Green Software Architectures: A Market-Based Approach

Govindaraj Rangaraj
The University of Birmingham
Birmingham, United Kingdom B15 2TT
govin.acs@googlemail.com

Rami Bahsoon
The University of Birmingham
Birmingham, United Kingdom B15 2TT
r.bahsoon@cs.bham.ac.uk

1. INTRODUCTION

Software systems architects are continually faced with the challenge of scaling up software systems architectures to sup- port constantly growing load of users' processing needs and data. Scaling up the architectures to meet these needs does certainly introduce additional energy cost. For example, to meet the scalability requirements, additional hardware and software resources may need to be deployed. Reducing the energy demands in such architectures while meeting the scalability requirements, are always challenging. We explicate the attention to power as an architectural constraint/property that need to be analyzed in relation with scalability. Current research and practice to distributed software architecture approaches are green-unaware. They don't provide the primitives for reasoning and managing power consumption. We argue that the software engineering should be green aware, where the software engineering and design activities should not only be judged by their technical merits, but also by their contributions to energy savings. In particular, the software system architecture appears to be the appropriate level of abstraction to address green-aware concerns. Software architectures should be green-aware, providing power management mechanisms as part of the architecture primitives. Furthermore, it looks plausible to leverage on advances in self-management software architectures [2], where self-managing power could be separated from the core system functionalities. We argue that there is a pragmatic need for new software architectural layer, which could be easily integrated with existing styles for self-managing the trade-offs between scalability and power. The power consumption can be minimized by only provisioning the required amount of resource at any given point of time. For example, architecture can be scaled only when the demand for the resource increases. Classical market-based economic theory is appealing for addressing this problem effectively in the context of supply/demand.

In this paper, we describe a standalone architectural layer for self-managing power consumption in a software system. This layer can be integrated with existing power-unaware architectures styles. The layer makes a novel use of the classical supply and demand, market-based economic theory, keeping the dynamic energy management process simple, intuitive, and appealing. It optimizes energy utilization of a runtime architectural instance by dynamically monitoring and matching the resource requirements (demand) with the resource availability (supply) in relation to various scalability scenarios. By scaling up/down the resource availability to match with its demand, we can avoid unnecessary wastages of power due to inaccurate resource provisioning. Experimental results shows that our framework improves the power savings, for a web based client-server architecture while maintaining the desired scalability requirements of a system.

2. PROPOSED APPROACH

Our proposed solution is a conceptual architectural level framework for automatic power management using market economic theory as depicted in figure 1. In next subsections, we present a brief overview of supply demand theory and describe the proposed architecture.

2.1 Economic theory. A market economy is described using two terms: Demand and Supply. Demand is defined as the quantity of goods consumers are willing to buy. Supply is defined as the quantity of goods the suppliers are willing to produce. Inflation scenario occurs when the supply of a good could not meet the demand. Similarly, a recession scenario occurs when the sup- ply is in excess with demand. In order to keep the economy in equilibrium (neither inflation nor recession), the supply should always be matched with the demand. At this point, all the goods supplied are consumed, without any wastage.

2.2 Proposed Architecture. We build on an analogy with the classical economic theory of 2.1. Figure 1 depicts the architecture of our pro- posed model. As shown in the figure, the economic layer of our framework is composed of four components. The Supply Manager (SM) and Demand Manager (DM) components deal with the supply i.e. resource availability and demand i.e. resource requirement concerns of a system. (The resource availability can be defined as number of nodes in a cluster and the resource requirements can be defined as number of incoming requests per second.) The Supply/Demand Coordinator (Coordinator) controls these two components to reach the equilibrium state where the resource availability matches its demand. In this state, unnecessary wastage of power due to resource over-provisioning can be avoided. For example, the Coordinator notifies the SM to decrease the resource availability whenever the resource demand decreases and vice versa. The additional strategic planning component is responsible for handling change management in the framework. The translational layer is used to map the abstract architectural concerns with its implementation concerns. The system layer represents the underlying run-time architectural instance which is being power managed. The sensors and actuators, attached to the system, monitor and control the system specific values. A detailed view of the components of economic layer is given below:

Demand Manager: It collects the resource demand in-formation of the underlying system through the sensors. For example, the resource demand of client server architecture can be determined by the number of pending requests in the queue. This information is collected for a specified time interval, consolidated and transformed into an abstract index value called demand index. The demand index is analogous to the number of goods demanded in the context of economic theory.

