From static, one-off System Evaluation to Continuous Benchmarking … and let’s retire HPL

SC22 Dallas, TX | hpc accelerates.

16.Nov. ’22 Birds of a Feather:

HPC System Test: Looking ahead to Post-Exascale Systems and HPC Ecosystems

Jens Domke Dr. rer. nat. jens.domke@riken.jp
Supercomputing Performance Research Team, RIKEN R-CCS, Kobe, Japan

Motivation – Heterogeneous, Diverse Future

Computer simulations create the 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?

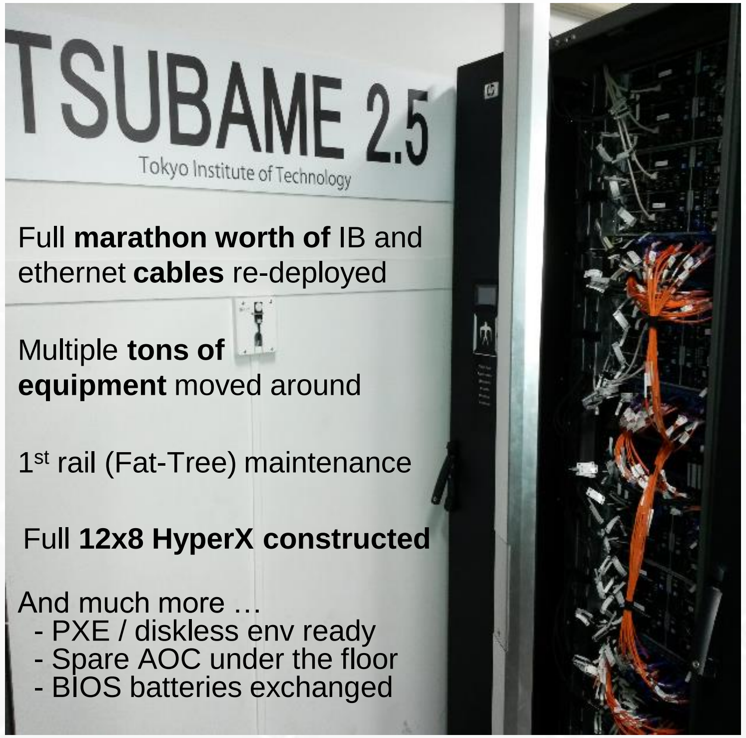
Proxy-Apps: What are they good for? Traditional approach…

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!**

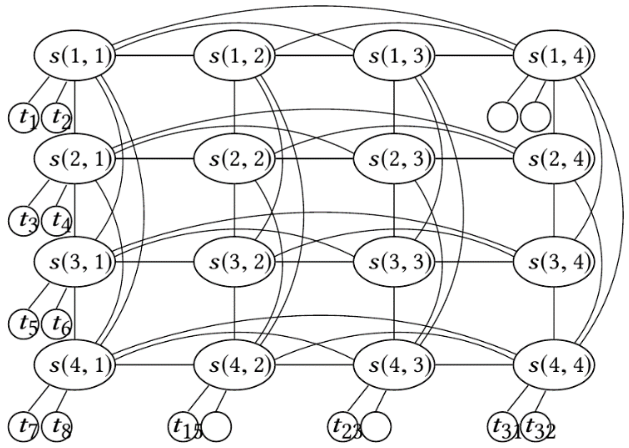


Fig.1: HyperX with n-dim. integer lattice (d_1,…,d_n) base structure fully connected in each dim.

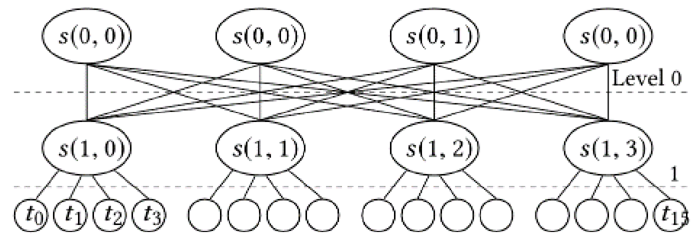
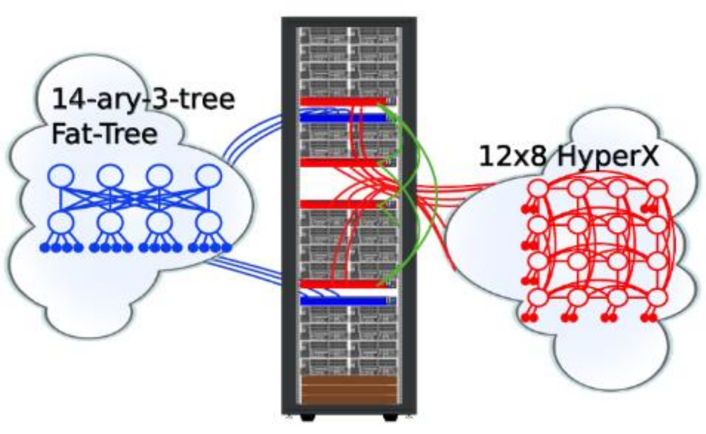


