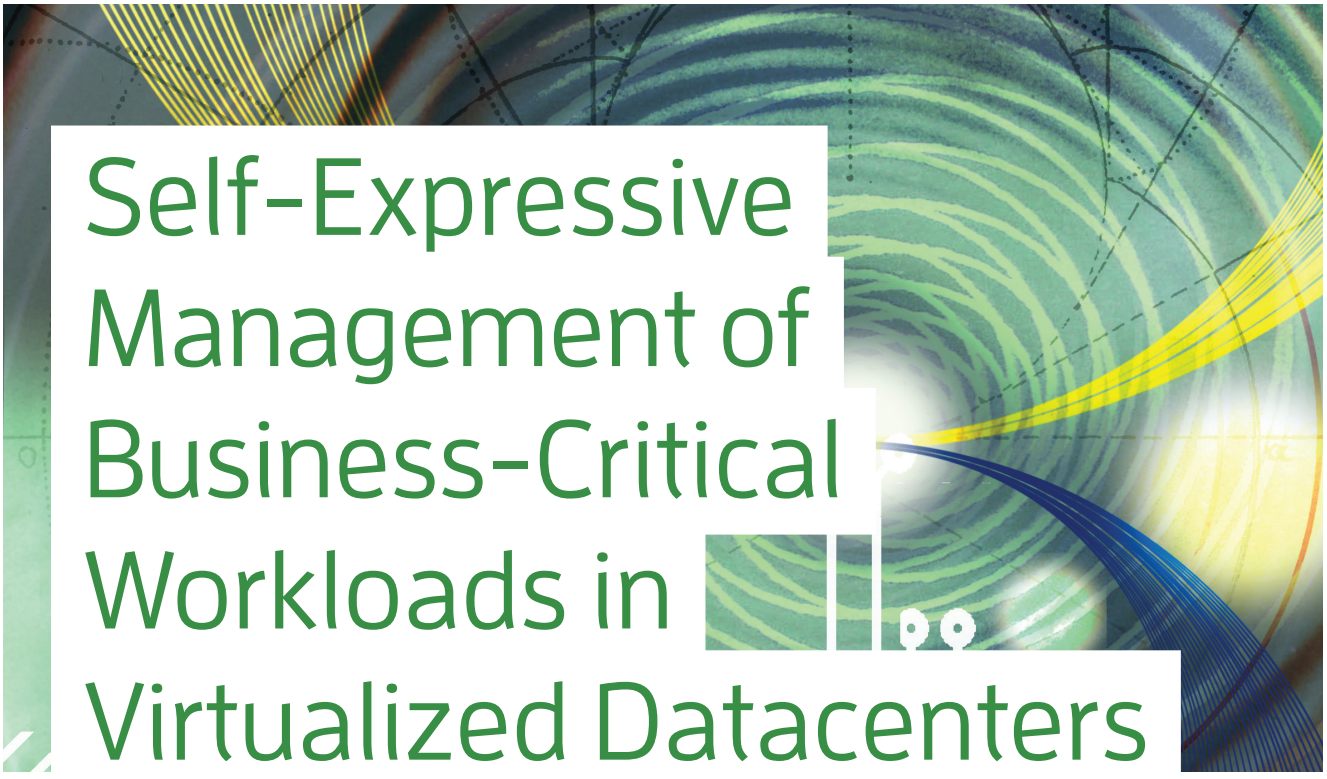

Self-Expressive Management of Business-Critical Workloads in Virtualized Datacenters

Vincent van Beek, ASP4ALL Bitbrains

Jesse Donkervliet, Tim Hegeman, Stefan Hugtenburg, and Alexandru Iosup, Delft University of Technology

The Mnemos resource management and scheduling architecture uses portfolio scheduling, topology-aware virtual-resource management, and state information to self-adapt to significant workload changes and to analyze risks. Simulations with real-world workload traces reveal the potential for significant cost savings.

To run their business-critical workloads, many large enterprises and governments are moving toward leasing computation from datacenters, which host the workloads as diverse digital services on virtualized resources. IDC predicts that by 2017 over three-quarters of business-critical data will reside in virtualized datacenters,[1] and major digital economies, such as the EU, the US, and Japan, are already expanding datacenter customers. The EU predicts that its digital economy will grow 50 percent from 2015 until 2020.[2]

Through resource virtualization, datacenters can service many workload types and respond to a variety of resource requirements, while still operating relatively modest physical resources. These economies of scale in cost, energy, and human resources are critical to a datacenter's ability to cost-effectively handle increasing demand, but maintaining them requires self-aware and self-expressive techniques to address increasing scale, changing architectures, and dynamic workloads. Automated state monitoring, for example, can enable intelligent scheduling and other decision making, but it must accommodate novel architectures driven by big data applications, which are becoming central to business-critical workloads. The sidebar "The Challenge of Business-Critical Workloads" describes workload characteristics

0018-9162/15/$31.00 © 2015 IEEE

THE CHALLENGE OF BUSINESS-CRITICAL WORKLOADS

Business-critical workloads are user-facing, back-end enterprise services that generally support business decisions and are typically contracted under strict service-level agreement requirements. Their downtime or even low performance will decrease revenue and productivity and possibly lead to financial loss, legal action, and departing customers.[1] Business-critical workloads often include applications based on Monte Carlo simulations, such as financial modeling and applications such as email, databases, customer relationship management, and collaborative and management services.