The demand index for a web based client- server architecture can be calculated as follows: number of pending requests/(number of active nodes * 100). This component also performs actions on the underlying system such as blocking the incoming requests from distributing to the server for processing, upon receiving notification from the Coordinator.

Supply Manager: It collects the resource availability information of the underlying system through the sensors. For example, the resource availability of cluster based client-server architecture can be determined by the number of active nodes and their respective loads in the cluster. This information is collected for a specified time interval, consolidated and transformed into an abstract value called supply index. The supply index is analogous to the number of goods supplied in the context of economic theory. The supply index for a web based cluster server is calculated by the ratio of cumulative available capacity of the cluster in percent- age (%) and the total capacity of the cluster in percentage (%). It also performs actions such as increasing or decreasing resource levels, on the underlying system, upon receiving notification from the Coordinator.

Supply/Demand Coordinator (Coordinator): The coordinator is the abstract component which acts as a mediator between the SM and DM by continuously monitoring the supply and demand indices. It consists of predefined configurations of what action to be taken when the demand index varies relative to the supply index. It obtains the index values from both SM and DM, compares them, chooses an appropriate action (to match the supply with the demand) and controls the SM and DM to execute the action. For example, whenever it encounters an inflation or recession constraint, it compares the supply and demand indices, decides whether to increase or decrease the resource supply and notifies the SM and DM to take action accordingly.

Strategic Planner: This component produces change management plans upon request from Coordinator. For example, it reconfigures the framework when the underlying system is integrated with external resource, on the fly, to meet unanticipated demand. It can also recreate failed components of the framework to address the fault-tolerance.

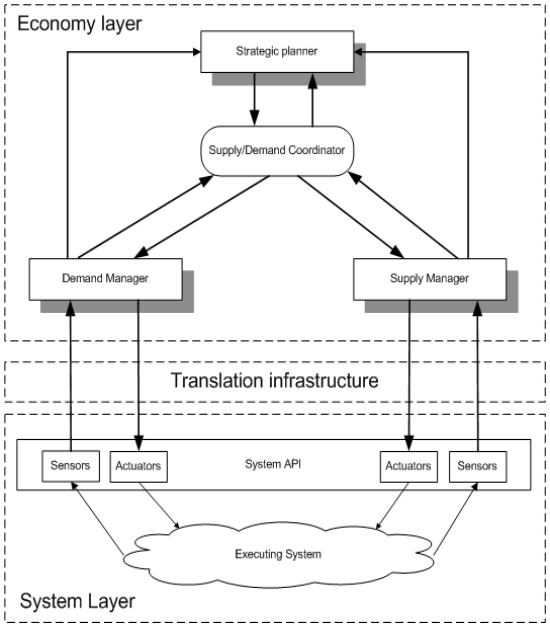

Figure 1: Economic framework

3. Working Scenarios

We describe an architecture scenario, which utilizes the proposed power management layer. We use an instance of client server architecture which consists of a serverCluster of replicated nodes to meet the scalability requirements. The client requests are sent to the server for processing through a request queue. Power saving can be achieved in such an environment by provisioning the resource according to the demand i.e. by dynamically switching off some of the nodes under lighter load conditions to save power. Likewise, scaling up can be achieved by dynamically switching on some of the nodes under burst load conditions. Our proposed framework automates this process efficiently and dynamically using supply and demand theory. The DM controls the demand layer of the cluster which is its request queue. The SM controls the supply layer of the cluster which is the ServerCluster itself. Both the lay- ers should have respective sensors and actuators attached to them for monitoring and controlling the changes respec- tively. The DM monitors the request queue and calculates the demand index. Similarly, the SM monitors the load in each cluster node and calculates the supply index at regular intervals. The SM and DM regularly update the latest in- dex values to the Coordinator which keeps tract of the same, anticipating for constraint violations.

Inflation Scenario: An inflation scenario occurs when the supply index value reaches below 0.2 (80% of the re-sources are utilized in the cluster). When the inflation constraint is flagged, the Coordinator compares the supply index with demand index to see if the demand index exceeds supply index by more than 0.2 (0.2 is the threshold differ-ence between supply and demand index configured in the Coordinator, but it can be varied according to the underly-ing system), which means that the supply cannot meet the demand requirements. If this condition occurs, the Coordi-nator starts self adaptation process by notifying the SM to increase the supply and wait for an acknowledgement from SM upon completion of the action. It also notifies the DM to decrease the demand temporarily, as overloading the cluster may increase the response time.

