Improving Twister Messaging System Using Apache Avro

Yuduo Zhou Yuan Luo Patanachai Tangchaisin

School of Informatics and Computing
Indiana University, Bloomington IN
{yuduo, yuanluo, ptangcha}@indiana.edu

Abstract

Twister[1], an iterative MapReduce framework developed by SALSA group in Pervasive Technology Institute of Indiana University takes MapReduce[2] paradigm a level further by catering applications that are iterative in nature. Twister uses publish/subscribe messaging system by using NaradaBrokering[3]. The problem triggered by this is the messaging experience delays when exchanging relatively large data messages in contrast to smaller control messages. This inspires us to implement another approach, the Remote Procedure Calls (RPC), for messaging to increase performance. We chose Apache Avro[4] as our RPC framework.

Project Goal

The goal of the project is to investigate and implement a new approach for Twister messaging system. Our approach is to replacing current Twister's broker system with Remote Procedure Calls and Serialization System.

Apache Avro[4] is a serialization and remote procedure call framework. It provides rich data structures and a compact, fast, binary data format. Avro relies on *schemas* for data serialization and de-serialization. It provides functionality similar to systems such as Thrift, Protocol Buffers, etc., but in a more compact and fast way.

By replacing the current NaradaBrokering System with Apache Avro, all computation nodes are enabled to send and receive data directly from each other, instead of going through the broker system. Therefore the communication overhead and the bottleneck between the publish/subscribe broker system and computation nodes is eliminated. We expect to see a better performance in many aspects from our implementation including speed, scalability, and communication efficiency.

Architecture

In Figure 1, Apache Avro is used for communication between Twister Driver – Twister Daemon and Twister Daemon – Twister Daemon. Instead of sending or receiving a message through a brokering system, each node can communicate with each other directly via RPC and serialization. Since Twister Driver currently maintains a list of nodes, it is not hard to locate all Twister Daemons and it can also encapsulate its address inside a message to let daemons send a response. The Twister Driver is also responsible for node identification in peer-to-peer communication and execution control.

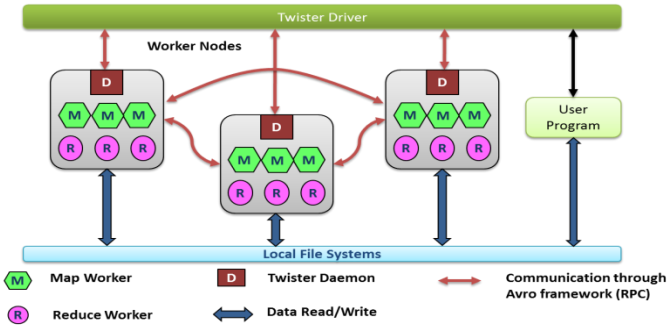

Figure 1 Twister architecture after integrated with Apache Avro

Evaluation Method

We will evaluate our implement in two aspects which are speed and scalability.

For the makespan, we will run existing Twister's programs such as K-means, BFS, matrix multiplication, etc. with our implementation. Then compare the result with original Twister and other MapReduce technology.

For scalability tests, first method is to run our program on a large scale page rank algorithm to investigate how our implementation will do on a data-intensive application. Second method is to verify that our program can scale up by making speed up chart when running it on a large number of machines.

Early Stage Result

In Figure 2, we present the result from running our Twister-Avro prototype version on a single machine with 4 workers. Our test machine is Intel Core 2 Quad CPU with 8 GB memory. We run K-means algorithm on different data sizes using our implement and original twister version. As we expect, removing brokering system can reduce communication overhead and improve system performance.

Data Size	Twister-Avro	Twister
40k	1.723 seconds	3.281 seconds
160k	2.096 seconds	3.807 seconds
640k	3.466 seconds	4.658 seconds
2560k	6.377 seconds	8.147 seconds

Figure 2 Time used for running K-means algorithm on various data size

Future Work

Since the prototype version of Twister-Avro shows a better performance than brokering system on a single machine. The next step is to evaluate our system with different algorithms and environments. Moreover, we will alter the current serialization implementation to Apache Avro binary encoder serialization, which is tested to have a faster transmission speed [6].

Conclusion

We implemented a new messaging mechanism using Apache Avro RPC for Twister, the iterative MapReduce Framework. Early stage testing proves that the performance and scalability can be significantly improved via the new system. As further development is ongoing, we hope more improvement can be achieved to make Twister more reliable and scalable.

We propose a poster that describes the Twister-Avro architecture and its capabilities. The poster will include an architecture diagram similar to that shown in Figure 1, protocol and message definition using Avro schemas, and evaluation of our system comparing to original Twister in various algorithms similar to that shown in Figure 2. In addition, we will include its performance chart when running in different number of machines.

References

[1] Jaliya Ekanayake, Hui Li, Bingjing Zhang, Thilina Gunarathne, Seung-Hee Bae, Judy Qiu, Geoffrey Fox. Twister: A Runtime for Iterative MapReduce, Proceedings of the First International Workshop on MapReduce and its Applications of ACM HPDC 2010 conference, Chicago, Illinois, June 20-25, 2010

[2] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: simplified data processing on large clusters. *Commun. ACM* 51, 1 (January 2008), 107-113

[3] Narada Brokering System, http://www.naradabrokering.org

[4] Apache Avro, http://avro.apache.org

[5] Shrideep Pallickara, Marlon Pierce, Harshawardhan Gadgil, Geoffrey Fox, Yan Yan, Yi Huang. A Framework for Secure End-to-End Delivery of Messages in Publish/Subscribe Systems. Proceedings of the 7th IEEE/ACM International Conference on Grid Computing (GRID 2006). Barcelona, Spain.

[6] Apache Avro in practice. http://blog.voidsearch.com/bigdata/apache-avro-in-practice