Bitrix Site Manager

Bitrix Web Cluster

Phone/Fax: +1 (703) 740-8301; e-mail: info@bitrixsoft.com, http://www.bitrixsoft.com

Contents

Phone/Fax: +1 (703) 740-8301; e-mail: info@bitrixsoft.com, http://www.bitrixsoft.com

Computer Clusters

The term “cluster” was first referenced in the 80's of last century by DEC with regard to computer systems. Since then, the cluster is defined as *a group of linked computers, working together closely thus in many respects forming a single computer* (wiki).

In practice, the cluster systems can be:

1. **High availability clusters** (also known as Failover Clusters) are implemented primarily for the purpose of improving the robustness and availability of services that the cluster provides.
2. **High performance clusters** for serving requests that can grow immensely.

Phone/Fax: +1 (703) 740-8301; e-mail: info@bitrixsoft.com, http://www.bitrixsoft.com

Introducing Bitrix Web Cluster

Providing scalable performance and uninterrupted availability is the task of utmost importance for any web project. Any business's website must always be available to the customers and the website pages must load in no time, irrespective of factors that may or may not affect the server operability.

Otherwise, the website visitors will simply be diverted to the rival websites because they could not wait for the product catalog or ordering page to load. What can be more frustrating, the search engines usually give lower ranking to such websites (this is exactly the way the Google engine works).

For large enterprises, it becomes a matter of reputation: a website is the face of the corporation.

At present, such problems can be easily tackled using solutions like Oracle RAC (Real Application Cluster) or Microsoft SQL Server which are undoubtedly trustworthy and effective platforms. The Bitrix solutions support Oracle and MS SQL in major versions. However, the following disadvantages come to light when the customers are making their decision:

- the cluster configurations of Oracle or MS SQL can only be used for providing the database availability and scalability; the problem of web server availability still requires solution;
- such solutions are extremely expensive;
- highly qualified personnel and system administrators is also an essential requirement.

Bitrix Web Cluster offers an integrated solution for all these problems because it provides easily configurable and flexible scalability and availability for the entire web project, not just a database or a web server individually.

There is one more thing that makes Bitrix Web Cluster extremely attractive to business owners and administrators: web clustering is supported at the kernel level of the Bitrix platform. The web developers need to make absolutely no changes in the source code of their web projects to migrate to cluster configuration.

Phone/Fax: +1 (703) 740-8301; e-mail: info@bitrixsoft.com, http://www.bitrixsoft.com

Chapter 1.
Tasks And Goals
Accomplished By Bitrix Web Cluster

Simultaneous resource scalability and assured fault tolerance provided by Bitrix' Web Cluster module solves a few of the most basic and most difficult tasks that face the developers and owners of a mission-critical web project.

Web Project Scalability And Enhanced Performance

As a web project matures, growth in popularity and traffic increases, as do the resources it requires: CPU, memory, disk space.

To a certain threshold, it is enough simply to increase resources. The site may migrate from virtual hosting to a VPS, then on to a dedicated server, with subsequent upgrades. But at some point, however powerful the dedicated server is, it will not be able to support the demands of the site. Furthermore, each update or migration of the site brings with it a breakage in service and a reconfiguration process.

Bitrix Web Cluster provides a flexible solution with real-time scalability of the specific resources (the database or web server) which need to be expanded, simply by adding new machines to the cluster.

Key features of the technology:

- machines (nodes) in the cluster may have completely different configuration: a virtual host, VPS, dedicated server, cloud instance, etc.;
- nodes can be distributed in any manner needed, even being located in separate data centers;
- adding nodes increases site performance proportionally to the capacity of the new resource.

Example: if your site is working on a single dedicated server with an Intel Quad Core processor, 4 Gb RAM and can process 40 queries per second during peak traffic, then the addition of a second server of similar capacity in a web cluster formation will increase performance by approximately 90-95%, processing up to 77 queries per second at peak load.

Similar additions to the cluster can be made until the desired level of performance is reached.

Phone/Fax: +1 (703) 740-8301; e-mail: info@bitrixsoft.com, http://www.bitrixsoft.com

Backup Of System Nodes, Fault Tolerance And Continuity Of Service

The second important issue is to ensure fault tolerance of the system and minimize downtime in the case of server failure or routine maintenance.

If a site is on a single server and that server goes off line, or when there is an emergency in a datacenter, the interruption in service can lead to many unpleasant circumstances:

- lost orders (for e-commerce sites);
- lost income from media or from context ads;
- damage to reputation;
- risk of loss of position in search engine rankings if indexing occurs during downtime.

Clustering of all site components (web servers and databases) with Bitrix Web Cluster minimizes downtime. Depending on how load balancing is managed throughout the cluster, downtime can either be greatly reduced, or in some situations completely eliminated.

How it works: when a node goes down, the cluster recognizes that the resource is offline, be it a database, web server, or memcached server, and the load on the system automatically is re-distributed among the remaining nodes.

Time of page creation may increase proportionately during peak load periods, but service will not be interrupted. If there is adequate reserve capacity in the cluster, performance during non-peak times will continue without perceptible change.

After the failed server(s) is diagnosed and the problems are corrected, it can be returned to work in the cluster to maximize site performance.

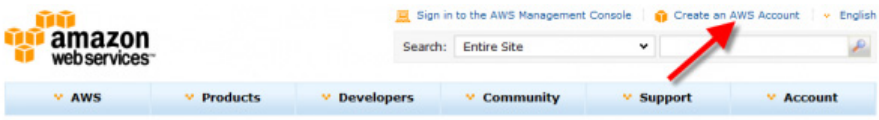
Chapter 2. Sample Implementation Using Amazon Web Services

Bitrix Web Cluster can be deployed on any hosting platform ranging from cloud hosting virtual machines to dedicated servers. At least two separate servers are required to render the cluster solution robust and reliable.

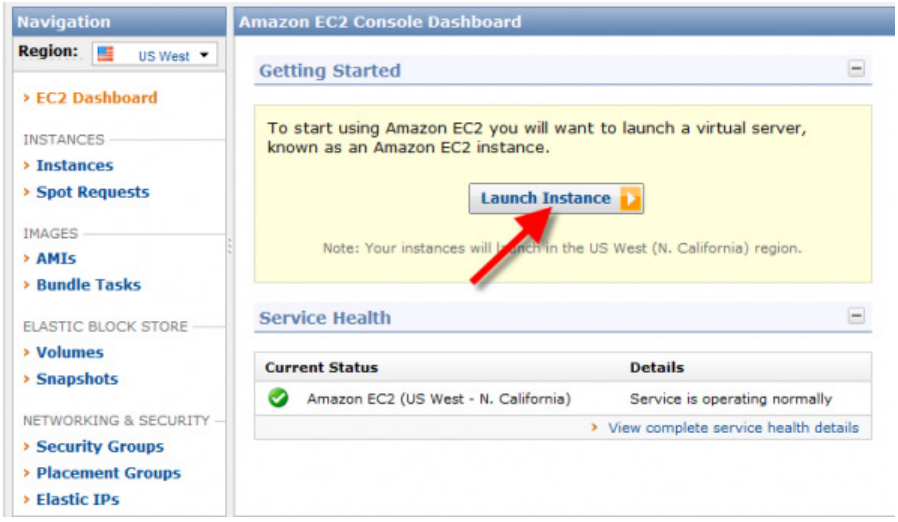
Creating Virtual Machines

This chapter discusses the process of the creation and configuration of the web cluster using the two Amazon Web Services virtual machines running on a cloud hosting platform. It is particularly convenient in its ability to provide the virtual servers of any configuration we may require.

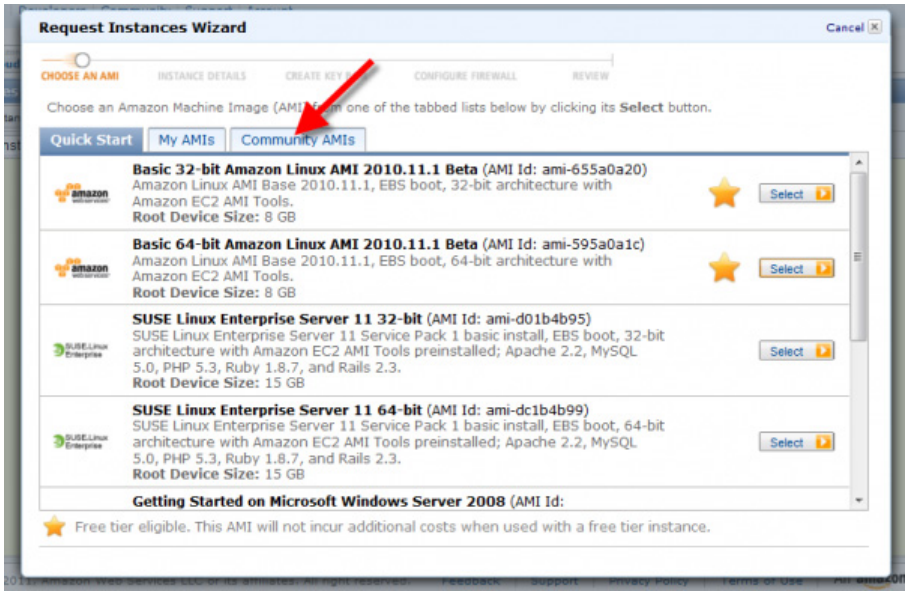
Create an Amazon Web Services account :



Then, create two virtual machines. Click **Launch Instance**:

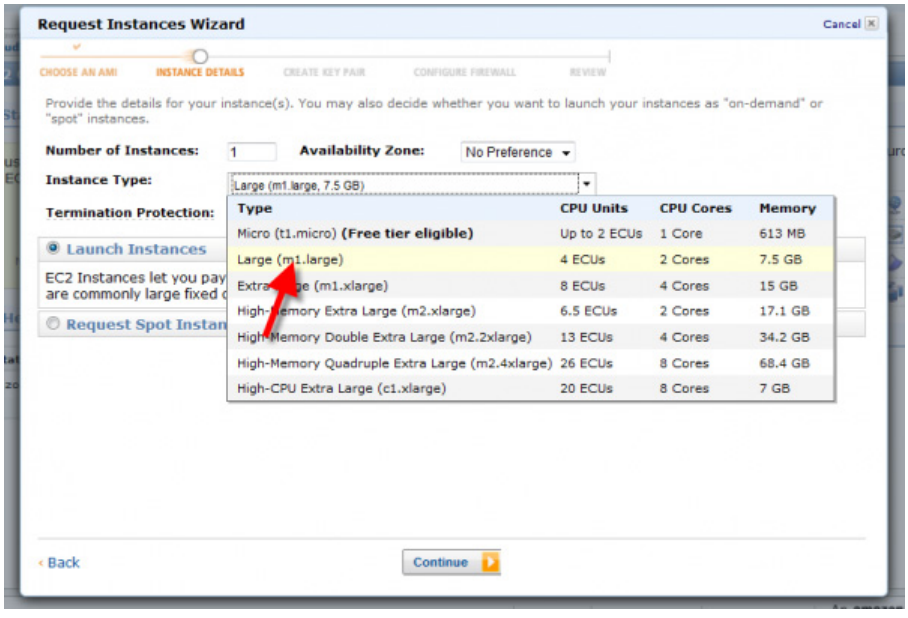


Click the tab **Community AMIs**. At the time of your reading this paper, the Bitrix Virtiual Machine AMI images should already be available for downloading.

