
  - Is this safe? No, globals are defined per process

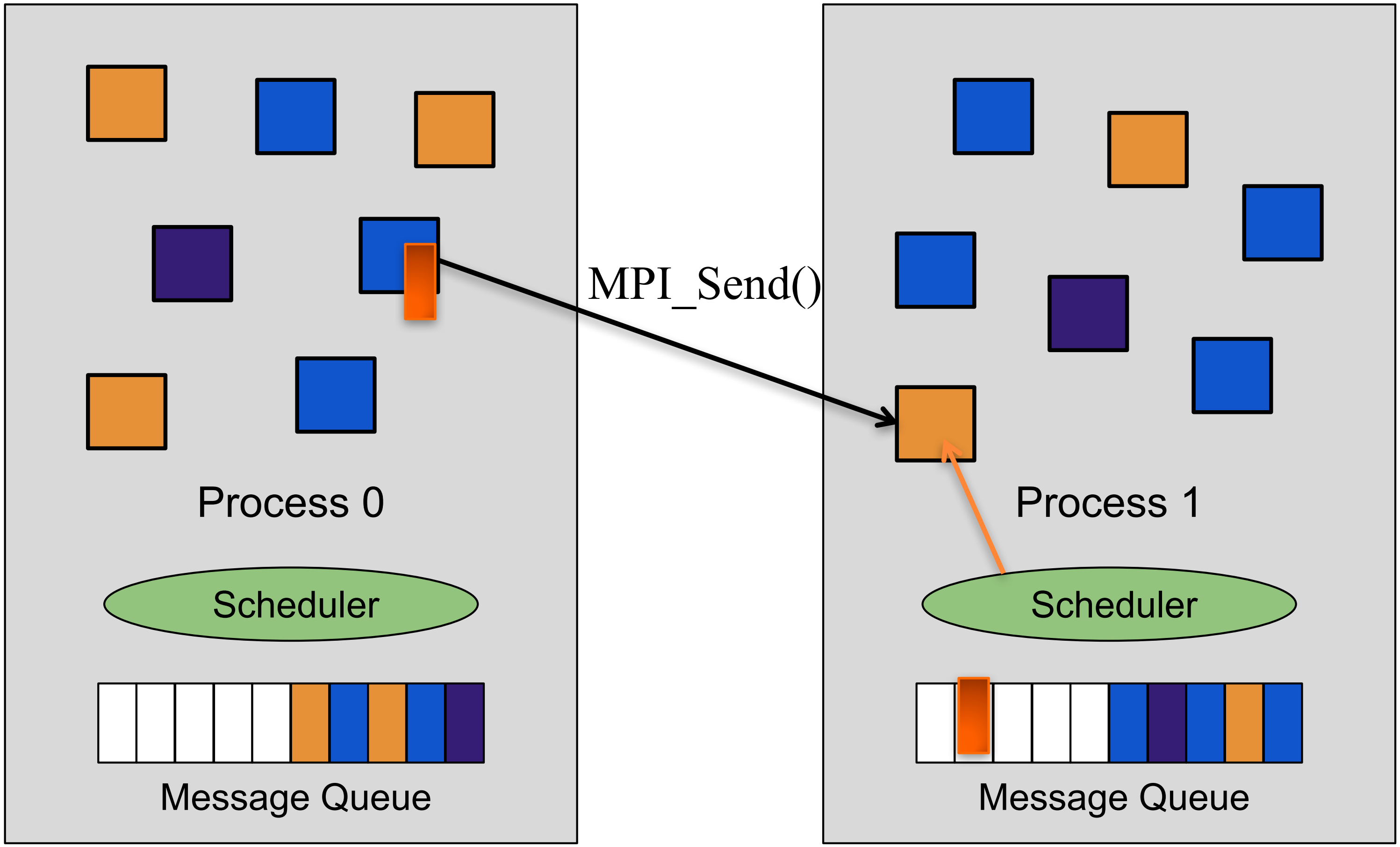


Thread Safety

- AMPI programs are MPI programs without mutable global/static variables
  A. Refactor unsafe code to pass variables on the stack
  B. Swap ELF Global Offset Table entries during ULT context switch
     - *ampicc -swapglobals*
  C. Swap Thread Local Storage (TLS) pointer during ULT context switch
     - *ampicc -tlsglobals*
     - Tag unsafe variables with C/C++ ‘thread_local’ or OpenMP’s ‘threadprivate’ attribute, or …
     - In progress: compiler can tag all unsafe variables, i.e. ‘icc -fmpc-privatize’

