Analysis of Parallel Applications on a High Performance–Low Energy Computer

Florina M. Ciorba[1], Thomas Ilsche[1], Elke Franz[2], Stefan Pfennig[2], Christian Scheunert[3], Ulf Markwardt[1], Joseph Schuchart[1], Daniel Hackenberg[1], Robert Schöne[1], Andreas Knüpfer[1], Wolfgang E. Nagel[1], Eduard A. Jorswieck[3], and Matthias S. Müller[4]

[1] Center for Information Sciences and High Performance Computing, Technische Universität Dresden, Germany
[2] Faculty of Computer Science, Chair of Privacy and Data Security, Technische Universität Dresden, Germany
[3] Faculty of Electrical Engineering, Chair of Communications Theory, Technische Universität Dresden, Germany
{firstname.lastname}@tu-dresden.de
[4] Rheinisch-Westfälische Technische Hochschule Aachen, Germany
Chair for Computer Science 12 - High Performance Computing
mueller@itc.rwth-aachen.de

Abstract. In this paper, we propose a holistic approach for the analysis of parallel applications on a high performance–low energy computer (called the HAEC platform). The HAEC platform is currently under design and refers to an architecture in which multiple 3-D stacked massively parallel processor chips are optically interconnected on a single board and multiple parallel boards are interconnected using short-range high-speed wireless links. Although not exclusively targeting high performance computing (HPC), the HAEC platform aims to deliver high performance at low energy costs, which are essential features for future HPC platforms. At the core of the proposed approach is a trace-driven simulator called `haec_sim` which we developed to simulate the behavior of parallel applications running on this hardware. We investigate several mapping layouts to assign the parallel applications to the HAEC platform. We concentrate on analyzing the communication performance of the HAEC platform running parallel applications. The simulator can employ two communication models: dimension order routing (DOR) and practical network coding (PNC). As a first example of the usefulness of the proposed holistic analysis approach, we present simulation results using these communication models on a communication-intensive parallel benchmark. These results highlight the potential of the mapping strategies and communication models for analyzing the performance of various types of parallel applications on the HAEC platform. This work constitutes the first step towards more complex simulations and analyses of performance and energy scenarios than those presented herein.

Keywords: performance, HAEC, simulation, network coding, routing.

1 Introduction

Energy efficiency is one of the greatest challenges in information and communication technology. A large part of the energy costs can be attributed to the transfer of information. Progress in energy efficient interconnections is necessary to allow high performance computing (HPC) and data centers to manage their energy costs while performing powerful applications. Future computing systems will largely consist of chips with energy efficient interconnects, such as IBM's Holey Optochip [7] or HP's Corona architecture [20].

The highly adaptive energy efficient computing (HAEC) platform [9] is a future computing system design aimed at dynamically adjusting the energy usage according to the workload without compromising on performance. It uses optical on-board [16] and wireless board-to-board [10] connections to mitigate the bandwidth and latency bottlenecks inherent in existing multiprocessor systems. Optical and wireless interconnects provide a wider opportunity for selecting different operation modes such that the energy consumption of individual links can be adjusted according to their load. We use an integrated approach of a highly scalable end-to-end simulation framework combining sufficient details of the application, processor, and network.

Our goal is to analyze the performance of applications executed on a high performance–low energy computer. We are concerned with questions regarding: (i) Modeling of the behavior of the various independent software and hardware components of such a system, (ii) Their integration into a holistic system model, and (iii) The prediction of the performance and energy costs of running applications on the HAEC platform. A more specific challenge on which we concentrate in this work is to predict the performance of the HAEC platform running (communication intensive) parallel applications.

Our approach is holistic and comprises multiple models. The application model is based on event traces obtained from running the parallel applications on existing platforms. This model is mapped onto the HAEC platform model using several mapping strategies. Our simulations employ two communication models to predict the behavior of parallel applications on the HAEC platform. The resulting simulated application traces form the basis for our analysis using state-of-the-art performance measurement and visualization tools.

