Microprocessors and Microsystems xxx (2015) xxx–xxx

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

ParaDIME: Parallel Distributed Infrastructure for Minimization of Energy for data centers

Santhosh Kumar Rethinagiri [a,*], Oscar Palomar [a], Anita Sobe [b], Gulay Yalcin [a], Thomas Knauth [c], Rubén Titos Gil [a], Pablo Prieto [a], Malte Schneegaß [d], Adrian Cristal [a,f], Osman Unsal [a], Pascal Felber [b], Christof Fetzer [c], Dragomir Milojevic [e]

[a] Barcelona Supercomputing Center – Centro Nacional de Supercomputación (BSC), Spain
[b] Université de Neuchâtel (Unine), Switzerland
[c] Technische Universität Dresden, Germany
[d] Cloud & Heat Technologies GmbH, Germany
[e] Interuniversitair Micro-Electronica Centrum (IMEC), Belgium
[f] IIIA – CSIC – Spanish National Research Council, Spain

ARTICLE INFO

Article history:
Received 23 December 2014
Revised 1 May 2015
Accepted 5 June 2015
Available online xxxx

Keywords:
Low power
Runtime energy optimization
Programming models
Error recovery
Approximate computing
Message passing accelerators

ABSTRACT

Dramatic environmental and economic impact of the ever increasing power and energy consumption of modern computing devices in data centers is now a critical challenge. On the one hand, designers use technology scaling as one of the methods to face the phenomenon called dark silicon (only segments of a chip function concurrently due to power restrictions). On the other hand, designers use extreme-scale systems such as teradevices to meet the performance needs of their applications which in turn increases the power consumption of the platform. In order to overcome these challenges, we need novel computing paradigms that address energy efficiency. One of the promising solutions is to incorporate parallel distributed methodologies at different abstraction levels.

The FP7 project ParaDIME focuses on this objective to provide different distributed methodologies (software–hardware techniques) at different abstraction levels to attack the power-wall problem. In particular, the ParaDIME framework will utilize: circuit and architecture operation below safe voltage limits for drastic energy savings, specialized energy-aware computing accelerators, heterogeneous computing, energy-aware runtime, approximate computing and power-aware message passing. The major outcome of the project will be a noval processor architecture for a heterogeneous distributed system that utilizes future device characteristics, runtime and programming model for drastic energy savings of data centers. Wherever possible, ParaDIME will adopt multidisciplinary techniques, such as hardware support for message passing, runtime energy optimization utilizing new hardware energy performance counters, use of accelerators for error recovery from sub-safe voltage operation, and approximate computing through annotated code. Furthermore, we will establish and investigate the theoretical limits of energy savings at the device, circuit, architecture, runtime and programming model levels of the computing stack, as well as quantify the actual energy savings achieved by the ParaDIME approach for the complete computing stack with the real environment.

1. Introduction

The growing popularity of cloud computing has greatly increased the scale of data centers resulting in significant increase in power dissipation. We have identified several issues at different levels (programming model, runtime, hardware and device) that need to be addressed in order to reduce the power consumption and increase the energy efficiency of data centers.

The first issue is that modern servers suffer from the vivid power fluctuations due to sudden changes in workload. This is attributed to lower utilization of processor power for running the software. In real scenarios, the processor needs only 20% of its processing power to execute the application. When scaling beyond

* Corresponding author at: BSC-Microsoft Research Centre, Nexus I Building, Office 303, Campus Nord UPC, Gran Capita 2-4, 08034 Barcelona, Spain. Tel.: +34 93 4054062; fax: +34 93 413 77 21.
E-mail address: santhosh.rethinagiri@bsc.es (S.K. Rethinagiri).

http://dx.doi.org/10.1016/j.micpro.2015.06.005
0141-9331/

the computing node to the data center, power consumption is independent of the computing load of the system. The nodes keep state in memory and on local disk which means that they cannot be turned off even if the load is low. For this issue, we need to propose a novel scheduling policy between the computing nodes to raise work load at the same time adhering to Service Level Agreements (SLAs).

The second issue is the trend in technology minimization and the associated gains in performance and productivity. On the one hand, we expect technology scaling to finally come face-to-face with the problem of dark silicon [1] (only segments of a chip can function concurrently due to power restrictions), which will push us to use devices with completely new characteristics that must be studied. On the other hand, as core counts increase, the shared memory model based on cache coherence will severely limit code scalability and increase energy consumption. Therefore, to overcome these problems, we need new computer architectures that are radically more energy efficient.

The third issue is related to the programming model for the data center applications. It must provide interface such as annotations with the applications to maximize the utilization of resources during runtime. It also must interact with the hardware to provide information that can be used to apply aggressive energy saving techniques. Moreover, it should not rely on the shared-memory paradigm to enable scaling to a high number of cores.

The ParaDIME project addresses these issues to minimize energy consumption of data centers. The high level objectives of the ParaDIME Project can be summarized as follows:

- Objective 1: To build a reference Individual and Multiple Data Center Infrastructure that incorporates new energy conscious workload scheduling techniques utilizing information from the runtime to radically decrease energy consumption; to quantify the energy savings from employing these techniques by running multiple applications to stress test this Data Center platform.
- Objective 2: To develop an energy-aware programming model driving an associated ecosystem, the ParaDIME Computing Node (applications, runtime and architecture based on existing Hardware) that showcases energy-efficient SW programming methodologies that radically decrease energy consumption; to quantify the energy savings from employing these methodologies by running multiple applications to stress test this Computing Node.
- Objective 3: To simulate a Future Computing Node based on novel HW design techniques and new emerging devices that function at the limit of CMOS scaling to radically decrease energy consumption; to quantify the energy savings from employing these methodologies.

The main outcomes of the project will be:

- Outcome 1 (Objective 1): A reference Individual and Multiple Data Center Infrastructure that will help European companies develop beyond green technology product offerings based on the most promising energy-efficient computing methodologies.
- Outcome 2 (Objectives 2 and 3): A roadmap that indicates the most promising individual or combination of energy-efficient SW and HW methodologies by quantifying the energy cost/ benefit for the Computing Node and (possibly) the Data Center making use of both novel configurations of existing HW as well as forward-looking architectures based on emerging devices.

The rest of the paper is organized as follows. Section 2 presents the proposed ParaDIME project flow and methodologies. Selection of the benchmarks is given in Section 3. Section 4 illustrates the preliminary results of methodologies proposed. Finally, we conclude in Section 5.

2. ParaDIME project flow and methodologies

ParaDIME stands for Parallel Distributed Infrastructure for Minimization of Energy. As the name states, this project focuses on the minimization and optimization of energy consumption for the data center. Fig. 1 presents an overview of the project which is based on several energy minimization methodologies. Before discussing the ParaDIME Approach in detail, we must first introduce the ParaDIME Infrastructure, which is shown in Fig. 1. There are two types of computing nodes in ParaDIME. The one used in the Data Center which is build using existing hardware and the ParaDIME Future Computing Node that represents the prototype developed with the simulator at the architectural-level as shown in Fig. 1. The various energy efficient methodologies, which are proposed in this project at different level are as follows:

At the highest level, the ParaDIME Infrastructure (initially based on existing Hardware) consists of the ParaDIME Computing Node, the Individual Data Center as well as Multiple Data Centers. The Infrastructure relies on the Scala Programming Model and thus natively supports programming languages running in the Java Virtual Machine as well as their external components and their associated runtimes and real hardware. It consists of the following basic components:

- The existence of ***applications*** is critical to showing proof-of-concept for our energy-efficient programming methodologies. Moreover, it provides the programming community as well as system integrators with an example as to how our programming model may be employed.
- ***Language extensions*** and programming language constructs also referred to as ***APIs*** are the most visible aspects of the ParaDIME Approach to the programmers. While it may be possible to add support to applications via explicit library calls, this approach is not satisfactory for large systems as it relies on coding conventions, leads to unnecessarily complex code and is typically error prone. The addition of new language constructs with well-defined semantics is the soundest approach to import support into existing languages.
- The ***runtime*** is the central component of the Computing Node, the Individual and Multiple Data Center Infrastructure as it implements the scheduling logic. It consists of several schedulers in order to orchestrate message passing at the Node level as well as to schedule jobs between nodes (Intra Data Center Scheduling) and between Data Centers (Multi- Data Center Scheduling).
- At both the Individual and Multiple Data Center level, the ParaDIME Infrastructure uses ***real hardware*** which allows us to analyze trade-offs with respect to current hardware as well as provides initial insight as to where the bottlenecks may be in the hardware of the future.

The ParaDIME Future Computing Node consists of the following:

- A ***simulated hardware platform*** is used expressly to verify new ideas for hardware support to increase energy efficiency.
- Finally, we will simulate at the Computing Node using inputs for ***the near- and far-future devices*** to provide initial insight into their behavior.

In the upcoming subsections, we will describe the different methodologies based on which this project is built upon.

Please cite this article in press as: S.K. Rethinagiri et al., ParaDIME: Parallel Distributed Infrastructure for Minimization of Energy for data centers, Microprocess. Microsyst. (2015), http://dx.doi.org/10.1016/j.micpro.2015.06.005

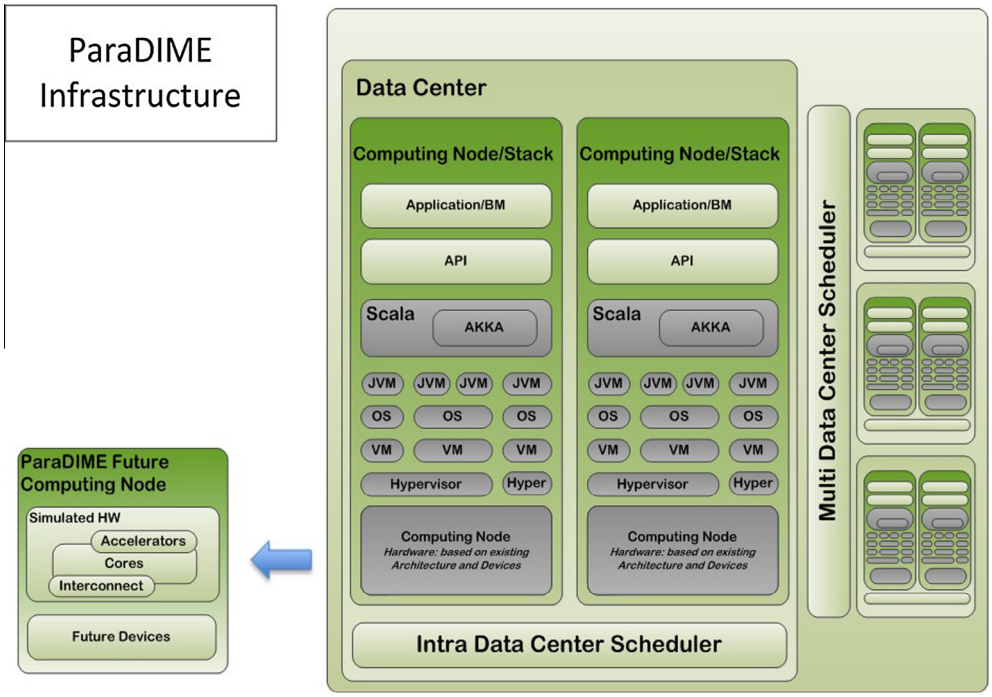


Fig. 1. ParaDIME project flow with two computing nodes (simulated and real hardware).

2.1. Device-level

2.1.1. Emerging devices

The geometric scaling has been the driving force during the past decades of CMOS dominance, which is losing steam, we expect the device scaling in the next decade to be tougher, relying on material science and engineering. Fig. 2 shows the logic V_{dd} scaling trend over different technology nodes. As is evident from the figure, the diminishing difference between V_{dd} and V_t results in performance degradation as well as induces more variability in the design. To balance this problem, commensurate electrostatic and mobility scaling is necessary. In the near future FinFET devices seem plausible as an effective means to extend MOS scaling for high-performance or low-power technologies at 20 nm and beyond. They provide sufficient protection against short-channel effects, especially "off-state" leakage current. Fig. 2 shows the device roadmap in the upcoming years. It is widely expected that around 11 nm and beyond, strained silicon may run out of steam and alternative channel materials will be required to achieve the performance targets at low power. As a result, III–V MOS devices have established itself as a viable candidate for technology nodes of 10–14 nm.

