Scheduling in Heterogeneous Decentralized and Unstructured Peer-To-Peer Systems to Improve the Computation Performance of Genomic Data

Velvizhi N.[1]* and Manjula D.[2]
1. Department of Computer Science and Engineering, Indira Institute of Engineering and Technology, Pandur, Tiruvallur-631203, INDIA
2. Department of Computer Science and Engineering, College of Engineering, Anna University, Chennai, INDIA
*vizhichezhian@yahoo.co.in

Abstract

Biotechnology applications and high performance computing goes hand in hand to handle the large datasets. Peer-to-peer (P2P) network sets up loosely coupled application-level overlay over the Internet for facilitating effective sharing of resources. Several applications, which implement simultaneously on heterogeneous platforms contend for CPU as well as network resources. In the current work, we take into consideration the issue of scheduling applications for ensuring fair as well as effective execution n a distributed network of processors. Among the set of all existing connected peers source peer request for service from other peers. Based on the response delay from the peers, the huge quantity of independent computational tasks is scheduled.

The aim of scheduling is the maximization of throughput of every application simultaneously guaranteeing fair distribution of resources between the applications. This shows how to build a periodic schedule for n tasks. For single-level communication, the solution is featured by the scheduling performance and the quantity of tasks finished per unit time (throughput). For multi-level communication, the method needs global knowledge of every application as well as platform parameter. For huge-scale platforms where huge data is processed like in genomic data in biotech applications, centralized schedulers globally coordinating might not be realistic. Hence, we examined decentralized schedulers of which utilize solely local data at every participant peer.

We evaluate their performance through simulations and contrast them to optimum centralized solutions got. Though our outcomes are grounded in basic assumptions, and do not explore every parameter (like the maximal quantity of tasks which could be performed on one node), they offer insights into the important question of fair as well as optimal scheduling heterogeneous applications.

Keywords: Unstructured peer-to-peer networks, scheduler, response delay, heterogeneous.

Introduction

Biotechnology applications dealing with genome data which consists of gene sequence require higher storage space and analyzing them are computationally intensive. It is seen that parallel or distributed architecture is required to processing the huge amount of information. Peer-to-peer networks are logical overlay networks situated atop a physical network, wherein every relates to a node in the P2P network, as well as is present in a host in the physical network. Every peer has equal role. The connections among the peers are logical, all of them corresponding to a physical route in the physical network. This is defined through a routing protocol and is comprised of more than a single physical link. Logical connections may be included in the P2P network randomly so long as a related physical route is discovered, i.e., the physical network is linked. The overlay topologies being flexible as well as the P2P network being controlled in a decentralized manner makes it appropriate for distributed application.

For instance, it may be utilized for distributed data sharing, wherein peers declare the information they possess as well as exchange data with one another via a loosely formed P2P network. Unstructured P2P systems are extremely infrastructure, because of their decentralized nature, it is capable of easily perform updates, have increased storage, as well as provides error-tolerant characteristics. Unstructured P2P networks do not have strictly organized peers or content.

Previous studies primarily focus on peer as well as content discovery, overlay topology formation, fairness as well as incentive problems and so on. But, not enough research has been performed for investigating the issue of data distribution that is a core element of all file sharing applications. Existing scheduling methods focuses on network traffic rather than scheduling the processes. The outcome of above scheduling methods leads the problem of minimal throughput along with minimal use of resources.

When more than one peer responds for request, the source peer may not be able to assign the task to the peer which is idle since there is no effective scheduler. Protocols for the allocation of computation resources such as CPUs as well as memories to processes as well as synchronizing several conflicting procedures are detailed for minimizing the computational time as well as response time, for maximizing throughput, as well as for minimizing memory space[1].

P2P file sharing resolves the issue through permitting peers to function as servers. It is interesting to note that in well-structured P2P file sharing networks, several taking part in the file sharing sessions implies improved performance, as every peer is capable of downloading concurrently from several peers. Because of the considerable performance enhancements with cooperative file sharing, there has been a rise in the usage of P2P file sharing application such as Bit Torrent, Gnutella[2], Kazaa[3] and Napster[4] etc.

