Volume 6, Issue 3, March 2016 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

An Energy-Saving Task in Cloud Computing Using Triggering Mechanism

K. Pandi Kumar (Asst. Prof.), S. Karthikeyan, D. Gowsik
Computer Science and Engineering, Dhanalakshmi College of Engineering,
Chennai, Tamilnadu, India

Abstract*: *High energy consumption is one of the key issues of cloud computing systems. Incoming jobs in cloud computing environments have the nature of randomness, and compute nodes have to be powered on all the time to await incoming tasks. This results in a great waste of energy. An energy-saving task scheduling algorithm based on the vacation queuing model for cloud computing systems is proposed in this paper. First, we use the vacation queuing model with exhaustive service to model the task schedule of a heterogeneous cloud computing system. Next, based on the busy period and busy cycle under steady state, we analyze the expectations of task sojourn time and energy consumption of compute nodes in the heterogeneous cloud computing system. Subsequently, we propose a task scheduling algorithm based on similar tasks to reduce the energy consumption. Simulation results show that the proposed algorithm can reduce the energy consumption of the cloud computing system effectively while meeting the task performance.

Keywords: cloud computing; independent task scheduling; energy-saving; Triggering mechanism.

I. INTRODUCTION

When you store your photos online instead of on your home computer, or use webmail or a social networking site, you are using a "cloud computing" service. If you are an organization, and you want to use, for example, an online invoicing service instead of updating the in-house one you have been using for many years, that online invoicing service is a "cloud computing" service. Cloud computing refers to the delivery of computing resources over the Internet. Instead of keeping data on your own hard drive or updating applications for your needs, you use a service over the Internet, at another location, to store your information or use its applications. Doing so may give rise to certain privacy implications. For that reason the Office of the Privacy Commissioner of Canada (OPC) has prepared some responses to Frequently Asked Questions (FAQs). We have also developed a Fact Sheet that provides detailed information on cloud computing and the privacy challenges it presents. As sleep states on different systems can be quite different, we reference existing sleep states the compute node will switch to sleep state with low power.

System Architecture

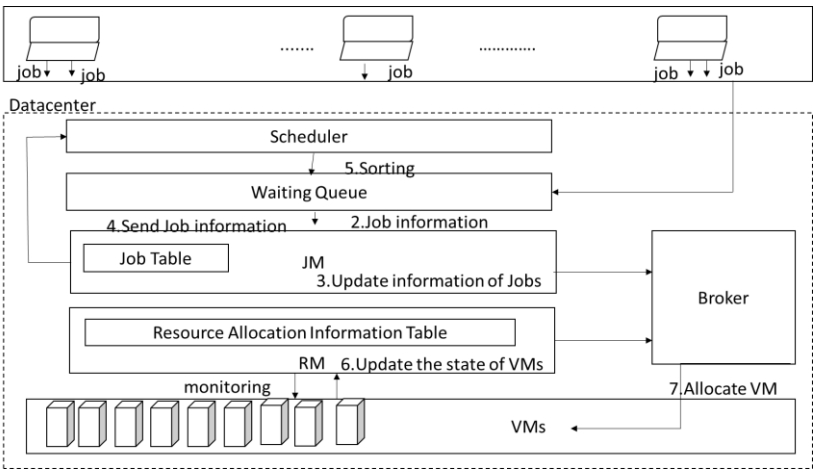


Fig. System Architecture

II. RUNNING

Their characteristics and requirements are varied tasks are independent. In addition, compute nodes running tasks are heterogeneous in cloud computing systems they have different hardware configurations, different computing capacities, and various power states. The time and power needed to switch state are also different. Compute node maintains its own task queue. When performing a task, a compute node is in the running state with high power. The Running state when a compute node is working if there are no tasks arriving at a compute node, the node goes through an idle period to avoid

frequent switches from the deep sleep state. The intervals of task arrival and service time for each compute node are independent. The task scheduler runs in the global dispatching central node. It receives task requests that arrive dynamically, and uses the first come first served strategy to perform the task scheduling algorithm and dispatch tasks to appropriate compute nodes. Compute nodes accept and perform tasks.

III. LAST OF RECOVERING

Today's computers generally support power saving states. The recovering time, namely wake up latency. The compute node in the idle state can receive tasks at once, therefore, the transition cost from idle state to running state can be ignored. Recovering state when a task arrives the compute node need to recover and then start to main the task. Transition state was also recovering state. Recovering state transitioned and woken up from sleep state to the running state. Recovering state switches different power state of node.

