Research Statement

Nilabja Roy

August 18, 2010

1 Doctoral Dissertation

Large-scale, component-based distributed systems form the backbone of many service-oriented applications, ranging from Internet portals in the enterprise domain to shipboard computing in the military domain. These applications must be designed to handle intense (and often bursty) client workloads, while satisfying stringent quality of service (QoS) requirements, including bounded client response times and high availability. From the service provider perspective, however, it is important to keep both the procurement and operational costs of the system resources low while improving the revenues. Consequently, deploying large-scale, component-based distributed systems can be formulated as a utility maximization problem that handles increased client workload while still assuring their QoS requirements, yet consuming fewer resources.

To maximize utility, its important to identify the average resource requirements of an application and then provision it accordingly. Average resource requirements, however, depend on a complex combination of workload, hardware, and application design [1, 2] and is thus difficult to compute. It is comparatively easier to determine the worst-case conditions for an application and compute its resource requirements.

Traditional approaches to resource allocation use analytical techniques to provision distributed systems resources. These approaches have generally focused on worst-case workload scenarios, which underutilizes resources and increases cost. A promising solution to this problem is auto-scaling of cloud computing resources, where resources can be increased or decreased as load increases or decreases. Implementing such a strategy, however, requires a function that maps the expected workload to resource requirements of applications. Moreover, this function must be accurate since otherwise resources provisioned using a flawed function will result either in idle resource (underutilized system) or in performance degradation (over-utilized system).

Developing the desired function is a hard problem for distributed systems due to a complex combination of issues, including the code (legacy, proprietary and third-party), operating layers (operating system, virtual machines, middleware) and hardware (multiple cores and processors). Traditional analytical modeling methods that map workload to resource requirement assume a simple view of the system and do not account for all relevant issues.

To overcome the limitations of prior work, in my research has developed an analytical modeling method called MAQ-PRO (Modeling and Analysis using Queuing, Placement and Replication Optimizations) . MAQ-PRO includes a correction factor and variable service time that captures systemic issues, such as complex coding, application layering, and hardware complexity. Results from extensive experiments [3] with MAQ-PRO indicate that the value of the correction factor and service time change with system utilization and thus affect application performance. Extensive profiling was conducted to estimate the correction factor and the service times [4].

The correction factor and the service time was then used with an enhanced version of the mean-value analysis (MVA) algorithm. MVA traditionally estimates performance parameters for given amount of resources and workload under simple scenarios. I extended MVA to use the correction factor and variable service times described above so that performance estimation is accurate in complex applications with multiple layers of code and using multiple cores/processors. The enhanced MVA algorithm can identify resource requirements for various different workload conditions and performance requirement.

Once an accurate analytical model is developed, it can be used with resource allocation algorithms (such as bin-packing based placement algorithms [5]) to identify a deployment strategy that maximizes utility. MAQ-PRO also contains a capacity planning method that uses the analytical models to determine component placement strategies. MAQ-PRO enhances traditional bin-packing strategies [4] by incorporating component-based software design to guide the placement of application components onto the computers to minimize resource utilization. Compared to conventional monolithic, tiered application architectures, the MAQ-PRO technique leverages the smaller granularity of components to increase flexibility in resource partitioning and utilization [6].

2 Ongoing Work and Future Directions

When a complex distributed application runs in a heavily loaded environment many factors influence its performance, including background jobs, workload variation, and minor faults. The assumptions made at static time (such as the correction factor described above) may no longer remain valid. What is needed, therefore, are analysis techniques that can handle these dynamic effects.

I have conducted experiments that indicate how multiple factors contribute to changes in runtime performance. For example, in an experiment involving a large-scale database application showed that as workload increases, the number of threads increase along with the amount of system activity, such as context switches, caching, swapping etc. This excess system activity also increases the processor and memory usage. It is essential to determine how workload causes an increase in the various system activities and how that in turn consume resources, such as CPU and memory.

My current research focuses on using machine learning techniques to relate workload and disturbances, such as background work or minor faults to system activity and in turn to resource consumption. The following two-phased learning steps are needed:

1. **Relate system activity to resource consumption**. In the first phase each system activity factor (such as context switches, swapping, or paging) can be modeled as an input feature while resource consumption is the target or output variable. A model selection process can be performed to quantify the effects of the various input features on resource consumption. Some features may be more dominant than others, *e.g.*, the number of context switch may represent the entire system activity, rather than considering all other factors. After the dominant factors are identified, regression models can be developed to characterize resource consumption as a function of system activity.

2. **Relate workload to system activity.** In the second phase system activity is related to workload and other disturbances. Disturbances can be assumed as changes in workload. Relating workload to system activity should be straightforward. A linear or piecewise linear function should suffice. The purpose would be to keep the model simple.

The results of applying these two phases point is an model that accurately characterizes workload to resource consumption at runtime. This model can be used to optimize resource usage in cloud computing environments. Application can be deployed with the minimum amount of resources using a static capacity planning method described in Section 1. As the application runs, online monitoring [7, 8, 9] can be used to refine the analytical models in the presence of dynamic factors, as described above. After workload increase is anticipated , online capacity planning can be performed to estimate the required increases in resources. These resources could be kept in a passive state to save power until workload actually increases.

As workload decreases, it may be necessary to release resources to reduce cost of resource usage.... As before, the model devised above can be applied to determine which resources to release to minimize excess costs. Combining these modeling techniques with elastic cloud computing provisioning mechanisms helps to systematically optimize resource usage and minimize cost.