The main contribution of this work is a holistic approach for analyzing the performance of parallel applications on the HAEC platform. We developed a trace-driven simulation framework (`haec_sim`) that employs three strategies for mapping applications to the target platform. Another major contribution is a novel communication model for the HAEC platform developed using network coding (NC) technology. This model has been implemented in the simulator in addition to standard routing. Even though these communication models do not account for transmission errors, they can easily be extended to address errors. Then, NC will outperform standard routing [1]. Given that the design of the HAEC platform is ongoing, the simulator will account for new aspects of the hardware that may otherwise be hard to capture by existing simulators.

2 Related Work

Topology aware mapping of parallel applications with regular and irregular communication patterns onto supercomputers has been studied in [4]. Mapping is also a very important area of research in network on chip (NoC) systems, where a major challenge in overall system design is to associate the intellectual property (IP) cores implementing tasks of an application with the NoC routers [17].

Many communication models in HPC belong to the LogP model family or the BSP model family [11]. Even though the parameters of these models capture significant characteristics of the underlying hardware, they do not explicitly account for the network topology. Routing [6] and network coding [1] are at a lower abstraction level than the LogP and BSP models and account for the network topology. For multiple concurrent flows, network coding can achieve higher throughput, lower latency, and better energy efficiency than standard routing.

BigSim [19] is a parallel trace-based simulator for predicting the performance of MPI applications on future large scale systems larger than those available today. COTSon [2] is a parallel simulation infrastructure for modeling clusters of multicore CPU nodes, networking, and I/O. It combines functional simulation for the behavior of devices and software, and timing simulators for the timing of all components. Apart from the compute performance, it also enables to simulate the power consumption. Dimemas [14] is a sequential trace-based simulator for predicting the performance of parallel MPI or multithreaded applications. The simulation model uses parameters such as relative processor speeds, network bandwidth and latency within and across nodes, the number of input and output links, and the processor scheduling policy. The network model assumes two-level buses. Existing trace-driven simulation approaches combine only a subset of all the aspects considered in this work, such as performance *or* energy efficiency, and application *or* system modeling.

3 Aspects of Application Analysis on Future Computing Systems

3.1 Simulation and Analysis Workflow

We employ trace-driven simulation to simulate future computing systems, such as the HAEC platform (cf. §3.3), using traces generated with the scalable performance measurement infrastructure for parallel codes Score-P (cf. §3.2). We developed a parallel trace-based simulation framework (`haec_sim`) for predicting the behavior of applications running on a future computing system (described via hardware and system software abstraction models). The simulation concentrates on maximizing performance, minimizing energy consumption, and optimizing communication. The simulated HAEC platform (cf. §3.3) employs a heterogeneous and adaptive communication model (cf. §4.2) to combine high application performance with high energy efficiency.

The proposed simulation and analysis workflow is illustrated in Fig. 1. The source code of a parallel application of interest represents the first step in the

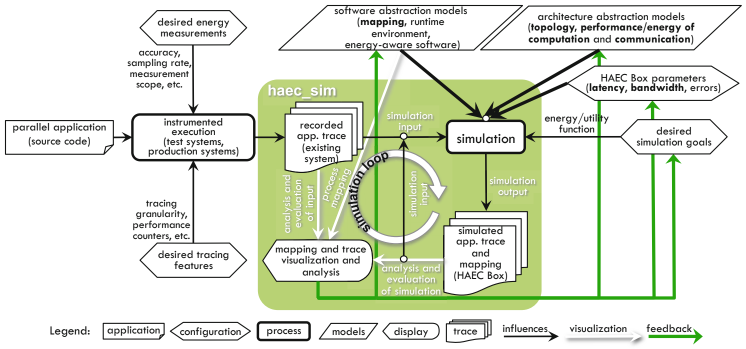

desired energy measurements
accuracy, sampling rate, measurement scope, etc.
software abstraction models (mapping, runtime environment, energy-aware software)
architecture abstraction models (topology, performance/energy of computation and communication)
HAEC Box parameters (latency, bandwidth, errors)
haec_sim
parallel application (source code)
instrumented execution (test systems, production systems)
recorded app. trace (existing system)
simulation input
simulation
energy/utility function
desired simulation goals
simulation output
simulation loop
process mapping
analysis and evaluation of input
tracing granularity, performance counters, etc.
desired tracing features
mapping and trace visualization and analysis
analysis and evaluation of simulation
simulated app. trace and mapping (HAEC Box)
Legend: application configuration process models display trace influences visualization feedback

