LA-FD: a Low-overhead Accrual Failure Detector

Xiao REN, Jian DONG∗, Hongwei LIU, Yang LI, Xiaozong YANG

School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China

Abstract

Failure detector is one of the fundamental components for building a distributed system with high availability. In order to maintain the efficiency and scalability of failure detection in a complicate large-scale distributed system, accrual failure detectors that can adapt to multiple applications have been studied extensively. In this paper, an accrual failure detector — LA-FD with low system overhead has been proposed specifically for current mobile network equipment on Internet whose processing power, memory space and power supply are all constrained. It does not rely on the probability distribution of message transmission time, or on the maintenance of history message window. By simple calculation, LA-FD provides adaptive failure detection service with high accuracy to multiple upper applications. The related experiments and results have also been presented.

Keywords: Failure Detection; Accrual Failure Detector; Adaptive

1 Introduction

Failure detector is one of the fundamental components for building a distributed system with high availability [1]. By providing the processes’ failure information in system, it supports the solution of many basic issues (such as consensus and atomic broadcasting, etc.) in an asynchronous system. Failure detection was proposed and formally defined by Chandra and Toueg [2] as an effective way to enhance the computational model of synchronous system. With the increasing demands on capability in distributed systems, failure detector has been widely applied to many fields including grid computing [3], cluster management [4] and peer-to-peer networks [5]. As a fundamental component, more and more requirements have been posed on the efficiency and scalability [6] of failure detector by the expanding system scale and increasingly complex distributed applications. How to achieve good detection speed and accuracy with low detection load has become a hot research topic in this field. Adaptive failure detector has been proposed as an important approach to solve this problem. It adjusts the detector’s configuration parameters automatically so that the system’s requirement on the indicator of effectiveness can be met with...
low load under different network environments. Chen [7] and Bertier [8] proposed a series of QoS-based adaptive detection algorithms based on probability network model. These algorithms have achieved adaptive adjustment on the quantitative control of detector parameters and greatly improved the detector’s control accuracy and effectively reduced detection load. However, with the development of various network applications, multiple applications are often running simultaneously in large-scale systems such as grid, P2P and cloud computing. They have different QoS requirements on failure detection. Taking into account the impact of load on scalability, we can’t afford configuring a separate failure detector for each application. Therefore, here comes another requirement on adaptive failure detector, that is, it can adapt to different QoS requirements demanded by multiple applications. This has become an important issue in the research of failure detection in large-scale distributed systems [6].

Hayashibara [9] has first launched the research in this area and proposed the concept of accrual detector. It allows a complete decoupling between monitoring and interpretation in traditional failure detection model. By outputting a continuous value associated with the status of a process rather than a binary value simply representing success or failure, upper applications can interpret detection results according to their own QoS requirements. Therefore, multiple applications can share the same detector and failure detection load can be effectively reduced in large-scale distributed systems. Currently many implementations of accrual detectors have been proposed and well applied to some well-known systems, such as Facebook [10]. However, with the development of applications in the Internet of Things and cloud computing, network access equipment has become diversified. Mobile terminals like cell phone and tablet PC have been used more widely. The majority of such equipment is embedded system whose processing power, memory space and power supply are all constrained. But the accrual detectors proposed previously require the probability distribution model for message transmission delay. For example, -detector uses normal distribution [11], Cassandra uses exponential distribution [10], and Benjamin uses gamma distribution [12]. Furthermore, those detectors need certain memory space to save a large history message window. At each detection cycle, a large amount of calculation is needed to compute probability distribution parameters and detector parameters. For most of mobile terminals, these system overheads about failure detection have important impact on system performance and battery consumption. And regarding failure detection itself, Gillen [13] has pointed out that the transmission delay caused by performance degradation would also have great impact on detection accuracy. Therefore, aiming at the mobile devices with constrained resource, we have proposed an accrual failure detector with low system overhead. It does not rely on the probability distribution of message transmission delay, or on the maintenance of history message window. Through simple calculation, it is able to provide adaptive failure detection service with high accuracy to multiple upper applications.

2 Algorithm Description

2.1 System model

We consider an asynchronous distributed system consisting of n processes, \( \prod = p_1, p_2, \cdots, p_n \). Each pair of processes is connected by a communication channel that can be used to send and receive messages. The type of failure is crash and channels are fair-lossy channels. No synchronized clock is assumed.
2.2 Basic failure detection strategy

Heartbeat is a common method to implement failure detectors. The detection modules detect each other’s status by sending heartbeat messages periodically at duration. According to different ways of implementation, there are two monitoring approaches: PUSH and PULL. For two processes $p$ and $q$ in system, where $q$ is monitoring $p$, the two basic approaches are described in the Fig. 1.

Both of the two approaches detect each other’s status by sending out heartbeat messages periodically at duration $\Delta t_i$. The difference is, in PUSH, the monitored process $p$ initatively sends a periodical message “I am alive” to process $q$, informing $q$ that $p$ is still alive; while in PULL, process $q$ sends a probing message “Are you alive?” to the monitored process $p$ periodically. After receiving the query message, the monitored process $p$ passively replies an “I am alive!” message to indicate its status. For traditional failure detectors based on timeout mechanism, an appropriate time-out value $\Delta t_0$ needs to be set. If no response message is received after $\Delta t_0$, the monitored process will be suspected as fails. Obviously, PULL approach needs twice the number of messages to get the same performance, but this does not affect its scalability (the increased multiple is a constant which is irrelevant to the number of processes). However, PULL is an initiative detection method which launches detection only when needed, and it does not need the assumption of a global synchronization clock. This is very important for current complicate large-scale distributed applications. Therefore, PULL employed as the basic detection strategy in this paper.

