Abstract

This paper deals with an efficient and robust parallel and distributed simulation framework for standalone Monte Carlo simulation based on a MapReduce computing framework. The Monte Carlo simulation method is inherently computing-intensive and requires many replicated simulation runs to get meaningful statistical results. Thus, it is important to reduce total simulation time by exploiting hardware and/or software as well as to reuse existing standalone simulation programs with little modification for the replicated simulations. To cope with this situation, we propose a general framework that turns a stand-alone Monte Carlo simulator into a chain of MapReduce jobs in order to run the simulation on a MapReduce framework such as Hadoop. A case study of an air defense simulation on 16-node Hadoop cluster illustrates that the proposed framework is feasible and fully utilizes the merit of the parallel and distributed computing environment.

Keywords: MapReduce, Monte Carlo Simulation, Parallel and Distributed Simulation, System Analysis

1. Introduction

Monte Carlo simulation has been used for decades to compute a variety of applications such as radiation transport, dose estimation, radiation shield, and so on, that has random variables in the system's nature and are too complex to solve analytically. It provides fairly accurate approximate solutions to a variety of mathematical problems by performing statistical sampling experiments through a computer. As more experiments are conducted and more accurate results are obtained, the required amount of replicated simulation runs are accompanied by computation costs and time. For example, if over 10,000 replications are needed to obtain accurate Monte Carlo simulation results and each replication takes only one minute, it takes about 7 days to get all the results of replications. Monte Carlo simulation is inherently computing-intensive and requires efficient hardware and/or software to reduce total simulation time. Moreover, it is a time-consuming job to collect and analyze the simulation results.

Many researches have attempted faster simulation methods to reduce time and effort for successful Monte Carlo simulation. Most of them have widely used parallel and distributed computing methods to overcome this time-consuming nature. DEXSim, CR-PADS, mJADES, and MEG are representative researches adapting parallel and distributed simulation frameworks to reduce total simulation time. These researches provide efficient simulation environments with best use of distributed hardware resources. As they focus on the speedup of the simulation and the utilization of the given resources, the simulation environments were developed on the basis of their specialized hardware and/or software. This means, however, that one needs to spend much time and cost to construct such environments. Also a simulator sometimes needs to be modified to fit into the environments. Moreover, they don't provide any mechanism to collect and analyze the simulation results. Thus we need a new simulation framework for both the kind of Monte Carlo simulation and for analysis of the results on the basis of a commodity computing framework to reduce time, effort, and cost.

Recently, cloud-computing technology such as MapReduce has been used because it gives advantages of cost, fault tolerance, and is relatively easy to transform existing problems to fit on the frameworks compared
to other platforms used for the previous researches. Although this computing method generally has been used for big-data processing, we found that it is of good use for simulation fields. Pratx et al.\textsuperscript{9} conducted research about Monte Carlo simulation using MapReduce framework. This research aims to port a specific Monte Carlo computation package for photon migration to MapReduce. However, it has some limitations for executing other legacy simulators on MapReduce. So a general-purpose framework is required to execute existing legacy simulators and analysis tools for reusability, modularity, and flexibility. Thus, we have conducted a preliminary study on an experimental frame based on the MapReduce framework for parallel and distributed simulation\textsuperscript{10} where we propose a parallel and distributed execution environment for executing a legacy simulator and collecting the simulation results. As the work doesn’t consider Monte Carlo simulation, it does not offer any sampling module and statistical analysis method of the results, which has been identified by another work\textsuperscript{11}. This paper extends and completes these insufficient parts of both of the previous works.

Figure 1 depicts the concept of the problem statement to propose an automated parallel and distributed Monte Carlo simulation framework from replicated simulation execution and statistical analysis based on MapReduce framework and commodity hardware. This paper proposes a general framework that turns a standalone Monte Carlo simulator into a chain of MapReduce jobs in order to run the simulation on a MapReduce framework that is operated on a cluster of commodity computers.

This paper is organized as follows. Brief related backgrounds about Monte Carlo simulation and MapReduce framework are presented in the next section. Then we propose the parallel and distributed Monte Carlo simulation framework based on the MapReduce framework in section 3, which is experimented by a case study in the following section. We conclude with summary and future works.

