LinpackR: An Implementation of the Linpack Benchmark for the R Programming Language

David J. Lilja
Department of Electrical and Computer Engineering
University of Minnesota
Minneapolis, MN 55455 USA
Email: lilja@umn.edu

***Abstract*—The popularity of R as a programming language has increased substantially over the past few years. Despite this rising popularity, there has been little systematic evaluation of the performance of R execution environments. To help initiate this evaluation process, this work introduces LinpackR, which is a port to the R language of the popular Linpack benchmark program. The use of this new benchmark program is demonstrated by comparing the standard R execution environment with an R environment that has been enhanced with Intel's Math Kernel Library (MKL), and with an optimized Fortran version of Linpack. Performance tests on a Xeon-based server show that the MKL enhancements increase the maximum performance of R execution up to 13.9 times, from 3.0 to 41.8 GFLOPS, when using a single thread of execution. Increasing the number of threads to twenty increases the performance 4.6 times to a maximum of 192.5 GFLOPS. Its maximum performance is still 40 percent below the maximum performance of optimized Fortran code at 321 GFLOPS, however. Additional tests on a two-core Pentium-based system show that the performance of the LinpackR benchmark increases approximately proportionally to increases in the clock rate, and that the choice of operating systems, either Ubuntu Linux or Windows 10, has no significant effect on performance.**

Keywords: *R benchmark programs, R performance, R programming language, performance benchmarking, Linpack, Top500*

I. Introduction

THE R programming language and associated execution environment [1] is used extensively for statistical computing, such as regression modeling [2] [3] [4] and a broad range of applied statistical analyses [5] [6] [7]. It also is being used more frequently for general programming applications [8] [9] [10] and high-performance numerical computing [11]. The popularity of R has grown significantly in the past few years [12]. Recently it has been estimated to be the fifth most popular programming language ranking between C++ at fourth and C# at sixth [13].

R is typically used in an interactive execution environment where the R code is interpreted rather than compiled. The execution environment has been ported to many different systems making it easy to run R programs across a range of systems. Given its popularity, there has been a surprising lack of systematic performance evaluation of R programs when executed using different interpreters, operating systems, and processor capabilities.

This work introduces the *LinpackR* benchmark program[1] for measuring the performance of different R execution environments when running on systems with different capabilities. It is a straight-forward port to the R language of the Linpack benchmark program [14] [15]. Linpack has a long history for measuring the performance of high-performance computing systems. It is perhaps best known for its use in the Top500 ranking of the world's fastest computer systems [16].

The metric of performance for LinpackR is the number of floating-point operations executed per second (FLOPS). This value is scaled by 10^9 to provide a GFLOPS rating. There are well-known issues with measuring performance using FLOPS since no single number can adequately capture the complexity of a computer system's performance [17]. Nevertheless, given the relative simplicity of the Linpack benchmark, its widespread use, and the fact that it provides a good indication of the peak performance of a computer system, it is a useful benchmark program with which to begin evaluations of the performance of R execution environments.

The use of this benchmark program is demonstrated by comparing two different R execution environments, the standard open-source implementation [1] and a version optimized for Intel processors [18]. These results are further compared to an optimized Fortran version of the standard Linpack benchmark. The performance effects of varying the number of parallel threads used to execute the benchmark and the problem size are observed. Finally, the performance impact of different operating systems and different clock speeds is shown.

II. The Standard Linpack Benchmark Program

The Linpack benchmark program [14] [15] solves a dense linear system of equations. The solution vector x is found for the system $Ax = b$ using LU factorization with partial pivoting. The matrix A is size $n \times n$ and x and b are vectors of length n. Both A and b are initialized to random values uniformly distributed between -1 and +1.

The original implementation of Linpack was written in Fortran and specified a fixed problem size, n. It has subsequently been ported to several different languages. Any solution method is allowed as long as the accuracy of the final solution is as good as that produced by the original approach

[1]The LinpackR source code is available at: z.umn.edu/linpackr

using LU factorization with partial pivoting. More precisely, the relative accuracy must be such that:

$$\frac{\|Ax-b\|}{\|A\|\|x\|n\epsilon} \leq O(1) \tag{1}$$

where ϵ is the machine precision for 64-bit floating-point arithmetic. If this inequality holds true for the x vector computed by the benchmark program, the solution is said to be valid.

