An Experience Report on Scalable Implementation of DDoS Attack Detection

Sri Yogesh Dorbala, Kishore. R and Neminath Hubballi

Discipline of Computer Science and Engineering
Indian Institute of Technology Indore
{cs110112, cs110118,neminat}@iiti.ac.in

Abstract. Distributed Denial of Service (DDoS) attacks are increasingly becoming powerful and crippling many networks and services in Internet. Many methods have been proposed to mitigate and detect DDoS attacks in the literature. These techniques require processing large amount of network traffic in real time. In order to process this bulky network traffic, in this paper we report an experimental investigation of scalable implementation. In our experiments we used distributed computing framework of Apache Hadoop to achieve the scalability. We implemented clustering and classification algorithms for detecting DDoS attack. Several experiments on a DDoS dataset and normal dataset of sizes ranging from 1 GB to 80 GB resulted in a performance improvement.

1 Introduction

Denial of Service (DoS) is a type of attack that aims to deny a legitimate user access to resource(s) for a prolonged time. These attacks have a huge impact depending on their intensity. In recent years these attacks have increased many fold and have crippled vital internet infrastructure. Attackers might have different intentions behind carrying out these attacks like financial gain and social protests.

To increase the intensity of DoS attacks, attackers use multiple sources with some of them across geographies. These multiple sources increase the complexity of attack detection. These attacks are known as **Distributed Denial of Service** (DDoS) attacks. The attacker initially takes control of many computers on internet by infecting those systems using different techniques. These infected systems are called botnets. Botnets are most commonly used sources for performing a DDoS attack. Bot master (usually attacker) sends commands to the botnets (slave machines) to launch attack against a particular target at a particular time.

According to a recent Verisign report on DDoS Trends [1], there is a 291% increase in attacks in the first quarter of 2014 when compared to that of 2013. This is also shown in Figure 1. This report also mention that the largest detected DDoS attack was generating traffic at the rate of 300GBps against Spamhaus website. Another instance of a major DDoS attack is on Root DNS Servers. Root DNS Servers do the translation from domain name to IP Address. In 2007, 2 of 13 root DNS servers suffered badly but the impact of the attack was very little thanks to the advanced measures like anycast and load balancing. If attacks on root DNS servers are successful, then access to many websites is forbidden as domain name to IP Address translation fails. If reliable and sufficient defenses are not in place to defend these attacks, the services and infrastructure under these attacks will be inaccessible [2].

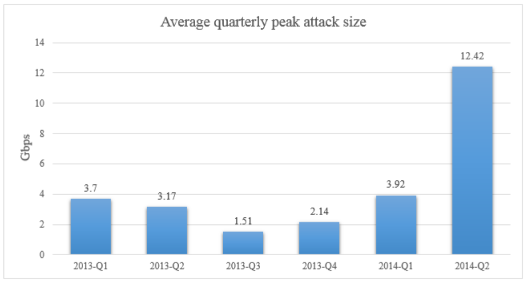

Fig. 1. DDoS Trends in 2014 [3]

Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks have been addressed by researchers in two main categories as preventive methods and detection methods. These techniques can work by monitoring traffic at the intermediate router level or individual host level. Techniques like Ingress filtering[4] and Egress filtering [5], MANAnet's Reverse Firewall [6] which is a reverse firewall limiting the rate of outbound traffic flow, excluding the acknowledgments (Ack's) to be sent to recently received packets, Unicast Reverse Path Forwarding [7] which discard IP packets that have been received on a different interface than the one used for sending a packet to that source and Traceroute work at the router level, while methods like Hop Count filtering [8], making initial sequence number of a TCP session a truly random number or a parameter of source and destination addresses [9] are deployed at the end host.

One of the challenge in detecting DDoS attacks is to collect and analyze large volume of network traffic in real time. This raises the question of scalability of detection methods. Recently few attempts are made to address this [10,11] issue. In this paper, we explore the scalability of DDoS detection techniques on Hadoop with two traffic pattern learning methods. First one is an unsupervised learning technique and second is a supervised learning method. In unsupervised learning method, only normal traffic is modeled and in the supervised learning method both normal and DDoS traffic is modeled. In both the cases a set of features extracted from the network traffic serve as feature vectors.

