Application Performance Modeling: Predictive Accuracy in the Presence of Simplifying Abstractions

Kevin J. Barker

With
Darren J. Kerbyson and Scott Pakin

Performance and Architecture Laboratory (PAL)
http://www.c3.lanl.gov/pal
Computer and Computational Sciences Division
Los Alamos National Laboratory

Performance Model Characteristics

Goals for performance modeling (What do we want from a model?):

- Predictive capability
  - » Variations in component performance (network, processor, etc.)
  - » Variations in system size
  - » Variations in network architecture/topology
- Simplicity
  - » Performance models should capture only those elements which actually impact application performance
- Accuracy
  - » How well do the model's predictions compare against measured runtimes on current systems?

$$T_{total} = N_{itr} \cdot \max_{PEs}(N_{cell} \cdot T_{comp} + T_{comm} - T_{overlap})$$

However…

Certain application characteristics are problematic

- Irregular domain partitioning
  - » “Strange” boundaries between processors affect communication volume and neighbor count
  - » Computation impacted by properties of local elements (e.g., material type)
  - » Varying cell counts across processors
- Global domain properties
  - » Ocean simulations with islands of land
- Adaptivity
  - » Neighbor relationships, boundary sizes, and local cell counts all vary over time

How to Model Such Applications?

- **We will look at three applications (Krak, VPIC, and HYCOM)**
  - Computation and communication requirements
    - » Vary across processors
    - » Do not necessarily remain static for length of run
    - » Are determined by characteristics of input deck and are unknown in advance
  - Communication patterns (i.e., neighbor sets) are determined at runtime
  - Input domain itself may be irregular (e.g., holes in the input as in HYCOM)
- **One approach is to develop a model for each processor in the system**
  - Very labor intensive, particularly at large scale
  - Many factors are not known in advance, making model development impossible
  - Would have to reformulate model for each processor count
  - May also need a model for each iteration!

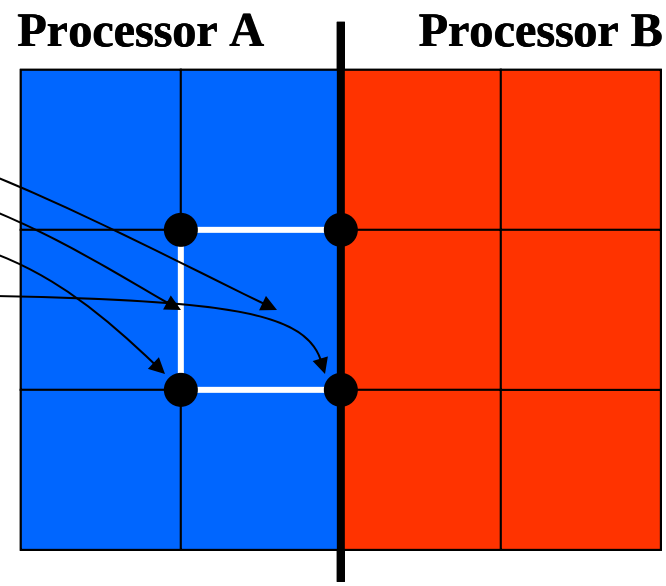
Better Approach: Abstraction

- **Idea is to develop a single model**
  - Describe all processors in the system…
  - …even though each may process a different workload
- **Abstraction trades off potential model accuracy for predictive capability**
  - How much accuracy is potentially lost?
- **Often relies on making key observations about application characteristics at large scale**
  - Predictions tend to become more accurate as processor count increases
  - This is OK, as we are generally interested in modeling performance at large scale

