Distributed Offloading of Tasks for Optimization Purposes

Nachiket Patel
nnpatel5@ncsu.edu
North Carolina State University
Raleigh, North Carolina, USA

Abstract

Heterogeneous distributed systems are more involved when it comes to balancing work loads due to the varying performance level of machines in the network. With an increase in the number of mobile devices and more reliable internet connections offloading demanding workloads to more powerful machines in a network can provide performance improvement. Tasks can be dynamically or statically offloaded, either whole tasks can be offloaded or parts of a task can be offloaded. Several things need to be considered before offloading a task; determine if the task can be offloaded effectively, which machines on the network can accept offloaded tasks, if the overhead of offloading is less than the overhead of waiting to run on the same machines.

Keywords: load balancing, decentralized, task offloading, distributed systems

1 Introduction

Load balancing is an integral part of distributed systems and there are several existing mechanisms which perform the task of load balancing in the form of system-based and application based [14]. Offloading can be done by either dynamically splitting up a task [8], through offloading entire tasks [15]. Splitting up tasks can be done with either through application based or system based adaptations [14]. Determining which machines/nodes to offload tasks to within the network requires keeping an up to date index of machines and their workloads. The main motivation behind implementing the mechanisms is to allow less powerful devices be capable of performing computationally expensive tasks which require powerful machines. Another motivation behind implementing this is to allow for dynamic load balancing across nodes in a network where there are machines idling and can be utilized to speed up computation.

There are challenges which will be encountered when implementing a mechanism which allows for offloading tasks, the main one is to decide if tasks should be split and distributed or individual tasks within a batch. The next thing to consider is to implement some mechanism of determining bandwidth of connections between nodes in the network, this will allow us to compute some part of the overhead involved in offloading the task. Another challenge will be how to determine which node to offload tasks to, this will require communicating with nodes in the network to establish which nodes have the appropriate resources available to offload tasks to. Furthermore, determining if the overhead of offloading the task is smaller than waiting for appropriate resources to become available on the current node where the task was received. Some of the sub-mechanisms which need to be implemented can be used from existing projects and research papers described in the related works section.

2 Related Work

There are various existing systems such as the puppeteer project which supports adaptation without needing to adapt the software. The puppeteer project is a a mechanism which allows for offloading tasks without modifying the application in mobile environments. The main components of the project are the kernel, driver, transcoder and policy [1]. A different approach to task offloading would use a system such as Coigen [3] which dynamically partitions which takes the approach of the system software being responsible for distribution decomposition. This paper discusses taking a binary of the task and using graphs to model the application and using a graph cutting algorithm to split the up into sub-tasks which can be offloaded.

Existing research such as the one described in [14] which presents an algorithm utilizing max cut with the utilization of classes as execution units to components of tasks which cannot be computed on the mobile device. The performance evaluation show that methods described provide significant improvements. Another similar paper which discusses dynamic task offloading is described in [7]. This method explores the offloading of tasks in low-latency networks with wireless connections. For splittable tasks the 'POST' algorithm [8] explores the the idea of using time slots to find appropriate time to offload tasks. It uses broadcasting to determine which nodes can be accesses for the offload. This could be something to consider when implementing the system. A similar but different paper described the offloading of pairs of tasks with the 'POMT' algorithm [15] which offlaods tasks without splitting theme with the aid of a performance metric called 'delay reduction ratio (DDR)'. The results show the algorithm proposed achieved near-optimal system delay, and further explores the offloading cost with respect to the number of decision slots. Another variation on the offloading problem is explored in [16] which showcases a peer-to-peer

task offloading algorithm for mobile cloud computing. The paper explores concepts such as the delay constraints in offloading decisions, how to retrieve results from offloaded nodes, and most importantly how to transport the workload. This source will examined more closely when implementing the offloading mechanism as it explores a lot of the challenges described in the introduction section.

Other things which might need to be considered are the type of distributed networks on which the offloading mechanism will be usable, and a potential concept of arbitrary topologies explored in [6]. This paper describes a framework which discusses the challenges of networks such as multi-hop wireless networks, embedded networks and more. The framework is constructed on three main components, the network model, task model, and the computation model. The findings in this paper show that a near-optimal performance performance in heterogeneous networks. For more insight into the design of an adaptive offloading mechanisms, results from [5, 9] which discuss load balancing algorithms which can be used as a starting point due to the effective problem being solved is that of optimization which correlates with load balancing. Both papers explore grid environments and show that this method is effective in minimizing the response time which can be beneficial when implementing offloading as some of the overhead comes from the transporting the task to the compute node and returning the results.

A problem not entirely discussed in the introduction was the possibility of failure of nodes on the network which can lead to failures of offloaded tasks. The paper [11] explores the concept of failure aware task offloading within heterogeneous mobile cloud computing. The paper describes the 'Dynamic Content Aware Task Offloading Algorithm (DCTOA)and Failure Aware Algorithm (FAA) schemes' which implements a checkpoint technique used to determine if a task has failed. The results show that the scheme described is more effective then static task offloading algorithms when it comes to relative percentage deviation.

2.1 Related Potentially Useful Unread Work

Other papers which were found but have not been read but might be of use include:

Dynamic content and failure aware task offloading in heterogeneous mobile cloud network [11].

Large scale user assisted multi-task online offloading for latency reduction in D2D-Enabled heterogenous networks [10].

Task offloading between smartphones and distributed computational resources [4].

Salable testbed for task offloading and development of heterogeneous edge computing [2].

