HYBRID INFRASTRUCTURE AS A SERVICE
A Cloud-oriented Provisioning Model for Virtual Hosts and Accelerators in Hybrid Computing Environments

Thomas Grosser, Gerald Heim, Wolfgang Rosenstiel
University of Tübingen, Computer Engineering Department, Tübingen, Germany

Utz Bacher, Einar Lueck
IBM Deutschland Research & Development GmbH, Ehningen, Germany

Keywords: Hybrid computing, Hybrid infrastructure, HyIaaS, Mainframe, Accelerators, Logical system modeling.

Abstract: State of the art cloud environments lack a model allowing the cloud provider to leverage heterogeneous resources to most efficiently service workloads. In this paper, we argue that it is mandatory to enable cloud providers to leverage heterogeneous resources, especially accelerators, to optimize the workloads that are hosted in accordance with the business model. For that we propose a hybrid provisioning model that builds on top of and integrates into state of the art model-driven provisioning approaches. We explain how this model can be used henceforth to handle the combinatoric explosion of accelerators, fabrics, and libraries and discuss how example workloads benefit from the exploitation of our model.

1 INTRODUCTION

Large data centers consist of many machines and various architectures. Virtualization is widely used to enable abstraction from hardware and provide isolation of systems and its data, as well as driving utilization of resources to achieve cost-effective data processing. Moreover, special-purpose hardware becomes popular to increase processing efficiency. The cloud paradigm has been accepted by data centers for fine-grained billing of customers. We propose to extend the cloud provisioning model towards heterogeneous collections of resources, which delivers *Hybrid Infrastructure as a Service* (HyIaaS for short). This leverages cost-effectiveness across all involved systems and special-purpose resources. In hybrid infrastructure services, accelerator resources can be shared, thereby increasing utilization of such attachments. Accelerator sharing is abstracted in the infrastructure, so that accelerator software runtime components do not need to be aware of any level of virtualization.

Business data is typically processed by robust, centralized systems, implemented on server farms or mainframes. Especially mainframes come with sophisticated reliability, availability, and serviceability capabilities (RAS). The strengths of such systems are efficient processing of transactional, data-intensive, and I/O-intensive workloads. On the other hand, compute and data-intensive workloads in the field of high performance computing (HPC) demand for additional compute resources. In order to move to hybrid systems, additional management functionality must be incorporated. As an example, the latest Mainframe IBM zEnterprise (IBM, 2010b) is extended with racks full of blades, which can be used as accelerators or appliances. The *Unified Resource Manager* running on the Mainframe provides common management across the system complex. Moreover, the first supercomputer to reach a peak performance of over one petaflops was the supercomputer built by IBM at the Los Alamos National Laboratory (LANL) (Koch, 2008). This setup is also a hybrid system, consisting of two different processor architectures. In this paper, we envision that cloud-oriented hybrid infrastructure provisioning services will become common practice in the future. Thereby, we discuss multi-tenant access and a consolidated programmer's view that allows abstraction from the underlying interconnect in a heterogeneous data center. With upcoming implementations of the proposed model it will be possible to evaluate the feasibility of our approach.

Grosser T., Heim G., Rosenstiel W., Bacher U. and Lueck E..
HYBRID INFRASTRUCTURE AS A SERVICE - A Cloud-oriented Provisioning Model for Virtual Hosts and Accelerators in Hybrid Computing Environments.
DOI: 10.5220/0003391001700175
In *Proceedings of the 1st International Conference on Cloud Computing and Services Science* (CLOSER-2011), pages 170-175
ISBN: 978-989-8425-52-2

2 RELATED WORK

In the course of hybrid system exploration we are working on two applications, which are outlined in this section, and will be used later in this paper to exemplify the application of the proposed model. The first application under investigation implements acceleration of compute-intensive tasks for insurance product calculations (Grosser et al., 2009). Compute-intensive tasks are executed by network-attached IBM POWER blades. Acceleration is achieved through leveraging parallelism and calculating insurance algorithms by just-in-time compiled code snippets on the attached blades. Requirements towards availability are met by tightly controlling execution from the host's off-loading facility.