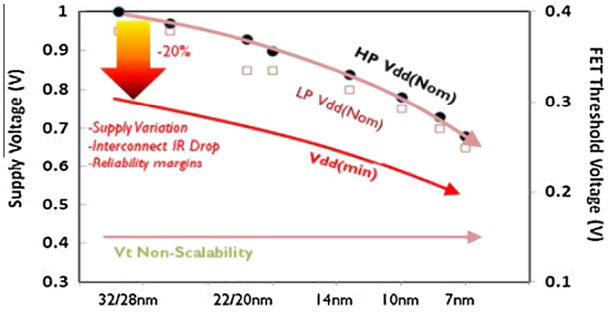


Fig. 2. Logic V_{dd} Scaling Trend-Asymmetric V_{dd} and V_t scaling trend hampers the required voltage headroom to maintain the performance scaling in advanced nodes. This headroom control with V_{dd} is viable to maintain energy-scalable architectures.

As we go beyond 10 nm scaling, the III–Vs are promising successors of MOSFETs due to their potential for sub-60 mV sub-threshold swing. Such a reduced swing will be a requirement for the ultra-low voltage operation of future generation of transistors. For the near future, the behavior of these devices is expected to follow past patterns, however the emerging devices for the mid to far future will require new logic cell concepts that steer us towards the energy-efficient operation and yield-aware design. ParaDIME will develop functional blocks such as adders, multipliers, register file as well as virtual prototypes of different ARM processors cores to enable architectural-level designers to predict the performance for both near and far-future technology devices to minimize the energy consumption.

2.1.2. Voltage limits

Due to the recent issues impacting device scaling as we approach the end of the CMOS roadmap, safe operation margins have been increasing. In particular, there is a substantial "tax" in the case of guard-bands for supply voltage. This guard-band is increasing due to systematic and random variability, increased thermal stresses and noise margins;. If we go below the safe limit and the associated guard-band, one might encounter sporadic errors while simultaneously saving energy dramatically. As an example, decreasing the supply voltage from 1.2 V to 0.7 V will lead to 3× dynamic energy savings. This will require support from other levels, for example using selective duplex replication at the architectural and programming model levels for driving the chip below safe V_{dd}.

2.2. Architectural-level

2.2.1. Efficient message passing

At the microarchitecture level, we have two goals for efficient message passing: (1) to leverage message passing to implement an efficient and scalable architecture and (2) to reduce the energy consumption of delivering messages. We will achieve the first goal by eliminating the structures and overhead required for cache-coherent shared-memory systems, thus improving the overall energy-efficiency of the system. We are aware that

Please cite this article in press as: S.K. Rethinagiri et al., ParaDIME: Parallel Distributed Infrastructure for Minimization of Energy for data centers, Microprocess. Microsyst. (2015), http://dx.doi.org/10.1016/j.micpro.2015.06.005

cache-coherency is the state-of-the-art for chip multiprocessors, and that it is very convenient for the programmer as data communication between cores is hidden. However, scaling to higher core counts is not trivial and alternatives are already being explored for many-core architectures [2]. We will address our second goal of reducing the energy consumption of delivering messages by providing two relatively simple mechanisms: including instructions to send messages to a particular level of the cache hierarchy of the destination core and avoiding copying data when sending messages to the same core.

On the top of that, we propose to use a message passing co-processor that accelerates the processing of messages by offloading the it to the co-processor and avoiding the OS/runtime software. One step on the road to achieving this accelerated processing of message is to have fast-task switching between the threads in the processor. The cost of spawning a thread can be very high and it is not cost effective if the task to execute is small. This would typically prevent programmers to specify small tasks, with the consequence of not expressing part of the available parallelism. We envision a system where the main core can spawn threads in a co-processor with much reduced cost, enabled by a very simple interface based on full/empty bits.

2.2.2. Operation below safe Vdd

Reducing the supply voltage of a circuit (Vdd) is a well-known technique for making trade-offs between performance and power [3]. Many commodity processors implement Dynamic Voltage and Frequency Scaling (DVFS) and offer a number of power modes that use different levels of supply voltage. However, the applicability of Vdd reduction is limited by safeguard bands that are necessary to ensure correctness of execution. Redundancy can be used to detect and correct errors, but full replication incurs significant energy overhead if not used carefully. In ParaDIME, we investigate techniques to lower the Vdd below the safe limits that only result in an overall reduction in energy consumption. We assume that the programmer indicates when performance can be aggressively traded off for reduced power dissipation (e.g., with the help of annotations at the programming level).

2.2.3. Reduced precision computing

In ParaDIME, we will approximate floating point computation by reducing the precision whenever the programmer indicates that computation does not require high precision, e.g. using the IEEE standard 754 format for floating point. Thus, floating point data will use fewer bits than standard format (e.g. 16), and both the floating point unit that operates them and the register file can be considerably smaller [4]. We will also implement a similar mechanism for integer values that are known to be narrow, i.e. that are inside a small range of values [5]. In a similar fashion, data caches and memory will be modified to exploit the smaller size of data, either by powering down some blocks or packing more data in the same capacity. This can also be leveraged to reduce the amount of data sent in messages.

2.2.4. Heterogeneous computing

In ParaDIME, we propose to implement different strategies that incorporate heterogeneity at two levels: architecture and device. At the architecture level, one promising way to deal with the dark silicon issue is to use heterogeneous processors, for example with several accelerators, where only a small number of them is powered on simultaneously. In ParaDIME, we will use specialized accelerators such as vector co-processors and heterogeneous CPU cores/processor in the same system such as ARM (big.LITTLE)[1] and NVIDIA's Quadro.[2] We will evaluate the power, performance and energy characteristics of common existing hardware accelerators, such as FPGAs, GPUs or DSPs and heterogeneous CPUs. At the device level, we will introduce heterogeneity to reduce energy consumption and to maintain the CPU performance without any degradation.

2.2.4.1. Architecture-level heterogeneity. In the context of ParaDIME, we will use the Power Estimation Tool at System-level (PETS) [6], for estimating and optimizing power. This tool simplifies application porting as well as enables the user to choose the processor architecture upon which to perform hardware/software co-simulation. PETS was initially developed for the evaluation of MPSoC systems [7–9]. In ParaDIME, we have extended it to model a variety of other systems, including GPUs [10], DSPs, FPGAs [11] and multi-core (dual- and quad- core) ARM processors [12]. Fig. 3 shows our architectural simulator for heterogeneous platforms.

2.2.4.2. Device-level heterogeneity. The main challenge is to reduce power consumption by reducing the supply voltage due to concerns of either reducing performance (due to reduced drive currents) or increasing leakage (when reducing threshold voltage simultaneously). The sub-threshold slope of the transistor is a key factor in influencing the leakage power consumption. In this work, we propose the use of III–V [13] devices that exhibit sub-threshold slopes steeper than the theoretical limit of 60 mV/decade found in CMOS devices. Consequently, III–Vs can provide higher performance than FinFETs [14] based designs at lower voltages. However, at higher voltages, the Ion of FinFETs is much larger than can be accomplished by the flushing mechanism employed in existing III–V devices. This trade-off enables architectural innovations through use of heterogeneous systems that employ both III–V and FinFET based circuit elements. Heterogeneous chip-multiprocessors that incorporate cores with different frequencies, micro-architectural resources and instruction-set architectures are already emerging. In all these works, the energy-performance optimizations are performed by appropriately mapping the application to a preferred core.

2.3. Programming-level

The programmer can influence software helping the overall system to be more energy-efficient. In this project, we focus on the interaction of applications with the hardware as well with the runtime.

The interface to the hardware can be seen as an extension of the programming model (API), which allows the programmer to indicate safe sections for lowering V_{dd}, as well as marking types and methods for calculating and storing values with reduced precision. For efficient message passing, we rely on the actor model.

The interface to the runtime (at data centers) provides distributed communication for enabling energy-efficient allocation and scheduling of resources (= static energy profiles). The definition of static energy profiles are configuration-based indications by the user for the runtime.

For ParaDIME, we decided to use Scala [15], a general-purpose language, that runs on top of the JVM and combines functional and object-oriented programming patterns. Since the release of Scala V2.10,[3] Scala includes the Akka framework, a library with actor model support.

[1] http://www.arm.com/products/processors/technologies/biglittleprocessing.php.

[2] http://www.nvidia.com/object/tegra.html.

[3] http://www.scala-lang.org/news/2013/10/01/release-notes-v2.10.3.html.

Please cite this article in press as: S.K. Rethinagiri et al., ParaDIME: Parallel Distributed Infrastructure for Minimization of Energy for data centers, Microprocess. Microsyst. (2015), http://dx.doi.org/10.1016/j.micpro.2015.06.005

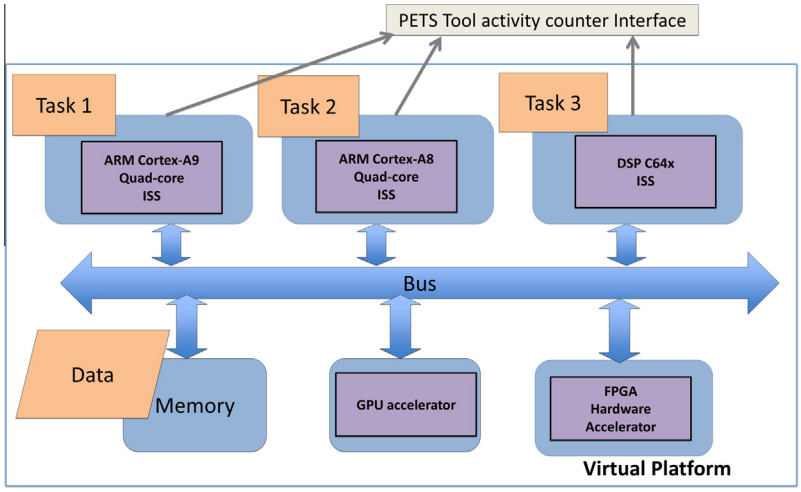


Fig. 3. Heterogeneous architectural simulation environment.

2.3.1. API: general ParaDIME annotation design

In addition to Scalas being compatible with the Java API, it is also possible to write a part of the code using other languages such as C or CUDA and then integrate these into the main Scala application rather easily. We have used this capability to define low-level annotations for indicating data types with reduced precision, and safe regions for operations below safe V_{dd}. For low-level annotations we use Scala annotation macros, which allow us to define annotations and with them linked macros that will be executed at compile time.

We define two general annotations (`@AlterMethod`, `@AlterType`) that can be applied to the definition of types as well as methods and are fairly generic, which allows us to adapt to the provided features in the course of the project. We do not support annotations that target method calls to avoid errors if the programmer forgets to reset the voltage or precision. With the annotations it is possible to add a variable amount of arguments in the form of `key = value` pairs, which can be combined as necessary.

To simplify the usage for the programmer, we define three profiles (nominal, safe and unsafe) for interacting with the hardware related to lowering the V_{dd}. The programmer can choose three types of annotations as specified in the following Listing 1. Although it is possible to define low V_{dd} also for types using the `@AlterType` annotation, it is not recommended to actually use them, since the change of voltage requires several instruction cycles (a compiler warning will be raised in such a case).

An annotation consists of the key-value pair `lowVdd` = ⟨`nominal`, `safe`, `unsafe`⟩. We chose these simple types to be independent from the supplied voltages of the hardware. Note that the nominal value represents the default case and does not have to be necessarily used yet. However, we leave it open to extend the annotations scope away from definition of methods and types to calls depending on the future development of ParaDIME.

Safe means that the hardware chooses reduced voltage, but above safe limits, whereas unsafe lowers the voltage radically such that errors are very likely and might have to be handled and error detection and recovery mechanisms are required. Which error detection and recovery mechanism is appropriate will be chosen by the hardware, the programmer only indicates whether a reliability mechanism is required.

