VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

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

A project submitted
in partial fulfillment of the requirements for the degree of
Bachelor of Technology in Computer Science and Engineering

By

Yashraj Agarwal (18BCI0183)

Bhawana Choudhary (18BCE2497)

Nimisha Jha (18BCE2420)

Course Instructor Prof. - Manoov R

Abstract
With the coming of innovative progression the measure of calculation performed matters as well as the time and speed of such calculation matters too. Limiting the calculation time has gotten one of the essential interests with the expanding outstanding task at hand as nobody needs to sit tight to for some preparing to happen as obvious by the need of invigorating and checking our web association when such handling takes longer than anticipated. As conveying the work alleviates the people, same can be applied to the machines where as opposed to having one single PC do all the preparing a gathering of PCs are utilized to finish them in less time and with more proficiency. Also, with the approach of distributed computing, sharing PC cycles and assets over the web has developed quickly. Notwithstanding, it might happen that dispersing the work among endless PCs can really expand the time delay as the information needs to head out to and from the cloud with expansion to the conveyance and genuine handling. So in this paper we set forward the upsides of utilizing the Fog Computing over the cloud simultaneously we attempt to enhance Scatter-Gather calculation so we can upgrade the time taken to deteriorate the outstanding burden and afterward appropriate the errand to a topographically positioned close by haze hub, so the remaining task at hand gets circulated on the hubs that are in the region so the time inertness can be limited.
Keywords
Scatter Gather, Workload Distribution, Fog Computing, Efficient distribution, Greedy paradigm, Dijkstra's algorithm

18BCI0183*, SCOPE, VIT University*

18BCE2497*, SCOPE, VIT University*

18BCE2420*, SCOPE, VIT University*

Introduction

There has been a remarkable expansion in the measure of information that is being produced and hence handling this information has gotten dramatically troublesome. To diminish the time inactivity and increment the client experience another model known as Fog Computing is utilized, which can work on the organization edges and thus it eliminates the requirement for sending the information to the cloud for investigation as the work should be possible in the systems administration hub itself, which relatively is path quicker than sending information to the cloud for investigation and afterward sitting tight for the outcomes.

In Fog Computing notwithstanding, various hubs or edges are topographically appropriated which handle the approaching solicitation from the client and cycles it to give the outcome. Despite the fact that with the utilization of Fog figuring the client experience is improved still if the outstanding task at hand is tremendous, the calculation time would at present be essentially huge and the degree of client experience expected would not accomplished. As registering in a haze hub likewise requires a ton of time as by sending the information to haze hub, just an opportunity to send and get the information is diminished however the handling time stays unaltered. So as to undermine the time taken for handling the main conceivable substitute is to apply Divide-and-Conquer that is the accessible information is

decayed so the underlying colossal information is broken into more modest information which are then circulated over various mist hubs so the undertakings are registered in corresponding to one another, in this way diminishing the computational time by a factor relative to the quantity of disintegrated parts.

While alluding to the papers [1-6] where the appropriation of outstanding burden or Scattering as alluded is important for the Scatter-Gather convention, which is performed based on the undertaking need alone. In such cases despite the fact that the errand having most noteworthy need is appointed first, the separation between the offloading hub and the preparing hub isn't represented, which probably won't prompt the ideal outcomes. That is the reason we make an adjustment in the Scatter-Gather Protocol with the end goal that the task isn't based on need of errand rather based on the separation between the offloading hub and the handling hub. In the calculation representing the separation, not just the accessibility of the hub is thought about however the separation between them is additionally contemplated to guarantee that the offloading is even worth the time, as it may so happen that the offloading itself takes so much time that it makes up for its accessibility and the outcomes are not productive.

The decision of the hub on which undertaking is to be offloaded/moved is chosen utilizing the ravenous worldview by executing the Dijkstra's algorithm, which finds the most brief way between the two hubs. We run a few emphasess of the calculation which assesses the separation between the first errand hub and the wide range of various accessible hubs in an offered span to discover which hub is the nearest one which is than chosen for offloading. We have likewise run a few emphasess to figure out which disintegration procedure would be most appropriate for the errand close by. The organization and record of all the deterioration and the offloading is finished by a Master mist hub, whose usefulness is to keep a watch on the quantity of accessible hub and their good ways from the mentioning source. In addition, it is the ace hub which controls the dispersing, that is, appropriation of the disintegrated errands to the accessible hubs and furthermore accumulates the individual registered outcome and assembles them and afterward conveys the last processed response to the client.

Literature Survey

Seshadri et al (2015) [1] presents the utilization of "Dissipate Gather Protocol" for ascertaining the location interpretation in the memory. As the memory space is very tremendous and for changing the organization address the calculation can be applied to break down the undertaking and every one of the organization address interpretation can be acted in discrete chip set of the memory and afterward is assembled to bring about a last organization address interpretation.

Deshpande et al (2014) [4] have introduced the paper where the ousting time during the virtual machine movement can be diminished by eliminating access so the assets are opened up, so the undesirable virtual machine memory isn't conveyed during relocation. The excess, basic virtual machine memory is dissipated to middle of the road has, which proceeds with an expanding influence and pass on the information till it arrives at its objective where the objective accumulates these memory to make into single enormous unique document memory.

Kandalla et al (2010) [5] shows and investigates the working of "MPI SCATTER" and "MPI GATHER" on various scales for various calculations and proposes activities that are progressive based and tasks that are geography mindful. By this paper,

Kandalla et al portrayed how various calculations act and scale down when actualized in correspondence with Scatter-Gather Protocol. Rad et al (2016)[15] examines the organization mindful VM arrangement in cloud for processing by utilizing the subterranean insect state calculation. Verma et al(2016)[18] means to limit correspondence time for VMs of same assistance type by characterizing the organization mindful VM arrangement enhancement issue in haze processing.

