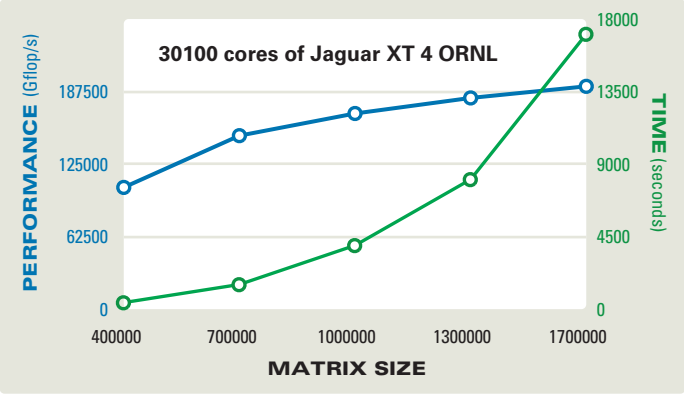
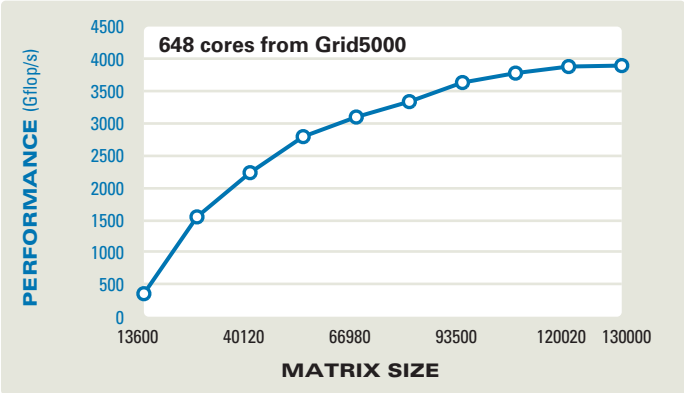
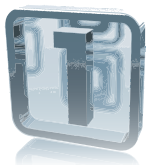
HPL

HPL is a portable implementation of the High Performance LINPACK Benchmark for distributed memory computers.

- Algorithm: recursive panel factorization, multiple lookahead depths, bandwidth reducing swapping
- Easy to install, only needs MPI+BLAS or VSIPL
- Highly scalable and efficient from the smallest cluster to the largest supercomputers in the world







HPL was used to obtain a number of results in the current TOP500 list, including the #1 entry.

HISTORY OF THE BENCHMARK

1974 **LINPACK software is released**
Solves systems of linear equations in FORTRAN 66

1977 **LINPACK 100 released**
Measures system performance in Mflop/s and solves 100x100 linear systems

1986 **LINPACK 1000 released**
Any language allowed and the linear system of size 1000 can be used

1989 **LINPACKDv2 released**
Extends random number generator from 16384 to 65536

1991 **LINPACK Table 3 (Highly Parallel Computing)**
Any size linear system is allowed

1993 **TOP500 first released**
With CM-5 running the LINPACK benchmark at nearly 60 Gflop/s

1996 **9th TOP500 is released**
With the 1st system breaking the 1 Tflop/s barrier: ASCI Red from Sandia National Laboratory

2000 **HPLv1 is released**
By Antoine Petitet, Jack Dongarra, Clint Whaley, and Andy Cleary

2008 **31st TOP500 is released**
With the 1st system breaking the 1 Pflop/s barrier: Roadrunner from Los Alamos National Laboratory

HPLv2 is released
The new version features a 64-bit random number generator that prevents the benchmark failures from generating singular matrices which was the problem with the old generator.

2009 **Peta flop/s are spreading**
The upgrades of the machines hosted at ORNL results in shattering the 2 Pflop/s theoretical peak barrier for Cray's XT5 Jaguar. Also, the first academic system reaches 1 Pflop/s theoretical peak: University of Tennessee's Kraken.

2010 **GPUs are coming**
The performance growth at Peta-scale is now fueled by a new breed of GPUs that now power systems with over 2 Pflop/s in LINPACK performance.

2011 **Multicore strikes back** by exceeding 8 Pflop/s using over half a million of modified SPARC VIII cores.

2012 **Even more cores** needed to be #1: a million and a half cores were required to break 15 Pflop/s barrier.