Similarly to lowering the V_{dd}, a programmer can annotate types and methods for using reduced precision. Again, we provide three profiles, the programmer can choose from the following key-value pairs: `precision` = ⟨`standard`, `reduced`, `radical`⟩. The hardware chooses the appropriate values linked to these profiles,

```
@AlterMethod("lowVdd=nominal")
@AlterMethod("lowVdd=safe")
@AlterMethod("lowVdd=unsafe", "lowVddReliability=true")
```

Listing 1. Annotations for lowering V_{dd}.

whereas standard means full precision. As specified by the hardware it is possible to add type annotations to Float and Double definitions. In all cases the hardware will use fewer bits to represent data with reduced precision.

Open questions to consider include (1) the specifics of annotations at method declaration, (2) the effects of calculations with values on reduced precision and (3) the effects of annotations on definition of variables with custom types. The second point might be handled under the hood.

2.3.2. Efficient message passing: actor model and Scala STM extension

We proposed several extensions for improving the efficiency of the actor model that are more or less visible to the programmer. We first introduced concurrent message processing by encapsulating each processing within an actor in a transaction (using Transactional Memory (TM)) [16]. With sequential processing, access to the state will be suboptimal when operations do not conflict (e.g., modifications to disjoint parts of the state, multiple read operations). TM can guarantee safe concurrent access in most of these cases and can handle conflicting situations by aborting and restarting transactions. However, we noticed that in cases of high contention, the performance of parallel processing dropped close to or even below the performance of sequential processing. Hence, we need to reduce the contention. We presented an extension of this work [17], which is in contrast to the other extensions visible to the programmer because we introduce a new method call to Scala STM. We propose a combination of two approaches: (1) relaxing the atomicity and isolation for some read-only operations and (2) determining the optimal number of threads, executing transactional operations dynamically throughout the execution of the algorithm.

2.3.3. Static energy profiles

In this section, we define static energy profiles, which will be extended in the course of the project. As an example, the profiles can be low energy, economy, and high performance (similar to battery usage modes on a PC). If the programmer indicates that the application should run in low energy mode, the ParaDIME

Please cite this article in press as: S.K. Rethinagiri et al., ParaDIME: Parallel Distributed Infrastructure for Minimization of Energy for data centers, Microprocess. Microsyst. (2015), http://dx.doi.org/10.1016/j.micpro.2015.06.005

framework can make decisions on allocating the required resources: e.g., low number of Virtual Machines (VMs) on low-performance CPUs, limited parallelization, low number of actors, etc.

In ParaDIME, we target the static energy profiles towards data-centers, i.e., we can use the energy profiles for managing the allocation of VMs and any runtime-level decision. We plan to use performance counters as well as power estimation to validate energy profiles.

2.4. Runtime-level

2.4.1. Operation below safe Vdd

We want to increase the energy-efficiency at the CPU level by decreasing the CPUs supply voltage. Modern CPUs already incorporate energy-efficiency measures. The processor supports several power states in the real machine. The ACPI standard defines exactly four different power states: C0–C3. Besides power states, the processor may also support different performance states. The number of performance states differs between processors. They are numbered P0, P1, . . . , Pn. Each successively higher state reduces the processor's performance, because the voltage and/or frequency are reduced.

Based on the application annotations the Vdd can be lowered. This is achieved, for example, by annotating non-critical sections. The runtime also provides automatic voltage and frequency scaling and allows for low and near threshold operation. It will provide the current state information to the application.

2.4.2. Energy-efficiency at the data center level

To increase the energy efficiency at the data center, we plan to increase average utilization levels. By pushing utilization levels up, the comparatively high baseline power consumption is compensated for. We plan to combat the previously mentioned drawbacks of high utilization levels by executing a mix of compute tasks on each server. We distinguish between two types of tasks: interactive and batch. Interactive tasks have stringent performance requirements expressed as service level agreements (SLAs).

Batch tasks, on the other hand, have turnaround times 2–3 magnitudes larger than interactive tasks, i.e., hours or days. This flexibility allows us to achieve utilization values of 90% and higher. Each server executes a mix of interactive and batch tasks. We have to ensure that interactive jobs never account for more than, say, 50% of the load. Additional capacity is consumed by batch tasks. Whenever there is a spike in interactive load, batch tasks will yield their resources to the interactive tasks. As soon as the surge in interactive load subsides, the batch tasks will continue executing, occupying all available spare resources. In ParaDIME, we propose energy efficient scheduling decisions for runtime based on the information given to it by the hardware and application.

We also propose mechanism to support for migrating applications between physical servers if it deems this beneficial with respect to energy efficiency and also a mechanism to swiftly reactivate suspended virtual machines.

2.4.3. Energy-proportionality at the data center level

Energy-proportional computing is a concept where the power required by a computing system is directly proportional to the work performed. A standard commodity server is typically not energy-proportional. An energy-proportional server would draw 0 W at 0% utilization. The power drawn would increase linearly with utilization. Even though individual components, here servers, may not be energy proportional, it has been shown that energy-proportionality can be approached at the aggregate level. A server with close to 0% utilization can be switched off, while the work is taken over by the remaining servers. We propose a novel energy-proportional placement decisions of virtual machines and also a mechanism to switch off and which workloads must be moved between servers. Migration decisions follow a cost/benefit analysis.

2.4.4. Carbon-aware scheduling between multiple data centers

Energy efficiency is less important if sufficient cheap and carbon emission-free energy sources are available. Because energy is a growing cost factor for data center operators, reducing the overall consumption in turn reduces the overall operating expenditures. Coupled with penalties for carbon emissions the urge to cut energy consumption is even stronger. If, however, a cheap and green energy source is available, the overall consumption may suddenly be secondary. When data centers have access to alternative energy sources, say solar and coal, the question of where to process a task is then also dependent on where energy is cheap, plentiful, and green. Within ParaDIME, we propose making scheduling decisions across data centers to select the "greenest" data center among those available. The placement decision is based on information about projected energy availability, cost, and heat demand.

2.4.5. Heterogeneous computing

CPUs are general purpose microprocessors. But even they have a multitude of special purpose circuitry to help with, e.g., floating point operations and streaming data manipulation (SSE1/2/3/4). Besides the CPU, there are other components, which take over specialized tasks. The most prominent example is the graphics processing unit (GPU). The GPU is an accelerator for graphics processing. Rendering 3D scenes is a complex task which can, however, be sped up significantly with special-purpose hardware. The idea of accelerators is to implement certain functionality in hardware instead of executing it in software on a general purpose processing unit. By offloading tasks to the accelerator, the CPU is free to do alternative work, or sleep if there is nothing else to do. The accelerator, because it is specialized, will perform the same task more efficiently. The runtime proposes mechanisms to indicate when a task can be sent/offloaded to an accelerator and also a mechanisms to turn accelerators off.

2.4.6. Energy-efficient storage

The energy-efficient storage system is an object store with a simple interface to get, put, update, and delete objects. Objects are binary data blobs as far as the storage system is concerned. Each object is replicated *R* times, where *R* is the replication factor. The replication factor is tunable. It allows different trade-offs for data availability and storage overhead. Besides the minimum replication factor *R*, there exist additional copies of popular objects. These exist solely to cope with increased read requests. Whenever the aggregated client read throughput exceeds the available bandwidth of live replicas, additional copies are brought online. This ensures that the storage system only consumes energy in proportion to the client demands. In ParaDIME, we propose an interface to the application to persistently store data. The interface is linked to the application in the form of a library. The library provides the basic primitives to create, read, update, and delete data objects. To the library each object is an opaque binary string. The storage library provides an additional level of encapsulation: the details of accessing the storage system can be changed without re-writing the dependent applications.

2.5. Communication across the layers

Several tasks developed in ParaDIME involve cooperation between several layers, sharing information using Look-Up-Tables or annotations among others. There are two tasks that put together the device and hardware layers: lowering the

Vdd for the future node and heterogeneous devices. We use a look-up table that characterizes the processor components for a specific technology and Vdd. The values of the table are fed to the simulator at the hardware layer in order to model the future nodes. Between the Hardware and the programming model layers and between the programming model and the runtime layers, we use annotations developed at the programming model level for the future data center node level prototype and for the current data centers respectively in order to implement the lowering of the Vdd, recovery mechanism and fixation of the actor models for the particular core. From the runtime layer to multi-data center layer, the communication is performed by using annotations based workload partitioning and VMs deployment across the geographically distributed data centers. Furthermore, we also work on the compiler (KEMU) based optimization techniques that will enable the data centers to run with better energy efficiency.

3. Benchmarks

We began our selection process by surveying a wide range of algorithms and their available implementations (as benchmarks) with respect to a set of high level, ''nonstarter'' criteria. In other words, we would not further consider any benchmark that did not meet these initial criteria, which can be summarized as follows: (1) The algorithms (implemented as benchmarks) and applications must be parallelizable while at the same time easily implementable in shared memory. (2) The benchmark must have few enough dependencies in order to be able to apply message passing or the actor model. (3) Energy must be a key factor; the benchmark must be deployable on an energy-aware platform that would allow us to clearly measure and demonstrate an energy/performance trade-off. (4) The benchmark allows for approximate data types (necessary for approximate computing). Having selected the initial candidate benchmarks, we next worked with representatives from the other parts of the ParaDIME infrastructure to define a set of additional requirements based on the methodologies that we hoped to implement and test.

3.1. K-means benchmark

The K-means algorithm groups objects in an N-dimensional space into K clusters. This algorithm is not embarrassingly parallel and may benefit from optimistic concurrency. In first year of the ParaDIME, we released several versions of K-means. The first version that we released is our research baseline which is a reimplementation of the sequential K-means application as found in STAMP [18] using Scala. Second, we implemented a multi-threaded shared memory version of K-means. Finally, we released an actor-based implementation to show the performance differences for message passing. Our proposed actor implementation of K-means consists of two types of actors: the (a) (always one) coordinating actor and the (b) (many) worker actors. Worker Actors (WAs) claim responsibility for processing a disjoint chunk of the input dataset and executing the multi-threaded shared memory K-means algorithm. Further, we provide a simple heterogeneous implementation, in which some actors can run on the CPU while others are executed on a GPU. With Akka actors it is easy to separate K-means tasks and to connect the GPU part written in CUDA with the CPU part written in Scala with the help of JNI.

In ParaDIME, we have started to provide support for low-level annotations (Reduced precision computing, Operation below safe Vdd) and their integration into K-means. In this implementation, we have finalized the definition of annotations for the usage in applications. While on a semantic level these annotations and macros have been tested, we will focus on integrating K-means with the simulator focusing on Low Vdd. Furthermore, we will investigate the different trade-off by lowering the Vdd (by altering the frequency of a CPU) on current hardware platform.

3.2. Hydraulic sub-surface simulation (Hydra) application

Multiple-point geostatistics [19] is a prominent tool that has proven effective for performing geostatistical simulations. At its core, the technique analyzes the relationships between multiple variables in several locations at a time. In general, the cost associated with the deterministic determination of the hydraulic properties of the subsurface is prohibitively high. Hence, the aim of multiple-point geostatistical simulation is to simulate the hydraulic properties of the subsurface based on a given number of samples. A simulation consists of a Training Image (TI) and a Simulation Grid (SG). To simulate an unknown point in the SG, the algorithm locates n informed (or known) points in the SG, which are located closest to the unknown point. We then record all the offsets from the unknown point to the known points as well as the values found at these offset locations. Then, for each of the points in the TI, we compute the values located at recorded offset locations and compare them to the values we observed in the SG. Using a domain specific distance criteria, we find the distance between both sets of values. If the distance falls below some predefined threshold, we stop the search and fill the unknown point on the SG using a value from the TI. Hydra is already implemented in CUDA and has a graphical user interface that allows for executing a mixture of batch and interactive tasks. Additionally, the base implementation of Hydra is of the appropriate size and complexity. Finally, it lends itself to possible use in a data center, making it particularly desirable for testing the storage API that will be provided by the runtime. This is particularly important, because none of the other applications to which it was compared were suitable for testing this aspect of the infrastructure.