2. Background

2.1 Monte Carlo Simulation

Generally, the concept of Monte Carlo simulation is a combination of Monte Carlo method and simulation theory, depicted in Figure 2. Monte Carlo method is a sort of statistical sampling theory that tries to estimate the statistics of an original population with a limited number of sampled populations generated by a given probability density function\textsuperscript{12}. Meanwhile, a model is an abstraction of a real system in a specific interesting aspect. When the model is too complex to get a closed form of analytical model we have to make a simulation model, which is turned into a computer simulation program. Furthermore, if the model contains random variables, the simulation should repeat with a limited number of sampled input data to produce outputs in order to statistically estimate the behavior of the original real system, which we call a Monte Carlo simulation. Thus, it inherently requires a huge computation cost in proportion to the number of sampled inputs, which is a major drawback of Monte Carlo simulation.
2.2 MapReduce

Hadoop is a representative big data platform for reliable, scalable, large scale distributed computing\(^{13}\). Hadoop consists of MapReduce and Hadoop Distributed File System (HDFS)\(^{14}\). MapReduce is an efficient framework for large-scale distributed data processing based on the divide and conquer paradigm. MapReduce works by breaking the processing into map and reduce as shown in Figure 3. Map and reduce are executed in parallel on the different machines within the Hadoop cluster by MapReduce framework. Map processes a key-value pair to generate a set of intermediate key-value pairs. Reduce accepts an intermediate key and a set of values for that key, and merges together these values to form a smaller set of values. The user can specify map/reduce functions, and types of input/output\(^{13}\). The input data that consists of key value pairs is stored on the HDFS. They are split into fixed-size blocks, and allocated to the user-specified maps. The map function is applied on the input block to produce intermediate key-value pairs. The intermediate data is partitioned by the key, and the grouped records are shuffled to the appropriate reduces. Then, the shuffled records are merged and sorted in the node that a reduce task located. Each reduce sequentially processes key-value pairs by the user-specified reduce function to generate the final output key-value pairs.

2.3 Hadoop Streaming

To implement the proposed work, it is required to execute simulators and statistical programs on the MapReduce framework. Because MapReduce applications are executed in the form of source code, it is difficult to run an executable legacy simulator on the framework. Therefore, an interface is required to run one on the MapReduce framework. Hadoop provides a utility called Hadoop Streaming\(^{15}\). It allows creating and running MapReduce jobs with any executable program as the Mapper and Reducer. Executable programs communicate with Hadoop Streaming through UNIX streams shown in Figure 4. They read the input key-value pairs from standard input (stdin) line by line, and emit the output key-value pairs to standard output (stdout).

3. Proposed Framework

In this section, we propose a framework for Monte Carlo simulation adapting Hadoop MapReduce. Figure 5 shows the mapping between Monte Carlo (MC) simulation and MapReduce (MR) framework. In this proposed work, the sampling phase of Monte Carlo simulation could be transformed to the sampling module. The Monte Carlo simulation part could be assigned to the mapper task of MapReduce, and the statistical part of the analysis of simulation results could be assigned to the reducer task of MapReduce.

3.1 Overall Architecture of MC-MR Framework

As shown in Figure 6, the proposed framework is composed of three major phases; sampling, simulation,
and analysis phase. The first phase of a Monte Carlo simulation framework is a sampling part that includes a sampling module. The sampling module generates random values of its input random variables according to their probability density functions. It also generates scenarios from input variable set using the method of design of experiment 16. The sampling module outputs the inputs of the simulator that contains the initial values for simulation. The second phase is a simulation part that executes a simulation model on the map of MapReduce. The simulation model is assigned to each mapper task for parallel and distributed simulation. The inputs of the mapper are appropriate parameters of the sampling module, such as seeds or means and variances of input random variables. The output of the mapper at key-value pairs, which contains simulation results such as simulation time and performance index. The last phase of the framework is an analysis part that analyzes the result of the simulation. This phase is assigned to each reducer task according to the number of simulation results. For examples, if there are two parameters to analyze, each value of the parameters is sent to each reducer task after the simulation. Then, the results are gathered and analyzed by the statistical software, which can be a commercial or user-developed application, located on the reducer task.

