Redesigning Transaction Load Balancing on Electronic Funds Transfer Scenarios

Rodrigo da Rosa Righi, Cristiano André da Costa, Luiz Gonzaga Jr, Kleinner Farias, Alexandre Luis Andrade, Lucas Graebin

Universidade do Vale do Rio dos Sinos - Applied Computing Graduate Program
Unisinos Av. 950, São Leopoldo, RS, Brazil

{rrrighi, cac, lgonzaga, kleinnerfarias}@unisinos.br, {alexandreluisandrade, lgraebin}@gmail.com

ABSTRACT

The use of electronic medias for payment has been increasingly adopted, instead of employing money in currency paper or check directly. Considering this electronic funds transfer (EFT) scenario, we developed a model called GetLB which comprises not only a completely new and efficient scheduler but also a cooperative communication infrastructure for handling heterogeneous and dynamic environments. The scientific contribution consists of a scheduling heuristic that combines static data from transactions and dynamic one from processing nodes to overcome the limitations of Round-Robin. Besides the GetLB's description, this article also presents a prototype evaluation by using both traces and configurations obtained with a real EFT company.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods—*Scheduling*; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Algorithms, Performance, Management, Experimentation

Keywords

Load balancing, cooperation, scheduling, heuristic, GetLB

1. INTRODUCTION

Nowadays, we can observe that the use of electronic medias for payment operations is increasingly adopted, instead of using money in currency paper and check [1]. Normally, an electronic transaction is related to either a purchase or balance requisition and runs through a round-trip path from one terminal up to a processing center [2]. POS (Point of Sale), EFT (Electronic Funds Transfer), ATM (Automatic Teller Machine) and mobile devices are examples of the most used terminals [3]. After arriving in the provider company, transactions are received by a switch that acts as a scheduler that assigns them to processing machines, or PMs.

Both resource management and scheduling are key services for getting efficiency when dealing with the observed growing of transactions per second (TPS) on processing companies [4]. The standard mechanism for mapping transactions to PMs is the so-called Round-Robin, in which employs a list of resources in a circular fashion [5, 6]. Round-Robin method represents an easy way to achieve an optimal scheduling when both sets of transactions and PMs are characterized by a homogeneous system. However, since the electronic transactions are heterogeneous (withdraw, balance, prepaid telephony, etc), Round-Robin algorithm can distribute them for processing on highly-loaded machines, leaving others with moderated load idle.

In this context, this article presents a proposal for load balancing framework called GetLB. It acts as an alternative to employing the Round-Robin method and presents the following structure: (i) cooperative communication; (ii) scheduling and; (iii) notification. The communication topic concerns the efficient and cooperative interaction between the switch and PMs for collecting scheduling data. In its turn, the scheduling is responsible for assigning transactions in accordance with their types (requirements of CPU, network, memory and disk) and dynamic data regarding PMs and network status. Notifications are useful for enlarging or dropping the number of PMs without loosing system availability. The input workload adopted for tests was based on traces obtained with a real Brazilian service provider company.

2. GETLB: LOAD BALANCING MODEL FOR EFT SYSTEMS

GetLB was structured with the following design decisions in mind: (i) the scheduling heuristic algorithm runs in the switch module and must work with up to date information regarding the PMs; (ii) the heuristic scheduling must combine relevant data in order to compose the notion of load; (iii) PMs must be capable to notify the switch; (iv) the framework must deal with heterogeneous resources at both communication and computing levels.

2.1 Cooperative and Extensible Architecture

The proposed architecture is shown in Figure 1. The first

SAC'14 March 24-28, 2014, Gwangju, South Korea.

http://dx.doi.org/10.1145/2554850.2555121

difference when comparing it to the traditional approach comprises the network. In GetLB, the disposition of the elements in the same network is not mandatory. The only prerequisite consists of the fact that each element must be accessed through an IP address. Regarding the switch perspective, the time for accessing a PM should not be the same for another one. The same can be applied for communications from PM components to internal subsystems. Although the Internet offers a network latency often 1000 times greater than local area networks, the flexibility offered by GetLB brings the following benefits: (i) given that the resources of an institution are limited, it is possible to add additional ones from business partners in order to support a specific demand; (ii) it helps the growth of the enterprise, since some countries presents strong regulations that claim EFT being processed by MPs located in national territory.