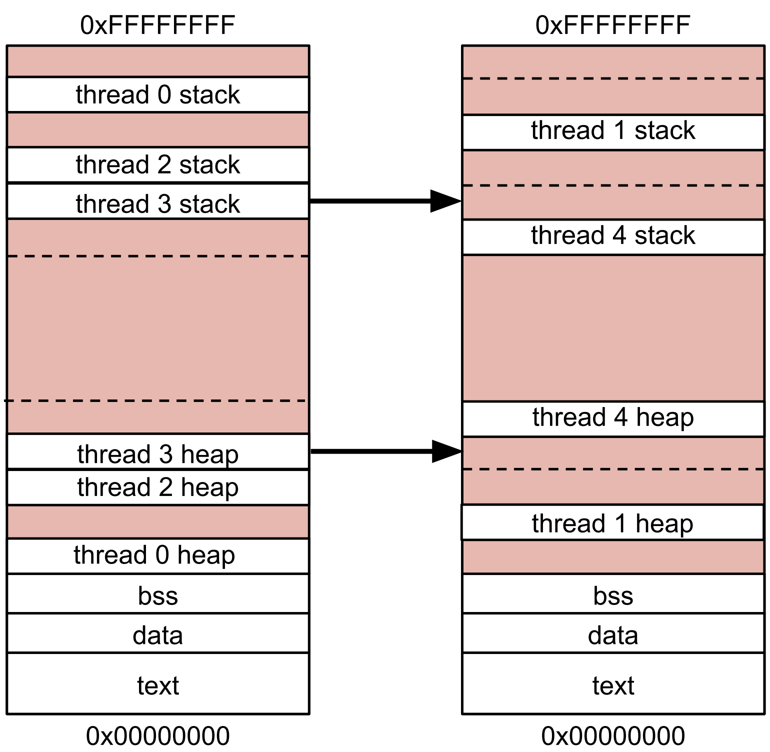
PPL UIUC

Message-driven Execution



Migratability

- AMPI ranks are migratable at runtime across address spaces
  - User-level thread stack & heap
- Isomalloc memory allocator
  - No application-specific code needed
  - Link with '-memory isomalloc'



Migratability

- AMPI ranks (threads) are bound to chare array elements
  - AMPI can transparently use Charm++ features

- ‘int AMPI_Migrate(MPI_Info)’ used for:
  - Measurement-based dynamic load balancing
  - Checkpoint to file
  - In-memory double checkpoint
  - Job shrink/expand

Applications

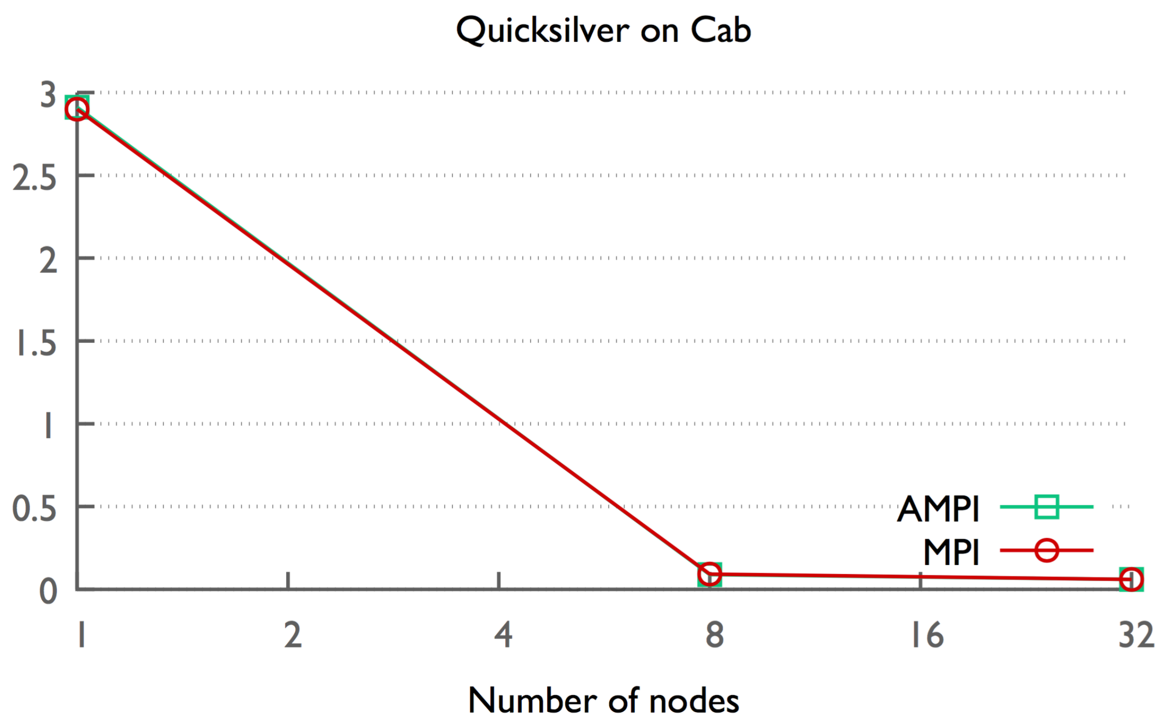
- LLNL proxy apps & libraries
- Harm3D: black hole simulations
- PlasComCM: Plasma-coupled combustion simulations

LLNL Applications

- Work with Abhinav Bhatele & Nikhil Jain
- Goals:
  - Assess completeness of AMPI implementation using full-scale applications
  - Benchmark baseline performance of AMPI compared to other MPI implementations
  - Show benefits of AMPI's high-level features

