Byzantine Fault Tolerance in YARN

Josh Fuerst
Purdue University
fuerst@purdue.edu

Rachna Goyal
Purdue University
goyal15@purdue.edu

Josh Reese
Purdue University
reese5@purdue.edu

Derek Schatzlein
Purdue University
dschatzleinop@gmail.com

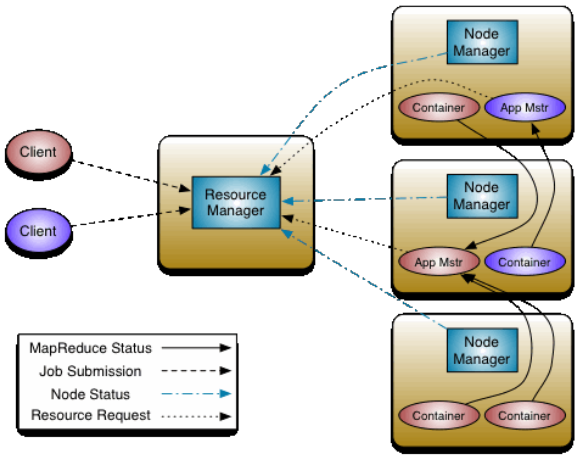

Figure 1: YARN Architecture

ABSTRACT

For this project we attempted to implement Byzantine Fault Tolerance (BFT) within YARN. The goal was to provide anyone using the YARN framework with the ability to have resources and tasks automatically duplicated. Our task focused specifically on the protion of BFT which deals with inconsistent output.

1. INTRODUCTION

Need to introduce some stuff here... bla bla bla fuck fuck fuck...

2. RELATED WORK

Talk about Mesos, YARN, Spark, Byzantine Fault Tolerance

3. SYSTEM DESIGN

Our system design began with just YARN in mind. Our hopes were to analyse the system and implement BFT entirely inside of YARN. Our hopes were to keep the YARN changes just within the API such that any future frameworks which use the same API's could utilize our changes. We also hoped that if our changes were within the standard API current YARN frameworks could have BFT without any changes. We later discovered that many current frameworks do not use YARN's API to handle the entire process. Spark is once such example. Spark uses the standard API to request containers. Spark however, starts its own separate scheduler which bypasses many of the communications needed to implement BFT.

3.1 YARN

We used the provided examples within YARN to determine how the system functions. As it turns out YARN provides two simple classes AMRMClient and NMClient. AMRMClient handles the communication between the ApplicationMaster(AM) and the ResourceManager(RM). This is how the AM sends requests to the RM for new containers. It is also how the RM reports back when the new containers are allocated. The NMClient class is what handles the communication between the NodeManager(NM) and the AM. This is where the communication is sent to launch processes in the containers.

Our system design creates a new class Byzantine. We altered the NMClient and AMRMClient classes to each create an instance of our Byzantine class. The overall idea is that whenever there is a communication from AMRMClient or NMClient we intercept that communication. This means when the AM requests a container we can intercept that resource request and replicated it. This means if we are in byantine mode, for each intercepted resource request we duplicate it so we have 4 total requests. As this is happening we save off the request in a table. As the containers are allocated we intercept the communication from the RM back to the AM. We save each replica containers information in the table with its respective resource request. This is done by matching the requested resources with those available in the container. We do not deliver the replica communication to the AM. Once all of the container replicas have been allocated we forward only one allocated container status back to the AM. We do this to hide all of the Byzantine replication from the AM. This means that existing AMs can run with our version of YARN with no changes.

At this point the AM has received communication that one container has been allocated. This is the point where the AM will begin to form a ContainerLaunchContex. This encompasses the information needed to run a task on the container. This includes a path to the .jar file which will be run on the container. Once the launch context is built the AM uses the NMClient API to send communication to the launched container which tells it to start. At this point we intercept the communication and tell all of the containers replicas to perform the same actions. We can do this because we have a table which maps each container to all of its replicas.

Method	Park Cost	User involvement	Accuracy	Automated	User Visibility
Landmarks	None	Passive	Low	No	At Gate
Sensor Systems	High	Passive	High	Yes	At Gate/Online
Active Mobile Apps	None	Active	High/Low	Yes	Mobile Device (online)
Our Application	None	Passive	High/Low	Yes	Mobile Device (online)

Table 1: Comparison of Queue Tracking Systems

At this point YARN will start running the provided task on the container. Upon completion the NM will report a completion status to the RM who forwards the status back to the AM via the AMRMClient API. In a similar fashion we intercept these communications. Again we hide the replicas completions from the AM. Once we have seen the completion status from all the replicas we must verify that they all have the same output. To do this we simply compare the output of each container. The containers output file name must be defined by a call to our Byzantine class. At this point if we have a successful Byzantine run we report that back to the AM. If not we report a Byzantine failure back to the AM. In the event of a Byzantine failure we assume it is up to the AM to decide what to do next.

