The Anatomy Study of Load Balancing in Cloud Computing Environment

Shu-Ching Wang, Ching-Wei Chen
Department of Information Management
Chaoyang University of Technology
Taiwan, R.O.C.
{scwang; s10114901}@cyut.edu.tw

Kuo-Qin Yan, Shun-Sheng Wang
Department of Business Administration
Chaoyang University of Technology
Taiwan, R.O.C.
{kqyan; sswang}@cyut.edu.tw

Abstract—**In recent years, network bandwidth and quality have improved dramatically, in fact, much faster than the enhancement of computer performance. Cloud computing is an Internet-based resource sharing system in which virtualized resources are provided as a service to users over the Internet. Cloud computing refers to a class of systems and applications that employ distributed resources for use in various applications; these computing resources (service nodes) are utilized over a network to facilitate the execution of complicated tasks. However, Cloud computing resources are heterogeneous and dynamic, connecting a broad range of resources. Thus, when selecting nodes for the execution of a task, the dynamic nature of Cloud computing nodes must be considered. To most effectively utilize the available resources, they have to be properly selected according to the requirements of each task. This study proposes a hybrid load balancing policy to maintain the efficient performance and stability of a Cloud computing environment.**

Keywords-Distributed System; Cloud Computing; Scheduling; Load Balancing; Makespan

I. INTRODUCTION

The Internet is a constantly and rapidly developing global network system, and in order to keep pace with its development, network bandwidth must also constantly develop. Cloud computing is one of such developments, allowing for more applications for Internet users [1,6,7]. Cloud computing environments consist of many commodity nodes that can cooperate to perform specific services.

Users are able to access operational capabilities in Cloud computing environments much faster than they could with Internet applications [3]. However, the infrastructure of the Internet is continuously growing and evolving, progressively allowing the provision of ever more Internet application services. In a distributed computing system, components allocated to different places or in separate units are connected so that they may collectively be used to greater advantage [4]. In addition, Cloud computing has greatly encouraged distributed system design and applications to support user-oriented service applications [7]. Furthermore, many Cloud computing applications, such as YouTube, offer greater user convenience [7].

As technology advances, Cloud computing provides better large-scale resource sharing in a broad-field Internet access environment [1,14]. The limitation of space on conventional distributed systems can thus be eliminated in order to achieve cross-platform compatibility, and to fully exploit the significant resources of all available computers [1,14].

Cloud computing over the Internet provides many applications for users, like Facebook, YouTube, etc. Therefore, determining how best to utilize the advantages of Cloud computing and to ensure that each task is assigned the required resources in the shortest possible time is an important task.

Resources are distributed in a Cloud computing environment, and the stability and performance of each resource varies. In other words, Cloud computing environments are dynamic and composed of heterogeneous resources. Thus, resource selection and task distribution are of particular importance. This study proposes a hybrid load balancing policy that selects an effective node set in the static load balancing stage in order to lower the odds of ineffective nodes being selected, and makes use of the dynamic load balancing stage to ensure that tasks and resources are efficiently balanced. When a node status is changed, a new substitute can be located in the shortest time to maintain execution performance.

The remainder of this paper is organized as follows. Section II focuses on related works. The proposed hybrid load balancing policy is described in Section III. Section IV discusses the design of the simulation experiment. Section V provides the experiment results. Finally, conclusions are presented in Section VI.

II. RELATED WORKS

Cloud computing is a form of distributed computing in which massively scalable IT-related capabilities are provided to multiple external customers “as a service” using Internet technologies [14]. Amazon [10] provides many applications through Amazon Web Services (AWS) as a Cloud computing environment, allowing users to rent required infrastructure or application services [11]. With AWS, users can request computing power, storage and other services, and then access suitable IT infrastructure services on demand [11].

Cloud providers have to achieve a large, general-purpose computing infrastructure and virtualization of that infrastructure for different customers and services in order to provide multiple application services. Furthermore, a software package developed by ZEUS allows Cloud providers to easily and cost-effectively offer every customer a dedicated application delivery solution [15]. The network

 ISBN: 978-1-61208-280-6

framework provided by ZEUS can also be used to develop new Cloud computing methods [13-15]. Based on the ZEUS network framework and the properties of Cloud computing structures, this study uses a three-level hierarchical topology.