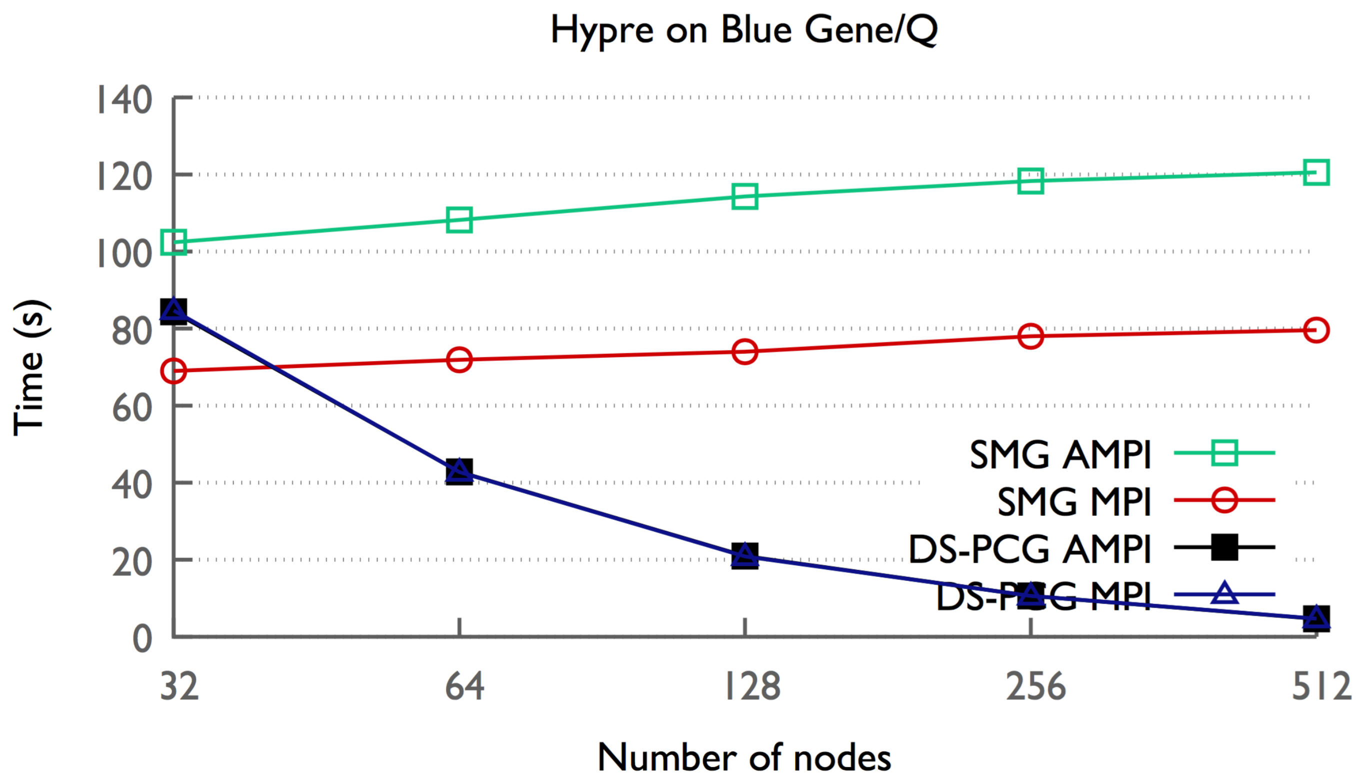
LLNL Applications

- Quicksilver proxy app
  - Monte Carlo Transport
  - Dynamic neutron transport problem