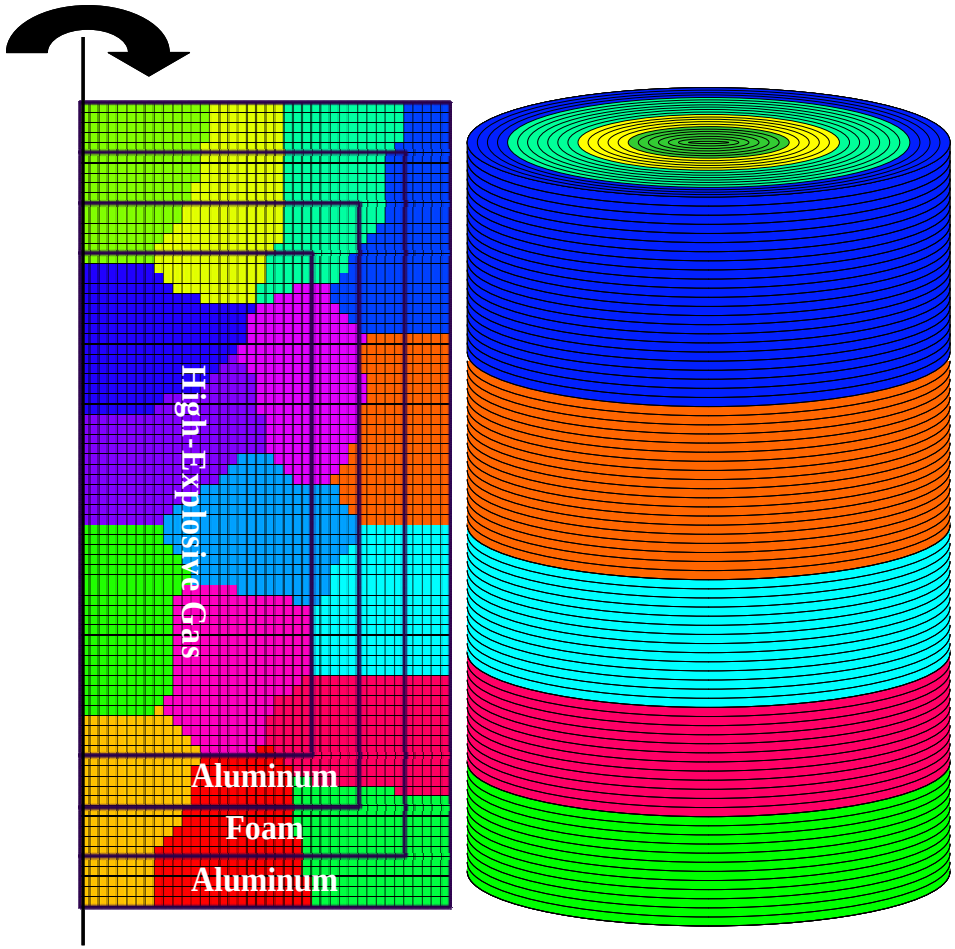
Case Study #1: Krak

- **Production hydrodynamics code developed at LANL**
  - Simulates forces propagating through objects composed of multiple materials
  - >270K lines of code, >1600 source files
  - Object-oriented Fortran dialect
- **Typically executes in strong-scaling mode (fixed global domain size)**

- **Objects mapped onto grid**
  - Grid composed of “cells”
  - Cell defined by “faces”
  - Faces connect “nodes”
  - “Ghost nodes” on PE boundary
- **Processing flow moves through series of time-steps that calculate object deformation caused by high-energy forces**



Krak Input Description



Before Rotation **After Rotation**

- **Three grid sizes studied**
  - Small : 3,200 Cells
  - Medium : 204,800 Cells
  - Large : 819,200 Cells
- **Cells contain one of three material types**
  - Aluminum
  - Foam
  - High Explosive (HE) Gas
- **Regular grid decomposed into irregular subgrids (colors – shown for 16 processors)**
- **Metis partitioning optimized for edge-cuts leads to irregular domain shapes and sizes**

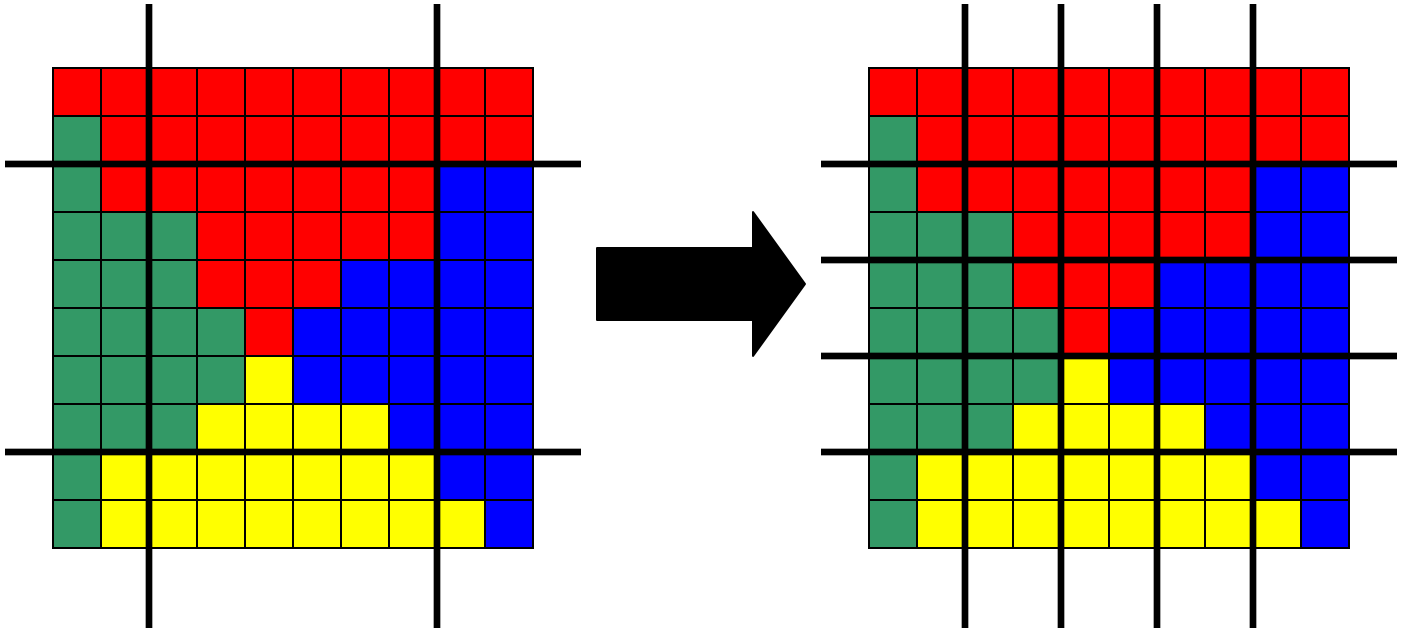
Krak Performance Model

- **Given our per-processor performance model, how do we model Krak?**
  - **Computation**
    - » Per cell computation cost of each material
    - » Number of cells of each material in each sub-grid
  - **Communication**
    - » Boundary length between sub-grids
    - » Collectives
- **All of these are determined by the exact partitioning of the input spatial grid, which cannot be known in advance**
- **Any resulting model would not satisfy goals of simplicity and predictive ability**
- **Abstraction is the key…**

Key Observations

Due to Strong Scaling:

