Energy minimization at all layers of the data center: The ParaDIME Project

Oscar Palomar, Santhosh Rethinagiri, Gulay Yalcin, Ruben Titos-Gil, Pablo Prieto,
Emma Torrella, Osman Unsal, Adrian Cristal (BSC), Pascal Felber, Anita Sobe, Yaroslav Hayduk,
Mascha Kurpicz (UniNe), Christof Fetzer, Thomas Knauth (TUD), Malte Schneegaß,
Jens Struckmeier (Cloud&Heat), Dragomir Milojevic (IMEC)

***Abstract*—The main objective of the ParaDIME project has been to minimize energy consumption at all levels of the data center. On the one hand, we have considered what can be achieved on currently existing systems, via improvements of the programming model and the runtime system. On the other hand, we investigated which techniques and design decisions can make future computing nodes more energy efficient. We have successfully proposed and developed several methodologies that enable up to 60% energy savings in data center's components.**

I. Project Context and Objectives

Thanks to Moore's Law, the energy-efficiency of computers has been improving exponentially over the decades, i.e. one can perform more and more computations per joule. Nevertheless, the total energy consumption of the worldwide IT infrastructure has been increasing dramatically, and the main reason for this increase in energy consumption is that the number and scale of data centers has skyrocketed. At its core, the ParaDIME[1] Project ("Parallel Distributed Infrastructure for Minimization of Energy") aimed to provide methodologies that reduce the energy consumption of data centers by two approaches: First, we investigated energy-efficient computing techniques in the form of enhanced programming models and runtime improvements, which can be used with existing hardware. Second, we also investigated novel hardware architectures that leverage future device technology.

We can synthesize all these methodologies into the following high-level objectives of the ParaDIME Project:

- Objective 1: To develop and evaluate new energy-aware workload scheduling methodologies for Individual and Multiple Data Center infrastructures to radically decrease energy consumption.
- Objective 2: To develop and evaluate an energy-aware programming model that showcases energy-efficient software programming methodologies based on message passing for Computing Nodes that radically decrease energy consumption.
- Objective 3: To develop and evaluate new hardware methodologies for future Computing Nodes that use future devices or novel microarchitectures to radically decrease energy consumption.

ParaDIME was a three year research project launched in September 2012 with a total budget of 3,2M€, including 2.5M€ funding from the European Community's Seventh Framework Programme. The partners of the project were Barcelona Supercomputing Center – Centro Nacional de Supercomputación (BSC, coordinator), IMEC (Belgium), Technische Universität Dresden (Germany), Université de Neuchâtel (Switzerland) and Cloud&Heat (Germany).

[1]http://www.paradime-project.eu/

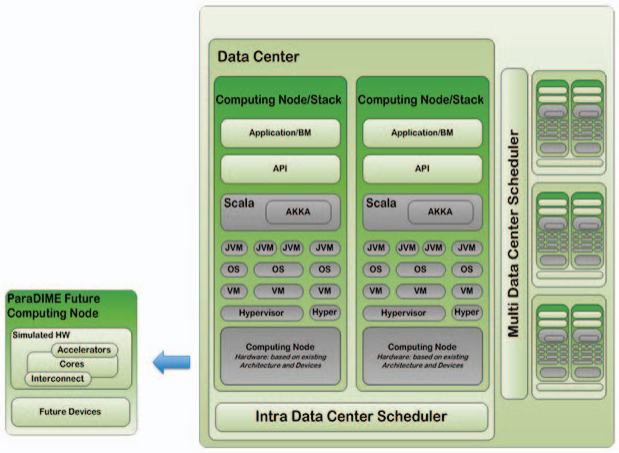

Fig. 1. The ParaDIME infrastructure.

II. Main Results

The ParaDIME project minimized energy consumption of data-centers by using different methodologies at the different levels of the system, as shown in Figure 1. In this section we present the main results achieved in each one of the levels, completing the preliminary results published in [R+15a].