Different solution techniques other than LU factorization may require a different number of floating-point operations. These differences then would affect the performance result as measured by the GFLOPS value. To eliminate this difference when reporting performance results, the total number of floating-point operations, F, for the Linpack benchmark is defined to be:

$$F = 2n^3/3 + 2n^2 \tag{2}$$

where n is the problem size.

The final performance metric, GFLOPS, then is computed as:

$$GFLOPS = \frac{F}{10^9 T} \tag{3}$$

where T is the time in seconds required to compute the final result, x.

III. The LinpackR Benchmark Program

The LinpackR benchmark program is a straight-forward port of the standard Linpack program to the R programming language. The key portion of the program that is timed computes the solution vector as follows:

```
system.time(x <- solve(A,b), gcFirst=TRUE)[3]
```

The actual computation is performed using R's built-in `solve()` function. This function takes the matrix, A, and the right-hand side, b, as input parameters and returns the solution vector x. The enclosing `system.time()` function measures the time required to execute the `solve()` function. The `gcFirst` parameter is set to `TRUE` so that all memory garbage collection operations are performed before the `solve()` function is executed. This operation ensures that the system memory starts in a clean state each time the `system.time()` function is called to measure the execution time.

The `system.time()` function returns a vector consisting of three values - the user time, the system time, and the total elapsed time. The "`[3]`" appended to the function call selects the third value of the vector returned by the function, which is the total elapsed time, as the execution time used in computing the performance metric. Before the `system.time()` function is called, there is some additional code to initialize A and b to random values uniformly distributed between -1 and +1. Following this function is the code to compute the number of floating-point operations using Equation 2, the GFLOPS performance metric value using Equation 3, and the validation of the final solution with Equation 1.

Using the `solve()` function in LinpackR demonstrates the simplicity of writing in R what would be relatively complex code in many other languages. This simplicity of expression is one of the important advantages of the R language. Additionally, using this function allows the execution environment to use optimized libraries when executing the `solve()` function. One such library can produce substantial performance gains, as we observe in the following sections.

TABLE I
THE SOFTWARE VERSIONS USED IN THE PERFORMANCE EVALUATION.

Name	**Source**	**Version**
Fortran-MKL	Optimized Fortran [20]	mklb_p_2017.1.013
R-MKL	MKL-optimized R interpreter [18]	3.3.2
R-base	Standard R interpreter [1]	3.2.3

IV. Experimental Evaluation Methodology

Two different versions of the R execution environment are compared in this demonstration of LinpackR. The first is the original open source version developed and maintained by the R Foundation for Statistical Computing [1]. This version is referred to as *R-base* in the remainder of this paper. It is the version that most users are likely to find installed on their systems.

The second version of the R execution environment, which is also open source, has been developed by the Microsoft R Application Network (MRAN) [18]. The primary difference compared to R-base is that this version incorporates Intel's Math Kernel Library (MKL) [19] into the execution environment. The Math Kernel Library includes many functions for numerically-intensive operations that have been highly optimized to take advantage of vectorized instruction sets and multiple computing cores. This version is referred to as *R-MKL* in this paper.

These R execution environments are compared to a compiled Fortran version of the standard Linpack benchmark. This version has been highly optimized by Intel using the MKL functions [20] and is referred to as *Fortran-MKL* in the following performance comparisons. The specific versions of the software used are summarized in Table I.

The performance of these different R and Linpack benchmark implementations was evaluated on the computing systems shown in Table II. The Xeon system is a server-class system with ten processing cores. Each core is capable of running two simultaneous threads using Intel's Hyperthreading technology. The Pentium-based systems are single-user PCs and are actually all the same physical hardware. The system was evaluated once when running Linux and again when running Windows 10. Both operating system configurations were executed at the base clock frequency of 3.2 GHz, and when over-clocked to 4.3 GHz. This system contains two processing cores, each of which can execute a single thread. Note that the MKL-enhanced software can take advantage of all available cores in these systems.

V. Performance Comparisons

The LinpackR benchmark program was executed under the R-base and R-MKL execution environments on each of the system configurations shown in Table II. The Fortran-MKL

TABLE II
THE COMPUTING SYSTEMS USED IN THE PERFORMANCE EVALUATION.

