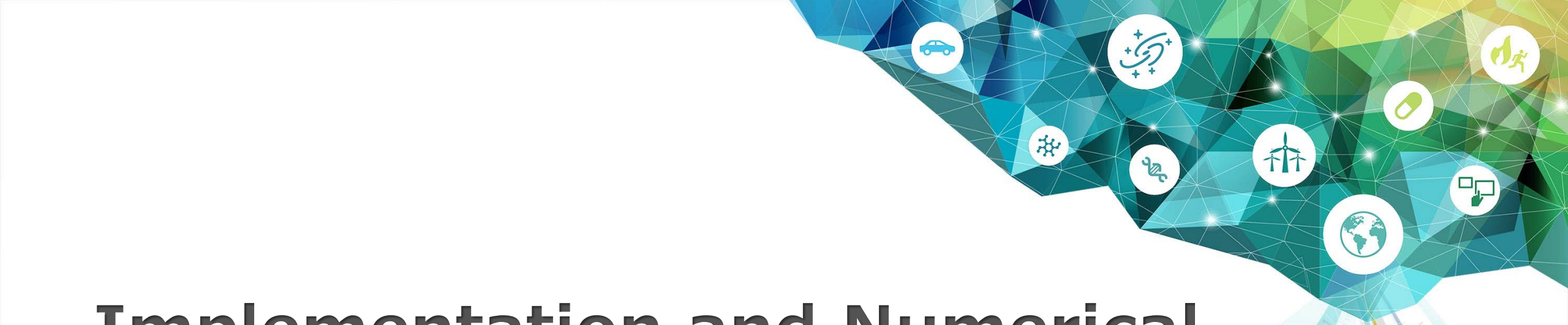


Implementation and Numerical Techniques for One EFlop/s HPL-AI Benchmark on Fugaku

Shuhei Kudo, Keigo Nitadori,
Takuya Ina, and Toshiyuki Imamura*
Center for Computational Science, RIKEN

11th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems

Outline

- **Fugaku and Mixed-precision computing**
  - Fugaku system configurations
  - Mixed-precision on scientific code and benchmark

- **HPL-AI**
  - The rule defined in 2019
  - Preliminary analysis of the benchmark matrix
  - Numerical and Implementation techniques

- **Numerical experiments on Fugaku**
  - Performance analysis
  - Ever largest benchmark result in May 2020

Supercomputer Fugaku

- **A new Japanese flagship supercomputer successor to K**
  - the official launch available for public service is 2021
  - targeting traditional HPC and AI
- **CPU: A64FX (158,976 in total)**

  ARMv8.2+SVE(512bit x 2pipes) +clock(2.0/2.2GHz) and HBM2(32GB, 1TB/s)

  48cores(+2/4 assistant cores)/CPU
- **Interconnect: TofuD**
  - the network interface embedded in the A64FX microprocessor
  - injection bandwidth per node equaling 40.8 GB/s
  - low-latency communication with hardware-offloaded communication capabilities

- **Through benchmark tests**
  - HPL-AI to demonstrate capability for mixed-precision computation

  → Indirectly, new AI workload

	K	Fugaku	Summit
FP64	10.62PF	537PF	200PF
FP32	10.62PF	1.07EF	400PF
FP16	--	2.15EF	3.3EF
INT8	--	4.30EO	--

Mixed-precision on Scientific study

- **For the linear solver**
  - Classical and investigated by Wilkinson (1988) and Moler (1967).
  - Haidar et al. examined the feasibility of FP16 for the mixed-precision solver for well-conditioned matrices (ScalA17, SC18).
  - Yamaguchi et al. Finite Element solver with FP21-32-64 (WACCPD/SC19)
  - Idomura et al. SSOR preconditioner using FP16 (SC20)
- **Applications, Not only FP but also INT**
  - Neural Net Trained to Recognize Extreme Weather Patterns (GBP@SC18)
  - Modified CoMet algorithm on TC's to find fundamental biology (GBP@SC18)
- **Other potential mixed-precision studies, …**
  - For example, Mukunoki et al. Ozaki scheme (JCAM), to emulate FP64 by TensorCores (ISC20)
- **Of course, we can not stop listing up more works in DNL.**