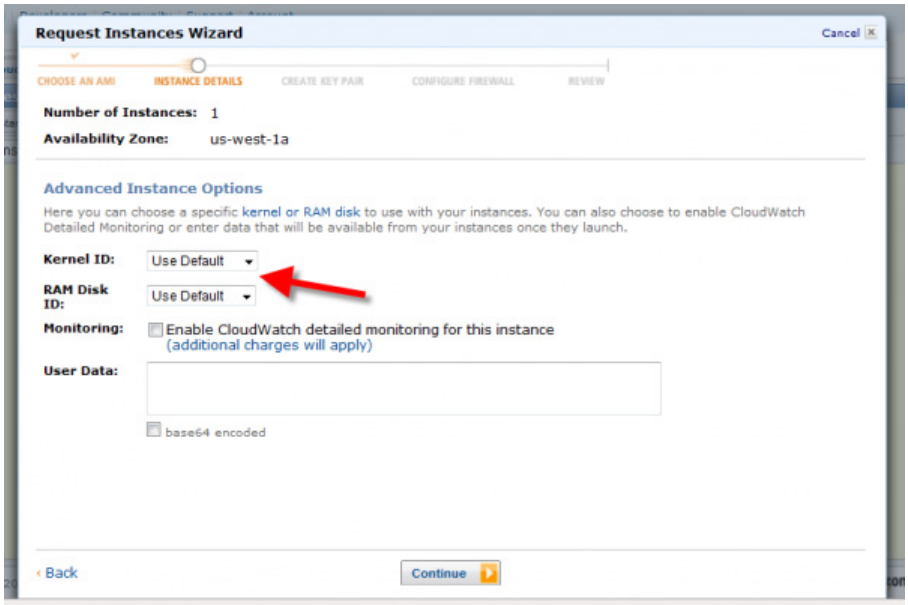


Select the image **ami-6d7f2f28** containing CentOS_5.4_x64. Bitrix Web Environment for Linux also supports Fedora 8-14 (i386) and Red Hat Enterprise Linux 5 (i386, x86_64).

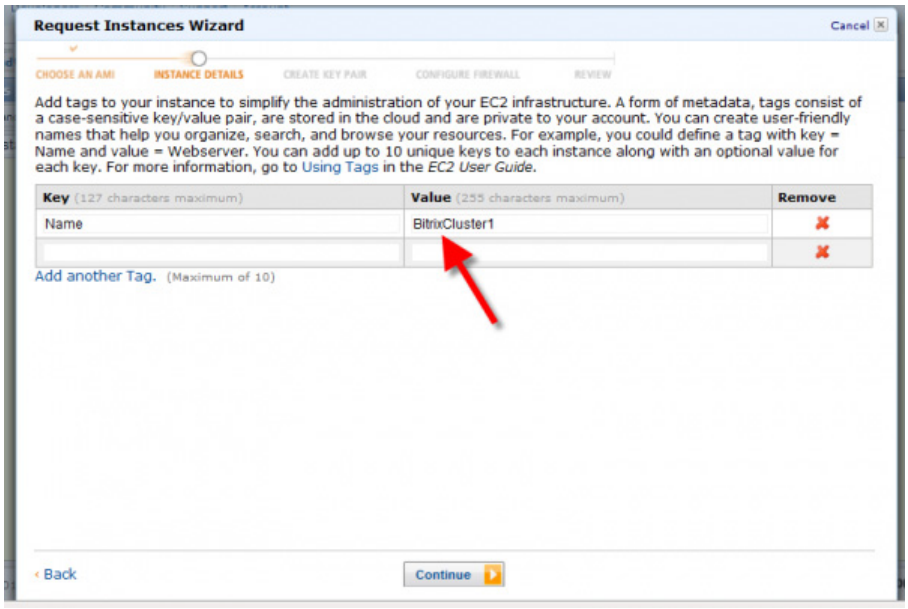
Depending on the expected load, select the hardware configuration for your virtual machine. In our example, we select "m1.large": 2 cores 2 GHz with 7.5GB RAM:



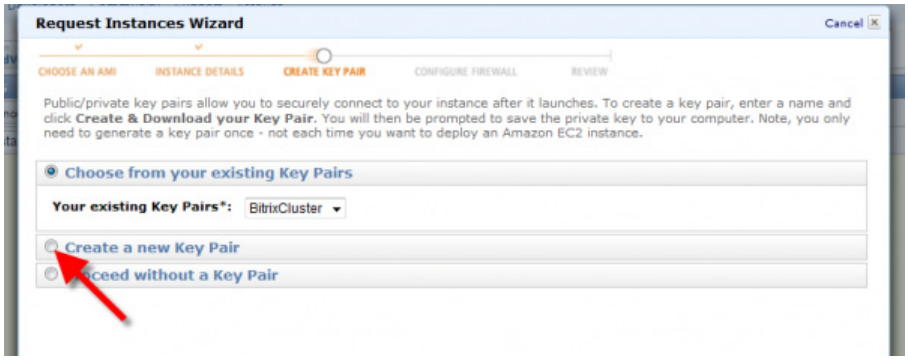
Leave the *Linux kernel* and *ramdisk* as is:



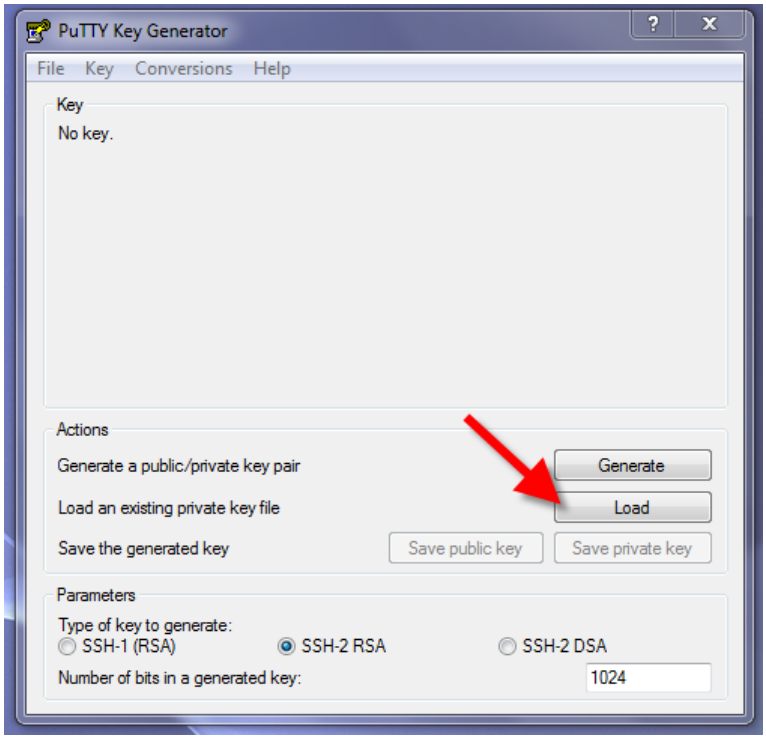
Give your machine a readable and clear name:



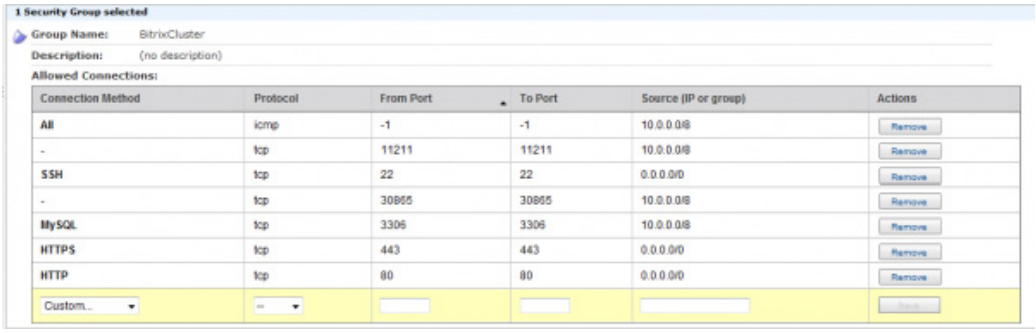
Create or select a key pair for further authorization using SSH:



If you are on a Windows platform, import the private key you have just created to PuTTYgen and save it as a Putty private key. Since the virtual machines use the private key for access, it is a good idea to provide a strong password for the key.

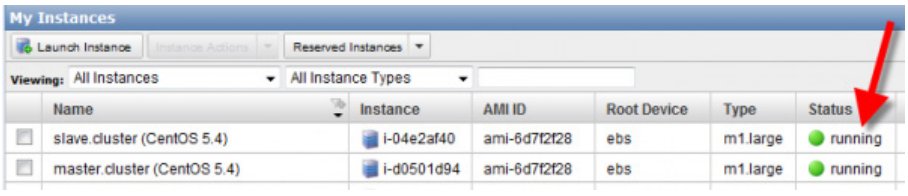


Create or select a previously selected security group and provide the following parameters:



The port 11211 is used by **memcached**, and the port 30855 – by **csync2**. As soon as you have finished configuring the cluster, remember to close access to memcached's ports.

Run the second VM in the similar fashion:



In the end, we have two running virtual machines with CentOS 5.4 64 bit installed.