WORKLOAD CHARACTERISTICS

Business-critical workloads differ significantly from scientific and analytic workloads in the level of customer data sensitivity. Because details about datacenter customers' software cannot be revealed, software requirements are typically expressed in virtual machines (VMs) instead of as applications. Over the past two decades, business-critical workloads have changed from sequential jobs and Web applications to a mix of long-running services and high-performance applications that are both computation-intensive (MPI) and data-intensive (MapReduce and Pregel). If this recent history is any indication, workload characteristics are likely to continue changing significantly.

Although datacenter operators benefit from the shift to leasing computation, they also face interesting new challenges in resource management and scheduling related to rising volume, diverse requirements, and rapid workload changes. When workload volume is high, VM scheduling must be fully automated with minimal risk of low performance. Intelligent resource managers must be aware of the network topology, for example, to ensure that critical datasets are not placed on the same physical machine or on machines that are likely to fail together.

MANAGEMENT STRATEGIES

Traditional approaches to managing datacenters cannot cope with these challenges. Developing or even selecting scheduling policies is error-prone and ephemeral, because new workload requirements often invalidate the previous approaches and scheduler selection remains a challenge.[2,3] Instead, many datacenters use simple approaches combined with human expertise,[3] but human resources are scarce, error-prone, and often too slow to respond to dynamic customer requirements and uncertain workload conditions.

Self-* principles provide the roots of a radically different approach to handling dynamically changing requirements. Self-awareness in computer systems is the ability to alter behavior in some beneficial way, without human intervention,[4] while self-expressiveness is the notion that a system knows its own state, context, goals, values, objectives, and constraints.[5] We believe that self-awareness and self-expressiveness can offer key adaptation capabilities to datacenters managing business-critical workloads.

References

1. S. Shen, V. van Beek, and A. Iosup, "Statistical Characterization of Business-Critical Workloads Hosted in Cloud Datacenters," *Proc. 15th IEEE/ACM Int'l Symp. Cluster, Cloud, and Grid Computing* (CCGRID 15), 2015; www.pds.ewi.tudelft.nl/~iosup/business-critical-datacenter-workloads15ccgrid.pdf.
2. D. Feitelson, "Packing Schemes for Gang Scheduling," *Proc. 2nd Workshop Job Scheduling Strategies for Parallel Processing* (JSSPP 96), LNCS, vol. 1162, 1996, pp. 89–110.
3. D. Klusácek and S. Tóth, "On Interactions among Scheduling Policies: Finding Efficient Queue Setup Using High-Resolution Simulations," *Proc. 20th Int'l Conf. Parallel Processing* (Euro-Par 14), LNCS, vol. 8632, 2014, pp. 138–149.
4. H. Hoffmann et al., "SEEC: A General and Extensible Framework for Self-Aware Computing," tech. report MIT-CSAIL-TR-2011-046, Computer Science Dept., MIT, 2011; http://hdl.handle.net/1721.1/67020.
5. P.R. Lewis et al., "A Survey of Self-Awareness and Its Application in Computing Systems," *Proc. 5th IEEE Int'l Conf. Self-Adaptive and Self-Organizing Systems Workshops* (SASOW 11), 2011, pp. 102–107.

that are driving datacenters to more efficient scheduling approaches.

To meet these needs, we developed Mnemos, a self-expressive architecture for resource management and scheduling in virtualized datacenters. Mnemos has two main self-aware and self-expressive components:

- *Datacenter-wide portfolio scheduler.* Scheduling policies are typically designed for each new workload (and sometimes application) type, always with much effort and cost. Mnemos' portfolio scheduler is self-aware and self-expressive and thus can continuously select an appropriate policy from those in its portfolio and quickly reflect on each policy's usefulness.
- *Virtualization-aware scheduler.* Distributed data–intensive applications, such as Hadoop, perform poorly when running on virtual machines (VMs) mapped to the same physical machine, and these applications can lose data when hosted together on a machine that fails. Mnemos' virtualization-aware scheduler, Nebu, is designed to automatically handle these applications by running them on VM clusters that are agnostic to both the virtualization provider and the distributed application, and thus require no human-managed mapping.

