DADSTORM
A simple, fault-tolerant and real-time stream processing system

DAD 2016-2017
Group 12:
Daniel Fermoselle n^o 78207
João Marçal n^o 78471
Tiago Rodrigues n^o 78692

Abstract

DADSTORM is a simple but reliable stream processing system. It's mainly used by Instituto Superior Tecnico Students.
The main features are: the 3 possible semantics of tuple processing it can have, fault-tolerance to f faults per operator with a synchronous model of detection, 3 modes of tuple routing and last but the not the least 5 types of operators. This system is composed by a Puppet Master, Process Creation Service, Operators with their Replicas, Tuples and ThreadPools.

1 Introduction

Nowadays with the rising interest on big data streaming information is getting more and more important and we want to process that data as fast as possible even though existing the possibility of having faults. In order to get that information in a reliable way we developed DADSTORM. This system is reliable through the passive replication we use, allowing us to get tolerance to f faults using just f+1 replicas.
Our system process tuples based on the type of operator which can be UNIQ, COUNT, DUP, FILTER and CUSTOM. With these operators we can get the data processed in a way that let us get the information we want.

2 Programming Model

In this section, we provide a high-level overview of the programming model, highlighting the key concepts.
In DADSTORM data is represented as string Tuples, i.e, a collection of strings.
To begin with we have a puppet master who will start all the operators on all the machines with the help of the process creation service this last will already be located in all the machines that will run operator replicas.
The puppet master has an intuitive interface inside which we have a box where we can introduce a path to a configuration file to start all the operator's replicas as well has their inputs, routing, operator spec and address.
The tuples are stored inside a file that will be accessed by the first operator of the stream. The operators are composed by repicas which can be located inside different machines.
All the replicas might fail but we assure f fault tolerance to silent failures for each operator. We assume that none of the first operator replicas can fail and that at least there is one replica alive per operator to guarantee that we have at least f+1 replicas for each operator, being f the number of fautls of each operator.
A quick overall. So first of all we start the puppet master which will through the configuration file introduced and with the help of the process creation service create all the operator's replicas. Then the replicas of the first operator will read the .dat file which has the begining tuples to be processed. After this initialization all the replicas will receive tuples and process them, sending the result to the next operators, unless it is a replica of the last operator this one don't send anything just processes tuples.
To finish this section the user of our system doesn't have to worry about how the system is distributing the operations, the user just has to create the configuration file and let the puppet master read then just execute and all the data will be processed, getting in the end the desired information.

3 DADSTORM Abstractions

In this section we will talk a little bit about the abstractions we created for this system, begining with the tuple and all its variations, having an overview about the operator, the replicas. To finish the section we will talk about the parser, the RepInfo and the ConfigInfo.

3.1 Tuple

Tuple, this is one of the most important abstractions of our system if not the most one. The Tuple is a possible infinite collection of strings, it also has a unique id to be used when the tuple processing semantics requires so. The tuples are processed and when this happen it is created another tuple having as content the result of processing the previous tuple. To help mostly the processing semantics we created also 3 more variations of the tuple, which are: AckTuple, TimerTuple and Tuple2TupleProcessed. The AckTuple is no more than a tuple associated with the url of the replica who sent the tuple this url can be used in the future. The TimerTuple is a tuple associated with a timer of the limit the tuple has to be acked otherwise it'll be resent. Last but not the list there is the Tuple2TupleProcessed this abstraction contains 2 tuples, the first tuple when processed by some replica generates the second tuple, this association can be useful first to compare if the tuple was already processed and then to return the result of processing the first tuple if necessary.

3.2 Operators and Replicas

3.3 Parser

3.4 RepInfo

3.5 ConfigInfo

4 Architecture and Implementation

4.1 Tuple

4.2 Operators and Replicas

4.3 Puppet Master

4.4 Process Creation Service

5 Discussion

6 Production Experiences

7 Evaluation

8 Conclusion