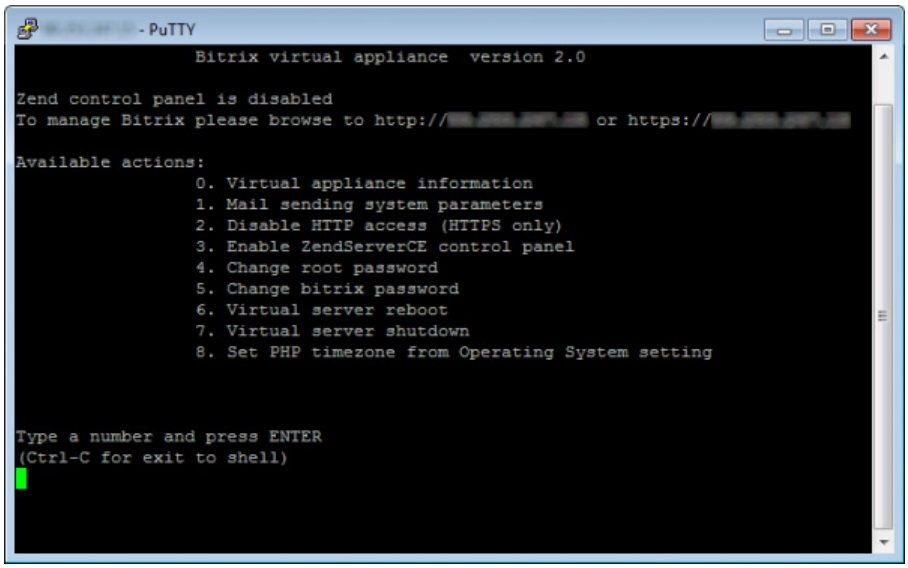
For this platform we are using the package Bitrix Web Environment 2.0 for Linux which is available free of charge. This suite has been designed for fast and undisturbed installation and configuration of the entire set of software required by Bitrix solutions on Linux platforms.

Bitrix Web Environment installs the following applications:

- mysql-server 5.*;
- web-server (Apache 2.2.*);
- zend-server-ce-php-5.3;
- mod-php-5.3-apache2-zend-server;
- geoip module;
- nginx;
- memcached;
- stunnel.

You are free to use any other web platform, OS or server software configuration. However, Bitrix Web Environment:

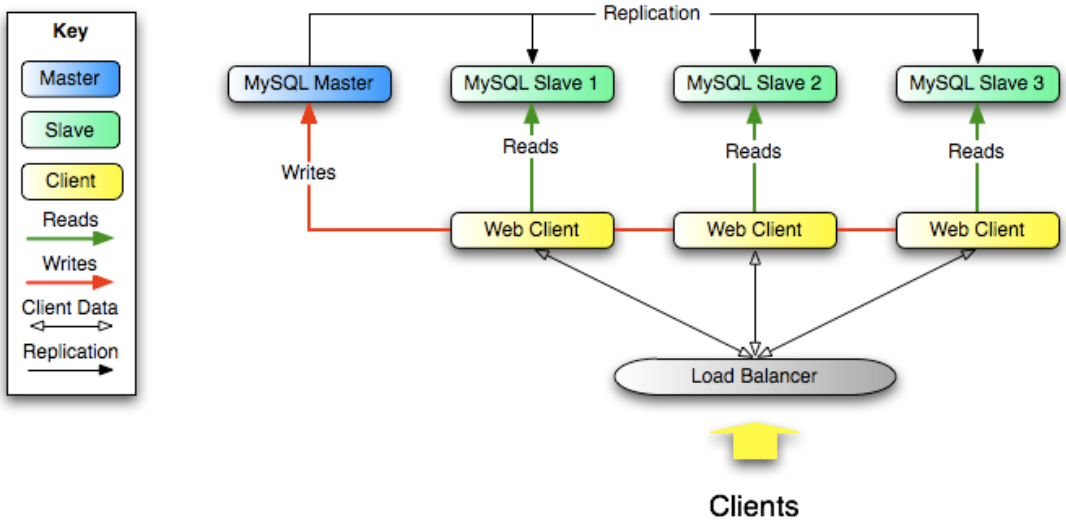
- provides two- to three-fold server performance boost compared to default software and configuration;
- requires minimum time and effort to deploy the system: you get a fully configured environment in minutes; Bitrix Web Environment saves 200 to 300 hours of system administration;
- Bitrix Web Environment is well balanced for heavy load.



MySQL Replication Configuration. Slave-To-Master Emergency Switching

MySQL replication is usually performed to achieve the following goals.

1. Increase database performance by adding more servers to it for better adaptation to growing load. The increase in performance is due to allocation of a master server for saving the new data and a slave server for reading the existing data. This solution proves itself efficient when used with web applications because most of their database requests are to retrieve data.
2. Because data is replicated to the slave, and the slave can pause the replication process, it is possible to run backup services on the slave without corrupting the corresponding master data. To perform a whole binary backup (which may be essential with InnoDB), the master can be temporarily suspended while the backup is being performed and then restarted.
3. Online backup and data accessibility: whenever the master goes offline, one of the slaves always has the up-to-date copy of the data; the slaves continue to serve requests.



For more information about MySQL database replication and use cases, please refer to MySQL database documentation.

Achieving Robust Replication

For maximum reliability of replication, configure the MySQL parameters as follows:

- innodb_flush_log_at_trx_commit = 1
- sync_binlog = 1
- sync_relay_log = 1
- sync_relay_log_info = 1

However, these settings, while being the most reliable solution, may cause database performance degradation. For better performance, you can use just a subset of these parameters. The drawback is that the loss of transaction data is possible if a transaction is under way while the database server accidentally goes offline or crashes. The faster but less robust settings are as follows:

- innodb_flush_log_at_trx_commit = 2

- sync_binlog = 0

Dynamic Hostname

If the server has a dynamic IP address (or a hostname), the following parameters need to be set explicitly: relay-log, relay-log-index.

Privileges

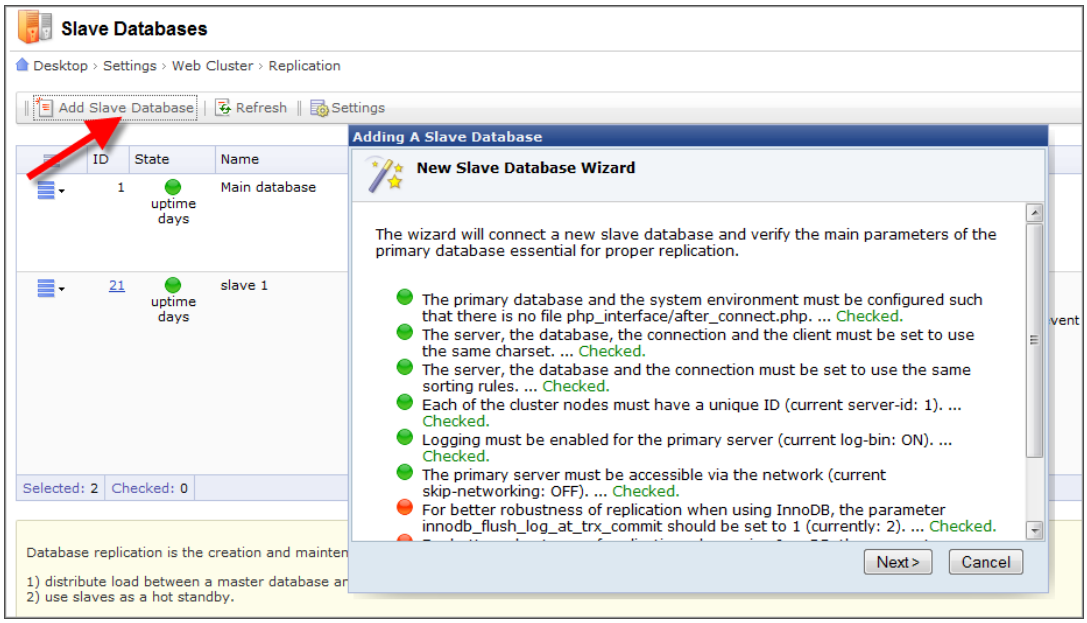
Setting up replication requires that the access rights SUPER, REPLICATION CLIENT and REPLICATION SLAVE are granted to the master and the slaves.

Time Zones

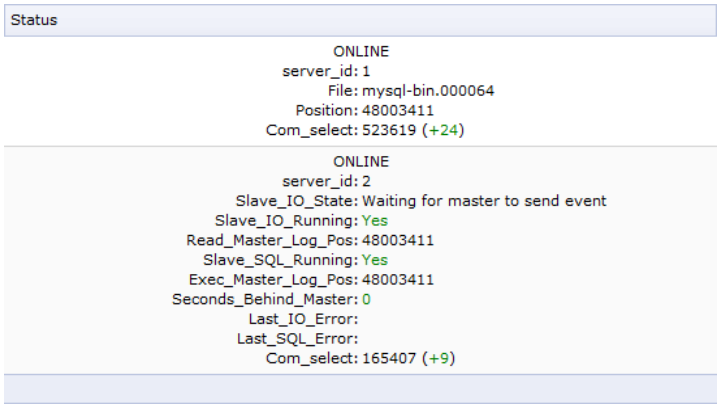
If the database cluster servers are physically placed in different data centers, you have to set the same system time zone for both the master and the slaves.

Setting Up Replication In Control Panel

When you add a slave server, the replication wizard verifies the parameters required:



After verification, Control Panel will show the slave status:



Replication Administration

Once configured, replication performs smoothly and requires only minimum administration. However, it is recommended to check it now and then using the OS monitoring tools (nagios, zabbix, monit, linux-ha).

In the unlikely event of an error occurring on the slave side, the latter needs to be reinitialized by loading data from the master. To do so, open *Settings* > *Web Cluster* > *Replication* in Control Panel; click the button **Stop Using Database** and then click **Use Database**.

Backup

Slaves can be safely stopped for the purpose of the creation of logical or binary backup copy using the MySQL and operating system tools. The master server remains online and continues processing requests.

Slave-To-Master Switching

In the event of the master server going offline, the cluster needs to be switched to using another server as master, manually or by running a script. It is common to use a slave server storing the most recently replicated data.

The following guidelines show how to switch slave to master safely.