For huge-scale platforms, especially those in which resource availability alters with time, centralized schedulers might not be desirable. Only local information like the current capacities of processor's neighbours will probably be available. The major aim of the current work is the investigation into if decentralized scheduling protocols are capable of reaching an optimum throughput. If every application has an unlimited supply of tasks, every application ought to focus on the maximization of the average quantity of tasks processed every time unit. When the quantity of tasks is extremely huge, optimization of the steady-state throughput permits the derivation of periodic asymptotically optimum schedules for the makespan [5].

We look into the issue of scheduling applications requiring rapid response time (rather than higher throughput). The set of applications has sets of computationally-heavy jobs which ought to be completed as fast as possible, as well as the jobs that are organized as a directed acyclic graph (DAG). For those applications, the temporal structure of availability as well as resources selection as per the structure is crucial to the performance [6].

Related Work

The present models of distributed systems, that include grid computing as well as P2P, computation suffers from an issue of knowledge as well as resources fragmentation. Because of the drastic rise in broadband users, P2P file sharing systems are deployed on a huge scale over the Internet. Many algorithms have been implemented to schedule the task based on the quantity of peers connected in the network. The global scheduling issue has been analysed in [7] through modeling it as a minimal-cost network flow issue. The local scheduling issue, provided a set of pull parents as well as their buffer map has been discovered to be NP-hard in[12].

Though mesh-pull has been shown to provide simplicity, robustness, as well as higher bandwidth usage, the pull method results in longer delays [8]. Peers are capable of maximizing the download speed, through request of various pieces from various peers simultaneously. A scheduling method is required for peers to decide the pieces to demand, as well as to whom the requests ought to be made. A poor scheduling protocol might result in each peer obtaining almost the same set of pieces, and subsequently reduces the quantity of file piece sources that a peer may concurrently download from [13].

BitTorrent utilizes the *Rarest Element First (REF)* protocol, wherein the pieces which is not available with most peers are downloaded initially. The protocol is excellent at improving the accessibility of various file pieces and is effective in the distribution of every piece from the original source to various peers over the network [11]. But, the simulations reveal that *REF* is not an optimum scheduling protocol. The completion time of several protocols is evaluated and it is shown that the popular *Fastest Node First* protocol could lead to solutions that are poorer than the optimum by an unbounded factor [9].

Most Demanding Node First (MDNF) design analyses the quantity of cycles required for the whole distribution of the entire queries to every peer which is connected in the network. Here, in this case the design considers only the number of queries that are scheduled for processing. The other measures like processing time, waiting time and turnaround time are not considered for the computation of performance. But it schedules the transmissions node by node. The maximal quantity of transmissions for every cycle is not achievable [10].

To overcome the above issue we designed **Optimal Response Scheduling (ORS) Algorithm,** which performs better than MDNF.

Problem Definition

Peer-to-Peer Network Architecture: This system assumes that the users are connected with logical connections linking each pair of peers (that is, entirely connected graph, a sample network is given in Figure 1).

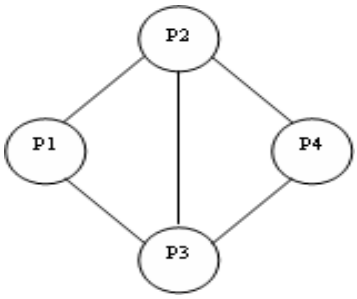

Fig. 1: Example of P2P Network

In fig. 1, four peers called P1, P2, P3 as well as P4 that are linked via common networks. In the below instance, Peer "P1" requires some data that is present in"P2","P3" or "P4". As peer P1 requires data, "P1"may be taken into consideration as a source peer and "P1" has the information given below:

- Peers linked to it
- Time of joining and leaving the network

Design of Optimal Response Scheduling (ORS) Algorithm: The system is an effective technique to schedule number of queries with their logical neighbors. In ORS design, every peer who are linked to the network are in ready

state for probing or computing the queries. The optimum route protocol though which the query may be transmitted.

