Co-Authors:

Prof. S. Matsuoka
Ivan R. Ivanov
Yuki Tsushima
Tomoya Yuki
Akihiro Nomura
Shin'ichi Miura
Nic McDonald
Dennis L. Floyd
Nicolas Dubé

Tokyo Tech

Hewlett Packard Enterprise

HyperX Topology

First At-Scale Implementation and Comparison to the Fat-Tree

Jens Domke, Dr. rer. nat. < jens.domke@riken.jp >
High Performance Big Data Research Team, RIKEN R-CCS, Kobe, Japan

RIKEN R-CCS Computer simulations create the future

Outline

- **5-min high-level summary**
- From Idea to Working HyperX
- Research and Deployment Challenges
  - Alternative job placement
  - DL-free, non-minimal routing
- In-depth, fair Comparison: HyperX vs. Fat-Tree
  - Raw MPI performance
  - Realistic HPC workloads
  - Throughput experiment
- Lessons-learned and Conclusion

1^{st} large-scale Prototype – Motivation for HyperX

➔ **First large-scale 2.7 Pflop/s (DP) HyperX installation in the world!**

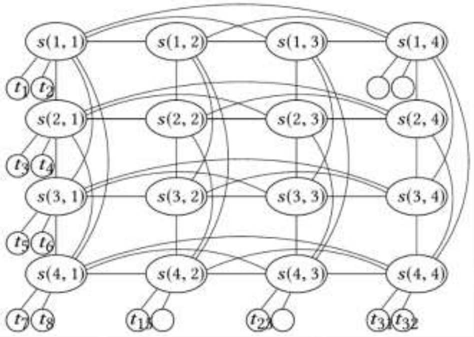


Fig.1: HyperX with n-dim. integer lattice (d_1,…,d_n) base structure fully connected in each dim.

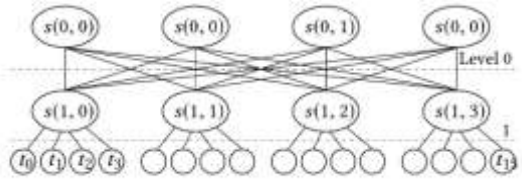
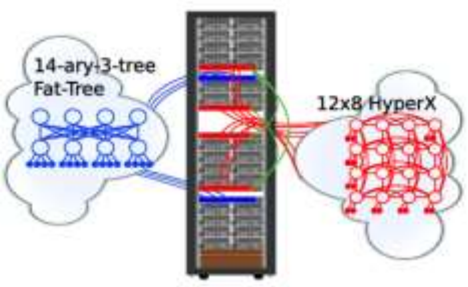


Fig.2: Indirect 2-level Fat-Tree

TokyTech's 2D HyperX:

- **24 racks** (of 42 T2 racks)
- 96 QDR switches (+ 1st rail) **without adaptive routing**
- 1536 IB cables (720 AOC)
- **672 compute nodes**
- **57% bisection bandwidth**



Theoretical Advantages (over Fat-Tree)

- **Reduced HW cost** (less AOC / SW)
- **Only needs 50% bisection BW**
- **Lower latency** (less hops)
- **Fits** rack-based **packaging**

Evaluating the HyperX and Summary

1:1 comparison (as fair as possible) of
672-node 3-level Fat-Tree and 12x8 2D HyperX

- NICs of 1st and 2nd rail even on same CPU socket
- Given our HW limitations (few "bad" links disabled)

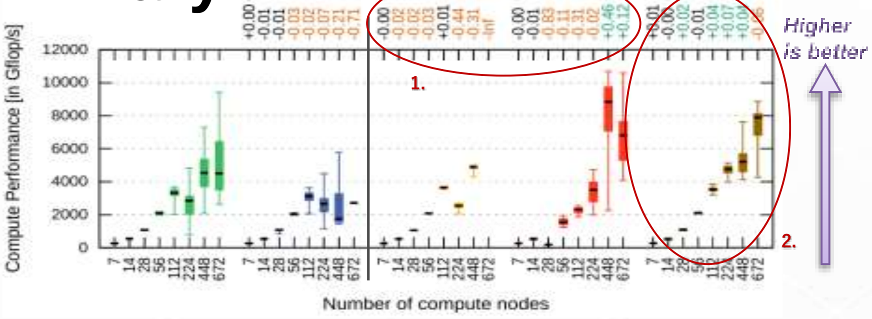


Fig.3: HPL (1GB pp, and 1ppn); scaled 7→ 672 cn

Wide variety of benchmarks and configurations

- 3x Pure MPI benchmarks
- 9x HPC proxy-apps
- 3x Top500 benchmarks
- 4x routing algorithms (incl. PARX)
- 3x rank-2-node mappings
- 2x execution modes

Fig.4: Baidu's (DeepBench) Allreduce (4-byte float) scaled 7→ 672 cn (vs. "Fat-tree / ftree / linear" baseline)

Primary research questions

Q1: Will reduced bisection BW (57% for HX vs. ≥100% for FT) impede performance?

Q2: Two mitigation strategies against lack of AR? (→ e.g. placement vs. "smart" routing)

1. Placement mitigation can alleviate bottleneck
2. HyperX w/ PARX routing outperforms FT in HPL
3. *Linear* good for small node counts/msg. size
4. *Random* good for DL-relevant msg. size (+/− 1%)
5. "Smart" routing suffered SW stack issues
6. FT + ftree had bad 448-node corner case

Conclusion

HyperX topology is promising and cheaper alternative to Fat-Trees (even w/o adaptive R) !

