

WhizzBee

Web Server 1.0a

TECHNICAL OVERVIEW

Abstract

This document gives information technology professionals a technical overview of WhizzBee Web Server 1.0a, a new web server running on PC/Linux platforms, and its distinctive features. These features include performance and reliability improvements such as support for clustering, cooperative caching, efficient packet routing, service monitoring and dynamic load-balancing. Benchmark results show that WhizzBee Web Server outperforms its competitor, with improvement reaching as much as 1,522% in an 8-node configuration.

Copyright © 2001 Whizz Technology Ltd. All rights reserved.

“Linux” is a registered trademark of Linus Torvalds. Other product and company names mentioned herein might be the trademarks of their respective owners.

The information contained in this document represents the current view of Whizz Technology Ltd. on the issues discussed as of the date of publication. Because Whizz must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Whizz, and Whizz cannot guarantee the accuracy of any information presented after the date of publication.

This Technical Overview is for informational purposes only. WHIZZ TECHNOLOGY LTD. MAKES NO WARRANTIES, EXPRESSED OR IMPLIED, IN THIS DOCUMENT.

Whizz Technology Ltd.

<http://www.whizztech.com/>

info@whizztech.com

WhizzBee Support

support@whizztech.com

1/2001

Table of Contents

1	INTRODUCTION.....	1
1.1	CLUSTERING TECHNOLOGY	1
1.2	FEATURES OF WHIZZBEE WEB SERVER 1.0A	1
2	WHIZZBEE WEB SERVER ARCHITECTURE	3
2.1	OVERVIEW	3
2.2	CLUSTER VIRTUAL IP ADDRESS	4
2.3	WHIZZ IP TRAFFIC CONTROL HOST (WITCH)	4
2.4	EC-CACHE ENGINE	4
3	USER INTERFACE AND MANAGEMENT TOOLS	6
3.1	INSTALLATION	6
3.2	DAY-TO-DAY ADMINISTRATION.....	7
3.3	CLUSTER MONITORING AND EVENT LOGGING.....	7
4	HIGH AVAILABILITY SUPPORT.....	8
4.1	DISPATCHER FAILURE.....	8
4.2	WEB SERVER NODE FAILURE.....	8
4.3	SAFE CLUSTER ADMINISTRATION	8
5	PERFORMANCE EVALUATION	9
5.1	TEST ENVIRONMENT	9
5.2	TEST METHOD	9
5.3	BENCHMARK RESULTS.....	10
6	SUMMARY	13

1 INTRODUCTION

A web server is the core component that powers a web site. The performance of the web server is a crucial factor in the success of a web site and links directly to customers' satisfaction.

WhizzBee Web Server is a scalable web server based on clustering technology, and has a design goal to outperform many of its competitors in the global market. WhizzBee Web Server is developed on top of the open-source Apache web server, with a number of performance enhancement features not present in the original Apache.

This report describes the system architecture and important features of WhizzBee Web Server 1.0a.

In this section, we provide background information about clustering technology. We also outline the unique features built into WhizzBee Web Server 1.0a.

1.1 Clustering Technology

We rely on clustering in the design of WhizzBee Web Server. A cluster can be defined as a group of computers interconnected by a fast network. The aggregate power of a cluster can easily outperform that of a standalone server machine, and a cluster can scale up easily by adding more computers. The deliverable performance is virtually unlimited. Clustering technology offers cost-effective high availability solutions as single points of failure can easily be avoided by using redundant or duplicated components.

In order to realize the potentials of clusters, certain software components have to be developed in order to glue together the physically distributed computers and make them work cooperatively like a single machine, i.e., Single System Image (SSI). Furthermore, by implementing the necessary failure detection and recovery mechanisms, tasks running on a failed machine can be migrated to, or shared by, other running machines.

1.2 Features of WhizzBee Web Server 1.0a

WhizzBee Web Server 1.0a incorporates a number of advanced features that make it an ideal solution for constructing highly scalable and reliable web servers.

EC-Cache technology. Backend I/O access is the performance bottleneck for most server systems. Existing cluster-based solutions seem to have paid little attention to file caching effects. Precious system time is often spent unnecessarily in fetching already-cached-in files from the backend. With EC-Cache technology, files that have already been stored in physical memories will be re-used whenever possible, thus minimizing the backend I/O traffic.

