Chapter 2

PROGRAMMING MODELS AND RUNTIMES

Georges Da Costa[1] and Alexey L. Lastovetsky[2] and Jorge G. Barbosa[3] and Juan C. Díaz-Martín[4] and Juan L. García-Zapata[5] and Matthias Janetschek[6] and Emmanuel Jeannot[7] and João Leitão[8] and Ravi Reddy Manumachu[9] and Radu Prodan[10] and Juan A. Rico-Gallego[11] and Peter Van Roy[12] and Ali Shoker[13] and Albert van der Linde[14]

Keywords: Performance Modelling, Energy Modelling, Heterogeneous Platforms, Optimization Techniques, Cloud Computing; Edge Computing; Process Placement; Graph Partition; Sustainability; Energy awareness; Availability; Scalability.

Several millions of execution flows will be executed in Ultrascale Computing Systems (UCS), and the task for the programmer to understand their coherency and for the runtime to coordinate them is unfathomable. Moreover, in link with USC large scale and their impact on reliability the current static point of view is not more sufficient. A runtime cannot consider to restart an application because of the failure of a single node as statically several nodes will fail every days. Classical management of these failures by the programmers using checkpoint-restart are also too limited due to the overhead at such scale.

[1]University of Toulouse, France
[2]University College Dublin, Ireland
[3]Faculdade de Engenharia da Universidade do Porto, Portugal
[4]University of Extremadura, School of Technology, Spain
[5]University of Extremadura, School of Technology, Spain
[6]Institute of Computer Science, University of Innsbruck, Austria
[7]INRIA Bourdeaux Sud-Ouest, LaBRI, Université de Bourdeaux, France
[8]Universidade Nova de Lisboa, Portugal
[9]University College Dublin, Ireland
[10]Institute of Information Technology, University of Klagenfurt, Austria
[11]University of Extremadura, School of Technology, Spain
[12]Université Catholique de Louvain, Belgium
[13]HASLab, INESC TEC & University of Minho, Portugal
[14]Universidade Nova de Lisboa, Portugal

Emerging programming models that facilitate the task of scaling and extracting performance on continuous evolving platforms, while providing resilience and fault tolerant mechanisms to tackle the increasing probability of failures throughout the whole software stack, are needed to achieve scale handling (optimal usage of resources, faults), improve programmability, adaptation to rapidly changing underlying computing architecture, data-centric programming models, resilience, energy efficiency.

One key element on the ultrascale front is the necessity of new sustainable programming and execution models in the context of rapid underlying computing architecture changing. There is a need to explore synergies among emerging programming models and runtimes from HPC, distributed systems and big data management communities. To improve the programmability of future systems, the main changing factor will be the substantially higher levels of concurrency, asynchrony, failures and heterogeneous architectures.

UCS need new sustainable programming and execution models, suitable in the context of rapidly changing underlying computing architecture, as described in [1]. Advances are to be expected at three levels: Innovative programming models with higher level abstraction of the hardware; breakthrough for more efficient runtimes at large scale; cooperation between the programming models and runtime levels.

Furthermore all the programming ecosystem must evolve. A large number of scientific applications are built on the message passing paradigm which needs a global point of view during the programming phase and usually require global synchronization during execution. But even at lower granularity, classical libraries must evolve. As an example, a large number of scientists use the linear algebra BLAS libraries for their optimized behavior on current supercomputers. Improving the performance of this library on UCS would prove largely beneficial.

This chapter explores programming models and runtimes required to facilitate the task of scaling and extracting performance on continuously evolving platforms, while providing resilience and fault-tolerant mechanisms to tackle the increasing probability of failures throughout the whole software stack. However, currently no programming solution exists that satisfies all these requirements. Therefore, new programming models and languages are required towards this direction. The whole point of view on application will have to change. As we will show, the current wall between runtime and application models leads to most of these problem. Programmers will need new tools but also new way to assess their programs. Also, data will be a key concept around which failure-tolerant high number of micro-threads will be generated using high-level information by adaptive runtime. One complex element comes from the difficulty to test these approaches as UCS systems are not yet available. Most of the following explorations are extrapolated to UCS scale but only actually proven an currently existing infrastructure.

