Scientific Research Publishing

Journal of Software Engineering and Applications, 2017, 10, 777-786
http://www.scirp.org/journal/jsea
ISSN Online: 1945-3124
ISSN Print: 1945-3116

Application Resource Management for Highly Computational Applications in the Operational Environment: A Critical Review

Joseph Balikuddembe, Jael Gudu

Department of Networks, College of Computing and Information Sciences, Makerere University, Kampala, Uganda
Email: jbalikud@cis.mak.ac.ug, jaelgudu@mu.ac.ke

How to cite this paper: Balikuddembe, J. and Gudu, J. (2017) Application Resource Management for Highly Computational Applications in the Operational Environment: A Critical Review. *Journal of Software Engineering and Applications*, **10**, 777-786.
https://doi.org/10.4236/jsea.2017.109043

Received: December 22, 2016
Accepted: August 14, 2017
Published: August 17, 2017

Abstract

Computational resources have such a significant influence on the operation of any software application, it is therefore important to understand how these applications utilize these resources. Modern resource-intensive enterprise and scientific applications are creating a growing demand for high performance computing infrastructures. They constantly interact with and rely heavily on complex resources. However, they often operate in resource-limited environments yet they often handle massive data, both in size and complexity. Software application services, processes or transactions compete for the much required but scarce resources. This creates the need to improve the existing resource allocation and management issue in such operational environments, as well as propose new ones, if necessary. Software developers try to analyze application operation environment using diverse analysis and design methods. Our aim therefore, is to design a tool that is able to work with a hybrid of adaptive and prediction-based resource management and allocation models while applying the priority based job scheduling algorithm to try and solve the application resource management challenges currently being faced in such environments, even if, partially.

Keywords

Resource Management, Resource-Intensive Applications, Resource Allocation, Design

1. Introduction

There is growing interest in improving the resources utilization, such as memory and processor time efficiency of large scale enterprise applications [1]. A system

resource can be defined as a provider of a set of capabilities and they entail a network, hardware (memory) and software [2] that are needed to execute any transaction initiated in and by the application. "[1]" notes that memory usage (as a resource) is perhaps the most important factor in system performance. In support [3] argues that processor time and memory resources have such a significant influence on the operation of any software application, it is therefore important to understand how these applications utilize them. Continued expansion of software applications represents the need for more of such resources [4].

Resources should be distributed among the different application transactions in such a way that there should not be starvation [5]. Monitoring their utilization helps software engineers know how applications make demands on the resources and how those resources respond to each job or transaction [3]. This is what resource management is about; monitoring the availability of system resources, allocating the resources and provisioning of the resources [6] as demand arises. In this regard, interest is in controlling how capabilities provided by system resources and services are made available to other entities, whether users, applications, services. For improper resource management is susceptible to the underutilized and wastage of resources. The net result on user application experience is poor service delivery in these applications and data centers. [7] mentions that it is desirable to not just avoid wasting resources as a result of underutilization, but to avoid lengthy response times as a result of over-utilization.

There are many ways to provide resource management controls proposed by various researchers, we categorize them as: 1) resource management Middleware techniques; 2) Prediction-based resource allocation techniques [8] [9]; 3) Web services; and 4) Virtualization Technology. However, our study proposes a self-adaptive and prediction-based resource allocation mainly because they allow for resource allocation at an application level. The operational environment of the used is a disease surveillance environment. However, it is important to note that there is no single technique or method which provides the complete solution for transaction and resource management challenges in dynamic operational environment in which these disease surveillance applications are deployed. Even so, this does not negate the fact that software developers are trying to analyze application operation environment using diverse analysis and design methods [10]. Methods here include the different ways of monitoring a utilization amount of resources within a server and identifying a resource-strained server to avoid an activation of capacity upgrade on demand (CUoD) [11].

2. Background

2.1. Resource-Intensive Applications

Modern resource-intensive enterprise and scientific applications are creating a growing demand for high performance computing infrastructures [12]. These applications constantly interact with and rely heavily on complex resources and are all stretching the limits of the organization's existing resources [13]. Such

applications are characterized by complex processes [14] that sometimes need to run for extended periods of time [15].