Name	Type	CPU	Clock	Memory	OS	Cores	Max Threads
Xeon	Server	Intel Xeon E5-2660 v3	2.6 GHz	16 GB	Linux Ubuntu 16.04.1 LTS	10	20
Pentium-L	PC	Intel Pentium G3258	3.2/4.3 GHz	8 GB	Linux Ubuntu 16.04.1 LTS	2	2
Pentium-W	PC	Intel Pentium G3258	3.2/4.3 GHz	8 GB	Windows 10 Ver. 1607	2	2

version of the Linpack benchmark also was executed on each of these systems. The problem size, n, was varied from 1000 $\times$ 1000 to a value large enough to exceed the size of each system's physical memory. The performance metric used to compare the results is GFLOPS, as reported by the benchmark program.

Each program was executed multiple times to obtain a measure of the variance of the performance. This variance is plotted as 95 percent confidence intervals in the following graphs. There were no other programs running on these systems during the tests, other than the normal background processes of the operating system. As a result, in most cases the confidence intervals are so small that they are not visible on the graphs. The solution error as measured by Equation 1 is similar for both the R and Fortran versions, which is expected since both versions use 64-bit precision arithmetic.

A. *Effect of Parallel Threads*

The performance of the various versions of the benchmark program when running on the Xeon system is shown in Figure 1. The two solid lines, one at the top and one near the bottom, show the performance of the standard Fortran-MKL version of Linpack when running with 20 threads and one thread, respectively. Both of these lines show a typical performance characteristic for Linpack as the problem size is varied [15]. When the problem size is small, the execution overheads, such as loop overhead, memory accessing times, cache misses, and so forth, are relatively large with respect to the actual floating-point computations used to solve the system of equations. As the problem size increases, the performance tends to increase as this overhead is amortized over additional useful computations. Eventually, the problem size becomes large enough that it does not fit in the physical memory. This can result in paging to virtual memory on the secondary disk memory, which causes the performance to decrease. The problem size in this figure does not become quite large enough to see this decrease in performance for Fortran-MKL, although it is seen for the LinpackR results.

The bottom-most line in this figure shows the performance of LinpackR when executed under the R-base execution environment. This performance is constant at approximately 3.0 GFLOPS as the problem size is varied. In contrast, the single-thread Fortran-MKL performance levels out at approximately 44.8 GFLOPS. The difference between the R-base performance and the Fortran-MKL performance with a single thread shows the overhead of executing the benchmark using the standard R execution environment compared to a compiled

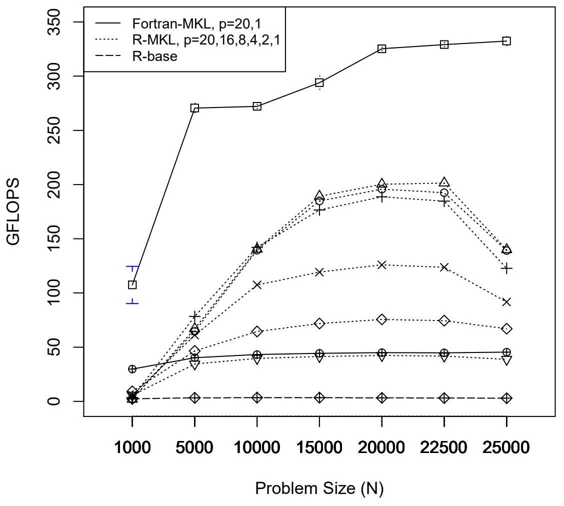


Fig. 1. The LinpackR performance of the Xeon server for the different R execution environments and the optimized Fortran code.

Fortran program. This difference is likely due to R being an interpreted language and that Fortran-MKL uses the optimized numerical libraries while R-base does not. Further study is need to verify this supposition, however.

The dotted lines in the middle of the figure show the performance of LinpackR when executed using the optimized R-MKL execution environment as the number of threads is increased from 1, 2, 4, 8, 16, to 20. Note that the bottom-most dotted line, which shows R-MKL executed with a single thread, mostly overlaps the solid line, which shows the performance of Fortran-MKL executing with a single thread. This result shows that the performance of the R execution environment optimized with the MKL routines is only slightly lower than the optimized Fortran version when executed with a single thread. Note that the performance of R-MKL does begin to drop slightly when the problem size increases to 25,000 while Fortran-MKL continues to maintain its performance.