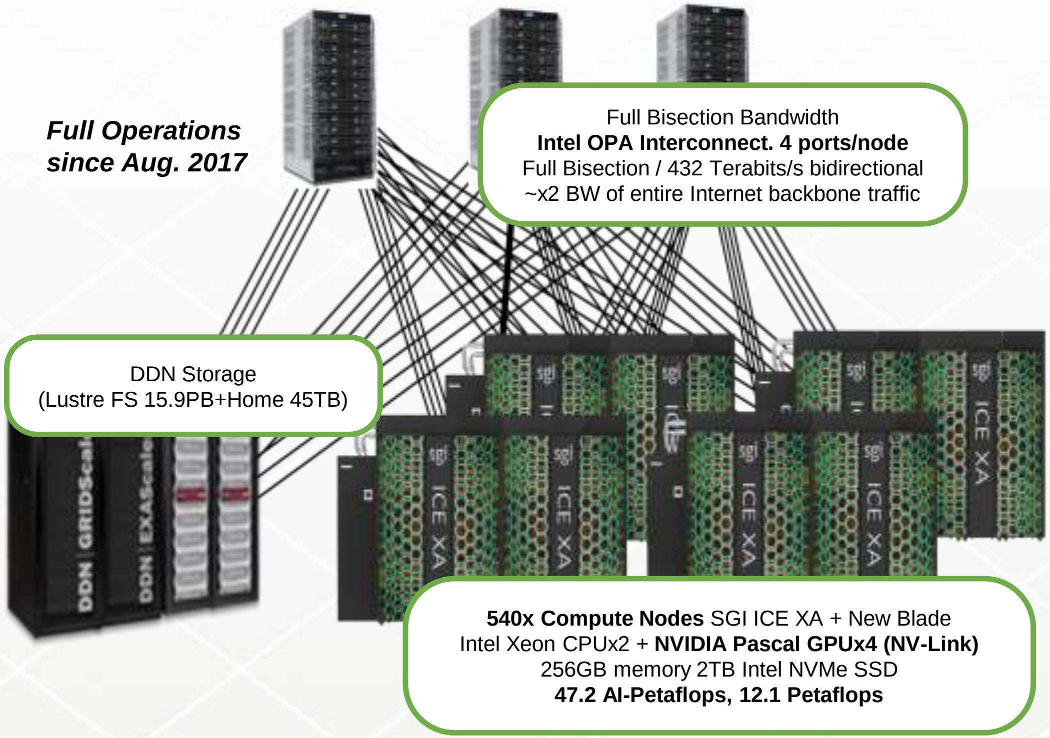
Outline

- 5-min high-level summary
- From Idea to Working HyperX
- Research and Deployment Challenges
  - Alternative job placement
  - DL-free, non-minimal routing
- In-depth, fair Comparison: HyperX vs. Fat-Tree
  - Raw MPI performance
  - Realistic HPC workloads
  - Throughput experiment
- Lessons-learned and Conclusion

TokyoTech's new TSUBAME3 and T2-modding

New TSUBAME3 – HPE/SGI ICE XA

Full Operations since Aug. 2017

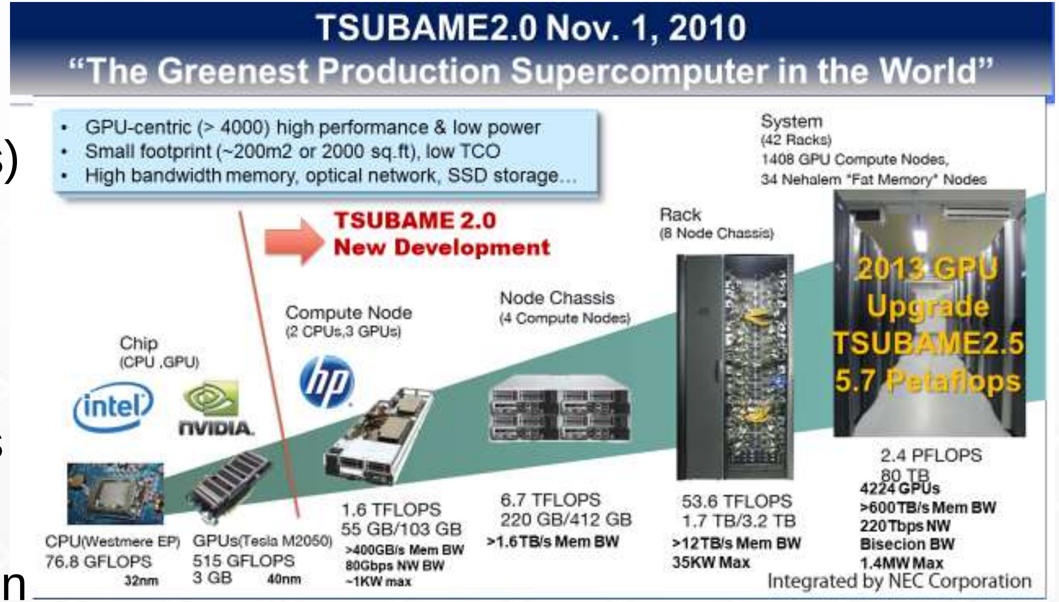


But still had 42 racks of T2…

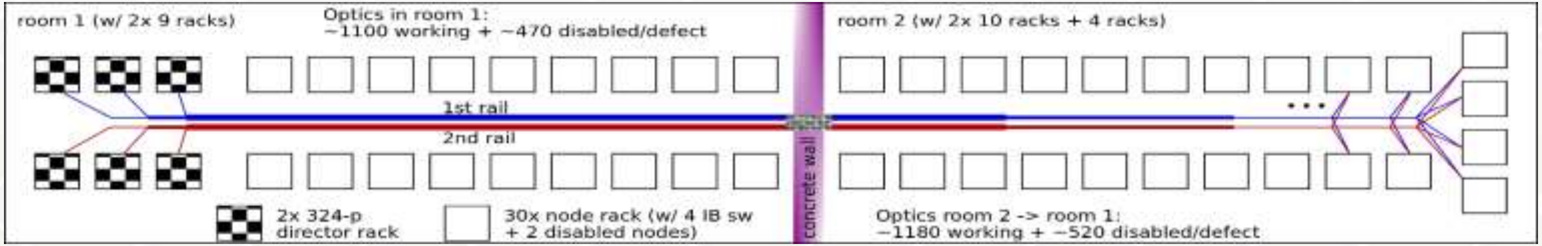
Results of a successful HPE – TokyoTech R&D collaboration to build a HyperX proof-of-concept

TSUBAME2 – Characteristics & Floor Plan

- **7 years** of operation ('10–'17)
- **5.7 Pflop/s** (4224 Nvidia GPUs)
- **1408 compute nodes** and ≥100 auxiliary nodes
- 42 compute racks in 2 rooms +6 racks of IB director switches
- Connected by **two separated QDR IB networks** (full-bisection fat-trees w/ 80Gbit/s injection per node)

