FAIR RESOURCE ALLOCATION USING GOSSIP PROTOCOL AND MERGING CLOUD

S.ABDUL ANSARI[*]

K.RAMIAH ILANGO[**]

***Abstract*:**

In this project, we give an optimal solution for resource allocation problem and an efficient heuristic for the load balancing problem. Objective is to create a decentralized system to allocate resource dynamically in cloud to avoid the demand for resources.Cloud tenant running a collection of virtual appliances that are hosted on the cloud infrastructure, with services provided to end users through the public Internet.In large-scale, dynamic, fully distributedsystems constructing and maintaining a topology is a highly non-trivial problem. In this paper we identify topology management as an abstract service, independent of application or function, and propose a gossipbased scheme called T-MAN for the construction of a large class of different topologies. The topology is defined by a ranking function which can be based on any appropriate property of components, including geographical location, semantic proximity, bandwidth, or simply on an abstract, pre-defined topology overran ID space like a ring or a torus.A gossip protocol is a style of computer-to-computer communication protocol used for easily access the data in network. Modern distributed systems often use gossip protocols to solve problems because it allocates the resource in network for user access data to avoid delay.

KEYWORDS:*Gossip protocol, epidemic protocol, aggregation, Real-time monitoring.*

[*] Final B.E, Dept. of CSE, SBM College of Engineering & Technology, Dindigul

[**] Asso.Prof, Dept. of CSE, SBM College of Engineering & Technology, Dindigul

Introduction

The goal of this research is to investigate the use of gossip protocols for decentralized real-time monitoring .Recent research in gossip protocols suggests that these types of protocols may help engineering a new generation of monitoringsystems that are highly scalable and fault tolerant. To date, however, no gossip-based monitoring systems have been built, nor has gossip-based monitoring protocols been evaluated against alternatives, such as tree-based monitoring protocols. Gossip protocols, also known as epidemic protocols, are round-based distributed algorithms. During a round, each node selects a subset of other nodes to interact, whereby the selection function is often probabilistic. A push-pull paradigm, whereby two nodes exchange state information, process this information and update their local states frequently.Using time slot negotiation, We provide lot of offers to the users. We use merging cloud for dynamic resource allocation. There are qualitative and quantitative differences between tree-based and gossip-based aggregation. First, gossip-based aggregation protocols tend to be simpler in the sense that they do not maintain a distributed tree in the management overlay. Second, in tree-based aggregation, the result of an aggregation operation is available on the root node of the tree, while in gossip-based aggregation the result is available on all nodes. Third, failure handling is very different for both types of protocols. If a node fails, tree based aggregation protocols are generally able to reconstruct the aggregation tree and re-compute the aggregate. In gossip protocols, the computation of the aggregate is unstructured, and a node failure generally cause sir recoverable information loss–a phenomenon we refer to as “mass loss” in this paper. This problem of mass loss has not been sufficiently studied to date and thus needs to be addressed first when one wishes to perform a comparative assessment of tree-based and gossip based aggregation.The difficulty is to engineer such message rate adaptation schemes while obtaining good performance, i.e. short detection times, and low probabilities for false positives and false negatives. In this paper we examine the design space for gossip-based protocols augmented with a rate adaptation scheme using push-synopses, a gossip protocol for computing aggregates introduced by Kempe.As the baseline.This thesis investigates designs that enable the individual components of a distributed system to work together and coordinate their actions towards a common goal. While the basic motivation is to contribute towards engineering principles for large-scale autonomous systems, this research addresses the problem in the context of resource management in clusters that provide web services. Such clusters overwebapplications that must scale to a large number of clients and generate customized dynamic content that matches the clients' preferences.