However, since a multi-level hierarchical network topology can increase the resource cost of data storage [2], Wang et al. proposed a three-level hierarchical framework [8]. In this framework, service nodes in the third level of the framework are used to execute subtasks, service managers in the second level are used to divide the task into logical independent subtasks and a request manager in the first level is used to assign tasks to a suitable service manager.

The system performance of a Cloud computing environment can be managed and enhanced based on comprehensive status information of each node in the system. There are several methods of collecting the relevant information from nodes, including broadcasting, polling and agents.

Agents have been used extensively in recent years [9]. They have inherent navigational autonomy, and can ask to be sent to other nodes. In other words, an agent does not have to be installed on each visited node, and can collect the relevant information from each node participating in a Cloud computing environment, such as CPU utilization, remaining CPU capability, remaining memory, transmission rate, etc. Therefore, when an agent is dispatched, it does not require any control or connection, and travel flow in maintaining system can be reduced [13]. In this study, the agent is used to gather relevant information, and to reduce wasted resources.

This study also includes a system load balancing policy, and a scheduling algorithm for heterogeneous resources [5, 13]. Generally, load-balancing policies for distributed systems can be categorized into static and dynamic policies [5]. Static load balancing uses simple system data, and based on these data, tasks are distributed through mathematic formulas or other adjustment methods [5]. Dynamic load balancing determines how best to assign tasks to each node in the distributed system. When the system is overloaded, the task causing the overloading will be moved to other nodes and processed for dynamic balance. However, this migration of tasks induces extra system overhead [5].

Therefore, task scheduling will affect the load balancing performance of a system. The following are two typical task-scheduling methods:

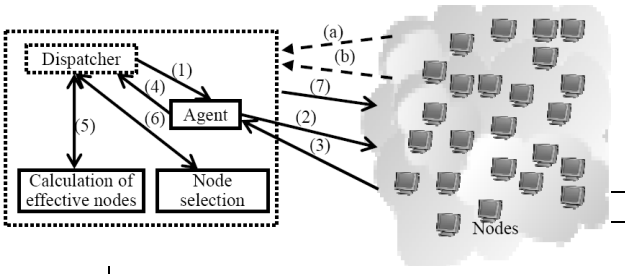
(1) Minimum Completion Time (MCT) assigns each task, together with the minimum expected completion time of each task, to nodes in arbitrary order [5]. This results in some tasks being assigned to nodes that do not have the required minimum execution time for that task [3].

(2) Min-min establishes the minimum completion time for every unscheduled task, and then assigns the tasks to nodes based on the minimum completion offered by each node. The minimum completion time for all tasks is considered, and Min-min can schedule tasks in such a way as to achieve the lowest overall make-span [3].

Many load balancing polices and scheduling algorithms are used to maintain system performance. However, the number of available nodes changes constantly. System performance maintenance, therefore, becomes a complex and difficult process in this dynamic environment. This paper, therefore, proposes a hybrid load balancing policy in order to achieve efficient load balancing.

III. THE HYBRID LOAD BALANCING POLICY

In this section, the proposed hybrid load balancing policy is explained. The structure of the proposed system consists mainly of a dispatcher and nodes. The relationship of roles in this hybrid load balancing policy is described in Figure 1.



(a) When any node can provide its resources, a "join" message and related hardware information will be transmitted to the dispatcher.
(b) When a node can no longer provide resources, it transmits an "exit" message to the dispatcher.

The progressive stages of static load balancing are as follows:
(1) When a request to execute a task is proposed, the dispatcher dispatches an agent to collect the relevant information of each node.
(2) The agent collects the relevant information of the node, such as remaining CPU capability, remaining memory, etc.
(3) The node send its information to the agent.
(4) The agent provides all node-related information to the dispatcher.
(5) The dispatcher builds a table of effective nodes.
(6) The dispatcher selects an effective node set from the table of effective nodes by node selection.
(7) The dispatcher assigns subtasks to the selected nodes.

Figure 1. The interaction of roles

The objectives of the dispatcher include maintaining the load balance, monitoring the status of each node, selecting the nodes for task execution, and assigning tasks for each node. In order to ensure the efficiency of the dispatcher in performing these tasks, an agent will be designed as follows:

(1) The mechanism of an agent mainly collects the relevant information of each node. The information will be provided to the dispatcher to maintain the load balancing of system.

