Journal of Systems Architecture 000 (2016) 1–15

Contents lists available at ScienceDirect

Journal of Systems Architecture

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

IOTSim: A simulator for analysing IoT applications

Xuezhi Zeng[a,*], Saurabh Kumar Garg[b], Peter Strazdins[a], Prem Prakash Jayaraman[c], Dimitrios Georgakopoulos[c], Rajiv Ranjan[d]

[a] Australian National University, Research School of Computer Science, Canberra, ACT 2914, Australia
[b] University of Tasmania, Australia
[c] RMIT, Melbourne, Australia
[d] Newcastle University, UK

ARTICLE INFO

Article history:
Received 25 January 2016
Revised 9 June 2016
Accepted 30 June 2016
Available online xxx

Keywords:
Internet of things (iot)
Big data
Mapreduce
Cloud computing
Programming model
Modelling and simulation

ABSTRACT

A disruptive technology that is influencing not only computing paradigm but every other business is the rise of big data. Internet of Things (IoT) applications are considered to be a major source of big data. Such IoT applications are in general supported through clouds where data is stored and processed by big data processing systems. In order to improve the efficiency of cloud infrastructure so that they can efficiently support IoT big data applications, it is important to understand how these applications and the corresponding big data processing systems will perform in cloud computing environments. However, given the scalability and complex requirements of big data processing systems, an empirical evaluation on actual cloud infrastructure can hinder the development of timely and cost effective IoT solutions. Therefore, a simulator supporting IoT applications in cloud environment is highly demanded, but such work is still in its infancy. To fill this gap, we have designed and implemented IOTSim which supports and enables simulation of IoT big data processing using MapReduce model in cloud computing environment. A real case study validates the efficacy of the simulator.

© 2016 Elsevier B.V. All rights reserved.

1. Introduction

According to a study by IBM, we are creating 2.5 quintillion (2.5×1018) bytes of data every day as of 2012 through different sensing devices [1, 2]. IDC (International Data Corporation) predicts that from 2005 to 2020, the digital universe will grow by a factor of 300, from 130 exabytes to 40,000 exabytes, or 40 trillion gigabytes (more than 5200 gigabytes for every man, woman, and child in 2020). From 2012 until 2020, the digital universe will double every two years. It can be declared that we are in the era of "Big Data" which is accelerated by the Internet of Things (IoT) [3]. Such "Data Explosions" have led to the next grand challenge in computing known as the 'Big Data' problem [4-7], which is defined as the practice of collecting and analysing structured and unstructured data sets flowing at a volume and velocity that is too large and too fast to store, process, and interpret manually or using traditional data management applications.

Gartner forecasted that there will be nearly 50 to 100 billion devices and sensors in the Internet of Things ecosystem by 2020 [8]. Once all these devices and sensors are connected with each other, IoT will enable more new and innovative applications that support not only our daily basic needs but also solve economic, environmental and health problems. Such enormous number of devices connected to internet provide many kinds of services and generate Big Data [9] that needs to be processed and analysed for knowledge extraction. Due to their intrinsic nature, IoT applications require lots of IT resources if users want fast analysis of their large datasets. Thousands of CPUs, hundreds of terabytes of storages and very high speed interconnections are demanded. In order to support these IoT applications, a reliable, elastic and agile platform is essential. Cloud computing is one of the enabling platforms to support IoT applications.

Cloud computing [10–13] is a model for on-demand access to a shared pool of configurable resources (e.g. compute, networks, servers, storage, applications, services, and software) that can be easily provisioned by three commonly deployed cloud service models namely Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS). For example, IaaS can be used to implement custom gateway interfaces to support IoT devices or sensors. Consumers can set up arbitrary services and manage the devices or sensors via cloud resource access control.

* Corresponding author.
E-mail addresses: xuezhi.zeng@anu.edu.au (X. Zeng), saurabh.garg@utas.edu.au (S.K. Garg), peter.strazdins@anu.edu.au (P. Strazdins), prem.jayaraman@rmit.edu.au (P.P. Jayaraman), dimitrios.georgakopoulos@rmit.edu.au (D. Georgakopoulos), raj.ranjan@ncl.ac.uk (R. Ranjan).

http://dx.doi.org/10.1016/j.sysarc.2016.06.008
1383-7621/© 2016 Elsevier B.V. All rights reserved.

PaaS can provide a platform to access IoT data and on which custom IoT applications (or host-acquired IoT applications) can be developed. SaaS can be provided on top of the PaaS solutions to offer the provider's own SaaS platform for specific IoT domains such as smart city, healthcare, video surveillance etc.

Therefore, It is well understood that cloud computing platforms are well suited for hosting IoT applications as they offer an elastic hardware resources (e.g. CPU, Storage, and Network) that can be scaled on-demand for handling large quantities of data from IoT applications with uncertain volume, variety, velocity, and query types [14-16]. These two technologies are inherently and increasingly getting entwined with each other. Because of this, evaluation and analysis of IoT applications in a real cloud computing environment can be a challenge for several reasons:

- It is not cost-effective to procure or rent a large scale datacentre resource pool that will accurately reflect realistic application deployment and let practitioners experiment with dynamic hardware resource and big data processing framework configurations, and changing data volume, velocity, and variety
- Frequently changing experiment configurations in a large-scale real test bed involves lot of manual configuration, making the performance analysis itself time-consuming. As a result, the reproduction of results becomes extremely difficult
- The real experiments on such large-scale distributed platform are sometimes impossible due to multiple test runs in different conditions
- It is almost impractical to set up a very large cluster consisting hundreds or thousands of nodes to test the scalability of the system

An obvious solution to the aforementioned problems is to use a simulator supporting IoT application processing. A simulator not only allows us to measure scalability of computing resources for IoT applications efficiently, but also enables to determine the effects of various independent variables (i.e., datacentre configuration, Virtual Machine (VM) configuration, VM number, job configuration, MapReduce (MR) combination) on different dependent variables (i.e., average execution time, maximum execution time, minimum execution time, make span, VM computation cost, network cost) which will be detailed in Section 5.2 and Section 5.3 respectively. Thus, IoT simulator will be a very useful tool to facilitate both researchers and commercial entities equally to analyse, test and design IoT applications with far less cost and time.

As the Cloudsim simulation software is the best choice to simulate cloud computing resources [17], we have designed and implemented a simulator called IOTSim on top of Cloudsim, where we can simulate the behaviour of IoT applications utilizing MapReduce framework to process the big data generated from different sensing devices. The key contributions of IOTSim lie in extending Cloudsim with 1) IoT application model support and 2) enabling processing of IoT data using big data system (i.e., MapReduce) in Cloud Computing environment. The proposed simulator also allows modelling and simulation of network usage between storage and processing virtual machines, and between individual VM.

The rest of this paper is organized as follows: Section 2 presents the general IoT architecture with its definition described and discusses the requirements for modelling IoT-based applications within a simulator. Section 3 conducts an extensive literature review of simulators in cloud computing environment and those simulators that specifically targets the MapReduce model. Section 4 details the design and implementation of the proposed IOTSim simulation framework. In Section 5, simulation results to show the efficacy of the proposed simulation tool are discussed. Section 6 concludes the paper and points out some future work.

2. Requirement for modelling IoT-based applications

The Internet of Things (IoT) is a network of networks, in which objects, animals or people are provided with unique identifiers and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction [18]. The IoT allows people and things to be connected anytime, anyplace, with anything and anyone through the information and communications infrastructure to provide value-added services [8]. The IoT has evolved from the convergence of wireless technologies, microelectromechanical systems (MEMS) and the Internet. In a general way, IoT is formed by three layers [19-21].

- Physical Layer:
  - Perception layer: which is the bottom layer whose function is to gather and transform data to readable digital signals with RFID, sensors, etc. All the data collection and data sensing part is done on this layer [22].
  - Network layer: a middle layer which collects the data perceived by the perception layer and sends digital signals to corresponding platforms via network. This layer may only include a gateway, having one interface connected to the sensor network and another to the Internet.
- Virtual Layer: represents the cyber representation of the physical world entities. In most cases, the virtual layer is deployed on cloud computing infrastructure eliminating the need for owning, housing and maintaining computing resources. It leverages a combination of advanced batch and streaming processing technologies to provide useful analytical insight for different type of IoT applications that have historical and real time requirements.
- Application layer: is on the top layer, which performs the final presentation of data. Application layer receives information from the lower layer and provides global management of the application presenting that information. According to the needs of user, Application layer presents the data in the form of: smart city, healthcare, video surveillance and other many kinds of applications [23].

A typical relationship of IoT-based applications and cloud computing is shown in Fig. 1. IoT applications currently require a combination of batch and streaming data across cloud resources [24]. One of the well-known and established batch processing technology is MapReduce which is a distributed parallel computing framework [25]. Typical implementations of the MapReduce model include Disco [26], Mars [27], Phoenix [28], Hadoop [29] and Google's implementation [30]. Among them, Hadoop, which is inherently designed for batch and high throughput processing jobs, has proven itself as the de facto solution to big data processing. In this version of the proposed IoT simulator, we have implemented the batch processing requirements of an IoT application.

2.1. Big data processing platforms: mapreduce

The nature of the IoT, or properly speaking, of the data that IoT devices generate, leads to the "big data approach" . This is the idea of running data processing in a scale-out fashion on commodity hardware, using distributed data processing framework such as MapReduce for data incentive IoT-based applications.

MapReduce is a predominant framework for large-scale distributed data processing based on the divide and conquer paradigm [30]. MapReduce works by breaking the processing into map and reduce phases. Map task and reduce task are executed in parallel on the different machines within the Hadoop cluster by MapReduce framework. Map performs filtering and sorting operations, and reduce performs summary operations. The user can specify map/reduce functions, and types of input/output.

Please cite this article as: X. Zeng et al., IOTSim: A simulator for analysing IoT applications, Journal of Systems Architecture (2016), http://dx.doi.org/10.1016/j.sysarc.2016.06.008

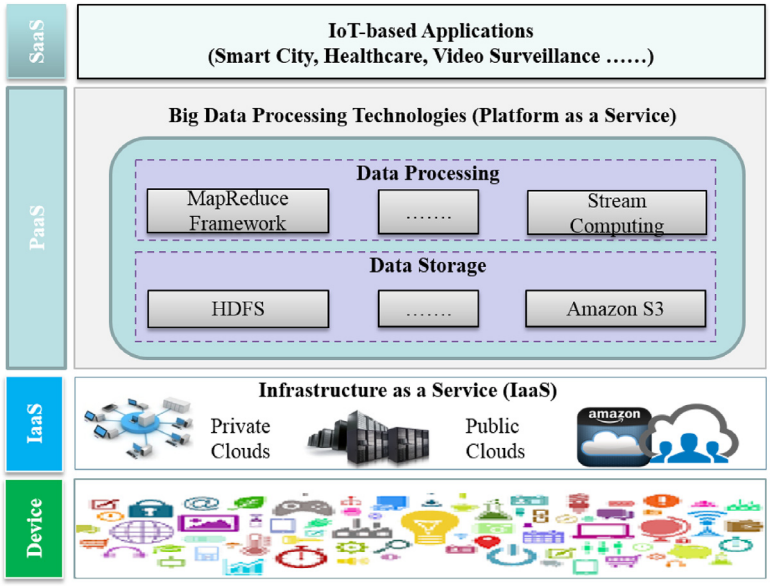


Fig. 1. Relationship of IoT-based application and cloud computing.