Efficient packet routing. Traditional software routers present in most cluster-based web servers tend to limit the scalability of the system because of routing latency. WhizzBee Web Server includes a specially designed high-performance routing agent called WITCH (Whizz IP Traffic Control Host) that considerably cuts down on the routing latency within the cluster.

Single system image. WhizzBee Web Server integrates the power of low-cost PC computers and makes them work cooperatively like a single machine. A WhizzBee Web Server is accessible to the outside world by a single cluster virtual IP address or hostname. The fact that the system is physically distributed is completely hidden.

Improved availability. All system services are monitored by specially designed heartbeat protocols. Backup and duplicated components are there ready to take over the running services whenever system or application failures occur. There is no single point of failure in WhizzBee Web Server.

Dynamic load-balancing. WhizzBee Web Server includes a dynamic-load collector that collects real time load information of the cluster nodes. This information helps to balance the workloads among the web servers and is used to coordinate the access of remote caches.

Integrated management services. WhizzBee Web Server significantly reduces the costs associated with typical administrative tasks of web server clusters by providing convenient tools for software installation, configuration, maintenance, and monitoring.

2 WHIZZBEE WEB SERVER ARCHITECTURE

This section presents the architectural design of WhizzBee Web Server 1.0a, and its operations.

2.1 Overview

WhizzBee Web Server works by seamlessly integrating the computing resources of independent server nodes into a powerful, yet scalable, web server. The whole unit is accessible to the outside world by a single cluster Virtual IP Address (VIP) and hostname. Figure 1 illustrates the system architecture of WhizzBee Web Server 1.0a.

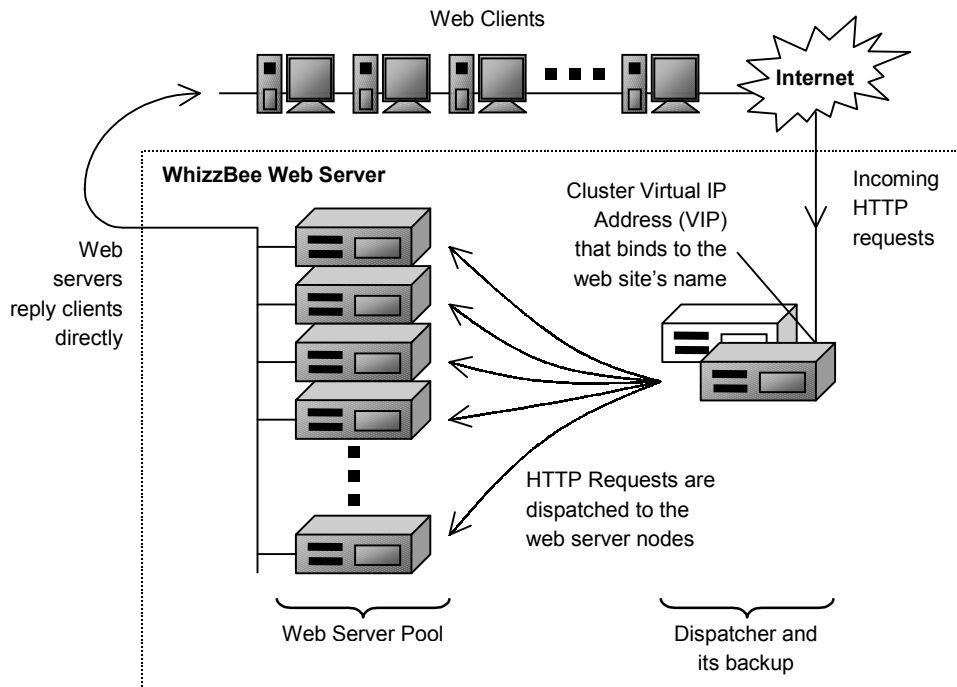


Figure 1: WhizzBee Web Server System Architecture

As shown in the figure, WhizzBee can be logically divided into two levels:

1. Dispatcher

There are two software modules involved, which could exist in the same computer node. WITCH (Whizz IP Traffic Control Host) accepts incoming HTTP requests, and dispatches the requests to the web server nodes according to some distribution policy using an efficient algorithm. CLSERV (Cache-Load Server) gathers the real-time load information of the web server nodes and coordinates the access of the cooperative caches.

2. Web Server Pool

Each web server in the pool is responsible for receiving and processing the HTTP requests dispatched from the dispatcher, and delivering the resulting web contents to the clients directly through the public network. A web server in the pool runs the Apache web server software, but with special routines that have been added for supporting the cooperative cache feature.