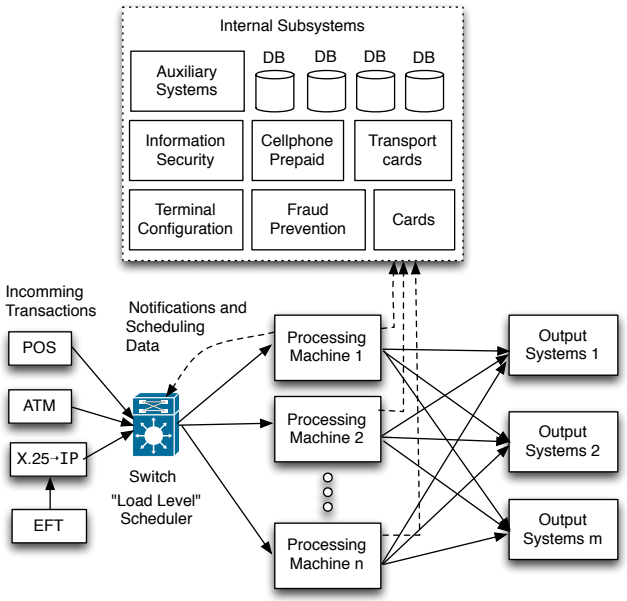

Figure 1: GetLB architecture

Other relevant aspect of GetLB architecture concerns the heterogeneity treatment. Besides exploring this feature at network level, GetLB also deals with PMs that present different features such as CPU clock, access times to the switch and subsystems, as well as different configurations of primary and secondary memories. So the switch works with a vector that contains information of all PMs. We designed a pull-based interaction between the switch and PMs, where each processing machine updates its part on the remote vector periodically. The cooperation among the architecture elements enables switch to act only with in-memory data for performing scheduling calculus. The updating period will inform how recent is data regarding CPU, memory and network from the processing machines. Due this approach, notifications allows to inform the switch about a particular event asynchronously. Notifications will not be covered in this paper in details.

2.2 Transaction Scheduling

The switch module identifies the type of a transaction, which has its own CPU and I/O requirements, and finds the most suitable target for its processing. Considering a modeling where transactions and PMs are heterogeneous and PMs can be also seen as a dynamic environment, we developed a scheduling heuristic called LL (Load Level). LL can be viewed as a decision function $LL(i,j)$ where i means a specific type of transaction while j denotes a candidate target PM for receiving transaction i. For each new transaction i, the switch will calculate n equations $LL(i,j)$, where n means the number of processing machines. In this way, the lowest result will inform the target that will receive a transaction. $LL(i,j)$ can be obtained by computing Equation 1.

$$LL(i,j) = Recv(i,j) + Proc(i,j) \quad (1)$$

$$Recv(i,j) = bytes(i)\ .\ transfer(j) \quad (2)$$

$$Proc(i,j) = transaction(i,j) + \sum_{z=0}^{m-1} transaction(z,j) \quad (3)$$

$$transaction(i,j) = \frac{instructions(i)}{clock(j)\ .\ (1 - load(j))} + \frac{RAM(i)\ .\ serviceRAM(j)}{freeRAM(j)} + \frac{HD(i)\ .\ serviceHD(j)}{freeHD(j)} + sub(i,j) \quad (4)$$

$$sub(i,j) = \sum_{y=0}^{x-1}[(2\ .\ sub_a(y,j)) + sub_c(y)]\ .\ sub_r(i,y) \quad (5)$$