- Close client access to the web application

If a two tier configuration is the case (nginx front-end and Apache back-end), it is recommended to disable the front-end's access to back-end returning a maintenance message to clients instead.

- Wait for the slaves to finalize synchronization

Stop the binary log thread at all slaves:

```
STOP SLAVE IO_THREAD;
SHOW PROCESSLIST;
```

Wait for the slave thread (SQL_THREAD) to return the message *"Has read all relay log; waiting for the slave I/O thread to update it"*. It is not advised to stop the slave server by executing 'STOP SLAVE' because not all SQL commands of the relay log might be fulfilled (due to delays etc.) while switching to master mode wipes the relay log causing potential data loss.

- Prepare a new master server

Make sure the target slave server maintains binary log while not registering master requests:

```
SHOW VARIABLES LIKE 'log_bin';
log_bin       | ON
SHOW VARIABLES LIKE 'log_slave_updates';
log_slave_updates | OFF
```

Now, stop the slave server completely (both the binary log read and SQL command threads):

```
STOP SLAVE;
RESET MASTER;
```

RESET MASTER is required to clear the new master's binary log, otherwise the old commands may be executed at the connected slave servers. This might be the case (rare but possible) when a server had been used as master with binary log on, then turned to slave mode not using binary log and is now finally being turned back to master mode again.

Now the new master server is ready to process requests.

- Reconnect slaves to a new master

Run the following commands at each slave:

```
STOP SLAVE;
CHANGE MASTER TO MASTER_HOST='#new_master_host_name#';
START SLAVE;
```

Executing 'CHANGE MASTER ...' clears the slave's relay log and sets the default read position of the slave's master binary log reader (which is the first log file, position 4 – at the very beginning of the master's binary log).

- Switch the web application to the new master server

Now the web cluster needs to be reconfigured to use the new master server. Add the server information ($DBHost) to */bitrix/php_interface/dbconn.php*. If the static content synchronization interval is too long, start the synchronization process at master manually by executing the command 'csync2 - x' (see 4.4.2. Synchronization: Using Special Sync Mechanisms).

- Open access to the web application

For a two tier configuration, remove the maintenance message and enable the front-end's access to back-end.

Now the cluster is using the new master server.

- Connect the old master server as slave

Open *Settings* > *Web Cluster* > *Replication* in Control Panel and click the button **Add Slave Database**. Follow the wizard instructions. The server will be reinitialized; master data will be propagated to the new slave server.

Vertical Sharding: Delegation Of Queries From Selected Modules

The data tables of the modules of your choice can be moved to another database thus rebalancing requests across multiple servers. In Control Panel, go to *Settings* > *Web Cluster* > *Sharding* and click **Add New Database**. After the database has been connected, you have two options of moving the module data.

The first method is for MySQL only. Open the module databases page, pick a required database and select **Use Database** in the database's action menu. The data migration wizard will start.

Another method is to uninstall the module and install it again. The first step of the installation wizard shows a list of databases available from which you can pick a destination database for the module data. However, this method does not allow you to transfer the module's existing tables to a new installation.

The following modules currently support vertical sharding:

- Web Analytics
- Search

Web Server Clustering

The use of multiple servers implicates that any request can be processed by any of the web cluster nodes. Such modus operandi entails several requirements.

1. **Each of the servers must have the same information (files).** For example, an image file uploaded by a user to one of the servers must be available at any other server.
2. **User session must be transparent across all the servers within the cluster.** When a user becomes authorized at one of the servers, they must be recognized as authorized at any other server. Likewise, ending a session at any one of the servers must end it at all the other servers.

It is worth noting that not all of these problems can be fully addressed at only the web application level. Nevertheless, the discussion below discloses possible

approaches to tackling these issues; shows the implementation examples and describes the clustering tools provided by the Web Cluster module.

The following two methods exist to synchronize server data.

- The use of shared data storage.
- The use of the synchronization mechanisms to put the local storages of different servers in sync.

We shall discuss each of the two methods below, and describe how to use the memcached server pool to reduce the number of the files that need to be synchronized.

Each server of a web cluster can be included in the server monitor using the Control Panel form at *Settings* > *Web Cluster* > *Web Servers*.

At each server, a special page must exist showing the Apache web server statistics (using mod_status). For Bitrix Web Environment, follow the guidelines below.

1. Add the following lines to the Apache configuration file (*/etc/httpd/conf/httpd.conf*):

```
<IfModule mod_status.c>
ExtendedStatus On
<Location /server-status>
    SetHandler server-status
    Order allow,deny
    Allow from 10.0.0.1
    Allow from 10.0.0.2
    Deny from All
</Location>
</IfModule>
```

The *Location* directive denotes the address at which the statistics is going to be available. Namely, the *Allow from* lines define the IP addresses of the servers. You can specify all the web cluster servers.

Reload the Apache configuration files using the command:

```
# service httpd reload
```

2. Edit the nginx configuration files (*/etc/nginx/nginx.conf*) by adding the following lines to the first section (*server*):

```
Location ~ ^/server-status$ {
  proxy_pass http://127.0.0.1:8888;
}
```

Reload the nginx configuration files:

```
# service nginx reload
```

3. Edit /home/bitrix/www/.htaccess:

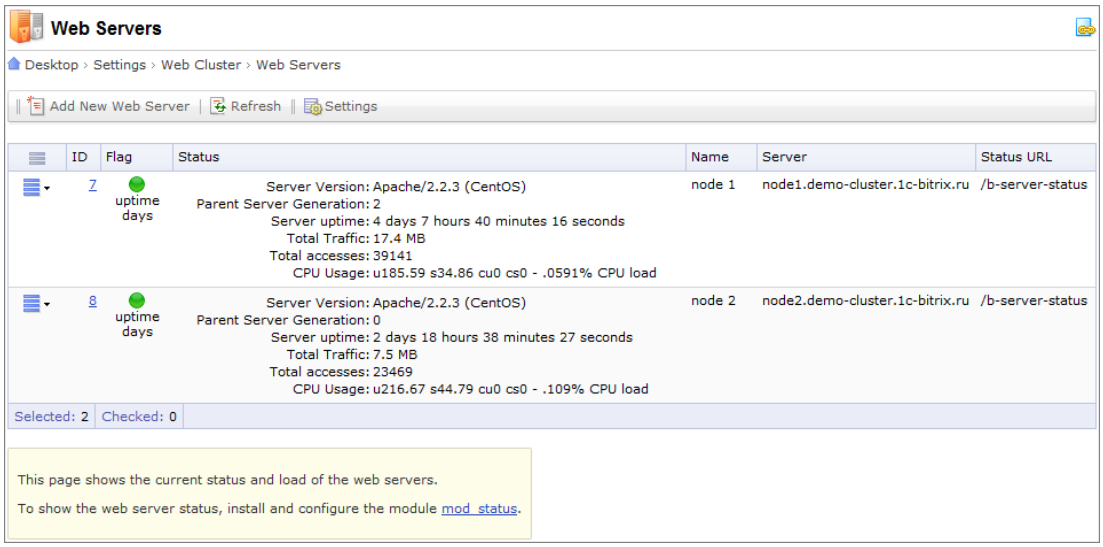
find this line:

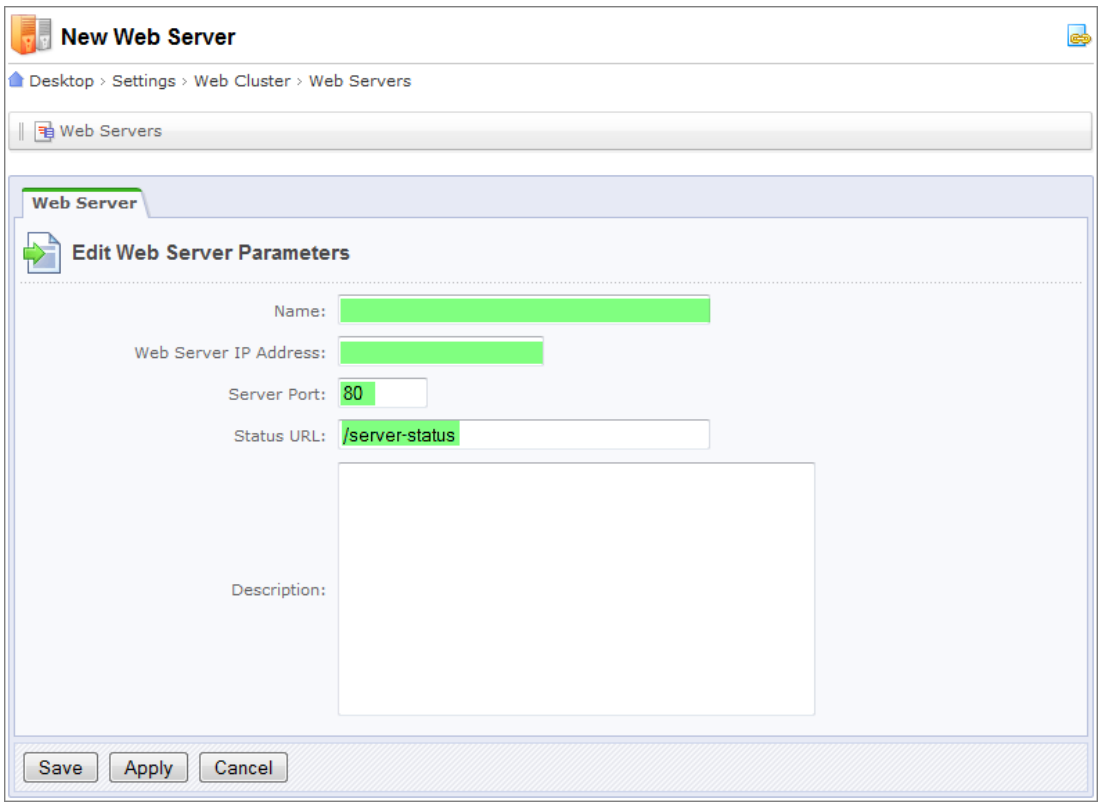
```
RewriteCond %{REQUEST_FILENAME} !/bitrix/urlrewrite.php$
```

add the following line immediately after it:

```
RewriteCond %{REQUEST_URI} !/server-status$
```

Once you have done with the changes, add the *server-status*' address to the cluster configuration:





Synchronization: Using A Shared Data Storage

There are many known different approaches to setting up a shared data storage among which are the straightforward sharing of the Bitrix system folder at one of the nodes and the more sophisticated tricks like using large slave SAN/NAS solutions.