The detailed design and functions of the above software modules are described in the following sections.

2.2 Cluster Virtual IP Address

From the point of view of web clients, WhizzBee Web Server behaves as a single web server as it can be accessed through a cluster VIP. All cluster nodes, including the dispatcher and web server nodes, listen to the HTTP service port of the VIP. Replies to ARP (Address Resolution Protocol) requests for the VIP, however, are disabled at all web server nodes, and only the dispatcher would answer those ARP requests. Therefore, the dispatcher is the sole entrance for incoming packets. It dispatches the packets to the web server nodes, and the web server nodes send packets with the VIP as source to the web clients directly. The following steps describe how WhizzBee Web Server accepts and replies to a typical HTTP request.

1. A web client browser generates an HTTP request that targets at WhizzBee, i.e., the VIP.
2. The dispatcher picks up the packets from the network. It replaces the MAC address (Medium Access Control Address) of the packets with that of the best available web server node and dispatches the packets back to the network.
3. The chosen web server node accepts the packets that target at its own MAC address, reconstructs the HTTP request, and processes it.
4. After processing the HTTP request, the web server node sends the HTTP reply to the client directly through the VIP. In other words, it sends packets that originate from the VIP.
5. The web client browser receives the HTTP reply from WhizzBee Web Server.

2.3 Whizz IP Traffic Control Host (WITCH)

WITCH is an IP-level packet router that runs as a loadable kernel module at the dispatcher node. Upon receipt of an IP packet of an incoming HTTP request, WITCH replaces the MAC address of the packet with that of the best available web server node and dispatches the packet back to the network.

Compared to the TCP-level software routers used in some popular server products, WITCH is superior in its simplicity and low routing overhead. It is a key component in the design of WhizzBee Web Server to maintain a high degree of load-balancing among the web server nodes.

2.4 EC-Cache Engine

One of the innovative features of WhizzBee Web Server 1.0a is the Extensible Cooperative Caching Engine (EC-Cache Engine). Each web server node in WhizzBee Web Server contributes a portion of

its physical memory as cache to the cluster. By integrating these disparate memory pieces, a large virtual file cache is formed. A file fetched from the backend I/O store to a web server node will be cached in the physical memory of the server node. When another web server node needs the same file, it asks that other node to perform a cache-forwarding operation, instead of initiating an I/O access itself. Figure 2 illustrates the software configuration of the EC-Cache Engine.

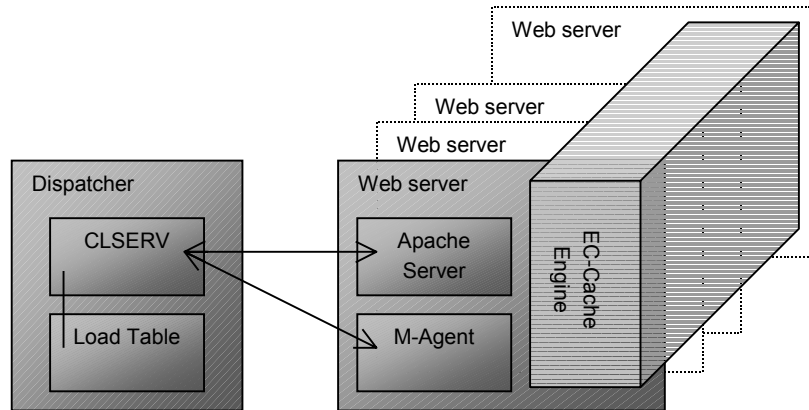


Figure 2: EC-Cache Engine

As shown in the figure, each web server node runs a special “M-Agent” and an Apache web server. The M-Agents running at the web server nodes periodically send dynamic load information to the central CLSERV (Cache-load server). The CLSERV running at the dispatcher node considers the real-time load information of the web server nodes and coordinates the access of the cooperative caches.

In order to better utilize the limited physical memory space, WhizzBee Web Server periodically removes outdated or less-frequently accessed files from the cache by using an efficient garbage-collection algorithm. The goal is to maximize the cooperative-cache-hit rate, while maintaining the highest data locality at each of the web server nodes.

