A Practical Approach to Performance Analysis and Modeling of Large-Scale Systems

Kevin J. Barker, Adolfy Hoisie, Darren J. Kerbyson

Fundamental & Computational Sciences Directorate

Pacific Northwest NATIONAL LABORATORY

Proudly Operated by Battelle Since 1965

Tutorial Outline (the plan!)

	Page	Duration
Introduction and motivation		**20 mins**
Performance metrics & pitfalls		**30 mins**
Performance modeling methodology		**40 mins**
COFFEE BREAK		**30 mins**
Abstractions		**30 mins**
Case Studies		
I: SWEEP3D		**60 mins**
LUNCH BREAK		**90 mins**
II: SAGE		**30 mins**
III: DNS3D		**30 mins**
Applications of modeling		
I: Rational system integration		**30 mins**
COOKIE BREAK		**30 mins**
II: Novel Architectures: Blue Waters		**40 mins**
III: Performance comparison of large-scale systems		**40 mins**
Conclusions, lessons learned, wrap-up		**10 mins**

Introduction and Motivation

" 20% of a project's time is spent in trying to understand what to build, 80% is spent building it, and no time is spent trying to understand deeply, how well the design decisions were made in terms of performance delivered to users, and hence, how to proceed on the next system design."

- David Kuck,
Kuck & Associates, Inc. and
Univ. of Illinois, Emeritus
"High-Performance Computing"
Oxford U. Press, 1996

What is This Tutorial About?

- Performance modeling
  - Analytical techniques that encapsulate performance characteristics of applications and systems and enable a predictive capability
  - Techniques developed at LANL
  - Emphasis on full applications
  - No dependence on specific tools
    - » Although data collection is vital
- Applications of performance models: performance prediction
  - Tuning roadmap for current bottlenecks
  - Architecture exploration for future systems
  - Software / algorithm changes
  - System installation diagnostics: "Rational System Integration"
  - Result: replace benchmarks with models

What is This Tutorial Really About?

- Insight into performance issues
  - Performance modeling is the only practical way to obtain *quantitative* information on how to map real applications to parallel architectures rapidly and with high accuracy
- With this insight you become a more educated buyer/seller/user of computer systems
  - Help you become a "performance skeptic"
  - Show how to integrate information from various levels of the benchmark hierarchy
  - Show why "naïve" approaches sometimes don't work

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Why Performance Modeling?

- Other performance analysis methods fall short in either accuracy or practicality:
  - Simulation (UCLA, Dartmouth, UIUC)*
    - » Greatest architectural flexibility but takes too long for real applications
  - Trace-driven experiments (UIUC, Barcelona)*
    - » Results often lack generality
  - Benchmarking (~ everybody)
    - » Limited to current implementation of the code
    - » Limited to currently-available architectures
    - » Difficult to distinguish between real performance and machine idiosyncrasies

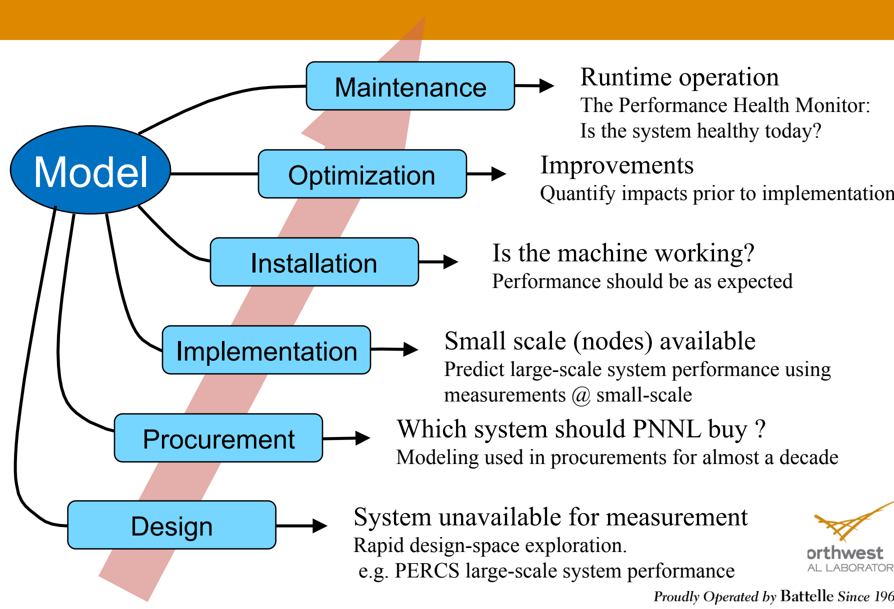
* Partial lists

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Why Performance Modeling?

- Parallel performance is a multidimensional space:
  - Resource parameters: # of processors, computation speed, network size/topology/protocols/etc., communication speed
  - User-oriented parameters: Problem size, application input, target optimization (time vs. size)
  - **These issues interact and trade off with each other**
- Large cost for development, deployment and maintenance of both machines and codes
- Need to know in advance how a given application utilizes the machine's resources

Why Performance Modeling?



Why Performance Modeling?

- Go beyond what traditional performance tools offer
- Traditional tools tell you "what program did" and "when it did it" - profilers
- We are "tools-neutral"
  - You choose (TAU, PABLO, PARADYN, VAMPIR, PAPIPROF, etc.)
- The performance model is the tool
  - But modeling cannot be fully automated
- Many uses
  - Isolate bottlenecks
  - Plan ahead with "What if?" scenarios by varying problem size, network parameters, computation speed, etc.

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Why Performance Modeling?

- From the application-centric point of view: workload characterization
- Is the application sensitive
  - to network bandwidth?
  - to network latency?
  - to computation speed?
- What would the speedup be if we used a different parallel decomposition method?
  - Give an indication of performance improvement before investing the effort to recode
  - Ultimately, performance-engineer applications from design phase

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Modeling Successes

- Machines
  - ASCI Q
  - ASCI BlueMountain
  - ASCI White
  - ASCI Red
  - CRAY T3E
  - Earth Simulator
  - Itanium-2 cluster
  - BlueGene/L
  - BlueGene/P (early design)
  - CRAY X-1
  - ASC Red Storm
  - ASC Purple
  - IBM PERCS
  - IBM Blue Waters
  - AMD-based clusters
  - Clearspeed accelerators
  - SiCortex SC5832
  - Roadrunner
- Codes
  - SWEEP3D
  - SAGE
  - TYCHO
  - Partisn
  - LBMHD
  - HYCOM
  - MCNP
  - POP
  - KRAK
  - RF-CTH
  - CICE
  - S3D
  - VPIC
  - GTC

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Performance Modeling Process

- Basic approach:

$$T_{run} = T_{computation} + T_{communication} - T_{overlap}$$
$$T_{run} = f\,(T_{1\text{-}CPU}\ ,\ \text{Scalability})$$

  where $T_{1\text{-}CPU}$ is the single processor time

- We are not using first principles to model single-processor computation time.
  - Rely on measurements for $T_{1\text{-}CPU}$. May be:
    - » time per subgrid,
    - » time per cell,
    - » calculated using measured rate and # of FLOPS per subgrid

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Performance Modeling Process

- Simplified view of the process
  - Distill the design space by careful inspection of the code
  - Parameterize the key application characteristics
  - Parameterize the machine performance characteristics
  - Measure using microbenchmarks
  - Combine empirical data with analytical model
  - Iterate
  - Report results
- The huge design space requires careful choice of metrics
  - Reporting results itself requires a methodology.
- With all this in mind, here is the tutorial outline:

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Tutorial Outline (the plan!)

Performance Metrics

"Planet's Largest Supercomputer Accepted After Rigorous Tests"

- *Headline, Los Alamos National Laboratory* Newsbulletin, *date unknown*

-"XYZ to Increase Price/Performance with the support of the new 64-bit Intel Xeon"

- "Based on the 9.6 GHz XYZ processor, code named XYZ, the new server family has achieved eight world record benchmarks "

Why the Great Interest in Performance Metrics?

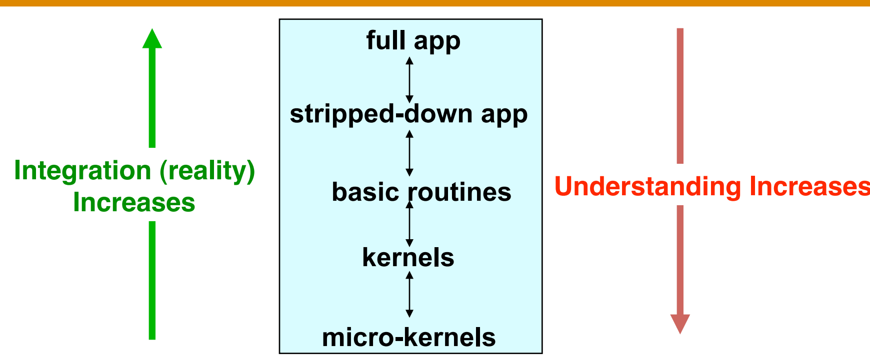
- Reliance on performance metrics is tempting because:
  - Metrics appear to allow performance to be distilled into a single number
    - System *X* capable of peak performance of *N* Tflop/s
  - Metrics appear to allow rapid comparisons between systems
    - System *X* achieves 30% higher performance on LINPACK than System *Y*
  - Metrics appear to yield intuitive insight into system performance
- However…

Be Skeptical of Performance Metrics

- There are so many metrics out there
  - Some indication of the complexity of parallel application performance
- Creating metrics to describe parallel performance is hard
  - Metrics describe only *aspects* of total performance
    - » Total system performance is impacted by many components (compute speed, network performance, memory performance, etc.)
    - » But we are ultimately interested in achievable application performance!
- Performance metrics are easily abused
  - E.g., Flop/s easily manipulated with problem size
- To get the full picture, a workload-specific performance model is necessary!

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Metrics Trade Realism for Understanding



Micro-kernels:

- Attempt to generalize performance
  - May represent characteristics of a large number of applications
- Are the easiest to understand and discuss
  - But this is a poor representation of reality!

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

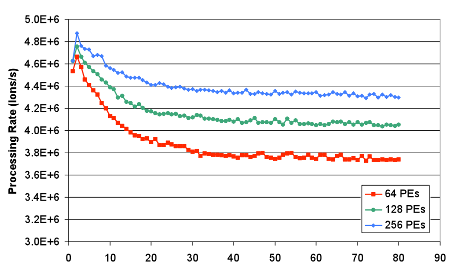
Types of Metrics: Direct Measures

- Absolute time
  - Difference between start and finish
    - » Measured as maximum dedicated wall-clock time over all processors
      - But what constitutes "dedicated?"
      - Easiest metric to measure
  - Best performance measure for
    - » Tracking performance improvements
    - » Comparisons between systems for the same app
    - » Historical comparison when the application is "frozen"
  - But tells us little about how well the resources are being used
    - » Cannot be used to predict performance
      - Due to architectural changes
      - Due to software changes
    - » Does not give any performance insight!

Types of Metrics: Direct Measures

Processing Rate: operations per unit time

- Application-specific rates:
  - » E.g., cells processed per unit time
  - » Careful! Are we talking about compute time only or total time?
    - May be difficult to separate computation and communication times
    - Rates may change with system size due to parallel overheads
    - Rates may also change with problem size due to memory effects



GTC – Plasma modeling code: Stride through memory varies with processor count, incurring TLB miss penalties at 64 processors!

Beware of Flop/s – this is often unreliable!

Types of Metrics: Indirect Measures

- Performance improvement
  - % Improvement in some metric due to some feature
  - Normalized time (be careful if you average)
- Efficiency
  - (typically observed rate / peak rate)
- Scalability / Speedup
  - Performance improvement due to parallelism
- Other indirect measures of performance
  - Cache hit/miss ratio, % vectorization, average vector length, % parallel, etc.
- Difficult but important: Cost / performance
  - Cost too difficult to "measure" so we concentrate on performance
  - Should be *actual* cost / *actual* performance

Pacific Northwest
NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Common Metrics: Flop/s

- Number of floating-point operations / time
- Problems:
  - Can be artificially inflated (by algorithm, code by compilation)
  - Single precision or double precision?
  - No convention for counting flops & flop instruction sets differ:
    - » A = B * C + D? A = A + B * C A = B * C? A = B? A = A / B?
  - Need an unambiguous means to measure # of flops
    - » Relates to workload hierarchy (easiest for lowest levels)
    - » Still doesn't work well for codes with small numbers of flops
- Use with care!
  - Not useful for comparing amongst machines
  - Not useful for comparing different apps
  - May be useful for providing utilization of a given machine

Pacific Northwest
NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Common Metrics: Efficiency

- Measure of how well resources are being used
- Of limited validity by itself
  - Can be artificially inflated
  - Biased towards slower systems
- Example 1: Efficiency of applications

	Solver Flops	Flops	Mflop/s	% Peak	Time (s)
Original	64 %	29.8×10^9	448.8	5.6 %	66.351
Optimized	25 %	8.2×10^9	257.7	3.2 %	31.905

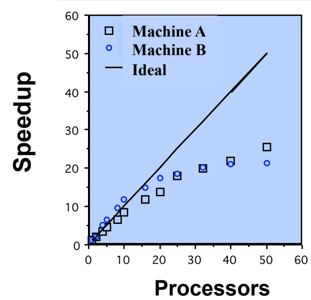
- Example 2: Efficiency of systems
  - SAGE (timing_b) on SGI Origin2000
    - (250 MHz, 500 MFLOPS Peak per CPU, 2 FLOPS per CP):
    - Time = 522 sec.; MFLOPS = 26.1 (5.2% of peak)
  - SAGE (timing_b) on Itanium-2
    - (900 MHz, 3600 MFLOPS Peak per CPU, 4 FLOPS per CP):
    - Time = 91.1 sec; MFLOPS = 113.0 (3.1% of peak)

Pacific Northwest
NATIONAL LABORATORY

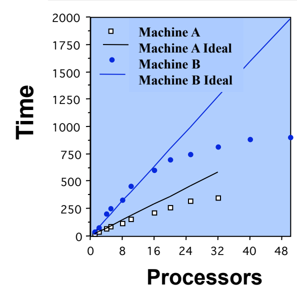
Proudly Operated by Battelle Since 1965

Common Metrics: Speedup

Relative performance:



Absolute performance:



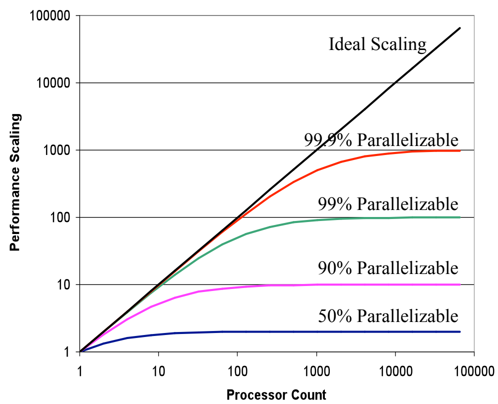
Speedup is only one characteristic of a program – it is not synonymous with performance.
In this comparison of two machines the code achieves comparable speedups but one of the machines is faster.

Pacific Northwest
NATIONAL LABORATORY

Proudly Operated by Battelle Since 1965

Amdahl's Law and Speedup

- Amdahl's Law bounds the speedup due to any improvement
  - $S = T/T' = 1/[f_v/r + (1-f_v)]$
  - Example: What will the speedup be if 20% of the exec. time is in inter-processor communications which we can improve by 10X?
    $S = T/T' = 1/ [.2/10 + .8] = 1.22$ (i.e., 22% speedup)



- Amdahl's Law forces diminishing returns on performance
  - Invest resources where time is spent
  - The slowest portion will dominate
  - Cannot sustain linear speedup
- Amdahl's Law + Murphy's Law: If any system component can damage performance, it will!

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

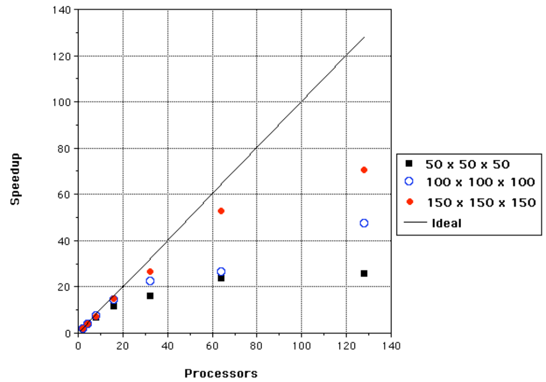
Application Scaling

- Strong Scaling
  - Motivation
    - » What is the largest # of procs I can effectively utilize?
    - » What is the fastest time I can solve a given problem?
  - Global problem remains constant; subgrid size decreases with *P*
    - » Memory requirements decrease with *P* - super-linear speedup?
    - » Surface-to-volume ratio increases with *P*
- Weak Scaling
  - Want to use a larger machine to solve a larger problem *in the same time*
  - Global problem size grows proportionally with *P*
    - » Per-node memory requirements stay constant
    - » Surface-to-volume ratio may remain constant
  - Ideally, time to solution remains constant
    - » Linear speedup possible, but only in terms of available parallelism
    - » Other overheads may increase with *P*, e.g., collectives

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Strong Scaling: Sweep3D

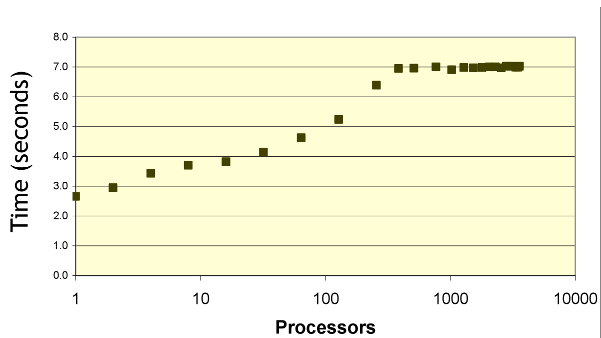
Manipulating problem sizes can also manipulate observed performance



Problem sizes should be chosen to reflect workload, not to portray the machine in its best light!

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Weak Scaling: SAGE



- Weak scaling would indicate runtime should remain constant
- However, characteristics of SAGE prevent ideal weak scaling at small scale
  - SAGE is highly optimized; this is not a defect of the code
- Cannot rely on speedup (or any other metric) alone to understand performance!

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Common Pitfalls: Unrealistic Problem Size

- Is the problem you are studying sensible?
  - Beware of benchmarks that use unrealistically large problem sizes
    - » This tends to improve parallel efficiency (mask parallel overheads with large amounts of computation)
- Is the problem being run in the appropriate scaling mode?
  - This will impact computation/communication ratio at large scale
- This is more than a quantitative difference
  - At scale, applications that are actually communication bound can appear computation bound

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Simple Metrics Don't Give the Whole Story

- The problem is not the metrics themselves but how they are used
- It is always dangerous to use a single metric by itself
  - This is especially true when examining relative performance
    - » How does System *A* compare with System *B*?
  - Keep in mind that micro-kernels and benchmarks only approximate reality
    - » Application performance may be markedly different

To gain true insight into application performance, a performance model is necessary

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Summary

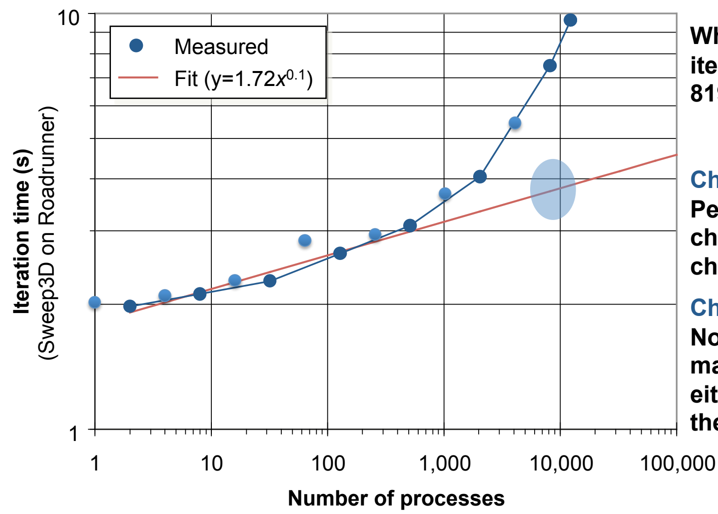
- Metrics can be useful for gleaning insight into system/application performance
  - Distill complex performance information into a single number
  - Don't necessarily represent reality, so be careful
- The large number of metrics indicates the complexity of analyzing performance of parallel codes and systems
- Some common pitfalls:
  - Confusing speedup, flop rate, efficiency with absolute performance
  - Ignoring Amdahl's law by assuming sustainable linear speedup
  - Using an unrealistic problem size
- As you move further away from reality (i.e., micro-kernels instead of applications), you must exercise more care in interpreting results!

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Tutorial Outline (the plan!)

