Co-design with Proxy-Apps: A match made in heaven?

EU-SW-CoDesign WS @ISC'22 – 02. June 2022

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

Motivation – Heterogeneous Future

- The “Cambrian explosion” was just the start! (50+ in this fig.)

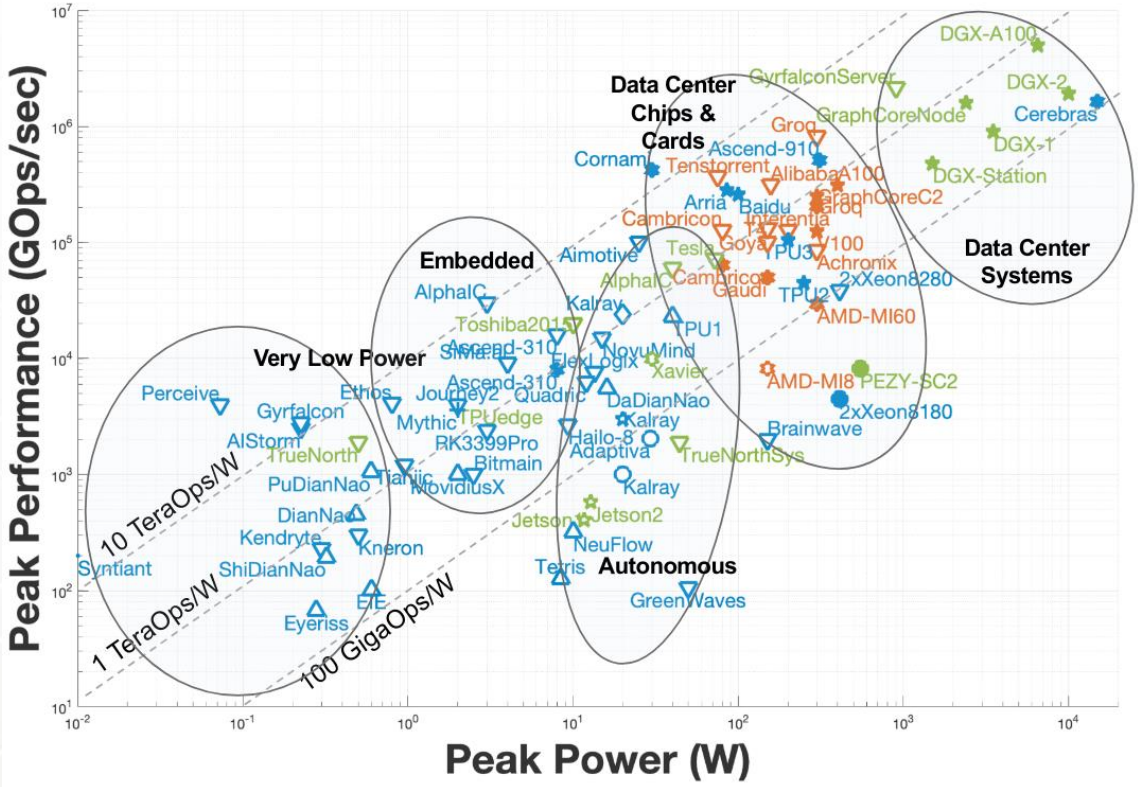


Fig. 2. Peak performance vs. power scatter plot of publicly announced AI accelerators and processors.

Source:
Albert Reuther, et al.
“Survey of Machine Learning Accelerators”, 2020

Motivation – Heterogeneous, Diverse Future

- Increase on almost all fronts:
  - Specialized chips in phones, FPGAs in Intel/AMD CPUs, …
  - Number of programming languages (Rust, Julia, …) and paradigms (CUDA, HIP, oneAPI, Kokkos, RAJA, …)
  - New workloads: containerization, DL/ML, big data, workload-chaining, etc.
  - New topologies: HyperX, Slimfly, Dragonfly, Megafly, …

➔ How to make sense of it all? (and fast)

➔ How to determine the best architecture per site?

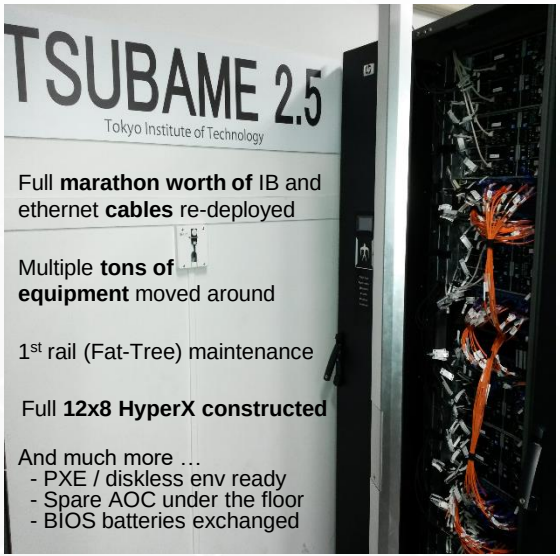
Proxy-Apps: What are they good for? Traditional approach…

Computer simulations create the future

“Use benchmark X and run workload Y and report back.”

