Good afternoon!

My name is Valery Bashkatov.

I am working for Progress Technologies company, which is the official distributor of Progress Software in Russia, the CIS countries and Latvia.

I work as an OpenEdge Principal Consultant, what means that in the zone of my responsibility are everything what related to OpenEdge platform - starting with technical support and ending with training of new Progress database administrators.

Today I'll tell you about how to implement load balancing for the new Progress Application Server for OpenEdge.

I don't pretend to describe the completeness of this topic, but only want to provide a starting point and to demonstrate the simple configuration of Apache Tomcat Load Balancing to spread the load between PAS for OpenEdge instances.
Why a load balancing?

- High Availability
- High Reliability
- High Scalability

For a typical application that is serviced by a small number of users and that does not have serious business requirements, a load balancing may not be needed.

But the situation is completely different with those applications that are key to business. For example, applications that handle billing operations.

If a service of this kind "falls", then some business process also will cease to function.

In this case, we have a single point of failure.

To eliminate it we should configure load balancer cluster for our application server.

As result, if one of the application servers fails or if we need to perform maintenance, the load balancer will forward all user requests to the second application server.

So, load balancing is a solution to the problem of a single point of failure of application server.

Then, as we know, application servers may use a lot of memory, especially when a large number of users work with it.

The more users, the more server resources are needed to process their requests.

So, load balancing it is also scalability and optimization of resource use.

-------------------
https://esj.com/Articles/2012/06/18/5-Reasons-To-Use-Load-Balancers.aspx?Page=1
https://blog.wsol.com/what-advantages-does-load-balancing-provide-for-web-servers
<table>
<thead>
<tr>
<th>Classic OpenEdge AppServer</th>
<th>New PAS for OpenEdge</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Progress Name Server Load Balancer</td>
<td>- Apache HTTP</td>
</tr>
<tr>
<td>Name Server, Admin Server</td>
<td>- Apache Tomcat</td>
</tr>
<tr>
<td></td>
<td>- Amazon Elastic Load Balancing</td>
</tr>
</tbody>
</table>

For more complete information on configuring and working with Apache Tomcat Load Balancing refer to Internet resources, for example, https://httpd.apache.org/ and http://tomcat.apache.org/.

In the classic OpenEdge AppServer to implement load balancing used Progress Name Server Load Balancer.

But load balancing is quite different in PAS for OpenEdge - there is no NameServer or ubroker.properties file to implement load balancing.

Instead, for the new Progress Application Server load balancing is implemented by using standard HTTP options based on one of the third-party technologies such as Apache HTTP, Apache Tomcat or Amazon Elastic Load Balancing.

There are many more third-party load balancing products available. Most of them are Open Source projects and are available for free.

But it’s not possible to recommend the best load balancing option for your application – it’s depending on your requirements and it’s only for your choice.

I chose Tomcat Load Balancing for this presentation because some time ago I had a support case from one of our clients, where they asked to help in the implementation of load balancing for a new application server.

So, we had to look for information and to explore this issue. And as a result, we were able to configure the working version of the load balancing cluster and I decided that this topic will be interesting for you too.

Let’s start.
On this slide shows the configuration of the simple load balancer cluster, which we implement in this session.

The first server is an external Apache Tomcat server. There is nothing except for a web server.

But I will not describe how this server is configured, since this is a topic for a separate presentation, and I do not use it in my demo.

All you need here, is to remember that in a real production environment external web server - it's a public server which may be available from the Internet and it must be placed in the demilitarized zone (DMZ) in terms of safety.

-----------------------------------------

CAN AN APACHE VULNERABILITY AFFECT OEPAS?

If the application is using Tomcat directly an Apache web server vulnerability will not affect OEPas.

The Apache web server is a native code HTTP server, while Tomcat is a Java based J2EE server. Vulnerabilities reported against Apache are not the same as vulnerabilities reported against Tomcat.

The only exception to this is if Apache is being used in the front and mod_proxy is used to redirect requests to Tomcat.

https://knowledgebase.progress.com/articles/Article/can-an-apache-vulnerability-affect-oepas
When users connect to the external Apache server, we redirect them to the local Apache server.

This server acting as a load balancer and placed on separate machine in the local network.

In this part, we will perform its installation and configure local Apache for load balancing.
Apache installation is very simple. So I'll just show you the common steps on this slide.

The first we should install Apache using YUM, since I use CentOS 7. Then we allow access to Apache through the firewall, to open http or https ports depending on your requirements.

