International Journal on Data Science and Technology
2018; 4(2): 42-48
http://www.sciencepublishinggroup.com/j/ijdst
doi: 10.11648/j.ijdst.20180402.11
ISSN: 2472-2200 (Print); ISSN: 2472-2235 (Online)

Performance Engineering for Scientific Computing with R

Hui Zhang

Computer Engineering and Computer Science Department, University of Louisville, Louisville, USA

Email address:
h0zhan22@louisville.edu

To cite this article:
Hui Zhang. Performance Engineering for Scientific Computing with R. *International Journal on Data Science and Technology*. Vol. 4, No. 2, 2018, pp. 42-48. doi: 10.11648/j.ijdst.20180402.11

Received: March 26, 2018; **Accepted**: May 26, 2018; **Published**: June 26, 2018

Abstract: R has been adopted as a popular data analysis and mining tool in many domain fields over the past decade. As Big Data overwhelms those fields, the computational needs and workload of existing R solutions increases significantly. With recent hardware and software developments, it is possible to enable massive parallelism with existing R solutions with little to no modification. In this paper, three different approaches are evaluated to speed up R computations with the utilization of the multiple cores, the Intel Xeon Phi SE10P Co-processor, and the general purpose graphic processing unit (GPGPU). Performance engineering and evaluation efforts in this study are based on a popular R benchmark script. The paper presents preliminary results on running R-benchmark with the above packages and hardware technology combinations.

Keywords: Performance Evaluation, R, Intel Xeon Phi, Multi-Core Computing, GPGPU

1. Introduction

R, open-source version of the language S, is best known as package that performs statistical analysis and creates plots. Over the years, R has evolved as a high-level language environment for performing complex calculation and simulation in a variety of scientific computing tasks. R has high-level functions to operate on matrices and perform numerical analysis as well as advanced data analytics.

Although R has been adopted in many scientific domains as a high productive analytic tool, R faces the even greater challenges to scale up the computation with large data set. Recent hardware and software developments have potential to enable massive parallelism with existing R solutions with little to no modification.

The goal of this paper is to evaluate three such approaches to speed up R computations with the utilization of the latest hardware technology including multi-core, many core (GPU) technologies, and Intel Many Integrated Core architecture (MIC).

2. Background

Significant efforts have been made in developing accelerator cards that can easily increase the parallel processing potential in recent years. A general purpose graphic processing unit (GPGPU) extends parallel functions and technologies traditionally embedded in graphic processing units to handle more generic computations. Computational solutions can utilize the parallel features provided by GPU through programing interface such as OPENCL and CUDA. Most recently, the Intel Xeon Phi SE10P Co-processor (Xeon Phi) integrate 60 processing cores and 8GB memory in a single card. A critical advantage of the Xeon Phi co-processor is that, unlike GPU-based co-processors, the processing cores run the Intel x86 instruction set (with 64-bit extensions), allowing the use of familiar programming models, software, and tools. In addition to allowing the host system to offload computing workload partially to the Xeon Phi, it also can run a compatible program independently.

To utilize those new hardware enabled parallelism, a common usage model is to rewrite some basic functions or processing flow with the corresponding parallel version supported by the particular hardware. The code redevelopment requires the user to have extensive knowledge in both existing R code as well as the parallel mechanism supported by the additional packages. On the other hand, R enables linking to other shared mathematics libraries to speed up many basic computation tasks for linear algebra computation. One option to utilize the Intel Many Integrated Core is to use Intel Math Kernel Library (MKL). Lately some R packages have been developed that use the latest multi-core and GPU libraries to give substantial speed-ups to existing linear algebra functions in R. One such example is HiPLARM1 which targets the underlying LAPACK routines and replaces them with the latest linear algebra libraries that take advantage of multi-core CPU and GPU hardware.

This paper presents the study's preliminary results on running R-benchmark with the above packages and hardware technology combinations. Section 2 details the experimental setup. The section 3 presents the initial findings.

