

Life at the Near Petascale Edge: A Tale of Two Applications

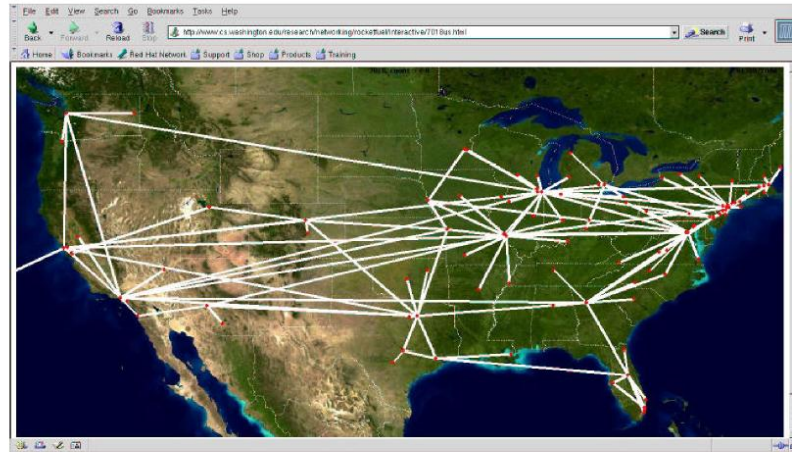
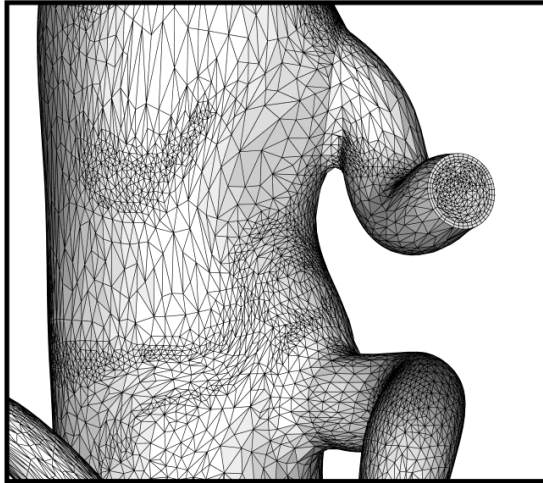


Fig. 5. AT&T Network Topology (AS 7118) from the Rocketfuel data bank for the continental US.

Christopher D. Carothers

Department of Computer Science

Rensselaer Polytechnic Institute

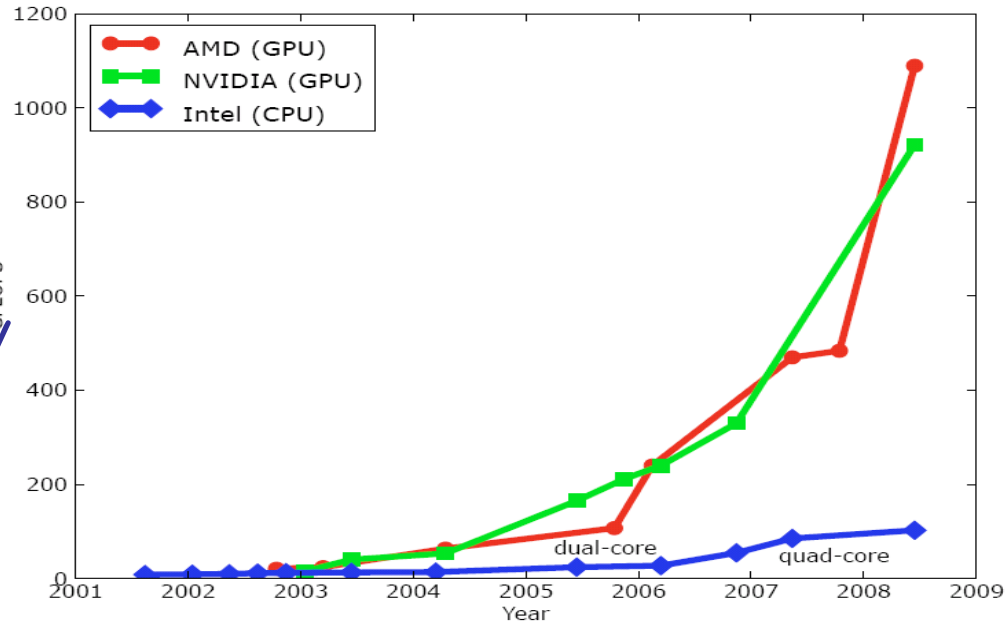
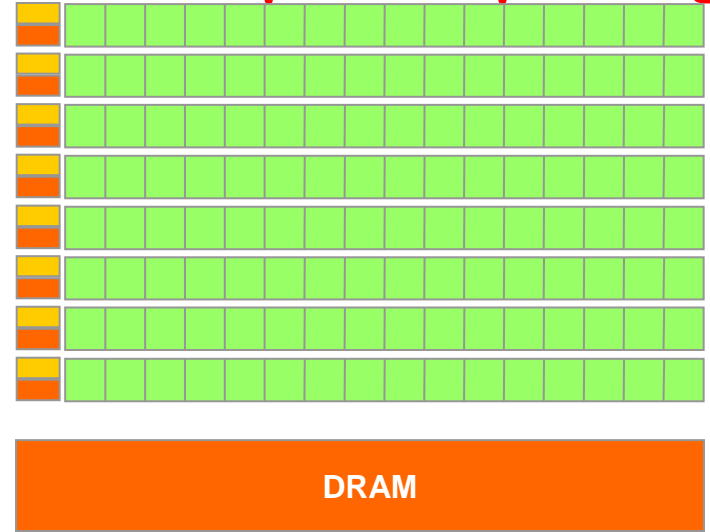
chrisc@cs.rpi.edu



Outline

- Overview of HPC Platforms
 - CUDA - Compute Unified Device Architecture
 - IBM Blue Gene/L, /P & /Q
- App 1: PHASTA -- CFD Solver using MPI
 - Implementation
 - Performance Results
- App 2: ROSS - PDES using MPI
 - Implementation
 - Performance Results
- Summary and Future Challenges

CUDA - Massively Parallel Desktop Computing



- Impressive Performance Capability
 - Calc: 1288 GFLOPS vs. 32 GFLOPS
 - Memory BW: 144 GB/s vs. 8 GB/s
 - Threads: 10's of 1000s of active threads
 - Upto 512 cores
- Highly available platform & relatively low cost:
 - Telsa C2050: \$2500
 - GTX 480: \$500
- No free lunch:
 - Push lots of details onto programmer!!
 - Exposes all layers of memory hierarchy
 - Radically changes program structure from serial
 - Cards have limited memory resources (peak of 6 GB today)

CUDA Example: Reduction

- What are challenges and optimizations to performing a reduction operation
 - E.g., sum, min or max across an array of data elements
 - Turns out memory bandwidth is the key limiting factor
 - There are seven levels of optimization required to obtain peak performance



G80 Performance for 4M element reduction

	Time (2^{22} ints)	Bandwidth	Step Speedup	Cumulative Speedup
Kernel 1: interleaved addressing with divergent branching	8.054 ms	2.083 GB/s		
Kernel 2: interleaved addressing with bank conflicts	3.456 ms	4.854 GB/s	2.33x	2.33x
Kernel 3: sequential addressing	1.722 ms	9.741 GB/s	2.01x	4.68x
Kernel 4: first add during global load	0.965 ms	17.377 GB/s	1.78x	8.34x
Kernel 5: unroll last warp	0.536 ms	31.289 GB/s	1.8x	15.01x
Kernel 6: completely unrolled	0.381 ms	43.996 GB/s	1.41x	21.16x
Kernel 7: multiple elements per thread	0.268 ms	62.671 GB/s	1.42x	30.04x

Kernel 7 on 32M elements: 73 GB/s!