### 3.2 Process of the MC-MR Framework

This section presents an overall simulation process using the proposed framework as shown in Figure 7. The detail of the process is as follows.

#### 3.2.1 Sampling Phase: Requirement Analysis and Input Generation

First of all, we should establish the modeling and simulation objective of the target system and draw the requirement matrix can be used to recognize the relation between the object and the performance index. We should also specify appropriate parameters such as seeds or means and variance related to inputting random variables manually. The number of Monte Carlo simulations and the probability density functions are also significant elements to specify in this process. The sampling module automatically combines the foregoing information and generates experimental scenarios for Monte Carlo simulation.

#### 3.2.2 Simulation Phase: Simulator Allocation and Simulation

After the scenario generation from sampling module, the output scenarios are stored to the HDFS. The MapReduce splits them into an individual block of scenario and then each block is sent to a map. The map is automatically assigned to each CPU core of Hadoop cluster by the MapReduce following the total number of scenarios. Efficient simulation and analysis could be possible with the MapReduce framework because the framework also provides load balancing and resource management for the map and reduce operation. At the same time, the simulator (simulation model and simulation engine) is written in the Local File System (LFS) of the machine where the map is assigned.

Allocated scenarios are sent to legacy simulator using Hadoop streaming. The simulator receives the initial input parameters (scenario) from stdin and emits the output to stdout in the form of key-value pairs after the simulation. For these processes, the input and output interface of the simulator needs some modifications. The simulator is executed on each of the CPU cores of the machine by using the simulation engine stored in the LFS. All of the processes in the simulation phase are automatically progressed by the supporting of the MapReduce framework.

---

Figure 6. Overall Architecture of Proposed Framework.

Figure 7. Simulation Process of Proposed Framework.
3.2.3 Analysis Phase: Result Allocation and Analysis

After the simulation is finished, the simulator outputs the result to its own map. Then, the output is allocated to a particular reducer by the “partition” function. The partition function is given the key and the number of reducers and returns the index of the desired reducer. If there are more than two results that have to be analyzed, each result has several keys in this proposed framework. The results of mappers are partitioned to a particular reducer according to the assigned keys. And they are sent to the reducer by the function called “shuffle” through the network. The reducer merges and sorts the results before analyzing them using statistical software. For instance, suppose the simulator outputs two parameters to analyze, such as “Param A” and “Param B.” Param A from all mappers are sent to reducer 1, and Param B from all mappers are sent to reducer 2. Reducer 1 collects and analyzes only the result of Param A, and same for reducer 2 with Param B.

The statistical software receives the simulation outputs from stdin and emits the analysis results to stdout in the form of key-value pair, as aforementioned. In this paper, we use statistical language R for the analysis. After the analysis, outputs like statistical values of the simulation, optimized input parameters or extracted internal state are collected and saved to the HDFS by the reduce. This process is performed in parallel, therefore faster data analysis can be possible using the proposed framework. The input and output specifications of each process are shown in Table 1.

4. Case Study

In this section, we applied a case study to demonstrate the feasibility of the proposed framework. We used 16-node Hadoop cluster, which consisted of one master machine and 15 slave machines, for the case study. Each machine within the cluster had quad-core Intel i5-3550 CPUs running at 3.3 GHz, DDR3 4 GB RAM, and was running on Ubuntu-12.04 32bit. We used Hadoop-1.1.2. The machines were connected to a gigabit Ethernet switch with two trunked gigabit ports per machine. We will first describe a simulator that depicts an air defense system and R program for statistical analysis, then we will design the experiments, and conclude by analyzing the experimental results.

4.1 Applications

4.1.1 Air Defense Simulator

In this experiment we used an air defense system simulator that detects and nullifies missiles from an enemy. It is modeled using Discrete Event System Specification (DEVS) and is running on the DEVSimLinux, discrete event system simulation engine. Its objective is to analyze the effectiveness of the air defense system depending on the variation of various parameters. The air defense simulator consists of an “experimental frame” for simulation control and a “simulation model” describing the real system. The simulation model is mainly composed of four parts: missile, radar, weapon, and a C2A (Command and Control and Alert) system model. The C2A system receives target information from radar systems, makes decisions based on algorithms, and sends attack orders to the weapon systems for air defense.