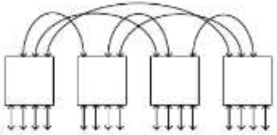


2-room floor plan of TSUBAME2

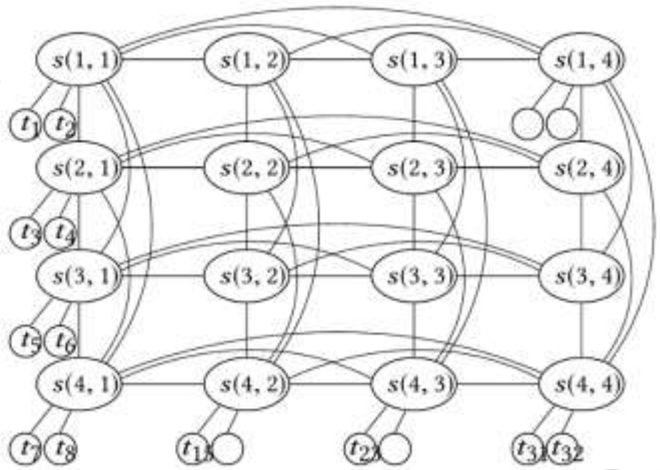


Recap: Characteristics of HyperX Topology

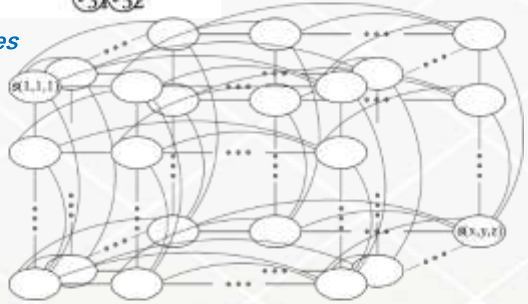
- Base structure
  - Direct topology (vs. indirect Fat-Tree)
  - **n-dim. integer lattice** $(d_1,\ldots,d_n)$
  - **Fully connected** in each dimension
- Advantages (over Fat-Tree)
  - **Reduced HW cost** (less AOC and switches) for similar perf.
  - **Lower latency** when scaling up
  - **Fits** rack-based **packaging** scheme
  - Only **needs 50% bisection BW to provide 100% throughput** for uniform random
- But… (theoretically)
  - **Requires adaptive routing**



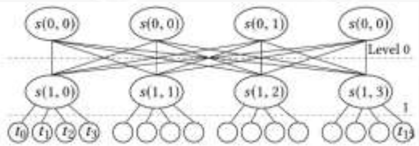
a) 1D HyperX with $d_1 = 4$



b) 2D (4x4) HyperX w/ 32 nodes



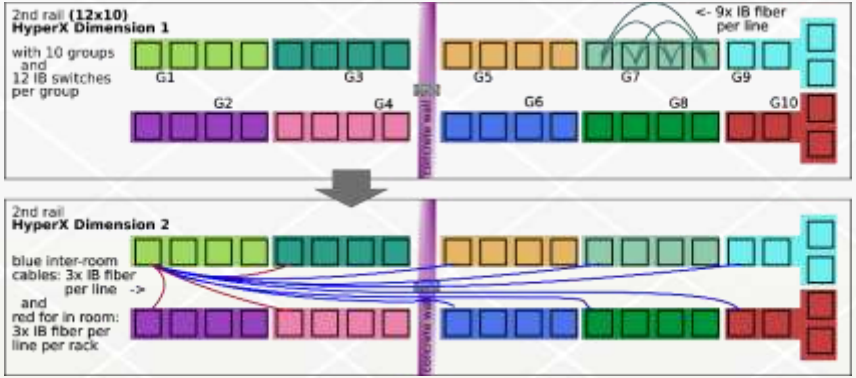
c) 3D (XxYxZ) HyperX



d) Indirect 2-level Fat-Tree

Plan A – A.k.a.: Young and naïve ☺

- Scale down #compute nodes
  ➔ 1280 CN and keep 1st IB rail as FT
- Build **2nd rail with 12x10 2D HyperX** distributed over 2 rooms
- Theoretical **Challenges**
  - **Finite** amount/length of IB **AOC**
  - Cannot remove **inter-room AOC**



Fighting the **Spaghetti Monster**



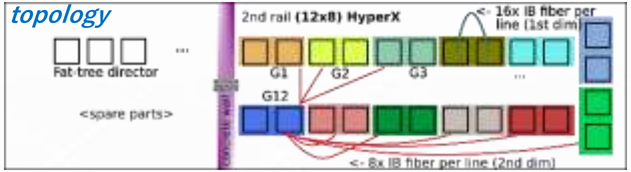
- 4 gen. of AOC ➔ mess under floor
- "Only" ≈900 extracted cables from 1st room using cheap students labor

Still, **too few cables, time, & money** …

~~Plan A~~ ➔ **Plan B !**

Plan B – Downsizing to 12x8 HyperX in 1 Room

Re-wire 1 room with HyperX topology

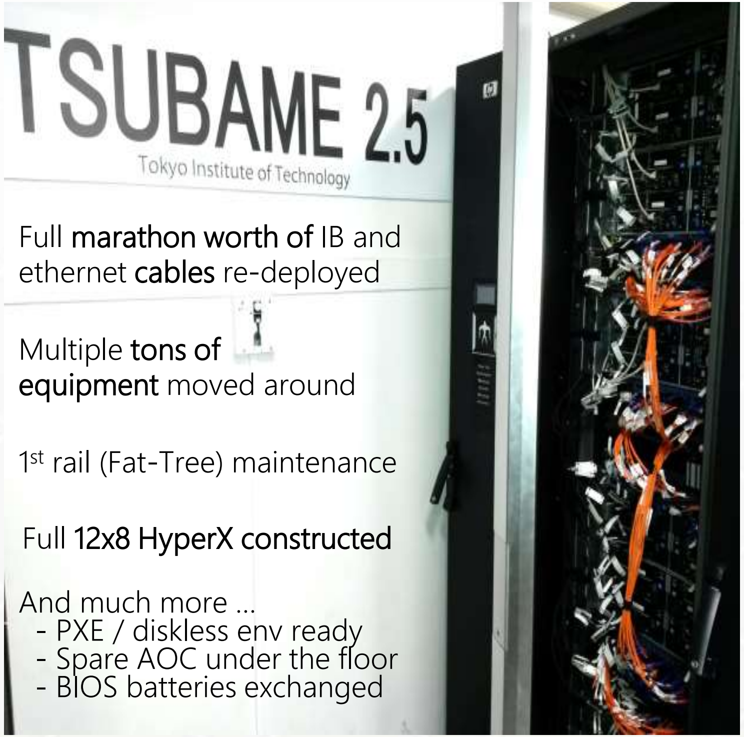


For **12x8 HyperX** need:

- **Add 5th + 6th IB switch** to rack
  ➔ remove 1 chassis
  ➔ 7 nodes per SW
- Rest of Plan A mostly same
- **24 racks** (of 42 T2 racks)
- **96 QDR switches** (+ 1st rail)
- **1536 IB cables** (720 AOC)
- **672 compute nodes**
- **57% bisection bandwidth**
- +1 management rack

Rack: back

Rack: front



Full **marathon worth of** IB and ethernet **cables** re-deployed

Multiple **tons of equipment** moved around

1st rail (Fat-Tree) maintenance

Full **12x8 HyperX constructed**

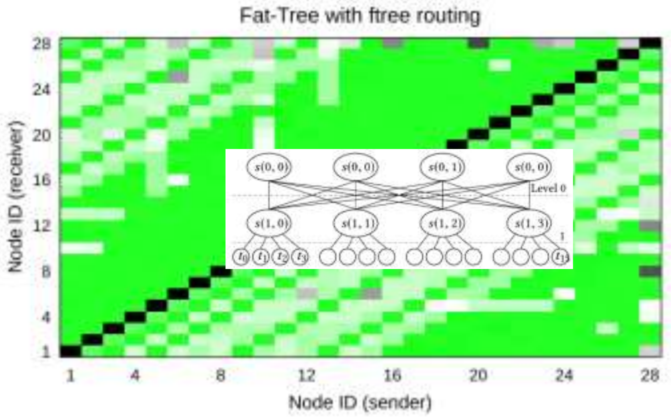
And much more …
- PXE / diskless env ready
- Spare AOC under the floor
- BIOS batteries exchanged