The second application scenario focuses on numeric and data-intensive calculations of seismic modeling problems. Seismic algorithms are parallelized on various types of accelerators, each having its advantages and disadvantages (Clapp et al., 2010). FPGAs can be exploited efficiently, by implementing pipeline-based stream processing architectures. Besides FPGAs, we also investigate in CPU SIMD processing, threads, MPI, and OpenCL. By using OpenCL we are moreover able to evaluate multiple architectures, like different types of CPUs and GPUs. The focus of this application is to have a broad range of accelerators and programming environments, in order to evaluate different compositions of a hybrid system and its resulting performance.

3 PROBLEM STATEMENT

In the cloud computing anatomy there are three layers to deliver infrastructure, platform, and software as a service, i.e. IaaS, PaaS, and SaaS, respectively (Vaquero et al., 2008). To service infrastructure, physical deployment methods are used to implement the established cloud services. Infrastructure deployment methods are realized using model-based approaches, which enable abstraction from the underlying hardware. Today, no cloud provisioning models exist for offering *multiple* adapted accelerator environments[1]: complexity is imposed on the administrator to build workload optimized solutions based on resources of various types and architectures. A composite system built from host resources, connectivity, and accelerator resources needs to be provisioned as a single entity in order to establish hybrid infrastructure. Abstraction of the physics allows for both accelerator sharing and flexibility of the cloud service provider. Isolation needs to be ensured to create a multi-tenancy capable landscape.

Table 1 lists the manifoldness of an arbitrary heterogeneous data center. The table allows to create *any* combination of host, accelerator, fabric, including the desired programming platform.

Table 1: Each possible combination imposes challenges to handle hosts and interrelationships to multiple accelerators. In addition, the used library must be adapted to the underlying system and interconnect.

Host	Accelerator	Fabric	Library
i386	Cell/B.E.	Ethernet	OpenCL
x86_64	GPGPU	InfiniBand	MPI
s390	Cluster	PCIe	DaCS/ALF
POWER	FPGA/DRP	Myrinet	SOAP
Itanium	DataPower	Quadrics	CUDA

Manufacturers develop all kinds of multi-core and many-core accelerators, each having its targeted application domain. This includes, among others, multi-core CPUs, GPGPUs, Cell/B.E., and FPGAs. Arising standards for heterogeneous computing, like OpenCL, enable to use GPGPUs for number crunching. An OpenCL implementation is provided, among others, for Nvidia GPUs (NVIDIA, 2010), Cell/B.E., and POWER (IBM, 2010a). OpenCL enables easy parallelization over a device's compute units, exploiting data-parallel computing power. In order to scale an accelerated application, multiple devices must be incorporated, thus additional management functionality is needed in the host code to queue operations efficiently. As another example, some application domains fit efficiently onto FPGAs, eventually forming appliances. Multiple access to these resources must also be shared. Regarding the fabric, in HPC, InfiniBand may be preferred over Ethernet. Traditional Ethernet networking allows to send messages, while InfiniBand channel adapters also provide RDMA capabilities. Also, Ethernet adapters may be equipped with RDMA capabilities to reduce communication overhead, though. Unlike networking, PCIe is an attachment approach where devices are accessible through the memory-mapped PCI address space. Thus, the use of PCIe-attached accelerators is different from network-attached ones. From an infrastructure point of view, there is a variety of available hosts, available accelerators, fabrics, programming environments, frameworks, and libraries. In the following sections, we introduce the model and discuss how example workloads benefit from hybrid infrastructure.

[1] Meanwhile, Amazon EC2 offers Nvidia-based GPU compute racks (http://aws.amazon.com/ec2/hpc-applications/).

4 HYBRID PROVISIONING MODEL

To achieve abstraction from the underlying heterogeneous data center, an object-relationship model is used to define a logical system structure. By using this, host systems, compute nodes, and accelerators are specified, including additional information on their interrelationship. That is, in the *logical view* the system is modeled consisting of hosts, accelerators, and the connection. This enables to model connectivity requirements between two nodes. Using this representation, it is possible to create arbitrary logical configurations of a hybrid system.

