Bundling Communication Messages in Large Scale Cloud Environments

Ali Sianati, Azzedine Boukerche, Robson De Grande
Paradise Research Laboratory–Electrical and Computer Engineering
University of Ottawa
Ottawa, Canada
asianati@uottawa.ca, bouckerch@site.uottawa.ca, rdgrande@site.uottawa.ca

***Abstract*— Cloud computing has been receiving a growing attention due to its features on the provisioning of computing resources for distributed processing. A lot of interest lays on the enabled benefits of flexible and elastic management of cheap and reliable computing sources though virtualization. The cheap characteristic brought from virtualization helps data centers to host more than one operating system on any machine. However, the time-sharing nature of virtualization and the existence of more software layers lead to larger delays in execution and network communications. The bundling of network messages directed to the same destination can be used as means to reduce the number of network transfers. In this paper, the effect of different message bundling aspects closely related to cloud applications is investigated. Analysis shows that message bundling effectively enhances communication efficiency; however, it is directly influenced by parameters such as number of messages, length of the bundling cycle, and number processing units. Thus, the highest performance gain is achieved by flexibly adjusting according bundling parameters towards data communication in large-scale cloud environments.**

Keywords—Cloud; Network; Distributed; Bundling;

I. INTRODUCTION

As an evolution of grid computing, Cloud computing introduces the provision of computational resources under the concept that they are shared among different users through the Internet [1][2][3]. Cloud computing can be observed as a successful attempt to replace small and midsize data centers while not being super-efficient keeps it from hitting scientific data centers where high performance computing grids are needed [5]. Despite grid computing, cloud provisions resources based on users' demands and flexibly re-schedules under-utilized resources when they are idle [4]. This tactic helps cloud vendors to lease their resources when no one is using them. However, this kind of resource provisioning exposes delays in overall system performance because it is possible that a request to a resource gets rejected due to the lack of resources or being used by several users at the same time [2][6][7]. Cloud vendors seek to ensure users that they receive 99.95% quality of service and in the case of 0.05% lack of resources, users will not get charged [2][17].

Although cloud vendors are using the most powerful servers and devices in the market, they cannot cope with the scientific data centers. Heavily used virtualization puts a lot of delay on the execution time of tasks, and overlay network and time-sharing nature of the cloud computing software puts extra delays on end to end delivery time. Since, cloud is based on a software over software paradigm, it worth to change the underlying software of cloud computing and improve its performance. However, this is not the case with the public clouds where user cannot access the underlying cloud software. Hence, application layer solutions are highly demanded for solving the issues with the cloud's long delays.

Message bundling is one the most suited application layer solutions to any environment with high network load. Being an application layer solution makes message bundling applicable to any cloud environment and not only the private clouds. It makes use of high bandwidth network of clouds and tries to regulate the network traffic between two nodes by reducing the total messages sent.

In this paper, new and extensive analyses are performed to evaluate message bundling under different circumstances defined according to predefined parameters that are expected to affect bundling behavior and system performance. Processing Unit Count, Inter-communication messages count, and Bundling Cycle are taken in to account as the main parameters that essentially configure a Cloud environment and can determine its performance. Message bundling can be tuned based on the system and network conditions or can be configured based on the desired communication throughput. Our results show that with proper setup, bundling can save up to 35% of system performance.

The rest of this paper is organized as follows. In Section II, a brief description of related work is presented. In Section III, the motivation for this study about message bundling in Clouds is delineated. In Section IV, Message bundling is described. In Section V, the parameters that influence the performance of bundling are introduced. We define our experimental test bed through simulations and the results obtained; we also propose a general equation that consider and summarize all conclusions from the results. Finally, Section VI draws the final conclusions and presents the future work.

II. RELATED WORK

Cloud computing extensively employs the virtualization paradigm in which multiple operating system instances can be run on one hardware instance simultaneously [8]. In the recent years, hardware virtualization and higher memory capacities helped vendors to increase overall quality of service while keeping their costs. This represented a

considerable step forward for the expansion of cloud computing. However, hardware virtualization has not yet hit networking components since the management of networking is not as straightforward as the management of processing.

As with the advancement of virtualization, virtual instances (VI) on clouds can present their own network setup. This feature indicates that VIs in the different physical networks can be connected through the same virtual network to communicate with each other. Hence, this kind of virtual networking in clouds demands the management through a set of layers to create an overlay network on top of a physical network [9], causing processing delays and likely generating overhead for applications and systems with overly-intensive communication. In the cloud architecture, the Network Device Driver(NDD) consists in one of the major software layers in cloud software stack [10]. NDD is responsible for routing the packets in virtual networks through physical networks; it works as an overlay added on top of the physical network mediating and masking the requests coming from the operating system. This layer not only provides the routing but also employs some mechanisms to perform security checks against malwares and viruses. Besides processing and queuing delays generated in the Network Driver layer, the nature of time-sharing in clouds also constitutes another issue for the communication performance. Whenever a host OS claims the CPU from the NDD, all packets inside the NDD need to wait until the next CPU round. Only after the CPU is granted for processing, the NDD can perform the routing and security checks. This issue introduces extra delays on top of cloud networking and consequently on the overall communication performance. Thus, this overhead makes cloud networks suffer from a high delay in packet delivery [11]. Experiments in [23] show that a guest domain running Linux on Xen can only achieve 2.9 Gb/s of throughput, whereas native Linux can achieve above 9.3 Gb/s on the same machine.

