Converging Interconnect Fabric Requirements for HPC and Warehouse Scale Computing

John Shalf
Department Head for Computer Science
CTO: National Energy Supercomputing Center
Lawrence Berkeley National Laboratory

ExaComm at the ISC Conference
July 16, 2015
Frankfurt, Germany
Overview

• **HPC Application Performance in the Cloud**
  – **Hardware**: is cloud correctly formulated for HPC?
  – **Software**: are HPC apps correctly formulated for the cloud?

• **Use similar CPU’s, but HPC/Cloud fabric requirements diverge**
  – **Old CW**: Cloud/datacenter requirements not aligned with HPC
  – **New CW**: Cloud requirements are more aligned than ever with HPC requirements

• **HW: New Cloud fabrics closer to HPC requirements**
  – Converging on high-performance internal networks
  – Still divergent on variability and fault recovery

• **SW: New formulations of HPC apps. closer to Cloud requirements**
  – Even exascale machines will have high variability & fault rates
  – Potential solutions would benefit both environments
  – But formulation requires fundamental rethink of math & algorithms
Measuring Cloud Value Proposition (2009)

• Evaluate the effectiveness of Clouds for HPC
  – Potentially replace DOE HPC Centers with credits in the “cloud”

• Performance Barometer: Performance of HPC apps in the cloud
  – What was the original cloud value proposition (circa 2009)

• Convergent requirements for processors
  – Massive parallelism
  – COTS x86 nodes

• Divergence requirements for fabrics
  – Cloud: externally facing TPC/IP standard fabrics
  – HPC: internally facing high performance custom fabrics
Using HPC Workload to Assess Performance
(NERSC Sustained System Performance/SSP)

SSP represents delivered performance for a real workload

- CAM: Climate
- GAMESS: Quantum Chemistry
- GTC: Fusion
- IMPACT-T: Accelerator Physics
- MAESTRO: Astrophysics
- MILC: Nuclear Physics
- PARATEC: Material Science

- SSP: aggregate measure of the workload-specific, delivered performance of a computing system
- For each code measure
  - FLOP counts on a reference system
  - Wall clock run time on various systems
  - $N$ chosen to be 3,200
- Problem sets drastically reduced for cloud benchmarking

$$SSP = N \left( \prod_{i=1}^{M} P_i \right)^{\frac{1}{M}}$$
Application Rates and SSP

Carver: Infiniband Cluster
Franklin: Cray XT4 custom interconnect
Lawrencium: GigE cluster
EC2: high-instance (cloud)
Correlation with Communication Performance

![Graph showing the correlation between %Communication and Runtime Relative to Lawrencium for different systems like PARATEC, MAESTRO, IMPACT-T, GTC, MILC, and fvCAM. The graph plots Ping Pong Latency, Ping Pong Bandwidth, Random Ring Latency, and Random Ring Bandwidth.]
Variability (in compute and communication)

• Modest variability in compute
  – Not as bad as communication, but substantially more than HPC systems
  – HPC codes not well structured to handle this variability

• Lot of variability in communication performance
  – Virtualization
  – Cross-traffic (contention)
  – Not topologically compact in cloud (*greedy scheduler puts things on different subnets*)
Overall Assessment in 2009

• Significantly lower interconnect performance
  – High overheads due to virtualization and TCP/IP overheads
  – Low effective bandwidth compared to IB or HPC-custom Nets.

• Significant performance variability
  – Stacking of VM’s for spot-priced hardware
  – Expensive to use un-Virtualized (or non-stacked) hardware
  – Shared interconnect without placement affinity for instances (hotspots, contention, traffic interference)

• Significantly higher MTBF: Variety of transient failures, including
  – Boot failures and intermittent virtual machine hangs;
  – Inability to obtain requested resources.