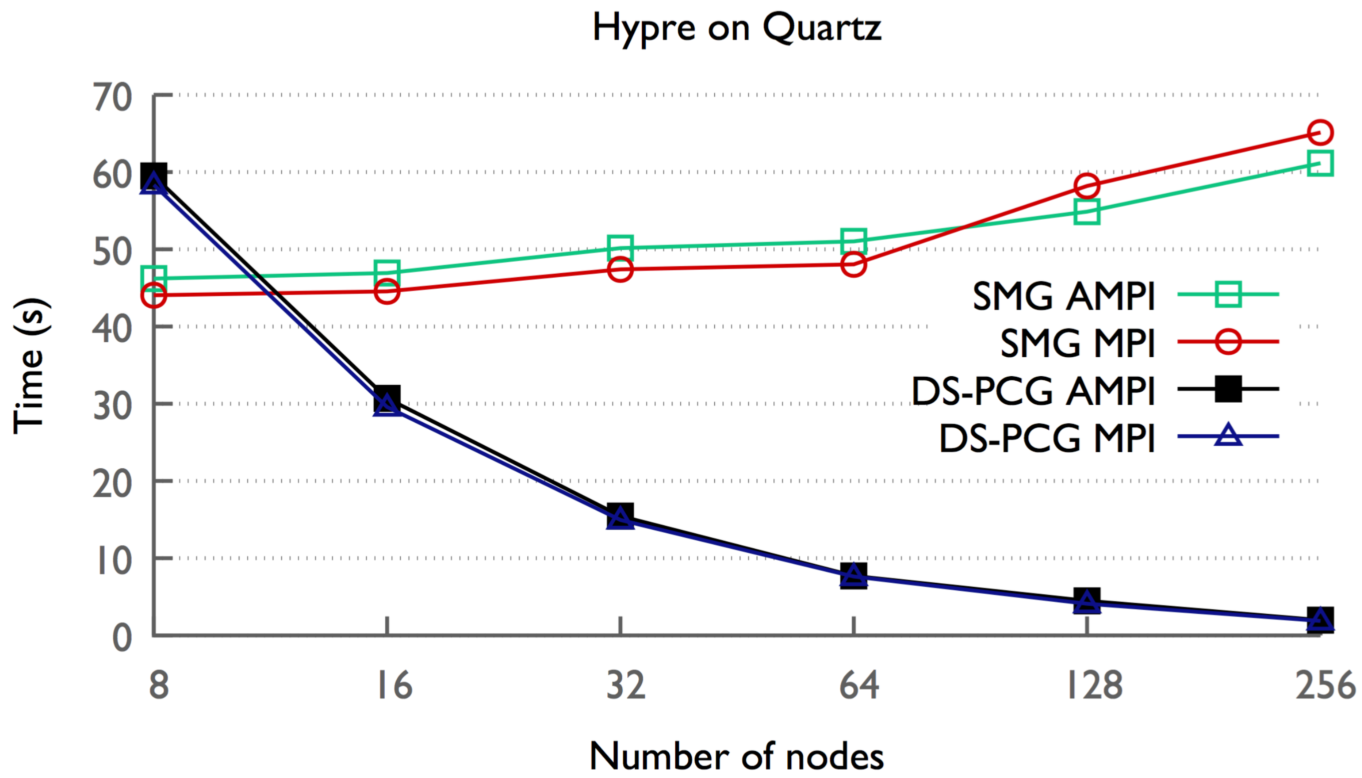
LLNL Applications

- Hypre benchmarks
  - Performance varied across machines, solvers
    - SMG uses many small messages, latency sensative



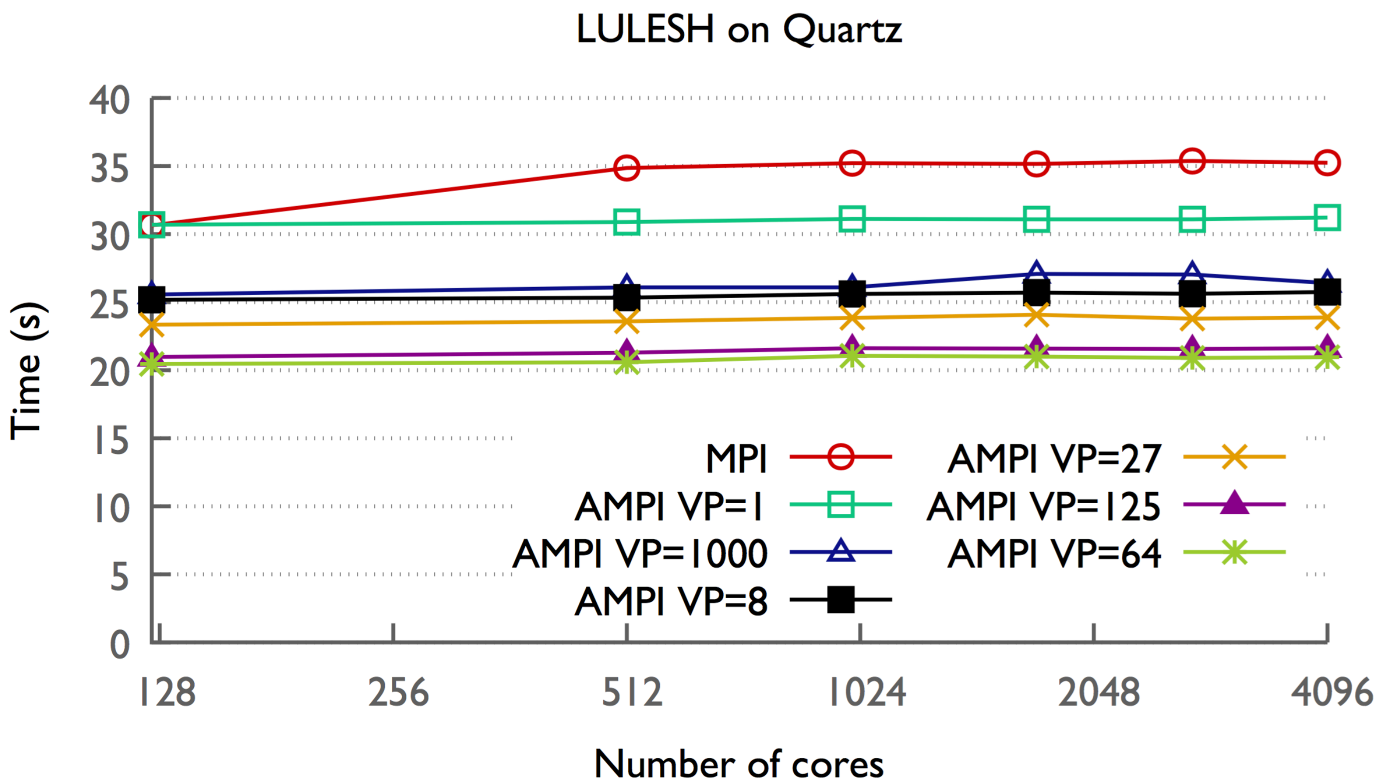
LLNL Applications

- Hypre benchmarks
  - Performance varied across machines, solvers
    - SMG uses many small messages, latency sensative