Motivation and Main contributions

- **Almost full system size and speed benchmark on Fugaku**
  - 126,720 nodes (5/6 system), and 91%~>95% of the peak
- **Performance beyond one Exa Flop/s**
  - Achieved World record in floating point benchmarking

- **Potential of Mixed-precision computing FP16-32-64, etc.**
- **Indirect examination to AI workload**
- **Learn of HP-AI benchmark**
  - We learned a lot via preliminary analysis
  - Several techniques (numerical and implementation):
    - Less iteration by single-iteration iterative refinement
    - Numerical stability by Scaling and reordering
    - Reduction of memory footprint
    - HW-supported asynchronous communication

Preliminary analysis of the HPL-AI matrix

- **The HPL-AI rule (Nov. 2019)**

  Solving $Ax = b$ by

  - the LU factorization of a matrix, and
  - the iterative refinement method (IR).

  LU: $A = LU$

  - Must consist of $2/3n^3 + O(n^2)$ flops.
  - **Not critical on accuracy/precision at this stage**

  IR: $x^{(n+1)} = x^{(n)} + A^{-1}(b - Ax^{(n)})$

  - solves the system of linear equations **with 64bit-precision accuracy**
  - tweak the IR algorithm without restriction, but IR should use the LU factors.

- The HPL-harness must be less than 16 until 49 iterations

$$\frac{\|Ax - b\|_\infty}{\|A\|_\infty \|x\|_\infty + \|b\|_\infty} \times (n \cdot \varepsilon)^{-1} < 16$$

- **HPL matrix vs. HPL-AI matrix**

  [HPL] a randomly generated matrix.

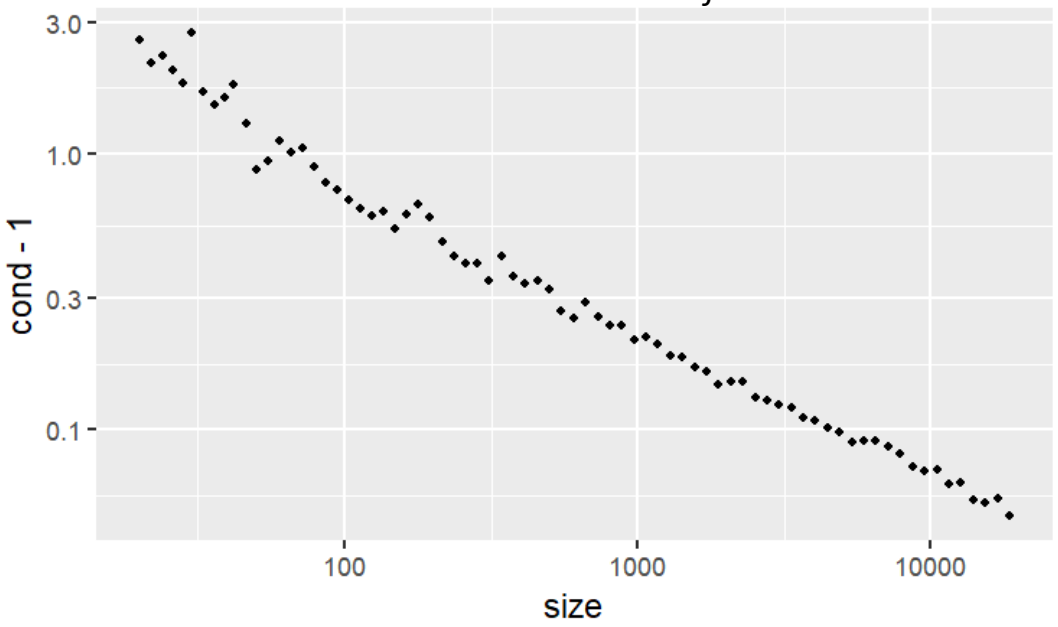
  On the other hand,

  [HPL-AI] an absolute sum of off-diagonals = weakly diagonally dominant

  Diagonal: O(*n*), off-diagonal: O(1)

  → Not necessary pivoting!

