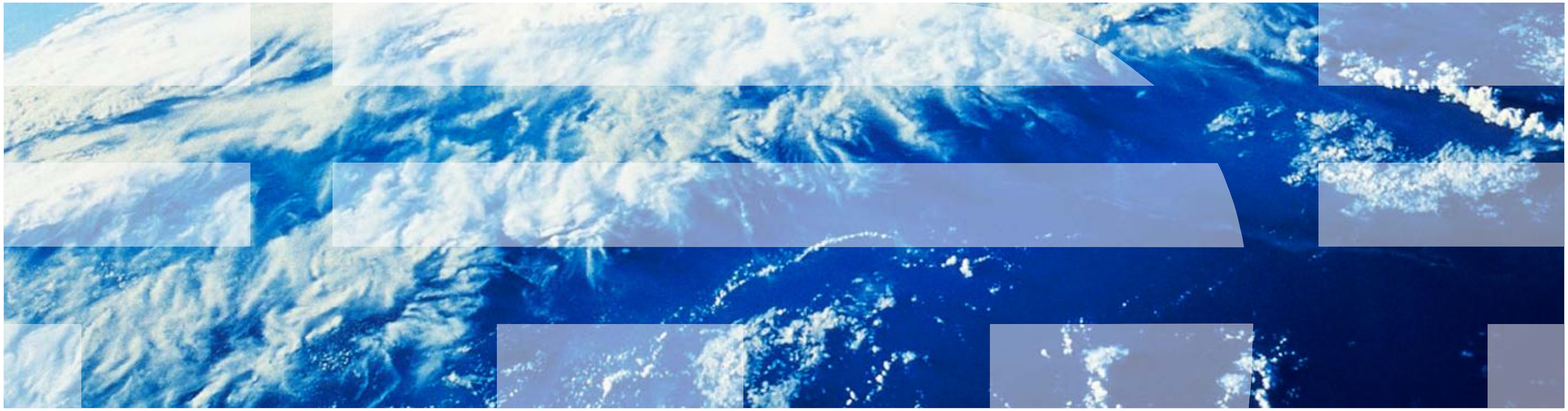
Towards Massively Parallel Simulations of Massively Parallel High-Performance Computing Systems

Robert Birke, German Rodriguez, Cyriel Minkenberg
IBM Research — Zurich

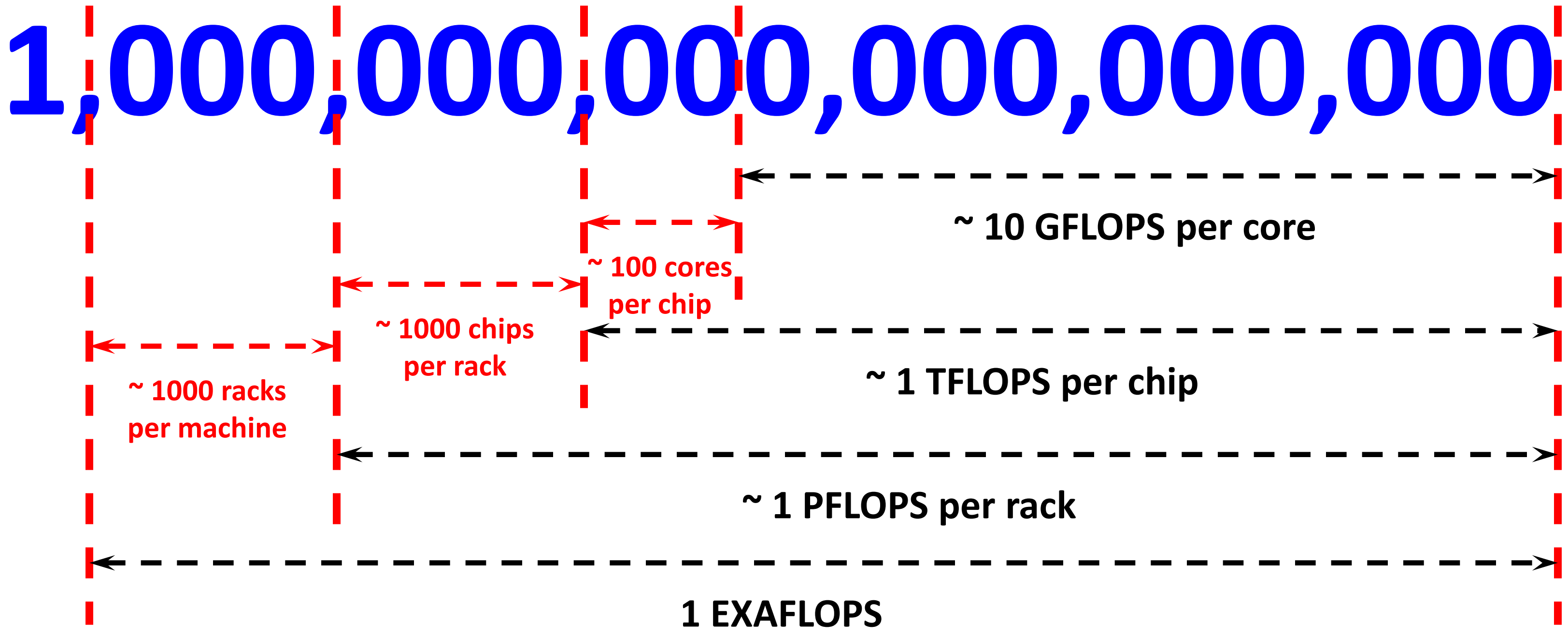


Outline

- **High-performance computing: Road to exascale**
- **Role of the interconnection network**
- **Workload-centric simulation of HPC systems**
  - Performance prediction of benchmarks & applications
  - Impact of communication subsystem on application performance: Cost-performance optimization
- **Tool chain: Instrumentation, simulation, visualization**
- **Parallelization of our network simulator**
- **Porting Omnest & Venus to IBM Blue Gene**
- **Results**
- **Conclusions**

Towards exascale computing

- Current #1 on Top 500: 10 Petaflop/s
- 1 Exaflop = 10^{18} floating point operations per second
- Timeframe: 2018



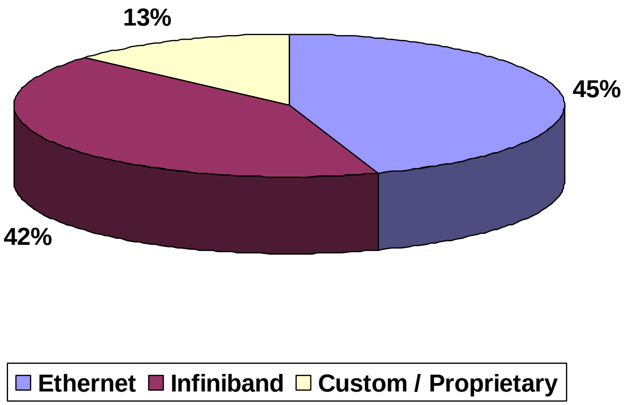
A few examples of current HPC systems

				Actual [www.top500.org]	Theoretical
	GFLOPS /core	Cores/chip GFLOPs/chip	Chips/rack Cores/rack TFLOPS/rack	#racks Cores/machine TFLOPS/machine [peak]	Max. #racks Cores/machine TFLOPS/machine [peak]
Fujitsu K Computer	16	SPARC64 VIIIfx 8 128	102 816 13	RIKEN 12.7 MW 864 705’024 11’280	
Cray XT5-HE Jaguar	10.4	AMD Opteron 6 62.4	192 1’152 12	ORNL 7 MW 194 224’256 2’331	
IBM BG/L	2.7	PowerPC 440 2 5.5	1’024 2’048 5.6	LLNL 2.3 MW 104 212’992 596	128 262’144 717
IBM BG/P	3.4	PowerPC 450 4 13.6	1’024 4’096 13.9	Jülich 2.3 MW 72 294’912 1’003	256 1’048’576 3’558
IBM BG/Q Prototype	12.8	PowerPC A2 16 204.8	1’024 16’384 209.7	TJ Watson 39 kW 8 8’192 105	512 8’388’608 107’366
IBM p775 (PERCS)	31.2	POWER7 8 249.6	384 3’072 96	No top500 entry yet	170 524’288 16’320