--HPC procurement

Opportunity for new topologies – HyperX



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

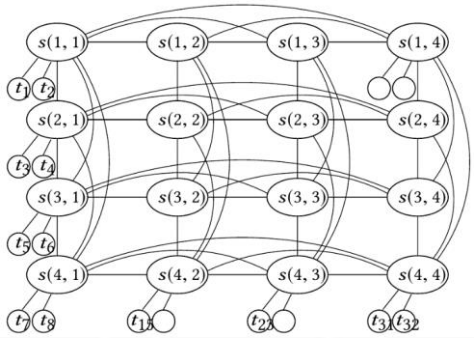


Fig.1: HyperX with n-dim. integer lattice (d_1,…,d_n) base structure fully connected in each dim.

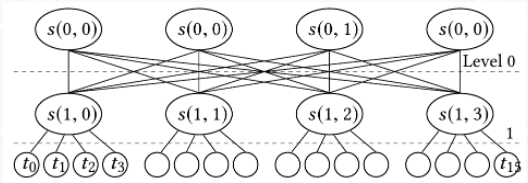
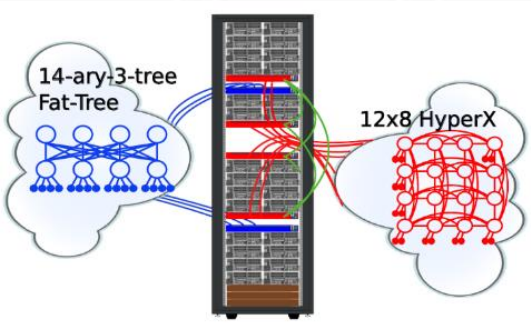


Fig.2: Indirect 2-level Fat-Tree

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**

J. Domke et al. "HyperX Topology: First at-scale Implementation and Comparison to the Fat-Tree"

Opportunity for new topologies – HyperX

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)

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

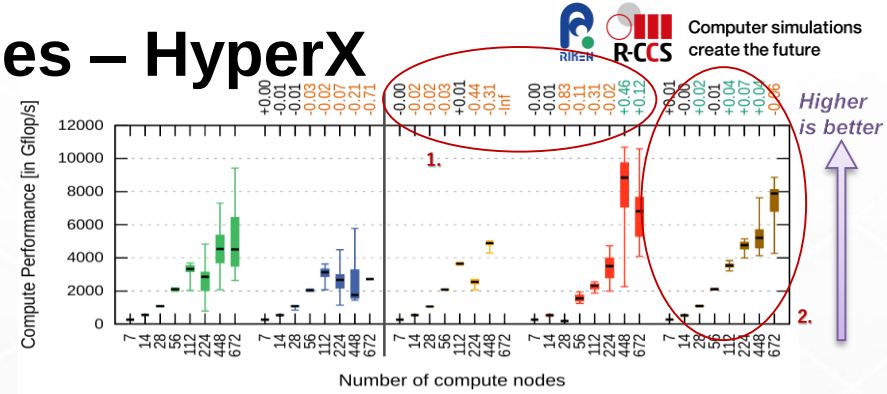


Fig.3: HPL (1GB pp, and 1ppn); scaled 7→672 cn

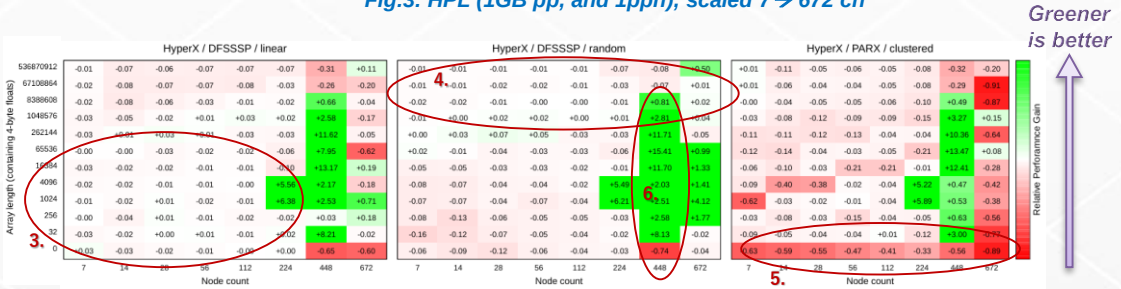


Fig.4: Baidu's (DeepBench) Allreduce (4-byte float) scaled 7→672 cn (vs. "Fat-tree / ftree / linear" baseline)

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*) !

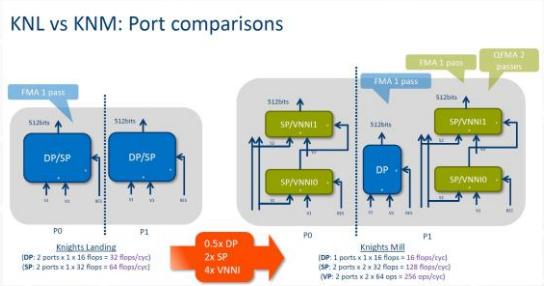
Proxy-Apps: Motivating vendors / domain scientists…