3. Evaluation Environment

3.1. R Benchmark

The investigation used the R-25 benchmark script for testing performance of different approaches2. The testing script includes fifteen common computational tasks grouped into three categories: Matrix Calculation, Matrix functions and Programmation. The fifteen tasks are listed in Table 1:

Table 1. *Translation of benchmark number to R-25 benchmark description for all R-25 plots.*

#	R25 Benchmark Task Description
1	Creation, transp., deformation of a 2500×2500 matrix (sec)
2	2400×2400 normal distributed random matrix
3	Sorting of 7,000,000 random values
4	2800×2800 cross-product matrix
5	Linear regression over a 3000×3000 matrix
6	FFT over 2,400,000 random values
7	Eigenvalues of a 640×640 random matrix
8	Determinant of a 2500×2500 random matrix
9	Cholesky decomposition of a 3000×3000 matrix
10	Inverse of a 1600×1600 matrix
11	3,500,000 Fibonacci numbers calculation (vector calc.)
12	Creation of a 3000×3000 Hilbert matrix (matrix calc.)
13	Grand common divisors of 400,000 pairs (recursion)
14	Creation of a 500×500 Toeplitz matrix (loops)
15	Escoufier's method on a 45×45 matrix (mixed)
16	Total time for all 15 tests (not averaged)
17	Overall mean (sum of means of all tests)

3.2. Compute Environment – Stampede Supercomputing Cluster

The evaluation work used the Stampede cluster at Texas Advanced computing Center as the high performance computing environment for performance testing. Stampede supports several latest hardware technologies for improved computational performance including using Xeon Phi accelerators and/or NVIDIA Kepler 20 GPUs for large matrix calculations. In this test, each compute node has two Intel Xeon E5-2680 processors each of which has eight computing cores running @2.7GHz. There is 32GB DDR3 memory in each node for the host CPUs. The Xeon Phi SE10P Coprocessor installed on each compute node has 61 cores with 8GB GDDR5 dedicated memory connected by an x16 PCIe bus. The NVIDIA K20 GPUs on each node have 5GB of on-board GDDR5. All compute nodes are running CentOS 6.3. For this study the stock R 3.01 package is used. The package is compiled with the Intel compilers (v.13) and built with Math Kernel Library (MKLv.11).

4. Acceleration Strategies

The performed experiment focused on the benefit of R programs using the latest multi-core, GPGPU, and Intel Co-processor technologies:

4.1. Exploiting Xeon Phi Co-Coprocessor with MKL

To utilize Xeon Phi co-processor, one option is to use Intel Math Kernel Library (MKL) [1]. MKL which includes a wealth of routines to accelerate application performance and reduce development time such as highly vectorized and threaded linear algebra, fast fourier transforms (FFT), vector math and statistics functions. It has been reported that the compiling R with MKL can provide three times improvements out of box [2].

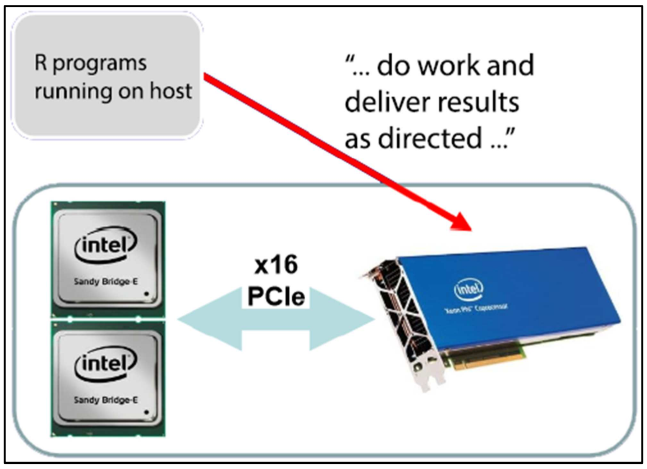

R programs running on host
"... do work and deliver results as directed ..."
intel Sandy Bridge-E
intel Sandy Bridge-E
x16 PCIe
intel Xeon Phi Coprocessor