Fig.2: Indirect 2-level Fat-Tree

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**



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

Theoretical Advantages (over Fat-Tree)

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

Proxy-Apps: Motivating vendors / domain scientists…

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

--most HPC beginner classes

More Flop/s ➔ more science?!

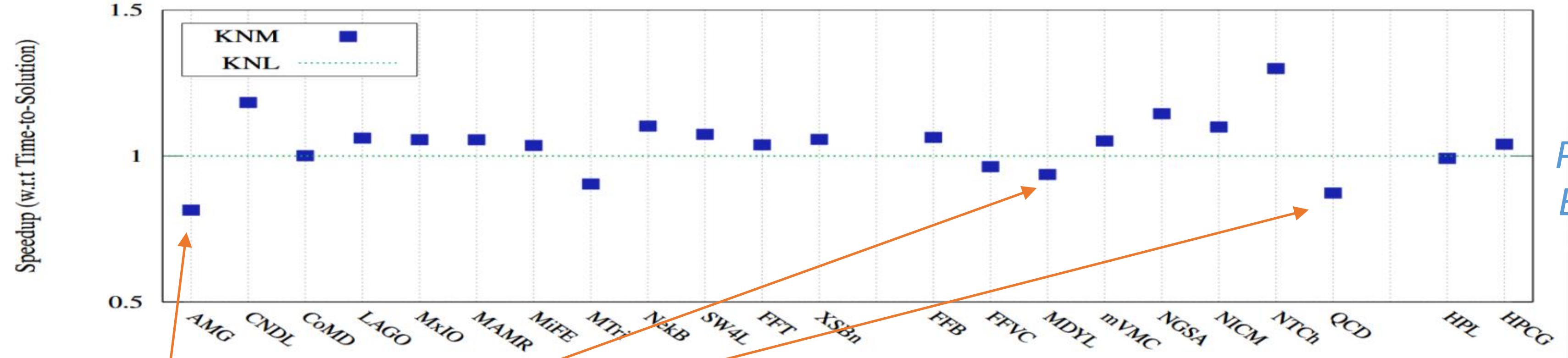
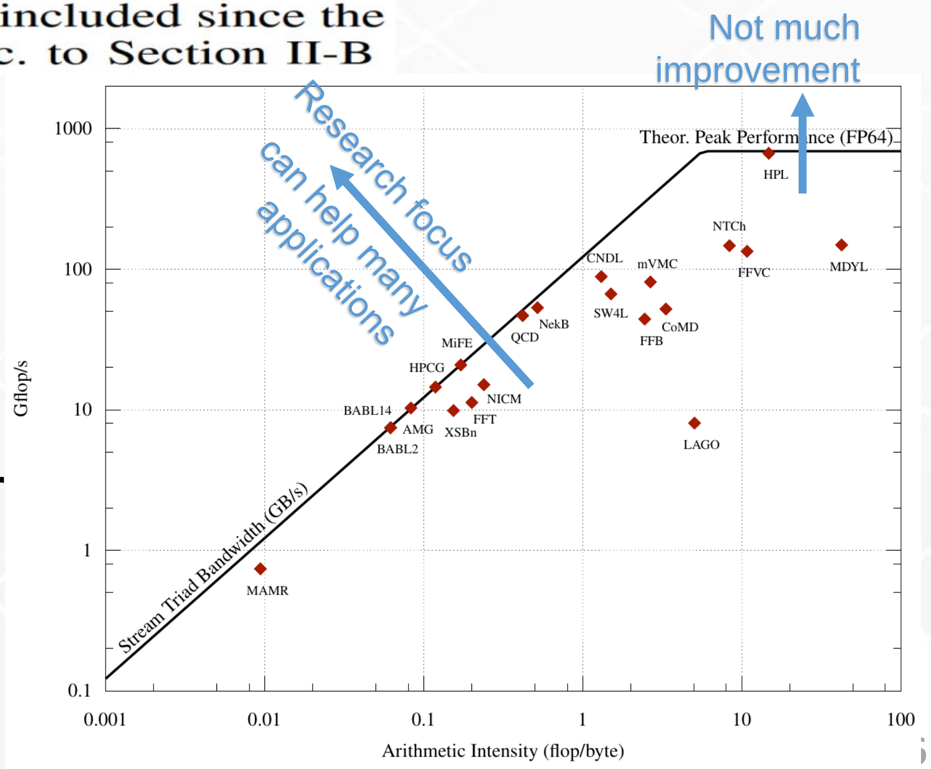