The complexity of UCS computing architecture integrating in a hierarchical heterogeneous way multicore CPUs and various accelerators makes many traditional approaches to the development of performance and energy efficient applications ineffective. New sustainable approaches based on accurate and sustainable application-level performance and energy models have a great potential to improve

the performance and energy efficiency of applications and create a solid basis for the emerging USC programming tools and runtimes. Section 2.1 of this chapter covers this topic by describing accurate models of the hardware and software usable during the design phase, but also means or reasoning on these models. With these tools, it becomes possible to adapt and tune finely applications during the design phase to run efficiently on large scale heterogeneous platforms.

Optimizing UCS usage is difficult due to the large number of possible use-cases. In particular ones such as Scientific workflow, it becomes possible to use a dedicated abstraction. As scientific workflow scheduling for UCS is a major challenge, the impact of proposing a particular abstraction along-with dedicated runtime harnessing the particularities of this abstraction leads to a high improvement of the efficiency of using a UCS. The approaches to solve this challenge are covered in section 2.2. In this section, both the *Abstract part* (linked with the design and programming of the workflow) and the *Concrete part* (linked with its actual scheduling and execution) are described. This specific high-level abstraction shows that link between programming models and runtime helps to simplify the task of programmers to harness the power of the underlying large scale heterogeneous systems.

With the emergence of UCS, a new computing revolution is coming: Edge computing. Instead of harnessing computing power directly from large scale datacenters, new proposal comes from the possibility to interconnect and coordinate large number of distributed computing nodes. Due to the explosion of IoT applications the aggregated Edge computing power is increasing extremely fast. These two systems (Edge and UCS) share the difficulty to manage large number of distributed execution flows in a dynamic and heterogeneous environment. These similarities is explored in Section 2.3 where key elements of programming models and runtime for large scale Edge computing are explored.

Due to the scale of UCS, even classical management operation of the platform becomes complex. As an example, section 2.5 shows how a simple operation such as graph partitioning becomes complex at large scale. This operation is central in the management of a platform as it is needed to minimize communication between nodes when used for placing the tasks. In this section several challenges are addressed such as the scale but also the heterogeneity of tasks, computing nodes and networking infrastructure.

This chapter concludes with a description of the main global challenges linked to programming models, runtimes and the link between these two as described in NESUS roadmap[2].

2.1 Using Performance and Energy Models for Ultrascale Computing Applications

Ultrascale systems, including high performance computing, distributed computing and big data management platforms, will demand a huge investment of heterogeneous

Tool-driven: Several tools will be needed to use efficiently UCS. Some tools can be provided by software, but also abstract models and new programming paradigms helping programmers to better use the available resources are helpful. Due to the scale of the systems, one key element will be resource-efficient models for automatic recovery from minute-to-minute failures. As security is often forgotten by programmers, software-defined security models will be needed on large scale distributed infrastructure to simplify its usage. One way to increase security and privacy will be to create new secure Privacy-Preserving data management algorithms such as machine learning. To address code sustainability and adaptation evolution on code production is needed such as source-to-source translators and MDE (Model Driven Engineering) in order to adapt to the underlying hardware.

In order to support some of those challenges, several breaktroughs are expected in order to reach proper support for programmers and users in the Ultrascale context as described in the NESUS research roadmap[2]:

Improve the programmability of complex systems Due to the size of these systems, it is no more possible for the programmer to have a precised and detailed global view of the state of its application. Thus he needs to have support from programming frameworks to simplify this view;

Break the wall between runtime and programming frameworks Exascale sytems are so complex that runtime need high level information from the programmers and the programmer need some information on the runtime to understand how to harness its power;

Enabling behavioral sensitive runtime. Runtime cannot run application as black boxes anymore as large scale systems are composed of a large number of interconnected elements. Network profile must be known to reduce impact on neighboor applications for example.

References

[1] Da Costa G, Fahringer T, Gallego JAR, et al. Exascale machines require new programming paradigms and runtimes. Supercomputing frontiers and innovations. 2015;2(2):6–27.

[2] Sousa L, Kropf P, Kuonene P, et al. A Roadmap for Research in Sustainable Ultrascale Systems. University Carlos III of Madrid; 2017. 0. Available from: https://www.irit.fr/~Georges.Da-Costa/NESUS-research_roadmap.pdf.

[3] Fortune S, Wyllie J. Parallelism in Random Access Machines. In: Proceedings of the Tenth Annual ACM Symposium on Theory of Computing. STOC '78. New York, NY, USA: ACM; 1978. p. 114–118. Available from: http://doi.acm.org/10.1145/800133.804339.

[4] Valiant LG. A Bridging Model for Parallel Computation. Commun ACM. 1990 Aug;33(8):103–111. Available from: http://doi.acm.org/10.1145/79173.79181.