CCNI

COMPUTATIONAL CENTER for
NANOTECHNOLOGY INNOVATIONS

```

template <unsigned int blockSize>
__global__ void reduce6(int *g_idata, int *g_odata, unsigned int n)
{
    extern __shared__ int sdata[];

    unsigned int tid = threadIdx.x;
    unsigned int i = blockIdx.x*(blockSize*2) + tid;
    unsigned int gridSize = blockSize*2*gridDim.x;
    sdata[tid] = 0;

    while (i < n) { sdata[tid] += g_idata[i] + g_idata[i+blockSize]; i += gridSize; }
    __syncthreads();

    if (blockSize >= 512) { if (tid < 256) { sdata[tid] += sdata[tid + 256]; } __syncthreads(); }
    if (blockSize >= 256) { if (tid < 128) { sdata[tid] += sdata[tid + 128]; } __syncthreads(); }
    if (blockSize >= 128) { if (tid < 64) { sdata[tid] += sdata[tid + 64]; } __syncthreads(); }

    if (tid < 32) {
        if (blockSize >= 64) sdata[tid] += sdata[tid + 32];
        if (blockSize >= 32) sdata[tid] += sdata[tid + 16];
        if (blockSize >= 16) sdata[tid] += sdata[tid + 8];
        if (blockSize >= 8) sdata[tid] += sdata[tid + 4];
        if (blockSize >= 4) sdata[tid] += sdata[tid + 2];
        if (blockSize >= 2) sdata[tid] += sdata[tid + 1];
    }

    if (tid == 0) g_odata[blockIdx.x] = sdata[0];
}

```

Final Optimized Kernel

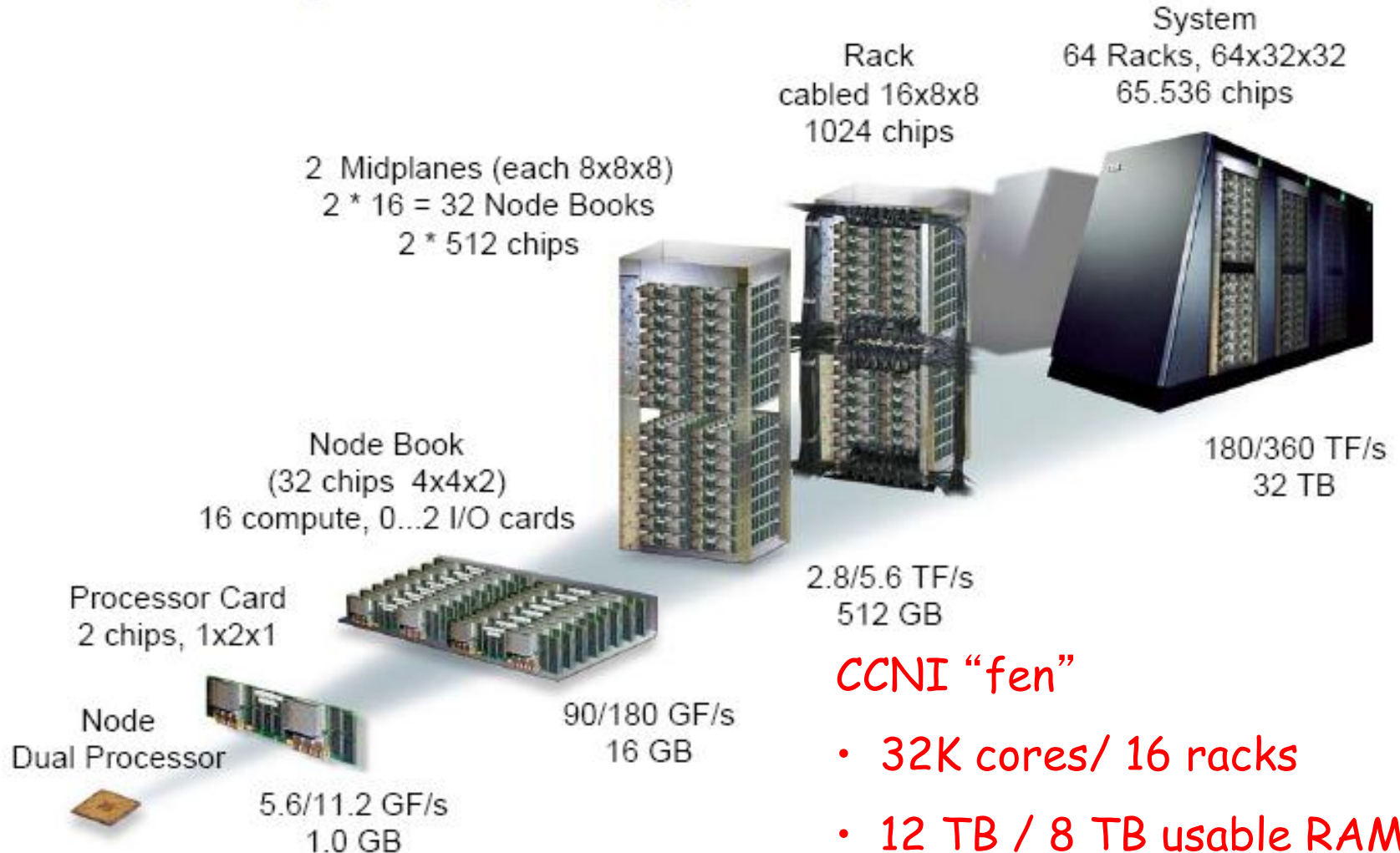
```

// int specific reduce on CPU
int reduceCPU(int *data, int size)
{
    int sum = data[0];
    for (int i = 1; i < size; i++)
    {
        sum += data[i];
    }
    return sum;
}

```



Blue Gene /L Layout

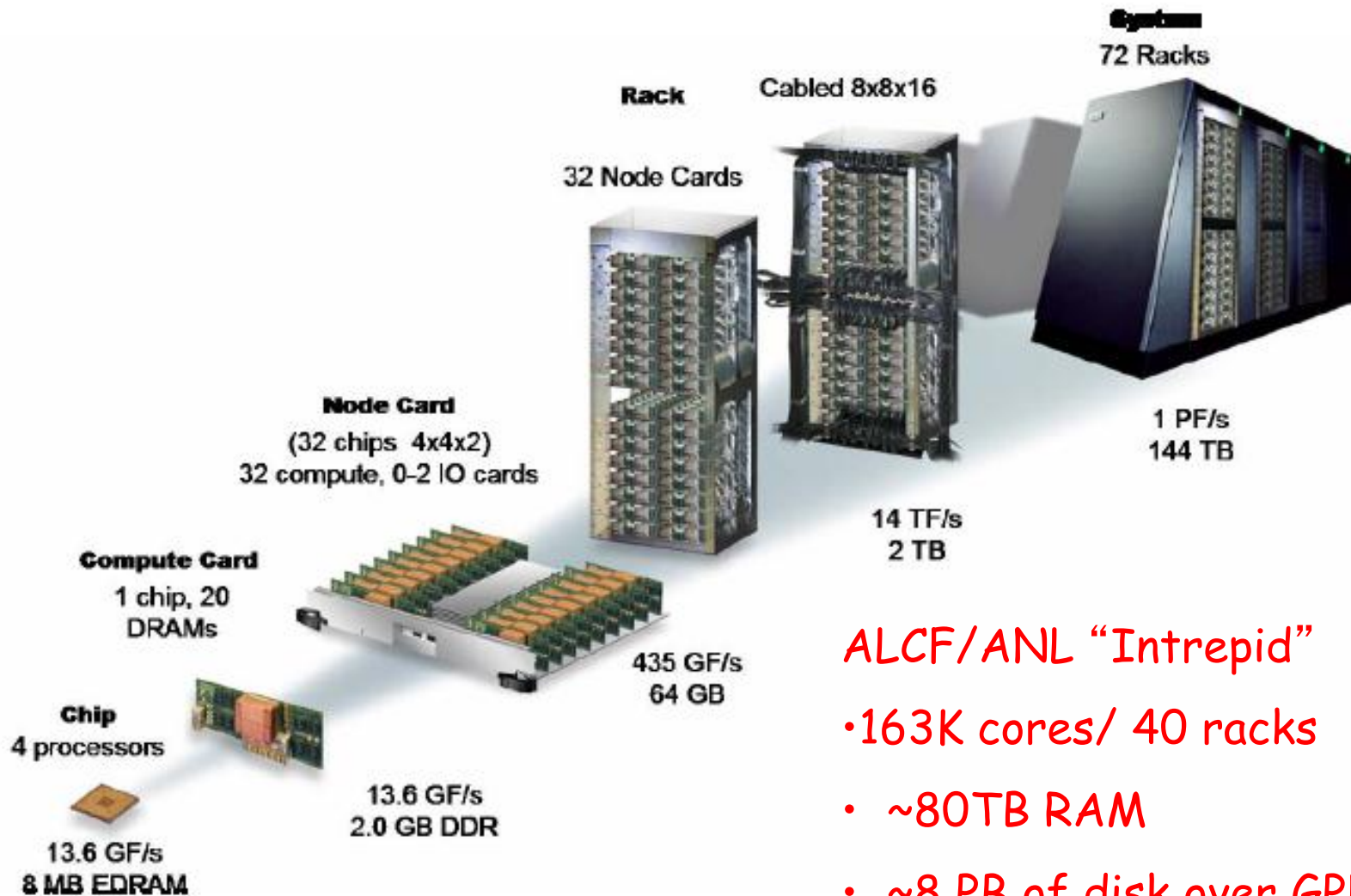


CCNI "fen"