After receiving notification, the SM controls the under-lying cluster (with the help of the actuator) to activate a new node and DM controls the request queue of the un-derlying cluster (with the help of actuator) to stop sending the low priority requests to the cluster. Once a new node is activated, SM updates the Coordinator with an acknowledg-ment and the new supply index value which is (200-85)/(200)= 0.575. The Coordinator in turn compares this with the new demand index obtained from DM which is 110/(2*100)= 0.55. Since the difference is now below 0.2, it updates the demand manager to normalize the demand distribution (start distributing the low priority requests). Thus, minimal resource configuration is sufficient for the architecture at initialization in order to save power. The architecture can be scaled dynamically in the later stage as the demand increases.

Recession Scenario: Recession scenario occurs when the supply index goes above 0.4 (only 60% of the cluster load is utilized). Let us assume that there are three active nodes in the cluster which are running respectively at 40%, 50% and 60% of the load. Assume that the number of requests that are queued in the system at present is 60. Demand index at this point is 60/(3*100) = 0.2 and the supply index is ((100-40) + (100-50) + (100-60))/300 = 0.5.

As the supply index is above 0.4, a recession constraint is flagged in the Coordinator. When the recession constraint is flagged, the Coordinator compares the two indices to see if demand index exceeds the supply index by 0.2, which means there is excess resource available. If this condition occurs, the Coordinator starts the self-adaptation process by notifying the SM to decrease the supply and waits for the acknowledgement. After receiving the notification, the SM deactivates a node which is running at 40% load by transferring its load into other nodes with 20% each. Now, the new supply index is (100-(50+20)) + (100-(60+20)) / 200 = 0.25 and the new demand index is 60/(2*100) = 0.3. The difference is activated simultaneously which reduces the activation time by almost half and hence improves the overall performance. Similarly, it notifies the SM to scale down the resources further when the demand index decreases rapidly. Deactivating simultaneous nodes without any delay will result in significant power saving. It also shows how effectively our framework responds to burst increase in the demand by scaling the architecture at higher rate. These different scenarios clearly shows the effectiveness of our approach in managing the trade-offs between scalability and power.

Higher inflation/lower recession scenarios: After notifying the SM to increase the supply due to an inflation constraint, the coordinator continues monitoring the latest supply and demand indices. If the demand index grows at a higher rate, for example, when the new supply cannot match the current demand, it notifies the SM to scale up the resource further without waiting for the acknowledgment for the previous notification.

4. Conclusion and Further References

We have reported on ongoing work on Green Software Architectures. Further references of the work can be found in [1] with discussion of closely related work and evaluation of the approach. We are further extending the work to address green-aware concerns in various software architectural styles.

In conclusion, green-aware constraints such as power brings new challenges to the way we systematically develop, maintain, manage, evolve, and scale software architectures. The paper discusses a dynamic self-management software architecture framework for power based on economic theory. We have discussed various scenarios to demonstrate the effectiveness of our approach in saving power using supply/demand theory.

Different power un-aware architectures can benefit from our approach with minimal modification. Many modern architectural paradigms, such as cloud, which has clear separation of supply and demand, can also benefit from our approach. As a part of our future research, we are looking at relating other architectural dependability requirements with power. We are also working on extending the functionality of DM to raise inflation or recession scenarios to improve the overall self adaptation process.

The research will raise the understanding of evolution trends in dynamic systems, and improve their quality and robustness through dependability and power measurement and control. More widely, we hope the research results will feed into long-term vision of helping in reducing power consumption and CO2 emissions in ICT infrastructures.

5. References

[1] G. Rangaraj and R. Bahsoon(2010). A Market-based Approach for Self-Managing Power in Software Architectures – in submission. Technical Report, School of Computer Science, University of Birmingham, CSR-10-01.

[2] J. Kramer and J. Magee. Self-managed systems: an architectural challenge. In FOSE '07: 2007 Future of Software Engineering, pages 259–268, 2007.

[3] V. Nallur and R. Bahsoon(2010). Design of a Market-Based Mechanism for Quality Attributes Tradeoffs of Services in the Cloud, To appear, in the Proceedings of the 25th ACM Symposium of Applied Computing (ACM SAC 2010), 2010.