Resource intensive applications can be either data-intensive or compute-intensive applications [14]. Data-intensive applications handle massive amounts of data commonly referred to as "big data". Another category is the user-intensive applications, an aspect introduced by [16]. Such include web applications that must satisfy the interaction requirements of thousands if not millions of users, which can be hardly fully understood at design time. Both data-intensive and user-intensive software applications could be considered as compute-intensive applications. This is because they often require continuously increasing power of computing resources and storage volume that are in many cases required on-demand for specific operations in data lifecycle [17]. This therefore calls for a capacity planning approach as suggested by [18] in most organizations that are not in a position to purchase additional computational resources each time need arises due to financial constrains.

2.2. Transaction Processing in Resource-Intensive Applications

Transaction processing is one of the essential features of such an enterprise application, since they perform computationally intensive work [15] [19]. For the transaction processing to happen efficiently and effectively, considerable amounts of resources are required to complete execution in a reasonable timeframe [14]. Each time a transaction is initiated it creates the demand for a resource [19] and there are instances where two or more workloads can place a demand for the resources at the same time. We deduce that for the extended time such a workload is running it will be assigned available resources and will consume the resources until its execution is complete which could lead to other applications being denied or put on hold (waiting) since the resource is committed elsewhere. The resources must therefore be sliced and shared between machines running potentially heterogeneous workloads [6] [20], consisting of many short jobs (transactions) and a small number of large jobs (transactions) that consume the bulk of the available resources, without incurring extra resource and performance costs [21] [22]. That is why it is important to determine how best to maximize the performance of the system even in such circumstances. It is also important that software applications must have the ability to self-adapt to meet changes in such execution environment and yet still maintain expected qualities-of-service [23].

3. Design Concerns and Methods

3.1. Design Concerns for Resource-Intensive Applications

Insufficient attention to workloads and the interactions of the applications with the available (limited) resources at the application design time can reduce the quality and effectiveness of the software [13], yet there is an increasing demand, from the users, for software products with increasing quality [24]. We infer that

for any application design, it is important to understand the management of the application transactional processes and its implication within its operational environment. Accordingly, scheduling of transactions and management of their execution time are important performance requirements of any enterprise application [25] [26].

Designing for such situations is a challenging task because resource management in resource limited/constrained environment, in itself, is a hard problem due to the following: the large scale nature and complexity of such applications [27] [28] [29] [30]; the heterogeneity of resource types and their interdependencies; the variability and unpredictability of the load [2] [28]. Failure to interpret different complexities that can arise in the course of an application development within the planning process can result in huge bottlenecks. This therefore means that a lot of attention should go to the planning of the architecture [31] [32], and that is where the design comes in.

The underlying goal of any design process or decisions that are to be incorporated in the final architecture is ensuring that quality attribute are achieved [33] since these attributes are used as a bridge to connect business goals and software architectures [32]. In planning the components and how they will relate, it is important work out how the different categories of the heterogeneous workloads will be processed [21] and that is where the method comes in. There is need to assign available local resources for these applications which is capable of executing on it [6], but in order to improve resource utility, resources must be properly allocated and load balancing must be guaranteed [34]. It is also important to note that the resource demand of an application can change over time [6] and a software application designer needs to factor in such changes in the design plan. The analysis drawn from this is that resource management in transaction processing and management is an important design decision and concern in the software engineering discipline.

3.2. Design Methods for Resource-Intensive Applications

Define There are different techniques are being used to target specific design and implementation issues in the dynamic environment. "[35]" argues that there is always a trade-off in terms of performance, availability, consistency, concurrency, scalability and elasticity. Virtualization focuses heavily on the physical level of resource allocation for the transactions [36], while web services, such as Elastic Compute Cloud (EC2) adopted by [37], focuses on involving the users by giving them a chance to be able to monitor the allocation of resources and applications and also monitor the custom metrics generated by a customer's applications and service, an aspect that will not be covered in this work. "[38] [39]" Present adaptive resource management middleware techniques that achieve the QoS requirements of the system.