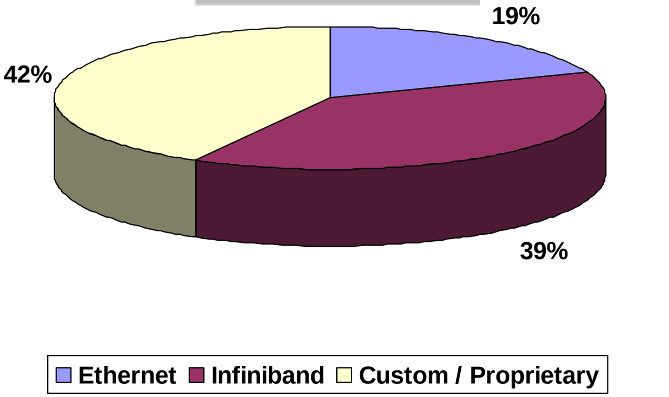
Top 500 Interconnects

- **List of November 2011**
- **Dominated by Ethernet and Infiniband**
  - Ethernet by volume
  - Infiniband by FLOPS
- **Proprietary still plays a significant role**
  - Cray XT (Seastar) / XE (Gemini)
  - IBM Blue Gene
  - IBM p775
  - Myrinet, Quadrics
- **10 GigE: 14 installations**

Number of systems



FLOPS (Rmax)



Source: www.top500.org

HPC interconnect trends

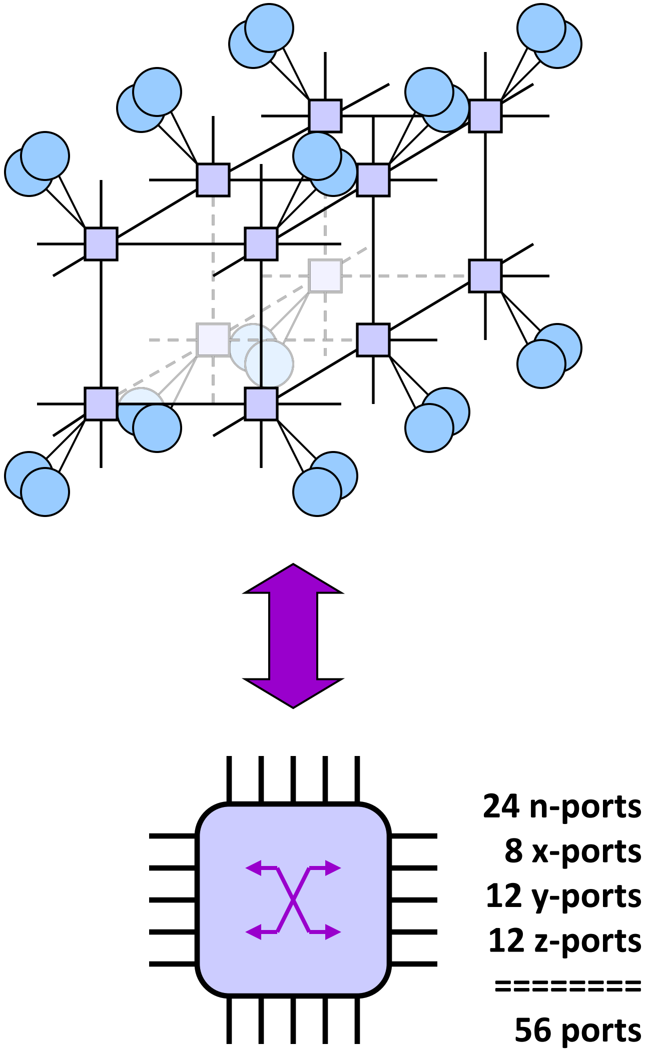
- **The latency-optimum switch radix is increasing**
  - Aggregate switch bandwidth for a given technology is fixed
  - Should it be divided among many narrow or few wide ports?
    - Serialization latency vs. hop count
  - As aggregate bandwidth increases, optimum switch radix also increases
  - Large radix switches are indeed becoming available

- **Moving away from typical 2D/3D mesh and torus topologies**
  - Require low-radix switches
  - Too many hops: large worst-case latency
  - Low bisection bandwidth

- **Attractive topologies**
  - High-radix fat trees; “slim” trees
  - “Concentrated” k-ary n-cubes
  - High-dimensional k-ary n-cubes
  - Dragonflies

- **Network-assisted acceleration of collectives**

- **Very-high-end systems use tightly integrated custom networks**
  - IBM BlueGene
  - IBM PERCS
  - Cray XE6 (“Baker”)

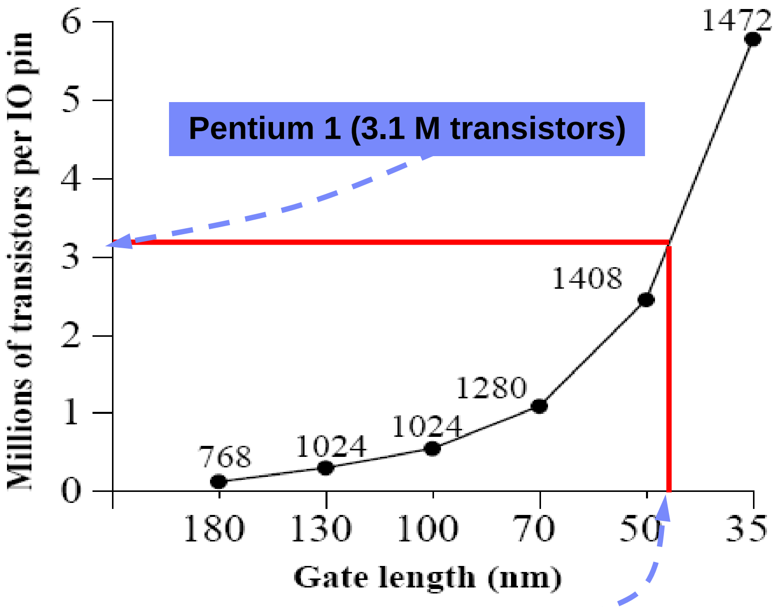


24 n-ports
8 x-ports
12 y-ports
12 z-ports
========
56 ports

The interconnect – no longer 2nd class citizen?

- **Conventional wisdom**
  - – Computation = expensive
  - – Communication = cheap
  - ➔Corollary: Processor is king

- **Then something happened…**
  - – Computation
    - Transistors became “free” ➔ more parallelism: superscalar, SMT, multi-core, many-core
    - Huge increase in FLOPS/chip
  - – Communication
    - Packaging & pins remained expensive
    - Scaling of per-pin bandwidth did not keep pace with CMOS density
  - ➔Consequence
    - Comp/comm cost ratio has changed fundamentally
    - Memory and I/O bandwidth now an even scarcer resource



Pentium 1 (3.1 M transistors)

For the 45nm node, all I/O and power & ground connections of an Pentium1-equivalent chip will have to be served by ONE package pin!

