RC22743 (W0303-076) March 17, 2003
Computer Science

IBM Research Report

HotRod: An Autonomic System for Dynamic Surge Protection

Edwin R. Lassettre, Joseph L Hellerstein, Maheswaran Surendra, David W. Coleman, Yixin Diao, Steven E. Froehlich, Lawrence S. Hsiung, Todd W. Mummert, Mukund Raghavachari, Geoffrey K. Parker, Lance Russell, Veronica P. Tseng, Noshir C. Wadia, Peng Ye
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

An Autonomic System for Dynamic Surge Protection

Ed Lassettre, David W Coleman, Yixin Diao, Steven Froehlich, Joseph L. Hellerstein,
Larry Hsiung, Todd Mummert, Mukund Raghavachari, Geoff Parker, Lance Russell,
Maheswaran Surendra, Veronica Tseng, Noshir Wadia, Peng Ye
IBM Corporation

Introduction

Today's information technology (IT) departments are besieged with uncertainty. New applications are deployed, but their resource demands are unknown. Businesses offer promotional discounts, but the volume of customer responses is uncertain. Traditionally, these situations have been addressed by over-provisioning IT resources and/or manual intervention to adjust for resource imbalances. Unfortunately, both approaches are undesirable. The first is costly in terms of equipment, licenses, electricity, etc.; the second requires expert operators who are in short supply and costly as well. This paper discusses an approach we have developed to react rapidly to workload surges so as to preserve service level objectives in a cost effective way.

Examples of subscriber overloads abound. On September 11, 2001, CNN (along with other news agencies) saw traffic double every seven minutes to a total of twenty times the normal volume until the web site was overwhelmed (Bill Lefebvre, CNN Internet Technologies, 2001 LISA Invited talk "Facing a world crisis"). The Victoria's Secret web site had a similar experience as a result of an advertising campaign during the 1999 Super Bowl (http://www.cnn.com/TECH/computing/9902/05/vicweb.idg/). Others have noted that "sites such as Encyclopaedia Britannica, egg.com, and H&R Block have suffered massive overload from subscribers" (http://www.zeus.com/library/articles/webvital.htm).

Many systems have been developed to adapt to changes in workload, but not to automate surge protection. The ThinkProvision technology of Think Dynamics (http://www.thinkdynamics.com), HP's Utility Data Center (UDC) (http://www.hp.com), and Sun's N1 (http//:www.sun.com) initiatives provide convenient ways for operators to move resources between systems. However, no automation is provided to determine *when* resources are moved. The MVS workload manager (Aman et al., "Adaptive algorithms for managing a distributed data processing workload," **IBM Systems Journa** l, 1997, 36) incorporates algorithms for adjusting resource allocations to achieve service level objectives. These algorithms assume that actions take effect immediately (e.g., changing CPU priorities) and so do not address actions with substantial startup delays, such as server provisioning in response to workload surges. The DynamicIT technology of ProvisionSoft (http://www.provisionsoft.com) uses long-term forecasts to anticipate well-understood time-of-day effects. However, it does not address unexpected workload surges. Indeed, Chandra et al. ("Impact of Space-Time Multiplexing Granularity on Provisioning in On-Demand Data Centers", TR03-3, Computer Science Department, University of Mass.) report that there is considerable value in rapidly adjusting resource allocations.

The foregoing motivates our development of a system that automates the response to workload surges. At the core of this system employs three technologies: adaptive forecasting, on-line capacity planning, and rapid configuration management. Adaptive forecasting detects and anticipates the progression of surges. On-line capacity planning determines the resources needed to defend service levels. Rapid configuration management addresses the resource requirements by tuning, provisioning, and/or workload throttling to adapt to the onset and subsiding of unexpected surges.

Architecture and Algorithms

The system is structured into three layers as shown in Figure 1. The Application layer provides the business function. In our prototype, a two-tier web application is used that consists of one or more application servers and a database server. In general, we require that the application layer scale horizontally. That is, resources may be added and removed without requiring a system shutdown (e.g., by adding application servers).