• Bottom Line
  – Cost to operate HPC center $50M/year
  – Cost to operate equivalent Top500/Linpack system in cloud $200M/year
  – Cost to operate equivalent app. performance facility in cloud $1B/year
### Cloud/Datacenter NEEDS high performance internal fabric
*(requirements are starting to align with HPC)*

- **Old Hardware Drivers for Clouds**
  - **COTS:** Lowest cost off-the-shelf Ethernet gear *(HPC pushed towards high-performance fabrics for best TCO.)*
  - **External vs. Internal:** TCP/IP primarily to external network loads for web services *(HPC primarily internally focused traffic patterns)*
  - **Throughput vs. Overhead:** Throughput valued more than low latency + overheads *(HPC needed lower latency)*
  - **Overheads:** Stacked VMs for elasticity and dynamic loads *(but hurt HPC due to performance heterogeneity and overheads)*
  - **Contention:** Provision nodes for loosely coupled random traffic *(tightly coupled jobs: provision contiguous topologies)*

- **New Developments in Drivers for Cloud/Datacenter**
  - **Bikash Koley, Google Inc. (OI2012):** 80%+ of Google traffic now internal facing *(used to be the other way around)*
  - **Nathan Farrington, Facebook (OI2013):** Every **1kb** of external traffic entering the datacenter generates **930kb** of internal traffic.
Facebook seeing huge internal traffic requirements
(Nathan Farrington, Facebook Inc., Presented at OIC2013)

HTTP request amplification

This 1 KB HTTP request generated 930 KB of internal network traffic
Egress traffic from one rack

 Majority of Facebook Traffic is Intra-Cluster  
(Nathan Farrington, Facebook Inc., Presented at OIC2013)
New Requirements

• New Requirements
  – Need lower Overhead to send minimum sized messages
  – Higher intra-datacenter bandwidth

• Cloud/Datacenter Responses
  – Willing to sacrifice full TCP/IP stack (designed for wide area) to get higher performance and lower overheads inside of datacenters
  – Push towards hardware technology to reduce bandwidths and latencies
  – Push towards leaner more efficient software stacks
Google “Pluto Switch”
(fell off truck somewhere in Iowa)

OMG: Google is building its own semi-custom switches!
Responses to New Cloud/Datacenter Requirements

- **Custom Intra-Center Fabrics** (*Google not waiting !!*)
  - Who the heck cares what you use for transport within the center?
  - Meaner/Leaner communications software stack
  - Modify switches to use more effective congestion control or avoidance
    - TCP/IP just uses inefficient lagging indicators of congestion or waits for packet drop + AIMD avoidance
  - Optical Circuit Switches: why use packet switching for persistent flows? *(e.g. OpenFlow, but make it a hard circuit for QoS guarantees)*

- **System on Chip (SOC):** Move NIC into CPU chip (silicon motherboard)
  - *Use Moore’s law* to put more *peripherals* onto chip instead of more *cores*
  - *Reduces component count* *(reduces cost, size and complexity of motherboards)*
  - *Reduces power* *(fewer off-chip connections)*
  - *Reduces sources of failure* *(fewer solder joints and connectors... ask ORNL about that)*
  - *Increases performance* *(factor of 20x reduction in software overheads)*
Response to the NEW Cloud/Datacenter Requirements

• Intel Acquisition Spree
  – **Qlogic**: Infiniband
  – **Fulcrum**: Ethernet
  – **Cray**: Aries (custom HPC)
  – **Quote from Raj Hazra** (to HPCwire 2012): "We are seeing the role of the fabric far less like a network, in a loosely coupled sense, and far more like a system bus at the datacenter level"

• AMD Acquisitions
  – **SeaMicro**: Atom + fabric to connect many together for datacenter

• Ecosystem Disruption: We can no longer mix-n-match the processor with the interconnect (**CPU choice defines fabric**)

• Bottom Line: If you are a CPU vendor who wants to play in the datacenters, need the NIC on the CPU chip
Old/New Conception of Cloud/Datacenters
(Simplified Conceptual Model)

**Old Conception**

- Designed for externally facing TCP/IP
- Nearly 100% Std. TCP/IP ethernet inside and out

The Datacenter/Cloud

- DMZ
- Router
- 90+% traffic

Backbone
Old/New Conception of Cloud/Datacenters

(Simplified Conceptual Model)

New Conception
Need to Handle Internal Data Mining/Processing
Design for 80+% internal traffic

Cloud/Datacenter
80+% of traffic

DMZ Router
10% traffic

Backbone

High-Performance Fluid Center
Low Overhead, High Bandwidth, Semi-custom internal interconnect