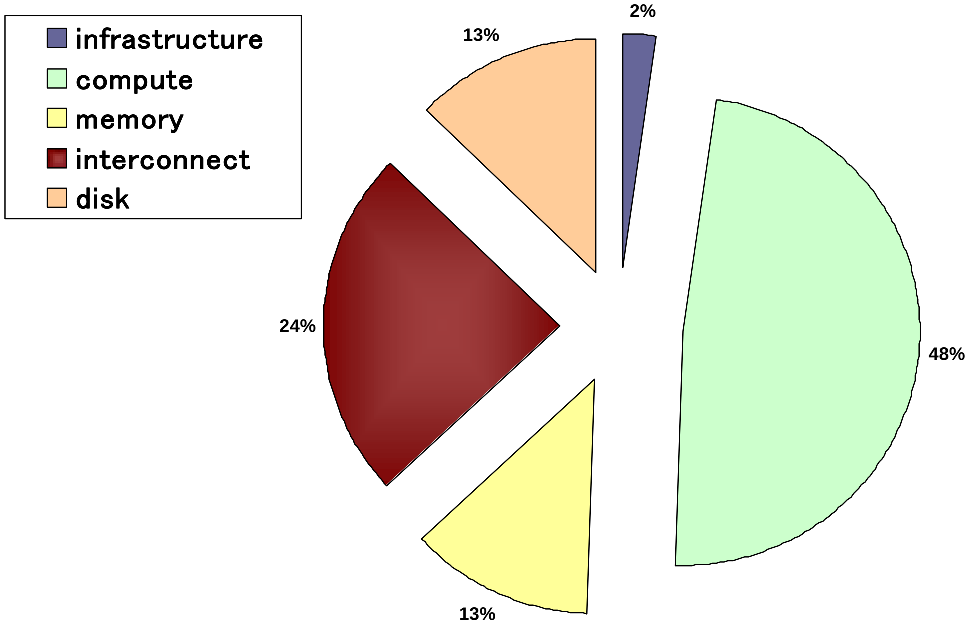
Source: Intel & ITRS

Consequences of scarce bandwidth

- **Performance of communication-intensive applications has scaled poorly**
  - because of lower *global* byte/FLOP ratio
- **Yet *mean* utilization is typically very low**
  - because of synchronous nature of many HPC codes; regularly alternating comp/comm phases
  - massive underutilization for computation-intensive applications (e.g. LINPACK)
- **Full-bisection bandwidth networks are no longer cost-effective**
- **Common practice of separate networks for clustering, storage, LAN has become inefficient and expensive**
  - File I/O and IPC can’t share bandwidth
    - I/O-dominated initialization phase could be much faster if it could exploit clustering bandwidth: poor speedup, or even slowdown with more tasks…

Interconnect becoming a significant cost factor

- Current interconnect cost percentage increases as cluster size increases
- About one quarter of cost due to interconnect for ~1 PFLOP/s peak system



	Fat Tree	*Torus 1*	*Torus 2*	*Torus 3*
Compute	**63.3%**	**72.5%**	**76.5%**	**79.7%**
Adapters + cable	**10.4%**	**11.9%**	**12.6%**	**13.1%**
Switches + cables	**26.3%**	**15.6%**	**10.9%**	**7.2%**
Total	**100%**	**100%**	**100%**	**100%**

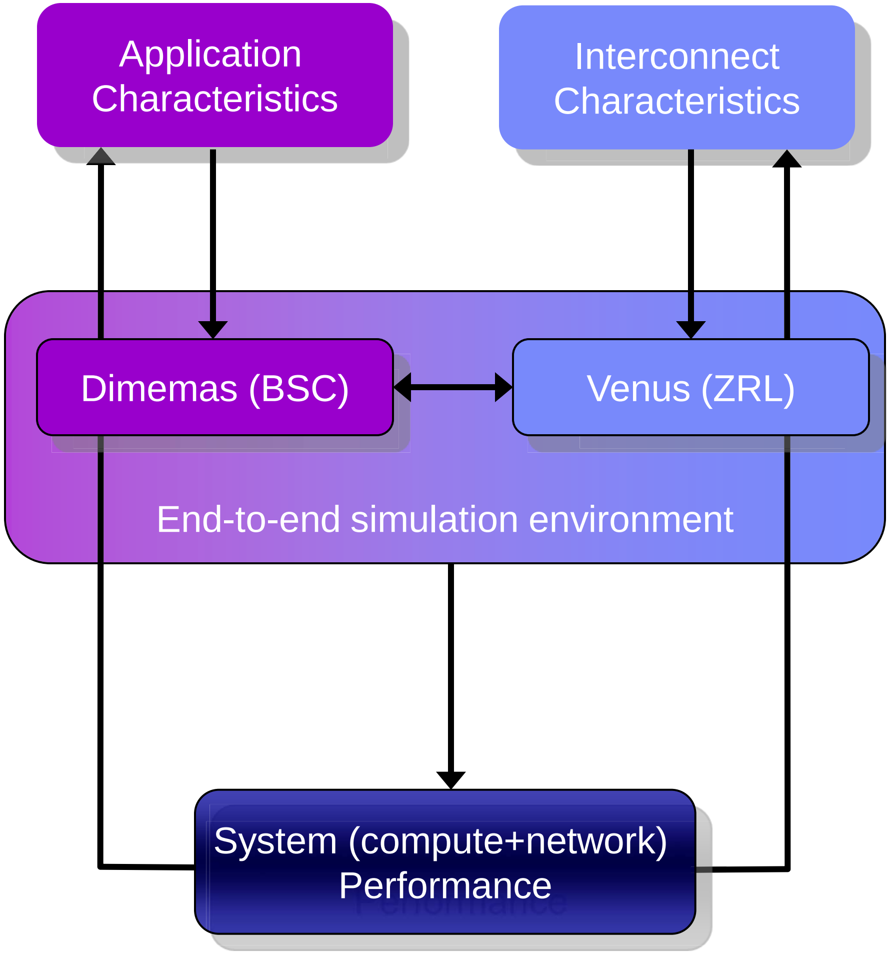
Modeling of large-scale HPC systems

- **Dichotomy in performance evaluation in computer architecture**
  - Node architecture
    - Execution-driven, cycle-accurate CPU (ISA), cache and memory models
    - Highly accurate
    - Too much detail to scale to large node counts
  - Network architecture
    - Highly accurate (flit level) at network level
    - Several orders of magnitude fewer network nodes than CPU cores
    - Network node much simpler than CPU core
    - But usually driven by either purely random or purely deterministic and **non-reactive** traffic patterns

