Progress®



Progress Application Server for OpenEdge:
Machine Sizing Guide

OpenEdge®

Table of Contents

Copyright.....5

Overview.....7

Machine Sizing Test Information.....8
Performance Tuning Basics.....9
Configuring Your Testing and Monitoring Infrastructure.....10
PAS for OpenEdge Tuning Parameters.....11
Performance Testing and Machine Sizing Process.....13

Appendix A: Gather System Information.....15

Monitoring with Top Command.....15
Sample Scripts for Gathering System Information.....16

Copyright

© 2021 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

These materials and all Progress® software products are copyrighted and all rights are reserved by Progress Software Corporation. The information in these materials is subject to change without notice, and Progress Software Corporation assumes no responsibility for any errors that may appear therein. The references in these materials to specific platforms supported are subject to change.

DataDirect Cloud, DataDirect Connect, DataDirect Connect64, DataDirect XML Converters, DataDirect XQuery, DataRPM, Defrag This, Deliver More Than Expected, DevReach (and design), Icenium, Inspec, Ipswitch, iMacros, Kendo UI, Kinvey, MessageWay, MOVEit, NativeChat, NativeScript, OpenEdge, Powered by Chef, Powered by Progress, Progress, Progress Software Developers Network, SequeLink, Sitefinity (and Design), Sitefinity, Sitefinity (and design), SpeedScript, Stylus Studio, Stylized Design (Arrow/3D Box logo), Styleized Design (C Chef logo), Stylized Design of Samurai, TeamPulse, Telerik, Telerik (and design), Test Studio, WebSpeed, WhatsConfigured, WhatsConnected, WhatsUp, and WS_FTP are registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and/or other countries.

Analytics360, AppServer, BusinessEdge, Chef Automate, Chef Compliance, Chef Desktop, Chef Habitat, Chef WorkStation, Corticon.js, Corticon Rules, Data Access, DataDirect Autonomous REST Connector, DataDirect Spy, DevCraft, Fiddler, Fiddler Everywhere, FiddlerCap, FiddlerCore, FiddlerScript, Hybrid Data Pipeline, iMail, JustAssembly, JustDecompile, JustMock, KendoReact, NativeScript Sidekick, OpenAccess, PASOE, Pro2, ProDataSet, Progress Results, Progress Software, ProVision, PSE Pro, Push Jobs, SafeSpaceVR, Sitefinity Cloud, Sitefinity CMS, Sitefinity Digital Experience Cloud, Sitefinity Feather, Sitefinity Insight, Sitefinity Thunder, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, SmartDataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, Supermarket, SupportLink, Unite UX, and WebClient are trademarks or service marks of Progress Software Corporation and/or its subsidiaries or affiliates in the U.S. and other countries. Java is a registered trademark of Oracle and/or its affiliates. Any other marks contained herein may be trademarks of their respective owners.

July 2021

Product version: Progress OpenEdge 12.4

1

Overview

This document describes how to size your Progress Application Server (PAS) for OpenEdge machine for use in production. Sizing your machine means establishing performance benchmarks and running performance tests against those benchmarks. This document is intended for OpenEdge administrators who want to ensure they are getting the best performance from their PAS for OpenEdge application. However, if the OpenEdge administrator finds memory leaks or performance issues in the ABL code, they will need help from OpenEdge developers in fixing those errors.

To get the best performance out of your OpenEdge web application, size your PAS for OpenEdge machine to your application and network load. To gauge the most efficient sizing of your machine, run performance tests to see how many resources your application is using. We recommend starting the performance test with a low number of clients to find a baseline for your application. When you scale your application, you can see how many clients your machine can handle for your application and tune the performance as necessary. At Progress, we ran a PAS for OpenEdge Load Test on several machines running a sample application. This document describes our test benchmarks and results, and also provides the framework to effectively size your machine for your application.

For details, see the following topics:

- Machine Sizing Test Information
- Performance Tuning Basics
- Configuring Your Testing and Monitoring Infrastructure
- PAS for OpenEdge Tuning Parameters
- Performance Testing and Machine Sizing Process

Machine Sizing Test Information

This section describes the test and test environments we used on an sample application, which is a modified version of the "ATM Test". The ATM test has been around for over 15 years at Progress and is designed to mimic a real-world use-case of an OpenEdge application. More information about the ATM Test can be found on Progress Communities.

Application and test description:

- Application emulates an Automatic Teller Machine (ATM).
- Functionality includes updating account, bank, teller, and deposit tables to an OpenEdge database.
- The application is completely state-free using the WEB transport.
- The test used Amazon machines in the same region.
- The test ran the PAS for OpenEdge on an isolated system with the database and test driver on two other machines, configured as follows:
  - The test driver used JMeter from a Windows machine.
  - PAS for OpenEdge resided on different size Linux machines.
  - The database is on a separate Linux machine. The database was scaled large enough that it and the machine were never a bottleneck (memory, CPU, and record contention were not issues).