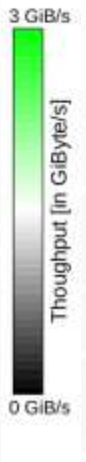
➔ **First large-scale 2.7 Pflop/s (DP) HyperX installation in the world!**

Missing Adaptive Routing and Perf. Implications

- TSUBAME2's older gen. of **QDR IB** hardware has **no adaptive routing** ☹
- HyperX with **static/minimum routing** suffers from **limited path diversity** per dimension
  ➔ results in high congestion and low (effective) bisection BW
- Our example: 1 rack (28 cn) of T2
  - Fat-Tree **>3x theor. bisection BW**
  - **Measured** 2.26 GiB/s (FT; ~**2.7x**) vs. 0.84 GiB/s for HyperX

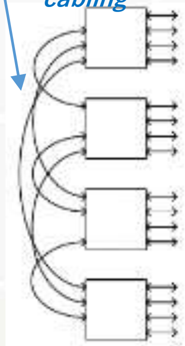
Mitigation Strategies???

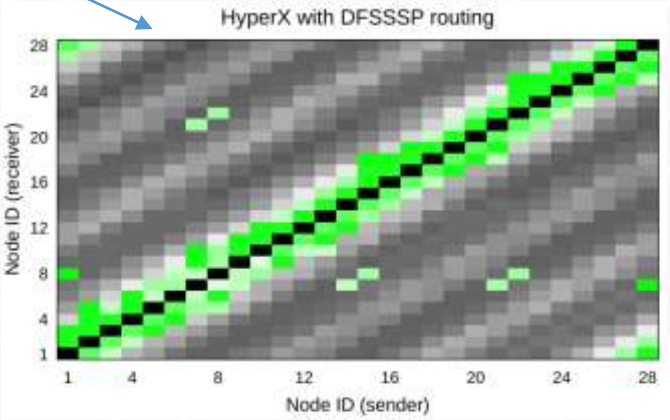




Measured BW in mpiGraph for 28 Nodes

HyperX intra-rack cabling

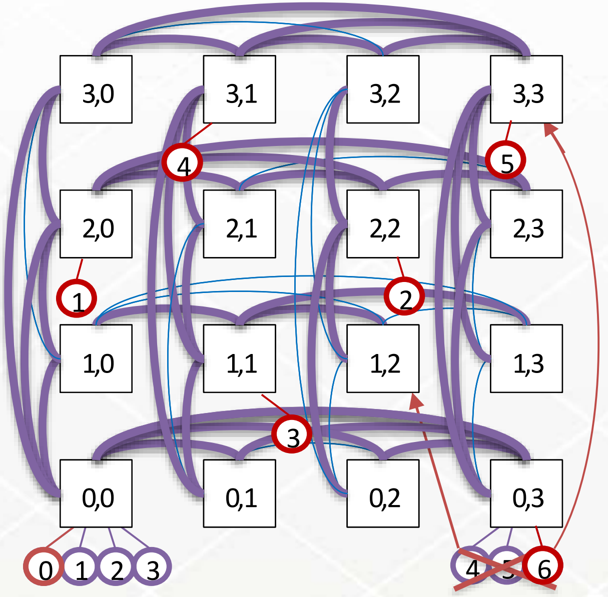




Option 1 – Alternative Job Allocation Scheme

Idea: spread out processes across entire topology

- Increases path diversity for incr. BW
- Compact allocation ➔ single congested link
- Spread out allocation ➔ nearly all paths available
- **Our approach: randomly assign nodes** (Better: proper topology-mapping based on real comm. demands per job)
- Caveats:
  - Increases hops/latency
  - Only helps if job uses subset up nodes
  - Hard to achieve in day-to-day operation

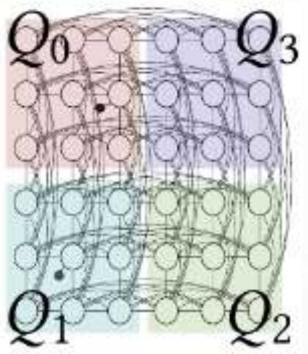


2D HyperX

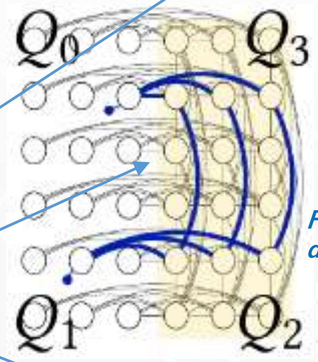
Option 2 – Non-minimal, Pattern-aware Routing

Idea (Part 1): **enforcing non-minimal routing** for higher path diversity (not universally possible with IB)
(+ Part 2) while integrating **traffic-pattern and comm.-demand awareness** to emulate adaptive and congestion-aware routing

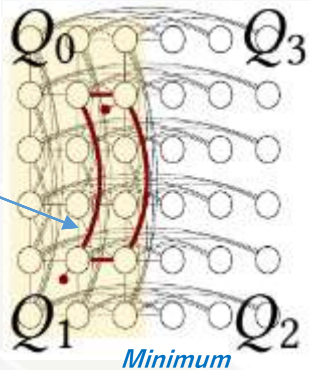
- Pattern-**A**ware **R**outing for hyper**X** (**PARX**)
  - "Split" our 2D HyperX into **4 quadrants**
  - Assign **4 "virtual LIDs"** per port (IB's **LMC**)
  - Smart link removal and path calculation
- **Optimize** static routing **for process-locality and know comm. matrix** and balance "useful" paths across links:
  - Basis: DFSSSP and SAR (IPDPS'11 and SC'16 papers)
- Needs support by MPI/comm. layer
  - Set LID_i^{dest} based on msg. size (**lat: short; BW: long**)



