Improvising Scatter-Gather Protocol for Efficient Distribution of Workload among Fog Nodes.

Aditya Shaha[1], Pratik Kejriwal[2], V Shashank[3]

Abstract

With the advent of technological advancement not only the amount of computation performed matters but the time and speed of such computation matters as well. Minimizing the computation time has become one of the primary interests with the increasing workload as no one wants to wait to for some processing to take place as evident by the need of refreshing and checking our internet connection when such processing takes longer than expected. As distributing the work relieves the individuals, same can be applied to the machines where instead of having one single computer do all the processing a group of computers are employed to complete them in less time and with more efficiency. And with the advent of cloud computing, sharing computer processes and resources over the internet has grown rapidly. However, it may happen that distributing the work among so many computers can actually increase the time delay as the data has to travel to and from the cloud with addition to the distribution and actual processing. So in this paper we put forward the advantages of using the Fog Computing over the cloud at the same time we try to improve on Scatter-Gather algorithm so that we can optimize the time taken to decompose the workload and then distribute the task to a geologically placed nearby fog node, so that the workload gets distributed on the nodes that are in the vicinity so that the time latency can be minimized.

Keywords

Scatter Gather, Workload Distribution, Fog Computing, Efficient distribution, Greedy paradigm, Dijkstra's algorithm

[1] *15BCE0227, SCOPE, VIT University*
[2] *15BCE0226, SCOPE, VIT University*
[3] *15BCE0042, SCOPE, VIT University*

Introduction

There has been an unprecedented increase in the amount of data that is being generated and thus processing this data has become exponentially difficult. To reduce the time latency and increase the user experience a new model known as Fog Computing is employed, which can operate on the network edges and hence it removes the need for sending the data to the cloud for analysis as the work can be done in the networking node itself, which comparatively is way faster than sending data to the cloud for analysis and then waiting for the results.

In Fog Computing however, a number of nodes or edges are geologically distributed which handle the incoming request from the user and processes it to give the result. Though with the use of Fog computing the user experience is improved still if the workload is huge, the computation time would still be significantly large and the level of user experience expected would not achieved. As computing in a fog node also requires a lot of time as by sending the data to fog node, only the time to send and receive the data is reduced but the processing time remains unchanged. In order to undercut the time taken for processing the only possible alternate is

to apply Divide-and-Conquer that is the available data is decomposed so that the initial huge data is broken into smaller data which are then distributed over different fog nodes so that the tasks are computed in parallel to each other, thereby decreasing the computational time by a factor proportional to the number of decomposed parts.

While referring to the papers [1-6] where the distribution of workload or Scattering as referred is part of the Scatter-Gather protocol, which is performed on the basis of the task priority alone. In such cases even though the task having highest priority is assigned first, the distance between the offloading node and the processing node is not accounted for, which might not lead to the desired results. That is why we make a modification in the Scatter-Gather Protocol such that the assignment is not on the basis of priority of task rather on the basis of the distance between the offloading node and the processing node. In the algorithm accounting for the distance, not only the availability of the node is taken into consideration but the distance between them is also taken into consideration so as to ensure that the offloading is even worth the time, as it might so happen that the offloading itself takes so much time that it compensates for its availability and the results are not fruitful.

The choice of the node on which task is to be offloaded/shifted is selected using the greedy paradigm by implementing the Dijkstra's algorithm which finds the shortest path between the two nodes. We run several iterations of the algorithm which evaluates the distance between the original task node and all the other available nodes in a given radius to find out which node is the closest one which is than selected for offloading. We have also run several iterations to determine which decomposition technique would be best suited for the task in hand. The administration and record of all the decomposition and the offloading is done by a Master fog node, whose functionality is to keep a watch on the number of available node and their distance from the requesting source. Moreover, it is the master node which controls the scattering, that is, distribution of the decomposed tasks to the available nodes and also gathers the individual computed result and compiles them and then delivers the final computed answer to the user.

