Simulator of a “Weather” Cloud

Ksenia Khramenkova
State University of Aerospace Instrumentation
Saint-Petersburg, Russia
ksu.khramenkova@gmail.com

Olivier Hermant, Renaud Pawlak
Institut Supérieur d'Electronique de Paris
Paris, France
{renaud.pawlak,olivier.hermant}@isep.fr

Abstract

In this article a cloud simulator is considered. It is created for modeling a cloud architecture, for possibility to modify architecture or algorithms and for analyzing result and performance, to be able to avoid introducing the errors in a real cloud. We consider two methods of request allocation policies to the nodes and compare results of their behavior in terms of interaction with nodes’ cache-memory.

Index Terms: Cloud Computing, modeling, algorithms, cache.

I. Introduction

“Cloud” computing is a relatively recent term, built on decades of research in virtualization, distributed computing, utility computing, and more recently networking, web and software services. It implies a service oriented architecture, reduced information technology overhead for the end-user, great flexibility, reduced total cost of ownership, on-demand services and etc [1].

Cloud computing delivers infrastructure, platform, and software as services, which are made available as subscription-based services in a pay-as-you-go model to consumers. These services in industry are respectively referred to as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).

Cloud computing can be defined as “a type of parallel and distributed system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumers”. Some examples of emerging Cloud computing infrastructures are Microsoft Azure [2], Amazon EC2 [3], Google App Engine [4], and Aneka [5] [6].

In this article the creation of a cloud simulator and the working with nodes are considered. An algorithm of sending requests by the controller to the nodes is implemented so that the largest possible part of the requests is already in the chosen node's cache memory. The goal of this task management algorithm is to reduce the processing time of requests.

In the first section we talk about terms of cloud computing. The second section describes a real cloud. The third section introduces the cloud simulator, the fourth describes results and the last is a conclusion.

II. Cloud description

CAP 2020 is a French company that aims at providing various services for the French agriculture, especially services that help the farmers to analyze agricultural data in their fields in order to perform more efficient and environmental-friendly farming. CAP 2020 uses different sources of existing data such as weather data coming from existing weather stations, or specific weather or agricultural sensors. Examples of such sensors can be temperature, soil or air humidity sensors, soil acidity, and so on. Each farmer or farming group may use its own way to gather data, including M2M platforms. For cost reasons, most farmers use existing sensor networks and M2M platforms such as european-wide weather services like Meteo France and Numtech. However, the precision of such continental services is not sufficient for the farmers who need to work at very small scale to optimize their production (1km x 1km areas). Thus, farmers use specific sensor networks such as PESSL (weather) or Agriscope (weather and agriculture specific sensors). CAP 2020 integrates the data coming from all these sources with interpolation in order to provide homogeneous services to the farmers, as close as possible to their fields. The weather service of CAP 2020 allows the farmers to define Virtual Weather Stations (VWS) in their fields for accessing weather history and forecast at a very precise point.

The provided data will be calculated from weather data grids (a set of points and associated data) coming from our main providers, and from many local sensor data which will be integrated to the grids through a global interpolation algorithm implemented and hosted within a dedicated server by Estimages, a company specialized in data interpolation and imaging for oil company. Thus, in order to know the value of a given data for a given VWS, we take the available grids, including Estimages generated grids, and we perform a local interpolation operation. The grids which are selected for local interpolation will depend on the VWS's user and its access rights to a given data type and precision for that given data type. Indeed, for the same VWS, there will be several possible grids providing the data, but of different precisions. There are currently the following precisions available in the service:

- low precision: square grids providing one point every 15km (15km x 15km precision), these are used only for backup only, in case the standard or higher precision are not available
- standard precision: square grids providing one point every 5km (5km x 5km precision)
- high precision: square grids providing one point every 1km (1km x 1km precision)
- ultra-high precision: lower than 1km precision (not implemented yet) [7].

III. Weather cloud simulator

A cloud simulator is used for modeling cloud architecture for performance analysis and optimization. During a simulation bugs can be discovered and fixed before moving on to the implementation phase in which debugging will be a lot of work.

