GSoCFailureDetector

GSoC 2010: ZooKeeper Failure Detector Model

- Student: Abmar Barros (abmargb at gmail dot com)
- Assigned mentor: Flavio Paiva Junqueira

Abstract

ZooKeeper servers detect the failure of other servers and clients by counting the number of 'ticks' for which it doesn't get a heartbeat from other machines. This is the 'timeout' method and it works very well; however it is possible that it is too aggressive and not easily tuned for some more unusual ZooKeeper installations. This project's goals are to abstract the failure detector to a separate module, to implement several failure detectors and to compare their appropriateness for ZooKeeper.

The full gsoc-zookeeper-failuredetector.pdf is attached to this page.

Roadmap

1. Discuss the project with the community (dev/user lists), asking for suggestions and requirements and decide which type and which methods are to be implemented (Community Bonding Period)
2. Study the chosen failure detection methods specification and the ZooKeeper code (24th May)
3. Isolate the failure detector model in the ZooKeeper code (14th June)
4. Implement the chosen failure detector methods (28th June)
5. Evaluate the QoS metrics for the implemented methods (26th July)

Failure detection methods' references

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

Specific objectives

The ones with strike-through have already been finished

1. ~~Write pseudo-codes for the proposed algorithms~~
2. ~~Create FailureDetector interface~~
3. ~~Write implementations and tests of the FailureDetector interface based on the proposed algorithms~~
4. ~~Refactor client-side code of the client-server monitoring to use the proposed FailureDetector interface~~
5. ~~Make the failure detection and its parameters configurable on the client~~
6. ~~Refactor server-side code of the client-server monitoring to use the proposed FailureDetector interface~~
7. ~~Refactor the code of the server-server monitoring to use the proposed FailureDetector interface~~
8. ~~Make the failure detection and its parameters configurable on the server (to server-server and client-server monitoring)~~
9. ~~Evaluate the QoS metrics with experimentation~~
10. ~~Write Forrest docs~~

Related JIRA

- https://issues.apache.org/jira/browse/ZOOKEEPER-702
- https://issues.apache.org/jira/browse/ZOOKEEPER-811
- https://issues.apache.org/jira/browse/ZOOKEEPER-810
- https://issues.apache.org/jira/browse/ZOOKEEPER-812

Progress Report

Community bonding period

- Studied ZooKeeper code regarding failure detection
- Studied the proposed failure detection algorithms

05/Jun/10

- Discussed whether FD instance should run in the same thread of the application
- Proposed first version of the FailureDetector interface. It does not consider application messages as heartbeats.
- Adapted ClientCnxn class to use the proposed interface.

11/Jun/10

- Written pseudo-codes for the proposed failure detection algorithms
- Started discussion on how could the application scheduling interval could interfere on adaptive FD methods.

22/Jun/10

- Attached the classes of the initially proposed FD methods (Phi Accrual, Chen, Bertier, Fixed Heartbeat) and the corresponding unit tests.
- Included suggestions Flavio gave concerning package naming and method scope.

28/Jun/10

- Started discussion on how to configure the FD method and its parameters at the client and server sides.

02/Jul/10

- Enhanced pseudocodes documentation
- Created appMessageReceived() and appMessageSent() methods. These methods allow the Failure Detector to use application messages as heartbeats, which reflects the ZooKeeper case.
- Added command line options to the client side in order to configure failure detector method and its parameters.
- Unit tests expanded to comply with new methods.
- Enhanced javadocs for each Failure Detector implementation

08/Jul/10

- Adapted failure detectors to work on both client and server sides of the client-server monitoring.
- Refactored server-side code to use the FailureDetector interface.
- Created a new FailureDetector, which groups monitoreds by their tick time, similar to what the SessionTracking does.
- Made the server-client failure detector (and its parameters) options of the ZooKeeper configuration file.

23/Jul/10

- Made Learners track client heartbeats and send statistical data of the heartbeat sampling window to the Learner
- Analyzed paper about application message used as heartbeat.

30/Jul/10

- Refactored the code of the Leader-Learner monitoring in order to use failure detector module
- Added options in configuration file to configure Leader-Learner monitoring
- Fixed bugs in FDs initialization and in some cases of session registering
- Fixed bug in server recovery on the client side

09/Aug/10

- Set up first experiments in Emulab
- Instrumented code to report some detection metrics as false suspicion and detection time
- Enhanced java documentation, started to write forrest docs

14/Aug/10

- Kept on changing experimentation scenarios to check the behaviour of the failure detectors
- Tuned some initialization parameters of the failure detectors based on some experimentation results
- Finished writing forrest documentation
- Added a new parameter to phi accrual FD - the minimum window size to begin timeout adaptation.
- Added a moderation step to Bertier FD, to comply with second level task described in his paper
- Made Chen's alpha parameter configurable, and not a quarter of the timeout

16/Aug/10

- Refactored the way default values are passed to failure detectors
- Finished experimentation and written experiment report

Experimentation

Experimental design

- **First batch of tests**:
  - 1 client and 1 server connected by an transcontinental link (Campina Grande-Brazil / Newark-USA)
  - client sending async operations to server
  - client running during 10 min (average)
  - link = 1MBps, 250ms
  - timeout = 5000ms
  - replication = 5

    - used the following failure detectors:
        - Fixed heartbeat
        - Chen (alpha = 0, 1000)
        - Bertier (moderationstep = 0, 1000)
        - Phi accrual (threshold = .5, 8; minwindowsize=50)