NAS service using NFS or SMB/CIFS

If a web cluster is comprised by the two nodes, one of the them may host a NFS/CIFS server processing file requests from the other nodes.

For better performance, a dedicated, optimized for file operations NFS/CIFS server can be set up for using by all the cluster nodes.

You may be interested in the following technologies to achieve the maximum performance and robustness of the web cluster storage: OCFS, GFS, Lustre, DRBD, SAN.

Synchronization: Using Special Sync Mechanisms

A completely different approach is the use of distributed storages whose data is continuously synchronized.

There exist a lot of software applications providing the means necessary for such synchronization, ranging from simple file copy tools (ftp, scp) to specialized server synchronization programs (rsync, unison).

After much research, we have come to using csync2 in the web cluster due to the following features it provides.

- High performance and speed.
- Low resource consumption (CPU and disk operations). This allows running the sync process as frequently as needed thereby mirroring data at the distributed servers almost instantly.
- This tool is easy to configure to support as many servers as required.
- The file delete operations can be synchronized.
- Secure data exchange (SSL).

To get the full documentation on using and configuring csync2 please refer to the official website: http://oss.linbit.com/csync2/. In this paper, we are going to elaborate on a practical, real world implementation of a two server web cluster using csync2.

Some of the Linux distribution packages (e.g. Ubuntu) have csync2 in their repository available. However, this is not the case with CentOS: csync2 must be installed manually.

Here is one of the possible installation techniques.

- Install the required packages (csync2 uses rsync; xinetd is going to be used to run the csync2's server):

```
# yum install rsync librsync-devel xinetd
```

- Install libtasn1 – a library to handle X.509 (required by csync2):

```
# wget
http://ftp.freshrpms.net/pub/freshrpms/redhat/testing/EL5/cluster/x86_64
/libtasn1-1.1-1.el5/libtasn1-1.1-1.el5.x86_64.rpm
# wget
http://ftp.freshrpms.net/pub/freshrpms/redhat/testing/EL5/cluster/x86_64
/libtasn1-1.1-1.el5/libtasn1-devel-1.1-1.el5.x86_64.rpm
# wget
http://ftp.freshrpms.net/pub/freshrpms/redhat/testing/EL5/cluster/x86_64
/libtasn1-1.1-1.el5/libtasn1-tools-1.1-1.el5.x86_64.rpm
# rpm -ihv libtasn1-1.1-1.el5.x86_64.rpm libtasn1-devel-1.1-
1.el5.x86_64.rpm libtasn1-tools-1.1-1.el5.x86_64.rpm
```

- Install sqlite (version 2):

```
# wget
http://ftp.freshrpms.net/pub/freshrpms/redhat/testing/EL5/cluster/x86_64
/sqlite2-2.8.17-1.el5/sqlite2-2.8.17-1.el5.x86_64.rpm
# wget
http://ftp.freshrpms.net/pub/freshrpms/redhat/testing/EL5/cluster/x86_64
/sqlite2-2.8.17-1.el5/sqlite2-devel-2.8.17-1.el5.x86_64.rpm
# rpm -ihv sqlite2-2.8.17-1.el5.x86_64.rpm sqlite2-devel-2.8.17-
1.el5.x86_64.rpm
```

- Now, install csync2:

```
# wget
http://ftp.freshrpms.net/pub/freshrpms/redhat/testing/EL5/cluster/x86_64
/csync2-1.34-4.el5/csync2-1.34-4.el5.x86_64.rpm
# rpm -ihv csync2-1.34-4.el5.x86_64.rpm
```

- After the installation, the SSL certificates are available. If required, you can generate them anew:

```
# openssl genrsa -out /etc/csync2/csync2_ssl_key.pem 1024
# openssl req -new -key /etc/csync2/csync2_ssl_key.pem -out
/etc/csync2/csync2_ssl_cert.csr
# openssl x509 -req -days 600 -in /etc/csync2/csync2_ssl_cert.csr -
signkey /etc/csync2/csync2_ssl_key.pem -out
/etc/csync2/csync2_ssl_cert.pem
```

- Now generate a key file at one of the hosts:

```
# csync2 -k /etc/csync2/csync2.cluster.key
```

Once the key file has been generated, copy it to all the servers that will be included in the synchronization process, to */etc/csync2/*.

- The following code is an example of the configuration file */etc/csync2/csync2.cfg*:

```
group cluster
{
  host node1.demo-cluster.ru node2.demo-cluster.ru;

  key /etc/csync2/csync2.cluster.key;

  include /home/bitrix/www;
  exclude /home/bitrix/www/bitrix/php_interface/dbconn.php;

  auto younger;
}
```

Here:

- *host* is one or more addresses of the server whose data will be synchronized;
- *key* is the key file;
- *include* is the directory whose contents will be synchronized;
- *exclude* specifies the directories that you do not want to get synchronized;
- *auto* defines the conflict resolution strategy to be applied when the same file is changed at all the servers simultaneously. The parameter "*younger*" gives priority to the server possessing the latest copy according to the time stamp.

- The directive *host* deserves special attention.

In the csync2's configuration file, more than one group (according to the group *"directive"* as shown above) may exist. csync2 ignores the groups in which no machine's local name is specified (which can be obtained by running the command *hostname*).

Therefore, the machine names must not change. To define IP's for the known machine names, use the file */etc/hosts*. To assign a fixed name to a server, specify it in any file, for example:

```
# echo "node1.demo-cluster.ru" > /etc/hostname
```

And then, assign it in the startup script file */etc/init.d/network*. Add the following line to */etc/init.d/network*, prior to the *exit* command:

```
/bin/hostname -F /etc/hostname
```

9. If each new server comes with a complete copy of data existing on other servers (for example, with Amazon EC2 it can be done by creating a snapshot of any server *instance*, and then feeding it to a new *instance*), you can quickly initialize the files using the command:

```
# csync2 -cIr
```

This command must be executed at each host.

- At each host, csync2 should includes the two parts: the server and the client. To enable the server component of csync2, comment the line *"disable = yes"* in */etc/xinetd.d/csync2*, restart the service xinetd and set the latter to run at the system start-up:

```
# service xinetd restart
# chkconfig xinetd on
```

- The client is started by running *csync2* with an *"-x"* key.

You can determine the required frequency of data updates and set *csync2* to run using *cron*. To do so, add the line to */etc/crontab*:

```
*/5 * * * * root /usr/sbin/csync2 -x >/dev/null 2>&1
```

This command sets *csync2* to run each 5 minutes.

Cache Clustering (memcached)

For centralized and robust cache data storage, the Bitrix solutions takes advantage of the memcached server cluster which is in use by many web project like Youtube, Facebook, Twitter, Google App Engine just to name a few.

To run the memcached server on Bitrix Web Environment (Linux), execute the commands:

```
# chkconfig memcached on
# service memcached start
```

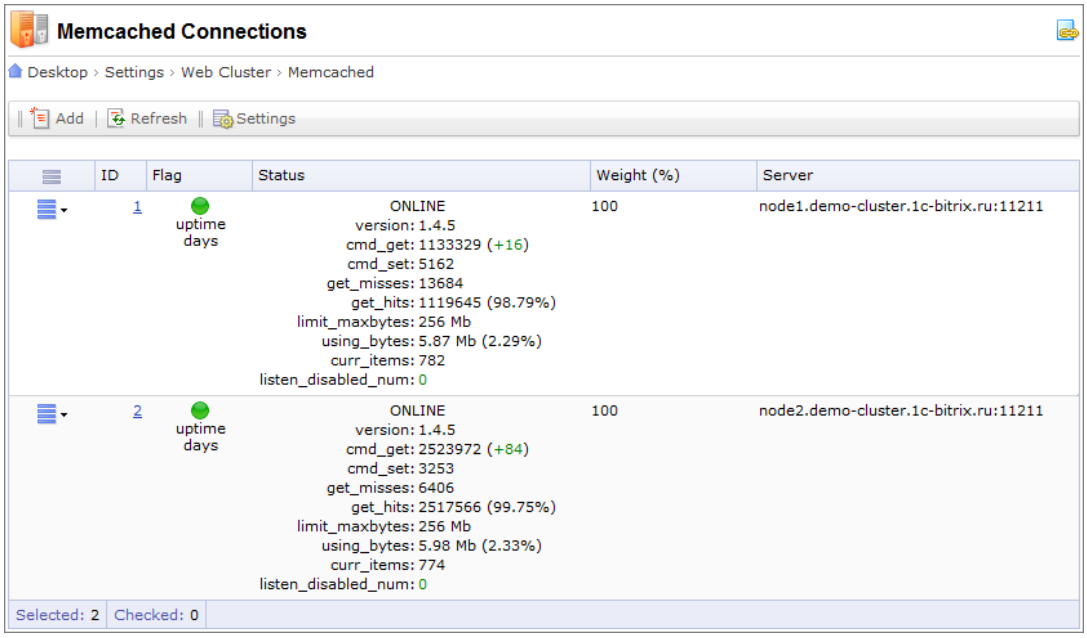
Centralization Brings About Better Performance

For the sake of maximum performance of the web cluster, a centralized shared cache data storage has been devised. The cache data created by any node can be accessed and used by the other nodes of the cluster. A positive effect is that the larger is the node, the more effect the centralized cache produces.

Discarding Old Data For Better Performance

memcached uses the LRU algorithm which discards the least recently used data first. In practice, this prevents the cache from growing infinitely which may be the case due to web developers errors. The LRU algorithm automatically tracks the age of cache lines and maintains the maximum possible efficiency of the cache: both in terms of speed and resource consumption.

