2014 IEEE 22nd International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems

Concurrent and Distributed CloudSim Simulations

Pradeeban Kathiravelu
INESC-ID Lisboa
Instituto Superior Técnico, Universidade de Lisboa
Lisbon, Portugal
Email: pradeeban.kathiravelu@tecnico.ulisboa.pt

Luis Veiga
INESC-ID Lisboa
Instituto Superior Técnico, Universidade de Lisboa
Lisbon, Portugal
Email: luis.veiga@inesc-id.pt

***Abstract*—Cloud Computing researches involve a tremendous amount of entities such as users, applications, and virtual machines. Due to the limited access and often variable availability of such resources, researchers have their prototypes tested against the simulation environments, opposed to the real cloud environments. Existing cloud simulation environments such as CloudSim and EmuSim are executed sequentially, where a more advanced cloud simulation tool could be created extending them, leveraging the latest technologies as well as the availability of multi-core computers and the clusters in the research laboratories. This research seeks to develop Cloud2Sim, a concurrent and distributed cloud simulator, extending CloudSim while exploiting the features provided by Hazelcast, Infinispan and Hibernate Search to distribute the storage and execution of the simulation.**

I. Introduction

Simulations empower the researchers with an effective and quicker way to test the prototype developments of their research. As cloud computing environments consist of data centers and applications distributed on a planetary-scale, cloud simulations are used for evaluating algorithms and strategies that are under research and development. While some of the simulators are general-purpose, others focus on a narrower domain. CloudSim [1], EmuSim [2], SimGrid [3], and GreenCloud [4] are some of the mostly used general-purpose cloud simulation environments. MDCSim [5] and DCSim [6] are simulators designed specifically for datacenter simulation. OverSim [7], PlanetSim [8], and PeerSim [9] are simulators for peer-to-peer and overlay networks. Many grid simulators such as SimGrid [3] evolved into cloud simulators, or have been extended into a cloud simulator.

Cloud simulation environments require a considerable amount of memory and processing power to simulate a complex cloud scenario. Processors are increasingly becoming more powerful with multi-core architectures. Computing clusters in the research laboratories themselves could be used to run complicated large simulations in a distributed manner, as in BOINC derivatives [10]. However, current simulation tools provide limited support to utilize these resources, as they are mostly written with a sequential execution model targeting to run on a single server. This work researches and implements a concurrent and distributed cloud simulator, named "Cloud2Sim", using the distributed shared memory provided by Hazelcast [11] and Infinispan [12], while exploiting the search capabilities offered by Hibernate search [13].

Originally developed as GridSim, a grid simulation tool, CloudSim was later extended as a Cloud Simulation environment. Initially having GridSim as a major building block [14], CloudSim was further developed as a cloud simulator on its own. Due to its modular architecture which facilitates customizations, it is extended into different simulation tools such as CloudAnalyst [15] and NetworkCloudSim [16]. Developed in Java, CloudSim is portable. CloudSim can be easily modified by extending the classes, with a few changes to the CloudSim core. Its source code is open and maintained. Hence, CloudSim was picked as the core module to build the distributed simulator.

In the remaining of the paper, we will discuss the preliminary background information on CloudSim in section II. Section III discusses design and implementation of Cloud2Sim, and how CloudSim is customized and extended to be a distributed and concurrent cloud simulator. Section IV presents some preliminary evaluation results. Section V closes the paper with some conclusions and ongoing efforts.

II. Background

A. CloudSim

CloudSim defines the parameters of the cloud environments such as hosts, VMs, applications, and datacenters by the instances of different classes. $Datacenter$ is the resource provider which simulates infrastructure as a service. Multiple hosts are created inside datacenters [17]. There should be at least one datacenter in the system, for CloudSim to start execution. $DatacenterBroker$ is responsible for application scheduling and coordinating the resources. Datacenter broker functions as the coordinating entity of resources and user applications. A single broker or a hierarchy of brokers can be initiated depending on the simulation scenario.

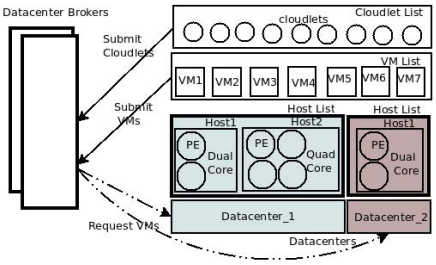

Fig. 1. CloudSim scheduling operations

Figure 1 shows how a cloud environment is represented by the architecture of CloudSim in a high level, focusing on the resource scheduling. CPU unit is defined by Pe (Processing Element) in terms of million instructions per second (MIPS).