Next configure Apache to start at boot. And check how this work by start Apache and verifying status.

As you can see at this step here is nothing too hard.

Next, we should install mod_jk module.

------

hostnamectl set-hostname apache
systemctl restart systemd-hostnamed
Installation of MOD_JK

- Make sure that we have installed all the necessary modules
  
  ```
  yum install httpd-devel apr apr-devel apr-util apr-util-devel
  gcc gcc-c++ make autoconf libtool
  ```

- Download the Mod_jk
  
  ```
  mkdir -p /opt/mod_jk/
  cd /opt/mod_jk
  wget http://www.eu.apache.org/dist/tomcat/tomcat-connectors/jk/tomcat-connectors-1.2.42-src.tar.gz
  tar -xvzf tomcat-connectors-1.2.42-src.tar.gz
  cd tomcat-connectors-1.2.42-src/native
  ```

- Compile and install the module in native/ directory:
  
  ```
  ./configure --with-apxs=/usr/bin/apxs --enable-api-compatibility
  make
  libtool --finish /usr/lib64/httpd/modules
  make install
  ls -la /etc/httpd/modules/mod_jk.so
  ```

Mod_jk is the Apache module that will be used to provide our cluster with its load balancing and proxy capabilities.

It uses the Apache JServ Protocol (AJP) to facilitate fast communication between Tomcat servers and the Apache Web Server.

In other words, mod_jk allows us to establish a connection between Apache and a back-end application server like PAS for OpenEdge.

The mod_jk module is distributed separately from Apache as part of the Tomcat project.

Mod_jk installation process is simple, but it requires compilation.

So first, make sure that we have installed all the necessary modules for mod_jk.

Then download the mod_jk from the official web-site (at the time of this writing, the current version number of the module was 1.2.42).

Next compile and install the module in “native/” directory.

If everything goes with no errors, then in the directory “/etc/httpd/modules” should be appear mod_jk.so library.

------

httpd-devel (important)
Configuring Apache for load balancing

- httpd.conf:

  #workers.properties load balancing config
  LoadModule jk_module apache_install_dir/modules/mod_jk.so
  JkWorkersFile apache_install_dir/conf/workers.properties
  JkShmFile apache_install_dir/logs/mod_jk.shm
  JkLogFile apache_install_dir/logs/mod_jk.log
  JkLogLevel [debug/error/info]
  JkMount /status status
  JkMount /* lb

Next, we'll have to set up the mod_jk module in the Apache configuration file. This configuration is a two-step process

**First**, let's configure the module itself.

This is done by adding a few lines to the configuration file. The following important parameters are presented here:

- **LoadModule** - this makes the mod_jk module available for use.
- **JkWorkersFile** - the path to the worker configuration file, which we will create in the next step.
- **JkShmFile** - the path to the shared memory file for the module.
- **JkLogFile** - the path to the module log file.
- **JkLogLevel** - the level of logging for the module. The valid values for this attribute are "debug", "error" or "info".
- **JkMount** - this is used to map a certain URL pattern to a specific worker configured in the workers configuration file. Here, we use it twice - once to enable "/status" as the access URL for a virtual monitoring worker, and once to map all requests we want to be handled by the cluster to the virtual worker that contains the load balancing capability (lb).

The second step I am will describe after we creating worker instances.
The following two servers contain working instances of application server, to which the load balancer will forward user requests for processing.

As worker instances, among which will be implemented load balancing, we will create two physical worker instances of the PAS for OpenEdge, each of which will run in its own server.
Creating and configuring the worker instances

Node 1

▪ proenv>

$DLC/servers/pasoe/bin/tcman.sh create -p 8820 -P 8821 -s 8822 $WRKDIR/node1

- p 8820 – port number for http;
- P 8821 – port number for https;
- s 8822 – port number for shutdown;

$WRKDIR/node1 – the path and the name of the instance directory.

