A Visualization Tool for Automatic Overload Mitigation

We advocate an approach for building self-adaptive Web services in which the operator plays an important role and remains "in the loop." By better visualizing underlying connections between components and load, we claim that operators can become better decision makers. An example of the type of visualization we advocate is given in Figure 3, in which the pie chart shows the percentage of traffic that is correlated to our bottleneck. Within the correlated slice, specific request types are enumerated. From this simple graph, an operator can see which requests would be affected by selective admission control, as well as what percentage of the total traffic they represent.

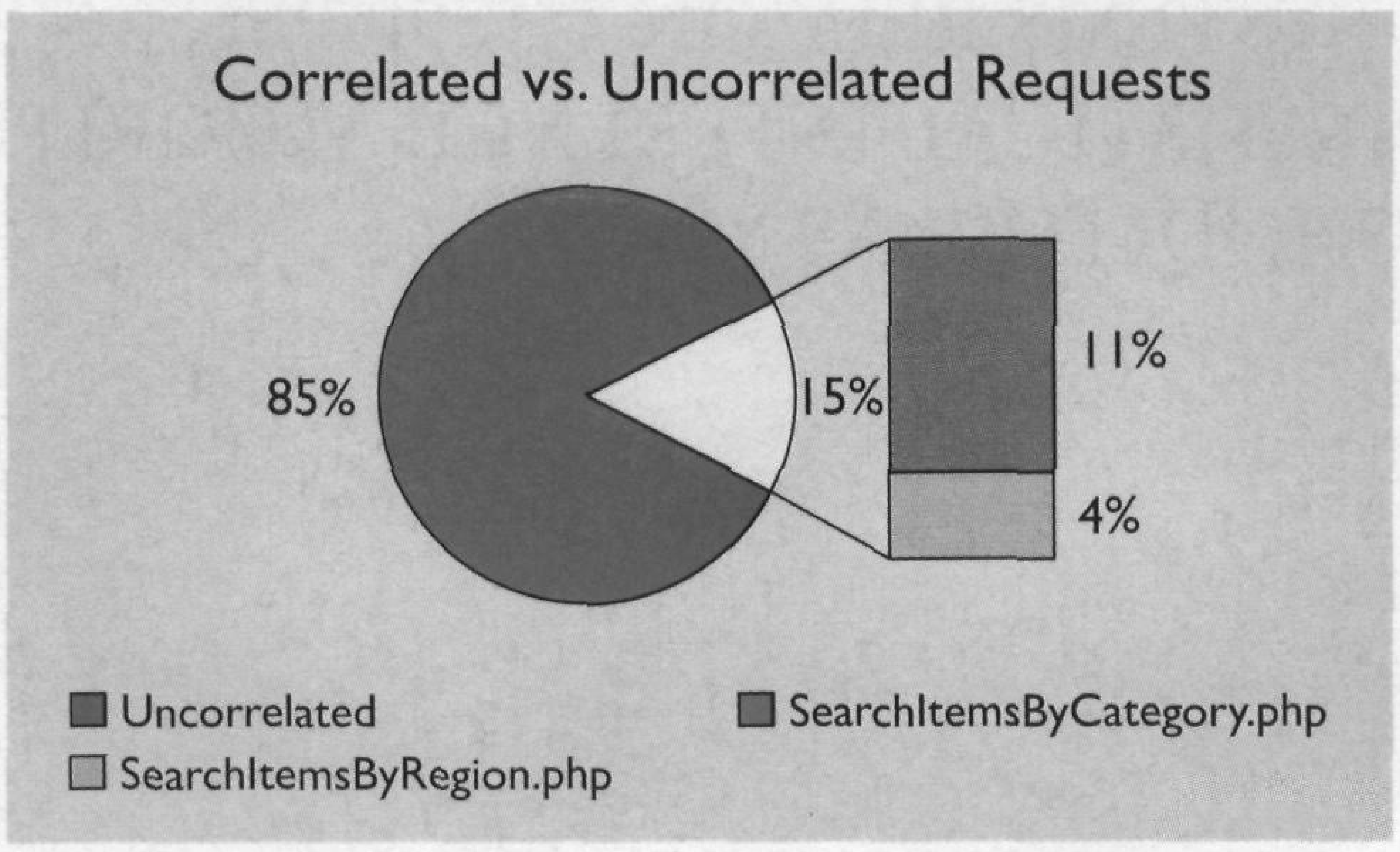

Figure 3. This visualization shows the requests identified by our system as candidates for selective admission control. Additionally, the graph shows their percentage of the total number of requests.

Revising the three motivations from the previous section, we see that the visualization component reduces the disruption fear by providing a point for the operator to see information needed to diagnose and pinpoint performance problems. Once operators feel more comfortable with the tool, it can be made more automatic. Secondly, to cope with rapidly changing services, visualization tools allow operators to choose whether to implement throttling depending on formal or informal business rules that are known to the operator. Again, as the tool is used more often, some admission control decisions might be programmed to take effect automatically without operator involvement. Lastly, we address the distributed ownership of components. By visualizing request effect through the system, observations across different components (often in different parts of the enterprise) can be correlated into one display that gives more insight to the system's operation.

Scenario	Total Requests	Correlated URLs	Requests/ Second	Average Session Time (s)	Maximum Request Time (s)
Stock	756,137	112,521 (14.9%)	462	670 s	154.7 s
Selective Admission Control	1,143,264	105,964 (9.3%)	782	872 s	32.7 s

Table 2. Performance measurements for a stock deployment and one that utilizes selective admission control. Both measurements represent 30 minutes of elapsed time with 3,500 concurrent clients. A session represents a series of operations on the auction site.

Effective Actuators for Admission Control

Once a subset of the requests are identified as candidates for selective admission control, the operator needs a way to reduce the rate at which they arrive. This can be done at the HTTP level through 503 TOO BUSY status messages, or at the network level through bandwidth shaping. We chose to implement the throttling at the network level because that did not involve modifying the Web tier. Correlated requests were sent over a bandwidth-limited network path. The effect on the end user for such requests is they take longer than they used to. This means that sessions, which consist of multiple, individual requests, might take longer than before.

To tie together the visualization tool and the actuators for admission control, we envision an interface in which each request type is listed, along with its likelihood of relieving the noticed bottleneck based on our statistical findings. Such a display resembles a "top talkers" graph. In Figure 3, they would be able to select requests identified by the bar graph. Once selected, new filters could be sent to the Web server (in the case of HTTP-based throttling), or to the network appliance (for network-level throttling). In either case, the operator would have a tactile way to see the effect of their choice on both the bottleneck and the arriving traffic.

Results

We have deployed a Web service based on the RUBiS auction site that embodies the four mechanisms outlined in this article. Our testbed consists of an IBM BladeCenter with two Nortel Layer 2-7 Gigabit switches. The Nortel switches can perform deep packet inspection to identify HTTP request types (based on URL pattern matching) at gigabit rates and assign VLAN tags to packets that should be subject to admission control. To perform the bandwidth throttling, we use the Linux Traffic Control (`tc`) extensions to reduce the rate of correlated requests from 3.5Mbit/s (the baseline rate) to 1Mbit/s.

As Figure 1 and Table 2 show, performing this selective admission control greatly improves the performance of the Web service for users who are not causing bottlenecks. In our deployment, the number