- 32K cores/ 16 racks
- 12 TB / 8 TB usable RAM
- ~1 PB of disk over GPFS
- Custom OS kernel



Blue Gene /P Layout



ALCF/ANL "Intrepid"

- 163K cores/ 40 racks
- ~80TB RAM
- ~8 PB of disk over GPFS
- Custom OS kernel



Blue Gene: L vs. P

Property		BG/L	BG/P
Node Properties	Node Processors	2 * 440 PowerPC	4 * 450 PowerPC
	Processor Frequency	0.7 GHz	0.85 GHz
	Coherency	Software managed	SMP
	L1 Cache (private)	32KB / core	32KB / core
	L2 Cache (private)	14 stream prefetching	14 stream prefetching
	L3 Cache size (shared)	4 MB	8 MB
	Main Store/Node	512 MB, later 1 GB version	2 GB
	Main Store Bandwidth	5.6 GB/s (16B wide)	13.6 GB/s (2*16B wide)
	Peak Performance	5.6 GF/node	13.6 GF/node
Torus Network	Bandwidth	6*2*175MB/s = 2.1 GB/s	6*2*425MB/s = 5.1 GB/s
	Hardware Latency (Nearest Neighbor)	200 ns (32B packet) 1.6 us (256B packet)	100 ns (32B packet) 800 ns (256B packet)
	Hardware Latency (Worst Case)	6.4 us (64 hops)	3.0 us (64 hops)
Collective Network	Bandwidth	2*350MB/s = 700 MB/s	2*0.85GB/s = 1.7 GB/s
	Hardware Latency (Round Trip Worst Case)	5.0 us	3.0 us
System Properties	Peak Performance	360 TF (64k nodes)	1 PF (72k nodes)
	Total Power	1,7 MW	TBD (about 2,3 MW)

Blue Gene /Q will have 1.6M cores (6.4 M threads) & use a 5-D Torus



NSF PetaApps: Parallel Adaptive CFD

•PetaApps Components

- **CFD Solver**
- Adaptivity
- **Petascale Perf Sim**
 - **Fault Recovery**
- Demonstration Apps
 - Cardiovascular Flow
 - Flow Control
 - Two-phase Flow

Ken Jansen (PD),
Onkar Sahni,
Chris Carothers,
Mark S. Shephard

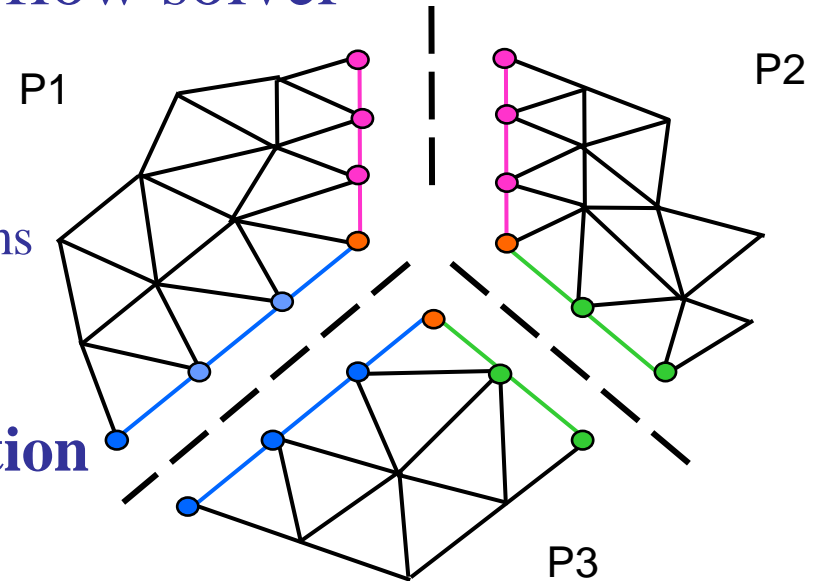


Scientific Computation Research Center
Rensselaer Polytechnic Institute

Acknowledgments: Partners: Simmetrix, Acusim, Kitware, IBM
NSF: PetaApps, ITR, CTS; DOE: SciDAC-ITAPS, NERI; AFOSR
Industry: IBM, Northrup Grumman, Boeing, Lockheed Martin, Motorola
Computer Resources: TeraGrid, ANL, NERSC, RPI-CCNI

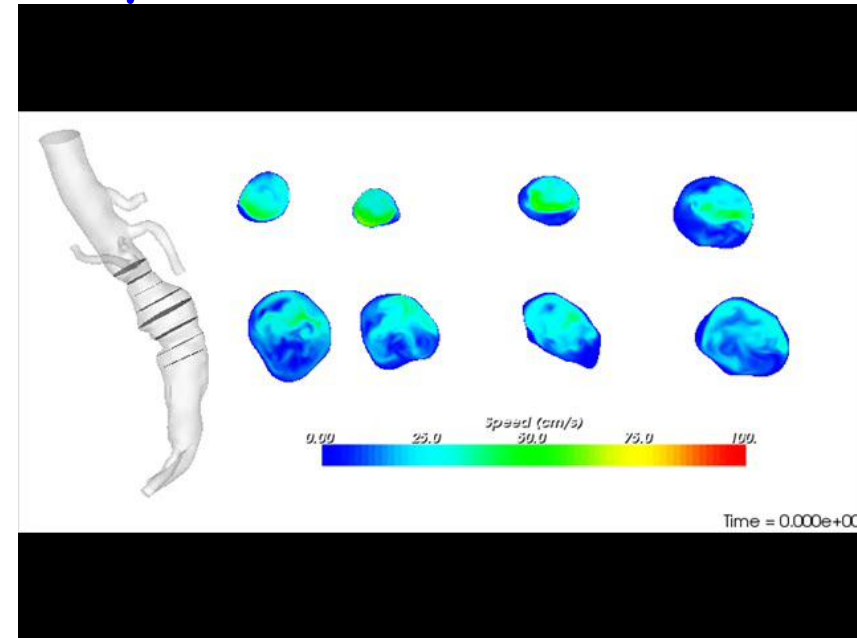
PHASTA Flow Solver Parallel Paradigm

- Time-accurate, stabilized FEM flow solver
- Two types of work:
 - **Equation formation**
 - $O(40)$ peer-to-peer non/blocking comms
 - Overlapping comms with comp
 - Scales well on many machines
 - **Implicit, iterative equation solution**
 - Matrix assembled on processor **ONLY**
 - Each Krylov vector is:
 - $q=Ap$ (matrix-vector product)
 - Same peer-to-peer comm of q PLUS
 - Orthogonalize against prior vectors
 - **REQUIRES NORMS= \Rightarrow MPI_Allreduce**
 - This sets up a cycle of global comms. separated by modest amount of work
 - Not currently able to overlap Comms

