EC2-LB

An application scaling system in Elastic Compute Cloud

Authors

Włodzimierz Funika(1,2), Kamil Mazurek(1), Wojciech Kruczkowski(1)

(1) Department of Computer Science, AGH - University of Science and Technology, al. Mickiewicza 30, 30-059 Krakow, Poland
(2) ACC CYFRONET AGH, ul. Nawojki 11, 30-950 Krakow, Poland

funika@agh.edu.pl, mkamil@student.agh.edu.pl, krwojc@student.agh.edu.pl

Introduction

Since the implementation of Amdahl's law in 1960s parallel computing has changed, but the goal was always the same: to execute computations as fast as possible. Many models of parallel computing have been introduced over the years, for example: Grid Computing, computation on multi processor machines and computing on clusters. Lately, a new approach known as Cloud Computing has been introduced.

A Cloud Computing system must combine dynamic allocation with effective external and internal cloud communication and data storing. One of the main cloud features is transparency - people who are using a cloud system do not need to know what is the system architecture.

With transparency a problem comes to solve - balancing the load of machines and using them with the best performance.

Motivation

The concept of our project relates to the UE Urban Flood project. It is aimed at processing huge quantities of data coming from a network of sensors by the Internet.

Our aim is to create a mechanism for processing these data in form of a system which executes jobs while exploiting load balancing procedures for Amazon EC2 [1]. The idea is to create an experimental hybrid architecture in which some parts of the system are centralised, whereas other ones are not.

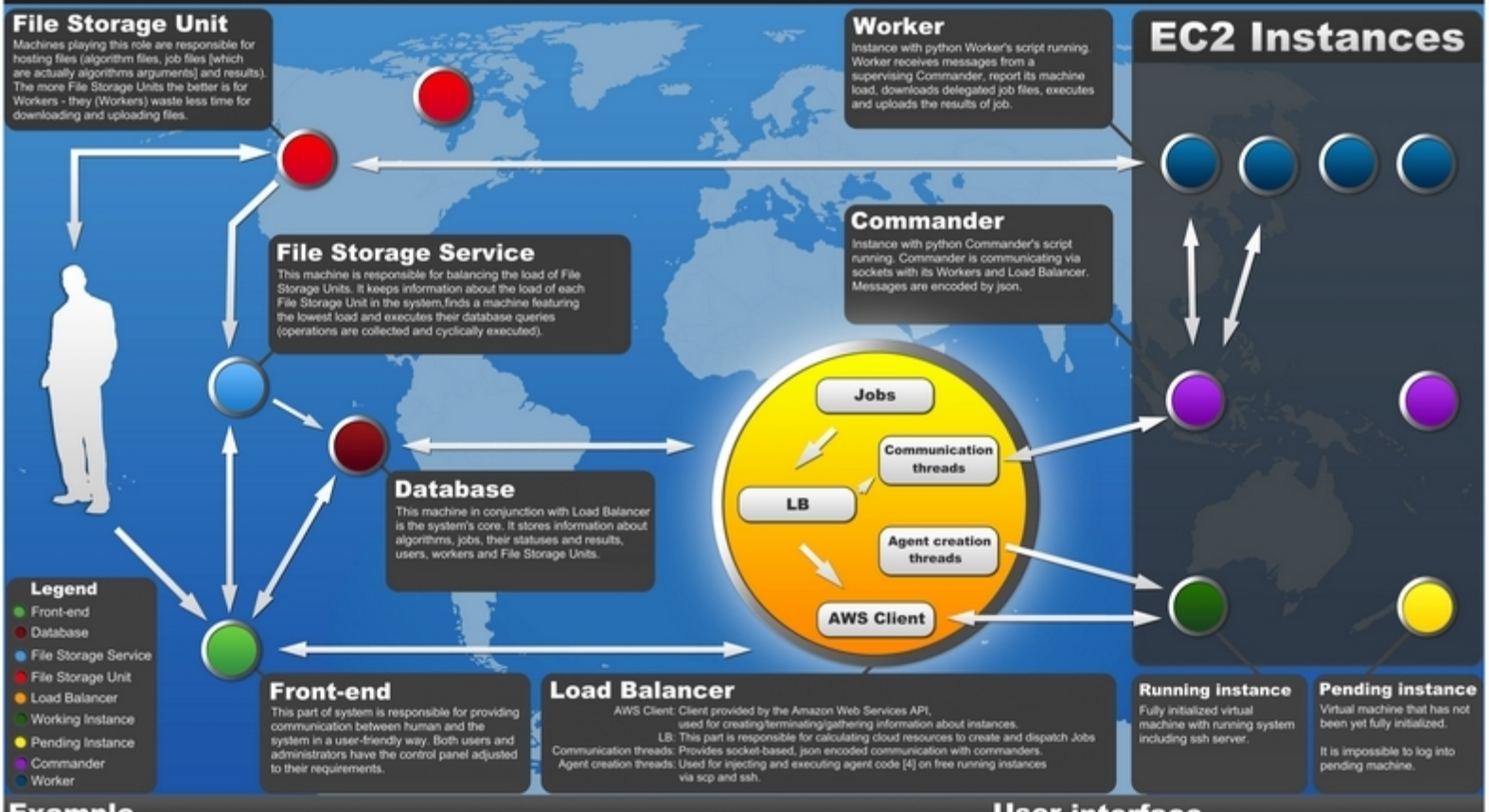
We try to balance dependencies between the system units by using a hierarchic structure consisting of masters and slaves [5], with one distinct root unit. Scientific objectives are focused on measuring the efficiency of such a solution and finding the best load balancing algorithm under given load conditions.