To evaluate Mnemos, we simulated its use with traces of real-world, business-critical workloads in a datacenter infrastructure that includes a multicluster setup. Our goal was to determine the degree to which Mnemos ensures reliable and high-performance service to datacenter customers. Our results show that both the portfolio and virtualization-aware schedulers can help datacenter operators understand and lower the risk of suboptimal performance. Overall, Mnemos decreases operational risk with an acceptable performance penalty. In our experiments, the portfolio scheduler that equips Mnemos made decisions about large-scale problems in a matter of minutes.

MNEMOS ARCHITECTURE

Mnemos uses self-* principles to fully automate operation across multiple physical clusters within the entire datacenter and across multiple datacenters that share an operational domain. Its focus is to reliably execute business-critical workloads, including running high-performance computing (HPC) and big data applications, while maintaining good performance with little risk of data loss, even with highly dynamic workloads. Mnemos, currently in the prototype stage, can handle a wide range of business-critical applications, although it is not yet completely generic.

As Figure 1 depicts, Mnemos combines the typical components of a datacenter resource manager and scheduler—VM manager, system monitor, and application (app) managers—with self-aware and self-expressive components. These core components cover the typical roles in a self-aware computing framework's decoupled observe-decide-act loop:[3] the system monitor is the observer, the portfolio scheduler is the decider, and the VM manager is the actor or executer. Other than these core components, Mnemos uses tools familiar to most datacenters because replacing existing components with self-* components is likely to occur gradually.

VMs and vClusters

Mnemos manages user requests transparently without the need for additional administration. Datacenter users embed their applications in VMs, either as a single VM or VM cluster (vCluster). Single VMs are suitable for running low-load webservers and small database applications, whereas vClusters are better for multitier Web, big data, and HPC applications.

To facilitate hosting applications in a single VM or in entire vClusters, Mnemos' app managers automatically manage common business-critical applications, such as Hadoop for big data applications and Microsoft HPC for HPC applications.

Because it distinguishes between big data and other applications, Mnemos addresses the current pressing need for placement strategies specific to big data applications. In Figure 1, App A represents a big data application running in a vCluster, and Apps B and C are regular applications running in a single VM and in a vCluster, respectively. Nebu places big data applications on hosts, using placement strategies that can guarantee data locality or distribution over multiple clusters. It then communicates placement decisions to the application, which can use this information to optimize its internal scheduling and its own job-placement strategies. The portfolio scheduler handles regular application scheduling directly through policies that aim to optimize performance and reduce oversubscription to physical hosts.

The VMs that users or app managers request run on physical resources, which Figure 1 shows in the datacenter layer. Multiple datacenters and one or

more clusters within each datacenter are connected through high-speed fiber optics. The clusters are built from physical hosts (which contain CPUs, memory, and networking), storage servers, and network routers. To run VMs, all the hosts run a hypervisor—a component for creating and running VMs—that grants the VMs use of physical resources, including CPU, memory, network, and storage.

Consistent with its focus on modern business-critical workloads, Mnemos differs from traditional architectures in several key aspects:

- Mnemos uses only single-cluster failure domains, whereas traditional architectures use multiclusters;
- through virtualization, Mnemos users can simultaneously occupy computing nodes, whereas in traditional HPC and grid-computing architectures, node use is mutually exclusive;[4] and
- finally, Mnemos uses a high-performance network, whereas typical grid-computing and cluster-based architectures rely on relatively slower and less expensive networks.