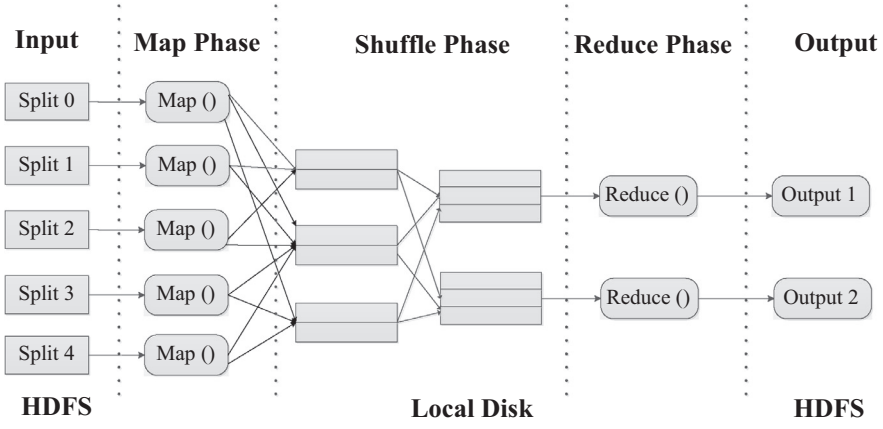


Fig. 2. The Structure of MapReduce model.

Fig. 2 illustrates the overall process of MapReduce. Input data stored on the Hadoop Distributed File System (HDFS) are split into fixed-size blocks, and each block is allocated to a map. Then user-specified map processes each key-value pair in the block; and outputs the result as a list of key-value pairs. The output of the map is partitioned by the key, and the grouped records are stored in the local disk and transferred to the different reducers, respectively (called shuffle). Then, the transferred records are merged and sorted in the node where a reduce task performs. Each reduce task sequentially reads key-value pairs, and processes them by the user-specified reduce function. Finally, the output records of the reduce task are written to the HDFS.

Fig. 3 presents the high level workflow of how a MapReduce job is executed in Hadoop. Specifically, MapReduce processing in Hadoop is handled by the JobTracker and TaskTracker daemons. The JobTracker maintains a view of all available processing resources in the Hadoop cluster and, as application requests come in, it schedules and deploys them to the TaskTracker nodes for execution. As applications are running, the JobTracker receives status updates from the TaskTracker nodes to track their progress. The JobTracker needs to run on a master node in the Hadoop cluster as it coordinates the execution of all the incoming jobs. An instance of the TaskTracker daemon runs on every slave node in the Hadoop

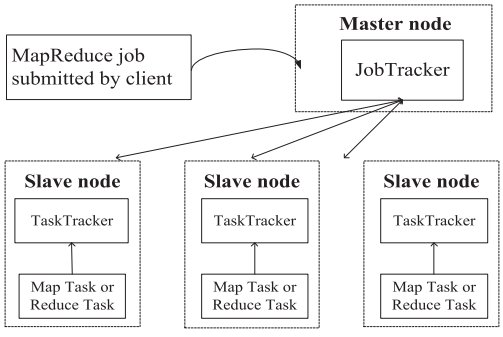


Fig. 3. The MapReduce workflow in Hadoop.

cluster, which means that each slave node has a service that ties it to the processing (TaskTracker) and the storage (Data Node), which enables Hadoop to be a distributed system. As a slave process, the TaskTracker receives processing requests from the JobTracker. Its primary responsibility is to track the execution of MapReduce workloads happening locally on its slave node and to send status updates to the JobTracker.

Please cite this article as: X. Zeng et al., IOTSim: A simulator for analysing IoT applications, Journal of Systems Architecture (2016), http://dx.doi.org/10.1016/j.sysarc.2016.06.008

2.2. Requirements for modelling IoT-based applications

There are three main issues in supporting the modelling of IoT-based applications within a Cloud simulator. The first issue is from the perspective of application configuration and modelling, and another is from cloud service providers' perspective in terms of processing and network. In the following sections, we discuss these issues and provide features that allow support of IoT-based applications for simulation of cloud computing environments.

2.2.1. Application requirement

As it can be noticed from the previous section, an IoT-based application generally processes large data sets stored in clouds after being collected from different devices. The simulator should thus allow modelling of different IoT-based applications depending on used big data processing platforms such as MapReduce. For example, a MapReduce-compatible IoT application may consist of one or more jobs that will split into a specific number of chunks of data. These chunks are processed as map tasks at the beginning, and thereafter, the intermediate output of map task is processed by the corresponding reduce task.

2.2.2. Big data processing requirement

To support IoT applications, big data processing technologies plays a central role. Hence, it is mandatorily required for the proposed simulator to meet the big data processing requirement. Depending on various IoT-based applications, the simulator should offer the capability that uses different big data processing technology to support batch processing or stream processing on the big data. Also, it should allow modelling and simulating the execution of multiple jobs simultaneously in a scalable manner as it happens in the real world.

2.2.3. Network and processing infrastructure requirements

An IoT-based application requires different types of storage that are commonly ambient in cloud-based data centres to store content from the various devices, thus, a storage layer should be modelled to simulate storage (such as Amazon S3, Azure Blob Storage, Hadoop HDFS) and retrieval of any amount of data, subject to the availability of network bandwidth. It is obvious that accessing files in storages at run-time incurs additional delay for IoT-based application execution. This is due to the latencies between the nodes and storages when transferring the data files through the IoT network. Hence, the design of network between hosts and storage is required to model the aforementioned delay.

3. Related works

Over the last decades, many simulation frameworks have been developed to facilitate researches on the behaviour of large-scale distributed systems for hosting various application services (e.g., social networking, web hosting, scientific applications, and content delivery). It is well understood that simulators offer an environment where performance evaluation studies can be conducted in a repeatable and controllable manner.

To the best of our knowledge, there is no simulator available which specifically targets the IoT environment. However, there are some closely related in terms of cloud simulator and MapReduce simulator.

3.1. Cloud simulator

Popular cloud simulators that are capable of simulating and modelling distributed system are typically classified into the following categories:

- Grid Computing: the typical simulation toolkits are GridSim [31], MicroGrid [32], GangSim [33], SimGrid [34], and OptorSim [35].
- Peer-to-Peer network models such as structured and unstructured overlay networks were simulated in PlanetSim [36]. In one study [37], PlanetSim was integrated with GridSim for evaluating the performance of decentralized and coordinated scheduling of scientific applications across multiple computational sites (clusters, supercomputers, etc.).
- Cloud computing model were simulated in GreenCloud [38], iCanCloud [39], Cloudsim [40] and its variants (CloudAnalyst [41], NetworkCloudsim [42], EMUSIM [43], MDCSim [44]) has been described and compared [45].
  - GreenCloud, which is a packet-level simulator (developed by extending NS-2) is capable of modelling behaviours of network links, switches, gateways, and other hardware resources (CPU and storage) in a cloud datacentre. The goal of this simulator is to simplify performance tests of energy-aware scheduling algorithms in cloud environments. GreenCloud is a packet-level simulator hence it requires extra memory and processing power to create and transmit packets across simulation entities.
  - iCanCloud: It is a simulation platform which is oriented towards the simulation of a wide range of Cloud Computing systems and their underlying architecture. It has the ability to model and simulate large environments (thousands of nodes) and distributed applications with a customizable level of detail.
  - Cloudsim is one of the widely used discrete event (its definition is detailed in Section 4.3) simulation frameworks as it is highly extensible and flexible. It provides models for all hardware resources including CPUs (virtual machine), storage and networks (network contention and delays) within multiple datacentres. Cloudsim has extensive support for application (e.g., scientific and web hosting applications) scheduling level simulation, as it provides cloud broker and cloud exchange (for federated datacentre resource pooling) entities.

3.2. MapReduce simulator

Further to the above cloud simulators, some researchers designed and implemented the simulation tools specifically targeted for MapReduce framework. Such MapReduce simulators include:

- MRPerf [46], which can serve as a design tool for MapReduce infrastructure and can help in designing new high performance MapReduce setup, and in optimizing existing ones. However, it cannot simulate complete behaviour of a Hadoop framework and [47] claimed that accurate results for jobs of different type of algorithms or different cluster configurations cannot be generated based on testing they performed on the MRPerf code.
- Mumak [48]: It is an open source Apache's MapReduce simulator which uses data from real experiments to estimate the completion time for Map and Reduce tasks with different scheduling algorithms. In cases where data from real experiments do not exist, Mumak cannot estimate completion time for Map and Reduce tasks.
- SimMR [49]: was developed in HP lab. It can replay execution traces of real workloads collected in Hadoop clusters (as well as synthetic traces based on statistical properties of workloads) for evaluating different resource allocation and scheduling ideas in MapReduce environments.
- MRSim [47]: It is a discrete event based MapReduce simulator. It is able to simulate different type of MapReduce applications with the ability to study with good accuracy the effect of

Please cite this article as: X. Zeng et al., IOTSim: A simulator for analysing IoT applications, Journal of Systems Architecture (2016), http://dx.doi.org/10.1016/j.sysarc.2016.06.008

dozens of job configuration parameters on the job performance. However, it was modelled and simulated using SimJava discrete event engine that has intrinsic weakness such as increased kernel complexity [50] and lack of support of some advanced operations [40]. Because of this, the SimJava layer has been removed from Cloudsim 2.0 onwards.
- MR-Cloudsim [51] was developed (by extending Cloudsim) for simulating MapReduce big data processing model. However, MR-Cloudsim has several limitations as it only supports simplistic, single-state Map and Reduce computation. Further, it lacks support for network link modelling, which is a critical element affecting the performance of MapReduce applications. Also, there is lack of support for allowing multiple MapReduce applications.