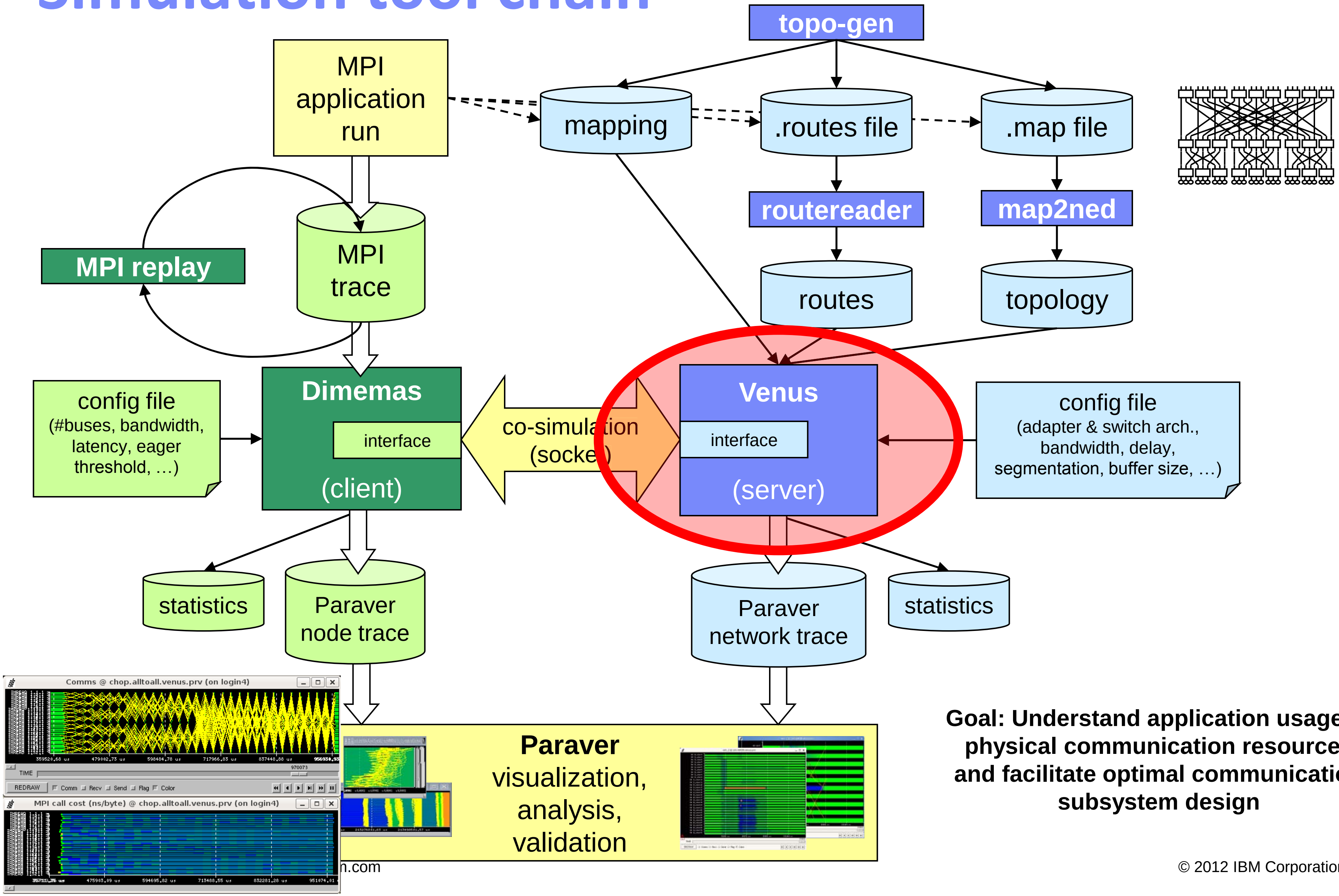
- **Need to adopt a holistic approach, taking into account application, node architecture, and network architecture**
  - Given the scale of current and future systems, parallel simulation is the only way to manage simulation run time and memory footprint

Application-level performance prediction

1. **Instrument applications to collect computation, communication and their inter-dependencies**
   - For apps or benchmarks of interest
2. **Collect traces on a production system**
   - e.g., BG/P, MareNostrum
3. **Perform full-system trace-driven simulations with Dimemas+Venus**
   - Tune model parameters to match reality
   - Perform parameter sensitivity studies
     - Network technology
     - Network topology
     - Routing, etc…
4. **Optimize**
   - Interconnect: e.g. performance/$
   - Application: e.g. communication scheduling


Application Characteristics
Interconnect Characteristics
Dimemas (BSC)
Venus (ZRL)
End-to-end simulation environment
System (compute+network) Performance

Simulation tool chain



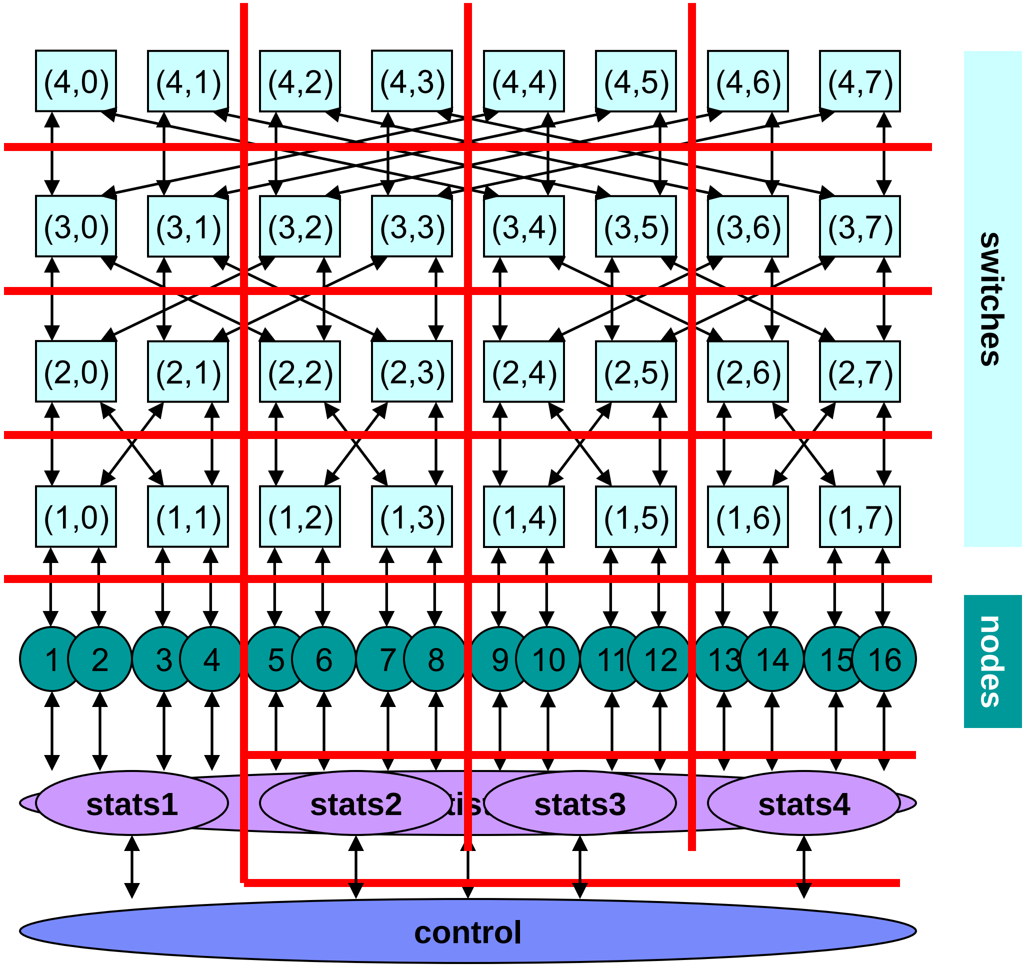
Goal: Understand application usage of physical communication resources and facilitate optimal communication subsystem design

Parallelizing our model

- **Obey the four commandments**
  - Thou shalt not use global variables
  - Thou shalt not invoke thy neighbor’s methods directly
  - Thou shalt not use dynamic topologies
  - Thou shalt provide sufficient lookahead
- **Message packing/unpacking**
  - Auto-generated by msgc in most cases
  - Customized or hand-coded message classes require explicit implementation
- **Partitioning: Omnest fixes**
  - Partitioning compound modules: Proxy gates
  - Lookahead between unconnected partitions
  - Partition assignment in ini file: Compound module must be assigned union set of all partitions in which any of its submodules reside
- **MPI buffer size**
  - Need to assign large MPI buffer; appears to scale with square of #partitions
  - We believe this can be optimized
- **Debugging parallel efficiency**
  - One major issue (see next slide)

Topology partitioning: Fat tree example

- **Option 1: Non-homogenous partitions; load balancing issues**
- **Option 2: Better homogeneity, but statistics module is a bottleneck**
- **Option 3: Split statistics module into one per partition**