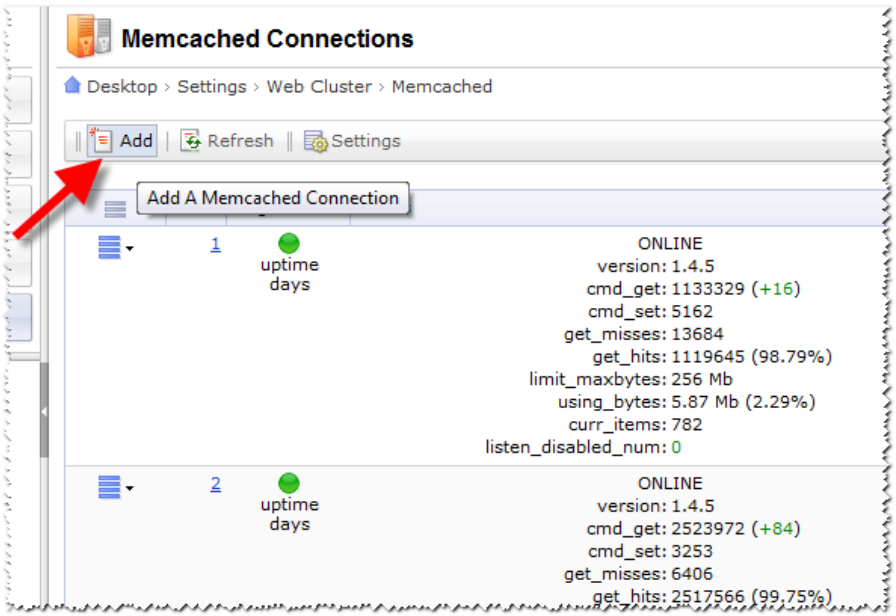
It is recommended that you watch the cache consumption for some time to choose the optimum cache size:



If the memcached servers are different in disk or RAM size (which may affect the cache size and performance), you can balance the server load and usage by assigning the appropriate weight values.

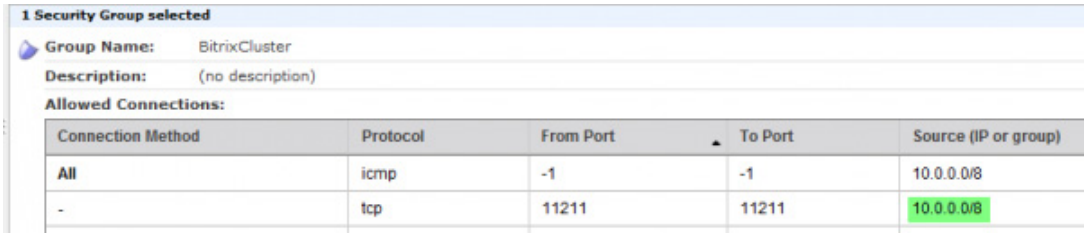
Improved Robustness And Scalability

Since the memcached cache data is distributed across multiple servers of the cluster, one or more (but not all, evidently) servers going offline will not disable the whole cache subsystem. If the cache size increases due to natural reasons, just connect a new memcached server to the cluster:



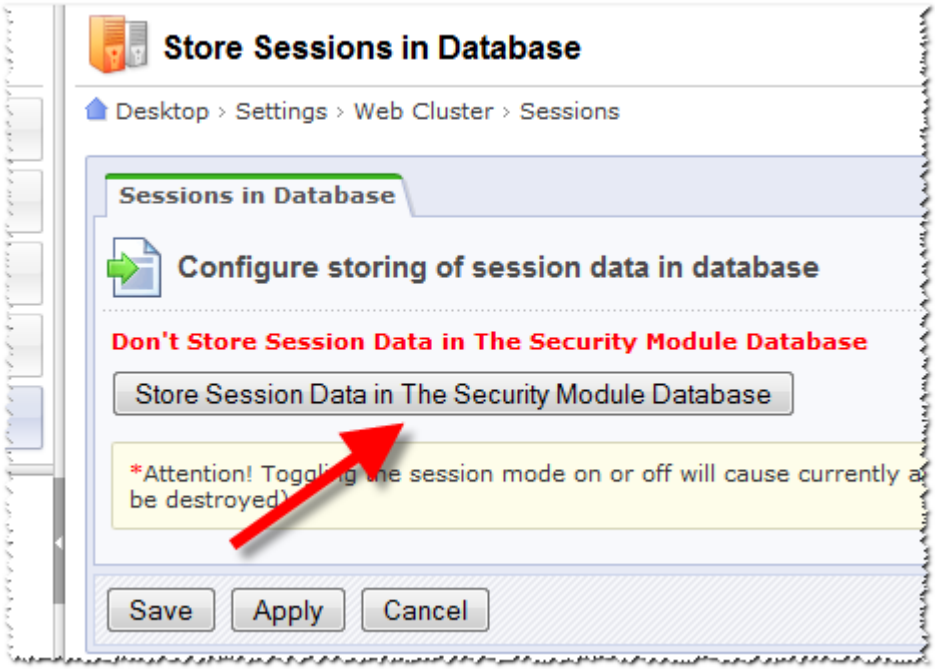
Security

The firewall must be configured to have the memcached servers accessible by the web cluster nodes *only*. When using Amazon Web Services, edit the security parameters as the example below shows.



Choosing Web Session Storage Location

Initially, the user session data is stored in the web server's file system. To view the current session storage status, go to *Settings* > *Web Cluster* > *Sessions*:

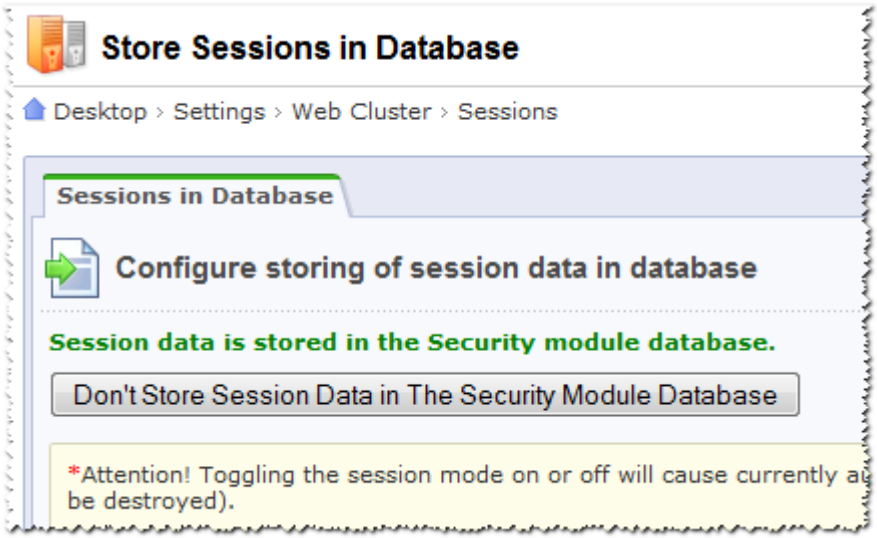


This session storage method is obviously the most convenient when web clustering is not the case. It provides highest performance possible: stress tests indicate that the page generation time is 3 to 5 percent less compared to storing sessions in the database.

However, using multiple servers in a web cluster may produce situations when a user authorization request is processed at server **A**, while the subsequent requests come to other servers – **B**, **C** etc. at which a user will not be recognized as authorized. Such behavior is obviously unacceptable.

The user session must be transparent across the entire web cluster.

This is easily achieved by storing the session data at the database. To enable this mode, click **Store Session Data in The Security Module Database** at *Settings* > *Web Cluster* > *Sessions* as shown above. The following figure shows the same Control Panel form after the mode has changed:



Methods To Balance Web Cluster Node Load

There is a wide range of load balancing tools, software and hardware. The hardware appliances (for example Cisco CSS) are usually expensive, while software tools are reasonably priced and almost as efficient as the hardware solutions.

Load Balancer (Amazon Web Services)

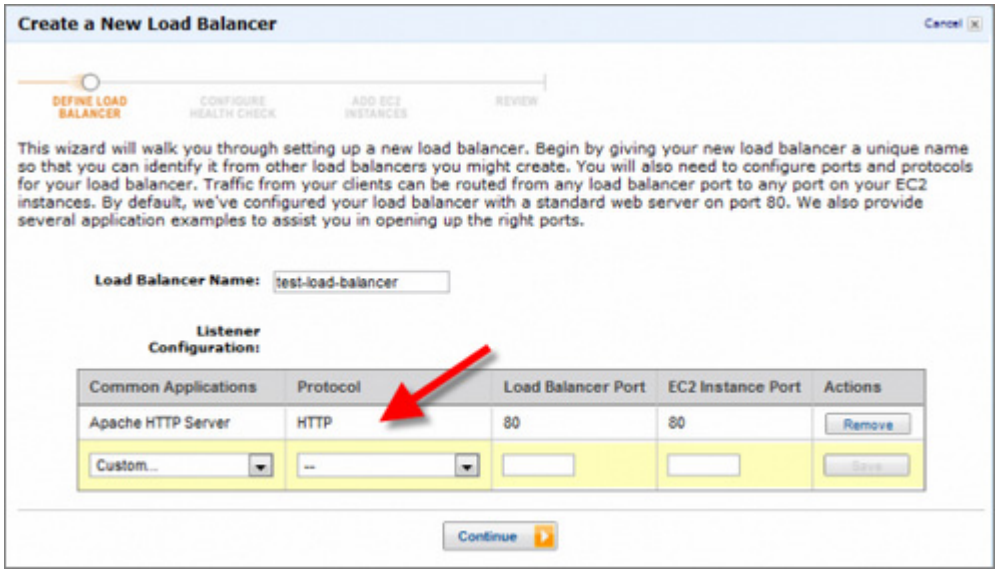
AWS Load Balancer uses a very simple algorithm (*"round robin"*) and is easy to configure. Notice that the balancer's DNS name is created automatically and you will have to create an easy readable CNAME:

```
my-load-balancer-1820597761.us-west-1.elb.amazonaws.com ->
www.mydomain.com
```

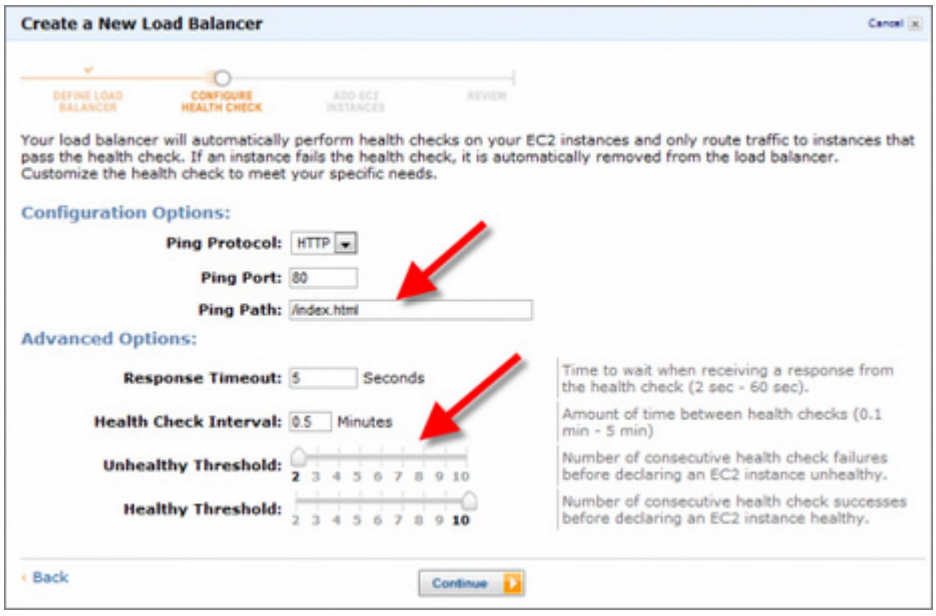
You will find the detailed information on AWS Load Balancer in the online documentation at *amazonwebservices.com*: Elastic Load Balancing.