As a common, simple, and costly measure adopted by several cloud vendors, extremely high bandwidth packet switch networks are used in their systems in order to provide the best quality of service. Even though such networks introduce high cost to the providers, they help delivering a large amount of data messages in a fraction of a second [10][11].

Network performance needs to be increased for cloud applications, and this gain can be achieved with costly methods, such as overprovisioning of resources, through the use of different techniques applied in different layers of the software stack to minimize the delays generated by the virtualization layers. Several attempts were made to improve the network performance in cloud computing.

OpenFlow was used in a method presented in [19] to create a new cloud network structure not only to improve the network performance but also to enhance the network security. In this method, the proposed structure introduces a total change of the entire cloud data center architecture, similar to one solution presented in [12]. Consequently, this is not applicable or feasible for public clouds where accessing the cloud software stack itself is impractical or impossible.

Multi queue NIC can help different guest domain to interact with the network device directly and bypassing the driver domain [25]. However, in clouds where guest domains can have their own overlay network, packets must again be copied to the driver domain in order to determine the physical address of the packet. A packet aggregation technique is introduced into the network layer of the cloud software in [12][13]. This technique is completely against the fragmentation technique. In this method, smaller packets are clustered into larger packets and then sent over the network. As a drawback, the aggregation approach needs to have access to the underlying cloud software where this is impossible in public clouds. This approach also needs to know the Maximum Transfer Unit (MTU) of the underlying network. In networks with large MTU, this approach can save a lot of bandwidth and end-to-end communications transfer time.

Another packet aggregation in the TCP layer is introduced along with TCP acknowledgment offload in [21]. This technique is very similar to the one described in [12][13]. The authors performed some modifications on the TCP layer to implement their solution and accommodate with the new approach. They put small packets for one TCP connection into larger packets on the sender side and send one acknowledgment for one large packet from the receiver side. While their method has some limitation on the type of the packets, like other network layer methods, it cannot be applied to public clouds.

A new XenSocket [22] is designed and implemented which can reduce the overhead of the original XEN intra-communication system. This new socket can be used between two VMs on the same machine. However, it is not a general solution for all communications in the cloud. The work described in [24] presents a packet aggregation technique in order to reduce total memory access for copying packets from quest domain into the network drivers shared memory. The technique presented in [23] changes the guest domain in a virtualized environment to eliminate the need to copy packets from guest domain to the network driver domain. Their approach uses direct memory access from host domain to the guest domain. These approaches are applicable in the private clouds where user can change the guest domain.

A different methodology is presented in [18]. This work proposes a new pricing scheme for which users with higher network demand are required to pay more. This scheme does not require changing any software stack in the cloud and only needs to be added to the monitoring software of the cloud that vendors use. By using this policy, cloud vendors can force cloud tenants to reduce their network usage by changing their applications behavior or compress their data. However, this cannot guarantee the impact of the high demands from Internet users. A service provider cannot force users to consume less resources or access less services, and by forcing this policy, a provider may also lose users. A work described in [20] attempts to solve the unpredictable behavior of Internet user by outlining a weighted linear combination of past observations. By predicting the user