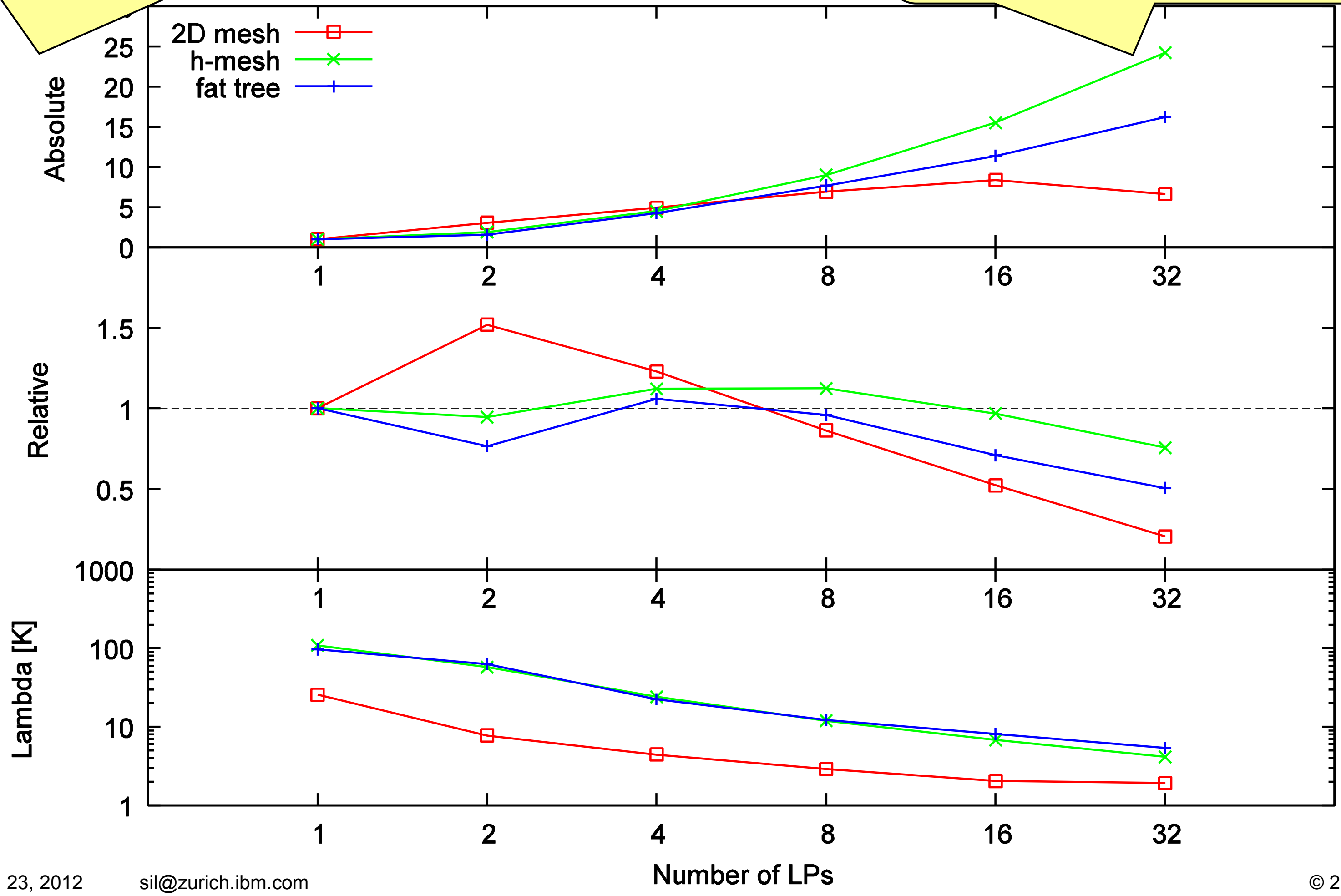
Simulated topologies

	2D mesh	3D mesh	Fat tree	Hierarchical mesh
Arity	k = 64	k = 16	k = 8	$k_{1,2,3}$ = 16, 8, 8
Degree	n = 2	n = 3	n = 4	n = 3
Bristling	p = 1	p = 1	p = 1	p = 4
#end nodes	4,096	4,096	4,096	4,096
#switches	4,096	4,096	4,096	1,024
#links	10,240	14,336	32,768	16,986
Switch radix	5	7	16	33
Diameter	126	45	6	3

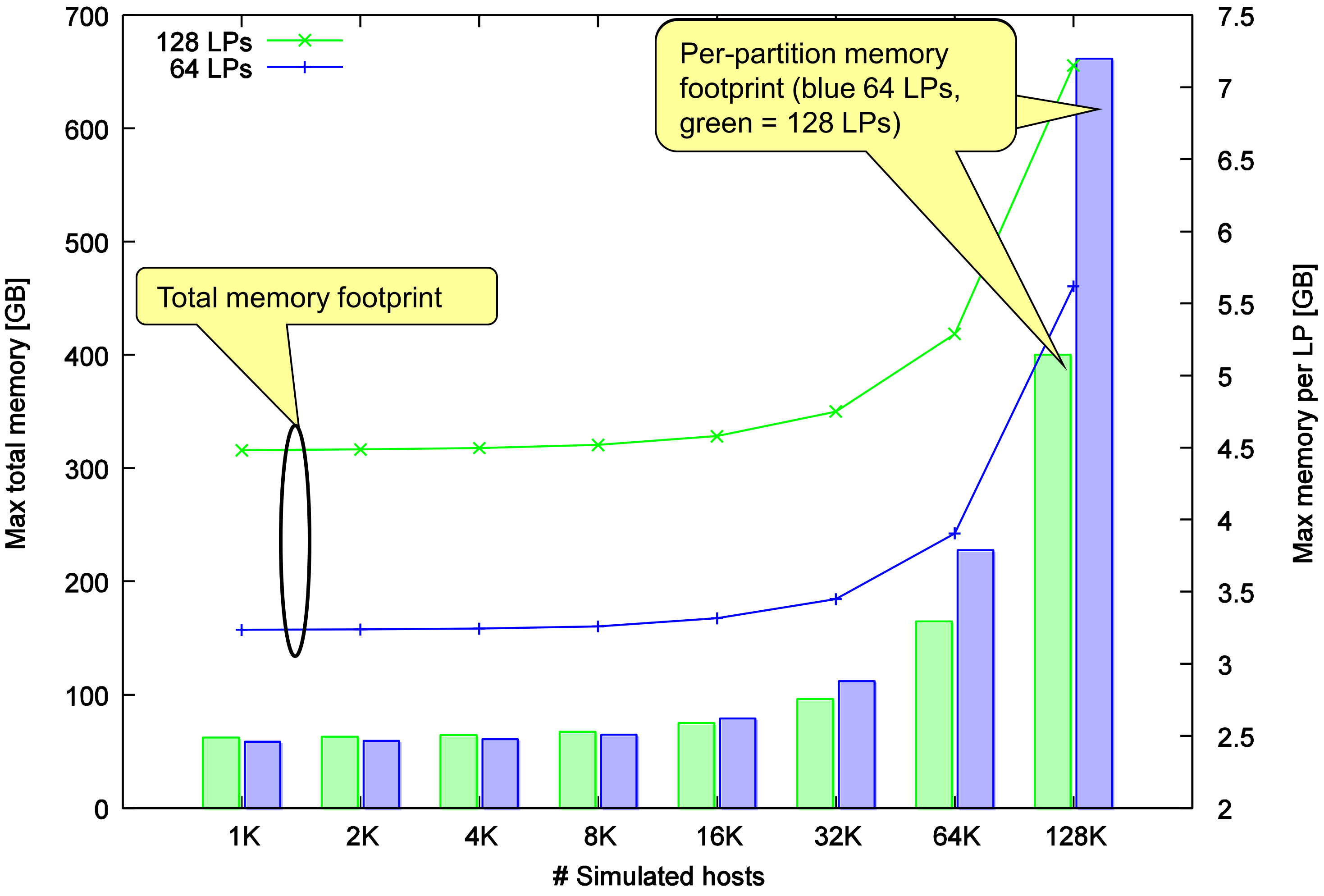
Speedup results on 32-core SMP 768 GB RAM

- Absolute speedup A(*N*) = T(*N*) / T(1)
- Relative speedup R(*N*) = A(*N*) / *N*
- Lambda = (*L* * *E*) / (τ * *P*)

$A_{mesh}(N) < A_{fat\text{-}tree}(N) < A_{h\text{-}mesh}(N)$.
Speedup has strong inverse correlation with network diameter: more hops implies more partition-boundary crossings!



