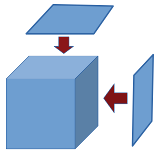
HPL-AI

Mixed Precision

HPL-AI JUNE 2021

		SITE	CORES	HPL RMAX EFLOP/S	TOP500 RANK	HPL-AI EFLOP/S	SPEEDUP
1	**Fugaku** Fujitsu A64FX, Tofu D	RIKEN Center for Computational Science JAPAN	7,630,848	0.442	1	2.0	4.5
2	**Summit** AC922 IBM POWER9, IB Dual-rail FDR, NVIDIA V100	DOE/SC/ORNL USA	2,414,592	0.149	2	1.15	7.7
3	**Selene** DGX SuperPOD, AMD EPYC 7742 64C 2.25 GHz, Mellanox HDR, NVIDIA A100	NVIDIA USA	555,520	0.063	6	0.63	9.9
4	**Perlmutter** HPE Cray EX235n, AMD EPYC 7763 64C 2.45 GHz, Slingshot-10, NVIDIA A100	DOE/SC/LBNL/NERSC USA	761,856	0.065	5	0.59	9.1
5	**JUWELS Booster Module** Bull Sequana XH2000 , AMD EPYC 7402 24C 2.8GHz, Mellanox HDR InfiniBand, NVIDIA A100, Atos	Forschungszentrum Juelich (FZJ) GERMANY	449,280	0.044	8	0.47	10
6	**HiPerGator** NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Infiniband HDR	University of Florida USA	138,880	0.017	23	0.17	9.9
7	**Wisteria/BDEC-01 (Odyssey)** PRIMEHPC FX1000, A64FX 48C 2.2GHz, Tofu D, Fujitsu	Information Technology Center THE UNIVERSITY OF TOKYO, JAPAN	368,640	0.022	13	0.10	4.5
8	**Berzelius** NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, A100, Infiniband HDR, Atos	National Supercomputer Centre (NSC) SWEDEN	59,520	0.005	84	0.05	9.9
9	**Flow Type II subsystem** PRIMERGY CX2570 M5, Xeon Gold 6230 20C 2.1GHz, NVIDIA Tesla V100 SXM2, Infiniband EDR	Information Technology Center NAGOYA UNIVERSITY, JAPAN	79,560	0.0049	87	0.03	4.3
10	**MTS GROM** NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, A100 40GB, Infiniband	#CloudMTS RUSSIA	19,840	0.0023	245	0.015	7

OVERVIEW

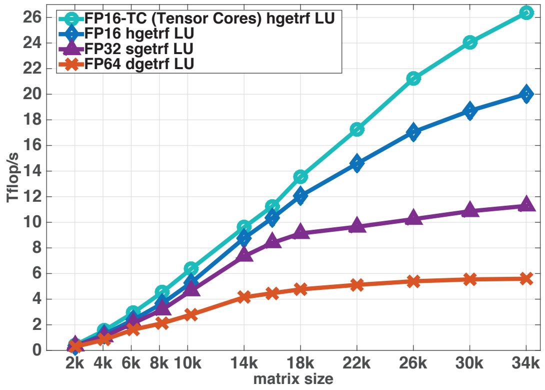


The HPL-AI benchmark seeks to highlight the emerging convergence of high-performance computing (HPC) and artificial intelligence (AI) workloads. While traditional HPC focused on simulation runs for modeling phenomena in physics, chemistry, biology, and so on, the mathematical models that drive these computations require, for the most part, 64-bit accuracy. On the other hand, the machine learning methods that fuel advances in AI achieve desired results at 32-bit and even lower floating-point precision formats. This lesser demand for accuracy fueled a resurgence of interest in new hardware platforms that deliver a mix of unprecedented performance levels and energy savings to achieve the classification and recognition fidelity afforded by higher-accuracy formats.

HPL-AI strives to unite these two realms by delivering a blend of modern algorithms and contemporary hardware while simultaneously connecting to the solver formulation of the decades-old HPL framework of benchmarking the largest supercomputing installations in the world. The solver method of choice is a combination of LU factorization and iterative refinement performed afterwards to bring the solution back to 64-bit accuracy. The innovation of HPL-AI lies in dropping the requirement of 64-bit computation throughout the entire solution process and instead opting for low-precision (likely 16-bit) accuracy for LU, and a sophisticated iteration to recover the accuracy lost in factorization. The iterative method guaranteed to be numerically stable is the generalized minimal residual method (GMRES), which uses application of the L and U factors to serve as a preconditioner. The combination of these algorithms is demonstrably sufficient for high accuracy and may be implemented in a way that takes advantage of the current and upcoming devices for accelerating AI workloads.