As the number of threads is increased, the performance of R-MKL also increases until its maximum performance of 192.5 GFLOPS is obtained with twenty threads at a problem size of $n = 22500$. This performance corresponds to a speedup of 4.6 compared to the single-thread performance of 41.8 GFLOPS for R-MKL at this problem size. Fortran-

MKL, in contrast, achieves 321.1 GFLOPS with 20 threads at $n = 22500$, as shown in the top-most line in this figure. This corresponds to a speedup of 7.2 compared to its single-thread performance of 44.8 GFLOPS.

B. Effect of Clock Rate and Operating System

The performance of the various configurations of the Pentium-based system is shown in Figure 2. The bottom row in this figure shows the performance when the processor is clocked at the manufacturer's recommended frequency of 3.2 GHz. The top row shows the performance of the same hardware after it has been manually overclocked to 4.3 GHz. The plots in the left column show the system's performance when running the Linux operating system while the right column shows the performance when running the Microsoft Windows 10 operating system.

The dashed line at the bottom of each plot is the performance of R-base. The performance of the Fortran-MKL version with two threads is shown by the solid line at the top of the plots. The two dotted lines in between Fortran-MKL and R-base show the performance of R-MKL with one and two threads.

Similar to its performance on the Xeon system, R-base achieves maximum performance of about 3.0 GFLOPS at a clock rate of 3.2 GHz on this system. This performance increases about 10 percent to 3.3 GFLOPS when the clock rate is increased 34 percent to 4.3 GHz. The maximum performance of Fortran-MKL with two threads increases 31 percent when the clock rate is increased, from 24.0 to 31.4 GFLOPS. R-MKL also shows this almost proportionate increase in performance with the increase in the clock rate, improving from 22.9 to 29.9 GFLOPS with two threads, and from 12.1 to 15.9 GFLOPS with a single thread. With two threads of execution, the maximum performance of R-MKL is only 4.8 percent below that of Fortran-MKL at the 4.3 GHz clock rate, and 4.6 percent below at the 3.2 GHz clock rate.

Comparing the left and right columns of this figure shows that there is no significant difference in the performance of the R execution environments or the Fortran implementation as the operating system is changed. This independence of the operating system holds true for all of the configurations and problem sizes at both clock rates.

VI. Related Work

Benchmarking has been widely used to evaluate the performance of computer systems [17]. Benchmark programs have been written in a variety of languages for a broad range of applications. Commonly used examples include the SPEC benchmarks [21] [22] [23] for evaluating workstations; Parboil [24], NAS [25], and Parsec [26] for parallel systems; SHOC [27] and Rodinia [28] for heterogeneous systems; MediaBench [29] and MiBench [30] for embedded systems; TPC [31] for transaction processing systems; and Graph500 [32] for evaluating non-numeric operations.

Benchmarking the performance of R execution environments has been typically more *ad hoc* with few R benchmark programs being widely adopted by the computer performance evaluation community. Urbanek [33] developed a collection of R functions that are intended to capture a range of operations that are typically performed in R programs. These functions are timed as a series of microbenchmarks to provide an overall measure of performance. Included are operations such as transposing a matrix, raising a matrix to a power, sorting a list of random numbers, computing a matrix cross-product, and performing linear regression over a large matrix.

The *rbenchmark* package [34] provides a wrapper around R's `system.time()` function. It will time the execution of given R expressions multiple times to measure their performance. It is not a benchmark program itself, however. Rather, it is a tool written in R that can assist in developing benchmark programs.

