Ad Hoc Networks 35 (2015) 37–50

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

An empirical study on implementing highly reliable stream computing systems with private cloud

Yaxiao Liu[*], Weidong Liu, Jiaxing Song, Huan He

Department of Computer Science and Technology, Tsinghua National Laboratory for Information Science and Technology (TNList), Tsinghua University, Beijing, China

ARTICLE INFO

Article history:
Received 8 December 2014
Revised 5 June 2015
Accepted 13 July 2015
Available online 26 July 2015

Keywords:
Stream computing
Cloud
Reliability
Banking system

ABSTRACT

Stream computing systems are designed for high frequency data. Such systems can deal with billions of transactions per day in real cases. Cloud technology can support distributed stream computing systems by its elastic and fault tolerant capabilities. In a real deployment environment, such as the pre-treatment system in Chinese top banks, the reliability based on user experience is key metrics for performance. Although many significant works have been proposed in the literature, they have some limitations such as less of architectural focus or difficult to implement in complex projects. This paper describes the reliability issue which is caused by the service downgrade in cloud. We use novel reliability analysis techniques, queuing theory, and software rejuvenation management techniques to build a framework for supporting stream data with low latency and fault tolerance. A real streaming system from a top bank is studied to provide the supporting data. Operational parameters such as rejuvenation window and time-out parameter are identified as key parameters for the design of a distributed stream processing system. An algorithm for reliability optimization, monitoring and forecast is also introduced. The paper also compares the improved result with original issues, which saved millions of money and reputations.

1. Introduction

Stream systems are designed to support continuous online data processing [1]. Many stream computing systems are designed to deal with the velocity characteristic of big data [2], which often requires stream systems to label, extract or generate events from original data. Since the data stream is continuous, the reliability of stream processing system will impact the quality of data outputs significantly.

The cloud computing technology can provide elastic support to stream processing systems by helping them to handle data volume fluctuations. The fast rejuvenation of cloud services can also provide fault tolerant support.

In some real cases, we find that operational parameters also play an important role in stream management. For example, if a distributed system used software rejuvenation for fault tolerant, the system might cause queue overflow before the completion of rejuvenation in cloud. We also observe some cases of reliability downgrade caused by cloud system hardware failure. Even if the stream system is distributed in different virtual server zones, the unexpected queue overflow would also be encountered. The unexpected reliability in stream systems would cause big loss in business cases. We decide to solve the challenge by improving reliability of cloud based stream systems from an architectural view.

In most cases, cloud service providers offer service-level agreements (SLAs) based on the 'availability' of cloud services [3]. 'Availability' means the 'up' time of single cloud services regardless user's experience. For example, a cloud based load balancing services may be blocked by tremendous data flow while all cloud virtual servers are 'up'. From cloud users' view, the system is 'unreliable', while from the cloud

[*] Corresponding author. Tel.: +8613910752612.
E-mail addresses: rootliu@gmail.com (Y. Liu), liuwd@tsinghua.edu.cn (W. Liu), jxsong@tsinghua.edu.cn (J. Song), heh906@gmail.com (H. He).

http://dx.doi.org/10.1016/j.adhoc.2015.07.009
1570-8705/© 2015 Elsevier B.V. All rights reserved.

provider's point of view, the services are 'available'. Reliability plays an important role here to link business case with cloud performance.

We propose several architecture patterns in banking private cloud [4]. The all patterns have specific needs in private cloud. Stream systems could be considered as a special application architecture pattern in cloud environment [1]. Our research is part of a general study of managing different banking application architecture patterns in cloud.

Cloud availability/reliability [5–8] have been discussed in many papers. The empirical study comes from the practice for designing stream processing systems on private cloud for a major bank in China. The stream system helps the bank to deal with 3rd largest data processing requirements in China with more than 200 million daily transactions.