[5] Culler D, Karp R, Patterson D, et al. LogP: towards a realistic model of parallel computation. In: Proceedings of the fourth ACM SIGPLAN symposium on Principles and practice of parallel programming. PPOPP '93. New York, NY, USA: ACM; 1993. p. 1–12.

[6] Gautier T, Besseron X, Pigeon L. KAAPI: A Thread Scheduling Runtime System for Data Flow Computations on Cluster of Multi-processors. In: Proceedings of the 2007 International Workshop on Parallel Symbolic Computation. PASCO '07. ACM; 2007. p. 15–23.

[7] Augonnet C, Thibault S, Namyst R, et al. StarPU: A Unified Platform for Task Scheduling on Heterogeneous Multicore Architectures. Concurr Comput : Pract Exper. 2011 Feb;23(2):187–198.

[8] Bosilca G, Bouteiller A, Danalis A, et al. DAGuE: A Generic Distributed DAG Engine for High Performance Computing. In: 2011 IEEE IPDPSW; 2011. p. 1151–1158.

[9] Bui TN, Jones C. A heuristic for reducing fill-in in sparse matrix factorization. In: Proceedings of the 6th SIAM Conference on Parallel Processing for Scientific Computing. SIAM; 1993. .

[10] Hendrickson B, Leland R. A Multilevel Algorithm for Partitioning Graphs. In: Proceedings of the 1995 ACM/IEEE Conference on Supercomputing. Supercomputing '95. ACM; 1995. Available from: http://doi.acm.org/10.1145/224170.224228.

[11] Catalyurek U, Aykanat C. Decomposing Irregularly Sparse Matrices for Parallel Matrix-Vector Multiplication. In: Proceedings of the Third International Workshop on Parallel Algorithms for Irregularly Structured Problems. IRREGULAR '96. Springer-Verlag; 1996. p. 75–86.

[12] Hendrickson B, Kolda TG. Partitioning Rectangular and Structurally Unsymmetric Sparse Matrices for Parallel Processing. SIAM Journal on Scientific Computing. 2000;21(6):2048–2072.

[13] Cierniak M, Zaki MJ, Li W. Compile-time scheduling algorithms for a heterogeneous network of workstations. The Computer Journal. 1997;40(6):356–372.

[14] Beaumont O, Boudet V, Rastello F, et al. Matrix multiplication on heterogeneous platforms. Parallel and Distributed Systems, IEEE Transactions on. 2001;12(10):1033–1051.

[15] Kalinov A, Lastovetsky A. Heterogeneous Distribution of Computations Solving Linear Algebra Problems on Networks of Heterogeneous Computers. Journal of Parallel and Distributed Computing. 2001;61:520–535.

[16] Lastovetsky AL, Reddy R. Data partitioning with a realistic performance model of networks of heterogeneous computers. In: Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International. IEEE; 2004. p. 104.

[17] Lastovetsky A, Twamley J. Towards a realistic performance model for networks of heterogeneous computers. In: High Performance Computational Science and Engineering. Springer; 2005. p. 39–57.

[18] Lastovetsky A, Reddy R. Data partitioning with a functional performance model of heterogeneous processors. International Journal of High Performance Computing Applications. 2007;21(1):76–90.

[19] Lastovetsky A, Reddy R. Data Distribution for Dense Factorization on Computers with Memory Heterogeneity. Parallel Computing. 2007 Dec;33(12).

[20] Lastovetsky A, Szustak L, Wyrzykowski R. Model-based optimization of EULAG kernel on Intel Xeon Phi through load imbalancing. IEEE Transactions on Parallel and Distributed Systems. 2017;28(3):787–797.

[21] Lastovetsky A, Reddy R. New Model-Based Methods and Algorithms for Performance and Energy Optimization of Data Parallel Applications on Homogeneous Multicore Clusters. IEEE Transactions on Parallel and Distributed Systems. 2017;28(4):1119–1133.

[22] Alexandrov A, Ionescu MF, Schauser KE, et al. LogGP: incorporating long messages into the LogP model - one step closer towards a realistic model for parallel computation. In: Proc. of the seventh annual ACM symposium on Parallel algorithms and architectures. SPAA '95. NY, USA; 1995. p. 95–105.

[23] Kielmann T, Bal HE, Verstoep K. Fast Measurement of LogP Parameters for Message Passing Platforms. In: Proceedings of the 15 IPDPS 2000 Workshops on Parallel and Distributed Processing. IPDPS '00. London, UK, UK: Springer-Verlag; 2000. p. 1176–1183.