Distributed load balancing algorithm for big data processing over multi-core cluster [12].

Dynamic QoS-aware multimedia service configuration in ubiquitous computing environments [13].

References

[1] Eyal de Lara, Dan Wallach, and Willy Zwaenepoel. 2000. Puppeteer: Component-Based Adaptation for Mobile Computing. *SIGOPS Oper. Syst. Rev.* 34, 2 (April 2000), 33. https://doi.org/10.1145/346152.346224

[2] B. Diao, K. Yang, Y. Mei, Q. Wang, C. Li, Z. An, and Y. Xu. 2019. A Scalable Testbed for Task Offloading and Deployment of Heterogeneous Edge Computing. In *2019 IEEE International Conferences on Ubiquitous Computing Communications (IUCC) and Data Science and Computational Intelligence (DSCI) and Smart Computing, Networking and Services (SmartCNS)*. 586–591. https://doi.org/10.1109/IUCC/DSCI/SmartCNS.2019.00123

[3] Galen C. Hunt and Michael L. Scott. 1999. The Coign Automatic Distributed Partitioning System. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation* (New Orleans, Louisiana, USA) *(OSDI '99)*. USENIX Association, USA, 187–200.

[4] Shigeru Imai. 2012. Task Offloading Between Smartphone and Distributed Computational Resources. *Faculty of Rensselaer Polytechnic Institute* (2012), 52. http://wcl.cs.rpi.edu/theses/imai-master.pdf

[5] Kashif Inayat and Seong Oun Hwang. 2018. Load balancing in decentralized smart grid trade system using blockchain. *Journal of Intelligent & Fuzzy Systems* 35, 6 (2018), 5901 – 5911. https://proxying.lib.ncsu.edu/index.php?url=https://search-ebscohost-com.prox.lib.ncsu.edu/login.aspx?direct=true&db=bth&AN=133721660&site=ehost-live&scope=site

[6] B. Liu, Y. Cao, Y. Zhang, and T. Jiang. 2020. A Distributed Framework for Task Offloading in Edge Computing Networks of Arbitrary Topology. *IEEE Transactions on Wireless Communications* 19, 4 (2020), 2855–2867. https://doi.org/10.1109/TWC.2020.2968527

[7] C. Liu, M. Bennis, M. Debbah, and H. V. Poor. 2019. Dynamic Task Offloading and Resource Allocation for Ultra-Reliable Low-Latency Edge Computing. *IEEE Transactions on Communications* 67, 6 (2019), 4132–4150. https://doi.org/10.1109/TCOMM.2019.2898573

[8] Z. Liu, Y. Yang, K. Wang, Z. Shao, and J. Zhang. 2020. POST: Parallel Offloading of Splittable Tasks in Heterogeneous Fog Networks. *IEEE Internet of Things Journal* 7, 4 (2020), 3170–3183. https://doi.org/10.1109/JIOT.2020.2965566

[9] R. Shah, B. Veeravalli, and M. Misra. 2007. On the Design of Adaptive and Decentralized Load Balancing Algorithms with Load Estimation for Computational Grid Environments. *IEEE Transactions on Parallel and Distributed Systems* 18, 12 (2007), 1675–1686. https://doi.org/10.1109/TPDS.2007.1115

[10] M. Sun, X. Xu, X. Tao, and P. Zhang. 2020. Large-Scale User-Assisted Multi-Task Online Offloading for Latency Reduction in D2D-Enabled Heterogeneous Networks. *IEEE Transactions on Network Science and Engineering* 7, 4 (2020), 2456–2467. https://doi.org/10.1109/TNSE.2020.2979511

[11] Q. u. A. Mastoi, A. Lakhan, F. A. Khan, and Q. H. Abbasi. 2020. Dynamic Content and Failure Aware Task Offloading in Heterogeneous Mobile Cloud Networks. In *2019 International Conference on Advances in the Emerging Computing Technologies (AECT)*. 1–6. https://doi.org/10.1109/AECT47998.2020.9194161

[12] Yuzhu Wang, Jinrong Jiang, Huang Ye, and Juanxiong He. 2016. A distributed load balancing algorithm for climate big data processing over a multi-core CPU cluster. *Concurrency and Computation: Practice and Experience* 28, 15 (2016), 4144–4160. https://doi.org/10.1002/cpe.3822 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.3822

[13] Xiaohui Gu and K. Nahrstedt. 2002. Dynamic QoS-aware multimedia service configuration in ubiquitous computing environments. In *Proceedings 22nd International Conference on Distributed Computing Systems*. 311–318. https://doi.org/10.1109/ICDCS.2002.1022268

[14] Xiaohui Gu, K. Nahrstedt, A. Messer, I. Greenberg, and D. Milojicic. 2003. Adaptive offloading inference for delivering applications in pervasive computing environments. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, 2003. (PerCom 2003).* 107–114. https://doi.org/10.1109/PERCOM.2003.1192732

[15] Y. Yang, Z. Liu, X. Yang, K. Wang, X. Hong, and X. Ge. 2019. POMT: Paired Offloading of Multiple Tasks in Heterogeneous Fog Networks. *IEEE Internet of Things Journal* 6, 5 (2019), 8658–8669. https://doi.org/10.1109/JIOT.2019.2922324

[16] C. Zhou and C. Tham. 2018. Deadline-Aware Peer-to-Peer Task Offloading in Stochastic Mobile Cloud Computing Systems. In *2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON).* 1–9. https://doi.org/10.1109/SAHCN.2018.8397142