3 Research Summary

Table 1 summarizes my research contributions and Table 2 summaries my publication record, which are classified according to individual topics that form my overall research portfolio.

References

[1] N. Roy, A. Dabholkar, N. Hamm, L. Dowdy, and D. Schmidt, "Modeling Software Contention using Colored Petri Nets," in *16th Annual Meeting of*

Table 1: Summary Of Research Contributions

Category	**Contributions**
Component Resource Requirement Identification	TargetManager: design and implementation of (1) distributed profiling framework, (2) implementation of profiling techniques for component resource profiling, and (3) customer behavior modeling for overall component resource requirement
Performance Estimation of Software Components	(1) Queuing theoretic models for large scale multi-tiered internet applications, (2) more accurate analytic models for high utilization, software contention and multiple processors/cores, (3) simulation modeling of multi-threaded application with software contentions and (4) markov chain modeling of soft real time systems
Application Component Placement	(1) Detailed comparative study of multiple bin-packing heuristics, (2) design and development of component placement heuristic based on worst-fit bin packing and (3) development of component replication and placement heuristic based on worst-fit bin packing.

the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), Baltimore, MD, Sep. 2008.

[2] F. Wolf, J. Balasubramanian, A. Gokhale, and D. C. Schmidt, "Component Replication based on Failover Units," in *Proceedings of the 15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA '09)*, Aug. 2009, pp. 99–108.

[3] A. G. Nilabja Roy and L. Dowdy, "A Novel Capacity Planning Process for Performance Assurance of Multi-Tiered Web Applications," in *To Appear in the Poster Proceedings of the 18th Annual Meeting of the IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '10)*. Miami Beach, FL, USA: IEEE, Aug. 2010.

[4] G. A. Roy, Nilabja and L. Dowdy, "A Capacity Planning Process for Performance and Availability Assurance of Multi-Tiered Web Applications," in *Submitted to HASE '10: Proceedings of The 12th IEEE International High Assurance Systems Engineering Symposium, San Jose, CA*, 2010.

[5] N. Roy, J. S. Kinnebrew, N. Shankaran, G. Biswas, and D. C. Schmidt, "Toward Effective Multi-capacity Resource Allocation in Distributed Real-time and Embedded Systems," in *Proceedings of the 11th International Symposium on Object/Component/Service-oriented Real-time Distributed Computing)*. Orlando, Florida: IEEE, May 2008.

Table 2: Summary of Publications

Category	**Publications**
Profile Driven Identification of Component Resource Requirement	1. Bulls-Eye: A Resource Provisioning Service for Enterprise Distributed Real-time and Embedded Systems , Proceedings of the International Symposium on Distributed Objects and Applications (DOA), Montpellier, France, Oct 30th - Nov 1st, 2006. [9] 2. Dynamic Analysis and Profiling of Multi-threaded Systems, Designing Software-Intensive Systems: Methods and Principles, Edited by Dr. Pierre F. Tiako, Langston University, OK, April, 2008. [7] 3. Design and Performance Evaluation of an Adaptive Resource Management Framework for Distributed Real-time and Embedded Systems, EURASIP Journal on Embedded Systems (EURASIP JES): Special issue on Operating System Support for Embedded Real-Time Applications, Edited by Alfons Crespo, Ismael Ripoll, Michael Gonzalez Harbour, and Giuseppe Lipari, 2008, pgs. 47-66. [8]
Performance Estimation of Component Based Software Applications and application placement	4. A Component Assignment Framework for Improved Capacity and Assured Performance in Web Portals, Proceedings of the 11th International Symposium on Distributed Objects, Middleware, and Applications (DOA'09) Vilamoura, Algarve-Portugal, Nov 01 - 03, 2009 [6]. 5. Modeling Software Contention using Colored Petri Nets, Proceedings of the 16th Annual Meeting of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), September 8-10, Baltimore, MD [1]. 6. The Impact of Variability on Soft Real-Time System Scheduling, Proceedings of the 15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2009), Beijing, China, August 24-26, 2009 [2]. 7. A Novel Capacity Planning Process for Performance Assurance of Multi-Tiered Web Applications, to be submitted to the 18th Annual Meeting of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), August 17-19, Miami, FL [3]. 8. Model-Driven Performance Evaluation of Web Application Portals, Model-Driven Domain Analysis and Software Development:

[6] N. Roy, Y. Xue, A. Gokhale, L. Dowdy, and D. C. Schmidt, “A component assignment framework for improved capacity and assured performance in web portals,” in *OTM ’09: Proceedings of the Confederated International Conferences, CoopIS, DOA, IS, and ODBASE 2009 on On the Move to Meaningful Internet Systems*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 671–689.

[7] D. G. Waddington, N. Roy, and D. C. Schmidt, “Dynamic Analysis and Profiling of Multi-threaded Systems,” in *Designing Software-Intensive Systems: Methods and Principles*, P. F. Tiako, Ed. Idea Group, 2007.

[8] N. Shankaran, N. Roy, D. C. Schmidt, X. D. Koutsoukos, Y. Chen, and C. Lu, “Design and Performance Evaluation of Resource-Management Framework for End-to-End Adaptation of Distributed Real-time Embedded Systems,” *Journal on Embedded Systems: Special issue on Operating System Support for Embedded Real-Time Applications*, 2008.

[9] N. Roy, N. Shankaran, and D. C. Schmidt, “Bulls-Eye: A Resource Provisioning Service for Enterprise Distributed Real-time and Embedded Systems,” in *Proceedings of the 8th International Symposium on Distributed Objects and Applications*, Montpellier, France, Oct/Nov 2006.