In ParaDIME, we have released a sequential version of Hydra and started with the implementation of the shared-memory version and the actor-based version. The parallelization of the implementation has two flavors, one parallelizing the SG and one parallelizing the TI. While the parallelization of the TI can be done implicitly, the parallelization of the SG is a bit harder as the number of dependencies is higher. Finally, we ported the TI parallelization version to also run partly on a GPU using CUDA.

4. Preliminary experimental results

In this section, we will present our preliminary results based on the previously proposed methodologies.

4.1. Device-level results

For given lithography assumptions (typically geometrical distances for the patterns that can be effectively printed out on an ASIC), we are now able to produce a usable standard cell technology library. By usable, we mean that the library is compatible with industry EDA flow tools technology library set. For a feasible lithography assumptions for the n10 technology, such technology library has been generated for a fixed temperature but targeting different corners (slow–slow, typical–typical, fast–fast) for a 4 discrete values of the Vdd (0.4, 0.5, 0.6 and 0.7 V). All these technology libraries have been used to characterize a 64-bit adder circuit in terms of: area, delay, leakage and total power. Out of all the data points typical circuit is selected and different performance indicators extracted. Using the distribution information of the delay on the critical path, the probability of the path failure has been

Please cite this article in press as: S.K. Rethinagiri et al., ParaDIME: Parallel Distributed Infrastructure for Minimization of Energy for data centers, Microprocess. Microsyst. (2015), http://dx.doi.org/10.1016/j.micpro.2015.06.005

extracted. The output of this step will now serve for architectural-level simulations. Once these are established, other circuits will be considered to provide a finer grain view.

In order to allow the below safe Vdd assessments, a complex flow has been set-up to span the complete IC integration life cycle starting with lithography assumptions, passing though device modeling, to standard cell generation and RTL circuit level analysis.

The whole flow is divided in 3 consecutive steps:

- ***Technology generation*** enables creation of the technology used as input for circuit synthesis and characterization.
- ***Circuit characterization*** provides a gate-level netlist for a given RTL and set of technologies (different Vdd and process variation corners).
- ***Below Vdd assessment*** produces the probability of the failure for a given critical path in the design.

We will now detail each step.

4.1.1. Step 1 – Technology generation

The automated technology generation framework is finalized (depicted on Fig. 4). For a given lithography assumptions (typically geometrical distances for the patterns that can be effectively printed out on an ASIC), we are now able to produce a usable standard cell technology library. By usable we understand an industry EDA flow tools compatible technology library set, typically exported in the form of.LIB (defines logic functionality timing, power, area and used for gate-level synthesis and place route) and.LEF (defines geometry used for place and route to generate the final circuit layout).

Based on lithography assumptions, we first build transistor device models. Once these models are stable (validated), we can proceed with the design of a reduced set of standard cells. Using advanced interpolation techniques this information is used to generate more exhaustive set of standard cell libraries in both *.LIB* and *.LEF* flavors.

For feasible lithography assumptions and for the n10 technology node, corresponding technology libraries have been generated for a fixed temperature but targeting different process corners (slow–slow, typical–typical, fast–fast) and for a 4 discrete values of the Vdd (0.4, 0.5, 0.6 and 0.7 V). All these technology libraries can be then used to characterize any design in terms of area, delay, leakage and total power, providing that we have a complete set of synthesizable RTL descriptions.

4.1.2. Step 2 – Circuit characterization

For the circuit characterization, we use a traditional design flow composed of usual synthesis, place and route steps. In the context of this work the circuit characteristics will be extracted post synthesis. The gain in accuracy using post placement and route circuit model would not be justified.

The synthesis process is being repeated for a different set of timing constraints supplied to the tool at each synthesis run for all Vdd targets and three process corners. For each of these runs, after synthesis the gate-level netlist is characterized in terms of area, critical path timing and power (total, dynamic, switching, leakage). The whole process is fully automated so that each new design requires very little manual intervention and some CPU time to produce all the necessary data.

For 4 voltages and 3 corners, even relatively simple design would take couple of hours to complete, namely due to numerous timing constraint targets. The synthesis tool assembles gates so that the constraints are met, with minimum total area.

Typical area curve exhibit three zones:

- Flat-least effort, reaches the objective easily.
- Exponential rise-area goes up, tool is using faster cells.
- Flat-Max F is reached, no point in adding extra cells, constraints are not met most likely.

The results that we obtain for a 64-bit adder are shown in Fig. 5 below:

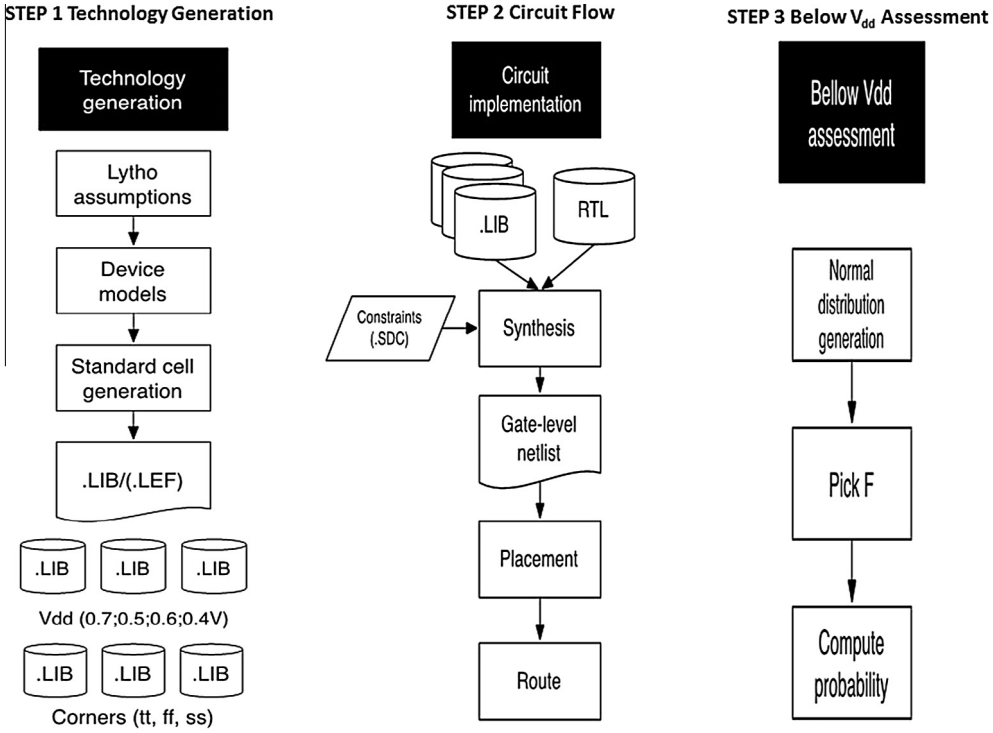


Fig. 4. Device characterization flow.

Please cite this article in press as: S.K. Rethinagiri et al., ParaDIME: Parallel Distributed Infrastructure for Minimization of Energy for data centers, Microprocess. Microsyst. (2015), http://dx.doi.org/10.1016/j.micpro.2015.06.005

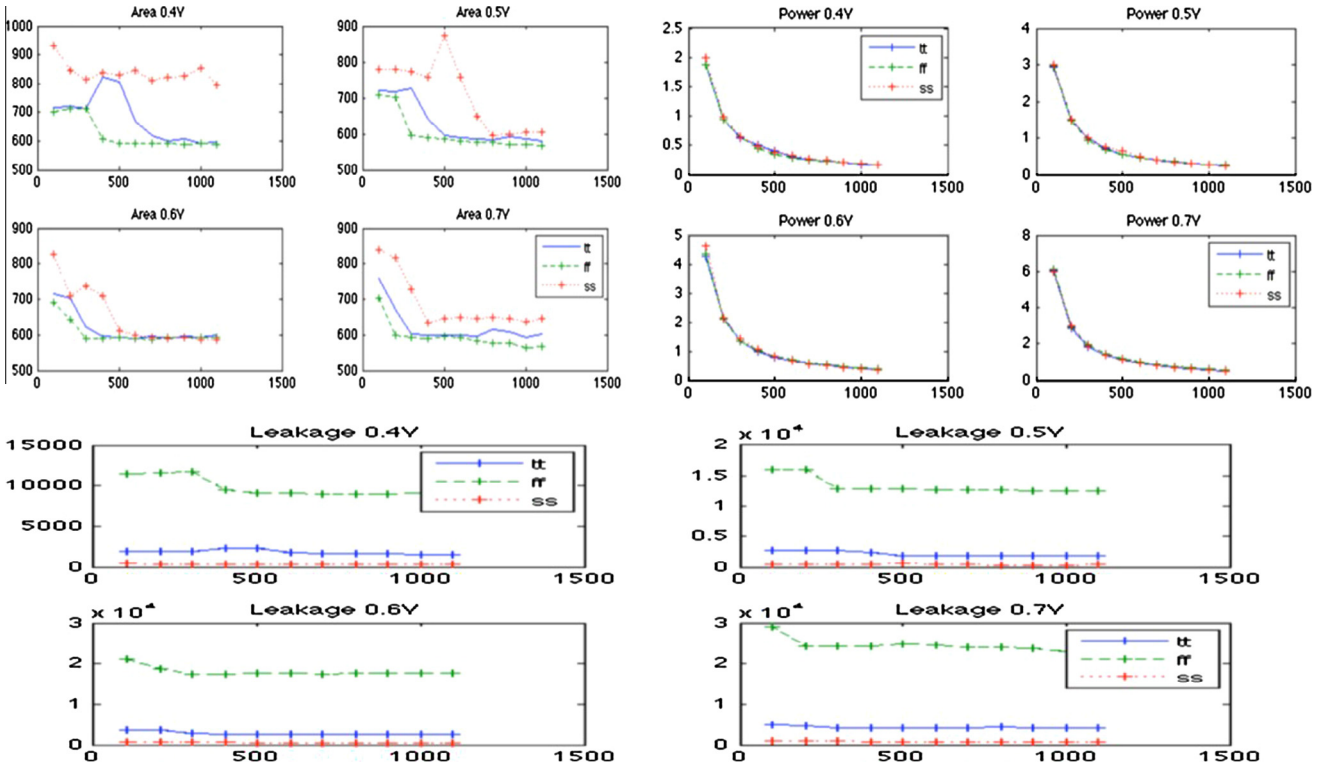


Fig. 5. Device characterization flow.

4.1.3. Step 3 – Below Vdd assessment

In order to generate the critical path failure probability, we take as input the delay information found after gate-level circuit characterization for different corners. Three corners slow–slow, typical–typical, fast–fast define −3 sigma, average and +3 sigma points of a normal distribution curve. For a:

- Given reference point (Max F/Max Vdd) we pick an operating point (F).
- We then look for the intersection of this F with the bell shaped curves for different Vdd.

The whole process of the normal distribution curve generation and the corresponding area (i.e. probability generation) is illustrated in Fig. 6.

4.2. Architectural-level results

4.2.1. Below safe Vdd

In this section, we analyze the feasibility of applying the error detection schemes with TM-based error recovery. We are specifically interested in how much we can lower the voltage while still providing high error detection capability. For the evaluation we consider the following two scenarios: (1) We investigate the energy overhead of the error detection schemes and the combined error detection and recovery and (2) combination of different error detection schemes. Our preliminary results are shown in Figs. 7 and 8.

In Fig. 7, we summarize the performance of all applications in the SPLASH benchmark by averaging their energy consumption. The energy consumption is normalized to the error-free base case in which 2 V supply voltage is used. From this graph (Fig. 7), we can observe that when a transaction consists of 100 instructions, Double Modular Redundancy (DMR) starts to outperform the base-case, when Vdd is 1.4 V (up to 28% reduction) or 1.2 V (up to 54% reduction). Due to the increase in the fault rate, the probability of faults causing rollbacks repeatedly becomes significantly high. Thus, the energy consumption of DMR increases drastically after this voltage level.

There is a trade-off between energy efficiency and reliability, as we can see for DMR and symptom-based error detection and TM recovery. Thus, we can for example combine symptom-based error detection and DMR for consuming less energy, but providing full reliability for critical parts. In Fig. 8, we analyzed the energy overhead of this combination in comparison to the base case and DMR only for a transaction size of 100 instructions. We assume that 30%, 50% or 70% of the application are only secured by symptom-based error detection. With this combination it is possible to lower the Vdd to 1 V (in comparison to 1.2 V with DMR only) and still be more efficient than the base case. Specifically, we reduce the energy consumption by 66% in comparison to the base case.