Now, let us add the balancer:

The balancer listens on the port 80 and reroutes requests to the port 80 of the web cluster nodes:

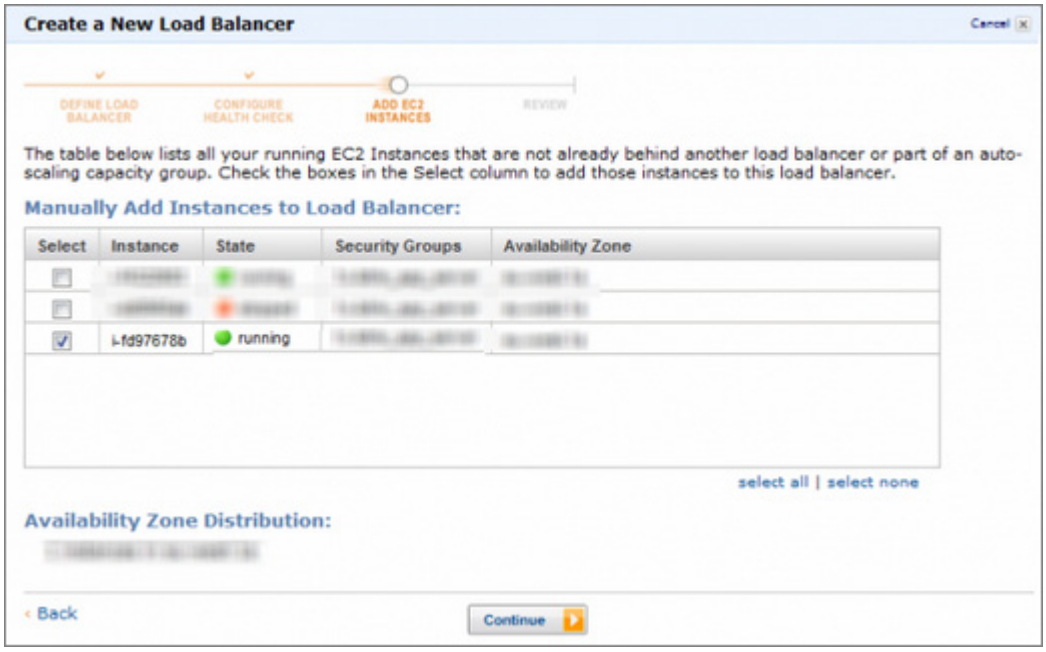


Configure the node's timeouts:



The page you specify in the field **Ping Path** should not be tracked by the **Web Analytics** module. Set other parameters according to the prospected load.

Select the nodes to be controlled by the balancer. The algorithm the balancer is based on (*"round robin"*) requires that you select nodes of similar performance. If the nodes are located in different data centers (*availability zones*), it is recommended to have the same number of nodes in each zone.



If your requirement is that all the requests of a particular client are processed by the same node, you need to enable such binding using the option **Enable Load Balancer Generated Cookie Stickiness**.

Now all the cluster nodes are available at a single domain name (e.g. *www.mydomain.com*) irrespective of their number. Whenever a node goes offline, the balancer just discontinues forwarding the client requests to such a node.

You will find more information about the balancer in the AWS documentation.

DNS

The simplest load balancing method is the use of round robin DNS.

The DNS specification allows to specify multiple different IN A records for the same name:

```
www   IN A   10.0.0.1
www   IN A   10.0.0.2
```

Here, the DNS server will return the whole list of IP addresses instead of a single one:

```
# host www.domain.local
www.domain.local has address 10.0.0.1
www.domain.local has address 10.0.0.2
```

Moreover, the DNS server will rotate the records and return different list for each request (though consisting of the same records). Therefore, the client requests become distributed among different IP addresses (or servers, in effect).

“round robin” is the simplest method possible, but being such it brings about serious drawbacks. We do not recommend using it.

The following is the summary of round robin DNS disadvantages.

- There is absolutely no criteria by which a certain IP address is selected from the supplied list on the client side. Depending on the implementation, a client may pick the first entry, or cache the most recently used value, or do any other selection.
- There is no means to determine the node availability or define the node weight. If one or more of the cluster nodes go offline, the requests will continue to be sent to the offline servers.
- The DNS cache lifetime is unacceptably long which will cause delay in service until all of the clients refresh their DNS cache after the breakdown of a node.

nginx

An alternative solution is to use the nginx HTTP server. The load balancing capability is provided by the ngx_http_upstream module.

Bitrix Web Environment comes equipped with nginx, so it is already installed at the web cluster servers (if you have chosen to resort for Bitrix Web Environment, of course) – you can use one of the servers for balancing purpose. However, it is recommended to set up a dedicated server for more flexible and undisturbed configuration and operation.

A simplest nginx configuration file (*/etc/nginx/nginx.conf*) providing balancing capability:

```
http {
    upstream backend {
        server node1.demo-cluster.ru;
        server node2.demo-cluster.ru;
    }
    server {
        listen   80;
        server_name  load_balancer;
        location / {
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header Host $host:80;

            proxy_pass  http://backend;
        }
    }
}
```

The section *"upstream"* specifies the addresses of the servers to which the balancing services will be provided.

The DNS table maps the website domain name to the IP address of the server running nginx.

Not specifying any other parameters essentially result in nginx using the round robin algorithm. However, **ngx_http_upstream** can be used for much more sophisticated scenarios.

1. In some cases, the desired behavior is to process all the requests of a certain client at the same server. The directive *ip_hash* causes requests to be distributed between upstreams based on the IP-address of the client.

```
upstream backend {
    ip_hash;
    server node1.demo-cluster.ru;
    server node2.demo-cluster.ru;
}
```

2. For clusters with servers of different capacity and/or performance, the servers can be assigned different weights. If not specified, the weight is equal to one. Use higher weight values for more powerful servers in a cluster.

```
upstream backend {
    server node1.demo-cluster.ru weight=3;
    server node2.demo-cluster.ru;
    server node3.demo-cluster.ru;
}
```

In this example, of every 5 requests they will be distributed like this: 3 requests on node1.demo-cluster.ru, and one request to the second and the third of servers.

It is not possible to combine *ip_hash* and *weight* methods for connection distribution.

3. The following parameters can be used to set thresholds for determining the offline servers.

- *max_fails* = the number of unsuccessful communication attempts within the time period (assigned by *fail_timeout*) after which it is considered inoperative;
- *fail_timeout* = the time during which there must occur *max_fails* number of unsuccessful communication attempts that would cause the server to be considered inoperative, and also the time for which the server will be considered inoperative (before another attempt is made). If not set, the time is 10 seconds.

Example:

```
upstream backend {
    server node1.demo-cluster.ru max_fails=3 fail_timeout=30s;
    server node2.demo-cluster.ru max_fails=3 fail_timeout=30s;
}
```

If the request to the server times out, the request will be transmitted to the following server and so forth. In this example: if 3 errors occurs within 30 seconds, such server is marked as offline for the next 30 seconds and will not be queried for this period of time.

Adding A Web Cluster Node

Assume that due to increasing traffic you need to add a new node to the web cluster. In practice, it means that you have to set up a new physical or virtual server and specify its parameters in the web cluster settings.

As you remember, we are creating our demo cluster on an Amazone cloud hosting which is why the following sequence describes actions required for AWS. However, it can be easily adjusted to any other environment.

1. Create a disk snapshot at one of the nodes.
2. Create a new disk from the snapshot. This disk will be used for the new node.
3. Set up an AMI virtual machine similar to the original node (1).
4. Select the hardware configuration depending on the expected load, or pick the same configuration as with the original node.
5. Stop the new node.
6. Disconnect the disk from the new node.
7. Connect the disk created at step 2.
8. Run the new node.
9. Assign the IP address to the new node if required.
10. Now, configure the new node. If using csync2, add the new node's domain name to csync2 parameters at all the other cluster nodes. Run the essential services at the new node: memcached, mysql-slave.
11. It is recommended to add the IP address assignments to the initialization scripts of the new node. Otherwise, you will have to do it manually each time you restart the node.
12. Synchronize the new node's data with that of the other nodes. Add the new node to the load balancer.
13. If the new node is used as a mysql-slave memcached server, register the services in the web cluster's Control Panel at *Settings* > *Web Cluster*.

Security

Since the web cluster is using additional services (centralized cache and synchronization), you need to pay certain attention to security issues.

Load Balancing And DDOS Attack Prevention

It is recommended to open the port 80 of the load balancer while closing external access to the HTTP ports at the web cluster servers. This will protect the nodes behind the balancer from being overloaded (which may occur, for example, during an extensive ad campaign), and substantially decrease the effectiveness of DDOS attacks.

Cluster Cache

Close public access to the memcached servers (port 11211), while making them available to the web cluster nodes. You may consider configuring the firewall as one of the simplest solutions.

Cluster Content Synchronization

If you are using csync2 for synchronization, close public access to it (port 30865), but make it available to the web cluster nodes.