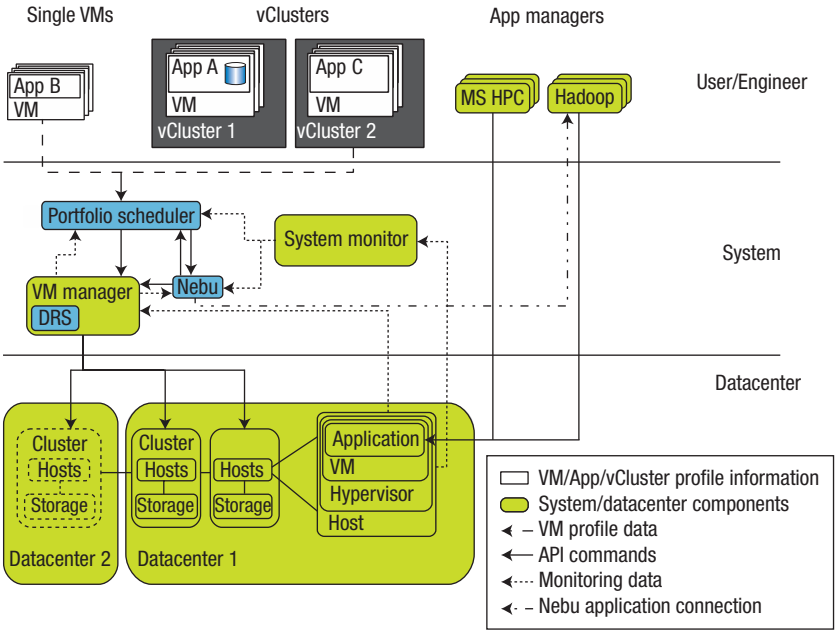

FIGURE 1. Overall model of the Mnemos datacenter-wide architecture. Components in *blue* use self-aware or self-expressive techniques. In addition to these, Mnemos incorporates familiar support components, which are freely or commercially available such as the app managers, system monitor, and virtual machine (VM) manager, to ensure that datacenters have the necessary tools as they gradually increase the architecture's self-awareness with future self-* components. VMWare's Dynamic Resource Scheduling (DRS) tool, which works with the VM manager, is an example of how the architecture can spread self-* capabilities.

Component overview

Of Mnemos' main components, only the portfolio scheduler and Nebu have self-aware and self-expressive capabilities. The VM manager and system monitor, which mediate between user requests and physical resources, enable self-* capabilities but alone cannot ensure them.

VM manager. The VM manager provisions and allocates VMs in each cluster, subject to the scheduler's intelligent decisions. It can be any of the many single-cluster and multicluster managers on the market, including the freely available Condor, Globus, and Mesos tools. The VM manager can also use self-aware commercial tools, such as VMWare's Dynamic Resource Scheduling tool, to allocate and migrate VMs across hosts within the same cluster.

System monitor. The system monitor maintains information about the datacenter's internal state by gathering information about the center's system components, including hosts, storage, networking, VMs, and the hypervisor. The system monitor can be any commercial or freely available monitor, such as Ganglia, or any of the monitoring tools that typically accompany commercial VM managers.

Self-* components. Mnemos' two self-* components—the portfolio scheduler and Nebu—are tools we created to intelligently manage both workloads and scheduling policies. Both use the VM manager and system monitor to gather information about the datacenter's current and historical states. Through these components, Mnemos enables self-expressive resource management and scheduling across multiple datacenters and multiple vClusters, and supports big data applications in distributed virtualized environments.

Table 1 summarizes the portfolio scheduler's and Nebu's self-aware and self-expressive features. Although the two have different functions, they have a common goal: to proactively gather system information that will increase their self-aware and self-expressive capabilities and use that information to greatly enhance a datacenter's functionality and efficiency.

Mnemos' portfolio scheduler is based on an earlier scheduler[4] that uses resources from public datacenters

TABLE 1. Self-aware and self-expressive features in Mnemos' portfolio scheduler and Nebu.

Feature	Self-* principle	Component
Autonomously learns the physical infrastructure hierarchy from monitoring tools, allowing adaptive virtual machine (VM) placement	Self-awareness	Portfolio scheduler, Nebu
Provisions VMs and VM clusters in one or more datacenter clusters, responding to observed behavior and decisions from the portfolio scheduler, allowing adaptive VM placement	Self-expressiveness	Portfolio scheduler
Monitors VM host mapping and detects changes in the infrastructure so that the placement policies can adapt and optimize for the new situation	Self-awareness	Portfolio scheduler, Nebu
Monitors available physical hardware in datacenters and observes infrastructure use, allowing adaptive behavior that can optimize datacenter use	Self-awareness	Portfolio scheduler, Nebu
Monitors VM behavior to make more informed decisions about VM placement	Self-awareness	Portfolio scheduler
Automatically deploys distributed applications across datacenters through placement policies that take into account observations from both physical and virtual infrastructures	Self-expressiveness	Nebu

(infrastructure-as-a-service clouds) to create a scheduling policy portfolio for scheduling HPC workloads. An evaluation showed that the portfolio outperformed any of the individual policies it contained.

Although Mnemos' portfolio scheduler is modeled after this scheduler, it is the first scheduler we know of that can operate across multiple clusters and datacenters. Adaptive scheduling techniques have been considered for datacenters[5] and clouds,[4,6,7] but not across multiple clusters and datacenters and not for the workloads Mnemos targets. Commercial tools such as PBS, Mesos, and Cloudera can schedule hierarchically, but they cannot schedule VM placement in multiple datacenters. Commercial cluster-management software, such as that offered from VMware, currently lacks an understanding of big data applications and a holistic view of datacenter management. As techniques similar to Mnemos' become more widespread, we expect this situation to change.

