Proceedings of the 2011 Winter Simulation Conference
S. Jain, R.R. Creasey, J. Himmelspach, K.P. White, and M. Fu, eds.

MODULAR PERFORMANCE SIMULATIONS OF CLOUDS

Peter Altevogt

IBM Germany Research & Development GmbH
Schoenaicher Str. 220
D-71032 Boeblingen, GERMANY

Tibor Kiss

Gamax Kft
Bartok Bela ut 15/D
1114 Budapest, HUNGARY

Wolfgang Denzel

IBM Research GmbH, Zurich Research Laboratory
Saeumerstr. 4
8803 Rueschlikon, SWITZERLAND

ABSTRACT

Performance and scalability are essential non-functional features of contemporary cloud solutions. Performance modeling and simulation techniques provide the tools required for cloud capacity planning and design. In this publication we describe a modular approach to simulate the hardware and software components of clouds. This approach supports the rapid construction of new cloud models by combining available simple or compound simulation modules, adding new cloud modules when required and adapting the implementation of existing ones if necessary. Key design points are a careful separation between hardware (infrastructure) modules and modules representing software workflows as well as the introduction of a system of a hierarchical request execution phases separating the simulation of high-level cloud workflows from the simulation of workflows at hardware component level.

1 INTRODUCTION

Cloud computing is a game changing technology to provide respectively to consume data center resources, see e.g. (Cloud Computing 2011) and references therein. Because cloud customers expect good and consistent response times to their requests irrespective of the workload already posted against the cloud, performance and scalability are essential here. Typical performance metrics here are e.g. the "number of concurrent users" that a cloud can support respecting certain response time constraints or throughput and response times of provisioning images (e.g. instances of operating systems) as a function of the number of concurrent provisioning requests in flight.

Performance and scalability of clouds handling such workloads significantly depend on the infrastructure available (in terms of physical server, networking and storage), as well as on the software heuristics to manage this infrastructure, cloud users, approval processes and reservation and provisioning of resources.

To provide a balanced, workload optimized design of clouds, cloud architects frequently have to rely on a rather incomplete set of measurements available only for small and medium sized clouds and then try to extrapolate to larger cloud infrastructures and workloads. Obviously, this is the traditional domain for performance modeling, either for analytic modeling (Bloch, Greiner, de Meer and Trivedi 2006; Kleinrock 1975; Kleinrock 1976) or for a simulation based modeling approach (Law and Kelton 2000; Banks, Carson II, Nelson and Nicol 2005). Although these technologies are well established e.g. in designing new microprocessors (Pasricha and Dutt 2008) or the performance analysis of networks (Bertsekas and

Gallager 1992) they need to be adapted to meet the special requirements of simulating clouds as described below:

- The hardware infrastructure of clouds consists of servers, networking switches and storage subsystems and all of these components need to be taken into account on an equal footing. This is in contrast to most of the performance simulation work focusing on (parts of) just one of these components.
- A cloud being a complex system with intricate interactions between hardware and software modules, we need to treat software workflows on an equal footing with the hardware infrastructure in simulating end-to-end performance.
- In general the software heuristics for managing and using a cloud change at a much higher rate than the available cloud hardware infrastructures; therefore it is important to introduce separate modules for simulating software heuristics and the hardware infrastructure to support a rapid implementation of new cloud software heuristics for an unchanged hardware infrastructure and vice versa.
- The market for cloud solutions being highly dynamic, we need to provide simulations of new clouds in a timely manner, i.e. we need to support a rapid prototyping here.
- We need to allow for selectively and rapidly adding details to the simulation of specific hardware or software components to increase the authoritativeness of the simulation if required by the stake holders of a simulation effort.

In this publication we will describe a framework for performance simulations of clouds addressing the above mentioned requirements. The key features are hierarchies of:

- simulation modules representing hardware infrastructure components as well as software workflows. These modules communicate with each other using messages using well defined interfaces (ports respectively gates). The software modules interact with the hardware modules requesting resources (e.g. processor cycles or bandwidth) from them.
- phases describing the execution of request workflows at different levels, e.g. at the cloud and the component level.

Simulation models of clouds can then in many cases be built by simply combining available simulation modules ("Lego bricks") and only modifying some associated parameter files .

As usual in performance modeling, a key point is to find an appropriate abstraction level of the system and workload under investigation. This level is largely determined by the goals of the modeling effort. In our case a main focus is to simulate end-to-end scalability of response times and throughput of various provisioning heuristics, administrative tasks and application workloads on different cloud infrastructures. The abstraction level as indicated in this publication may need to be adapted for other simulation goals. In this publication we will focus more on the software architecture of the simulation framework, but also provide details on the current modeling abstractions used.