Computer simulations create the future

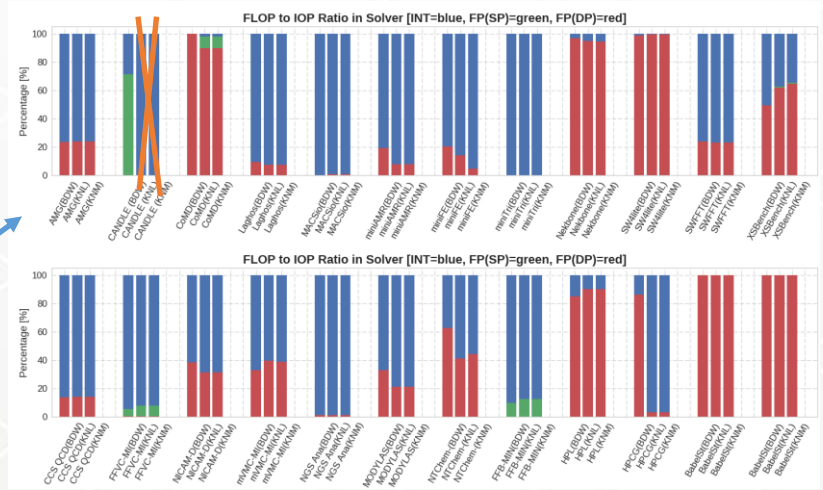
“Wanna do HPC? Then you need many and fast FP64 Units”

--most HPC beginner classes

More Flop/s ➔ more science?!

- Thanks to the (curse of) the TOP500 list, the **HPC community (and vendors) are chasing higher FP64 performance**, thru frequency, SIMD, more FP units, …
- **Motivation:**
  - **Less FP64 units**
    - **Saves power**
    - **Free chip area (ex: FP16)**
    - **Less divergence of "HPC-capable" CPUs from mainstream processors**
- Investigating many proxy-applications:
  - 

  - %FP32 vs. %FP64 vs. %Integer
    ➔ Integer (+DP) heavy (>50%; 16 of 22)
    ➔ Only 4 w/ FP32
    ➔ Only 1 mixed precision



J. Domke et al. "Double-precision FPUs in High-Performance Computing: an Embarrassment of Riches?"

Compare Time-to-Solution in Solver

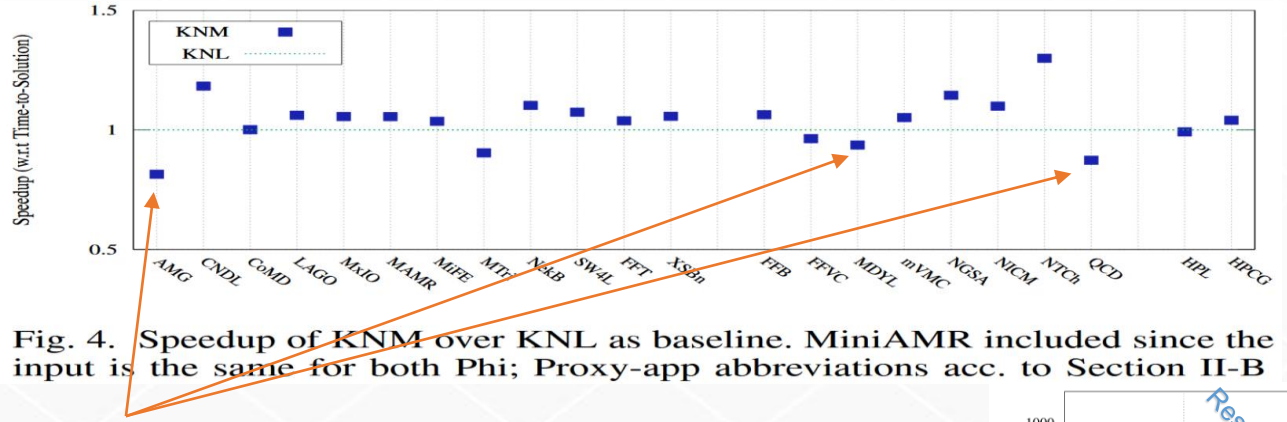
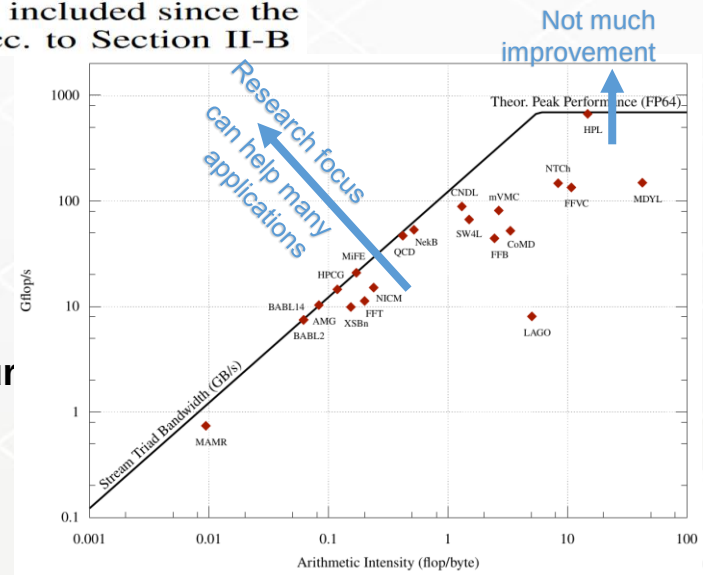


Fig. 4. Speedup of KNM over KNL as baseline. MiniAMR included since the input is the same for both Phi; Proxy-app abbreviations acc. to Section II-B

- **Only 3 apps seem to suffer from missing FP64 unit** (MiniTri: no FP; FFVC: only int+FP32)
- Options for **memory-bound** applications (almost all):
  - **Invest in memory-/data-centric architectures**
  - **Move to FP32/mixed precision ➔ less memory pressur**
- Options for **compute-bound** applications:
  - **Brace for less FP64 units** (driven by market forces) and less “free” performance (10nm, 7nm, 3nm, …then?)