A. ParaDIME Programming Model

In the context of ParaDIME our efforts were focused on the programmer's perspective on energy efficiency.

One of our main motivations to provide incentives for energy-efficient software was to provide solutions that are "programmable" or even transparent to the programmer. More specifically, the work was split into three parts: 1) efficient software-based message passing, 2) API support for lowering the supply voltage (V_{dd}) and 3) tools for power and cost awareness.

978-3-9815370-6-2/DATE16/

1) Efficient software-based message passing: For our efforts on the API level we chose Scala; a general-purpose language that runs on top of the JVM (Java Virtual Machine) and combines functional and object-oriented programming patterns. Scala provides many features, such as macros, support for transactional memory and a seamless integration with the Akka Framework. Akka is an implementation of the actor model. The actor model is a successful message-passing approach that introduces desirable properties such as encapsulation, fair scheduling, location transparency, and data consistency to the programmer [HBS73], [KSA09]. During this project we used and extended several features of Scala/Akka and evaluated them thoroughly [FHHS13], [HSH+13].

The main contributions are:

Static and dynamic parallel message processing: The key idea of our enhancement is to apply speculation, as provided by transactional memory (TM), to process more than one message at a time as if they were processed sequentially. In cases where these semantics are violated, we rely on the rollback capabilities of TM to undo the operations potentially leading to inconsistencies. To improve the performance of high contention workloads, we parallelize the processing of messages in a transactional context with messages that can run in a non-transactional context and further determine the optimal level of parallelism. With these measures we reached significant execution time reduction and consequently lowered the energy consumption without any additional effort for the programmer [HSF14], [HSF15].

Heterogeneous actors using GPUs: When executing actors on a heterogeneous platform (e.g. a system comprised of CPU and GPU cores), it is desirable for a programmer to be able to decide on a fine-grained level whether a task, encapsulated in an actor, should execute on a CPU or on the GPU. Our contribution is hence the design of an energy-efficient and programmable solution for heterogeneous actors. We investigated several strategies for implementing heterogeneous actors. We started from a manually crafted and optimized implementation, in which an actor calls CUDA code for the GPU using the Java native interface (JNI). A promising solution is to use a domain-specific language (DSL) as a basis for generating both CPU and GPU code whenever possible. The automatic scheduling extends the heterogeneous actor implementation presented before. We implemented a workload balancing scheduler that allows for finding the base share of CPU and GPU actors with a short profiling phase.

From a ParaDIME point of view there is a trade-off between programmability and energy-efficiency. Figure 2 summarizes the main results. JNI is the most efficient solution, leading to similar power consumption than a CPU-only implementation and radically reduced execution time of up to 80%. From a programmer's point of view the DSL actors might be more interesting, providing still a lowered energy consumption of around 40%. Automatic solutions or supporting solutions represent incentives for energy-efficient software and hence were one of the targets of ParaDIME. With the flexibility of the actor model, we improved the message processing throughput as well as increased the overall utilization of the system.

Fig. 2. Power dissipation (a), execution time (b) and energy consumption (c) of CPU and GPU implementations.

2) Support for lowering the V_{dd}: This feature evolved from simple annotations to sophisticated solutions, and finally tests on real hardware. Lowering the V_{dd} adds computational complexity, which means we need to call some method to adapt the frequency, for which we rely on tools provided by the operating system. We added annotations to our benchmarking applications k-means and hydra.

With this work we intended to provide mechanisms for future hardware to go below safe margins. To test the annotations, we evaluated our work on current hardware, i.e. safe voltage margins. We showed that our mechanisms are actually programmable and work, and detailed evaluation using hardware architecture simulators will be used to determine the energy savings.