Literature Survey

Seshadri et al (2015) [1] presents the usage of "Scatter-Gather Protocol" for calculating the address translation in the memory. As the memory space is quite vast and for changing the network address the algorithm can be applied to disintegrate the task and each of the network address translation can be performed in separate chip set of the memory and then is gathered to result in a final network address translation.
Deshpande et al (2014) [4] have presented the paper where the eviction time during the virtual machine migration can be decreased by removing access so that the resources are freed up, so that the unwanted virtual machine memory is not carried during migration. The remaining, essential virtual machine memory is scattered to intermediate hosts, which continues a ripple effect and pass on the data till it reaches its destination where the destination gathers these memory to make into single large original file memory.
Kandalla et al (2010) [5] shows and analyzes the working of "MPI_SCATTER" and "MPI_GATHER" on different scales for different algorithms and proposes operations that are hierarchical based and operations that are topology aware. By this paper, Kandalla et al described how different algorithms behave and scale down when implemented in correspondence with Scatter-Gather Protocol.
Rad et al (2016)[15] investigates the network aware VM placement in cloud for computing by employing the ant colony algorithm. Verma et al(2016)[18] aims to minimize communication time for VMs of same service type by defining the network aware VM placement optimization problem in fog computing.
From the numerous studies we conducted, we came to the conclusion that the current research in this field is about the optimization of processing time based on time latency and is more focused at ex-

tracting maximum work from nodes and fulfilling maximum requests from clients. In our research we try to apply similar methodology for a different problem which is quite untouched until now.

Problem Statement

The scenario for our problem is as follows: A cloud is assumed which is connected with N nodes (n_1, n_2, n_3,...n_N). While there are a maximum of C clients (c_1, c_2, c_3, ...c_C) that are connected in the said network of N nodes. For simplicity, we assume that all clients are identical to each other as are all the nodes.

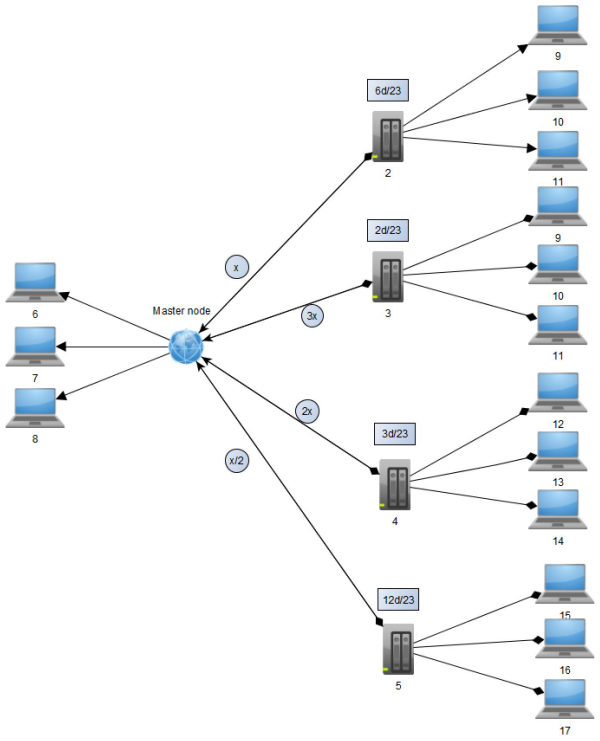


Figure 1. Server Visualization

From the diagram itself, it is quite evident that any computation that has to be done on the cloud, will take a long time even with higher computation speed of the cloud and that is because of the propagation time to and from the cloud and the client. The nodes are at a variable distance from the client. Traditionally the task would have been sent to the readily available fog node. But as discussed that would not be the best fit, alternately the node at the shortest distance from the client can be chosen, but it also creates the processing time problem.