Portfolio scheduler

The portfolio scheduler responds to changes in workload patterns across multiple users, datacenters, and clusters. Its name derives from the idea of a stock portfolio—a collection of stocks designed to mitigate risks and achieve better overall performance. Portfolio schedulers are by nature self-expressive, in that they adapt to reflect changed user demand, goals, or knowledge.

Adding self-awareness to the organizing mechanism, in our case, the portfolio scheduler, alleviates the need for engineers to tune the datacenter. The scheduler iteratively selects the best scheduling policy for the foreseeable future, from a set (portfolio) of constituent policies.

Portfolios can also become the basis for search instruments applicable to a variety of domains. For example, an exhaustive simulation of each portfolio item's behavior, whether it is a stock or a scheduling policy, can provide information essential to making investment or scheduling decisions. However, each new portfolio application can require a new design or significant adaptation.

For Mnemos' portfolio scheduler, we nontrivially adapted a previous portfolio scheduler developed for single-cluster operation.[4] Figure 2 shows the operational model. Operations have three main phases: equipping the portfolio with policies, selecting a policy, and applying a policy.

Equipping the portfolio. The system administrator equips the portfolio with a set of scheduling policies that are unique to the problem at hand. Policies generally address the scheduling of VMs, rack-based clusters, and datacenters and specifically address the requirements for scheduling business-critical workloads, such as affinity and anti-affinity for VM placement and load balancing across clusters and datacenters. Many other classes of scheduling policies exist, as the sidebar "Workload Scheduling Approaches" describes.

Policies range from simple round-robin or first-fit placement scheduling to complex policies based on resource utilization metrics and VM-specific characteristics.[8] For example, the complex BB vCluster policy (bottom of the list in Figure 2) groups VMs according to exclusion vectors and assigns groups to clusters on the basis of cluster-wide CPU availability, storage requirements, and network workload, taking into account the requested VM memory sizes.

Policy selection. The portfolio scheduler selects a single scheduling policy from the portfolio online. Online selection better aligns with the large scale of multiple datacenters because, unlike traditional periodic selection, it does not allow large request accumulations in the system's queues.

As part of the selection process, the portfolio scheduler simulates

WORKLOAD SCHEDULING APPROACHES

Resource scheduling in a datacenter usually takes place on different execution stack levels, starting at the CPU level and moving to the OS, hypervisor, and software levels. Usually the higher the stack level, the more information is available on workload context and characteristics.

Workload scheduling within a datacenter or across multiple centers is attracting increasingly more attention. From our survey of general scheduling policies applied in datacenters, grids, and clusters, we have identified four main scheduling approaches: specialized, single-tier, multitier, and meta.

SPECIALIZED

Specialized scheduling focuses on finding scheduling policies for very specific workload types, such as workflows and parallel and sequential jobs. Because scheduling performance is commensurate with the amount of available information, specialized scheduling mechanisms are often found at the software level, high in the execution stack. However, specialized scheduling policies can still make decisions at lower levels. For example, information about what is running on a VM can help intelligently allocate VMs to hardware.

SINGLE-TIER

Work on model-based scheduling focuses on finding single-tier workload models and using the insights these models provide to optimize future workload scheduling. These single-tier models are also used to test scheduling policies in simulators. Modelers use different modeling approaches, such as analytical and statistical, to develop workload models.

MULTITIER

Multitier applications often have two characteristics (multicolinearity and highly dynamic load patterns) that are very specific for this type of setups. Because Web applications are often hosted on multitier systems—webserver and database server—much research has been devoted to this topic. The resulting variety of approaches use a range of modeling strategies, including queue, prediction, and analytical models, to achieve effective dynamic resource allocation for multitier applications.

META

Meta scheduling requires a scheduling approach that works across multiple datacenters. However, many datacenters run a wide variety of workload types, so there is little chance of finding a one-size-fits-all policy. Portfolio scheduling, one recent approach of interest, works with a set of policies, choosing the best one for every workload subset. Consequently, datacenters can create less complex scheduling policies that work well for certain workload subsets without worrying if the policy works well for another complete workload.

the entire datacenter to determine the impact of the candidate policy on the datacenter's operations. The simulator gathers information about the datacenter's current system state and about current and past workloads. It also reports various operational and performance metrics, such as the maximum resource load and the expected response time for each workload unit (median, 99th percentile, and so on).

Using a utility function that the system administrator provides, the scheduler then selects the scheduling policy with the highest utility. The utility function, which is unique to the application domain, captures the datacenter operator's requirements—for example, minimizing the imbalance between each cluster's average and maximum workloads. Thus, in gathering information about the datacenter, the portfolio scheduler ensures that Mnemos is self-aware, and through the utility function, that Mnemos is self-expressive.