Although, the aforementioned two types of simulators were widely adopted in the study of the behaviours of cloud computing and MapReduce in distributed computing environment, they obviously lack the support of modelling and simulation for IoT applications. In contrast, the proposed IOTSim focuses on simulating IoT environment and aims to offer the following advantages compared to those existing simulators.

- support for simulation of IoT big data processing using MapReduce model or steam model in cloud computing environment
- support for modelling and simulation of large scale multiple IoT applications to run simultaneously in a shared cloud data centres
- support for modelling network and storage delays existing in the processing of IoT applications

4. Design and implementation of IOTSIM

Cloudsim [40], is an extensible simulation toolkit that enables modelling and simulation of cloud computing environments and application provisioning. It has many features, which make us choose it for building our simulator for analysing Iota Application. To be specific, Cloudsim supports modelling and creation of one or more Virtual Machines (VMs) on a simulated node of a datacentre with different hardware configurations, cloud-based tasks and their mapping to suitable VMs. It also allows simulation of multiple datacentres to enable a study on federated and associated policies for migration of VMs for reliability and automatic scaling of applications. In addition, Cloudsim helps in modelling user applications having independent jobs, and design and analysis of different hardware configurations, VM provisioning and scheduling policies. Hence, Cloudsim can pave the way for us to design and implement our simulation tool specific for IoT-based applications. In fact, Bashar [17] had done a critical evaluation on various cloud computing simulators and his study has concluded that Cloudsim is the best choice if research has to be done by using a simulation software. In the following section, we will details how we design and implement IOTSim by extending Cloudsim.

4.1. Proposed architecture

Illuminated by the works of Cloudsim [40], our IOTSim simulator is designed using the layered architecture with support for big data processing framework. Fig. 4 shows components of the Cloudsim architecture with the key elements of IOTSim (shown by dark boxes). In this section, we outline the general layered architecture of IOTSim. The detailed design and functionality of our proposed IOTSim's components will be discussed in the later sections.

- Cloudsim Core Simulation Engine Layer: the bottommost layer, which is a simulation engine that supports several core functionalities, such as queuing and processing of events, creation of Cloud system entities (services, host, datacentre, broker, and virtual machines), communication between components, and management of the simulation clock.
- Cloudsim Simulation Layer: this layer provides support for modelling and simulation of virtualized Cloud-based datacentre environments including dedicated management interfaces for virtual machines (VMs), memory, storage, and bandwidth. The fundamental issues such as provisioning of hosts to VMs, managing application execution, and monitoring dynamic system state are handled by this layer. This layer consists of several sublayers that model the core elements of Cloud Computing. The bottommost sublayers model datacentre, cloud coordinator, and network topology. These components help in designing Infrastructure-as-a-Service (IaaS) environments. The VM Services and Cloud Services provide the functionality to design resource (VM) and management and application scheduling algorithms.
- Storage Layer: this layer supports modelling different type of storage such as Amazon S3, Azure Blob Storage, and HDFS etc. where large datasets generated from devices are stored. In run time, IoT-based applications copy the data files from these storages, and write the intermediate data files to these storages when need. A storage delay will be incurred at this layer.
- Big Data Processing Layer: it includes two sub-layers. MapReduce sublayer is to support applications where a batch-oriented data processing paradigm is required while Streaming Computing sublayer aims to support applications that need a real-time processing paradigm. Depending on which IoT-based applications the customer will use, it can support processing for the big data generated from IoT devices or sensors using MapReduce or streaming computing model. Due to the limitation of Cloudsim mentioned in Section 3, the Big Data Processing Layer has been highly demanded in order to simulate and analyse IoT-based Applications. Take the MapReduce-compatible applications as an example, a MapReduce model needs to be fully implemented here where a set of new classes or entities such as JobTracker, TaskTracker, Mapper and Reducer work as does the real Hadoop and a series of events occur in some specific order to finish a Map/Reduce process. This layer plays an integral role in support big data processing towards IoT-based applications.
- User Code Layer: the top-most layer which exposes basic entities for hosts (number of machines, their specification and so on), IoT-based applications' configurations (Job Length and their requirements), VMs, number of users and their application types, and broker scheduling policies. This layer helps users to define their own simulation scenarios and configurations for validating their algorithms.

4.2. Design considerations

4.2.1. Application model

The application models in IoT environment can vary from building and home automation to wearables applications areas. The current generation of IoT applications (such as smart city, smart healthcare, and video surveillance) combines multiple independent data analytics models, historical data repositories, and real-time data streams that are likely to be available across geographically distributed datacentres (both private and public). Typically, such applications need to process large amounts of data by using parallel big data processing technologies such as MapReduce.

Most simulators for cloud computing generally offer limited support for modelling execution of parallel and distributed applications. For example, Cloudsim allows modelling an application service or a cloud task by using a programming structure called "Cloudlet", which only represents single and atomic computation

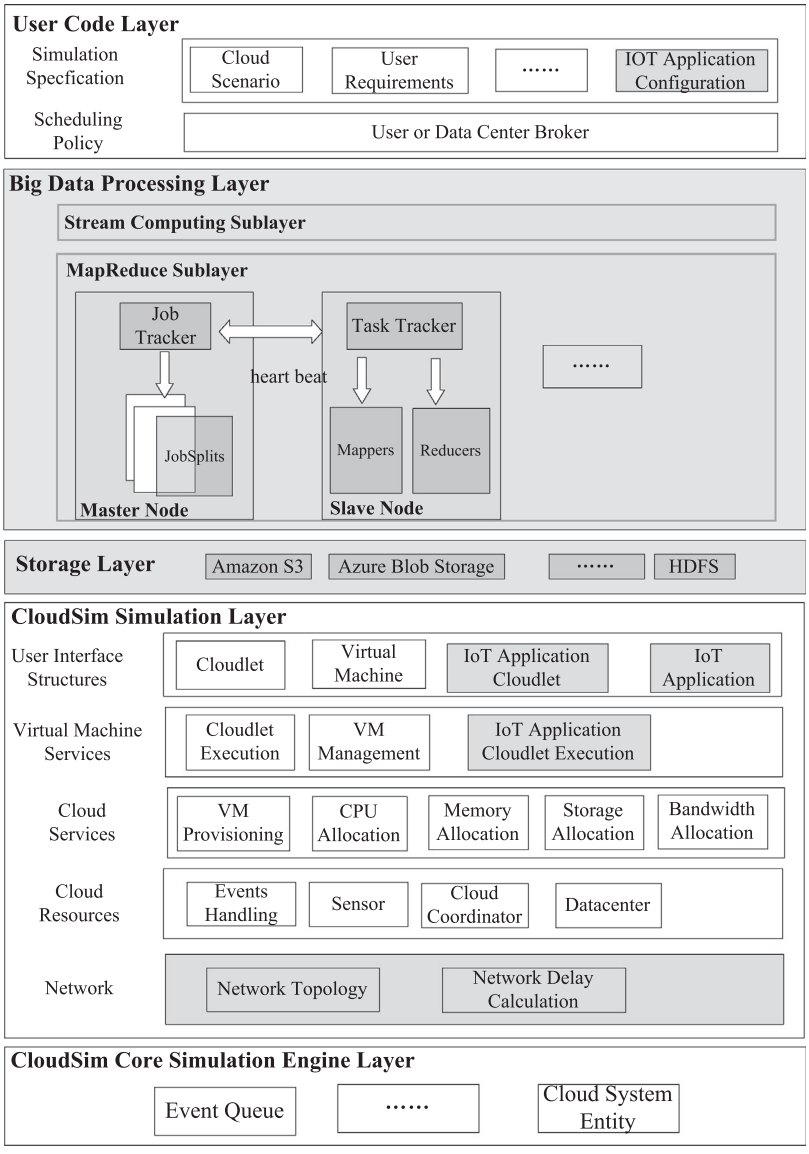


Fig. 4. The proposed layered architecture of IOTSim.

needs. This approach is obviously not appropriate for IoT applications scenarios. In order to support IoT applications simulation, one approach that could be used is to introduce two types of Cloudlet, one is MapCloudlet, and the other is ReduceCloudlet. The details of the two types of Cloudlets are described in the next Section 4.3.1. Both of them inherits from Cloudsim Cloudlet and has their own specific attributes and will be paired during the simulation. Also, ReduceCloudlet always runs after MapCloudlet of the same input.

4.2.2. Big data processing model

When dealing with IoT-based applications, parallel big data processing system has becomes the key to IoT applications. MapReduce, as a predominant big data processing framework is largely utilized by IOT applications. In Cloudsim, when Cloudlets come in, they are simply submitted to the cloud datacentre via broker. However, it doesn't suit for submitting MapReduce job in the same way due to the complexity of MapReduce workflow. Since JobTracker and TaskTracker play an integral part during the MapReduce processing as depicted in the above section, the IOTSim simulator needs to fully model and implement them as well as other notable features that MapReduce model has. Such features include: (1) there will be lots of communication and interaction between them on top of Cloudsim during the execution; (2) every separated Map task output has one Reduce operation; (3) Reduce operation has to come after Map operation of corresponding input; (4) multiple MapReduce jobs can be submitted and run simultaneously in a shared cloud-based big datacentres.

4.2.3. Network and storage model

As mentioned, a large amount of data that are generated from devices or sensors are stored in the Storage Layer. In runtime environment, a Map instance (mapper) operates in each slave node. It copies data which is saved in the above Storage Layer to its own local hard disk. When the data is copied and saved in the local hard disk, mapper starts processing the allocated map task by TaskTracker. Thereafter, the intermediate output will be generated and are associated with the Reduce instance. The paired reducer reads the intermediate output and starts processing the allocated reduce task. The final output will be written into the Storage Layer. Therefore, there are two typical network delay incurred that affect the performance of map or reduce task. In this scenario, network and storage model must be represented in a IOTSim simulator. One

Please cite this article as: X. Zeng et al., IOTSim: A simulator for analysing IoT applications, Journal of Systems Architecture (2016), http://dx.doi.org/10.1016/j.sysarc.2016.06.008

