Project no. 288570

PARAPHRASE

Strategic Research Partnership (STREP)
PARALLEL **P**ATTERNS FOR **A**DAPTIVE **H**ETEROGENEOUS **M**ULTICORE **S**YSTEMS

Final Report: Public Summary
D1.4

Due date of deliverable: 31st May 2015

Start date of project: October 1^{st}, 2011

Type: Deliverable
WP number: WP1
Task number: N/A

Responsible institution: University of St Andrews
Editor and editor's address: Kevin Hammond, University of St Andrews

Version 1.0 (*Revision* : 4622)

Project co-funded by the European Commission within the Seventh Framework Programme		
Dissemination Level		
PU	Public	√
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

1.1 Project Context and Objectives

A revolution is happening in computer hardware. After three decades during which microprocessor speeds increased almost 4000 times, we are starting to hit long-predicted physical limits on the speed of a single processor. Recent computers instead use two, four or more processor cores working together "in parallel", giving peak performance that is equivalent to a 5GHz, 10GHz or even 30GHz single processor, but at a fraction of the projected energy usage. The *effective* exploitation of such high performance is essential to support modern demands for computing power in the home, in industry, and in the economy at large. Combining this with low energy usage is crucial if the performance is to be delivered at a reasonable financial and environmental cost. Future hardware designs will harness even greater numbers of processor cores, perhaps in the thousands or millions, and perhaps with widely varying speeds and capabilities. These will be combined with advanced graphics processor units (GPUs) and other specialist units, such as "soft" processors, to give further performance and energy gains. In this way we will be able to meet society's future needs for computing power. While there are already significant challenges in building computers, such as those described above, from heterogeneous processor and other computing units, there are even greater challenges in building parallel software that can use them effectively, especially where non-specialist programmers are involved. In order to meet existing and future software needs, we must produce tools and techniques that make it easy for normal programmers to write software for heterogeneous parallel systems, but without significantly impacting the efficiency of the system as a whole. This is the challenge that has been successfully tackled by the ParaPhrase project.

1.1.1 Concrete Objectives

The concrete technical objectives of the ParaPhrase project were:

Objective 1: To develop high-level parallel design patterns that easily expose parallelism for a wide variety of parallel applications.

Objective 2: To develop efficient parallel implementations corresponding to these patterns that can be easily re-targeted to different hardware architectures.

Objective 3: To define a virtualisation of the parallel software in the form of a low-level component model defining well-defined component state, life-cycle, and interfaces (use and provide) that allows the re-mapping to, possibly heterogeneous, hardware devices.

Objective 4: To develop refactoring tools that support the parallel design process by allowing the straightforward inclusion of alternative parallelisations for the same software design.

Objective 5: To develop adaptation mechanisms that dynamically and efficiently re-map software components to the available hardware components.

Objective 6: To validate the overall ParaPhrase approach using a representative set of industrially-derived high-performance applications.

All of these objectives have been achieved. New pattern-based parallel software development tools have been produced that enable programmers to generate efficient parallel programs at a fraction of the effort required by expert parallel programmers. The refactoring tools support alternative parallelisations using advanced new ideas of *pattern discovery* and *program shaping* to help the programmer identify and introduce parallelism. They build on both existing and new patterns of parallelism, determined from large-scale industrial and scientific use cases. Pattern implementations have been provided in both C++, via the FastFlow library, which has been extended and enhanced in the course of the project, and in Erlang, via the Skel library which has been newly developed in the project. Dynamic mapping techniques have been developed that abstract across heterogeneous combinations of CPUs and GPUs, using a component model to allow code to be automatically and dynamically mapped between CPUs and GPUs. The effectiveness of the tools has been demonstrated using large-scale industrial and scientific applications. We have shown that using the automated ParaPhrase technologies, it is possible to easily achieve an improvement in performance over the original program source that is close to or the same as that produced by an expert parallel programmer, but at a fraction of the effort and without needing such specialist skills. This potentially has a major impact on future parallel software development, empowering normal software developers, while dramatically reducing development/testing time, and also eliminating major classes of bug, such as deadlocks.

The results of the project have been disseminated widely and publicly, with over 80 scientific publications produced in the course of the project and over 100 presentations given to various groups. The majority of the project publications and deliverables are available for free public access. The work that has been done in the project is now being exploited by third parties, including major industrial concerns, who have found it to yield significant cost savings and major advantages in terms of ease of parallelisation by non-experts. It is being taken forward through a number of commercial spin-off activities, as well as through further national and international research projects that will engage new users of the ParaPhrase technologies, including major multinational companies. In this way, the results of the ParaPhrase project will provide significant long-term benefits and impact.

1.2 The ParaPhrase Approach

The tools developed in the ParaPhrase project link together to provide a coherent set of tools supporting a structured parallel development process and methodology. Figure 1.1 gives an overview of how the ParaPhrase tools can be used. Programs written in the source language (either C++ or Erlang) are passed to the relevant refactoring tool (the ParaPhrase C++ refactoring tool for C++ or the ParTE tool for Erlang). The tools are used by the programmer to identify and isolate components in the program that are amenable to parallelisation. As part of this process, it may be necessary to restructure (shape) the program. This gives a potentially parallelisable version of the original program.