Test parameters and bottleneck indicators

After performing historical analysis of application, we found that the baseline test turn-around time was around 100ms (milliseconds). Testing showed if there was a bottleneck in CPU or memory, the elapsed turn-around time increased dramatically. Your application may have a longer turn-around time, so start running performance measurements with minimal load to get a baseline of best turn-around time and increase load until you see a marked increase in elapsed time.

Also, please note the following observation:

- In addition to Average Elapsed time, CPU seemed to be the other large indicator of performance degradation. When the load average started to equal the amount of CPUs, then elapsed turn-around time increased immediately. If load average equals the vCPU, then the machine is at 100% load.
- If you use 100% of RAM, the machine starts swapping RAM for disk space and degrading trip time. This was not seen in the testing.
- Total responses indicates the amount of requests served in the testing timeframe.

The following table shows our tests and responses:

Amazon Machine Linux	**c3.large**	**c3.xlarge**	**c3.2xlarge**	**c3.4xlarge**
vCPU	2	4	8	16
RAM	3.75 Gb	7.5 Gb	15 Gb	30 Gb
concurrent clients	50	125	250	500

openedge.properties	Set the `maxAgents=1` Set the `maxConnections PerAgent=50`	Set the `maxAgents=1` Set the `maxConnections PerAgent=125`	Set the `maxAgents=1` Set the `maxConnections PerAgent=250`	Set the `maxAgents=1` Set the `maxConnections PerAgent=500` Set `pcs.as.executor .maxthreads=500`
Load Average	1.42	2.95	9.11	16.5
Total Responses	14,174	35,650	71,371	139,948
Avg Elapsed Time	113 ms	103 ms	99 ms	95 ms

As you can see in the tests that we ran, we were able to compensate for the larger testing loads by editing the `openedge.properties` files and setting the maxConnectionsPerAgent property accordingly. When the clients continue scaling up, there are other properties that must be adjusted. These properties are discussed in the "PAS for OpenEdge Tuning Parameters" section of this document. For extended discussion on application tuning, see the PAS for OpenEdge Machine Tuning Guide, which goes into detail regarding best practices.

Note:

The difference between sizing and tuning an application is that in sizing the application you want to answer is, "How many clients running my application can I handle on a given machine?" This requires running tests. With application tuning, the question you are trying to answer is, "How can I get the best performance for my application at scale?" In-depth tuning questions and answers can be found in the PAS for OpenEdge Machine Tuning Guide.

Performance Tuning Basics

There are four possible bottlenecks in any application environment: CPU, Memory, Disk I/O, and Network. There can be several reasons why any of these bottlenecks may occur. To properly size or tune your application, you must check each component for bottlenecks and relieve them until you have located all areas with dragging throughput in your application environment.

The following list shows examples of symptoms and causes of bottlenecks in each component:

Table 1: CPU Bottlenecks

Symptoms	High Load – Load number equals or near-equals number of CPUs
	0% Idle time
Causes	Not enough CPUs
	Too much swapping (memory to disk)
	Application issue

Table 2: Memory Bottlenecks

Symptoms	0% Memory available
	Swap space being used
Causes	Memory Leaks
	Not enough memory for application

Table 3: Disk I/O Bottlenecks

Symptoms	Poor performance in the PAS for OpenEdge agent
Causes	Files or logs on network drives
	Too much swapping
	Slow disk I/O

Table 4: Network Bottlenecks

Symptoms	Poor performance from the PAS for OpenEdge agent to the client
Causes	DNS look up issues
	Poorly configured network
	Saturated network devices

Find the bottleneck in any of these four areas by looking at the symptoms and alleviate the cause by making one change at a time. Then run your test again to see how your change worked, then find the next bottleneck.

Configuring Your Testing and Monitoring Infrastructure

This section describes how to replicate our performance tests in your environment.

Testing Infrastructure

The testing infrastructure is for sizing your PAS for OpenEdge instance, so you will want to use three machines to run the tests to ensure that there is no performance cross-contamination. For example, if you ran your test harness on the same client as your PAS for OpenEdge instance, that could induce performance lag. We recommend running the performance test on three different machines:

1. Your client machine or test harness
2. Your PAS for OpenEdge instance and application
3. Your database

You can use this data later to scale your application if PAS for OpenEdge and the database are on the same machine. However, trying to size your PAS for OpenEdge application with the database running on the same machine would introduce too many variables.

Testing harness

You will need to select a test harness. You could choose from any number of testing tools, such as Apache JMeter, LoadRunner, or other commercial or open-source tool.