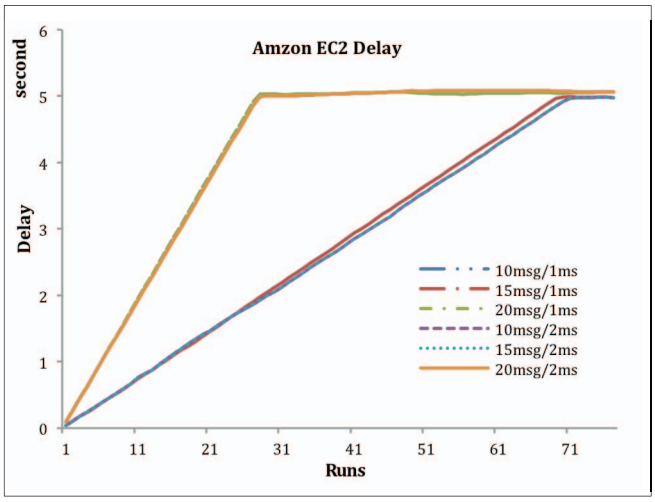

Figure 1. Amazon EC3 network delay under high network load

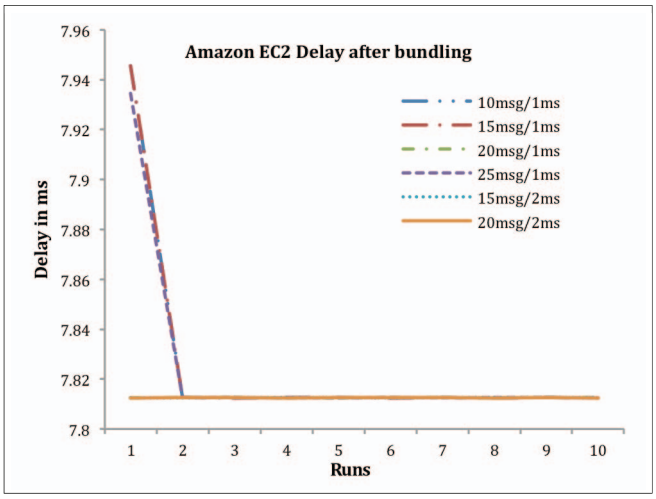

Figure 2. Bundling effect on Amazon EC2 servers

behavior, a tenant may not pay more for the agreed services, but the cloud vendor still suffers from the high load on its network [25].

As stated above, most of the works attempt to improve the protocols and algorithms in the lower layers of the network stack, but in some cases it is possible to reduce the total network delay in the application layer instead of lower layers. This is due to the fact that applications can change their behavior to fit into cloud constraints where changing the cloud infrastructure is unfeasible. Message Bundling is one of the techniques used for reducing network delays and basically acts like packet aggregation. As a benefit for the existing models and implementations, this technique is an application layer solution for systems with high network load. . Consequently, any system can employ it with no need to change infrastructure of the underlying network. This unique characteristic of the Message bundling makes it the perfect candidate for regulating network traffic in the cloud.

III. MOTIVATION

A research on public clouds in 2008 revealed that cloud network could handle large and small size messages in the same way with high delivery delay [11]. Nevertheless, there is no document reporting that clouds still suffer from this issue, or the Cloud software has improved to enable better performance on communication. As a result, a study observing the Cloud behavior in the case of high network load, in terms of message count, been conducted for determining the aspects that might influence performance.

A. Testing Environment

The experimental analysis consists in a real scenario of a public Cloud, so we conducted the experiments on Amazon EC2 servers. Since communication delay was the focus of the analysis, a client/server setup was designed and implemented. We created one t2.medium instance with windows server 2012 installed and another t2.micro instance with windows server 2012 installed. The former instance was used as the sender and the latter was used as a receiver. We implemented two simple applications. One of these applications, named sender, was responsible for generating messages in a controlled fashion, and the other process, named receiver, was responsible for accepting the messages and forwarding them back to the sender. The sender puts its current time stamp in each message and calculates the round trip time upon receiving the message forwarded back from the receiver.

B. Computation overhead

In order to make sure that processes do not add any computational delay on top of network delay, an optimization on the implementation of them forced the use of almost atomic actions. The delay which is introduces by running atomic operations are so small comparing to that of network delays. As a result, we could ignore these delays.

C. Experimental results

Some conclusion have drawn from our experiments on the Amazon EC2 network. The results indicated some of the sources that cause the communication delays. As shown in Figure 1, in the case in which a large number of messages are sent to a particular server destination, the network delay increases in a linear fashion. These results are similar to those presented in [11] in 2008, which describes despite the improvements in the past years, Clouds still cannot tolerate an extremely high number of messages at the same time. Consequently, this constraint on the communication in the Cloud led us to explore the advantages of message bundling as a form to reduce the quantity of messages sent in contrast to the increase of message size.

As described in its definition, message bundling can tackle the network delay in clouds by aggregating several messages into one message destined to one receiver [14]. This method performs the opposite of message fragmentation where messages are split into smaller chunks. In order to evaluate message bundling in real environments, we conducted a second experiment on an Amazon EC2 server. This experiment was totally driven to evaluate the effect or benefit of bundling in networking. Therefore, we modified

our test application so that it bundles messages and sends one message at each bundling cycle. Figure 2 shows that despite the large bundling cycle used in the experiments, the network delay decreased to a constant value.

To the best of our knowledge, no one has ever investigated the effect of parameters similar to Processing Unit (PU) count, node count, and bundling cycle on Message Bundling. Thus, we evaluated these parameters in a wide range of values; the results of such analyses show the effect they produce on the performance of message bundling in cloud computing.

IV. Message Bundling

Message bundling can be applied on systems in different scopes, such as extending network protocols or network applications. In network protocol scope, if the packet length is smaller than the network maximum transfer unit (MTU), several packets are formed into one packet and sent [13]. In this way, the protocol benefits the most from the available network bandwidth. At the application level, this approach is not widely used for several reasons: (i) most applications are not time sensitive; and (ii) network bandwidth is a barrier that prevents applications from sending large messages. However, with the shift towards having distributed systems on cloud computing and having high bandwidth networks in clouds, we can start designing solutions using bundling on the application layer.