Estimated conditioned numbers of HPL-AI matrices



Significant techniques in Numeric

- **Three-precision mixed computing**

Require: An $n \times n$ matrix A, a vector b.
Ensure: The solution of the linear equations x, the LU factors L and U.

preprocess
(FP64) $D \leftarrow \mathrm{diag}(A)$, the diagonal part of A.
(FP64) $x \leftarrow D^{-1}b$, the initial guess.

LU decomp.
(FP32&16) Factorize $A = LU$ using the mixed-precision algorithm.

IR iter.
while the backward-error (HPL-harness) > 16 **do**
  (FP64) $r \leftarrow b - Ax$, the residual.
  (FP64) Solve $LUd = r$.
  (FP64) $x \leftarrow x + d$.
end while

- **Two notes on FP16 implementation**
  - [Scaling]
    - Value range on FP16 arithmetic is quite narrow, thus, we should avoid over/under-flows by appropriate scaling
  - [Ordering]
    - To prevent meaningless operations due to rounding, the order of operations should be devised

IEE754 half precision (binary16/FP16)

Sign 1bit Frac. 10bit

Exp. 5bit

$$O(n) + O(1) \approx O(n) \quad \text{when } n > 2^{10} = 1024$$

Scaling in LU factorization

$$A_{i,j}^{(m)} := s^{-1}\left(-\sum_{k=1}^{m}(sL_i^{(k)})U_j^{(k)}\right) + A_{i,j}^{(0)}$$

$$s = n^{\frac{3}{4}}, \|sL_i^{(k)}\|_{\max} \approx O(n^{-1/4})$$

Basic idea of our ordering in LU factorization

$$A_{i,j}^{(m)} := A_{i,j}^{(0)} - \sum_{k=1}^{m} L_i^{(k)} U_j^{(k)}$$

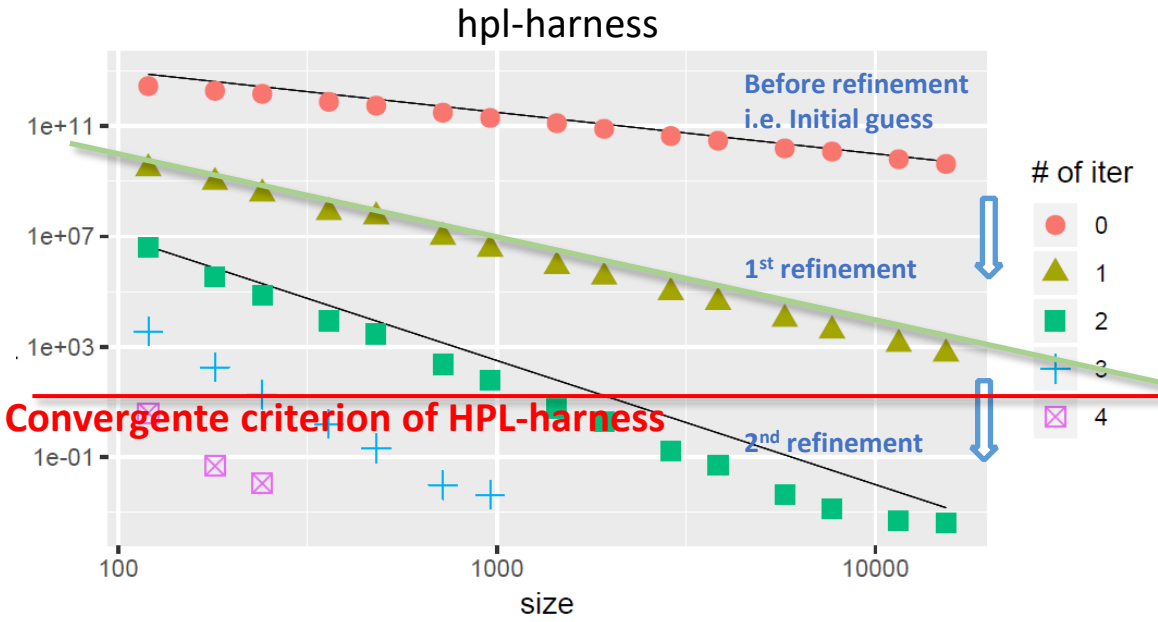
$$\Rightarrow \quad A_{i,j}^{(m)} := \left(-\sum_{k=1}^{m} L_i^{(k)} U_j^{(k)}\right) + A_{ij}^{(0)}$$

Significant techniques in Numeric

- **SIIR (Single-Iteration for IR)**

  Good guess $x_0 := D^{-1}b$ enables to skip IR iterations, if quick factorize (approx.) as $A \approx L_J U_J = ID$.

