Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 12, December 2014, pg.543 – 548

SURVEY ARTICLE

A Survey on Load Balancing in Heterogeneous Distributed Computing Environment

Javed Hussain
Research Scholar, Mewar University, Chittorgarh, Rajasthan, India

---

Abstract - Load balancing are important components of Distributed Systems. Designing efficient load balancing techniques has been always a challenges researcher. New Applications and architecture require new strategies for load balancing. Use of heterogeneous computing architecture requires partitions that account for uniform computing network and memory resources. Tremendous amount of literature is available for load balancing. In this paper important developments are put at one place and further issues of load balancing research are explored. We intend to provide a template for load balancing research before further researcher.

Keywords- Load balancing, distributed system, heterogeneous distributed computing

---

I. INTRODUCTION

The basic idea with load balancing is to share incoming connections across multiple hardware devices. For example a web site may receive many thousands of hits an hour. The web site load balancer needs to distribute these hits across multiple backend servers in order to process the requests in a reasonable amount of time. If only one server handled all the incoming requests most of us would have given up on the Internet by now due to how slow it would be. There are two main types of load balancing:

- software load balancing
- hardware load balancing

With software load balancing the software accepts the incoming connections and distributes them across multiple servers. *The Apache Web Server* is a typical example of a software load balancer. It accepts web requests and shares them across multiple backend servers. With hardware load balancing a hardware device such as a switch does the load balancing. This is faster than using software. The two main types of hardware load balancing are:

- server load balancing
- switch load balancing

Which type of load balancing you use depends on the requirements of your system. The problem of load balancing continues to raise interesting challenges to researchers. The operating systems and distributed application design must include solutions for solving it. The main reason is to increase the economic efficiency. The main target of a distributed kernel is to provide supplementary computing power to a local user whenever it is needed. The most efficient solution is to use other partially loaded machines for computing sub-trees, in other words to use a remote computing solution [1]. This process is known as load balancing and must be automatically made with respect to the end user needs. The classic approach is to use algorithms, which try to solve the problem by analyzing the current state of the system. This way has some disadvantages like the need of medium or large computing power and due to this problem the scalability is poor on larger systems. The other approach is the one which try to estimate some how the future system state in order to propose a more stable solution. The assignment of work to processors is critical in parallel simulations. It maximizes application performance by keeping processor idle time and interprocessor communication as low as possible. In applications with constant workloads static load balancing can be used as a pre-processor to the computation. Other applications such as adaptive finite element methods have workloads that are unpredictable or change during the computation. Such applications require dynamic load balancers that adjust the decomposition as the computation proceeds. Numerous strategies for static and dynamic load balancing have been developed including recursive bisection (RB) methods [3-5], space-filling curve (SFC) partitioning [6-10] and graph partitioning (including spectral [4,11], multilevel [12-14], and diffusive methods [15-17]). These methods provide effective partitioning for many applications, perhaps suggesting that the load-balancing problem is solved. RB and SFC methods are used in crash [18], particle [6, 18], and adaptive finite element simulations [3, 8, 19]. Graph partitioning is effective in traditional [13, 14], adaptive [20, 21], and multiphase [22, 23] finite element simulations, due, in part, to high quality serial (Chaco [24], METIS [14], Jostle [25], Party [26], Scotch [27]) and parallel (ParMETIS [28], PJostle [25]) graph partitioners.

II. RELATED WORK

Earlier focus was given to homogeneous distributed computing environment, but present research has a focus on heterogeneous computing as point by Moons *et al.* in [29]. They also presented a detail discussion DACNOS (Distributed Academic Computing Network Operating System) which is a system for general solution for running distributed applications in heterogeneous network. Bond in [30] explored an experimental task allocation system of UNIX called STARS. He showed the utility of STARS for load sharing and thus maximizing the throughput of distributing computing. He showed that a load sharing environment increases global system performance for little overhead or user effort. Winckler [31] has surveyed on context-sensitive load balancing policies and information requirements in decentralized architecture. He has presented techniques for utilizing job context information in distributed computing system. He discusses context-sensitive sequencing rules as main part of load balancing policies and advantages of workload predictability for dynamic look ahead planning. A topic of the discussion was the server state and work plan information requirements of the proposed policies which vary in a wide range. Additionally, communication protocol suggestions to efficiently support the policies especially in highly loaded decentralized organized systems are included. Implementation of this kind of sequencing rules for load balancing in distributed computing system could result in a significant increase in performance especially in highly loaded systems where response times are a major issue. For better performance in parallel application load balancing is a key factor where tasks are created in dynamic fashion. In many computations, tasks have priorities, and solutions to the computation may be achieved more efficiently if these priorities are adhered to in parallel execution of the task. Sinha *et al.* in [32] described development of more efficient prioritized load balancing strategies that takes care of load imbalances, balances prioritized by centralizing queue, and balances memory requirement of processors by using tokens.