1. **Sub-grids become more homogeneous as system size increases (figure below)**
2. **Assuming each sub-grid to be square is reasonable at large system sizes**



Small System Size **Large System Size**

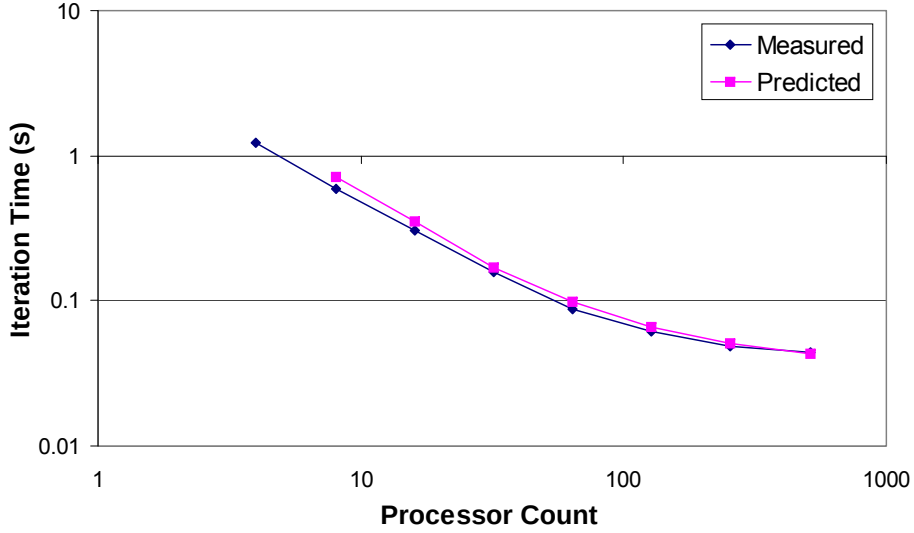
Performance Model

Abstractions result in simplified performance model:

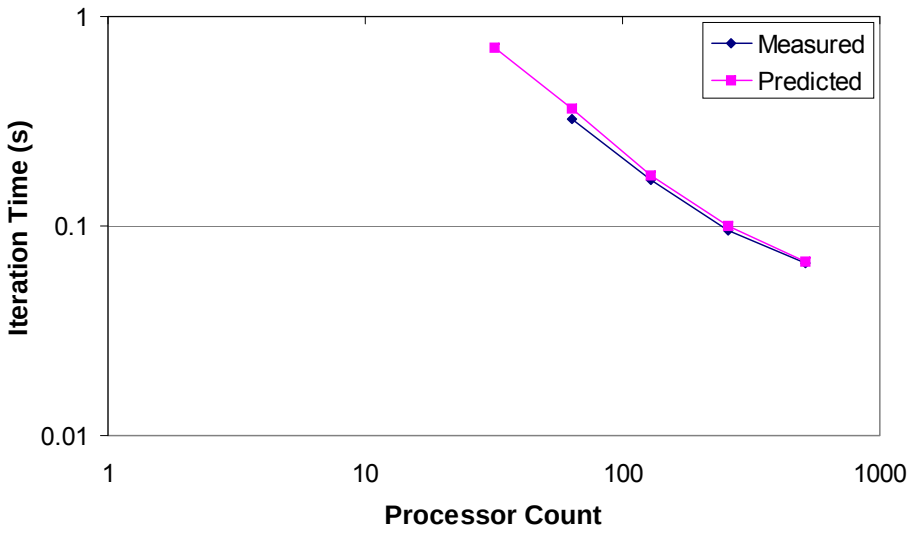
- **Computation**
  - Each subgrid contains the same number of cells
  - All cells are of the most computationally intensive material
  - All subgrids are square in shape
  - Per-cell cost derived from measuring compute times of subgrids of varying sizes
- **Communication**
  - Each subgrid is modeled with four neighbors in 2D
  - All boundaries are the same length
  - All boundary faces touch only a single material
  - Communication consists of boundary exchanges and collectives

Will such abstractions reduce the effectiveness of the performance model?