To create PAS instance we use TCMAN utility with CREATE option. We should identify HTTP and HTTPS port number and shutdown port number. Worker instance of PAS for OpenEdge can be in any directory on the server. For our convenience, we will create them in the standard OpenEdge work directory (WRKDIR)

--------

Random Number Generator

Adjust “write_wakeup_threshold” if creating PAS slow to create new instance for virtual machine since

rngd -r /dev/urandom

https://www.certdepot.net/rhel7-get-started-random-number-generator/
Creating and configuring the worker instances

Node 1

- **Activate AJP13**
  - `cd $WRKDIR/node1/bin`
  - `tcman.sh feature AJP13=on`
- Set the AJP13 port number
  - `tcman.sh config psc.as.ajp13.port=8009` or `tcman.sh create -j <ajp13-port> ...
- Open AJP13 port in the firewall
  - `firewall-cmd --permanent --add-port=8009/tcp`
  - `firewall-cmd --reload`

Apache JServ Protocol (AJP) – a binary protocol that enables transfers incoming requests from the web server to the application server.

After creating a PAS instance we need to activate AJP13 protocol.

**AJP13** – *it is a binary protocol that enables transfers incoming requests from the web server to the application server. It is generally used in a load-balanced systems. It also supports the monitoring of state of the server.*

By default in PAS for OpenEdge this protocol is disabled, so we must enable it using TCMAN utility to set feature AJP13 to ON.

Next, set the AJP13 port number.

By default, this number is 8009. But in case when in one server work multiple instances of PAS for OpenEdge it is recommended for each instance establish a unique AJP13 port number.

In conclusion we open AJP13 port in the firewall because through this port the working instance will interact with users via the load balancer.

Reload the firewall.

-------------

Hot to change ports in an existing instance?

- `tcman config psc.as.http.port=
- `tcman config psc.as.https.port=
- `tcman config psc.as.ajp13.port=
- `tcman config psc.as.shut.port=`
Creating and configuring the worker instances

Node 1

- Activate APSV / REST / WEB / SOAP
  
  - ./conf/openedge.properties
    
    (node1.ROOT.APSV)
    
    adapterEnabled=1
    enableRequestChunking=1
    useHTTPSessions=1
  
  - OEPROP.sh

    oeprop.sh node1.ROOT.APSV.adapterEnabled=1

- Start PAS instance node1

  tcman.sh start

In the PAS for OpenEdge, the Client access to applications through special protocols: APSV, REST, WEB and SOAP.

By default, in the production version of the PAS for OpenEdge for security reasons all communication protocols disabled.

In the development version these protocols are enabled by default.

In our example to demonstrate the work of load balancing, we will use APSV protocol designed to interact with the application server through the ABL.

So it is necessary to activate this protocol.

To do it we used “adapterEnable” property with a value of “1” in the instance configuration file openedge.properties.

We can edit the configuration file manually or by using a script “oeprop”.

After that we can start the instance.
Creating and configuring the worker instances

Node 2

- $DLC/servers/pasoe/bin/tcman.sh create -p 8820 -P 8821 -s 8822 $WRKDIR/node2
- cd $WRKDIR/node2/bin
- tcman.sh feature AJP13=on
- tcman.sh config psc.as.ajp13.port
- firewall-cmd --permanent --add-port=8009/tcp
- firewall-cmd --reload
- oeprop.sh node2.ROOT.APSV.adapterEnabled=1
- tcman.sh start

To create a worker instance node2, we need to perform the same actions as when creating the node1 instance.

Because node2 we placed on a separate server, the ports for that instance we leaving the same as for node1.

Differ is only in instance name.
Mod_jk requires one "workers.properties" file where load balancing is configured. Workers are defined in the properties file and represent actual or virtual workers. In our example this file defines the virtual load-balancing instance, the virtual metrics-gathering instance, and the two actual worker instances that execute client requests. The last two is a PAS for OpenEdge instances.

To create and deploy the worker.properties file:

1. Create a preliminary worker.properties file by running the tcman workers command in the bin directory of each PAS instance.
2. Combine the worker.properties files that you have generated for all instances into a single worker.properties file and place it in the conf directory of the Apache HTTP server.
3. Comment out the property, worker.common.host, in the combined worker.properties file.
4. Add a workers.instance_name.host=host_name property to each instance's configuration.
5. Save and close the file.
Creating and deploying the worker.properties file

- Follow these steps for generate preliminary worker.properties file:
  - node1
    cd node1/bin
    tcman.sh workers
    ls -la /usr/wrk/node1/temp
  - node2
    cd node2/bin
    tcman.sh workers
    ls -la /usr/wrk/node1/temp

You can create this workers file manually or use TCMAN script with option WORKERS.

Follow these steps for use TCMAN:
1. For instance node1: 
   a. Connect to a virtual machine node1.
   b. Open proenv.
   c. Change directory to bin:
      cd node1/bin
   d. Execute command to generate workers.properties file:
      tcman.sh workers

workers.properties file will be created in the /usr/wrk/node1/temp directory.