Increase in web traffic distributed multi server web site can provide scalability, and flexibility to cope with growing client demands. Many DNS (Domain Name Server) based schedulers have been proposed in literature, mainly for multiple homogeneous server. Heterogeneous web servers not only increase the complexity of DNS scheduling problem but also algorithm for homogeneous distributed systems not directly applicable. To propose new policies called adaptive TTL (Time to Live) algorithms that take into account of both the uneven distribution of client request rates and heterogeneity of web servers to adaptively set the TTL for each address mapping request. Colajanni *et al.* in [33] conclude that adaptive TTL strategies show low computational complexity and high robustness, and do not require many system state information. Bohn *et al.* in [34] find that when range of processing power is narrow, some benefits can be achieved with asymmetric load balancing. When the range of processing power is broad, dramatic improvement in performance are realized. Their experiments has shown unto 92% improvements when asymmetrically load balancing a modified version of the NAS parallel benchmarks' LU application on a heterogeneous cluster of Linux powered PCs. They conclude that the removal of older hardware is unnecessary even when the newer hardware has more than twice the performance.

Earlier for high performance computing network of workstation may be employed. In such an environment, partitioning a target job based on only the first order moments (means) of system parameter is not optimal. Lee *et al*. in [35] is proposed to consider the second order moments (standard deviation) also in load balancing in order to minimize execution time of target job on a set of work stations where the round robin job scheduling policy is adopted. It has been verified through computer simulation that the proposed static and dynamic load balancing scheme can be significantly reduce execution time of target job in a NOWs (Network of workstations) environment, compared to cases where only the means of the parameter are used.

In parallel computing and distributed system design the load balancing is a problem. León *et al.* [36] presented a new heuristic approach in assuring an optimal load balancing into a local computer network using cognitive behavioral modeling. They conclude that the simulation results are in good agreement with the proposed mode.

For parallel computation, Data partitioning and load balancing are important components. Many different partitioning strategies have been developed, with great effectiveness in parallel applications. New applications and architectures require new partitioning features. Existing algorithms must be enhanced to support more complex applications. Increased use of heterogeneous computing architectures requires partitioners that account for non-uniform computing, network, and memory resources. Devine *et al.* in [37] discuss their approaches to addressing these issues within the Zoltan Parallel Data Services toolkit. They conclude that the load-balancing problem is not yet solved. Developing data grids has become a major concern to make grids attractive for a wide range of data-intensive applications. Storage subsystems are most likely to be a performance bottleneck in data grids and therefore Qin [38] focuses to design and evaluate a data-aware load balancing strategy to improve the global usage of storage resources in data grids. He built a model to estimate the response time of job running at a local site or remote site. In light of this model, can calculate slowdowns imposed on jobs in a data grid environment. He propose a load-balancing strategy that aims to balance load of a in such a judicious way that computation and storage resources in each site are simultaneously well utilized. He conducted experiments using a simulated data grid to analyze the performance of the proposed strategy. Experimental results confirmed that their load-balancing strategy could achieve high performance for data-intensive jobs in data grid environments.

III. CRITICAL RESEARCH ISSUES FOR LOAD BALANCING

During our literature survey we explored that the techniques of load balancing are yet to be improved at many fronts. In this section we put our findings that will provide a template to the researchers interested in the area of the load balancing especially for the heterogeneous distributed computing environments.

Due to slow deployment of parallel programs, it is difficult to write efficient parallel programs. In parallel programming environment two points are important i) When will particular task be executed ii) Which processor will perform that task. Most of the parallelizable application do not have regular structure for efficient parallelization, therefore such type of applications require load balancing to perform efficiently. The load in these applications may change over time thus requiring rebalancing.

In recent years, new types of parallel computers have appeared. Networks of commodity workstations are making parallel computation available to group of researchers. Workstation networks present new issues for the application programmer. In addition to application imbalance, a parallel program must be concerned with background load from

other simultaneous users. Parallel programs may run on clusters of workstations on an interactive user's desks, where the primary user only permits parallel computation when the computer is not being used interactively. Finally, computational clusters may expand over time, but with the rapid increase in computational power, new processors are likely to be faster than the older machines that they are supplementing. To maximize throughput, load balancers in parallel applications must account for all these factors.

Work migration is a unified scheme for handling both application-specific and externally-arising load imbalance. The difficulty with migrating work is that either work is repartitioned in an application-specific way, placing the burden on the application programmer, or that automatic migration is supported, but with poor accuracy, due to the lack of application-specific knowledge.