(2) Many factors are considered when a node is selected. Thus, a Value Function (VF) [9] is given to

 ISBN: 978-1-61208-280-6

determine the value of each candidate node, and to provide a reference for selecting effective nodes.

(3) All candidate nodes will be organized into a table of effective nodes. Whenever each node joins or exits the system, the node table will be updated. When any node in the execution aggregate accomplishes its assigned task, it will be transferred to the waiting aggregate, ready for another assignment.

Nodes assist in the execution of tasks in this system. When any node is available or in a busy state, it must transmit its status message to the dispatcher.

In order to maintain system load balancing, this study proposes a hybrid load balancing policy. The proposed policy is carried out in two phases. In the first phase, a static load balancing policy selects an appropriate node for each task. In the second phase, a dynamic load balancing policy, a new node is found to take over the task as soon as the task cannot be completed by the assigned node.

When a request for task execution is made, the task must be divided into several subtasks. The lowest requirements of each subtask determine the threshold on the table of candidate nodes. Nodes passing the threshold are considered as candidate nodes. All candidate nodes can be organized and built into a table of nodes optimized for the proposed task, and the number of required nodes can be determined. If the total number of nodes in the table is smaller than the required number, a portion of subtasks will first be assigned to effective nodes, and the remaining subtasks will be processed when new nodes are added to the table of candidate nodes.

In a dynamic distributed system, the effectiveness of nodes may vary with time. The variation of node status can be identified in two conditions. First, when the dispatcher receives the message that a certain node can no longer provide resources, and second, when the execution of a certain node exceeds the expected time. When either of the above conditions occurs, the dispatcher will launch the agent mechanism for confirmation. If the node remains effective, the distribution of tasks will not be readjusted, but the node's execution of the task will re-estimated. If the node is confirmed to be ineffective, the highest value available node will be selected to replace the ineffective node.

IV. DESIGN SIMULATION EXPERIMENT

In a heterogeneous Cloud computing environment, the performance of nodes varies. In addition, subtasks actually vary in size. Thus, task completion time may vary with the execution order. The properties of both MCT and Min-min are suitable for this experiment, and will be employed and compared with our proposed method. The progression of the experiment is as follows:

(1) The task is divided into 10 independent subtasks.

(2) 10 nodes are selected and assigned tasks by the three different task-scheduling methods.

(3) If any of nodes cannot complete the assigned subtask, new nodes are selected to take over, and the task is then redistributed and re-executed.

According to the above assumptions, this experiment is carried out in two stages. In the first stage, the network simulator, Network Simulation Version 2 (NS-2) [12], is used to dynamically create a Cloud computing environment. In the second, Cloud computing environments with 100, 200, 300, 400, 500, 600, 700, 800, 900 and 1000 nodes are dynamically created with NS-2. Randomly sized data packages are generated and transmitted at a constant bit rate. The transmission rates between the dispatcher and each node are tested. To simulate the heterogeneity of nodes in Cloud computing environments, the CPU capability, memory size, CPU usage and memory usage, and past task completion rate of each node are randomly generated. In addition, the effective time of each node is generated at random, and then multiplied by the past task completion rate in order to reflect the relation between the past task completion rate and the effective time of the node.

According to the Computing Resources (CR) and Amount of Data Transmission (ADT) required to execute the task [7], four scenarios are given:

Scenario 1: CR is large, and ADT is small
Scenario 2: CR is small, and ADT is large
Scenario 3: CR and ADT are large
Scenario 4: CR and ADT are small

In the VF, decision variables can be given different settings, according to the factor focused in the actual application. In the experiment, the available CPU capacity, size of available memory, transmission rate and the past completion rate were the four factors regarded as the threshold for the VF to select nodes and the decision variables for the nodes to estimate their values.

After the decision variables of VF are determined, to make every decision variable comparable, each variable must be quantified. In this experiment, the available CPU capacity and the size of available memory are quantified by the percentage of remaining CPU capacity and memory of each node. Because the transmission rate between the dispatcher and each node is limited to their network bandwidth, the network bandwidth of the dispatcher is taken as the denominator to quantify the transmission rate of each node.