1526-7539/14 $31.00 © 2014 IEEE
DOI 10.1109/MASCOTS.2014.70

Multi-core processors are created by adding multiple Pe objects to the list of processing elements. All processing elements of the same machine have the same processing power (MIPS). Processing elements are the shared resources. Cloudlets represent the applications that share these resources among them. Status of a processing element can be FREE (1), BUSY/Allocated (2), or FAILED (3), indicating its availability for the cloudlet.

Each of the VMs is assigned to a host. Each cloudlet is assigned to a VM, and the processing elements are shared among the VMs in a host and among the executing cloudlets in the VMs. Complicated real-world cloud scenarios can be simulated by appropriately extending the broker and the other classes. Virtual machines and cloudlets are created and added to the respective lists. Once the simulation is started, the list of cloudlets and virtual machines are submitted to the broker. The broker handles the allocation of VMs to the hosts and cloudlets to the VMs, and leads the simulation behavior such as deciding which of the available cloudlets to be executed next.

B. Hazelcast

Hazelcast provides the distributed implementations for the java.util.concurrent package. By extending the concurrent hashmaps, executor service, and other data structures to function in a distributed environment, Hazelcast facilitates a seamless development and deployment of a distributed execution environment. Computer nodes running Hazelcast can join or create a Hazelcast cluster using either multicast or TCP-IP. Multiple Hazelcast instances can also be created from a single node by using different ports, hence providing a distributed execution inside a single machine. The Hazelcast monitoring tool monitors the execution and the distribution of the data structures across the partitions, as well as the status and health of the nodes. As a distributed in-memory data grid, Hazelcast has been already used in researches, mostly to distribute the storage across multiple instances [18].

C. Infinispan

Infinispan is a distributed key/value data-grid [12]. When used as a cluster-aware data-grid over multiple nodes, Infinispan can execute applications that would not run on a single node/computer due to the limited availability of resources. By utilizing multiversion concurrency control, Infinispan permits concurrent readers and writers, opposed to the coarse grained Java concurrency control and synchronization. Hence, when used as a local in-memory cache, Infinispan outperforms ConcurrentHashMap.

While Infinispan can be used as a distributed cache for scaling the storage and execution out, fault-tolerance can be achieved with Infinispan as a replicated cache. Built-in eviction allows Infinispan to store huge objects that do not fit into the memory, by integrating with a persistency layer consisting of relational or NoSQL databases. Infinispan has been used in many researches, as an in-memory data-grid. Infinispan depends on two-phase commit based replication, which can further be made more efficient with partial replication techniques with weak consistency [19].

Hazelcast and Infinispan have similar functionality, and both can be used as an in-memory cache to build a Data-as-a-Service solution. While Infinispan has been optimized to function as a distributed as well as a local cache, Hazelcast targets mostly to be a distributed cache.

III. CLOUD2SIM

Cloud2Sim, our contribution, is designed to be run on top of a cluster. It uses Hazelcast and Infinispan to distribute the storage of VM, Cloudlet, and datacenter objects and also to distribute the execution to the instances in the cluster. Users have the freedom to choose Hazelcast based or Infinispan based distribution, customize to use both, or develop their own distribution methodology.

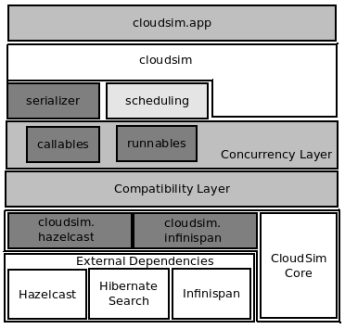

Fig. 2. Cloud2Sim Architecture

A. Architecture and Design

Figure 2 depicts a layered architecture of Cloud2Sim, hiding the fine architectural details of CloudSim. Classes of CloudSim are extended and a few are also modified. Hazelcast, Infinispan, and Hibernate Search are used unmodified. The packages $cloudsim.hazelcast$ and $cloudsim.infinispan$ respectively integrate Hazelcast and Infinispan into the simulator. $HazelSim$ is a singleton class that is responsible for initiating the Hazelcast clusters and ensuring that the minimum number of instances are present in the cluster before the simulation begins. Properties of the cluster such as whether the caching should be enabled in the simulation environment, and when the unused objects should be evicted from the instances are configured by $hazelcast.xml$. Hazelcast can also be configured programmatically for Cloud2Sim using $HazelSim$. $HzObjectCollection$ provides access to the distributed objects such as Hazelcast maps. $InfiniSim$ provides similar functionality for the Infinispan based distribution.