Memory footprint



Per-partition memory footprint (blue 64 LPs, green = 128 LPs)

Total memory footprint

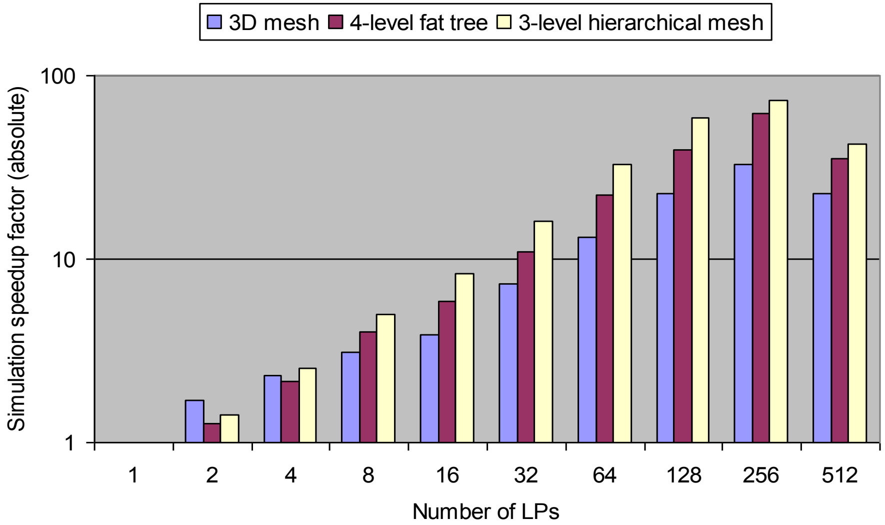
Porting Omnest and Venus to Blue Gene

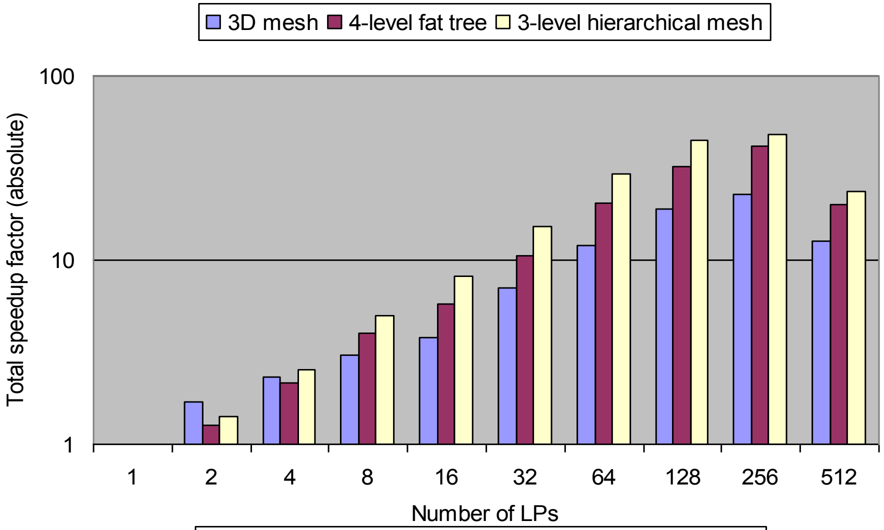
- **Porting for gcc as well IBM xlc**
  - Overall, porting was fairly smooth
  - xlc generally pickier than gcc
  - Once Omnest was ported, model porting was straightforward

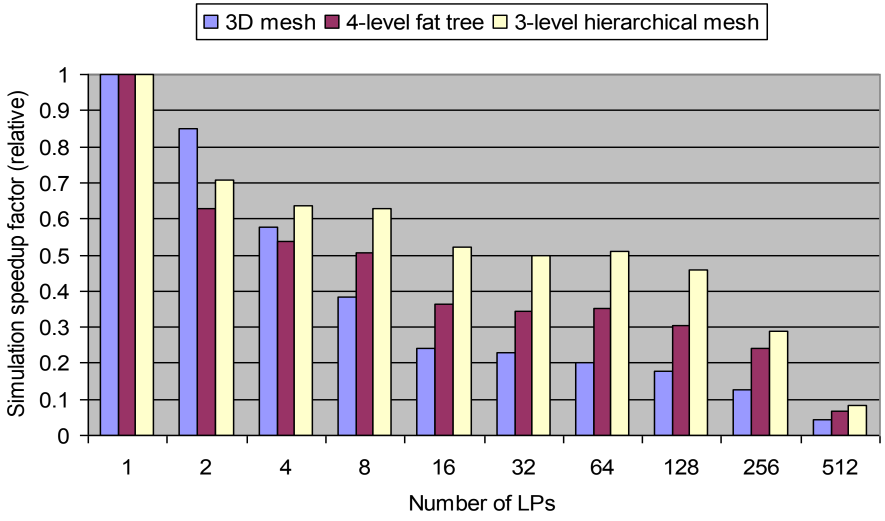
- **Mostly nitty-gritty details**
  - Please refer to the paper
  - (or get in touch with us)

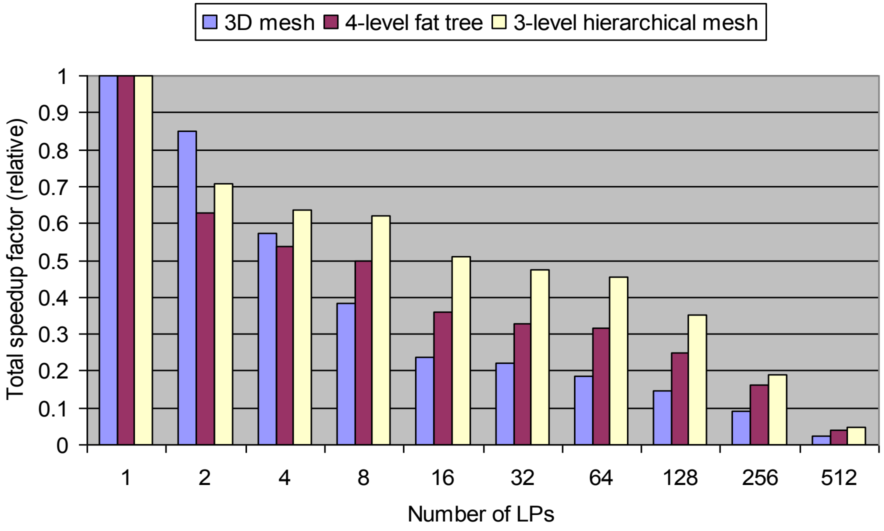
- **One nasty issue with xlc related C++ name mangling**
  - Obscure, hard-to-debug crash
  - …with an almost trivial solution

