A Failure Detection System for Large Scale Distributed Systems

Andrei Lavinia, University Politehnica of Bucharest, Romania
Ciprian Dobre, University Politehnica of Bucharest, Romania
Florin Pop, University Politehnica of Bucharest, Romania
Valentin Cristea, University Politehnica of Bucharest, Romania

ABSTRACT

Failure detection is a fundamental building block for ensuring fault tolerance in large scale distributed systems. It is also a difficult problem. Resources under heavy loads can be mistaken as being failed. The failure of a network link can be detected by the lack of a response, but this also occurs when a computational resource fails. Although progress has been made, no existing approach provides a system that covers all essential aspects related to a distributed environment. This paper presents a failure detection system based on adaptive, decentralized failure detectors. The system is developed as an independent substrate, working asynchronously and independent of the application flow. It uses a hierarchical protocol, creating a clustering mechanism that ensures a dynamic configuration and traffic optimization. It also uses a gossip strategy for failure detection at local levels to minimize detection time and remove wrong suspicions. Results show that the system scales with the number of monitored resources, while still considering the QoS requirements of both applications and resources.

Keywords: Clustering Mechanisms, Failure Detection, Fault Tolerance, Gossip Strategies, Large Scale Distributed Systems

INTRODUCTION

Large scale distributed systems (LSDS) are hardly ever “perfect”. Due to their complexity, it is extremely difficult to produce flawless designed distributed systems. While until recently the research in the distributed systems domain has mainly targeted the development of functional infrastructures, today new requirements have emerged among which fault tolerance is needed by more and more modern distributed applications, not only by the critical ones. The clients expect them to work despite possible faults occurring.

Although the importance of fault tolerance is today widely recognized and many research projects have been initiated in this domain, the existing systems often offer only partial solutions that follow a particular underlying distributed architecture. Traditional fault detection solutions, in particular, fail to work properly in the context of LSDS because of the large

DOI: 10.4018/jdst.2011070105
number of monitored processes involved, the high probability of message loss, the dynamic nature of the underlying topologies and the unpredictable delays in message delivery.

In this paper we present implementation details for a failure detector designed for highly dynamic LSDS. Based on the architecture previously proposed in (Dobre et al., 2009), it combines adaptive and accrual detection approaches with a hierarchical design, for scalability and performance, and uses gossip protocols for a more accurate detection. The detector is specifically designed to meet requirements of a reliable failure detector. Its architecture allows applications to specify different QoS detection levels, while offering scalability, generality and non-intrusive characteristics.

Failure detection consists in monitoring processes and failure detectors throughout the systems and detecting errors in the shortest time according to the fault tolerance requirements of the distributed applications. The interpretation of the monitoring information is based on the concept of suspicion level. The progressive detection feature brings a new approach in terms of predicting the time of arrival of the next heartbeat message, as well as interpreting and updating the suspicion level. The system is able to deal with both transitional and permanent errors.

The implementation of the system is based on two strategies. The first one uses a hierarchical approach based on dynamic clustering to solve the scalability issue. The second strategy leverages the gossiping technique in order to remove wrong suspicions and decrease the time needed to detect errors.

The solution proposed in this paper combines the advantages of existing approaches in order to minimize their limitations so as to effectively treat problems such as message explosion, scalability, flexibility, dynamism message delays and adjustment to variable network conditions and various fault tolerance requirements coming from applications.

The rest of this paper is structured as follows. First we present related failure detection strategies for distributed systems. The next section describes the proposed architecture, highlights key elements of the implementation of the failure detector, and details on the models and protocols being used. Experimental results demonstrating the validity and performance of the proposed solution are discussed next. Finally, we lay out the conclusions and future work.

RELATED WORK

Fault tolerance in LSDS is based on one form of a failure detection system. Such a failure detector is generally capable of running a detection algorithm and it can communicate with other services which it monitors. This model was first proposed, in the form of an “oracle” detection service, by Chandra and Toueg (1996).

The failure detection module is independent to the main application flow and is being responsible with the monitoring of a subset of the processes within the monitored system and maintaining a list of those it currently suspects to have crashed. A process can query its local failure detector module at any time to check its status. The list of suspected processes is permanently updated such that, at any time, new processes can be added and old ones removed. This failure detector is considered unreliable because it is allowed to make mistakes up to a certain degree. Therefore a module might erroneously suspect some correct process (wrong suspicion) or fail to detect processes that have already crashed. At any given time two failure detector modules may have different lists of processes.

The most common implementation of local failure detection is based on the heartbeat strategy. In this approach a failure detector module periodically sends a heartbeat message to other modules to inform them that it is still alive. When a module fails to receive a heartbeat from another module for a predetermined period of time (timeout) it concludes the remote process had crashed. There is a tradeoff, however, for the timeout values being considered. Short timeouts ensure quick detection, but with a high prob-
Implementation of a DES Environment
www.igi-global.com/chapter/implementation-des-environment/38266?camid=4v1a