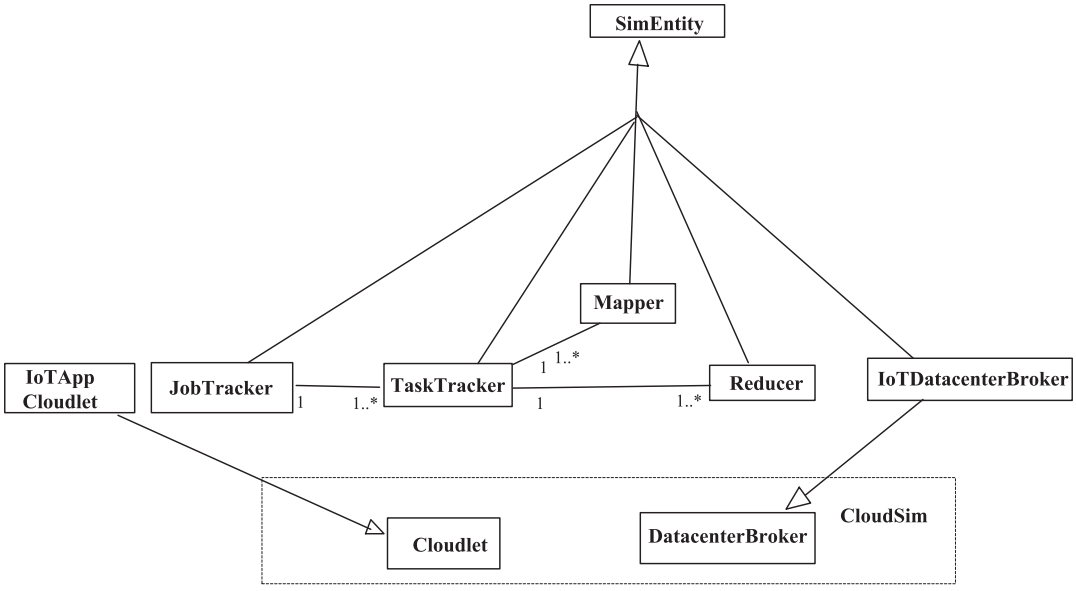


Fig. 5. Class diagram of IOTSim.

of feasible methods is to calculate the network consumption when copying data from Storage Layer by mapper or copying intermediate output from local disk by reducer. It is enough to guarantee the accuracy of simulator while calculating and presenting the network delay incurred during the processing.

4.3. Design of entities/classes

We would like to introduce the core components of Cloudsim before we present the design of IOTSim. There are two main parts in Cloudsim simulation: entity and event:

- Entity: refers to something which can individually and independently exists. It is able to send messages to other entities and process received messages as well as trigger and handle events. Each entity is initiated at the beginning and shutdown at the end during the simulation.
- Event: represents a simulation event which is passed between the entities in the simulation. Each event carries all the related information about an event between two or more entities such as event type, init time, time at which the event should occur, finish time, time at which the event should be delivered to its destination entity, the source entity and the destination entity as well as the data that has to be passed to the destination entity. Since Cloudsim is a discrete event-driven simulator, it is dependent of the series of events that are generated in some specific order. Without this order the simulation is impossible.

4.3.1. Application modelling design

To model the MapReduce for IoT-based applications, the following classes have been designed.

- MapCloudlet: This class, which is inherited from Cloudlet, models the atomic map task which will be submitted by DatacenterBroker and executed in VM. It extends Cloudlet with specific attributes.
- ReduceCloudlet: This class, which is inherited from Cloudlet, models the atomic reduce task which will be submitted by DatacenterBroker and executed in VM. It extends Cloudlet with specific attributes.

4.3.2. MapReduce modelling design

To model a MapReduce process and behaviour within a datacenter, the following classes/entities have been added to the IOTSim.

- JobTracker: represents an entity which receives the jobs summited by user, gets the data from storage, splits the jobs according to user requirements and schedules them to the TaskTracker node for execution. As applications are running in VM, the JobTracker receives status updates from the TaskTracker nodes to track their progress. If JobTracker finds that all Mappers has finished their tasks successfully, then it will communicates with TaskTracker to launch the corresponding Reducer to execute reduce task. It produces the intermediate output results after each Mapper finishes its work and provides input for Reducer
- TaskTracker: These classes represent entities which receive processing requests from the JobTracker. Its primary responsibility is to track the execution of map and reduce tasks happening locally on its slave node and to report the status updates of each map and reduce tasks to the JobTracker. TaskTracker manages the processing resources on each slave node in the form of processing slots — the slots defined for map tasks and reduce tasks. It schedule the split map tasks and reduce tasks to the corresponding Mapper and Reducer, and monitors the status updates of them
- Mapper: These classes represent entities which receive the request of TaskTracker and communicate with IoTDatacenterBroker to submit the corresponding map tasks to be executed in datacentre. It frequently reports the status of map tasks to TaskTracker
- Reducer: These classes represent entities which receive the request of TaskTracker and communicate with IoTDatacenterBroker to submit the corresponding reduce tasks to be executed in datacentre. It frequently reports the status of reduce tasks to TaskTracker.

Fig. 5 shows the above main classes or entities and their interrelationship in IOTSim.

Fig. 6 summaries the workflows of these entities. Once an IoT-based job has been submitted, JobTracker begins to track a simulated job and split the job. It creates one map task for each split. Then, TaskTracker starts to track each map task and sends messages to the JobTracker via event mechanism which reports the

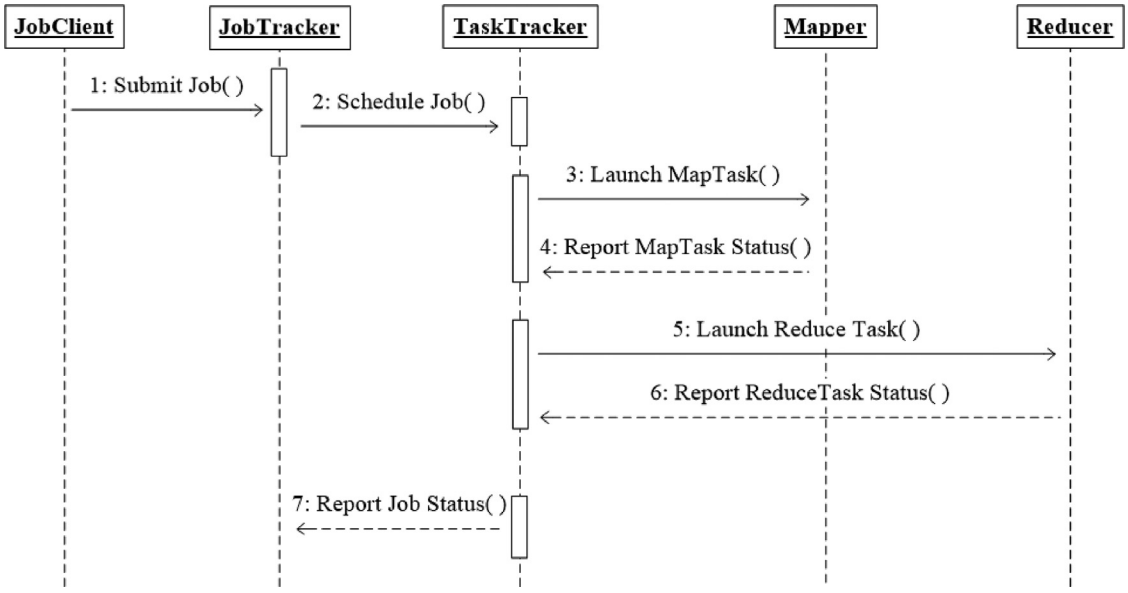


Fig. 6. Sequency diagram: communication between IOTSim entities.

task status to the JobTracker. Once TaskTracker is aware that the current tasks are finished, then, it starts to run a new task. If all the map tasks are finished, then TaskTracker reports it to the JobTracker, and is ready to run the corresponding reduce task.

4.4. Implementation

As mentioned before, our simulator extends Cloudsim. Hence, the implementation consists of two parts according to the aforementioned design decisions: modification and addition. Modifications are done on the original Cloudsim code including Datacentre, CloudTag, and DatacenterBroker etc. Upon the requirements stated in Section 2, the big data processing paradigm need to be implemented.

Fig. 7 shows the flow of how IOTSim works. Firstly, IOTSim initializes a series of entities, for example, Datacentre, Broker, JobTracker and TaskTracker. A specific number of VMs has been created with the pre-defined configuration (stated in Table 2). It also accepts multiple user-defined MapReduce jobs. When MapReduce jobs come in, JobTracker splits them into a number of blocks and schedules them to TaskTracker for execution. TaskTracker allocates the split job into the corresponding Mapper and Reducer. Then, Mapper and Reducer submit the corresponding map tasks and reduce tasks to the VMs where they are executed. The status of the map tasks and reduce tasks are reported to TaskTracker. In the runtime, the enhanced DatacenterBroker object works linking each object. When the linking ends, the simulator takes an action on captured event time. An event occurs when Cloudsim creates, executes, and terminates each object such as datacenter, broker, VM, JobTracker and TaskTracker. runClockTick() function works checking each SimEntity object and this state which is runnable at the event time. If the state is runnable, each SimEntity object classifies its own operable events. Each entity checks simulating tag and operates each request. Each object has one of various tags. They consist of entity creation, acknowledge, characteristic setting, event pause, move, submit, migration, termination and etc. At the event time that the cloudlet process is submitted, the simulator calculates all submitted cloudlet's processing time. During the event processing, a new event may be created. When new event is created, send() or sendNow() function is called. These functions notify that one event time is created. When the all event time is over, then simulating ends and the simulation result is reported. The simulation reports consist of each MapReduce jobs' information (name, split number, job length etc.), status, executed in which datacentre and which VM, and processing result (job id, VM id, start time, execution time, finish time, VM computation cost etc.).

4.5. Extensions to cloudsim

We have extended and enhanced Cloudsim with new functionality so that it can support executing multiple CloudletLists sequentially. The work originated from a functionality limitation existing in current Cloudsim. Internally, the broker has a very simple operation and has a single CloudletList, which means if you submitted multiple CloudletLists to the broker, they are always merged to a single list, and they are handled as if only one submission were made. However, it is not suitable for a real MapReduce framework as the reduce operation can only start after the corresponding map operation.

In order to solve this problem, we implemented a new broker called IOTSimBroker which inherits the original broker in Cloudsim. This new broker can accept the multiple Cloudlets and execute them sequentially. By virtue of this, the new broker can guarantee execution of the reduce task after corresponding map task has ended.

5. Evaluation

Currently, we implemented MapReduce model as one of the big data processing paradigm while keeping the stream computing model as future work. To evaluate the efficacy of IOTSim, we conduct a number of experiments and collect results. The experiment results were collected in a machine that had one Intel i7-5500 U Core 2.40 GHz and 16GB of RAM memory. All of these hardware resources were made available to a VM running Windows 7 SP1 that was used for running the simulator.