Figure 1. *Adopting offload model on Stampede cluster at XSEDE: an R program running on the host can "offload" work by directing the MIC to execute a specified block of code. The host also directs the exchange of data between host and MIC.*

```
# enable mkl mic offloading
export MKL_MIC_ENABLE=0

# from 0.0 to 1.0 the work division
export MKL_HOST_WORKDIVISION=0.3
export MKL_MIC_WORKDIVISION=0.7

# Make the offload report big to be visible:
export OFFLOAD_REPORT=2

# now set the number of threads on host
export OMP_NUM_THREADS=16
export MKL_NUM_THREADS=16

# now set the number of threads on the MIC
export MIC_OMP_NUM_THREADS=240
export MIC_MKL_NUM_THREADS=240
```

Figure 2. *Configuring environment variables to enable automatic offloading to Intel Xeon Phi Coprocessor. In this sample script, 70% of computation is offloading to Phi, while only 30% is done on host.*

The evaluation further exploited an offload model to automatically offload MKL operations to Intel

Co-processor. As illustrated in Figure 1, while MKL can automatically manages the computing details, further performance improvement can be obtained by distributing the work across the compute host and the many-integrate-core (MIC).

On Stampede offloading to Xeon Phi can be enabled by setting environment variables as opposed to making modifications to existing R programs (see e.g., Figure 2 for a sample script to enable 70% offloading to Phi.)

4.2. Exploiting Multi-Core and GPU Technologies with HiPLAR

General purpose graphic processing units (GPGPU) extend parallel functions and technologies traditionally embedded in graphic processing units to handle more generic computations. The Matrix package currently uses the BLAS and LAPACK linear algebra libraries to perform its operations. These are the de-facto libraries for performing linear algebra operations, however, they are only designed to run on single core CPUs and are not designed for modern CPU architectures and accelerators.

HiPLAR's over-arching goal is to provide easy access to the latest computational architectures using the latest linear algebra libraries and to do so in an easy and user friendly manner. With the installed suite of R packages the user can achieve large speed-ups with little understanding of the complexities of multi-core and GPU computing.

Another important feature of HiPLAR is its auto-tuning capability: for users that have multi-core CPUs and an NVIDIA GPU, an auto-tuning feature is provided that calculates the optimal configuration for the problem and the hardware. Currently, HiPLAR provides two packages to target linear algebra functions in the standard R release and Matrix packages.

Using the PLASMA library for multi-core CPUs and the MAGMA library for NVIDIA GPUs users can see large speed-up in R codes that use linear algebra routines from simple matrix multiplication to Cholesky and LU decomposition. Figure 3 shows an example where a minimal change (one liner) is applied to the R program to leverage substantial speed-ups by HiPLAR.

```
library(Matrix);
n <- 8192;
X <- Hilbert(n);
A <- nearPD(X);
system.time(B <- chol(A$mat));
# user  system elapsed
# 97.990   0.356  98.591
library(HiPLARM)
system.time(B <- chol(A$mat));
# user  system elapsed
# 1.012   0.316   1.337
```

Figure 3. *A brief example to highlight the benefits of using HiPLARM.*

5. Performance Tuning

Based on the previous observation of significant performance improvement of benchmark version of R computation using MKL and offload model, the study tests R25 benchmark script by choosing work-sharing at the 30% host (16 threads) 70% coprocessor (240 threads) sweet spot (see e.g., [3]).

The evaluation effort tested R25 with multi-core and GPU acceleration using HiPLAR package.

