A Dynamic Load Balancing Strategy for a Distributed Biometric Authentication System

Ana-Maria Pricochi, Andreea Salinca, Sorin Mircea Rusu and Bogdan Ivascu
Research & Development, SOFTWIN, Bucharest, Romania

Keywords: Authentication System, Biometrics, Distributed Computing, Dynamic Signature, Load Balancing, Online Signature, Service-Level-Agreement, Task Scheduling.

Abstract: Biometrics are a nowadays trend in developing secure systems. Biometric authentication systems must provide a higher processing capacity, as biometric verifications are complex. Distributed computing relieves the computational burden, but the efficient scheduling of biometric authentication requests is still an open problem. This paper presents a distributed system for handling handwritten signature authentication requests, with a dynamic load balancer ensuring service-level-agreement compliance, low response times, fair use of resources and high availability. The load balancing method uses the service-level-agreements information, the classification of processing nodes and the classification of authentication tasks.

1 INTRODUCTION

The need for security constantly increases in the modern web world. In 2011, the Internet Crime Complaint Center received over 300,000 complaints, 28,915 being identity theft related crimes.

Biometry emerges as a reliable solution for security, providing a high level of authentication and identification. People do not have to remember passwords or personal identification numbers (PINs) or to possess something (e.g. smartcards). While these can be stolen, lost or forgotten, biometric data authentication implies `something which a person is or does` (Zhang, 2002).

Biometric authentication methods extract features and match an input data with a previously stored template data. Such operations have a greater complexity than computing the hash of a password and comparing two password hashes. Biometric authentication systems demand fast response times and high availability. In the web environment users can experience distress when an e-commerce transaction confirmation or a website login delays. The unavailability of authentication systems should not restrict the access to a critical area of a building.

Specialized web-based systems can provide biometric authentication services to applications needing to enhance their security. The complexity and the large number of authentication requests require high processing capacity systems. A solution is the usage of parallel and distributed computing. Considering cost/performance ratio and viability, distributed computing represents the best option (Grosu et al., 2002).

Modern distributed systems use heterogeneous architectures with both multi-core and many-core processing units (Li et al., 2011), using specialized co-processors (like Cell Broadband Engine Architecture – Cell/BE) and CPUs with graphics processing units (GPUs) or field programmable gate arrays (FPGAs) (Brodtkorb et al., 2010). Distributed heterogeneous architectures have proven their performance. Accordingly to Top 500, 62 of the world's best supercomputers use accelerators/co-processors, 2 being in the first 10.

A challenge in distributed systems is scheduling tasks while ensuring a balanced load of the processing nodes and complying with quality of service (QoS) requirements, particularly when nodes are heterogeneous (Grosu et al., 2002). In this case, the optimal mapping of tasks is, in general, a NP-hard problem (Braun et al., 2001).

The current paper has the following structure: Section 2 sets the background. Section 3 introduces our biometric authentication system and section 4 shows our load balancing method. Section 5 details the implementation. Section 6 discusses the results obtained so far. Section 7 presents the conclusions.

Pricochi A., Salinca A., Rusu S. and Ivascu B..
A Dynamic Load Balancing Strategy for a Distributed Biometric Authentication System.
DOI: 10.5220/0004354400650070
In *Proceedings of the 9th International Conference on Web Information Systems and Technologies* (WEBIST-2013), pages 65-70
ISBN: 978-989-8565-54-9

2 BACKGROUND

In systems with deadlines or other constraints, finding the optimal task assignment is not feasible.

To simplify the task scheduling problem, studies consider heuristics, dynamic programming and relaxation of requirements. Heuristics, popular approximation techniques, include Opportunistic Load Balancing, Min-min, Max-min, Minimum Execution Time, Minimum Completion Time, Tabu, Genetic Algorithm, Simulated Annealing (Braun et al., 2001). Another popular choice are directed acyclic weighted task graphs (DAGs) algorithms.

2.1 Related Work

(Naji et al., 2011) present a fingerprint biometric system, with a 2.47 seconds average task processing time and an overall accuracy of 99.48%. Tests are made on a server, although they prospect the usage of multiple authentication servers.

(Trevathan and McCabe, 2005); (Trevathan et al., 2009) studied distributed systems securing online payments using handwritten signatures. In their best testing configuration, the authentication method has a False Acceptance Rate (FAR) of 1.1% and a False Rejection Rate (FRR) of 1.0%, with no remarks on system capabilities.

(You et al., 2003) use a centralized scheme for task scheduling using mobile multi-agents that dynamically allocate and migrate tasks. The task scheduling algorithm provides an average round trip time of 5 to 7 seconds for 100 authentication trials.