[24] Bosque JL, Perez LP. HLogGP: a new parallel computational model for heterogeneous clusters. In: Cluster Computing and the Grid, 2004. CCGrid 2004. IEEE International Symposium on; 2004. p. 403–410.

[25] Lastovetsky A, Mkwawa IH, O'Flynn M. An accurate communication model of a heterogeneous cluster based on a switch-enabled Ethernet network. In: Parallel and Distributed Systems, 2006. ICPADS 2006. 12th International Conference on. vol. 2; 2006. p. 6 pp.–.

[26] Cameron KW, Ge R, Sun XH. $\log_m P$ and $\log_3 P$: Accurate Analytical Models of Point-to-Point Communication in Distributed Systems. Computers IEEE Transactions on. 2007;56(3):314–327.

[27] Rico-Gallego JA, Díaz-Martín JC, Lastovetsky AL. Extending τ–Lop to model concurrent MPI communications in multicore clusters. Future Generation Computer Systems. 2016;61:66 – 82.

[28] Rico-Gallego JA, Lastovetsky AL, Díaz-Martín JC. Model-Based Estimation of the Communication Cost of Hybrid Data-Parallel Applications on Heterogeneous Clusters. IEEE Transactions on Parallel and Distributed Systems. 2017 Nov;28(11):3215–3228.

[29] Clarke D, Zhong Z, Rychkov V, et al. FuPerMod: a software tool for the optimization of data-parallel applications on heterogeneous platforms. The Journal of Supercomputing. 2014;69:61– 69.

[30] Beaumont O, Boudet V, Rastello F, et al. Matrix Multiplication on Heterogeneous Platforms. IEEE Trans Parallel Distrib Syst. 2001 Oct;12(10):1033–1051. Available from: http://dx.doi.org/10.1109/71.963416.

[31] Malik T, Rychkov V, Lastovetsky A. Network-aware optimization of communications for parallel matrix multiplication on hierarchical HPC plat-

forms. Concurrency and Computation: Practice and Experience. 2016 03/2016;28:802–821.

[32] Bellosa F. The benefits of event: driven energy accounting in power-sensitive systems. In: Proceedings of the 9th workshop on ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system. ACM; 2000. .

[33] Isci C, Martonosi M. Runtime power monitoring in high-end processors: Methodology and empirical data. In: 36th annual IEEE/ACM International Symposium on Microarchitecture. IEEE Computer Society; 2003. p. 93.

[34] Economou D, Rivoire S, Kozyrakis C, et al. Full-system power analysis and modeling for server environments. In: In Proceedings of Workshop on Modeling, Benchmarking, and Simulation; 2006. p. 70–77.

[35] Basmadjian R, Ali N, Niedermeier F, et al. A methodology to predict the power consumption of servers in data centres. In: 2nd International Conference on Energy-Efficient Computing and Networking. ACM; 2011. .

[36] Bircher WL, John LK. Complete System Power Estimation Using Processor Performance Events. IEEE Transactions on Computers. 2012 Apr;61(4):563–577.

[37] Hong H Sunpyand Kim. An Integrated GPU Power and Performance Model. SIGARCH Comput Archit News. 2010 Jun;38(3):280–289.

[38] Song S, Su C, Rountree B, et al. A Simplified and Accurate Model of Power-Performance Efficiency on Emergent GPU Architectures. In: 27th IEEE International Parallel & Distributed Processing Symposium (IPDPS). IEEE Computer Society; 2013. p. 673–686.

[39] CUPTI. CUDA Profiling Tools Interface; 2018. Available from: https://developer.nvidia.com/cuda-profiling-tools-interface.

[40] Wang H, Cao Y. Predicting power consumption of GPUs with fuzzy wavelet neural networks. Parallel Computing. 2015 May;44:18–36.

[41] Top500. Top 500. The List - November 2017; 2018. Available from: https://www.top500.org/lists/2017/11/.

[42] Shao YS, Brooks D. Energy Characterization and Instruction-level Energy Model of Intel's Xeon Phi Processor. In: Proceedings of the 2013 International Symposium on Low Power Electronics and Design. ISLPED '13. IEEE Press; 2013. .

[43] Ou J, Prasanna VK. Rapid energy estimation of computations on FPGA based soft processors. In: SOC Conference, 2004. Proceedings. IEEE International; 2004. .