The middleware performs QoS monitoring and failure detection, QoS diagnosis, and reallocation of resources to adapt the system to achieve acceptable levels

of QoS; for an application to be able to adaptive applications needs to be self-aware, resource-aware and also context aware [40]. This means it should be able to understand its current environment of operation and the current circumstances, and the resources available [41]. Prediction and Forecasting should be important components of many real-world enterprise applications [42]. Such capabilities help in reducing the risk of making wrong decisions while allocating resources to transactions [9]. Our aim therefore, is to be able to work with a hybrid of adaptive and prediction-based resource management and allocation models to try and solve the application resource management challenges currently being faced in such environments, even if, partially. The resulting tabulation is an on demand plan (type of resources provision plan based on needs of each transaction or workload) [5] identified using the hybrid model.

The on demand plan, such as the one shown in **Table 1**, helps ensure that the software engineers are fully aware of the types of jobs to be executed and the resource demand for each transaction during the design of the application and the modules for the application.

3.3. Workload Classification and Execution

Priority based job scheduling algorithm proposed by [43] [44] to help us identify the level of priority of each job. "[43]" mentions that in a workload there are some jobs that are executed prior to others depending on the nature of the job. Disease surveillance has the following components that are used to classify task: case detection, data recording, data compilation, and data transmission. Transactions from disease surveillance applications can be within any of these components and there must be a provision that all resources are made available to requesting users in efficient manner to satisfy their needs. But the question is how? Of the components mentioned which one should be executed before the other?

For example, the graph in **Figure 1** indicates that the most urgent task is given the available shortest execution time, while the lease urgent task will be executed within the last 3.5 time block. It is therefore important have a criteria that can

Table 1. Sample on demand plan.

Transaction	Network	CPU	Memory
T1 (Retrieve user details)	Demand 45% of network in 10 minutes	Currently utilizing (20%)	Demand (30% of memory in 15 minutes)
T2 (Upload updated information)	Demand (In 30 minutes needs 60% of network)	Demand (30% of CPU in 5 minutes)	Currently utilizing (50% job will release resource in 15 minutes)
T3 (Execute natural language processing)	Currently utilizing (80% will release in 20 minutes)	Currently utilizing (30%)	Demand (10% of memory in 2 minutes)

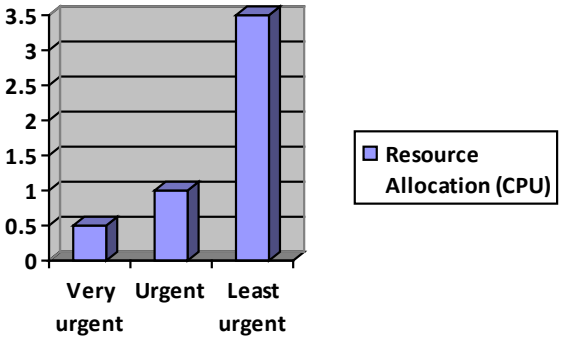

Figure 1. Sample job mapping based on urgency of output.

be used to initiate the tasks and then executing the tasks when the criteria is met, a process known as task scheduling; a vital part that assigns tasks to suitable resources for execution. "[9]" proposes classification criteria that entail the following: Performance goals, Resource requirements and Business importance.

This approach is important in task classification where the concern is in adding value to a business process as shown in Table 2. For example in a disease surveillance environment, based on the output it will be improvement in patient monitoring where high risk conditions. However, it leaves out the element of understanding the nature of the task or workload and how that will contribute to the overall performance of an application; yet this study infers that if an application is "heavy" dealing with large quantities of data it will consume more resources and will in turn affect the performance of the application.

Table 2. Sample task classification.

