Title: ZooKeeper Failure Detector Model

Student: Abmar Grangeiro de Barros abmargb@gmail.com

Organization: Apache Software Foundation / ZooKeeper

Assigned mentor: Flavio Paiva Junqueira

Content:

The project goals

ZooKeeper servers detect the failures of the components they are interested in (others servers and clients) by using a regular heartbeat strategy as described in [1] and in [2], with a fixed timeout value. This method can be classified as an aggressive one, but it works fine for the majority of ZooKeeper installations. However, in some more unusual ZooKeeper installations, such as in a wide-area network, or in a mobile ad-hoc network, this method may not reach the expected QoS.

The goals of this project are to abstract the failure detector to a separate Java module, to implement several methods of failure detection and to compare and contrast their appropriateness for ZooKeeper.

According to [3], failure detection implementations may be classified between traditional and accrual ones. The traditional ones output a boolean value to the application, deciding themselves whether the monitored object has failed. Accrual methods output a value in a continuum scale to the application rather than information of a boolean nature, so it can decide about the monitored failure in different levels of confidence. In order to abstract the failure detection model it is mandatory to decide whether strategy – tradition or accrual – will be used, so it can be established an abstraction level.

Also, failure detection methods based in heartbeats may have a fixed or a variable timeout value. When the timeout value changes, the failure detection is classified as an adaptive one. The goal of adaptive failure detectors is to adapt to changing network conditions. In this project, I intend to implement and compare at least two of the adaptive methods (aside the fixed-timeout one), considering the ones described in [1], [2] and [3], and possibly others, suggested by the mentor/community. It is important to state that the Phi-accrual method can be easily converted in a traditional one by the addition of a threshold to the suspicion level that, if reached, determines that the monitored object has failed.

The last goal of this project, that is to evaluate the methods, can only be achieved by defining metrics. In this project, I am willing to use the set of primary metrics proposed in [1] to evaluate the quality of service (QoS) of the failure detectors that were implemented.

Roadmap

During the Community Bonding Period the type of failure detection we are using (traditional or accrual) is to be decided, and we also decide which are the failure detection methods I am going to implement.

Below are the steps I am planning to take during this project:

1. Study the chosen failure detection methods specification and the ZooKeeper code (24th May)

In this step I will make a further and critical reading on the articles that describe the failure detectors mechanisms and to write pseudocodes for each algorithm. Also, I will deeply study the ZooKeeper code regarding failure detection and will draw some diagrams (classes and sequence) to better understand how the mechanism currently works.

The threats for the success of this step live on the partial/mis understanding of the processes to be studied. The full comprehension of the mentioned subjects is essential for the rest of the project.

2. Isolate the failure detector model in the ZooKeeper code (14th June)

Based on the diagrams drawn in the first step, in this step I will isolate the code regarding failure detection to a separate Java module in the ZooKeeper code. This isolation will enable the implementation of new failure detectors in a quicker and more elegant way. In this step I will also re-implement the current failure detector method under this new architecture.

The difficulty of this step is based on how the failure detection code is scattered across the ZooKeeper code.

3. Implement the chosen failure detector methods (28th June)

In this step I will basically write the failure detectors algorithms pseudo-codes in Java under the architecture I proposed in the previous step. This development will be guided by unit tests, which should be written based on the methods' descriptions.

This step will check whether the proposed architecture is general enough to support the implementation of several failure detectors. In addition to this, the validation of the implemented failure detection methods will also be a challenge in this step.

4. Evaluate the QoS metrics for the implemented methods (26th July)

This step seems to be the most challenging one. I am planning to instrument the code using Aspect Oriented Programming[6], so I can forge error scenarios by simulating, for instance, message delivery delays. I also need to research about the properties of wide-area and adhoc networks to better adjust values for these simulations. After simulating the different methods for the same scenarios, they will be statistically compared according to their QoS metrics values, which are outputted from the simulations.

Achieve a good model for the failures and delays of a real network is the big challenge in this step. Although real workloads for the described scenarios are difficult to be found, their usage in the simulations should reduce this validity threat.

All artifacts generated are to be uploaded to the GSoC site.

Once I am done, there will be an extra effort to integrate the code written during this project to the ZooKeeper code base.