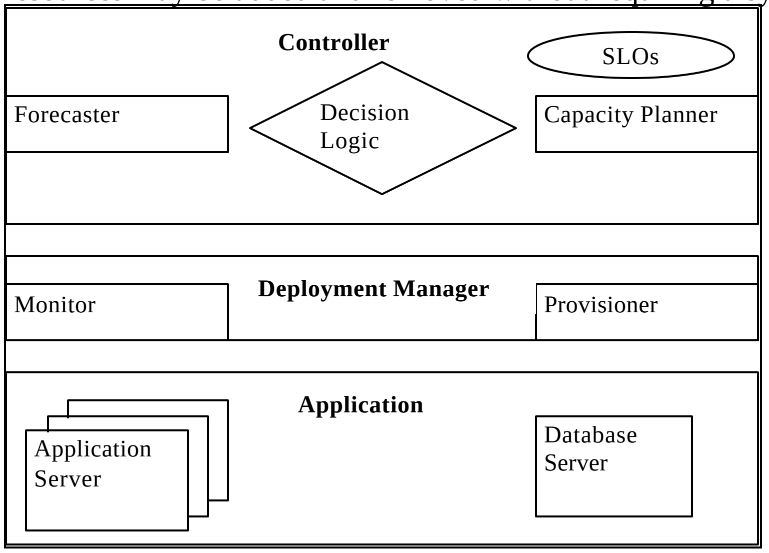

Figure 1: Architectural layers

The Deployment Manager provides a generic interface for monitoring and configuring the application layer. In our current implementation, the data monitored are transaction rates and response times, and configuration management focuses on provisioning (specifically the addition and removal of application servers). The architecture assumes that for each resource type there is a provisioning function that manages a resource pool that can be shared among applications. Longer-term configuration management will address not only resource provisioning but also adjustments to configuration parameters (e.g., buffer pool sizes) and setting admission control parameters.

The Controller is responsible for maintaining service levels by determining the state of the application layer and initiating appropriate actions if a service level objective (SLO) violation is anticipated or if SLOs can be maintained in a more cost-effective way. Figure 2 depicts the information and control flow within the Controller. The forecaster generates workload forecasts. The capacity planner determines the resources needed to maintain service level compliance for the forecasted workload. The decision logic manages the information flow and determines the resource (or other) adjustments needed.

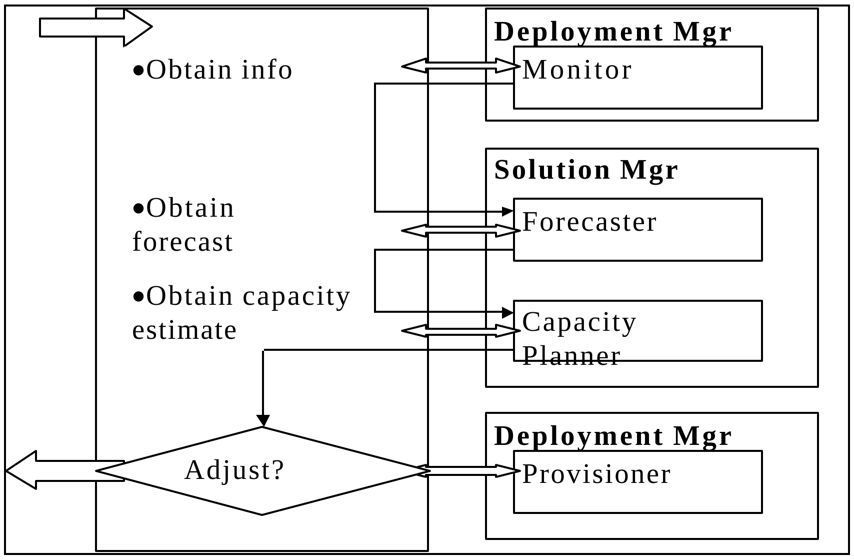

Figure 2: Control and information flow

The system operates based on six time intervals.

1. The measurement interval (*M*) is the interval between collecting measurements from the application. This should be long enough to moderate measurement variability and reduce collection overheads but short enough to respond to workload changes.
2. The control period (*P*) is the interval between executions of the control engine. Clearly, *P* should be no smaller than *M*, but it may be larger if the overhead of control is high.
3. The startup delay (*S*) is the time required to have an action executed. In our prototype, this relates to the time to provision or de-provision a server for an application. Data gathered during a startup interval is suspect since it includes transient effects that are indistinguishable from noise. As a result, the control engine is dormant during startup intervals.
4. The forecast horizon (*H*) is the time into the future for which workload demands are predicted. The forecast horizon must be long enough to anticipate the action to be taken, but no longer since forecast accuracy decreases as *H* increases. The forecast horizon should be an integral multiple of the control period.
5. The overflow interval (*O*) is the time between occurrences of situations in which resources need to be adjusted to avoid violating a service level objective. The control engine can respond if *O* is no larger than *P+2S*. (The doubling is needed since the Controller is not active during the startup delay.)
6. The underflow interval (*U*) is the time until the service level objective could be satisfied with fewer resources. As with the overflow interval, the Controller can handle a *U* no smaller than *P+2S*.