In order for a message bundler to be useful, it needs to collect from several application sources, similarly to distributed systems; it creates one message from them and sends it to the destination. Working as a networking middleware, instead of submitting messages directly to the underlying network through the network drivers and libraries, applications submit their messages to the bundler. The bundler sends the collected messages to a bundler at each respective destination, which is in charge of delivering the messages to destination application. In order to organize the transmission of messages, each bundler contains multiple queues where each of them is assigned to one designated destination.

A bundler receives messages from several sources and holds only one queue for each destination. In practical means, destinations are organized based on their IP address. The debundler component in each destination is responsible for splitting the received message into original messages.

A bundler also presents a crucial property for its functioning and for avoiding performance loss out of queuing delays, a timer. The timer defines the frequency in which the bundler should send the bundled messages to their destination. The best-fit setup of this timer presents a trade-off on the amount of bundled messages and end-to-end delivery delay. If the timer interval, bundling cycle, is too small, the bundling efficiency would be similar to that of sending messages without bundling; there is not enough time for the bundler to collect a reasonable number of messages. If the bundling cycle is too large, the bundler gathers a large number of messages and reduces the network load, but it introduces high delivery delay for messages that arrived in the beginning of the bundling cycle.

The principal functioning of bundling is totally driven by the incoming messages. When a message is received at a bundler, it checks its queues. If the queue for the message destination already exists, it inserts the message in the queue. Otherwise, it will create a new queue based on the destination IP address and put the message in the queue. In order for a debundler to be able to split messages, the bundler puts small offset tags in the bundled message string. Each offset determines the next message index in the bundled message. In the next section of this paper, we investigate the parameters affecting bundler performance and efficiency when applied in the context of a Cloud environment for the improvement of communication performance.

V. Performance Analysis

Even though it is clear that bundling messages reduces delays by decreasing the number of transmitted messages, bundling performance may not generate any gain if not configured according to the characteristics of the environment and application scenario. A set of major parameters is considered in this performance analysis to determine the correct combination of factors that might adjust the bundling and direct its efficiency. Such parameters are Processing Unit count, Inter-communication Message count, and Bundling cycle. A higher number of processing units leads to larger fragmentation of bundling sets in multiple destinations; a greater inter-communication message count enables more bundling to take place; and a longer bundling cycle promotes larger clustering of messages but increases the chances of bundling delays.

The optimal result in bundling is achieved only through an approach that truly combines and represents the conditions an environment presents. A thorough analysis of such conditions is conducted empirically. Hence, in the rest of this section we will further explain the three major introduced parameters and their respective effect on message bundling by doing several analyses on simulated cloud environment.

A. Simulation Environment

In order to evaluate the performance of bundling on a large-scale distributed environment, we simulated the behavior of a distributed system deployed on a cloud environment. For this analysis, we implemented a simulation in Java that mimics the communication and aggregated factors that might influence data transfer among parts. The simulator has been designed to examine the Cloud behavior in large-scale environments, in which hundreds of PUs are allocated for execution in public or private Cloud

In our scenarios, nodes are deployed randomly on PUs. After the deployment is concluded, each node randomly picks a list of nodes to communicate with. Each PU has a bundler with fixed bundling cycle. During the execution, nodes submit their messages to the bundler instead of the underlying network. A bundler sends local messages to their destinations right away and puts all other inter-communication messages in its queues. At the end of each

TABLE I. SIMULATION SETUP PARAMETERS

Name	Range
Nodes	5000
PUs	250,500,1000
Message creation	300-6000
Intercommunication message ratio to total messages	90%
Network Delay	200 µSec
Bundling Cycle	1/10 of network delay

cycle, the bundler creates different bundled messages for different destination PUs. The simulated network considers delays and bandwidth, which will significantly influence the data transfer. Results are collected based on the transmission delays, which comprehends the moment from which messages are dispatched in a node and received at the respective destination.

B. *Processing Unit and Processes*

The availability of resources totally drives the performance and execution of any running application. These resources include processing unit, memory, and network. PUs play an important role in computing intensive applications. Even though memory is essential for the execution of processes, we assume in this work that our distributed system is not memory intensive and PU intensive, yet it receives enough PU and memory to run. Nodes of the distributed system are deployed on PUs considering the overall PU load. The deployment of nodes can be done using random or static partitioning where in Random partitioning there is no intercommunication optimization while in Static partitioning a small amount of optimization is done [15][16]. In case of a good partitioning is established, the nodes that communicate with others the most are kept close together in terms of network distance, and in most cases they are kept in the same PU. Being close to each other reduces the total intercommunication overhead in the system.