Aging of the circuit can heavily modify its behavior. In ParaDIME, one of the objectives is defining the feasibility of lowering supply voltage for future devices under the effect of aging. Aging has an important role on voltage scaling since number of timing errors increases when the circuit gets older. A similar aging effect can be observed on NAND flash memories in which when the circuit gets older, the number of programming and retention errors increases. These errors are well researched and modeled in the literature [20]. We have studied these existing models and presented architectural mechanisms to correct programming errors so that we increase the lifetime of NAND flash memories [21]. The Neighbor-Assisted Error Correction (NAC) increases the lifetime of NAND flash memories. In Fig. 9, the extended lifetime due to NAC is divided into three regions based on NAC strength: stage 1, stage 2, and stage 3. To guarantee system reliability, the raw Bit Error Rate (BER) must be less than the acceptable raw BER (i.e., 10^3) of the baseline ECC. Thus, the maximum P/E cycle lifetime of the baseline flash memory without NAC is only 18 k P/E cycles, as shown in Fig. 9. NAC increases the P/E cycle lifetime by 22% (22 k

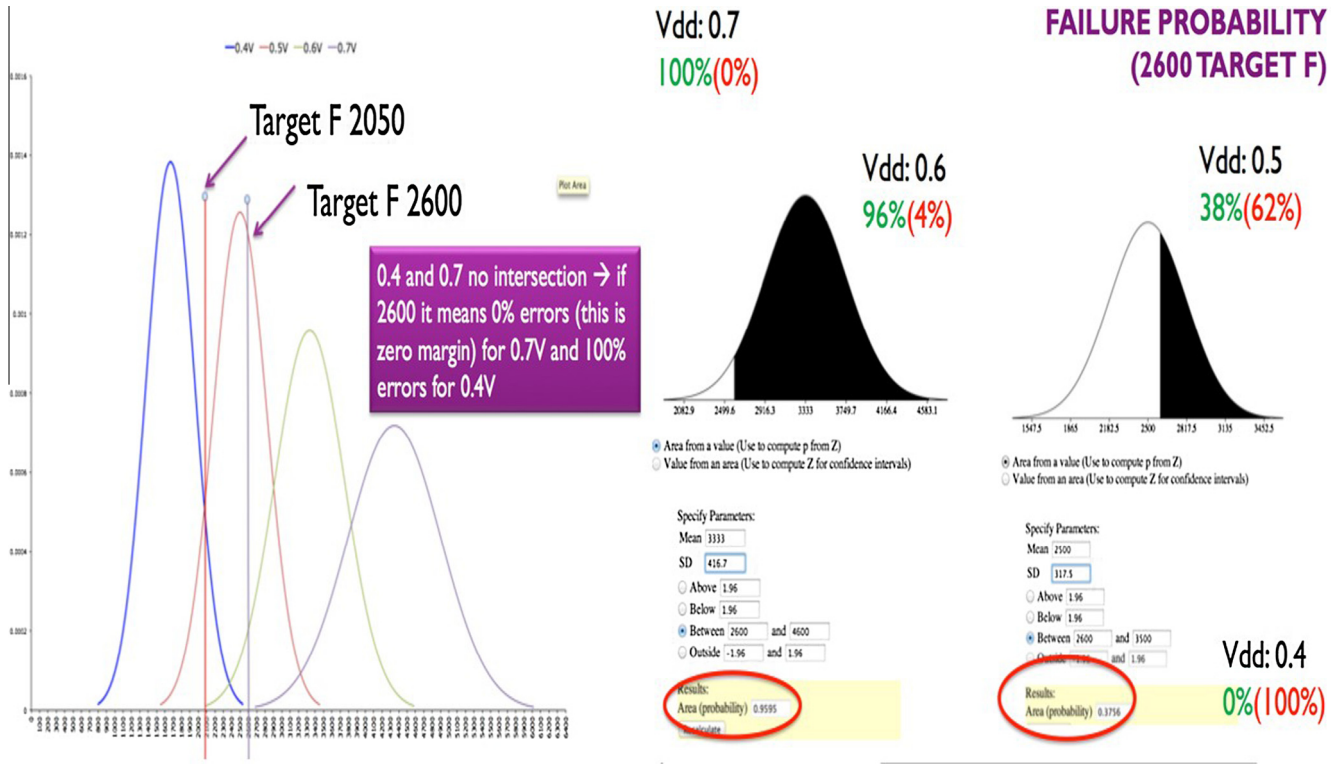


Fig. 6. Failure probabilities.

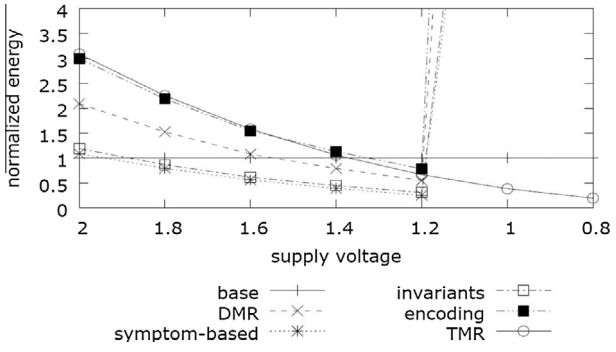


Fig. 7. Energy for transactions with 100 instructions.

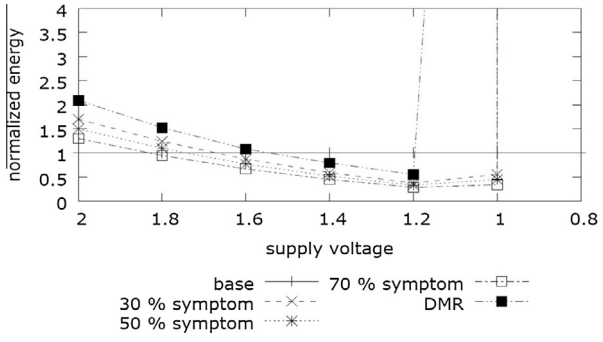


Fig. 8. Combination of different error detection schemes.

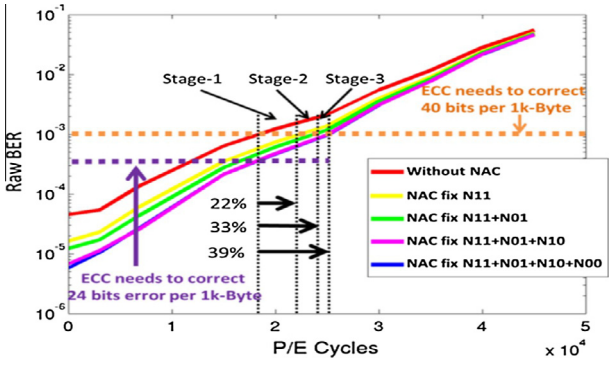


Fig. 9. NAND flash memory aging.

P/E cycles), 33% (24 k P/E cycles) and 39% (25 k P/E cycles) respectively for different strengths. We conclude that NAC is effective in improving flash memory lifetime and as NAC strength is increased lifetime improvement increases.

4.2.2. Multicore and heterogeneous computing at the architectural-level

As we mentioned in Section 2.2.4, we used PETS tool to estimate power for multi-core and heterogeneous processor at the architectural-level. The processor architecture which we use in this project for heterogeneous computing are DSP C64x [22], GPU Tegra3 [10], FPGA Xilinx Zynq 6951910 and Multi-core Cortex-A9 [23]. All the cores/processors execute the same workload. Fig. 10 shows the total energy consumption in mJ for K-means application which is one of the selected benchmarks for this project. In terms of energy consumption, we observe that until a certain number of cores, the total system energy consumption decreases as the number of execution cycles is reduced and then it tends to stabilize as the system performance improves. But increasing the number of processors over a certain limit tends to be futile, as it just adds new conflicts at the bus level, leading to more waiting cycles. When comparing the energy consumed by each of the systems to perform the same task, we observe that FPGA is more efficient compared to the other but programming using HDL is tedious.

Please cite this article in press as: S.K. Rethinagiri et al., ParaDIME: Parallel Distributed Infrastructure for Minimization of Energy for data centers, Microprocess. Microsyst. (2015), http://dx.doi.org/10.1016/j.micpro.2015.06.005

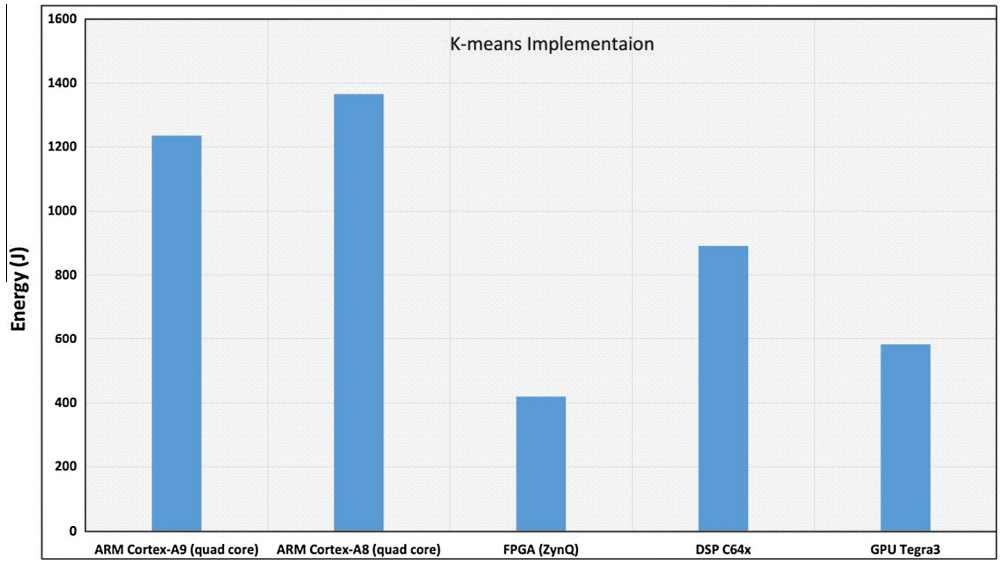


Fig. 10. Energy estimation of Kmeans for different hardware architecture implementation.

GPU is second most energy efficient surpassing DSPs and multicores. If we couple a low power arm with GPU then it will be more energy efficient than have big INTEL or AMD cores.

We have also completed the implementation of the vector processor on an FPGA [24]. We have implemented the open-source VESPA [25] soft vector processor on an Altera board, and we have extended the design to implement a new memory architecture (PVMC, a novel Memory Controller that is optimized for vector architectures). We have vectorized several application kernels and results have shown that our design has higher performance and consumes small time than VESPA and that both vector designs outperform and consume lesser execution time than the baseline scalar processor as shown in Fig. 11.

4.2.3. Reduced precision computing

We evaluated the improvement on performance and power dissipation of the reduced precision solution applied to the floating point operations. The analysis has been made in two phases: first, on the core side, the floating point unit has been modified to operate with reduced precision values, the same has been done with the register file. Then, the solution has been applied to the cache storage, for this analysis we choose to virtually increase the cache size allowing multiple reduced blocks to be stored in the same cache block. System modeled has a single core with out-of-order execution, and two levels of cache of 64 KB and 2 MB respectively. In Fig. 12, we can see the impact in performance of the reduced precision computing for a couple of SPEC2006 floating point applications, cactusADM and zeusmp. We can observe how the solution manages to obtain up to 13% speed-up. The percentage of floating point operations in these applications are 43% (cactusADM) and 35% (zeusmp), and most of these operations are double precision (64 bits), while the reduced precision values use 16 bits. Also, the significant amount of multiplications and divisions in these applications help, as these are the operations where the improvements in performance are higher. In Fig. 13, the reduction in dynamic power consumption of the chip can be observed. As can be seen, most of the dynamic power consumption benefits come from the processor side, specially from the floating point unit, which is an important power consumer in this kind of applications. On the other hand, the cache solution obtains a more humble improvement, although the number of main memory accesses decreases by almost 10%, which also implies lower pressure on the memory bandwidth.