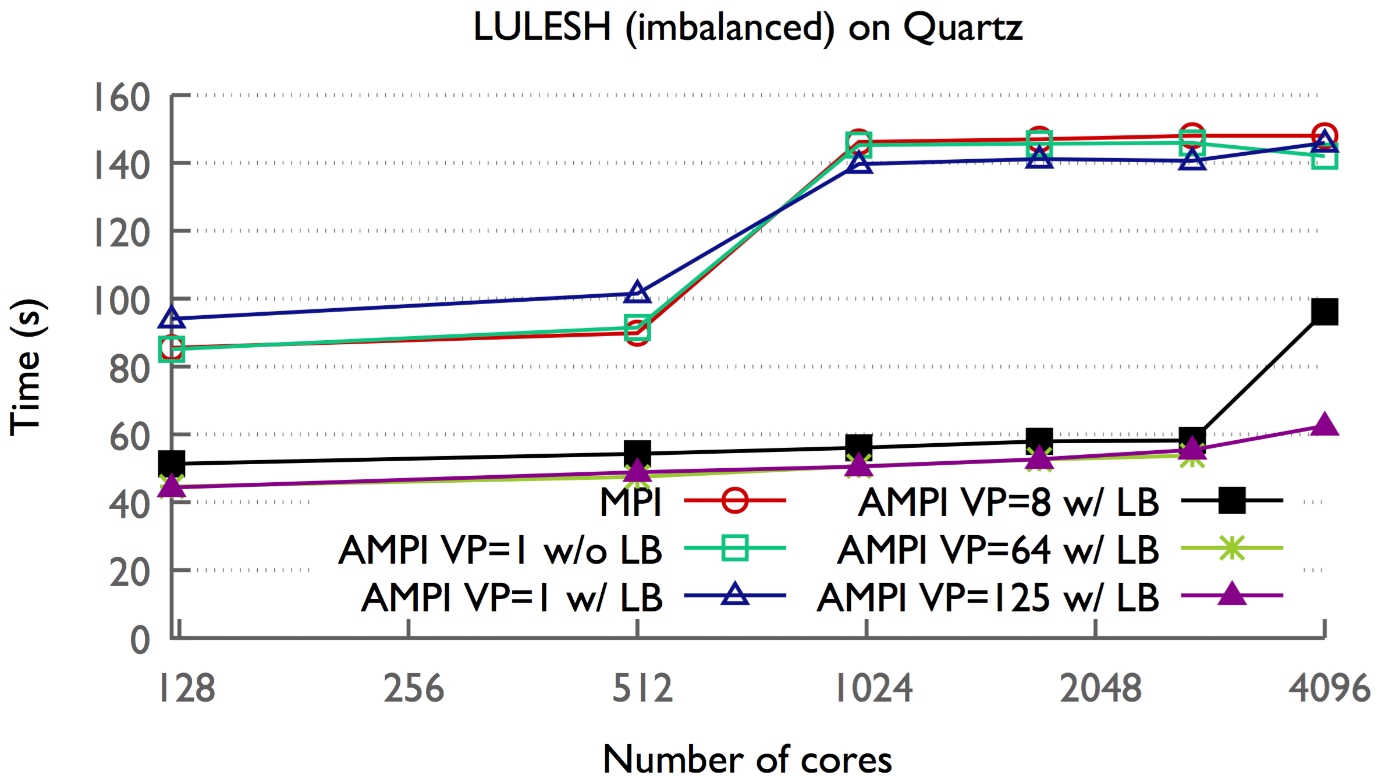
LLNL Applications

- LULESH 2.0
  - Shock hydrodynamics on a 3D unstructured mesh



LLNL Applications

- LULESH 2.0
  - With multi-region load imbalance

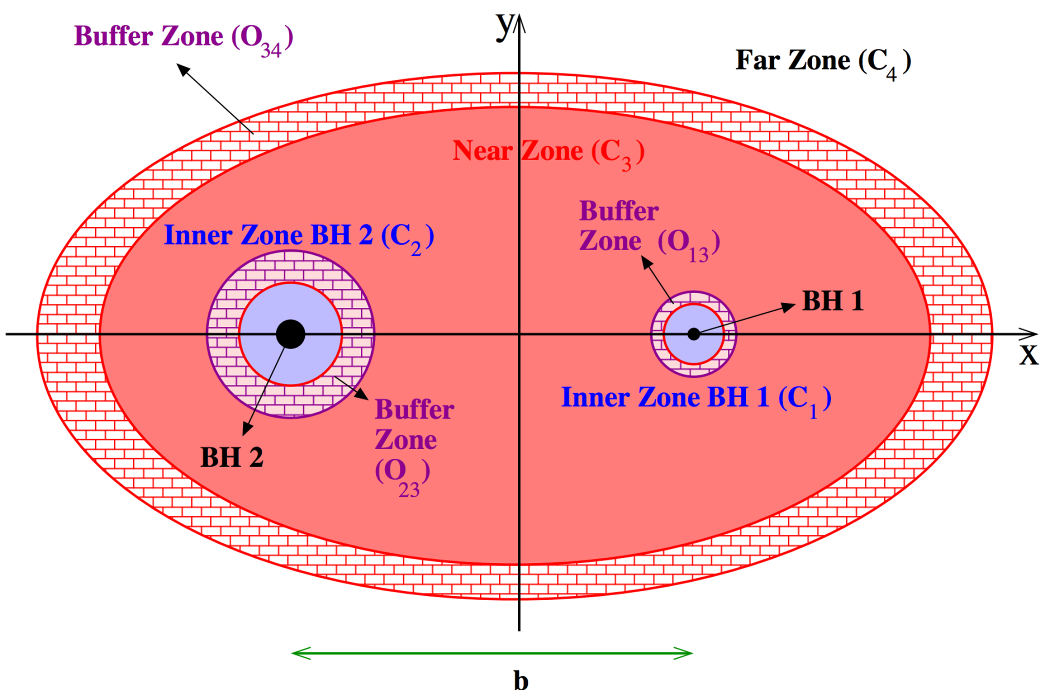


Harm3D

- Collaboration with Scott Noble, Professor of Astrophysics at the University of Tulsa
  - PAID project on Blue Waters, NCSA