Proxy-Apps: Influencing architecture…

Computer simulations create the future

“Wanna do HPC? Then you need fast [S/D]GEMM.”
--every HPC beginner class

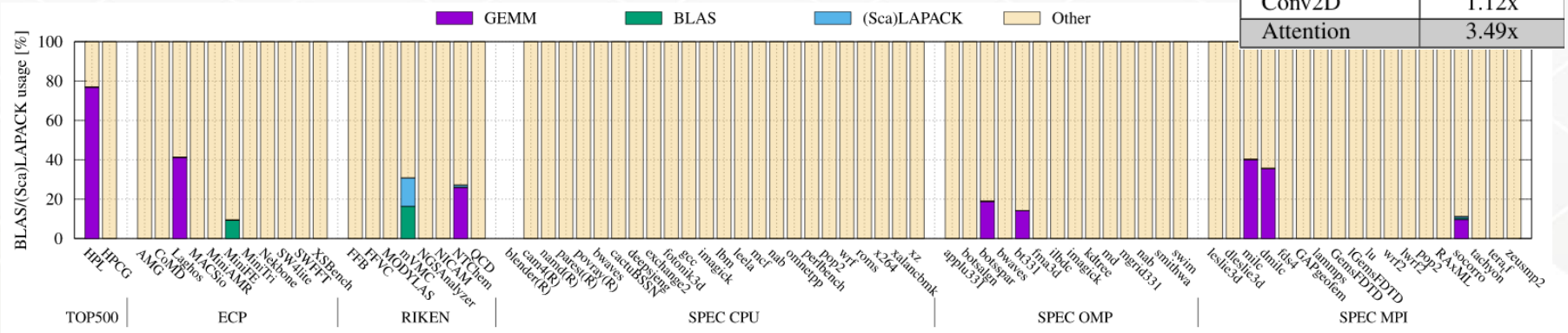
BLAS / GEMM utilization in HPC Applications

Computer simulations create the future

J. Domke et al. "Matrix Engines for High Performance Computing: A Paragon of Performance or Grasping at Straws?"

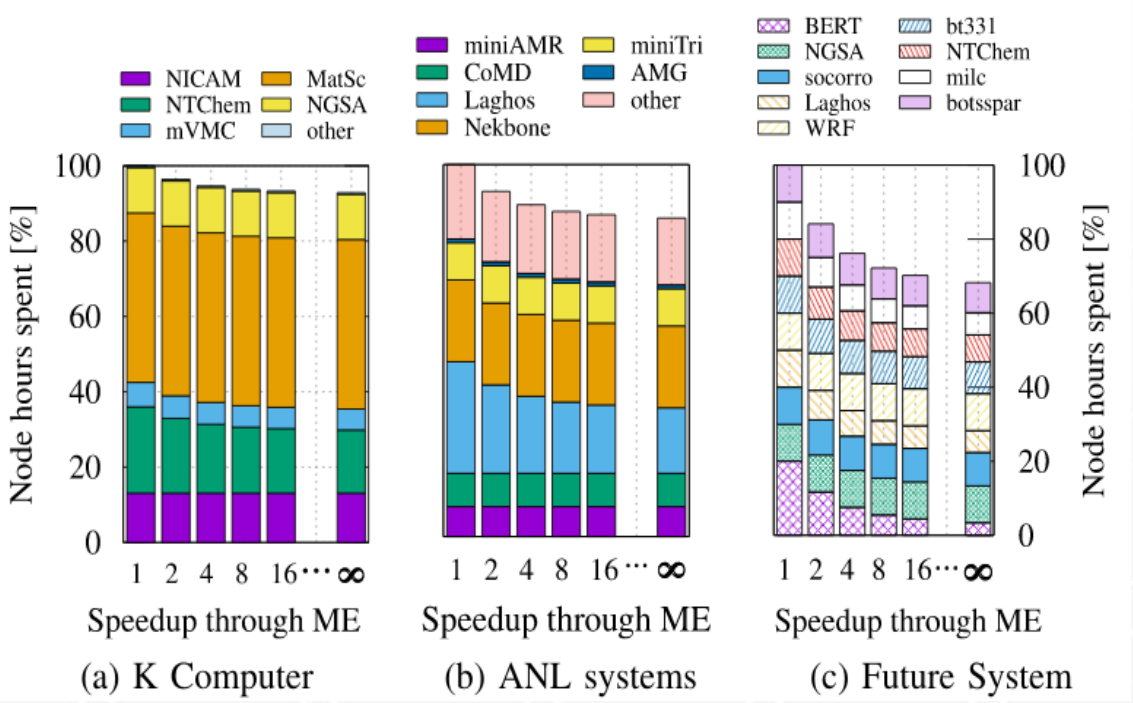
- Analyzed various data sources:
  - **Historical data from K computer**: only 53,4% of node-hours (in FY18) were consumed by applications which had GEMM functions in the symbol table
  - **Library dependencies**: only 9% of Spack packages have *direct* BLAS lib dependency (51.5% have indirect dependency)
  - **TensorCore benefit for DL**: up to 7.6x speedup for MLperf kernels
  - **GEMM utilization in HPC**: sampled across 77 HPC benchmarks (ECP proxy, RIKEN fiber, TOP500, SPEC CPU/OMP/MPI) and measured/profiled via Score-P and Vtune