	Page	Duration
Introduction and motivation		**20 mins**
Performance metrics & pitfalls		**30 mins**
Performance modeling methodology		**40 mins**
COFFEE BREAK		**30 mins**
Abstractions		**30 mins**
Case Studies		
I: SWEEP3D		**60 mins**
LUNCH BREAK		**90 mins**
II: SAGE		**30 mins**
III: DNS3D		**30 mins**
Applications of modeling		
I: Rational system integration		**30 mins**
COOKIE BREAK		**30 mins**
II: Novel Architectures: Blue Waters		**40 mins**
III: Performance comparison of large-scale systems		**40 mins**
Conclusions, lessons learned, wrap-up		**10 mins**

What Makes Performance Prediction Challenging?



What's the expected iteration time of an 8192-process run?

Challenge #1
Performance characteristics may change at scale.

Challenge #2
Nonlinear behavior may be caused by either the system *or* the application.

Curve fitting does not provide performance *insight*!

Pacific Northwest NATIONAL LABORATORY
Operated by Battelle Since 1965

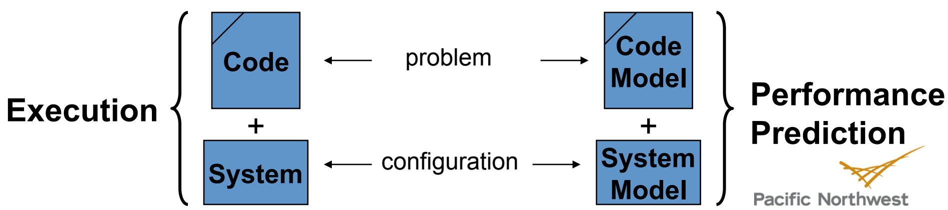
What is a Performance Model?

- Analytical expression of performance in terms of application and system characteristics
  - May be embodied as mathematical formulas, Excel spreadsheets, Perl scripts, etc. (It doesn't matter.)
- Precise description of an application in terms of system resources
  - Which resources substantially determine execution time?
    CPU speed/core count, network latency/bandwidth/topology, memory hierarchy sizes/speeds, …
  - When is each resource used?
    during an iteration, between iterations, every nth iteration, …
  - What determines how much each resource is used?
    processor count, memory capacity, physics modules included, …

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Attributes of a Performance Model

- Succinctly encapsulates application behavior
  - Abstracts application into communication and computation components
  - Focuses on first-order effects, ignoring distracting details
- Separates performance concerns
  - Inherent properties of application structure (e.g., data dependencies)
  - System performance characteristics (e.g., MPI latency)



Approach for Modeling a System

- Focus on first-order effects
  - No need to know performance of each transistor, line of firmware, etc.
  - Concentrate on factors that impact application performance
- Split modeling effort into two components:
  - Single-processor execution time
  - Scaling properties
- Single-processor execution time
  - For simplicity, treated as an input to the performance model
  - May be determined by actual execution, simulation, estimates based on similar processors, or other approaches
- Scaling properties
  - Core part of this tutorial
  - What aspects of a system are important at scale?
  - Processor count, network topology, messaging performance, communication-offload capabilities, scaling of collective operations

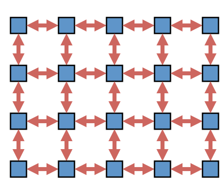
Approach for Modeling an Application

- Focus on first-order effects
  - No need to know performance of each line of code or CPU instruction
  - Concentrate on factors that impact performance
- Parameterize computation and communication patterns
  - Number of cells (or other unit of computation) per process?
  - Work per cell? *Constant? Function of cell count?*
  - Communication peers? *1D/2D/3D nearest neighbor? Gray code?*
  - Bytes/messages per peer? *Constant? Function of cell count?*
  - Collective communication type/frequency? *Reduction every iteration? All-to-all every N iterations?*
- White-box approach
  - Determine the above with instrumentation, profiling, and experimentation
  - Confirm by examining source code
  - (If you're the author, you may already know many of the answers)

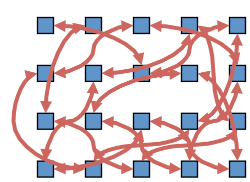
Pacific Northwest
NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Commonly Encountered Application Characteristics

- Single Program Multiple Data (SPMD) execution
  - Each process runs the same code but on different segments (called *subgrids*) of a global data structure
- Local, logical neighbor communication
  - Boundary data at subgrid edges is communicated between processes



Common

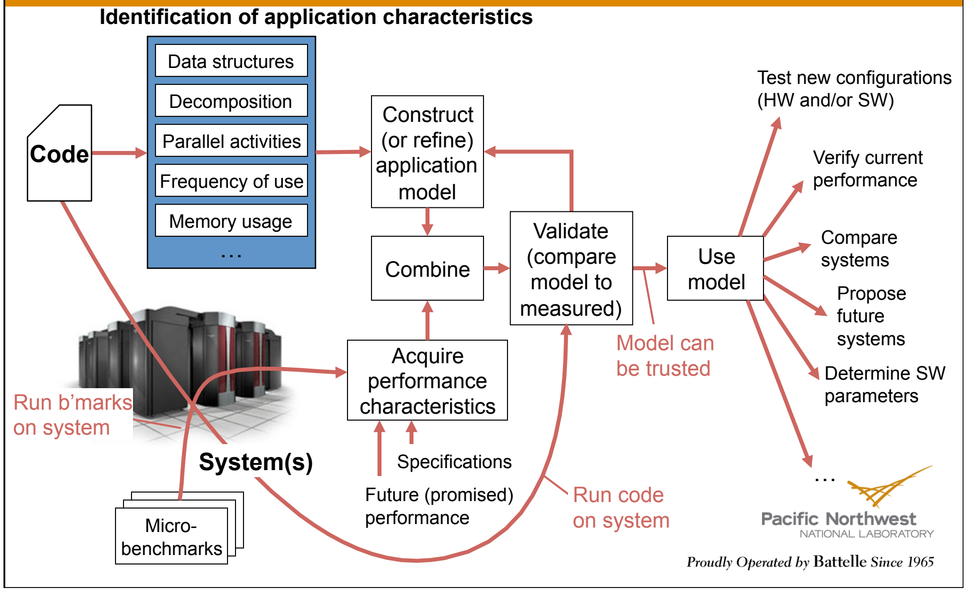


Not common

- Occasional global (collective) communication
  - Reduce (or all-reduce) to determine convergence criteria

Pacific Northwest
NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

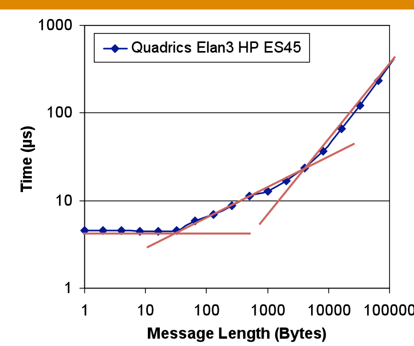
A Performance Modeling Process Flow



A Time-Based View of Communication

- Model may need to know time for an arbitrary-sized message
  - Avoid tables; generalize time with $T_{msg}(L) = t_0 + L / r_\infty$
  - *Caveat*: May be a function of L: $T_{msg}(L) = t_0(L) + L / r_\infty(L)$
- Useful approach:
  - Use a piecewise, linear fit for T_{msg}

$0 \le L \le 32$	$T_{msg}(L) \approx 5\ \mu s$
$64 \le L \le 1024$	$T_{msg}(L) \approx 5\ \mu s + 15L$ ns
$L > 1024$	$T_{msg}(L) \approx 10 \mu s + 3.4L$ ns



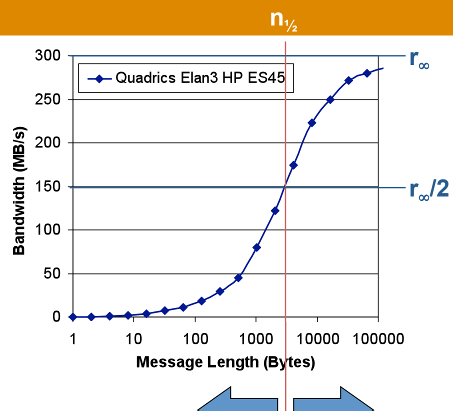
Note: log-log scale

- Often, a three-piece linear model suffices
  - t_0 dominates $t_B \times L$
  - t_0 and $t_B \times L$ are close
  - $t_B \times L$ dominates t_0

$T_{msg}(L) = t_0 + L / r_\infty$ (or $t_0 + t_B \times L$)

T_{msg}	is the time to send a message of length L,
t_0	is the start-up time (a.k.a. latency),
r_∞	is the asymptotic peak bandwidth, and
t_B	is the asymptotic time per byte ($1 / r_\infty$)

A Rate-Based View of Communication



- Time is dominated by t_0
- Reduce latency to most improve performance

- Time is dominated by $1/r_\infty$
- Increase bandwidth to most improve performance

- What message length yields half the peak data rate ($r_\infty/2$)?
  - We call this value $n_{½}$
- Solving $r_\infty/2 = t_0 + n_{½}/r_\infty$ for $n_{½}$ gives us $n_{½} = t_0 \cdot r_\infty$
- $n_{½}$ separates latency-bound communication from bandwidth-bound communication
- Implications in terms of the application's message sizes:
  - If $n_{½}$ is small, a higher-bandwidth network may improve performance
  - If $n_{½}$ is large, a lower-latency network may improve performance
  - If $n_{½}$ is large, message aggregation may improve performance
- A performance model can quantify each of the preceding performance improvements

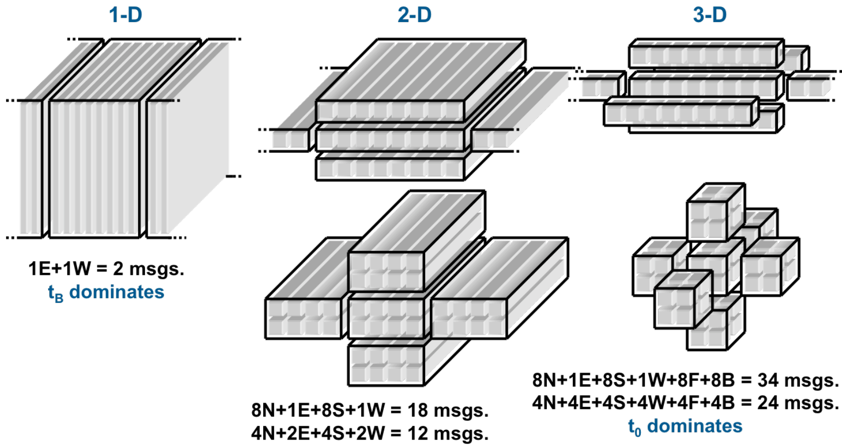
Other Factors Affecting Communication Performance

- Intrasocket vs. intranode vs. internode performance
  - Intrasocket usually slightly faster than intranode and much faster than internode
  - Internode communications may ∴ dominate performance
- Network channel sharing (NIC contention)
  - Processors within a node share external network connections
  - E.g., a node with 4 sockets, 4 cores/socket and a single 2GB/s NIC may deliver <128MB/s per processor if processors communicate simultaneously
- Network topology and routing
  - Messages routed through the network may collide
  - Increases effective T_{msg}
  - Collision frequency depends on application characteristics
- Uni- and bidirectional communication
  - Bidirectional comm. may take longer than equivalent unidirectional comm.
- Collectives
  - Some may be supported in hardware
  - Scaling properties vary by collective (and implementation)

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

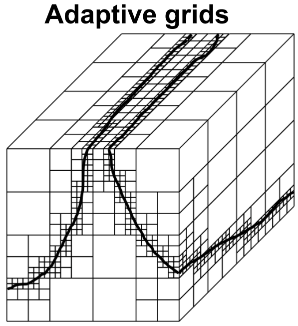
Application Modeling: Data Decomposition

- Mapping subgrids to processes affects the number of subgrid surfaces exposed between adjacent processes
- *Example*: Various decompositions of a 3-D grid, subgrid size = 8



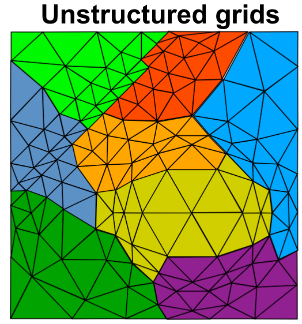
More Complex Data Structures

Adaptive grids



- Cells refined into 2×2×2 smaller cells
- Subgrids no longer regular

Unstructured grids

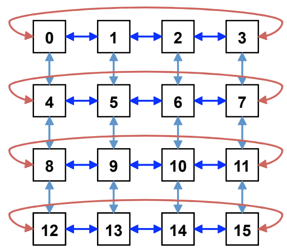


- Mesh composed of quads, tets, etc.
- Decomposed using partitioner, e.g., Metis

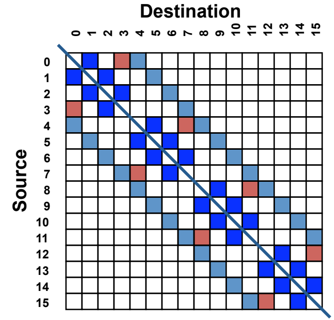
- Communication pattern is data set/partitioning dependent
- In our experience, these can be approximated by dense grids
  - e.g., with AMR surface increases by 2/3 power of volume
  - Irregular: approximate # of neighbors and communication volume

Identifying Communication Patterns

Logical communication
(2-D, cyclic in X, open in Y)



Pattern representation

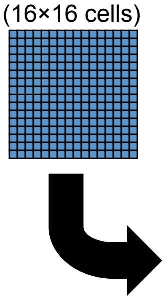


- Communication pattern may be an aggregate over an iteration
- *Tip*: Examine the pattern at each communication call point or, even better, each call stack

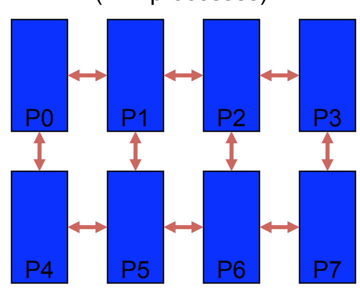
- Output from parallel-performance profiler
  - Symmetric iff equal data between processors in both directions
  - Major diagonal (±0) blank—processors do not send to themselves
  - 1st off diagonal (±1): normal communications in X
  - 3rd off diagonal (±3): wraparound in X (because P_x = 4 here)
  - 4th off diagonal (±4) communications in Y (again, because P_x = 4)

Application Modeling: Process Placement

Application's spatial grid (16×16 cells)

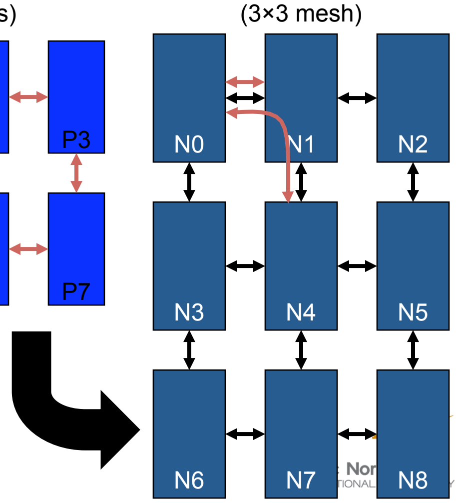


Application's logical process layout (4×2 processes)



(N.B.: subgrid size is 4×8 cells)

Physical topology of hardware (3×3 mesh)



Warning: It's easy to confuse the logical and physical layouts when developing a model. Be careful!

Proudly Operated by Battelle *Since 1965*

Frequency of Operations and Scaling

- Frequency and/or size of operations can depend on:
  - Scale (# of processors and/or data set size)
  - Dynamic characteristics (e.g., in some cycles a load balance occurs)
- Example: allreduce (for SAGE)

size words	1PE	2PE	4PE	8PE	16PE	32PE	64PE
1	358	547	555	546	562	558	582
2	0	400	0	0	0	0	0
3	10	10	10	10	10	10	10
4	0	0	431	0	0	0	0
8	0	0	0	393	0	0	0
14	10	10	10	10	10	10	10
16	0	0	0	0	455	0	0
32	0	0	0	0	0	437	0
64	0	0	0	0	0	0	530

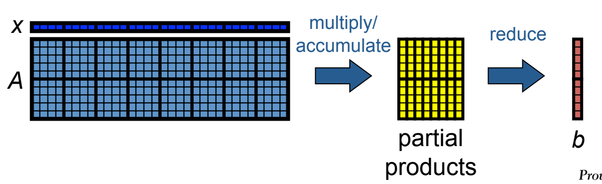
f(# AMR steps, # load balances)

Constant

Also, f(#PEs)

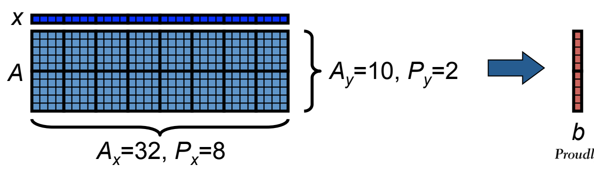
Putting It All Together: A (Very) Simple Performance Model

- "Application" to model: matrix-vector multiply
- Scatter columns of matrix *A* and elements of vector *x* from process 0 to all other processes
- All processes multiply/accumulate their fragments of *A* and *x* to produce a single vector per process (i.e., using a bunch of dot products)
- All processes collectively sum (i.e., reduce) the values in each row to produce vector *b* distributed across the first column of processes
- Process 0 gathers the fragments of *b* from the column 0 processes to produce a complete *b* vector



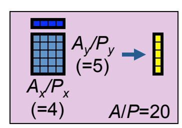
Putting It All Together: A (Very) Simple Performance Model (cont.)

- What parameters affect this application's performance?
  - A_x, A_y: # of columns and rows in A (→ # of elements in x and in b) (For convenience, let $A \equiv A_x \cdot A_y$ be the total # of elements in matrix A)
  - P_x, P_y: # of processes across and down (logical arrangement) (For convenience, let $P \equiv P_x \cdot P_y$ be the total # of processes)
  - T_{ma}: Time to perform a single multiply-accumulate
  - $T_{sc}(L,P)$: Time to scatter L doublewords to each of P processes
  - $T_{red}(L,P)$: Time to reduce L doublewords from each of P processes
  - $T_{ga}(L,P)$: Time to gather L doublewords from each of P processes



Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Modeling Matrix-Vector Multiplication



- Start with the "fundamental equation of modeling":
  - $T_{run} = T_{computation} + T_{communication} - T_{overlap}$
- Determine computation time
  - Each process is assigned A_x/P_x columns and A_y/P_y rows
  - All process work in parallel (SPMD-style), so the time for *each* process is also the time for *all* processes
  - # of multiply-accumulates per process = $(A_x/P_x)(A_y/P_y) = A/P$
  - Therefore, $T_{computation} = T_{ma} \cdot A/P$
- Determine communication time
  - Process 0 must scatter subgrids of A and x to all P-1 other processes
  - Scatter time = $T_{sc}(A/P, P-1) + T_{sc}(A_x/P_x, P-1)$
  - Each row of processes must reduce A_y/P_y values to process column 0
  - Reduction time = $T_{red}(A_y/P_y, P_x-1)$
  - Process 0 must gather a subvector of b from the processes in column 0
  - Gather time = $T_{ga}(A_y/P_y, P_y-1)$
  - Therefore, $T_{communication} = T_{sc}(A/P, P-1) + T_{sc}(A_x/P_x, P-1) + T_{red}(A_y/P_y, P_x-1) + T_{ga}(A_y/P_y, P_y-1)$
- Determine overlap of communication and computation: *none*

Sample Matrix-Vector Multiplication Model "What If?" Analyses

- What if we ran on a 1,000,000-CPU system?
  - Plug P=1,000,000 and suitable array sizes into the model
- What if the code were modified to use SIMD, vector, or fused multiply-add instructions?
  - Measure (or estimate) new T_{ma} and plug into model
- What if our network had hardware support for collectives?
  - Estimate new $T_{sc}(L,P)$ and $T_{ga}(L,P)$ and plug into model

Model variations:

- Model improvements
  - *Example*: Taking cache effects into consideration
  - Hardware changes (larger/smaller cache) or input parameters (fewer/more cells per subgrid) determine if subgrid fits in cache
  - T_{ma} must be made to depend on subgrid size: $T_{ma}(A/P)$
- Code changes
  - *Example*: Breaking up the scatter into pieces and interleaving these smaller scatters with computation
  - $T_{overlap}$ must represent communication/computation overlap

Summary of Our Approach to Performance Modeling

- Separation of:
  - Application factors (as identified from a functional point of view)
  - System factors (what it costs to perform certain functionality)
- Separation of:
  - Single-processor issues: normally measured or otherwise stated
  - Multiprocessor issues: scalability, parallel operations
- Application factors
  - Decomposition of global data structure into per-process units
  - Scaling behavior
  - Parallel activity (determined by looking at communication profiles, e.g., using a communication matrix)
  - Frequency of various operations (boundary exchanges, collectives, etc.)
- System factors
  - Typically measured using microbenchmarks or stated for future hypothetical machines

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Tutorial Outline (the plan!)