In the second phase of developing a parallel program, the same tools are used to structure the components into patterns. Formally motivated rewriting rules are used to refactor the source program to introduce implementations of the patterns that have been identified, under programmer control. Information about the expected parallel performance is used to guide the choice of alternative patterns and/or implementations. This information may be obtained either from high-level parallel cost models or from the ParaPhrase Performance Enhancement Infrastructure (PEI), that was developed in the course of the ParaPhrase project. This give a parallel version of the program, completing the high-level aspects of the parallelisation process.

In the final phase of parallelisation, low-level performance and systems information is taken into account, again through the PEI, and the program is mapped to the available heterogeneous hardware resources. For C++, we exploit the FastFlow skeleton library that provides skeleton instantiations for both CPUs and for GPUs, as extended in the ParaPhrase project. For Erlang, we either use the Skel library for CPU-only code that was developed in the ParaPhrase project, or the prototype Lapedo version of the Skel library that provides dynamic hybrid versions of the Skel skeletons that are capable of being allocated to either CPUs or to GPUs as required. Complex scenarios exploit the PEI tool to provide (near-)optimal static or dynamic mappings to the available heterogeneous hardware.

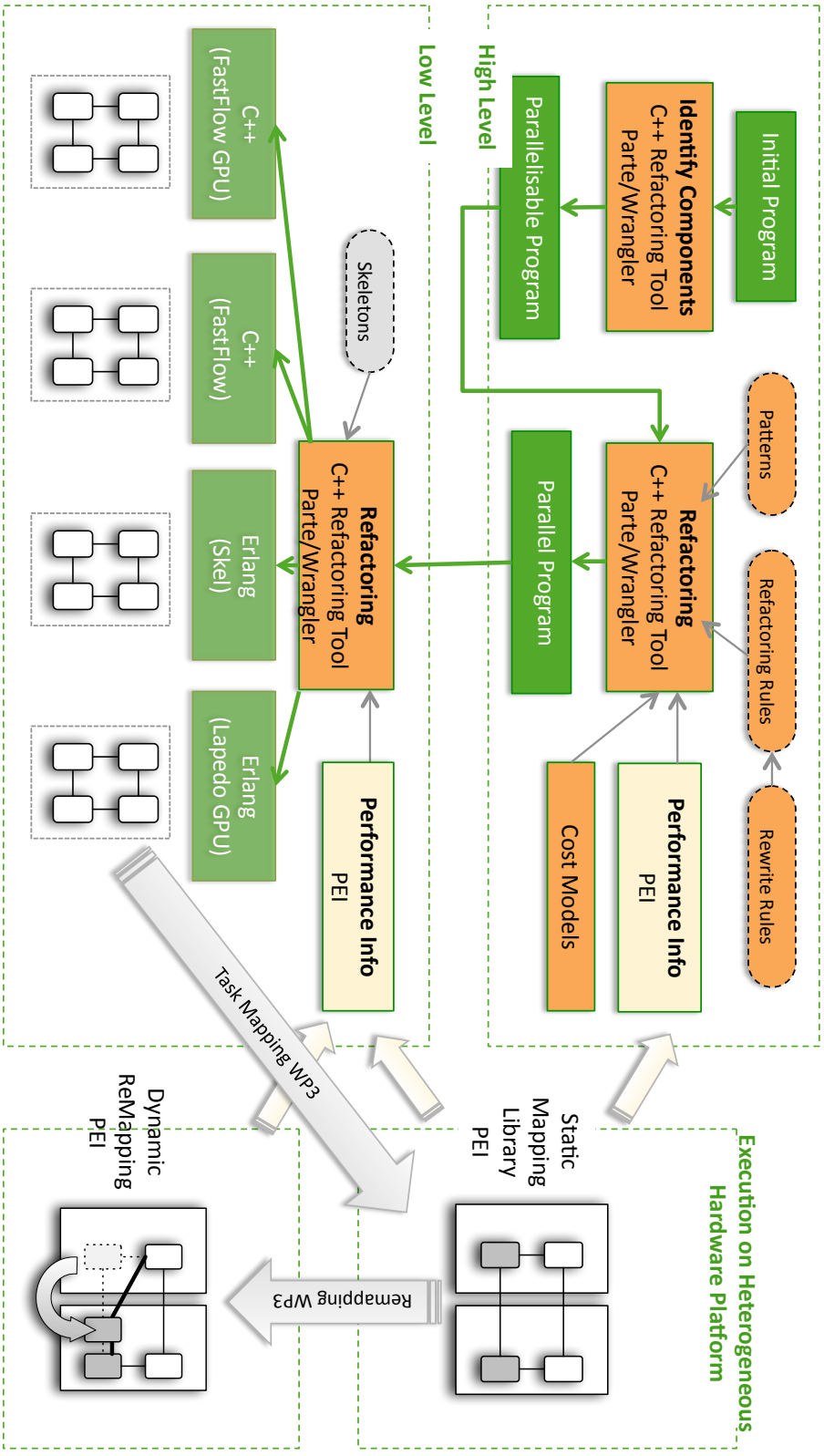

Figure 1.1: The ParaPhrase ToolChain

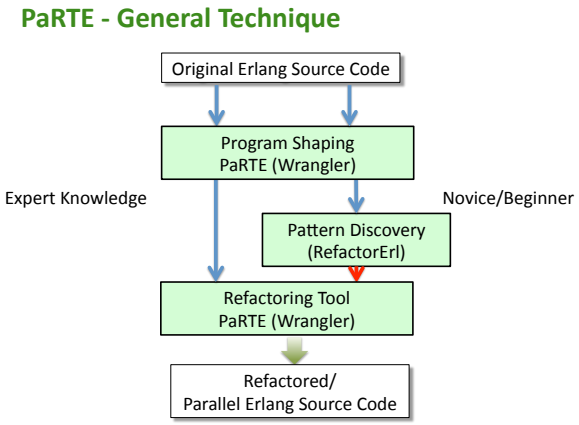

Figure 1.2: How to use the ParTE ToolChain

1.2.1 Using the ParTE Toolchain for Erlang

Figure 1.1 explains how to use the jointly-developed ParTE toolchain for Erlang. In the initial phase of parallel software development, the original Erlang source program is shaped using a new set of transformations that were developed in the course of the project. These transformations may be used to, for example, eliminate accidental dependencies, to remove side-effects, or to change data structures so that it is easier to exploit paralellism. The transformations are deployed using the widely used Wrangler refactoring tool for Erlang.

Once the program is shaped, or if shaping is not needed, new static analysis techniques, developed in the course of the ParaPhrase project and implemented in RefactorErl, are used to discover the parallel patterns and associated implementations that are most likely to yield the best parallelisations. All possible patterns are considered and cost models/the PEI are used to order these in terms of their predicted performance (best first). The programmer may select one of these, and this can then be automatically applied by the Wrangler tool to introduce one of the Skel/Lapedo skeleton structures.

Alternatively, where a programmer has expertise or knowledge, or where the pattern discovery tool fails to find suitable candidates, then the transformations can be applied directly by the programmer, by choosing an appropriate automatic refactoring. The tools check that the selected transformation is valid, so ensuring the safety of the parallelisation.