Benchmark	Speedup
BERT	3.39x
Cosmoflow	1.16x
VGG16	1.71x
Resnet50	1.97x
DeepLabV3	1.75x
SSD300	1.78x
NCF	0.97x
GEMM	7.59x
GRU	3.67x
LSTM	5.69x
Conv2D	1.12x
Attention	3.49x



Estimated Benefit by MEs for HPC Centers

- Thought experiment: Assume we have/had GEMM units in past or future systems.
  - Known: node-hour by domain
  - Sample application with highest BLAS utilization
  - Estimate the node-hour reduction assuming different speedup by ME (2x–8x is realistic dep. on precision)
  - Future system includes 20% DL workloads, other science domains ~10% each



- Results w/ ideal conditions + 4x ME speedup: 5.3% less on K, 10.8% @ANL, 23.8% future system

➔ **HPC can utilize MEs when they come for free, but it's no magic bullet as for DL workloads**

➔ Explore more/other alternatives for Fugaku-next!

Proxy-Apps: Functionality and Regression Testing…

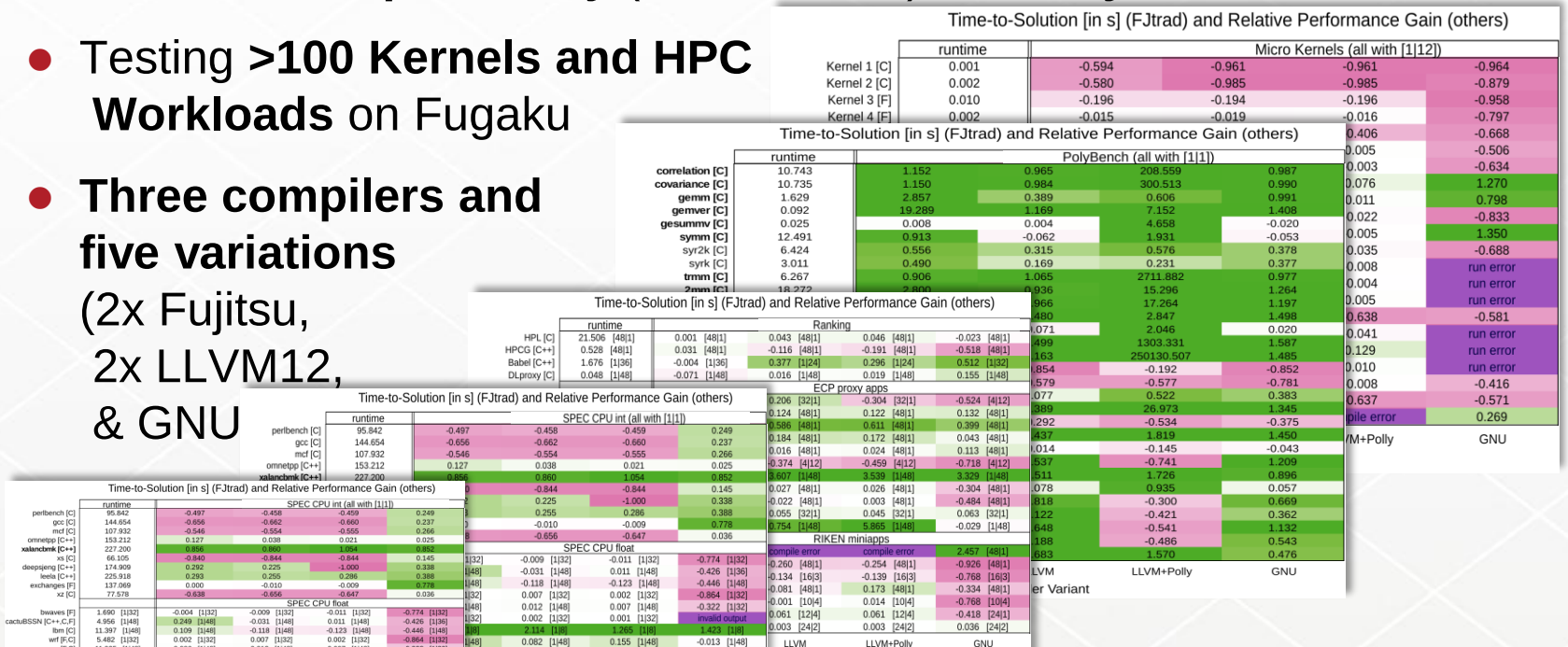
Computer simulations create the future

“Porting an application to A64FX? Just use fcc and –Kfast.”