Fig. 1. Proposed trace-driven simulation and analysis workflow

proposed workflow. This is followed by specification of performance and energy features desired to be collected using Score-P [13]. The instrumented application is executed either on an energy measurement test system or on high performance computing production systems. The resulting execution trace forms the input to the simulation. This trace can be visualized and analyzed using Vampir [12]. In addition to the input trace, the simulator (`haec_sim`) contains and employs various software and architecture abstraction models. The software abstraction models include the mapping of the processes in the input trace to the HAEC platform topology (cf. §4.1), the operating system, and energy-aware software. The architecture abstraction models include the topology of the HAEC platform (cf. §3.3), a model for predicting the energy consumption of running the desired application on the HAEC platform, and a communication model that describes how will communication be carried out over the wireless and optical links of the HAEC platform (cf. §4.2). The HAEC platform parameters refer to latency, bandwidth, and error rates. The desired simulation goals also form an input to the simulator and may include the optimization criteria (or metrics) such as performance (time) or cost (energy). The output of the simulation is an event trace describing the predicted behavior of the initial application if it were executed on the HAEC platform. Similar to the input trace, the output trace can also be visualized and analyzed with Vampir. Simulation-based analysis results in valuable feedback that can be provided to the abstraction models, to tune the target system parameters, and to adjust the desired simulation parameters to gain more insight towards the goals of the analysis.

3.2 Modeling Applications

Modeling and simulation of the performance and energy consumption of parallel applications require a detailed description of their characteristics and their behavior on various computing platforms. This can be provided in several ways, such as via: (i) Expert application knowledge, (ii) Conceptual application models, (iii) Distribution parametrized (or stochastic) models (e.g., profiles), and (iv) Recording of application event traces on existing computing systems.

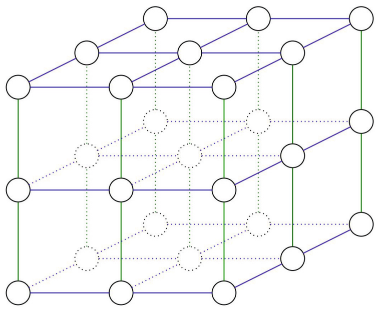
The first two approaches are not easily amenable to a broad range of parallel applications and require a significant modeling effort. The third one may not provide the fine-grained level of detail that is necessary to capture correlations and interference effects in the application. The last approach is more generic and can be employed to derive application descriptions even for highly complex applications [18].

Discrete event traces capture the runtime behavior of parallel applications on existing systems and form the application model for simulating their performance on target or future computing platforms. Traces preserve the dynamic application behavior and can yield meaningful results even for small changes in the model [18]. An application trace consists of a time-ordered sequence of discrete events including functions execution, communication operations, and management of parallelism [12]. In addition, runtime hardware performance characteristics, including energy measurements, can be recorded. We use Score-P [13] to record the execution of parallel applications in the OTF2 [8] file format.

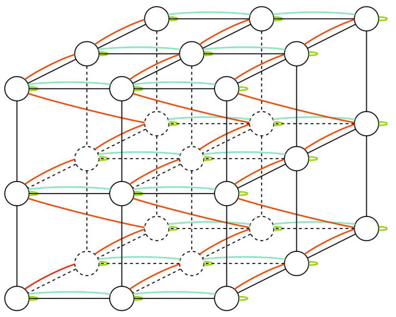
3.3 Modeling a High Performance–Low Energy Computer

The HAEC platform refers to a new high performance–low energy parallel computer architecture [9]. In this architecture, the compute nodes consist of 3-D stacked processor chips with thousands of ‘thin’ cores [15] offering massive intra-node parallelism. This parallelism is not modeled explicitly in `haec_sim` and is abstracted. Thus, a collection of many lightweight application threads are represented as a single coarse-grain application process. Several such processes can run concurrently on a single compute node or across multiple compute nodes and we assume that the ‘thin’ cores are not oversubscribed.