Fig. 4. Speedup of KNM over KNL as baseline. MiniAMR included since the input is the same for both Phi; Proxy-app abbreviations acc. to Section II-B

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

- **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 pressure**
- 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…

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

--every HPC beginner class

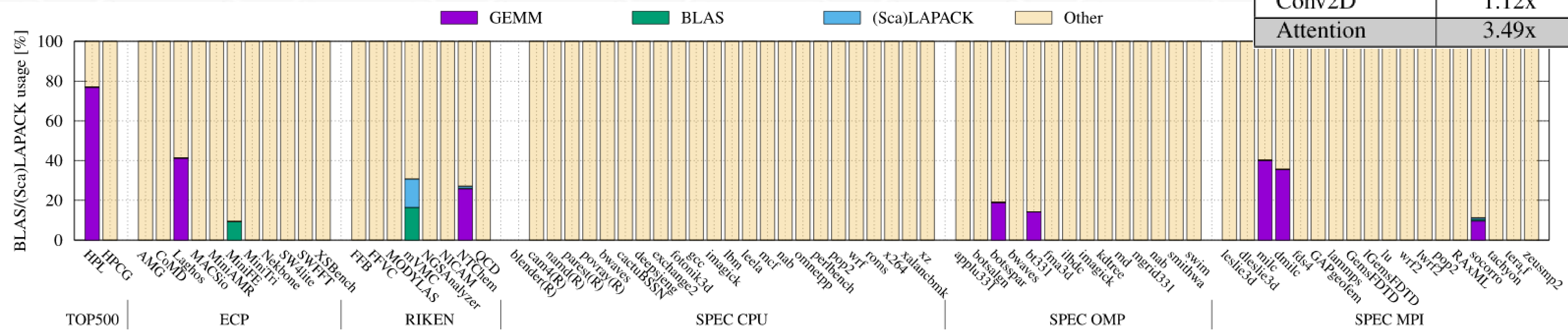
BLAS / GEMM utilization in HPC Applications