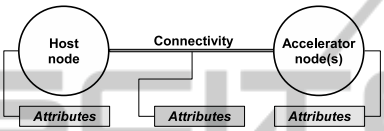

Figure 1: The logical system structure is built using an object-relationship model.

The object-relationship model in figure 1 enables cloud providers to implement the attributes according to their particular business model. A cloud provider with a very simple and homogeneous data center could for example focus on a model that allows a customer to specify attributes like number of CPUs, memory, number of GPGPUs and required throughput, regarding the interconnect. In contrast, another cloud provider with a more advanced heterogeneous data center allows the customer to specify the host architecture including a specific accelerator architecture, thereby offering various attributes. Attributes are used to operate on multiple levels, i.e. to reflect physical attributes, business goals, deployment constraints, and performance requirements.

The use of a model as abstraction allows transformations to generate physical deployment configurations. In this section, we study model-based approaches that are used to deploy physical systems using automated processes. Based on this, we propose a model flow to handle hybrid systems. To service *Hybrid Infrastructure as a Service*, we claim that accelerator resources are connected to the host system and appear as fabric-attached devices in the infrastructure system. The idea is to provide a single reliable node instead of multiple individual nodes. To accommodate multiple users, an accelerator sharing model is discussed which allows to handle different types of accelerators. According to our hybrid application scenarios, considerations regarding the runtime management are introduced as well.

4.1 `libvirt` on Linux

The model proposed in this section is intended to be integrated as part of `libvirt` (Coulson et al., 2010) on Linux-based systems in the future. We aim to use `libvirt` as part of our hybrid management prototype, as it allows to manage virtual machines, virtual networks, and storage. Adding extensions to `libvirt` may enable creation, attachment, and operation support of accelerators, forming hybrid infrastructure.

4.2 Prerequisites

Physical deployment is an automated process to service connectivity, security, and performance. Model-based approaches are used to service physical topology designs according to transformation rules and constraints. There are several approaches that can be adapted to setup infrastructure in constrained environments (Eilam et al., 2004). The underlying object-relationship model shown in figure 1 is used to represent logical resources including the components' relationships. The flow depicted in figure 2 represents the transformation process to a physical deployment. The user provides a *Logical System Model* that is transformed to a *Physical Deployment Model* by using an *Physical System Model*, which represents hardware available in the data center. At each stage of the transformation the model incorporates attributes representing business goals and configuration dependencies.

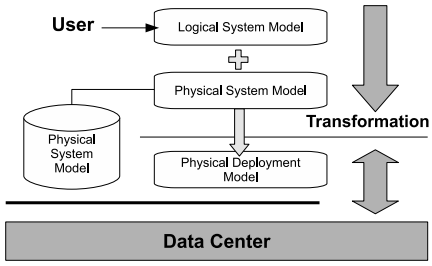

Figure 2: The physical deployment includes VM creation, network deployment, etc.

Using model-based approaches it is also possible to define platform services and applications by separating business logic and IT deployment. Transformation rules are then used to create platform-specific models from platform-independent models. Targeted applications are service-oriented architectures with Web service interfaces (Koehler et al., 2003). Moreover, frameworks for enhanced autonomic management of distributed applications are enabled using similar techniques (Dearle et al., 2004). The optimal use of hardware in large heterogeneous data cen-

ters leads to distributed service management (Adams et al., 2008). The proposed architecture uses detailed knowledge about available resources including resource interrelationships to achieve sharing. To implement a refinement process, the deployment is gradually improved as the system communicates the current status of the physical resources back into the logical model to adjust further deployment configurations. As in other model-based approaches, rules and constraints are used during transformation which are denoted as attributes in figure 1.