- **Second batch of tests**:
    - 200 clients and 1 server connected in an emulated WAN in emulab
    - client sending async operations to server
    - clients running during 10 min (average)
    - link = 2MBps, 250ms, message loss probability of 0.1
    - timeout = 5000ms
    - used the following failure detectors with fixed parameters:
        - Fixed heartbeat
        - Chen (alpha = 1250)
        - Bertier (moderationstep = 1000)
        - Phi-accrual (threshold = 2)

Results

- **First batch of tests**:

Method	Average detection time	Stddev of the detection time	False suspicions
Fixedhb	4731.8	299.6985	0/5
Chen (alpha=0)			5/5
Chen (alpha=1000)	1810.8	347.3632	0/5
Bertier (moderation step = 0)	784.6	483.5642	0/5
Bertier (moderation step = 1000)	1228.2	804.5773	0/5
Phi accrual (threshold = 0.5)	714.6667	521.9745	2/5
Phi accrual(threshold = 8.)	1574.75	602.7799	1/5

- **Second batch of tests**:
    -
        - In these tests, Fixed heartbeat and Bertier's strategies did not present any false suspicion. With the given alpha, Chen's presented 13/200 false suspicions, and the Phi-accrual, with the windowminsize parameter equals to 0, have made false suspicion on all the clients. Below, we show the average detection time of all methods but the Phi-accrual: `http://www2.lsd.ufcg.edu.br/~abmar/zk/fd-comparison.png`
        - The Phi-accrual method must be evaluated again with a better windowminsize parameter in a scenario with a greater duration, so the warm-up period is not considered.

Concluding remarks

As expected, we noticed that the fixed heartbeat method works well when we run ZooKeeper in a controlled environment, where the network behavior is expected. In this cases we can tune the fixed timeout after some network analysis. However, in scenarios where we have a changing network behavior, such in a WAN, the adaptive methods can be a good pick. Below, there is an overview of each failure detector:

- **Fixed heartbeat**: In average, with default parameters, the fixed heartbeat strategy had the highest detection time, but with no false suspicion. However, if the timeout is not well defined, failures may take a long time to be detected, or false suspicion rate would be increased. As said before, this strategy is useful when there is a controlled environment, in which the network can be characterized.
- **Chen**: This strategy requires some assumption over the network, once the administrator needs to define the alpha parameter - the safety margin for the estimation. However, with default parameters, Chen et al. method performed well in a WAN deploy. It managed to decrease the average detection time with a low false suspicion rate.
- **Bertier**: Bertier et al initially proposed a failure detector that requires no assumption over the network but a single moderation step to be added to the estimation when the monitored is at a suspected state when a heartbeat is received. With these experiments, we have come to same conclusion as Hayashibara et al: that this failure detector is very sensitive to message loss and fluctuation in the arrival times of heartbeats. In this sense, the moderation step turned out to be an important parameter for this failure detector. With a moderation step of 1000, Bertier's failure detector reached a higher average detection time than the Chen's method, but lower than the fixed hearbeat strategy. It is worth to mention that Bertier's failure detector was primarily designed to be used over local area networks (LANs), that is, environments wherein messages are seldom lost. As we could see, with a single client Berties's method stands out with a low detection time and no false suspicions, even with the moderation step equals to 0.
- **Phi-accrual**: The phi-accrual is the method that requires less information about the network behavior. However it relies on a large sampling window to perform a good estimation. As we could see, in the experiments that a minimum window size was not used, there was a huge number of false suspicions. The effect of the threshold is only noticeable when there is some deviation from the average. The phi-accrual stands out in a WAN with unknown behavior, but it is mandatory to set a good (high) initial timeout value for the warm-up period of the method, which happens while the minimum window size is not reached.

Design decisions

Should a failure detector instance (FD) run in a separate thread from the application?

- Drawbacks
    - There will be another thread competing for CPU, and its inclusion should add some overhead.

    - Concurrency issues must be handled.
- Benefits
    - The FD will run in a more independent way, and it will notify the application of changes using listeners or callbacks. If it runs in the same thread, the application must signal the failure detector of changes, and also retrieve its status periodically. In other words, the application code will be coupled to FD code.
- Decided to use the FD on the same thread of the application

How to use application message in adaptive failure detectors?

- Decided to just delay the estimated arrival time if the next message, and to not use this message in the timeout adaptation.

Due to the usage of application messages as heartbeat, the actual heartbeats are not sent regularly. How to compute the next estimated arrival time?

- Decided to use interarrival heartbeat times. When a application message is received, the time of the last heartbeat received in shifted.

How to report sampling window statistical data from Learners to Leader?

- Decided to do heartbeat tracking on the Learners, and then mean and standard deviation of the interarrival heartbeat times is reported to Leader. A new method in the FailureDetector interface was created to comply with this requirement.

Future work

- Update C client to use the Failure Detector module. This may require to have all failure detectors implemented in C. https://issues.apache.org/jira/browse/ZOOKEEPER-848
- Analyze the overhead of the timeout computation on adaptive FDs.
- Contrast adaptive FDs behaviour with sampling window full and in a warm-up period (when sampling window is not full).
- Extend experimentation in order to cover other scenarios, such as different number of nodes, experiment duration, infrastructure (link characteristics) and failure detection parameters.