	Page	Duration
Introduction and motivation		**20 mins**
Performance metrics & pitfalls		**30 mins**
Performance modeling methodology		**40 mins**
COFFEE BREAK		**30 mins**
Abstractions		**30 mins**
Case Studies		
I: SWEEP3D		**60 mins**
LUNCH BREAK		**90 mins**
II: SAGE		**30 mins**
III: DNS3D		**30 mins**
Applications of modeling		
I: Rational system integration		**30 mins**
COOKIE BREAK		**30 mins**
II: Novel Architectures: Blue Waters		**40 mins**
III: Performance comparison of large-scale systems		**40 mins**
Conclusions, lessons learned, wrap-up		**10 mins**

Predictive Accuracy in the Presence of Simplifying Abstractions

Goals for performance modeling :

- 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?

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

Pacific Northwest NATIONAL LABORATORY

Proudly Operated by Battelle *Since 1965*

Things Aren't Always So Simple

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

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

How to Model Such Applications?

- We will look at two applications (Krak and HYCOM)
  - Computation and communication requirements
    - Vary across processors
    - 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

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

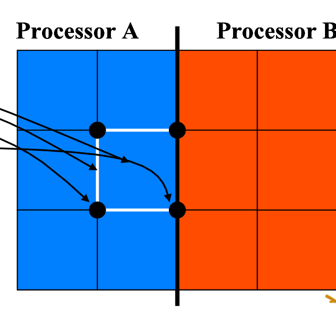
Better Approach: Abstraction

- 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
- 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

Pacific Northwest
NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

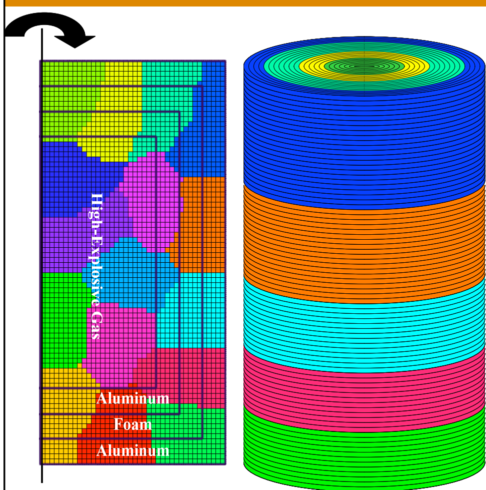
Case Study #1: Krak

- 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



Pacific Northwest
NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Krak Input Description: Irregular Subgrids and Multiple Materials



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

Pacific Northwest NATIONAL LABORATORY

Proudly Operated by Battelle *Since 1965*

Krak Performance Model

- Performance models separate application runtime into components:
  - 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
- 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
- Complexity can be managed with abstraction

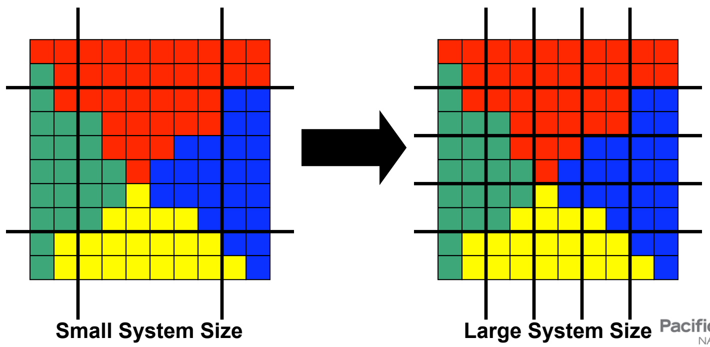
Pacific Northwest NATIONAL LABORATORY

Proudly Operated by Battelle *Since 1965*

Key Observations: Strong Scaling Behavior at Large Scale

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



Pacific Northwest NATIONAL LABORATORY

Proudly Operated by Battelle *Since 1965*

Abstractions Simplify Performance Model Components

Abstractions result in simplified performance model:

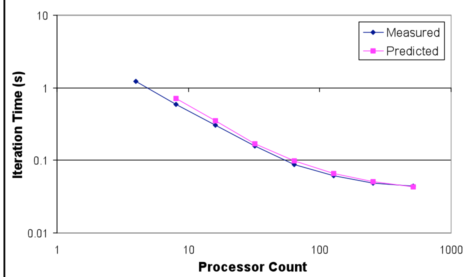
- Computation
  - Each subgrid contains the same number of cells
  - All cells are of the most computationally intensive material
  - All subgrids are square in shape
  - Per-cell cost derived from measuring compute times of subgrids of varying sizes
- Communication
  - Each subgrid is modeled with four 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?

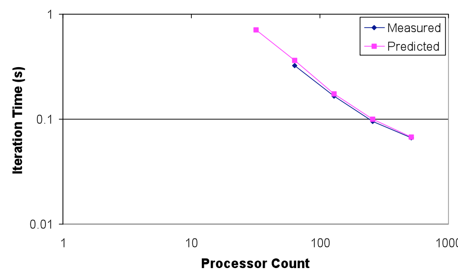
Pacific Northwest NATIONAL LABORATORY

Proudly Operated by Battelle *Since 1965*

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

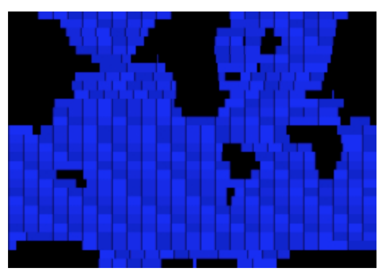
Pacific Northwest NATIONAL LABORATORY

Proudly Operated by Battelle *Since 1965*

Case Study #2: 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

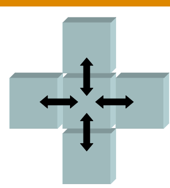
HYCOM Performance Model

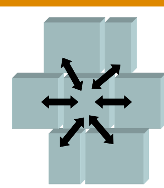
Again, performance model has two primary components:

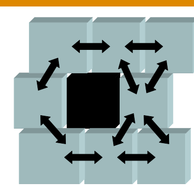
- Computation
  - Simple relative to Krak
    - » Computational cost dictated by largest subgrid
    - » Subgrid size is known in advance
  - Fractional subgrids incur idle time
- Communication
  - Interprocessor communication required to exchange boundary information between subgrids
  - Regular communication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

Where do we need abstraction?

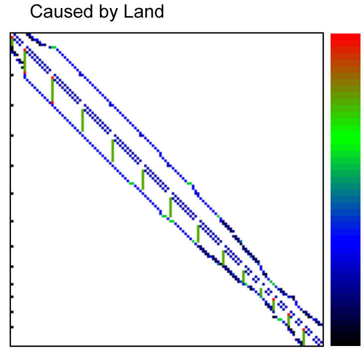
Modeling HYCOM Communication


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
  - Size of boundary faces often leads to large messages, even at large scale
- **"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?



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

Remember, goal is not to model each processor but to model the performance of the parallel system

Model Validation

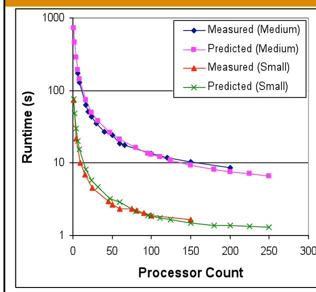
Input Decks:

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

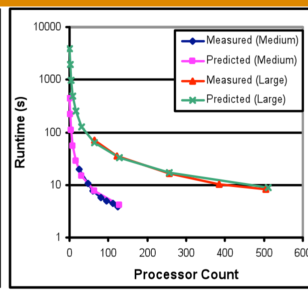
Machine Parameters:

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

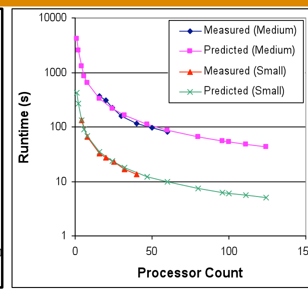
Model Validation



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

Conclusions

- Scientific applications are often not regularly partitioned
  - 3rd party partitioning software
  - Inconsistencies in global domain caused by inhomogeneous features
  - Irregular communication patterns
- 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

Pacific Northwest NATIONAL LABORATORY

Proudly Operated by Battelle *Since 1965*

Tutorial Outline (the plan!)

Case Studies

Three case studies chosen from many applications that have been modeled

1) Sweep3D
   - Deterministic S_N Transport
     - Structured mesh
     - 2-D data decomposition
     - Pipelined wavefront processing

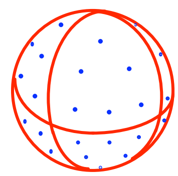
2) SAGE
   - Hydrodynamics code
     - Structured Adaptive mesh
     - 1-D data decomposition

3) DNS3D
   - Direct numerical turbulence simulation
     - Structured 3D mesh
     - 2-D data decomposition

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Case Study I: S_N Transport

- Solve the particle transport equation, where the density distribution of particles N(x̲, E, Ω, t) is the unknown
- Use discrete directions Ω
  - S_N has N*(N+2) total directions spread out in 3-dimensions
  - e.g., S_6 has 48 total directions, or 6 directions per octant

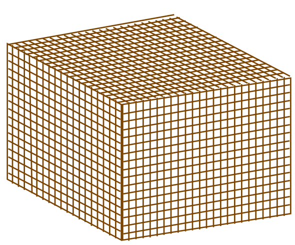


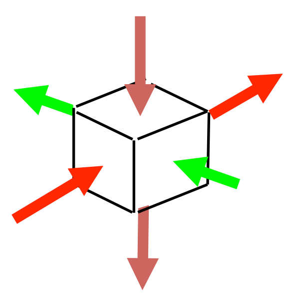
- SWEEP3D code: 1-group, Cartesian-grid kernel (http://www.c3.lanl.gov/par_arch/Software.html)

"Performance and Scalability Analysis of Teraflop-Scale Parallel Architectures Using Multidimensional Wavefront Applications", A. Hoisie, O. Lubeck, H. Wasserman, Int. J. of High Performance Computing Applications, Sage Science Press, 14(4), Winter 2000

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Cell Update

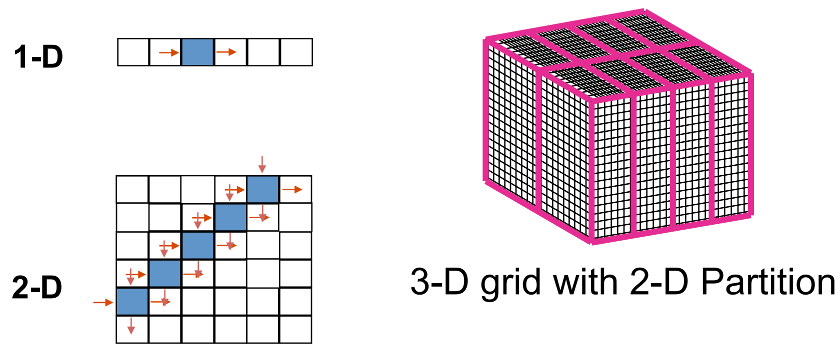




3 inflows & 3 outflows
cell balance equation(s)

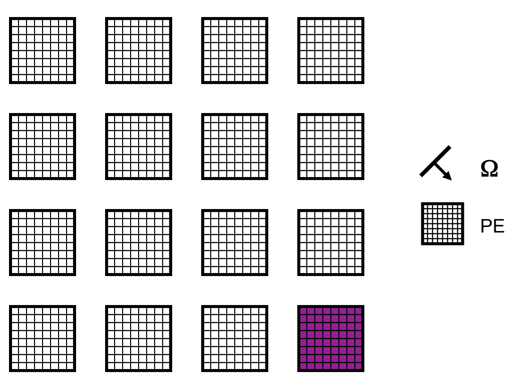
Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

S_N Wavefronts (Sweeps)

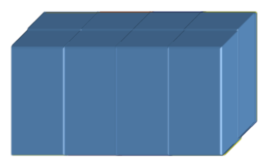


3-D Spatial Grid Using 2-D Decomposition

- 2-D decomposition results in a PE holding a several contiguous columns of data (diagram shows top view of 3-D spatial grid)
- Processor utilization is limited by the number of wavefronts (directions) from a corner point (quadrant)
  - for S_6 transport, only 6 wavefronts per octant (12 per quadrant)

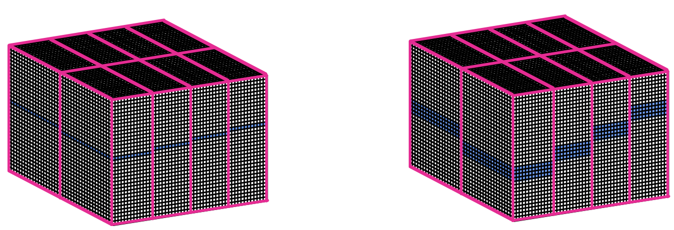


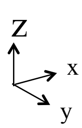
3D Wavefront with 2D Partition



3D Wavefront with 2D Partition

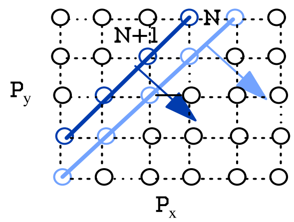
2D Domain decomposition with “blocking”

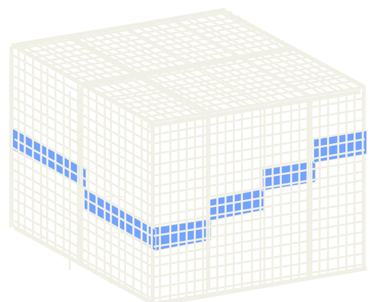




Blocking in “z” Leads to tradeoff: Parallel Efficiency vs. Communication Intensity

Wavefront Abstraction with Message Passing



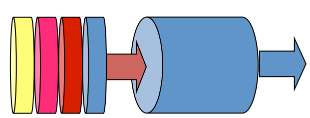


Pipelined wavefront abstraction:

```
for each octant
   for each angle-block
    for each z-block
       receive west
       receive north
            compute sub-grid
       send east
       send south
    end for
  end for
end for
```


Basic Pipeline Model

- N_{sweep} wavefronts "scan" the processor grid.
- Each scan requires
  N_s steps.
- There's a delay of d between scans.
- The total number of steps, S, for all wavefronts is

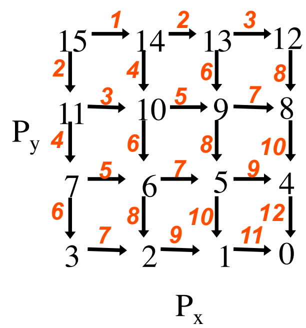


$$S = N_s + d(N_{sweep} - 1)$$

- The challenge is to find N_s and d.
- For S_N: N_{sweep} = zblocks * angleblocks * octants

Communication Pipeline

$$N_s^{comm} = 2(P_y - 1) + 2(P_x - 1)$$
$$d^{comm} = 4$$



Processor Nodes
Message Numbers

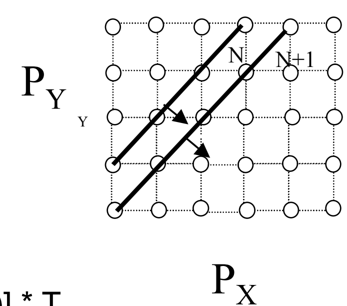
T_{comm} = [2(*Px* + *Py* - 2) + 4(N_{sweep} - 1)] * T_{msg}

$$T_{msg} = t_s + t_B L$$

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Computation Pipeline

$$N_s^{comp} = P_x + P_y - 1$$
$$d^{comp} = 1$$



T_{comp} = [(*Px* + *Py* - 1) + (N_{sweep} - 1)] * T_{cpu}

$$T_{cpu} = \left(\frac{N_x}{P_x} * \frac{N_y}{P_y} * \frac{N_z}{K_b} * \frac{N_a}{A_b}\right)\frac{N_{flops}}{R_{flops}}$$

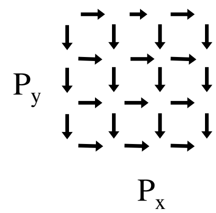
Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Alternative Modeling Approaches?

$(P_y-1)*P_x$ procs have South neighbors: all send
$(P_y-1)*P_x$ procs have North neighbors: all receive
$(P_x-1)*P_y$ procs have East neighbors: all send
$(P_x-1)*P_y$ procs have West neighbors: all receive

==

$N_{msg} = [(P_y-1)*P_x + (P_x-1)*P_y]$ pairs of send/receives



A) $T = N_{msg} * T_{msg} + (P_x * P_y) * T_{cpu}$

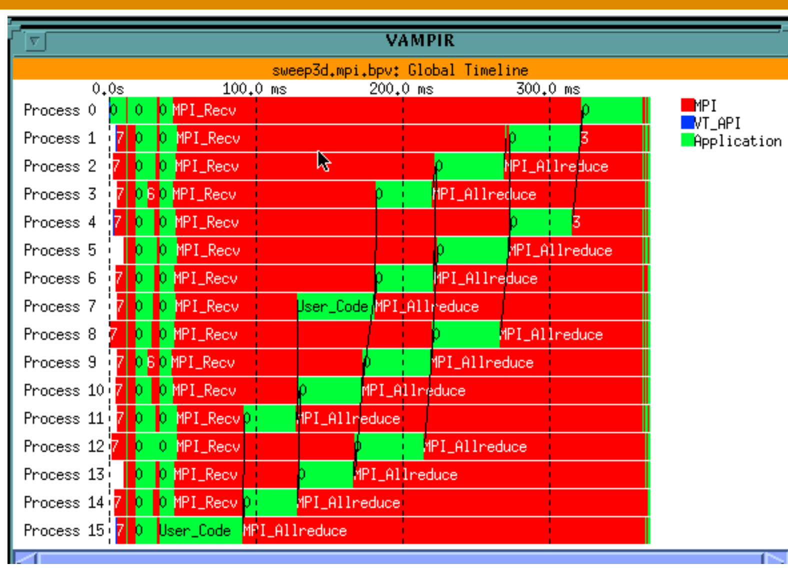
B) $T = P_x * P_y * 2 * T_{msg} + (P_x * P_y) * T_{cpu}$

Do you see any problem with any of these 2 alternative approaches?

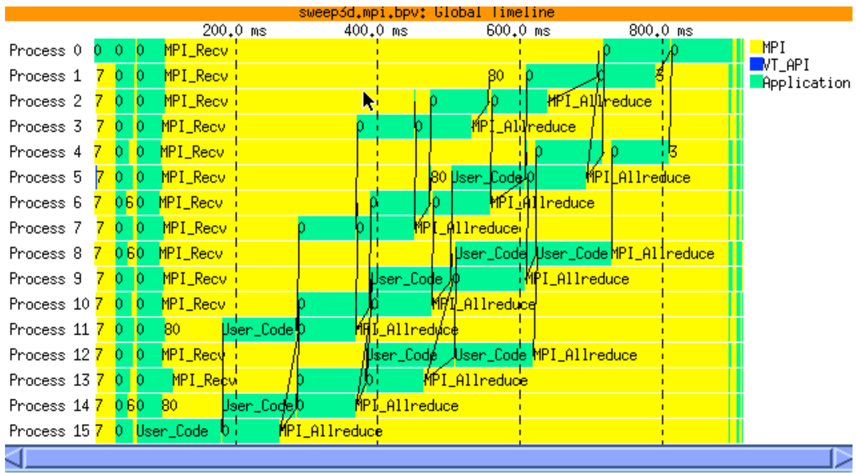
A) is a (wrong) upper bound. B) is a (wrong) lower bound. Both fail to accurately describe the overlap in communication and computation. Both fail to account for the delays due to the different repetition rates of the two types of wavefronts. Both are wrong…but don't feel bad if you almost agreed to one of them…we struggled with this for quite some time.