Based on the abovementioned four scenarios, task completion time and number of task redistributions are factors for evaluation. To verify that the nodes selected by VF perform better than those selected by other methods, VFs of different sets of weight are evaluated, and each set is simulated 100 times to obtain objective data. Of the VFs of different weights, the worst weight set that produces the longest completion time (VF-MAX), the average completion time of all weights set (VF-AVG) and the best weight set that produces the minimal completion time (VF-MIN) will be compared to other task-scheduling methods

 ISBN: 978-1-61208-280-6

using the task completion time and number of task redistributions.

V. EXPERIMENT RESULTS AND ANALYSES

The experiment results of Scenario 1 (Figures 2 and 3) indicate that the nodes with the largest CPU resources are selected first, disregarding the execution order for subtasks in MCT. The Min-min selects the combination with the node and subtask that consume the shortest time, so the required task completion time is shorter than in MCT. However, Min-min does not consider the memory provided by the node; subtasks may be re-distributed to nodes with insufficient memory. In addition, the VF considers not only CPU capacity, but also memory and transmission rate and past task completion rate of each node. Even if the task completion time with the worst weight set is close to Min-min, VF still consumes a shorter completion time than Min-min does, and the number of task redistributions are fewer than in Min-min. Thus, VF is more effective in maintaining the load balance.

When the amount of data transmitted is very large, a significant amount of time may be spent on data transmission. This will result in a node being unable to complete the assigned subtask in the effective time, and the subtask will have to be redistributed and re-executed (Scenario 2). As the VF considers these factors, under the various weight sets, fewer task redistributions (Figure 4) and a lower task completion time (Figure 5) are required.

From Figures 6 and 7, it is known that MCT does not consider the execution order. The largest subtask may not be able to find the node with the largest resources, and appropriate nodes cannot be searched for, or assigned to subtasks (Scenario 3). Min-min selects nodes by the shortest completion time instead of the order of subtasks. Therefore, it requires a shorter task completion time than does MCT. As the VF considers multiple factors, nodes that can provide stable resources will be selected first. Even in the worst weight set, task completion time is greater than in Min-min, but with fewer task re-distributions. Therefore, the nodes selected by the VF may not be able to produce the best results in all weight sets; however, they are still effective in maintaining the load balance of a system.

Figures 8 and 9 show that MCT and Min-min only consider CPU capability, and not the memory and transmission rate (Scenario 4). Thus, during task execution, subtasks may be redistributed and a longer task completion time may be incurred. The VF considers multiple factors, so a shorter completion time is required than in almost all the other methods. In addition, task redistribution is almost unnecessary. In other words, the task can be completed in the effective time in almost all cases.

In a Cloud computing environment, the nodes are composed of resources. Since each node has a different hardware structure, nodes cannot be selected based on a single condition (such as available CPU capacity). Therefore, the properties of the task to be executed must be considered. It is known from the above results that when selecting nodes, if the properties of a task and the resources that nodes can provide are not considered, the task to be executed will be repeatedly reassigned and re-executed, thus prolonging the task completion time and lowering the execution performance of system. The VF, however, takes the node resources, transmission rate and past completion rate into consideration. By estimating the value of each node using these factors, the nodes that can provide relatively better resources will be selected. The results of the experiments prove that the hybrid load balancing policy, whether in terms of the best weight set, the worst weight set, or the average time, is far more effective than the other methods in reducing the number of task redistributions and completion time, as well as enhancing the execution performance of the system.