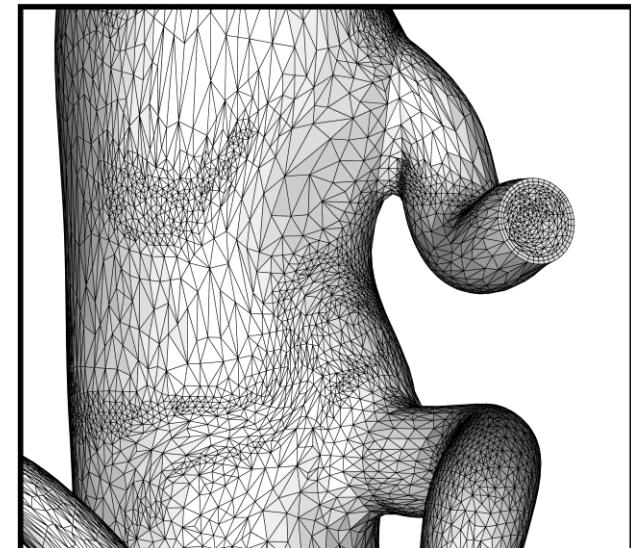


Parallel Implicit Flow Solver - Incompressible Abdominal Aorta Aneurysm (AAA)

Cores (avg. elems./core)	IBM BG/L RPI-CCNI	
	t (secs.)	scale factor
512 (204800)	2119.7	1 (base)
1024 (102400)	1052.4	1.01
2048 (51200)	529.1	1.00
4096 (25600)	267.0	0.99
8192 (12800)	130.5	1.02
16384 (6400)	64.5	1.03
32768 (3200)	35.6	0.93

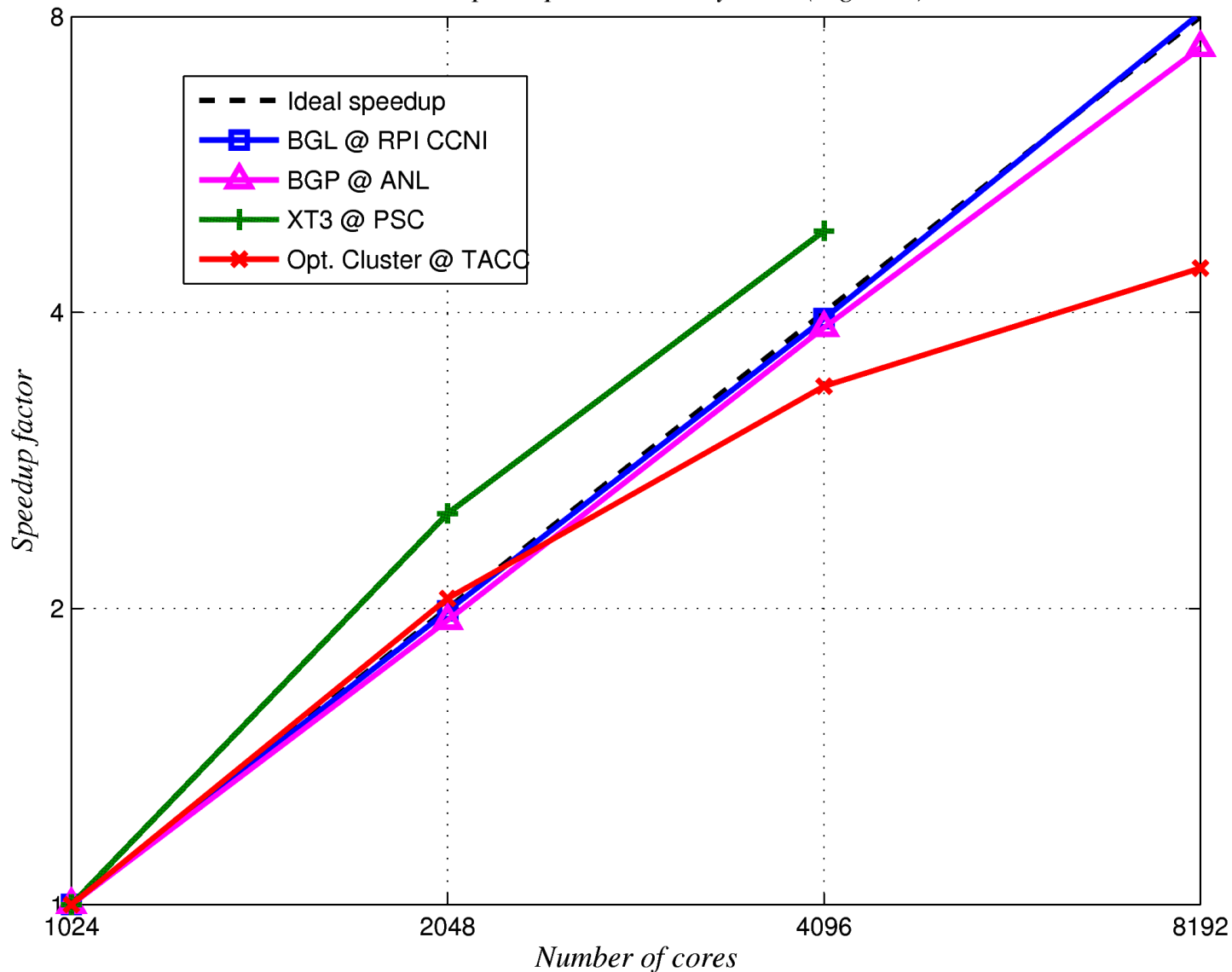


**32K parts shows modest degradation
due to 15% node imbalance**



Scaling of “AAA” 105M Case

PHASTA speedup on various systems (logscale)



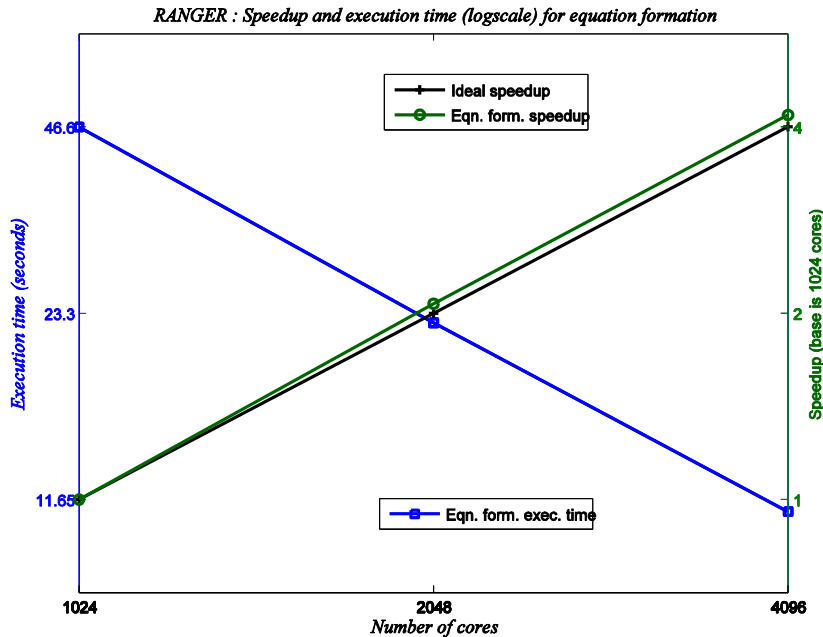
AAA Adapted to 10^9 Elements: Scaling on Blue Gene /P

#of cores	Rgn imb	Vtx imb	Time (s)	Scaling
32k	1.72%	8.11%	112.43	0.987
<i>128k</i>	<i>5.49%</i>	<i>17.85%</i>	<i>31.35</i>	<i>0.885</i>

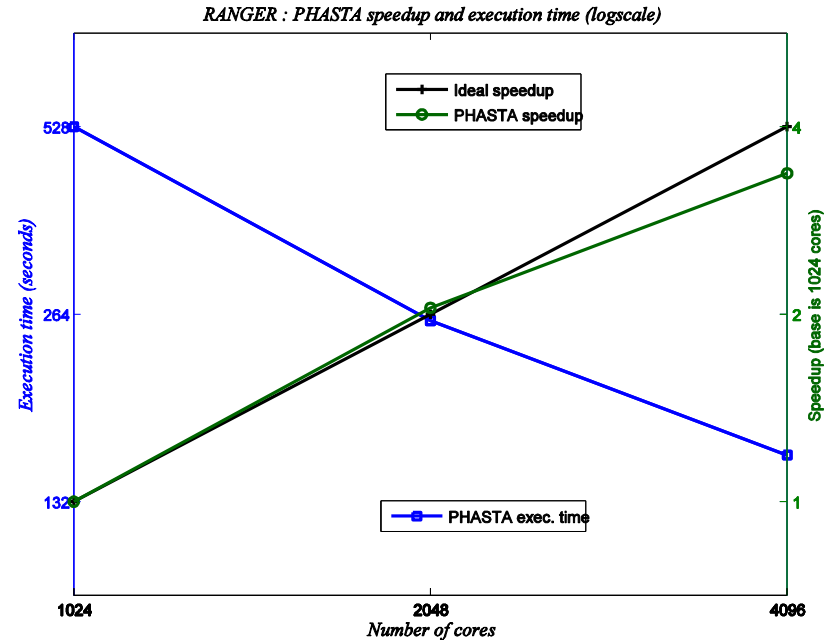
New: @ 294,912 cores → 85% scaling
relative to base of 16K cores

Work Analysis on Linux Cluster

Equation Formation



Implicit Solve



- Issue can be *avoided* by keeping element/node count high to shelter communication.
- Allows bigger problems to be solved but misses opportunity for time compression.