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

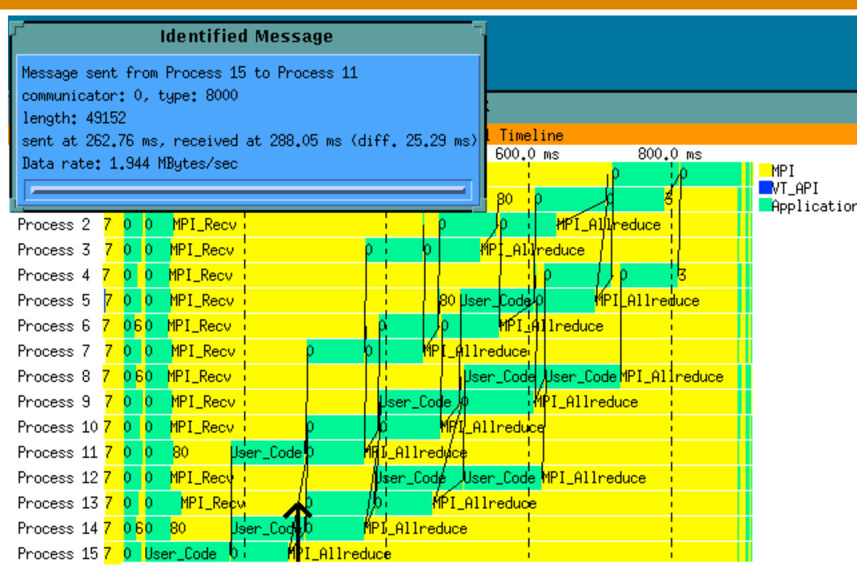
Trace Analysis with One Wavefront



Trace Analysis with Two Wavefronts

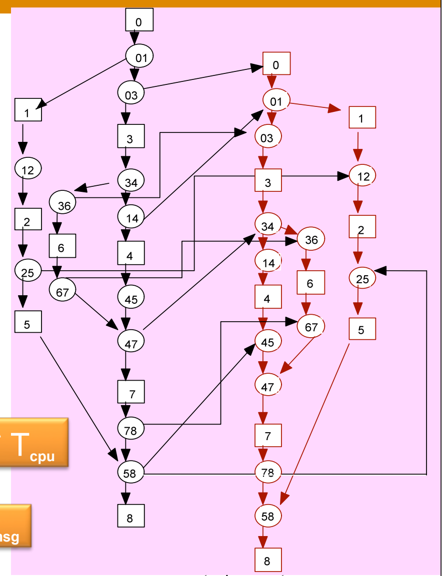


Trace Analysis with Two Wavefronts



Combining Pipelines

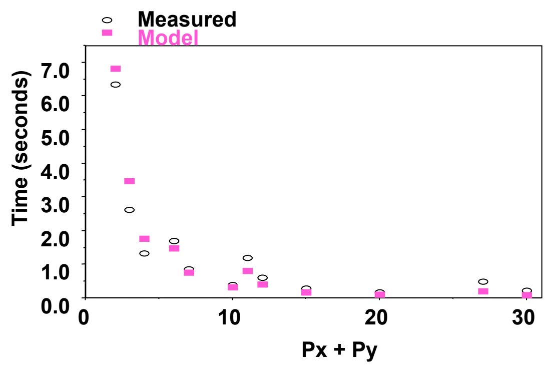
$T_{total} = T_{comp} + T_{comm}$?



$T_{comp} = [(Px + Py - 1) + (N_{sweep} - 1)] * T_{cpu}$

$T_{comm} = [2(Px + Py - 2) + 4(N_{sweep} - 1)] * T_{msg}$

Validation: Strong Scalability

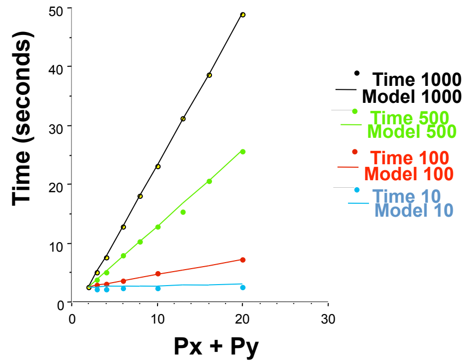


Pacific Northwest
NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Blocking Strategies

- Larger block sizes lead to increased computation / communication ratio.
- For wavefront algorithms smaller blocks yield higher parallel efficiency.

SWEEP for Several K Block Sizes



Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

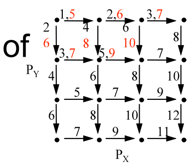
S_N Transport on Clusters of SMPs

- Goal: understand how decreased connectivity affects algorithmic performance.
  - **Obvious latency / BW effects, but is this the whole story?**
- Obvious relevance to many large-scale systems

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Summary So Far...

- SWEEP3D results assumed that a logical processor mesh can be imbedded into the machine topology such that
  - **each mesh node maps to a unique processor and**
  - **each mesh edge maps to a unique router link.**
- This is required to maintain the concurrency of communications within a wavefront.

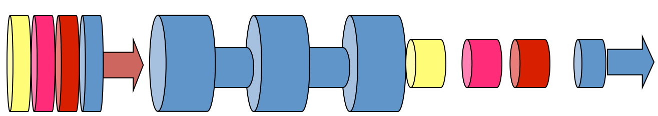


- We now examine cases with reduced connectivity.
- Q: What happens to d and N_{steps} ?

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Cluster of SMPs: "Pipeline with Bottlenecks" Model

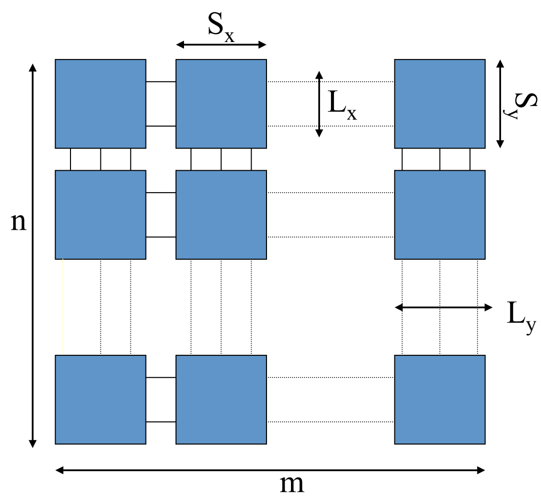
$$S = N_s(I) + d(I)F(N_{sweep}) \quad ?$$



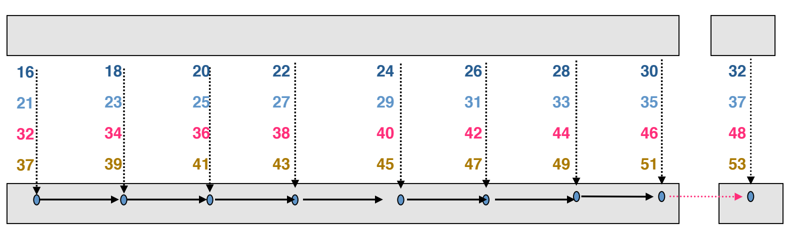
"A General Predictive Performance Model for Wavefront Algorithms on Clusters of SMPs", A. Hoisie, O. Lubeck, H. Wasserman, F. Petrini, and H. Alme, In Proc. of ICPP, Toronto, Canada, August 2000.

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

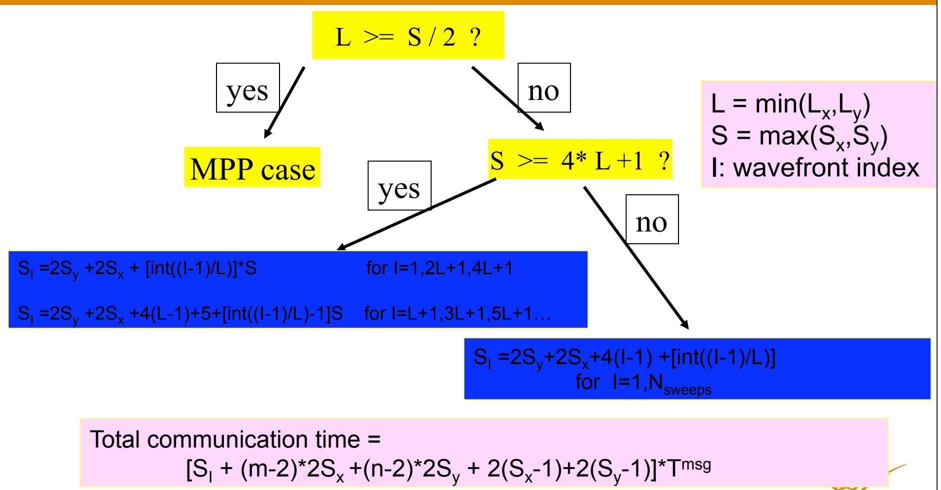
Cluster of SMPs: Notation



Sx = Sy = 8; Lx = Ly =1



Cluster Model



Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Extending to Multiple Octants

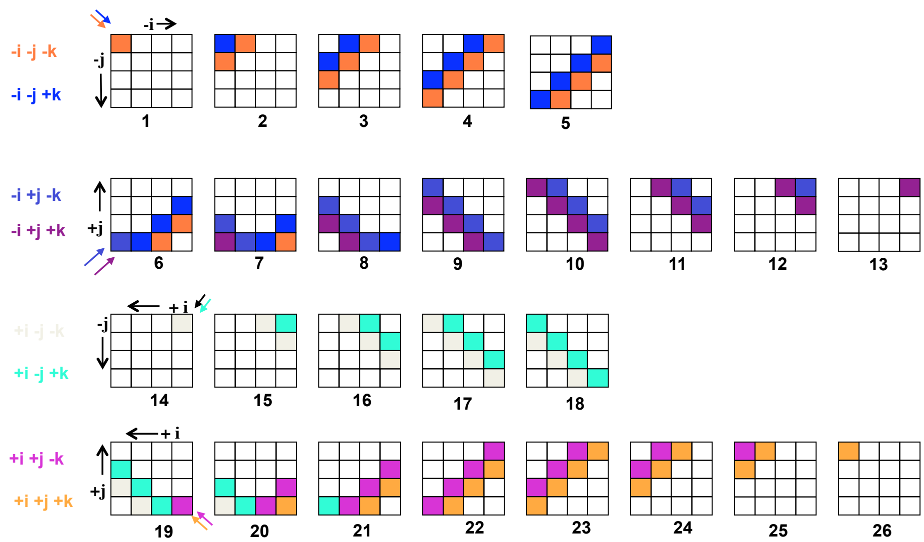
- Model so far represents sweeps generated by angle/k-block loops
- Application consists of multiple octants, multiple iterations
- Iteration dependence added as multiplicative term
- Multiple octants
  - extends the pipeline length
  - include dependences between octants.

Pipelined wavefront abstraction:

```
for each octant
   for each angle-block
    for each z-block
       receive east
       receive north
         compute subgrid
       send west
       send south
    end for
  end for
end for
```

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Multiple Octant Processing



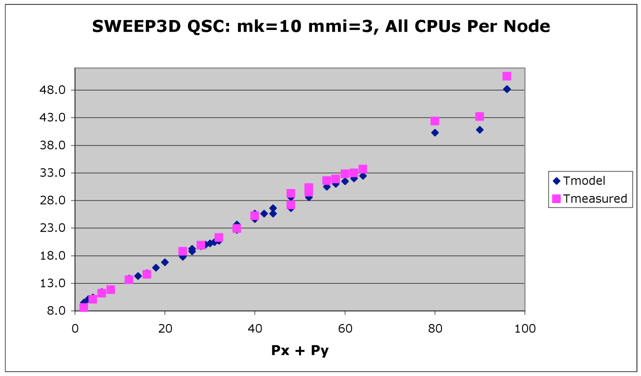
Multiple Octant Processing

Originating Octant for Sweep	Delay (to next Sweep)
-i -j -k	1
-i -j +k	P_y
-i +j -k	1
-i +j +k	P_x+P_y-1
+i -j -k	1
+i -j +k	P_y
+i +j -k	1
+i +j +k	P_x+P_y-1
Total steps	$2P_x+4P_y+2$

- Result: Pipeline length is 3 times longer than that of 1 octant for $P_x = P_y$ (but much less than 8 times longer).
- Result: The pipeline length is asymmetric with respect to the processor grid.

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Validation: Multiple Octant Processing



Compaq ES40 Cluster (4 Processors per SMP)
Model Parameters include:

T_{cell} = 120ns
T_s = 11µs
T_B = 3.4ns (for message size of 12000 bytes)

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Lessons from SWEEP3D Model

- Development of *application* microkernel benchmarks was important:
  - Create a version of the code with computation eliminated
  - Create a version of the code with communication eliminated
- Work from the inside to the outside of the loop nest
- Model communication/computation/overlap
- Validate
- Re-iterate as new factors come into play

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Tutorial Outline (the plan!)

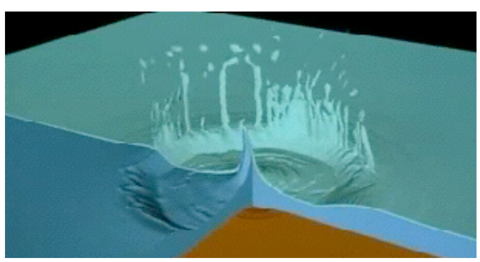
Pacific Northwest
NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Case Study II: Hydrodynamics

- SAGE – SAIC's Adaptive Grid Eulerian hydrocode
- Hydrodynamics code with Adaptive Mesh Refinement (AMR)
- Applied to: water shock, energy coupling, hydro instability problems, etc.
- Represents a large class of production ASCI applications at Los Alamos
- Routinely run on 1,000s of processors
- Scaling characteristic: Weak
- Data Decomposition (Default): 1-D (of a 3-D AMR spatial grid)

"Predictive Performance and Scalability Modeling of a Large-Scale Application", D.J. Kerbyson, H.J. Alme, A. Hoisie, F. Petrini, H.J. Wasserman, M. Gittings, in Proc. SC, Denver, 2001

SAGE Uses: Example Meteor Impact on Water



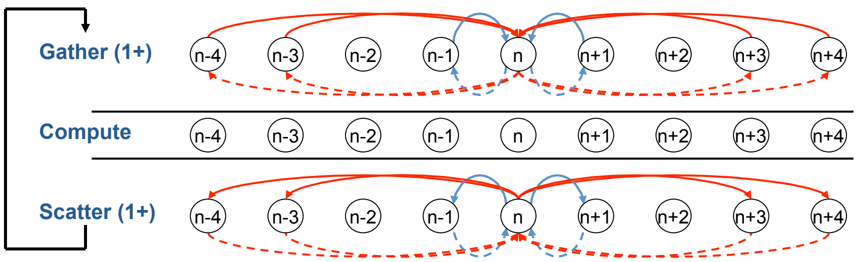
One-kilometer iron asteroid struck with an impact equal to about 1.5 trillion tons of TNT, and produced a jet of water more than 12 miles high

Wave velocities for the largest asteroid will be roughly 380 miles an hour. Initial tsunami waves are more than half a mile high, abating to about two-thirds of that height 40 miles in all directions from the point of impact.

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Processing Flow in SAGE

- SAGE consists of many repeated 'stages' per cycle:

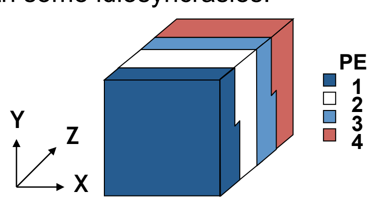


  - Gather: obtain boundary data from remote PEs
  - Compute: computation specific to a 'stage'
    (computations for all stages are considered together in a single PE timing)
  - Scatter: update boundary data on remote PEs
- Also, several collectives occur during each cycle (Allreduce)

SAGE Data Decomposition (1-D Slab)

- Decomposition determines boundary sizes between sub-grids
  - Amount of traffic for gather/scatter communications
- SAGE uses 1-D 'slab' decomposition, with some idiosyncrasies:
- First *B* blocks of 2x2x2 cells assigned to PE1 …

  (*E* = numcells_PE = *B**8)



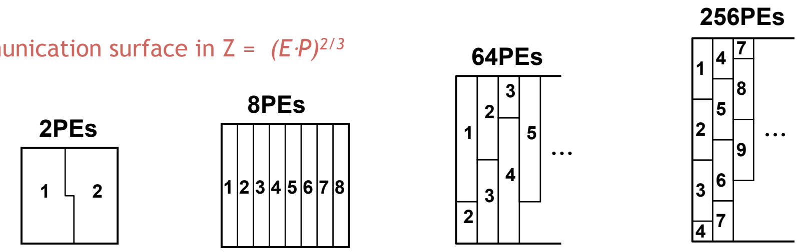
- Total grid volume ~ E · P (Weak-scaling)
  - Volume is constrained by the side of the spatial cube being even
- Boundary exchanges occur in all three dimensions
  - in Z: largest boundary exchange depends on size of spatial cube!
  - in Y: depends on side of spatial cube
  - in X: constant at 4 elements
- N.B. the communication costs increase with scale

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

SAGE Data Decomposition (1-D Slab)

- The spatial grid is a cube by default
- Due to weak scaling, the size of the spatial grid grows with the no. of PEs
- Hence, the communication surface in Z also grows (up to a point)

Communication surface in Z = $(E \cdot P)^{2/3}$



- At a certain scaling point, a single foil of cells is held on more than one processor which limits the communication traffic
- However, distance between processors increases!

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

A Bit of Algebra: Scaling Analysis

- The total volume is: $V = E \cdot P = L^3$
- The volume of each sub-grid is: $E = l \cdot L^2$

  where P is the number of PEs, l is the short side of the slab (in the Z dimension) and L is the side of the slab in X and Y directions (assuming a cubic grid)
- The surface of the slab, L^2, in the X-Y plane is: $L^2 = V^{2/3} = (E \cdot P)^{2/3}$
  i.e. communication grows with the number of processors!
- Partitioning in 1-D results in $L/(2P)$ 'foils' of width 2 on each PE:

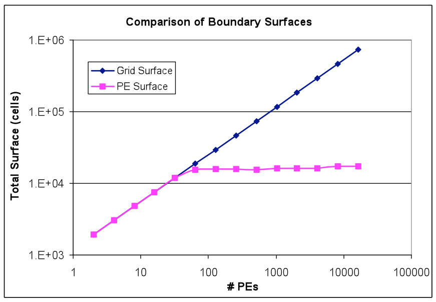
$$(E \cdot P)^{1/3}/2P \;=\; (E/8P^2)^{1/3}$$

- When this has a value less than one, a processor will contain less than a single foil, i.e. when $P > SQRT(E/8)$ the number of processors involved in boundary exchange increases!
- Also, there is a maximum distance between the processors that hold a foil, termed the "PE Distance" (*PED*)

Scaling Characteristics

i) Surface size in Z

Represents the size of boundary transfers between processors

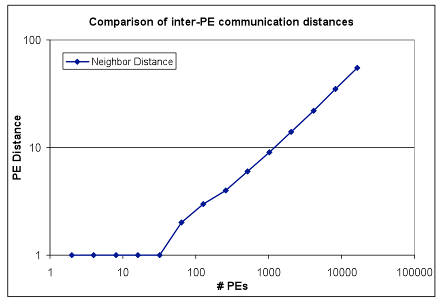


Surface split across PEs: $P > \sqrt{(E/8)}$

(e.g., for E = 13,500 $P > 41$**)**

ii) PE Distance

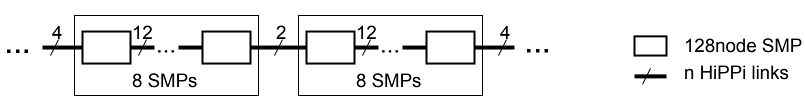
Minimum logical distance between processors for boundary transfers



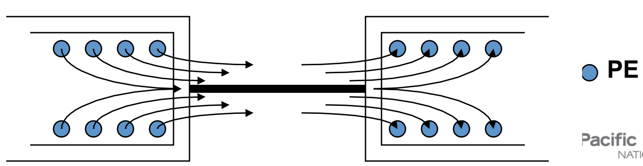
PE distance = $\lceil (8P^2/E)^{1/3} \rceil$

Effect of Network Topology

- PE distance determines the no. of out-of-node communications that take place on single a gather-scatter
- The max. no. communications equals the no. PEs in a node
- For example, on ASCI Blue Mountain:



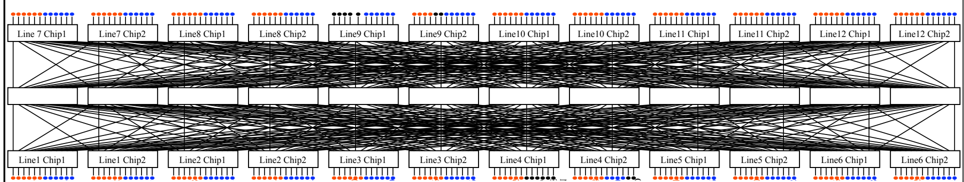
- PE distance results in many PEs communicating across a small number of links:



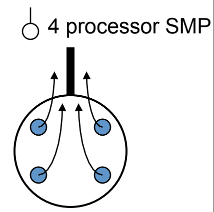
Pacific Northwest
NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Effect of Network Topology (cont.)

- On a fat-tree network, the PE distance effect is smaller:
  - smaller nodes (typically 4 processors per node)
  - Also fat tree network enables communication between nodes with approximately equal performance



- PE distance has maximum effect when all cores communicate out of SMP node.
- The important aspects for the model are:
  - Number of processors/cores per node
  - Number of communication channels per node



Performance Model for SAGE

- Encapsulates code characteristics
- Parameterized in terms of:
  - Code (e.g., cells per PE), Mapping,
  - System (CPU speed, communication latency & bandwidth, memory etc.)