Based on the total available resources, the performance of a distributed system may vary. In cloud environments, total execution cost is proportional to total PUs. However, if we reduce the PUs count, we lose more parallelism and add up more time cost to the system. On the other hand, increasing the number of PUs in distributed systems has a drawback and it is the inter-communication messages needed to connect two remote PUs. Another side effect of changing the number of PUs is on the bundler. As Figure 3 depicts, PU count presents a direct relation to the bundling performance. This is due to the fact that as the total PUs in the system increases, the chance of having more destinations per bundler increases as well. More possible destinations per bundler mean more queues in each bundler. Hence, each queue in the bundler has less chance to collect more messages and aggregate them in to a large one.

In this set of simulation we used the values in TABLE I. As we increase the PU count in the system, while keeping the message creation at 3000, in fact, we are creating more destinations per PU and more bundlers in the system. This presents two negative impacts on the system. First, increasing the queues increases the total number of bundled messages sent out, considering that each one adds a delay in the system. Second, with more bundlers in each PU, fewer messages are collected and bundled with each one. As a result, in a bundled system, the total PUs used for simulation must be kept as low as possible. In this way, we can make sure that the bundlers do not generating extra messages per destination.

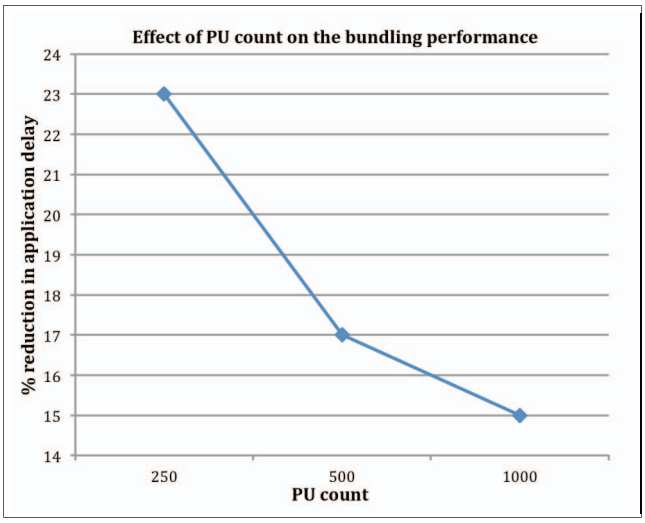

Figure 3. Effect of PU count on bundling performance

C. *Inter-Communication Messages generated*

The basic nature of communication in distributed systems is message passing. From this standpoint, waiting times between communicating parts is constantly present in a distributed system or application. Due to the dependencies among distributed elements, inter-communication messages implicate on larger cumulative delays.

As stated before, networks in clouds suffer from high delays. Hence, in a system with a large number of processing nodes or nodes which generate a massive amount of inter-communication messages, the system wastes a significant amount of time waiting for messages in the network to arrive. For this reason, we have to consider the quantity of messages that are generated in the system and which fraction of them is destined to go through inter-communications between PUs. The main task of a message bundler is to deal with the inter-communication messages and put each one in its queue. More inter-communication messages generated in the system enables the bundling process to collect more messages in each queue of the bundler. Consequently, a higher bundling efficiency is achieved.

Figure 4 and Figure 5 demonstrate the effect of inter-communication messages and messages submitted to the bundler on system delay respectively. As depicted in the graphs, more inter-communication messages are created, more reduction is observed in the system delay using bundler. In a system without a bundler, each message adds its own delay to the overall system delay. Hence, growing the message quantity, increases the overall delay. In a system with bundling feature enabled, the total delay is proportional to the total number of bundled messages, which itself is proportional to PUs. Since PU count is unchanged during the

