Report

How to talk new computers into working harder

Author(s):
Vömel, Christof

Publication Date:
2012

Permanent Link:
https://doi.org/10.3929/ethz-a-006824731 →

HOW TO TALK NEW COMPUTERS INTO WORKING HARDER

CHRISTOF VÖMEL[‡]

Abstract. In their 2008 Swiss Computer Science Challenge Award competition, the Hasler Foundation asks applicants to identify current grand challenges in computer science. In this current submission, we discuss how novel hybrid, heterogeneous high performance computing architectures as exemplified by IBM's Roadrunner pose enormous usability and productivity problems to programmers and users. We argue that addressing these difficulties is important as this kind of architectures will become more popular not only in high-end supercomputing but also in the mainstream. We show that significant advances are required in a variety of areas including programming models and languages, compilers, runtime systems, and debugging and performance analysis tools. In conclusion, the computer science community as a whole needs to undertake a collaborative effort to make hybrid computing more usable for real-world scientific applications and thus beneficial for society.

1. Problem statement. On May 25, 2008, the 'Roadrunner' supercomputer [7] installed at the Los Alamos National Laboratories, NM, USA, broke for the first time in history the mythical 'petaflop barrier.' It achieved the almost unimaginable performance of about 1.026×10^{15} floating point operations per second [28, 37] on the LINPACK benchmark [25], making it the world's most powerful computer [24] and 'roughly equivalent to the combined computing power of 100,000 of today's fastest laptop computers' [19].

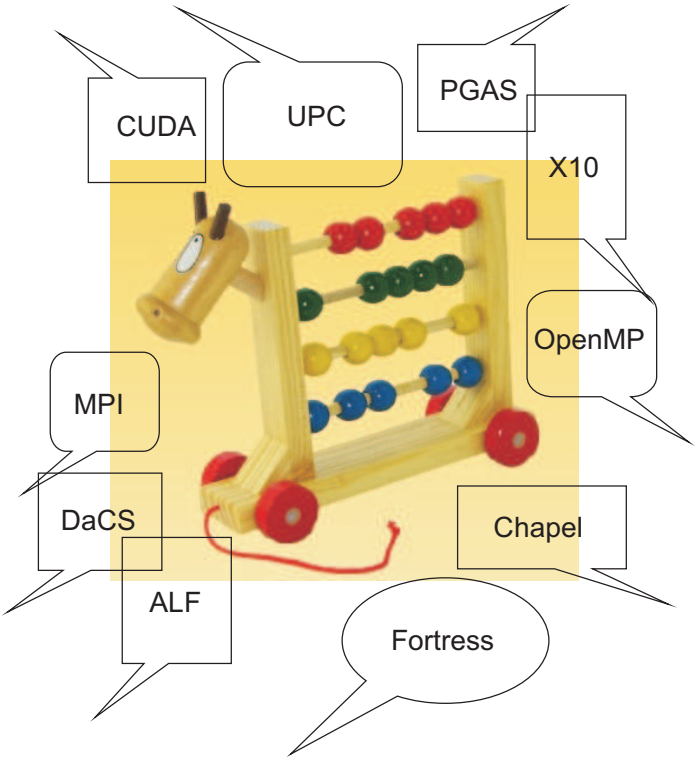

FIG. 1.1. *How should one talk new computers into working harder? [6]*

[‡]Institute of Computational Science, ETH Zürich, CAB, Universitätsstraße 6, 8092 Zürich, Switzerland, cvoemel@inf.ethz.ch

Of specific interest is here the particular hybrid architecture of Roadrunner [19]: it consists of AMD dual-core Opteron chips (as one would find them in a desktop) as well as Sony Cell Broadband microprocessors (which in turn consist of a Power Processor Element augmented with eight Synergistic Processing Elements each and were first used in the Playstation 3). One may argue whether this hybrid, heterogeneous design can be considered a herald, a precursor of future commodity systems. Yet, judging by the technical presentations at leading conferences such as Supercomputing [4], there is an undeniable trend of performing computations on a variety of devices that not only include multicores but also the cell microprocessor or graphics processing units.