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



Proxy-Apps: Functionality and Regression Testing…

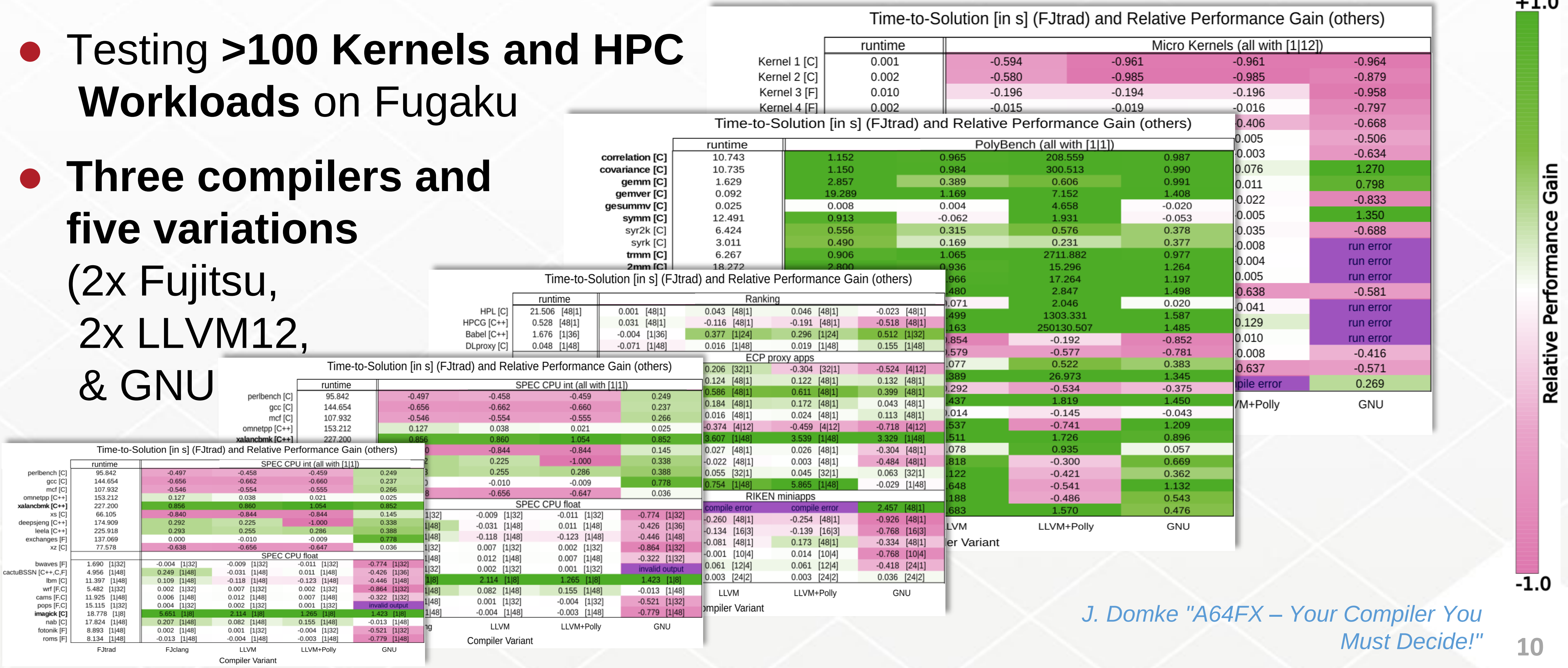
Computer simulations create the future

“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…

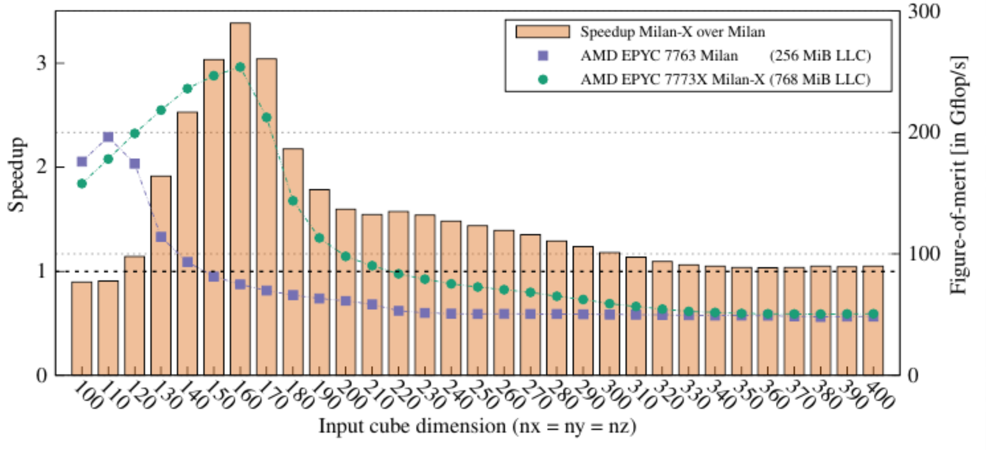
Computer simulations create the future

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

--naive me

LARge Cache processor w/ 3D-stacked SRAM

Computer simulations create the future



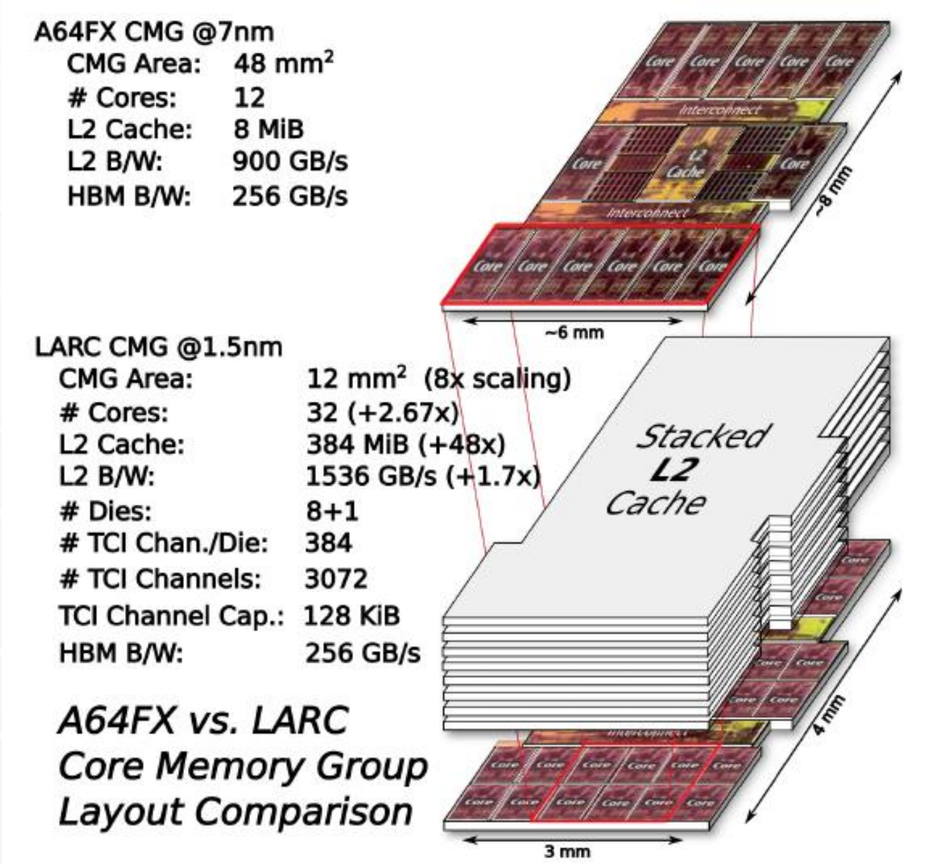
Motivating State-of-the-Art

- **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

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



Proxy-Apps: Few years of user experience …

“Uff... (to say it politely).”

--me

Drowning in overlapping Proxy-Apps (>>100)

- HPC challenge benchmark (HPCC)
- Exascale Computing Project (**ECP**) Proxy Applications
- Center for Efficient Exascale Discretizations (**CEED**) Miniapps
- DOE's **CORAL-2** Benchmarks and RIKEN's **Fiber** / TAPP
- European Union's PRACE Unified European Application Benchmark Suite (**UEABS**)
- **SPEC**, BenchCouncil, Intel's **HiBench**, DeathStarBench, Baidu's DeepBench, and MLCommons' **MLPerf**

cos θ-similarity for **cache-related** *perf. counters*

	ExaMiniMD	LAMMPS	MiniQMC	QMCPack	sw4lite	sw4	SWFFT	HACC	pennant	snap
ExaMiniMD	0.00	5.02	54.54	38.73	11.70	12.49	6.58	6.38	13.21	7.13
LAMMPS	5.02	0.00	54.69	38.62	15.66	16.27	4.87	6.38	13.60	10.88