The rest of the publication consists of sections describing the simulation of the cloud components (software and hardware), the simulation of the cloud workloads and finally providing details on workload generation for clouds.

The current implementation of the framework uses the OMNEST Simulation Framework (OMNEST 2011; Varga and Hornig 2008), which nicely supports building modular simulations.

2 SIMULATION OF CLOUD COMPONENTS

In this section we describe our approach to simulate the various hardware and software components of a cloud. Although clouds are in general highly complex systems, they frequently consist of a rather small set of fundamentally different building blocks as shown in the taxonomy of Figure 1. Furthermore, the leaf components of this taxonomy can be modeled as a combination of a few fundamental hardware com-

ponents with modules representing associated software workflows. This naturally leads to a modular design of a cloud simulation framework supporting the construction of complex clouds out of a few basic parameterized simulation modules.

We start with outlining the design of the simulation modules representing the basic hardware (infrastructure) components, i.e. multi-core processor systems, networking switches and storage subsystems. This also includes a description of some modeling abstractions we made. Then we describe the modules representing software workflows and demonstrate how they can be combined with hardware modules to form the basic cloud modules of Figure 1.

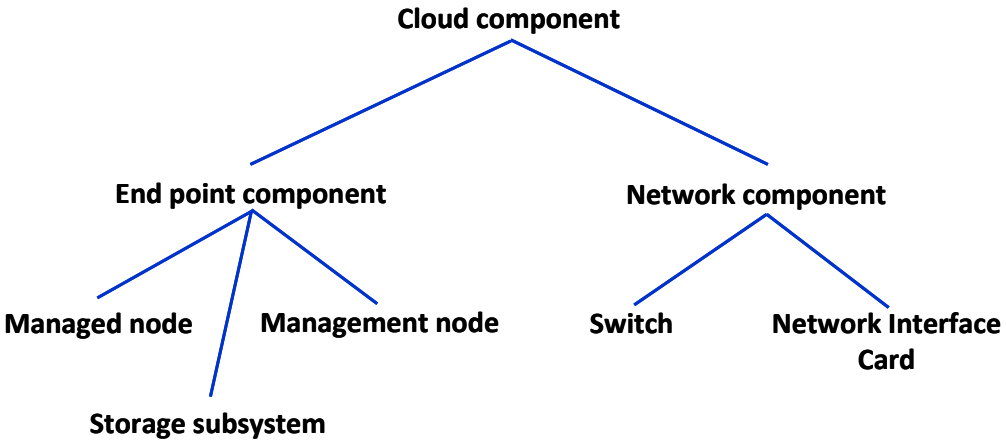

Figure 1: Taxonomy of key cloud components.

These basic cloud modules can then be combined into more complex compound modules representing e.g. a cloud resource pool or cluster consisting of a set of managed systems, storage subsystems and switches . A simulation of a complete cloud may then be built using such resource pool simulation modules, adding simulation modules representing the management subsystem and an appropriate workload generator.

2.1 Basic Hardware Simulation Modules

Our modeling approach for the basic hardware components is as follows: we think of the hardware components providing the active resources (in form of processor cycles or bandwidth associated with disk or network IO) required by the requests posted against the cloud to perform their operations as required by the business logic.

A request executing at a hardware simulation module first tries to allocate the required resources (passive ones like amount of memory or active ones like processor cycles) to proceed. If it is successful, it decreases the available resources accordingly and proceeds to be delayed for a specified amount of time by scheduling an appropriate event in the future event list. If not, the request is inserted into a queue for the required resource and waits. When encountering the scheduled event, the request deallocates the requested resources again, increases the available resources accordingly and retrieves the requests waiting in the appropriate queue to be scheduled immediately so that they can proceed to try to allocate the resources they require.

2.1.1 Multi-core Processor Systems

Our current model of multi-core processor systems consists of the following components:

- processor cores
- memory

- a generic shared resource representing e.g. the processor bus and the Memory Interface Controller (MIC)

Processor cycles are modeled as active resources, while memory and the generic shared resource represent the passive resources required to access the processor cycles. A request at a multi-core processor needs to allocate (part of) the available cycles for a specified amount of time to proceed. After its processing has completed, it releases the cycles again allowing other requests to allocate them if required, see the beginning of this section. The required resources are parameters of the simulation that can be provided in form of parameter files or via the OMNEST graphical user interface.

We support a partial allocation of the available cycles at the processing cores to implicitly model the common situation that a request can not make use of all of the available cycles due to waiting for locks or the completion of IO operations. We also model waiting for locks to become available explicitly when required, see section 2.3.