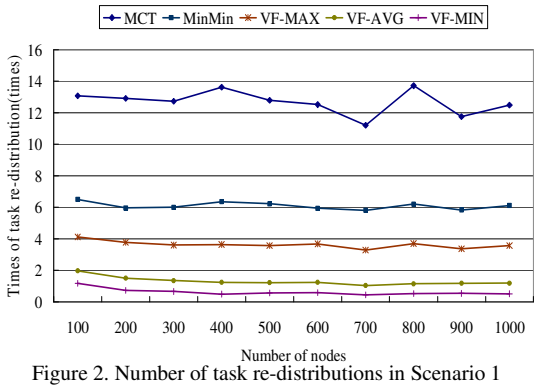


Figure 2. Number of task re-distributions in Scenario 1

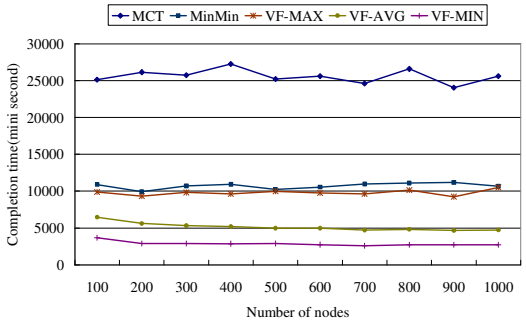


Figure 3. Task completion time in Scenario 1

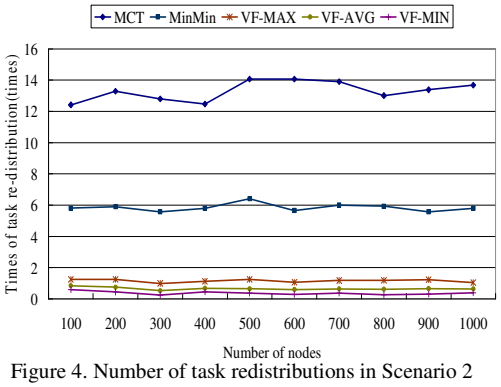


Figure 4. Number of task redistributions in Scenario 2

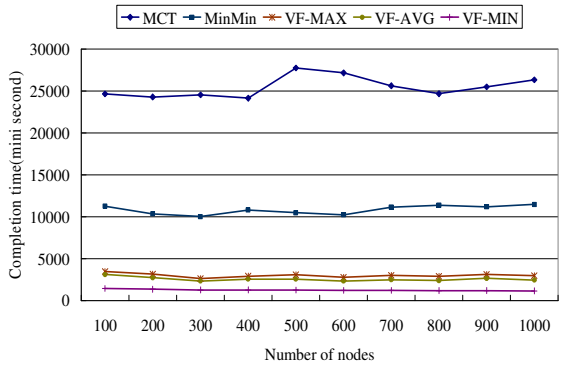


Figure 5. Task completion time in Scenario 2

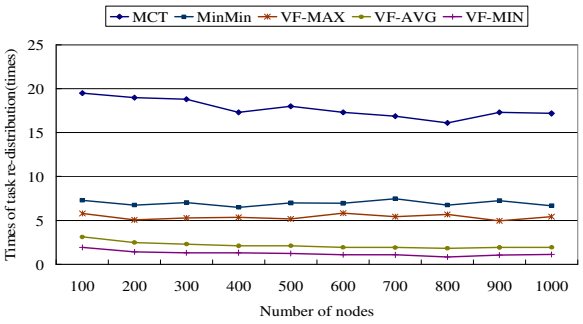


Figure 6. Number of task redistributions in Scenario 3

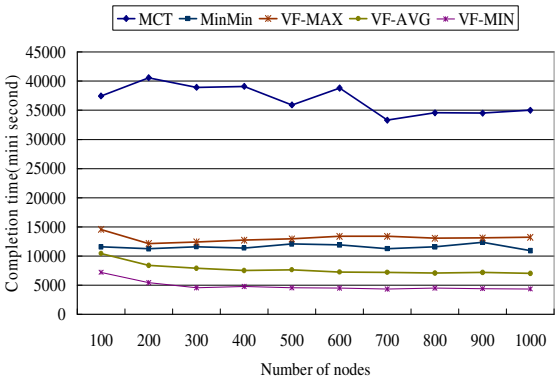


Figure 7. Task completion time in Scenario 3

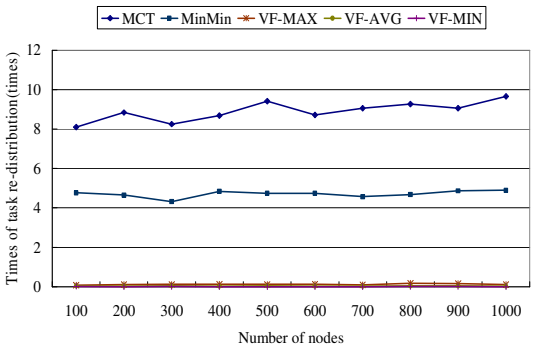


Figure 8. Number of task redistributions in Scenario 4

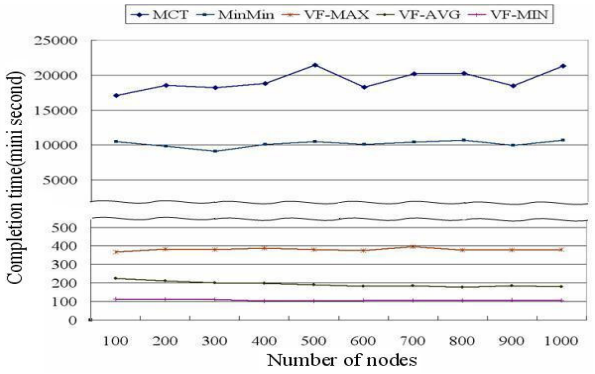


Figure 9. Task completion time in Scenario 4

VI. CONCLUSION AND FUTURE WORK

Cloud computing environments offer many available resources; however, the availability of nodes that provide these resources dynamically changes over time. In this paper, a hybrid load balancing policy is proposed for Cloud computing environments in order to efficiently distribute tasks to available nodes with the required resources for the completion of those tasks in the shortest possible time. The proposed policy is carried out in two phases. In the first phase, a static load balancing policy selects an appropriate node for each task. In the second phase, a dynamic load balancing policy, a new node is found to take over the task as soon as the task cannot be completed by the assigned node.

Since Cloud computing environments are more complicated than traditional distributed systems, it follows that if this policy can achieve efficient load balancing in Cloud computing environments, then it can also solve load balancing issues in other distributed systems.

ACKNOWLEDGMENTS

This work was supported in part by the Taiwan National Science Council under Grants NSC101-2221-E-324-032.

REFERENCES

[1] Y. Gong, Z. Ying, and M. Lin, “A survey of cloud computing,” Lecture Notes in Electrical Engineering, vol. 225, 2013, pp. 79-84.

[2] C.L. Hu and T.H. Kuo, “A hierarchical overlay with cluster-based reputation tree for dynamic peer-to-peer systems,” Journal of Network and Computer Applications, vol. 35, Issue 6, November 2012, pp. 1990-2002.