From an applications points of view, it is very desirable to leverage the individual strengths of each of these devices to the benefit of an overall more performant or economic simulation. One striking example for such gains has been highlighted by the 2006 Supercomputing Best Paper awarded for 'Scalable Algorithms for Molecular Dynamics Simulations on Commodity Clusters' [13].

However, users and programmers must indeed be brave to face this new world. Already it is well recognized that it is hard to 'write programs that execute efficiently on highly parallel computing systems' [11]. Moreover, heterogeneity from joining various different devices into a 'single computer' does amplify such existing problems to a new level. In [53], the director of the NERSC computing center criticizes that 'the race for each major performance milestone, [...] has resulted in a de-emphasis of software.' Indeed, in a worst case scenario, each device will require its own programming, execution, and performance models, resulting in a quasi-Babylonian confusion that imposes substantial investments in code and algorithm design and that hampers efficient code execution within an acceptable percentage of peak-performance. In a case study [41] on four applications, the authors report that up to 50% of the existing code base (in the best case: only 10%, on an already highly optimized code) had to be modified to adapt the software for Roadrunner. A description of some modifications such as the implementation of a specialized Message Passing Relay via IBM's proprietary Data Communications and Synchronization interface [50] is given in [8], by the example of a Particle-In-Cell code VPIC. The authors also report a measured sample performance of 15% of peak, about a factor five smaller than for the LINPACK benchmark. Based on these findings, we claim that at the current stage, there is no way to solve realistic and relevant scientific problems on such novel computers at a reasonable level of abstraction without incurring a significant performance penalty. We therefore consider it an urgent matter to address this usability and productivity problem and to make solutions available to users across scientific disciplines.

2. Difficulty and impact. Before the advent of Roadrunner, scalability issues on existing massively parallel computers inspired the DARPA High Productivity Computing Systems project that is 'focused on providing a new generation of economically viable high productivity computing systems for national security and for the industrial user community' [2]. In [33], the director of the Mathematics and Computer Science Division at Argonne National Laboratory emphasizes the cross-disciplinary requirements regarding compilers, runtime systems, debugging and performance analysis tools. Arguably, all these issues become even more serious in the context of hybrid (or: heterogeneous) hardware, further illustrating the difficulties already mentioned in Section 1.

In [28], IBM's chief engineer for Roadrunner summarizes the issues as follows: 'The ability to optimize memory flow across the system will be the critical factor in unleashing the performance from these hybrid machines. "This feels very similar

to the change we made when we went from shared memory to distributed memory. [...] Now we have to figure out how to get around this memory bandwidth wall and heterogeneous cores."'

On the other hand, addressing these problems promises to provide large benefits to computational science and in consequence to society in general. One success story, scalable molecular dynamics on commodity clusters, has been given in Section 1. To stress the impact on society, we quote the US National Science Foundation Blue Ribbon Panel on Simulation-Based Engineering Science who state nothing less than their belief that 'the computational and simulation engineering sciences are fundamental to the security and welfare of the United States' [38].

It thus seems reasonable to conclude that a substantial enhancement of productivity on heterogeneous computers is both difficult and important from the scientific and the societal point of view.

3. Related work. In order to harness massively parallel computing, there are two major schools of thought, one advocating a mixed MPI [29, 40]- OpenMP [20] strategy, the other one promoting a partitioned global address space approach like for example Universal Parallel C [14]. Furthermore, in the framework of the HPCS project mentioned in Section 2, DARPA promotes the development of next-generation HPC languages [39, 47] including Cray's Chapel [1], IBM's X10 [5], and Sun's Fortress [3].