In our previous work [4], we provide a methodology to manage the cloud deployment pattern from business, cloud and physical layer. In this paper, we focus on the stream aspects with the same layers. We analyze the mathematical considerations to manage reliability. We also describe how it is important to implement and manage the pre-treatment processing stream in Chinese top banks. After careful study, we provided 3 layers' improvement solutions. The solutions are proved to be successful by the deployment in production environment. The deployment shows an improvement by 80% under the same issues. We provide both novel methodology and empirical data to support our contributions.

Our contributions include:

1. Define reliability as an architectural metric for cloud based business services. The definition could be measured by the variance of reliability instead of calculation of many uncertain components.
2. Design a methodology to use above metrics to evaluate cloud based services components. For complexity reason, the study focuses on stream computing in cloud.
3. Resolve business challenges in high volume transactions in Chinese top banks. The solution prevents ATM processing system to lose of thousands of transactions, which help the bank to maintain both revenue and reputations.

This paper is organized as follows:

- In Section 2, we review the related work in this area and introduce our contributions on architectural methodology and the variance of reliability records.
- In Section 3, we describe the challenges from a banking system and provide analysis based on the framework in Section 2.
- In Section 4, we provide the optimization solutions and results comparison.
- In Section 5, a prospection of our work is provided together with conclusions.

2. Related works and our contributions

2.1. Related works on cloud reliability

Cloud reliability could affect the stream system's fault tolerance significantly. Some researches [5,6] showed concerns about reliability and availability in cloud computing virtual machines.

In a common on premise IT environment, IT availability is defined as [5,9]:

$$\text{Availability} = \frac{\text{Total Service Time} - \text{Downtime}}{\text{Total Service Time}}. \tag{2.1}$$

Service reliability is defined as the possibility of a service to perform its intended function under stated condition [9]. Reliability cares about the system to provide acceptable services, which means correct or accurate service delivered within an acceptable time. If a system is up but takes long time to perform, it is considered unreliable.

Another formula may express the idea more clearly [5]:

$$\text{Reliability} = \frac{\text{Number of Successful Responses}}{\text{Total Number of Responses}} \tag{2.2}$$

For a specified checkpoint, a service could obtain track record of possibility of successful return from (2.2).

A cloud system contains multiple architectural layers, such as bare metal servers, hypervisors and cloud management software. The work by [8] identified that, in a large cloud enabled datacenter (i.e. Microsoft cloud), the most frequent failure events come from storage systems, especially the storage system with multiple disks and RAID controllers. From an application user's view, the business transactions' reliability may or may not be affected by the failure. It depended on whether there was a high availability solution to avoid the impact or the application could store its data in different storage systems.

Authors in [5] gave several guidelines for cloud deployment, such as deploying application nodes in different virtual machines to share load and deploying virtual machines in different hypervisor zones to avoid a single point failure. They also discussed the latencies for different requirements. Other research work [3,6,8] gave more details in practice on using elastic capabilities to support stream fault tolerance. One thing to be noted is that the 'rapid elastic' capabilities can support both low latency and fault tolerance.

In our study, we find that we can use the variance of reliability for a specific architectural pattern to manage reliability instead of accurate variables which are more difficult to obtain.

2.2. Related works on distributed stream processing

Recently, internet architecture and protocol optimization is also a hot topic [12,18], it's related to our paper as an entry point for distributed stream processing. The velocity attribute of big data often requires a stream processing system to provide high performance capabilities. Since data source will send data packages continuously, stream systems have to process data in limited time slot.

Streams can use parallel sub-streams to scale up the required time slots. Each sub stream will handle data labeling, business event identification or word counting tasks simultaneously. Low latency design will ensure the processing not to block future data flow. Fault tolerant design will help stream system not to lose processing capabilities.

There are several systems designed to focus on large-scale and low-latency stream computation, such as Apache S4 [10], Twitter Storm [11], etc.

In the S4 system, the system keeps computation inside memory and partitions the data. It will only restart a