Gather & Scatter Communications — In-Box SMP Memory Contention

$$T_{cycle}(P,E) = T_{comp}(E) + T_{GScomm}(P,E) + T_{allreduce}(P) + T_{mem}(P,E)$$

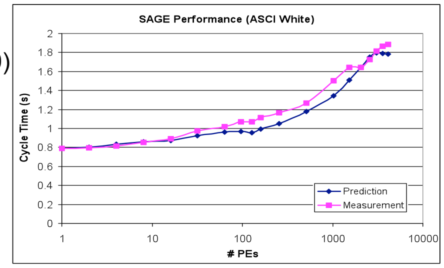
Computation — Allreduce Comms

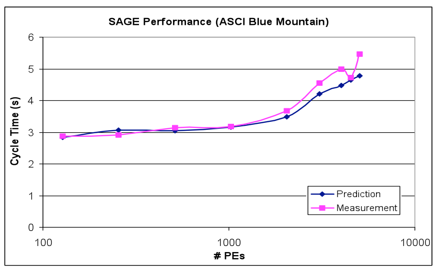
Application	E	Cells per PE
Mapping	Surfaces in X, Y, Z	Size of boundaries in X, Y & Z
System	$P^{\dagger}$, $P_{SMP}{}^{\dagger}$	#PEs, & #PEs per SMP box
	$CL^{\dagger}$	Communication Links per SMP
	$L_c{}^{\times}$, $B_c{}^{\times}$	Latency and Bandwidth
	$T_{comp}(E)^{\times}$	Time to process E cells
	$T_{mem}(P)^{\times}$	Memory contention per cell on P PEs.

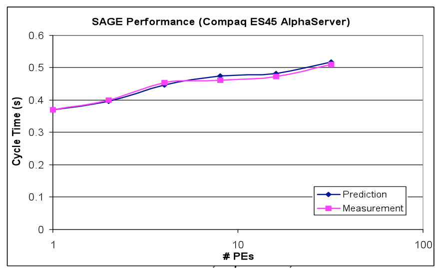
† System specification
× Measured / Benchmarked

Initial Validation

- Validated on large-scale platforms:
  - ASCI Blue Mountain (SGI Origin 2000)
  - CRAY T3E
  - ASCI Red (intel)
  - ASCI White (IBM SP3)
  - Compaq Alphaserver SMP clusters







Validation Summary

System	Number of Configurations tested	Maximum Processors tested	Maximum error (%)	Average error (%)
ASCI Blue (SGI O2K)	13	5040	12.6	4.4
ASCI Red (Intel Tflops)	13	3072	10.5	5.4
ASCI White (IBM SP3)	19	4096	11.1	5.1
ASCI Q (HP AlphaServer ES45)	24	3716	9.8	3.4
TC2K (HP AlphaServer ES40)	10	464	11.6	4.7
T3E (Cray)	17	1450	11.9	4.1
Roadrunner (Opteron & Cell)	15	6120	6.0	3.8
Dawn (Blue Gene/P)	17	144K	9.8	4.8
Lobo (AMD Barcelona)	15	4352	6.8	3.9

- Model is highly accurate (typically error < 10%)

Pacific Northwest NATIONAL LABORATORY

Proudly Operated by Battelle *Since 1965*

Lessons from SAGE Model

- Thorough understanding of data-decomposition leads to explanation of scaling effects
- Repetition of primary operations:
  - Boundary gather/scatters
- Computation encapsulated into a single processor time even though computation in stages varies
- Dependence on the node size – leading to contention in inter-node communications
- Model has not changed since development even though code is under active development
  - Even though frequency of operations and single processor time has changed, and
  - Also represents several derivatives of the code

Tutorial Outline (the plan!)

Direct Numerical Simulation (DNS3D)

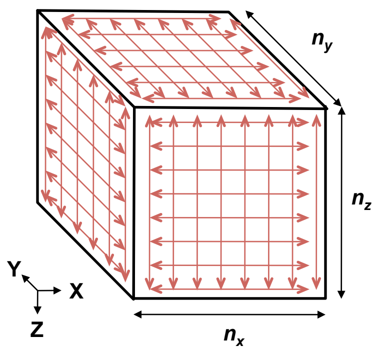
- NSF petascale application for Blue Waters
  - Simulation of homogeneous turbulence in 3D
- Performance predictions were a part of NSF proposal
- Petascale problem set-up:
  - 12,288 x 12,288 x 12,288 grid
  - Requires 10,000 iterations
  - Target runtime is 40 hours on full Blue Waters system
- Implementation chosen was DNS3D
  - Iterative:
    - » Each time-step uses a 4-stage Runge-Kutta (RK) stepping scheme
  - Use of inbuilt FFT routines
    - » Other libraries possible (e.g. FFTW)

DNS3D processing flow

```
For each time-step
   For each RK stage 1..4
      Inverse 3D FFT (3 variables)
      Spectral computation
      Inverse 3D FFT (3 variables)
      Real-space computation
      3D FFT (3 variables)
      Spectral computation
      RK time stepping
```

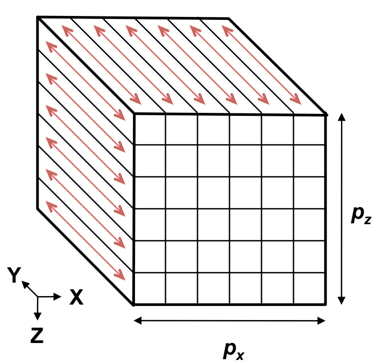
- An RK stage consists of 3x 3D-FFTs and 6x inverse 3D-FFTs
- Total of 12 + 24 3D-FFTs per time-step
- For modeling purposes we are not concerned with the precise ordering of operations but rather combine computation activities together, and communication activities together (assuming no communication/computation overlap)

General principle of a 3D-FFT: Done as sequence of 1D-FFTs



- Unit of computation is a 1D-FFT
- Three steps:
  - 1D-FFTs across *X*
  - 1D-FFTs across *Y*
  - 1D-FFTs across *Z*
- Assuming $n_x \times n_y \times n_z$ grid points there are:
  - $n_y \times n_z$ 1D-FFTs of size n_x
  - $n_x \times n_z$ 1D-FFTs of size n_y
  - $n_x \times n_y$ 1D-FFTs of size n_z

Parallel Decomposition done in 2D to reduce communication



- $p_x \times p_z$ processors
- n_x / p_x by n_z / p_z "pencils" per processor
- All 1D-FFTs along "pencils" are local to a processor (no communications)
  - need transpose between 1D-FFTs to ensure pencil locality
- Three steps:
  - 1D-FFT
    - » transpose
  - 1D-FFT
    - » transpose
  - 1D-FFT

Outline of DNS3D Model

$$T_{iteration} = T_{comp} + T_{transpose} + T_{collective}$$

where

T_{comp} – sequential time to process bundle of pencils

$T_{transpose}$ – time for 32 $Y \leftrightarrow Z$ & 36 $Y \leftrightarrow X$ and transposes

$T_{collective}$ – time for a single collective per iteration (small and ignored later)

Note that there is an additional I/O component for dumping data to disk. In the default setup this occurs every 200 iterations.

Pacific Northwest NATIONAL LABORATORY

Proudly Operated by Battelle Since 1965

Computation time

- Computation time split into two parts
  - The 1D-FFTs (non linear with #grid_points, n_{gp})
  - Spectral, real-space, & RK computation (linear with the #grid_points)

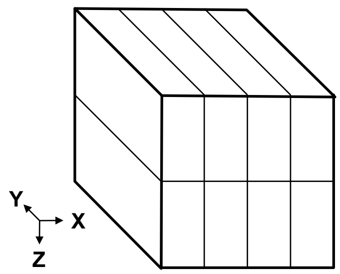
$$T_{comp} = 4 \times N_{pencils} \times (n_y \times T_{RK}(n_y) + 9 \times \begin{pmatrix} T_{1D\_FFT}(n_x) \\ T_{1D\_FFT}(n_y) \\ T_{1D\_FFT}(n_z) \end{pmatrix})$$

where $N_{pencils} = (n_x / p_x) \times (n_z / p_z)$

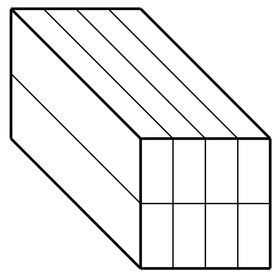
- Note forward and inverse FFTs assumed similar performance
- Parameters $T_{RK}(x)$, and $T_{1D\_FFT}(x)$ can be either measured on the target platform at small scale or by obtained by simulation

Measuring $T_{1D\_FFT}(s)$

- Example on an 8-processor node (or 8-core processor)
- Need to be careful when measuring $T_{1D\_FFT}(s)$
  1. if global problem $n_x = n_y = n_z = s$ fits into a nodes memory then measure
  2. otherwise need to have a reduced number of pencils: $n_x` \times n_y \times n_z`$ where $n_y = s$ (FFT size of interest)



- $n_x = n_y = n_z = s$
- **All 1D-FFTs equal size**



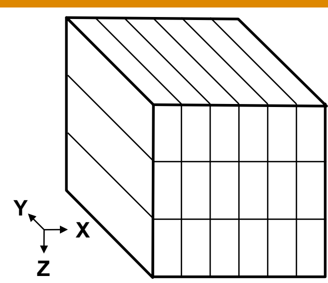
- $n_x` \times n_y \times n_z`$ $(n_y = s)$
- **1D-FFTs not equal size**
- **Need to isolate FFTs**

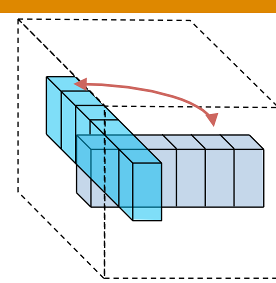
Pacific Northwest NATIONAL LABORATORY

Proudly Operated by Battelle Since 1965

Transpose time

Example: X <-> Y transpose





- Transpose time consists of two components

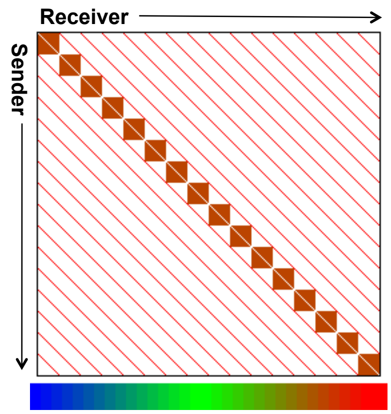
$$T_{transpose} = T_{local} + T_{remote}$$

- where T_{local} is for local copies and T_{remote} is for communication costs (and buffer reads / writes)

Volume	Y ↔ X	Y ↔ Z
Local	$\frac{1}{p_x} \cdot \frac{V}{p}$	$\frac{1}{p_z} \cdot \frac{V}{p}$
Remote	$\frac{(p_x-1)}{p_x} \cdot \frac{V}{p}$	$\frac{(p_z-1)}{p_z} \cdot \frac{V}{p}$

Proudly Operated by Battelle Since 1965

Communication matrix: Example measured from a 16x16 processor run



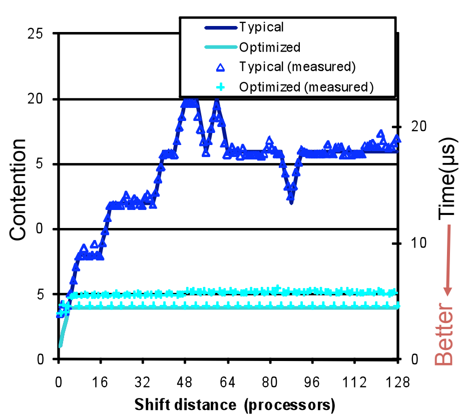
- Symmetric about major diagonal
  - Equal communication sent / received
- Matrix for full iteration
  - Y ↔ Z transposes are the "boxes" along the major diagonal
  - Y ↔ X transposes are the other diagonals
  - Note: MPI task allocation done in Z then X dimensions
- Diagonals represent a logical "shift"
  - $P_i \rightarrow P_{i+d}$
- Communication pattern can also indicate approach for good mapping
  - Want to minimize inter- vs. intra-node communications

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Contention in Network Impacts on Communication Performance

Two main sources:

1. Number of cores sharing a NIC
2. Contention in network when messages collide (share a channel)



- Shift: $P_i \rightarrow P_{i+d}$
  - where d = 1..128
- Infiniband Cluster
  - node = 4-cores
- Typical: contention generally increases with shift distance
- Optimized: max of 4 (bottleneck is node-size, PEs)

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Modeling Transpose Time

$$T_{transpose} = T_{local} + T_{remote}$$

$$T_{local} = (V / P).\ T_{transpose}(n_x, n_y, n_z)$$
$$T_{remote} = 32.T_{shift}(Y \leftrightarrow X) + 36.T_{shift}(Y \leftrightarrow Z)$$

- T_{local} is measured on a small (single-chip) run
  - Assumed to be the time for the transpose excluding intra-chip comms
- $T_{shift}()$ = average of the shift communication times
  - For Y↔X $d = 1..p_x\text{-}1$
  - For Y↔Z $d = p_x .. p_x{}^*p_z$ *step* p_x
- Can use modeled contention factors if measurements not possible
  - For Infiniband can be optimized
  - Some observations for BlueWaters later …

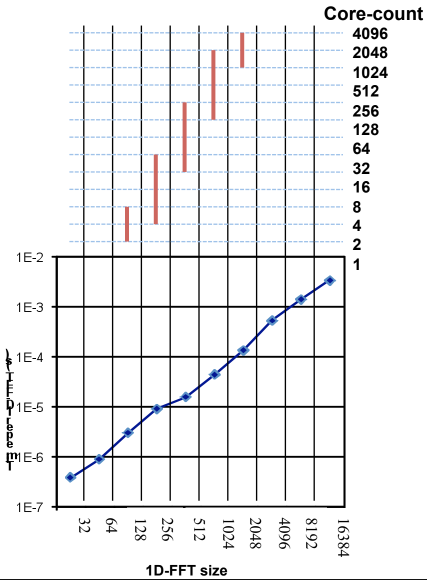
Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Validation to 512 Cores Showed High Accuracy

- Testbed: 256 node cluster
  - dual-socket dual-core Opteron,
  - 4x SDR Infiniband
- Problem setup:
  - Weak scaling: $V = 128{}^*128{}^*128$ on one processor-core
  - Power of two core counts
    - » round-robin scaling of p_x, p_z
    - » round-robin scaling of n_x, n_y, n_z
- Sub-grid shape varies with scale
  - Gets longer and narrower
    - » $(n_x / p_x) \times n_y \times (n_z / p_z)$
  - 1D-FFT sizes increase with scale

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

1D-FFT sizes vary with scale

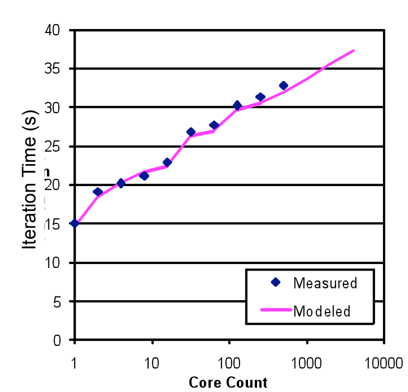


- Time for 1-D FFTs measured
- On testbed:
  - Increase with size
  - Some 2nd-order effects also
- FFT sizes used increases at distinct scale for our problem setup:
  - $(n_x / p_x) \times n_y \times (n_z / p_z)$
  - round-robin p_x, p_z
  - round-robin n_x, n_y, n_z
- Leads to interesting time curve with FFT size

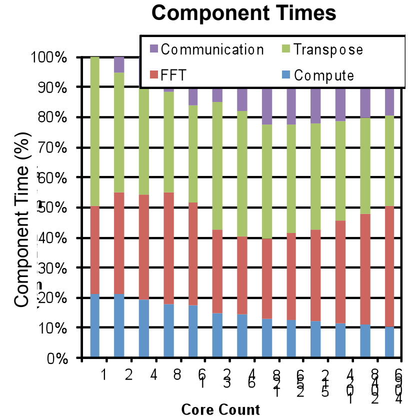
Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Validation shows high accuracy

Measurement vs. modeled iteration time



Component Times

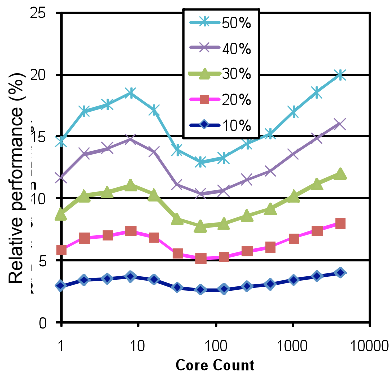


- High accuracy observed
  - Errors: 3.9% (max), 2.2% (avg)
- Component times shows increase in FFT times with scale

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Performance Exploration Using the Model: 1) Use of other FFT libraries

- Consider impact on change in T_{1D_FFT} from our baseline testbed
  - Not specific to a particular FFT implementation
  - But rather used as a guide to see if such a change is worthwhile

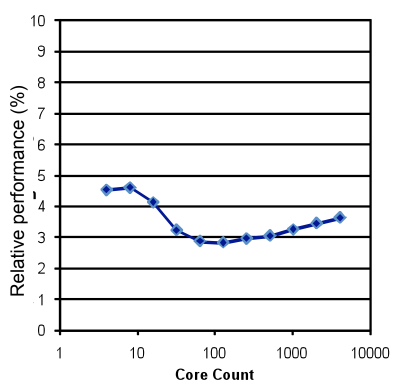


- 1D-FFT times assumed to be faster by between 10-50%
- Graph shows improvement in iteration time compared with baseline
- Form of curves reflects the measured 1D-FFT times for the current implementation

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Performance Exploration Using Model: 2) Optimized communications

- Possibility of overlapping *some* communication with computation
  - During last stage of FFT (remember log(N) stages), resultant data could start be communicated as part of the transpose
  - Requires optimization of the implementation
  - Is it valuable to undertake such optimizations ?



- **Assumptions:**
  - Each stage of a 1D-FFT takes constant time
  - Communication can be 100% overlapped during last stage of FFT
- **Performance improvement relative to testbed baseline**

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

DNS3D summary

- Non-linear effects pose interesting modeling factors
  - Size of 1D-FFTs increase with scale
  - Number of FFTs per core decreases
  - Cannot measure compute-cost on a sub-grid at small-scale and add in communication costs for large-scale
- Significant communication
  - Two types of transpose
  - Nearly all data for FFT is communicated to neighbors (most non-local)
- Modeling shows high accuracy
- Model currently in use to examine options for Blue Waters
  - We will also use it as a part of the performance acceptance testing

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Tutorial Outline (the plan!)

	Page	Duration
Introduction and motivation		**20 mins**
Performance metrics & pitfalls		**30 mins**
Performance modeling methodology		**40 mins**
COFFEE BREAK		**30 mins**
Abstractions		**30 mins**
Case Studies		
I: SWEEP3D		**60 mins**
LUNCH BREAK		**90 mins**
II: SAGE		**30 mins**
III: DNS3D		**30 mins**
Applications of modeling		
I: Rational system integration		**30 mins**
COOKIE BREAK		**30 mins**
II: Novel Architectures: Blue Waters		**40 mins**
III: Performance comparison of large-scale systems		**40 mins**
Conclusions, lessons learned, wrap-up		**10 mins**

Applications of Modeling

More than any other time in history, mankind faces a cross-roads. One path leads to despair and utter hopelessness. The other, to total extinction. Let us pray we have the wisdom to choose correctly.

- *Woody Allen*

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Rational System Integration

- When introduced, ASCI Q was the largest production ASCI system:
  - 20Tflops peak performance
  - 2048 HP AlphaServer ES45 nodes
  - 8192 Alpha EV68 processors, operating at 1.25GHz (2-fp per cycle)
- HP/Compaq was announced as supplier of ASCI Q in August 2000
- Majority of nodes were in production by end of 2002

Question (circa 2001) :

What level of performance will ASCI Q achieve?

Answer:

Use performance modeling!

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

ASCI Q at Los Alamos

ASCI Q Performance Data: History

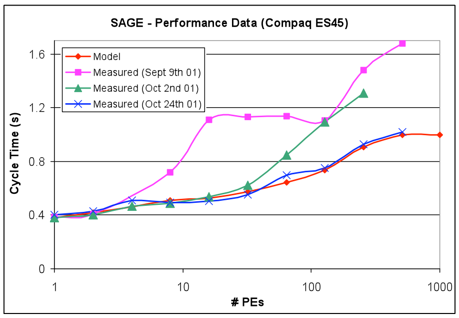
- Measured ASCI Q performance from the first nodes manufactured to the full sized machine
  - Installed in stages
  - 2 upgrades during installation: PCI bus (33MHz to 66MHz), and Processor (1.0GHz to 1.25GHz with increased L2 cache).