From the various examinations we led, we reached the resolution that the flow research in this field is about the advancement of preparing time dependent on time dormancy and is more engaged at separating greatest work from hubs and satisfying most extreme solicitations from customers. In our exploration we attempt to apply comparative philosophy for an alternate issue which is very immaculate up to this point.

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.

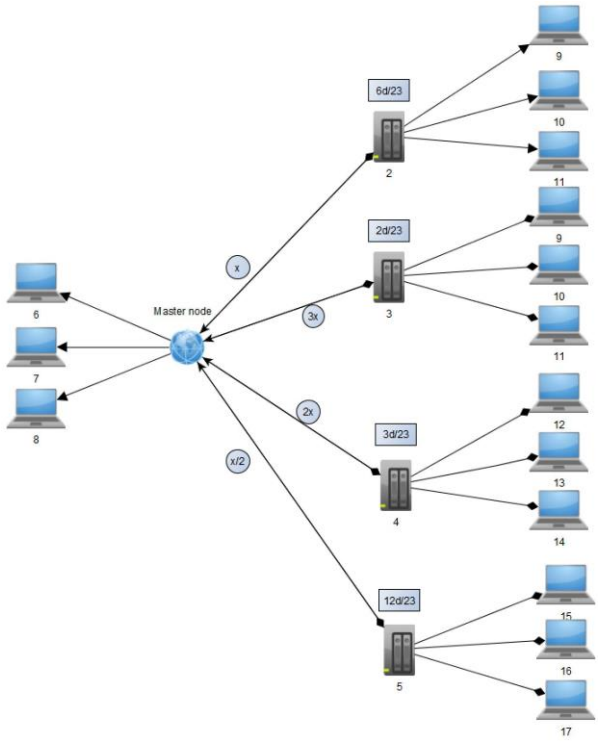


Figure 1. Server Visualization

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.

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.

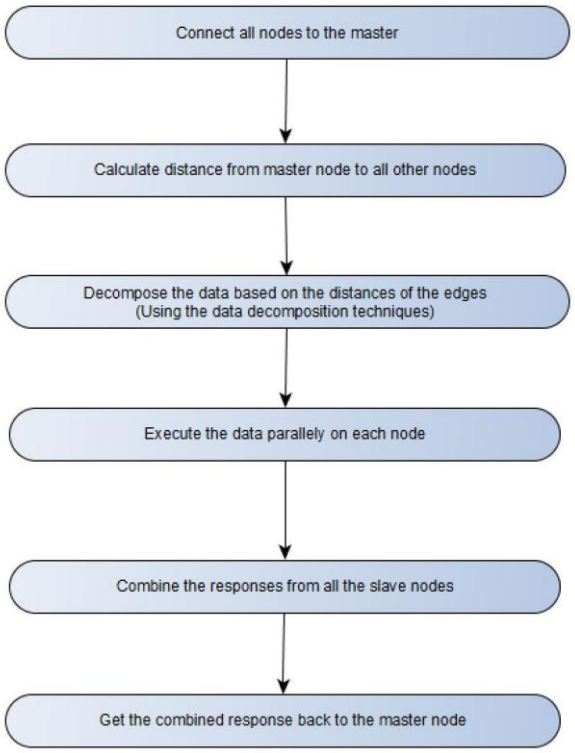


Figure 2. Flow of Proposed Model

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, Dijkstra'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.

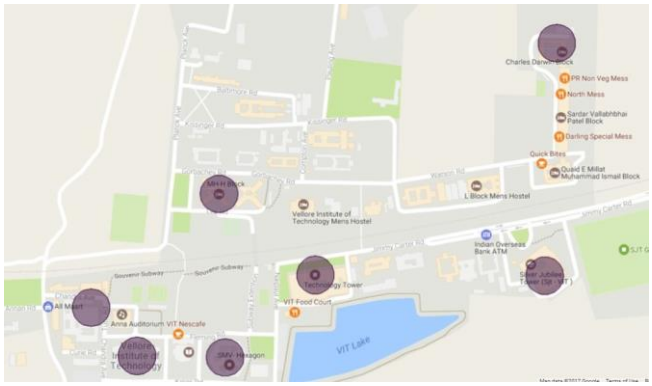


Figure 3. Simulated Fog Nodes in VIT University

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 ScatterGather 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

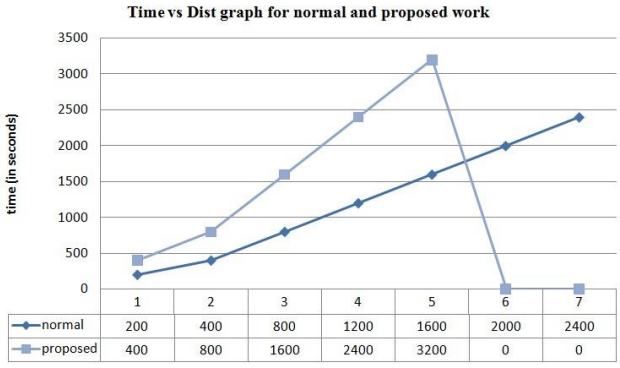
The desired result is achieved in the following order: data decomposition at the master node, which received the data from one of its slave nodes, which in turn had received it from the client. Once the data is decomposed, the master node distributes the data (Scatter) to a number of available fog nodes 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".