The compatibility layer enables the execution of the same CloudSim simulations, on top of either the Hazelcast and Infinispan based implementations, or the pure CloudSim distribution, by abstracting away the dependencies on Hazelcast and Infinispan. The concurrency layer consists of callables and runnables for asynchronous invocations to concurrently execute. As complex objects should be serialized before sending them to other instances over the wire, custom serializers were written for Vm, $Cloudlet$, $Host$, $Datacenter$, and the other distributed objects.

The $scheduling$ package provides enhancements to the existing application scheduling capabilities of CloudSim. Matchmaking-based scheduling algorithms have to search

through the object space to find a matching resource for the application requirements [20], [21]. Hibernate search is used to handle the scheduling in highly complex scenarios that involve searching large maps consisting of VMs, cloudlets, and the user requirements. An additional layer of $cloudsim.app$ on top of CloudSim provides user level utility functionalities, assisting the construction of simulations. $config.properties$ is used to input CloudSim specific parameters such as the number of resources and users to be present in the simulation, such that simulations can be run with varying loads, without recompiling. DatacenterBroker and Datacenter are extended to provide a distributed execution. Extended brokers and their interaction with the resources and cloudlets are depicted in Figure 3.

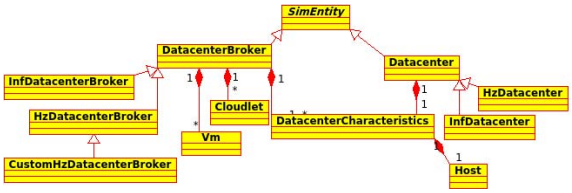

Fig. 3. Class Diagram of Cloud2Sim Brokers

B. Implementation Details

Distributing the simulation environment has been implemented using an incremental approach.

1) Distributed Storage: Initially, Hazelcast was just used to provide a distributed storage. An instance of the $Initiator$ class initiates a Hazelcast instance and keeps the node connected to the Hazelcast cluster. $Simulator$, the simulator instance is run from the master class, while an instance of $Initiator$ is run from the other instances. Instances of Hazelcast $IMap$ are used as the data structure. Hazelcast monitoring tool indicated equal partitioning of storage across all the instances. It was possible to execute the memory-hungry applications that would not run in a single node, as the required memory to store the objects exceeded the available memory in any single node in the cluster. However, distributing the complex VM and Cloudlet objects introduced communication and serialization costs.

2) Distributed Execution: Classes extending the $Runnable$ and $Callable$ interfaces are used to submit the VMs and Cloudlets concurrently. Further, Hazelcast $IExecutorService$ make the execution distributed. Initially, master and workers were implemented as different classes, named $Simulator$ and $SimulatorSub$. Later, the Simulator instances were unified, such that a same $Simulator$ class can be run from all the instances. The first instance to join the cluster becomes the master, where other instances will be the workers. Master executes the core fractions of the logic which should not be run parallel for a correct execution. The different approaches of distributed execution is depicted by Figure 4.

3) Data locality: Pulling data from each of the nodes for execution has a higher communication cost. To overcome this, Data locality features provided by Hazelcast are exploited to send the logic to the data instead. Callables and runnables are made to implement $HazelcastInstanceAware$ interface, where each member of the cluster executes part of the logic on the objects that are stored in the local partitions of the respective nodes.

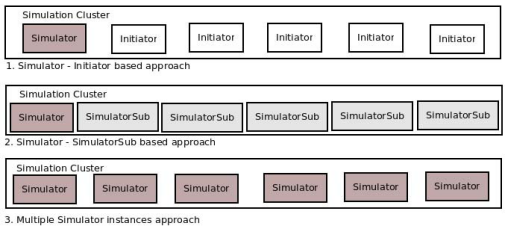

Fig. 4. Partitioning Approaches

TABLE I. EXECUTION TIME (SEC) FOR CLOUDSIM VS. CLOUD2SIM

nodes	Simple Simulation	Simulation with a workload
CloudSim	3.678	1247.400
Cloud2Sim (1 node)	20.914	1259.743
Cloud2Sim (2 nodes)	16.726	120.009
Cloud2Sim (3 nodes)	14.432	96.053
Cloud2Sim (6 nodes)	20.307	104.440