IV. MODULES

Module I: Request New Jobs and Jobs Information

- Users send a service jobs to the cloud service system. The service jobs arrive at the waiting queue.
- JM gets the information of the job regarding nj , tj , and worst-case execution time (WCET) from the waiting queue.

Module II: Update Jobs Information

- JM manages the information in the Job Table, and updates whenever a new job is arrived.
- To sort the jobs, the scheduler gives deadline scores to jobs.
- The system gives importance to all the jobs in by using round robin algorithm.

Module III: Sorting the Jobs into Priority Wise

- When the jobs allocate to the server, server will sort the jobs into priority .
- Then the jobs will be sent to the queue .

V. .EXISTING SYSTEM

Existing datacentres is less than 30%. One reason is that cloud computing provides on-demand services with incoming jobs that are stochastic, at times dense, and at other times sparse.

In order to meet the requirements of tasks in time, the cloud datacentre keeps compute nodes powered on, waiting for tasks to arrive. As a result, for most of the time, compute nodes in the cloud are in an idle state, which leads to a significant waste of energy[.

Disadvantage:

- Waiting for tasks to arrive.
- More time in ideal stage.

VI. PROPOSED SYSTEM

It analyse the expectations of task sojourn time and energy consumption of a cloud computing system based on the busy period and busy cycle under steady state[1].Based on our analysis of the partial derivatives of energy consumption with respect to idle time and the variance of service time.

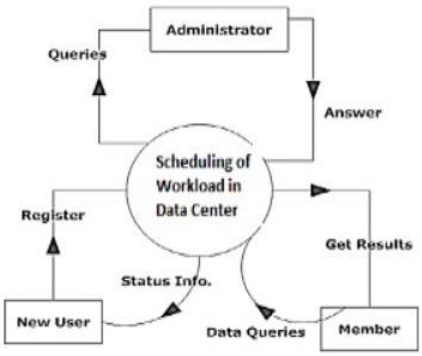
IT conclude that energy can be saved by reducing the variance of service time while[3] scheduling tasks. Based on our analysis, a task scheduling algorithm based on similar tasks to optimize energy consumption[4].

Advantage:

- Avoid ideal stage.
- More energy saving.

Data Flow Diagram

Data Flow Diagrams (DFD) of Heterogeneous Workload Consolidation Technique implemented by designing a Workload Consolidation Cloud Portal (WC Cloud Portal).[2]



VII. SOFTWARE ENVIRONMENT

A. Java Technology

Java is a general-purpose computer programming language that is concurrent[5], class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another.[8]

Java applications are typically compiled to byte code that can run on any Java virtual machine (JVM) regardless of computer architecture[10]. Java is, as of 2015, one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers. Java was originally developed by James Gosling at Sun Microsystems (which has since merged into Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. [12]

The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them. The original and reference implementation Java compilers, virtual machines, and class libraries were originally released by Sun under proprietary licenses. As of May 2007, in compliance with the specifications of the Java Community Process, Sun relicensed most of its Java technologies under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java (byte code compiler), GNU Class path (standard libraries), and Iced Tea-Web (browser plugin for applets).

B. Java Platform

One design goal of Java is portability, which means that programs written for the Java platform must run similarly on any combination of hardware and operating system with adequate runtime support. This is achieved by compiling the Java language code to an intermediate representation called Java byte code, instead of directly to architecture-specific machine code. Java byte code instructions are analogous to machine code, but they are intended to be executed by a virtual machine (VM) written specifically for the host hardware. End users commonly use a Java Runtime Environment (JRE) installed on their own machine for standalone Java applications, or in a web browser for Java applets.

Standardized libraries provide a generic way to access host-specific features such as graphics, threading, and networking. A major benefit of using byte code is porting. However, the overhead of interpretation means that interpreted programs almost always run more slowly than programs compiled to native executable would. Just-in-Time (JIT) compilers were introduced from an early stage that compiles byte codes to machine code during runtime. Java is platform independent. But as Java virtual machine must convert Java byte code into machine language which depends on the operating system being used, it is platform dependent.