Rest of this paper is organized as follows. In section 2 related work for detecting and mitigating DDoS attacks is elaborated. In section 3 a very brief overview of Hadoop which is a distributed computing framework is given. In section 4 two methods for DDoS detection namely unsupervised and supervised learning techniques are discussed. In section 5 experiments done to evaluate the performance of the two methods is described. In section 6 a discussion on lessons learnt are briefed. Finally, section 7 concludes the paper.

2 Literature Survey

There are number of methods proposed in literature to detect and mitigate DDoS attacks. Mainly these methods are of following types.

2.1 Methods deployed at source of attack

D-WARD [12] is a source based preventive measure for DDoS attack. Here both inbound and outbound traffic is monitored and compared with already computed models of normal traffic flows. If the current flows show huge difference with previously computed models they are identified as contributors to attack.

MUlti-Level Tree for Online Packet Statistics (MULTOPS) and Tabulated Online Packet Statistics (TOPS) MULTOPS [13] is a heuristic method to detect DDoS attacks. It uses a tree data-structure to capture rate statistics for various subnets in Internet. MULTOPS makes an assumption that at any time, traffic flow between two devices is proportional. Using this, if a flow between two hosts or subnets is not proportional then one of the host or subnet is either a source or a destination of an attack. The main disadvantage with this method is use of dynamic tree data structure. Attacker can leverage this to exhaust the memory resources at monitoring machine.

Proactive Surge Protection [14] is a filtering technique where packets originating from a source which is injecting more number of packets than previously seen are discarded by routers assuming they are contributing to DDoS attack. Backward Traffic Throttling [15] is a similar approach where traffic is prioritized based on historical rates observed rather than discarding.

Source-based methods make a poor choice in mitigating DDoS attacks because of following 3 reasons.

1. The attacker may employ different techniques to distribute the sources of attacks across geographies, making it impossible to know if one particular source is an attacker.
2. As the DDoS traffic is aggregated at the victim's end, the volume of traffic monitored/collected at a source might not be sufficient to draw any tangible conclusions.
3. Although source based methods are effective in mitigating DDoS attacks, they require cooperation from Internet Service Providers (ISPs). However these ISPs require huge investments and maintenance costs and in return find no incentive in implementing these methods.

2.2 Methods deployed at the destination of attack

These methods are implemented at the receiver end or victim of an attack sometimes called as destination. These methods have to be more sophisticated as they usually take into account all the traffic from multiple sources converging in on this destination unlike Source-based methods, where we have only one source.

Packet Marking [16,17,18,19]: In this method routers in the path from source to destination of a packet add their identity to the packet. This enables us to find the path which the packet has taken to reach destination. If there are any significant changes in the path compared to historic values those packets are suspected. One of the main challenge of this method is storing the entire path in the packet, as packet might take a longer path in which case whole path cannot be stored.

History-based IP filtering [20,21]: In this method statistics for various IP addresses is maintained. These statistics include rate of packets exchanged with the receiver. Any significant changes in these rates are detected as attack. This method can narrow down the access list to those IP addresses which were collected at an earlier time when there was no attack. A serious disadvantage of this method is some of the normal traffic is also denied access if those IP addresses were not seen earlier.

Hop-count filtering [22,8,23]: In this method, the source IP address and corresponding hop-count of every arriving packet is logged. Subsequently, incoming packets with those source IP addresses are checked for their hop-count. Significant deviations in the originally logged and new hop-count is detected as suspicious and packets are identified as spoofed. Since spoofed packets are commonly used in DDoS attacks, a significant percentage of spoofed packets are detected as DDoS attacks.

3 Hadoop

Apache Hadoop [24] is an open-source framework for distributed storage and processing of very large files. Hadoop cluster is a network of interconnected systems with Hadoop installed. Commodity computers can be configured to create a Hadoop cluster to run jobs in parallel.

Hadoop File System (HDFS) and MapReduce [25] are main components of Hadoop Framework. HDFS is the file storage system of Hadoop. Hadoop enables us to store data in multiple locations in the cluster by splitting files into blocks and also replicating file-blocks for data redundancy in case of system (node) failure. This has another advantage of data locality for computation, which plays a big role in distributed computing. MapReduce allows us to run jobs in parallel on different nodes in the cluster. This distributed computing achieves scale in computation.