[44] Wang X, Ziavras SG, Hu J. System-Level Energy Modeling for Heterogeneous Reconfigurable Chip Multiprocessors. In: 2006 International Conference on Computer Design; 2006. .

[45] Al-Khatib Z, Abdi S. Operand-Value-Based Modeling of Dynamic Energy Consumption of Soft Processors in FPGA. In: International Symposium on Applied Reconfigurable Computing. Springer; 2015. p. 65–76.

[46] Lively C, Wu X, Taylor V, et al. Power-aware predictive models of hybrid (MPI/OpenMP) scientific applications on multicore systems. Computer Science-Research and Development. 2012;27(4):245–253.

[47] PAPI. Performance Application Programming Interface 5.6.0; 2018. Available from: http://icl.cs.utk.edu/papi/.

[48] Bosilca G, Ltaief H, Dongarra J. Power profiling of Cholesky and QR factorizations on distributed memory systems. Computer Science-Research and Development. 2014;29(2):139–147.

[49] Witkowski M, Oleksiak A, Piontek T, et al. Practical Power Consumption Estimation for Real Life HPC Applications. Future Gener Comput Syst. 2013 Jan;29(1).

[50] Jarus M, Oleksiak A, Piontek T, et al. Runtime power usage estimation of HPC servers for various classes of real-life applications. Future Generation Computer Systems. 2014;36.

[51] Lastovetsky A, Manumachu RR. New Model-Based Methods and Algorithms for Performance and Energy Optimization of Data Parallel Applications on Homogeneous Multicore Clusters. IEEE Transactions on Parallel and Distributed Systems. 2017;28(4):1119–1133.

[52] McCullough JC, Agarwal Y, Chandrashekar J, et al. Evaluating the Effectiveness of Model-based Power Characterization. In: Proceedings of the 2011 USENIX Conference on USENIX Annual Technical Conference. USENIXATC'11. USENIX Association; 2011. .

[53] Hackenberg D, Ilsche T, Schöne R, et al. Power measurement techniques on standard compute nodes: A quantitative comparison. In: Performance analysis of systems and software (ISPASS), 2013 IEEE international symposium on. IEEE; 2013. p. 194–204.

[54] Rotem E, Naveh A, Ananthakrishnan A, et al. Power-Management Architecture of the Intel Microarchitecture Code-Named Sandy Bridge. IEEE Micro. 2012 March;32(2):20–27.

[55] O'Brien K, Pietri I, Reddy R, et al. A Survey of Power and Energy Predictive Models in HPC Systems and Applications. ACM Computing Surveys. 2017;50(3). Available from: http://doi.org/10.1145/3078811.

[56] Shahid A, Fahad M, Reddy R, et al. Additivity: A Selection Criterion for Performance Events for Reliable Energy Predictive Modeling. Supercomputing Frontiers and Innovations. 2017;4(4).

[57] Treibig J, Hager G, Wellein G. Likwid: A lightweight performance-oriented tool suite for x86 multicore environments. In: Parallel Processing Workshops (ICPPW), 2010 39th International Conference on. IEEE; 2010. p. 207–216.

[58] Mobius C, Dargie W, Schill A. Power Consumption Estimation Models for Processors, Virtual Machines, and Servers. IEEE Transactions on Parallel and Distributed Systems. 2014;25(6).

[59] Inacio EC, Dantas MAR. A Survey into Performance and Energy Efficiency in HPC, Cloud and Big Data Environments. Int J Netw Virtual Organ. 2014 Mar;14(4).

[60] Tan L, Kothapalli S, Chen L, et al. A survey of power and energy efficient techniques for high performance numerical linear algebra operations. Parallel Computing. 2014 Dec;40.

[61] Dayarathna M, Wen Y, Fan R. Data Center Energy Consumption Modeling: A Survey. IEEE Communications Surveys & Tutorials. 2016;18(1):732–794.

[62] Mezmaz M, Melab N, Kessaci Y, et al. A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems. Journal of Parallel and Distributed Computing. 2011;71(11):1497 – 1508.

[63] Fard HM, Prodan R, Barrionuevo JJD, et al. A Multi-objective Approach for Workflow Scheduling in Heterogeneous Environments. In: Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (Ccgrid 2012). CCGRID '12. IEEE Computer Society; 2012. p. 300–309.

[64] Beloglazov A, Abawajy J, Buyya R. Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. Future Generation Computer Systems. 2012;28(5):755 – 768. Special Section: Energy efficiency in large-scale distributed systems.