3) Tools for power and cost awareness: To know how much power/energy an application or a process can save, we need to know how much it actually consumes. Hence, we focused on estimating the power consumption of a single process running in a virtualized environment with a tool called BitWatts [C+14], [CKF+15]. BitWatts and its extensions are a great basis for energy-efficient software, because only when we are able to identify power hotspots, we can take actions against them. These results can be used as a foreground for energy-efficient scheduling, pricing, programming, etc.

To attribute the energy consumption of a data center, we need to include not only the servers, but the entire infrastructure. Hence, we developed a model called EPAVE to account the overall data center's costs to a single user. EPAVE keeps the philosophy of the Cloud: the pay-as-you-go model but based on energy consumption. The costs of a virtual machine (VM) indeed depend on the physical resources reserved for it (static costs) and on the utilization made of these resources (dynamic costs). Moreover, the energy costs of a VM are predictable for the static part, and bounded for the dynamic part. Thus the user knows the maximal costs of the VM, and is able to estimate the real costs if the behavior of the running application and their energy consumption is known.

Energy-efficient scheduling starts at the application level, by understanding the effects of software on the underlying hardware in a heterogeneous data center. Our vision is that in a heterogeneous setup it is possible to estimate the power profile of an application with limited effort. Our initial studies

show some promising results, having estimation errors not higher than 10% [KSF14]. Hence, although our approach is still naïve, we are able to reach results similar to state-of-the-art power estimation algorithms.

B. ParaDIME Runtime

The ParaDIME runtime includes two schedulers, an intra data center scheduler and a multi data center scheduler. The runtime also covers the storage layer, where we developed custom hardware and software. The unifying goal was always to reduce the overall power consumption and to make the infrastructure as a whole more energy-efficient.

1) Intra Data Center scheduler: This scheduler optimizes the placement of VMs within a single data center. We developed technologies to reduce the cost of virtual machine migrations [KF15], [KKHF14], [KF14]. By keeping a copy of the VM at each server the VM visits, we can reuse a significant part of the cached copy on an incoming migration. Instead of transferring the entire state between the migration source and destination over the network, we can (partially) reuse the cached copy stored on the local hard disk. This allows us to change the virtual machine assignment more often. Previous work reported that energy efficiency gains of 15% to 50% are possible with VM consolidation [VBJ14]. With cheaper and more frequent consolidations, we hope to exceed these savings.

2) Multi Data Center scheduler: The multi data center scheduler assigns virtual machines between a set of highly distributed small data centers. We initially aimed to schedule tasks according to the energy mix, prioritizing green energy sources. However, every Could&Heat (C&H) data center uses 100% renewable energy, hence we abandoned this initial approach. Instead, the scheduling decision is driven by the amount of energy that can be reused for secondary purposes, e.g., warm water and heating. Cloud and Heat data centers are powered with green electricity, and don't require a cooling system. The byproduct heat generated by the servers is repurposed to heat buildings and drinking water. This process saves energy for heating and significantly lowers CO_2 levels. As the C&H infrastructure is co-located within residential buildings, the reusable heat is an indirect indicator of how much additional carbon dioxide is avoided, by reusing the server's waste heat. The multi data center scheduler matches cloud jobs with heat demands from connected deployments, aiming at maximizing heat reuse.

3) Energy-efficient storage: At the storage layer, we developed a custom logic board that connects a single hard drive to two servers, shown in Figure 3. Using this logic board we were able to power off servers and still access all hard drives, which was previously impossible. We are now able to power off up to 50% of the servers and still access all the hard drives. At the same time, we also adapted the popular key-value store Cassandra to run on a variable number of servers. Because Cassandra uses redundancy to increase availability, we were able to shut off unused servers during low-demand periods. In a typical scenario with 3-way replication, we are able to power off 2 out of 3 Cassandra servers while keeping all of the data accessible [TKF14].