Fig. 1 presents the schema of the cloud simulator architecture. The controller is connected to each node via the channels. The bandwidth of each channel is adapted dynamically depending on the number of active channels, so as to be closer of a real network. Each node contains resources like CPU power, memory and cache (Cache algorithm in nodes is a Last Recently Used algorithm) and provides a number of services which uses the node's resources in a predefined way. Modeling the cache memory in nodes is of crucial importance since it has been noticed, in the real Cloud of section 2 [7], that requests on files already in cache memory takes up to four times less, which must be reflected in this simulator.

The controller receives requests from users. After the controller by task allocation algorithm sends them to the nodes. Each node processes the request and performs the specified service. The node then issues a response, which is a result of that service like computation results or storage, back to the controller. The controller sends the response to the user. This is done using a discrete synchronized time, step by step. In particular, we aim to measure the "logical time" taken by a request to be processed.

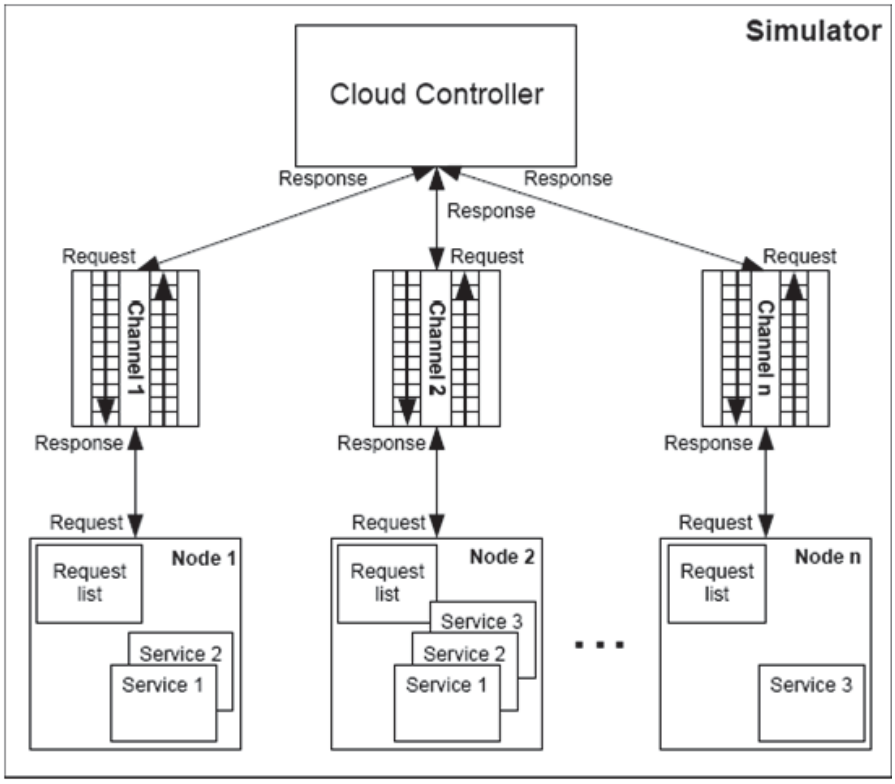

Fig. 1 Schema of cloud simulator

There are several auxiliary modules in the cloud simulator such as:

- Services which are provided by nodes.
- Requests which demand the node to perform a specific service.
- Responses which are the result to the specific service. They are sent in return to the controller.

The cloud controller aims to store nodes and requests statistics, send requests to chosen nodes (via the channel) and get response from nodes (via channel).

A request is characterized by its service type, its size and the list of documents it needs from the hard disk drive, that are supposed to be identical in each node. For the simulation, we chose to have 1000 different documents, that may be present in the cache memory of a node, or not. The size of the cache memory of each node is 250.

Each service implements three methods which, depending on the request's size, evaluate the memory and the CPU work required and calculate the output data size. The CPU work required may depend on the percentage of cache hit as in the simulation of the Weather Cloud of Section II, or may not. It also depends on the input data size in different manner, for instance for a naive sorting algorithm, it is quadratic on the input size.