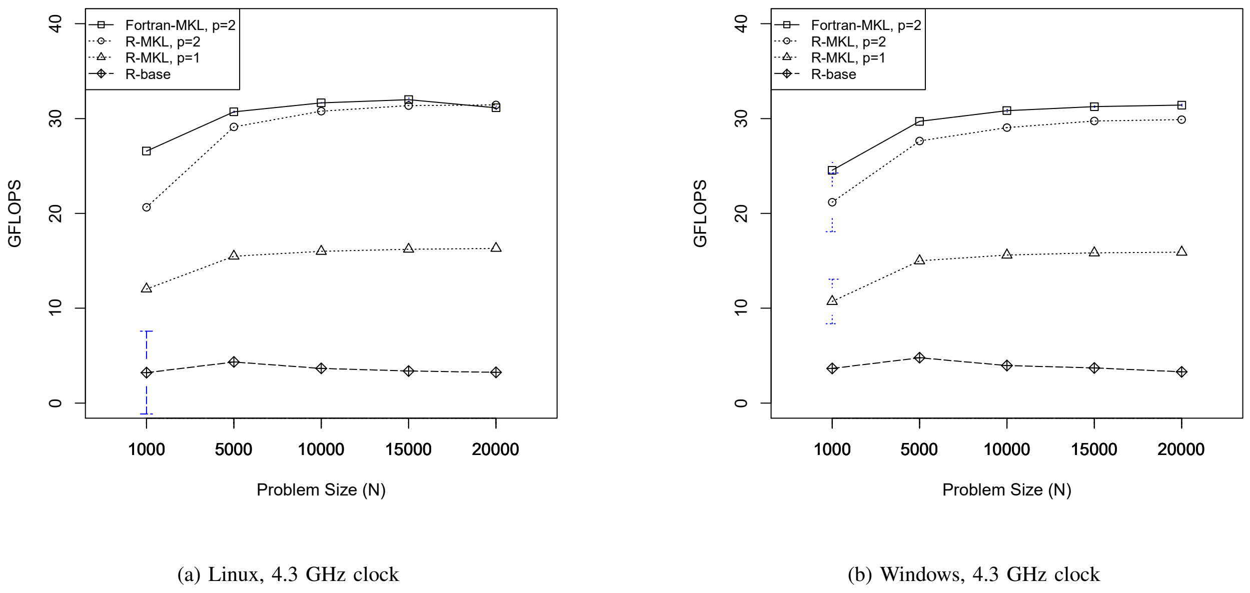
In contrast to prior work, this paper introduces the LinpackR benchmark program to provide a standardized benchmark for systematically evaluating the performance of R execution environments. Since it is ported from the widely used Linpack benchmark, results from LinpackR can be directly compared with any computer system that has traditional Linpack results, as well as providing performance comparisons across R execution environments.

VII. Conclusions

This work introduced the LinpackR benchmark program. It was used to compare the performance of two different R program execution environments, the standard R Project distribution [1] and a version [18] enhanced with Intel's Math Kernel Library (MKL) [19]. The R execution results were further compared with an optimized Fortran version of the Linpack benchmark program. Results were obtained on both a Xeon-based server-class computer system and a Pentium-based desktop system while executing with different numbers of threads, different processor clock frequencies, and different operating systems.