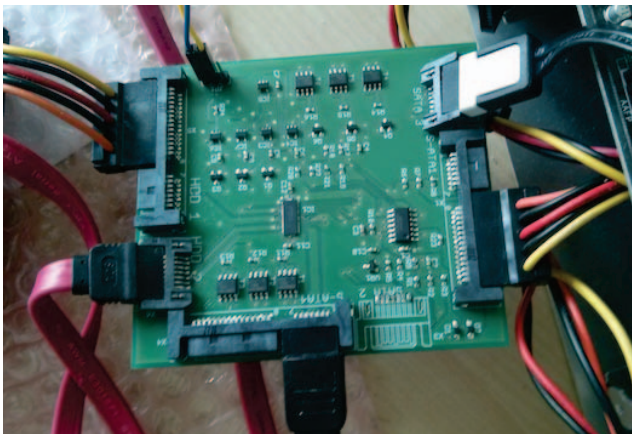

Fig. 3. Disk switch board prototype.

C. ParaDIME Future Hardware Node

We proposed and evaluated several techniques to improve the energy efficiency of Future Hardware Nodes. We envision such nodes using heterogeneous many-core processors, given the technology and industry trends. Thus, among other methodologies (e.g. reduce precision of floating point values) not discussed here, we focused on how to reduce energy consumption of such processors.

1) Heterogeneous computing: To deal with heterogeneity, a key decision is determining the most appropriate mapping of tasks to the cores (e.g. big.LITTLE processors [RPC$^+$14], or systems that combine FPGA, GPU and CPU cores [RPA$^+$15]). To achieve that, it is crucial to know how much energy a task would consume when running on each core, which can differ considerably in such heterogeneous systems, as illustrated in Figure 4. In ParaDIME, we researched power estimation methods and proposed tools that estimate per-task energy in a variety of core types, when combined with real hardware or simulators [RPA$^+$14]. Furthermore, we combined these tools with schedulers that consider not only which is the core type with the lowest energy for a given task, but also their occupancy and the cost of moving data if two dependent tasks execute in different cores. We save 40% of energy consumption on average by using our proposed heterogeneous systems composed of FPGA, GPU and CPU cores [R$^+$15b], when compared against multicore processor. Furthermore, with the scheduler methodology, we reduce 20% of the power and 35% of the energy on average across different types of heterogeneous platforms.

2) Efficient hardware-message passing: The popularity of message-passing is growing highly, e.g., with the use of actor-based programming models. This seems a very adequate way to program for many-core systems. Nevertheless, shared memory is likely to remain the design-of-choice for chip manufacturers in the coming years for many-core processors, given their wide versatility and the vast experience that

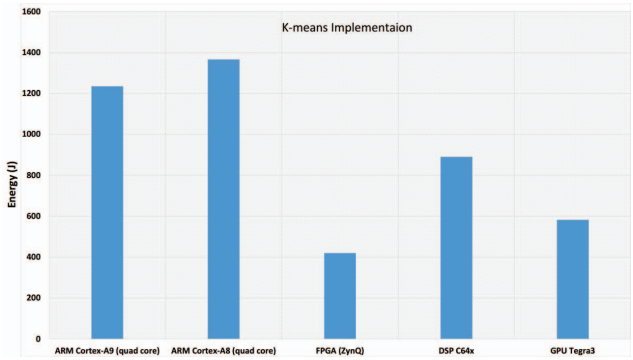

Fig. 4. Energy consumed in different computing devices.

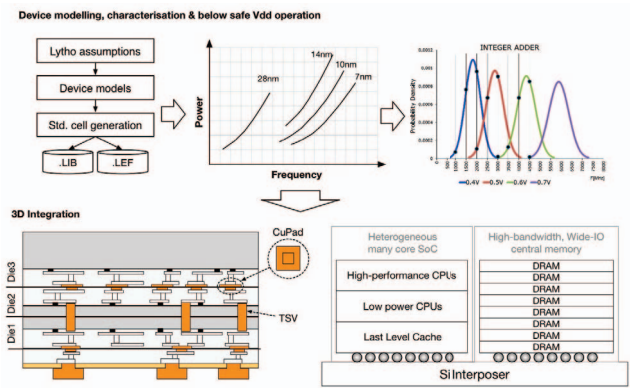

Fig. 5. The framework developed in ParaDIME for studying below safe V_{dd} operation (top), and our 3D packaging model (bottom).