Policy application. The last operational step is to apply a selected policy any time it is needed until another policy selection step invokes another policy. For business-critical workloads, policies decide where to place VMs. Inside the VMs, customers deploy applications that are then outside the policy scheduler's control.

Virtualization-aware scheduler

To our knowledge, Nebu is the first virtualization-aware scheduler that is agnostic to both the VM manager and application and that emphasizes Hadoop-based and distributed database applications.[9] Its main goals are

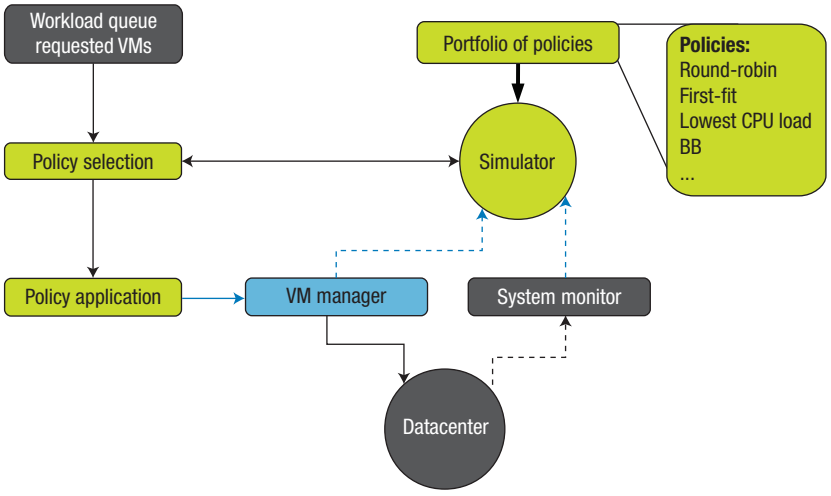

FIGURE 2. Operational model of the Mnemos portfolio scheduler. Both the VM manager and system monitor gather information about the datacenter from the simulator (dotted arrows), which contributes to the portfolio scheduler's self-* capabilities. Blue dashed denotes these capabilities in action. For example, the arrow from "Policy application" to the VM manager ensures the self-expressiveness of the Mnemos portfolio scheduler by triggering the provisioning of VMs and vClusters in one or more datacenter clusters.

to prevent data loss when resources fail and to reduce physical resource contention from the accidental collocation of VMs.

Nebu informs big data applications, or their app manager, of their VM host's physical topology, thus enabling self-aware and self-expressive application scheduling. Nebu also includes a system-level scheduler that can, without application input, intelligently place VMs so that they do not overlap.

As Figure 3 shows, Nebu's architecture includes three layers:

- *an application extension layer*, which provides distributed applications with host topology knowledge and feeds application requirements to the portfolio scheduler;
- *a VM manager extension layer*, which extracts meaningful information and transmits Nebu's scheduling decisions; and
- *a middleware layer*, which comprises the Nebu resource manager and a component that interacts with the portfolio scheduler; this connects the other layers and decides on the basis of app manager and VM manager information if new VMs are needed.

To enable topology-aware VM placement and the scheduling of big data applications in virtualized environments, Nebu collects information from the VM manager on the physical network topology, as well as host (cluster and datacenter) and storage locations. Storage locations can be in the host or in a dedicated network-attached storage solution. Nebu combines this information with knowledge about the virtual infrastructure, particularly VM location on the physical infrastructure, and its basic understanding of network topology and network types (within a cluster and between clusters or datacenters).

EXPERIMENTAL RESULTS

Experiments to understand the effects of self-* properties for datacenters that support business-critical workloads are not easy to perform. A production datacenter is rarely available for that use, and experimentation can be costly. Even trace collection is hampered by legal and business restrictions.

Fortunately, thanks to the generosity of ASP4ALL Bitbrains, which has a multidatacenter infrastructure, we obtained traces of real-world business-critical workloads (http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains) and secured resource time, which we used to evaluate the portfolio scheduler and Nebu.

Because of space limitations, we present key results only for the portfolio scheduler. In general, the results of the Nebu experiment showed that adding self-awareness to big data application schedulers can significantly increase reliability with an acceptable performance penalty.[9]

Portfolio scheduler implementation

We implemented a complete portfolio scheduler for the production datacenter infrastructure, including a set of placement policies for both a single VM and vCluster that covered the policies in the portfolio in Figure 2.

Workload

The collected workload traces capture long-term, large-scale operations: more than 1,300 VMs for over three operational months. Combined, the VMs have more than 17 TBytes of memory and consumed more than five million CPU-core hours.[10]

Evaluation process

The portfolio scheduler bases its selection on the scores expressed as utility functions based on the datacenter operator's goals, such as minimizing the resource overload or maximizing revenue. In practice, datacenter operators provide a utility function as a portfolio scheduler configuration. We considered six possible goals, which are reflected in six configurations:[8]

- *MinScore*. Mitigate the risk of scarcity for every resource (CPU,

memory, disk, and network).