The approach using C++ and the ParaPhrase refactorings in Eclipse is similar to that described above, except that there is no automated support for pattern discovery, and that program shaping has been less well-explored in this case. Extending pattern discovery and program shaping to C++ (and other language settings) is an important issue that is the subject of ongoing research and development.

1.3 Availability of Tools and Documents

Subject to licencing/commercial etc. issues, the software tools described above have all been linked from the project website (http://www.paraphrase-ict.eu). The sources to the majority of these tools are freely available from open source software repositories (GitHub etc), linked from the same location. The same project website also contains a large number of public technical deliverables that describe the tools and how to use them, the underlying analysis and other techniques, and some of the research results that have been obtained. It also contains newsfeeds, documentation, tutorials, podcasts and a list of past project events etc. Further research results are described in the high-quality research papers that have been produced in the course of the project, as listed below. The majority of these papers have been placed in freely accessible public repositories, e.g. APERTO (http://aperto.unito.it, TRAP (http://trap.ncirl.ie) or PURE (http://risweb.st-andrews.ac.uk), and are available for open access.

[1] M. Aldinucci, S. Campa, M. Danelutto, P. Kilpatrick, and M. Torquati. Targeting distributed systems in FastFlow. In *Euro-Par 2012: Parallel Processing Workshops*, volume 7640 of *Lecture Notes in Computer Science*, pages 47–56. Springer, 2013.

[2] M. Aldinucci, S. Campa, M. Danelutto, P. Kilpatrick, and M. Torquati. Design patterns percolating to parallel programming framework implementation. *International Journal of Parallel Programming*, 42(6):1012–1031, 2014.

[3] M. Aldinucci, S. Campa, M. Danelutto, P. Kilpatrick, and M. Torquati. Pool Evolution: a Domain Specific Parallel Pattern. In *Proc. HLPP 2014: 7th Intl. Symposium on High-level Parallel Programming and Applications (HLPP)*, Amsterdam, The Netherlands, July 2014.

[4] M. Aldinucci, S. Campa, M. Danelutto, P. Kilpatrick, and M. Torquati. Pool evolution: A parallel pattern for evolutionary and symbolic computing. *International Journal of Parallel Programming*, pages 1–21, 2015.

[5] M. Aldinucci, S. Campa, P. Kilpatrick, F. Tordini, and M. Torquati. An abstract annotation model for skeletons. In *FMCO: 10th International Symposium on Formal Methods for Components and Objects–Revised Selected Papers*, volume 7542 of *Lecture Notes in Computer Science*, pages 257–276, Turin, 2013. Springer.

[6] M. Aldinucci, M. Danelutto, L. Anardu, M. Torquati, and P. Kilpatrick. Parallel patterns + macro data flow for multi-core programming. In *Proc. of Intl. Euromicro PDP 2012: Parallel Distributed and network-based Processing*, pages 27–36, Garching, Germany, Feb. 2012. IEEE.