WhizzBee Web Server’s innovative EC-Cache Engine considerably reduces backend I/O traffic. This in turn removes the I/O bottleneck that is present in most web server clusters. Furthermore, the cache-forwarding algorithm ensures better utilization of physical memory as the collective memory could hold a larger portion of the whole web site than traditional buffer caches. Most importantly, the consideration of dynamic load information prevents the web server nodes from being over-loaded by cache-forwarding requests. These claims are verified by the benchmark results shown in Section 5.

3 USER INTERFACE AND MANAGEMENT TOOLS

WhizzBee Web Server 1.0a provides a set of management tools that make typical administrative tasks easy. It has two choices of user interface, web-based GUI and command-line. The web-based interface allows administrators to configure and to monitor the cluster from anywhere in the world. The command-line tools provide an alternative to those who are used to console operations. All administrative tasks can be done in both user interfaces. Figure 3 illustrates the management infrastructure of WhizzBee Web Server 1.0a.

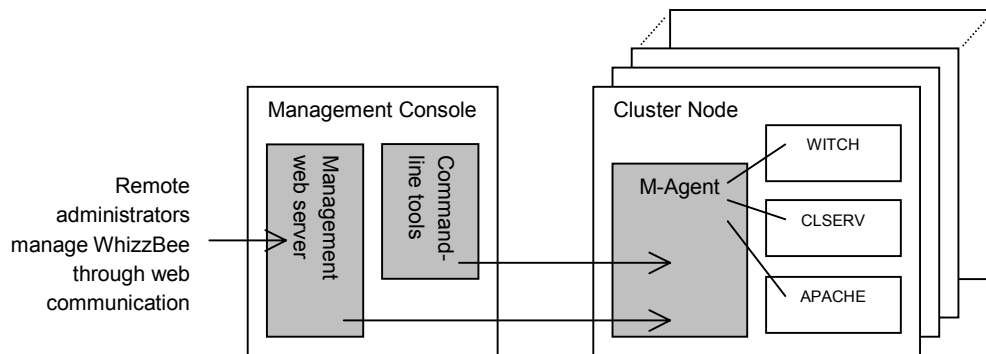


Figure 3: WhizzBee Web Server 1.0a Management Infrastructure

As shown in the figure, each cluster node runs an M-Agent that comes with the WhizzBee Web Server 1.0a package. The M-Agent is responsible for starting/stopping local WhizzBee Web Server services and reporting system statuses and statistics. The Management Console can be a dedicated computer running Linux, or it can be one of the cluster nodes. Remote administrators can connect to the management web server running at the Management Console, which would redirect any management request to the M-Agents in the cluster nodes.

Instead of using the web interface, administrators can also perform the same administrative tasks by running the command-line programs at the Management Console.

3.1 Installation

WhizzBee Web Server 1.0a takes only three steps to install.

1. Installation of the WhizzBee Web Server 1.0a RPM package at each of the cluster nodes. This is the only step that requires the administrator to login to every cluster node. M-Agents are automatically started after the installation of the RPM package.
2. Installation of the WhizzBee Web Server 1.0a Management RPM package at the Management Console.
3. The administrator either runs the command-line configuration tool "wbconfig", or connects to the management web server through a web browser, to configure and to start the WhizzBee Web

Server services. The configuration procedure involves defining the role of each cluster node, and specifying several parameters for the WhizzBee Web Server services.

3.2 Day-to-day Administration

Administrators can easily perform day-to-day administrative tasks by using the convenient tools provided. For instance, administrators can add/delete web server nodes to/from the cluster by issuing a single command. Similarly, WhizzBee Web Server services can be started, stopped, or restarted easily.

3.3 Cluster Monitoring and Event Logging

Activities of the WhizzBee Web Server services are recorded in log files. Besides, the dynamic statuses of the cluster nodes can be gathered and viewed through a web browser at any time. System status information of a cluster node includes memory utilization, CPU load, service healthiness, cache-hit ratio, etc. While the command-line tools can provide sufficient analytical data, the web-based interface can further compile the data into tabular and graphical formats.

4 HIGH AVAILABILITY SUPPORT

Businesses on the Internet are most intolerant of server outages. By using clusters that are equipped with advanced failure detection and recovery software, tasks running on the failed machine can be migrated to, or shared by, other running machines. In this section, we describe the high availability (HA) support of WhizzBee Web Server 1.0a.

The HA design goal of WhizzBee Web Server 1.0a is to remove all the single points of failure in the web server cluster. The following paragraphs describe how this is achieved.