Using the concept of execution phases described in more detail below, we can model the execution of requests at multi-core processors in terms of a sequence of computational phases, each phase possibly having completely different requirements in terms of resources and execution times required.

A configurable scheduler has also been implemented as part of the multi-core processor system modules, representing the appropriate operating system or the hypervisor functionality. Currently we frequently do not have separate modules modeling system software or middleware, but associate the resources consumed here with the application level requests.

2.1.2 Disks

We model disks as devices providing a specified amount of bandwidth as an active resource. Requests may then allocate a bandwidth for sequential and random read and write IO operations for an appropriate amount of time depending on the (segmented) data size associated with the IO operation. The remaining available bandwidth at the disk is then updated accordingly for all types of IO, i.e. the bandwidth available for newly arriving requests will be reduced accordingly.

Furthermore, we model simple disk caches parameterized by cache hit probabilities and appropriately increased bandwidths.

2.1.3 Switches

Analogous to disks, switches are also modeled to provide bandwidth as an active resource and the request execution as described at the beginning of section 2.1 is also valid here. The routing functionality of switches will be implemented as part of the associated software modules as described in section 2.3. Assuming a switch consists of an internal fabric and a number of attached ports, we associate a maximum bandwidth with the internal fabric and maximum bandwidths with each of its ports allowing for modeling of blocking respectively non-blocking switches.

2.2 Compound Hardware Simulation Modules

In this section we describe how the basic hardware simulation modules can be combined to create compound simulation modules representing more complex hardware components.

2.2.1 Multi-core Processor Systems with local Disks

It is straightforward to combine the basic simulation modules of multi-core processors and disks into compound modules for a multi-core processor systems with a local disk subsystem attached (hosts, nodes), see Figure 2 below.

We can now model the execution of requests here in terms of a sequence of alternating computational and disk IO phases, each phase possibly having completely different requirements in terms of processor cycles or disk bandwidth. Furthermore, we support the concept of background requests, i.e. a computational request is accompanied by concurrent disk IO requests or vice-versa; see Figure 3 below.

2.2.2 Storage Subsystems

Because most of the recent storage subsystems combine a set of multi-core processors with disk arrays, see e.g. (Smart businesses are turning to IBM Scale Out Network Attached Storage from IBM 2011), we can use the same simulation modules here as for multi-core processor systems with local disks and simply use a different sequence of execution phases (e.g. only disk IO phases with computational background) and appropriate parameters.

2.3 Software Simulation Modules

Software simulation modules implement the workflows (business logic) of requests posted against the cloud, e.g. requests for provisioning resources (e.g. retrieving images from an image repository and installing them on a managed node) or requests from users for computation or communication. Key attributes of a request used for implementing its workflow are:

- its type, which is initialized once at request creation time at the workload generator, see section 3 for details.
- its cloud-level phase, which is updated frequently during execution at the modules
- the module ports associated with its arrival and departure
- the maximal number of concurrent requests of each type allowed during each processing phase.

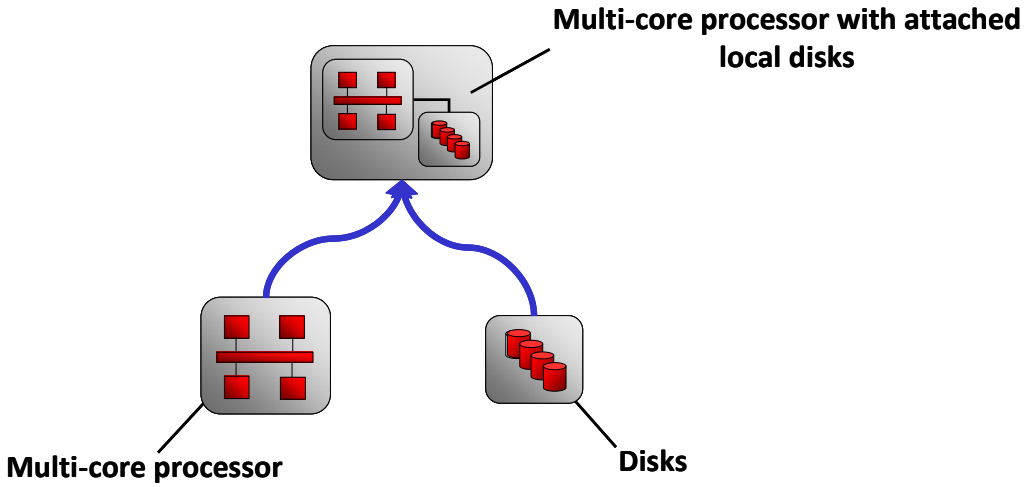

Figure 2: Combining simulation modules for a multi-core processor and for disks into a simulation module for a multi-core processor system with attached local disks. Requests are routed to the multi-core processor module first and then from there to the disk module during disk IO phases.