Performance Model Validation



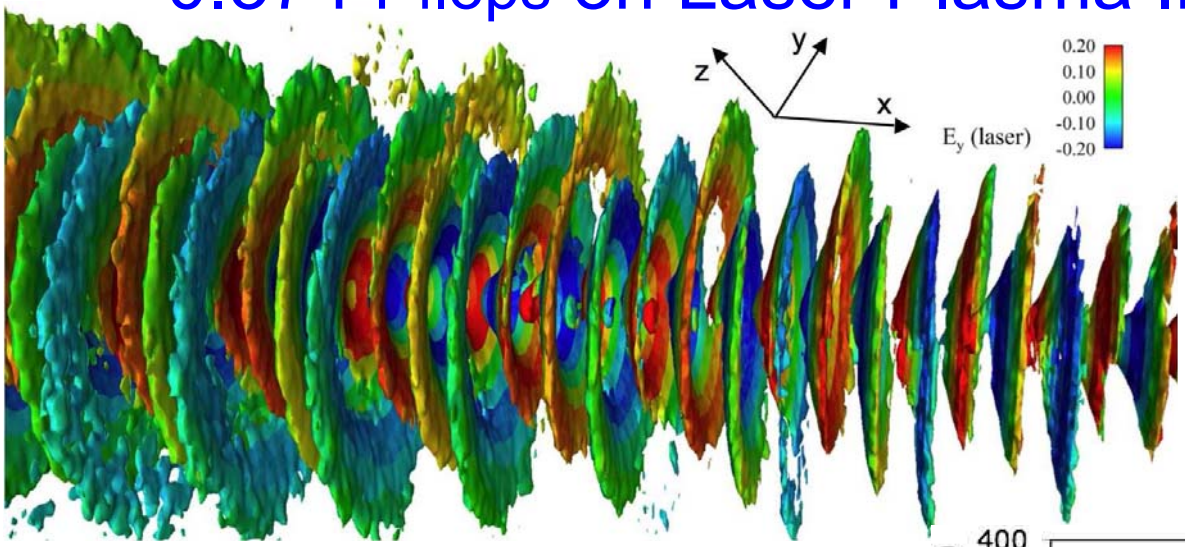
Medium Problem Size



Large Problem Size

- **Measurements taken on 256 node (dual-socket, dual-core 2.0GHz Opteron) cluster connected with Infiniband**
- **Assuming homogeneous material distribution more realistic for large processor counts**
- **Error less than 3% at 512 processors**
- **Communication overheads overwhelm benefits of increased parallelism at large processor counts**

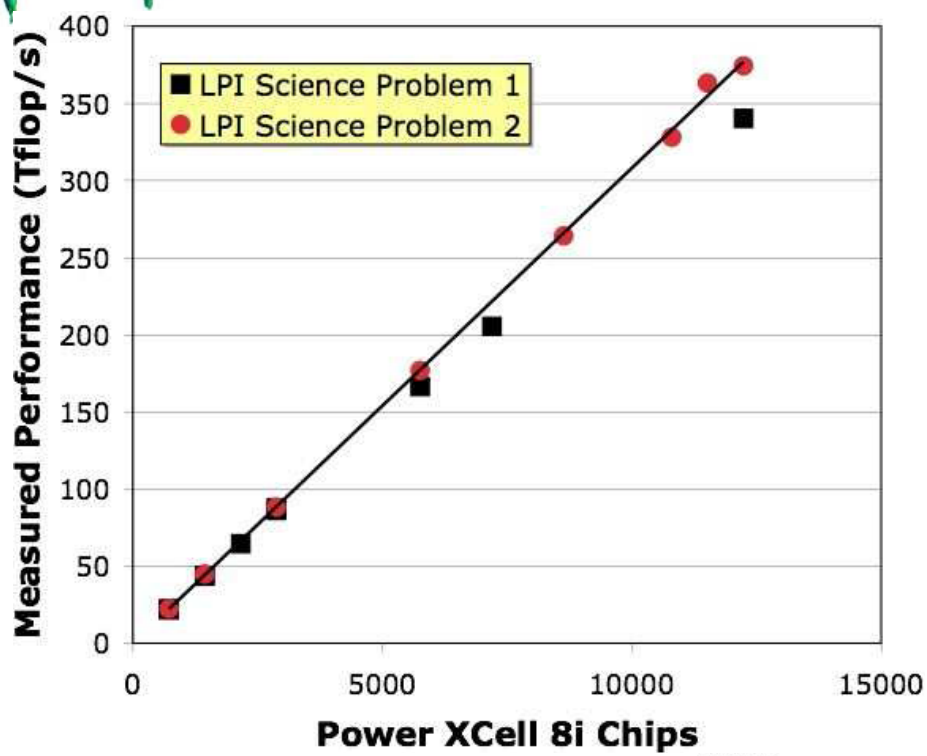
Case Study #2: Modeling VPIC on Roadrunner 0.374 Pflops on Laser Plasma Interaction



3D laser plasma interaction

Isosurfaces of Electric Field
Color indicates laser field

- **Several large-runs on pre-production Roadrunner**
- **Almost linear scaling observed**
  - Overhead of communications is low



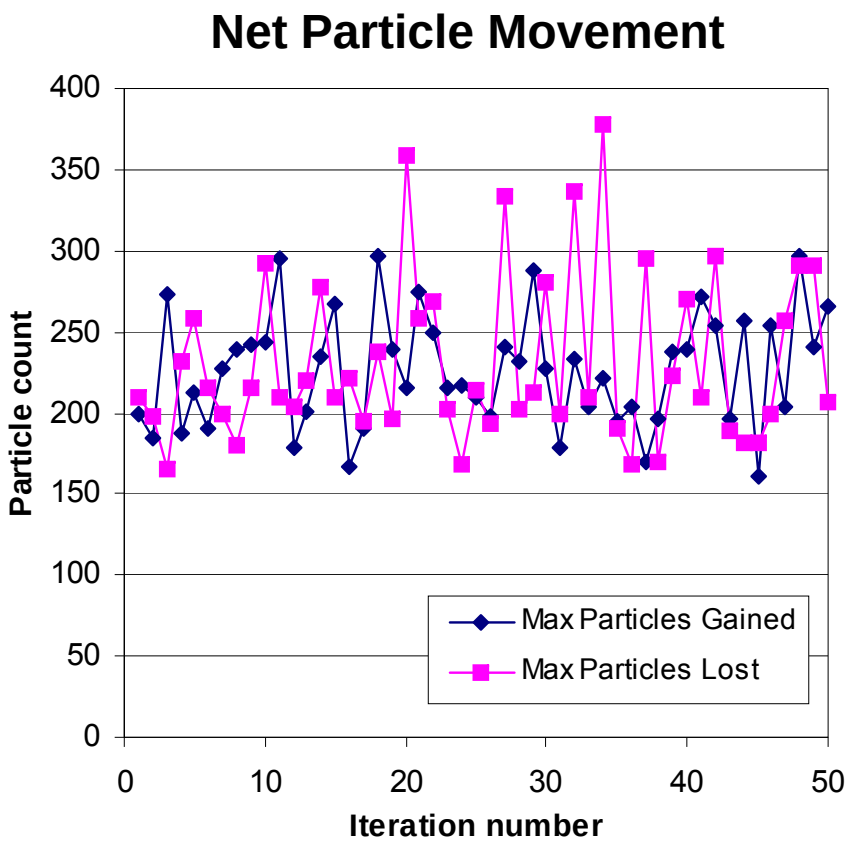
"0.374 Pflop/s Trillion-particle Particle-in-cell Modeling of Laser Plasma Interactions on Roadrunner", Bowers, Albright, Bergen et. Al., Gordon Bell Finalist, SC'08