  Only one iteration is enough, if n>100,000.



- **Lazy initialization (ordering the initial values come last)**

  The large gap between the initial elements and the lower part like

  $$\{\mathrm{diag}\} = O(n), \{\mathrm{others}\} = O(1) \text{ and } \{\mathrm{lower}\} \to O(1/n)$$

  causes information drops in summation when n>2^{24} in FP32 → Even FP16 is much harder.

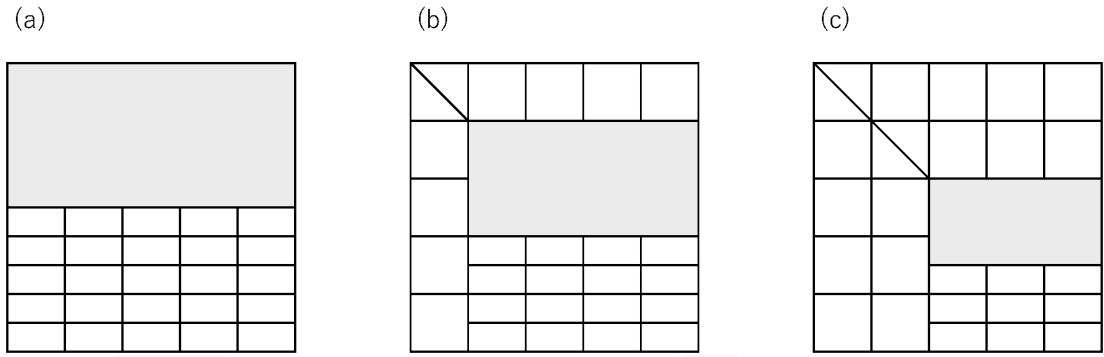
  HGEMM ← A blocked summation approach in FP16 to avoid satiation by accumulating the block result converted to FP32

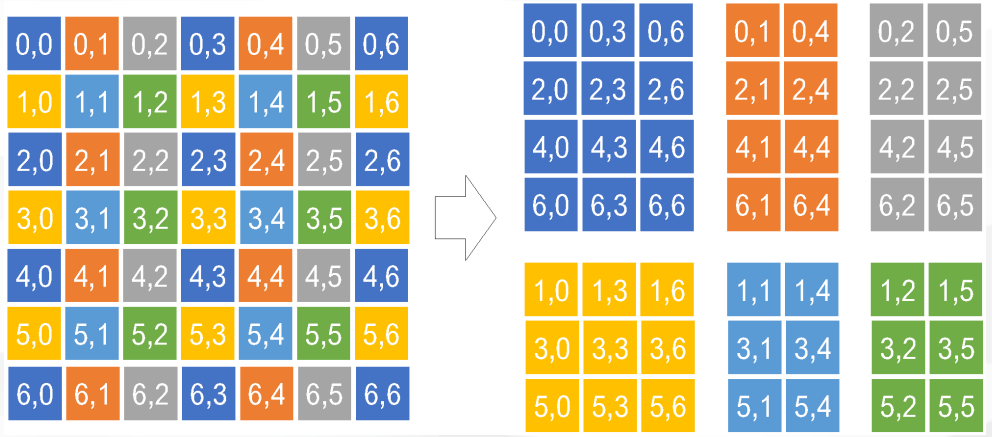
$$A_{I,J}^{(n/b-1)} = \begin{cases} \mathrm{lu}\left(A_{I,I}^{(0)} - \sum_{k=1}^{I-1} A_{I,k}^{(k)} A_{k,J}^{(k)}\right), & \text{if } I = J \\ \left(B_{I,J}^{(I)}\right)^{-1}\left(A_{I,J}^{(0)} - \sum_{k=1}^{I-1} A_{I,k}^{(k)} A_{k,J}^{(k)}\right), & \text{if } I < J \\ \left(C_{I,J}^{(J)}\right)^{-1}\left(A_{I,J}^{(0)} - \sum_{k=1}^{J-1} A_{I,k}^{(k)} A_{k,J}^{(k)}\right), & \text{otherw} \end{cases}$$

$$B_{I,J}^{(I)} = \mathrm{triu}\left(A_{I,I}^{(I)}\right), \text{ and } C_{I,J}^{(J)} = \left(\mathrm{stril}\left(A_{J,J}^{(J)}\right) + I_b\right)$$