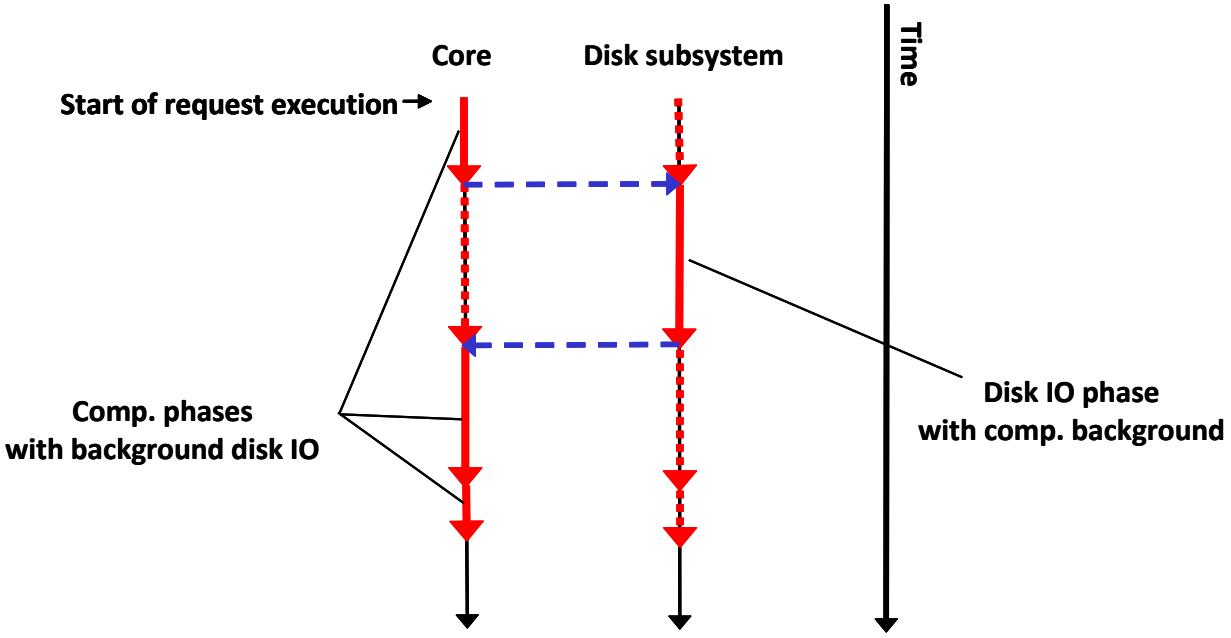

Figure 3: A sequence diagram representing the execution of computational and disk IO phases at a multi-core processor node: we model the execution of requests in terms of a series of computational and disk IO phases possibly including associated background requests (indicated by the dotted line).

This last limitation is frequently encountered at so called "critical regions" in software workflows where several requests (threads) need to be serialized to consistently update a shared variable (see Figure 4) or due to the management system of the cloud protecting the managed infrastructure from being overloaded, see (Configuration Maximums VMware* vSphere* 4.1 2011).

Currently we have implemented software simulation modules for management nodes, managed nodes (hypervisors), other administrative nodes like a VMware* vCenter* Server (VMware* vCenter* Server 2011), storage subsystems and network switches.

The software switch simulation modules play a special role gluing together the other modules representing hosts and storage subsystems. They currently support 3 different types of switch ports associated with the 3 major types of networks available in a cloud: the management, customer and storage LAN ports. The software switch simulation modules route the incoming requests according to their input port, type and execution phase and are completely configurable using parameter files. This is a key feature for rapid prototyping.

Software simulation modules always need to be accompanied by appropriate hardware simulation modules providing the resources necessary for the requests to proceed executing their business logic, see Figure 5 and the next section.