The patent of (Yu et al., 1999) describes a web-based authentication system where the task allocation considers the type of biometrics specific servers can process and the content of their local template database. (Takagi, 2008) presents a load balancing apparatus using the load evaluation, the performance property of each authentication device (clock frequency) and the occupancy rate. (Miller et al., 2007) propose a biometric authentication system where the load balancing of biometric matching engines occurs at router level.

Previously mentioned methods concentrate on performance goals. The literature does not cover sufficiently the scheduling biometric authentication tasks to comply with client contracts. Algorithms related to machine learning can be optimized to any goal, but they lead to long scheduling times.

2.2 Proposed Model

The proposed model is a load balancing algorithm in a heterogeneous distributed biometric authentication system, using dynamic handwritten signatures. To comply with client contract, we define load balancing as a provisioning optimization problem. Fast mean response times and efficient utilization of computing resources are secondary goals.

The task distribution method uses worker nodes and tasks classification to make the assignment. The load balancing algorithm is dynamic, using statistical data to make state estimates.

The system uses heterogeneous worker nodes, assembling a worker farm in a local area network. Node configurations include CPUs and CPUs with GPUs. Although machines with GPUs solve faster, more tasks, simple dual-core CPUs can successfully process tasks requiring few processing resources.

3 BIOMETRIC AUTHENTICATION SYSTEM

The proposed method schedules tasks in an automated system of authentication through biometric signature (ATHOS). ATHOS is a SoA (Software Oriented Architecture) based system, offering online signatures authentication facilities to applications that improve their security measures.

Multiple clients can concurrently use the system. The ATHOS system conducts authentication transactions for its client based on SLAs (Service Level Agreements), mutually agreed upon and including terms like: accuracy of authentication, availability, bandwidth, average response time.

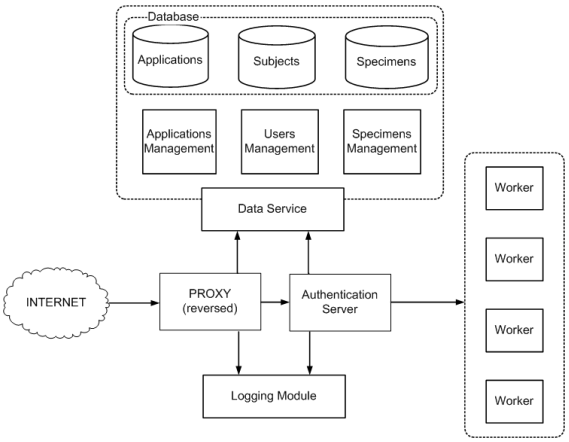

Figure 1: Biometric authentication center.

The ATHOS system comprises authentication centers with the following modules (Figure 1):

- A reversed *proxy server*, serving as a system

gateway, filtering and validating requests;
- An *authentication server*, with load balancing capabilities, formulating tasks based on the authentication requests and stored templates, and further distributing them to worker nodes;
- A cluster of *worker nodes*, running processes executing the authentication tasks received;
- A *logging module*, gathering and storing usage information during the system's operation.
- A *data service*, used for the management of the system databases;

To provide high availability, the ATHOS system can include several biometric authentication centers with mirrored databases and back-up servers for critical components, like the authentication server.

4 LOAD BALANCING SCHEME

The scheme refers to a multi-class dynamic centralized load balancing algorithm, assigning authentication tasks with different SLA demands to heterogeneous machines. The load balancing scheme must ensure that, for each client, the response time and the number of requests served complies.

ATHOS system requires an adaptive load balancing scheme as: the task arrival time is random, the workflow volume and SLA types vary, and the worker nodes have dynamic configurations and can process non-authentication tasks.

4.1 Task Model

The ATHOS system uses online signatures acquired with a special pen-like device, capturing both movement and graphic information while a person signs (Rusu et al., 2011).

The users subscribe templates, consisting of several specimen signatures. If the system needs to authenticate a user, the user offers an input signature to be verified. The authentication method compares template and input signatures based on the distances between the feature vectors extracted from the raw signals of the signatures. A classifier processes the distances to set the authentication response (Andrei et al., 2010). More than 1400 feature vectors comparisons are processed for each authentication. This method proved a FAR of 1.24% and a FRR of 14.59% on an evaluation of more than 7000 signatures, including skilled forgeries (Salinca et al., 2012).

Tasks are authentication requests, as they occur on regular basis and have greater complexity, influencing system performance. A user usually registers and deletes signatures once. Tasks have expiration time, priority and belong to a complexity class. A task expires when its response time exceeds the response time defined by its contract.