- Harm3D is used to simulate & visualize the anatomy of black hole accretions
  - Ideal-Magnetohydrodynamics (MHD) on curved spacetimes
  - Existing/tested code written in C and MPI
  - Parallelized via domain decomposition

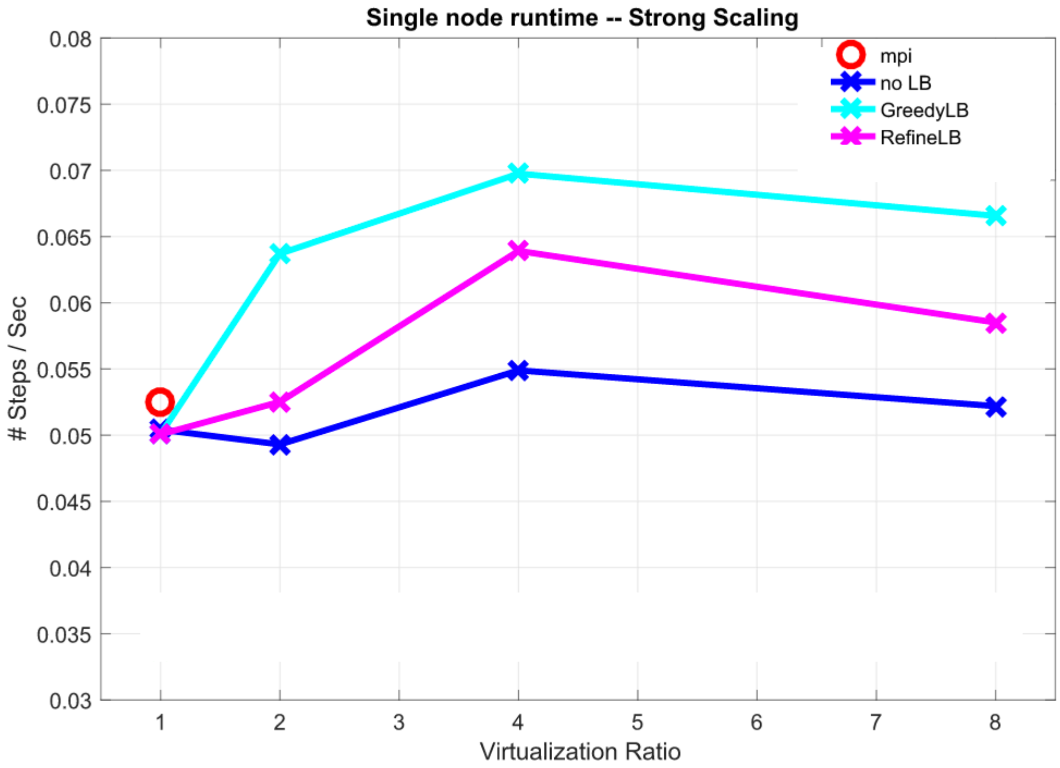
Harm3D

- Load imbalanced case: two black holes (zones) move through the grid
  - 3x more computational work in buffer zone than in near zone