Quadrants



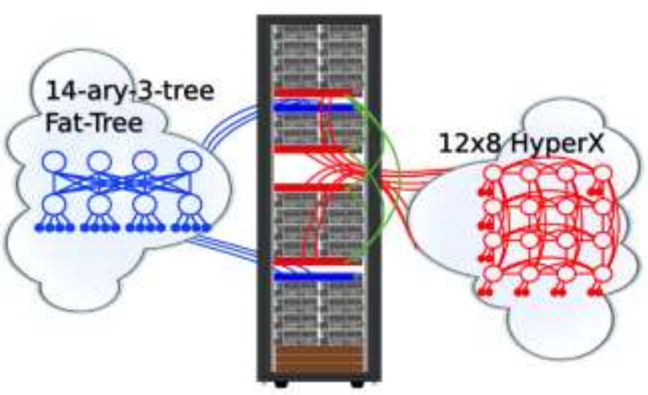
Forced detours



Minimum paths

Methodology – 1:1 Comp. to 3-level Fat-Tree

- Comparison **as fair as possible** of 672-node 3-level Fat-Tree and 2D HyperX
  - NICs of 1st and 2nd rail even on same CPU socket
  - Given our HW limitations (few “bad” links disabled)



- 2 topologies: Fat-Tree vs. HyperX
- 3 placements: **linear | clustered | random**
- 4 routing algo.: **ftree | (DF)SSSP | PARX**
- 5 combinations: **FT+ftree+linear (baseline) vs. FT+SSSP+cluster vs. HX+DFSSSP+linear vs. HX+DFSSSP+random vs. HX+PARX+cluster**

- …and many benchmarks and applications (all with 1 ppn):
  - **Solo/capability runs**: 10 trials; #cn: **7,14,…,672** (or pow2); conf. for **weak-scaling**
  - **Capacity evaluation**: **3 hours; 14 app**lications (32/56 cn); **98.8%** system util.

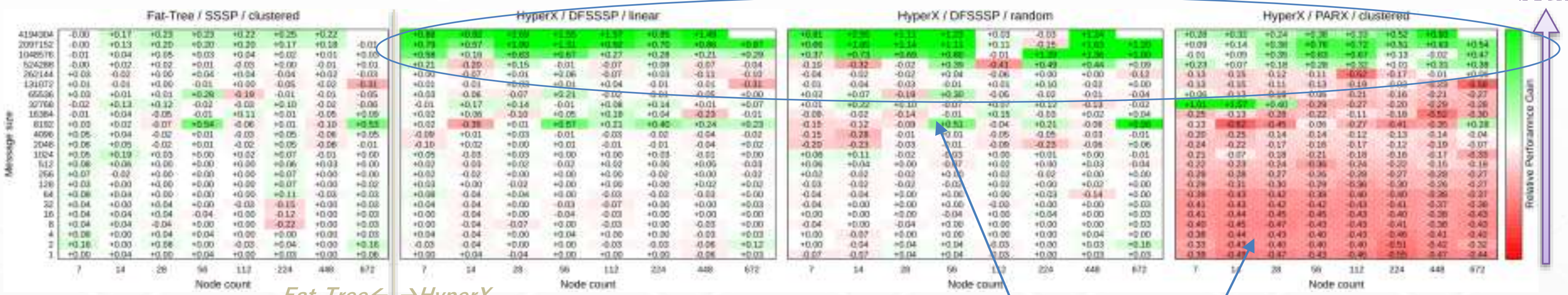
Benchmarks and (real-world) HPC Applications

- **MPI BMs** to evaluate peak perf.
- Applications sampled broadly from **range of HPC workloads**
  - Requ.: parallel implementation and “good” input (wrt. runtime)
  - 4x **ECP** proxy-apps
  - 3x **RIKEN** R-CCS priority apps
  - 1x **Trinity** BM (for NERSC systems)
  - 1x **CORAL** procurement BM
- …and the usual **“TOP 500”** BMs

➔ Should give good indication of HyperX topo. performance

Raw MPI	Workload
Intel’s IMB	Various MPI benchmarks (here limited to: **MPI-1 collectives**)
Netgauge eBB	Measure (routing-induced) **effective bisection bandwidth** of topology
Baidu’s Allred.	Evaluate **MPI traffic of Deep Learning** workload for various msg. sizes
x500	**Workload**
HPL	**Solves dense system** of linear equations **Ax = b**
HPCG	**Conjugate gradient** method on **sparse matrix A** to solve Ax = b
Graph500	Performs distributed **breadth-first search** (BFS) on a large graph
Proxy-Apps	**Workload**
AMG	Algebraic **multigrid solver** for unstructured grids
CoMD	Generate **atomic transition** pathways between any two structures of a **protein**
miniFE	Proxy for **unstructured implicit finite element** or finite volume applications
SWFFT	**Fast Fourier transforms (FFT)** used in by HW-Accel. Cosmology Code (HACC)
FFVC	Solves the 3D unsteady **thermal flow** of the incompressible fluid
mVMC	Variational **Monte Carlo** method for interacting fermion systems
NTChem	Molecular electronic structure calculation of std. **quantum chemistry** approaches
MILC	**Quantum chromodynamics (QCD)** simulations using lattice gauge theory
LLNL’s qb@ll	First-principles **molecular dynamics (MD)** using DFT