3.2 Spark

4. EVALUATION

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam id mauris adipiscing, pellentesque ligula sit amet, luctus leo. Duis congue nisl metus, id pretium nulla auctor vel. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed mauris nisl, tincidunt nec purus sagittis, condimentum aliquet eros. Pellentesque eu dolor lectus. Nunc gravida, ipsum at pretium mollis, neque est porta massa, sed iaculis odio tellus vitae est. Nunc accumsan interdum condimentum. Fusce sit amet neque ut erat commodo molestie id ac est.

Nunc sagittis lacus mattis, lobortis nibh ac, varius justo. Aliquam vestibulum enim et molestie lobortis. Praesent magna nulla, auctor egestas dictum quis, condimentum quis elit. Pellentesque iaculis bibendum pellentesque. Donec id semper sem. Nulla lacinia in leo fringilla laoreet. Integer in purus non nunc dapibus sagittis. Nunc dapibus neque nec ligula pretium, et rhoncus sapien pulvinar. Phasellus a enim rhoncus, ultricies tortor ut, vulputate enim. Aliquam odio orci, pharetra tempus ante at, mattis vehicula justo. Donec blandit ligula felis, placerat fermentum nibh mollis sit amet. Nunc sed ultrices leo. Cras ut justo sed lacus accumsan condimentum sit amet in justo. Suspendisse dui diam, venenatis a nisi at, sollicitudin gravida nisi.

5. LIMITATIONS AND FUTURE WORK

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam id mauris adipiscing, pellentesque ligula sit amet, luctus leo. Duis congue nisl metus, id pretium nulla auctor vel. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed mauris nisl, tincidunt nec purus sagittis, condimentum aliquet eros. Pellentesque eu dolor lectus. Nunc gravida, ipsum at pretium mollis, neque est porta massa, sed iaculis odio tellus vitae est. Nunc accumsan interdum condimentum. Fusce sit amet neque ut erat commodo molestie id ac est.

Nunc sagittis lacus mattis, lobortis nibh ac, varius justo. Aliquam vestibulum enim et molestie lobortis. Praesent magna nulla, auctor egestas dictum quis, condimentum quis elit. Pellentesque iaculis bibendum pellentesque. Donec id semper sem. Nulla lacinia in leo fringilla laoreet. Integer in purus non nunc dapibus sagittis. Nunc dapibus neque nec ligula pretium, et rhoncus sapien pulvinar. Phasellus a enim rhoncus, ultricies tortor ut, vulputate enim. Aliquam odio orci, pharetra tempus ante at, mattis vehicula justo. Donec blandit ligula felis, placerat fermentum nibh mollis sit amet. Nunc sed ultrices leo. Cras ut justo sed lacus accumsan condimentum sit amet in justo. Suspendisse dui diam, venenatis a nisi at, sollicitudin gravida nisi.

6. CONCLUSIONS

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam id mauris adipiscing, pellentesque ligula sit amet, luctus leo. Duis congue nisl metus, id pretium nulla auctor vel. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed mauris nisl, tincidunt nec purus sagittis, condimentum aliquet eros. Pellentesque eu dolor lectus. Nunc gravida, ipsum at pretium mollis, neque est porta massa, sed iaculis odio tellus vitae est. Nunc accumsan interdum condimentum. Fusce sit amet neque ut erat commodo molestie id ac est.

Nunc sagittis lacus mattis, lobortis nibh ac, varius justo. Aliquam vestibulum enim et molestie lobortis. Praesent magna nulla, auctor egestas dictum quis, condimentum quis elit. Pellentesque iaculis bibendum pellentesque. Donec id semper sem. Nulla lacinia in leo fringilla laoreet. Integer in purus non nunc dapibus sagittis. Nunc dapibus neque nec ligula pretium, et rhoncus sapien pulvinar. Phasellus a enim rhoncus, ultricies tortor ut, vulputate enim. Aliquam odio orci, pharetra tempus ante at, mattis vehicula justo. Donec blandit ligula felis, placerat fermentum nibh mollis sit amet. Nunc sed ultrices leo. Cras ut justo sed lacus accumsan condimentum sit amet in justo. Suspendisse dui diam, venenatis a nisi at, sollicitudin gravida nisi.

7. REFERENCES