As appealing as this might sound, none of these languages seems prepared to allow high-level programming of hybrid computers at this point. On the contrary, for Roadrunner one is currently limited to IBM's proprietary Accelerator Library Framework which provides wrappers for parallel kernels, and to their Data Communications and Synchronization interface [49, 50, 51, 52]. In fact, the IBM chief engineer for Roadrunner concedes in [28] that 'the software model they have constructed is just a start for making hybrid systems, like Roadrunner, easily programmable. When you combine multiple computing technologies (i.e., heterogeneous instruction sets, multicore processors, vector SIMD units, local memory stores, explicit DMA, on-chip CPU/memory networks, remote accelerators and cluster computing) the developer is going to need a framework that provides some level of hardware independence. For the first cut at this, IBM decided to go the library route as a relatively easy way to glue together the different binaries and help take the complexity out of the heterogeneity. Later versions could involve new programming languages and compiler/runtime technologies.'

4. Personal involvement. My personal interests in hybrid computers are those of a concerned programmer and user. I am working in the field of High Performance Computing and am seriously worried about the perspective of loosing more and more of my human time on trying to achieve reasonable performance on novel architectures such as Roadrunner. From past experience, I am convinced that similar designs will appear gradually and become ubiquitous eventually. In a historic assessment of the TOP500 supercomputer benchmark [36], it is noted that in general, 'it will take six to eight years for any system to move from position one to 500 and eight to ten years to move from position 500 to notebook level.'

Some of my previous related work includes a Sparse BLAS implementation [27], adapting numerical software to Symmetric Multi-Processors [9], as well as providing software [22, 23, 34, 35, 44, 45] for the LAPACK [10, 32] and ScaLAPACK [18] libraries. Last but not least, I worked on automatic performance tuning [31] which may be one of a number of interesting approaches to addressing part of the issue, see Section 5.

5. Possible directions of research. The previous discussion indicates that in order to enable higher productivity and better usability of hybrid computers, one needs to address issues on a variety of levels including those mentioned in Section 2, 'compilers, runtime systems, debugging and performance analysis tools.' It certainly is ambitious to address all of these within a time frame of 2-3 years.

However, there are three aspects which are of particular interest in my own research. With adequate funding to buy a small scale hybrid computer and to allow collaborations with colleagues, the following pieces of research could be conducted within that time.

First, to improve ease of use, one has to increase the abstraction level at which algorithms can be formulated. In an ideal world, a programmer or user could afford to be oblivious to the underlying architecture whereas the current reality on a hybrid computer is exactly the opposite. With the Formal Linear Algebra Method Environment [12, 42, 43], there already exists one established mechanism for expressing and deriving computational algorithms. It seems promising to study whether these ideas can be beneficially applied in the context of hybrid architectures. One possible collaborator on this would be Professor Bientinesi from RWTH Aachen.

Second, one has to reinvestigate existing algorithms in order to better expose fine grain parallelism and data locality so that a future optimizing compiler could take advantage of them. Some initial work in this direction, aimed at multicore and cell architectures individually, has been done by the group of Professor Dongarra at the University of Tennessee [15, 16, 17]. It is natural to research if and how this work can be extended to the hybrid combination of the two architectures.

Third, one should leverage part of the available computing cycles to automatically tune performance-critical parameters if they can be identified for the application at hand. With the program design space becoming larger and larger, one should involve automation in the development process as much as possible. The Atlas [26, 48] and the Sparsity [30]/Oski [21, 46] projects establish successful precedents for this concept. It is an interesting topic of research to study its usefulness on a hybrid architecture. One possible collaborator in this research would be Professor Vuduc from Georgia Tech.

While advances in the aforementioned directions of research will not alone solve the complicated problem outlined here, they do constitute important pieces of the overall mosaic. Hybrid architectures are a reality as supercomputers now, and we expect them to become widespread as commodity architectures within the next 10-15 years. Making them usable for real-world applications and thus beneficial for society will be a rewarding effort involving large parts of the computer science community.