Progress

Progress of the project will be reported regularly at a blog[4].

Currently, I've briefly already studied the ZooKeeper code regarding the failure detector and have read some failure detectors methods descriptive articles for writing the proposal.

Who am I?

An MSc candidate on Computer Science at Universidade Federal de Campina Grande, 2 semesters left. I have five years experience with Java development. In 2006 I joined the Smart Pumping team – a decision support software based on field sensors for a brazilian major oil company (Petrobrás), and since late 2007 I am part of the OurGrid[5] team – an open source middleware for P2P grid computing.

My experience on failure detection in distributed system comes from the OurGrid middleware, in which every component must monitor the failure of the ones it communicates with. We have isolated the failure detection model and we have developed a failure detection implementation based on a pinging strategy with a fixed timeout value. I'm interested in participating in this project (aside from the funding and the learning) in order to aggregate the experience of joining another open source project to my career, and also to use the acquired knowledge in the OurGrid project.

References

[1] Wei Chen, Sam Toueg, Marcos Kawazoe Aguilera, "On the Quality of Service of Failure Detectors," IEEE Transactions on Computers, vol. 51, no. 1, pp. 13-32, Jan. 2002, doi:10.1109/12.980014

[2] Marin Bertier , Olivier Marin , Pierre Sens, Implementation and Performance Evaluation of an Adaptable Failure Detector, Proceedings of the 2002 International Conference on Dependable Systems and Networks, p.354-363, June 23-26, 2002

[3] Naohiro Hayashibara, Xavier Défago, Rami Yared, Takuya Katayama, "The Φ Accrual Failure Detector," srds, pp.66-78, 23rd IEEE International Symposium on Reliable Distributed Systems (SRDS'04), 2004

[4] http://abmargb.blogspot.com/

[5] http://www.ourgrid.org

[6] http://www.eclipse.org/aspectj/

Public Reviews

On 17th April 2010 18:37 Abmar Barros wrote:

Here are some issues discussed with the project mentor that may need clarification:

About the project goals

- It is important to consider that WAN and Internet characteristics such as Network delay, Packet loss, Packet corruption, Disconnections, Packet re-ordering and Jitter are the ones that could lead an aggressive failure detector to reach a higher error rate, and, consequently, to not meet the expected QoS.
- According to the project mentor, nowadays there is no use case for ZooKeeper in adhoc networks, so in this sense, I should focus the experiments in WAN and Internet scenarios.
- The primary metrics to be used in methods' evaluation are Detection Time, Mistake Recurrence Time and Mistake duration [1].

On the Community Bonding Period

- The type and the methods of the failure detectors to be implemented are to be discussed on the Apache Jira system (https://issues.apache.org/jira/) or on the zookeeper-dev list, depending on how concrete the discussion is. It is mandatory to have every discussion summarized and reported.

About the third step

- I will guide the development using the TDD (Test Driven Development) technique. First I will write tests according to the methods' description and then I will write the methods' implementation so they can pass in the tests.
- After defining the failure detector class and having its interface designed, it is mandatory to check whether the chosen failure detectors comply (syntactically and semantically) with the interface.

About the fourth step

- Besides the evaluation over the simulations, real tests are definitely necessary. We will arrange an infrastructure similar to the test scenarios in order to deploy and run the proposed tests.

About code integration

- Changes to the code are proposed on the corresponding Jira as diffs to the trunk code. Patches are discussed, and if applicable, eventually committed.

About progress report

- Besides reporting progress in my personal blog, I will report my progress on tools that are typically used by the project: discuss any changes in the corresponding Jira, create a wiki page on the hadoop wiki (http://wiki.apache.org) and keep updating such page during the project.

My previous experience on Failure Detection

- As I stated in the proposal, I have helped developing the failure detector for the OurGrid middleware. Although we can tune the heartbeats delay and the detection timeout, these values are statically defined. So in this sense, this failure detector follows a very aggressive model.
- We have had some issues regarding false failure suspicion while deploying OurGrid over the Internet and WAN. I think an adaptive failure detector should be a better pick in these scenarios. This previous experience also motivates me for this project -- I think ZooKeeper may face similar problems in similar scenarios.