[7] M. Aldinucci, M. Danelutto, P. Kilpatrick, M. Meneghin, and M. Torquati. An efficient unbounded lock-free queue for multi-core systems. In *Proc. Euro-Par 2012: 18th Intl. Conf. on Parallel Processing*, volume 7484 of *LNCS*, pages 662–673, Rhodes Island, Greece, Aug. 2012. Springer.

[8] M. Aldinucci, M. Danelutto, P. Kilpatrick, C. Montangero, and L. Semini. Managing adaptivity in parallel systems. In *FMCO: 10th International Symposium on Formal Methods for Components and Objects–Revised Selected Papers*, volume 7542 of *Lecture Notes in Computer Science*, pages 199–217, Turin, 2013. Springer.

[9] M. Aldinucci, M. Danelutto, P. Kilpatrick, and M. Torquati. Targeting Heterogeneous Architectures via Macro Data Flow. *Parallel Processing Letters*, 22(2), June 2012.

[10] M. Aldinucci, M. Danelutto, P. Kilpatrick, and M. Torquati. Targeting heterogeneous architectures via macro data flow. In *Proc. HLPGPU 2012: Intl. Workshop on High-level Programming for Heterogeneous and Hierarchical Parallel Systems*, pages 1–6. HiPEAC, Jan. 2012.

[11] M. Aldinucci, M. Danelutto, P. Kilpatrick, and M. Torquati. FastFlow: High-Level and Efficient Streaming on Multi-Core. In S. Pllana and F. Xhafa, editors, *Programming Multi-core and Many-core Computing Systems*, Parallel and Distributed Computing, chapter 13. Wiley, 2013.

[12] M. Aldinucci, G. Peretti Pezzi, M. Drocco, C. Spampinato, and M. Torquati. Parallel visual data restoration on multi-GPGPUs using stencil-reduce pattern. *International Journal of High Performance Computing Application*, 2015.

[13] M. Aldinucci, G. Peretti Pezzi, M. Drocco, F. Tordini, P. Kilpatrick, and M. Torquati. Parallel video denoising on heterogeneous platforms. In *Proc. of Intl. Workshop on High-level Programming for Heterogeneous and Hierarchical Parallel Systems (HLPGPU)*, 2014.

[14] M. Aldinucci, S. Ruggieri, and M. Torquati. Decision tree building on multi-core using fastflow. *Concurrency and Computation: Practice and Experience*, 26(3):800–820, 2014.

[15] M. Aldinucci, C. Spampinato, M. Drocco, M. Torquati, and S. Palazzo. A parallel edge preserving algorithm for salt and pepper image denoising. In *Image Processing Theory, Tools and Applications (IPTA), 2012 3rd International Conference on*, pages 97–104. IEEE, 2012.

[16] M. Aldinucci, F. Tordini, M. Drocco, M. Torquati, and M. Coppo. Parallel stochastic simulators in system biology: the evolution of the species. In *Proc. of Intl. Euromicro PDP 2013: Parallel Distributed and network-based Processing*, Belfast, Nothern Ireland, U.K., Feb. 2013. IEEE.

[17] M. Aldinucci, M. Torquati, C. Spampinato, M. Drocco, C. Misale, C. Calcagno, and M. Coppo. Parallel stochastic systems biology in the cloud. *Briefings in Bioinformatics*, 15(5):798–813, 2014.

[18] T. Baumann and J. Gracia. Cudagrind: Memory-usage checking for cuda. In A. Knüpfer, J. Gracia, W. E. Nagel, and M. M. Resch, editors, *Tools for High Performance Computing 2013*, pages 67–78. Springer International Publishing, 2014.

[19] T. M. Baumann and J. Gracia. Cudagrind: A valgrind extension for CUDA. In M. Bader, A. Bode, H. Bungartz, M. Gerndt, G. R. Joubert, and F. J. Peters, editors, *Parallel Computing: Accelerating Computational Science and Engineering (CSE), Proceedings of the International Conference on Parallel Computing, ParCo 2013, 10-13 September 2013, Garching (near Munich), Germany*, volume 25 of *Advances in Parallel Computing*, pages 763–772. IOS Press, 2013.

[20] R. Behrends, K. Hammond, A. Konovalov, H.-W. Loidl, P. Maier, S. Linton, and P. Trinder. HPC-GAP: Engineering a 21st-Century High-Performance Computer Algebra System. Submitted to Concurrency and Computation: Practice and Experience (CPE), 2015.

[21] J. Berthold, H.-W. Loidl, and K. Hammond. PAEAN: Portable Runtime Support for Physically-Shared-Nothing Architectures. *Journal of Functional Programming*, 2015. (under revision).

[22] S. Boob, H. González-Vélez, and A. M. Popescu. Automated instantiation of heterogeneous fast flow CPU/GPU parallel pattern applications in clouds. In *PDP 2014: 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pages 162–169, Torino, Italy, Feb. 2014. IEEE.