EXISTING FEATURES & RELATED WORKS:

Cloud tenant running a collection of virtual appliances that are hosted on the cloud infrastructure, with services provided to end users through the public Internet. The disadvantage of this existing feature is Performance Computational is very less and cannot solve the dynamic resource allocation problem.A gossip protocol is a style of computer-to-computer communication protocol used for easily access the data in network. Modern distributed systems often use gossip protocols to solve problems because it allocate the resource in network for user access data to avoid delay.The traditional approach to the detection of threshold crossings of network-wide aggregates is using an aggregation Protocol to continuously compute the aggregate on a node and to evaluate on that node the threshold conditions every time theaggregate is updated. Several results, both centralized,and decentralized, that improve on this approach have been published recently. The common goal is achieving efficiency by reducing protocol overhead, compared to the traditional approach, when the aggregate is far from the threshold. In general, local thresholds define conditions under which nodes refrain from sending updates of their local states, thereby reducing the protocol overhead. In a recent work , we proposed the protocol TCA-GAP which augments a spanning tree-based aggregation protocol with a rate adaptation scheme for TCA generation. The key idea is to recursively assign local thresholds to subtrees, and to introduce a scheme for reassignment of these local thresholds once the threshold conditions are violated. In this paper, we use gossip protocols for detecting threshold crossings of network-wide aggregates. Gossip protocols, also known as epidemic protocols, are protocols that generally execute in periodic rounds. They can be characterized by asynchronous and often randomized communication among nodes in a networkThe main design criteria are:

Efficiency:

The communication and processing overhead of the protocol should be small, specifically during periods where the aggregate is far (above or below) the threshold.

Quality of detection:

The protocol should achieve short delays for detecting threshold crossings, and false positives and false negatives should be rare.

Scalability:

The protocol should allow for efficient operation with high quality of detection in large networks with at least thousands of nodes.

Controllability:

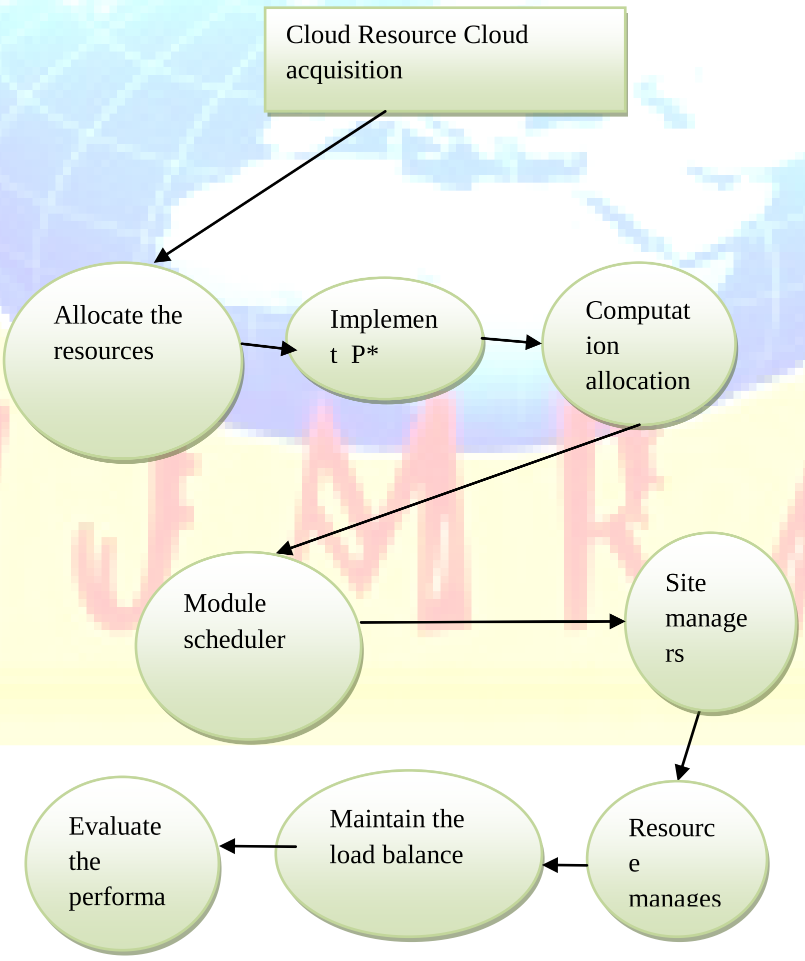
The protocol should allow for controlling the tradeoff between quality of detection and protocol overhead through management parameters that can be adjusted at runtime.The core contribution of this thesis is the introduction of self-management capabilities into the design of cluster-based large-scale services. To achieve this, we have developed, implemented and evaluated a decentralized design for resource management in these systems. The design makes service platforms dynamically adapt to the needs of customers and to environmental changes, while giving service providers the capability to adjust operational policies at runtime. The design provides the same functionality as the advanced service architectures mentioned above (i.e., service diterentiation, efficient operation and controllability), while, at the same time, assuming key properties of peer-to-peer systems (i.e., scalability and adaptability).In order to achieve the design goals listed above, our design of the middleware for resource management follows four principles. The _rst three of them are characteristic to many peer-to-peer systems. First, in order to facilitate scalability, each node has only partial knowledge of the system, and it receives and processes state information from a subset of peers, the size of which is independent of the system size. Second, each node can adapt and change its role at runtime. Initially, all nodes are functionally identical in the sense that they have the capability to manage their resources locally and to determine which applications they over. Third, each node runs a number of local control mechanisms independently and asynchronously from its peers.Mehyar present in a gossip protocol for averaging local values on a dynamically changing network graph. Convergence of the protocol is proved using the asynchronous framework of Bertsekas and Tsitsiklis presented in. No analytical upper bound on convergence time is reported inthe paper. The protocol in is robust to node failures, and there are similarities to the protocol presented in this paper (i.e., G-GAP), particularly regarding the use of recovery information. Our protocol G-GAP differs from the protocol in that the recovery mechanism can be easily separated from an underlying, non-robust protocol (i.e., push-synopses). This allows us to establish convergence bounds, by showing that there is a limit to the number of rounds needed to process recovery information, before the protocol reverts to the behavior of the underlying protocol. Most works on gossip-based aggregation protocols consider the aggregation functions AVERAGE or SUM for protocol description and evaluation.