[65] Kessaci Y, Melab N, Talbi EG. A Pareto-based Metaheuristic for Scheduling HPC Applications on a Geographically Distributed Cloud Federation. Cluster Computing. 2013 Sep;16(3):451–468.

[66] Durillo JJ, Nae V, Prodan R. Multi-objective energy-efficient workflow scheduling using list-based heuristics. Future Generation Computer Systems. 2014;36:221 – 236.

[67] Freeh VW, Lowenthal DK, Pan F, et al. Analyzing the Energy-Time Trade-Off in High-Performance Computing Applications. IEEE Trans Parallel Distrib Syst. 2007 Jun;18(6).

[68] Ahmad I, Ranka S, Khan SU. Using game theory for scheduling tasks on multi-core processors for simultaneous optimization of performance and energy. In: Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on; 2008. p. 1–6.

[69] Balaprakash P, Tiwari A, Wild SM. In: Jarvis AS, Wright AS, Hammond DS, editors. Multi Objective Optimization of HPC Kernels for Performance, Power, and Energy. Springer International Publishing; 2014. p. 239–260.

[70] Drozdowski M, Marszalkowski JM, Marszalkowski J. Energy trade-offs analysis using equal-energy maps. Future Generation Computer Systems. 2014;36:311–321.

[71] Marszalkowski JM, Drozdowski M, Marszalkowski J. Time and Energy Performance of Parallel Systems with Hierarchical Memory. Journal of Grid Computing. 2016;14(1):153–170.

[72] Reddy R, Lastovetsky A. Bi-Objective Optimization of Data-Parallel Applications on Homogeneous Multicore Clusters for Performance and Energy. IEEE Transactions on Computers. 2018;64(2):160–177.

[73] Juve G, Chervenak A, Deelman E, et al. Characterizing and profiling scientific workflows. Future Generation Computer Systems. 2013;29(3):682–692.

[74] Fahringer T, Prodan R, Duan R, et al. ASKALON: A Development and Grid Computing Environment for Scientific Workflows. In: Taylor IJ, Deelman E, Gannon DB, et al., editors. Workflows for e-Science. Springer; 2007. p. 450–471.

[75] Altintas I, Berkley C, Jaeger E, et al. Kepler: an extensible system for design and execution of scientific workflows. In: Proceedings. 16th International Conference on Scientific and Statistical Database Management, 2004.; 2004. p. 423–424.

[76] Tristan Glatard DLXP Johan Montagnat. Flexible and Efficient Workflow Deployment of Data-Intensive Applications On Grids With MOTEUR. The International Journal of High Performance Computing Applications. 2008;22(3):347–360.

[77] Taylor I, Shields M, Wang I, et al. Triana Applications within Grid Computing and Peer to Peer Environments. Journal of Grid Computing. 2003 Jun;1(2):199–217.

[78] Kacsuk P. P–GRADE portal family for grid infrastructures. Concurrency and Computation: Practice and Experience. 2011;23(3):235–245.

[79] Deelman E, Vahi K, Juve G, et al. Pegasus, a workflow management system for science automation. Future Generation Computer Systems. 2015;46:17–35.

[80] Janetschek M, Prodan R, Benedict S. A Workflow Runtime Environment for Manycore Parallel Architectures. Future Generation Computer Systems. 2017;75:330–347.

[81] Durillo JJ, Prodan R, Barbosa JG. Pareto tradeoff scheduling of workflows on federated commercial clouds. Simulation Modelling Practice and Theory. 2015;58:95–111.

[82] Arabnejad H, Barbosa JG. Budget constrained scheduling strategies for online workflow applications. In: International Conference on Computational Science and Its Applications. Springer; 2014. p. 532–545.

[83] Ullman JD. NP-complete scheduling problems. Journal of Computer and System sciences. 1975;10(3):384–393.

[84] Topcuoglu H, Hariri S, Wu MY. Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Transactions on Parallel and Distributed Systems. 2002 3;13(3):260–274.

[85] Wieczorek M, Hoheisel A, Prodan R. Towards a General Model of the Multi-Criteria Workflow Scheduling on the Grid. Future Generations Computer Systems. 2009;25(3):237–256.

[86] Maheswaran M, Ali S, Siegel HJ, et al. Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. Journal of parallel and distributed computing. 1999;59(2):107–131.

[87] Arabnejad H, Barbosa JG. List scheduling algorithm for heterogeneous systems by an optimistic cost table. IEEE Transactions on Parallel and Distributed Systems. 2014;25(3):682–694.