Parallel efficiency on BG/P

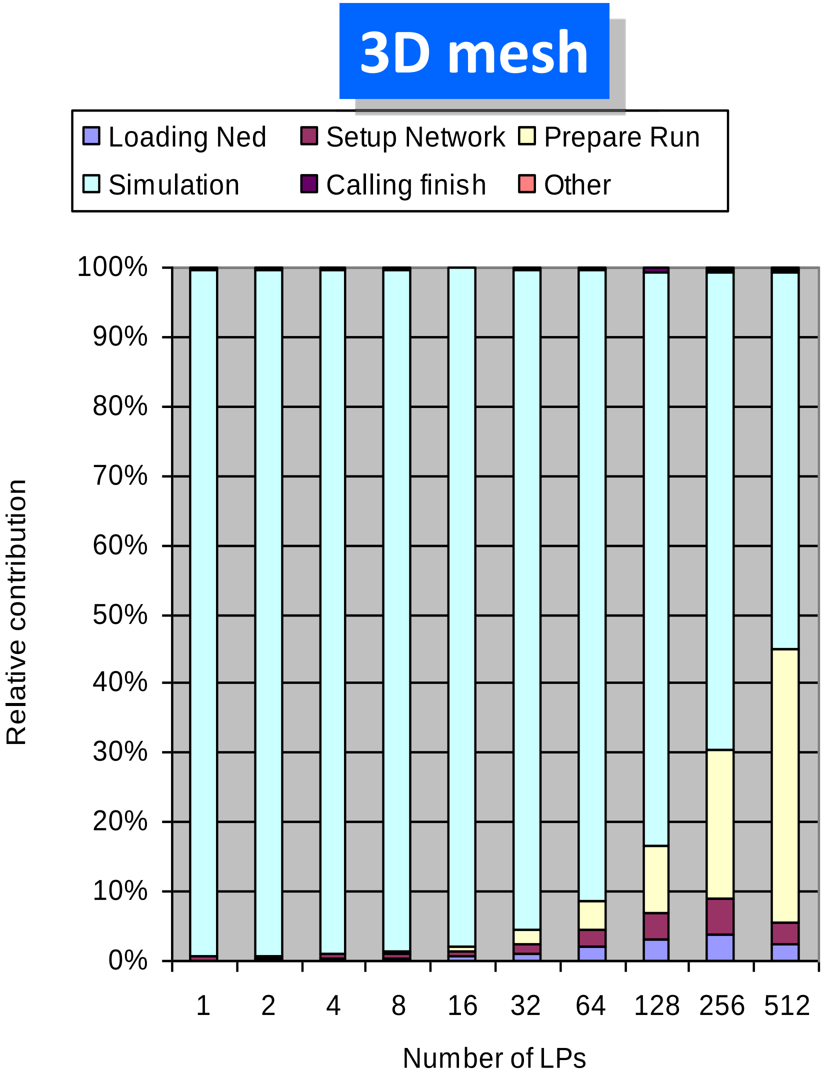


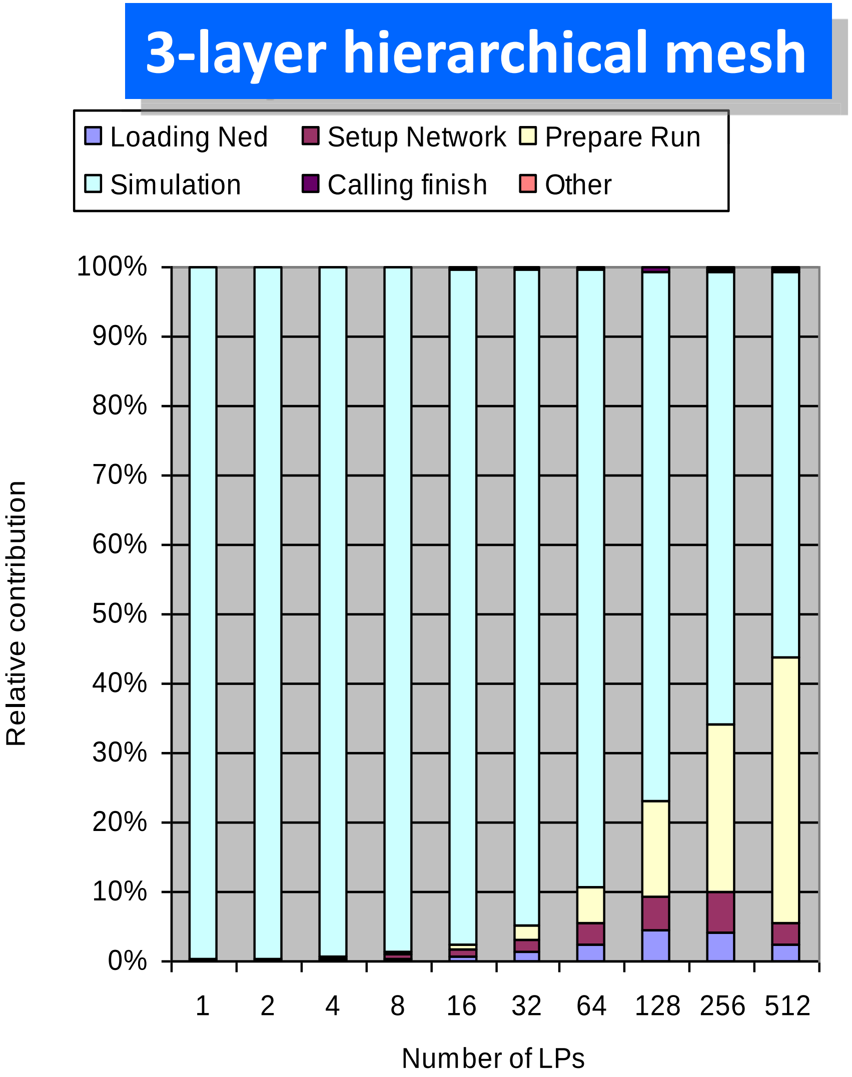




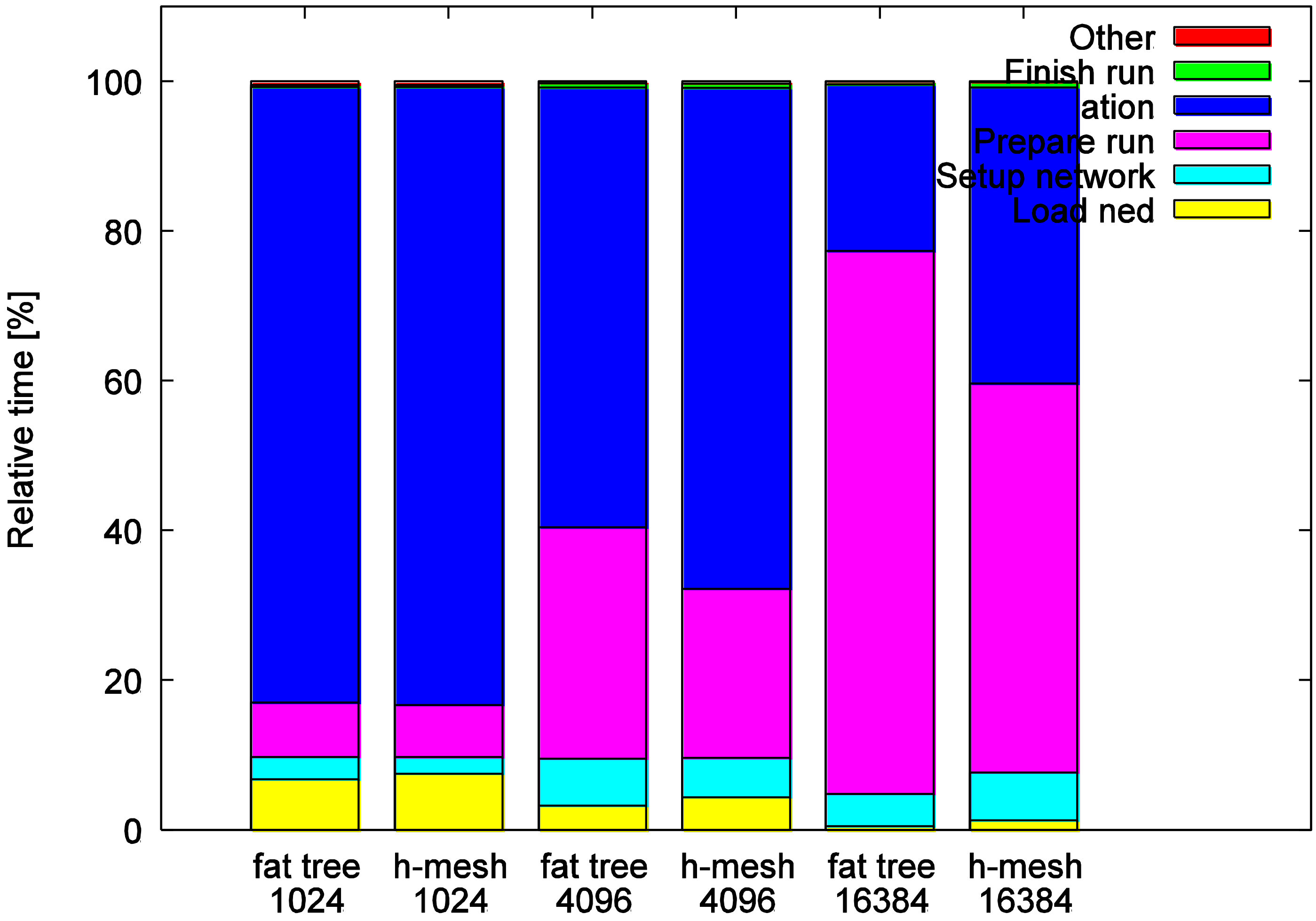


Relative runtime contributions (1)

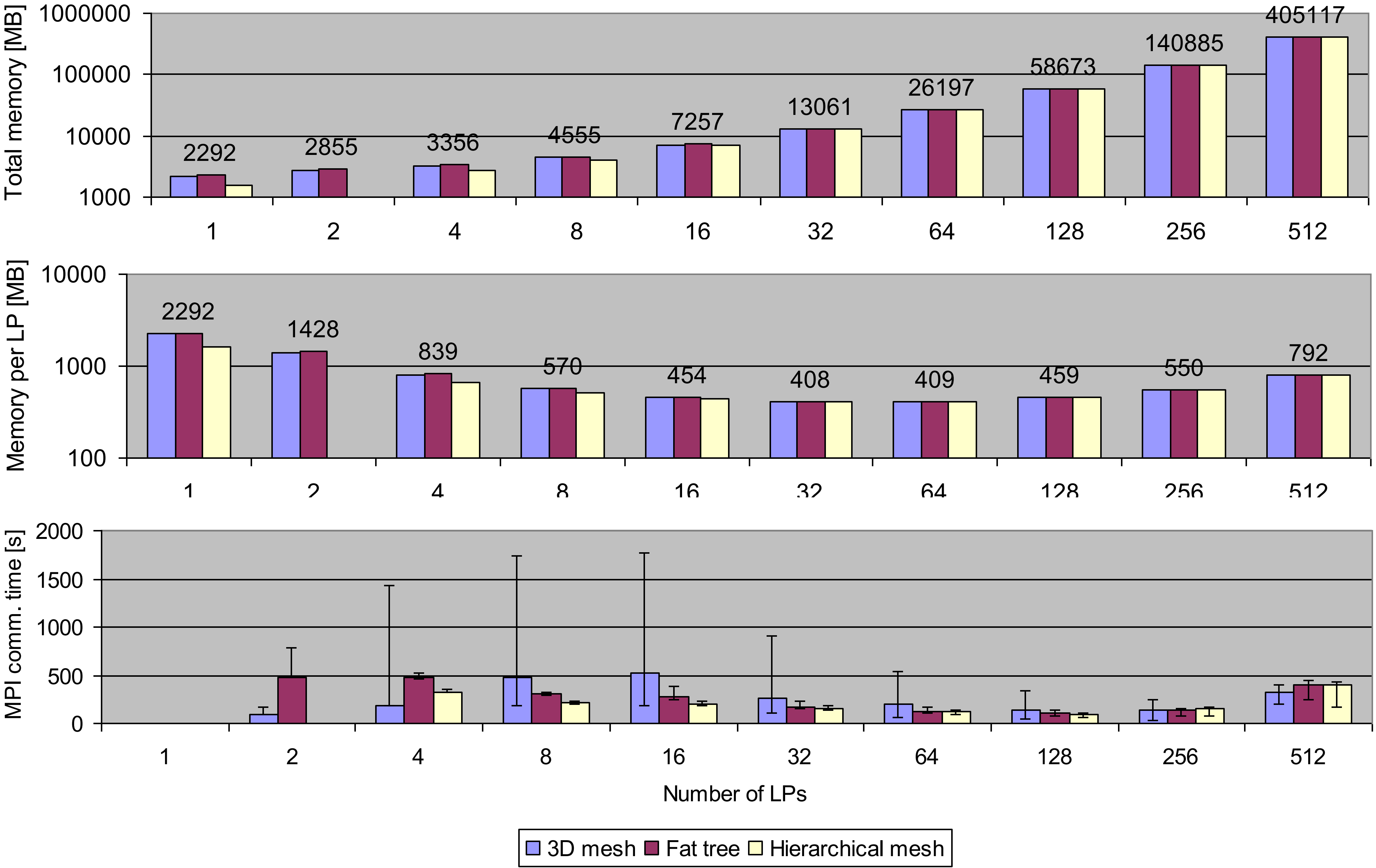




Relative runtime contributions (2)



Memory footprint



Next steps

- **Share required code changes with community**
- **Reduce per-partition memory footprint**
  - Wasteful MPI buffer allocation?
  - Potential to improve parsim MPI implementation?
- **Deterministic parallel execution**
  - In theory, results should be independent of the number of partitions (for same traffic pattern)
  - In practice, this is very hard to achieve
- **This approach can currently only be applied in conjunction with Venus-internal traffic generators**
  - Stochastic traffic (random spatial and temporal pattern)
  - Deterministic traffic (predetermined spatial and temporal pattern)
  - Workload models (collectives, benchmarks, mini-apps, skeleton apps)
- **Trace replay part is still sequential!**
  - …and so is the co-simulation interface
  - For true scalability, trace replay needs to be parallel as well
- **Apply our optimization methodology to the Venus/Omnest simulator itself**

Conclusions

- **Communication subsystem is a critical component of peta- and exascale HPC systems**
  - Performance-limiting factor for communication-heavy codes
  - Account for an increasingly significant fraction of system cost and power

- **Modeling and simulating such systems requires a PDES approach**
  - Workload-oriented approach is essential to achieve optimal cost/performance balance for specific uses
  - Manage simulation times and memory footprint
  - Omnest provides the right support
  - Parallelization requires a certain coding discipline

- **Successfully ported Omnest & Venus to the Blue Gene platform**

- **Reasonable parallel efficiency can be achieved without much tuning**
  - Relative speedup of >25% with up to 256 partitions
  - Depends quite significantly on simulated topology (diameter)

Questions?