Flow Solver Parallel Paradigm (REVIEW)

- Time-accurate, stabilized FEM flow solver
- Two types of work:

- Equation formation

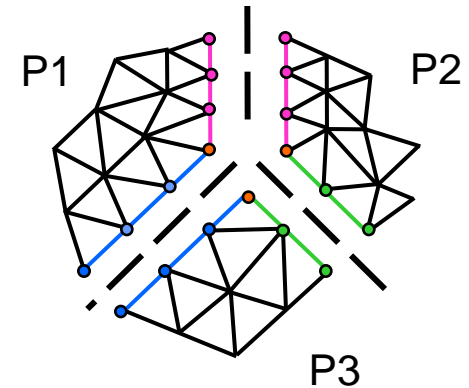
- $O(40)$ peer-to-peer non/blocking comms
- Overlapping comms with comp
- Scales well on many machines

- Implicit, iterative equation solution

- Each Krylov vector is:

- $q=Ap$ (matrix-vector product)
- Same peer-to-peer comm of q PLUS
- Orthogonalize against prior vectors
- **REQUIRES NORMS=>MPI_Allreduce**

- This sets up a cycle of global comms. separated by modest amount of work
- Not currently able to overlap Comms
- **Even if work is balanced perfectly, OS jitter can imbalance it.**
- **Imbalance WILL show up in MPI_Allreduce**
- Scales well on machines with low noise (like Blue Gene)



OS Noise Test Problem

- Test code created to mimic the problem
- For iter=1..LARGE-NUMBER
 - do 1 million MADDs (WORK)
 - MPI_Allreduce (COMM)
- Endfor

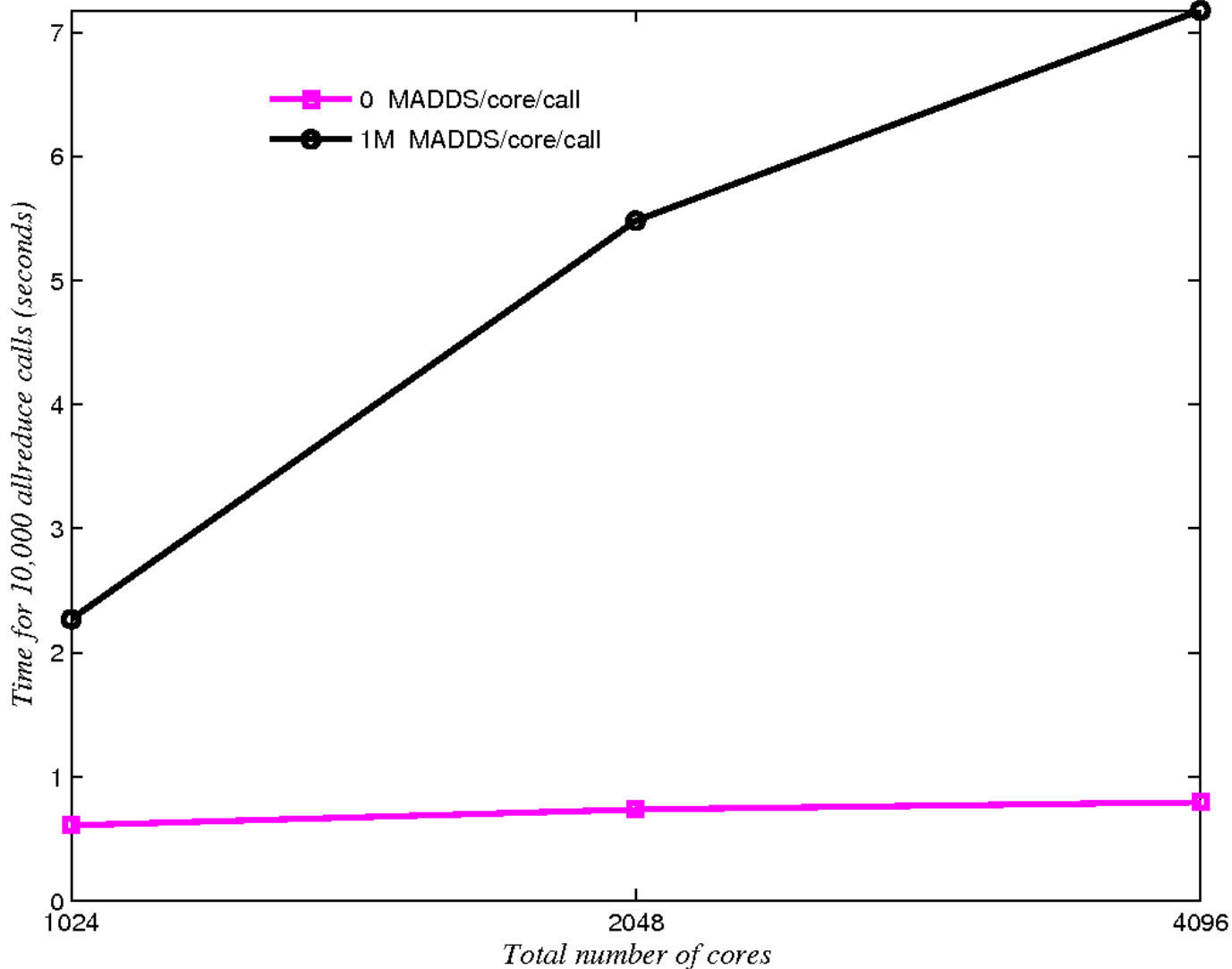
- Timers can be wrapped around WORK or COMM

Observe that REAL work is being done and not just a fixed timer amount of fake work.


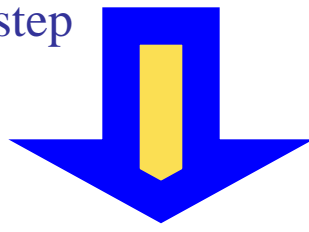
Care needs to be taken to avoid compiler optimizing the work loop away...

How OS Jitter Effects MPI Allreduce

RANGER : Time for 10,000 allreduce calls with and without FLOPs in between



Phasta Summary...

- Complex geometry/physics=> Real world Apps
- Implicit solvers: Complexity  but n_{step} 
- Excellent scaling results
- Understanding influence of OS noise
- Big Science AND FAST SCIENCE
- Methods used (FEM) representative of commercial Computer Aided Engineering (CAE) software
- Adaptivity brings real geometry problems into reach of solution in a USEFUL time frame
- Activity on viz co-processing with Kitware (ParaView)
- Approaching the era where time compression is sufficient to enable experiential fluid dynamic design: where “domain experts” can visually interact/iterate a design and experience fluid dynamic response to each tweak

DES Ex: Movies over the Internet

- Suppose we want to model 1 million home ISP customers downloading a 2 GB movie
- How long to compute?
 - Assume a nominal 100K ev/sec seq. simulator
 - Assume on avg. each packet takes 8 hops
 - 2GB movies yields 2 trillion 1K data packets.
 - @ 8 hops yields 16+ trillion events