The priority of the task j depends on the SLA of the issuing application i and additional user data (Equation 1). The SLA defines an average response time ($T[i]$) and a maximum number of requests per minute ($N[i]$). $T_{\min}$ and $N_{\max}$ are the minimum, respective maximum of the mentioned values, for all SLAs. The group priority discriminates different users of the same application ($P_{group}[i]$). α, β and δ balance the weight of SLA parameters and group priority. They are specific to certain system deployments and have a great influence on the SLA compliance. Preliminary tests tune these values for the required performances.

$$P[j] = \alpha \frac{T_{\min}}{T[i]} + \beta \frac{N[i]}{N_{\max}} + \delta P_{group}[i] \quad (1)$$

Handwritten signatures have a complexity, consisting in loops, intersections, length, direction, acceleration and pressure variations. The complexity translates into a number of features and a computational complexity. Table 1 presents example processing times (on the same machine) for tasks belonging to the 5 signature complexity classes used in ATHOS.

Table 1: Execution times for signature complexity classes.

Very low	Low	Average	High	Very high
22 ms	40 ms	70 ms	110 ms	133 ms

Another classification is urgent and non-urgent tasks. An urgent task has such an expiration time that further delaying its assignment leads to deadline.

4.2 Worker Node Model

Worker nodes are independent computing machines with multi-core or many-core processing units. Many-core systems comprise GPUs. A machine runs multiple authentication processes, depending on its number of cores and multithreading capabilities. Processes on multi-core units solve one task at a time. Processes on many-core units solve multiple tasks in parallel (packages).

Worker nodes register to the authentication server, stating the number and type of authentication processes and receiving a list of load thresholds. The

node notifies the authentication server when reaching the processor or memory load thresholds.

The characteristics stored for each authentication process include: type and capacity, current load state of the hosting machine, stability and average response times for all task classes and, if necessary, all package sizes. The stability of an authentication process is the degree of confidence that it completes processing tasks, within the expected time, without failure or compromising response. The stability is the rate of successfully processed tasks, without deadline violation.

4.3 Load Balancing Method

The load balancing method implies the accumulation of authentication tasks and, eventually, distributing them to authentication processes. The distribution is a continuous repetitive process which, at each step:

- Selects a number of tasks;
- Classifies tasks as urgent or non-urgent;
- Selects the best workers for the current ratio of urgent and non-urgent tasks;
- Assigns urgent tasks to the best workers for this purpose;
- Assigns non-urgent tasks to the best workers for this purpose.

The major components of the load balancing scheme are the state estimation and the distribution module. These components interact with other modules and data structures (as shown in Figure 2).

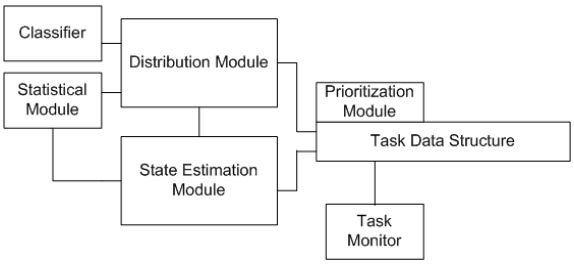

Figure 2: The architecture of the load balancing scheme.

Accumulating tasks means storing requests on arrival in a prioritized structure. The distribution module selects first the tasks of SLA demanding applications. To prevent starvation of requests with less demanding SLAs, a task monitor periodically upgrades low priority tasks. The number of selected tasks depends on the worker farm capacity.

The distribution module determines the proportionality between urgent and non-urgent task in the selected list. The classifier module uses this information, along with state estimates, to determine the scoring of each worker. When most tasks are urgent, the classifier highlights processes that have high processing capabilities (multiple authentication processes and/or parallelization capabilities), low response times and great stability (redistribution avoidance). Low performance processes solve the remaining non-urgent tasks. If non-urgent tasks dominate, the classifier determines for the few urgent tasks fast processes with low capacity. It highlights high capacity nodes, with low stability and high response time for non-urgent tasks. High performance processes remain unused, if possible. Unassigned tasks in the current step return to the prioritized structure, with an increased priority.

State estimation and control operations run concurrently with the distribution. The statistical module stores data of previously solved tasks, for short periods of time (to reflect the recent state). The state estimator evaluates authentication processes, for new worker nodes in the farm and for nodes not recently used. The evaluation includes test tasks that cover all complexity classes and package sizes. The state estimator receives worker node information on event (processor and memory load changes, start/stop of processes). State estimates for authentication processes include average response times for different complexity classes and package sizes, operational state and load level.