4 Proposed Method

In this section we describe our methods for detecting DDoS attack. Here we use two learning techniques namely supervised and unsupervised learning techniques for detecting DDoS attacks. Learning is based on interval summary of network traffic. Each interval summary has the information of amount of network traffic flowing in and out of network. Both the techniques are based on interval summary calculated on per second basis. Each interval summary has the information as shown in Table 1.

Table 1. Interval Summary

1	Number of Packets
2	Average Packet Length
3	Number of TCP packets
4	Number of UDP Packets
5	Number of ICMP Packets
6	Distinct Source IP Addresses Seen
7	Distinct Source ports
8	Distinct Destination IP Addresses Seen
9	Distinct Destination ports

An example interval summary is shown below.

Interval[t_1, t_2]: 173, 597.24, 171, 2, 0, 6, 6, 12, 12

This interval summary constitute the input feature vectors for both supervised and unsupervised learning methods. As we can notice, the summary calculates few statistics from network traffic. A sample of traffic statistics of DARPA 99 and CAIDA 2007 dataset is shown in Table 2. indicating the difference between normal and DDoS traffic . We can notice that the difference is clearly visible.

Table 2. Comparison between Normal and DDoS Traffic

Properties	Normal (1 minute)	DDoS (1 minute)
Distinct Source IP's	22	7519
Distinct Destination IP's	26	7522
Distinct Source Ports	826	65490
Distinct Destination Ports	670	234

4.1 Unsupervised Technique

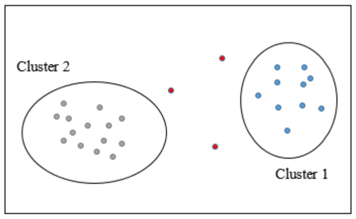

Fig. 2. Data Clustering and Outliers

Unsupervised learning is a method where hidden structure in the data is found from unlabeled dataset. This method summarizes the data and explains the essential characteristics of the dataset. As the data is unlabeled, there is no feedback provided to correct mistakes in learning. Data clustering is one such unsupervised learning method which we have used for DDoS attack detection. In this method DDoS attacks are detected as cases of outliers. Figure 2 shows a view of outliers in the dataset. In the figure there are two clusters and few points shown in red color constitute outliers.

Cluster is a group of objects with similar characteristics and Clustering is the method of forming these clusters from the set of unlabeled data. Any data object (vector) which does not fall into the clusters is termed as an outlier. Our approach is to form clusters of normal dataset and later detect whether a given input data object is within the cluster or an outlier, i.e., a normal one or DDoS attack instance.

I) *Training*: The first stage of this method requires the formation of clusters. Hence in training phase normal instances are grouped into clusters. Clusters are represented by a cluster center and each cluster has a radius.

II) *Detection:* In the detection phase a given input instance is to be detected as an outlier or not. In order to do this, calculated distance of the given input vector with respect to the center of the cluster is used. Those instances whose distances are within the radius of cluster are considered normal and those whose distance is outside the radius of the cluster is termed as outlier or DDoS attack instance.

In order to form clusters and find the distance of any input vector with cluster center a distance measure is used. In this case Manhattan distance given in equation 1 is used as a distance measure.

$$d(x,y) = \Sigma_{k=1}^{n} |x_k - y_k| \tag{1}$$

4.2 Supervised Technique

Supervised learning is a method where labeled training data is used to create a model in order to classify future data. The algorithm has to correctly label the unknown data in testing dataset by comparing it to the model created from the training dataset. Just like previous case this method also has a training phase and testing phase.

I) *Training:* We use k-nearest neighbors classification algorithm, to classify testing data into normal or DDoS traffic, with the use of labeled training data. Since K-nearest neighbor classification requires identifying closest neighbors of a testing vector, training phase is trivial which is the training dataset itself.

II) *Testing:* In this phase, each test vector's K nearest neighbors are found and a majority voting is done to label it. In order to calculate the nearest neighbors Manhattan distance metric is used. K closest neighbors in the training dataset are found for every interval in Testing dataset based on a distance measure. Distance of each testing data object (vector) to each training data object is computed using the Manhattan distance measure as in previous case.