For the second instance, we will perform the same steps.
Example of worker.properties files

NODE 1

worker.list=node1
worker.common.type=ajp13
worker.common.host=node1
worker.common.socket_timeout=10
worker.common.connect_timeout=10000
worker.common.socket_keepalive=true
worker.common.ping_mode=I
worker.common.ping_timeout=10000
worker.common.retry_interval=100
worker.common.recovery_options=7
worker.node1.port=8009
worker.node1.reference=worker.common

NODE 2

worker.list=node2
worker.common.type=ajp13
worker.common.host=node2
worker.common.socket_timeout=10
worker.common.connect_timeout=10000
worker.common.socket_keepalive=true
worker.common.ping_mode=I
worker.common.ping_timeout=10000
worker.common.retry_interval=100
worker.common.recovery_options=7
worker.node2.port=8009
worker.node2.reference=worker.common

This is an example of the two the preliminary “worker.properties” files for Node 1 and Node 2 instances, which we created using the TCMAN.
Then we should merge a preliminary “workers.properties” files by copy/paste and at the same time remove the duplicate rows.

We should get a file with the following contents as shown on the slide.

Now correct the merged file.
Our configuration defines two virtual workers, and two actual workers, which map to my Tomcat servers.
Let's study this file in parts.

The first is common section.
Here I made only one change.
The virtual workers “status” and “lb” I defined in the **worker.list** property, because I’m refer to them in my apache configuration.
Edit the merged worker.properties file

```properties
worker.node1.port=8009
worker.node1.host=172.16.95.160
worker.node1.reference=worker.common
worker.node1.lbfactor=1

worker.node2.port=8009
worker.node2.host=172.16.95.161
worker.node2.reference=worker.common
worker.node2.lbfactor=1
```

The next section is a section for our actual worker instances.
Here, I define workers for each of my servers, using the port values from the AJP connectors and hosts of that servers.
I’ve also included optional property for these workers, “lbfactor”.
The higher the number of this property, the more preference mod_jk will give that worker when load balancing.
For example, if I had given the servers lbfactors of 1 and 3, I would find that the round-robin load balancing would prefer one server over the other with a 3 to 1 ratio.
Correct the merged worker.properties file

```properties
worker.lb.type=lb
worker.lb.balance_workers=node1,node2
worker.lb.sticky_session=True
worker.lb.sticky_session_force=True
worker.lb.method=Request

worker.status.type=status
```

Lastly, I’ve got a little configuration for my virtual workers. I’ve set the load balancer worker to have type “lb” and listed the workers which represent Tomcat in the “balance_workers” property.

If I had any further servers to add, I would define them as a worker and list them in the same property.

Load balancers use a variety of methods to make sure that requests are sent to the machine that has the most current session data.

The easiest of these, and the one we will use for this example, is called "sticky sessions". **Sticky sessions** are an important feature if you rely on jSessionIDs and are not using any session-replication layer.

If sticky_session is True a request always gets routed back to the node which assigned this jSessionID.

If that host should get disconnected, the request will be forwarded to another host in our cluster, although not too successfully as the session-id is invalid in it’s context. You can prevent this by setting **sticky_session_force** to True. In this case if the host handling the requested session-id fails, Apache will generate an internal server error 500.

This is especially important for an application server that interacts with an OpenEdge database.

The latest version of mod_jk enables sticky sessions by default.

If **method** is set to Request the balancer will use the number of requests to find the best worker. Accesses will be distributed according to the lbfactor in a sliding time window.

This is the default value and should be working well for most applications.
The only configuration that the status worker needs is to set the type to status.
Here is an example of the final workers.properties file.
Final workers.properties file

1. Save worker.properties into /etc/httpd/conf on the Apache server

2. Start of Apache HTTP Web Server:
   systemctl restart httpd

Save the changes and put the final workers.properties file into etc/httpd/conf/ directory on the Apache server.
And perform restart of Apache HTTP Web Server.
Now it's time to show how it all works in practice.
Thank you for your attention and welcome to the "Progress - OpenEdge Zone" - the largest group of Progress friends in social networks!
Additional materials
Verifying Load Balancing: testserver.p

DEFINE OUTPUT PARAMETER vCustName AS CHARACTER NO-UNDO.