FIND OUT MORE AT
https://icl.bitbucket.io/hpl-ai/

PERFORMANCE Xgetrf routine



PUBLICATIONS

Azzam Haidar, Stanimire Tomov, Jack Dongarra, Nicholas J. Higham
Harnessing GPU Tensor Cores for Fast FP16 Arithmetic to Speed up Mixed-Precision Iterative Refinement Solvers In Procedeedings of SC18, 2018.
https://dl.acm.org/citation.cfm?id=3291719

Erin Carson and Nicholas J. Higham
Accelerating the Solution of Linear Systems by Iterative Refinement in Three Precisions SIAM J. SCI. COMPUT., Vol. 40, No. 2, pp. A817–A847, 2018.
https://epubs.siam.org/doi/pdf/10.1137/17M1140819

Erin Carson and Nicholas J. Higham
A New Analysis of Iterative Refinement and its Application to Accurate Solution of Ill-Conditioned Sparse Linear Systems 2017.
http://eprints.maths.manchester.ac.uk/2537/

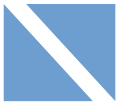
Nicholas J. Higham, Srikara Pranesh, and Mawussi Zounon
Squeezing a Matrix into Half Precision, with an Application to Solving Linear Systems 2018.
http://eprints.maths.manchester.ac.uk/2678/

Pierre Blanchard, Nicholas J. Higham, Florent Lopez, Theo Mary, and Srikara Pranesh
Mixed Precision Block Fused Multiply-Add: Error Analysis and Application to GPU Tensor Cores 2019.
http://eprints.maths.manchester.ac.uk/2733/

ICL INNOVATIVE COMPUTING LABORATORY

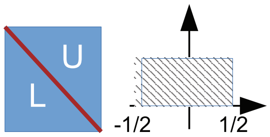
THE UNIVERSITY OF
TENNESSEE
KNOXVILLE

HPL-AI RULES



The idea for the benchmark is to solve a system of linear equations to 64-bit floating point accuracy by doing a mixed-precision factorization of a matrix and compute an approximate solution from the low-precision factorization (LU decomposition), and then use an iterative method like GMRES in 64-bit precision to iterate with the approximate low-precision solution to compute a final solution obtaining the accuracy one would have achieved by LU decomposition in 64-bit floating point arithmetic. The low-precision LU factors should be used as a preconditioner in the iterative algorithm.

The benchmark should use the HPL benchmark harness (https://www.netlib.org/benchmark/hpl/) with a modification of the matrix generator. The generator will produce a non-symmetric matrix with the diagonal entries being the sum of the off-diagonal rows, this will force the matrix to be diagonally dominant.



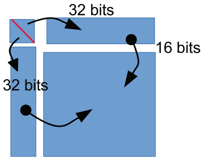
$$r_\infty \xrightarrow[n\to\infty]{} \frac{\frac{2}{3}n^3}{T_{tot}}$$

In an attempt to obtain uniformity across all computers in performance reporting, the algorithm used in solving the low-precision system of equations in the benchmark procedure must numerically conform to an LU factorization with partial pivoting. In particular, the operation count for the algorithm must be $\frac{2}{3}n^3+O(n^2)$ double precision floating point operations even though double-precision arithmetic is not required.

The HPL harness computes a backward-error: $\frac{\|Ax-b\|_\infty}{\|A\|_\infty\|x\|_\infty+\|b\|_\infty}\times(n\times\epsilon)^{-1}$, where ϵ is the machine precision in 64-bit floating point arithmetic (on IEEE machines this is $\epsilon=2^{-53}$) and n is the size of the problem. There is no restriction on the problem size.

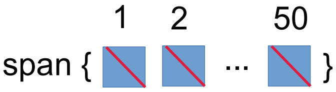
$$FP_{64}\left(\frac{\|Ax-b\|_\infty}{\|A\|\|x\|+\|b\|}\right)\approx\varepsilon\,\theta(n)$$

The implementation is allowed to do balancing to get the numbers within range of the floating point format, but the time to do the balancing must be included in the time to solution.



The factorization can use mixed precision during its construction, e.g., the panel factorization and triangular solves can be done in 32-bit arithmetic and the Schur complement (matrix-matrix multiply) can be computed in 16-bit arithmetic with 32-bit accumulation.

The computation rate is based on the time to solve the problem: factor the matrix in lower precision, perhaps balance the matrix to prevent overflow, perform GMRES in 64-bit floating point arithmetic using the LU factors as a preconditioner. If the implementation takes more than 50 iterations, the method should trigger a failure and the run is not valid.



In computing a rate of execution, $\frac{2}{3}n^3+\frac{3}{2}n^2$ operations ($\frac{2}{3}n^3-\frac{1}{2}n^2$ accounts for LU factorization and $2n^2$ for the subsequent back- and forward-solves) will be divided by the complete time to solution to achieve operations per second.

```
for i = 1..50
  | x_i = (LU)^{-1}
  | …
end
```

As part of the submission of results we expect the submitter to provide a detailed explanation of the algorithm used in the submission.

We have provided a reference implementation whose purpose is to show how the benchmark could be implemented. We do not expect this to be used in actually running of the benchmark. Optimizations should be applied to achieve higher performance than the reference implementation could achieve. The reference implementation can be found here on Bitbucket (https://bitbucket.org/icl/hpl-ai/)

FIND OUT MORE AT
https://icl.bitbucket.io/hpl-ai/

ICL INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY OF TENNESSEE
KNOXVILLE