[23] I. Bozó, V. Fördős, D. Horpácsi, Z. Horváth, T. Kozsik, J. Kőszegi, and M. Tóth. Refactorings to enable parallelization. In J. Hage and J. McCarthy, editors, *Trends in Functional Programming*, volume 8843 of *Lecture Notes in Computer Science*, pages 104–121. Springer International Publishing, 2015.

[24] I. Bozó, V. Fördős, Z. Horváth, M. Tóth, D. Horpácsi, T. Kozsik, J. Kőszegi, A. Barwell, C. Brown, and K. Hammond. Discovering parallel pattern candidates in Erlang. In *Proceedings of the Thirteenth ACM SIGPLAN Workshop on Erlang*, pages 13–23, New York, NY, USA, 2014. ACM.

[25] I. Bozó, Z. Horváth, T. Kozsik, and M. Tóth. Static analysis to discover Divide-and-Conquer algorithms. In Preparation, 2015.

[26] S. Brinkmann and J. Gracia. Cppss – a C++ library for efficient task parallelism. In *Proc. Third International Conf. on Advanced Communications and Computation (INFOCOMP 2013), Lisbon, Portugal*, Lisbon, Nov. 2013.

[27] C. Brown, M. Danelutto, K. Hammond, P. Kilpatrick, and A. Elliott. Cost-directed refactoring for parallel erlang programs. *International Journal of Parallel Programming*, 42(4):564–582, 2014.

[28] C. Brown, K. Hammond, M. Danelutto, and P. Kilpatrick. A language-independent parallel refactoring framework. In *Proc. WRT 2012: Fifth Workshop on Refactoring Tools*, pages 54–58, New York, NY, USA, 2012. ACM.

[29] C. Brown, K. Hammond, M. Danelutto, P. Kilpatrick, H. Schöner, and T. Breddin. Paraphrasing: Generating parallel programs using refactoring. In *FMCO: 10th International Symposium on Formal Methods for Components and Objects–Revised Selected Papers*, volume 7542 of *Lecture Notes in Computer Science*, pages 237–256, Turin, 2013. Springer.

[30] C. Brown, V. Janjic, K. Hammond, K. Idrees, C. Glass, M. A. Wafai, M. Goli, and J. McCall. Bridging the Divide: A New Methodology for Semi-Automatic Programming of Heterogeneous Parallel Machines. *Computer Science: Research and Development (CSRD)*, 2015. (to appear).

[31] C. Brown, V. Janjic, K. Hammond, H. Schoner, K. Idrees, and C. Glass. Agricultural reform: More efficient farming using advanced parallel refactoring tools. In *Proc. PDP 2014: 22nd Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pages 36–43, Feb 2014.

[32] D. Buono, M. Danelutto, S. Lametti, and M. Torquati. Parallel patterns for general purpose many-core. In *Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on*, pages 131–139. IEEE, 2013.

[33] D. Buono, M. Danelutto, T. D. Matteis, G. Mencagli, and M. Torquati. A lightweight run-time support for fast dense linear algebra on multi-core. In *Proceedings Software Engineering: Parallel and Distributed Computing and Networks: Artificial Intelligence and Applications - 2014 PDCN '14*, 2014.

[34] A. Byrski and M. Kisiel-Dorohinicki. Memetic computing in selected agent-based evolutionary systems. In *Intl. Proc. of European Conference on Modelling and Simulation 2014*, 2014.

[35] S. Campa, M. Danelutto, M. Goli, H. González-Vélez, A.-M. Popescu, and M. Torquati. Parallel patterns for heterogeneous CPU/GPU architectures: Structured parallelism from cluster to cloud. *Future Generation Computer Systems*, 37:354–366, 2014.

[36] S. Campa, M. Danelutto, H. González-Vélez, A. M. Popescu, and M. Torquati. Towards the deployment of FastFlow on distributed virtual architectures. In *Proc. of ECMS 2013*, pages 518–524, Alesund, 2013.

[37] D. Castro and K. Hammond. Skeletor: A DSL for Describing Type-based Specifications of Parallel Skeletons. In *Proc. Workshop on High-Level Programming for Heterogeneous and Hierarchical Parallel Systems (HLPGPU 2014)*, 2014.

[38] D. Castro, K. Hammond, E. C. Brady, and S. Sarkar. Structure, Semantics and Speedup: Reasoning about Structured Parallel Programs using Dependent Types. *Journal of Functional Programming*, 2015. (submitted).

[39] M. Danelutto and R. D. Cosmo. A Minimal Disruption Skeleton Experiment: Seamless Map & Reduce Embedding in OCaml. *Procedia Computer Science*, 9(0):1837 – 1846, 2012. Proc. ICCS 2012: International Conference on Computational Science.

[40] M. Danelutto and M. Torquati. A RISC building block set for structured parallel programming. In *Proc. PDP 2013: 21st Euromicro International Conference on Parallel, Distributed and Network-Based Processing*, pages 46–50. IEEE, 2013.

[41] M. Danelutto and M. Torquati. Loop parallelism: A new skeleton perspective on data parallel patterns. In *Proceedings of the 2014 22Nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, PDP '14, pages 52–59, Washington, DC, USA, 2014. IEEE Computer Society.