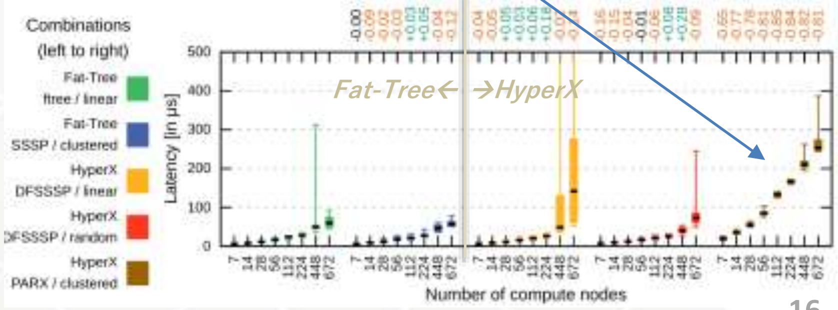
MPI – Subset of Intel's MPI Benchmarks



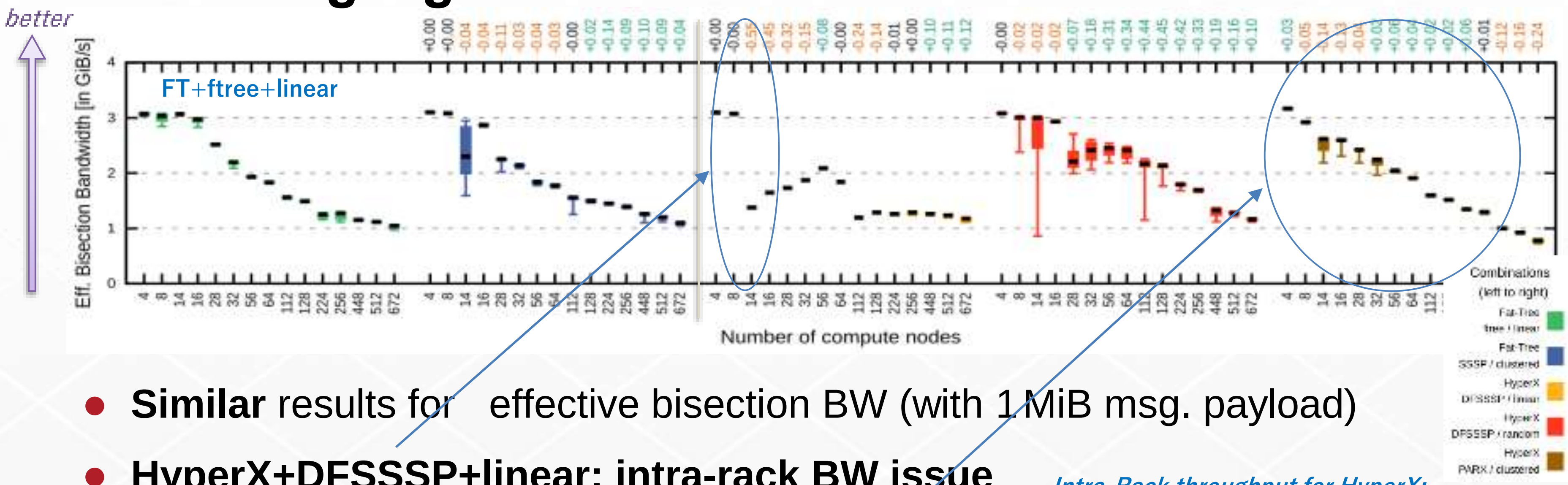
a) *IMB Gather – Relative gain over FT+ftree+linear*

- Tested Barrier, Bcast, **Gather**, Scatter, (All)reduce, Alltoall
- Here: HyperX **competitive** for small and **outperforms FT** for large msg.
- Performance issue in PARX (highly likely: unoptimized *bfo* PML)
- **Overall**: HX sometimes better or worse depending on MPI coll., msg. size, routing, & alloc. … **no clear winner!**
- Good results despite missing AR

b) *IMB Barrier*

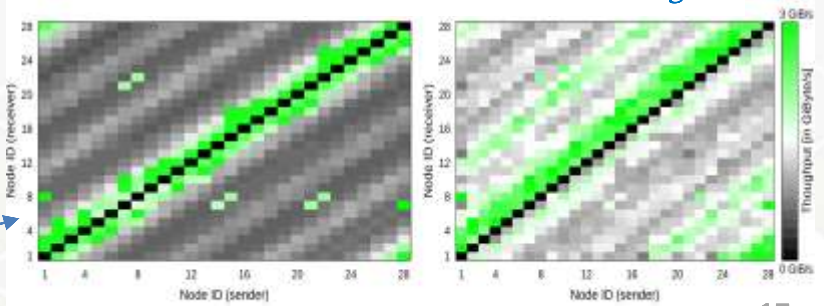


MPI – Netgauge's eBB Benchmark



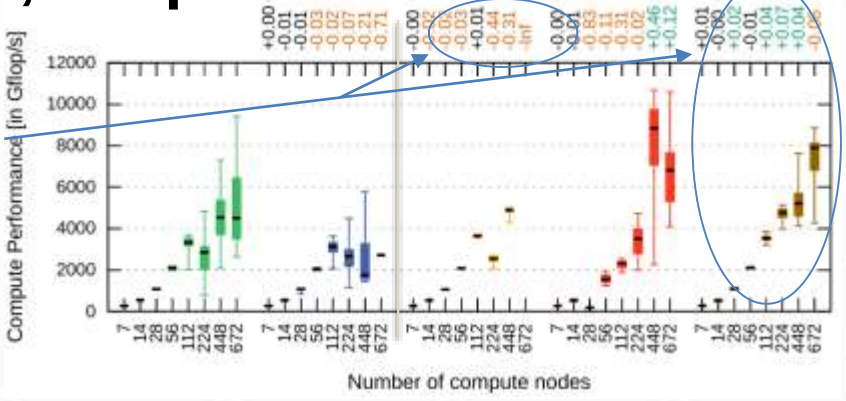
- **Similar** results for effective bisection BW (with 1 MiB msg. payload)
- **HyperX+DFSSSP+linear: intra-rack BW issue**
- Longer/more paths as enabled by **PARX alleviates perf. drop** (→ indicates theor. benefits when getting HX with AR)
- Similar to PARX vs. minimal routing in intra-rack case, cf. 28-cn mpiGraph BM

Intra-Rack throughput for HyperX: DFSSSP vs. PARX routing

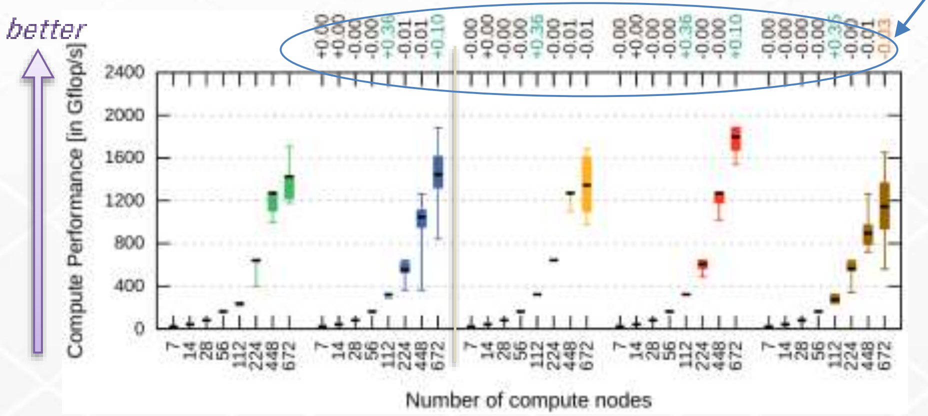


x500 Benchmarks – HPL, HPCG, Graph500

- HPL suffers from compact alloc. on HX but **HyperX beats FT with PARX** routing
- HX & FT perform **same for HPCG**