Significant techniques in Implementation

- **On-the-fly technique**
  - The matrix elements are generated in the on-the-fly manner during the LU and IR phases to **avoid the memory consumption of the FP64 matrix, and Lazy initialization as well.**
- **Double-decker layout for multi-data formats**
  - The memory regions of the **FP16 and FP32** matrices are laid out in an **overlapping** fashion
- **MPI parallelization with HW-offloaded communications**
  - The patterns of computation and communication are the same as in HPL, except for the non-obligatory pivoting.
  - Most of the communication is asynchronously offloaded to TofuD, thereby extending the overlapping part with a tiny overhead and latency.





Conditions of numerical experiments

- **Limited access during the installation phase of Fugaku.**

1. **Micro Fugaku and Very Early access program** (Dec 2019 to March 2020)
2. **Early access program III:**
   - April to mid May 2020, up to 27,648 nodes.
3. **Open for privileged users**
   - May 8 and 9 and 14 and 15, up to 126,720 nodes.

- Run the code with **2.0GHz (normal mode)** in these periods.
- **4MPI processes per node** → 12threads per 1process
- **Fujitsu compiler and Fujitsu MPI environment**
  - Matrix footprint equal to 23.7GiB (large) or 5.9GiB(tiny).

Node	2D-Shape	Peak(DP)	Peak(HP)	N(tiny)	N(large)	NB
432	24x18	1.33PF	5.31PF	829,440	1,658,880	288/320/576/640
1,728	48x36	5.31PF	21.23PF	1,658,880	3,317,760	↓
6,912	96x72	21.23PF	84.93PF	3,317,760	6,635,520	↓
27,648	192x144	84.93PF	339.74PF	6,635,520	13,271,040	↓

Experiments on smaller cases (NB=576)

- **Two matrix footprint cases**
  - Both: >91% of the peak
  - Efficiencies between large and tiny settings → only 4%
- **Suggestion**
  - **Even if it is a small case, the tiny setting is enough** to confirm the performance of Fugaku, comparing the HPL result on Fugaku (~80%).

Tiny(upper) and Large (bottom)

Node	T (sec)	Pflop/s	Efft %
432	78.6	4.83	91.13
1728	156.7	19.42	91.48
6912	312.6	77.88	91.70
27648	625.0	311.6	91.73

Node	T (sec)	Pflop/s	Efft %
432	603	5.05	95.12
1728	1203	20.24	95.32
6912	2403	81.07	95.45
27648	4807	324.1	95.40

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,299,072	415,530.0	513,854.7	28,335

x 0.80865

https://www.top500.org/lists/top500/list/2020/06/

Preliminary Benchmark result

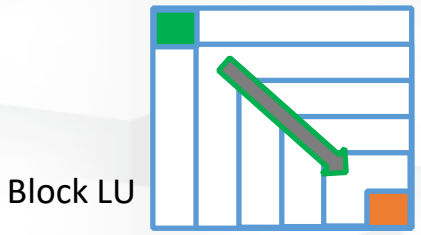
- **Benchmarked on a limited part of Fugaku (2.0GHz normal mode)**

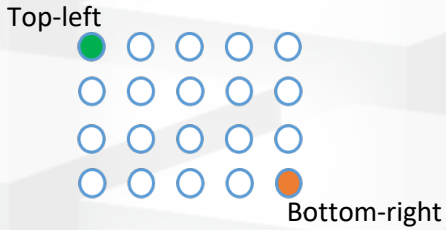
Node	Shape	N(tiny)	N(large)	Block size
6,912	96x72	3,317,760	6,635,520	288/320/ 576/640