We conclude that *H* should be close to *P+2S*. The values of *P* and *S* depend on the specifics of the application layer. In our prototype, the Provisioner uses the WAS 5.0 cellular cluster and its AddNode command to incorporate the new server into the cluster. This results in an *S* that is approximately 20 seconds. After some experimentation, we found that setting *M* and *P* to 10 seconds works well. Incorporating a 10 second "safety margin", we set *H* to 60 seconds.

The capacity planner inputs workload forecasts and the SLO and outputs the resource requirements (which in our prototype is the number of application servers). For the capacity planner, we incorporated an IBM internal tool that has been widely used in service engagements over the last two years. The capacity planner provides both performance estimates and the ability to determine resource requirements given the workload and the SLO.

The forecaster, which provides the workload predictions used by the capacity planner, has been challenging to develop. While a variety of forecasting techniques exist, switching between long-term and short-term forecasts in a robust way requires some invention. Our approach employs Box-Jenkins ARIMA (autoregressive, integrated, moving average) models and dynamically adjusts the model order (the number of autoregressive terms) based on the stability of the estimates. The latter is determined by checking if the poles of the transfer function lie within the unit circle (a requirement for stability in discrete time systems). This approach simplifies matters in that we do not need to eliminate surges from the historical data. However, it also means that the training data used by the long-term forecaster includes the unexpected surges. In some cases, this results in the prediction of phantom surges. This problem led to further refinements.

Results

Figures 3-5 show the results of running our autonomic system for a workload in which simulated users request various business operations, and there are occasional workload surges caused by a large influx of simulated users. The service level objective is that response time should be under 2 seconds.

Figure 3 plots the actual (blue or darker line) and forecast (lighter or green line) business operations per second (BOPS), the metric used to characterize workload. We see that during the non-surge period, BOPS hovers around 20. When a surge begins (e.g., 9:37, 9:49), BOPS increase rapidly to a peak of 120, or a factor of 6 over the non-surge period. Forecast BOPS (the green or lighter line) closely approximates the actual BOPS during periods of normal workload, such as 9:30 to 9:35. Initially, the forecast underestimates the actual BOPS because the forecast is based on non-surge data. After a minute, the short-term forecaster is used, and accuracy improves considerably. However, forecasts made during surges are not nearly as accurate as those during the non-surge periods.

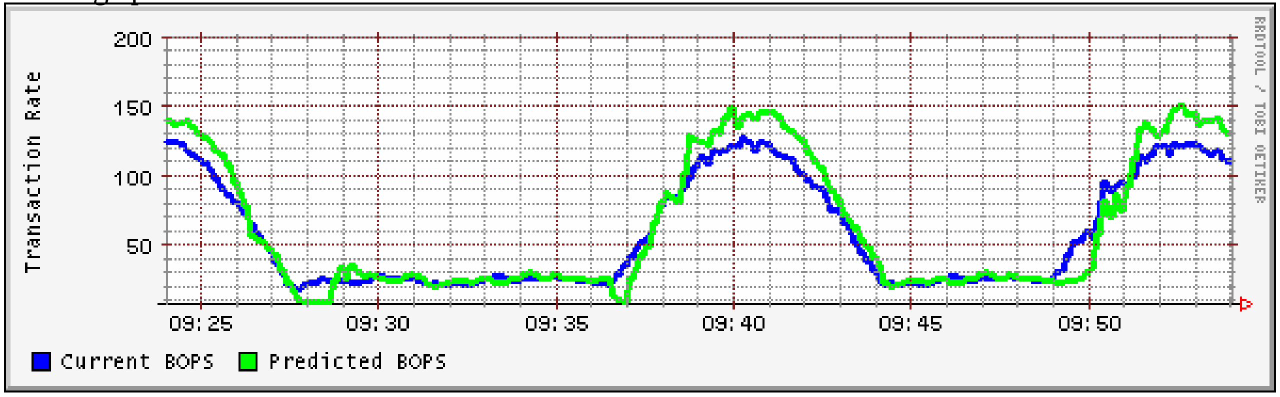

Figure 3: Predicted and Actual Business Operations Per Second (BOPS)

Figure 4 shows the changes of the state and the number of application servers in response to the actual and forecast demands on the system. When a rapid increase in load (and associated increase in response time) is detected at 9:37, a server is added (the leading dark or blue section). A second server is added at 9:38 as the short-term forecaster anticipates the progression of the surge. As the surge subsides around 9:42, servers are released (the trailing red or less dark section).