Scope of research

Main objective of this research is to measure the efficiency of such an architecture (shown on picture below). In our tests we are using Eucalyptus Community Cloud [2] and Cloud resources provided by Future-Grid [3]. Yet our resources are limited - it is possible to create up to X instances. We are also limited by network communication bandwidth which may affect jobs execution. We are able to use only one virtual machine so far, what means that all the modules (except EC2 ones) are running on the same server. In the future we are going to test it with real-life applications.

In our research we seek to investigate the efficiency of our hybrid structure, find the most efficient load balancing algorithm for it and to provide a fast and user-friendly system.

Architecture



Example

1. User adds a new algorithm to System using Front-end (it is saved in the Database).
2. User adds a new job to System using Front-end (it is saved in the Database).
3. File Storage Service finds a File Storage Unit with the lowest load.
4. Front-end redirects User to the found File Storage Unit.
5. User uploads files on the File Storage Unit.
6. File Storage Unit updates its load and informs the File Storage Service about new files stored on it.
7. File Storage Services updates information about the File Storage Units load and updates information about the job and algorithm in the Database (now their statuses are "Ready to process").
8. Load Balancer gets the jobs to be executed from the Database and changes their statuses to "In progress".
9. Load Balancer scales cloud resources to fit the execution of the current amount of jobs.
10. Load Balancer dispatches jobs between Commanders using load balancing algorithms.
11. For each job to dispatch Load Balancer sends a message via sockets to Commander.
12. When Commander receives the message it dispatches tasks between its Workers.
13. Worker receives the job execution request message.
14. Worker queues a job downloading.
15. Worker downloads job files from the File Storage Unit.
16. When the download is finished Worker queues a job execution.
17. After finishing the job execution results are uploaded to the File Storage Service.
18. File Storage Unit updates its load and informs the File Storage Service about new files stored on it.
19. File Storage Services updates information about the File Storage Units load and updates information about the job in the Database (now its status is "Complete").
20. User can download results using Front-end.

User interface

System control is provided by Front-end

Future work

To eliminate complicated source injection into instances we are planning to create our own virtual machine image which will provide Worker and Commander code as firmware.

We will also try to test our architecture on a larger set of machines and cloud resources.

References

[1] Amazon EC2 web site: http://aws.amazon.com/ec2
[2] Eucalyptus web site: http://open.eucalyptus.com
[3] Future Grid web site: https://portal.futuregrid.org/
[4] Cetnarowicz, K. From algorithm to agent. In: Sloot, P. et al. (eds.) Proc. 9th international conference ICCS 2009, LNCS 5544, Springer, pp. 825–834.
[5] Antonis, K., Garofalaskis, J., Mourtos, I., and Spirakis, P.: A Hierarchical Adaptive Distributed Algorithm for Load Balancing, Journal of Parallel and Distributed Computing, Vol. 64 (2004): 151-162

Acknowledgements

The authors are grateful to prof. K. Cetnarowicz and dr. K. Rycerz for valuable discussions.
Our thanks go to Tomasz Bartynski for fruitful co-operation.
We also want to thank Future-Grid https://portal.futuregrid.org/ for delivering cloud resources.
This research is partly supported by the UrbanFlood project and the European Union within the European Regional Development Fund program no. POIG.02.03.00-00-007/08-00 as part of the PL-Grid Project (http://plgrid.pl).

INNOVATIVE ECONOMY
NATIONAL COHESION STRATEGY

CYFRONET

AGH

Future Grid