5.1. Experiment scenario

IoT applications are enabling smart city initiatives all over the world. Smart city includes many different components (i.e., smart transportation, smart healthcare, smart energy) [52,53], where big data processing technologies such as MapReduce play an integral part. An example scenario could be from smart transportation that

Please cite this article as: X. Zeng et al., IOTSim: A simulator for analysing IoT applications, Journal of Systems Architecture (2016), http://dx.doi.org/10.1016/j.sysarc.2016.06.008

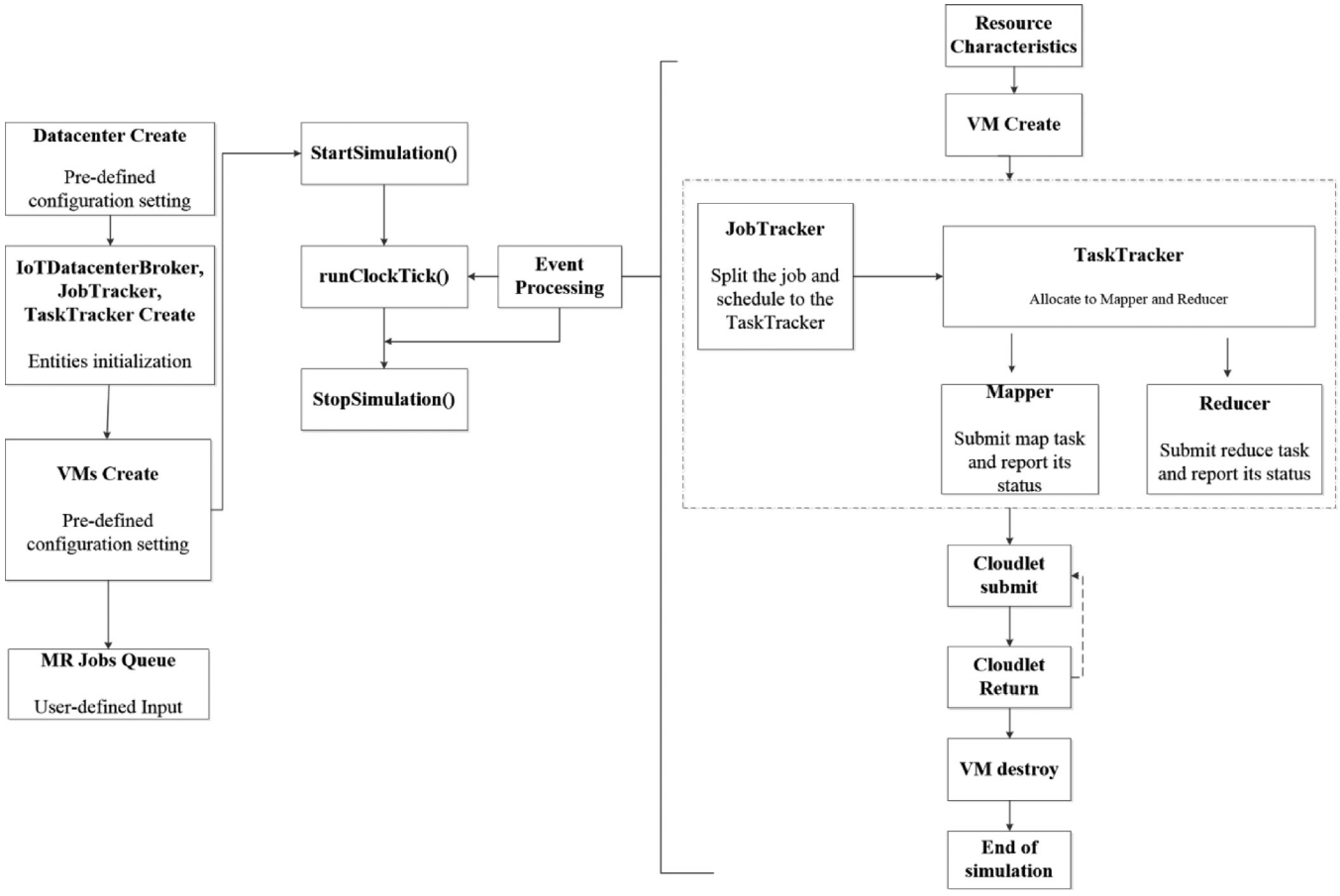


Fig. 7. The basic workflow of how IOTSim works.

a city council is planning to optimise and expand its road network. For such a scenario to be feasible, it is important that large datasets such as road network, road traffic, commuter requirements etc. are collected and stored in the cloud infrastructure. Further, the council needs to process and analyse this big data stemming from IoT devices to extract relevant knowledge such as highly utilized roads, traffic patterns, risky roads etc. Using such datasets is not an easy task due to its huge size and non-standard format [54], hence, the MapReduce paradigm is used to speed up data extraction, indexing, and querying from this big data in such scenarios. For our experimental evaluations, we have considered the smart city application described earlier for modelling. We considered there is one such application job or multiple jobs coming as an input into our simulated environment. The incoming jobs are processed using MapReduce approach which produces the corresponding output once the jobs are finished. The output presents the related information regarding the jobs including job type, job length, job size, the VM ID where map task or reduce task is being processed, start time, execution time, and finish time etc.

For comparisons, we have summarized four group experiments. In each group, we change an independent variable, while keeping others constant. Hence, we can clearly observe the result and make an appropriate analysis on how dependent variable is impacted by independent variable in details and if it matches the real world.

In the experiment, we define a series of key independent variables and dependent variables. independent variable include datacentre configuration, VM configuration, VM number, job configuration, MR combination while dependent variable include average execution time, maximum execution time, minimum execution time, make span, delay time, VM computation cost, network cost. These independent variables are the main impact factors that affect the above dependent variables. They will be detailed in the next section.

It is worth noting that all of the above group experiments (which are detailed in Section 5.4) are with two cases considered.

- Without Network Delay: in this scenario, JobTracker splits the MapReduce jobs according to user requirements and schedules them to the TaskTracker node for execution immediately when it receives the jobs summited by user. Also, when all the Mappers finish the map tasks successfully, the corresponding reduce tasks begin to execute immediately. This scenario means no network delay is incurred during the whole period that the job is running in the simulated big data environment.
- Network Delay: in this scenario, JobTracker firstly gets the data from storage (HDFS) for each MapReduce job when the simulation begins, this causes the first delay that job starts after the simulation clock. By subtracting the start time of map task and the start time of simulation clock time (also refer to the formula of Delay Time stated in Section 5.3.5), the result can be visualised. When all Mappers finish the map tasks, each Mapper will produces an intermediate output, then, the corresponding Reducer begins to work after it reads the intermediate output (in Hadoop, this is the shuffle process). Obviously, this causes the second delay. The result can also be visualised by subtracting the start time of reduce task and the finish time of the corresponding map task.

Table 1
Datacenter configuration.

pesNumber // number of cpus in a host	500
RAM //host memory (MB)	20,480
Storage //host storage (MB)	1,000,000
Bandwidth //amount of bandwidth	1000
MIPS //millions of instructions per second	1000

5.2. Independent variables

5.2.1. Datacenter configuration

Datacentre models the physical hardware that is offered by big data application provider. It encapsulates a set of computing hosts that can either be homogeneous or heterogeneous with their hardware configurations (memory, CPU, storage) where specific number of hosts and VMs are generated and run. Table 1 lists the datacentre configuration that is going to be consistent throughout the entire evaluation.

In Cloudsim, there are a few other parameters including OS, system architect, and VMM to be defined for initialization. They are not factors that can affect the aforementioned dependent variables. Hence, such parameters are irrelevant in our experiments.

5.2.2. VM configuration

VM component stores the following characteristics related to a VM: processor, memory, storage size, bandwidth and MIPS in Cloudsim. It should be submitted to the broker ahead of the simulation. VM parameters are monitored in Cloudsim, which means each sum of VM parameter must be less than the corresponding datacentre configuration. For a simplified reason, we define three types of VM (Small, Medium, and Large) which is compatible with typical computing infrastructure in Amazon etc. provider. The configuration of three types of VM is listed in Table 2.

5.2.3. VM number

Further to the above VM Configuration itself, we will also specify VM Number. This independent variable refers to the quantity of VM in Datacentre. VM, which is hosted in the host, is the basic computing unit where the jobs will be really processed. In our experiment, we will change the VM Number when required. For the sake of simplicity, we set the VM Number to be 3, 6 or 9. We can definitely set a very large number of VM, however, it is limited by the capacity of the Datacentre.

5.2.4. Job configuration

In Cloudsim, the complexity of an application is abstracted in terms of its computation requirements. Every application service has been modelled by Cloudlet. It has a pre-assigned instruction length and data transfer overhead that it needs to undertake during its life cycle. For the sake of simplicity, we define three types of job configuration as following.

5.2.5. MR combination

This represents the specific joining number of map task and reduce task. For the proposed IOTSim, when datacentre receives a job with Job Length and data size specified, the joining number of map task and reduce task in each job (which we call MR Combination) should also be considered throughout the experiment. For example, M1R1 means there are one map task and one reduce task for this job. In the same way, M20R1 means there are twenty map tasks and one reduce task for this job.

5.3. Dependent variables

5.3.1. Average execution time

It refers to the average execution time of Map/Reduce job. Its value is given by

$$\text{Average Execution Time} = \frac{\sum_{i=1}^{nm} \text{et}_\text{m}(\text{i})}{\text{nm}} + \frac{\sum_{j=1}^{nr} \text{et}_\text{r}(\text{i})}{\text{nr}},$$

where $et_m(i)$ is the execution time of map task i and $et_r(j)$ is the execution time of reduce task j. nm means the number of map tasks in this job, and nr represents the number of reduce tasks in this job.

5.3.2. Maximum execution time

It means the maximum execution time of MapReduce Job. Its value is given by

$$\text{Maximum Execution Time} = \max\left(et_\text{m}(i)\right) + \max(et_\text{r}(j))$$

5.3.3. Minimum execution time

It represents the minimum execution time of Map/Reduce Job. Its value is given by

$$\text{Maximum Execution Time} = \min\left(et_\text{m}(i)\right) + \min(et_\text{r}(j))$$

5.3.4. Make span

It is the time span of Map/Reduce job from start to finish. It is calculated by

$$\text{Make Span} = \text{ft}_r(nr),$$

where $ft_r(nr)$ means the finished time of reduce task nr.

5.3.5. Delay time

It means the discrepancy between reduce task starts and map task starts. It is calculated by

$$\text{Delay Time} = st_m(nm) + st_r(nr) - ft_m(nm),$$

where $st_m(nm)$ means the start time of map task nm and $st_r(nr)$ means the start time of reduce task nr.

5.3.6. VM computation cost

It refers to the CPU computation cost ($ unit) incurred when VM runs a Cloudlet. It is calculated by

$$\text{VM Computing Cost} = \left(\sum_{i=1}^{nvm} \text{et}_\text{m}(\text{i}) + \sum_{j=1}^{nvm} \text{et}_\text{r}(\text{j})\right) \times \text{VM Cost per Unit Time}$$

where $et_m(i)$ means the execution time of VM i when running the map task, while $et_r(j)$ means the execution time of VM j when running the reduce task.

