Customer Case Study

Industry:
Community Website

Problem Statement

The site is a Web 2.0-like community, with users creating profiles, image galleries and blogs - all running on mostly custom developed software. A message board, running a well known open source solution.

The multi-tiered existing setup included a front-end web farm of 26 web servers, running Apache/PHP and Apache/Tomcat, dedicated Jboss farm of 12 servers, multiple DB servers, including Oracle and MySQL clusters. NFS file store resided on a file appliance from a well known storage vendor.

For full functionality, user needed to register on the site, but anonymous browsing was allowed.

Customer started experiencing problem as its online community started to see more user registration and traffic. Each page-view typically resulted in execution of code on Tomcat and Jboss servers and multiple Database queries . Page load time increased to around 10 seconds during peak period, load utilization on most servers spiked in direct correlation to user traffic. User complains started to mount, visitors curve flat-lined and registrations started to drop.

An internal estimate projected having to double hardware, to allow for doubling of registered users. As existing hosting capacity simply could not accommodate for such dramatic increase in space, power and cooling and spend required to move to a different datacenter was very substantial, the situation started to look very bleak. It was clear that architecture was not sustainable and the code base needed an overhaul. The development team was busy maintaining parity with competition and significant code changes were not possible.

aiCache to the rescue

An aiCache proof of concept was implemented, using 4 server, each a dual dual-core 64bit Intel system with 32GB RAM.

Upon examining the typical use-cases and HTTP traffic patterns, aiCache was configured to cache most pages for 60 seconds. That included output that rendered the community home page: "What's new", "most popular group", "featured users". User's home pages, galleries and blogs were also enabled for caching. Same for user forums - home page, forum fronts, thread and article views were all enabled for caching. The same for images, .js, .css and all other auxiliary content - these were configured for a 7 day Time to live.

A large on-line community web site.

Technology: PHP and Java-based Open-Source and custom-developed SW. Apache/PHP, Apache/Tomcat and Apache/Tomcat/Jboss setups.

Back-end databases: MySQL and Oracle.

Original HW footprint - in excess of 60 servers.

Typical HW server configuration:
1RU FF server with 2 dual-core Intel or AMD CPUs, 4GB or 8GB of RAM, redundant network connections. DB clusters: dual quad-cores, with storage on the SAN.

The farms are load balanced via dedicated, redundant load balancers and protected via dedicated, redundant firewalls.

aiCache HW:
4 dual-socket dual-core 64bit Intel-based 1RU servers, with 16GB of RAM. As customer's IT org was already quite familiar with one of commercial Linux distros, where was no learning curve required for system admin and web infrastructure teams.

The results.

Post aiCache deployment, traffic to the origin server farm was reduced by about 65%. The bulk of setup has become low CPU - even during the busiest hours of the day. The number of connections to the DB servers was reduced, as was the load on file appliances.

As a result of traffic reduction, a decision was made to downsize web farm to 6 servers, down from 26. There where able to downsize Jboss servers to 4 from 12, and reduce the Database servers to just a 2 x 2-node clusters. Setup was further simplified by removing a number of DB read-replica servers, as these became unnecessary.

The relocation to a different datacenter/hosting cage has become unnecessary and significant growth can still happen at existing datacenter, due to significant reduction of footprint.

Additional benefits.

aiCache's rich instrumentation was put to good use, allowing for real-time monitoring of user traffic and identification of, and alerting on, slower code on origin servers.

aiCache's SNMP integration is also utilized - a number of SNMP OIDs are collected, charted and alerted on by a well known open source monitoring package.

aiCache's selective log suppression feature is utilized to control the size of log files.

As origin servers now have available capacity, content compression was turned on the origin web servers.

An effort is on the way to further streamline the setup by configuring on-demand cache expiration feature of aiCache - to allow for longer TTL for dynamic content, while refreshing it on-the-fly in case of content changes.

An interesting decision was made to create a custom CLI script that utilizes "sir" command - "inventory sorted by requests" to see more popular URLs in real-time and integrate some of the output into "most popular right now" feature in some section fronts.

Savings:

- 60 new servers at ~U$5000 per: U$300,000.
- OS licenses: about U$42K
- DB licenses: about U$120K
- Server install and setup charges: about U$18K
- re-purposing of 24 servers and OS licenses for other applications: U$120K

Total CapEx saved:
more than $600K.

The estimate doesn't include the projected relocation expense, estimated at additional CapEx of $180K and much higher run rates.

Being able to stay up under even heaviest traffic, while reducing HW footprint, saving space, power and cooling in existing datacenters: *priceless.*