The state estimation module detects system full loading: the simultaneous compliance with all SLAs is not possible with the current worker farm. At full load, the system degrades all SLA parameters with a certain percentage. If the performance degradation exceeds 25%, the system refuses incoming requests, returning a `System too busy message`.

Equation 2 reflects the condition for full load. $\#A$ is the number of applications, $\#T_i$ is the number of tasks issued by application i, $t[j]_{statistic}$ is the average statistic time for task j class, $t[i]_{SLA}$ is the SLA guaranteed time of application i and T_C is the statistically determined control time.

$$\sum_{i=1}^{\#A}\sum_{j=1}^{\#T_i} t[j]_{statistic} \geq \sum_{i=1}^{\#A}\sum_{j=1}^{\#T_i} (t[i]_{SLA} + T_C) \qquad (2)$$

The load balancing scheme allows redistribution, when a worker node fails during task processing or it delays the response. We choose redistribution as the worker farm is reliable and redundancy wastes computing resources.

5 IMPLEMENTATION

To evaluate the performance of the proposed load balancing model, we test real case scenarios for different system configurations.

We deploy the system, consisting of several Web Services, on multiple servers. We implement the Web Services using the Windows Communication Foundation (WCF) API and the C# programming language. Internet Information Services (IIS) is the web hosting solution. We configure the Web Services and IIS for optimal performances.

We simulate concurrent requests using JMETER and soapUI. The test model includes multiple SLAs.

The flow of client authentication requests consists of real signatures, acquired in a previous data collection stage (Salinca et al., 2012). The signature database has more than 7000 signatures, collected from 113 people. The selected signatures meet the distribution of different signature classes identified in the database.

6 RESULTS

We run various tests, including different loads and worker nodes configurations, over various periods.

Table 2 shows the characteristics of different worker nodes: number of authentication processes (column 2), number of tasks handled in parallel (column 3), average response time for medium complexity tasks (column 4), average number of tasks processed per minute (column 5).

Table 2: Characteristics of different worker nodes.

Hardware configuration	No. proc.	Task capacity	Avg. time	Task rate
Quad-core (multithreading)	5	1	510 ms	600
Quad-core (multithreading)	8	1	551 ms	650
Dual-core with 2 GPUs	2	5	554 ms	700
Quad-core with 3 GPUs	3	7	290 ms	4000

Table 3 shows the SLA compliance for both low and high system loads, comparing an earlier version of the load balancing (column 3) and the presented model (column 4). The earlier version assigns tasks on arrival to the first available machine capable of respecting the tasks deadline. The compliance is the percent of time in which the system respected the stated SLA parameter, considering each application separately.

Table 3: Client SLA compliance comparison for a low load simulation and a high load simulation.

Load	SLA parameter	Compliance in earlier version	Compliance in current version
Low	Maximum requests number	100%	100%
	Average response time	96.35%	100%
High	Maximum requests number	98.4%	99.6%
	Average response time	86.54%	90%

The standard deviation of overall performance degradation for each application, in the high load simulation, decreased from 16% to 14%. This means an increased fairness of performance degradation for applications with different SLAs.

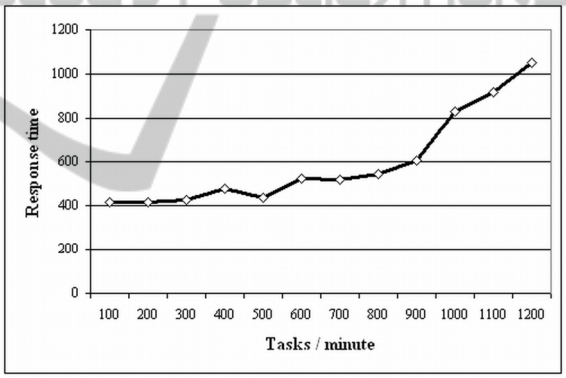

Figure 3: Average response times for different tasks rates.

Figure 3 shows the evolution of system response times for different system loads (different task rates). The system comprised two worker nodes, the second and fourth configuration from Table 2.

Table 4: The task distribution between two machines (one with a CPU and one with 2 GPUs) and between the GPUs.