5 Experiment Details

In our experiments we used two datasets for evaluation. First dataset is DARPA 99 week 1 and 3, outside traffic which is used as normal dataset and second one is CAIDA 2007 DDoS attack dataset. CAIDA 2007 dataset is DDoS traffic of an hour collected from a real attack in internet. A summary of these two datasets is shown in Table 3. From these datasets we extracted packet header details using CLI version of Wireshark called tshark. This is a text file containing the header details of all packets. Text file of packet header details of DARPA 99 datset is around 1 GB where as similar file of the DDoS dataset is around 22GB. We can notice that, there are 405325 intervals of one second each in the DARPA 99 dataset while CAIDA 2007 dataset has 2455 intervals. A java program is written to process this text file to generate interval summary per second. Although, DDoS dataset contains packets of only one hour network traffic it has many more packets compared to 5 days of normal traffic in DARPA 99 dataset.

Table 3. Dataset Description

Properties	**Normal**	**DDoS**
Number of Packets	1,39,41,518	37,11,32,316
Number of Intervals	4,05,325	2,455

5.1 Hadoop Configuration

We setup a Hadoop cluster with 4 nodes to study the scalability of network traffic processing. We used open source Apache Hadoop version 2.4.1 in our setup. Other configuration details of this cluster are below.
Hadoop Cluster Configuration:
Number of PC's: 4
Processor: Intel Core 2 Duo E8400 @ 3.00 GHz
RAM : 8 GB (Each node)
HDD : 500 GB (Each node)
Network : 100 Mbps DLink Switch connected using 1 Gbps Ethernet Cable

5.2 Performance of Interval Summary

In order to assess the performance of Hadoop in processing these text files into interval summaries we conducted an experiment. In this experiment we used files of different sizes containing packet header details as input and generated interval summaries of one second each. Table 4 shows a comparison of processing time of a 4 node Hadoop cluster with a single node machine. We can notice that, the time taken for processing 1 GB file in a single node machine is faster in comparison to multinode Hadoop cluster. A similar analysis on a file of 20 GB (this is a

DDoS dataset) brought down the difference between the two significantly with single node system still being the better of the two. We ran another experiment with a file size of 40 GB. This 40 GB file was generated by replicating DDoS dataset packets. In this case performance of Hadoop is found to be better than the single node machine. A similar experiment with 80 GB (generated similarly) data shows Hadoop's superiority over a single node machine. We can conclude from this experiment that Hadoop performance increases with increase in input dataset size.

Table 4. Processing Time

	File Size			
	1 GB	**20 GB**	**40 GB**	**80 GB**
Single Node Machine	30 sec	7 min	25 min	52 min
Multi-Node Hadoop	90 sec	13 min	23 min	45 min

5.3 Unsupervised Learning

As discussed previously interval summaries are used as data vectors to detect DDoS attacks. In unsupervised learning method we used interval summaries of only normal data (for training) from DARPA 99 datset. We experimented by creating different number of clusters from this dataset and the performance is reported here. The dataset details used for unsupervised learning is shown in Table 5. In the first experiment we created only one cluster by taking the average of all vectors in the training set.

Table 5. Dataset Used for Unsupervised Learning

Properties	**Normal**		**DDoS**	
	Training	**Testing**	**Training**	**Testing**
Number of Intervals	2,02,662	2,02,663	0	2,455

Table 6 shows the performance of unsupervised learning method. Out of the 202663 intervals in the normal dataset 50 of them are incorrectly labeled as DDoS instances and the rest all are correctly labeled as normal instances. Likewise, 45 of the total 2455 intervals of the DDoS were wrongly labeled as normal intervals. We use *Recall* and *Precision* as the metrics to evaluate the performance of the system. These two are given in equation 2 and equation 3 respectively. There is a *Recall* of 98.17% and *Precision* of 97.97%. in this method.

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

Table 6. Confusion Matrix - Complete

	Normal	**DDoS**
Normal	2,02,663	50
DDoS	45	2,410

In the second case we created one cluster for every protocol[1]. We report results for each of these protocols. Table 7. shows performance of TCP interval summaries. We used only TCP packets to generate interval summaries. Out of 337,650 intervals created, 50% of the data i.e, 1,68,825 are used for training and remaining 50% are used for testing. As we can see from the table 30 of them are incorrectly labeled as DDoS intervals. We can notice that 99.98% of normal instances are correctly labeled. Similarly, 56 of the total 2432 TCP intervals are misinterpreted as normal intervals. *Recall* and *Precision* of this experiment are 97.69% and 98.75% respectively.