77.88PFlop/s
91.70 %

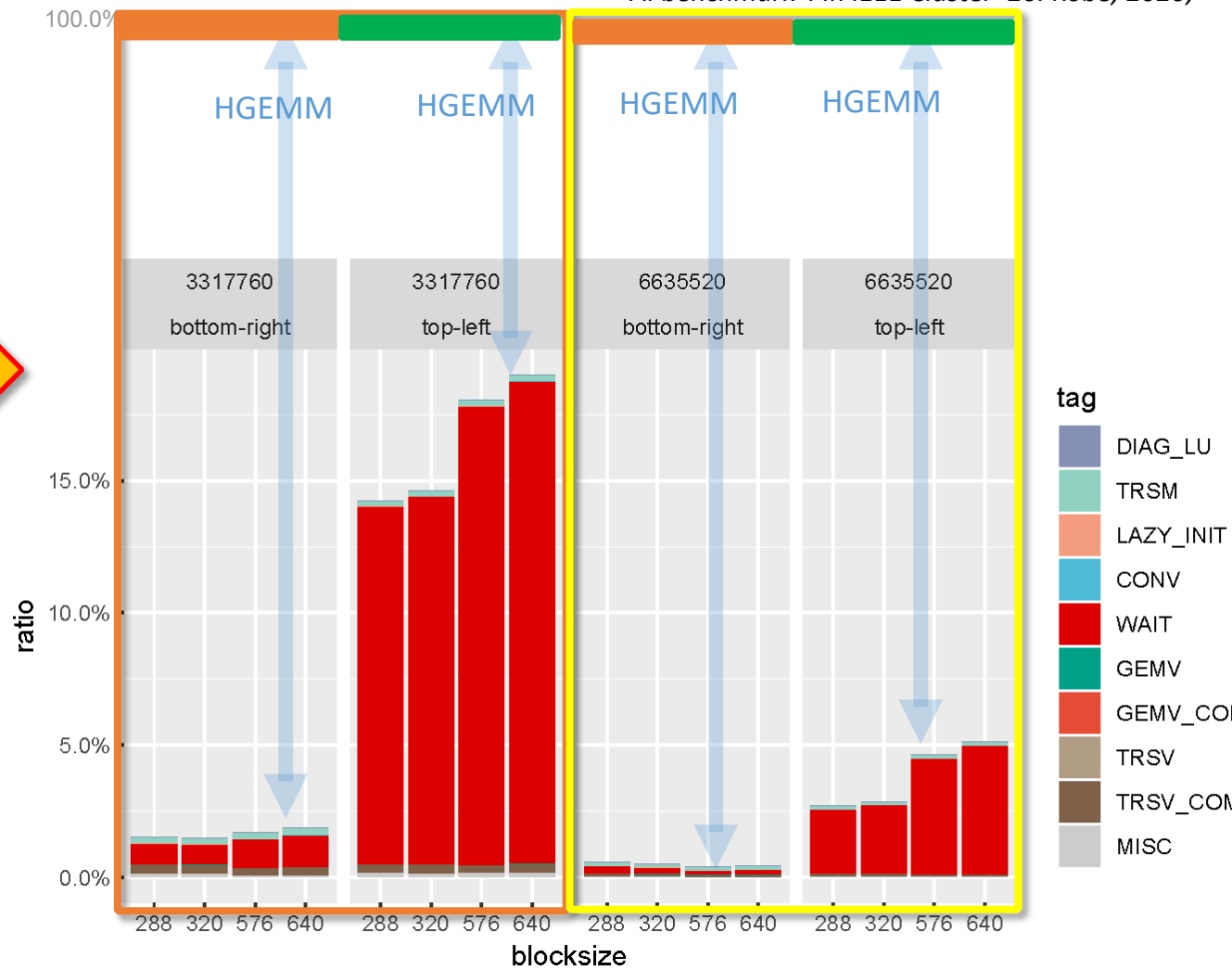
81.07PFlop/s
95.45 %

- **We can read time breakdown:**
  - HGEMM: 82~97%
  - WAIT (load imbalance) 2~18%
    ← depending on the place on PE grid





S. Kudo et al. "Prompt report on Exa-scale HPL-AI benchmark". In IEEE Cluster '20. Kobe, 2020,



Time breakdown, when 6912nodes=331776cores
Tiny: 5.9 GiB (left), Large: 23.7GiB (right), per node.