Two task management algorithms are used by the controller:

- Round Robin (RR) algorithm does not analyzes the request. The controller selects a node pointed to by a circular counter from a list, after which the counter is incremented, modulo the total number of nodes. This algorithm does not need to know anything about nodes or its cache-memory tables.
- The algorithm with an analysis of the contents of the request (AAR) analyzes the content of the request namely the array of documents requested by user. The algorithm assumes that documents between 0 and 249 are in node 1, between 250 and 499 – in node 2, between 500 and 749 – in node 3 and between 750 and 999 – in node 4. The controller receives and analyzes the percentage of requested documents (cache hit) for each node and chooses by the biggest value where the current request have to go. Thus cache hit have to grow and time required for processing have to decrease.

VI. Simulation results

As a result of the modeling we get a table containing: task identifier, service type, node identifier, start time, finish time, cache hit percentage, input data size, output data size.

The values of the last two parameters can vary. They depend on the called service and of the chosen size of the input. The task identifier is the number of the request. This parameter is important for analyzing dense stream of requests. Because in a cloud there can be low-power nodes, which slowly process requests, there is a parameter node identifier. It is needed to check each node in the cloud and watch them during the entire simulation. The start time describes time when request entered node. Finish time describes the time when the response reaches controller. If work was with cache in table there is the percentage of cache hit.

The comparisons are done for RR and AAR algorithms; array of documents in request is filled randomly. The service analyzed is a simulation of the Weather Service discussed in Section II, that uses the cache memory.

The fig. 2 shows processing request time against the amount of requested documents. With increasing volume requested documents the processing time using the AAR grows, but the processing time using the RR algorithm decrease. If the amount of requested documents is more than 100 per request, the idea to analyze requests fails, because processing time by node is the same.

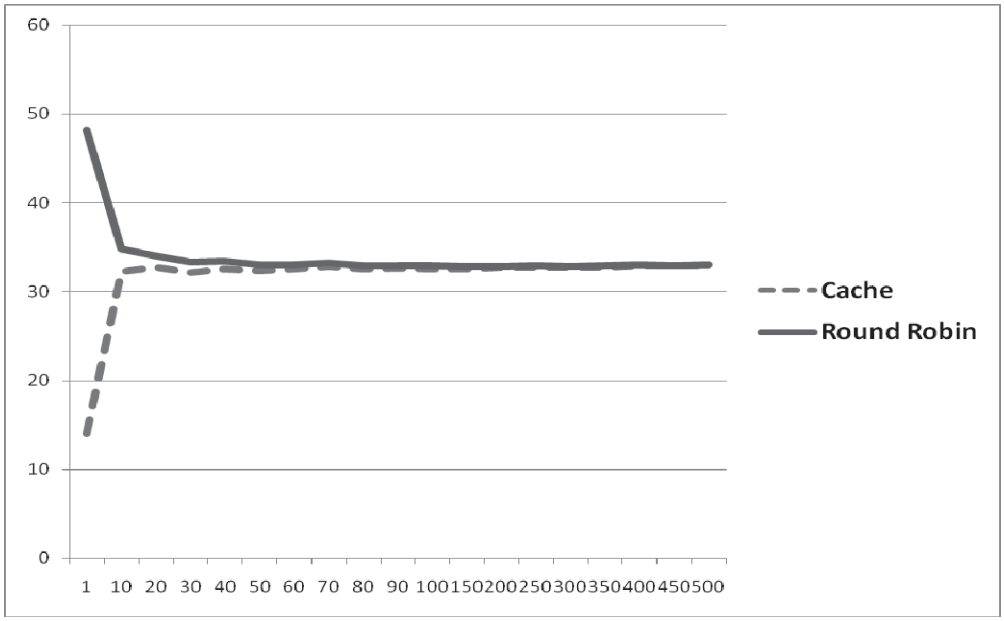

Fig. 2 The processing request time

The percentage of cache hit decreases when the amount of requested documents grows (see fig. 3). When the number of requested documents is 100 per request, the work of AAR is equal to the work of the RR algorithm. So it does not matter what algorithm has been chosen in the controller to distribute requests.