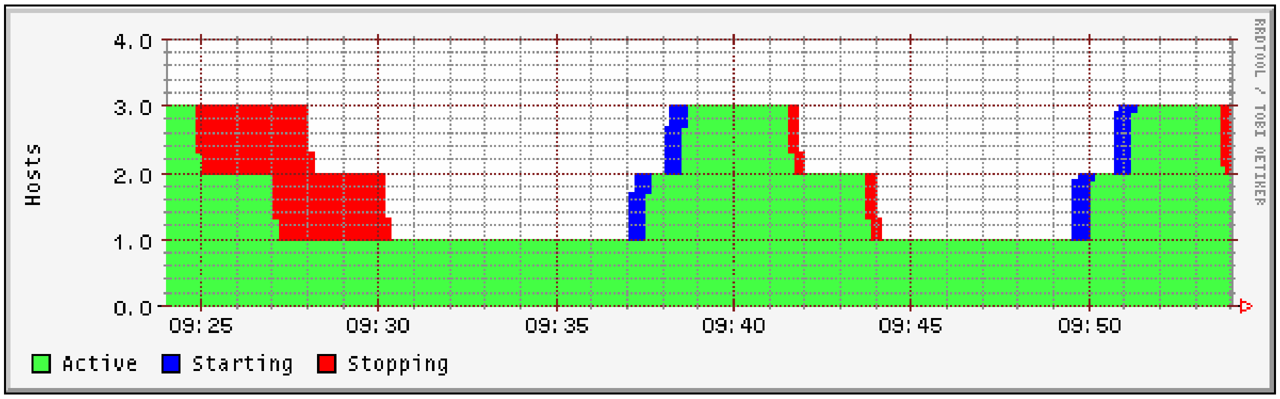

Figure 4: Number of Application Servers

Figure 5 depicts the effect of these actions on response times. We see an initial bump in response time around 9:37. In part, this is due to the increased load that cannot be handled until the server has completed its startup phase. But there is also some delay introduced by the action of adding a server. A similar response time bump occurs at 9:39 as a second new server comes on-line. These actions are able to contain response times within the SLO. We note that because of the stochastics of the system, different surges result in different control actions and different responses, as evidenced by the response time peak at 9:50 followed by the much smaller response time bumps.

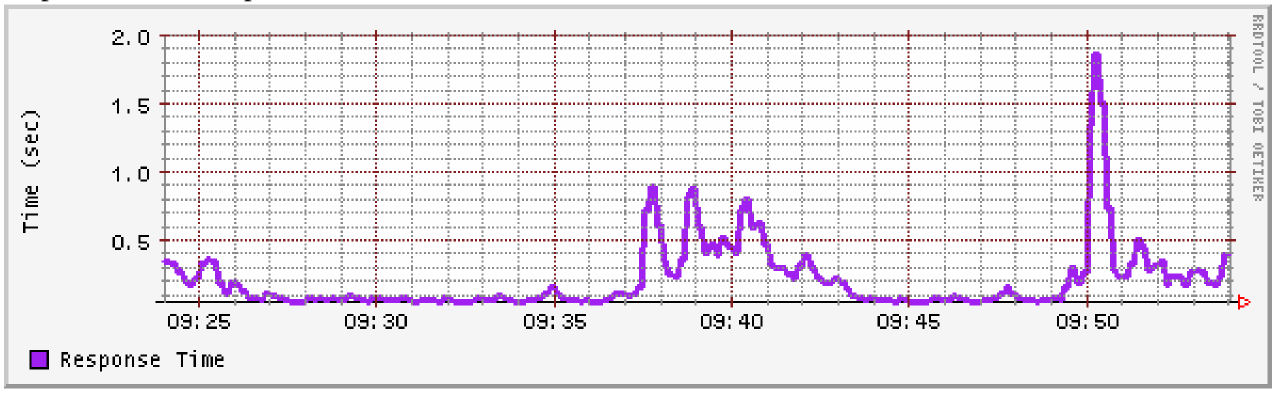

Figure 5: Response Time of Application

Conclusions

The autonomic system we describe integrates three technologies that provide a cost effective approach to handling unexpected workload surges. Adaptive forecasting provides a way to anticipate the trajectory of workload demands once a surge is detected. On-line capacity planning determines the resources required to maintain a service level objective (SLO) based on the anticipated workload. Rapid configuration management addresses the resource requirements by tuning, provisioning, and/or workload throttling to adapt to the onset and subsiding of unexpected surges.

Our future work will address the handling of multiple workloads, incorporating tuning as well as provisioning actions, and consider remote resources that are accessed through Grid services.