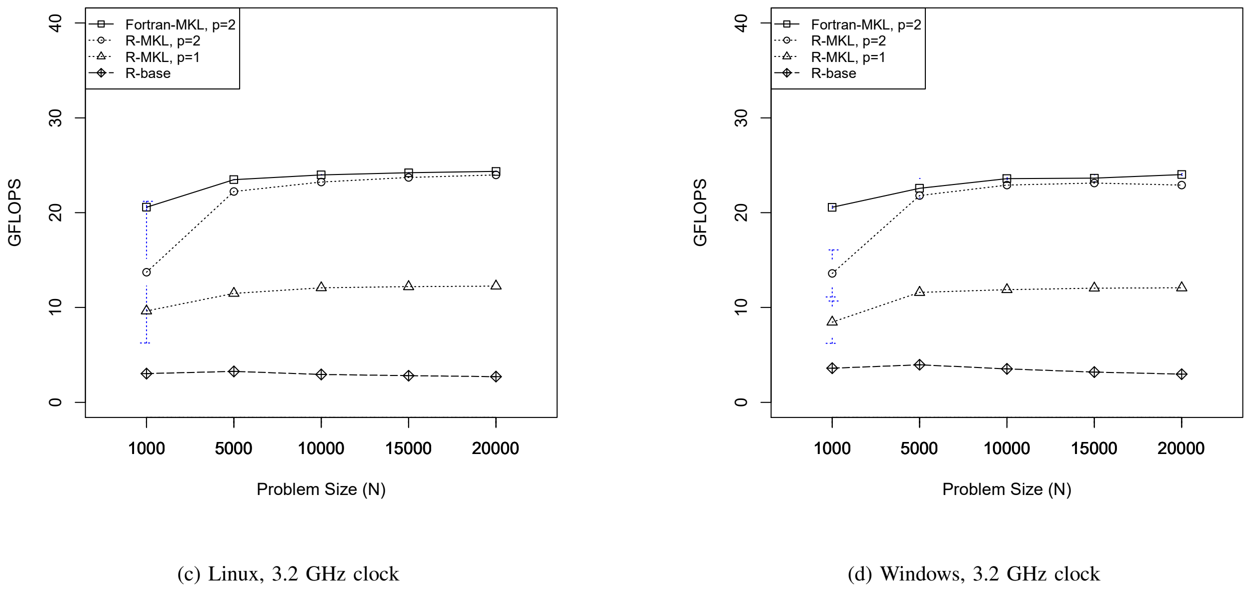
The results show that adding the MKL optimizations to the R interpreter increased the performance of single-threaded execution up to 13.9 times on the Xeon system, from 3.0 to 41.8 GFLOPS. On the Pentium system, the optimizations improved the performance of single-threaded execution by a factor of 4.0 times, from 3.0 to 12.1 GFLOPS, when running at a clock frequency of 3.2 GHz. When the clock frequency was increased to 4.3 GHz, the performance improved by 4.8 times, from 3.3 to 15.9 GFLOPS. The performance of the Pentium system increased almost proportionally with a change in the clock frequency for both of the R execution environments, and for the Fortran implementation, when executed with the same number of threads.

Increasing the number of threads from 1 to 20 increases the maximum performance of R on the Xeon system by 4.6 times, from 41.8 to 192.5 GFLOPS. This maximum performance of R on this system (192.5 GFLOPS) is 40 percent below the maximum performance of the optimized Fortran version at 321.1 GFLOPS. Additionally, the R interpreter reaches its maximum performance at a smaller problem size than the Fortran version. Finally, the choice of the operating system, either Windows 10 or Linux, has no significant effect on performance.



(a) Linux, 4.3 GHz clock

(b) Windows, 4.3 GHz clock



(c) Linux, 3.2 GHz clock

(d) Windows, 3.2 GHz clock

Fig. 2. The performance of the Pentium-based system with different clock frequencies and operating systems.

This work has shown that the LinpackR benchmark program can be used to effectively measure and compare the performance of R execution environments across multiple computer system configurations. It further showed the advantages of increasing the clock frequency and the number of parallel threads when executing a large numerical function in R. Future work includes a deeper analysis of the causes of the performance differences among the R execution environments and the optimized Fortran implementation.

References

[1] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2016. [Online]. Available: https://www.R-project.org/
[2] J. Faraway, "Practical regression and ANOVA using R," 2002.
[3] F. E. Harrell, *Regression modeling strategies*. Springer, 2015.
[4] D. J. Lilja, *Linear Regression Using R: An Introduction to Data Modeling*. University of Minnesota Libraries Publishing, 2016.
[5] P. Dalgaard, *Introductory statistics with R*. Springer, 2008.
[6] N. M. Radziwill, *Statistics (The Easier Way) with R: An informal text on applied statistics*. Lapis Lucera, 2015.
[7] A. P. Field, J. Miles, and Z. Field, *Discovering statistics using R*. Sage Publications, 2012.
[8] N. S. Matloff, *The art of R programming*. No Starch Press, 2011.

[9] P. Teetor and M. K. Loukides, *R cookbook*. O'Reilly Media, 2011.

[10] H. Wickham, *Advanced R*. Chapman and Hall/CRC Press, 2014.

[11] N. S. Matloff, *Parallel computing for data science*. CRC Press, 2015.

[12] N. Diakopoulos, "Top programming languages trends: The rise of big data," *IEEE Spectrum (online)*, July 26, 2016, accessed: 2016-12-15.