[42] M. Danelutto and M. Torquati. Structured parallel programming with core FastFlow. In *Central European Functional Programming School, 5th Summer School, CEFP 2013, Cluj-Napioca, Revised Selected Papers*, number 8606 in LNCS. Springer Verlag, 2014.

[43] M. Danelutto, M. Torquati, and P. Kilpatrick. A green perspective on structured parallel programming. In *Euromicro PDP, Special session on Energy aware computing, Proceedings of Euromicro Conference on Parallel, distributed and network based processing*. IEEE Press, 2015.

[44] R. Debski. High–performance simulation–based algorithms for an alpine ski racer's trajectory optimization in heterogeneous computer systems. *International Journal on Applied Mathematics and Computer Science*, 24(3):551–566, 2014.

[45] M. Drocco, M. Aldinucci, and M. Torquati. A dynamic memory allocator for heterogeneous platforms. In *Advanced Computer Architecture and Compilation for High-Performance and Embedded Systems (ACACES) – Poster Abstracts*, Fiuggi, Italy, 2014. HiPEAC.

[46] M. Drocco, C. Misale, G. Peretti Pezzi, F. Tordini, and M. Aldinucci. Memory-optimised parallel processing of Hi-C data. In *Proc. of Intl. Euromi-*

cro PDP 2015: Parallel Distributed and network-based Processing, pages 1–8. IEEE, Mar. 2015.

[47] H. Ferreiro, V. Janjic, L. Castro, and K. Hammond. Kindergarten Cop: Dynamic Nursery Resizing for GHC. In *Trends in Functional Programming (TFP 2015)*, 2015.

[48] H. Ferreiro, V. Janjic, L. M. Castro, and K. Hammond. Repeating history: Execution replay for parallel Haskell programs. In *Trends in Functional Programming*, volume 7829 of *Lecture Notes in Computer Science*, pages 231–246, St. Andrews, June 2013. Springer.

[49] K. Fürlinger, C. Glass, J. Gracia, A. Knüpfer, J. Tao, D. Hünich, K. Idrees, M. Maiterth, Y. Mhedheb, and H. Zhou. Dash: Data structures and algorithms with support for hierarchical locality. In *Euro-Par 2014: Parallel Processing Workshops*, pages 542–552. Springer, 2014.

[50] M. T. Garba and H. González-Vélez. Asymptotic peak utilisation in heterogeneous parallel CPU/GPU pipelines: a decentralised queue monitoring strategy. *Parallel Processing Letters*, 22(2):1240008[13 pages], 2012.

[51] M. T. Garba, H. González-Vélez, and D. L. Roach. GPU acceleration for hermitian eigensystems. *T. Computational Collective Intelligence*, 10:150–161, 2013.

[52] M. Goli, M. T. Garba, and H. González-Vélez. Streaming dynamic coarse-grained CPU/GPU workloads with heterogeneous pipelines in FastFlow. In *HPCC: 14th IEEE International Conference on High Performance Computing and Communication*, pages 445–452, Liverpool, 2012. IEEE.

[53] M. Goli and H. González-Vélez. Heterogeneous algorithmic skeletons for fast flow with seamless coordination over hybrid architectures. In *PDP: 21st Euromicro International Conference on Parallel, Distributed and Network-Based Processing*, pages 148–156, Belfast, 2013. IEEE.

[54] M. Goli and H. González-Vélez. Heterogeneous algorithmic skeletons for fast flow with seamless coordination over hybrid architectures. In *PDP: 21st Euromicro International Conference on Parallel, Distributed and Network-Based Processing*, pages 148–156, Belfast, 2013.

[55] M. Goli and H. González-Vélez. N-body computations using skeletal frameworks on multicore CPU/graphics processing unit architectures: an empirical performance evaluation. *Concurrency and Computation: Practice and Experience*, 26(4):972–986, 2014.

[56] M. Goli, J. McCall, C. Brown, V. Janjic, and K. Hammond. Mapping parallel programs to heterogeneous cpu/gpu architectures using a monte carlo tree

search. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 2932–2939. IEEE, 2013.

[57] J. Gracia, C. Niethammer, M. Hasert, S. Brinkmann, R. Keller, and C. Glass. Hybrid MPI/StarSs – a case study. In *Proc. ISPA 2012: 10th IEEE International Symposium on Parallel and Distributed Processing with Applications, July 10-13, Madrid, Spain*, pages 48–55, 2012.

[58] D. Grzonka, J. Kolodziej, J. Tao, and S. U. Khan. Artificial neural network support to monitoring of the evolutionary driven security aware scheduling in computational distributed environments. *Future Generation Computer Systems*, 2015. (in press).

[59] K. Hammond, M. Aldinucci, C. Brown, F. Cesarini, M. Danelutto, H. González-Vélez, P. Kilpatrick, R. Keller, M. Rossbory, and G. Shainer. The ParaPhrase project: Parallel patterns for adaptive heterogeneous multicore systems. In *FMCO: 10th International Symposium on Formal Methods for Components and Objects–Revised Selected Papers*, volume 7542 of *Lecture Notes in Computer Science*, pages 218–236, Turin, 2013. Springer.