industry has accumulated on this domain. This mismatch provokes substantial efficiency losses. Thus, we investigated hardware support that enables more efficient execution of such workloads in commodity many-core processors. We tackled the overheads of transmitting a message via shared memory by providing instructions that copy directly the message data from private cache of the sender to private cache of the receiver, instead of using any intermediate buffer in a shared cache [TGPUC15]. The consequence is a greatly reduced number of cache accesses and coherence protocol messages, with a subsequent reduction of energy spent on sending the data, on average 50% of the L1 data cache dynamic energy. We benchmarked the proposal with several unmodified MPI applications and limited modifications to a state-of-the-art MPI library (MPICH), reducing the energy-delay product by 11%. Please note the MPI applications we used ummodified have a high computation to communication ratio. Finally, we extended the proposal with a very simple coprocessor-based implemented, that frees the CPU to do useful computation while data is sent. The performance improvements results in further energy savings: 80% on average less dynamic energy consumed on the L1 data cache during data transfer.

3) Lowering the V_{dd}: In addition to these processor-level approaches, we investigated how to make each core more energy efficient. We studied how to aggressively push V_{dd} lowering, a well known power- saving technique, beyond the safe limits required to ensure correct operation of the units [YUC13], [Y+14]. We were able to save energy when we combine this with low overhead error detection and correction techniques. For example, in the L1 caches, we saved up to 60% energy with our circuit-driven replication scheme [YSUC14]. We analyzed as well the impact of using per-unit V_{dd} reduction. For instance, the energy consumption of the integer adder can be reduced up to 50% with only less than 10% performance overhead in the application.

D. ParaDIME Future Devices

As Figure 5, at the device level we have used future device technology models to estimate the behavior of circuits operating at below the safe V_{dd}. Moreover, we have also studied advanced packaging solutions.

1) Advanced device technology: We have proposed an integrated framework for generation of usable technology libraries based on advanced photolithography assumptions. The framework is using low-level device models (TCAD) to establish the basic building blocks of any gate: the transistor. Further, based on these transistors models, complete standard cell libraries have been built for 14, 10 and 7nm technology nodes (.lib/.lef) that can be used within any ASIC (Application Specific Integrated Circuit) design flow (synthesis, place & route). Different device structures and geometries have been explored to understand power/performance/area (i.e. cost) trade-offs.

2) Below safe V_{dd} operation framework: Using the generated technology libraries described above, and a dedicated framework for ASIC implementation, we implemented (synthesis) a typical set of digital sub- circuits that are part of any CPU (adder/multiplier/register file etc.). Using design automation scripts, specifically developed for this task, different sub-circuits have been characterized for both safe and below-safe V_{dd} operating conditions. The results have been passed to architecture-level simulator developed for ParaDIME future computing node to derive the actual power savings at architecture/application level.

3) Advanced packaging assessment: Using the 3D roadmap of IMEC, we were able to select appropriate stacking technologies to allow efficient heterogeneous design integration at system-level [MOM+15], [M+13]. A novel architecture, ParaDIME Computing Node, has been proposed, taking the advantages of both device and packaging worlds. Based on system level assumptions, different experiments were performed to highlight various performance benefits of future 3D tech heterogeneous systems. These experiments spanned all abstraction levels in the system design, starting with device modeling, up to full system integration including 3D system design. With these experiments we were able to quantify the impact of both device and packaging level technologies on the system characteristics and validate the proposed approach. Finally, we showed that such integration could be even cost-effective, if carefully and smartly designed. And this is crucial

since cost is the key ingredient for the acceptance of any realistic guess on future technologies and architectures.