Date	# Nodes	Comments
March '01	8	First ES45 cluster available (HP Marlborough)
9th Sept '01	128	First machine at LANL, 33MHz PCI bus
24th Sept '01	128	Some faulty H/W replaced
24th Oct '01	128	O/S patch improved Quadrics Performance
4th Jan '02	512	PCI bus @ 66MHz (but not on all nodes)
2nd Feb '02	512	All @ 66MHz PCI, some nodes configured out
20th April '02	512	All nodes available and running
13th June '02	2	First 1.25GHz nodes (HP Marlborough)
20th Sept '02	1024	QA testing (1.25GHz processors)
25th Nov '02	1024	QB Performance variability testing
25th Jan '03	1024	QB Performance optimization
1st May '03	2048	QA+QB combined testing (20Tflop peak)

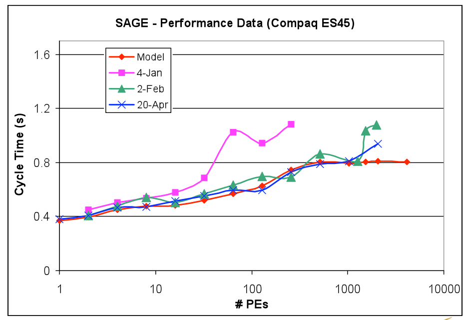
Performance Expectations Provided by Models

- Predictions made in April '01
- Further predications made for PCI upgrade, and CPU upgrade

Late 2001:



Early 2002: upgraded PCI



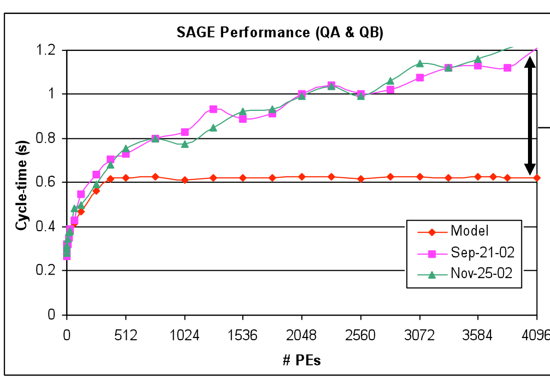
→ Model used to validate measurements!

Pacific Northwest NATIONAL LABORATORY

Proudly Operated by Battelle *Since 1965*

1024-Node Performance (Late 2002)

- Performance consistent across both phases of ASCI Q (each with 1024 nodes)
- Measurements were ~80% longer than model



There is a difference
WHY ?

Pacific Northwest NATIONAL LABORATORY

Proudly Operated by Battelle *Since 1965*

Model Includes Known Factors

- Model includes:
  - Computation characteristics of application
  - Communication requirements
  - Scaling characteristics
- Model approach is iterative: as new (understood) factors come into play they must be incorporated
- Without a model then it would not be possible to identify if there is a problem or not!
- If there are some unknown factors then we need to:
  - Identify
  - Understand
  - Model
  - And possibly optimize the application/system

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Quotation

"[W]hen you have eliminated the impossible, whatever remains, however improbable, must be the truth."

Sir Arthur Conan Doyle

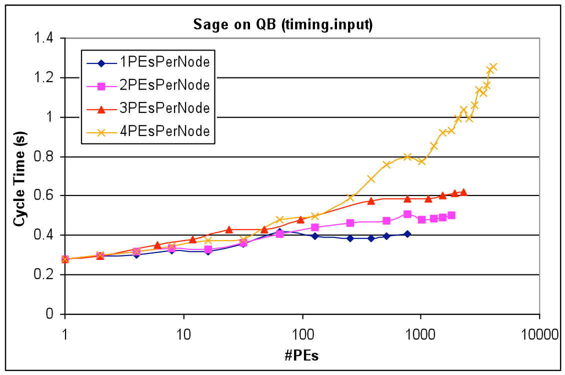
Sherlock Holmes
and the case of
The Missing Supercomputer Performance

"The Case of the Missing Supercomputer Performance: Achieving Optimal Performance on the 8,192 Processors of ASCI Q", F. Petrini, D.J. Kerbyson, S. Pakin, in Proc. of IEEE/ACM SC03, Phoenix, AZ, November 2003.

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Using Fewer PEs Per Node

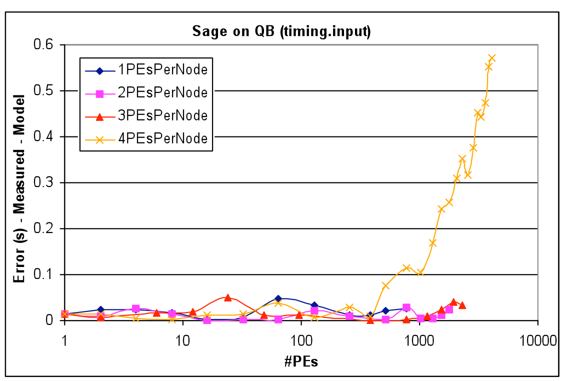
- Performance using 1, 2, 3, and 4 PEs per node
  - reduces the number of compute processors available



Pacific Northwest
NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Using Fewer PEs Per Node (2)

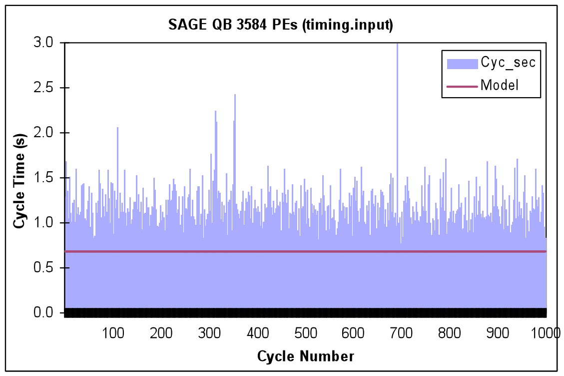
- Measurements match model almost exactly for 1, 2, and 3 PEs per node!



Performance issue only occurs when using 4 PEs per node

Performance Variability (1)

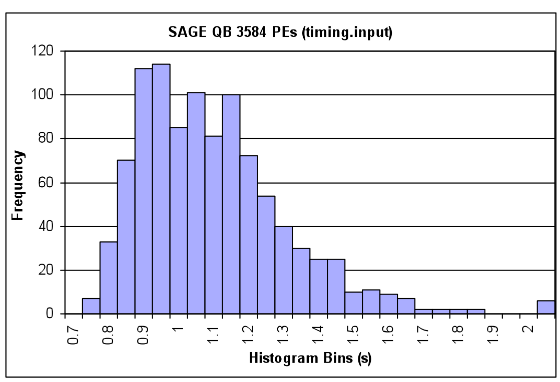
- Cycle time varies from cycle to cycle



Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Performance Variability (2)

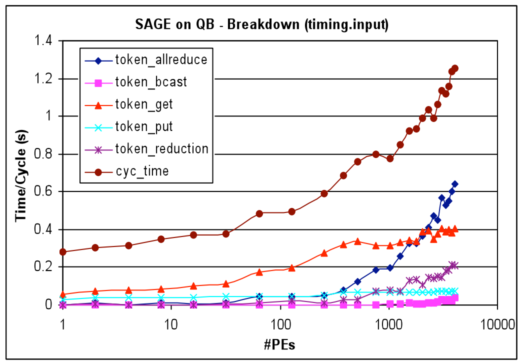
- Histogram of cycle time over 1000 cycles
- Over factor of 4 in range (0.75s → 3s)



Performance issue has variability (some cycles are not affected!)

SAGE Performance Components

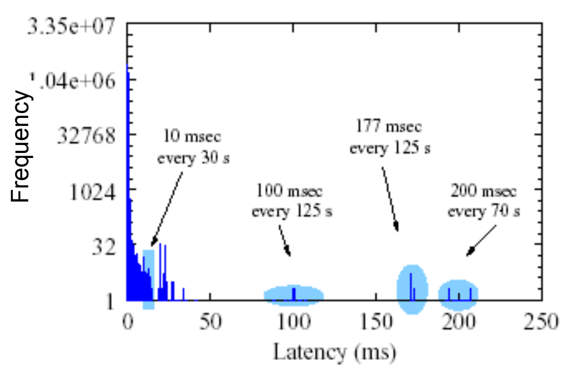
- Examine components of SAGE:
  - Put/Get (point-to-point boundary exchange)
  - Collectives (allreduce, broadcast, reduction)



Performance issue seems to occur only on collective operations

Delays Observed by a Micro-Benchmark

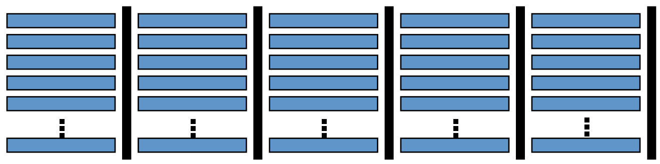
- Simple computation benchmark took exactly 1ms to execute
- Executed 1million iterations per processor
- Histogram plotted of time actually taken per node (= 4 PEs)



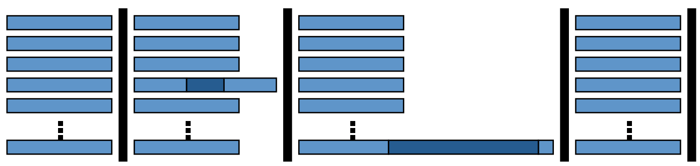
Pacific Northwest
NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Unknown Factor was Caused by the OS

- An application is usually a sequence of a computation followed by a synchronization (collective):



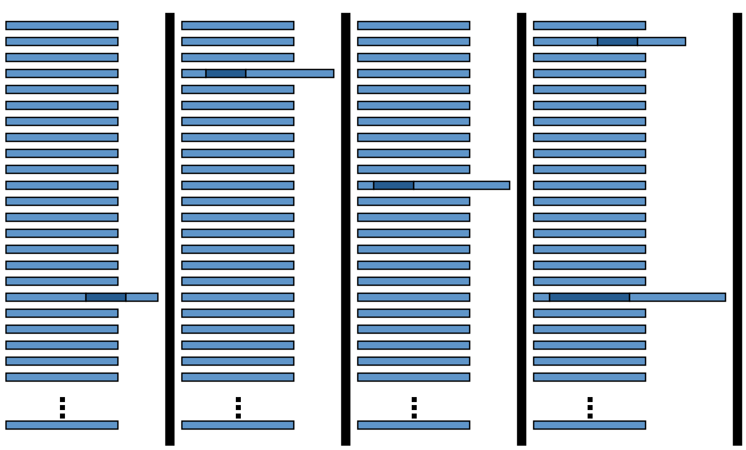
- But if an event happens on a single node then it can affect all the other nodes



"Computational Noise"

Pacific Northwest
NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

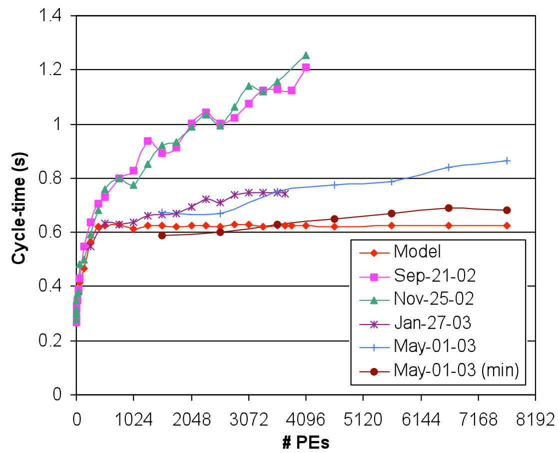
Effect Increases with Scale



- The probability of a random event occurring increases with the node count.

Pacific Northwest
NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

After OS Refinements



- Performance ASCI Q now within ~10% of our expectation
- Without a model we would not have identified (and solved) the poor performance!

Rational System Integration - Summary

- Models predicted ASCI Q performance in advance of installation
  - Based on single node performance, network performance, and knowledge of application factors
- Models can provide valuable performance data
- Do not believe everything you measure!
- Where possible have at least two data points for the same performance point from different sources
  - If there is a difference: diagnose and identify source of problem

Without modeling, it may have taken longer to realize there was a problem with ASCI Q!

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Tutorial Outline (the plan!)

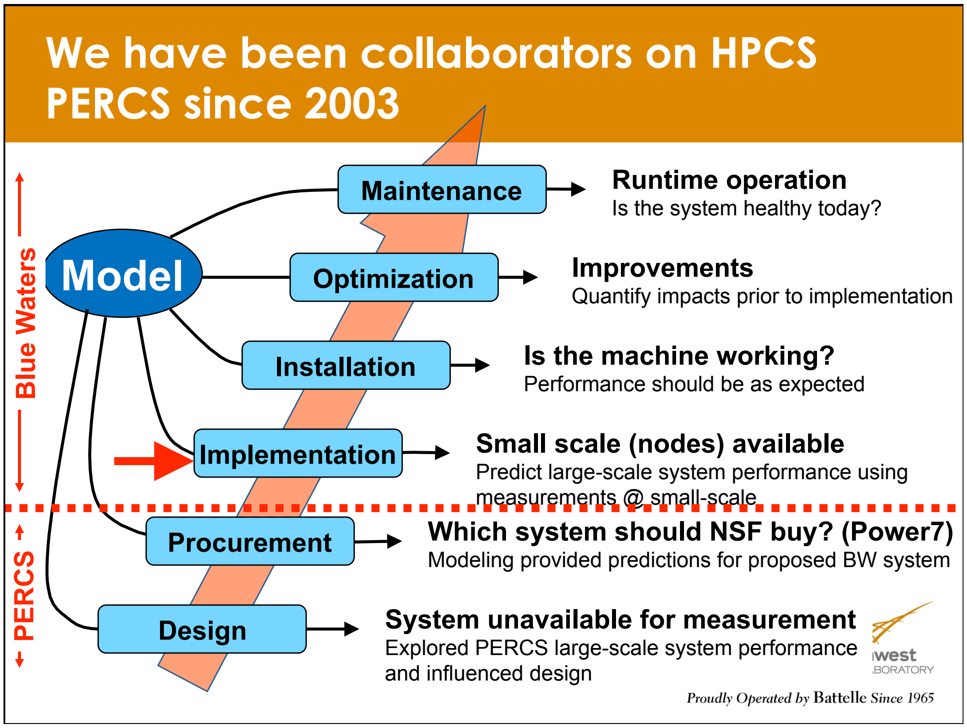
An Overview of Blue Waters for Modeling

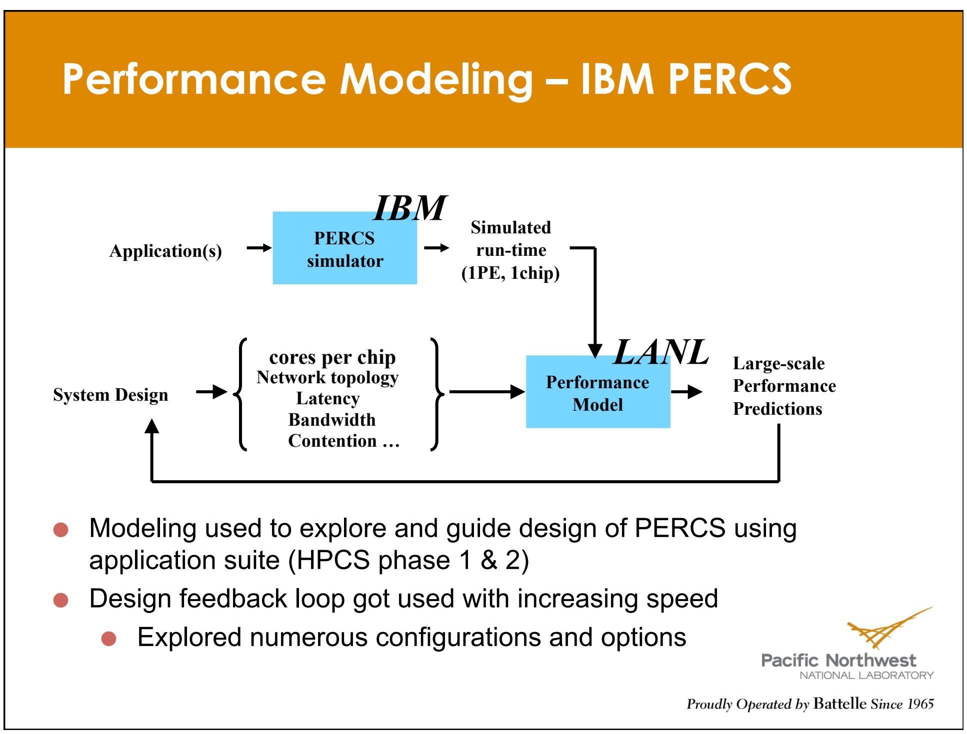
Key Aspect: Relay experiences on the reasoning for a new architecture when performance modeling

- Blue Waters: Innovative design
  - Different to current high end systems
  - Not a mesh, Not a Fat-tree, Not accelerated
- Large Core-count (> 200K)
- Large peak (multi peta-flop)
- Deep System Hierarchy
  - Communications
    - Differences in channel bandwidths and latencies
  - Task mapping

Note: This information is based on the view from our perspective. It is NOT an official view of either IBM or NCSA.

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

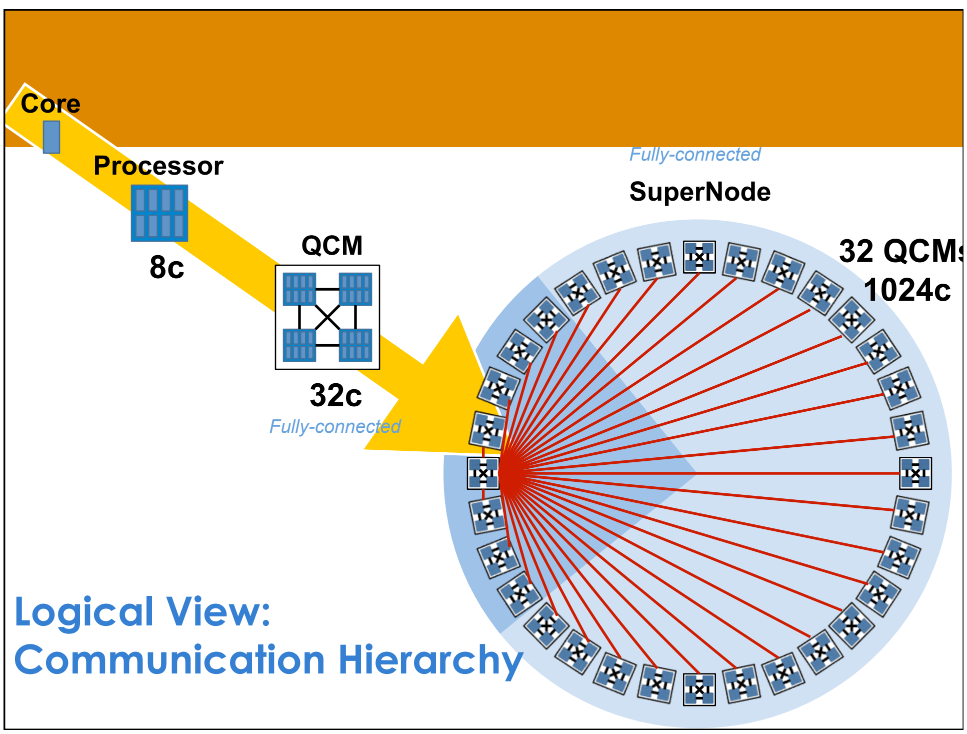

We have been collaborators on HPCS PERCS since 2003
Blue Waters
PERCS
Model
Maintenance
Runtime operation
Is the system healthy today?
Optimization
Improvements
Quantify impacts prior to implementation
Installation
Is the machine working?
Performance should be as expected
Implementation
Small scale (nodes) available
Predict large-scale system performance using measurements @ small-scale
Procurement
Which system should NSF buy? (Power7)
Modeling provided predictions for proposed BW system
Design
System unavailable for measurement
Explored PERCS large-scale system performance and influenced design
Proudly Operated by Battelle Since 1965


Performance Modeling – IBM PERCS
IBM
Application(s)
PERCS simulator
Simulated run-time (1PE, 1chip)
LANL
System Design
cores per chip
Network topology
Latency
Bandwidth
Contention …
Performance Model
Large-scale Performance Predictions
Modeling used to explore and guide design of PERCS using application suite (HPCS phase 1 & 2)
Design feedback loop got used with increasing speed
Explored numerous configurations and options
Pacific Northwest
NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