Multiple compute nodes on a single board are interconnected using optical waveguides [16] and multiple such boards are interconnected using board-to-board high-speed wireless links [10]. The on-board optical links have high data transmission rates, low transmission errors, and their topology is 2-D mesh [16]. The board-to-board wireless links are arranged around a compute node using very large Butler matrices (antenna arrays of 8x8 or 16x16) which correspond to narrow beams. When this is considered for both for transmitter and receiver nodes, the interference decreases significantly and can be neglected [10]. The wireless antennas use a beamforming architecture with phase shifters which enables suppression of signals from directions that are not desired. The placement of the wireless antennas around the compute nodes yields a 1-D mesh topology between neighboring boards. The 3$\times$3$\times$3 HAEC platform topology is schematically illustrated in Fig. 2a.



(a) 3×3×3 HAEC platform topology. Circles indicate compute nodes, blue/green lines indicate optical/wireless links, respectively.



(b) *lu.C.81* mapped onto the 3×3×3 HAEC platform using *block xyz* mapping. Color lines denote inter-process logical communications (red: ∼80k, turquoise: ∼160k, green: ∼240k).

Fig. 2. The HAEC platform (a) without and (b) with an assigned application

4 Modeling and Simulation Results

4.1 Mapping Applications to Systems

Simulating the behavior of any parallel application on the HAEC platform requires that the application processes be mapped to the nodes of the HAEC platform topology illustrated in Fig. 2a. High quality mappings increase the likelihood of achieving high performance and low energy consumption. The objectives of mapping application processes to compute nodes are *reducing the overall communication cost* and *maximizing parallelism*.

The *process-to-node* mapping is fixed for the duration of the simulation, i.e., no process is migrated. We compare three single-pass mapping strategies: xyz, block xyz, and random. These strategies are oblivious of the application communication requirements. The *xyz* mapping identifies the compute node to assign to an application process by first increasing the x coordinate of the last assigned node until every node along the x dimension is assigned, then by increasing the y coordinate and, finally, the z coordinate in the same manner. When the number of application processes is larger than the number of nodes in the system, the strategy proceeds in a round-robin fashion. Otherwise, *xyz* is the default mapping strategy. *Block xyz* mapping is similar to *xyz* and maps $\lfloor N/(d_x \cdot d_y \cdot d_z) \rfloor$ application processes to a single compute node, where d_x, d_y, and d_z are the number of nodes in the x, y, and z dimensions, respectively. *Random* mapping assigns application processes to compute nodes in a random fashion, and may result in unassigned system nodes when $N > (d_x \cdot d_y \cdot d_z)$. The three strategies result in different distributions of the application processes to the HAEC platform nodes, which in turn yield different numbers of intra-node and inter-node logical communications.

Table 1. Comparison of three process-to-node mappings for *lu.C.81*

Mapping	IePLC	IaNLC	IeNLC	AVG IeNLC	MIN IeNLC	MAX IeNLC
xyz		0	11,639,408	228,223	161,658	242,490
block xyz	11,639,408	**4,364,778**	7,274,630	173,205	80,829	242,488
random		646,633	10,992,775	99,934	80,829	242,488

To evaluate our approach we use the *lu* benchmark from the NPB 3.3 suite [3]. We chose *lu* because it performs a high number of point-to-point (unicast) messages and a small number of collective (multicast) messages. We use problem class C and execute it with 81 MPI processes (denoted *lu.C.81*) on 6 compute nodes of our current HPC production system[1].

In preparation for the simulations described in §4.2, we used the above strategies to map *lu.C.81* to the 27 nodes of the HAEC platform. The number of inter-process logical unicast communications (IePLC) of the benchmark is 11,639,408. These communications are illustrated in Fig. 2b where *lu.C.81* is mapped to the HAEC platform using *block xyz*. As comparison metrics (cf. Table 1), we use the number of intra-node logical communications (IaNLC), number of inter-node logical communications (IeNLC), and the average, minimum, and maximum number of IeNLC between any node pair. The *block xyz* strategy yields the smallest IeNLC value, which results in the largest IaNLC value. Thus, it is expected that *block xyz* results in the best overall simulated performance.

In reality, a single MPI process of *lu* represents more than single 'thin' core parallelism (e.g., as it is the case in the multi-zone version of this benchmark). In our approach, we abstract this parallelism and consider that a single MPI process partially or entirely exploits the available intra-node parallelism. When multiple MPI processes are mapped to the same compute node, we assume that they equally share the 'thin' cores of the node. In this work we concentrate on the inter-node communication requirements of applications mapped to the HAEC platform, and model them explicitly.