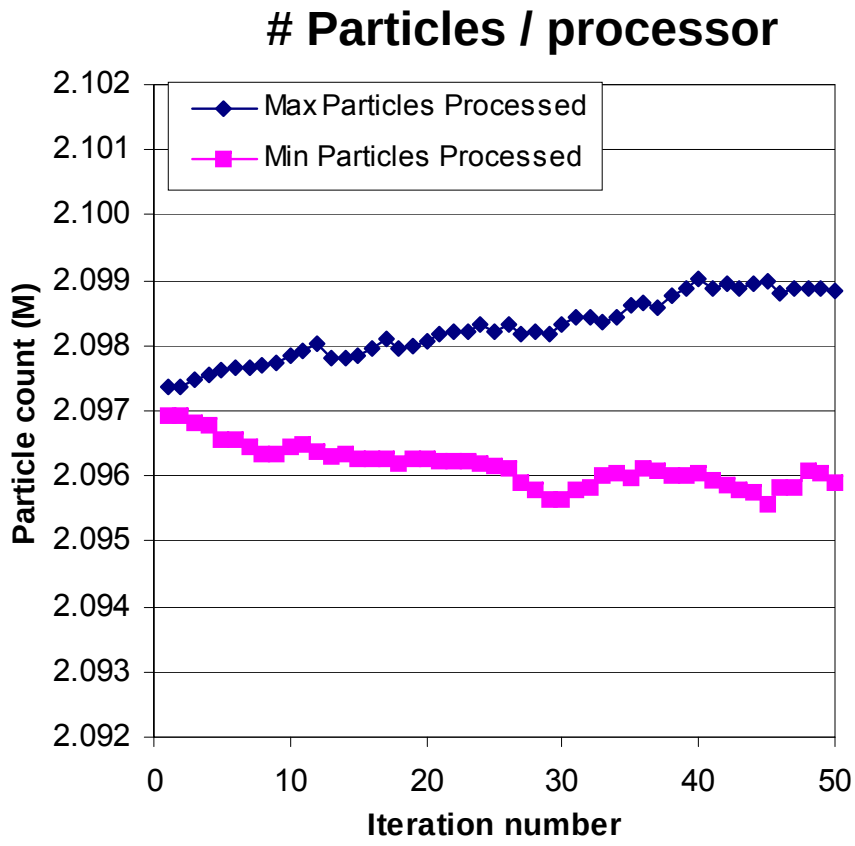
VPIC Overview

- **Plasma Particle-in-Cell**
  - 3-D volume containing multiple particle species (ions and electrons)
    - » Split into Voxels, each contain ~equal # of ions & electrons
  - ions and electrons can move between voxels
    - » some communication per iteration (small-medium sized messages)
  - Parallel Decomposition: in 1-D, 2-D or 3-D
  - Weak-scaling: constant work per processor
  - Periodic particle sorting to aid data layout in memory
- **Roadrunner implementation is Cell-centric**
  - PPE farms out work to SPEs
  - Opterons used for message relay between Cells
- **Benchmark:**
  - 16x16x16 voxels per processor, voxel contains 512 particles / species (2)
- **Laser Plasma Interaction:**
  - 13x14x14 voxels per processor, voxel contains 6420 particles / species (3)