As the total time for computation is evaluated by using the equation:

$$total_t = (2 * propagation_t) + processing_t \quad (1)$$

where $total_t$ is the total time taken for the computation while $propagation_t$ is the time taken for the data to travel from client to the node (considering that the travel time to and from the client to node as well as node to client is same) and $processing_t$ is the time taken by the processor to compute the task. Though the propagation time which accounts for a large computation time is minimized, the processing time is not reduced which is of major concern.

Now by applying Scatter-Gather algorithm [1-6], the decomposition of data is performed and the decomposed data is distributed such that the processing time ($processing_t$) is reduced. This algorithm though looks good on paper may incur some latency as the tasks are divided based on their priority and not based on their individual distances with the nodes, which in turn might result in the propagation delay thereby increasing $propagation_t$ which will result in increased $total_t$.
So it becomes clear that in order to reduce the $total_t$ we not only need to optimize $propagation_t$ or $processing_t$ but we have to optimize both of them.

Methodology

We visualize the system in a Master-Slave relation model. For communication one of the node is selected as the Master node which controls other devices and processes, which acts as its slave. Once the master-slave relationship is established, the master controls all the communication between the slaves as the slave nodes cannot communicate directly. The master node then applies the suitable decomposition technique and then the decomposed data is sent for load sharing purpose which is implemented in the form of Work Offloading (the transfer of work from one node to another without the users notice).

Architecture

The proposed system has a layered model, in order to give the system modularity. All the processes are divided in such a manner that the processing of one layer is isolated from the other which also leads to abstraction. The first layer is the cloud, which is divided into a number of cloud-lets also known as fog nodes. The second layer is the master fog nodes which acts as the supervisor to the slave fog nodes monitoring them and are geologically placed so that they do not increase the propagation latency. The third layer is the slave fog nodes which are connected to the master node and are actually responsible for communicating with the clients which are in the fourth layer.

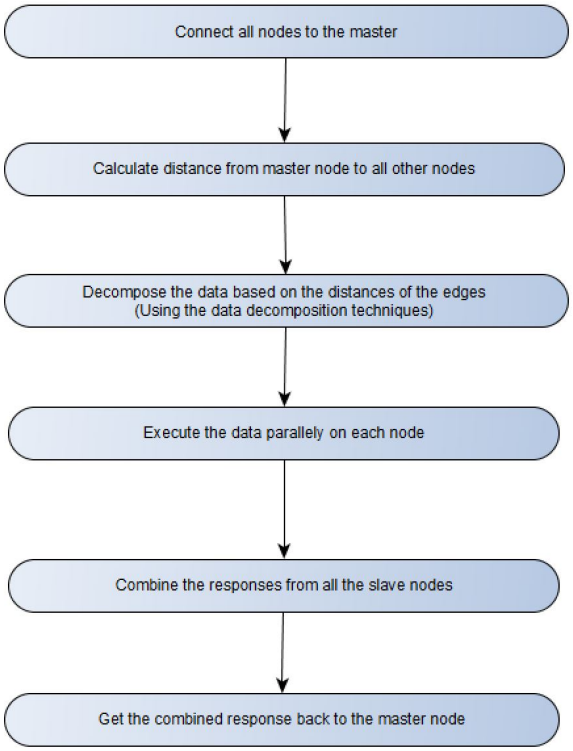


Figure 2. Flow of Proposed Model

Even though in both the levels i.e. master and slave layers, the nodes present are just fog nodes but the distinction between them are that the master node has more privileges than the slave nodes and it acts as a mediator between different slave nodes, as no two slave nodes have direct communication between them, so the only way to interact is through the master node. The master node decomposes the original task to a number of smaller tasks, which are then sent to different slave fog nodes which compute them and send it back to the master fog node which combines all the slave results and then ultimately sends it back to the user. It also supervises and monitors the availability and distance of all the slave nodes, fetches the data from all the slave nodes interprets them and if necessary redirects the data between slave nodes or communicate it back to the client.