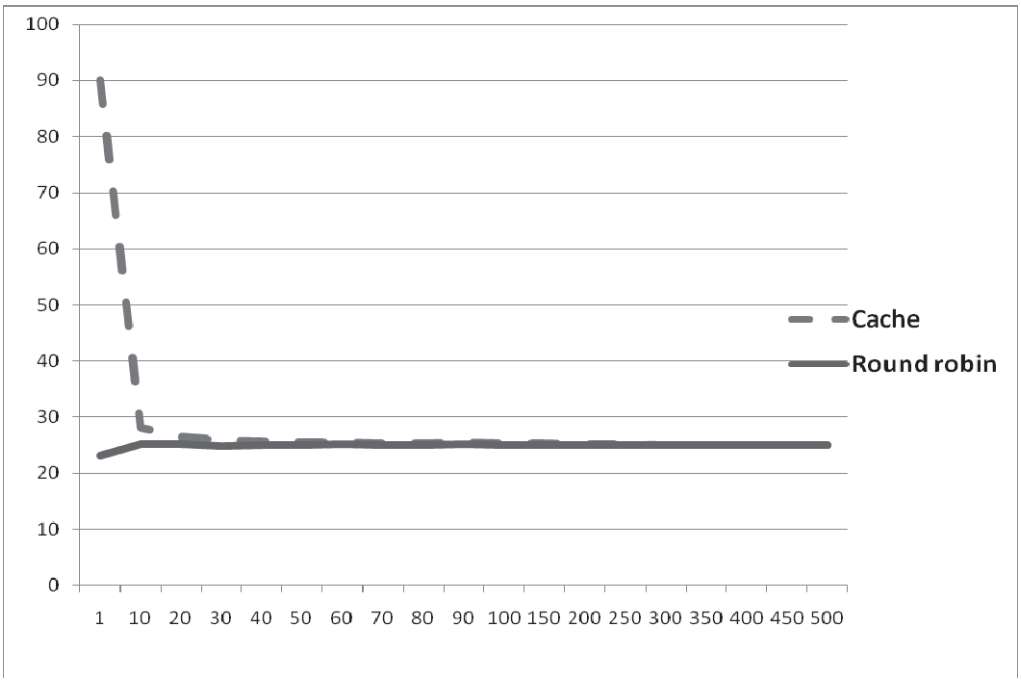

Fig. 3 The percent of cache hit

V. Conclusion

This article discusses the simulation of a cloud that contains information about weather in France. Comparison was done for 2 algorithms: for Round Robin algorithm and for the algorithm with an analysis of the contents of the request. The results of the analysis are: the second algorithm gives a small gain in time and cache hit on small amount of requested data. So, we need to improve the algorithm with an analysis of the contents of the request. When requests are distributed by controller, it has to know dynamically what documents are already in cache-memory in each node. After analyzing array of requested documents in current request, the controller have to choose the optimal node for the current request, from the point of view of working with cache-memory.

References

[1] Mladen A. Vouk, Cloud Computing – Issues, Research and Implementations, Journal of Computing and Information Technology - CIT 16, 2008, 4, 235–246 doi:10.2498/cit.1001391.

[2] D. Chappel, Introducing the Windows Azure Platform, http://www.microsoft.com/windowsazure/Whitepapers/introducingwindowsazureplatform/.

[3] Amazon Web Services, Auto-scaling Amazon EC2 with Amazon SQS, http://aws.amazon.com/articles/Amazon-EC2/1464.

[4] Google.com, Google App Engine General Questions, http://code.google.com/intl/ru-RU/appengine/kb/general.html.

[5] Y. Wei, K. Sukumar, C. Vecchiola, D. Karunamoorthy, R. Buyya, Aneka Cloud Application Platform and Its Integration with Windows Azure, Cornell University Library, http://arxiv.org/abs/1103.2590.

[6] R. Buyya, R. Ranjan and R. N. Calheiros, Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities.

[7] R. Pawlak, Z. Kazi-Aoul, D. Boisgontier, Private Cloud-Computing to the Rescue, ISEP.