Netty vs ZeroMQ in Realtime Analytics

SUNANDA UNNI[1,*]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[*] *Corresponding authors:suunni@indiana.edu HID: S17-IO-3022*

project-000, April 12, 2017

Performance of realtime analytics on Big Data depends on efficiency of inter and intra process communication. There are many messaging frameworks supported by HPC-ABDS like Netty, ZeroMQ, Kyro, ActiveMQ, LMAX disruptor. Performance of messaging framework is measured by throughput of the messaging framework. We are comparing the throughput of Netty with ZeroMQ based on research done by IBM and Yahoo.

Keywords: I524, Netty, Storm, Distributed Computing System, Real time Analytics

https://github.com/cloudmesh/sp17-i524/raw/master/paper1/S17-IO-3022/report.pdf

1. INTRODUCTION

Big Data upsurge has seen a better performance requirement from Data Processing Pipelines. Another goal for the organization is to get timely results for the data crunching. Structured and unstructured data generated by heterogeneous sources are collected and processed in batches or in realtime. To improve upon the processing time distributed computational cluster set ups are used. Distributed computational model requires a lot of communication between the processes running on different nodes. There are many messaging frameworks available in HPC-ABDS for realtime analytics and specifically for interprocess communication namely ZeroMQ, Kyro. We are comparing the data of experiments done at Yahoo and IBM for throughput using the different messaging frameworks.

Netty [1] is an asynchronous event-driven network application framework for rapid development of maintainable high performance protocol servers and clients. Netty book "Netty in Action" [2] mentions Netty to have a performance superior to standard Java NIO API thanks to optimized resource management, pooling and reuse and low memory copying.

2. DESIGN OF A DISTRIBUTED REALTIME COMPUTATIONAL SYSTEM

In a distributed cluster worker processes, executors and tasks makes a running topology required for data processing and analytics. A reliable messaging framework is crucial for the inter process, intra process and inter topology communication.

2.1. Worker Process Illustration

An illustration of worker process is as shown in Fig. 1.

2.2. Running Topology

A running topology as illustrated in Fig. 2.

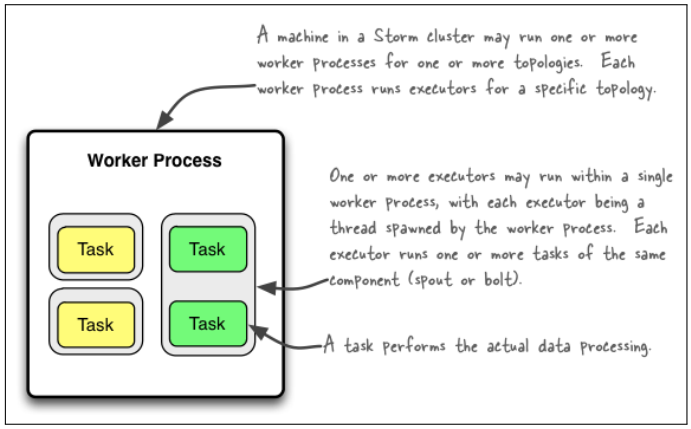

Fig. 1. The relationships of worker processes, executors (threads) and tasks in Storm, adopted from [3]

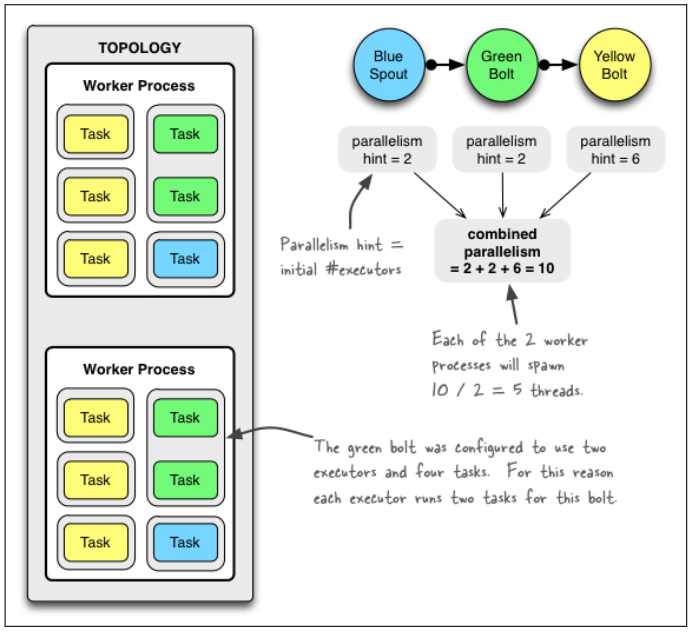

Fig. 2. Example of a Running Topology, adopted from [3]

Each of the above worker process is running on the same or different node in the distributed cluster. Worker process(executor or task) communicates with other Worker processes.