Table 2
VM configuration.

VM Type	Small	Medium	Large
Image Size //amount of storage (MB)	10,000	20,000	40,000
Ram //VM ram (MB)	512	1024	2048
MIPS //millions of instructions per second	250	500	1000
Bandwidth //amount of bandwidth	1000	1000	1000
pesNumber //number of CPUs in a VM	1	2	4
VMCostPerSec // the cost of processing in VM($)	1	2	4

Please cite this article as: X. Zeng et al., IOTSim: A simulator for analysing IoT applications, Journal of Systems Architecture (2016), http://dx.doi.org/10.1016/j.sysarc.2016.06.008

Table 3
Job configuration.

Job Type	Small	Medium	Big
Job Length // the length (expressed in millions of instructions) of this job to be executed in the Datacenter	362,880	725,760	1,451,520
Data Size // the data size (in MB) of this job before submitting to a Datacenter	200,000	400,000	800,000

5.3.7. Network cost

This dependent variable means the network cost ($ unit) incurred when job task gets data from storage and reduce task gets data from intermediate output of map task. It is calculated by

$$\text{Network Cost} = \text{DelayTime} \times \text{NetworkCost per Unit Time}$$

These dependent variables are very important factors when analysing IoT-based applications. It is not hard to understand they are functions of the aforementioned independent variables, which means the value of these dependent variables change as the independent variables vary. For example, given the same IoT application job, if we increase the VM number within the capability that the datacentre can offer, then, the make span, average execution time etc. may changes because more computing resources are leveraged by map or reduce tasks in the big data processing process.

5.4. Experiment results

In Group 1 experiment, we set specific job configuration (as presented in Table 3), VM configuration and VM number, for example job type (Small Job), VM type (Small VM), VM number = 3 and give different MR Combination from M1R1 to M20R1. We then start simulation in either Without Network Delay or Network Delay case respectively. After the simulation successfully finishes, we collect the data of Average Execution Time, Max Execution Time, Min Execution Time, Make Span and VM Computing Cost and Network Cost (only applicable for Network Delay case). The results of Group 1 experiment are presented in the following line chart.

In Fig. 8(a), generally speaking, Execution Time (Average, Max, and Min) fluctuates as MR Combination increases. When number of map task is smaller than the VM number (equals to 3), it can be observed the execution time (Average, Max and Min) is identical because datacentre provides more VMs than the map taks of the job need, which means some VMs are idle. Meanwhile, execution time (Average, Max and Min) decreases rapidly because more map tasks are generated and executed simultaneously in datacentre, such that less execution time is consumed. However, if the number of map task is greater than the VM number, execution time (Average, Max, and Min) begins to flatten and the discrepancy between them is becoming narrow as MR combination increases. This is because VMs are constrained in their computing resources and many map tasks compete to be processed in these VMs, such that the effect on reducing execution time by increasing MR combination is getting more and more insignificant. All the above statement also apply to the situation when VM number is larger than 3.

Fig. 8(b) compares the make span in Network Delay case with that in Without Network Delay case. Clearly, it is observed that the make span of former is slightly larger than the make span of the latter and the disparity between them is getting narrow as MR combination increases. This is because the delay incurs when copying data from storage and getting intermediate data generated by map task in the former case, and the delay becomes less and less as MR Combination increases.

In Group 2 experiment, we set job configuration (Small Job), VM configuration (Small VM) and MR combination (from M1R1 to M20R1) and keep them invariant in the experiment. But this time, VM number varies. In order to get better visualization result, we set the VM number with appropriate discrepancy (i.e., VM number equals 3, 6, 9).

Fig. 9 shows the comparison of average execution time between these three cases (i.e., VM number = 3, 6, 9). When number of map task is smaller than VM number, their average execution time are equal. Afterwards, the chart shows more VMs results in obvious less average execution time. To be precise, when VM number increases from 3 to 6, the average execution time is reduced by 40% on average and 50% if it further increases to 9. This is because more VM resources are available for processing the same job.

The comparison of network cost is presented Fig. 10. Interesting, even though VM number has been changed, the network cost is identical. This is because, given the same job, the data size is identical, which results in the same network delay.

In Group 3 experiment, we set job configuration (Small Job) and VM number (equals 3) and provide different VM configuration (from Small VM to Large VM) in our experiment. The results of Group 3 experiment are presented in the following figure.

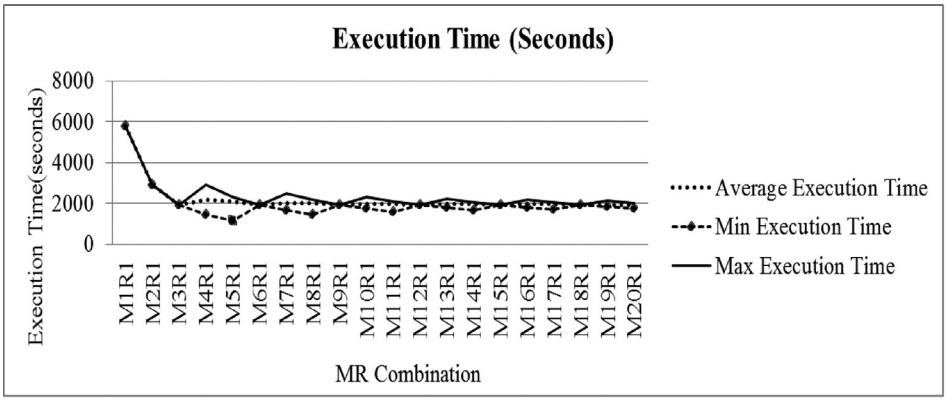
Fig. 11 shows average execution time decreases exponentially if we provide higher-profile VM. Precisely, Medium VM gets approximately 60% less average execution time, while Large VM consumes about 80% less average execution time when compared with Small VM. As presented in Table 2, Large VM has four times MIPS as much as Small VM, and Medium VM offers twice MIPS than Small VM, hence, higher configured VM can definitely provide more computing capacity.

In Group 4 experiment, VM configuration (Small VM) and VM number (equals 3) are set, while job type varies (from Small Job to Big Job) in our experiment. The VM computation cost is compared in the following figure.

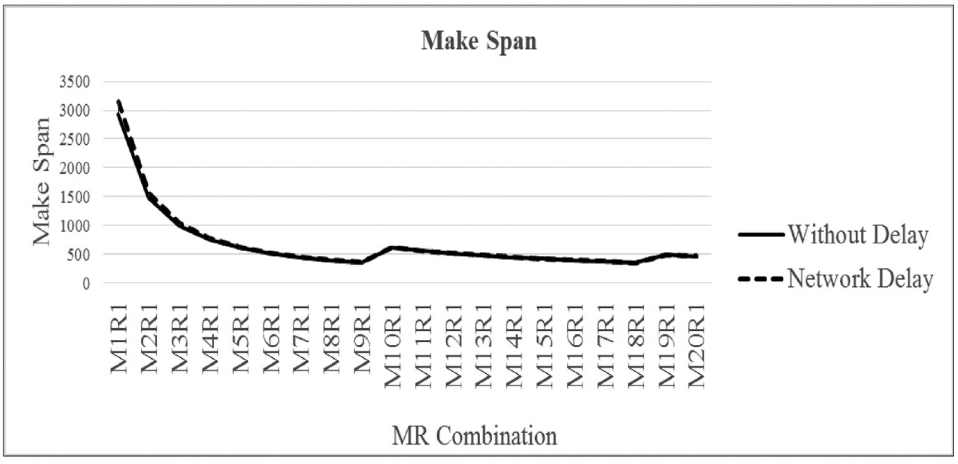
Fig. 12 shows Big Job costs VM computing resource twice as much as Medium Job and the VM computation cost of Medium Job is twice than Small Job. As presented in Table 3, Big Job doubles its job length (MI) than Medium Job while Medium Job double its job length (MI) than Small Job, hence when the same quantity of VMs with identical VM configuration are offered, higher-workload Job has linearly increased its VM computation cost, which matches the real world.

In summary, we can conclude from the above observations that efficacy of IoTSim has been proven through a number of experimental results. IOTSim largely extended Cloudsim's functionality to support for modelling and simulation of multiple IoT applications running simultaneously in a shared cloud data centres. In this version of the proposed IOTSim simulator, it is capable of simulating batch-oriented IoT applications by using MapReduce model with a high degree of accuracy. IOTSim is able to support simulation of IoT-based big data processing using MapReduce model with the ability to study the correctness and effect of independent variables i.e., VM configuration, VM number, job configuration, MR combination) on the dependent variables (i.e., average execution time, maximum execution time, minimum execution time, make span, VM computation cost and network cost). IOTSim enables the researchers to analyse how a MapReduce-compatible IoT application performs in certain environment. The simulation result provides better perspective to analyse IoT-based applications using MapReduce model in Cloud Computing environment with less cost and time.

Please cite this article as: X. Zeng et al., IOTSim: A simulator for analysing IoT applications, Journal of Systems Architecture (2016), http://dx.doi.org/10.1016/j.sysarc.2016.06.008



(a)



(b)

Fig. 8. Effects of independent variables on dependent variables by changing MR Combination: (a) execution time (Average, Max, Min); (b) Make Span.

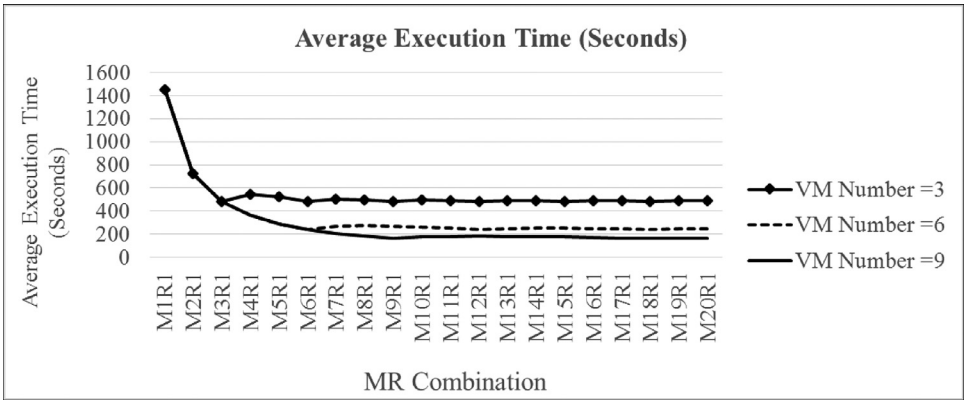


Fig. 9. Comparison of average execution time when changing VM Number.