MiniQMC	54.54	54.69	0.00	17.15	47.12	46.08	50.02	48.98	42.16	49.15
QMCPack	38.73	38.62	17.15	0.00	32.64	31.67	33.92	32.94	26.29	33.78
sw4lite	11.70	15.66	47.12	32.64	0.00	1.15	13.41	11.40	11.15	5.07
sw4	12.49	16.27	46.08	31.67	1.15	0.00	13.74	11.70	10.69	5.69
SWFFT	6.58	4.87	50.02	33.92	13.41	13.74	0.00	2.24	9.09	8.80
HACC	6.38	6.38	48.98	32.94	11.40	11.70	2.24	0.00	7.86	6.87
pennant	13.21	13.60	42.16	26.29	11.15	10.69	9.09	7.86	0.00	9.37
snap	7.13	10.88	49.15	33.78	5.07	5.69	8.80	6.87	9.37	0.00

Source: D. Richards et al. "Best Practices for Using Proxy Apps as Benchmarks"

- **Traditional**: HPL, HPCG, stream, Graph, and Intel's IMB, …

Lessons-Learned from using Proxy-Apps

- Inherently **complex** and **implementation biases**
  - Mostly implemented in Fortran and C[/C++], and tuned over the years
  - Highly tuned CUDA (and “legacy” CPU) from ECP efforts
- Huge porting & **maintenance overhead**
  - Macros, separate code paths, hand-written makefiles, …
  - Costly refactoring for: data layouts, parallelization strategies, accelerators
- Insufficient **inputs, testing, documentation**
  - Issues with non-std compilers and applying perf. analysis tools
  - Strong- and weak-scaling, varying input sizes, independent of #MPI/#OMP
- Lack of efficiency reporting
  - *How much performance achieved vs. the peak theoretical performance?*

Let's clean up this mess …

"Octopodes to the rescue."
--RIKEN & DOE

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?