Once all the above are computed, the query is forwarded via the optimum route, and the time taken for sending as well as receiving the queries are recorded as Adjacency matrix. The peers are assigned Time quantum as well as system time. When response time of the peers is more than the time quantum provided, the query is resent. In such case, the wait time is also included with the processing time when computing the time taken or required for responding.

The protocol discovers every possible path from the source peer to the destination peer, and therefore discovers the optimum route. This protocol is different from the previously described protocols, as it calculates the waiting time, processing time, and turnaround time. The complexity of the protocol is is $O(n^2)$:

1. Set TTL = 0
2. Obtain the Adjacency Matrix [A] for the Network and copy to route [][].
3. Obtain every probable route on the basis of matrix A
4. To find Optimum Route ()
For k: = 1 to number of vertices
For I := 1 to number of vertices
For j: = 1 to number of vertices
Route[i][j] = minimum (route[i][j], route[i][k]+route[k][j]);

Wherein I represents the source, j represent destination, k represents the intermediary node while route[i][j] retains the shortest route.
5. The system time is noted while forwarding the query
6. Compute Actual Time.
7. Processing time = A[source,destination]+A[destination,source]
8 Waiting time WT(n)=Waiting time(n-1)+processing time(n-1)
9. Response time RT(n) = Processing time(n)+ waiting time(n)
10.Turnaround time TT(n) = processing time(n) + waiting time(n)
11. Compute Average Waiting Time(AWT) = $\frac{\sum_{i=1}^{n} WTi}{n}$
12. Average Response time(ART) for "n" quantity of queries = $\frac{\sum_{i=1}^{n} RTi}{n}$
13. Average Turnaround Time(ATT) for "n" quantity of queries = $\frac{\sum_{i=1}^{n} TTi}{n}$
14. In the event of no reception of response; use
If (Waiting Time == System time)
If (Response Received)
If (flag==1)then
Success (process finished)
Exit
End if
Else Resend the queries
End if
End if

Experiment and Results

Process execution comprises a cycle of CPU execution as well as waiting time to get the CPU. Initially the assumption is CPU is idle and 50 Queries were sent through the optimal path and the processing time for every query is computed by the time of process is submitted to that of completion time. The average processing time is computed by the ratio of total quantity of query processed and that of number of query sent. During the process execution, the following metrics were measured and the respective time is noted in table 1.

Throughput: The quantity of processes finished per unit time. This time varies from process to process.

Waiting Time: The time spent on the ready queue is the waiting time.

Response Time: The quantity of time taken to begin responses to the queries.

Turnaround Time: The turnaround time includes the wait time to get into the CPU and the wait time in the ready queue.

Table 1
Computation of turnaround time with ORS Algorithm

S.N.: No. of Queries (in 10's)	Average Processing Time (in millisec)	Average Waiting Time (in millisec)	Average Turnaround Time (in millisec)
1 5	2.10	1.03	3.13
2 10	3.90	2.07	5.97
3 15	5.92	3.10	9.02
4 20	7.99	3.98	11.97
5 25	10.78	5.02	15.8

ORS with MDNF: We have described the design of MDNF for further enhancing ORS. The system is formulated with MDNF infrastructure and forwarded a set of queries and the processing and waiting time so as to compute turnaround time. Likewise, the same set of queries are forwarded and query response time is observed with ORS. The data obtained via this design is given below:

Table 2
Comparison of Turnaround Time (MDNF AND ORS)

S.N.: No. of Queries (in 10's)	Average Turnaround Time (in millisec) (MDNF)	Average Turnaround Time (in millisec) (ORS)
1 5	6.09	3.13
2 10	10.46	5.97
3 15	16.90	9.02
4 20	21.43	11.97
5 25	32.65	15.8

Figure.2 displays the comparative outcomes of MDNF as well as ORS, wherein the x-axis denotes the quantity of queries, while y-axis denotes the average Turnaround time per query in ms. The graph indicates that the average turnaround time of ORS is minimized around 40 percent in comparison to MDNF. In figure.2, we note that the convergent speed of MDNF is the slowest, hence its total performance in dynamic settings is poorer than ORS'. ORS performs better than MDNF, all things considered.