$LL(i,j)$ is obtained by calculating the time required for receiving and processing a given transaction i in a target machine j. So, the term $Recv(i,j)$ considers the time required to transfer all bytes of the transaction from the switch to the PM. For that, $transfer(j)$ comprises the time to send 1 byte between both communication entities. Equation 2 can represent the most significant parameter on LL equation since this part should be the most onerous one for accessing PMs around the Internet. $Proc(i,j)$ corresponds to the processing time of all transactions mapped to machine j, including the candidate transaction i. Thus, Equation 3 can be divided in two sub-elements: (i) a prediction of computation time for transaction i on PM j; (ii) a prediction of all m transactions that have already mapped to PM j previously and remain on its input queue.

Equation 4 is used for measuring the execution time of a transaction i on PM j. It can be divided in four subparts: the first denotes CPU operations, the second and the third I/O operations and the fourth refers to subsystems. Each transaction is defined by the following parameters: (i) number of instructions; (ii) number of RAM operations; (iii) number of HD operations; (iv) which subsystems are necessary for computing a transaction and the number of interactions with each one; (v) access time to each subsystem. Considering that the use of local area network is not mandatory, the aforementioned parameter v is useful to add a delay for using subsystems distributing along different Internet domains. In Equation 4, $serviceRAM(j)$ and $serviceHD(j)$ denote the mean I/O time for performing a single instruction of write in memory and disk, respectively. The parameters $freeRAM(j)$ and $freeHD(j)$ are percentages the indicate the availability of each I/O resource in a specific moment.

The time involving the subsystems is computed by $sub(i,j)$ in accordance with Equation 5. Each type of transaction i must access x subsystems. Thus, $sub_a(y,j)$ considers the

time spent by PM j for accessing the particular subsystem y through network interaction. This time is multiplied for 2 in order to consider a round-trip evaluation. The field $sub_c(y)$ refers to the service time of the subsystem y and $sub_r(i, y)$ represent the number of times that subsystem y is called for the complete computation of i. Finally, the switch has two tables for helping the scheduling process: (i) types of transactions with requirements; (ii) subsystems with computation times. Data needed for the functions $bytes(i)$, $instructions(i)$, $RAM(i)$, $HD(i)$, $sub_c(y)$ and $sub_r(i, y)$ are fixed (taken by querying these tables).

3. EVALUATION RESULTS

We implemented a prototype in Java using RMI (Remote Method Invocation) on two types of interactions: (i) "PM→Switch" for updating PM data to the switch and for sending notifications; (ii) "Switch→PM" for transaction dispatching. Table 1 presents the results when using an homogeneous cluster. The time displayed in this table has a hh:mm:ss notation and denotes the time difference from the beginning until the last transaction processing. Clearly, the distributing transactions with RR is balanced, but this fact does not imply on better performance, regarding the processing time perspective. Although the machines are homogeneous, the set of transactions is not, and RR maps them to resources cyclically without observing their characteristics.

The results with the heterogeneous cluster are shown in Table 2 (4 PMs with 2.4 GHz and 2 PMs with 1.2 GHz). RR penalizes the final time because some transactions are sent to machines with lower capacity. Thus, besides the set of transactions being characterized as an heterogeneous system, and the resources being classified as well, it contributes for degrading the RR performance. GetLB dispatches more than 90% of the amount of transactions for the collection of four machines with greater capacity. Despite the gain of 11.51% for GetLB, the machines PM0 and PM1 are underutilized, it happens due the interval of 1 second for updating scheduling data (PM to switch).