The air defense simulator represents the situation of defending a base against enemy missiles as shown in Figure 8. When missiles are fired, disposed radars detect them and send the detection information to the C2A system. Then the C2A system assigns weapon systems according to the algorithms and orders an attack to defend the base. Finally the simulator measures the time for operations and the defense rate in accordance with the success or failure of the attack.

4.1.2 R for Statistical Computing

R is a highly extensible language and environment for statistical computing and graphics. It provides a wide variety of statistical (linear and nonlinear modelling,
To get the desired results, full-factorial design is used for the input parameters and their values. We implemented a sampling module that generates a scenario file by adopting the parameters and their values as inputs for the sampling module. Therefore, the sampling module makes a total of 108,000 replications. Further, R program calculates the statistical values of simulation time and analyzes defense success rate to find the desired scenarios. We implemented an R code for statistical analysis that satisfies aforementioned requirements.

### 4.3 Experimental Result

After the experiment was completed, we got the analysis result stored in HDFS as shown in Figure 10 (a). It shows the experimental results, depicting scenario number, performance index, and statistical values such as simulation time. We can easily find the desired scenarios from the large number of scenarios. We can also perform the statistical analysis using an implementation of R.

Also, we analyze how much the proposed framework can reduce execution time. We compared total execution time of the Hadoop platform with that of single node. In this experiment, two simulators can be executed simultaneously in one node because the number of map slots is 2. So, theoretical speed up of the proposed work should be 30 times with 15 slave nodes. However, the maximum speed up was only about 27 times due to overhead of the Hadoop platform. Also, Figure 10 (b) shows that the more the scenarios increase, the more the utilization of the proposed framework also increases. This is because the ratio of Hadoop overhead becomes smaller, as the

---

**Figure 8.** Scenario of air defense simulation.

**Figure 9.** Experimental process using proposed framework.

**Table 2.** Experimental design – Parameter set

| Type of Parameter | Value of Parameter |
|-------------------|--------------------|
| Number of Replications | 1000 |
| Radar Detection Range | 3, 4, 5, 6 km |
| Number of Radars | 1, 2, 3 |
| Period of C2A | 60, 70, 80 seconds |
| Weapon Range | 1.5, 2.0, 2.5 km |
| Total Scenario Number | 1,080 ( = 4x3x3x3 ) |
| Total Replication Number | 108,000 ( = 1000x4x3x3x3 ) |

---

**Figure 10.** (a) Experimental Result Stored in HDFS (left). (b) Execution Time of Proposed Framework (right).
number of scenarios increases. Consequently, we know that the proposed parallel and distributed Monte Carlo framework is more efficient for simulations with larger numbers of scenarios.

5. Conclusions

This paper proposes a parallel simulation framework that exploits a MapReduce framework for Monte Carlo simulation. Generally, a Monte Carlo simulation requires a lot of time to execute because it needs many replicated simulation runs for meaningful statistical results. The proposed work provides an efficient framework to reduce total simulation time using parallel and distributed environments with little modification of the simulator. The sampling module of the proposed work generates input scenarios according to the simulation objective. Then, the MapReduce framework allocates the scenarios to each mapper task that assigned to a CPU core and executes the simulator. In the same way, simulation results are analyzed by statistical software allocated to each reducer task. The proposed work fully utilizes the existing Hadoop cluster without any modification or set-up.

To demonstrate the feasibility and extensibility of the proposed work, we apply a DEVS simulator describing the air defense situation. Over one hundred thousand simulations were conducted on the 16-node Hadoop cluster. Using the proposed framework, we appropriately utilized the existing hardware resource, and then we carried out a fast Monte Carlo simulation and statistical analysis through R. For future works, we need to apply several experimental techniques supporting faster data collection and data analysis.