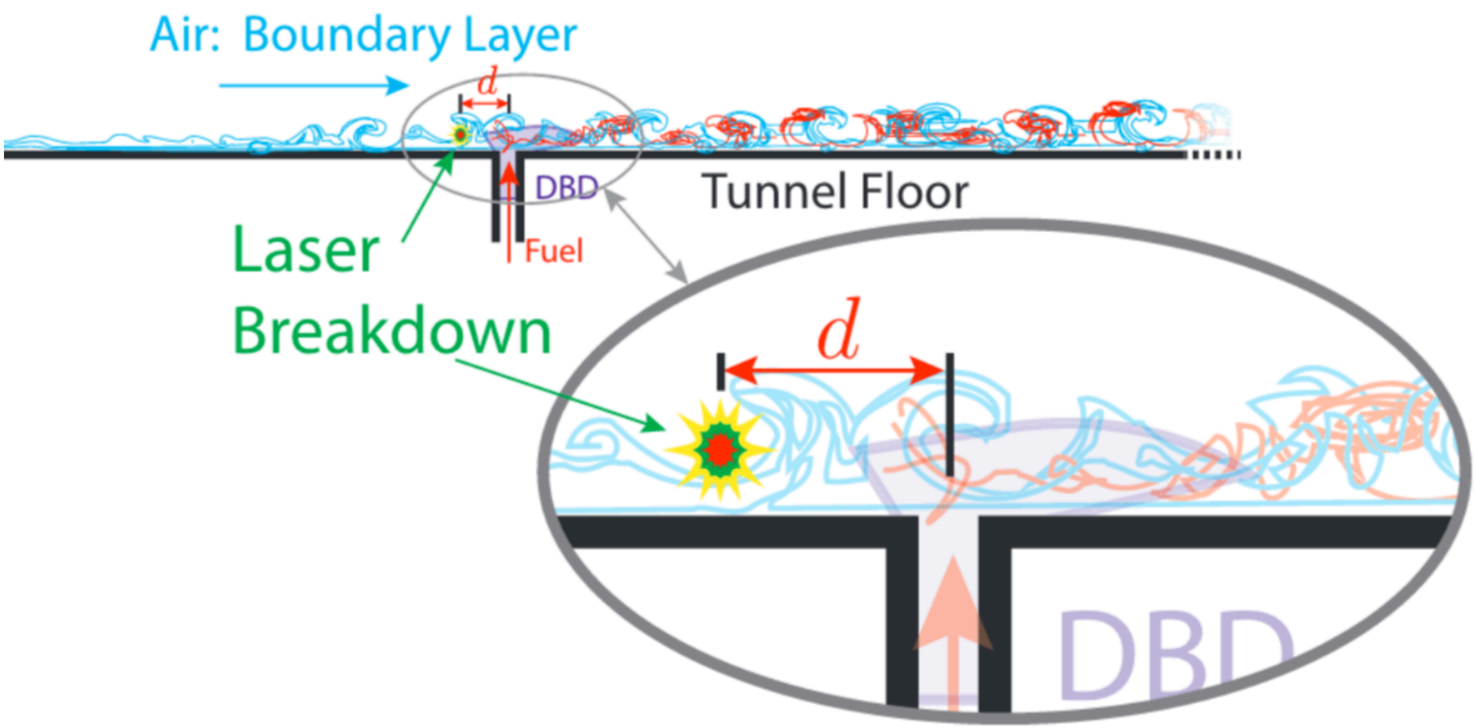
Harm3D

- Recent/initial load balancing results:



PlasComCM

- XPACC: PSAAPII Center for Exascale Simulation of Plasma-Coupled Combustion

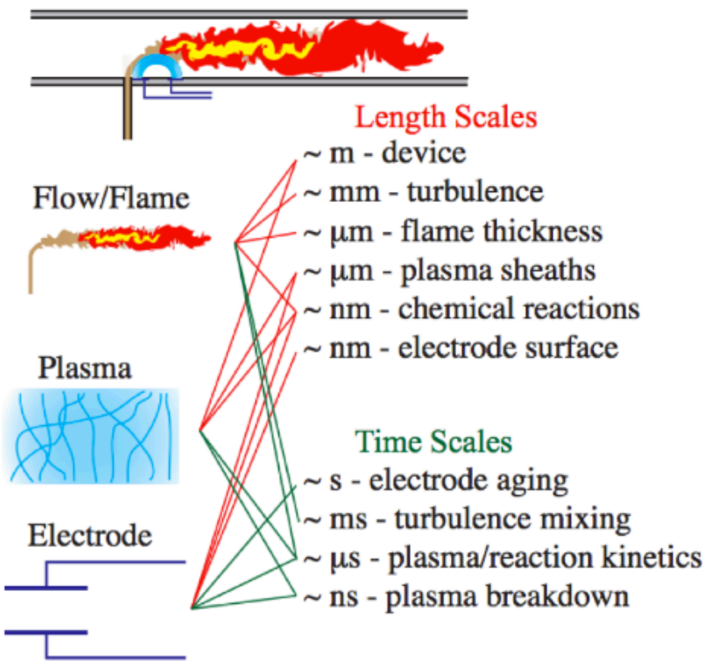


PlasComCM

- The “Golden Copy” approach:
  - Maintain a single clean copy of the source code
    - Fortran90 + MPI (no new language)
  - Computational scientists add new simulation capabilities to the golden copy
  - Computer scientists develop tools to transform the code in non-invasive ways
    - Source-to-source transformations
    - Code generation & autotuning
    - JIT compiler
    - Adaptive runtime system