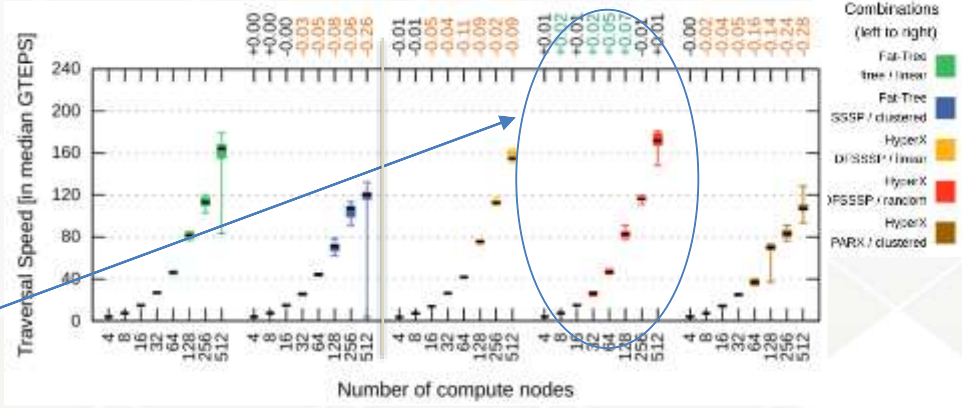


a) HPL (1GB pp)



b) HPCG

- **HyperX** w/ DFSSSP + rand allocation **outperforms FT for Graph500**



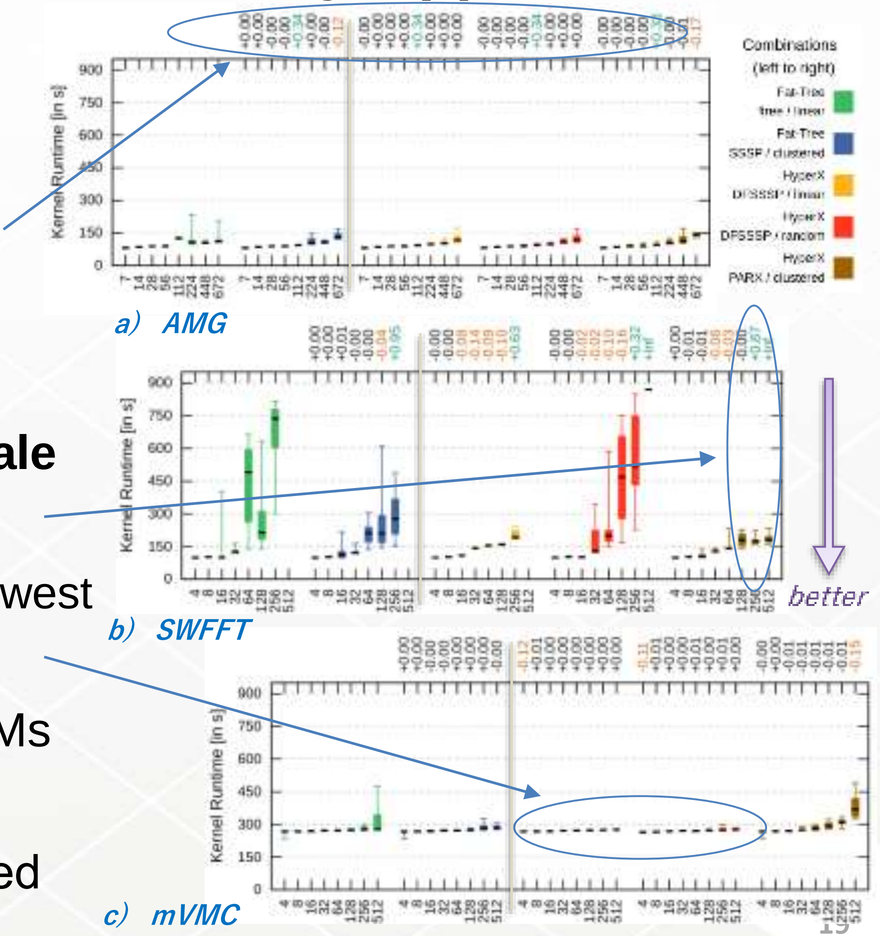
c) Graph500

Realistic Workloads – Procurement-/Proxy-Apps

- Subset of HPC workloads; reporting **kernel/solver times** (no pre-/post proc.)
- Almost no noticeable difference (all **within $^+/_-$1%** rel. gains) when switching **Fat-Tree → HyperX** for some apps
- **SWFFT**: **PARX** best option for HyperX (pattern-aware?) and **only option to scale to 512** nodes (all 10 in 233 s; see "+Inf")
- **mVMC**: HyperX/DFSSSP(/linear) shows lowest performance variability

➔ PARX overall less "bad" cf. raw MPI BMs (proxy-apps only ≈20% on avg. in MPI)