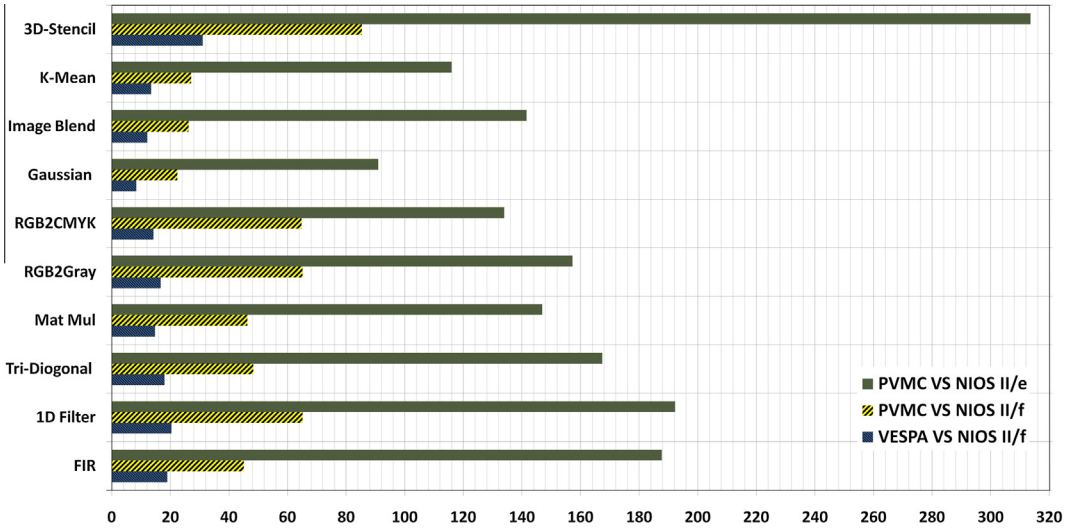


Fig. 11. Speed up in terms of clock cycles comparison between scalar the Nios processors, the VESPA and the proposed vector processor with an PVMC extension.

Please cite this article in press as: S.K. Rethinagiri et al., ParaDIME: Parallel Distributed Infrastructure for Minimization of Energy for data centers, Microprocess. Microsyst. (2015), http://dx.doi.org/10.1016/j.micpro.2015.06.005

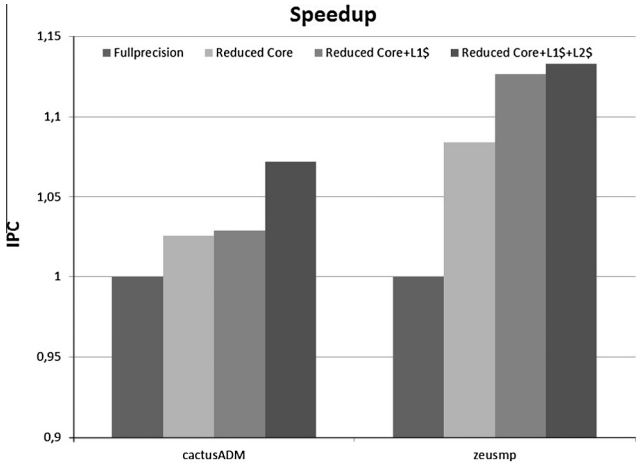


Fig. 12. Reduced precision speed-up results.

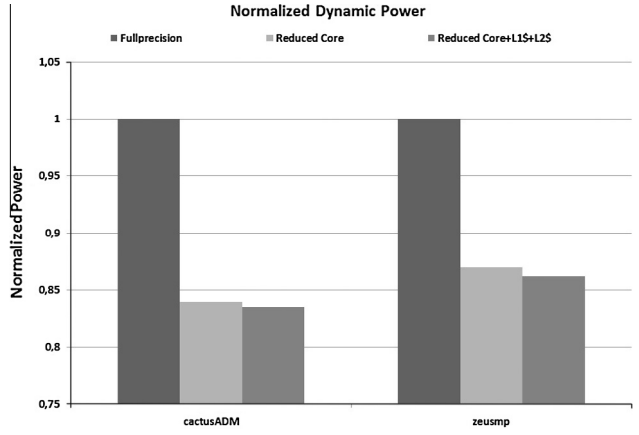


Fig. 13. Reduced precision dynamic power optimization.

4.3. Programming-level results

Experiments in this section were executed on an i3 machine with 2 cores and 4 threads, using 16 clusters. Intel(R) Core(TM) i3-2120 CPU @ 3.30 GHz. We estimate the power consumption on process level using PowerAPI [26].

Considering power estimation, we see in Fig. 14(a) that the sequential implementation requires the lowest power. The reason is that only one of the cores is occupied. The other implementations use all of the cores; this fact is reflected by almost doubled power consumption. However, with parallel systems, we can considerably improve the performance, having impact on energy consumption of the application (energy = power $*$ time). In Fig. 14(a) shows that actors and thread implementations can reduce the execution time, but the execution consumes around the same amount of energy. To actually gain from concurrent executions it is necessary to scale more as shown in Fig. 14(b). For these results, we execute the same experiment with an increasing number of threads on a 48-core AMD Opteron. Starting from similar numbers with 4 threads, it can be seen that the actor implementation scales better than the lock-based implementation. Additionally, from 8 threads the improvement of execution time is high enough to save energy, leading to a final improvement of 90% points over the sequential execution considering execution time.

Another interesting research work is about programming model support for parallelism. Currently, it is done either with control structures or abstractions. Control structures like threads and processes are used to explicitly control parallelism and the access to shared data. Abstractions realized with paradigms such as message passing (e.g., using actors), tasks and dataflow graphs provide implicit parallelism and simplicity to the programmer. Abstractions keep the typical concurrency hazards away from the programmer, who can develop his or her applications in a purely sequential manner.

In classic dataflow programming [27,28], we differentiate between nodes and arcs that are organized within directed graphs. Nodes are state-less sequential functional code-blocks that communicate through input and output arcs with other nodes. The actor model (specifically Akka as part of Scala) can be compared to dataflow programming, as actors are working independently and exchange messages. However, actors are able to keep state although in an isolated fashion. This makes the actor model rather comparable to flow-based programming [29], a particular form of dataflow programming. Flow-based programming also shares principles with the MapReduce [30] model. Here, data is broken into chunks and implicitly processed in parallel in a map phase and later merged in a reduce phase. To show the capability of Akka to express flow-based programming, we implemented the K-means benchmark using the MapReduce programming model. The implementation comprises map-actors that work on their part of the input independently in parallel and forward their results to the reduce actor.

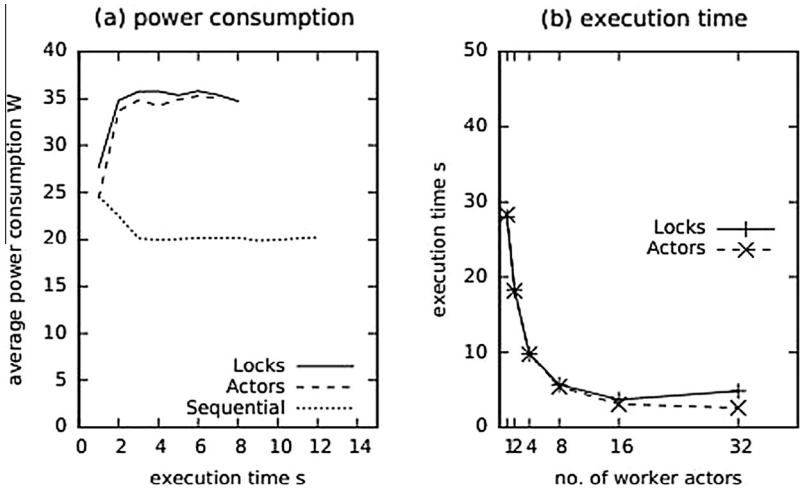


Fig. 14. (a) Power consumption comparison of running on 4 threads and (b) execution time comparison with increasing number of threads.

Please cite this article in press as: S.K. Rethinagiri et al., ParaDIME: Parallel Distributed Infrastructure for Minimization of Energy for data centers, Microprocess. Microsyst. (2015), http://dx.doi.org/10.1016/j.micpro.2015.06.005

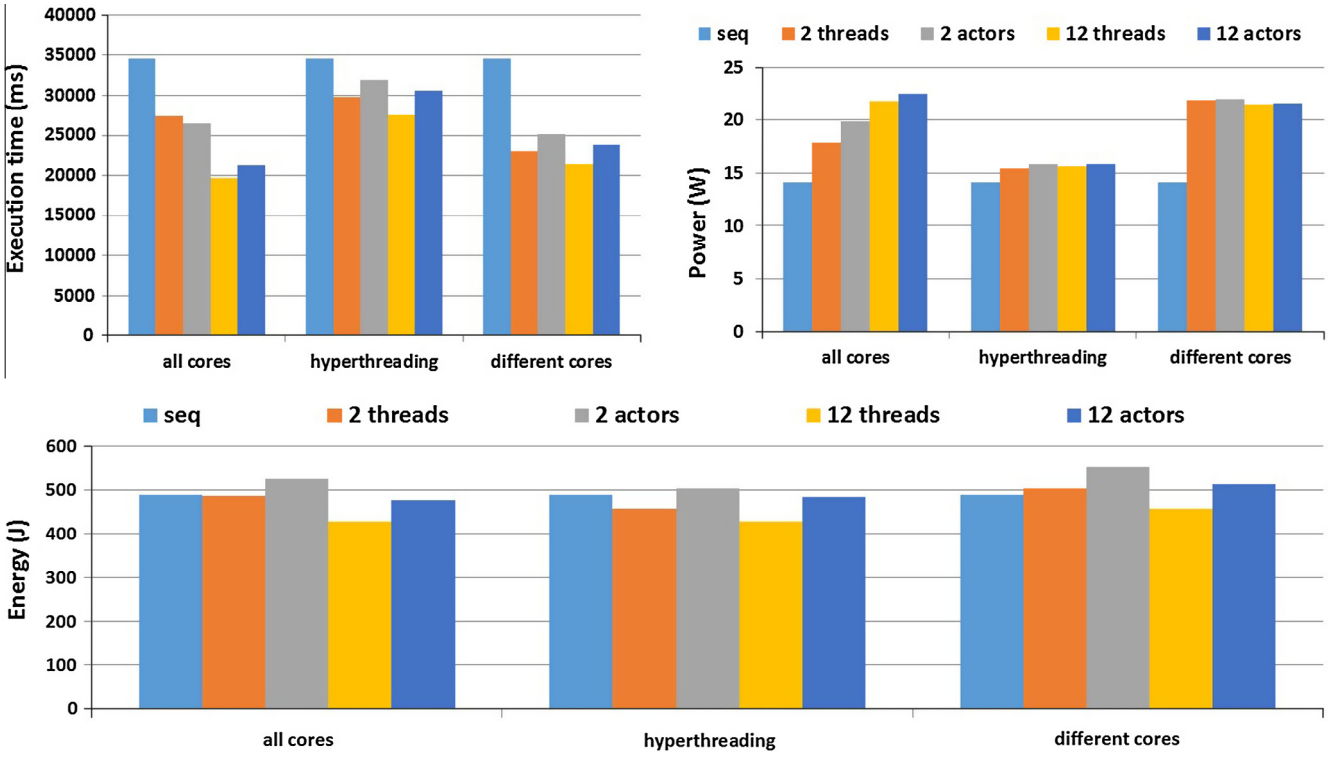


Fig. 15. Power, energy and execution time comparison.

MapReduce usually expects stateless tasks however, when expressing iterative algorithms like K-means, some data might have to be copied multiple times. Hence, the map actors are reused in the next iteration and keep state (the initial input). The reduce actor merges the results and decides whether a new iteration is necessary.

We implemented K-means once using threads and shared memory and once using actors. We compare the execution time and power consumption with the sequential K-means implementation. The experiments are done on an i3-2100 having 3.1 GHz, 2 cores and 4 threads (hyperthreading), 6 GB RAM, using a recent Ubuntu setup and Java 1.7.