Task	Performance goals	Resource requirements	Business importance
T1 (case detection)	Will require more resources, but might be transmitted or shared faster	- 50% CPU - 10% Bandwidth - 80% Memory and Storage	Important in single patient monitoring or disease surveillance
T2 (data recording)	Will require little resources, depending on amount being recorded at any given cycle or time	- 20% CPU - 5% Bandwidth - 10% Memory and Storage	Important for information preservation
T3 (data transmission)	Will require little resources, depending on amount being transmitted at any given cycle or time	- 10% CPU - 2% Bandwidth - 5% Memory and Storage	Important for information sharing

"[36]" proposes an approach that considers the nature and characteristics of the job. They mention that the following three characteristics can be used: Volume: This refers to the quantity of data which is generated and it determines the size only; Variety: This specifies the category to which the data belong for example in a disease surveillance environment the categories are (case identification, case reporting etc); and Velocity: It specifies the speed at which the data is generated for example social media data comes in fast while hospital data comes in on either a weekly or monthly basis. The challenge with using this approach is

that it is fully focused on the job leaving out other important things that need to be considered such as the available resources. "[37]" describes an approach to workload classifications based on task resource consumption needs and patterns; such as the time needed to execute the job (short or long duration), CPU and memory demand. However, the challenge with this approach according to is that it does not perform intra-cluster analysis to derive a detailed workload model and it also neglects the user patterns which are as important as the tasks in the overall workload model.

4. Conclusion

Achieving efficient resource allocation is one of the most challenging problems. But the question is how to choose a method that will give efficient resource allocation and management. This study has proposed a self adaptive resource aware approach which will give a better application capacity planning view to the software engineers during the design of the applications. It is important that software engineers know how applications make demands on the resources and how those resources respond to each job or transaction considering the fact that the computational resources are scarce. A priority based scheduling approach will help in ensuring that jobs which are considered most urgent, especially where disease surveillance is concerned, are given the priority that they deserve in terms of execution. This is because each job will be classified based on its nature and business value. This however does not mean that all resources will be committed to jobs considered "most important or urgent", but the application will avail tasks based on how much resources need an aspect that is very much needed.

References

[1] Urgaonkar, R., *et al.* (2010) Dynamic Resource Allocation and Power Management in Virtualized Data Centers. *Network Operations and Management Symposium* (*NOMS*), 2010 *IEEE*, Osaka, 19-23 April 2010, 479-486. https://doi.org/10.1109/NOMS.2010.5488484

[2] Ngenzi, A., *et al.* (2015) Dynamic Resource Management in Cloud Datacenters for Server Consolidation.

[3] Microsoft (2016) Watching How Programs Use System Resources. https://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/sag_mpmonperf_14.mspx?mfr=true

[4] Yang, C. (2010) Green Power Management with Dynamic Resource Allocation for Cloud Virtual Machines. *IEEE* 13*th International Conference on High Performance Computing and Communications* (*HPCC*), Banff, 2-4 September 2011, 726-733.

[5] Beeda, K. and Manikandan, A.R. (2015) Secure and Optimal Resource Allocation in Cloud Environment for Media Streaming Applications. *International Journal of Research and Engineering*, **2**, 68-73.

[6] K.L.A. and Suresh, S. (2015) A Survey on Dynamic Resource Management Technologies in Cloud Datacenter. *International Journal of Advanced Research in Computer and Communication Engineering*, **4**.

[7] Yazir, Y.O. (2010) Dynamic Resource Allocation in Computing Clouds Using Distributed Multiple Criteria Decision Analysis. *IEEE 3rd International Conference on Cloud Computing*, Miami, 26 August 2010, 91-98.

[8] Jokhio, F. (2013) Prediction-Based Dynamic Resource Allocation for Video Transcoding in Cloud Computing. 21*st Euromicro International Conference*, Belfast, 27 February-1 March 2013, 254-261.

[9] Gorde, B.P.A., Gandhi, N., Mishra, R. and Pathak, R. (2016) Prediction Based Outcome for Media Streaming Applications. *International Journal of Computer Applications*, **1**, 11-15.