REFERENCES

[1] Chapel, The Cascade High-Productivity Language. http://chapel.cs.washington.edu/.
[2] DARPA High Productivity Computing Systems. http://www.highproductivity.org/.
[3] Fortress. http://research.sun.com/projects/plrg/Fortress/overview.html.
[4] The International Conference for High Performance Computing, Networking, Storage, and Analysis. http://www.supercomputing.org/.
[5] The X10 Programming Language. http://domino.research.ibm.com/comm/research_projects.nsf/pages/x10.index.html.
[6] Abakus Esel Holzspielzeug (Wooden donkey abakus). Original image from http://www.beefoxy.de, an online store for wooden toys. Edited with Corel Draw X4 by CV, 2008.
[7] Los Alamos Lab: High Perfomance Computing: Roadrunner. http://www.lanl.gov/orgs/hpc/roadrunner/, 2008.
[8] B. Albright, K. Bowers, and Y. Lin. Roadrunner Technical Seminar Series.

Adapting VPIC to Roadrunner. Los Alamos National Laboratory LA-UR-08-2672. http://www.lanl.gov/orgs/hpc/roadrunner/pdfs/Bergen - VPIC/RRVPIC.pdf, 2008.

[9] P. R. Amestoy, I. S. Duff, S. Pralet, and C. Vömel. Adapting a parallel sparse direct solver to architectures with clusters of SMPs. *Parallel Computing*, 29:1645–1668, 2003.

[10] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK User's Guide*. SIAM, Philadelphia, 3. edition, 1999.

[11] K. Asanovic, R. Bodik, B. Catanzaro, J. Gebis, P. Husbands, K. Keutzer, D. Patterson, W. Plishker, J. Shalf, S. Williams, and K. Yelick. The landscape of parallel computing research: A view from berkeley. Technical Report UCB/EECS-2006-183, Dept. of Electrical Engineering and Computer Sciences, UC Berkeley, 2006.

[12] P. Bientinesi, E. S. Quintana-Ortí, and R. A. van de Geijn. Representing Linear Algebra Algorithms in Code: The FLAME Application Programming Interfaces. *ACM Trans. Math. Software*, 31(1):27–59, 2005.

[13] K. J. Bowers, E. Chow, H. Xu, R. O. Dror, M. P. Eastwood, B. A. Gregersen, J. L. Klepeis, I. Kolossvary, M. A. Moraes, F. D. Sacerdoti, J. K. Salmon, Y. Shan, and D. E. Shaw. Scalable Algorithms for Molecular Dynamics Simulations on Commodity Clusters. Proceedings of the ACM/IEEE SC2006 Conference, Tampa, Florida, http://sc06.supercomputing.org/schedule/pdf/pap259.pdf, 2006.

[14] The Berkeley UPC Project. http:upc.lbl.gov, 2006.

[15] A. Buttari, J. Dongarra, and J. Kurzak. LAPACK Working Note 185. Limitations of the PlayStation 3 for High Performance Cluster Computing. Technical report ut-cs-07-597, UT Knoxville, 2007.

[16] A. Buttari, J. Langou, J. Kurzak, and J. Dongarra. LAPACK Working Note 190. Parallel Tiled QR Factorization for Multicore Architectures. Technical report ut-cs-07-598, UT Knoxville, 2007.

[17] A. Buttari, J. Langou, J. Kurzak, and J. Dongarra. LAPACK Working Note 191. A Class of Parallel Tiled Linear Algebra Algorithms for Multicore Architectures. Technical report ut-cs-07-600, UT Knoxville, 2007.

[18] J. Choi, J. Demmel, I. Dhillon, J. Dongarra, S. Ostrouchov, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. ScaLAPACK: A portable linear algebra library for distributed memory computers - design issues and performance. *Computer Physics Communications*, 97:1–15, 1996.

[19] M. Corrado. Fact Sheet & Background: Roadrunner Smashes the Petaflop Barrier. http://www-03.ibm.com/press/us/en/pressrelease/24405.wss, 2008.