IV. EVALUATION

A cluster with 6 identical nodes (Intel(R) Core(TM) i7-2600K CPU @ 3.40GHz and 12 GB memory) was used for evaluations. Table I shows the time taken to simulate a round robin application scheduling with 200 users, 15 datacenters each with 20 hosts, 200 VMs, and 400 cloudlets. CloudSim outperformed Cloud2Sim in the base execution without an actual workload, due to the inherent overloads involved in Cloud2Sim. Cloud2Sim with 2 or 3 nodes showed a considerable improvement in the execution time when the cloudlets contained a relevant workload to be simulated once scheduled.

A fair matchmaking-based scheduling [20], [21] scenario is implemented to depict a practical use case of distributed execution of simulations. While other parameters are kept constant as in the previous scenario, the number of cloudlets was changed. The workload of this execution is a matchmaking-based cloudlet scheduling. Each cloudlet and VM has a variable length or size. Each cloudlet requires the executing VM to have a minimal size, which is a function of the cloudlet length. Cloudlets search the object space to find the best fit for this specification, and bind themselves to the VM that is the best fit. While ensuring that the minimal specifications are met, cloudlets also ensure fairness, by not binding to a VM that is much larger than their specification size. This avoids overloading the large VMs, and schedules a fair share of cloudlets to the VMs that they are bound to, in a round robin manner.

Figure 5 depicts the time taken for simulations with different number of cloudlets to complete the scheduling, with multiple nodes. Execution time for CloudSim was almost the same as the simulation time in a single node in Cloud2Sim, except when the simulation size was very small. As the simulation size becomes larger with large number of VMs and cloudlets, simulation time grows exponentially due to the increasing search and matchmaking space, when running on a single instance. This exponential growth is handled and mitigated when running on multiple instances, as the execution is evenly distributed across the instances.

As more instances are added, simulation performs faster. The performance gain, or the percentage improvement in the simulation time for the multiple instances is shown by Figure 6.

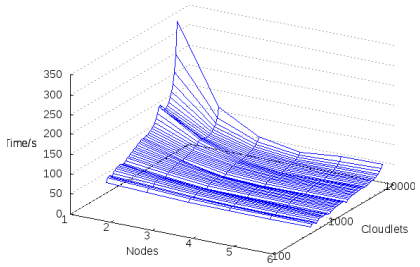

Fig. 5. Simulation Time for Matchmaking-based scheduling

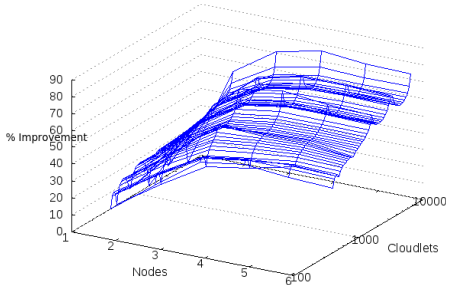

Fig. 6. Speedup - Percentage Improvement of the Distributed Execution

As shown by the simulation experiments, Cloud2Sim provides a considerable performance gain to the simulations, compared to their serial execution. It also exhibits a positive scalability for larger simulations, handling the simulations effectively through a distributed execution.

V. Conclusion

Simulation tools try to portray a geo-distributed decentralized environment using network and topology simulation code that is serial and manipulating a large global state that is considered consistent. Hence, their performance is far from ideal. Cloud2Sim attempts to address this concern by extending the existing CloudSim cloud simulator, utilizing distributed shared memory projects. A Hazelcast based Cloud2Sim implementation has already been completed, and is being fine-tuned and benchmarked against CloudSim for multiple scenarios. Infinispan and Hibernate Search based distributed simulations are currently researched and designed. Cloud2Sim[1] scales reasonably well, and distributes the storage and execution almost equally among all the instances. Upon completion, Cloud2Sim will have the advantages of CloudSim, while being efficient, faster, customizable, and scalable.

Acknowledgments: This work was partially supported by national funds through FCT - Fundação para a Ciência e a Tecnologia, under project PEst-OE/EEI/LA0021/2014.

References

[1] Calheiros, R.N., Ranjan, R., De Rose, C. A. F. and Buyya, R. "CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services," *Technical Report, GRIDS-TR-2009-1, Grid Computing and Distributed Systems Laboratory,* The University of Melbourne, Australia.