[10] Amid, A. and Moradi, S. (2013) A Hybrid Evaluation Framework of CMM and COBIT for Improving the Software Development Quality. *International Journal of Software Engineering and Knowledge Engineering*, **6**, 9. https://doi.org/10.4236/jsea.2013.65035

[11] Dawson, C. (2013) Method and System for Cost Avoidance in Virtualized Computing Environments. http://www.google.com/patents/US8347307

[12] Beloglazov, A. and Buyya, R. (2010) Energy Efficient Resource Management in Virtualized Cloud Data Centers. 10*th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, Melbourne, 17-20 May 2010, 826-831.

[13] Shin, S.Y. (2015) Resource Specification for Prototyping Human-Intensive Systems. *International Conference on Fundamental Approaches to Software Engineering*, 332-346. https://doi.org/10.1007/978-3-662-46675-9_22

[14] Buyya, R. and Vecchiola, C. (2013) Mastering Cloud Computing. Tata McGraw Hill Education Private Limited, New York.

[15] IBM (2016) Compute-Intensive Programming Mode. https://www.ibm.com/support/knowledgecenter/SSAW57_8.0.0/com.ibm.websphere.nd.doc/info/ae/ae/cgrid_schintensve.html

[16] Ghezzi, C. (2014) Mining Behavior Models from User-Intensive Web Applications. *Proceedings of the 36th International Conference on Software Engineering*, Hyderabad, 31 May-7 June 2014, 277-287. https://doi.org/10.1145/2568225.2568234

[17] Demchenko, Y. (2016) CYCLONE: A Platform for Data Intensive Scientific Applications in Heterogeneous Multi-Cloud/Multi-Provider Environment. *IEEE International Conference on Cloud Engineering Workshop* (*IC2EW*), 4-8 April 2016, Berlin, 154-159.

[18] Huang, C., He, D. and Miao, B. (2014) A Survey of Resource Management in Multi-tier Web Applications. *IEEE Communications Surveys & Tutorials*, **16**, 1574-1590.

[19] Mihindukulasooriya, R.C.N. and Gutiérrez, M.E. (2014) Seven Challenges for RESTful Transaction Models. *Proceedings of the 23rd International Conference on World Wide Web*, Seoul, 7-11 April 2014, 949-952. https://doi.org/10.1145/2567948.2579218

[20] Halevy, A.Y., *et al.* (2005) Enterprise Information Integration: Successes, Challenges and Controversies. *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, Baltimore, 14-16 June 2005, 778-787. https://doi.org/10.1145/1066157.1066246

[21] Deldago, P. (2015) Hawk: Hybrid Datacenter Scheduling. *Proceedings of the USENIX Annual Technical Conference*, Santa Clara, 8-10 July 2015, 499-510.

[22] Gember, A.A.A. and Dragga, C. (2012) ECOS: Practical Mobile Application Offloading for Enterprises. *Proceedings of the 2nd USENIX conference on Hot Topics*

in Management of Internet, Cloud, and Enterprise Networks and Services, San Jose, 4 April 2012, 4-4.

[23] Cheng, S.-W. and Garlan, D. (2012) Stitch: A language for Architecture-Based Self-Adaptation. *Journal of Systems and Software*, **85**, 2860-2875. https://doi.org/10.1016/j.jss.2012.02.060

[24] Miguel, G.R.J.P. and Mauricio, D. (2014) A Review of Software Quality Models for the Evaluation of Software Products. *International Journal of Software Engineering & Applications*, **5**. https://doi.org/10.5121/ijsea.2014.5603

[25] Lin, W. (2011) A Threshold-Based Dynamic Resource Allocation Scheme for Cloud Computing. *Procedia Engineering*, **23**, 695-703. https://doi.org/10.1016/j.proeng.2011.11.2568

[26] Singh, Y.J. (2010) Dynamic Management of Transactions in Distributed Real-Time Processing System.