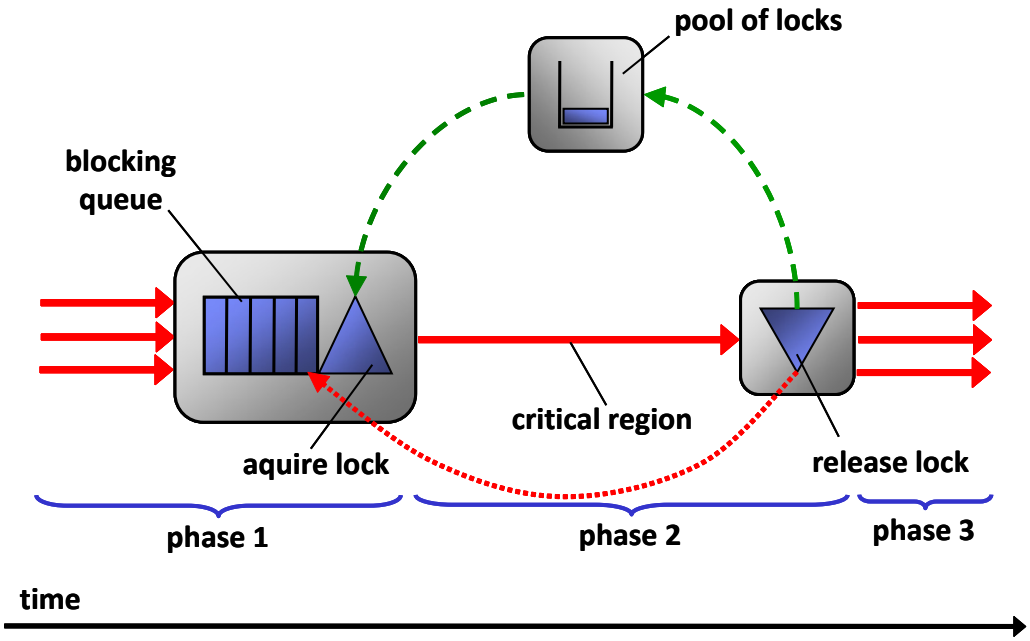

Figure 4: This figure indicates how we can apply our concept of execution phases to model a "critical region" during execution of a software workflow: several incoming requests of phase 1 may compete for a lock (token) required to enter the "critical region" (phase 1) allowing only 1 request in flight. The non-successful requests have to wait in a queue, while the request owning the lock can proceed. Once this request has completed processing in the critical region, it returns the lock to the pool and initiates popping off another request from the head of the queue to acquire the lock and proceed to the "critical region". After leaving the "critical region" and entering phase 3, we again may have more than one request concurrently in flight. The heuristics described here is analogous to the one outlined at the beginning of section 2.1 for requests requiring resources on hardware level.

2.4 Basic Cloud Simulation Modules

The basic simulation modules of clouds representing e.g. management nodes, managed nodes, the storage subsystem and network switches can now be easily created by combining hardware- and software modules; see Figure 5 and the associated caption for more details on how a software workflow module is combined with a multi-core processor system with local disks.

The careful separation of software and hardware simulation modules is one of our key design decisions and has several benefits:

- the development cycles of new hardware and software architectures for clouds are generally quite decoupled; this can be nicely reflected in different development cycles for the associated simulation modules
- it is straightforward to combine various software modules with a single hardware module or vice versa modeling e.g. several guest operating systems or virtual machines on one hypervisor or distributing requests to a cluster of physical nodes,.
- support of software and hardware driven simulations, i.e. we may focus on detailed simulations of software workflows using rather generic and simple simulation modules for the hardware components or analyze the performance of a detailed simulation model of a special hardware component (e.g. a switch) exposed to typical cloud workload patterns.

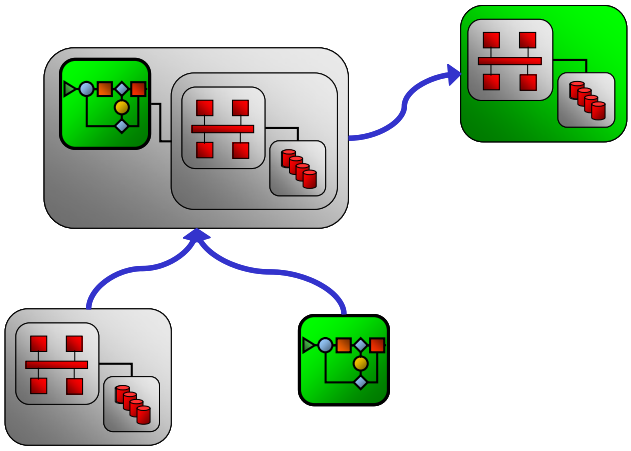

Figure 5: A simulation module representing a software workflow (green) is combined with a multi-core processor system with attached local disks to form a basic cloud simulation module (e.g. a management node): the software module sends out requests to the hardware modules to allocate a specified amount of resources for a certain amount of time as required to execute its business logic. The green box at the top right of the figure indicates the combination of hardware- and software modules and we will use this icon and similar ones in figures below to represent cloud components.