Table 1: Results with homogeneous cluster

	GetLB			Round-Robin		
Machine	Number transac.	%	Total time	Number transac.	%	Total time
PM0	1470	18	00:04:45	1362	16,66	00:05:51
PM1	1192	15	00:04:47	1362	16,66	00:05:52
PM2	1467	18	00:05:21	1361	16,66	00:06:26
PM3	1365	17	00:04:46	1361	16,66	00:05:52
PM4	1377	17	00:04:38	1361	16,66	00:05:44
PM5	1297	17	00:04:06	1361	16,66	00:05:27
	Total 8168	Total 100	Highest 00:05:21	Total 8168	Total 100	Highest 00:06:26

Table 2: Results with heterogeneous cluster

	GetLB			Round-Robin		
Machine	Number transac.	%	Total time	Number transac.	%	Total time
PM0	653	7,99	00:07:10	1362	16,67	00:10:20
PM1	5	0,06	00:00:15	1362	16,67	00:10:22
PM2	1787	21,88	00:09:59	1361	16,66	00:11:08
PM3	1865	22,83	00:09:08	1361	16,66	00:10:18
PM4	1940	23,75	00:09:01	1361	16,66	00:10:10
PM5	1918	23,48	00;09:11	1361	16,66	00:10:21
	Total 8168	Total 100	Highest 00:09:59	Total 8168	Total 100	Highest 00:11:08

4. CONCLUSION

This paper presented GetLB - a load balancing model for electronic funds transfer scenarios. GetLB's **technical contribution** consists of its cooperative and extensible communication framework, which considers an autonomic interaction between processing machines (PMs) and the switch element for updating scheduling and notifications data. This organization optimizes the decision making regarding transactions assignment since network interaction does not take place. The **scientific contribution** of GetLB comprises its sche-duling heuristic called Load Level or LL. For each incoming transaction, LL functions are calculated according to the number of candidate machines and the lowest level indicates the target. LL considers both static data about the type of each transaction, and dynamic one about the network, CPU, memory, disk and subsystems status. Both resources and transactions levels of heterogeneity were evaluated in the current paper. Future work includes tests with resource dynamics and the use of notifications.

Acknowledgments

This work was partially supported by FAPERGS, GetNet and CNPq.

5. REFERENCES

[1] L. Xiaojing, W. Weiqing, and Z. Liwei, "Analysis of the impact of ecommerce to the changing of economic growth mode," in *IEEE Symp. on Robotics and Applications (ISRA)*, 2012, pp. 698–700.

[2] R. Sastre, S. Bascon, and F. Herrero, "New electronic funds transfer services over ip," in *IEEE Electrotechnical Conf.*, 2006, pp. 733 –736.

[3] C. Araujo, E. Sousa, P. Maciel, F. Chicout, and E. Andrade, "Performance modeling for evaluation and planning of electronic funds transfer systems with bursty arrival traffic," in *Intensive Applications and Services, INTENSIVE '09.*, 2009, pp. 65–70.

[4] E. Sousa, P. Maciel, C. Araujo, and F. Chicout, "Performability evaluation of eft systems for sla assurance," in *Parallel Distributed Processing, 2009. IPDPS.*, 2009, pp. 1–8.

[5] T. Tchrakian, B. Basu, and M. O'Mahony, "Real-time traffic flow forecasting using spectral analysis," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 13, no. 2, pp. 519 –526, june 2012.

[6] S. N. Mehmood Shah, A. K. B. Mahmood, and A. Oxley, "Analysis and evaluation of grid scheduling algorithms using real workload traces," in *Int. Conf. on Management of Emergent Digital EcoSystems*, ser. MEDES '10. ACM, 2010, pp. 234–239.

[7] P. Liang and J. Bigham, "A taxonomy of electronic funds transfer domain intrusions and its feasibility converting into ontology," in *Communications, 2006. APCC '06.*, 2006, pp. 1–5.

[8] M. Maurer, I. Brandic, and R. Sakellariou, "Adaptive resource configuration for cloud infrastructure management," *Future Generation Computer Systems*, vol. 29, no. 2, pp. 472 – 487, 2013.

[9] A. Aboulnaga and S. Babu, "Workload management for big data analytics," in *Int. conference on Management of data*, ser. SIGMOD '13. New York, NY, USA: ACM, 2013, pp. 929–932.