PREPOSED SYSTEM

Push-pull paradigms, whereby two nodes exchange state information, process this information and update their local states frequently.Using time slot negotiation, We provide lot of offers to the users. We use merging cloud for dynamic resource allocation. A new concept "merge clouding" has to introduce in order to overcome resource allocation. The resource allocation will be done by combining more than one cloud server when user demand is larger. Therefore, the load balancing is satisfied.A new concept "merge clouding" has to introduce in order to overcome resource allocation. The resource allocation will be done by combining more than one cloud server when user demand is larger. Therefore, the load balancing is satisfied.Evaluation metrics: We evaluate the protocols using the following metrics. First, we measure the protocol overhead as the average number of messages processed/sec/node (Note that this value is fixed for baseline NNx protocols and depends only
on the overlay graph and the protocol round rate). Second,we evaluate the quality of threshold crossing detection by measuring the detection delay and accuracy of detection. The detection delay is measured as the difference between the time the protocol reports a crossing and the time the actual crossing occurs, as explained below. The accuracy of detection is measured by computing the fractions of false positives (alerts raised by the protocol without no corresponding actual alerts occurring) and false negatives (cases where the protocols fail to raise alerts when actual crossings have occurred). In graphs illustrating the measurement results, 95% confidence

intervals are given wherever appropriate.In all scenarios, a local variable represents the number of HTTP flows that enter the network at a specific router, and the aggregate of those variables represents the average number of such flows in the network. We simulate the behavior of the local variables using packet traces captured at the University of Twenty. The first trace, which we call the UT trace, has been created as follows. We sampled every second the number of HTTP flows from those original traces, which produced traces that give the evolution of the number of HTTP flows over time. Then, we divided the new traces into segments of 150sec each.