[60] K. Idrees, M. Nachtmann, and C. W. Glass. Evaluation of fastflow technology for real-world application. In *Sustained Simulation Performance 2013*, pages 77–88. Springer, 2014.

[61] K. Idrees, C. Niethammer, A. Esposito, and C. W. Glass. Performance evaluation of unified parallel C for molecular dynamics. In *7th International Conference on PGAS Programming Models*, page 237, Edinburgh, 2013. EPCC.

[62] V. Janjic, A. Barwell, and K. Hammond. Using Erlang Skeletons to Parallelise Realistic Medium-Scale Parallel Programs. In *Proc. Workshop on High-Level Programming for Heterogeneous and Hierarchical Parallel Systems (HLPGPU 2014)*, 2014.

[63] V. Janjic, C. Brown, A. Barwell, and K. Hammond. Lapedo: Hybrid Skeletons for Programming Heterogeneous Multicore Machines in Erlang. In Preparation, 2015.

[64] V. Janjic, C. Brown, and K. Hammond. Heuristics for Mapping Patterned Applications to Heterogeneous CPU/GPU Systems. In Preparation, 2015.

[65] V. Janjic, C. Brown, M. Neunhoeffer, K. Hammond, S. Linton, and H.-W. Loidl. Space exploration using parallel orbits: a study in parallel symbolic computing. In *International Conference on Parallel Computing (ParCo 2013)*, 2013.

[66] V. Janjic and K. Hammond. Using Load Information in Work-Stealing on Distributed Systems with Non-Uniform Communication Latencies. In *Proc.*

EuroPar 2012: 2012 International European Conference on Parallel and Distributed Computing, 2012.

[67] M. Kazirod, W. Korczynski, and A. Byrski. Agent-oriented computing platform in python. In *Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2014 IEEE/WIC/ACM International Joint Conferences on*, volume 3, pages 365–372, Aug 2014.

[68] M. Kowol, A. Byrski, and M. Kisiel-Dorohinicki. Agent-based evolutionary computing for difficult discrete problems. *Procedia Computer Science*, 29(0):1039 – 1047, 2014. 2014 International Conference on Computational Science.

[69] D. Krzywicki, Ł. Faber, A. Byrski, and M. Kisiel-Dorohinicki. Computing agents for decision support systems. *Future Generation Computer Systems*, 37(0):390 – 400, 2014.

[70] D. Krzywicki, J. Stypka, P. Anielski, Ł. Faber, W. Turek, A. Byrski, and M. Kisiel-Dorohinicki. Generation-free agent-based evolutionary computing. *Procedia Computer Science*, 29(0):1068 – 1077, 2014. 2014 International Conference on Computational Science.

[71] S. Linton, K. Hammond, A. Konovalov, C. Brown, P. W. Trinder, H.-W. Loidl, P. Horn, and D. Roozemond. Easy composition of symbolic computation software using scscp: A new lingua franca for symbolic computation. *Journal of Symbolic Computation*, 49:95–119, 2013.

[72] V. Marjanovic, J. Gracia, and C. Glass. Performance Modeling of the HPCG Benchmark. In S. A. Jarvis, S. A. Wright, and S. D. Hammond, editors, *High Performance Computing Systems. Performance Modeling, Benchmarking and Simulation - 5th International Workshop, PMBS 2014, New Orleans, LA, USA, November 16, 2014. Revised Selected Papers*, volume 8896 of *Lecture Notes in Computer Science*, pages 172–192. Springer, 2015. Held as part of SC14.

[73] C. Misale. Accelerating bowtie2 with a lock-less concurrency approach and memory affinity. In M. Aldinucci, D. D'Agostino, and P. Kilpatrick, editors, *Proc. of Intl. Euromicro PDP 2014: Parallel Distributed and network-based Processing*, Torino, Italy, 2014. IEEE. (Best paper award).

[74] C. Misale, M. Aldinucci, and M. Torquati. Memory affinity in multithreading: the bowtie2 case study. In *Advanced Computer Architecture and Compilation for High-Performance and Embedded Systems (ACACES)*, Fiuggi, 2013. Poster.

[75] C. Misale, G. Ferrero, M. Torquati, and M. Aldinucci. Sequence alignment tools: one parallel pattern to rule them all? *BioMed Research International*, June 2014. doi:10.1155/2014/207041.

[76] M. Moniruzzaman, K. Idrees, R. M., and J. Gracia. An adaptive load-balancing task-scheduler for FastFlow. In *Proceedings of the Fifth International Conference on Advanced Communications and Computation (INFOCOMP)*. IARIA XPS Press, 2015. accepted.

[77] C. Niethammer, C. Glass, and J. Gracia. Avoiding Serialization Effects in Data-dependency Aware Task Parallel Algorithms for Spatial Decomposition. In *Proc. ISPA 2012: 10th IEEE International Symposium on Parallel and Distributed Processing with Applications, July 10-13 2012, Madrid, Spain*, pages 743–748, 2012.