Sample Security Configuration

For web cluster nodes:

Port	Status
22 (TCP; SSH)	Open for the administrator subnet
80 (TCP; HTTP)	Open for the web cluster subnet
443 (TCP; HTTPS)	Open for the web cluster subnet
3306 (TCP; MySQL)	Open for the web cluster subnet
11211 (TCP; memcached)	Open for the web cluster subnet
30865 (TCP; csync2)	Open for the web cluster subnet

For the load balancer:

Port	Status
80 (TCP; HTTP)	Open to all
443 (TCP; HTTPS)	Open to all

Chapter 3.
Web Cluster Stress Testing.
Analysis And Conclusions

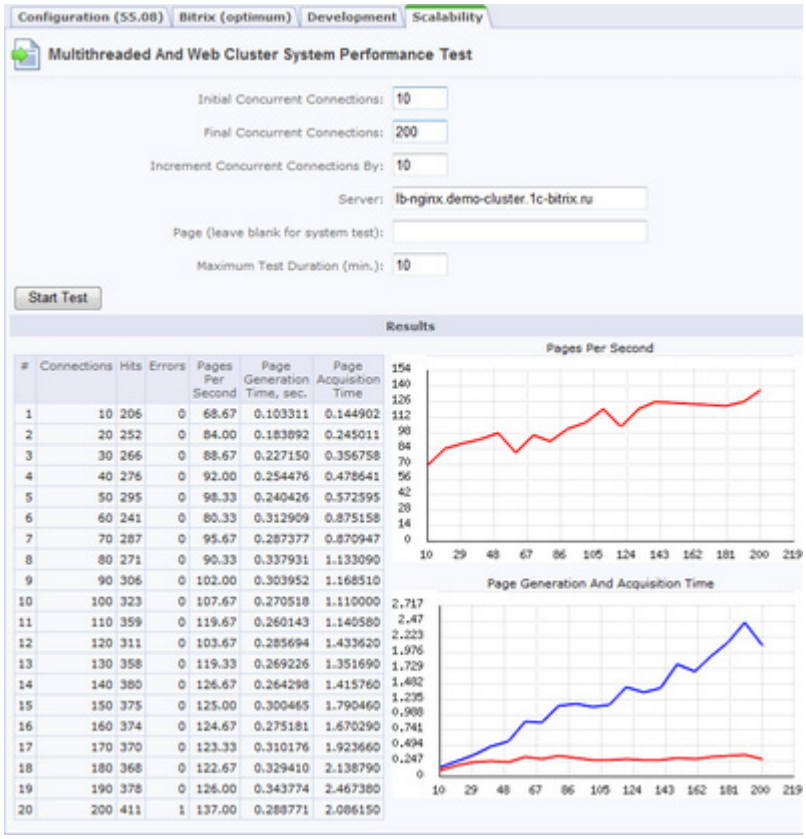
There are a good deal of tools currently available for the stress testing of web systems, ranging from simple programs (ab of Apache, siege, httperf) to powerful and sophisticated applications that allow to program any scenario and display exhaustive diagnostic information (JMeter, tsung, WAPT).

Since version 10.0, the Bitrix solutions have a built-in stress test tool.

> **Notice:** to minimize the impact the testing script may have on the test result, it is recommended to run it at a separate host.

The figures and text below illustrates the use of this tool.

Emulation Of Increasing Load



#	Connections	Hits	Errors	Pages Per Second	Page Generation Time, sec.	Page Acquisition Time
1	10	206	0	68.67	0.103311	0.144902
2	20	252	0	84.00	0.183892	0.245011
3	30	266	0	88.67	0.227150	0.356758
4	40	276	0	92.00	0.254476	0.478641
5	50	295	0	98.33	0.240426	0.572595
6	60	241	0	80.33	0.312909	0.875158
7	70	287	0	95.67	0.287377	0.870947
8	80	271	0	90.33	0.337931	1.133090
9	90	306	0	102.00	0.303952	1.168510
10	100	323	0	107.67	0.270518	1.110000
11	110	359	0	119.67	0.260143	1.140580
12	120	311	0	103.67	0.285694	1.433620
13	130	358	0	119.33	0.269226	1.351690
14	140	380	0	126.67	0.264298	1.415760
15	150	375	0	125.00	0.300465	1.790460
16	160	374	0	124.67	0.275181	1.670290
17	170	370	0	123.33	0.310176	1.923660
18	180	368	0	122.67	0.329410	2.138790
19	190	378	0	126.00	0.343774	2.467380
20	200	411	1	137.00	0.288771	2.086150

The tab 4 of the performance monitor, **Scalability**.

The input parameters:

- *Server* – in this case, the stress load is being applied to the balancer (nginx) which distributes the load among the two web servers (a web server, MySQL and memcached are running at each server and are added to the web cluster);
- *Page* – the test URL;
- *Initial Concurrent Connections, Final Concurrent Connections, Increment Concurrent Connections By* define the stress load parameters.

In this test, we are starting with 10 concurrent connections and increase their number to 200 at an interval of 10.

The second chart shows:

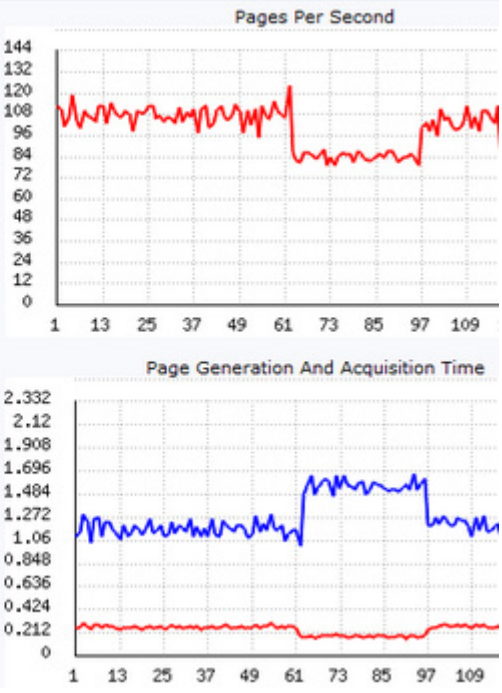
- the whole system is well balanced; the load increase does not degrade the system performance because the page generation time remains almost flat (the red line);
- the page transmission time increases due to queue growth (the blue line).

Emulating MySQL Slave Server Breakdown

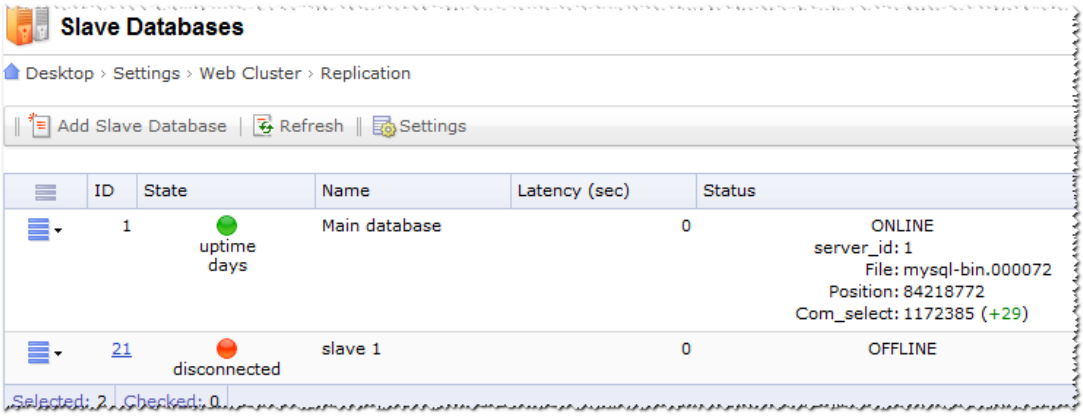
This test imitates the breakdown of one of the web cluster machines (the one running the MySQL slave database).

Results

#	Connections	Hits	Errors	Pages Per Second	Page Generation Time, sec.	Page Acquisition Time
1	100	339	0	113.00	0.254384	1.091420
2	100	333	0	111.00	0.259116	1.124530
3	100	303	0	101.00	0.294063	1.296410
4	100	319	0	106.33	0.272213	1.233670
5	100	356	0	118.67	0.238060	1.042020
6	100	312	0	104.00	0.287923	1.253230
7	100	302	0	100.67	0.283896	1.266900
8	100	331	0	110.33	0.251741	1.103020
9	100	320	0	106.67	0.278816	1.221660
10	100	319	0	106.33	0.275613	1.222020
11	100	313	0	104.33	0.268375	1.161730
12	100	339	0	113.00	0.260976	1.129540
13	100	337	0	112.33	0.247677	1.076180
14	100	308	0	102.67	0.262798	1.192910
15	100	343	0	114.33	0.256211	1.104170
16	100	330	0	110.00	0.250797	1.120340
17	100	320	0	106.67	0.275137	1.198950
18	100	321	0	107.00	0.262410	1.156470
19	100	330	0	110.00	0.242258	1.113700



Let us look at the slave server status: all services are suspended:



The small peaks indicate the time when the data is synchronized.

The surge on the blue graph is just the moment when the slave server was offline.

Phone/Fax: +1 (703) 740-8301; e-mail: info@bitrixsoft.com, http://www.bitrixsoft.com

Chapter 4.
Web Cluster Configuration: Use Cases

Before you start considering your options, you first have to do a thorough analysis of the type and the amount of load your web project is experiencing, and make a reasonable forecast for the observable future based on the current load and your plans. For this purpose, you can use any monitoring application like munin, zabbix, Apache's server-status etc.

Armed with this knowledge, you can make a sketch of the web cluster configuration and capacity required for your project. Some of the possible scenarios are outlined below.

High CPU load; moderate database load; the content is almost persistent

Reduce CPU load by adding more nodes to the cluster. Put the nodes behind the load balancer which in essence distributes CPU load across the nodes.

Setting up a memcached server at each node will decrease CPU load even further because the cache will be created at only one of the nodes for use by the others.

Use a csync2 or nfs/cifs server at one of the nodes for synchronization.

Database load is high and growing

Set up for master/slave replication. The more slave nodes is in the cluster, the less is the master database load. If your slave nodes are of different capacity or performance, use the "**Load, %**" parameter to let them operate according to their power.

If the database load is still growing:

1) Optimize master and slave nodes. Since the master is primarily for writes, it should be optimized for write operations, while slaves should be configured in favor of better read speed.

2) Add more slave nodes.

Cache is huge and rebuilds frequently; the content is highly volatile

Use cache more effectively. Run the memcached server at each node and allocate some GB of RAM to the former.

Use a dedicated nfs/cifs or ocfs/gfs server for synchronization.