6. References

1. Cashwell ED, Everett CJ, Rechard OW. A practical manual on the Monte Carlo method for random walk problems. Research Report LA-2120. Physics and Mathematics. Los Alamos Scientific Laboratory of the University of California. 1957.
2. Fan X, Felsovalyi A, Sivo SA, Keenan SC. SAS for Monte Carlo studies: A guide for quantitative researchers. North Carolina: SAS Institute Inc; 2002.
3. Mundfrom DJ, Schaffer J, Kim MJ, Shaw D, Thongteeraparp A. Number of replications required in Monte Carlo simulation studies: A synthesis of four studies. Journal of Modern Applied Statistical Methods. 2011; 10(1):19–28.
4. McGeoch C. Analyzing algorithms by simulation: variance reduction techniques and simulation speedups. ACM Comput Surv. 1992; 24(2):195–212.
5. Choi CB, Seo KM, Kim TG. DEXsim: an experimental environment for distributed execution of replicated simulators using a concept of single simulation multiple scenarios. Simulation. 2014; 90(4):355–76.
6. Bononi L, Bracuto M, D’Angelo G, Donatiello L. Concurrent replication of parallel and distributed simulations. In Proceedings of Workshop on Principles of Advanced and Distributed simulation; 2005 Jun 1-3; Monterey, USA. Washington, DC: IEEE Computer Society; 2005.
7. Rak M, Cuomo A, Villano U. mJADES: Concurrent Simulation in the Cloud. In Proceedings of 2012 Sixth International Conference on Complex, Intelligent and Software Intensive Systems; 2012 Jul 4-6; Palermo, Italy. Washington, DC: IEEE Computer Society; 2012.
8. Page EH, Litwin L, McMahon MT, Wickham B, Shadid M, Chang E. Goal-directed grid-enabled computing for legacy simulations. In Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing; 2012 May 13–16; Ottawa, Canada. Washington, DC: IEEE Computer Society; 2012.
9. Pratx G, Xing L. Monte Carlo simulation of photon migration in a cloud computing environment with MapReduce. J Biomed Opt. 2011; 16(12):125003.
10. Kim BS, Lee SJ, Kim TG, Song HS. MapReduce Based Experimental Frame for Parallel and Distributed Simulation Using Hadoop Platform. Paper presented at: 2014 European Conference on Modeling ans Simulation; 2014 May 27-30; Brescia, Italy.
11. Song HS, Son BK. Parallel Computing Framework for Stand-alone Monte Carlo Simulation in a Cloud Computing Environment. Proceeding of The 2nd International Conference on Digital Policy and Management; 2015 Jan 21-24; Bangkok, Thailand.
12. Metropolis N, Ulam S. The Monte Carlo Method. J Am Stat Assoc. 1949; 44(247):335–41.
13. White T. Hadoop – The Definitive Guide. Sebastopol: O’REILLY®, YAHOO!” PRESS; 2012.
14. Shvachko K, Kuang H, Radia S, Chansler R. The Hadoop Distributed File System. Paper presented at: 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies; 2010 May 3-7; Incline Village, USA.
15. Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. Commun ACM. 2008; 51(1):107–13.
16. Hadoop Streaming. 2013. Available from: http://hadoop.apache.org/docs/r1.2.1/streaming.html.
17. Antony J. Design of Experiments for Engineers and Scientists. Oxford: Butterworth-Heinemann; 2003.
18. Kim TG, Sung CH. Objective-driven DEVS modeling using OPI matrix for performance evaluation of discrete
event systems. Proceedings of the 2007 Summer Computer Simulation; 2007 Jul 15-18; San Diego, USA. San Diego: Society for Computer Simulation International; 2007.

19. Cho BG, Kim DY, Kim SH, Youn C. Real-time distributed simulation environment for air defense system using a software framework. J Def Model Simul. 2007; 4(2):64–79.

20. Zeigler BP, Praehofer H, Kim TG. Theory of Modeling and Simulation, Second Edition. Waltham: Academic Press; 2000.

21. Kim TG, Sung CH, Hong SY, Hong JH, Choi CB, Kim JH, Seo KM, Bae JW. DEVSim++ Toolset for Defense Modeling and Simulation and Interoperation. J Def Model Simul. 2011; 8(3):129–42.

22. James G, Witten D, Hastie T, Tibshirani R. An Introduction to Statistical Learning: with Applications in R. New York: Springer; 2013.