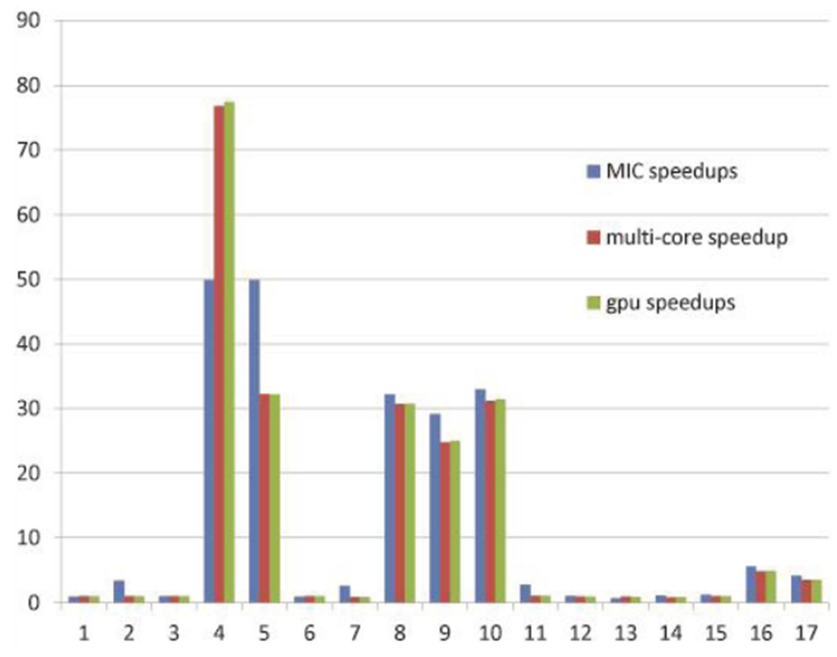


Figure 4. *Basic vectorized and matrixed operations can obtain significant speed-ups by using offload model with MKL and MIC, multi-core, and GPU technologies.*

Figure 4 indicates the obtained speed-ups from the three strategies proposed and evaluated in this study. Significant speed-ups are consistently achieved over various matrices size and matrix based functions. At an R level, the user will notice no difference between using Matrix and using the three acceleration strategies. These methods/packages strive to retain the optimization within the Matrix package in R. So the user will notice no difference between using the offload model, the functions of HiPLAR packages or indeed, results. This feature enables programmers easily access to the latest computational architectures though the linked linear algebra libraries.

6. Use Case: Accelerating Mathematical Knot Simulations with R

The creation of mathematical 3D curves and knots (closed 3D curves) can often be facilitated with a 2D drawing interface. One often constructs an initial configuration for an object while neglecting most issues of geometric placement.

For example, when knot diagrams are drawn, only bare projections are needed with relative depth ordering indicated at crossings while precise 3D depth information is unimportant (e.g., Figure 5 (a)).

The next task that comes naturally is to topologically refine these initial embeddings, not only to make the geometry look more pleasant, but also to remove crossings into the minimal number. One way to refine the initial embedding is to embed the initial graph into 3-dimensional space and replace the vertices with electrostatically charged masses and replace each edge with a spring to form a mechanical system.

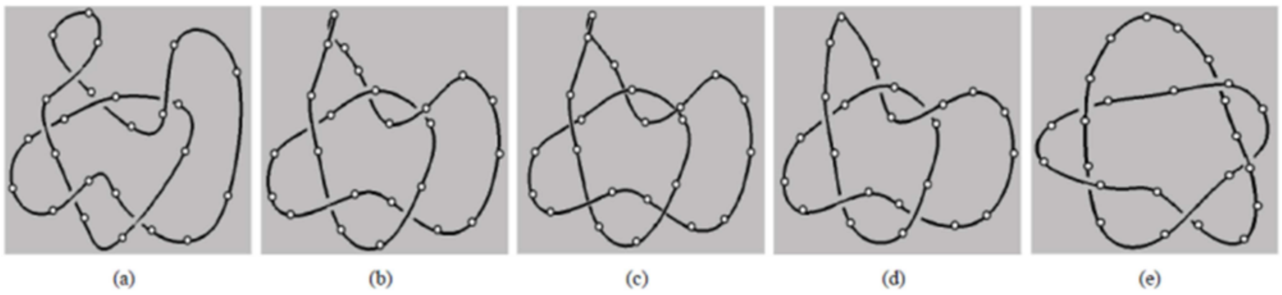


Figure 5. *Typical screen images of the self-deformation. The simple closed curve (a knot 5^1) relaxes, with the proposed force laws and collision avoidance mechanism. During the relaxation, the knotted string preserves the its topological structure.*