6. Conclusion and future work

Nowadays, increasing IoT-based applications rely on Clouds to run big data processing platform to process massive amounts of data generated from billions of devices and conduct data analytics for knowledge extraction. However, setting-up such environment is very challenging and a tedious task and is expensive in terms of cost and time. To address this problem, we proposed designed and implemented IOTSim. IOTSim allows simulation of IoT application by inherently supporting big data processing system such as with MapReduce to facilitate researchers and commercial organizations to understand and analyse the impact and performance of IoT-based applications. Our simulator is built on top of a widely used simulator, i.e., Cloudsim. However, we have extensively extended and improved the existing functions of Cloudsim.

Although MapReduce is a popular distributed processing framework for batch oriented IoT applications, it has obvious restrictions and limits to handle those IoT applications that have real-time and low-latency requirements. Instead, stream processing is highly re-

Please cite this article as: X. Zeng et al., IOTSim: A simulator for analysing IoT applications, Journal of Systems Architecture (2016), http://dx.doi.org/10.1016/j.sysarc.2016.06.008

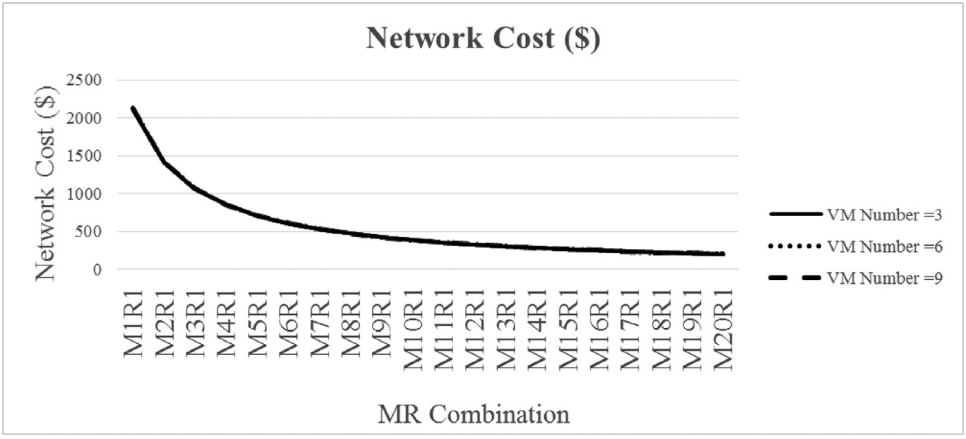


Fig. 10. Comparision of network cost when VM Number changes.

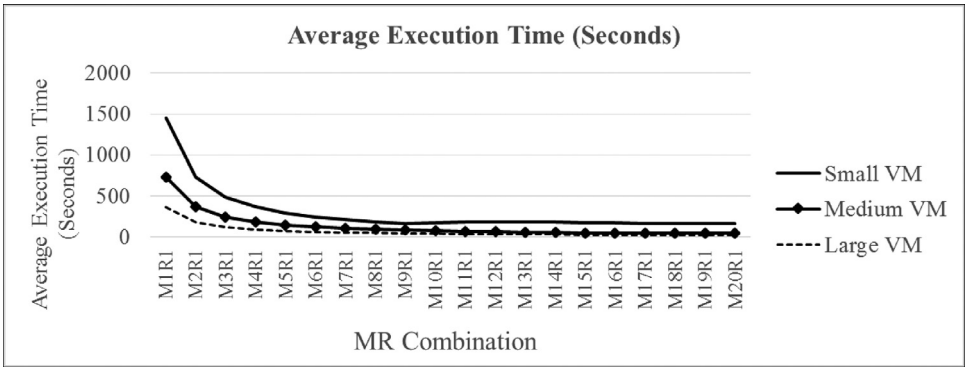


Fig. 11. Comparison of average execution time when changing VM configuratoin.

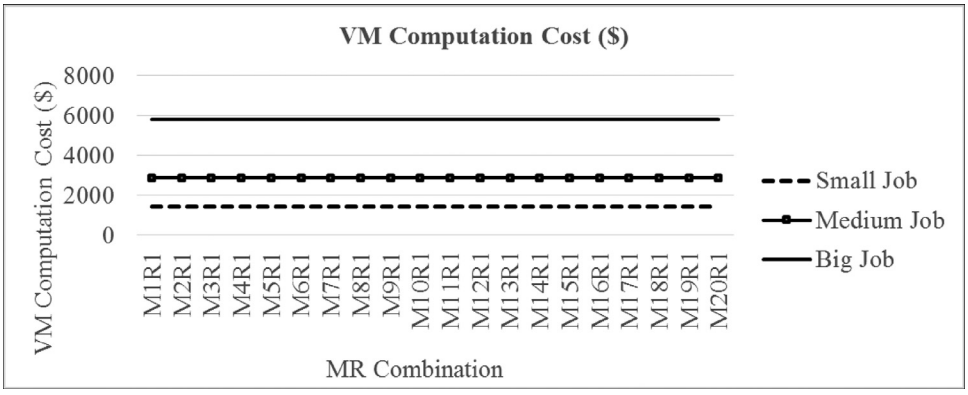


Fig. 12. Comparion of VM computation cost when changing job configuration.

quired and has been identified as the ideal platform to process such IoT applications in real time. Hence, in order to achieve simulation of the stream computing sublayer in our proposed architecture, much of our future work In the future, we would like to investigate the stream processing paradigm and techniques to design and implement stream computing model as a supplementary to our current simulation works.

References

[1] B.T. Rao, L.S.S. Reddy, Survey on improved scheduling in Hadoop mapreduce in cloud environments, Int. J. Comput. Appl. 34 (9) (2012) 29–33.
[2] Y. Kouki, T. Ledoux, SLA-driven capacity planning for cloud applications, in: *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, 2012, pp. 135–140.
[3] R.H. Weber, R. Weber, Internet of Things, Springer, 2010.
[4] X. Wu, X. Zhu, G.-Q. Wu, W. Ding, Data mining with big data, Knowl. Data Eng. IEEE Trans. 26 (1) (2014) 97–107.
[5] W. Fan, A. Bifet, Mining big data: current status, and forecast to the future, ACM SIGKDD Explor. Newslett. 14 (2) (2013) 1–5.
[6] Z. Deng, X. Wu, L. Wang, X. Chen, A. Zomaya, D. Chen, Parallel processing of dynamic continuous queries over streaming data flows, Parallel Distrib. Syst. IEEE Trans. 26 (3) (2015) 834–846.
[7] C. Perera, A. Zaslavsky, P. Christen, D. Georgakopoulos, Sensing as a service model for smart cities supported by internet of things, Trans. Emerg. Telecommun. Technol. 25 (1) (2014) 81–93.
[8] C. Perera, A. Zaslavsky, P. Christen, D. Georgakopoulos, Context aware computing for the internet of things: a survey, Commun. Surv. Tut. IEEE 16 (1) (2014) 414–454.
[9] C. Perera, A. Zaslavsky, P. Christen, et al., Sensing as a service model for smart cities supported by internet of things[J], Trans. Emerg. Telecommun. Technol. 25 (1) (2014) 81–93.
[10] L. Wang, R. Ranjan, J. Chen, B. Benatallah, Cloud Computing: Methodology, Systems, and Applications, CRC Press, 2011.

Please cite this article as: X. Zeng et al., IOTSim: A simulator for analysing IoT applications, Journal of Systems Architecture (2016), http://dx.doi.org/10.1016/j.sysarc.2016.06.008