[78] G. Oláh, D. Horpácsi, T. Kozsik, and M. Tóth. Type inference in Core Erlang to support test data generation. *STUD UNIV BABES-BOLYAI SER INFO*, LIX(1):201–215, 2014.

[79] A. Secco, I. Uddin, G. Peretti Pezzi, and M. Torquati. Message passing on infiniband RDMA for parallel run-time supports. In M. Aldinucci, D. D'Agostino, and P. Kilpatrick, editors, *Proc. of Intl. Euromicro PDP 2014: Parallel Distributed and network-based Processing*, Torino, Italy, 2014. IEEE.

[80] D. D. Sensi, M. Danelutto, and M. Torquati. Energy driven adaptivity in stream parallel computations. In *Proceedings of Euromicro Conference on Parallel, distributed and network based processing*. IEEE Press, 2015.

[81] T. Serban, M. Danelutto, and P. Kilpatrick. Autonomic scheduling of tasks from data parallel patterns to CPU/GPU core mixes. In *High Performance Computing and Simulation (HPCS), 2013 International Conference on*, pages 72–79. IEEE, 2013.

[82] H. Simões, P. Vasconcelos, M. Florido, S. Jost, and K. Hammond. Automatic amortised analysis of dynamic memory allocation for lazy functional programs. In *Proceedings of the 17th ACM SIGPLAN international conference on Functional programming*, pages 165–176. ACM, 2012.

[83] A. Singh and H. González-Vélez. Hierarchical multi-log cloud-based search engine. In *Eighth International Conference on Complex, Intelligent and Software Intensive Systems, CISIS 2014*, pages 211–219, Birmingham, UK, July 2014. IEEE.

[84] L. Siwik and R. Dreżewski. Evolutionary multi-modal optimization with the use of multi-objective techniques. In L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. Zadeh, and J. Zurada, editors, *Artificial Intelligence and Soft Computing*, volume 8467 of *Lecture Notes in Computer Science*, pages 428–439. Springer International Publishing, 2014.

[85] C. Spampinato, I. Kavasidis, M. Aldinucci, C. Pino, D. Giordano, and A. Faro. Discovering biological knowledge by integrating high throughput data and scientific literature on the cloud. *Concurrency and Computation: Practice and Experience*, 26(10):1771–1786, 2014.

[86] J. Stypka, P. Anielski, S. Mentel, D. Krzywicki, W. Turek, A. Byrski, and M. Kisiel-Dorohinicki. Parallel patterns for agent-based evolutionary computing. *Computer Science*, 2015. (in press).

[87] V. Subotic, S. Brinkmann, V. Marjanovic, R. Badia, J. Gracia, C. Niethammer, E. Ayguade, J. Labarta, and M. Valero. Programmability and portability for Exascale: Top Down Programming Methodology and Tools with StarSs. *Journal of Computational Science*, 2012.

[88] J. Swann and K. Hammond. Towards 'Metaheuristics in the Large'. In *Proc. MIC 2015: 11th Metaheuristics International Conference, Agadir, Morocco*, June 2015. (to appear).

[89] F. Tordini, M. Aldinucci, and M. Torquati. High-level lock-less programming for multicore. In *Advanced Computer Architecture and Compilation for High-Performance and Embedded Systems (ACACES) – Poster Abstracts*, Fiuggi, Italy, 2012. HiPEAC.

[90] F. Tordini, M. Drocco, I. Merelli, L. Milanesi, P. Liò, and M. Aldinucci. Nuchart-ii: a graph-based approach for the analysis and interpretation of hi-c data. In *Proc. of the 11th Intl. meeting on Computational Intelligence methods for Bioinformatics and Biostatistics (CIBB 2014)*, LNBI, Cambridge, UK, 2015. Springer. To appear.

[91] F. Tordini, M. Drocco, C. Misale, L. Milanesi, P. Liò, I. Merelli, and M. Aldinucci. Parallel exploration of the nuclear chromosome conformation with NuChart-II. In *Proc. of Intl. Euromicro PDP 2015: Parallel Distributed and network-based Processing*. IEEE, Mar. 2015.

[92] P. Trinder, K. Hammond, M. Cole, G. Michaelson, and H.-W. Loidl. Resource Analyses for Parallel and Distributed Coordination. *Concurrency: Practice and Experience*, 25(3):309–348, 2013.

[93] V. Ubarhande, A.-M. Popescu, and H. González-Vélez. Novel data-distribution technique for Hadoop in heterogeneous cloud environments. In *Nineth International Conference on Complex, Intelligent and Software Intensive Systems, CISIS 2015*, Blumenau, Brazil, July 2015. IEEE. Accepted for publication.

[94] P. B. Vasconcelos, S. Jost, M. Florido, and K. Hammond. Type-based allocation analysis for co-recursion in lazy functional languages. In *Programming Languages and Systems - 24th European Symposium on Programming, ESOP*

2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings, pages 787–811, 2015.

[95] S. Wesner, J. Gracia, and C. Glass. How to Keep Pace with Fast-Changing Hardware. In *Proc. SIMULATION-2012, May 16-18, Kiev, Ukraine*, 2012.