Table 7. Confusion Matrix- TCP

	Normal	**DDoS**
Normal	1,68,825	30
DDoS	56	2,376

Table 8. shows the ICMP data vector detection performance. A cluster is generated by using 50% of the 8059 ICMP intervals in the normal dataset. Out of 4029 normal testing instances all are correctly labeled as normal intervals thus giving a 100% correct labeling. *Recall* and *Precision* in this case are 99.95% and 100%.

Table 8. Confusion Matrix - ICMP

	Normal	**DDoS**
Normal	4,029	0
DDoS	1	2,454

5.4 Supervised Learning

As described earlier in supervised learning both normal and DDoS datset is used for training a classification algorithm. We used K-nearest neighbor classification

[1] Except UDP, whose number is very small in CAIDA 2007 datset

algorithm for supervised learning. Since K-nearest classification is computationally very expensive we used a smaller representative dataset for this experiment. Table 9 shows dataset details. We have divided the entire data set into two sets as training and testing. Training phase had 1200 normal intervals from DARPA 99 and another 1200 DDoS intervals from CAIDA. The testing data set had 175,117 normal intervals and 1255 DDoS intervals.

Table 9. Dataset Description

Properties	Normal		DDoS	
	Training	Testing	Training	Testing
Number of Intervals	1,200	1,75,117	1,200	1,255

With *K=5* we found nearest neighbors for each of the testing interval and labeled it with majority vote i.e., if 3 of its neighbors are labeled normal in training dataset the testing interval is also labeled as normal and otherwise. Table 10 shows the classification performance of this method. Table 11 shows the time taken to classify the testing dataset into normal or DDoS traffic. As we can see the time taken for performing the classification on a single node machine is slower than Hadoop cluster. We can notice that *Recall* is 100% and *Precision* is 99.52%.

Table 10. Confusion Matrix - Supervised

	Normal	DDoS
Normal	1,75,111	6
DDoS	0	1,255

Table 11. Classification Time

	Time (minutes)
Single Node	30
Hadoop	25

6 Discussion

As we noticed from above results although Hadoop resulted in performance improvement in comparison to a single node processing, the improvement is not substantial. There are several reasons for this including cluster setup and configuration. Following are few insights gained from our experiments.

1. **Data Size:** For MapReduce to perform well, input data size should be larger. On smaller datasets the overhead in performing Map and Reduce operations will be the major contributors to the processing time. We could not increase the dataset size further because of hardware constraints.

2. **Suitability of Algorithms for Parallel Processing:** Suitability of algorithm for parallel computation also contributes to the performance improvements. In our case the unsupervised learning method is not computationally involved where as K-nearest neighbor classification is not inherently parallelizable.
3. **Network Latency:** To use input data in sizes of few Giga Bytes, we need a very high speed network to transfer the portion of input files between nodes of cluster. Due to this network latency in transferring the file parts to slave nodes the total computation time is increased. In our experiments we used a low end 8 port switch which happened to be a bottleneck.
4. **Caching:** Hadoop has the facility to cache frequently accessed files to reduce the file-read time. By caching the data, a local copy of file is kept on all slave nodes. This reduces the time required for file access as compared to accessing the file from Master before the Map phase. We observed that, this feature reduces the network latency to a considerable extent and improve the performance in comparison with experiments without caching.

7 Conclusion

In this paper we described an experimental evaluation of DDoS attack detection with traffic interval summaries. We conducted scalability study of two detection methods namely unsupervised and supervised learning methods on Hadoop and compared its performance with a single node machine. An important observation is that although Hadoop improves performance there are several factors which affects its performance including size of data, suitability of algorithm for parallel computation, network latency, etc. In order to achieve the scale all of these need to be addressed.