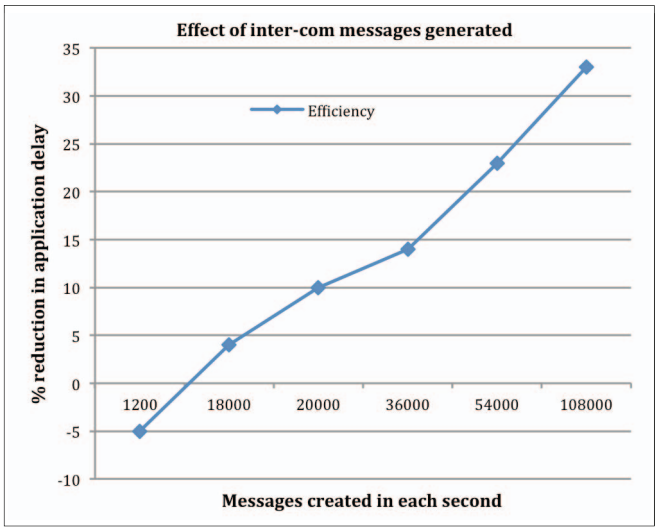

Figure 4. Message creation effect on bundling

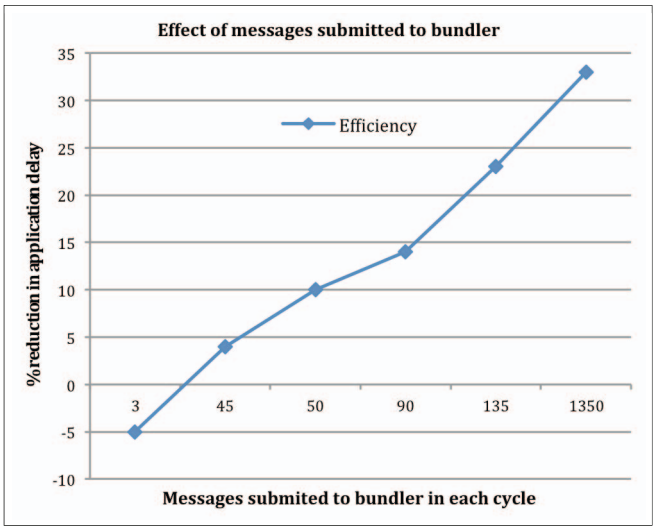

Figure 5. Message submission to bundler effect on bundling

runtime, as we increase the total messages created in the system, the total bundled messages will not increase with the same rate as messages are created. Thus, system delay becomes proportional to the frequency in which PUs communicate with each other.

D. *Bundling Cycle*

In a timer-based bundler, the cycle is determinant in the functioning of the bundling of messages due to its direct influence on the amount of bundled messages and communication delay. The timeout defining the end of each cycle restricts the amount of collected messages in each queue to be sent to the destination. Bundling cycle can be tuned based on the network or application conditions during the runtime or may be fixed at the system startup. Flexibility of the bundling cycle is useful in very dynamic environments. In environments with small amount of inter-communication messages, choosing smaller bundling cycle is preferred since a bundler can submit the queued message as soon as possible before waiting for more messages to arrive. Moreover, the existence of time sensitive messages, such as control messages, does not allow bundler to wait for a long time to build up the bundling packet before it is sent or for the timeout of the cycle to expire. Hence, a small bundling cycle is suitable in this scenario. In other scenarios where lots of inter-communication messages are generated, larger bundling cycle can gain performance by creating larger bundled messages and consequently reducing the total time spent in the network.

Figure 6 depicts three direct results in accordance with network delay: (i) bundling cycle is less than the half of the network delay: bundler does not have enough time to aggregate messages and sends more small messages which adds to the total delay, (ii) bundling cycle is larger than half of the network delay: bundler has a lot of time to gather messages, while messages arrived in the beginning of the cycle have to wait a lot which adds to the total system delay, (iii) bundling cycle is set to the half of the network delay: this value for the bundling cycle is like a tradeoff between gathering less or more messages and results in the best performance gain in the system.

As Figure 6 shows, the increase of the bundling cycle to the half of the network delay directly improves the total efficiency. However, if the bundling cycle becomes larger than the half of the network delay, the overall efficiency of the system decreases. Although larger bundling cycles lead to more messages collected and aggregated, they generate more queuing delays. Thus, messages arriving at the bundler need to wait until bundler times out and sends them. Hence, messages that have arrived at the beginning of the bundling cycle need to wait more, and in case of increasing the bundling cycle, they face even more delays.

E. *Network Congestion*

Another result from using bundling is shown in Figure 7 where reductions in the network congestion are plotted. In case of using message bundling, the network can benefit more from its MTU and generate less packets per message. This helps router buffers to accumulate fewer packets. As a result, congestion is less possible in networks where the message bundling is used. As a matter of fact, reducing network congestion leads to better network efficiency in cloud computing. An important point is as we increased the total messages generated in the network, network efficiency increased with a high rate. However, as the aggregated message size gets greater than the MTU size, the efficiency increasing rate drops due to fragmentation in the lower layers of the network stack. In this set of simulations, we set MTU five times greater than the message size and used a very naïve and simple buffer management in routers.

F. *Performance gain Function*

Based on the results, a general function which defines the system efficiency in terms of network delay is derived:

$$E(P,I,B,D) = \frac{I^{\alpha}.B^{\beta}.(B-D)}{P^{\gamma}} \quad (1)$$

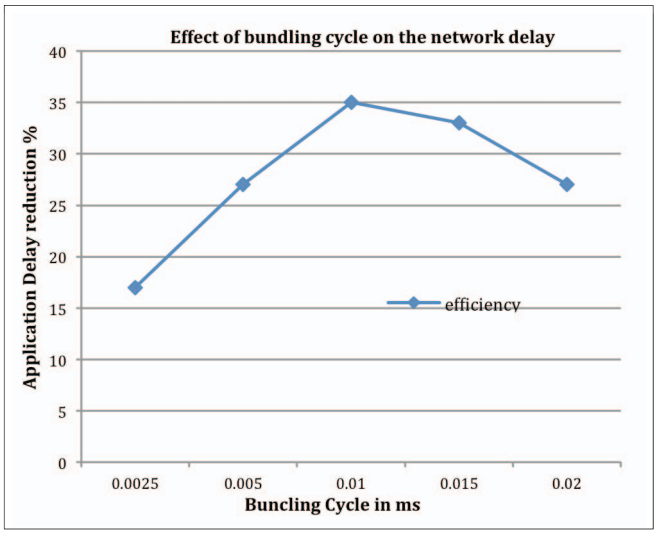

Figure 6. Bundling cycle effect on bundler performance

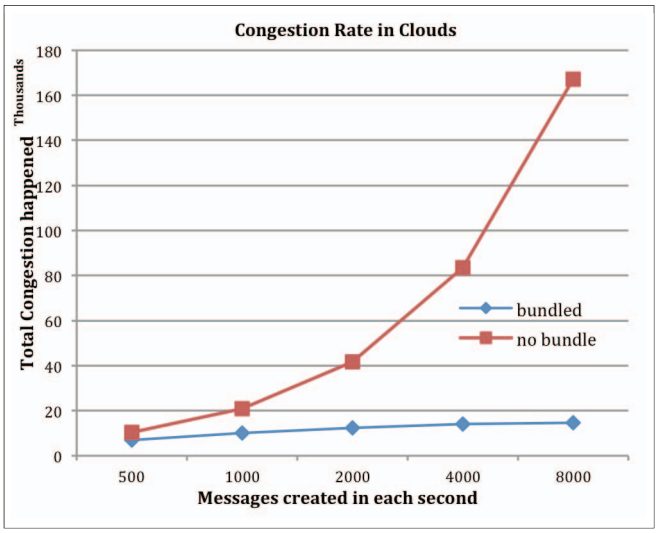

Figure 7. Message creation effect on congestion

Where E represents the bundling efficiency and P, I, B and D represent PU count, Inter-communication Message count, Bundling cycle, and network delay respectively. In this function, inter-communication count and bundling cycle shows a positive impact on the system efficiency while PU count presents a negative impact. However, if the bundling cycle is greater than the network delay, it will have a negative impact on the overall communication performance. In this formula α, β and γ are used as weights to adjust the formula. The term α reflects the networking conditions: delay, number of messages, and load. The term β is related to the bundling and closely dependent on α, bringing up a correction on the length of the bundling cycle for a given network delay. The term γ reflects on the fragmentation of communication channels and saturation of network resources. For instance, in a private cloud where network is not congested, inter-communication messages exercise a lower influence on bundling, so α can be assigned a smaller related value. In another hand, in scenarios where queuing delay is important, we can increase β to better represent the effect of the bundling cycle. Furthermore, in public clouds where more PUs costs more, larger γ is desired to control the total network efficiency in contrast to cost. Since the characteristics drastically changes according to the environment, resources, and applications, a dynamic adjustment of these values is needed for deploying systems with less concerns on delays and communication efficiency.

VI. CONCLUSION

In the recent years, cloud has paved its way to the information technology world. However, time-sharing nature of cloud computing and high end to end delivery delays in Cloud networks prevents it from competing with scientific data centers. In this paper, we showed how Message bundling as an application layer solution could be used in Clouds. We Studied the effect of different parameters on the cloud message bundling, similar to PU count, inter-communication messages count, and bundling cycle were investigated and showed which of them presents the negative and positive impact on the overall system performance. The results obtained in the experiments and discussed in this paper can serve as a stepping block for developers and system administrators to tune up bundling for specific application environments and save considerable time in the system execution time and communication resources.

Furthermore, the results we obtained from public Cloud servers shows that Cloud servers still suffer from high end to end delivery delays and this issue keeps an open door for future researches on cloud computing network performance.

As a future work, we will extend bundling model to that of buffer based model in which bundler timer is replaced with buffer capacity of the bundler. Moreover, we will combine both timer based and buffer techniques into hybrid model. We believe hybrid model will definitely enable cloud computing to reshape the networking landscape.

REFERENCES

[1] M. Hamdaqa and L.Tahvildari, "Cloud Computing Uncovered: A Research Landscape, " In Advances in Computers, pp. 41-85, 2012, ISBN: 978-0-12-396535-6.

[2] L.M.Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition," In ACM SIGCOMM Computer Communication Review, vol 39, issue 1, pp. 50-55, 2009.

[3] M. Hamdaqa, T. Livogiannis, and L. Tahvildari, "A Reference Model for Developing Cloud Applications, " In Proceedings of The first International Conference on Cloud Computing and Services Science, pp. 98-103, 2011.

[4] I. Foster, Y. Zhao, I. Raicu, and L. Shiyong, "Cloud Computing and Grid Computing 360-Degree Compared," In Proceeding of Grid Computing Environments Workshop, pp.1-10, 2008.

[5] K.R. Jackson, L.Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H.J. Wasserman, and N.J. Wright, "Performance Analysis of High Performance Computing Applications on the Amazon Web Services Cloud," In Proceedings of IEEE Second International Conference on Cloud Computing Technology and Science, pp.159-168, 2010.