[13] S. Cass, "The 2016 top programming languages," *IEEE Spectrum (online)*, July 26, 2016, accessed: 2016-12-15.

[14] J. J. Dongarra, "Performance of various computers using standard linear equations software," *SIGARCH Comput. Archit. News*, vol. 20, no. 3, pp. 22–44, Jun. 1992. [Online]. Available: http://doi.acm.org/10.1145/141868.141871

[15] J. J. Dongarra, P. Luszczek, and A. Petitet, "The linpack benchmark: past, present and future," *Concurrency and Computation: Practice and Experience*, vol. 15, no. 9, pp. 803–820, 2003. [Online]. Available: http://dx.doi.org/10.1002/cpe.728

[16] E. Strohmaier, J. Dongarra, H. Simon, M. Meuer, and H. Meuer, "The TOP 500 list," 2017, accessed: 2017-01-27.

[17] D. J. Lilja, *Measuring Computer Performance: A Practitioner's Guide*. Cambridge University Press, 2000.

[18] MRAN, *The Microsoft R Portal*, Microsoft R Application Network, 2016, accessed: 2017-01-20. [Online]. Available: https://mran.microsoft.com/

[19] Intel, *Intel Math Kernel Library*, Intel, 2016, accessed: 2017-01-20. [Online]. Available: https://software.intel.com/en-us/intel-mkl

[20] ——, *Intel Math Kernel Library Benchmarks*, Intel, 2016, accessed: 2017-01-20. [Online]. Available: https://software.intel.com/en-us/articles/intel-mkl-benchmarks-suite

[21] J. L. Henning, "SPEC CPU suite growth: An historical perspective," *SIGARCH Comp. Arch. News*, vol. 35, no. 1, pp. 65–68, Mar. 2007.

[22] ——, "Spec cpu2000: Measuring cpu performance in the new millennium," *Computer*, vol. 33, no. 7, pp. 28–35, 2000.

[23] ——, "Spec cpu2006 benchmark descriptions," *SIGARCH Comp. Arch. News*, vol. 34, no. 4, pp. 1–17, sep 2006.

[24] J. A. Stratton, C. Rodrigues, I.-J. Sung, N. Obeid, L.-W. Chang, N. Anssari, G. D. Liu, and W.-m. W. Hwu, "Parboil: A revised benchmark suite for scientific and commercial throughput computing," *Center for Reliable and High-Performance Computing*, vol. 127, 2012.

[25] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber *et al.*, "The nas parallel benchmarks," *The International Journal of Supercomputing Applications*, vol. 5, no. 3, pp. 63–73, 1991.

[26] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The parsec benchmark suite: Characterization and architectural implications," in *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*. ACM, 2008, pp. 72–81.

[27] A. Danalis, G. Marin, C. McCurdy, J. S. Meredith, P. C. Roth, K. Spafford, V. Tipparaju, and J. S. Vetter, "The scalable heterogeneous computing (shoc) benchmark suite," in *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units*. ACM, 2010, pp. 63–74.

[28] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," in *Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on*. Ieee, 2009, pp. 44–54.

[29] C. Lee, M. Potkonjak, and W. H. Mangione-Smith, "Mediabench: a tool for evaluating and synthesizing multimedia and communications systems," in *Proceedings of the 30th annual ACM/IEEE international symposium on Microarchitecture*. IEEE Computer Society, 1997, pp. 330–335.

[30] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "Mibench: A free, commercially representative embedded benchmark suite," in *Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop on*. IEEE, 2001, pp. 3–14.

[31] R. Nambiar, N. Wakou, A. Masland, P. Thawley, M. Lanken, F. Carman, and M. Majdalany, "Shaping the landscape of industry standard benchmarks: contributions of the transaction processing performance council (tpc)," in *Technology Conference on Performance Evaluation and Benchmarking*. Springer, 2011, pp. 1–9.

[32] R. C. Murphy, K. B. Wheeler, B. W. Barrett, and J. A. Ang, "Introducing the Graph 500," *Cray Users Group (CUG)*, 2010.

[33] S. Urbanek, "R benchmark 2.5," June 2008, accessed: 2017-01-27.

[34] W. Kusnierczyk, "Package 'rbenchmark': Benchmarking routine for R," February 20 2015, accessed: 2017-01-27.