4.2 Application Performance for Different Communication Models

It is possible that the HAEC platform topology dynamically changes at runtime given the presence of wireless links. To accurately model the communication behavior of applications running on the HAEC platform, the communication models must account for the shape and characteristics of the interconnection network topology. Hence, we consider routing and network coding as alternative communication models. For the scope of this work the topology is assumed to be fixed (a 3-D mesh illustrated in Fig. 2a).

In standard routing, data packets are forwarded by intermediate nodes in a first come first serve manner. Network Coding (NC) [1] allows to increase throughput, energy efficiency, and robustness of data transmission in comparison to standard routing. These benefits result from the basic concept of NC to compute linear combinations of data packets instead of simply forwarding them.

[1] `https://doc.zih.tu-dresden.de/hpc-wiki/bin/view/Compendium/SystemTaurus`

The min-cut max-flow (the number of packets that can simultaneously be transmitted by a sender) of a network can be achieved using NC in unicast scenarios (a single sender transmits data to a single receiver) as well as in multicast scenarios (one or more senders transmit data to multiple receivers). NC can also be beneficial due to enhanced transmission robustness against node/link failures. Receivers require sufficient linear independent data packets to be able to decode by solving a system of linear equations, hence, the loss of single data packets can be mitigated.

To study the benefits of network coding versus routing in the context of the HAEC platform, we implemented both models in the simulator. The routing model is based on *dimension order routing* (DOR) [6]. Using DOR in a 3-D mesh (such as the one in Fig. 2b), packets are first routed in the x dimension, then in the y dimension, and lastly in the z dimension. The network coding approach is based on *practical network coding* (PNC, [5]), a practical implementation of random linear network coding. Random refers to the selection of the coefficients needed for computing the linear combinations of the data packets. In view of sending, the data packets are organized into matrices of s_w rows $\times$ $(s_w + n_s)$ columns, called generations (or windows), where s_w is the number of data packets per generation, and n_s is the number of data symbols per packet. The data packets are augmented by a global encoding vector that reflects all linear combinations applied to the data packets. Hence, the receiver does not need to know the randomly selected coefficients for decoding the combinations. For each packet, the first s_w columns contain the global encoding vector. In PNC, only data packets from one generation can be combined. PNC employs the same path selection between (sender,receiver) pairs as DOR.

At the moment, both communication models address only unicast communication. In unicast communication, NC is beneficial in case of packet loss caused by errors or attacks. In the simulations reported below, communication is assumed to be error-free. Thus NC will not outperform routing. However, integrating NC as a communication model in the simulator enables future evaluations in which certain packet loss rates will be considered.

Using NC for communication requires accounting for additional associated costs. In our case, the forwarding nodes are not burdened with additional computational effort for receiving the linear combinations of packets and for forwarding them. However, both sender and receiver nodes must perform additional operations, such as computing linear combinations or solving a system of linear equations. We assume that the nodes of the HAEC platform have sufficient computational resources; thus the additional operations will not significantly decrease efficiency. Analysis of the energy consumption of these operation will be conducted in future work. Regarding communication overhead, the fact that some additional information is transmitted (e.g., global encoding vector and generation identifier) needs to be considered. In comparison to the payload, which in our context refers to the amount of data symbols per packet, the cost of transmitting this additional information is also negligible.

Table 2. Parameters used in the simulation

Parameter	Notation	DOR	PNC
latency	l	$1\,\mu$s	
bandwidth	b	250 Gbit/s	
packet size	s_p	288 bytes	
sending delay	d_{out}	100 ns	
receiving delay	d_{in}	100 ns	
delay per hop	d_h	$d_{out} + s_p/b + l + d_{in}$	
acknowledgment processing delay	d_a	$d_{out}/2$	
delay intermediate node	d_i	d_a	
packet processing delay	d_p	0.625 ns	
finite field size	s_{ff}	8 bits	
window ID	s_{wid}	4 bytes	
delay sender node	d_s	$2{\cdot}d_{out}$	$2{\cdot}d_{out} + s_w \cdot d_p$
delay recv. node	d_r	$2{\cdot}d_{out}$	$2{\cdot}d_{out} + s_w^2 \cdot d_p$
payload/packet	L_p	$s_p - s_{wid}$	$s_p - s_w \cdot s_{ff} - s_{wid}$