Apache Storm is a free and open source distributed realtime computation system. Storm makes it easy to reliably process unbounded streams of data, doing for realtime processing what Hadoop did for batch processing [4].

As per [5], Users are free to stitch together a directed graph of execution, with Spouts (data sources) and Bolts (operators). Architecturally, it consists of a central job and node management entity dubbed the Nimbus node, and a set of per-node managers called Supervisors. The Nimbus node is in charge of work distribution, job orchestration, communication, fault-tolerance, and state management (for which it relies on ZooKeeper).

The parallelism of a Topology can be controlled at 3 different levels: number of workers (cluster wide processes), executors (number of threads per worker), and tasks (number of bolts/spouts executed per thread)

Communication is handled at different levels in Storm using-

- Intra-worker communication in Storm (inter-thread on the same Storm node): using LMAX Disruptor
- Inter-worker communication (node-to-node across the network): using ZeroMQ or Netty
- Inter-topology communication: nothing built into Storm, you must take care of this yourself with e.g. a messaging system such as Kafka/RabbitMQ, a database, etc.

For inter-worker communication Storm uses ZeroMQ by default in older versions and can use Netty in Storm versions later than 0.9, as the network messaging backend.

3. PERFORMANCE - YAHOO ENGINEERING EXPERIMENT

Yahoo experimented in 2013 for using a pluggable messaging layer for Inter-worker communication using Netty [6]. A simple speed of light test was run to benchmark and measure performance of messaging layer. This can be done by pushing messages between different Spouts and Bolts.

3.1. Setup and Configuration of Cluster

In the cluster setup 38 nodes, 3 nodes were dedicated to zookeeper, 1 to nimbus and the UI, and 34 for supervisor and logviewer processes. Each node had 2 x 2.4GHz xenon processors each with 4 cores and hyperthreading enabled for a total of 16 threads of execution.

3.2. Netty Behavior with small Topology

The Netty behavior with Topology of 3 workers running on 3 nodes. In the below Fig 3, the Y-axis represents number of messages sent and X-axis represent the size of messages. A 100% increase in number of messages is seen with Netty as compared to zeromq.

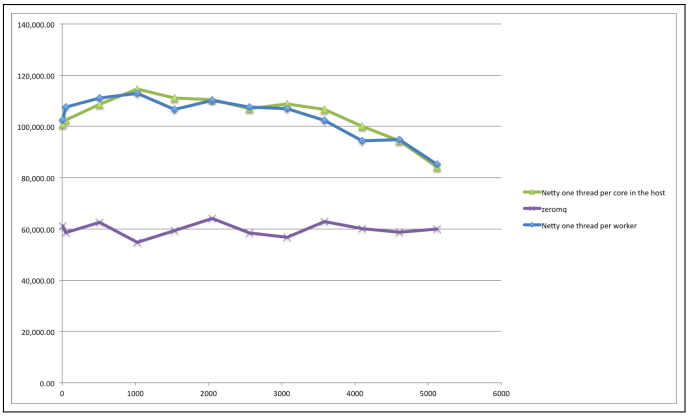

Fig. 3. Small Topology- Netty vs ZeroMQ, adopted from [6]

3.3. Netty Behavior with large Topology

The Netty behavior with Topology of 100 workers, 100 spouts, 500 bolts spread out in 5 levels of 100 bolts each, and 100 ackers is shown in Fig 4.