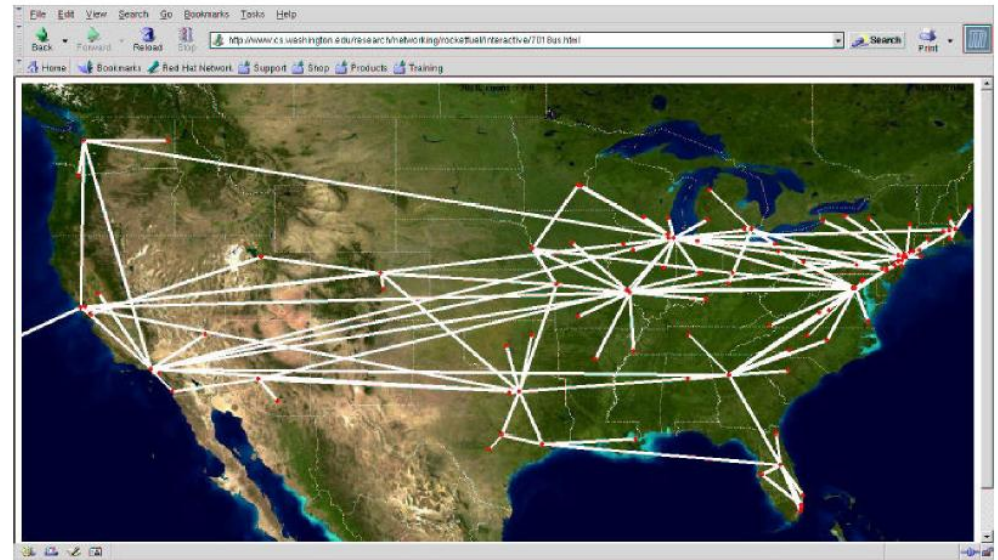
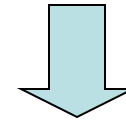


Fig. 5. AT&T Network Topology (AS 7118) from the Rocketfuel data bank for the continental US.

- 16+ trillion events @ 100K ev/sec

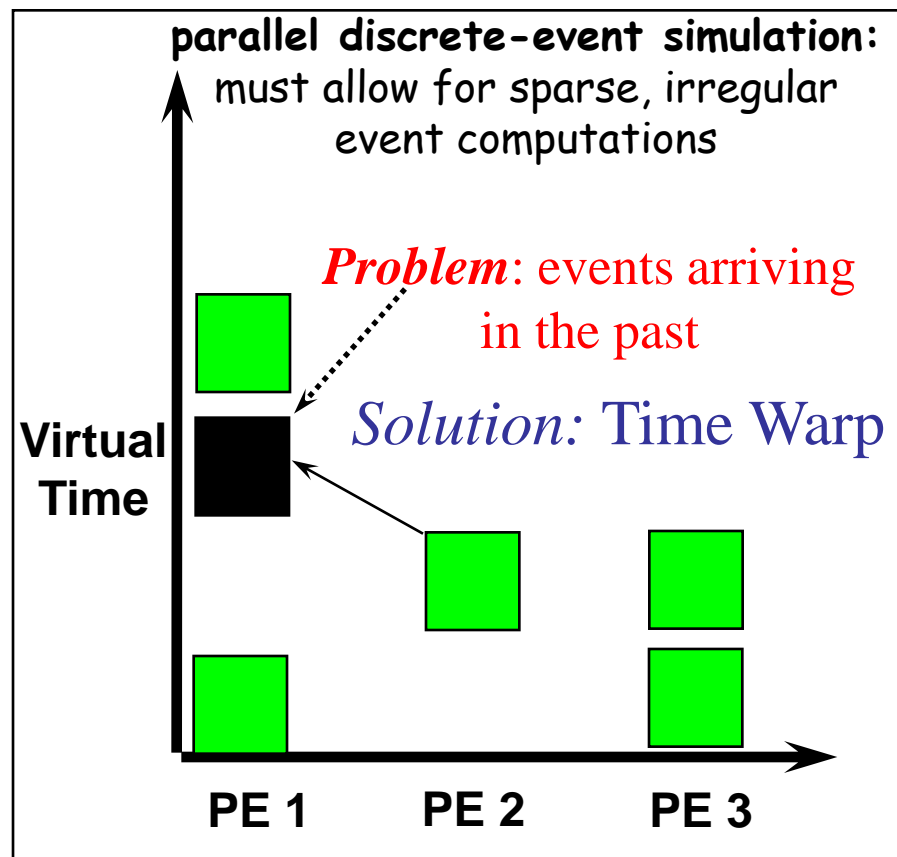
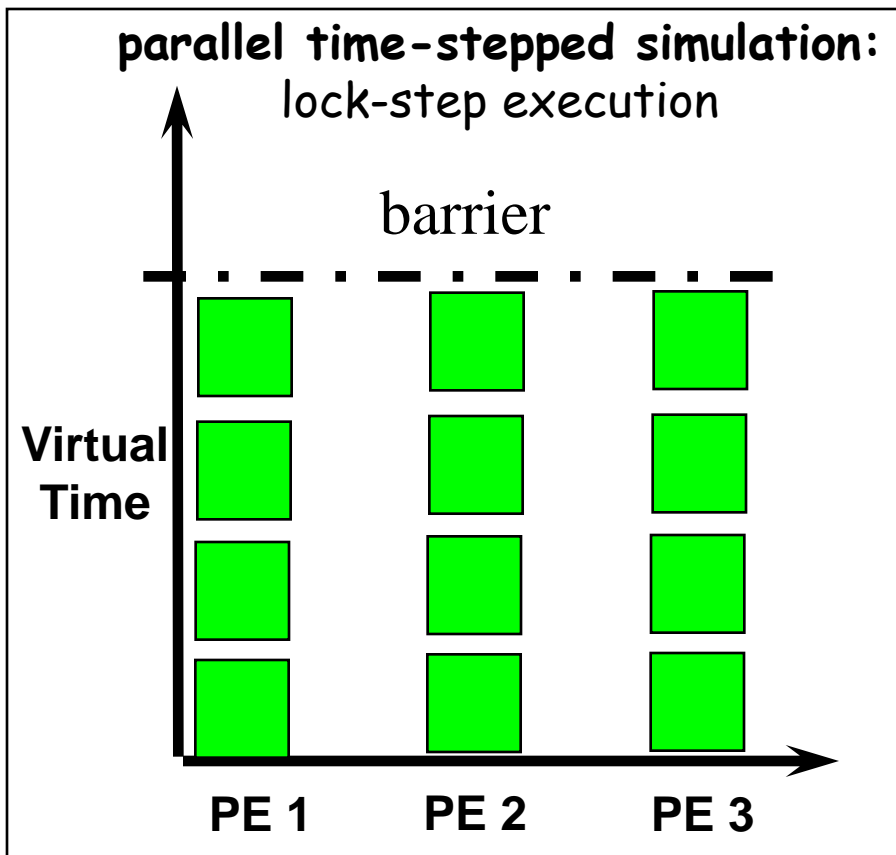


Over 1,900 days!!! Or
5+ years!!!

Need massively parallel simulation
to make tractable



How to Synchronize Parallel Simulations?