- *MaxScore*. Pack as many VMs as possible into a cluster, which leads to high resource utilization and thus income for the datacenter operator, but has the highest risk of resource scarcity.
- *MinMem*. Mitigate the risk of memory scarcity for many requests, each with small amounts of requested memory.
- *MaxMem*. Mitigate the risk of memory scarcity for several requests, each with large amounts of requested memory.
- *MinCPU*. Mitigate the risk of CPU scarcity for many requests, each with small amounts of requested CPU capacity.
- *MaxCPU*. Mitigate the risk of CPU scarcity for several requests, each with large amounts of requested CPU capacity.

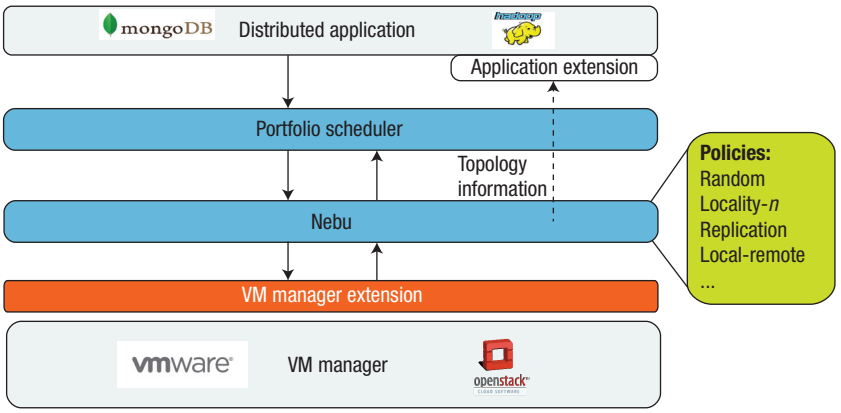

FIGURE 3. How Nebu fits in the core Mnemos architecture. Nebu collects information from the VM manager on the physical network topology and cluster and datacenter storage locations, and combines this information with knowledge about the virtual infrastructure. Blue denotes self-* elements.

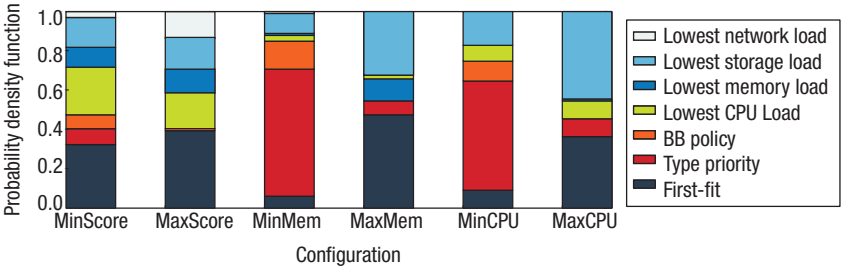

FIGURE 4. Evaluation of policy distribution to minimize and maximize VM and vCluster placement scores. The *y*-axis shows the fraction of times the portfolio scheduler chose the policy, while the *x*-axis shows scheduler configurations that correspond to the datacenter operator's various goals.

Selection frequency

To understand the usefulness of portfolio scheduling, we measured selection frequency for each policy selected and determined the distribution of the chosen policies for each of the six goals. As Figure 4 shows, no scheduling policy was selected more than three fourths of the time, and no policy was never selected. This is strong evidence that self-* principles manage selection efficiently. Using a single policy that never changes makes little sense, particularly when several scheduling policies are frequently selected, as in the MinScore and MaxScore goals.

Virtualized datacenters provide important infrastructure for digital economies, but they also raise new challenges in resource management and scheduling. Through experiments based on real-world workload traces, we found that self-awareness and self-expressiveness, when considered in architectures such as Mnemos, can enhance response to significant workload changes, prevent data loss during failures, and lower the risk of resource scarcity.

Mnemos is part of a larger project on datacenter management, developed in collaboration with ASP4ALL Bitbrains and TU Delft. We have already implemented the Mnemos prototype, tested the portfolio scheduler and Nebu, and are currently extending the Mnemos conceptual framework with more service-level agreements and objectives and more in-depth network and CPU performance metrics. In the near future, we plan to extend the Mnemos prototype implementation to a full-scale operational environment, which we will use to test our design and policies under real-world conditions and with user feedback.