Crunchy TCP/IP Exterior

Execution Models Review: April 3, 2013
Looks Like Conceptual Diagram of a Typical HPC System

New Conception
Need to Handle Internal Data Mining/Processing
Design for 80+% internal traffic

Cloud/Datacenter
80+% of traffic

DMZ Router
10% traffic

Backbone

High-Performance Fluid Center
Low Overhead, High Bandwidth, Semi-custom internal interconnect

Crunchy TCP/IP Exterior
Convergence with HPC Requirements? (scorecard)

- **COTS:** Lowest cost off-the-shelf Ethernet gear (HPC pushed towards high performance fabrics for best TCO.)
- **External vs. Internal:** TCP/IP primarily to external network loads for web services (HPC primarily internally focused traffic patterns)
- **Throughput vs. Overhead:** Throughput valued more than low latency + overheads (HPC needed lower latency)
- **Contention:** Provision nodes for loosely coupled random traffic (tightly coupled jobs: provision contiguous topologies)
- **Performance Variation:** Dynamic behavior for elasticity and cost (but hurt HPC due to performance heterogeneity and overheads)
- **Resilience:** Loosely coupled jobs, depend on software to tolerate failure (HPC tightly coupled parallelism depends on HW to avoid failures... software not very tolerant of faults)
Today we have a bulk synchronous, distributed memory, communicating sequential processes (CSP) based execution model

- We’ve evolved into it over two decades
- It will require a lot of work to carry it forward to exascale
- The characteristics of today’s execution model are mis-aligned with emerging hardware trends of the coming decade

Due Diligence to examine alternative execution models for HPC

- Alternatives exist, and they look promising (e.g. async Ems such as SWARM, HPX, Charm)
- Use modeling and simulation to evaluate for DOE HPC applications
- This can guide our hardware/software trade-offs in the codesign process, and expands options for creating more effective machines

Asynchronous EMs: aligned with cloud/datacenter SW challenge

- Make software more tolerant of asynchrony than BulkSync/MPI
- Make software more tolerant of failures than current approaches
Converging HPC Software with Cloud Hardware
(meeting the cloud half-way)

• Meet the cloud half-way by redesigning to tolerate asynchrony and faults

• If successful would make cloud safe for HPC
  – Enables more aggressive energy savings options
  – Side-benefit of overcoming the huge technical challenge of keeping 1B+ processors executing in lock-step, which sounds absurd on the face of it.
Rewrite of GTC to get Maximum Async Execution
(not a simple port of the application... need to reformulate the math)

- Eliminated Sources of Synchronization
- See some signs of async behavior/benefit (best task granularity > 7tasks/core
- But still suffer from barriers (still a long way to go to match “stupid mode”)

Issues
- Poisson solve creates implicit barrier (have to complete globally before moving on)
- Not just a CS exercise (don’t get to port to new library and call for success)
- Requires deep reformulation of the underlying math & algorithms to see benefits
**Preliminary Observations**

- **We’ve spent 20 years rewriting Scientific numerical algorithms to eliminate non-bulk-sync behavior**
  - It's difficult to reverse such a global and comprehensive effort
  - It’s a deep reformulation of the fundamental math/algorithm
  - It is NOT simply a matter of porting to the correct runtime/communication system (or correct version of MPI3)

- **Potential benefits are substantial (so still working)**
  - Convergence with cloud/datacenter fabrics
  - Enables CPU vendors to pursue more aggressive power reduction strategies (NTV operation, fine-grained speed throttling, etc...)
  - Enable better approach to fault recovery
Conclusions for Hardware
(cloud datacenter hardware realignment)

- Big data analytics pushing the cloud to high performance internal networking
  - TCP/IP to outside world
  - High perf. custom nets or stripped down protocol internally

- Server CPU vendors responding with Integrated NICs
  - Natural consequence of SOC integration
  - Huge performance, cost and power benefits
  - Custom protocols to reduce CPU overheads

- Implication: Hardware more aligned with HPC
Conclusions for Software (HPC software realignment)

- HPC hardware expected to be more heterogeneous and prone to faults
  - Future requirements for HPC software more aligned with what we consider deficiencies of today’s cloud
  - Redesigning software to embrace solutions that overcome heterogeneity and resilience aligns with Cloud *and* exascale