4.1 Dispatcher Failure

The dispatcher is at the heart of the cluster. Its failures could bring down the whole cluster as all web server nodes rely on it to route incoming HTTP requests. Therefore, it has to be designed and implemented very carefully.

As shown in Figure 1, there is a backup dispatcher that sends heartbeat checks to the active one. If the active dispatcher fails to respond to the checks, the WITCH module running at backup dispatcher will automatically be promoted to become the active one to take over the routing responsibility. This ensures that the web server functions would not be affected by dispatcher node failures.

CLSERV also shares similar HA features. There is a stand-by CLSERV running at the backup dispatcher and it would take over the responsibility whenever the online dispatcher fails to respond to heartbeat checks.

4.2 Web Server Node Failure

CLSERV periodically sends heartbeat checks to the M-Agents running at the web server nodes. If a web server node fails to respond to the checks, CLSERV will regard it as offline. In this case, CLSERV will not recommend other web server nodes to request for cache-forwarding from this failed node. Furthermore, CLSERV runs a special system call to inform WITCH to stop dispatching HTTP requests to this failed node.

4.3 Safe Cluster Administration

Besides avoiding failures, WhizzBee Web Server also guarantees that normal administrative tasks would not affect the availability of the cluster. For instance, dynamic addition/deletion of cluster nodes does not require the services to be suspended and restarted.

5 PERFORMANCE EVALUATION

Several benchmark tests have been carried out to evaluate the performance of WhizzBee Web Server 1.0a. The results were compared to those of TurboCluster Server 4.0. This section reports the benchmark and comparison results for these products.

5.1 Test Environment

The tests were carried out on a cluster platform with the following configuration.

Dispatcher	: Pentium III 600 MHz, 128 MB RAM
Web servers x 8	: Pentium III 667 MHz, 256 MB RAM
NFS Server	: Pentium III 600 MHz, 128 MB RAM
OS	: RedHat Linux 6.2 (kernel 2.2.14-50)
Network	: LEVEL ONE 100 Mbps Fast Ethernet network switch
Clients x 9	: Pentium III 600, 128 MB RAM running Windows 98 Second Edition

All the clients and cluster nodes are connected to the Fast Ethernet switch.

5.2 Test Method

Test Suite	: WebBench 3.0
Duration	: 30 minutes
No. of threads	: 5 (per client)
Access Pattern	: Shown in Table 1.

Table 1: WebBench Access Pattern

<i>Dataset Sub-class</i>	<i>File Size (Bytes)</i>	<i>Probability of Retrieval</i>
1	223	0.2
2	735	0.08
3	1522	0.12
4	2895	0.2
5	6040	0.15
6	11426	0.17
7	22132	0.07
8	41518	0.008
9	87360	0.001
10	541761	0.001

Dataset Size : ~1.2GB (20 replicas of the dataset that ships with WebBench)
 Backend-I/O store : Two settings were tested:
 (Setting 1) A centralized NFS server;
 (Setting 2) Local hard disks of the web server nodes.

It should be emphasized that we have run several tests that last for 8 hours, but the results were the same as those that ran for 30 minutes. Therefore, 30 minutes was considered long enough for the servers to reach equilibrium.

5.3 Benchmark Results

Tests were carried out for different numbers of web server nodes in WhizzBee Web Server and the results were compared with those of TurboCluster Server 4.0 with Apache 1.3.14 (TC4). It should be emphasized that the Apache web servers used in both tests have the same set of modules loaded. The only difference lies in that WhizzBee Web Server uses a customized proxy module in order to support cooperative caching, whereas the test for TC4 uses the standard one. All the results are shown in Table 2-4. The terms used in the tables are explained below.

Requests completed per second	The performance of a web server is measured in terms of the number of HTTP requests it can handle. This figure is the total number of HTTP requests the server has successfully handled in one second.
% improve over TC	This is the percentage improvement of the performance that WhizzBee Web Server can achieve over TC4. This percentage is calculated by: $((\text{requests completed/s by WhizzBee Web Server} - \text{requests completed/s by TC4}) / \text{requests completed/s by TC4}) * 100 \%$
Speedup factor	This number represents how well a web server scales up with the number of web server nodes. It is calculated by: $\text{requests completed/s by } n \text{ nodes} / \text{requests completed/s by } 1 \text{ node}$
Global cache hit ratio	This is the percentage of requests that are handled by retrieving files from the cooperative caches, instead of the I/O store. The higher the ratio, the lighter the I/O traffic. This term is only applicable to WhizzBee Web Server.