III. Lessons learned and recommendations

We recommend the following research directions at the different levels based on our findings:

A. Programming model

On the API level, we have shown that the actor model is beneficial for heterogeneous programming. Having this in mind, we would recommend to continue the efforts in the direction of dynamic scheduling of the tasks, e.g., using work-stealing approaches. Other projects have intestigated schedulers for data-flow execution models [GS15], [G+14]. For non-iterative workloads, and workloads with different resource utilization patterns this might lead to better utilization of resources and improved energy efficiency.

On the power estimation level, we were able to model accurately the power characteristics of Intel CPUs regardless of its hardware features and additionally covered virtual environments. As a next step, we would recommend to work on manufacturer independent models, e.g., by introducing a learning phase per model. With the knowledge of model-dependent patterns, hardware simulators can be extended and even power characteristics of future devices might be predictable.

On the energy-efficient scheduling level, we showed that with little effort, we are able to provide coarse-grain information about repetitive workloads. As a future direction, we would recommend to include the idle costs of heterogeneous data centers in scheduling decisions for distributing workloads or shutting down the non-needed machines, i.e., combining the efforts of the runtime.

B. Runtime

At the runtime level, we envision a large decentralized infrastructure of small scale data centers, motivated by the gains in efficiency demonstrated by Cloud&Heat (C&H). New application scenarios are enabled by this widely distributed infrastructure. Physical proximity is paramount for scenarios where low latency, around one millisecond, is required. For example, with autonomous cars and their coordination to optimize the traffic flow, computation must happen close to the source of the data, i.e., the cars. We see a great potential to position the distributed infrastructure of C&H, to enable these and other upcoming use cases.

C. Hardware architecture

We summarize our recommendations regarding hardware architecture into two directions which we consider have the higher priority:

Improve and extend: We have not exhausted the research on the most promising hardware techniques proposed in ParaDIME. For example, the power estimation tools would benefit from extending to other core types, and the scheduling algorithms that use them are initial approaches, there is substantial room for improvement. In particular, automated task- mapping for complex heterogeneous platforms (e.g. with three types of cores: CPU, GPU and FPGA) guided by our power estimation tools is highly desirable. We consider that continuing the research on lightweight reliability mechanisms for operation below safe V_{dd} will lead to further energy savings. Moreover, such advances in energy-efficient error detection and recovery are likely to reduce energy consumption of reliability mechanisms in general, which will grow in importance in future systems. Our per-unit V_{dd} lowering technique has shown high potential for energy savings and should be studied in more detail.

Extend software integration and improve the compiler: The proposed technique for efficient message passing should be coupled with other message passing implementations to achieve its full potential (in particular, with actor-based programming models). That said, its adoption has the potential to change the mentality of MPI programmers, so that MPI programs are not necessarily aggressively optimized and structured to minimize communication. If the cost of communicating is low, programmers can create smaller tasks, since less computation is required to amortize the communication cost. Smaller task granularity can lead to exploit more parallelism. Another aspect where software plays a key role is heterogeneous computing. Currently, compiler-generated code for accelerators is less efficient than low-level accelerator- specific code, with the resulting risk for programmability. Finding ways to overcome this is a crucial step for allowing most programmers to truly exploit the energy efficiency possible with heterogeneous platforms that we have shown.

D. Device and packaging

At device and packaging levels, future technologies have anticipated the way in which future systems should be built. While we still do have some future with the current CMOS technologies (up to 7nm at least), the road to wide adoption of these technologies is not going to be simple (complexity and cost). Heterogeneous integration (at technology level) is most likely to be the only way to enable the usage of such advanced technologies at device-level, and this is only possible if appropriate stacking technologies (3D, 2.5D) are used. To take the best of these technologies the co-optimization at device/architecture/system/ application should be put in place. The current basic architectural model of a computing node should thus change, because current technology-homogeneous approach in ASIC integration will not hold any more.