Source: 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

Example of one Octopode: Matmul

- Input **shapes**: such as squared, rectangular, and tall/skinny
- Various numerical **precisions** (i.e., from fp128 to bfloat16, etc)
- **Batched** and non-batched executions modes
- **Dense** matrix-matrix operations, matrix-vector, sparse matrices
- **Sparse** matrix: random, realistic blocks, *Matrix Market*

➔ Use C++ templating to generate as many variants as possible to train ML models

➔ One Octopode for each distinct compute phase or math kernel

➔ In-between Berkeley MOTIFs and Proxy-Apps

Usage of Octopodes for Co-Design

- "What needs & how can it be computed" <u>not</u> "Here is how you have to do it"
- For **performance modeling** of real workloads: identify compute phases which can be mapped to one or more Octopodes ➔ combine perf. model of the 'easier to understand' Octopodes ➔ approx. perf. model of full workloads
- For **vendors**:
  - Allowed tuning freedom for the Octopodes, i.e., changes of algo., implementation, integer/float. precision, data layout, etc., *as long as* intended result is the same
  - Accurately model consumer workloads ➔ Less over/under-selling of hardware
- **Porting** of user codes to new system:
  - Act as demonstrator for users to show how to port
  - ML/AI to identify phases can be used as helper for porting of real codes
- **Better suited for co-design tools**, e.g. compiler tests, regression testing, simulators (gem5/SST/CODES/…), quick "What-If" tools, etc.

Summary and Call for Co-Design Collaboration

Computer simulations create the future

Octopodes will be the common language between HPC users, system operators, co-designers, and vendors to describe the to-be-solved scientific challenges, what needs to be computed, and how it can be computed, in an abstract way.

Long: arxiv.org/abs/2204.07336

Short: ieeexplore.ieee.org/document/9789513

EDITORS: Kathryn Mohror, mohror1@llnl.gov
John M. Shalf, jshalf@lbl.gov

DEPARTMENT: LEADERSHIP COMPUTING

Preparing for the Future—Rethinking Proxy Applications

Satoshi Matsuoka, Jens Domke, Mohamed Wahib, and Aleksandr Drozd, *RIKEN Center for Computational Science, Kobe, 650-0047, Japan*

Andrew A. Chien and Raymond Bair, *Argonne National Laboratory, Lemont, IL, 60439, USA*

Jeffrey S. Vetter, *Oak Ridge National Laboratory, Oak Ridge, TN, 37831, USA*

John Shalf, *Lawrence Berkeley National Laboratory, Berkeley, CA, 94720, USA*

A considerable amount of research and engineering went into designing proxy applications, which represent common high-performance computing (HPC) workloads, to co-design and evaluate the current generation of supercomputers, e.g., RIKEN's supercomputer Fugaku, ANL's Aurora, ~~or~~ ORNL's Frontier. This process was necessary to standardize the procurement while avoiding duplicated effort at each HPC center to develop their own benchmarks. Unfortunately, proxy applications force HPC centers and providers (vendors) into an undesirable state of rigidity, in contrast to the fast-moving trends of current technology and future heterogeneity. To accommodate an extremely heterogeneous future, we have to reconsider how to co-design supercomputers during the next decade, and avoid repeating past mistakes.

Supercomputing is the art of mapping a scientific question onto hundreds of trillions or quadrillions of transistors, as in the case of the currently fastest supercomputers in the world, by explo-

and on perfecting component integration to assemble the supercomputers. But the projected end of Moore's law and Dennard's scaling in the early 2000s required a rethinking, culminating in an intensified co-design

Future Directions and Concerns

- Proper benchmark sets and documentation
- Let’s try to retire HPL ☺
- CI/CD/CB and Octopodes
- Be mindful about power/cycles dedicated to benchmarking

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 < jens.domke@riken.jp >
    https://bit.ly/3faax8v
    https://bit.ly/3tLVwBZ ← **Currently hiring for SPR Team!**
- 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