--Fujitsu

“Silver bullet” compiler choice for A64FX?

- Issue: unexpected advantage of Xeon vs. A64FX in PolyBench
- **Performance portability** (x86→A64FX) **not easy** to achieve
- Testing **>100 Kernels and HPC Workloads** on Fugaku
- **Three compilers and five variations** (2x Fujitsu, 2x LLVM12, & GNU



J. Domke "A64FX – Your Compiler You Must Decide!"

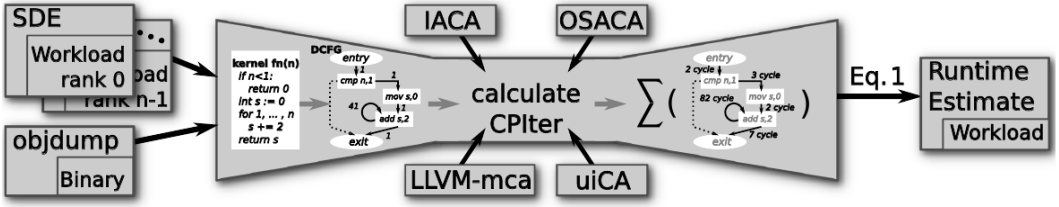
Proxy-Apps: Investigating new CPU architectures…

“Can a simulator handle those complex codes? Surely it must be possible, right, RIGHT?”

--naive me

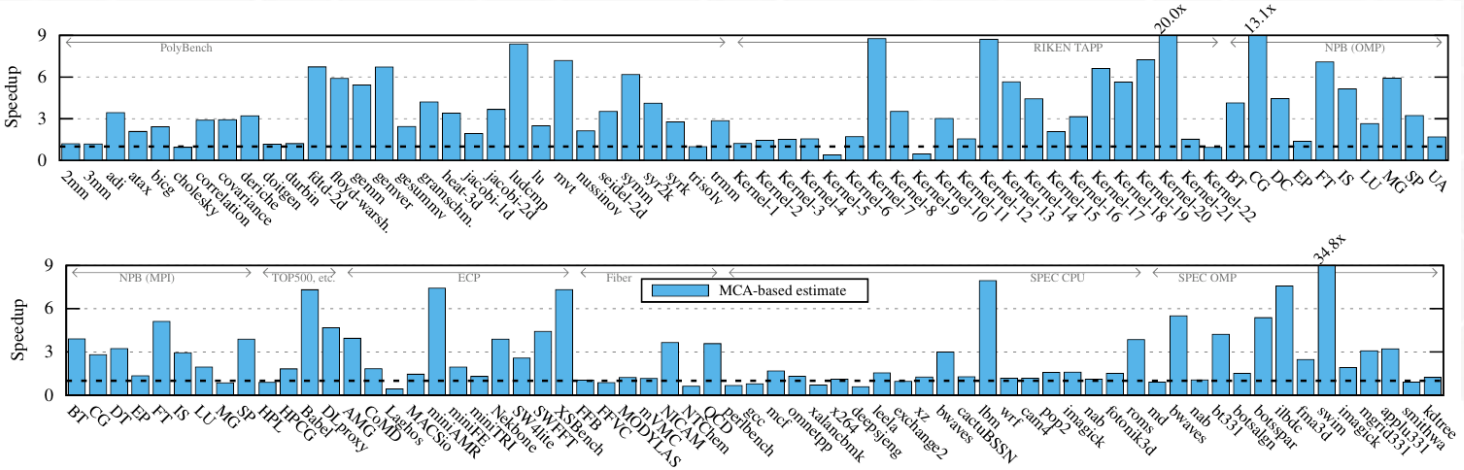
What-if we can fit all data into L1D?

Machine-Code-Analyzers (MCA)

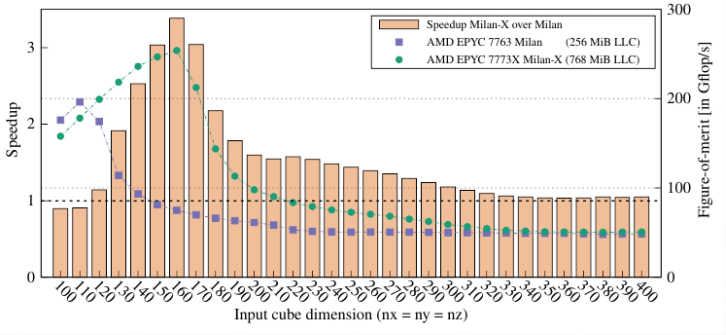


- Estimate the throughput of proxy-app/workload's basic blocks
- MCA tools ignore memory subsystem (assume optimistic L1 hits for all load/store ops)
- Build a BB graph and estimate the runtime ➔ equiv. to "infinitely" large L1 ➔ "speed of light" simulation which trades accuracy for sim. speed (only ~1000x slowdown)

➔ High speedup potential for almost all proxy and micro-bench. (again: HPC is memory-bound)