[20] L. Dagum and R. Menon. OpenMP: An Industry-Standard API for Shared Memory Programming. *Comp. Sci. Eng.*, 5(1):46–55, 1998.

[21] J. Demmel, J. Dongarra, V. Eijkhout, E. Fuentes, A. Petitet, R. Vuduc, R.C. Whaley, and K. Yelick. Self-Adapting Linear Algebra Algorithms and Software. *Proceedings of the IEEE*, 93(2):293–312, Feb. 2005.

[22] J. W. Demmel, O. A. Marques, B. N. Parlett, and C. Vömel. Performance and Accuracy of LAPACK's Symmetric Tridiagonal Eigensolvers. *SIAM J. Sci. Comput.*, 30(3):1508–1526, 2008.

[23] I. S. Dhillon, B. N. Parlett, and C. Vömel. The design and implementation of the MRRR algorithm. *ACM Trans. Math. Software*, 32(4):533–560, 2006.

[24] J. Dongarra, H. Meuer, E. Strohmaier, and H. Simon. 31^{st} TOP500 List of World's Most Powerful Supercomputers Topped by World's First Petaflop/s System. http://www.top500.org, 2008.

[25] J. J. Dongarra, P. Luszczek, and A. Petitet. The LINPACK Benchmark: past, present and future. *Concurrency and Computation: Practice and Experience*, 15(9):803–820, 2003.

[26] J. J. Dongarra and R. C. Whaley. Automatically tuned linear algebra software. Technical Report UT-CS-97-366, University of Tennessee, 1997.

[27] I. S. Duff and C. Vömel. Algorithm 818: A Reference Model Implementation of the Sparse BLAS in Fortran 95. `http://www.netlib.org/netlib/toms/818`, 2002.

[28] M. Feldman. IBM Roadrunner Takes the Gold in the Petaflop Race. http://www.hpcwire.com/topic/processors/IBM_Roadrunner_Takes_the_Gold_in_the_Petaflop_Race.html, 2008.

[29] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: Portable Programming with the Message Passing Interface*. MIT Press, Cambridge, MA, 1994.

[30] E. Im, K. Yelick, and R. Vuduc. Sparsity: Optimization Framework for Sparse Matrix Kernels. *Int. J. High Perf. Comp. Appl.*, 18:135–158, 2004.

[31] T. Katagiri, C. Vömel, and J. W. Demmel. Automatic Performance Tuning for the Multi-section with Multiple Eigenvalues Method for Symmetric Tridiagonal Eigenproblems. In *Applied*

Parallel Computing. State of the Art in Scientific Computing, pages 938–948. Springer's Lecture Notes in Computer Science, LCNS-4699, 2007.

[32] LAPACK 3.1. http://www.netlib.org/lapack/lapack-3.1.0.changes, 2006.

[33] E. Lusk. HPCS Languages Move Forward: A Q&A with Rusty Lusk. http://www.hpcwire.com/topic/developertools/HPCS_Languages_Move_Forward_A_QA_with_Rusty_Lusk.html, 2006.

[34] O. A. Marques, B. N. Parlett, and C. Vömel. Computations of Eigenpair Subsets with the MRRR algorithm. *Numerical Linear Algebra with Applications*, 13(8):643–653, 2006.

[35] O. A. Marques, C. Vömel, J. W. Demmel, and B. N. Parlett. Algorithm 880: A Testing Infrastructure for Symmetric Tridiagonal Eigensolvers. *ACM Trans. Math. Software*, 35(1), 2008.

[36] H. W. Meuer. The TOP500 Project: Looking Back Over 15 Years of Supercomputing Experience. *Informatik Spektrum*, 31(3):203–222, 2008.

[37] J. Niccolai. Q&A: What Roadrunner's petaflop Top500 milestone is all about. http://www.computerworld.com/action/article.do?command=viewArticleBasic&taxonomyName=servers&articleId=9099718&taxonomyId=&intsrc=kc_feat, 2008.