[3] S. Nesmachnow, F. Luna, and E. Alba, “An efficient stochastic local search for heterogeneous computing scheduling,” Proc. IEEE 26th International Parallel and Distributed Processing Symp. Workshops & PhD Forum (IPDPSW), 21-25 May 2012, pp. 593-600.

[4] N. Olifer and V. Olifer, Computer network: principles, technologies and protocols for network design. John Wiley & Sons, 2006.

[5] B. Sahoo, D. Kumar, and S.K. Jena, “Observing the performance of greedy algorithms for dynamic load balancing in heterogeneous distributed computing system,” Proc. 1st International Conf. on Computing, Communication and Sensor Networks- CCSN,(2012), vol. 62, 2012, PIET, Rourkela, Odisha, pp. 265-269.

 ISBN: 978-1-61208-280-6

[6] A. Vouk, “Cloud computing- issues, research and implementations,” Information Technology Interfaces, June 2008, pp. 31-40.

[7] L.H. Wang, J. Tao, and M. Kunze, “Scientific cloud computing: early definition and experience,” Proc. 10th IEEE International Conf. on High Performance Computing and Communications, 2008, pp. 825-830.

[8] S.C. Wang, W.P. Liao, K.Q. Yan, and S.S. Wang, “Towards a load balancing in a three-level cloud computing network,” Proc. 2010 3rd IEEE International Conference on Computer Science and Information Technology (IEEE ICCSIT 2010), Chengdu, China, 9-11 July 2010, pp. 108-113.

[9] K.Q. Yan, S.C. Wang, C.P. Chang, and J.S. Lin, “A hybrid load balancing policy underlying grid computing environment,” Computer Standards & Interfaces, 2006.

[10] Amazon web services, http://aws.amazon.com/, April 2, 2013.

[11] Application delivery networking, application acceleration, Internet traffic management system: Zeus.com, http://www.zeus.com/, April 2, 2013.

[12] The network simulator - NS-2, http://www.isi.edu/ nsnam/ns/, April 2, 2013.

[13] Load balancing, load balancer, http://www.zeus.com/products/ zxtmlb/index.html, April 2, 2013.

[14] What is cloud computing?, http://www.zeus.com/Cloud_ computing/Cloud.html, April 2, 2013.

[15] ZXTM for cloud hosting providers, http://www.zeus.com/Cloud_ computing/ for_ Cloud providers.html, April 2, 2013.

 ISBN: 978-1-61208-280-6