The vertices are placed in some initial layout and let go so that the spring systems and electrical forces on the masses move the system to a minimal energy state. Two basic forces are used, an attractive mechanical force applied between adjacent masses on the same spring and a repulsive electrical force applied between all other pairs of masses:

1. attractive mechanical force — the mechanical force is a generalization of Hooke's law, allowing for an arbitrary power of the distance r between masses, $Fm = Hr^1+b$, where H is a constant;
2. repulsive electrical force — the electrical force also allows for a general power of the distance, $Fe = Kr-(2+a)$, where r again is the distance between the two masses, and K is a constant. The electrical force is applied to all pairs of masses excluding those consisting of adjacent masses on the same link.

In most of the preliminary results [4], [5], [6] shown in this work, the parameters used • = 1 and • = 2.

For this force-directed algorithm to be applicable to the principal test case of mathematical curves positioned in R3, it is imperative that any proposed evolution should respect topological constrains: it does not involve cutting the curve or passing the curve through itself. Parallel to the force laws previously specified, the self-intersection problem is solved in the proposed approach by requiring that the position of each mass be updated one at a time, and collision avoidance is strictly performed to determine if one is heading towards one of the following two potential collisions:

1. point-segment collision — a vertex of a 3D curve is going towards a link of the curve and the distance is less than a predefined threshold distance
2. segment-segment collision — a link of a 3D curve is going towards another link and their distance is less than a predefined threshold distance