C. Java Virtual Machine

The heart of the Java platform is the concept of a "virtual machine" that executes Java byte code programs. This byte code is the same no matter what hardware or operating system the program is running under. There is a JIT (Just In Time) compiler within the Java Virtual Machine, or JVM. The JIT compiler translates the Java byte code into native processor instructions at run-time and caches the native code in memory during execution. The use of byte code as an intermediate language permits Java programs to run on any platform that has a virtual machine available.

The use of a JIT compiler means that Java applications, after a short delay during loading and once they have "warmed up" by being all or mostly JIT-compiled, tend to run about as fast as native programs.Since JRE version 1.2, Sun's JVM implementation has included a just-in-time compiler instead of an interpreter. Although Java programs are cross-platform or platform independent, the code of the Java Virtual Machines (JVM)[7] that execute these programs is not. Every supported operating platform has its own JVM.

D. Java Class Library

The Java Class Library (JCL) is a set of dynamically loadable libraries that Java applications can call at run time. Because the Java Platform is not dependent on a specific operating system, applications cannot rely on any of the platform-native libraries. Instead, the Java Platform provides a comprehensive set of standard class libraries, containing the functions common to modern operating systems.[14].

JCL serves three purposes within the Java Platform:

Like other standard code libraries, they provide the programmer a well-known set of useful facilities, such as container classes and regular expression processing.

The library provides an abstract interface to tasks that would normally depend heavily on the hardware and operating system, such as network access and file access.

Some underlying platforms may not support all of the features a Java application expects. In these cases, the library implementation can either emulate those features or provide a consistent way to check for the presence of a specific feature.

VIII. TESTING

Software Testing

Testing a program consists of providing the program with a set of test inputs (or test cases) and observing if the program behaves as expected. If the program fails to behave as expected, then the conditions under which failure occurs are noted for later debugging and correction.The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of

components, sub-assemblies, assemblies and/or a finished product it is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

Unit Testing

Unit testing is undertaken after a module has been coded and successfully reviewed. Unit testing (or module testing) is the testing of different units (or modules) of a system in isolation. In order to test a single module, a complete environment is needed to provide all that is necessary for execution of the module. That is, besides the module under test itself, the following steps are needed in order to be able to test the module Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produces valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of business process performs accurately to the documented specifications and contains clearly defined inputs and expected result

Integration Testing:

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Fuctional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation and user manuals.

Functional testing is centered on the following items:

Valid Input: is used to identified classes of valid input must be accepted.

Invalid Input: is used to identified classes of invalid input must be rejected.

Functions is used to identified functions must be exercised.

OOTPUT is used to identify classes of application outputs.

Systems/Procedures: is used to interfacing systems or procedures must be invoked. Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive

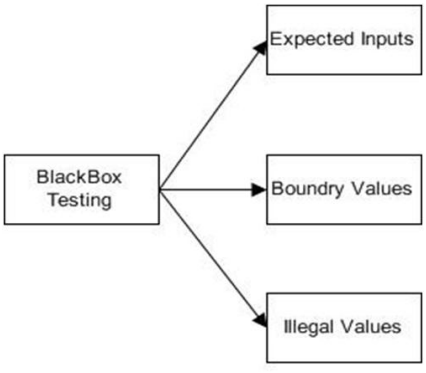
Processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Testing:

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process link and integration points.

Black Testing:

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements Document. It is a testing in which the software under test is treated, as a black box .you **cannot "see" into it. The test provides inputs and responds to outputs without** Considering how the software works.



Test Strategy and Approach

Field testing will be performed manually and functional tests will be written in Detail.

Test Objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.
- Features to be tested
- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Alpha Testing

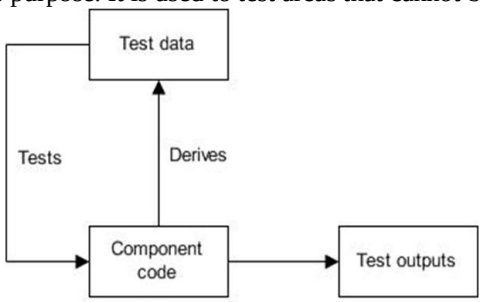
In software development, alpha test will be a test among the teams to confirm that your product works. Originally, the term alpha test meant the first phase of testing in a software development process. The first phase includes unit testing, component testing, and system testing. It also enables us to test the product on the lowest common denominator machines to make sure download times are acceptable and freeloaders work.