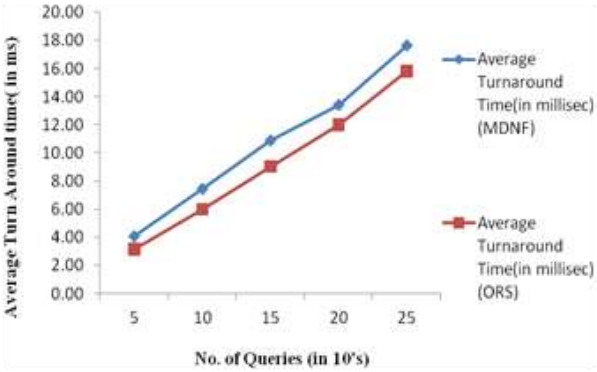

Fig. 2: Comparison on Turnaround Time

Conclusion and Future Work

Optimal Response Scheduling (ORS) Algorithm is suggested for reducing the query response time in P2P environments. The system is scalable as well as entirely distributed. Furthermore, it does not need global knowledge when nodes are optimizing their logical neighbours. The performance benefits of ORS are consistent with various time intervals as well as varying quantity of queries. ORS attains around 40 percent lesser turnaround time than MDNF. The outcomes from the experiments reveal that ORS considerably performs better than all other current methods available. Moreover, the system may be improved with the scheduling technique which can determine the maximal quantity of tasks which can be performed on one node.

References

1. Haiyan Luo, Wei An, Song Ci and Dalei Wu, A distributed utility-based scheduling for peer-to-peer video streaming over wireless networks, *Wireless Communications and Mobile Computing*, **16(12)**, 1556-1569 **(2015)**

2. The official BitTorrent website, http://www.bittorrent.com

3. The official Gnutella website, http://www.gnutella.com

4. The official Kazaa website, http://www.kazaa.com

5. The official Napster website, http://www.napster.com

6. Banino C., Beaumont O., Carter L., Ferrante J., Legrand A. and Robert Y., Scheduling Strategies for Master-Slave Tasking on Heterogeneous Processor Platforms, *IEEE Trans. Parallel and Distributed Systems*, **15(4)**, 319-330 **(2004)**

7. Bahman Javadi, Derrick Kondo, Jean-Marc Vincent and David P. Anderson, Discovering Statistical Models of Availability in Large Distributed Systems, An Empirical Study of SETI@home, *IEEE Trans. Parallel and Distributed Systems*, **22(11)**, 1896-1903 **(2011)**

8. Zhang M., Xiong Y. and Zhang Q., On the optimal scheduling for media streaming in data-driven overlay networks, Proc, IEEE GLOBECOM, New York **(2006)**

9. Zhang M., Zhang Q., Sun L. and Yang S., Understanding the power of pull-based streaming protocol: Can we do better?, *IEEE J, Sel. Areas Commun*., **25(9)**, 1678–1694 **(2007)**

10. Khuller S. and Kim Y.A., On broadcasting in heterogeneous networks, Proc. ACM-SIAM Symposium on Discrete Algorithms, **(2004)**

11. Jonathan S.K. Chan, Victor O.K.Li. and King-Shan Lui, Performance Comparison of scheduling algorithms for Peer-to-Peer Collaborative File Distribution, *IEEE Journal on Selected Areas in Communications*, **25(1)** **DOI:** 10.1109/JSAC. 2007.070115 **(2007)**

12. Risson J. and Moors T., Survey of research towards robust peer-to-peer networks: Search methods, *Comput. Netw.*, **50(17)**, 3485–3521 **(2006)**

13. Demetrios Zeinalipour-yazti, Vana Kalogeraki and Dimitrios Gunopulos, P Fusion: A P2P architecture for internet-scale content-based search and retrieval, *IEEE Transactions on Parallel and Distributed Systems,***18(6)**, 804-817 **(2007)**.