Solution

Master node sends a request to all the slave nodes after a regular interval of time or when the client requests for resources or processors, requesting the data regarding the nodes like CPU utilization, the slave nodes in response sends the data regarding its availability. After receiving all the data, the master node analyzes and processes the data. As per the CPU utilization and the distance between the node and the client, the master also identifies the target node(s), the nodes which are underutilized and are in close proximity with the client. Such target nodes are decided and the master node overviews the transfer of data or process from the client to the fog nodes.

Before work offloading all the data that are sent by the client are analyzed and decomposed in order to distribute the workload in all the target nodes, in such cases all the existing processes that are required to process the allotted data are transferred to the node, without the user noticing the such offloading. The possible candidate nodes to which the work needs to be offloaded is selected by the Greedy paradigm based on their respective CPU Utilization and the distance between the node and the client. Once the task is completed the fog nodes on its own returns the data back to the master node. The master node checks the order of the distribution of work, in which order was the data sent to different fog nodes. Accordingly the data is then collected and analyzed by the master, which is again combined to form a single answer that needs to be sent to the client.
For identifying the node(s) closest to the client, Di-

jkstra's algorithm is employed while for selecting the node for work offloading, the Greedy paradigm is used. The above proposed algorithm is an improvement on the Scatter-Gather algorithm by implementing the Dijkstra's algorithm and employing the Greedy paradigm.

Experimental Setup

For the purpose of simulating our algorithm we have set up few fog nodes in the VIT University itself. The setting up of Fog nodes was achieved though Google APIs (Application Programming Interface) where the address, where the Fog Node is to be setup was given and the output was the establishment of the Fog Node in that location. The Technology Tower being in the center of VIT (roughly) was chosen to be the master Node. All the other Fog Nodes, namely the fog nodes in [SMV, Silver Jubilee Tower, GDN Building, H-Block (Men's Hostel), N-block (Men's Hostel)] were chosen as slaves.

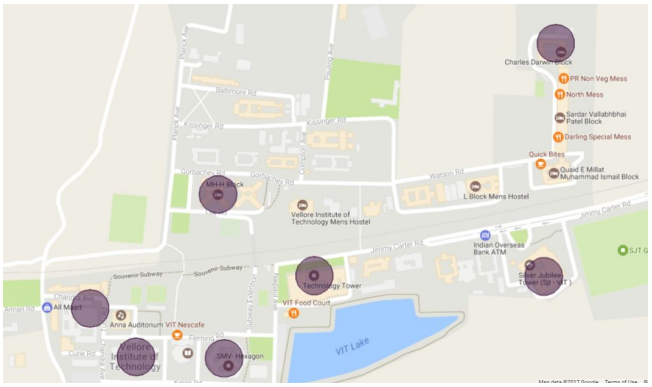


Figure 3. Simulated Fog Nodes in VIT University

The algorithm was iterated after certain interval of time, where the master node (in this case Technology Tower) received the information of the clients requesting for processors. This information was then processed at the master wherein it is checks all the close by nodes to see that, are the nodes underutilized and if such nodes are in close proximity to the client, requesting the processors and the resources. The master also analyzes the data to be processed and then it decomposes the data on the run-time using any of the Decomposition techniques: Input Data Decomposition, Output Data Decomposition, or Input/Output Data Decomposition, which is decided based on the data set.
Once the data is decomposed, the unit data is sent to all the target nodes which are then processed in isolation, with absolute no interference from other nodes and their processing. Once the processing is performed result is sent to the master which combines them to a single result and is presented to the client. In the process, it was assumed that the clients requesting are at some distances from their respective fog node as well the distances between the nodes.

Figure 4. Distribution based on Shortest Path

Thus using Dijkstra's algorithm the geologically immediate node was found and the work was offloaded. This process was continued recursively while all the requests were offloaded to the nearby nodes. While the decomposition and its eventual recombination of result is performed by the Scatter-Gather Protocol, though the priority is based on the distance between the client and the fog nodes rather than solely on the priority of the task.