6.1. Accelerating the Force-Driven Knot Simulation

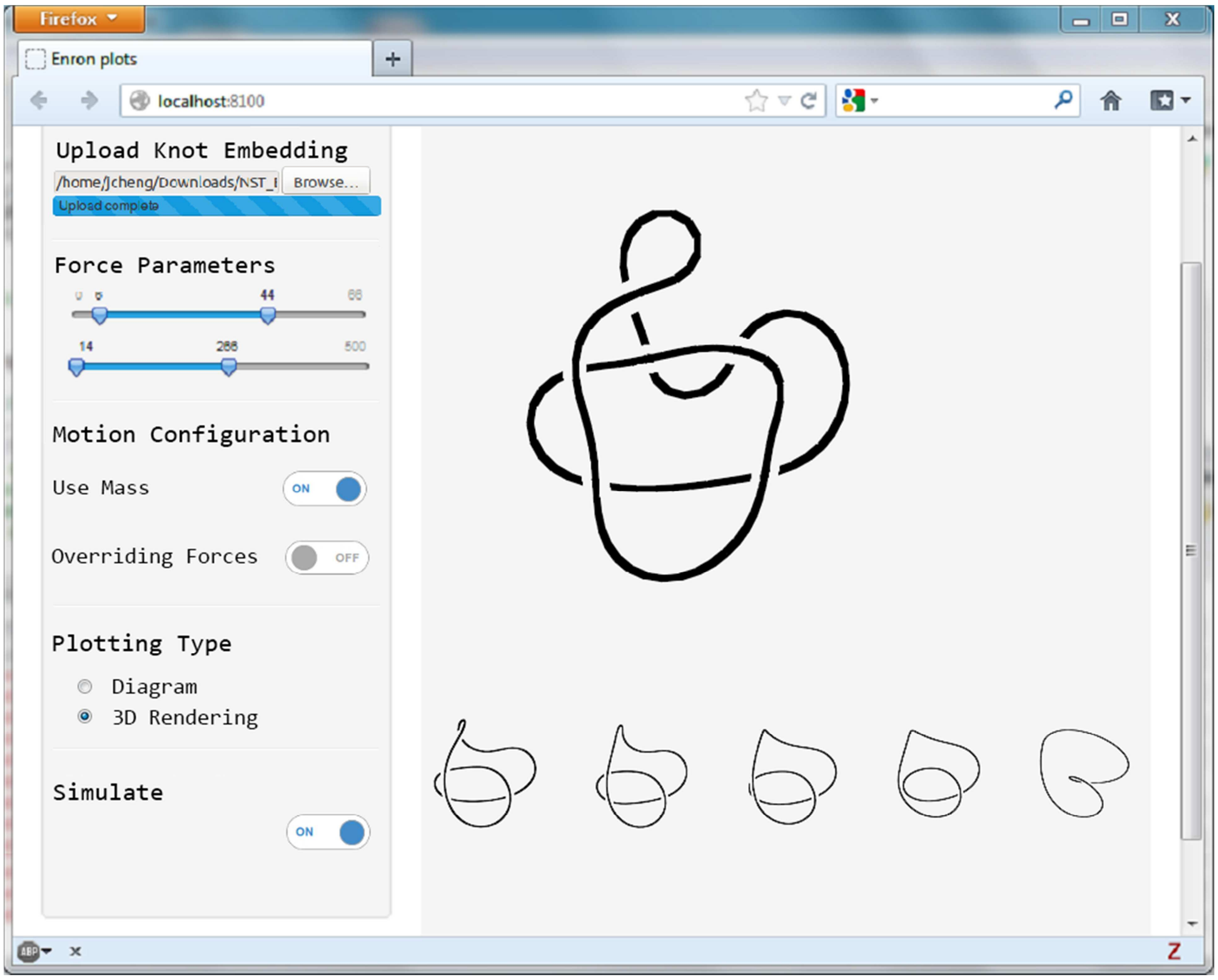
A good portion of the algorithms is concerned with Linear Algebra Computation (LAC), and heavily vectorized operations performed over and over in a large number of iterations. One promising direction for accelerating the computation is to utilize the latest hardware advance and exploit hardware-enabled massive parallelism to accelerate the LAC in the force-driven knot algorithms.

The compute-intensive part of the knot simulation algorithm is for distance calculation. Both point-segment collision and segment-segment collision avoidance are heavily relying on distance calculation between points and line segments in 3-dimensional space. The core algorithms are implemented and optimized with vectorized and matrixized R code and exploit GPGPU to accelerate the core computational components. We have recently completed a preliminary study on accelerating R computation with hardware enabled parallelism [3], and obtained promising results by adopting this technology in several domain-specific scientific investigations [7], [8], [9]. Meanwhile, many geometry studies require the examination of phenomena under different conditions, e.g., with different relaxation models, or with different intervening forces. Such tasks and simulations can be executed in a pleasing parallel way, which can be accelerated by parallel computing on multi-core and multimode data infrastructure (see e.g., initial results in [10], [7], [11].)

6.2. Extracting the Key Moments

Extracting key moments is a simple yet effective form of summarizing a long mathematical evolution or comparing among multiple evolutions. In most applications of topological refinement, the interests do not include each step in the path followed by the object model. Rather, those key moments and the final conformation are of greater importance for investigation and presentation. The fascinating question here is whether there is a way to extract the key moments of the deformations in high dimensions by identifying the sequence of "frames" where each item differs by one critical topological change.

In the case scenario, the critical changes can be computed in the knot presentations (e.g., see work in [4], [12]) by identifying the minimal number of crossing points among all possible 2D projections. If the number of crossing points changes, the evolution is considered at the new critical moment. In this way, w associated key moments can be identified and provide a much clearer visualization and navigation interfaces for users to perceptualize the entire evolution process (see e.g., Figure 6).