Because of the associated costs of the future emerging technologies (and eventual practical considerations such as IO and analogue system performance), advanced devices will have to be combined in technologically heterogeneous systems. Also, this would mean that the given process could be fine-tuned for a specific function, especially memories, where the requirements for the process technology are different from the ones used for logic. In this project we only partially answered the question on the efficiency of such systems. But it did set-up the groundwork for future exploration of heterogeneous system at technology, architecture, system and application levels.

While this work quantified important system parameters in the presence of the realistic process assumptions, a lot of room is still left for further system optimization and refinements. Further, while fully supported by the IC community, Wide IO DRAMs will need a very solid software support to really unleash all their potential. That support is fully lacking in the Operating System/Application development layers as of today.

IV. Conclusion

Researchers from the ParaDIME project have successfully proposed and developed several methodologies that enable energy savings up to 60% in the data center. These methodologies tackle several critical power-related research challenges: starting from using different current and future devices to multi data center VM scheduling.

In the programming model level, the focus has been set on shifting from the shared memory model to an actor-based message passing programming model, focusing on the programmer's perspective on energy efficiency. In the runtime level, we developed two schedulers, an intra data center scheduler and a multi data center scheduler. The runtime also covers the storage layer, where we developed custom hardware and software. The unifying goal was always to reduce the overall power consumption and to make the infrastructure as a whole more efficient. In the hardware level, ParaDIME researchers have proposed and simulated several methodologies for improving energy-efficiency of the Future computing node, including lowering the V_{dd} below safe levels, heterogeneous computing and support for efficient message passing, as well as studied future device and packaging technologies.

Considering these results, that have resulted in near 50 publications in international peer-reviewed conferences, journals and workshops, we consider the ParaDIME project a success.

Acknowledgments

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under the ParaDIME Project (www.paradime-project.eu), grant agreement n. 318693.

References

[C+14] M. Colmant et al. Bitwatts: A process-level power monitoring middleware. In *Middleware 2014 Posters/Demos*, 2014.

[CKF+15] M. Colmant, M. Kurpicz, P. Felber, L. Huertas, R. Rouvoy, and A. Sobe. Process-level power estimation in vm-based systems. In *EuroSYS*, 2015.

[FHHS13] P. Felber, D. Harmanci, Y. Hayduk, and A. Sobe. Concurrent message processing using transactional memory in the actor model. In *EURO-TM Workshop on Transactional Memory (WTM, no proceedings)*, 2013.

[G+14] Roberto Giorgi et al. Teraflux: Harnessing dataflow in next generation teradevices. *Microprocessors and Microsystems*, 38(8, Part B), 2014.

[GS15] R. Giorgi and A. Scionti. A scalable thread scheduling co-processor based on data-flow principles. *Future Generation Computer Systems*, 53, December 2015.

[HBS73] C. Hewitt, P. Bishop, and R. Steiger. A universal modular actor formalism for artificial intelligence. In *IJCAI'73: Proceedings of the 3rd international joint conference on Artificial intelligence*, 1973.

[HSF14] Y. Hayduk, A. Sobe, and P. Felber. Dynamic concurrent message processing with transactional memory in the actor model. In *TRANSACT 2014*, 2014.

[HSF15] Y. Hayduk, A. Sobe, and P. Felber. Dynamic message processing and transactional memory in the actor model. In *IFIP International Conference on Distributed Applications and Interoperable Systems, DAIS*, 2015.

[HSH+13] Y. Hayduk, A. Sobe, D. Harmanci, P. Marlier, and P. Felber. Speculative concurrent processing with transactional memory in the actor model. In *International Conference on Principles of Distributed Systems (OPODIS)*, 2013.

[KF14] T. Knauth and C. Fetzer. Dreamserver: Truly on-demand cloud services. In *SYSTOR*, 2014.