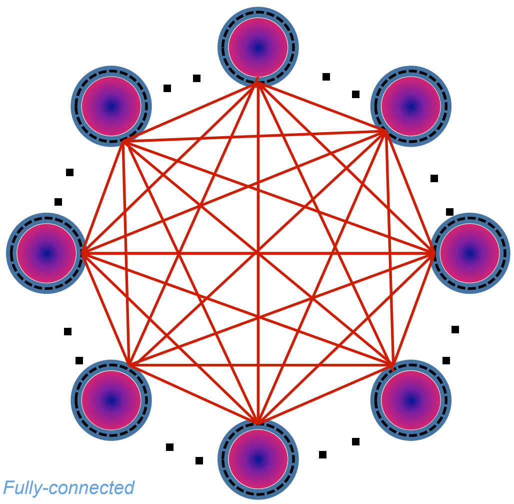
Processor Hierarchy

- Processor = 8x Power7 cores
  - on-chip shared L3 cache
  - 2x memory controllers supporting 8 channels DDR3 memory
- Quad-Chip-Module (QCM) = 4x Processors
  - Single socket
  - Direct communication channels between all 4 processors
  - Connection to communications Hub (Torrent)
- Drawer = 8x (QCM + Hubs)
  - Each Hub has a connection to each Hub on the same board
- SuperNode = 4x Boards
  - Each Hub has a connection to each other Hub on other boards in the Supernode
- System = up to 513x SuperNodes
  - Each SN has a connection to each other SN

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965



System = up to 513 Supernodes

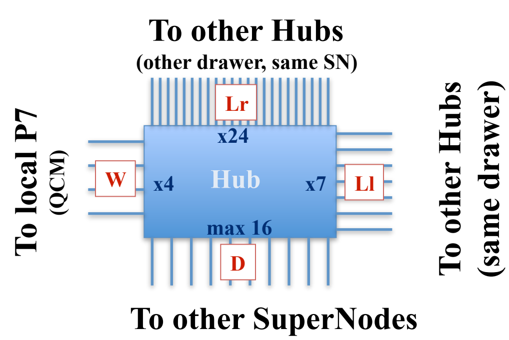


Fully-connected

- One channel between any two SNs
  - (8 in example)
- Communication from one SN to another can be done through an intermediate SN
  - Two stage comms

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Communication Hierarchy provided by Hubs



~1TB/s throughput

W (x4): local P7 QCM
Ll (x7): local SN, same drawer
Lr (x24): local SN, same SN
D (max 16): distant SN

Bandwidth ↑ Latency ↓

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Communication parameters for Modeling

- At present we have no measurements from hardware and can only assume parameter values
  - Bandwidths
  - Latencies
  - Collectives
- Values may vary depending on message size
- These will be firmed up over time

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Our (current) assumed communication parameters

	Latency	Bandwidth
QCM -> Hub (W)	0.05us	15 + 15 GB/s
Intra-drawer (Ll)	0.1us	15 + 15 GB/s
Intra-SN (lr)	0.2us	4 + 4 GB/s
Inter-SN (D)	~0.3us	7 + 7 GB/s

- Note this is only for small message latencies (per hop) and large-message bandwidths. Currently ignores detail which will be available closer to actual hardware delivery.
- Above does not include MPI software stack (0.5us)

Note that these numbers are for illustration purposes only and does not reflect actual performance characteristics of Blue Waters.

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

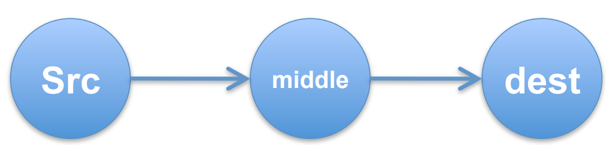
Routing Considerations

- Intra-SN
  - Direct Routing
  - Each Hub has direct connection to each other
- Inter-SN
  - Two possible operation modes:
    1. Direct Routing
       - Each SN pair has a *single* channel between them
    2. Indirect Routing
       - Use a middle SN “C”, when routing from SN “A” -> SN “B”
       - Take advantage of many of the channels from a SN
  - In the following we assume case 2

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Routing (continued)

- Inter-SN



Ll → D → Ll | Lr → D → Ll/Lr

- First step (Ll) enables use of any Hub in same drawer
  - 8*max16 = max 128 D links available (i.e. a max of 128 middle SNs)
- Middle step (Ll | Lr) routing to correct D-link exit
  - Only one D link to destination SN
- Last step routes within destination SN to dest Hub/core

Note: the maximum available D-links depends on system size

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Communication Cost

$$T_{msg} = L + S / B$$

$$T_{msg} = \sum L_i + \max (S / B_i)$$

- L_l = latency, B_i = bandwidth on link I, S= message size
- Sum latencies in the multi-hop routing
- Use the min bandwidth of the links used (max time)
- Straightforward. But …
  - Above only for single message without striping

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Communication Cost with striping

$$T_{msg} = \sum L_i + \max ((S / N_i) / B_i)$$

- N_i = number channels of type i
  - For inter-node:
    - Hub → 7 other Hubs (7x LI)
    - Hub → 128 other Hubs (128 x D)
    - But then fan into destination
  - In actual fact bandwidth limited by P7 → Hub Bandwidth
  - Also note message striped into 2KB packets
- Reasonably straightforward. But …
  - Above only for large <u>single</u> message with striping

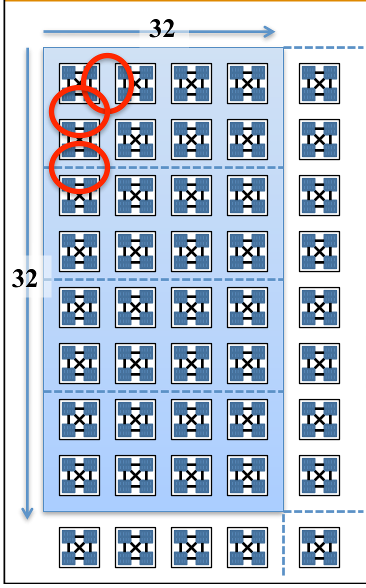
Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Communication Pattern Cost

$$T_{msg} = \sum L_i + \max (C_i \cdot (S / N_i) / B_i)$$

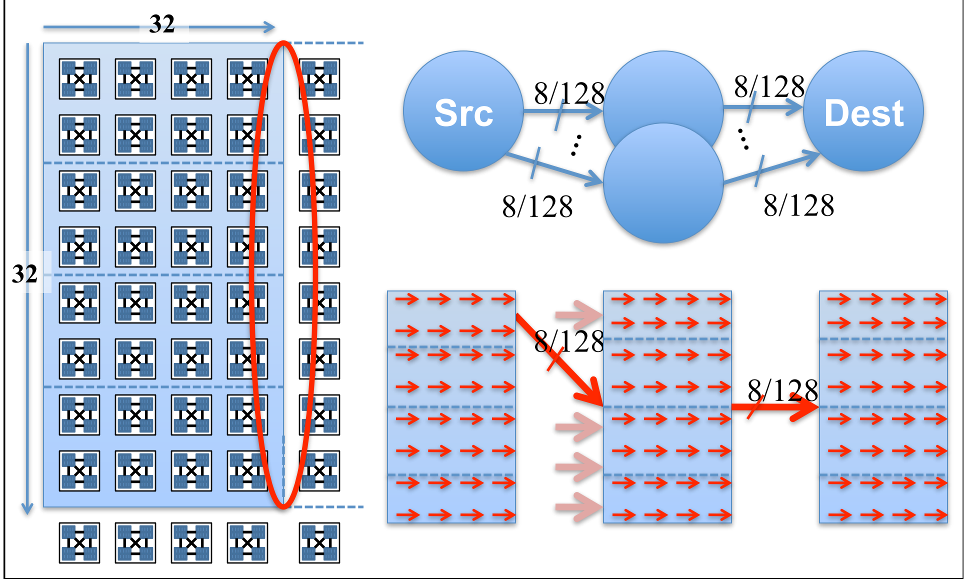
- C_i = contention (# of messages going over same channel)
- Value of C_i depends on communication pattern and also mapping
- Look at some examples:
  - 2D decomposition
  - Subset of All-to-all (e.g. DNS3D)
- We show what should be achieved, could be used to identify inefficiencies in practice

2D Example



- 1024 cores in an SN
- 2D: 32x32 processes
  - 4x2 per P7 processor
  - 2x2 x (4x2) per QCM =8x4
  - 4x8 QCMs
1. Intra-SN
  - +X-dim: C = 4 on LI channels
  - +Y-dim: C = 8 on either LI or Lr
- Note dependence on mapping

2D Example (continued) Inter-node: +X

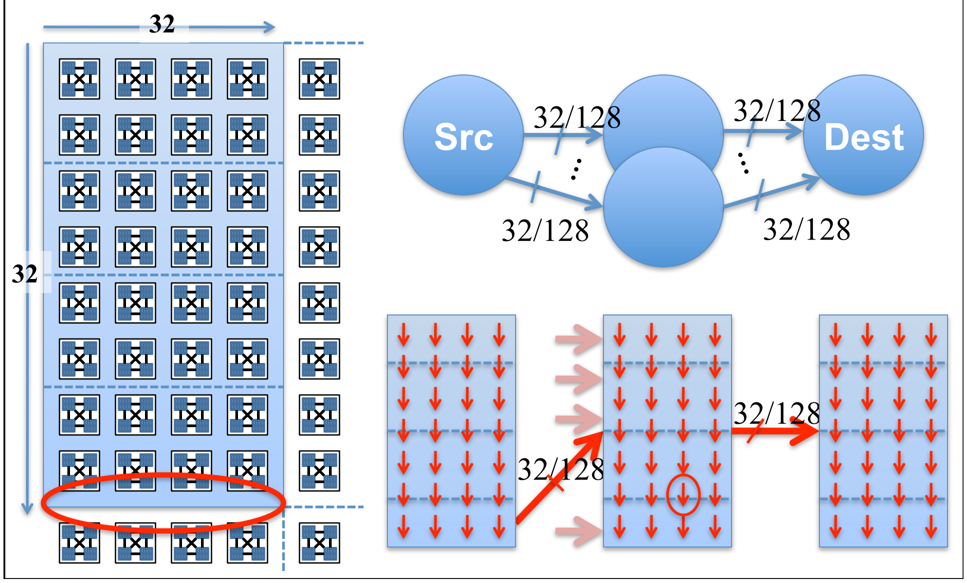


+X Dim

1. Intra-SN
   – C = 4 on Ll channels
2. Inter-SN
   – C = 2x 8/128 on D
   – C = 4 + 8/128 on Ll
   – C = 8/128 on Lr

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

2D Example (continued)
Inter-node: +Y

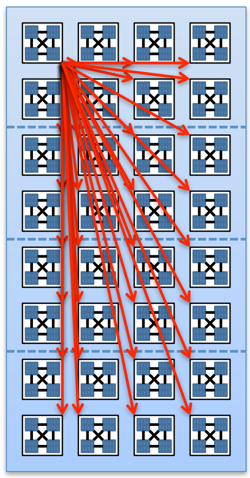


+Y Dim

1. Intra-SN
   - C = 8 on Ll channels
   - C = 8 on Lr channels
2. Inter-SN
   - C = 2x 32/128 on D
   - C = 8 + 32/128 on Ll
     = 8 + 32/128 on Lr

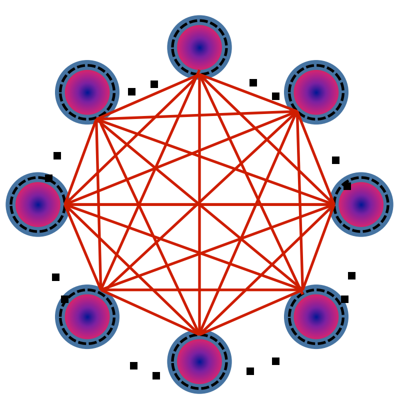
Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

All-to-all example: intra-node



- 1024 cores in an SN
  - All-to-all between all cores
  - C.f. one of the transpose in DNS3D
- C = 32x32 on each Ll
- C = 32x32 on each Lr

All-to-all example: inter-node



- 1024 cores in an SN
  - All-to-all between all cores
  - C.f. other transpose in DNS3D
- C = #SN x1024 on D
- Best case using direct routing

Summary Modeling Blue Waters

- We have provided a view of the Blue Waters processor and communication hierarchy
  - Focused on aspects impacting communication performance
  - Task layout and resulting communication contention
  - 2D example: intra- and inter-node communications
  - All-to-all performance
- Actual communication performance of Blue Waters not yet determined
- We are in the process of modeling several applications for Blue Waters for:
  - pre-delivery performance prediction
  - To assist with application and system optimizations
  - As tools for performance acceptance testing of the system
- Part of on-going performance modeling of the Power7 (since 2003)
- Stay tuned, it's going to be interesting !

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Tutorial Outline (the plan!)

	Page	Duration
Introduction and motivation		**20 mins**
Performance metrics & pitfalls		**30 mins**
Performance modeling methodology		**40 mins**
COFFEE BREAK		**30 mins**
Abstractions		**30 mins**
Case Studies		
I: SWEEP3D		**60 mins**
LUNCH BREAK		**90 mins**
II: SAGE		**30 mins**
III: DNS3D		**30 mins**
Applications of modeling		
I: Rational system integration		**30 mins**
COOKIE BREAK		**30 mins**
II: Novel Architectures: Blue Waters		**40 mins**
III: Performance comparison of large-scale systems		**40 mins**
Conclusions, lessons learned, wrap-up		**10 mins**

Large-Scale System Comparison

- Performance models can be used to compare the performance of large systems
  - Measurement is not always possible
    - Access may be limited
    - Systems may not yet be available (e.g., in the procurement of a future system)
  - Predict performance of a workload on a set of systems and compare
  - Determine the system characteristics that most limit performance
- We compare performance of three supercomputers on a realistic workload combining benchmarking and modeling
- The applications and their models for the workload considered, Sweep3D and SAGE, were described earlier

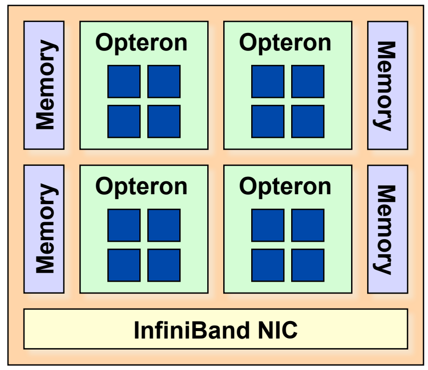
Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Systems Under Consideration

- Lobo: Conventional cluster
  - Commodity processors and network
- Dawn: Traditional massively parallel processor
  - Second-generation Blue Gene (Blue Gene/P)
  - Specially modified processors, custom networks
  - *Pros*: abundant parallelism, low-latency communication
  - *Cons*: weak processor cores, limited bandwidth
- Roadrunner: Hybrid, accelerated cluster
  - Commodity processors and network plus enhanced commodity processors as accelerators
  - *Pros*: immense peak performance per node, abundant parallelism
  - *Cons*: severely unbalanced communication-to-computation performance (few GB/s per flop/s) → significant NIC contention

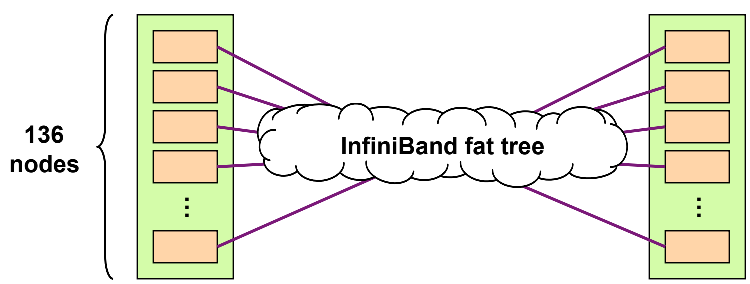
Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Lobo Node Architecture



- Quad-socket, quad-core CPUs
  - AMD Barcelona 8354 @ 2.2 GHz
- 32 GB of memory per node
  - 2 GB/core

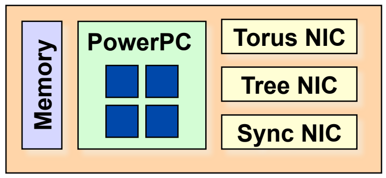
Lobo System Architecture



- 2 SUs × 136 nodes/SU × 4 sockets/node × 4 cores/socket = 4,352 cores (38.3 peak Tflop/s)
- 4x DDR InfiniBand (2 GB/s per link per direction)
- One 288-port InfiniBand switch

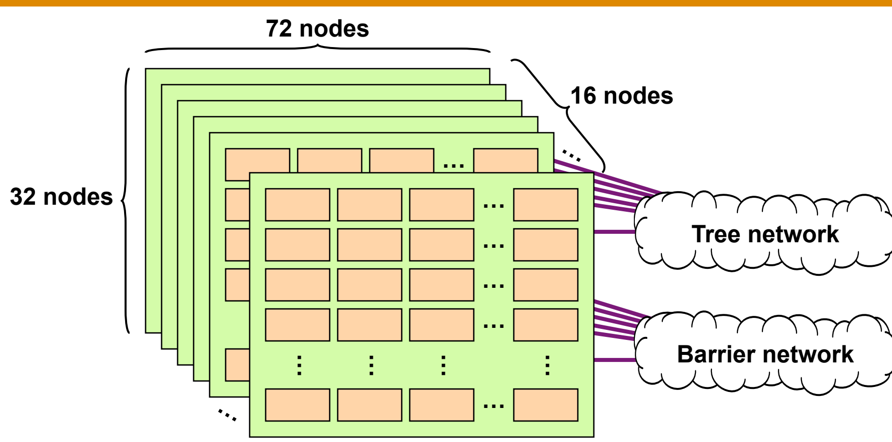
Pacific Northwest
NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Dawn Node Architecture



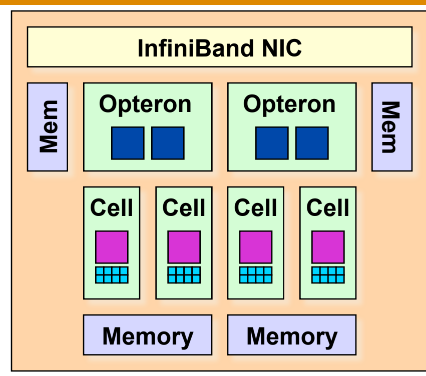
- Single-socket, quad-core CPUs
  - PowerPC 450d @ 850 MHz
- 4 GB of memory per node
  - 1 GB/core

Dawn System Architecture



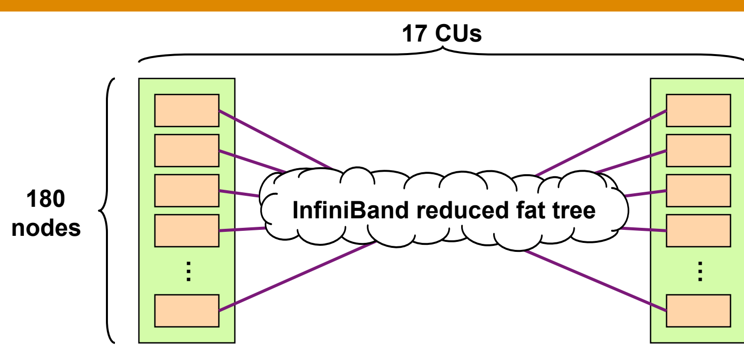
- 72 × 32 × 16 nodes × 4 cores/node = 147,456 cores (501.3 Tflop/s)
- 425 MB/s per torus link per direction × 6 links/node = 2.6 GB/s per direction per node

Roadrunner Node Architecture



- Dual-socket, dual-core CPUs
  - AMD Opteron 2210 @ 1.8 GHz
- 4 Cell/B.E. accelerators (one per CPU core)
  - PowerXCell 8i @ 3.2 GHz
- 32 GB of memory per node
  - 4 GB/Opteron core + 4 GB/Cell socket

Roadrunner System Architecture



- 17 CUs × 180 nodes/CU × {2,4} sockets/node × {2,9} cores/socket = 122,400 cores (1,393 peak Tflop/s)
- 4x DDR InfiniBand (2 GB/s per link per direction)
- 2 levels of InfiniBand (intra- and inter-CU)

Summary of Architectural Characteristics

Feature	Lobo	Dawn	RR
Cores/node	16	4	40
Nodes/system	272	36,864	3,060
Cores/system	4,352	147,456	122,400
Memory/node (GB)	32	4	32
Streams mem. BW/socket (GB/s)	7.4	10.0	22.2
Streams mem. BW/node (GB/s)	18.8	10.0	88.9
Network BW/node/dir. (GB/s)	2	2.5 (÷6)	2
Peak performance (Tflop/s)	38	501	1,393 (44 Base)