2.3 LA-FD failure detector

LA-FD employs PULL approach as the basic failure detection strategy. To simply the description, suppose the system is consisted of only two processes $p$ and $q$, where $q$ is monitoring $p$. The detection algorithm is shown in Fig. 2.

Fig. 2 shows that LA-FD failure detector is consisted of detection module and query module. The detection module located on process $q$ sends status probing message $mq_i$ to the monitored process $q$ at interval $t_i$. After receiving the probing message $mq_i$, process $p$ immediately replies an acknowledge message $ma_i$ to indicate its status. Everytime after receiving the acknowledge message $ma_i$, process $q$ needs to calculate the margin for the next detection message and records the transmission time of current detection message. When an upper application queries the detector, it will reply with a value of $q_p$. Then the upper application will set a threshold value $P$ according to its own requirement on detection accuracy. When $q_p > P$, process $p$ is suspected as failed.
3 Experiment Results and Analysis

In this section, we will analyze and compare the performance and overhead of LA-FD detector through experiments. In order to make the results more convincing, detection processes have been designed according to the configuration (ARM2440 processor, 400MHz clock speed, 512M RAM, 1200 mah battery capacity) of current mainstream mobile devices. Monitor process is located in Harbin and connected to Internet through WiFi. The monitored processes use the configuration described in section 2.3. There are two set of servers representing two typical network environments. One is located in Beijing (dataset 1) and the other is located in Pittsburgh (dataset 2). Experimental references are selected from several major implementations of accrual detectors such as Hayashibara’s $\phi$-failure detector ($\phi$-FD) [11], Benjamin’s new accrual detector (NAD) [14] and Avinash’s improved $\phi$-failure detector (I-$\phi$-FD) [10]. All experiments are focusing on two aspects of LA-FD: detection accuracy and system overhead.

3.1 Analysis of detection accuracy

The accuracy of accrual failure detector is usually affected by two main factors. One is detection delay, and lower detection delay will reduce the accuracy of detection results; the other is the threshold set by upper applications for the suspicion level $sl$, and higher threshold leads to higher accuracy. However, in comparative experiments, different implementations of accrual failure detector use different approaches to calculate $sl$ and its threshold. To help the comparison, for the same detection data, we have collected all the detection results and related data from different detectors under multiple sets of thresholds. The relationship between the average mistake rate ($\lambda_M$) and detection delay has been explored and the results are shown in Fig. 3.

It’s obviously shown in Fig. 3 that LA-FD has demonstrated higher detection accuracy. Under the same accuracy requirement, LA-FD has lower detection delay. I-$\phi$-FD based on exponential

---

Fig. 2: LA-FD failure detector

```
1. For process q:
2. detector_module:
3. for all $i>0$, at time $t_i$, send $mq_i$ to $p$;
4. upon receive $mq_i$ from $p$ do
5. if $i \geq sn_{current}$ then
6. margin $\leftarrow$ margin + $\alpha(|t_{current} - t_i \cdot \Delta t| - last_time)$;
7. last_time $\leftarrow t_{current} - i \cdot \Delta t$
8. $sn_{current} \leftarrow i + 1$
9. query_module:
10. $T_i \leftarrow |t_{current} - sn_{current} \cdot \Delta t|
11. if $T_i > 0$ then
12. $\rho_{sp} \leftarrow \exp\left(\frac{T_i}{last_time + margin}\right) - 1$
13. else $\rho_{sp} \leftarrow 0$
14.
15. For process $p$:
16. upon receive $mq_i$ from $q$ do
17. send $mq_i$ to $q$
```
distribution has the worst detection performance, so, the assumption of exponential distribution is only suitable for the specific P2P systems and normal distribution can describe the message transmission delay more accurately.

3.2 Comparison of system overhead

Since accrual failure detector needs to calculate detector parameters and maintain the history window of detection messages for each detection period, these two factors are the main reason for different system overhead in accrual detectors. A large history window will improve the prediction accuracy for model parameters and has certain impact on the calculation accuracy for detector parameters. Meanwhile, the maintenance of history window will cause more overhead. For the experiments in this section, we have selected different settings for historical window size and have made a detailed comparison of CPU utilization.

It can be seen from the Fig. 4 that the CPU overhead is the heaviest in $\varphi$-detector based on normal distribution and it grows the rapidest as the window size changes. This is because the workload is the most for calculating parameters of normal distribution model, and every time it needs the statistical data from the entire window. The overhead of LA-FD is the minimum, and
it isn’t affected by window size. Each process in the experiment shown in Fig. 4 only maintains 5 connections. In large-scale P2P systems, in order to maintain a high locating efficiency, each process is generally required to maintain \( \log N \) (\( N \) is the number of processes in the system) connections. Therefore, it is more significant in real system that LA-FD can reduce CPU overhead.

4 Conclusion

Accrual failure detector can adapt to the changes in network conditions and on this basis, it can satisfy the different QoS requirements on failure detection demanded from multiple applications. Accrual failure detector is a fundamental block to ensure the efficiency and scalability of applications in large-scale distributed system. Aiming at the characteristics that resource is constrained in mobile network equipment like cell phone and tablet PC, LA-FD has been proposed as an accrual failure detector of class \( \Diamond P_{ac} [9] \) in this paper. It does not need the probability distribution for message transmission time and the maintenance costs for message history window. LA-FD can provide adaptive detection service to multiple applications with very low overhead. Experimental analysis has shown that compared to several other implementations of accrual detectors, LA-FD has maintained high detection accuracy while effectively reducing system overhead and it has met the needs of major distributed applications.

References