Within our evaluations, we focus on comparing the transfer times of messages of applications running on the HAEC platform. To enable comparison between DOR and PNC, we assume that data packets are organized in windows of the same size as the generations. After sending one window (or generation) of data packets, the sender waits for the acknowledgment of receipt from the receiver before sending the next window of data packets. Given a payload L_p per data packet, sending a message of size m requires sending $n_p = \lceil m/L_p \rceil$ data packets and, hence, sending $n_w = \lfloor n_p/s_w \rfloor$ full windows (or generations) containing s_w data packets and a non-full window containing the remaining $n_r = n_p - s_w \cdot n_w$ data packets (if any). For the tests reported in the following, we set s_w to 5. Other parameters and their notation and values are given in Table 2.

Assuming the delay caused by sending a message over one hop (d_h) exceeds both the delay associated with preparing the data to be sent by the sender (d_s) and the associated delay at the receiver (d_r), the time to transfer x data packets over h hops between sender s and receiver r in the absence of errors is given by:

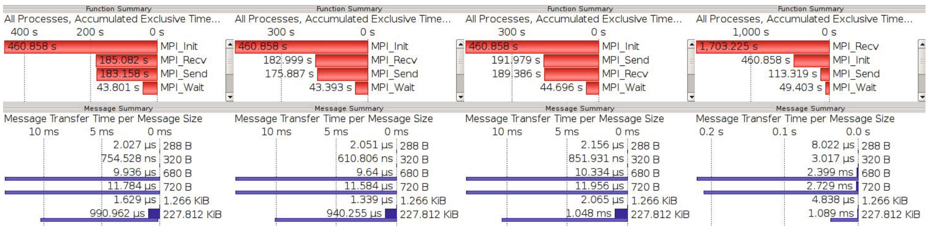
$$tt(x) = d_s + (h + x - 1) \cdot d_h + (h - 1) \cdot d_i + d_r, \tag{1}$$

where d_i denotes the delay associated with processing data packets at the intermediate nodes. When s and r are mapped to the same node, we assume $tt(x) = (d_s + d_r)/2$. The time needed for transmitting all data packets of message m is given by:

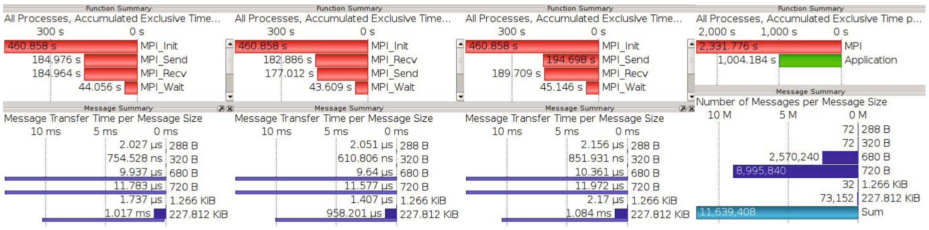
$$T(n_p) = tt(s_w) \cdot n_w + tt(n_r) + h \cdot (n_w + 1) \cdot (d_h + d_a), \tag{2}$$

where d_a refers to the delay associated with processing of an acknowledgment and $tt(s_w)$ is given by Eq. (1). Both DOR and PNC employ Eq. (1) and (2) with different payloads L_p and delays d_s and d_r (cf. Table 2). This holds for the error-free case.