No one system is clearly superior → use performance models to compare

Pacific Northwest National Laboratory
Proudly Operated by Battelle Since 1965

Model Accuracy

- Maximum modeled error excluding outlying “rogue” points

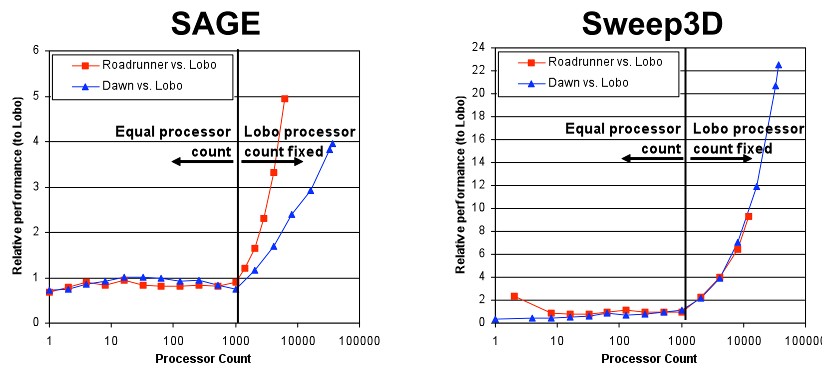
	Lobo	Dawn	Roadrunner	
SAGE	< 7%	< 10%	< 4%	
Sweep3D	< 14%	< 4%	< 8%	Non-Hybrid
			< 11%	Hybrid

FYI, two other applications we also looked at:

VPIC	< 6%	< 1%	< 4%	Non-Hybrid
			< 8%	Hybrid
Partisn	< 6%	< 12%	< 4%	

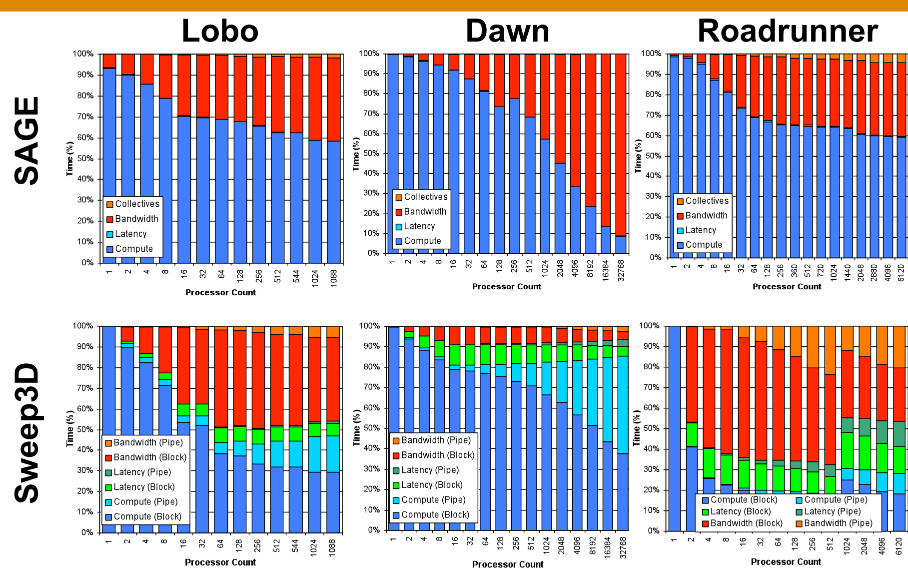
Pacific Northwest National Laboratory
Proudly Operated by Battelle Since 1965

Measuring Application Performance

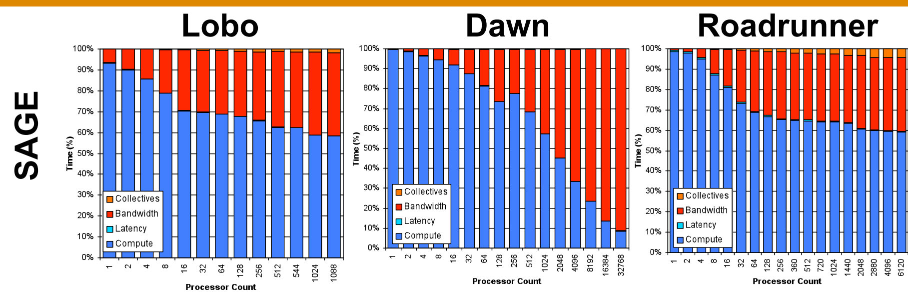


- Roadrunner Base > Dawn on SAGE
- Dawn > Roadrunner Hybrid on Sweep3D
- Can we use modeling to explain this discrepancy?

Using Modeling to Identify Performance Bottlenecks



Using Modeling to Identify Performance Bottlenecks

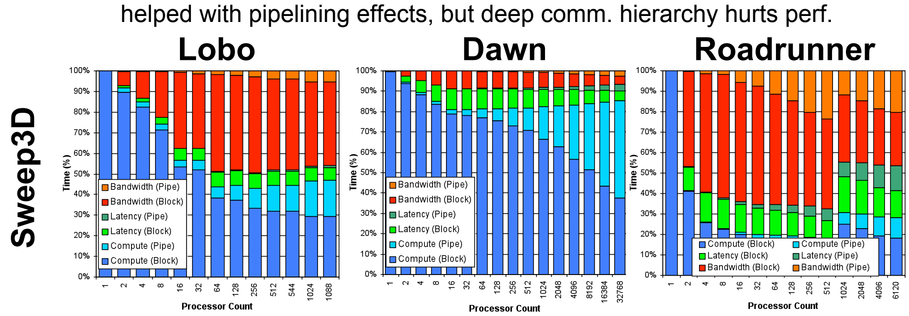


- SAGE transmits a large volume of large messages
- Lobo and Roadrunner Base (same IB fat-tree network) gradually lose performance to bandwidth
- Dawn's limited link bandwidth and susceptibility to network contention in the torus rapidly let bandwidth dominate performance

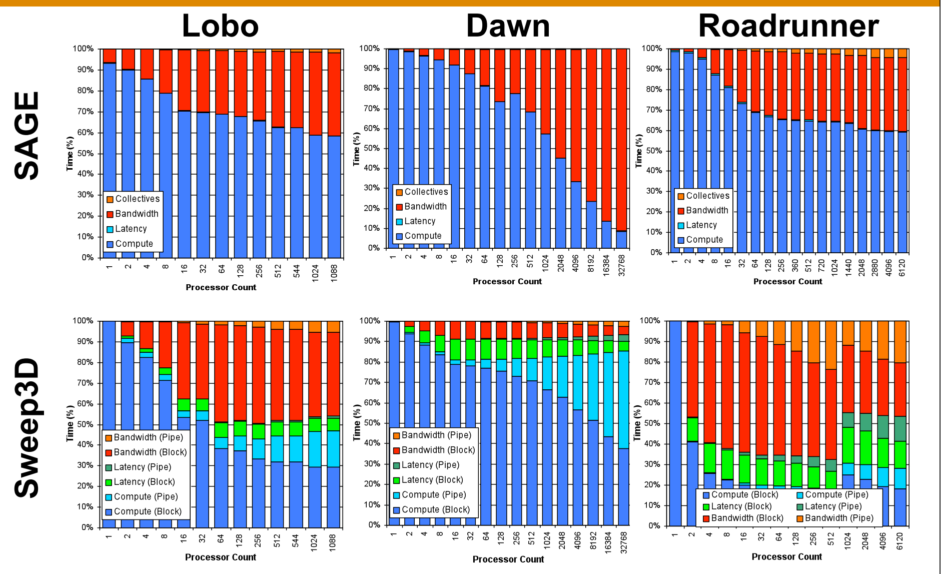
Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Using Modeling to Identify Performance Bottlenecks

- Sweep3D transmits a large number of small/medium-sized messages; also, pipeline effects limit parallel efficiency
- Would expect latency to dominate; in fact,
  - Few networks are bandwidth-optimized for Sweep3D's message sizes
  - Lobo is 50-50 compute/bandwidth due to NIC contention (16 procs)
  - Dawn spends 50% of its time stalled waiting for data (pipeline effects)
  - Roadrunner required different blocking at 2K procs; data aggregation helped with pipelining effects, but deep comm. hierarchy hurts perf.



Using Modeling to Identify Performance Bottlenecks



Summary

- Performance is workload-dependent
- Different systems → different bottlenecks
  - SAGE is compute-bound on Lobo and Roadrunner Base but bandwidth-bound on Dawn
  - Sweep3D is compute-bound on Dawn and Roadunner Base but communication bound on Roadrunner Hybrid and 50-50 compute/ communicate on Lobo
- Different applications → different bottlenecks
  - Dawn is bandwidth-bound on SAGE but compute-bound on Sweep3D
- Modeling can help explain performance measurements
  - Dawn has more processors than Roadrunner Base, but Roadrunner Base is faster on SAGE
    - » Model shows Dawn's relatively poor bandwidth limits its performance
  - Roadrunner Hybrid has higher per-node peak than Dawn, but Dawn is faster on Sweep3D
    - » Model shows Roadrunner Hybrid is bottlenecked by communication

Tutorial Outline (the plan!)

"If you've enjoyed this program just
half as much
as we've enjoyed doing it, then we've enjoyed
doing it twice as much as you."

\- Monty Python

"Some people have a way with words,
and other people...not have way."

\- *Steve Martin*

Pacific Northwest
NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Summary

- Modeling and predicting the performance of large-scale applications on HPC systems is one of the great challenges for computer science
- The predictive capability you have seen in this tutorial is currently being used for a variety of tasks at PNNL and elsewhere within DOE
- Our goal is to establish performance engineering as a standard practice

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Performance Engineering

Performance-engineered system: The components (application and system) are ***parameterized*** and ***modeled***, and a ***constitutive model is proposed*** and ***validated***.

Predictions are made based on the model. The model is meant to be ***updated, refined***, and ***further validated*** as new factors come into play.

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Capabilities and Limitations

- We do:
  - Model full applications
  - Validate on systems with thousands of CPUs
- We do not:
  - Model / predict single-CPU time
  - Account for memory contention within an SMP
    - Could be done empirically
  - Account for non-algorithmic comm/comm and comm/comp overlap
  - Account for operating system effects within the application model
    - These are measured and modeled separately
    - We still have a dedicated, single-application view
      - Throughput, scheduling issues modeled separately
  - Model / predict I/O performance

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Final Thoughts

- Application / architecture mapping is the key - not lists of basic machine characteristics (speeds & feeds)
  - Kernels alone do not characterize the performance of a supercomputer
- Performance studies need to address a specific workload
- Performance and scalability modeling is an effective “tool” for workload characterization, system design, application optimization, and algorithm-architecture mapping
  - The model is the tool
- Back-of-the-envelope performance predictions are risky (outright wrong?), given the complexity of analysis in a multidimensional performance space

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Acknowledgements and Disclaimers

- Previous colleagues at LANL
  - Kei Davis
  - Michael Lang
  - Scott Pakin
  - Jose Carlos Sancho
  - Harvey J. Wasserman
- Thanks to:
  - US Department of Energy

- Note: Any benchmark results presented herein reflect our workload. Results from other workloads may vary.

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Some Publications

Sweep3D

A. Hoisie, O. Lubeck, H. Wasserman, F. Petrini, H. Alme, "A General Predictive Performance Model for Wavefront Algorithms on Clusters of SMPs," In Proc. of ICPP, Toronto, Canada, August 2000.

A. Hoisie, O. Lubeck, H. Wasserman, "Performance and Scalability Analysis of Teraflop-Scale Parallel Architectures Using Multidimensional Wavefront Applications", The International Journal of High Performance Computing Applications, Sage Science Press, 14(4), Winter 2000.

A. Hoisie, O. Lubeck, H. Wasserman. Scalability Analysis of Multidimensional Wavefront Algorithms on Large-Scale SMP Clusters. In Proc. of Frontiers of Massively Parallel Computing '99, Annapolis, MD, February 1999.

SAGE

D.J. Kerbyson, H.J. Alme, A. Hoisie, F. Petrini, H.J. Wasserman, M. Gittings, "Predictive Performance and Scalability Modeling of a Large-Scale Application," in Proc. of IEEE/ ACM SC01, Denver, November 2001.

Tycho / UMT2K

D.J. Kerbyson, S.D. Pautz, A. Hoisie, "Predictive Modeling of Deterministic Sn Transport", in Performance Analysis for Parallel and Distributed Computing, Kluwer, 2003.

M. Mathis, D.J. Kerbyson, “A General Performance Model of Structured and Unstructured Mesh Particle Transport Computations", J. Supercomputing, Vol. 34, No. 2, pp. 181-199, 2005.

System Modeling and Comparisons

D.J. Kerbyson, A. Hoisie, H.J. Wasserman, "A Performance Comparison between the Earth Simulator and other Top 5 Terascale Systems on a characteristic ASCI Workload", to appear Concurrency & Computation Practice and Experience, Vol.17, No. 10, pp. 1219-1238, 2005.

D.J. Kerbyson, A. Hoisie, H.J. Wasserman, “A Comparison Between the Earth Simulator and AlphaServer Systems Using Predictive Application Performance Models," in Proc. IPDPS, Nice, France, April 2003.

D.J. Kerbyson, H.J. Wasserman, A. Hoisie, "Exploring Advanced Architectures using Performance Prediction", in Innovative Architecture for Future Generation High-Performance Processors and Systems, IEEE Computer Society Press, 2002, pp. 27-37

Resource Management

E. Frachtenberg, D. Feitelson, F. Petrini, J. Fernandez.”Flexible CoScheduling: Mitigating Load Imbalance and Improving Utilization of Heterogeneous Resources”, in Proc. IPDPS, Nice, France, April 2003. Best Paper Award.

E. Frachtenberg, F. Petrini, J. Fernandez, S. Pakin, S. Coll. "STORM: Lightning-Fast Resource Management", In Proc. IEEE/ACM SC2002, Baltimore, MD, November 2002

E. Frachtenberg, F. Petrini, S. Coll, W. Feng, "Gang Scheduling with Lighweight User-Level Communication. In Proc. ICPP2001, Wksp on Scheduling and Resource Management for Cluster Computing, Valencia Spain, September 2001.

Network Architecture

D. Addison, J. Beecroft, D. Hewson, M. McLaren, F. Petrini. “Quadrics QsNet II: A network for Supercomputing Applications” In *Hot Chips 14*, Stanford, CA, August 2003

S. Coll, F. Petrini, E. Frachtenberg, A. Hoisie, “Performance Evaluation of I/O Traffic and Placement of I/O Nodes on a High Performance Network.” In Wksp on Communication Architecture for Clusters 2002 (CAC '02), IPDPS, Fort Lauderdale, FL, April 2002.

F. Petrini, S. Coll, E. Frachtenberg, A. Hoisie, L. Gurvits,"Using Multirail Networks in High-Performance Clusters.” In IEEE Cluster 2001, Newport Beach, CA, October 2001

F. Petrini ,W. Feng, A. Hoisie, S. Coll, E. Frachtenberg., “The Quadrics Network (QsNet): High-Performance Clustering Technology.” In Hot Interconnects 9, Stanford University, Palo Alto, CA, August 2001.

F. Petrini, S. Coll, E. Frachtenberg, A. Hoisie, "Hardware and Software Based Collective Communication on the Quadrics Network. In IEEE International Symposium on Network Computing and Applications 2001 (NCA 2001), Boston, MA, October 2001

System Integration

F. Petrini, D.J. Kerbyson, S. Pakin, “The Case of the Missing Supercomputer Performance: Achieving Optimal Performance on the 8,192 Processors of ASCI Q”, in Proc. of IEEE/ACM SC03, Phoenix, AZ, November 2003. Best Paper Award.

D.J. Kerbyson, A. Hoisie, H.J. Wasserman, "Verifying Large-Scale System Performance During Installation using Modeling", in Hardware/Software Support for Parallel and Distributed Scientific and Engineering Computing, Kluwer, 2003.

System Performance Analysis

K. Davis, A. Hoisie, G. Johnson, D.J. Kerbyson, M. Lang, S. Pakin, F. Petrini, "A Performance and Scalability Analysis of the BlueGene/L Architecture", to appear in Proc. SuperComputing, Pittsburgh, November 2004.

D.J. Kerbyson, M. Lang, G. Patino, H. Amidi, "An Empirical Performance Analysis of Commodity Memories in Commodity Servers", in Proc. of ACM Workshop on Memory System Performance, Washington DC, June 2004.

D.J. Kerbyson, A. Hoisie, S. Pakin, F. Petrini, H. J. Wasserman, "A Performance Evaluation of an Alpha EV7 Processing Node", Int. Journal of High Performance Computing Applications, Vol. 18, No. 2, Sage Publications, 2004, pp. 199-209.

Performance Analysis - Book

S. Goedecker and A. Hoisie, "Performance Optimization of Numerically Intensive Codes (Software, Environments, Tools), Paperback - 173 pages, 2001, Society for Industrial & Applied Mathematics; ISBN: 0898714842

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

About the Authors

Adolfy Hoisie is a Laboratory Fellow, Director of the Center for Advanced Architectures, and the Leader of HPC at the Pacific Northwest National Laboratory. He joined PNNL in 2010 after spending 13 years at Los Alamo National Laboratory. From 1987 to 1997, he was a researcher at Cornell University. His area of research is performance analysis and modeling of systems and applications. He has published extensively, lectured at numerous conferences and other important events in his area worldwide. He was the winner of the Gordon Bell Award in 1996, and co-author to the recently published SIAM monograph on performance optimization.

Darren Kerbyson is a Laboratory Fellow at the Pacific Northwest National Laboratory. He received his BSc in Computer Systems Engineering in 1988, and PhD in Computer Science in 1993 both from the University of Warwick (UK). Prior to joining PNNL in 2010 he was the lead of the Performance and Architecture Lab at Los Alamos National Laboratory for almost 10 years. He was previously a senior member of faculty in Computer Science at the University of Warwick in the UK. His research interests include performance evaluation, performance modeling, and performance optimization of applications on high performance systems as well as image analysis. He has published over 130 papers in these areas over the last 20 years. He is a member of the IEEE Computer Society.

Pacific Northwest
NATIONAL LABORATORY

Proudly Operated by Battelle *Since 1965*

About the Authors (cont.)

Kevin Barker joined the HPC group at the Pacific Northwest National Laboratory in 2010. He previously spent 5 years as a member of the Performance and Architecture Lab (PAL) team at Los Alamos National Laboratory. His current research interests include developing performance modeling methodologies and tools for HPC systems and workloads as well as understanding how current and future architectures impact workload performance. He has published papers in the areas of dynamic load balancing, HPC middle-ware systems, performance modeling, and future network architectures. He received his B.S. in computer science from North Carolina State University in 1997, his M.S. in computer science from the University of Notre Dame in 2001, and his Ph.D. in computer science from the College of William and Mary in 2004.

Pacific Northwest
NATIONAL LABORATORY

Proudly Operated by Battelle *Since 1965*

Glossary

- bandwidth
  - The rate at which data can be transfered from one process to another, often measured in MB/s
- cell
  - A unit of application data (e.g., an array element); may correspond to a physical entity (e.g., an atom)
- CPU core
  - The minimal unit of hardware capable of computation
- (global) grid
  - An application's primary data structure, distributed across all processes; may correspond to a physical entity (e.g., a 3-D volume of particles)
- grid point
  - See *cell*
- latency
  - The time from when a sending process initiates a message transfer to when a destination process receives it, often measured in μs
  - (May imply a *minimally sized* message transfer)
- NIC (network interface controller)
  - A communication endpoint; a node's entry point into the interconnection network
- node
  - A component of a parallel system containing at least one CPU, NIC, and memory subsystem

Glossary (cont.)

- PE (processing element)
  - See *process*
- performance model
  - A formal expression of an application's execution time in terms of the execution times of various system resources
- process
  - A software construct capable of performing computation; has its own, private memory space
- processor
  - A socket
  - A CPU core
  - A process
- socket
  - A hardware package containing at least one CPU core and also typically caches, a memory interface, and signaling pins to connect to memory, other sockets, and I/O
- strong scaling
  - When increasing the process count, keeping the application's global grid size constant (and ∴ reducing the subgrid size proportionally); represents using parallelism to reduce execution time while keeping accuracy constant
- subgrid
  - A single process's subset of the global grid

Glossary (cont.)

- surface-to-volume ratio
  - The ratio of the number of cells at one process that must be communicated to another process divided by the total number of cells at that process; lower ratios indicate higher computational efficiencies
- weak scaling
  - When increasing the process count, increasing the application's global grid size proportionally (and ∴ keeping the subgrid size constant); represents using parallelism to increase accuracy while keeping time constant

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Performance Modeling

"Prediction is difficult - especially for the future."

- *Y. Berra*

"The future will be just like the present - only more so."

- *Groucho Marx*

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Case Studies

"One good, solid hope is worth a carload of certainties"

- *The Doctor, Dr. Who*

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*