Object migration provides a way of performing accurate and fine-grained automatic load balancing. Objects usually have small, well-defined regions of memory on which they operate, reducing the cost of migration. Using the Charm++ object model, the run-time system measures the work represented by particular objects, rather than deriving execution time from application-specific heuristics. Furthermore, the run-time system records object-to-object communication patterns, so the load balancer can asses the communication impact of migrating particular objects. We have developed strategies which take into account both the message sizes and the network hop length to minimize the total amount of communication.

Communication latencies form a significant factor in the performance of parallel applications on these large machines. The latencies are primarily due to network contention in the grid and torus networks, which are usually used in these large parallel machines. Our load balancing strategies minimize the impact of topology by heuristically minimizing the number of hops traveled by each communicated packet. They are not network specific and work for all classes of interconnection networks. Seed load balancing involves the movement of object creation messages, or seeds to create a balance of work across a set of processors. Several variations of strategies are being analyzed. In particular, we distinguish between global strategies, which may result in communication amongst all processors to exchange load information, and neighborhood strategies, which typically impose a dense graph organization on the processors, and restrict communication to neighbors only. Some strategies use averaging of loads to determine how seeds should be distributed, while others use receiver-initiated strategies, where a processor requests work from elsewhere when it is about to go idle. A strategy that places seeds randomly when they are created and does no movement of seeds thereafter is used as a baseline for comparison on numerous benchmarks.

IV. CONCLUSION AND FUTURE SCOPE

Load sharing is a simple and effective means of maximizing computational throughput in a distributed computing environment. By applying simple allocation strategies, the average response time is reduced. Load balancing or sharing environment increases system performance. Without load balancing, some processors in the system are idle while other processors having many ready tasks waiting to run when the average processor utilization is just below 100%. Load balancing functions make use of the processor resource more efficiently and produce great improvement under this condition. When the processor utilization is close to 100%, the processors are always busy. Future study includes consideration of communication among subtasks and performance analysis for real workloads on network of workstations. A researcher may focus on optimal achievement of object migration, work migration, seed load balancing and communication latency.

REFERENCES

1. A. S. Tanenbaum, "Distributed Systems," *Prentice Hall*, NY, 1993.

2. I. Hamburg, M.-H. Zaharia, "A Strategy for Multimedia-based Distributed Applications in Telecooperation," *EURASIP Conf.: ECMCS 2001*, Budapest, Hungary, 11-13, 2001.

3. M. J. Berger, S. H. Bokhari, "A partitioning strategy for non uniform problems on multiprocessors," *IEEE Trans. Computers C-36 (5) (1987) 570-580*.

4. H. D. Simon, "Partitioning of unstructured problems for parallel processing," *In Proc. Conference on Parallel Methods on Large Scale Structural Analysis and Physics Applications*, Pergammon Press, 1991.

5. V. E. Taylor, B. Nour-Omid, "A study of the factorization Fill-in for a parallel implementation of the finite element method," *Int. J. Numer. Meth. Engng. 37(1994) 3809-3823.*

6. M. S. Warren, J. K. Salmon, "A parallel hashed oct-tree n-body algorithm," *In Proc. Supercomputing '93, Portland, OR, 1993.*

7. J. R. Pilkington, S. B. Baden, "Partitioning with space Filling curves," *CSE Technical Report CS94-349, Dept. Computer Science and Engineering*, University of California, San Diego, CA (1994).

8. A. Patra, J. T. Oden, "Problem decomposition strategies for adaptive hp Finite element methods, *Computing Systems in Engg". 6 (2) (1995) 97-109.*

9. W. F. Mitchell, "Refinement tree based partitioning for adaptive grids," *In Proc. Seventh SIAM Conf. on Parallel Processing for Scientific Computing, SIAM, 1995, pp. 587-592.*

10. J. Flaherty, R. Loy, M. Shephard, B. Szymanski, J. Teresco, L. Ziantz, "Adaptive local refinement with octree load-balancing for the parallel solution of three-dimensional conservation laws," *J. Parallel Distrib. Comput. 47 (2) (1998) 139-152.*

11. A. Pothen, H. Simon, K. Liou, "Partitioning sparse matrices with eigenvectors of graphs," *SIAM J. Matrix Anal. 11 (3) (1990) 430-452.*
12. T. Bui, C. Jones, "A heuristic for reducing Fill in sparse matrix factorization," *In Proc. 6th SIAM Conf. Parallel Processing for Scientific Computing, SIAM, 1993, pp. 445-452.*

13. B. Hendrickson, R. Leland, "A multilevel algorithm for partitioning graphs," *In Proc. Supercomputing '95, ACM, 1995.*

14. [14] G. Karypis, V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *Tech. Rep. CORR 95-035, University of Minnesota, Dept. Computer Science, Minneapolis, MN (June 1995).*

15. G. Cybenko, "Dynamic load balancing for distributed memory multiprocessors," *J. Parallel Distrib. Comput. 7 (1989) 279-301.*
16. Y. Hu, R. Blake, "An optimal dynamic load balancing algorithm," *Tech. Report DL-P-95-011, Daresbury Laboratory,* Warrington, WA4 4AD, UK (Dec. 1995).