The instrumented *lu.C.81* benchmark (cf. Sec. 4.1) ran in 41.8 s and resulted in a trace of 1.4 GiB. This trace was given as input to `haec_sim`. We conducted six simulations: one for each of the three mapping strategies, and for each mapping we employed DOR and PNC as communication models. Each simulation was



(a) xyz, DOR (b) block xyz, DOR (c) random, DOR (d) input trace



(e) xyz, PNC (f) block xyz, PNC (g) random, PNC (h) input trace

Fig. 3. Function and message statistics of the simulated *lu.C.81* running on the 3×3×3 HAEC platform. (d) shows function and message statistics of the input trace while (h) shows function groups statistics and message counts per message size of the input trace. Visualization with Vampir [12].

conducted in parallel on 6 compute nodes using 81 simulation processes, and completed in 675 seconds. The duration of the simulated *lu.C.81* benchmark on the HAEC platform was between 23.7 s to 24.1 s for the different mappings using DOR and PNC (Fig. 3). Two types of statistics are shown for each simulated trace: (1) the accumulated exclusive time spent in MPI functions and (2) the average transfer times for the different message sizes. The following statistics are shown additionally for the input trace: (3) accumulated exclusive time spent in functions of group Application (green bar) and MPI (red bar) and (4) the number of messages grouped by message size. The original trace is shown only for illustration and not for comparison against the simulated traces.

The choice of mapping or communication model has no impact on the duration of the simulated benchmark, even though most of the time is spent in MPI functions in the input trace (cf. Fig. 3d). Note that the communication models only alter the duration of the following MPI functions: Send, Recv, Wait, and Irecv. Time spent in all other functions is the same in both input and simulated traces. There are differences in the times spent in these four MPI functions and in the transfer times per message sizes among the three mappings and the two communication models.

From a mapping strategy perspective, less time is spent in send, recv, and wait for *block xyz* mapping using DOR and PNC, than in any other case. From a communication model perspective, more time is spent in send and wait using PNC than DOR for all mappings. This confirms our expectation, given that the simulation assumed an error-free HAEC platform and knowing that use of NC for unicast communication is only beneficial in the presence of errors or attacks. However, less time is spent in recv using PNC than DOR. This may be due to indirect balancing effects, such as faster transmissions leading to longer waiting times on subsequent messages. Also, the effect of mapping and communication model on message transmission time depends on the message size.

5 Conclusion and Future Work

This paper presents a holistic approach that uses a trace-driven workflow to simulate and analyze the performance of parallel applications running on a high performance–low energy computer (the HAEC platform). We have presented an application model based on event traces, an abstract model for the HAEC platform, as well as three strategies for mapping parallel applications to the HAEC platform. We have developed a trace-driven simulator (`haec_sim`) which employs two communication models: dimension order routing and practical network coding. The simulation results conducted on a well known parallel benchmark show the potential of the mapping strategies and the communication models for analyzing the performance of various parallel applications on the HAEC platform.

There are multiple future work directions. Immediate directions include: simulation experiments on various parallel applications from the scientific community; development of energy consumption models for computation and communication operations; development of mapping strategies that take into account the communication patterns of the application; modeling of unicast communication in the presence of errors/attacks; and modeling of HAEC platform (compute and communication) resources management in order to address, e.g., congestion over communication links. Longer-term work directions include: modeling of multicast communications; development of support for migration of tasks across compute nodes to increase performance or decrease energy costs; development of a hybrid communication model that supports dynamic latency, bandwidth, and topology.

Acknowledgements. This work is supported by the German Research Foundation (DFG) in the Collaborative Research Center 912 “Highly Adaptive Energy-Efficient Computing”. The authors thank Mario Bielert and Miriam Debus for contibution in simulator implementation and visualization.

References

1. Ahlswede, R., Cai, N., Li, S.-Y.R., Yeung, R.W.: Network information flow. IEEE Trans. on Inf. Theory 46(4), 1204–1216 (2000)
2. Argollo, E., Falcón, A., Faraboschi, P., Monchiero, M., Ortega, D.: COTSon: Infrastructure for full system simulation. SIGOPS Op. Sys. Review 43(1) (2009)
3. Bailey, D., Barszcz, E., Barton, J., Browning, D., Carter, R., Dagum, L., Fatoohi, R., Fineberg, S., Frederickson, P., Lasinski, T., Schreiber, R., Simon, H., Venkatakrishnan, V., Weeratunga, S.: The NAS parallel benchmarks. RNR Technical Report RNR-94-007, NASA (March 1994)