[KF15] T. Knauth and C. Fetzer. Vecycle: Recycling vm checkpoints for faster migrations. In *Middleware*, 2015.

[KKHF14] T. Knauth, P. Kiruvale, M. Hiltunen, and C. Fetzer. Sloth: Sdn-enabled activity-based virtual machine deployment. In *Hot Topics in Software Defined Networking*, 2014.

[KSA09] R. K. Karmani, A. Shali, and G. Agha. Actor frameworks for the jvm platform: a comparative analysis. In *PPPJ '09: Proceedings of the 7th International Conference on Principles and Practice of Programming in Java*, 2009.

[KSF14] M. Kurpicz, A. Sobe, and P. Felber. Using power measurements as a basis for workload placement in heterogeneous multi-cloud environments. In *CrossCloudBrokers '14 (co-located to Middleware 2014)*, 2014.

[M+13] D. Milojevic et al. Design issues in heterogeneous 3d/2.5d integration. In *18th Asia and South Pacific Design Automation Conference, ASP-DAC 2013*, 2013.

[MOM+15] F.L.T. Maggioni, H. Oprins, D. Milojevic, E. Beyne, I. De Wolf, and M. Baelmans. 3d-convolution based fast transient thermal model for 3d integrated circuits: methodology and applications. In *Thermal Measurement, Modeling Management Symposium (SEMI-THERM), 2015 31st*, 2015.

[R+15a] S.K. Rethinagiri et al. Paradime: Parallel distributed infrastructure for minimization of energy for data centers. *Microprocessors and Microsystems*, 2015.

[R+15b] S.K. Rethinagiri et al. Trigeneous platforms for energy efficient computing of hpc applications. In *22nd annual IEEE International Conference on High Performance Computing (HiPC 2015) (to appear)*, 2015.

[RPA+14] S.K. Rethinagiri, O. Palomar, R.B. Atitallah, A. Cristal, O. Unsal, and S. Niar. Pets: Power and energy estimation tool at system-level. In *ISQED*, 2014.

[RPA+15] S.K. Rethinagiri, O. Palomar, J. Arias, A. Cristal, and O. Unsal. Facet: Fast and accurate power/energy estimation tool for cpu-gpu platforms at architectural-level. In *System-on-Chip Conference (SOCC), 2014 27th IEEE International*, 2015.

[RPC+14] S.K. Rethinagiri, O. Palomar, A. Cristal, O. Unsal, and M. Swift. Dessert: Design space exploration tool based on power and energy at system-level. In *System-on-Chip Conference (SOCC), 2014 27th IEEE International*, 2014.

[TGPUC15] R. Titos-Gil, O. Palomar, O. Unsal, and A. Cristal. Architectural support for direct message passing on shared memory multi-cores. In *The 44th International Conference on Parallel Processing (ICPP-2015)*, 2015.

[TKF14] F.L. Tena, T. Knauth, and C. Fetzer. Powercass: Energy efficient, consisten hashing based storage for micro clouds based infrastructure. In *IEEE 7th International Conference on Cloud Computing*, 2014.

[VBJ14] A. Verma, J. Bagrodia, and V. Jaiswal. Virtual machine consolidation in the wild. In *Proceedings of the 15th International Middleware Conference*, 2014.

[Y+14] G. Yalcin et al. Combining error detection and transactional memory for energy-efficient computing below safe operation margins. In *Parallel, Distributed, and Network-Based Processing (PDP)*, 2014.

[YSUC14] G. Yalcin, A. Seyedi, O. Unsal, and A. Cristal. Flexicache: Highly reliable and low power cache under supply voltage scaling. In *CARLA Latin American High Performance Computing Conference*, 2014.

[YUC13] G. Yalcin, O. Unsal, and A. Cristal. Faultm: error detection and recovery using hardware transactional memory. In *DATE*, 2013.