References

1. Verisign, "http://www.verisigninc.com/assets/report-ddos-trends-q22014.pdf."
2. M. Geva, A. Herzberg, and Y. Gev, "Bandwidth distributed denial of service: Attacks and defenses," *IEEE/ACM Transaction on Networking*, vol. 12, no. 1, pp. 54 – 61, 2014.
3. Verisign, "Ddos attack stats: http://www.stateoftheinternet.com/downloads/pdfs/resources-web-security-2014-q2-global-ddos-attack-report-infographic.pdf," 2014.
4. P. Ferguson and D. Senie, "(rfc 2827) Network ingress filtering: Defeating denial of service attacks which employ ip source address spoofing," 2000.
5. D. Distler, "Performing egress filtering- SANS Institute Infosec Reading Room - 2008."
6. MANANET, "The reverse firewall: Defeating ddos attacks emanating from a local area network."
7. CISCO, "http://www.cisco.com/web/about/security/intelligence/unicast-rpf.html."

8. H. Wang, C. Jin, and K. G. Shin, “Defense against spoofed IP traffic using hop-count filtering,” *IEEE/ACM Transactions on Networking*, vol. 15, no. 1, pp. 40–53, 2007.
9. F. Gont and S. Bellovin, “(rfc 6528) Defending against sequence number attacks,” 2012.
10. Y. Lee, W. Kang, and Y. Lee, “A hadoop-based packet trace processing tool,” in *Proceedings of the Third International Conference on Traffic Monitoring and Analysis*, pp. 51–63, 2011.
11. Y. Lee and Y. Lee, “Toward scalable internet traffic measurement and analysis with hadoop,” *SIGCOMM Computer Communication Review*, vol. 43, no. 1, pp. 5–13, 2012.
12. J. Mirkovic and P. Reiher, “D-ward: A source-end defense against flooding denial-of-service attacks,” *IEEE Transactions on Dependable Secure Computing*, vol. 2, no. 3, pp. 216–232, 2005.
13. T. M. Gil and M. Poletto, “MULTOPS: a datastructure for bandwidth attack detection,” in *In Proceedings of 10th Usenix Security Symposium*, pp. 23–38, 2001.
14. J. Chou, B. Lin, S. Sen, and O. Spatscheck, “Proactive surge protection: A defense mechanism for bandwidth-based attacks,” *IEEE/ACM Transaction on Networking*, vol. 17, no. 6, pp. 1711–1723, 2009.
15. Y. Gev, M. Geva, and A. Herzberg, “Backward traffic throttling to mitigate bandwidth floods,” in *In Proceedings of Global Communication Conference (GLOBECOME 12)*, pp. 904–910, 2012.
16. S. Savage, D. Wetherall, A. Karlin, and T. Anderson, “Practical network support for ip traceback,” in *SIGCOMM ’00: Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pp. 295–306, 2000.
17. T. Peng, C. Leckie, and K. Ramamohanarao, “Adjusted probabilistic packet marking for ip traceback,” in *Networking ’02: Proceedings of the Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications*, pp. 697—-708, 2002.
18. A. Belenky and N. Ansari, “IP traceback with deterministic packet marking,” *IEEE Communication Letters*, vol. 7, no. 4, pp. 162—-165,, 2003.
19. V. Paruchuri, A. Durresi, and S. Chellappan, “TTL based packet marking for ip traceback,” in *GLOBECOM ’08: Proceedings of the GLOBCOM Conference*, pp. 1–5, 2008.
20. M. Goldstein, C. Lampert, M. Reif, A. Stahl, and T. Breuel, “Bayes optimal ddos mitigation by adaptive history-based ip filtering,” in *Proceedings of the Seventh International Conference on Networking*, pp. 174–179, 2008.
21. F. Yi, S. Yu, W. Zhou, J. Hai, and A. Bonti, “Source-based filtering scheme against ddos attacks,” *International Journal of Database Theory and Application*, vol. 1, no. 1, pp. 9 – 20, 2008.
22. C. Jin, H. Wang, and K. G. Shin, “Hop-count filtering: An effective defense against spoofed ddos traffic,” in *CCS ’03: Proceedings of the 10th ACM Conference on Computer and Communications Security*, pp. 30–41, 2003.
23. U. Weinsberg, Y. Shavitt, and Y. Schwartz, “Stability and symmetry of internet routing,” in *IEEE International Conference on Computer Communications Workshops*, pp. 407–408, 2010.
24. Apache, “http://hadoop.apache.org/.”
25. J. Dean and S. Ghemawat, “Mapreduce: Simplified data processing on large clusters,” in *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, pp. 137–149, 2004.