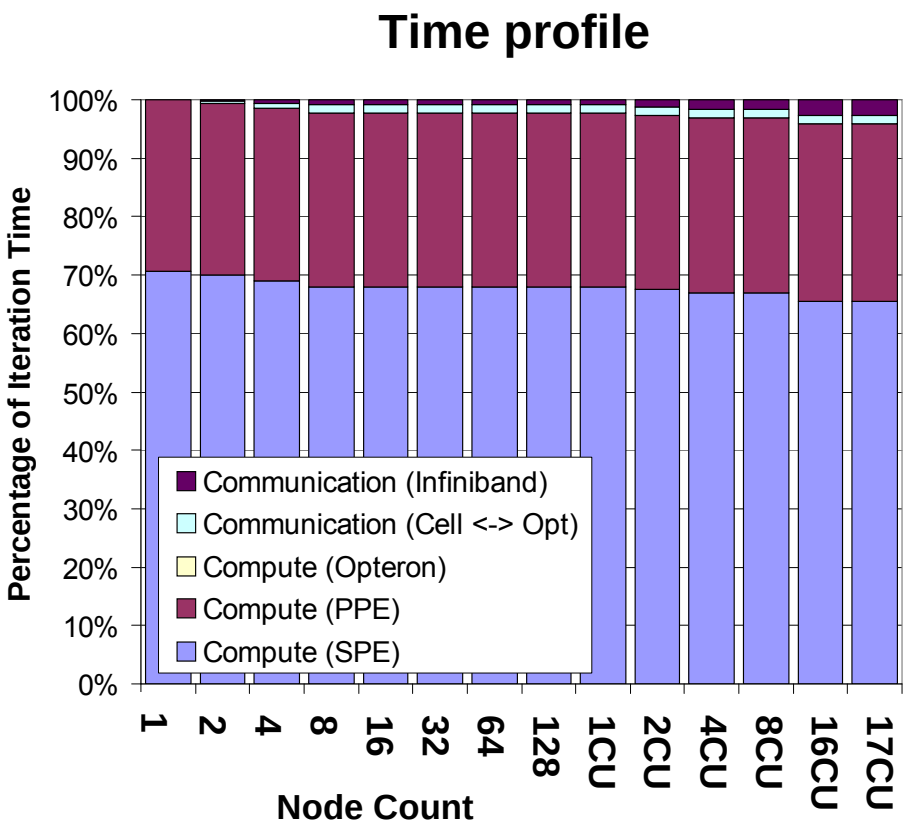
Abstracting Per-Node Particle Count

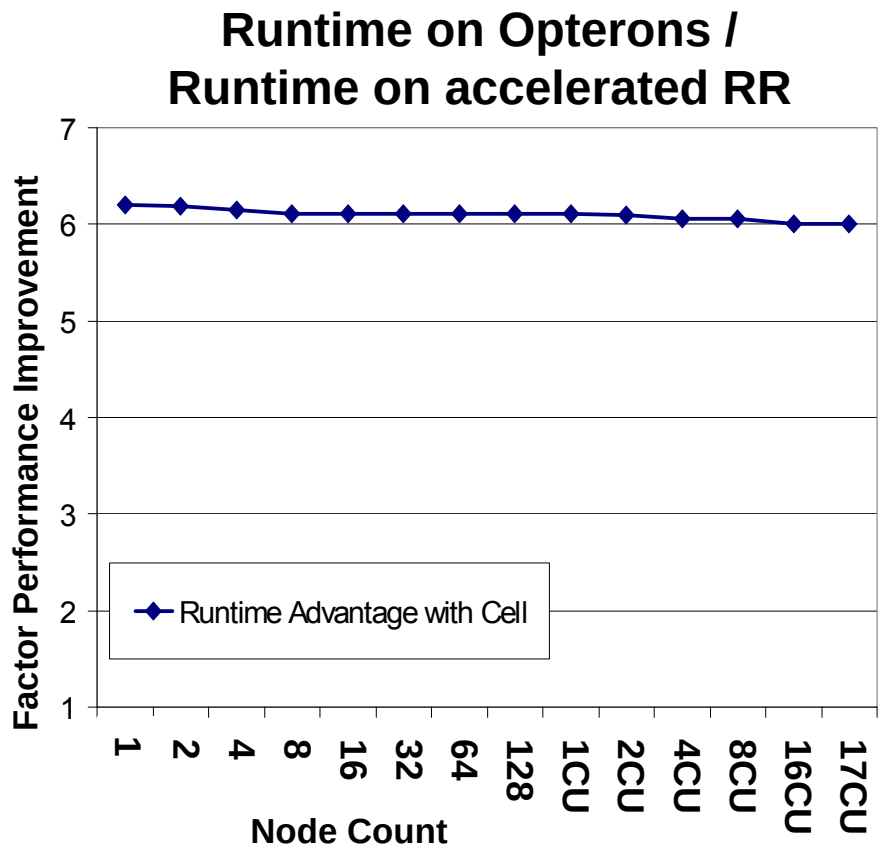
- **Number of particles per processor varies over iterations**
  - Input deck dependent





Initial Performance predictions





- **compute bound**
  - ~65% SPU, ~31% PPU
- **Very little communication overheads**
  - ~1% Cell ↔ Opteron, ~3% Infiniband

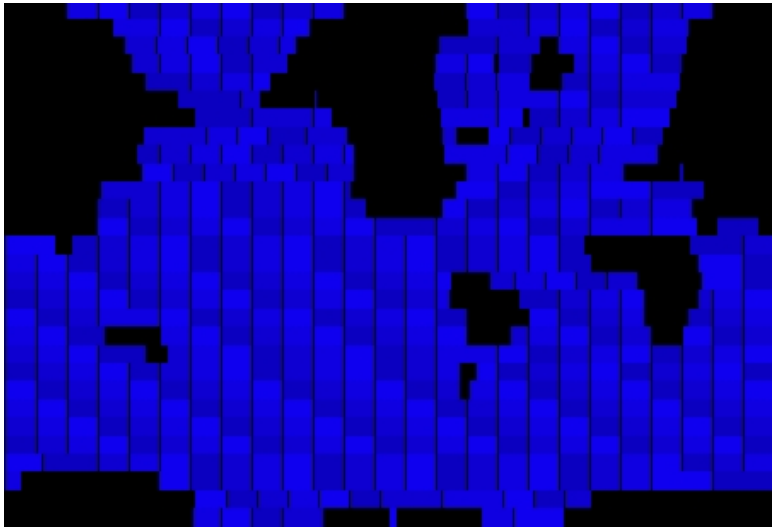
- **Very Good scaling**
- **~6x better performance using Cell (benchmark input)**