➔ No severe issues ☺ … but AR is desired



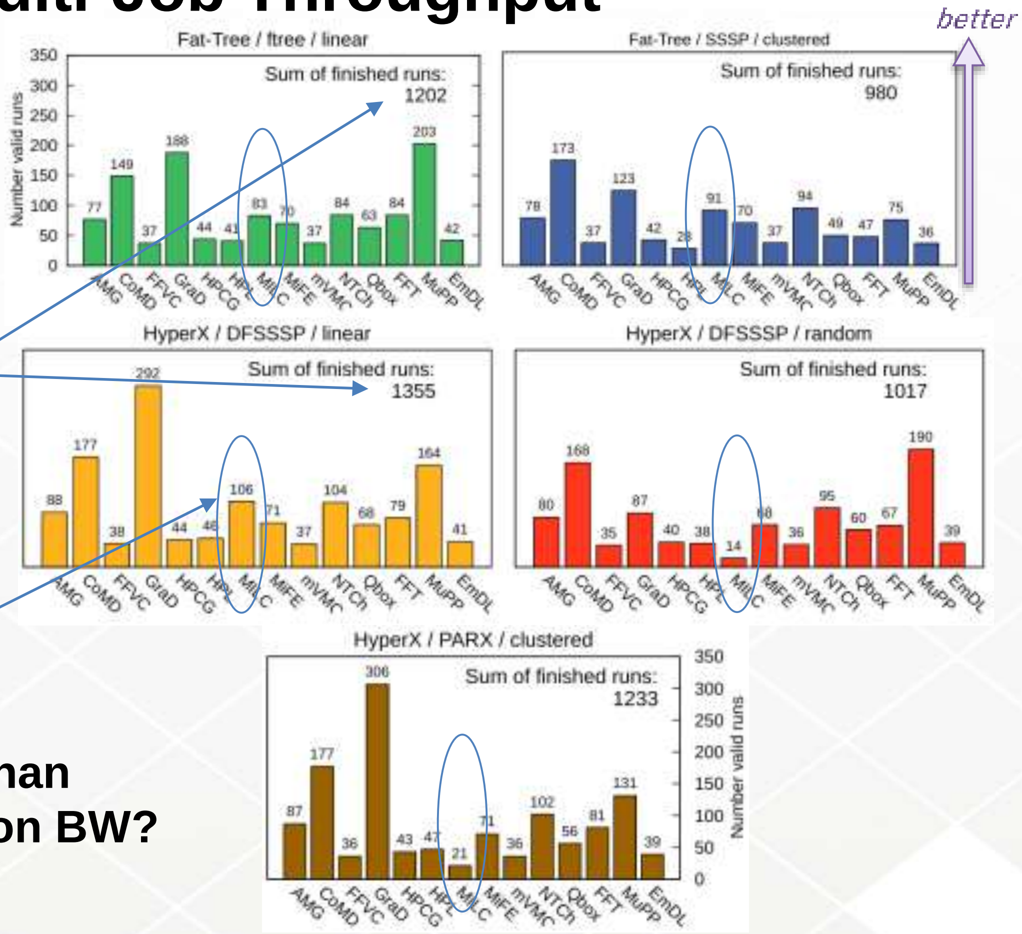
a) AMG

b) SWFFT

c) mVMC

Capacity Evaluations – Multi-Job Throughput

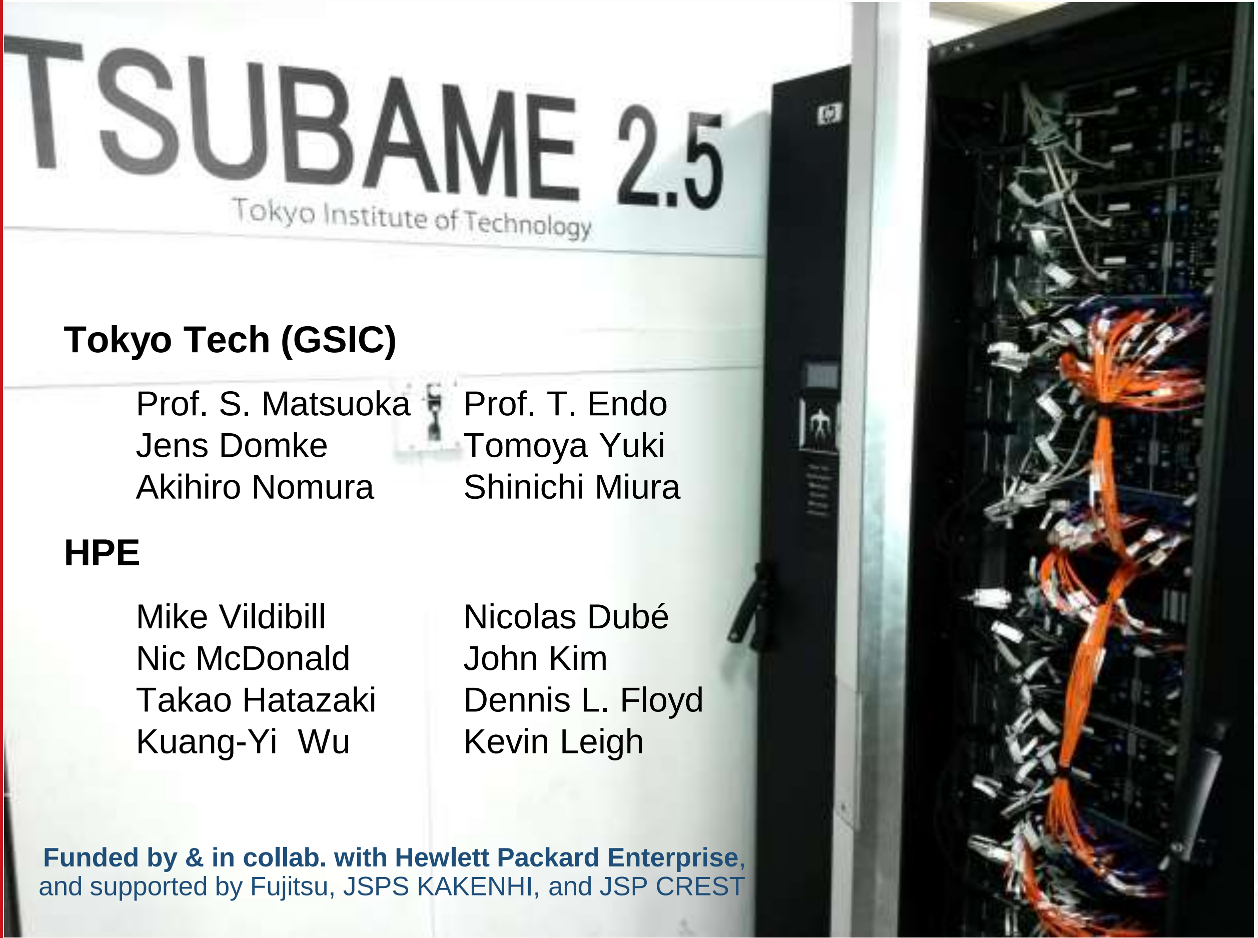
- More realistic scenario for most HPC centers (multi-job exec.)
- Metric: **#runs in 3h on shared network** (job alloc. fix w/ hostfile)
- Unexpected: **HX beats FT/ft/lin. by 12.7%** (DF/lin.) and 3% (PARX)
- **MILC** negatively **affected by inter-job interferences** (but linear alloc. on HX best among all 5)
- Linear vs. random vs. PARX: **Interferences have worse effect than bottlenecks in theoretical bisection BW?**



Lessons-learned and Conclusion

- Fun project (despite cable mess ☺) & enjoyable Univ./Industry collaboration
- **Deadlock-free routing is essential** for HyperX (in static case; likely for AR too)
- **PARX prototype shows potential** (→ could be adopted elsewhere) but MPI stack prohibited better results
- 2D HyperX (57% bisection BW; w/o AR) vs. under-subscribed 3-lvl Fat-Tree → our **12x8, 672-node HyperX did extremely well in all tests**
- **Open research**: **ideal job allocation** scheme and/or adaptive **routing** for different usage models (capacity vs. capability systems)
- **HyperX a compelling alternative…? Definitively!** ➔ Looking forward to next "real" HyperX system with adaptive routing!

Acknowledgements to HPE & Participants



Tokyo Tech (GSIC)

Prof. S. Matsuoka
Jens Domke
Akihiro Nomura

Prof. T. Endo
Tomoya Yuki
Shinichi Miura

HPE

Mike Vildibill
Nic McDonald
Takao Hatazaki
Kuang-Yi Wu

Nicolas Dubé
John Kim
Dennis L. Floyd
Kevin Leigh

Funded by & in collab. with Hewlett Packard Enterprise,
and supported by Fujitsu, JSPS KAKENHI, and JSP CREST

≥ 40 Tokyo Tech Student (and other) Volunteers

Nagashio, Shibuya, Aizawa, Takai
Ito, Oshino, Numata, Masukawa
Iijima, Minematsu, Muto, Oosawa
Yui, Hamaguchi, Asako, Fukaishi
Ivanov, Mateusz, Tam, Kitada
Ueno, Katase, Numata, Tsushima
Fukuda, Suzuki, Sena, Takahashi
Okada, Endo, Baba, Harada
Sogame, Higashi, Wahib, Alex
Artur, Bofang, Haoyu, Matsumura
Tsuchikawa, Yashima

Avail. software stack: gitlab.com/domke/t2hx