17. E. Leiss, H. Reddy, "Distributed load balancing: design and performance analysis," *W.M. Keck Research Computation Laboratory 5 (1989) 205-270.*

18. S. Plimpton, S. Attaway, B. Hendrickson, J. Swegle, C. Vaughan, D. Gardner, "Transient dynamics simulations: Parallel algorithms for contact detection and smoothed particle hydrodynamics," *J. Parallel Distrib. Comput. 50 (1998) 104-122.*

19. H. C. Edwards, "A Parallel Infrastructure for Scalable Adaptive Finite Element Methods and its Application to Least Squares C1 Collocation," *Ph.D. thesis, The University of Texas at Austin (May 1997).*

20. K. Devine, J. Flaherty, "Parallel adaptive hp-refinement techniques for conservation laws, *Appl. Numer. Math. 20 (1996) 367-386.*

21. K. Schloegel, G. Karypis, V. Kumar, "Multilevel diffusion algorithms for repartitioning of adaptive meshes," *Journal of Parallel and Distributed Computing 47 (2) (1997) 109-124.*

22. C.Walshaw, M. Cross, K. McManus, "Multiphase mesh partitioning," *App. Math. Modelling 25 (2000) 123-140.*

23. K. Schloegel, G. Karypis, V. Kumar, "Parallel static and dynamic multiconstraint graph partitioning, Concurrency and Computation," - *Practice and Experience 14 (3) (2002) 219-240.*

24. B. Hendrickson, R. Leland, "The Chaco user's guide, *version 2.0, Tech. Rep. SAND94-2692, Sandia National Laboratories, Albuquerque, NM (Oct. 1994).*

25. C. Walshaw, "The Parallel JOSTLE Library User's Guide," Version *3.0, University of Greenwich, London, UK (2002).*

26. R. Preis, R. Diekmann, "The PARTY partitioning library, user guide version 1.1," *Tech. Rep. tr-rsfb-96-024, Dept. of Computer Science, University of Paderborn,* Paderborn, Germany (Sept. 1996).

27. F. Pelligrini, "SCOTCH 3.4 user's guide," *Research Rep. RR-1264-01, LaBRI (Nov. 2001).*

28. G. Karypis, V. Kumar, "ParMETIS: Parallel graph partitioning and sparse matrix ordering library," *Tech. Rep. 97-060, Department of Computer Science, University of Minnesota,* available on the WWW at URL http://www.cs.umn.edu/~metis (1997).

29. H. Moons, P.Verbaeten and U. Holberg , "Distributed computing in Heterogeneous Environment," *In EUUG Spring '90 Conference Proceedings.*[30] A. Bond, "Load Sharing in a Distributed Environment," *In proceedings of Uniforum NZ '92, New Plymouth, 14-16 May 1992.*

30. A. Winckler, "Context Sensitive Load Blancing in Distributed Computing Systems," 1993 — *Proc. ISCA Int. Conf. on Computer Applications in Industry and Engineering*

31. A.B. Sinha L.V. Kale, "A load balancing strategy for prioritized execution of tasks," *International [1]Symposium on Parallel Processing, Newport Beach, CA, April 1993.*

32. M. Colajanni, S. Philip, "Dynamic Load Balancing in Geographically Distributed Heterogeneous Web Servers," *Proceedings of The 18th International Conference on Distributed Computing Systems 1998.*

33. C.A. Bohn, G.B. Lamont, "Asymmetric Load Balancing on a Heterogeneous Cluster of PCs," In *Proceedings of PDPTA'1999. pp. 2515-2522.*

34. S.-Y Lee, C.-H Cho, "Load Balancing for Minimizing execution time of target Job on a Network of Heterogeneous Workstations," *In: Proceedings of IASTED International Conference on Networks, Parallel and Distributed Processing, and Applications (NPDPA 2002).*

35. F. Leon M. H. Zaharia, D. Galea, "Load Balancing in Distributed Systems using cognitive Behavioral Models," *Bulletin of Technical University of lasi, tome XLVIII(LII), fasc. 1-4, pp 119-124,2002.*

36. K. D. Devine, E. G. Boman, R. T. heaphy, B. A. Hendrickson, J. D. Teresco, J. Faik, J. E. Flaherty, L. G. gervasio, "New Challenges In Dynamic Load Balancing" *In Applied Numerical Mathematics 52(2005).*

37. X. Qin, "Design and Analysis of Load Balancing Strategies in Data Grids," *In Future Generation Computer Systems: vol. 23, no. 1 pp. 132-137. Jan 2007.*

38. Parallel Programming Laboratory, *Department of Computer Science, University of Illinois at Urbana Champaign.*