We also implemented 2 actors/2 threads to match the number of cores. To investigate scaling and scheduling we increased the number of threads/actors to 12. We further investigated what the impact of core-pinning has on the execution, hence, we used taskset to pin the threads to specific cores. For the power consumption measurement we relied on a power meter (PowerSpy 2.0[4]) that reports power values every second. The energy consumption is then calculated as the average power multiplied by the execution time. As we can see in the following Fig. 15, the sequential implementation requires the least power, but the highest execution time, resulting in rather high energy consumption. In comparison, the multi-threaded implementations require more power, as they use more CPU resources but less execution time. If the performance is increased enough as in the case of 12 threads and 12 actors, the resulting energy consumption is lower than for the sequential execution. In general, however, the actor-based MapReduce performs worse than the shared memory version. The reason is the requirement of immutable messages, leading to overheads. Note that optimizations of the actor model, e.g., by providing mutable access for read-only data would reduce this overhead.

Another point is the scheduling impact. We can see the influence on execution time and power consumption becomes evident depending on which cores the tasks are scheduled. While the typical load balancing is trying to fill all cores equally, for lower power consumption it would be better to fill the cores one after another. However, in the total energy consumption (at least in this case) the results barely differ.

4.4. Data center-level results

4.4.1. Fast virtual machine resume

As outlined in Section 2.4, one aspect of ParaDIME is to increase the energy efficiency of a single data center. In this context, we identified a previously neglected class of applications with only sporadic resource requirements, for example, a web server which only answers a few requests per hour. For this class of infrequently accessed services, it makes sense to *suspend* the services while it is idle and only *resume* it when a new request arrives. While suspending idle services helps the resource provider to reduce its required capacity, it is important to reactivate the service swiftly once more work arrives. We have modified the open-source virtual machine emulator *qemu/kvm*, to resume virtual machines almost instantly. To evaluate our modifications, we performed benchmarks with different applications, storage technologies (HDD vs SSD), and storage locations (direct-attached vs networked). Fig. 16 illustrates the results for resuming a virtual machine from a checkpoint stored on a network-accessible SSD.

While Fig. 16 presents data for three different resume strategies, we focus on the *hybrid* resume variant. Depending on the application, it is possible to resume a virtual machine over the network in 1.0–2.8 s. While some applications, notably Mediawiki, take longer to resume, because they access more memory during the resume, other applications, e.g., Django and Rubis, take less time. This is the worst case delay only experienced on the first request.

[4] http://www.alciom.com/en/products/powerspy2.html.

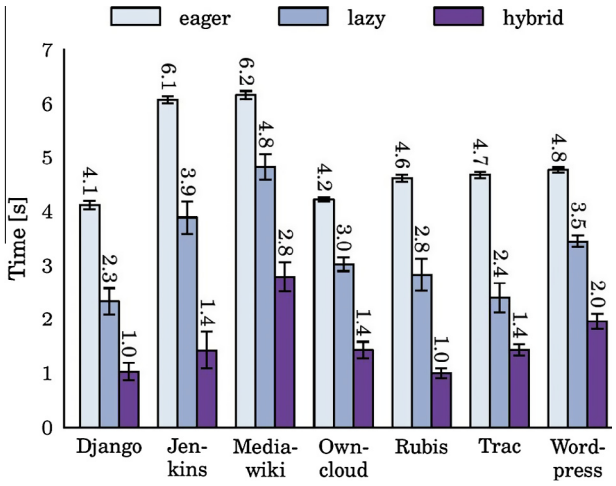


Fig. 16. Virtual machine resume times from remote SSD over a gigabit Ethernet network.

For subsequent requests, when the VM is already running again, will be answered much faster. Further measurements and a more detailed explanation are available in the original publication [31].

4.4.2. Periodic state synchronization

The goal to power off servers is complicated by directly attached storage, because as soon as server is offline, the stored data becomes inaccessible. In this context we developed a system named *dsync* to efficiently synchronize gigabytes of data between two machines. Fig. 17 compares the synchronization time taken for different synchronization methods. We observe that `dsync` is among the fastest methods, while referring the interested reader to the original publication [32] to learn more about the exact differences between the various methods. The wall-clock time taken to synchronize is only one aspect in which the methods differ. Resource consumption, such as disk I/O and computational overhead, are also important metrics to consider in this context.

Furthermore, ParaDIME project targets a geographically distributed infrastructure where tasks, in the form of virtual machines, are migrated between data centers. The excess heat each data center produces is used for secondary purposes, such as generating warm water and heating residential buildings. Hence, the decision when and where to migrate is mainly influenced by external factors such as the demand for more or less excess heat.

Moreover, virtual machine migration typically requires to send multiple gigabytes of data over the network. We minimize the data transfer by keeping a checkpoint of the VM at each data center the VM visits. A migration to a data center with an existing checkpoint must only copy the updates from the migration source to the destination. This reduces the migration traffic to only 40% of the original unoptimised case in a typical scenario.

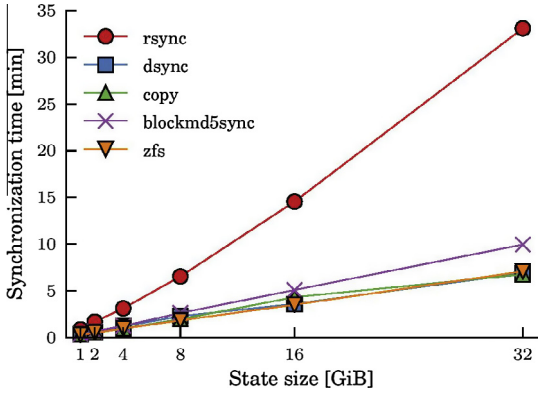


Fig. 17. The time to synchronize grows with the data set size. `rsync` is by far the slowest method while dsync is on par with `copy` and ZFS.

5. Conclusions

The ParaDIME project targets minimization of energy by proposing different distributed methodologies at different abstraction-levels for data centers. The first phase of this project (October 2012–June 2013) "Requirement specifications and concepts" has been successfully completed. This project is currently in the start of the final phase (October 2014–September 2015) "Full specification and Implementation".

From the device-level, we are currently investigating on 14 nm and 7 nm nodes of FinFET and III–V devices and exploring the stacking concept for data centers (2.5D and 3D). We also examine the concept of lowering the V_{DD} and the delay error rates for those device specifications.

At the architectural-level, we have started to implement techniques to improve the efficiency of message passing and lowering the V_{DD} for ARM and Intel processors. We have successfully implemented the heterogeneous part of this project with FPGA, DSP and GPU. Currently, we are exploring the device-level heterogeneity at the architectural-level.

At the programming-level, we have released the K-means implementation of scala code. Now, we are working on programmer friendly annotations by which software developers can mention and implement, in which part of the code, lowering of the V_{DD} and error recovery concept can be introduced.

At the run-time, we are now implementing below safe V_{DD} concept and trying to explore various other methods to promote green computing such as an energy efficient storage facility.

The outcome of this project will serve as a roadmap for future data center processors, will promote green computing and will serve as a example for the programmer to develop energy efficient software for data centers.

After completion of the project, most of the tools and the technologies developed within this project will be released to the research community as open-source.

Acknowledgment

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 No. 318693.

References

[1] H. Esmaeilzadeh, E. Blem, R.St. Amant, K. Sankaralingam, D. Burger, Dark silicon and the end of multicore scaling, in: Proceedings of the 38th Annual International Symposium on Computer Architecture, ISCA '11, ACM, New York, NY, USA, 2011, pp. 365–376. http://dx.doi.org/10.1145/2000064.2000108, URL http://doi.acm.org/10.1145/2000064.2000108.

[2] M. Baron, The single-chip cloud computer, Tech. rep., Microprocessor Report, June 2010.

[3] G. Yalcin, A. Cristal, O. Unsal, A. Sobe, D. Harmanci, P. Felber, A. Voronin, J.-T. Wamhoff, C. Fetzer, Combining error detection and transactional memory for energy-efficient computing below safe operation margins, in: 22nd Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), 2014, 2014, pp. 248–255, http://dx.doi.org/10.1109/PDP.2014.61.

[4] J. Tong, D. Nagle, R. Rutenbar, Reducing power by optimizing the necessary precision/range of floating-point arithmetic, IEEE Trans. Very Large Scale Integ. (VLSI) Syst. 8 (3) (2000) 273–286, http://dx.doi.org/10.1109/92.845894.

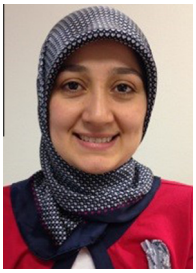
[5] O. Ergin, O. Unsal, X. Vera, A. Gonzalez, Reducing soft errors through operand width aware policies, IEEE Trans. Depend. Sec. Comput. 6 (3) (2009) 217–230, http://dx.doi.org/10.1109/TDSC.2008.18.