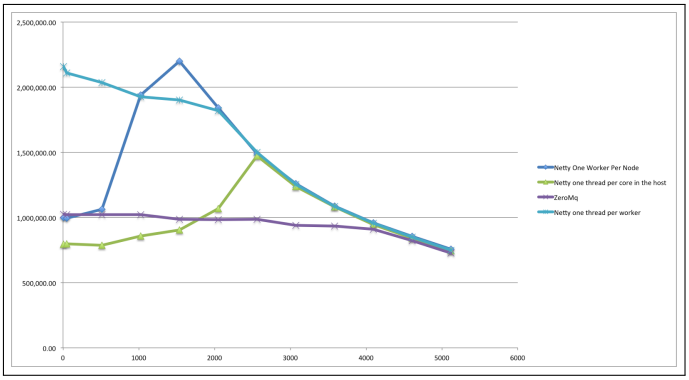

Fig. 4. Large Topology- Netty vs ZeroMQ, adopted from [6]

An increase is seen in number of messages as message size is increased however above behavior is due to less context switching as the message size increases. Post a peak at 2.5KB message size, network saturation occurs and we see a slope down where Netty performance matches ZeroMQ at 4KB message size.

To resolve and better Netty performance we need to-

- Reduce number of workers so single worker is running per node
- Make the number of threads configurable to default 1, which matches ZeroMQ default.

4. PERFORMANCE - IBM EXPERIMENT

Another verification of the numbers stated by Yahoo was done in IBM Whitepaper "Of streams and storms" [5].

In [5],the performance benchmarking is carried out by processing an Enron email dataset of half a million emails through data processing pipeline. The processing pipeline used IBM Infosphere Stream versus Apache Storm.

4.1. Setup and Configuration of Cluster

The cluster here consists of 6-nodes, 1 nimbus node, 1 zookeeper, 4 supervisory nodes. Each node with 3GHz dual-core Xeon cluster (HS21), 2 CPUs per node, 8GB RAM, dual 146GB drives, RHEL 6.2. It is mentioned in the paper that Storm with Netty is around 1.65 times faster (in terms of throughput) than Storm with ZeroMQ for a multinode setup and "Restricted Benchmark" which consists of a simple 3-stage pipeline: source, count, and sink on a 4-node configuration.

4.2. Report of Throughput

The below table shows the throughput number improvement for Storm with Netty(Storm 0.8.2) vs Storm with ZeroMQ(Storm 0.9.0.1) in Fig. 5.

Table 4: Storm 0.9 Results

System	Nodes	Dataset	Parallelism	Elapsed time	Throughput (emails/s)	CPU time
Storm 0.8.2	1	100%	4	21m 29s	1,270	1h 24m 5s
Storm 0.9.0.1				27m 28s	988.04	1h 28m 2s
Streams				3m 13s	8,438	0h 9m 48s
Storm 0.8.2	4	100%	4	5m 14s	5,213	1h 04m 34s
Storm 0.9.0.1				4m 43s	5,757	1h 04m 56s
Streams				1m 43s	31,637	0h 10m 12s
Storm 0.8.2	4	100%	8	9m 34s	5,659	2h 28m 56s
Storm 0.9.0.1				8m 58s	6,053	2h 18m 56s
Streams				1m 43s	31,637	0h 20m 47s

Fig. 5. Storm Results 0.9 vs 0.8.2, adopted from [p. 27 5]

5. CONCLUSION

Netty can be used as a pluggable Messaging framework for Distributed Computation System for Realtime Analytics for performance improvement however there is restriction of currently using it only for inter process communication. Netty is a pure Java solution which has a higher memory footprint of the application in comparison to ZeroMQ hence tuning is required for JVM heap size.

REFERENCES

[1] "Netty site," Web Page. [Online]. Available: http://netty.io/
[2] N. Maurer and M. A. Wolfthal, *Netty in Action*, 1st ed. Greenwich, CT, USA: Manning Publications Co., 2015. [Online]. Available: http://www.ebook.de/de/product/21687528/norman_maurer_netty_in_action.html
[3] M. Noll, "Understanding the parallelism of a storm topology," Web Page, Oct. 2012. [Online]. Available: http://www.michael-noll.com/blog/2012/10/16/understanding-the-parallelism-of-a-storm-topology/
[4] Apache Software Foundation, "Apache storm," Web Page, 2015. [Online]. Available: http://storm.apache.org/index.html
[5] Z. Nabi, E. Bouillet, A. Bainbridge, and C. Thomas, "Of streams and storms," *IBM White Paper*, pp. 1–31, Apr. 2014. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.695.5752&rep=rep1&type=pdf
[6] B. Evans, "Making storm fly with netty," Web Page, Oct. 2013. [Online]. Available: https://yahooeng.tumblr.com/post/64758709722/making-storm-fly-with-netty