2013 **Intel back on top** with over half of 100 Pflop/s of peak performance coming mostly from its Intel Xeon Phi acceleratos

ICL UT INNOVATIVE COMPUTING LABORATORY THE UNIVERSITY of TENNESSEE

IN COLLABORATION WITH OAK RIDGE National Laboratory

SPONSORED BY U.S. DEPARTMENT OF ENERGY

HPL

The original LINPACK Benchmark is, in some sense, an accident. It was originally designed to assist users of the LINPACK package by providing information on execution times required to solve a system of linear equations. The first "LINPACK Benchmark" report appeared as an appendix in the LINPACK Users' Guide in 1979. Over the years additional performance data was added, more as a hobby than anything else, and today the collection includes over 1300 different computer systems. In addition to the number of computers increasing, the scope of the benchmark has also expanded. The benchmark report describes the performance for solving a general dense matrix problem Ax=b at three levels of problem size and optimization opportunity: 100 by 100 problem (inner loop optimization), 1000 by 1000 problem (three loop optimization – the whole program), and a scalable parallel problem – the HPL. New statistics were required to reflect the diversification of supercomputers, the enormous performance difference between low-end and high-end models, the increasing availability of massively parallel processing (MPP) systems, and the strong increase in computing power of the high-end models of workstation suppliers (SMP). To provide this new statistical foundation, the TOP500 list was created in 1993 to assemble and maintain a list of the 500 most powerful computer systems. The list is compiled twice a year with the help of high-performance computer experts, computational scientists, manufacturers, and the Internet community in general. In the list, computers are ranked by their performance on the HPL benchmark. With the unprecedented increase in performance comes the price of increased time the benchmark takes to complete. As the chart below shows, the time to completion is increasing with the number of cores. In 2009, the #1 entry reported a run that took over 18 hours. In 2011, this time increased to nearly 30 hours.

To deal with this problem, a new version of HPL will offer a partial run option. This will allow the machine to run the benchmark only a fraction of the time it takes to run the full version of the benchmark without losing much of the reported performance. Initial runs on a system with over 30000 cores indicate only a 6% drop in performance with a 50% reduction in time. This becomes a viable alternative to, say, checkpointing to solve the problem of the rapidly increasing number of cores and the decrease of Mean Time Between Failures (MTBF) for large supercomputer installations.

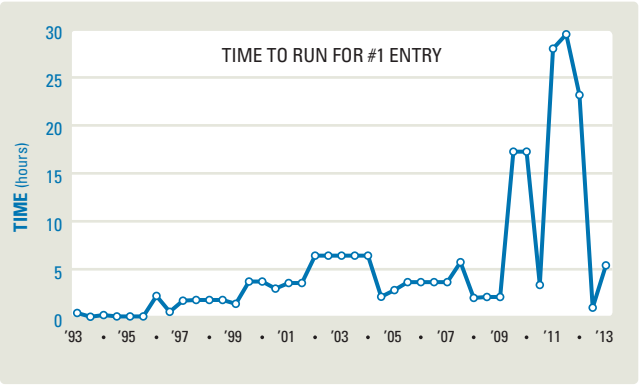
As shown below in the performance-time curve, the time is given in relative terms as a fraction of the maximum attained performance. In this manner, both time and performance may coexist on the same Y-axis because they both vary between 0% and 100%. In this setting it is now easy to perform a what-if analysis of the data as it is indicated with arrows. The question being answered by the figure is this: if the time to run the benchmark is reduced by 50% how much will the performance decrease? The answer is encouraging: the resulting performance drop will only be 5%. The sampled factorization provides the added benefit of stressing the entire system: the system matrix occupies the entire memory. Also, the result of the partial execution can be verified as rigorously, as is the case for the standard factorization.

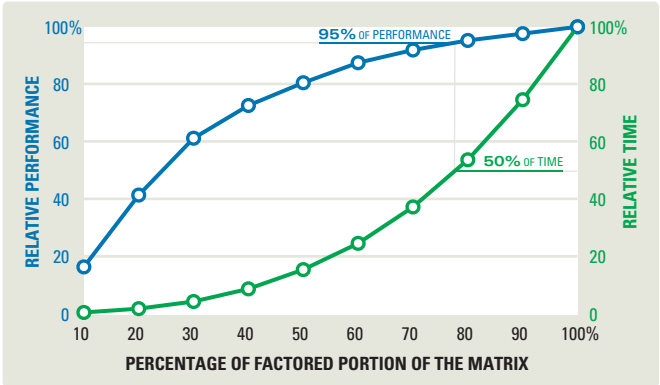
RELEASED OCTOBER 2012

HPL 2.1

The output now reports exact time stamps before and after the execution of the solver function HPL_pdgesv(). This could allow for accurate accounting of running time for data center management purposes. For example, for reporting power consumption. This is important for the Green500 project.

Fixed an out-of-bounds access to arrays in the HPL_spreadN() and HPL_spreadT() functions. This may cause segmentation fault signals. It was reported by Stephen Whalen from Cray.





ICL UT INNOVATIVE COMPUTING LABORATORY THE UNIVERSITY of TENNESSEE

IN COLLABORATION WITH OAK RIDGE National Laboratory

SPONSORED BY U.S. DEPARTMENT OF ENERGY

CITR CENTER FOR INFORMATION TECHNOLOGY RESEARCH

FIND OUT MORE AT http://icl.cs.utk.edu/**hpl**