Table 2: Requests completed per second by WhizzBee Web Server and TC (NFS-mounted dataset)

<i>Number of web server nodes</i>	1	2	4	8
WhizzBee	63	88 (1.4)	248 (3.9)	1233 (19.6)
TC4	60	69 (1.2)	74 (1.2)	76 (1.3)
% improve over TC	5%	28%	235%	1522%

Note: the numbers in parentheses indicate the speed up factor

Table 3: Requests completed per second by WhizzBee Web Server and TC (Local dataset)

<i>Number of web server nodes</i>	1	2	4	8
WhizzBee	61	150 (2.5)	586 (9.6)	1253 (20.5)
TC4	97	171 (1.8)	324 (3.3)	489 (5.0)
% improve over TC	-37%	-12%	80%	156%

Note: the numbers in parentheses indicate the speed up factor

Table 4: Global cache hit ratio of WhizzBee Web Server

<i>Number of web server nodes</i>	1	2	4	8
NFS-mount dataset	13.9	31.5	75.6	100.0
Local dataset	14.3	31.8	73.9	100.0

The following graphs show the results that we have obtained in visual form. They are Requests completed per second vs. number of nodes for NFS-mounted dataset (Figure 4), Requests completed per second vs. number of nodes for local dataset (Figure 5), and Global cache hit ratio vs. number of nodes (Figure 6), respectively.