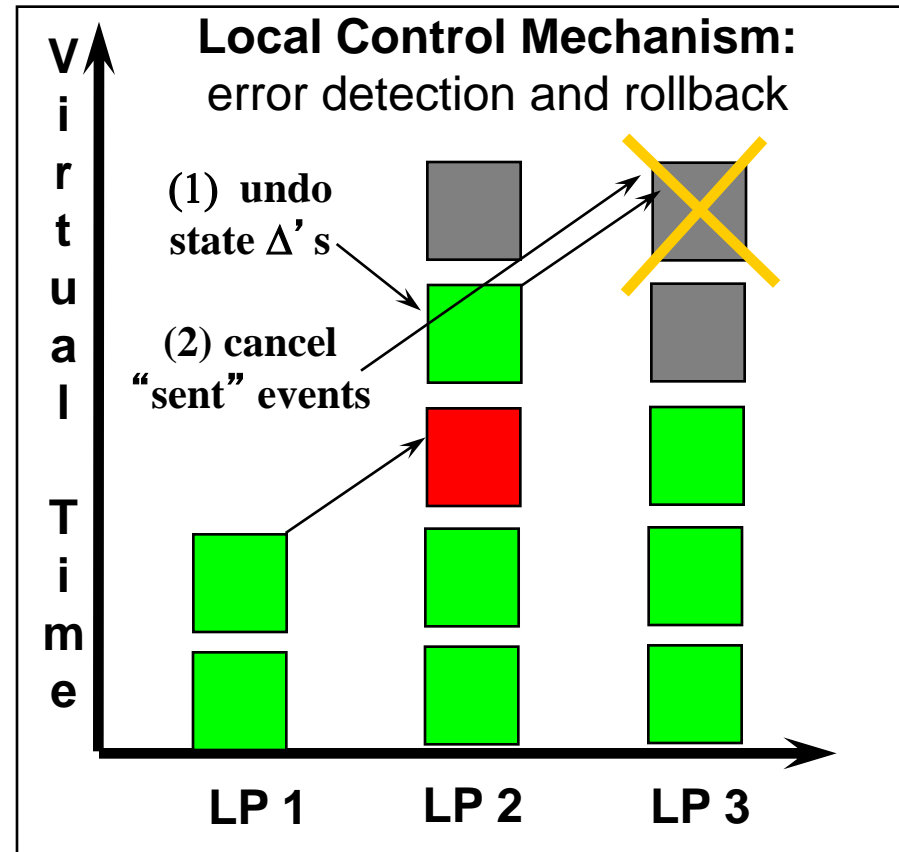
 processed event

 "straggler" event



Local Control Implementation

- **MPI_IRecv/MPI_Irecv** used to send/recv off core events
- Event & Network memory is managed directly.
 - Pool is allocated @ startup
- Event list keep sorted using a Splay Tree ($\log N$)
- LP-2-Core mapping tables are computed and not stored to avoid the need for large global LP maps.



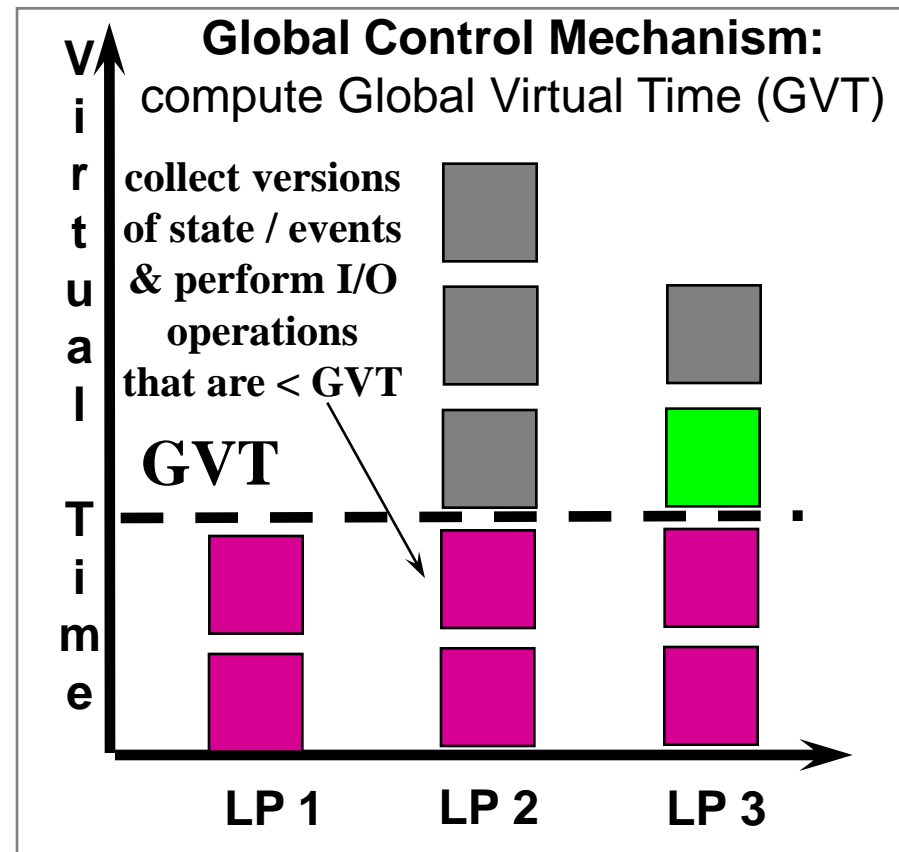
Global Control Implementation

GVT (kicks off when memory is low):

1. Each core counts $\#sent$, $\#recv$
2. Recv all pending MPI msgs.
3. MPI_Allreduce Sum on ($\#sent - \#recv$)
4. If $\#sent - \#recv \neq 0$ goto 2
5. Compute local core's lower bound time-stamp (LVT).
6. $GVT = \text{MPI_Allreduce Min on LVTs}$

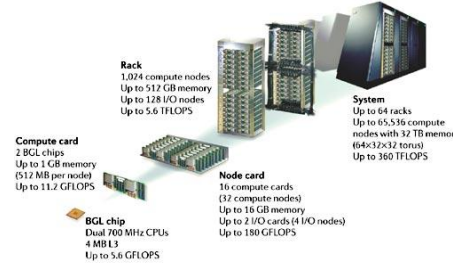
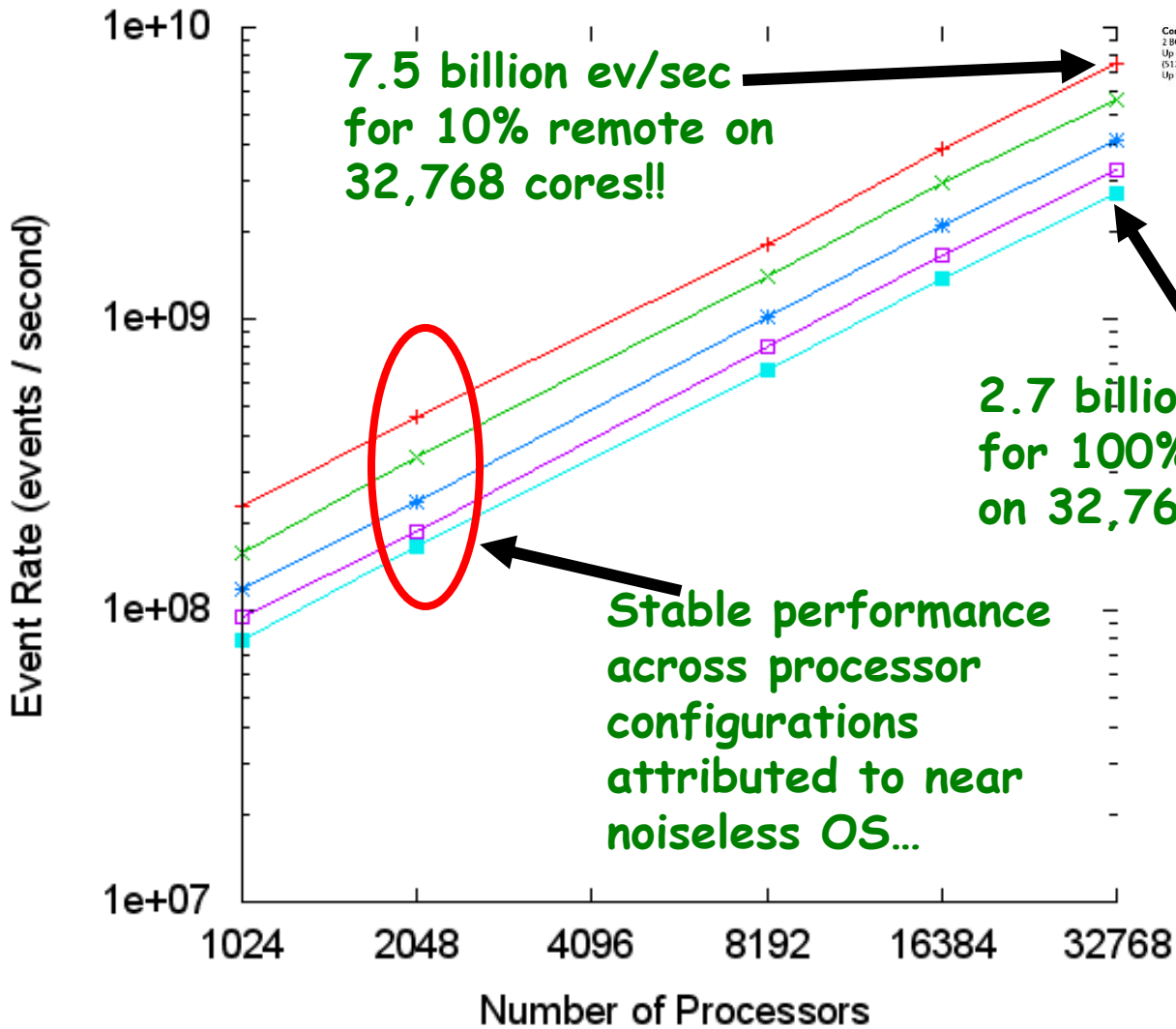
Algorithms needs efficient MPI collective

LC/GC can be very sensitive to OS jitter



So, how does this translate into Time Warp performance on BG/L & BG/P?