Tasks / min	% tasks on machine 1	% tasks on machine 2 (% on GPU 1 - % on GPU 2)
200	50%	50% (50% - 0%)
400	50%	50% (48% - 2%)
600	50%	50% (30% - 20%)
800	52%	48% (23% - 25%)
1000	46%	54% (27% - 27%)
1200	41%	59% (29% - 30 %)

The algorithm also ensures fair distribution between

worker nodes, to a certain degree. Table 4 presents the tasks assignment to different machines and authentication processes. Machine 1 is a quad-core with 5 authentication processes. Machine 2 is a workstation with 2 GPUs, each running an authentication process executing up to 5 tasks in parallel. At low loads, the tasks assignment is balanced among machines, while one authentication process on machine 2 remains free. As the load increases, the algorithm assigns more tasks to the free process, to the point it exploits machine 2 more, as it has better performance. The assignment between the 2 processes on machine 2 is balanced.

7 CONCLUSIONS

This paper presents a strategy for load balancing in heterogeneous distributed biometric authentication systems. The load balancing algorithm is a SLA optimization problem. Real case test scenarios, involving data acquired from potential users of such a system and actual system deployments show the high performance of the proposed algorithm. Furthermore, the experimental results confirm that such an authentication system can scale while still offering high QoS.

The algorithm manages to comply with all client SLAs in normal operation mode and complies with SLA parameters of 99.6%, respective 90% when the system exceeds its processing capacity. Experimental results demonstrate that rates of 6000 request per minute can be reached, establishing the scalability of the system.

ACKNOWLEDGEMENTS

This work is a result of the research in the project "Automated system of authenticating through biometric signature/ATHOS", co-funded from SOP IEC PA2: Research, Technological Development and Innovation for Competitiveness, Operation 2.1.2: "Complex research projects fostering the participation of high-level international experts".

REFERENCES

IC3, 2011. Internet Crime Complaint – U.S. Bureau of Justice, Internet Crime Report, http://www.ic3.gov/media/annualreports.aspx.

Zhang, D., 2002. *Biometric Solutions for Authentication in an E-World*. The Springer International Series in Engineering and Computer Science.

Grosu, D., Chronopoulos, A., Leung, M., 2002. Load Balancing in Distributed Systems: An Approach Using Cooperative Games. In *Proceedings of the 16th IEEE International Parallel and Distributed Processing Symposium*, pp. 501-510.

Braun, T., Siegel, H., Beck, N., Boloni, L., Maheswaran, M., Reuther, A., Robertson, J., Theys, M., Yao, B., 2001. A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing System. In *Journal of Parallel and Distributed Computing*, pp. 810-837.

Li, T., Narayana, V., El-Ghazawi, T., 2011. A Static Task Scheduling Framework for Independent Tasks Accelerated using a Shared Graphics Processing Unit. In *IEEE 17th International Conference on Parallel and Distributed Systems*, pp. 88-95.

Brodtkorb, A., Dyken, C., Hagen, T., Hjelmervik, J., Storaasli, O., 2010. In *Scientific Programming*, IOS PRESS, pp. 1-33.

Top 500 Super Computers, 2012. November 2012, http://www.top500.org/.

Naji, A., Housain, A., Zaidan, B., Zaidan, A., Hameed, S., 2011. Security Improvement of Credit Card Online Purchasing System. In *Scientific Research and Essays*, Academic Journals, pp. 3357-3370.

Trevathan, J., McCabe, A., 2005. Remote handwritten signature authentication. In *Proceedings of the 2nd International Conference on e-Business and Telecommunication Networks*, pp. 335-339.

Trevathan, J., McCabe, A., Read, W., 2009. Online Payments Using Handwritten Signature Verification. In *6th International Conference on Information Technology: New generations*, pp. 901-907.

You, J., Zhang, D., Cao, J., Guo, M., 2003. Parallel Biometrics Computing Using Mobile Agents. In *Proceedings of the International Conference on Parallel Processing*, pp. 305-312.

Yu, Y., Stephen, W., Hoffberg, M., 1999. *Web-Based Biometric Authentication System and Method*. U. S. Patent and Trademark Office, US5930804.

Miller, J., Willis, W., Lau, J., Dlab, B., 2007. *Multimodal Biometric Platform*. U. S. Patent and Trademark Office, US7298873.

Takagi, J., 2008. *Load Balancing Apparatus*. U. S. Patent and Trademark Office, US0031496.

Rusu, S., Dinescu, A., Diaconescu, S., 2011. *Systems and methods for assessing the authenticity of dynamic handwritten signature*. World Intellectual Property Organization WO/2011/112113.

Salinca, A., Rusu, S., Diaconescu, S., 2012. An approach to data collection in an online signature verification system. In *Proceedings of the WebIST'12,* pp. 251-254.

Andrei, V., Rusu, S., Diaconescu, S., 2010. Solutions for speeding-up on-line dynamic signature authentication. In *Proceedings of the 12th International Conference on Enterprise Information Systems*, pp. 121-126.