SYSTEM ARCHITECTURE



Cloud Consumer

This module relies on a global directory for routing requests from users on the Internet to access points to particular sites inside the cloud. Users access sites hosted by the cloud environment through the public Internet.

Cloud server

The cloud service provider owns and administers the physical infrastructure, on which cloud services are provided. It offers hosting services to site owners through a middleware that executes on its infrastructure.

Middleware access

The components of the middleware layer run on all machines. The resources of the cloud are primarily consumed by module instances whereby the functionality of a site is made up of one or more modules. In the middleware, a module either contains part of the service logic of a site or a site manager.

Resource allocation policy

The resource allocation process must be scalable both in the number of machines in the cloud and the number of sites that the cloud hosts. This means that the resources consumed (by the process) per machine in order to achieve a given performance objective must increase sub-linearly with both the number of machines and the number of sites.

Merging Cloud

In some situation, the customer has to wait in order to gain the services from the cloud server during the demand generated by the user.

In this case, other cloud servers will be merged with the current cloud servers.Therefore, the user demands are transformed to this merged cloud server.

CONCLUSION:

We have explored the use of gossip-type aggregation protocols for distributed detection of threshold crossings of aggregates.

Gossip protocols iteratively refine a local estimate of a global aggregate by nearest neighbor interactions. The key idea in our protocols is to let nodes dynamically adjust the protocol rate according to how far their local estimate of the aggregate is from the threshold. We have identified a family of protocols, organized according to the rate adjustment mechanism, the mechanism for triggering threshold crossing alerts, and whether or not the protocol exploits the symmetry in TCA detection by implementing a hysteresis-like functionality (dual modes). Key points in the design space have been evaluated, by simulation, for efficiency, quality of detection, scalability, and controllability.

REFERENCE:

[1] A. G. Prieto and R. Stadler, "A-GAP: An adaptive protocol for continuous network monitoring with accuracy objectives," Network and Service Management, IEEE Transactions on, vol. 4, pp. 2–12, June 2007.

[2] M. Dam and R. Stadler, "A generic protocol for network state aggregation," inProc. RadiovetenskapochKommunikation (RVK), 2005.

[3] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in FOCS '03: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science, p. 482, IEEE Computer Society, 2003.

[4] M. Jelasity, A. Montresor, and O. Babaoglu, "Gossip-based aggregation in large dynamic networks," ACM Trans. Comput. Syst., vol. 23, no. 3, pp. 219–252, 2005.

[5] F. Wuhib, M. Dam, R. Stadler, and A. Clemm, "Robust monitoring of network-wide aggregates through gossiping," in IM '07: 10th IFIP/IEEE International Symposium on Integrated Network Management, pp. 226– 235, May 2007.

[6] D. Breitgand, D. Dolev, and D. Raz, "Accounting mechanism for membership size-dependent pricing of multicast traffic.," in NGC '03: Networked Group Communication, pp. 276–286, 2003.

[7] F. Wuhib, M. Dam, and R. Stadler, "Decentralized detection of global threshold crossings using aggregation trees," Computer Networks, vol. 52, no. 9, pp. 1745–1761, 2008.

[8] M. Dilman and D. Raz, "Efficient reactive monitoring," IEEE Journal on Selected Areas in Communications (JSAC), vol. 20, no. 4, 2002.

[9] R. Keralapura, G. Cormode, and J. Ramamirtham, "Communicationefficient distributed monitoring of thresholded counts," in SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on

Management of data, pp. 289–300, ACM Press, 2006.

[10] L. Huang, M. Garofalakis, J. Hellerstein, A. Joseph, and N. Taft, “Toward sophisticated detection with distributed triggers,” in MineNet ’06: Proceedings of the 2006 SIGCOMM workshop on Mining network data, pp. 311–316, ACM Press, 2006.