As model-based approaches are used to implement various services, such models can also be defined to provision cloud-oriented infrastructure. The hardware of large heterogeneous data centers can be shared to multiple users, thus improving overall utilization of few specialized hardware. In order to further move from heterogeneous to managed hybrid infrastructure deployment, it is crucial to define models specific to accelerator management. This includes an accelerator runtime, which can be serviced as part of the hybrid infrastructure. The runtime is part of a software stack to query, allocate, and operate accelerator engines. Out-of-band monitoring may be used by the cloud service provider to obtain capacity and utilization data with respect to all resources in the hybrid environment. According to user-specified goals, the deployment of shared accelerator resources is configured autonomously.

4.3 Proposed Hybrid Model Flow

In contrast to figure 2, the *HyIaaS* deployment flow uses additional models and methods (shaded gray), as depicted in figure 3. The transformation requires an additional *Hybrid Connectivity Model* in order to reflect available fabrics in the data center. Attributes include bandwidth and latency allowing implementation of Quality of Service. Besides the physical deployment, a *Hybrid Setup Routine* is created, which initializes and configures involved resources, i.e. creates the host node and enables connectivity to designated accelerators. As soon as the fabric is shared between accelerator instances, isolation of the logical connectivity is included (e.g. via virtual LANs) to avoid interference of other host or accelerator nodes.

Another essential part of our flow is the *Resource Interrelationship Model*, which can define limitations and constraints used to set up the runtime environment or select connectivity and acceleration resources by meeting performance requirements. In the remainder of this section, the *Accelerator Sharing Model* and *Accelerator Runtime Environment* will be described more detailed.

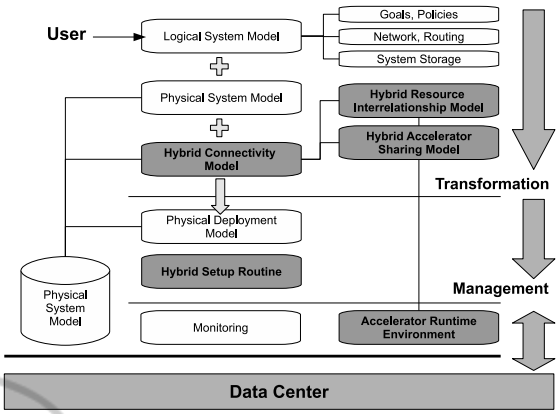

Figure 3: *Hybrid Infrastructure as a Service* flow.

4.4 Accelerator Sharing Model

While virtualization can improve overall resource utilization, the variety of accelerator architectures imposes several challenges on sharing and multi-tenancy. Figure 4 depicts three accelerators, each offering hardware resource abstraction at different levels. Virtualization offers entire virtual sets of accelerator hardware, each instance running the entire accelerator-resident software service stack. An example for this level of virtualization is a Linux-based accelerator framework running on general purpose CPUs virtualized by a hypervisor. Not all sorts of accelerators exhibit such mature virtualization capabilities. Therefore accelerator sharing can also be achieved on other levels by means of multi-tenancy in the accelerator-resident software service stack.

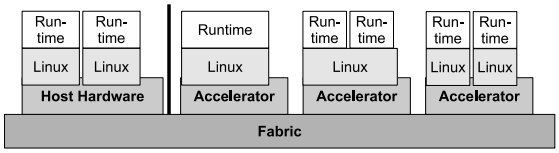

Figure 4: Sharing of accelerators at different level.

In correlation to figure 4 the schemes for accelerator abstraction and sharing include:

- Fully virtualized, using a hypervisor
- Time sharing, scheduling
- Queuing, with ordering and blocking
- Exclusive use of the accelerator

The sharing technique imposes implications to efficient data handling. This includes data and cache locality, scheduling, and connectivity to achieve adequate performance and utilization. Multi-tenancy of the accelerator service requires proper abstraction, which allows for isolation between different con-

sumers of virtual accelerators. Selection of the sharing technique can be performed by the transformation process of the HyIaaS flow. It may be guided by performance requirements as stated in the logical system model defined by the user. Selection of the accelerator architecture can be made an attribute of the logical system model, or can be determined as part of the transformation process, again guided by definitions of the user.