Top500 #1 record at June 2020

- **1.421 EHFLop/s (approximately 91% of the theoretical peak).**
  - 126,720 nodes of the non-boosted Fugaku system (5/6 of the full system)
  - The target matrix size is 13,516,800, and the computing time required was 1,158 sec.
  - Scored more than 2.5 times of that on the Summit system.
- **Ever fastest benchmark result, and**
- **Big epoch in the HPC community**
  - The world's first achievement to exceed the wall of exascale in a floating-point arithmetic benchmark.

```
!REMAP FOR 330 RACKS
jobid=567937
n=13516800 b=320 r=1056 c=480
2dbc lazy rdma full pack
numasize=4 numamap=CONT2D nbuf=3
epoch_size = 1689600
#BEGIN: Fri May 15 08:03:02 2020
!epoch 0/8: elapsed=0.000321, 0.000000 Pflops (estimate)
!epoch 1/8: elapsed=370.133893, 1468.210388 Pflops (estimate)
!epoch 2/8: elapsed=650.033814, 1464.253280 Pflops (estimate)
!epoch 3/8: elapsed=851.582777, 1461.317344 Pflops (estimate)
!epoch 4/8: elapsed=988.277782, 1457.670705 Pflops (estimate)
!epoch 5/8: elapsed=1072.277294, 1454.437530 Pflops (estimate)
!epoch 6/8: elapsed=1117.625903, 1450.088542 Pflops (estimate)
!epoch 7/8: elapsed=1143.143441, 1437.409844 Pflops (estimate)
# iterative refinement: step=  0, residual=5.0168606685474515e-04
hpl-harness=222677.730166
# iterative refinement: step=  1, residual=1.5618974863462753e-11
hpl-harness=0.006931
#END__: Fri May 15 08:22:50 2020
1158.483898010 sec. 1421151817.927741528 GFlop/s resid =
1.561897486346275e-11 hpl-harness = 0.006931424
1421.151818 Pflops, 91.267070 %
```

Analysis and Discussions

- **Performance**
  - **Well optimized. 91.27% in total ←←← HGEMM performs >95%** of the peak
- **Load-imbalance = 2–18% (in the case of the tiny setting)**
  - Mostly caused by data dependency of the LU decomposition.
  - **Broadcast operations → offloaded onto the TofuD functionality.**
- **Performance bottleneck**
  - A64FX has an on-die communication control unit → simple arch. and extremely low latency
  - for 8B put latency is **0.49-0.54[μs],** and 1MiB throughput attains **6.35[GB/s].**

- **On the other hand, case of the Summit code**
  - **Not sure for implementation of the partial pivoting : ours has no pivoting**
  - Tensor-Core's computational performance is extremely high,
  - However, latency between NIF and GPU memory is large.

→ Reasons for the large score gap between Summit and Fugaku.

Summary

- **We investigated three numerical and four implementation techniques of the HPL-AI benchmark**
  - Reported what we have learned, and the real tests on the supercomputer Fugaku.
  - Preliminary numerical analysis and implementation works, and we never seen numerical instability and inaccurate results.
- **Our HPL-AI implementation achieved >1Eflops**
  - demonstrated the world's first Exa-flops computing, 1.421 EFlop/s.
- **We intended to make our HPL-AI code open source soon.**
- **Only focused on performance (speed or flop/s), but other metrics are also significant. We should analyze and report them very shortly.**

We will update the result using the full Fugaku system in SC week.

Acknowledgement

- **We sincerely express our grateful acknowledgment to**
  - Prof. Satoshi Matsuoka, Director R-CCS,
  - The Flagship 2020 project,
  - Dr. Yutaka Ishikawa, Dr. Mitsuhisa Sato, Dr. Fumiyoshi Shoji, and Fujitsu company.

- **The results obtained on the evaluation environment in the trial phase do not guarantee the performance, power and other attributes of the supercomputer Fugaku at the start of its operation.**