Upload Knot Embedding
/home/jcheng/Downloads/NST_I
Upload complete

Force Parameters
0 5 44 88
14 288 500

Motion Configuration
Use Mass ON
Overriding Forces OFF

Plotting Type
Diagram
[✓ 3D Rendering]

Simulate ON

Figure 6. *MathSimWeb: a web-based interface to define mathematical knots' initial embedding, and generate knot images for the entire evolution.*

6.3. MathSimWeb: Putting Simulation and Visualization Together

The next focus is effective integration of algorithms and techniques so far, to enable and enrich users' mathematical experience with knot geometry and topology. Figure 6 shows a visual analysis system, called MathSimWeb, developed for exploring geometric data. This work to fully connect guided geometric relaxation (with a multi-view interface) and hardwareenabled accelerated computing (offloading math operations to GPGPU) for exploring new geometry.

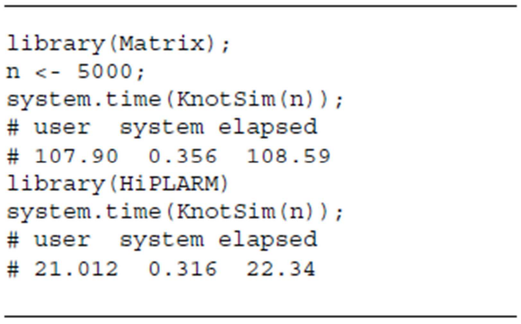
MathSimWeb leverages R Shiny's architecture deployed on a local lab cluster environment at UofL, and it will consist of two main parts:

1. a back-end module that exploits massively parallel solutions for geometry computation and ingests visualization archive (vectors) into a data store;
2. a front-end module that allows investigative geometric analysis at run time
   I. a central visualization panel that displays the geometry and its real-time evolution, and mathematical movies that can depict geometry's evolution with identified key moments.
   II. a dashboard for users to upload knot embedding and configure parameters for the simulations

General purpose graphic processing units (GPGPU) extend parallel functions and technologies traditionally embedded in graphic processing units to handle more generic computations. The Matrix package currently uses the BLAS and LAPACK linear algebra libraries to perform its operations. These are the de-facto libraries for performing linear algebra operations, however, they are only designed to run on single core CPUs and are not designed for modern CPU architectures and accelerators.

To accelerate the large number of iterations in the mathematical simulations, the library of High Performance Linear Algebra in R (HIPLAR) is used. HiPLAR's over-arching goal is to provide easy access to the latest computational architectures using the latest linear algebra libraries and to do so in an easy and user friendly manner. With the installed suite of R packages the user can achieve large speed-ups with little understanding of the complexities of multi-core and GPU computing. Another important feature of HiPLAR is its auto-tuning capability: users that have multi-core CPUs and an NVIDIA GPU can use an auto-tuning feature that calculates the optimal configuration for the problem and the hardware. Currently, HiPLAR provides two packages to target linear algebra functions in the standard R release and Matrix packages. Using the PLASMA library for multi-core CPUs and the MAGMA library for NVIDIA GPUs users can see large speed-up in R codes that use linear algebra routines from simple matrix multiplication to Cholesky and LU decomposition.

Figure 7 shows an example where a minimal change (one liner) is applied to the math simulation R program to leverage substantial speed-ups by HiPLAR.



```
library(Matrix);
n <- 5000;
system.time(KnotSim(n));
# user  system elapsed
# 107.90  0.356  108.59
library(HiPLARM)
system.time(KnotSim(n));
# user  system elapsed
# 21.012  0.316  22.34
```

Figure 7. *A brief example to highlight the benefits of using HiPLARM in accelerating mathematical simulations.*

7. Conclusion

The ultimate goal is to facilitate the manipulation and understanding of geometric structures. We now possess interactive graphics tools, computational algorithms, and increased computing powers that can extremely simplify and accelerate the process of making analogues diagrams, generating dynamic illustrations, and rendering perceptual clues, even for abstract mathematical entities and phenomena that occur in three- and high-dimensional space. By exploiting such tools, we feel that we can make a novel contribution to building intuition about classes of geometric and topological problems even beyond the third dimension.