Beta Testing

In software development, a beta test is the second phase of software testing in which a sampling of the intended audience tries the product out. Beta testing can be considered "pre-release testing." Beta test versions of software are now distributed to curriculum specialists and teachers to give the program a "real-world" test.

White Testing:

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.



The task scheduling of a heterogeneous cloud computing system using the vacation queuing theory. It analyze the average sojourn time of tasks and the average power of compute nodes in the heterogeneous cloud computing system under steady state. A task scheduling algorithm based on similar tasks. Simulation results show that the proposed algorithm can ensure task performance, while reducing the energy cost of a cloud computing system effectively.

ACKNOWLEDGMENT

The heading of the Acknowledgment section and the References section must not be numbered.

Causal Productions wishes to acknowledge Michael Shell and other contributors for developing and maintaining the IEEE LaTeX style files which have been used in the preparation of this template. To see the list of contributors, please refer to the top of file IEEETran.cls in the IEEE LaTeX distribution.

REFERENCES

[1] L. G. Valentini, W. Lassonde, U. S. Khan, N. Min-Allah, S. A. Madani, L. Juan, L. Zhang, L. Wang,N. Ghani, J. Kolodziej, et al., An overview of energyefficiency techniques in cluster computing systems,Cluster Computing, vol. 16, pp. 3–15, 2013.

[2] R. J, R. Kuo and C. W. Cheng, Hybrid meta-heuristicalgorithm for job shop scheduling with due date timewindow and release time,The International Journal ofAdvanced Manufacturing Technology, vol. 67, pp. 59–71,2013.

[3] G. Shen and Y. Zhang, Power consumption constrainedtask scheduling using enhanced genetic algorithms, in Evolutionary Based Solutions for Green Computing. Springer Berlin Heidelberg, 2013, pp. 139159.

[4] L. X. Wang, P. Y. Wang, and H. Zhu, Energy-efficientmulti-job scheduling model for cloud computing and itsgenetic algorithm,Mathematical Problems in Engineering,vol. 10, pp. 1–16, 2012.

[5] M. Y. Tan, S. G. Zeng, and W. Wang, Policy of energyoptimal management for cloud computing platform withstochastic tasks,Journal of Software, vol. 23, pp. 266–278,2012.

[6] S. Zikos and D. H. Karatza, Performance and energy awarecluster-level scheduling of compute-intensive jobs withunknown service times,Simulation Modeling Practice andTheory, vol. 1, pp. 239–250, 2011.

[7] L. Gong, H. X. Sun, and F. E. Waston, Performancemodeling and prediction of non-dedicated networkcomputing,IEEE Transactions on Computers, vol. 51, pp.1041–1055, 2002.

[8] W. Wang, Z. J. Luo, and B. A. Song, Dynamic pricingbased energy cost optimization in data center environment,Journal of Computers, vol. 36, pp. 600–615, 2013.

[9] Y. Z. Ma, The steady state theory of M/G/1 type multipleadaptive vacation queueing system, Ph. D. dissertation,Yanshan University, Qinhuangdao, China, 2006.

[10] A. Gandhi, M. Harchol-Balter, and A. M. Kozuch, Aresleep states effective in data centers? InS2012 InternationalConference on Green Computing (IGCC), 2012, pp. 1–10.

[11] M. Ware, K. Rajamani, M. Floyd, B. Brock, J. C. Rubio,F. Rawson, and J. B. Carter, Architecting for powermanagement: The IBM POWER 7 approach, inProc.IEEE 16th International Symposium on High PerformanceComputer Architecture, HPCA, 2010.

[12] H. Blume, V. J. Livonius, L. Rotenberg, T. G. Noll, H.Bothe, and J. Brakensiek, OpenMP-based parallelizationon an MPcore multiprocess or platform—A performanceand power analysis,Journal of Systems Architecture, vol.54, pp. 1019–1029, 2010.

[13] X. Y. Shi, H. X. Jiang, and J. K. Ye, An energy-efficient scheme for cloud resource provisioning based onCloudSim, inIEEE International Conference on ClusterComputing, 2011, pp. 595–599.

[14] D. T. Braun and H. Siegel, A comparison of eleven staticheuristics for mapping a class of independent tasks ontoheterogeneous,Journal of the Parallel and DistributedComputing, vol. 61, pp. 810–837, 2011.