ACKNOWLEDGMENTS

We thank ASP4ALL Bitbrains for access to data and expertise and for financial support. This research is also supported by the Nederlandse Oraganisatie voor Wettenschappelijke Onderzoek (Netherlands Organization for Scientific Research) Kennis Innovatie Mapping (KIEM; Mapping between Knowledge and Innovation) project Kiesa, and by the Commit project Commissioner.

ABOUT THE AUTHORS

VINCENT VAN BEEK is a doctoral student in computer science in the Parallel and Distributed Systems (PDS) group at the Delft University of Technology (TU Delft), the Netherlands, and a systems engineer at ASP4ALL Bitbrains. His research interests include distributed systems, big data, and cloud computing. Van Beek received an MSc in computer science from TU Delft. He is a member of IEEE. Contact him at vincent.vanbeek@bitbrains.nl.

JESSE DONKERVLIET is pursuing an MSc in computer science with an emphasis in distributed systems, as part of TU Delft's PDS group. His research interests include distributed systems, big data, and cloud computing. Donkervliet received a BSc in computer science from TU Delft. Contact him at j.donkervliet@gmail.com.

TIM HEGEMAN is pursuing an MSc in computer science with an emphasis in distributed systems as part of TU Delft's PDS group. His research interests include distributed systems, big data, and cloud computing. Hegeman received a BSc in computer science from TU Delft. Contact him at tim.m.hegeman@gmail.com.

STEFAN HUGTENBURG is pursuing an MSc in computer science with an emphasis on algorithms as part of TU Delft's Algorithmics group. His research interests include distributed systems, big data, and cloud computing. Hugtenburg received a BSc in computer science from TU Delft. Contact him at s.hugtenburg@gmail.com

ALEXANDRU IOSUP is an assistant professor of computer science in the PDS group at TU Delft. His research interests include distributed systems, big data, and cloud computing. Iosup received a PhD in computer science from TU Delft. He is a member of ACM, IEEE, and SPEC. Contact him at a.iosup@tudelft.nl.

REFERENCES

1. *Worldwide and Regional Public IT Cloud Services: 2013-2017 Forecast,* IDC; www.idc.com/getdoc.jsp?containerId=251730.
2. "Europe in a Nutshell," *EU 2020 Smart Growth*, European Commission; http://ec.europa.eu/europe2020/europe-2020-in-a-nutshell.
3. H. Hoffmann et al., "SEEC: A General and Extensible Framework for Self-Aware Computing," tech. report MIT-CSAIL-TR-2011-046, Computer Science Dept., MIT, 2011; http://hdl.handle.net/1721.1/67020.
4. K. Deng et al., "Exploring Portfolio Scheduling for Long-Term Execution of Scientific Workloads in IAAS Clouds," *Proc. ACM Supercomputing Conf.* (SC 13), 2013, pp. 1–12.
5. V. Nae, A. Iosup, and R. Prodan, "Dynamic Resource Provisioning in Massively Multiplayer Online Games," *IEEE Trans. Parallel Distributed Systems*, vol. 22, no. 3, 2011, pp. 380–395.
6. J. He et al., "On the Cost-QoE Tradeoff for Cloud-Based Video Streaming under Amazon EC2's Pricing Models," *IEEE Trans. Circuits and Systems Video Technology*, vol. 24, no. 4, 2014, pp. 669–680.
7. S. Farokhi et al., "Self-Adaptation Challenges for Cloud-Based Applications: A Control Theoretic Perspective," *Proc. ACM Int'l Workshop Feedback Computing* (IWFC 15), 2015; www.infosys.tuwien.ac.at/staff/sfarokhi/soodeh/papers/Soodeh-Farokhi_CameraReady_FeedbackComp-2015.pdf.
8. J. Donkervliet, T. Hegeman, and S. Hugtenburg, "Nebu: A Topology-Aware Deployment System for Reliable Virtualized Multi-Cluster Environments," bachelor's thesis, TU Delft, 2014; http://repository.tudelft.nl/view/ir/uuid:aa101139-5fe5-457d-85f5-cf939cfe3868.
9. V. van Beek, "Design and Evaluation of a Portfolio Scheduler for Business-Critical Workloads Hosted in Cloud Datacenters," master's thesis, TU Delft, 2015; http://repository.tudelft.nl/view/ir/uuid%3A43db2d0f-9593-4eac-901a-ecd7783805fc.
10. S. Shen, V. van Beek, and A. Iosup, "Statistical Characterization of Business-Critical Workloads Hosted in Cloud Datacenters," *Proc. 15th IEEE/ACM Int'l Symp. Cluster, Cloud, and Grid Computing* (CCGRID 15), 2015; www.pds.ewi.tudelft.nl/~iosup/business-critical-datacenter-workloads15ccgrid.pdf.

Selected CS articles and columns are also available for free at **http://ComputingNow.computer.org**.