2.5 Compound Cloud Simulation Modules

The basic cloud simulation modules can then be combined into compound simulation modules representing e.g. VMware* cluster or datacenters by connecting their ports; see Figure 6. The final step is then to combine these compound simulation modules into a complete cloud simulation model and include an appropriate workload generator (for details on the workload generator module see below) to post requests against it, see Figure 7.

3 SIMULATION OF CLOUD WORKLOADS

The workload posted against the cloud can be categorized at high level into customer and administrative workloads; for a more detailed taxonomy of cloud workloads (i.e. requests posted against the cloud); see Figure 8 below. Each request posted against the cloud is characterized by a set of attributes with the most important ones being its type (associated with the requests of the taxonomy described above) and its execution phases. Both attributes are instrumental for implementing the request workflow. A key design point in our framework is to decouple the phases on the cloud workflow level (used to implement the business logic) from the lower level phases required to implement the request processing at the hardware components; see Figure 9. This allows the rapid introduction of new cloud level workflows without the need to change the requests execution workflows at the hardware simulation modules or vice versa.

Other important attributes specify the requirements of requests for processor cycles or network and disk bandwidth during the various execution phases; these attributes are drawn from parameter distribution functions associated with the request type and phase at the appropriate simulation modules.

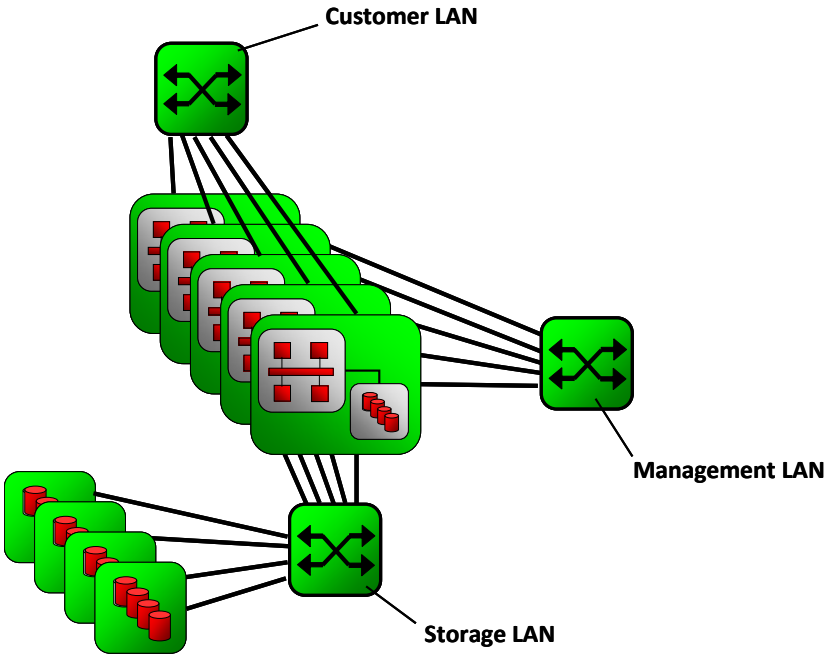

Figure 6: A typical cloud cluster consisting of managed nodes, a storage subsystem and appropriate LANs for customer access, management requests (e.g. provisioning of images) and storage access. The green colour indicates that we always use a combination of hardware and software simulation modules as discussed in section 2.4 above.