LARge Cache processor w/ 3D-stacked SRAM

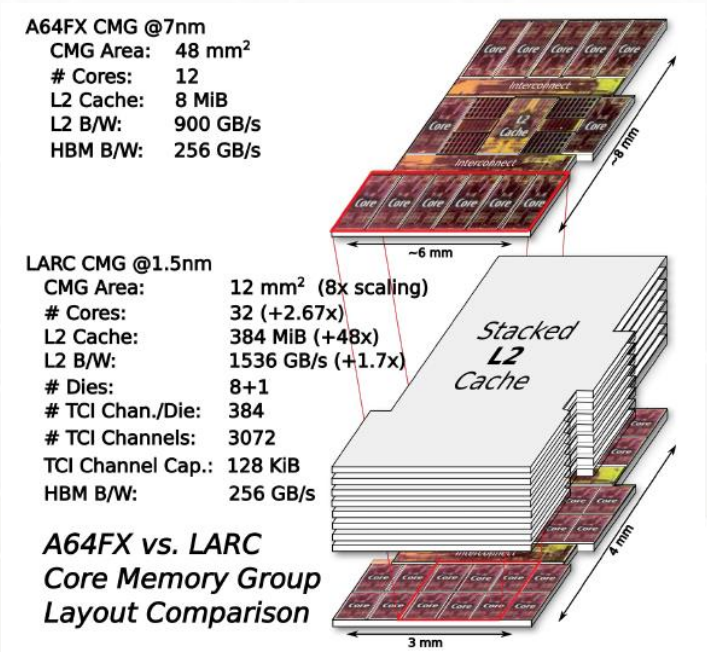


Motivating State-of-the-Art

- **MiniFE on AMD Milan vs Milan-X**
- **>3x speedup from larger LLC (L3) when problem fits into Milan-X' 768MB cache**

What-If extrapolation:

- Assuming we can 3D-stack **SRAM with up to 8 layers** on top or below cores
  ➔ projected L2 cache size & L2 bandwidth?
  ➔ projected performance gain for real apps?
- Explore ~7 years into the future for **1.5nm** fab technology
- Reclaim L2 area to add more cores



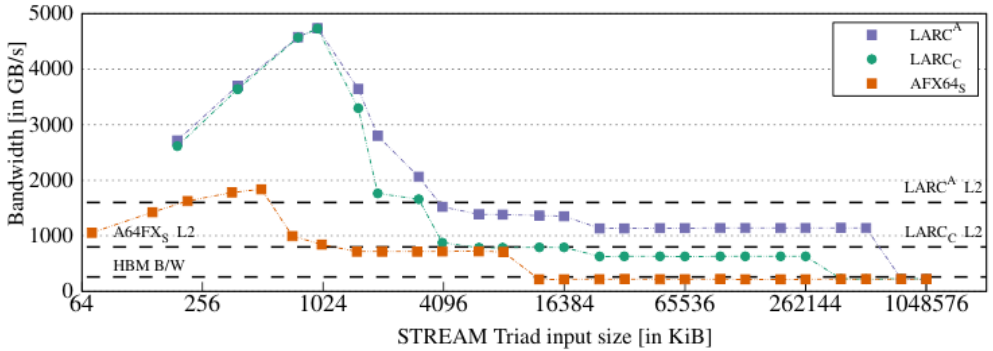
Minor alterations due to gem5 limitations

LARC' CMG not easily replicated

- Configs.: $LARC_c$< LARC < $LARC^a$
- Cycle-level accurate **simulation w/ gem5** (patches to fix bugs and scale cores)
- Base: **1 CMG of Fugaku's A64FX**
- Additional: $A64FX^{32}$ to separate effect of core increase from cache increase
- Only single-rank proxy-apps ☹

TABLE I
CHIP AREA AND SIMULATOR CONFIGURATIONS FOR GEM5

Configuration	$A64FX_S$	$A64FX^{32}$	$LARC_C$	$LARC^A$
Area (per CMG)	48 mm^2	N/A	12 mm^2	12 mm^2
Cores (per CMG)	12	32	32	32
Core config.	Arm v8.2 + SVE, 512bit SIMD width, 2.2 GHz OoO 128 ROB entries, dispatch width 4			
Branch pred.	Bi-mode: 16K global predictor, 16K choice predictor			
Per-core L1D	64KiB 4-way set-assoc, 3 cycles, adjacent line pref.			
L2 cache size/BW (per CMG)	8 MiB ~ 800 GB/s		256 MiB ~ 800 GB/s	512 MiB ~ 1600 GB/s
L2 config.	16-way set-assoc, 37 cycles, inclusive, 256B block			
Aggregated L2 size/BW (per CPU)	32 MiB ~ 3.2 TB/s		4096 MiB ~ 12.8 TB/s	8192 MiB ~ 25.6 TB/s
Main Memory	32 GiB HBM2, 4 channels, 256 GB/s			



Validation tests with STREAM Triad:

🡐 Expected behavior in L1, L2, and HBM

J. Domke et al. "At the Locus of Performance: A Case Study in Enhancing CPUs with Copious 3D-Stacked Cache"

Many patches, crashes, and moons later…

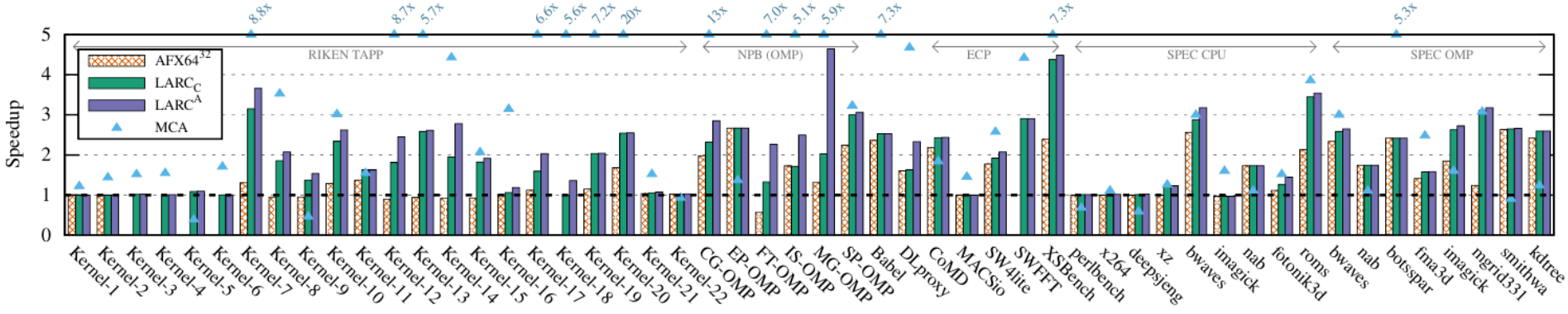
Computer simulations create the future

6-months of gem5 runs on a cluster yield:

- **29 of 52** apps show **≥2x speedup on LARC^{A}** compared to A64FX^{S} CMG (on 1/4th the area)
- For **23 of 29** the gain is **attributed to larger L2**
- Overall: **we estimate *(GM=)*9.77x speedup** for those proxy-apps **for a 16-CMG LARC CPU with 8GiB stacked L2** and 512 cores
- **App-specific tuning** for large cache ➔ even more possible (s. Ltaief et al. in SC'21)

TABLE II
L2 CACHE-MISS RATE [IN %] OF REPRESENTATIVE PROXY APPLICATIONS

Proxy-App	A64FX$_S$	A64FX^{32}	LARC$_C$	LARC^A
Kernel 12	36.6	47.6	10.5	9.1
Kernel 17	46.7	49.5	48.7	34.8
Kernel 19	73.8	69.6	49.1	48.9
FT-OMP	11.6	48.2	6.4	3.8
MG-OMP	59.8	70.9	29.4	0.4
XSBench	32.1	36.4	0.1	0.1



Proxy-Apps: Few years of user experience …

Computer simulations create the future

“Uff... (to say it politely).”
--me

Fugaku Enhancement & Co-Design for Future

- Superseding current proxy-apps: ***Octopodes***
  - Downsides w/ Fiber/proxy-apps (s. Fugaku R&D)
  - On-going collaboration / brainstorming phase with DOE labs (position paper release in Apr.'22)
  - Set of highly-parameterizable, easily-amendable, MOTIF-like problem representations
  - **Common "language" between HPC users, system operators, co-designers, and vendors** to describe the to-be-solved scientific problems: What needs to be computed, and how it can be computed?

https://www.pinterest.fr/pin/145170787976811341/

➔ Apply ML to identify, parameterize, and categorize compute phases

S. Matsuoka, J. Domke, M. Wahib, A. Drozd, A. Chien, R. Bair, J. S. Vetter, J. Shalf
"Preparing for the Future –Rethinking Proxy Applications"
to appear in Computing in Science & Engineering

Summary and Call for Co-Design Collaboration

- **Better & versatile simulators**
  - Consolidating and enhancing existing infrastructure
- More **testbeds of diff. scales**
  - Some netw. issues manifest at scale
  - Repurpose decommissioned system
  - Shared access
- **Better performance metrics** reflecting real-life **and tools** to collect them
  - Focus on more async., automated, easy-to-use for admins
- Cleaner **build environments** and user training
- Extensive **network co-design**
  - More insight into apps/workloads
- More **bottleneck analysis**
  - Fix the lack for memory-centric tools
- **1!** (better) **community test suite**
  - Easy migration, inter-op. testing, regression testing, etc.
- **Octopodes for co-design ☺**

Lots of exciting R&D challenges ahead!

Job & Collaboration Opportunities

Computer simulations create the future

- Collaborations and job opportunities:
  - **We are hiring!** Check out our research teams and open positions:
    https://www.riken.jp/en/research/labs/r-ccs/ and
    https://bit.ly/3faax8v
    < jens.domke@riken.jp >
- Internship/fellowship for students (Bachelor→PhD):
  - Fellowship: https://www.riken.jp/en/careers/programs/index.html
  - Internship: https://www.r-ccs.riken.jp/en/about/careers/internship/
- Supercomputer **Fugaku**:
  - Apply for node-hours:
    https://www.r-ccs.riken.jp/en/fugaku/user-guide/
  - Interactive, virtual tour:
    https://www.r-ccs.riken.jp/en/fugaku/3d-models/ and
    https://www.youtube.com/watch?v=f3cx4PGDGmg