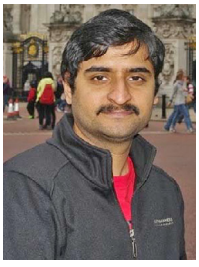
[11] L. Wang, G. Von Laszewski, A. Younge, X. He, M. Kunze, J. Tao, et al., Cloud computing: a perspective study, New Gen. Comput. 28 (2) (2010) 137–146.
[12] L. Wang, C. Fu, Research advances in modern cyberinfrastructure, New Gen. Comput. 28 (2) (2010) 111–112.
[13] L. Wang, D. Chen, Y. Hu, Y. Ma, J. Wang, Towards enabling cyberinfrastructure as a service in clouds, Comput. Elect. Eng. 39 (1) (2013) 3–14.
[14] D. Abadi, R. Agrawal, A. Ailamaki, M. Balazinska, P.A. Bernstein, M.J. Carey, et al., The beckman report on database research, ACM SIGMOD Rec. 43 (3) (2014) 61–70.
[15] R. Zhang, R. Jain, P. Sarkar, L. Rupprecht, Getting your big data priorities straight: a demonstration of priority-based QoS using social-network-driven stock recommendation, in: Proceedings of the VLDB endowment, 7, 2014.
[16] D. Chen, Z. Liu, L. Wang, M. Dou, J. Chen, H. Li, Natural disaster monitoring with wireless sensor networks: a case study of data-intensive applications upon low-cost scalable systems, Mobile Netw. Appl. 18 (5) (2013) 651–663.
[17] A. Bashar, Modeling and simulation frameworks for cloud computing environment: a critical evaluation, in: International Conference on Cloud Computing and Services Science, 2014, pp. 1–6.
[18] L. Atzori, A. Iera, G. Morabito, The internet of things: a survey, Comput. Netw. 54 (15) (2010) 2787–2805.
[19] Q. Xiaocong, Z. Jidong, Study on the structure of "Internet of Things (IOT)" business operation support platform, in: Communication Technology (ICCT), 2010 12th IEEE International Conference on, 2010, pp. 1068–1071.
[20] M. Aazam, I. Khan, A.A. Alsaffar, E.-N. Huh, Cloud of things: integrating Internet of Things and cloud computing and the issues involved, in: Applied Sciences and Technology (IBCAST), 2014 11th International Bhurban Conference on, 2014, pp. 414–419.
[21] D. Georgakopoulos, P.P. Jayaraman, M. Zhang, R. Ranjan, Discovery-driven service oriented IoT architecture, in: 2015 IEEE Conference on Collaboration and Internet Computing (CIC), 2015, pp. 142–149.
[22] D. Uckelmann, M. Harrison, F. Michahelles, Architecting the internet of things, Springer Science & Business Media, 2011.
[23] R. Khan, S.U. Khan, R. Zaheer, S. Khan, Future internet: the Internet of Things architecture, possible applications and key challenges, in: Frontiers of Information Technology (FIT), 2012 10th International Conference on, 2012, pp. 257–260.
[24] X. Zeng, P. Strazdins, S.K. Garg, L. Wang, Cross-layer SLA management for cloud-hosted big data analytics applications, in: Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on, 2015, pp. 765–768.
[25] B.S. Kim, S.J. Lee, T.G. Kim, H.S. Song, MapReduce based experimental frame for parallel and distributed simulation using Hadoop Platform, in: European Conference Modeling Simulation, 2014, pp. 664–669.
[26] P. Mundkur, V. Tuulos, J. Flatow, Disco: a computing platform for large-scale data analytics, in: Proceedings of the 10th ACM SIGPLAN workshop on Erlang, 2011, pp. 84–89.
[27] B. He, W. Fang, Q. Luo, N.K. Govindaraju, T. Wang, Mars: a MapReduce framework on graphics processors, in: Proceedings of the 17th international conference on Parallel architectures and compilation techniques, 2008, pp. 260–269.
[28] K. Taura, K. Kaneda, T. Endo, A. Yonezawa, Phoenix: a parallel programming model for accommodating dynamically joining/leaving resources, ACM SIGPLAN Notices 38 (10) (2003) 216–229.
[29] D. Agrawal, S. Das, A. El Abbadi, Big data and cloud computing, in: Proceedings of the 14th International Conference on Extending Database Technology - EDBT/ICDT '11, 443, 2011, p. 530.
[30] R. Lämmel, Google's MapReduce programming model—revisited, Sci. Comput. Program. 70 (1) (2008) 1–30.
[31] R. Buyya, M. Murshed, Gridsim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing, Concurren. Comput. 14 (13-15) (2002) 1175–1220.
[32] H.J. Song, X. Liu, D. Jakobsen, R. Bhagwan, X. Zhang, K. Taura, et al., The microgrid: a scientific tool for modeling computational grids, in: Supercomputing, ACM/IEEE 2000 Conference, 2000, p. 53.
[33] C.L. Dumitrescu, I. Foster, GangSim: a simulator for grid scheduling studies, in: Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on, 2005, pp. 1151–1158.
[34] H. Casanova, Simgrid: A toolkit for the simulation of application scheduling, in: Cluster Computing and the Grid, 2001. Proceedings. First IEEE/ACM International Symposium on, 2001, pp. 430–437.
[35] W.H. Bell, D.G. Cameron, L. Capozza, A.P. Millar, K. Stockinger, F. Zini, Simulation of dynamic grid replication strategies in optorsim, in: Grid Computing—-GRID 2002, Springer, 2002, pp. 46–57.
[36] P. García, C. Pairot, R. Mondéjar, J. Pujol, H. Tejedor, R. Rallo, Planetsim: A New Overlay Network Simulation Framework, Springer, 2005.
[37] R. Ranjan, A. Harwood, R. Buyya, Coordinated load management in Peer–to-Peer coupled federated grid systems, J. Supercomput. 61 (2) (2012) 292–316.
[38] D. Kliazovich, P. Bouvry, S.U. Khan, GreenCloud: a packet-level simulator of energy-aware cloud computing data centers, J. Supercomput. 62 (3) (2012) 1263–1283.
[39] A. Núñez, J.L. Vázquez-Poletti, A.C. Caminero, G.G. Castañé, J. Carretero, I.M. Llorente, iCanCloud: a flexible and scalable cloud infrastructure simulator, J. Grid Comput. 10 (1) (2012) 185–209.
[40] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A. De Rose, R. Buyya, CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Software 41 (1) (2011) 23–50.
[41] B. Wickremasinghe, R.N. Calheiros, R. Buyya, 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 on, 2010, pp. 446–452.
[42] S.K. Garg, R. Buyya, Networkcloudsim: Modelling parallel applications in cloud simulations, in: Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on, 2011, pp. 105–113.
[43] R.N. Calheiros, M.A. Netto, C.A. De Rose, R. Buyya, EMUSIM: an integrated emulation and simulation environment for modeling, evaluation, and validation of performance of cloud computing applications, Software 43 (5) (2013) 595–612.
[44] S.-H. Lim, B. Sharma, G. Nam, E.K. Kim, C.R. Das, MDCSim: a multi-tier data center simulation, platform, in: Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on, 2009, pp. 1–9.
[45] R. Malhotra, P. Jain, Study and Comparison of CloudSim simulators in the cloud computing, SIJ Trans. Comput. Sci. Eng. Its Appl. 1 (2013) September-October.
[46] G. Wang, A.R. Butt, P. Pandey, K. Gupta, Using realistic simulation for performance analysis of MapReduce setups, in: Proceedings of the 1st ACM workshop on Large-Scale system and application performance, 2009, pp. 19–26.
[47] S. Hammoud, M. Li, Y. Liu, N.K. Alham, Z. Liu, MRSim: a discrete event based MapReduce simulator, in: Fuzzy Systems and Knowledge Discovery (FSKD), 2010 Seventh International Conference on, 2010, pp. 2993–2997.
[48] A. Murthy, Mumak: map-reduce simulator, MAPREDUCE-728, Apache JIRA (2009).
[49] A. Verma, L. Cherkasova, R.H. Campbell, Play it again, SimMR!, in: Cluster Computing (CLUSTER), 2011 IEEE International Conference on, 2011, pp. 253–261.
[50] C. Simatos, Making Simjava Count *MSc. Project report*, The University of Edinburgh, 2002.
[51] J. Jung, H. Kim, MR-cloudsim: designing and implementing MapReduce computing model on CloudSim, in: ICT Convergence (ICTC), 2012 International Conference on, 2012, pp. 504–509.
[52] C. Yin, Z. Xiong, H. Chen, J. Wang, D. Cooper, B. David, A literature survey on smart cities, Sci. Chin. Inf. Sci. 58 (10) (2015) 1–18.
[53] A. Solanas, C. Patsakis, M. Conti, I. Vlachos, V. Ramos, F. Falcone, et al., Smart health: a context-aware health paradigm within smart cities, Commun. Mag. IEEE 52 (8) (2014) 74–81.
[54] L. Alarabi, A. Eldawy, R. Alghamdi, M.F. Mokbel, TAREEG: a MapReduce-based web service for extracting spatial data from OpenStreetMap, in: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, 2014, pp. 897–900.

Please cite this article as: X. Zeng et al., IOTSim: A simulator for analysing IoT applications, Journal of Systems Architecture (2016), http://dx.doi.org/10.1016/j.sysarc.2016.06.008

Xuezhi Zeng is a PhD Candidate in the Computer Systems Group of the Research School of Computer Science at the Australian National University. He received his master degree in information system from Fudan University. His research interests include distributed computing technologies, big data analytics and internet of things.

Saurabh Garg is currently working as a lecturer in the Department of Computing and Information Systems at the University of Tasmania, Hobart, Tasmania. He was one of the few Ph.D. students who completed in less than three years from the University of Melbourne in 2010. He has published more than 30 papers in highly cited journals and conferences with H-index 20. His doctoral thesis focused on devising novel and innovative market-oriented meta-scheduling mechanisms for distributed systems under conditions of concurrent and conflicting resource demand. He has gained about three years of experience in the Industrial Research while working at IBM Research Australia and India.

Peter Strazdins is an Associate Professor in the Computer Systems Group of the Research School of Computer Science at the Australian National University. He is a Senior Member of the IEEE and a Senior Fellow of the Higher Education Academy (SFHEA). Peter is the convenor for the Bachelor of Advanced Computing. He is also a Green Rep, a Delegate to the National Tertiary Education Union. From 2009–2013, he was the Associate Director of Education for RSCS ANU. Up to 2009, he was Convenor of the Coursework Masters programs. the CSIT Safety Co-ordinator, the chair of the CSIT Occupational Health and Safety Committee and co-ordinator of the CSIT Ride to Work Group.



Prem Prakash Jayaraman is currently a research fellow at RMIT University, Melbourne. His research areas of interest include, Internet of Things, cloud computing, mobile computing, sensor network middleware and semantic internet of things. Dr. Jayaraman is one of the key contributors of the Open Source Internet of Things project (OpenIoT) that has won the prestigious Black Duck Rookie of the Year Award in 2013. He has been the recipient of several awards including hackathon challenges at the Fourth International Conference on IoT (2014) at MIT Media Lab, Cambridge, MA and IoT Week 2014 in London and best paper award at IEA/AIE-2010. He was a postdoctoral research fellow at CSIRO Digital Productivity Flagship, Australia from 2012 to 2015. Prior to that, he worked as a research fellow and lecturer at the Centre for Distributed Systems and Software Engineering, Monash University, Melbourne, Australia. He has served as a program committee member for the Hawaii International Conference on System Sciences and Mobile Data Management Conferences and is a reviewer of many distributed systems and software engineering journals, including Elsevier's Future Generation Computer Systems, John Wiley & Sons' Concurrency and Computation: Practice and Experience, World Wide Web Journal, and IEEE Transactions on Cloud Computing

Dimitrios Georgakopoulos has recently joined RMIT as a professor at the School of Computer Science and Information Technology in Melbourne, Australia. Until June 2014 he was the Director of the Information Engineering Laboratory at CSIRO's ICT Centre. Before coming to CSIRO in 2008, Dimitrios held research and management positions in several industrial laboratories in the US. From 2000 to 2008, he was a Senior Scientist with Telcordia, where he helped found Telcordia's Research Centers in Austin, Texas, and Poznan, Poland. From 1997 to 2000, Dimitrios was a Technical Manager at the Microelectronics and Computer Corporation (MCC), and the Project Manager and Chief Architect of MCC's Collaboration Management Infrastructure (CMI) consortial project. From 1990–1997, Dimitrios was a Principal Scientist at GTE (currently Verizon) Laboratories Inc. He is an Adjunct Professor at the Australian National University and a CSIRO Adjunct Science Fellow.

Rajiv Ranjan is an Associate Professor (Reader) in Computing Science at Newcastle University, United Kingdom. Prior to that, he was a Senior Research and Julius Fellow at CSIRO, Canberra, where he was working on projects related to Cloud and big data computing. He has been conducting leading research in the area of Cloud and big data computing developing techniques for: (i) Quality of Service based management and processing of multimedia and big data analytics applications across multiple Cloud data centers (e.g., CSIRO Cloud, Amazon and GoGrid); and (ii) automated decision support for migrating applications to data centers. He has published about 110 papers that include 60+ journal papers. He serves on the editorial board of IEEE Transactions on Computers, IEEE Transactions on Cloud Computing, IEEE Cloud Computing, and Future Generation Computer System Journals. According to Google Scholar Citations his papers have received about 3450+ citations and he has an h-index of 24.

Please cite this article as: X. Zeng et al., IOTSim: A simulator for analysing IoT applications, Journal of Systems Architecture (2016), http://dx.doi.org/10.1016/j.sysarc.2016.06.008