Results

The desired result is achieved in the following order: data decomposition at the master node, which received the data from one of its slave nodes, which in turn had received it from the client. Once the data is decomposed, the master node distributes the data (Scatter) to a number of available fog nodes which receive a decomposed part of the original problem. Once the fog nodes receive the data they all execute the problem in parallel with each other and once it completes its execution returns the result to the

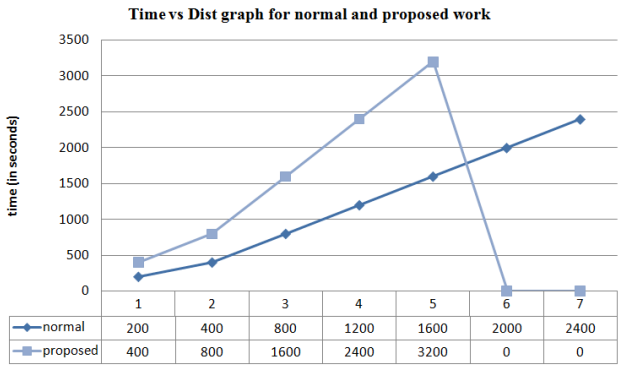
master node, which combines all the data from the nodes to which it had originally distributed the data. After getting all the decomposed result, it combines them all (Gather) to yield a final answer which is sent to the client.

To clearly understand the working of the system, let us take an example where a packet of length 10^9 bytes(1 GB) has to be propagated over a link of distance 25 km with a propagation speed of $2.5x10^8 m/s$ and transmission rate 2 Mbps. The time it takes is $10^4 s$ (2.7 minutes). However, for long distances the time taken will be more.

type\dist(in km)	5	10	20	30	40	50	60
normal	200	400	800	1200	1600	2000	2400
proposed	400	800	1600	2400	3200	00	00

Model vs Distance table

The total time required to data processing being 8600.0 for rational scatter gather and 8400.0 for efficient scatter gather (proposed through this paper). However, we assume that the processor works at maximum efficiency at 2 GB data processing and take same time for processing.
Case 1: 1 GB Distributed across nodes at distance [5, 10, 20, 30, 40, 50, 60] km from the master node;
Case 2: 2 GB Distributed in a Greedy fashion with node nearest getting 2 GB first.



Graph for time vs dist

Case 1 is the normal traditional way while case 2 is the proposed way using Scatter-Gather protocol in accordance with the Greedy Paradigm.

Conclusion

The basis of the paper is to reduce the computational time by reducing the propagation time and the processing time, so we use a data decomposition technique and then on the decomposed data Scatter-Gather Protocol algorithm is applied and compared with the results when not using the said mechanism.
In the absence of our proposed algorithm,the only time, computational speed/time is being conserved in the fog nodes are when they are manually decomposed and distributed or else they run utilizing their minimum requirement even though they are not yielding in their full potential.
Using the suggested algorithm, we were able to minimize the computation time by decomposing the data and then distributing them to fog nodes for computation and then receiving all the results from these fog nodes, in such a manner that the computation time is optimized.

References

[1] ”Seshadri, V., Mullins, T., Boroumand, A., Mutlu, O., Gibbons, P. B., Kozuch, M. A., & Mowry, T. C. (2015). Gather-scatter DRAM: in-DRAM address translation to improve the spatial locality of non-unit strided accesses. Proceedings of the 48th International Symposium on Microarchitecture, 267–280”.

[2] ”Deshpande, U., You, Y., Chan, D., Bila, N., & Gopalan, K. (2014). Fast server deprovisioning through scatter-gather live migration of virtual machines. IEEE International Conference on Cloud Computing, CLOUD, 376–383”.