- But HPC software re-alignment is much deeper issue
  - Not just choosing the correct communication library or language
  - Requires fundamental rethinking of algorithm and mathematics

- Implication: forcing function for HPC to push closer to clouds
Starting with the Gyrokinetic Toroidal Code

- GTC uses PIC method to simulate plasma microturbulence for fusion devices
- Written in F90 with MPI
- Scalable to thousands of processors
- Grid memory accesses depend on the order in which particles are processed.
- In a multithreaded implementation with a shared grid, multiple threads update grid locations in parallel.
- The case of random particle positions and parallel updates is similar to the GUPS benchmark. However, implementations usually exploit the fact that PIC is a physical many-body simulation method.

**GTC**

- Uses PIC method to simulate plasma microturbulence for fusion devices
- Written in F90 with MPI
- Scalable to thousands of processors
- Grid memory accesses depend on the order in which particles are processed.
- In a multithreaded implementation with a shared grid, multiple threads update grid locations in parallel.
- The case of random particle positions and parallel updates is similar to the GUPS benchmark. However, implementations usually exploit the fact that PIC is a physical many-body simulation method.
• Typical load imbalance performance loss can be between 10-30% of the runtime, depending on concurrency and problem definition
• Partially due to:
  – Dynamic nature of the computation- different numbers of particles move at each time step
  – Initial conditions – static load-imbalance
Initial Scaling Results Disappointing
(can’t port directly from MPI to Async)

- How does a fixed problem scale with machine size (strong scaling), oversubscription = 2
Rewrite of GTC to get Maximum Async Execution
(not a simple port of the application... need to reformulate the math)

- Eliminated Sources of Synchronization
- See some signs of async behavior/benefit (best task granularity > 7 tasks/core)
- But still suffer from barriers (still a long way to go to match stupid mode)

Issues
- Poisson solve creates implicit barrier (have to complete globally before moving on)
- Not just a CS exercise (don’t get to port to new library and call for success)
- Requires deep reformulation of the underlying math & algorithms to see benefits
Summary of HPC Clouds 2009

• Weaknesses of Cloud/Datacenter Technology for HPC
  – Ethernet inhibits MPI communication performance
  – Performance inconsistency due to virtualization and fabric loading hurts synchronous apps
  – Significantly greater performance range
  – Failure rates high for application benchmarks

• Building Clouds for Science
  – Use high-performance fabric (IB)
  – Shared global filesystem
  – Don’t stack VM images (or offer un-virtual access to hardware)
  – Improved fault resilience, or improved application resilience

• Not measured: I/O and network performance
Some other things for Clouds/Datacenters to Consider

• **Lightweight communication libraries**
  – Already seeing benefits of paring down TCP/IP protocol
  – Additional benefits realized from considering alternative APIs e.g. GAS

• **UPC or PGAS for data mining**
  – GAS is really good for irregular graph problems
  – NSA has been at this for a long time
  – Might have something to learn (although they won’t tell you)

• **Data locality/topology awareness**
  – Naïve random all-to-all is just bisection bandwidth buster (not much freedom for innovation in hardware or software)
  – Data movement distance is huge power cost (see Dally talk!)
  – If there is any exploitable locality, take advantage of it (understand that Google’s internal replacement for MapReduce does just that)
UPC Was Designed for Irregular Data-Mining Problems

More Regular

Message Passing Programming
Divide up domain in pieces
Compute one piece
Send/Receive data from others

MPI, and many libraries

More Irregular

Global Address Space Programming
Each start computing
Grab whatever / whenever

UPC, CAF, X10, Chapel, GlobalArrays
UPC Designed for Irregular “big data” Problems
*(plug to support HPC features like GAS in fabric)*

**More Regular**

Message Passing Programming
- Divide up domain in pieces
- Compute one piece
- Send/Receive data from others

*MPI, and many libraries*

**More Irregular**

Global Address Space Programming
- Each start computing
- Grab whatever / whenever

*UPC, CAF, X10, Chapel, GlobalArrays*

( NSA has been in big-data biz with UPC for long time)