[27] Gerostathpoulous, L. (2016) Self-Adaptation in Software-Intensive Cyber-Physical Systems: From System Goals to Architecture Configurations. *Journal of Systems and Software*, **122**, 378-397. https://doi.org/10.1016/j.jss.2016.02.028

[28] Jennings, B. and Stadler, R. (2013) Resource Management in Clouds: Survey and Research Challenges. *Journal of Network and Systems Management*, **23**, 567-619. https://doi.org/10.1007/s10922-014-9307-7

[29] Aktunc, O., Erol, B.A. and Garcia, J.D. (2012) Redesign of a Seismic Monitor Using Contextual Design. *International Journal of Software Engineering & Applications*, **3**. https://doi.org/10.5121/ijsea.2012.3601

[30] Njeru, M.E. (2014) Software Frameworks, Architectural and Design Patterns. *Journal of Software Engineering and Applications*, **7**, 670-678. https://doi.org/10.4236/jsea.2014.78061

[31] Balikuddembe, J.K., Osunmakinde, I.O. and Bagula, A. (2008) Software Project Profitability Analysis Using Temporal Probabilistic Reasoning: An Empirical Study with the CASSE Framework. *International Conference on Security Technology*, 138-150.

[32] Tan, L., Lin, Y. and Ye, H. (2012) Quality-Oriented Software Product Line Architecture Design. *Journal of Software Engineering and Applications*, **5**, 472-476. https://doi.org/10.4236/jsea.2012.57054

[33] Otero, C. (2012) Software Design Challenges. http://www.ittoday.info/ITPerformanceImprovement/Articles/2012-06Otero.html

[34] Balan, R.V.S. and Punithavalli, M. (2011) Decision Based Development of Productline: A Quintessence Usability Approach. *Journal of Scientific Computing*, **7**, 619-628. https://doi.org/10.3844/jcssp.2011.619.628

[35] Waqas, A. (2015) Transaction Management Techniques and Practices in Current Cloud Computing Environments: A Survey. *Int. J. Database Manag. Syst.*, **7**.

[36] Lee, E.K., Viswanathan, H. and Pompili, D. (2015) Proactive Thermal-Aware Resource Management in Virtualized HPC Cloud Datacenters. *IEEE Transactions on Cloud Computing*, **5**, 234-248. https://doi.org/10.1109/TCC.2015.2474368

[37] Kaleeswari and Juliet, N.M. (2014) Dynamic Resource Allocation by Using Elastic Compute Cloud Service. *International Journal of Innovative Research in Science, Engineering and Technology*, **3**, 12375-12379.

[38] Ravindran, B., Welch, L. and Shirazi, B. (2011) Resource Management Middleware for Dynamic, Dependable Real-Time Systems. *Real-Time Systems*, **20**, 183-196.

[39] Dharaskar, R.V., Tondre, V.S., Thakare., V.M. and Sherekar, S.S. (2011) Trends and

Prospectives of the Dynamic Resource Management Using Adaptive Techniques in Distributed System. *International Journal of Computer Science and Telecommunications*, **2**, 68-77.

[40] Becker, T. (2012) EPiCS: Engineering Proprioception in Computing Systems. 15*th International Conference on Computational Science and Engineering* (*CSE*), *2012 IEEE*. Nicosia, 5-7 December 2012, 353-360.

[41] Miraoui, M. (2011) Dynamic Context-Aware and Limited Resources-Aware Service Adaptation for Pervasive Computing. *Advances in Engineering Software*, **2011**, 11. https://doi.org/10.1155/2011/649563

[42] Wagner, N. (2011) Intelligent Techniques for Forecasting Multiple Time Series in Real-World Systems. *International Journal of Intelligent Computing and Cybernetics*, **4**, 284-310. https://doi.org/10.1108/17563781111159996

[43] Kumar, R.S. and Rekha, S. (2014) Priority Based Job Scheduling For Heterogeneous Cloud Environment. *International Journal of Computer Science Issues*, **11**, 114.

[44] Blessie, R. and Stanislas, A. (2014) A State of Art Scheduling Algorithms in Cloud Environment. *International Journal of Advance Research in Computer Science and Technology*, **2**, 476-481.