Los Alamos NATIONAL LABORATORY EST. 1943

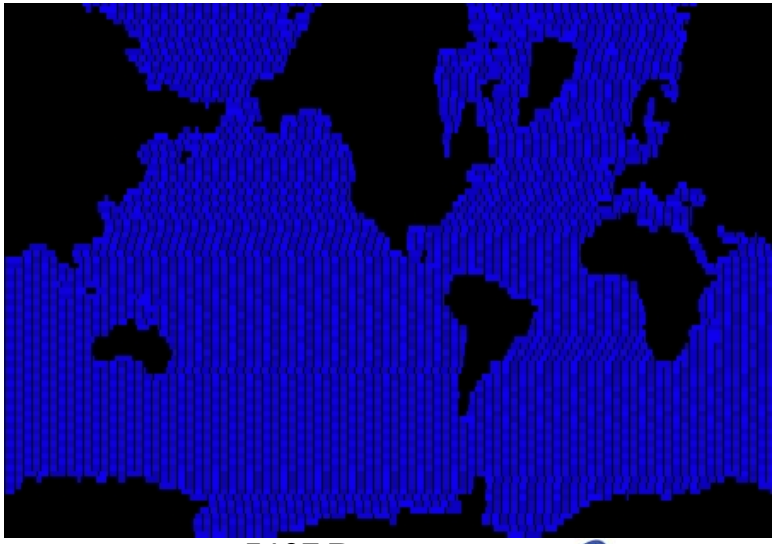
Case Study #3: HYCOM Ocean Model

- **Hybrid vertical (depth) coordinate scheme**
  - Transitions smoothly from deep ocean to shallow coastal regions
- **Parallel data decomposition:**
  - 3D spatial grid partitioned into "tiles" along 2 horizontal dimensions
  - Any tile consisting solely of land is removed
  - Each processor assigned a single (whole or partial) ocean tile
- **Strong scaling mode reduces time to solution for larger PE counts**
- **Approx. 25K lines of Fortran code**

"A Performance Model and Scalability Analysis of the HYCOM Ocean Simulation Application", Kevin J. Barker and Darren J. Kerbyson, Proc. Of IASTED PDCS 2005



504 Processors



5107 Processors

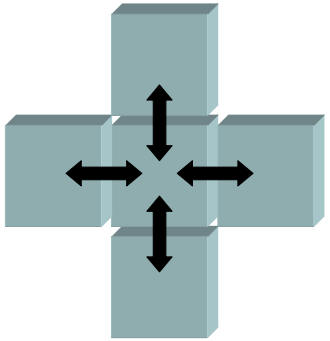
HYCOM Performance Model

Again, performance model has two primary components:

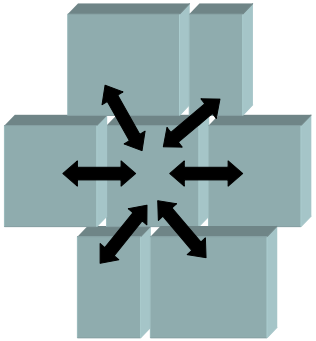
- **Computation**
  - Simple relative to Krak
    - » Computational cost dictated by largest subgrid
    - » Subgrid size is known in advance
  - Fractional subgrids incur idle time
- **Communication**
  - Interprocessor communication required to exchange boundary information between subgrids
  - Regular communication pattern is disturbed by land in the input region

Where do we need abstraction?

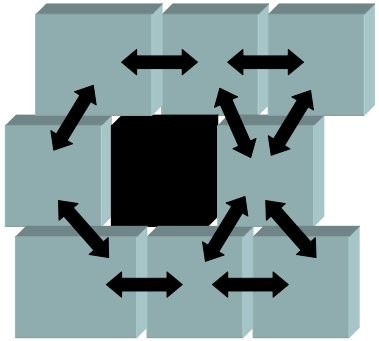
Modeling HYCOM Communication



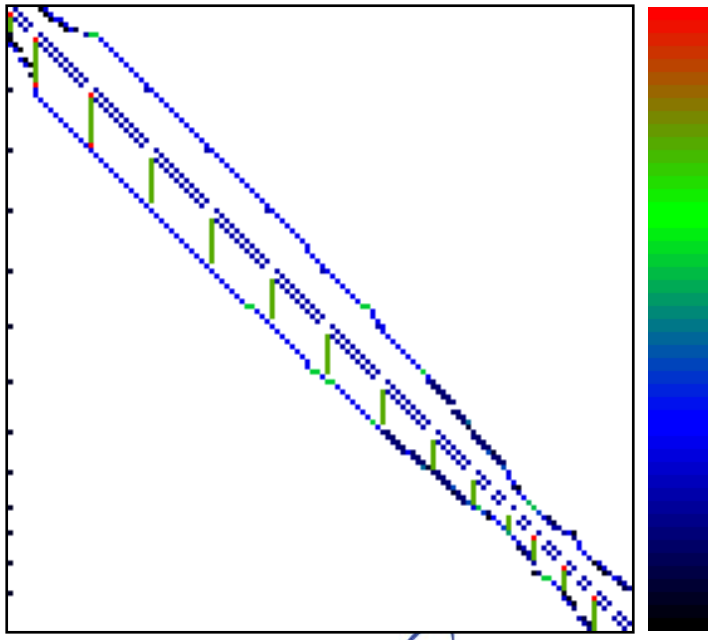
Regular Tile Layout



Irregular Tile Layout



Tile Layout with Gaps Caused by Land



- **2D Boundary Exchanges**
  - Neighbor count varies with tile layout and gaps
  - Msg sizes scale with size of subgrid boundary
  - Neighbor relationships do not span gaps
- **"Software Reductions"**
  - Step 1: Processors communicate with head of row
  - Step 2: Heads of rows communicates with "root" processor
  - How many processors are in each row or column?

Los Alamos NATIONAL LABORATORY EST. 1943

Modeling HYCOM Communication

What abstraction can be applied to simplify the model?