[6] S.-K. Rethinagiri, O. Palomar, O. Unsal, A. Cristal, R. Ben-Atitallah, S. Niar, Pets: power and energy estimation tool at system-level, in: 15th International Symposium on Quality Electronic Design (ISQED), 2014, 2014, pp. 535–542, http://dx.doi.org/10.1109/ISQED.2014.6783373.
[7] S. Rethinagiri, R. Ben Atitallah, S. Niar, E. Senn, J. Dekeyser, Fast and accurate hybrid power estimation methodology for embedded systems, in: 2011 Conference on Design and Architectures for Signal and Image Processing (DASIP), 2011, pp. 1–7, http://dx.doi.org/10.1109/DASIP.2011.6136852.
[8] S. Rethinagiri, R. Atitallah, J. Dekeyser, A system level power consumption estimation for mpsoc, in: 2011 International Symposium on System on Chip (SoC), 2011, pp. 56–61, http://dx.doi.org/10.1109/ISSOC.2011.6089692.
[9] S.K. Rethinagiri, R. Ben Atitallah, J.-L. Dekeyser, E. Senn, S. Niar, An efficient power estimation methodology for complex risc processor-based platforms, in: Proceedings of the Great Lakes Symposium on VLSI, GLSVLSI '12, ACM, New York, USA, 2012, pp. 239–244. http://dx.doi.org/10.1145/2206781.2206839, URL http://doi.acm.org/10.1145/2206781.2206839.
[10] S. Rethinagiri, O. Palomar, J. Arias Moreno, G. Yalcin, O. Unsal, A. Cristal, System-level power & energy estimation methodology and optimization techniques for CPU–GPU based mobile platforms, in: 2014 IEEE 12th Symposium on Embedded Systems for Real-time Multimedia (ESTIMedia), 2014, pp. 118–127, http://dx.doi.org/10.1109/ESTIMedia.2014.6962352.
[11] S. Kumar Rethinagiri, O. Palomar, J. Arias Moreno, O. Unsal, A. Cristal, Vppet: Virtual platform power and energy estimation tool for heterogeneous MPSoC based FPGA platforms, in: 24th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS), 2014, 2014, pp. 1–8, http://dx.doi.org/10.1109/PATMOS.2014.6951910.
[12] S.K. Rethinagiri, O. Palomar, R. Ben Atitallah, S. Niar, O. Unsal, A.C. Kestelman, System-level power estimation tool for embedded processor based platforms, in: Proceedings of the 6th Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools, RAPIDO '14, ACM, New York, NY, USA, 2014, pp. 5:1–5:8. http://dx.doi.org/10.1145/2555486.2555491, URL http://doi.acm.org/10.1145/2555486.2555491.
[13] C. Kang, R.-H. Baek, T.-W. Kim, D. Ko, D.-H. Kim, T. Michalak, C. Borst, D. Veksler, G. Bersuker, R. Hill, C. Hobbs, P. Kirsch, Comprehensive layout and process optimization study of Si and III–V technology for sub-7 nm node, in: 2013 IEEE International Electron Devices Meeting (IEDM), 2013, pp. 5.3.1–5.3.4, http://dx.doi.org/10.1109/IEDM.2013.6724566.
[14] P. Magnone, A. Mercha, V. Subramanian, P. Parvais, N. Collaert, M. Dehan, S. Decoutere, G. Groeseneken, J. Benson, T. Merelle, R. Lander, F. Crupi, C. Pace, Matching performance of FinFET devices with fin widths down to 10 nm, IEEE Electron Dev. Lett. 30 (12) (2009) 1374–1376, http://dx.doi.org/10.1109/LED.2009.2034117.
[15] S. Papadimitriou, K. Terzidis, S. Mavroudi, S. Likothanassis, Exploiting java scientific libraries with the Scala language within the ScalaLab environment, IET Softw. 5 (6) (2011) 543–551, http://dx.doi.org/10.1049/iet-sen.2010.0135.
[16] Y. Hayduk, A. Sobe, D. Harmanci, P. Marlier, P. Felber, Speculative concurrent processing with transactional memory in the actor model, in: R. Baldoni, N. Nisse, M. Steen (Eds.), Principles of Distributed Systems, Lecture Notes in Computer Science, vol. 8304, Springer International Publishing, 2013, pp. 160–175. http://dx.doi.org/10.1007/978-3-319-03850-6_12, URL http://dx.doi.org/10.1007/978-3-319-03850-6_12.
[17] Y. Hayduk, A. Sobe, P. Felber, Dynamic concurrent message processing with transactional memory in the actor model, in: 9th Workshop on Transactional Computing (co-located with ASPLOS 2014), ACM SIGPLAN, ACM SIGPLAN, Salt Lake City, Utah, USA, 2014.
[18] C.C. Minh, J. Chung, C. Kozyrakis, K. Olukotun, Stamp: Stanford transactional applications for multi-processing, in: IEEE International Symposium on Workload Characterization, 2008, IISWC 2008, 2008, pp. 35–46, http://dx.doi.org/10.1109/IISWC.2008.4636089.
[19] U.S. Ofterdinger, P. Renard, S. Loew, Hydraulic subsurface measurements and hydrodynamic modelling as indicators for groundwater flow systems in the Rotondo granite, Central Alps (Switzerland), Hydrol. Process. 28 (2) (2014) 255–278, http://dx.doi.org/10.1002/hyp.9568. http://dx.doi.org/10.1002/hyp.9568.
[20] Y. Cai, E. Haratsch, O. Mutlu, K. Mai, Error patterns in MLC NAND flash memory: measurement, characterization, and analysis, in: Design, Automation Test in Europe Conference Exhibition (DATE), 2012, 2012, pp. 521–526, http://dx.doi.org/10.1109/DATE.2012.6176524.
[21] Y. Cai, G. Yalcin, O. Mutlu, E.F. Haratsch, O. Unsal, A. Cristal, K. Mai, Neighbor-cell assisted error correction for MLC NAND flash memories, SIGMETRICS Perform. Eval. Rev. 42 (1) (2014) 491–504, http://dx.doi.org/10.1145/2637364.2591994. http://doi.acm.org/10.1145/2637364.2591994.
[22] S. Kumar Rethinagiri, O. Palomar, J. Arias Moreno, O. Unsal, A. Cristal, M. Biglari-Abhari, System-level power and energy estimation methodology for open multimedia applications platforms, in: IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2014, 2014, pp. 442–449, http://dx.doi.org/10.1109/ISVLSI.2014.38.
[23] S. Kumar Rethinagiri, O. Palomar, A. Cristal, O. Unsal, M. Swift, Dessert: Design space exploration tool based on power and energy at system-level, in: 27th IEEE International System-on-Chip Conference (SOCC), 2014, 2014, pp. 48–53, http://dx.doi.org/10.1109/SOCC.2014.6948898.
[24] T. Hussain, O. Palomar, O. Unsal, A. Cristal, E. Ayguade, M. Valero, PVMC: Programmable vector memory controller, in: IEEE 25th International Conference on Application-Specific Systems, Architectures and Processors (ASAP), 2014, 2014, pp. 240–247, http://dx.doi.org/10.1109/ASAP.2014.6868668.
[25] P. Yiannacouras, J. Steffan, J. Rose, Portable, flexible, and scalable soft vector processors, IEEE Trans. Very Large Scale Integ. (VLSI) Syst. 20 (8) (2012) 1429–1442, http://dx.doi.org/10.1109/TVLSI.2011.2160463.
[26] A. Noureddine, A. Bourdon, R. Rouvoy, L. Seinturier, A preliminary study of the impact of software engineering on greenit, in: First International Workshop on Green and Sustainable Software (GREENS), 2012, 2012, pp. 21–27, http://dx.doi.org/10.1109/GREENS.2012.6224251.
[27] M. Flynn, O. Pell, O. Mencer, Dataflow supercomputing, in: 22nd International Conference on Field Programmable Logic and Applications (FPL), 2012, 2012, pp. 1–3, http://dx.doi.org/10.1109/FPL.2012.6339170.
[28] Arvind D.E. Culler, Annual Review of Computer Science, vol. 1, 1986, Annual Reviews Inc., Palo Alto, CA, USA, 1986, Ch. Dataflow Architectures, pp. 225–253 <http://dl.acm.org/citation.cfm?id=17814.17824>.
[29] W.M. Johnston, J.R.P. Hanna, R.J. Millar, Advances in dataflow programming languages, ACM Comput. Surv. 36 (1) (2004) 1–34, http://dx.doi.org/10.1145/1013208.1013209. http://doi.acm.org/10.1145/1013208.1013209.
[30] J. Dean, S. Ghemawat, Mapreduce: simplified data processing on large clusters, Commun. ACM 51 (1) (2008) 107–113, http://dx.doi.org/10.1145/1327452.1327492. http://doi.acm.org/10.1145/1327452.1327492.
[31] T. Knauth, C. Fetzer, DreamServer: Truly On-Demand Cloud Services, in: International Systems and Storage Conference, ACM, 2014.
[32] T. Knauth, C. Fetzer, dsync: efficient block-wise synchronization of multi-gigabyte binary data, in: Large Installation System Administration Conference, 2013.

Dr. Santhosh Kumar Rethinagiri received his Bachelor of Engineering degree from the Anna University, Chennai, India, in 2006, the Master of Science degree in electrical engineering from KTH, Stockholm, Sweden in 2009 and the Ph.D. in computer science from the INRIA Lille Nord Europe in 2013. He worked with Synopsys for a year in Germany before getting into his PhD. He is currently working as a senior researcher with the Microsoft research group in Barcelona Supercomputing center, Spain. His research interests involve minimization of energy for data centers, power reduction for supercomputers with mobile computing chips and also FPGA based acceleration for data bases.

Dr. Oscar Palomar received his degree on Computer Sciences on 2002 from the Universitat Politècnica de Catalunya and his Ph.D. on Computer Architecture in 2011 from the same university. Since 2010 he has been working on the Barcelona Supercomputing Center in the Computer Architectures for Parallel Paradigms group. His research interests involve low-power vector architectures and energy minimization.

Dr. Anita Sobe received her Ph.D. in Computer Science at Alpen-Adria Universität Klagenfurt, Austria in 2012 with distinction. She was lecturer at the FH Kärnten, Austria as well as senior researcher at the networked and embedded systems group at Alpen-Adria Universität Klagenfurt, Austria. Currently, she is post-doctoral research fellow at the University of Neuchâtel, Switzerland working in the ParaDIME (EU FP 7 FET proactive project) targeting software–hardware solutions for energy-efficient data centers. Her research interests include parallel and distributed systems, energy-efficiency, self-organization in networked systems, virtualization, applications of transactional memory. She has been member of several program committees, including IEEE SASO, IEEE CCNC, WISES, TRANSACT, IEEE PerEnergy. She is a member of IEEE.



Dr. Gülay Yalçin holds BS degree in Computer Engineering from Hacettepe University and MS degree in Computer Engineering from TOBB University of Economics and Technology. Between 2009 and 2014, she pursued his Ph.D. degree in the Computer Architecture department at University Politècnica de Catalunya as a scholar of FI (Research Personal) while she was doing research in reliability at the Barcelona Supercomputing Center.

Thomas Knauth received his diploma in computer science from Technische Universitaet Dresden in 2008. He took a one-year break to work and travel in Australia, and rejoined the group in 2009. In 2014 he received his PhD degree also from TU Dresden.

Rubén Titos-Gil received the MS and Ph.D. degrees in Computer Science from t he University of Murcia, Spain, in 2006 and 2011, respectively. Between 2012 and 2014, he held a postdoc position at Chalmers University of Technology, Sweden. In April 2014, he joined B.Sc. and began working for the ParaDIME project. His research interests lay on the fields of parallel computer architecture and programming models, including synchronization, coherence protocols and memory hierarchy.

Dr. Pablo Prieto received the BS, MS, and Ph.D. degrees from the University of Cantabria, Spain, in 2006 and 2014, respectively, where he teached as assistant at the Department of Computers and Electronics. Now he colaborates at the BSC as senior researcher in the ParaDIME project. His research interests are focused on on-chip cache hierarchies, network on-chip and memory controller design.

Malte Schneegaß received his Diploma of Engineering degree from the University of Applied Science, Dresden, Germany, in 2009. From 2009 until 2011 he worked for a consulting company that is specialized in advising and planning building technologies for offices and museums. His main field was automation of buildings and HVAC devices. Since 2012 he is working for Cloud and Heat Technologies, Dresden as a research & development engineer. His work involves minimization of energy for data centers and the control design of C&H's innovative server rack.

Dr. Adrián Cristal is co-manager of the Computer Architecture for Parallel Paradigms research group at BSC. His interests include high-performance microarchitecture, multi- and many-core chip multiprocessors, transactional memory, and programming models. He received a Ph.D. from the Computer Architecture Department at the Polytechnic University of Catalonia (UPC), Spain, and he has a BS and an MS in computer science from the University of Buenos Aires, Argentina.

Dr. Osman Unsal received the BS, MS and Ph.D. degrees in Electrical and Computer Engineering from Istanbul Technical University (Turkey), Brown University (USA) and University of Massachusetts, Amherst (USA) respectively. Together with Dr. Adrian Cristal, he co-manages the Computer Architecture for Parallel Paradigms research group at BSC. His current research interests include many-core computer architecture, reliability, low-power computing, programming models and transactional memory.

Prof. Pascal Felber received his M.Sc. and Ph.D. degrees in Computer Science from the Swiss Federal Institute of Technology (EPFL). He has then worked at Oracle Corporation and Bell-Labs in the USA, and at Institut EURECOM in France. Since 2004, he is a Professor of Computer Science at the University of Neuchâtel, Switzerland, working in the field of dependable, distributed, and concurrent systems. He has published over 120 research papers in various journals and conferences.

Christof Fetzer received his Ph.D. from UC San Diego (1997). As a student he received a two-year scholarship from the DAAD and won two best student paper awards (SRDS and DSN). He was a finalist of the 1998 Council of Graduate Schools/UMI distinguished dissertation award and won an IEE mather premium in 1999. Dr. Fetzer joined AT&T Labs-Research in August 1999 and was a principal member of technical staff until March 2004. Since April 2004, he is head of the Systems Engineering Chair in the Computer Science Department at the Dresden University of Technology. He is the chair of the Distributed Systems Engineering International Masters Program at the Computer Science Department. Prof. Dr. Fetzer has published over 130 research papers in the field of dependable systems.

Dr. Dragomir Milojevic received his Ph.D. in Electrical Engineering from Université libre de Bruxelles (ULB), Belgium. He joined IMEC in 2004 where he first worked on multi-processor and Network-on-Chip architectures for low-power multimedia systems. Today, part of the INSITE program at IMEC, he is working on methodologies and tools for technology aware design of 3D integrated circuits. Dragomir Milojevic is associate professor at Faculty of Applied Sciences, ULB, where he co-founded Parallel Architectures for Real-Time Systems – PARTS research group. He authored or co-authored more than 50 journal and conference articles, and served as technical program committee member to several conferences in the field.

Please cite this article in press as: S.K. Rethinagiri et al., ParaDIME: Parallel Distributed Infrastructure for Minimization of Energy for data centers, Microprocess. Microsyst. (2015), http://dx.doi.org/10.1016/j.micpro.2015.06.005