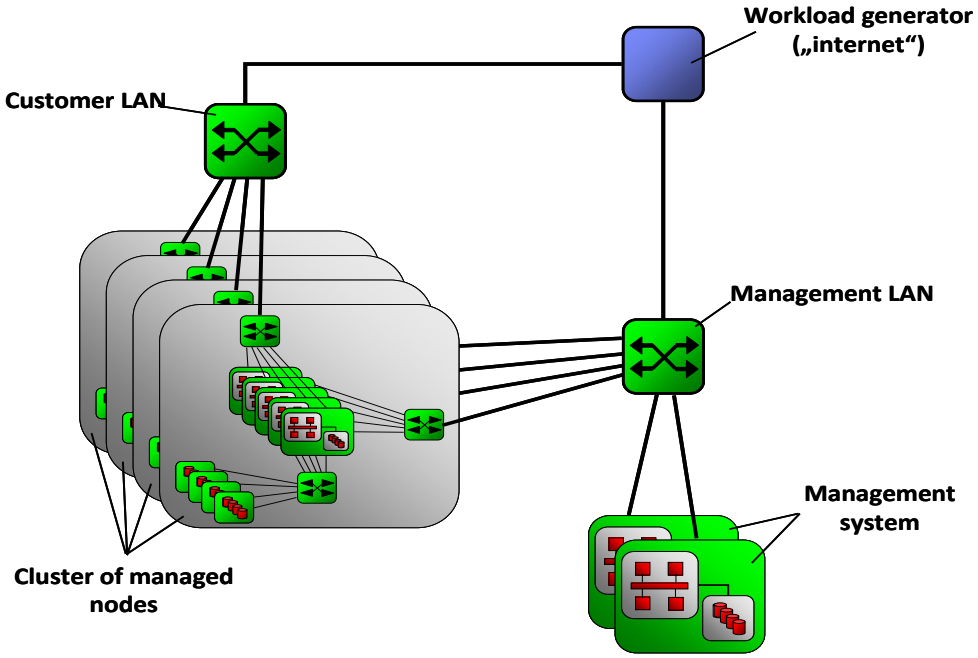

Figure 7: A typical cloud simulation consisting of various clusters of managed nodes including storage subsystems and appropriate LANs for customer and management traffic. The green colour indicates that we always use a combination of hardware and software simulation modules as discussed in section 2.4 above. For details concerning the workload generator, see section 4.

4 THE CLOUD WORKLOAD GENERATOR

In the workload generator module we have implemented all functionality related to generating, initializing and posting requests of various types against the cloud and collecting all request related statistics. Device related statistics like utilization are collected at the device simulation modules. Therefore, all requests need to return to the workload generator for collecting statistical data, even if they do not spend any simulation time on their way back to the workload generator.

We currently support the creation of open and closed streams of various request types possibly with bulk (batch) arrivals.

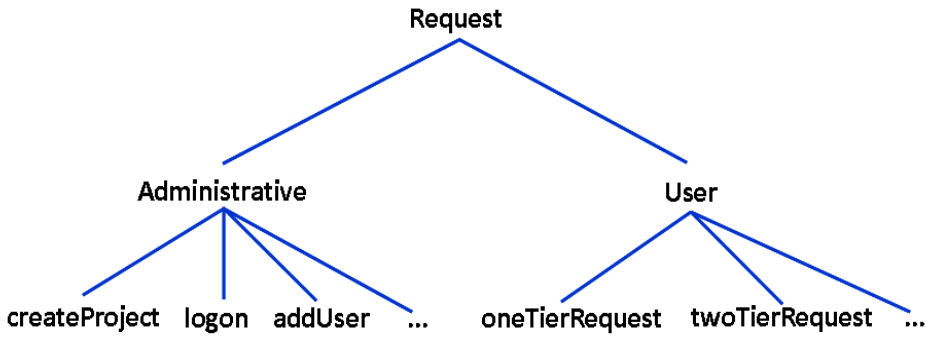

Figure 8: A taxonomy of some typical requests posted against clouds. The associated request types are used within the software simulation modules to implement the business logic. The hardware modules are of course agnostic with regard to the request type, i.e. they provide the available physical resources to whatever request asks for them.