- **Suppose we discount the presence of land**
  - Global domain completely covered by ocean
  - Each processor now has 4 immediate neighbors
  - Each row consists of an equal number of cells
  - All subgrid boundaries are the same size
- **Key observation is that messages are bandwidth bound, even at large scale**

Model Validation

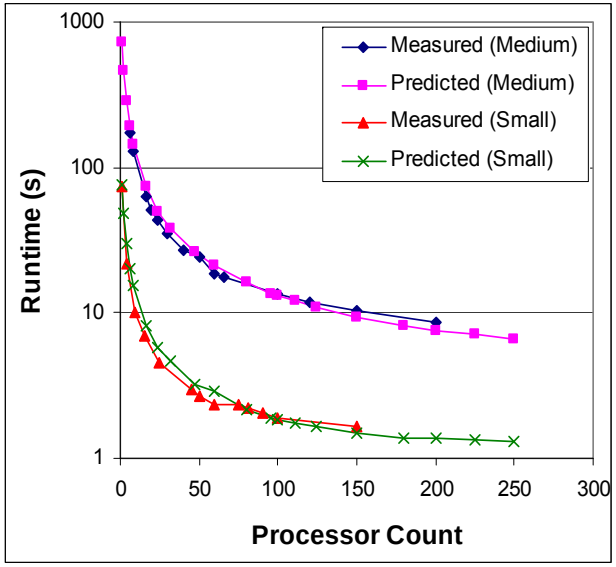
Input Decks:

Input Deck	Oceans	Grid Size (*X* x *Y* x *depth*)	Resolution
Small	Pacific	450x450x22	1/12 degree
Medium	All	1500x1100x26	1/4 degree
Large	All	4500x3298x26	1/12 degree

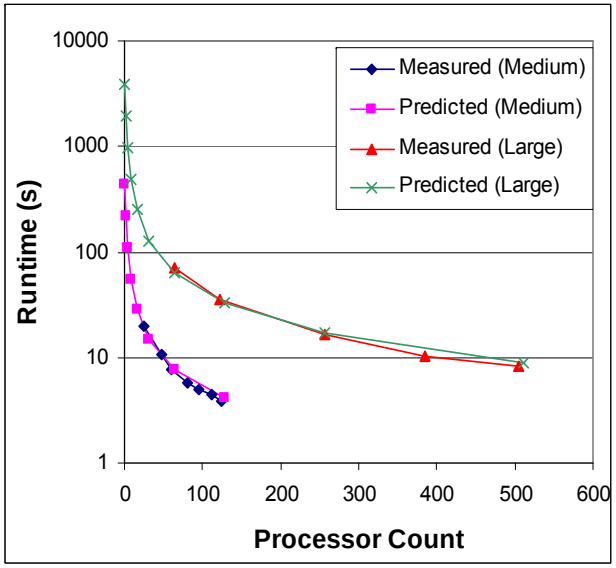
Machine Parameters:

Processor (PE) Type	Clock Speed	PEs/Node	Memory/ PE	Node Count	Network Type	NICs/ Node
HP Alpha EV 68	833 MHz	4	2 Gbytes	50	Quadrics QsNet	1
HP Alpha EV 68	1.25 GHz	4	4 Gbytes	126	Quadrics QsNet	1
Intel Itanium II	1.3 GHz	2	1 Gbyte	30	Quadrics QsNet	1

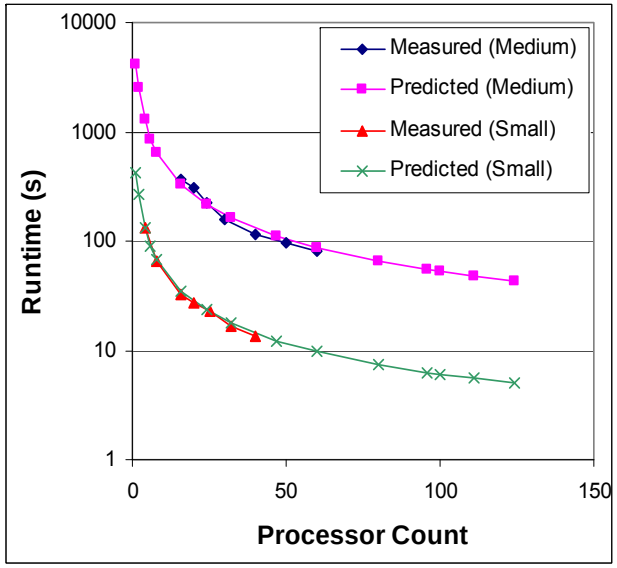
Model Validation



HP Alpha EV 68



HP Alpha EV 68 (2)



Intel Itanium II

System	Input	Mean Error (%)
Alpha EV 68	Small	17
	Medium	12
Alpha EV 68 (2)	Medium	7.7
	Large	7.9
Itanium II	Small	5.6
	Medium	8.5

Single baroclinic + 2 barotropic steps is minimum iteration size

Model typically accurate to within 10% of measurement

Los Alamos NATIONAL LABORATORY EST. 1943

Conclusions

- **Scientific applications are often not regularly partitioned**
  - 3rd party partitioning software
  - Inconsistencies in global domain caused by inhomogeneous features
  - Irregular communication patterns
  - Subdomain variations over time
- **Iteration time of loosely synchronous applications will be determined by the slowest process**
- **Assuming regularity can simplify modeling process**
  - Computational load across cells is homogeneous
  - Interprocessor communication pattern is the same everywhere
- **Accuracy is not negatively impacted, particularly at large scale**
  - Irregularity approximates regularity at large scale