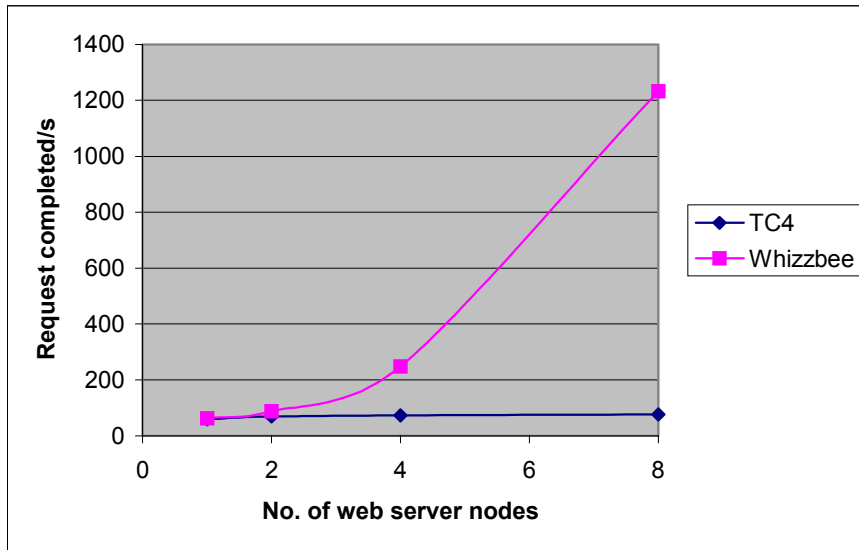


Figure 4: Requests completed/s vs. no. of nodes, NFS-mounted dataset

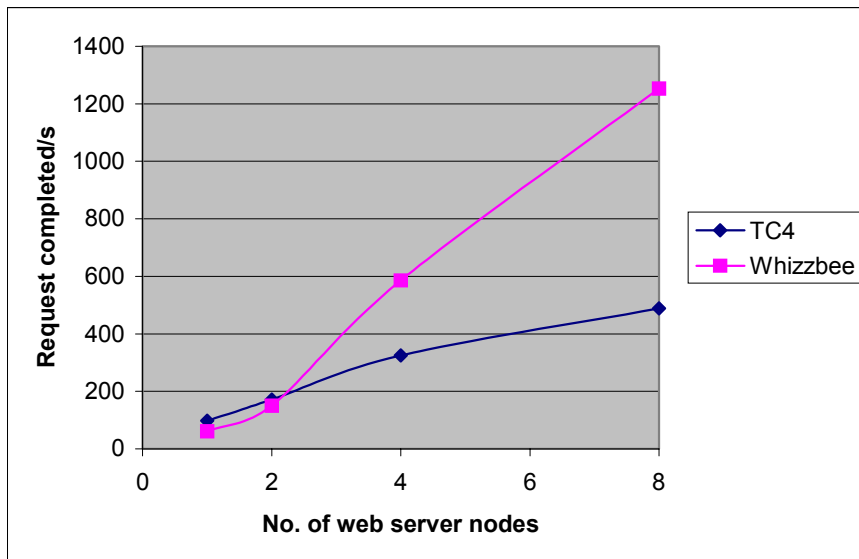


Figure 5: Requests completed/s vs. no. of nodes, local dataset

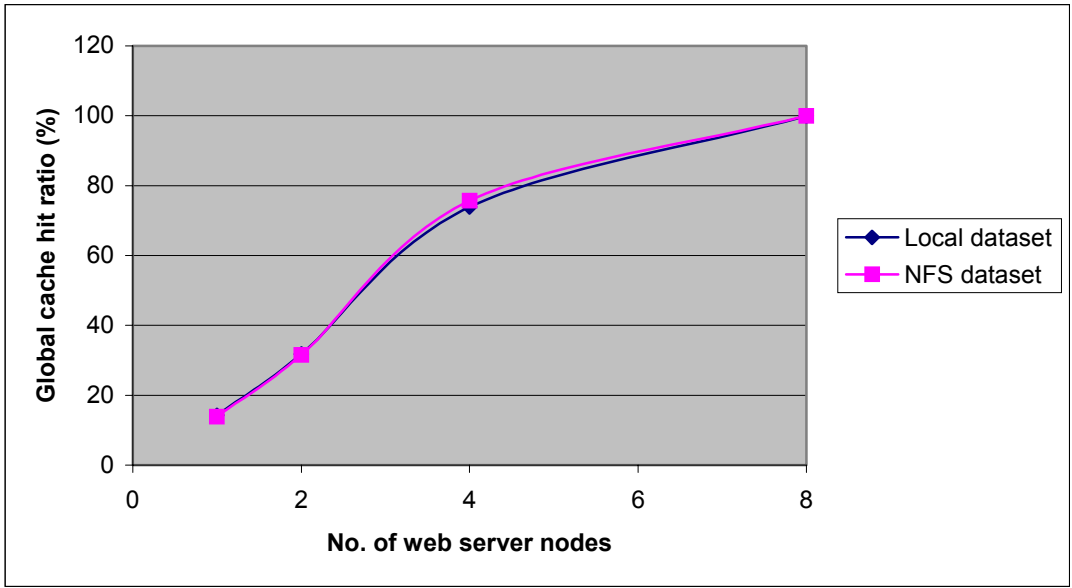


Figure 6: Global cache hit ratio vs. no. of nodes

As shown in the figures, WhizzBee Web Server achieves better scalability and performance than TC4 in most cases. Specifically, WhizzBee Web Server performs roughly 15 times better than TC4 in an 8-node configuration where the dataset is stored in the NFS server. It is obvious that the EC-Cache Engine does have a significant effect on reducing the load of the centralized storage. For the case of local dataset, WhizzBee Web Server still has over 150% improvement over TC4. This indicates that WhizzBee Web Server is able to better utilize the memory resource. In other words, WhizzBee Web Server achieves a considerably lower cost/performance ratio.

6 SUMMARY

WhizzBee Web Server is a next-generation web server that leverages the state-of-art clustering technology to deliver a quality server solution.

Better Performance. The WITCH component supports load-balancing intrinsically with its simple yet intelligent traffic distribution algorithm. All incoming Internet connection requests are distributed quickly based on a prescribed policy among all server components, leading to load-balancing with the fastest response and highest degree of availability. The overall WhizzBee Web Server system performance receives a big push by using the EC-Cache technology where a portion of physical memory in every server component is set aside as cache for cluster-wide access. WhizzBee Web Server intelligently makes use of all such memory pieces as a single system-wide cache resource.

Enhanced Availability. Servers are made more reliable and available with WhizzBee's unique routing technology and intelligent fault-detection and recovery facilities.

Ease of Installation, Administration and Maintenance. WhizzBee Web Server emphasizes the convenience and ease of installation, management and performance monitoring. WhizzBee Web Server provides two options for all tools: command-line and web-based, satisfying different needs and administrative styles. WhizzBee Web Server also maintains system and event logging. Network administrators can be assured that their web services will run non-stop reliably while all common events will be catered for automatically and gracefully. WhizzBee Web Server understands the mobility of nowadays network administrators and thus a web-based remote administration facility is provided. The web-based nature allows system reconfiguration, startup/shutdown and administration to be carried out anywhere.