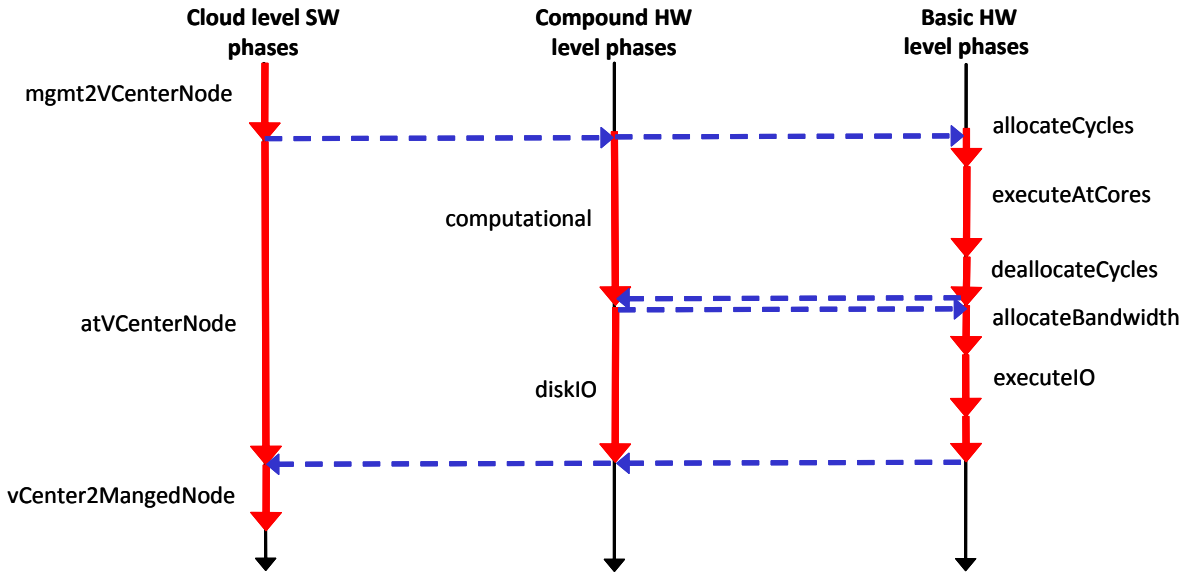

Figure 9: A (logical) sequence diagram indicating the relationships between the various levels of request execution phases: the cloud level phases (implemented in software modules) initiate sequences of computational and IO phases at the level of compound hardware modules and these phases initiate sequences of low-level phases at the basic hardware modules for resource management and request execution.

5 CONCLUSION

In this publication we have described the architecture of a modular framework for cloud performance simulations. This framework enables capacity planning and supports design of new cloud designs in a timely and accurate manner by supporting the creation of new cloud simulations by gluing together basic and compound simulation modules representing cloud building blocks. The implementation of the building blocks themselves may be adapted to satisfy accuracy or execution time constraints of various performance simulation studies. Key design points are the decoupling of the simulation of hardware and software components as well as of the simulation of workflows at cloud level and at component level.

OUTLOOK

We plan to refine the simulation of our current hardware and software modules, add more simulation modules, increase scalability by parallelization and implement the generation and simulation of complex customer workloads. Furthermore, we plan to enable the collection of additional statistical data and increase the general usability of our framework.

ACKNOWLEDGEMENTS

Trademark, service mark, or registered trademark of VMware Corporation in the United States, other countries, or both. Other brands and product names mentioned in this manual may be trademarks or service marks of their respective companies organizations and are hereby acknowledged.

REFERENCES

Banks, J., J.S. Carson II, B.L. Nelson and D.M. Nicol. 2005. *Discrete-Event System Simulation*, 4th Edition. Prentice Hall.

Bertsekas, D. and R. Gallager. 1992. *Data Networks*, 2nd Edition. Pearson Education.

Bloch, G., S. Greiner, H. de Meer and K.S. Trivedi. 2006. *Queueing Networks and Markov Chains*, 2nd Edition. Wiley-Interscience.

Configuration Maximums, VMware vSphere 4.1. Accessed May 25, 2011. http://www.vmware.com/pdf/vsphere4/r41/vsp_41_config_max.pdf. Link last verified on 25.05.2011.

IBM. "Smart businesses are turning to IBM Scale Out Network Attached Storage from IBM." Accessed May 13, 2011. http://public.dhe.ibm.com/common/ssi/ecm/en/tsb03016usen/TSB03016USEN.PDF.

Kleinrock, L. 1975. *Queueing Systems, Volume 1: Theory*. John Wiley.

Kleinrock, L. 1976. *Queueing Systems, Volume 2: Computer Applications*. John Wiley.

Law, A.M. and W.D. Kelton. 2000. *Simulation Modeling and Analysis*, 3rd Edition. McGraw-Hill.

OMNEST – High-Performance Simulation for All Kinds of Networks. Accessed May 7, 2011. http://www.omnest.com/.

Pasricha, S. and N. Dutt. 2008. *On-Chip Communication Architectures*, Elsevier Inc.

Varga, A. and R. Hornig. 2008. "An overview of the OMNeT++ Simulation Environment." In *Proceedings of First International Conference on Simulation Tools and Techniques for Communications, Networks and Systems* (SIMUTools'08), Marseille, France, March 2008.

VMware vCenter Server. Accessed May 7, 2011. http://www.vmware.com/products/vcenter-server/overview.html.

Wikipedia. "Cloud Computing." Accessed May 6, 2011. http://en.wikipedia.org/wiki/Cloud_computing.

AUTHOR BIOGRAPHIES

PETER ALTEVOGT is a performance architect at the IBM Germany Research & Development Laboratory. He holds degrees in Mathematics and Physics from the University of Heidelberg, and he holds a

doctorate in theoretical physics from the University of Karlsruhe. His email address is peter.altevogt@de.ibm.com.

TIBOR KISS is a performance engineer at Gamax Kft, working on projects at the IBM Zurich Research laboratory. He holds B.S degree in Computer Engineering from the College Of Dunaujvaros, Hungary. His e-mail is tibor.kiss@hu.ibm.com.

WOLFGANG DENZEL is a research staff member at the IBM Zurich Research laboratory, Switzerland. He holds M.S and Ph.D. degrees in Electrical Engineering from the University of Stuttgart, Germany. His email address is wde@zurich.ibm.com.