[38] J. T. Oden, T. Belytschko, J. Fish, T. J. R. Hughes, C. Johnson, D. Keyes, A. Laub, L. Petzold, D. Srolovitz, and Y. Sidney. Simulation-Based Engineering Science. Report of the National Science Foundation Blue Ribbon Panel on Simulation-Based Engineering Science, May 2006.

[39] A.G. Shet, W.R. Elwasif, R.J. Harrison, and D.E. Bernholdt. Programmability of the HPCS Languages: A Case Study with a Quantum Chemistry Kernel. *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pages 1–8, April 2008.

[40] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. J. Dongarra. *MPI: The Complete Reference*. MIT Press, Cambridge, MA, USA, 1996.

[41] J. A. Turner. The Los Alamos Roadrunner Petascale Hybrid Supercomputer. Overview of Applications, Results, and Programming. Los Alamos National Laboratory LA-UR-08-2412. http://www.lanl.gov/orgs/hpc/roadrunner/pdfs/Turner - RR Apps, Programming,& Results/RR_Seminar_Apps_Results_Programming.pdf, 2008.

[42] R. A. van de Geijn and E. S. Quintana-Ortí. *The Science of Programming Matrix Computations*. http://www.lulu.com, 1. edition, 2008.

[43] F. G. Van Zee, P. Bientinesi, T. M. Low, and R. A. van de Geijn. Scalable Parallelization of FLAME Code via the Workqueuing Model. *ACM Trans. Math. Software*, 34(2):Article 10, 29 pages, 2008.

[44] C. Vömel. LAPACK Working Note 194. A refined representation tree for MRRR. Technical report, University of Tennessee, Knoxville, TN, USA, 2007.

[45] C. Vömel. LAPACK Working Note 195. ScaLAPACK's MRRR algorithm. Technical report, University of Tennessee, Knoxville, TN, USA, 2007.

[46] R. Vuduc, J. Demmel, and K. Yelick. Oski: A library of automatically tuned sparse matrix kernels. *J. Phys.: Conf. Ser.*, 16:521–530, 2005.

[47] M. Weiland. Chapel, Fortress and X10: novel languages for HPC. Technical Report HPCxTR0706, The HPCx Consortium, 2007.

[48] R. C. Whaley, A. Petitet, and J. Dongarra. Automated Empirical Optimization of Software and the ATLAS Project. *Parallel Computing*, 27(1-2):3–35, 2001.

[49] C. Wright. Roadrunner Tutorial. Cell Broadband Engine - Architecture and Programming. Los Alamos National Laboratory LA-UR-08-2820. http://www.lanl.gov/orgs/hpc/roadrunner/pdfs/Roadrunner-tutorial-session-2-web1.pdf, 2008.

[50] C. Wright. Roadrunner Tutorial. Hybrid Programming: DaCS and ALF Examples. Los Alamos National Laboratory LA-UR-08-2817. http://www.lanl.gov/orgs/hpc/roadrunner/pdfs/Roadrunner-tutorial-session-5-web1.pdf, 2008.

[51] C. Wright. Roadrunner Tutorial. IBM Software Deveopment Kit for Multicore Acceleration. Los Alamos National Laboratory LA-UR-08-2819. http://www.lanl.gov/orgs/hpc/roadrunner/pdfs/Roadrunner-tutorial-session-3-web1.pdf, 2008.

[52] C. Wright, P. Henning, and B. Bergen. Roadrunner Tutorial. An Introduction to Roadrunner and the Cell Processor. Los Alamos National Laboratory LA-UR-08-2818. http://www.lanl.gov/orgs/hpc/roadrunner/pdfs/Roadrunner-tutorial-session-1-web1.pdf, 2008.

[53] K. Yelick. The Software Challenges of Petascale Computing. http://www.lbl.gov/CS/Archive/news111006.html.