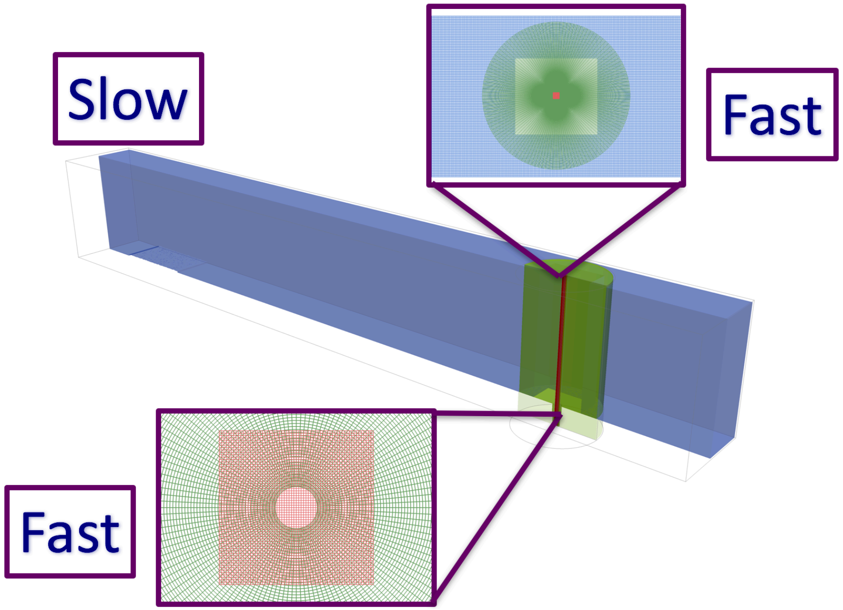
PlasComCM

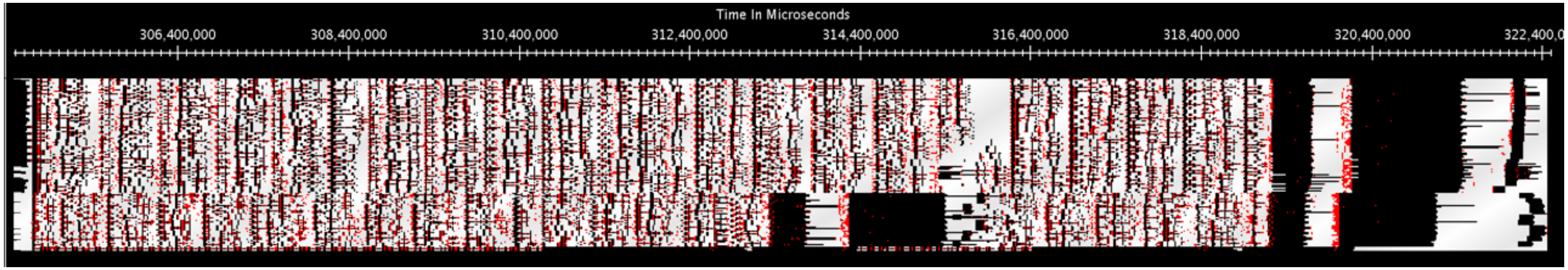
- Multiple timescales involved in a single simulation (right)
  - Leap is a python tool that auto-generates multi-rate time integration code
    - Integrate only as needed, naturally creating load imbalance
    - Some ranks perform twice the RHS calculations of others



PlasComCM

- The problem is decomposed into 3 overset grids
  - 2 ”fast”, 1 ”slow”
  - Ranks only own points on one grid
  - Below: load imbalance





PlasComCM

- Metabalancer
  - Idea: let the runtime system decide *when* and *how* to balance the load
    - Use machine learning over LB database to select strategy
    - See Kavitha's talk later today for details
  - Consequence: domain scientists don't need to know details of load balancing

Ranks	NoLB	GreedyLB	RefineLB	MetisLB	ScotchLB	HybridLB	DistributedLB	Predicted LB
128	1.00	-	-	-	-	-	-	-
256	0.88	1.00	0.90	0.86	0.92	0.87	0.90	1.00 (GreedyLB)
512	0.87	1.00	0.91	0.89	0.96	0.88	0.91	1.00 (GreedyLB)
1024	0.85	0.97	0.89	0.90	1.00	0.87	0.91	0.97 (GreedyLB)

PlasComCM on 128 cores of Quartz (LLNL)

Recent Work

- Conformance:
  - AMPI supports the MPI-2.2 standard
  - MPI-3.1 nonblocking & nbor collectives
  - User-defined, non-commutative reductions ops
  - Improved derived datatype support
- Performance:
  - More efficient (all)reduce & (all)gather(v)
  - More communication overlap in MPI_{Wait,Test}{any,some,all} routines
  - Point-to-point messaging, via Charm++'s new zero-copy RDMA send API

Summary

- Adaptive MPI provides Charm++'s high-level features to MPI applications
  - Virtualization
  - Communication/computation overlap
  - Configurable static mapping
  - Measurement-based dynamic load balancing
  - Automatic fault recovery

- See the AMPI manual for more info.

Thank you

OpenMP Integration

- Charm++ version of LLVM OpenMP works with AMPI
  - (A)MPI+OpenMP configurations on P cores/node:

Notation	Ranks/Node	Threads/Rank	MPI(+OpenMP)	AMPI(+OpenMP)
P:1	P	1	✔	✔
1:P	1	P	✔	✔
P:P	P	P		✔

  - AMPI+OpenMP can do >P:P without oversubscription of physical resources