4.5 Accelerator Runtime Environment

Figure 5 shows the individual layers of the HyIaaS model, with the accelerator runtime as top layer. The accelerator runtime environment can be provided by the cloud service provider as part of a pre-deployed image. It depends on the architecture and attachment type of the accelerator and offers a well-defined set of interfaces. Each combination of accelerator, fabric and API may result in a different implementation of the accelerator runtime environment, and some combinations may not exist at all. The API offered by the accelerator runtime also defines the interface that applications have to use to consume accelerator resources, and is therefore specified as part of the logical system model. In most cases, the user will start out with a desired API and derive lower levels of the system model from that. Use of future software as well as applicability to future accelerator architectures suggest the use of de-facto standards like OpenCL, rather than vendor-proprietary programming paradigms. Quality of Service improvements like redundant execution, job retry, and accelerator failover, can be implemented efficiently in the runtime environment, if not offered by the underlying fabric and accelerator infrastructure.

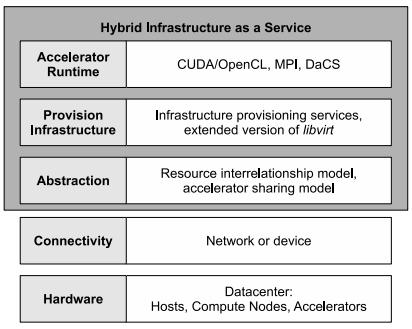

Figure 5: Essential functionality for the *Hybrid Infrastructure as a Service* model.

5 APPLICATION OF THE MODEL

In this section, we outline how the applications stated in section 2 can be represented using the proposed cloud-oriented hybrid infrastructure model.

5.1 Insurance Calculations

The insurance application is used for both batch and interactive processing. While acceleration of batch processing results in shorter batch windows and benefits from higher batch throughput in off-shift hours, the interactive processing of insurance calculations has requirements to handle workload peaks and must be able to return results with low latency. In both cases, hybrid infrastructure can enable workload optimized processing and improves the capacity of the overall solution. In the logical system model for this application, the host system and multiple CPUs as accelerators, connected through an IP-capable fabric, are defined. To scale out according to workloads, the logical system defines a $1 : n$ correlation of host and accelerators. In order to guarantee low-latency results, the network topology must be adapted to avoid bottlenecks. As this application is run on a mainframe, the acceleration has to incorporate RAS capabilities which are specified in the logical system and result in deployment of an appropriate accelerator runtime environment. In this case, the accelerator communicates with a host-side session manager to dispatch individual tasks.

5.2 Seismic Processing

The seismic application exhibits more degrees of freedom for selecting accelerators. The host application itself started out as cluster application, working on large amounts of seismic data. In order to achieve maximum compute throughput, the workload is distributed to several nodes, which also applies to the hybrid environment. The targeted accelerator can be a combination of the following: a cluster of homogeneous nodes, a cluster of OpenCL-enabled nodes (CPUs and GPUs), or a cluster of specialized FPGA accelerators. Besides FPGAs and GPGPUs (Clapp et al., 2010), the Cell/B.E. processor may also be considered (Perrone, 2009). For maximum bandwidth, InfiniBand may be favored over Ethernet, which is expressed by performance attributes in the logical system structure. In any of these combinations, the transformation has to find a suitable network topology by evaluating the resource interrelationship model, attach appropriate accelerators, and establish the proper runtime environment. So, hybrid infrastructure has to support selecting the right runtime with regard to the specified accelerator, fabric, and API. Assuming OpenCL is used as acceleration method, the run-

time may be extended to transparently switch to other OpenCL-enabled nodes.