Future directions of this work include extending the range of objects for which we can support to include more complex knots, links, and Riemann surfaces, and the use of divide-and-conquer strategies [13, 14] to accelerate large-scale mathematical simulations.

Acknowledgements

The computational experiments used Stampede at the Extreme Science and Engineering Discovery Environment (XSEDE) (resource allocation Award Number TGASC130037). The study was partly funded by NSF awards #1651581 and #1726532.

References

[1] Accelerating the intel math kernel library, 2007. M. Intel. Intel math kernel library, 2007.

[2] A hardware accelerator for the Intel Math Kernel. J. L. Gustafson and B. S. Greer. ClearSpeed whitepaper.

[3] Y. El-Khamra, N. Gaffney, D. Walling, E. Wernert, W. Xu, and H. Zhang. Performance evaluation of r with intel xeon phicoprocessor. In Big Data, 2013 IEEE International Conference on, pages 23–30. IEEE, 2013.

[4] Hui Zhang, Sidharth Thakur, and Andrew J. Hanson. Haptic exploration of mathematical knots. In ISVC (1), pages 745–756, 2007.

[5] Lin Jing, Xipei Huang, Yiwen Zhong, Yin Wu, and Hui Zhang. Python based 4d visualization environment. International Journal of Advancements in Computing Technology, 4 (16):460–469, September 2012.

[6] Hui Zhang, Jianguang Weng, and Andrew J. Hanson. A pseudo-haptic knot diagram interface. In *Proc. SPIE*, volume 7868, pages 786807–786807–14, 2011.

[7] Guangchen Ruan and Hui Zhang. *Conquering Big Data with High Performance Computing*, chapter Large-Scale Multimodal Data Exploration with Human in the Loop. Springer International Publishing, Springer International Publishing Switzerland, 2016.

[8] Jian Zou and Hui Zhang. *Conquering Big Data with High Performance Computing*, chapter High-Frequency Financial Analysis through High Performance Computing. Springer International Publishing, Springer International Publishing Switzerland, 2016.

[9] Weijia Xu, Ruizhu Huang, and Hui Zhang. *Conquering Big Data with High Performance Computing*, chapter Empowering R with High Performance Computing Resources for Big Data Analytics. Springer International Publishing, Springer International Publishing Switzerland, 2016.

[10] Hui Zhang, Huian Li, Michael J. Boyles, Robert Henschel, Eduardo Kazuo Kohara, and Masatoshi Ando. Exploiting hpc resources for the 3d-time series analysis of caries lesion activity. In *Proceedings of the 1st Conference of the Extreme Science and Engineering Discovery Environment: Bridging from the eXtreme to the Campus and Beyond*, XSEDE '12, pages 19:1–19:8, New York, NY, USA, 2012. ACM.

[11] Hui Zhang, Michael J. Boyles, Guangchen Ruan, Huian Li, Hongwei Shen, and Masatoshi Ando. Xsede-enabled highthroughput lesion activity assessment. In *Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery*, XSEDE '13, pages 10:1–10:8, New York, NY, USA, 2013. ACM.

[12] Hui Zhang, Jianguang Weng, and Guangchen Ruan. Visualizing 2-dimensional manifolds with curve handles in 4d. *IEEE Transactions on Visualization and Computer Graphics*, 20 (12):2575–2584, Dec 2014.

[13] Riqing Chen and Hui Zhang. *Large-scale 3D Reconstruction with an R-based Analysis Workflow*. In Proceedings of the Fourth IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT '17). ACM, New York, NY, USA.

[14] Hui Zhang, Yiwen. Zhong and Juan Lin, *Divide-and-conquer strategies for large-scale simulations in R*, 2017 IEEE International Conference on Big Data (Big Data), Boston, MA, 2017, pp. 3517-3523.