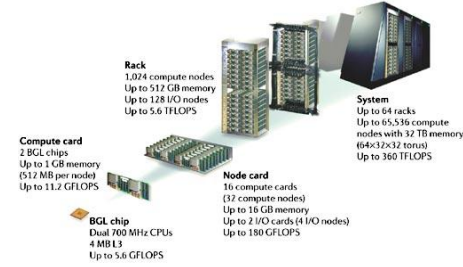
Blue Gene/L: Time Warp Scalability



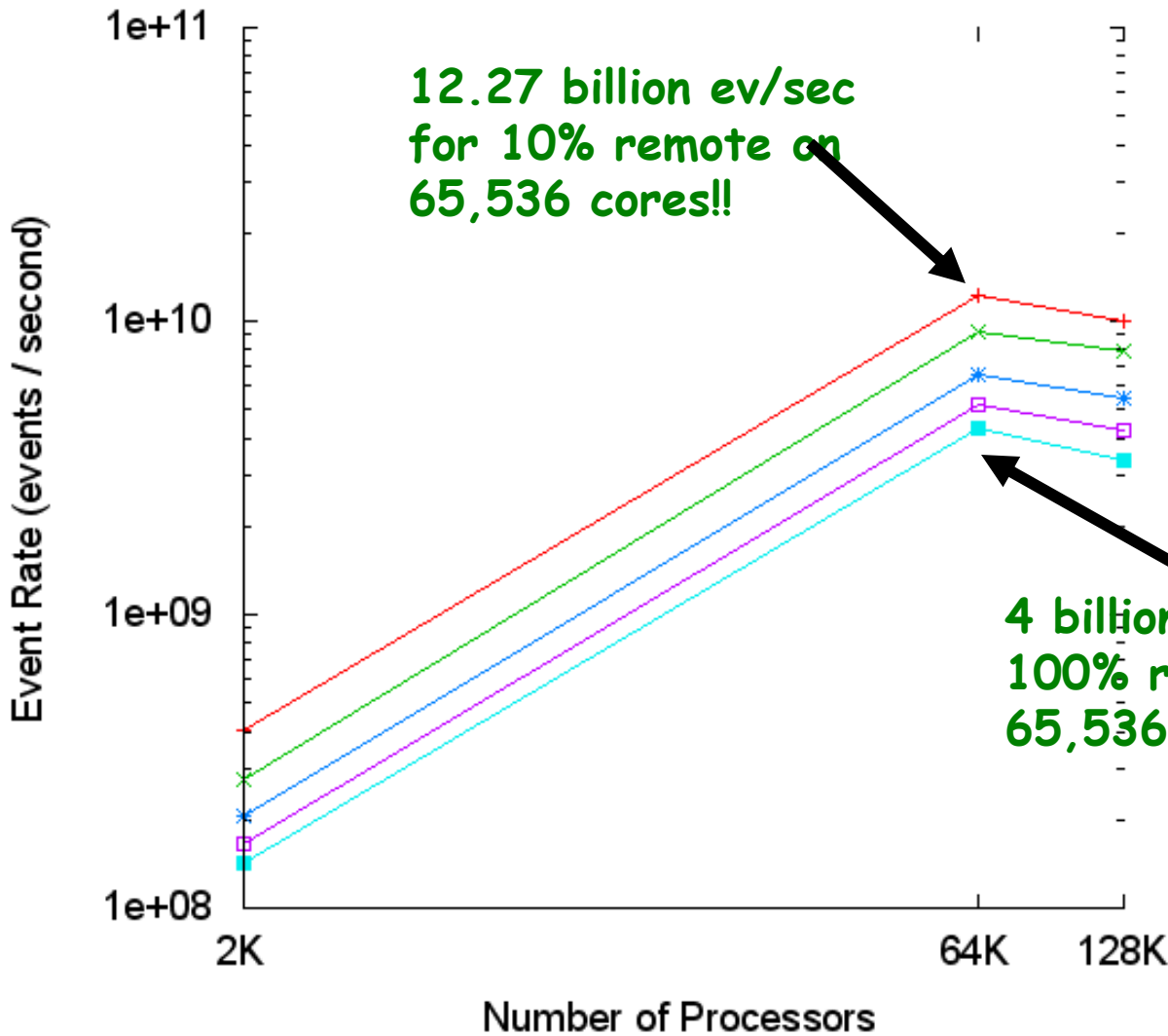
Copyright © 2006 Nature Publishing Group
Nature Reviews | Neuroscience

- +— 10% Remote
- *— 50% Remote
- 100% Remote
- x— 25% Remote
- 75% Remote

Blue Gene/P: Time Warp Scalability



Copyright © 2006 Nature Publishing Group
Nature Reviews | Neuroscience



12.27 billion ev/sec for 10% remote on 65,536 cores!!

4 billion ev/sec for 100% remote on 65,536 cores!!

—+—	10% Remote	—*—	50% Remote	—■—	100% Remote
—x—	25% Remote	—□—	75% Remote		

Movies over the Internet Revisited

- Suppose we want to model 1 million home ISP customers over AT&T downloading a 2 GB movie
- How long to compute with massively parallel DES?

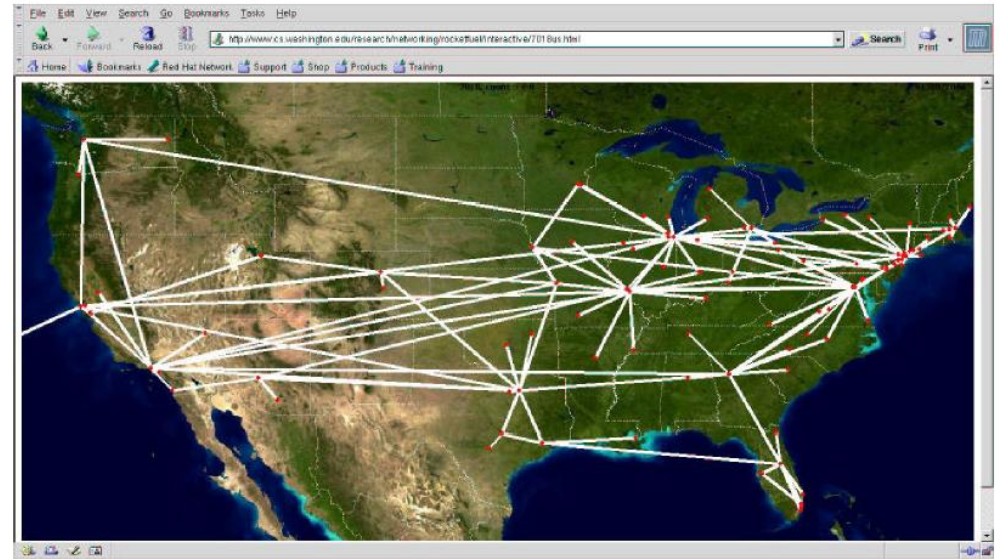


Fig. 5. AT&T Network Topology (AS 7118) from the Rocketfuel data bank for the continental US.

- 16+ trillion events @ 1 Billion ev/sec ...

~4.5 hours!!

Summary

- *Significant opportunities for using massively parallel computing systems in robotics system's research*
- *GPUs/CUDA*
 - *PRO: massive amount of compute power per \$\$*
 - *PRO: very available*
 - *CONS: very hard to program*
 - *CONS: Problem sizes limited by GPU memory space*
- *Supercomputer Systems*
 - *PRO: Growth in size (1.6M cores in 2012)*
 - *PRO: Relatively easier to program (student developer time less than GPU typically).*
 - *CON: batch queuing of jobs leads to longer wait times than desktop*
 - *CON: CPU hours limited, need proposals and win time!*