[6] M. Mao and A Humphrey, "A Performance Study on the VM Startup Time in the Cloud, " In Proceedings of IEEE 5th International Conference on Cloud Computing, pp. 423-430, 2012.

[7] A. Iosup, S. Ostermann, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, “Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing, ” IEEE Transaction on Parallel Distributed systems, pp. 931-945, vol. 22, 2011.

[8] J. Ekanayake and G. Fox: “High Performance Parallel Computing with Clouds and Cloud Technologies, ” In Proceedings of The First International Conference on Cloud Computing, pp. 20-38, 2009.

[9] K. Barabash, R. Cohen, D. Hadas, V. Jain, R. Recio, and B. Rochwerger, “A case for overlays in DCN virtualization, ” In Proceedings of the 3rd Workshop on Data Center - Converged and Virtual Ethernet Switching, pp. 30-37, 2011.

[10] G. Wang and T.S.E. Ng, “The Impact of Virtualization on Network Performance of Amazon EC2 Data Center, ” In Proceedings of The 29th IEEE Conference on Computer Communications, pp.1-9, 2010

[11] Edward Walker, “Benchmarking Amazon EC2 for high-performance scientific computing, ” LOGIN the Usenix Magazine, Vol. 33, No. 5 , pp. 18-23, 2008.

[12] M. Bourguiba, I. El Korbi, K. Haddadou, and G. Pujolle, “Improving virtual machines networking performance for cloud computing,” IFIP/IEEE International Symposium on Integrated Network Management, pp. 513-519, 2013.

[13] M. Bourguiba, K. Haddadou, I. El Korbi, and G. Pujolle, “Improving Network I/O Virtualization for Cloud Computing, ” IEEE Transaction on Parallel and Distributed Systems, vol. 25, pp. 673-681, 2014.

[14] R.M. Fujimoto, A.W. Malik, and A.J Park, “Parallel and Distributed Simulation in the Cloud, ” SCS Modeling and Simulation Magazine, Society for Modeling and Simulation, Vol. 1, No. 3, July 2010.

[15] A. Boukerche, “An adaptive partitioning algorithm for distributed discrete event simulation systems,” Journal of Parallel and Distributed Computing Volume 62, Issue 9, pp. 1454-1475, 2002.

[16] A. Boukerche and C. Tropper, “A static partitioning and mapping algorithm for conservative parallel simulations, ” In Proceedings of the eighth workshop on Parallel and distributed simulation, pp. 164-172, 1994.

[17] S. Ostermann, R. Iosup, N. Yigitbasi, and T. Fahringer, “A Performance Analysis of EC2 Cloud Computing Services for Scientific Computing,” In Proceedings of First International Conference on Cloud Computing, pp. 115- 131, 2009.

[18] H. Ballani, K. Jang, T. Karagiannis, C. Kim, D. Gunawardena, and G. O'Shea, “Chatty tenants and the cloud network sharing problem,” In Proceedings of the 10th USENIX conference on Networked Systems Design and Implementation, pp. 171-184, 2013 .

[19] Y.H. Chu; Y.T. Chen, Y.C. Chou, and M.C. Tseng, “A simplified cloud computing network architecture using future internet technologies,” In Proceeding of IEEE 13th Network Operations and Management Symposium, pp.1-4, 2011.

[20] K. LaCurts, J.C. Mogul, H. Balakrishnan, and Y. Turner, “Cicada: Introducing Predictive Guarantees for Cloud Networks,” In Proceedings of the 6th USENIX Workshop on Hot Topics in Cloud Computing, pp.14-14, 2014.

[21] A. Menon and W. Zwaenepoel ”Optimizing TCP Receive Performance”, In Proceedings of the USENIX 2008 Annual Technical Conference, pp. 85–98, 2008.

[22] X. Zhang, S. McIntosh, P. Rohatgi, and J.L. Griffin, ”XenSocket: a high-throughput interdomain transport for virtual machines,” In Proceedings of the ACM/IFIP/USENIX 2007 International Conference on Middleware, pp. 184-203, 2007.

[23] K.K. Ram, J.R Santos, Y. Turner, A.L. Cox, and S. Rixner, ”Achieving 10 Gb/s using safe and transparent network interface virtualization, ” In Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments, pp. 61-70, 2009.

[24] M. Bourguiba, K. Haddadou, and G. Pujolle, ”Packet aggregation based network I/O virtualization for cloud computing, ” Journal of Computer Communication, vol. 35, pp. 309-319, 2012.

[25] D. Abramson, J. Jackson, S. Muthrasanallur, G. Neiger, G. Regnier, R. Sankaran, I.Schoinas, R. Uhlig, B. Vembu, J. Wiegert, ”Intel virtualization technology for directed I/O”, Intel Technology Journal, vol.10, issue 3, www.intel.com/technology/itj/2006/v10i3/, 2006.