[3] ”Member, S. (n.d.). An efficient Power Flow Method for Distribution System Studies under various load models, 1–6.Miled, Z. Ben. (1998). Zina Ben Miled, 216–225”.

[4] ”Deshpande, U., Chan, D., Chan, S., Gopalan, K., & Bila, N. (2015). Scatter-gather live migration of virtual machines. IEEE Transactions on Cloud Computing, PP(99), 1–14”.

[5] ”Kandalla, K., Subramoni, H., Vishnu, A., & Panda, D. K. (2010). Designing topology-aware collective communication algorithms for large scale InfiniBand clusters: Case studies with Scatter and Gather. Proceedings of the 2010 IEEE International Symposium on Parallel and Distributed Processing, Workshops and Phd Forum, IPDPSW 2010”.

[6] ”Bhatt, S. N., Pucci, G., Ranade, A., & Rosenberg, A. L. (1993). Scattering and Gathering Messages in Networks of Processors. IEEE Transactions on Computers, 42(8), 938–949”.

[7] ”Al-Obasiat Y.; Braun, R. . (2007). A Multi-Agent Flexible Architecture for Autonomic Services and Network Management; Computer Systems and Applications. Computer Systems and Applications, 2007. AICCSA ’07. IEEE/ACS International Conference On;, 124–131”.

[8] ”Alonso-Monsalve, S., Garcia-Carballeira, F., & Calderon, A. (2017). Fog computing through public-resource computing and storage. 2017 2nd International Conference on Fog and Mobile Edge Computing, FMEC 2017, 81–87”.

[9] ”Atero, F. J., Vinagre, J. J., Morgado, E., & Wilby, M. R. (2011). A low energy and adaptive architecture for efficient routing and robust mobility management in wireless sensor networks. Proceedings - International Conference on Distributed Computing Systems, 172–181”.

[10] ”Baviskar, Y. S., Patil, S. C., & Govind, S. B. (2015). Energy Efficient Load Balancing Algorithm in Cloud Based Wireless Sensor Network”.

[11] ”Bhatt, S. N., Pucci, G., Ranade, A., & Rosenberg, A. L. (1993). Scattering and Gathering Messages in Networks of Processors. IEEE Transactions on Computers, 42(8), 938–949”.

[12]”Laredo, J. L. J., Guinand, F., Olivier, D., & Bouvry, P. (2017). Load Balancing at the Edge of Chaos: How Self-Organized Criticality Can Lead to Energy-Efficient Computing. IEEE Transactions on Parallel and Distributed Systems, 28(2), 517–529”.

[13] ”Member, S. (n.d.). An efficient Power Flow Method for Distribution System Studies under various load models, 1–6”.

[14] ”Miled, Z. Ben. (1998). Zina Ben Miled, 216–225”.

[15] ”Miraftabzadeh, S. A., Rad, P., & Jamshidi, M. (2016). Efficient distributed algorithm for scheduling workload-Aware jobs on multi-clouds. 2016 11th Systems of Systems Engineering Conference, SoSE 2016”.

[16] ”Pavithra, B., & Ranjana, R. (2016). A Comparative Study on Performance of Energy Efficient Load Balancing Techniques in Cloud, 1192–1196”.

[17] ”Sharma, G., Rai, S., Busch, C., Trahan, J. L., & Vaidyanathan, R. (2015). Work-Efficient Load Balancing. Proceedings of the International Conference on Parallel Processing Workshops, 2015–May, 27–36”.

[18] ”Verma, S., Yadav, A. K., Motwani, D., Raw, R. S., & Singh, H. K. (2016). An efficient Data Replication and Load Balancing Technique for Fog Computing Environment. International Conference on Computing for Sustainable Global Development (INDIACom), 2888–2895”.

[19] ”Xiao, Y., & Krunz, M. (2017). QoE and power efficiency tradeoff for fog computing networks with fog node cooperation. IEEE INFOCOM 2017 - IEEE Conference on Computer Communications, 1–9”.