4. Bhatele, A.: Automatic Topology Aware Mapping for Supercomputers. PhD thesis, University of Illinois at Urbana-Champaign (2010)
5. Chou, P.A., Wu, Y., Jain, K.: Practical network coding. In: Proc. Annual Allerton Conf. on Comm., Control, and Computing (2003)
6. Dally, W.J., Towles, B.: Principles and Practices of Interconnection Networks. Morgan Kaufmann (2004)
7. Doany, F.E., Lee, B., Rylyakov, A., Kuchta, D.M., Baks, C., Jahnes, C., Libsch, F., Schow, C.: Terabit/sec VCSEL-based parallel optical module based on Holey CMOS transceiver IC. In: Optical Fiber Communication Conf. and Expo. and the National Fiber Optic Engineers Conf. (2012)
8. Eschweiler, D., Wagner, M., Geimer, M., Knüpfer, A., Nagel, W.E., Wolf, F.: Open Trace Format 2: The next generation of scalable trace formats and support libraries. In: Applications, Tools and Techniques on the Road to Exascale Computing. Advances in Par. Co, vol. 22, pp. 481–490 (2012)
9. Fettweis, G., Nagel, W.E., Lehner, W.: Pathways to servers of the future. In: Design, Automation, Test in Europe, pp. 1161–1166 (2012)
10. Israel, J., Martinovic, J., Fischer, A., Jenning, M., Landau, L.: Optimal antenna positioning for wireless board-to-board communication using a butler matrix beamforming network. In: 17th Int'l ITG Workshop on Smart Antennas, pp. 1–7. VDE (2013)
11. Kielmann, T., Gorlatch, S.: Bandwidth-latency models (BSP, LogP). In: Padua, D. (ed.) Encycl. of Par. Co., pp. 107–112. Springer, US (2011)
12. Knüpfer, A., Brunst, H., Doleschal, J., Jurenz, M., Lieber, M., Mickler, H., Müller, M.S., Nagel, W.E.: The Vampir performance analysis tool-set. In: Resch, M.M., Keller, R., Himmler, V., Krammer, B., Schulz, A. (eds.) Tools for High Perf. Comp, pp. 139–155. Springer (2008)
13. Knüpfer, A., Rössel, C., Mey, D., Biersdorff, S., Diethelm, K., Eschweiler, D., Geimer, M., Gerndt, M., Lorenz, D., Malony, A., Nagel, W.E., Oleynik, Y., Philippen, P., Saviankou, P., Schmidl, D., Shende, S., Tschüter, R., Wagner, M., Wesarg, B., Wolf, F.: Score-P: A joint performance measurement run-time infrastructure for Periscope, Scalasca, TAU, and Vampir. In: Brunst, H., Müller, M.S., Nagel, W.E., Resch, M.M. (eds.) Tools for High Perf. Comp, pp. 79–91. Springer, Heidelberg (2012)
14. Labarta, J., Girona, S., Cortes, T.: Analyzing scheduling policies using Dimemas. Par. Co. 23(1-2), 23–34 (1997)
15. Marowka, A.: Back to thin-core massively parallel processors. Computer 44(12), 49–54 (2011)
16. Nieweglowski, K., Rieske, R., Henker, R., Schöniger, D., Ellinger, F., Wolter, K.-J.: Optical interconnects for adaptive high performance computing. In: IEEE Workshop Photonics and Microsys. (July 2013)
17. Sahu, P.K., Chattopadhyay, S.: A survey on application mapping strategies for network-on-chip design. J. Syst. Archit. 59(1), 60–76 (2013)
18. Sherman, S.W., Browne, J.C.: Trace driven modeling: Review and overview. In: 1st Symp. on Simul. of Computer Sys., pp. 200–207. IEEE Press (1973)
19. Totoni, E., Bhatele, A., Bohm, E.J., Jain, N., Mendes, C.L., Mokos, R.M., Zheng, G.,, L.: V Kale. Simulation-based performance analysis and tuning for a two-level directly connected system. In: 17th IEEE Intl. Conf. on Par. and Dist. Sys., pp. 340–347 (2011)
20. Vantrease, D., Schreiber, R., Monchiero, M., McLaren, M., Jouppi, N.P., Fiorentino, M., Davis, A., Binkert, N., Beausoleil, R.G., Ahn, J.H.: Corona: System implications of emerging nanophotonic technology. In: IEEE Intl. Conf. on Progr. Comprehension, pp. 153–164 (June 2008)