6 CURRENT AND FUTURE WORK

In the course of our current work, we started to implement the proposed model on Linux-based systems using `kvm` as hypervisor and `libvirt` as infrastructure provisioning engine. By using `libvirt`, the system description is provided as XML. To implement the flow of our model, hybrid infrastructure XML descriptions have to be defined and integrated into `libvirt`. In the current work, we ported libvirt to `s390` in order to enable the basic flow as depicted in figure 2. Using the application scenarios described in the previous sections, we address the proposed flow shown in figure 3. That way, the provided hybrid infrastructure can, for example, be modified or extended by accelerator elements as well as networking or device structures.

7 CONCLUSIONS

In this paper we propose a model to enable cloud provisioning of accelerator resources in heterogeneous data centers. This will allow cloud users to consume composite systems equipped with accelerator resources, instead of having to deal with a multiplicity of nodes and orchestration thereof. According to our proposal, the user specifies his requirements via a logical system structure, consisting of nodes, accelerators, and fabrics. The model flow allows producing a transformation engine, which maps the logical system structure to the actual hardware of the data center, fulfilling the user's requirements specified as model attributes. We introduce sharing of accelerators and multi-tenancy, which requires appropriate isolation at the accelerator entity as well as in the fabric. A runtime software stack allows for using the accelerator from the host, thereby abstracting from the environment's actual implementation. Such use of accelerator resources attached to host nodes results in a cloud paradigm to combine the advantages of workload optimized systems with the data center's commercial model: applications can improve compute efficiency through the use of accelerators, and the cloud provider can achieve high utilization and offer a cost-competitive environment.

REFERENCES

Adams, R., Rivaldo, R., Germoglio, G., Santos, F., Chen, Y., and Milojicic, D. S. (2008). Improving distributed service management using Service Modeling Language (SML). *Salvador, Bahia. IEEE*.

Clapp, R. G., Fu, H., and Lindtjorn, O. (2010). Selecting the right hardware for reverse time migration (in High-performance computing). In *Leading Edge*, pages 48–58, Tulsa, OK.

Coulson, D., Berrange, D., Veillard, D., Lalancette, C., Stump, L., and Jorm, D. (2010). libvirt 0.7.5: *Application Development Guide*. URL: http://libvirt.org/guide/pdf/.

Dearle, A., Kirby, G. N., and McCarthy, A. J. (2004). A Framework for Constraint-Based Deployment and Autonomic Management of Distributed Applications. *Autonomic Computing, International Conference on*.

Eilam, T., Kalantar, M., Konstantinou, E., Pacifici, G., Eilam, T., Kalantar, M., Konstantinou, E., and Pacifici, G. (2004). Model-Based Automation of Service Deployment in a Constrained Environment. *Research report, IBM*.

Grosser, T., Schmid, A. C., Deuling, M., Nguyen, H.-N., and Rosenstiel, W. (2009). Off-loading Compute Intensive Tasks for Insurance Products Using a Just-in-Time Compiler on a Hybrid System. In *CASCON '09*, pages 231–246, New York, NY, USA. ACM.

IBM (2010a). OpenCL Development Kit for Linux on Power. URL: http://www.alphaworks.ibm.com/tech/opencl.

IBM (2010b). The IBM zEnterprise System – A new dimension in computing URL: http://www-01.ibm.com/common/ssi/rep_ca/9/877/ENUSZG10-0249/ENUSZG10-0249.PDF.

Koch, K. (2008). Roadrunner Platform Overview. URL: http://www.lanl.gov/orgs/hpc/roadrunner/pdfs/Koch - Roadrunner Overview/RR Seminar - System Overview .pdf.

Koehler, J., Hauser, R., Kapoor, S., Wu, F. Y., and Kumaran, S. (2003). A Model-Driven Transformation Method. In *EDOC '03*, Washington, DC, USA. *IEEE Computer Society*.

NVIDIA (2010). Developer Zone – OpenCL. URL: http://developer.nvidia.com/object/opencl.html.

Perrone, M. (2009). Finding Oil with Cells: Seismic Imaging Using a Cluster of Cell Processors. URL: https://www.sharcnet.ca/my/documents/show/44.

Vaquero, L. M., Rodero-Merino, L., Caceres, J., and Lindner, M. (2008). A break in the clouds: towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, 39.