[88] Bittencourt LF, Sakellariou R, Madeira ER. Dag scheduling using a lookahead variant of the heterogeneous earliest finish time algorithm. In: Parallel,

Distributed and Network-Based Processing (PDP), 2010 18th Euromicro International Conference on. IEEE; 2010. p. 27–34.

[89] Armbrust M, Fox A, Griffith R, et al. A view of cloud computing. Communications of the ACM. 2010;53(4):50–58.

[90] Leitão J, Pereira J, Rodrigues L. Epidemic Broadcast Trees. In: Proceedings of SRDS.2007; 2007. p. 301–310.

[91] Leitão J, Pereira J, Rodrigues L. HyParView: A Membership Protocol for Reliable Gossip-Based Broadcast. In: Dependable Systems and Networks, 2007. DSN '07. 37th Annual IEEE/IFIP International Conference on; 2007. p. 419–429.

[92] Shapiro M, Preguiça N, Baquero C, et al. Conflict-free replicated data types. INRIA; 2011. RR-7687.

[93] Almeida PS, Shoker A, Baquero C. Efficient state-based crdts by delta-mutation. In: International Conference on Networked Systems. Springer; 2015. p. 62–76.

[94] Carlos Baquero PSA, Shoker A. Making Operation-Based CRDTs Operation-Based. In: Distributed Applications and Interoperable Systems - 14th IFIP WG 6.1 International Conference, DAIS 2014, Held as Part of the 9th International Federated Conference on Distributed Computing Techniques, DisCoTec 2014, Berlin, Germany, June 3-5, 2014, Proceedings; 2014. p. 126–140. Available from: http://dx.doi.org/10.1007/978-3-662-43352-2_11.

[95] Bonomi F, Milito R, Zhu J, et al. Fog Computing and Its Role in the Internet of Things. Proceedings of the first edition of the MCC workshop on Mobile cloud computing. 2012;p. 13–16. Available from: http://doi.acm.org/10.1145/2342509.2342513$\delimiter"026E30F$npapers2://publication/doi/10.1145/2342509.2342513.

[96] Yi S, Li C, Li Q. A Survey of Fog Computing: Concepts, Applications and Issues. In: Proceedings of the 2015 Workshop on Mobile Big Data. Mobidata '15. New York, NY, USA: ACM; 2015. p. 37–42. Available from: http://doi.acm.org/10.1145/2757384.2757397.

[97] Verbelen T, Simoens P, De Turck F, et al. Cloudlets: Bringing the cloud to the mobile user. In: Proceedings of the third ACM workshop on Mobile cloud computing and services. ACM; 2012. p. 29–36.

[98] Fernando N, Loke SW, Rahayu W. Mobile cloud computing: A survey. Future generation computer systems. 2013;29(1):84–106.

[99] Hu YC, Patel M, Sabella D, et al. Mobile edge computing—A key technology towards 5G. ETSI white paper. 2015;11(11):1–16.

[100] Cisco. Cisco IOx Data Sheet; 2016. Available from: http://www.cisco.com/c/en/us/products/collateral/cloud-systems-management/iox/datasheet-c78-736767.html.

[101] Dell. Dell Edge Gateway 5000; 2016. Available from: http://www.dell.com/us/business/p/dell-edge-gateway-5000/pd?oc=xctoi5000us.

[102] Milojicic DS, Kalogeraki V, Lukose R, et al. Peer-to-peer computing. 2002;.

[103] Jelasity M, Montresor A, Babaoglu O. Gossip-based aggregation in large dynamic networks. ACM Transactions on Computer Systems (TOCS). 2005;23(3):219–252.

[104] Akyildiz IF, Su W, Sankarasubramaniam Y, et al. Wireless sensor networks: a survey. Computer networks. 2002;38(4):393–422.

[105] Gilbert S, Lynch N. Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-tolerant Web Services. SIGACT News. 2002 Jun;33(2):51–59.

[106] Meiklejohn C, Van Roy P. Lasp: A Language for Distributed, Coordination-Free Programming. In: Proceedings of the 17th International Symposium on Principles and Practice of Declarative Programming (PPDP 2015). ACM; 2015. p. 184–195.

[107] Carvalho N, Pereira J, Oliveira R, et al. Emergent Structure in Unstructured Epidemic Multicast. In: Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07). Edinburgh, Scotland, UK; 2007. p. 481 – 490.