[2] Calheiros, R. N., Netto, M. A., De Rose, C. A., and Buyya, R. "EMUSIM: an integrated emulation and simulation environment for modeling, evaluation, and validation of performance of cloud computing applications," *Software: Practice and Experience, 00-00* 2012.

[3] Casanova, H. "Simgrid: A toolkit for the simulation of application scheduling," in *Cluster Computing and the Grid, 2001. Proceedings. First IEEE/ACM International Symposium*, 2001 (pp. 430-437).

[4] Kliazovich, D., Bouvry, P., and Khan, S. U. "GreenCloud: a packet-level simulator of energy-aware cloud computing data centers," in *The Journal of Supercomputing, 62*(3), 2012, pp. 1263-1283.

[5] Lim, S. H., Sharma, B., Nam, G., Kim, E. K., and Das, C. R. (2009, August). MDCSim: A multi-tier data center simulation, platform. In Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on (pp. 1-9). IEEE

[6] Tighe, M., Keller, G., Bauer, M., and Lutfiyya, H. "DCSim: A data centre simulation tool for evaluating dynamic virtualized resource management," in *Network and Service Management (CNSM), 2012 8th International Conference*, 2012 (pp. 385-392).

[7] Baumgart, I., Heep, B., and Krause, S. "OverSim: A flexible overlay network simulation framework", in *IEEE Global Internet Symposium*, 2007 (pp. 79-84).

[8] García, P., Pairot, C., Mondéjar, R., Pujol, J., Tejedor, H., and Rallo, R. (2005). Planetsim: A new overlay network simulation framework. In Software engineering and middleware (pp. 123-136). Springer Berlin Heidelberg.

[9] Montresor, A., and Jelasity, M. "PeerSim: A scalable P2P simulator," in *Peer-to-Peer Computing, 2009. P2P'09. IEEE Ninth International Conference*, 2009 (pp. 99-100). IEEE September.

[10] Silva, J. N., Veiga, L., and Ferreira, P. (2008, October). nuboinc: Boinc extensions for community cycle sharing. In Self-Adaptive and Self-Organizing Systems Workshops, 2008. SASOW 2008. Second IEEE International Conference on (pp. 248-253). IEEE.

[11] Johns, M. *Getting Started with Hazelcast. Packt Publishing Ltd*, 2013.

[12] Marchioni, F. *Infinispan Data Grid Platform. Packt Publishing Ltd*, 2012.

[13] Bernard, E., and Griffin, J. Hibernate search in action. Manning, 2009.

[14] Calheiros, R.N., Ranjan, R., De Rose, C. A. F. and Buyya, R. "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", *Softw. Pract. Exper. 2011; 41*:23-50. Published online 24 August 2010 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/spe.995.

[15] Wickremasinghe, B., Calheiros, R. N., and Buyya, R. "Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications," in *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference*, 2010, (pp. 446-452).

[16] Garg, S. K., and Buyya, R. "NetworkCloudSim: modelling parallel applications in cloud simulations," in *Utility and Cloud Computing (UCC), Fourth IEEE International Conference*, 2011, (pp. 105-113).

[17] Buyya, R., Ranjan, R., and Calheiros, R. N. (2009, June). Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities. In High Performance Computing & Simulation, 2009. HPCS'09. International Conference on (pp. 1-11). IEEE.

[18] Pachori, V., Ansari, G., and Chaudhary, N. "Improved Performance of Advance Encryption Standard using Parallel Computing".

[19] Ruivo, P., Couceiro, M., Romano, P., and Rodrigues, L. (2011, December). Exploiting total order multicast in weakly consistent transactional caches. In Dependable Computing (PRDC), 2011 IEEE 17th Pacific Rim International Symposium on (pp. 99-108). IEEE.

[20] Raman, R., Livny, M., and Solomon, M. "Resource management through multilateral matchmaking," in *High-Performance Distributed Computing, 2000. Proceedings. The Ninth International Symposium*, 2000, (pp. 290-291).

[21] Raman, R., Livny, M., and Solomon, M. "Matchmaking: Distributed resource management for high throughput computing," in *High Performance Distributed Computing, 1998. Proceedings. The Seventh International Symposium*, 1998 (pp. 140-146).

[1]The source code can be accessed from https://sourceforge.net/p/cloud2sim/code/ci/master/tree/, with user name, "cloud2sim" and password, "Cloud2Simtest".