FOR FIRST Customer NO-LOCK.
    vCustName = Customer.Name + ' on PAS '
    + SUBSTRING(SESSION:SERVER-CONNECTION-ID,INDEX(SESSION:SERVER-CONNECTION-ID,"." + 1).
END.

Put procedure testserver.p into "openedge" directory for each PAS instance.
Run procedure testclient.p

mpro -p testclient.p
Verifying Load Balancing: result

Procedure Editor - Run

Press

- UI Tours on PAS node2
- UI Tours on PAS node1
- UI Tours on PAS node2
- UI Tours on PAS node1
- UI Tours on PAS node2
- UI Tours on PAS node1
- UI Tours on PAS node2
- UI Tours on PAS node1

Procedure complete. Press space bar to continue.
Monitoring load balancing

- JK Status Manager
  - http://<lb-hostname>/status

To monitor the load balancer, we have several options.

The first of them is JK Status Manager, which available for us through web browser.
That is an example of monitoring web-page.

With JkStatusManager the tomcat cluster worker can temporarily disabled for maintenance reasons e.g. software installations, updates or application reconfiguration.

To disable a tomcat instance in a cluster set the worker status to 'disabled'. Before doing some maintenance tasks be sure that there are no active sessions remains on this tomcat worker.

The value 'disabled' means that no new further sessions will be created by the load balancer on this tomcat worker. If all sessions of the worker are finished or timed out the worker is cluster released and can be configured.

On this page, in addition to viewing statistics on the load balancer, we can manage our worker instances, for example,

Click 1

activate or deactivate an instance for maintenance,

Click 2

or change some of their properties.

-----------------------
http://www.gesea.de/techdocs.htm?id=67616
Other options it is log files of Apache, MOD_JK and PAS for OpenEdge instances.

Thus, if your balancer does not work, you should look at the log files and JK Status Manager.
How to add new PAS instance by copy existing

- Stop existing instance, for example node2
  - Disable and stop worker instance in the JK Status Manager
  - Stop PAS instance by "tcman.sh stop"
  - Clean logs by "tcman.sh clean"
- Copy instance by OS command
  - cp -R $WRKDIR/node2 $WRKDIR/node3
- Start existing instance (node2) and enable them for balancing in the JK Status Monitor
How to add new PAS instance by copy existing

- Register new instance
  - `$DLC/servers/paseo/bin/tcman.sh register node3 $WRKDIR/node3`

- Change ports for the new instance (node3)
  - `tcman config psc.as.http.port=<new http port>`
  - `tcman config psc.as.https.port=<new https port>`
  - `tcman config psc.as.ajp13.port=<new ajp13 port>`
  - `tcman config psc.as.shut.port=<new shutdown port>`

- Allows AJP13 port of new instance in the SELinux
  - `semanage port -a -t http_port_t -p tcp <AJP13 port>`
How to add new PAS instance by copy existing

- Add new worker instance into worker.properties
  - cd /etc/httpd/conf
  - vim worker.properties
  - Add
    worker.node3.port=<{AJP13 port}>
    worker.node3.host=<{worker host or IP}>
    worker.node3.reference=worker.common
    worker.node3.lbfactor=1
  - Add
    worker.lb.balance_workers=node1,node2,node3

- Restart Apache server
  - systemctl restart httpd
Because of SELinux policy, a service is normally allowed to run on a restricted list of well-known ports. For example, in the case of the httpd service, this list is 80, 443, 488, 8008, 8009, 8443.

To allow a service to use non-standard ports, you need to follow a specific procedure to change the SELinux policy.

This it important, if we want to add more than one instance of Tomcat (PASOE) to the same server with separate AJP13 ports to configure load balancing.

Install the setroubleshoot-server (to get the semanage command) if not installed yet:

```bash
yum install -y setroubleshoot-server
```

To get the list of all restricted ports by service, type:

```bash
semanage port -l
```

To get the list of well-known ports for the httpd service, type:

```bash
semanage port -l | grep -w http_port_t
```

To allow the httpd service to run on the 40100 tcp port (-a for add), type:

```bash
semanage port -a -t http_port_t -p tcp 40100
```
Disabling SeLinux if needed (for demo only)

```bash
sestatus
```

setenforce 0 - will only disable SELinux for now

Disable it permanently by opening the file `/etc/sysconfig/selinux` and replace the current SELINUX directive with a value of disabled so it looks like this:

```
SELINUX=disabled
```