[108] Balegas V, Serra D, Duarte S, et al. Extending Eventually Consistent Cloud Databases for Enforcing Numeric Invariants. In: Proceedings of SRDS 2015. Montréal, Canada: IEEE Computer Society; 2015. p. 31–36. Available from: http://lip6.fr/Marc.Shapiro/papers/numeric-invariants-SRDS-2015.pdf.

[109] Najafzadeh M, Shapiro M, Balegas V, et al. Improving the scalability of geo-replication with reservations. In: ACM SIGCOMM - Distributed Cloud Computing (DCC). Dresden, Germany; 2013. Available from: http://lip6.fr/Marc.Shapiro//papers/escrow-DCC-2013.pdf.

[110] Gotsman A, Yang H, Ferreira C, et al. 'Cause I'M Strong Enough: Reasoning About Consistency Choices in Distributed Systems. In: Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. New York, NY, USA: ACM; 2016. p. 371–384. Available from: http://doi.acm.org/10.1145/2837614.2837625.

[111] Akkoorath DD, Tomsic A, Bravo M, et al. Cure: Strong Semantics Meets High Availability and Low Latency. INRIA; 2016. RR-8858.

[112] van der Linde A, Fouto P, Leitão J, et al. Legion: Enriching Internet Services with Peer-to-Peer Interactions. In: Proceedings of the 26th International Conference on World Wide Web. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee; 2017. p. 283–292. Available from: https://doi.org/10.1145/3038912.3052673.

[113] Lasp: The Missing part of Erlang distribution;. Accessed: 2018-04-27. http://www.lasp-lang.org.

[114] Meiklejohn C, Enes V, Yoo J, et al. Practical Evaluation of the Lasp Programming Model at Large Scale. In: Proceedings of the 19th International Symposium on Principles and Practice of Declarative Programming (PPDP 2017). ACM; 2017. p. 109–114.

[115] Bichot CE, Siarry P. Graph partitioning. John Wiley & Sons; 2013.

[116] Shewchuk JR. Allow Me to Introduce Spectral and Isoperimetric Graph Partitioning; 2016. Available from: http://www.cs.berkeley.edu/~jrs/papers/partnotes.pdf.

[117] Bellman R. Introduction to matrix analysis. vol. 960. SIAM;.

[118] Chung FR. Laplacians of graphs and Cheeger's inequalities. Combinatorics, Paul Erdos is Eighty. 1996;2(157-172):13–2.

[119] Spielman DA, Teng SH. Spectral partitioning works: Planar graphs and finite element meshes. Linear Algebra and its Applications. 2007;421(2):284–305.

[120] Gantmakher FR. The theory of matrices. vol. 131. American Mathematical Soc.; 1998.

[121] Berman A, Plemmons RJ. Nonnegative matrices. vol. 9. SIAM; 1979.

[122] Fiedler M. Algebraic connectivity of graphs. Czechoslovak mathematical journal. 1973;23(2):298–305.

[123] Mohar B. Isoperimetric numbers of graphs. Journal of Combinatorial Theory, Series B. 1989;47(3):274–291.

[124] Van Driessche R, Roose D. An improved spectral bisection algorithm and its application to dynamic load balancing. Parallel computing. 1995;21(1):29–48.

[125] Hendrickson B, Leland R. An improved spectral graph partitioning algorithm for mapping parallel computations. SIAM Journal on Scientific Computing. 1995;16(2):452–469.

[126] Lancaster P, Tismenetsky M. The theory of matrices: with applications. Elsevier; 1985.

[127] Chevalier C, Pellegrini F. PT-Scotch: A tool for efficient parallel graph ordering. Parallel computing. 2008;34(6):318–331.

[128] Anderson E, Bai Z, Bischof C, et al. LAPACK Users' Guide (Software, Environments and Tools) 3rd Edition;.

[129] Bergamaschi L, Bozzo E. Computing the smallest eigenpairs of the graph Laplacian. SeMA Journal. 2018;75(1):1–16.

[130] Soper AJ, Walshaw C, Cross M. A combined evolutionary search and multilevel optimisation approach to graph-partitioning. Journal of Global Optimization. 2004;29(2):225–241.

[131] Zheng A, Labrinidis A, Pisciuneri PH, et al. PARAGON: Parallel Architecture-Aware Graph Partition Refinement Algorithm. In: EDBT; 2016. p. 365–376.

[132] Fiduccia CM, Mattheyses RM. A linear-time heuristic for improving network partitions. In: Papers on Twenty-five years of electronic design automation. ACM; 1988. p. 241–247.