There are a few points to keep in mind:

1. Ensure that the test machine always has excess CPU, Memory, and Network capabilities.
2. Ensure that you add some “think time” in writing your tests. Real applications never have all requests running immediately after the response. We ran our tests with a 1 to 3 second random "think time" interval.
3. Ensure that your test harness gathers round trip (a.k.a elapsed) times.
4. Ensure that your test harness gathers and reports errors.

Monitoring tools

Monitoring tools are essential for running your sizing or performance tests. You will want to monitor all three machines. The monitoring tools must be reliable, consistent and run in the background. The data must be archived while being run.

In our sizing tests, we used the following monitoring tools:

- Top – A Linux system performance monitoring tool that provides CPU, RAM, Swap (Disk space/speed) usage on the PAS for OpenEdge machine.
- OE Manager REST API – A PAS for OpenEdge monitoring tool that provides information about session manager and AMS agent such as client requests, agents used, sessions per agent, and memory size of agent ABL session.
- JMeter – JMeter was not only the test harness, but also monitors responses for elapsed time and errors.

See the Appendix for all the data recorded during each test from the three monitoring tools.

PAS for OpenEdge Tuning Parameters

Although there are many parameters available for tuning PAS for OpenEdge, you only need to know a few for most tuning situations. You should edit these parameters during your sizing tests to reduce the average elapsed time of your requests.

TCMan Properties

You can change the Tomcat maximum simultaneous connections using the OpenEdge proenv command-line utility.

The following commands show how to set the Tomcat Maximum simultaneous connections to the recommended number of 300 for your baseline test:

```
proenv> tcman.sh config psc.as.executor.maxthreads

proenv> psc.as.executor.maxthreads=300
```

To change the value, use the same command with an equal sign and value as seen in the following example:

```
proenv> tcman.sh config psc.as.executor.maxthreads=400

proenv> psc.as.executor.maxthreads=400
```

Adjust this parameter beyond the recommended 300 when you also wish to increase your maximum concurrent connections.

openedge.properties

The `openedge.properties` file allows you to set the number of agents, sessions, and connections per agent. The defaults are not tailored to your specific application, so expect to change them until you get them correct for you application and expected load.

The following properties show the `openedge.properties AppServer.SessMgr` defaults:

```
[AppServer.SessMgr]

maxAgents=10

maxConnectionsPerAgent=16

maxABLSessionsPerAgent=200

numInitialAgents=1
```

For your first test, you should set the properties small enough to be sure you don't encounter any bottlenecks. This will provide a baseline for your best expected round-trip (elapsed) time.

For your baseline test, you may want to change the following parameters:

```
[AppServer.SessMgr]

maxAgents=1

maxConnectionsPerAgent=50
```

Generally, we advise running as many connections per agent all on a single agent with as many sessions as it can handle before performance degrades. This generally gives the best performance and the best usage of resources.

These parameters should be increased as you test on larger machines.

Performance Testing and Machine Sizing Process

Follow these steps to run your performance tests and size your machine:

1. Start your database on a machine big enough to handle the requests.
2. Start your PAS for OpenEdge instance with the minimal client values connecting to the database at startup.

   In our baseline test, we set the following `openedge.properties` values to establish minimal client values:

   - `maxAgents=1`
   - `maxConnectionsPerAgent=50`

3. Start scripts to gather system information (top) and PAS for OpenEdge information (OE Manager REST API) from your monitoring tools. See the sample scripts provided in the Appendex for more information.

   Note: For further information about OpenEdge monitoring tools, see the OE Manager REST API or OE JMX documentation.

4. Start your test harness, for example JMeter, running a small concurrent user count (such as 10).
5. Verify best baseline performance. If a bottleneck is found at this stage, fix that issue before continuing the tests.
6. Save all testing information.
7. Increase concurrent user load, and `maxConnectionsPerAgent`, incrementally.

   - When you get to 300 concurrent clients, increase the Tomcat `maxthreads` values to be equal to `maxConnectionsPerAgent`.
   - Keep `maxAgent=1` unless your specific application requires multiple agents.

8. Increase all settings on the specific machine size until round-trip (elapsed) time gets longer.
9. Analyze your bottleneck.
10. Correct bottleneck (if CPU 100% or Memory 100%, increase machine size).

A

Gather System Information

This appendix shows the results of system monitors active during the ATM Test. It also includes sample scripts for collecting system information. These sample scripts must be trailored to your environment.

For details, see the following topics:

- Monitoring with Top Command
- Sample Scripts for Gathering System Information

Monitoring with Top Command

The following table shows the output of running the top command for monitoring our PAS for OpenEdge server during the ATM test:

Monitoring with TOP Sample Output:

top - 16:25:13 up 159 days, 14:33, 21 users, load average: 1.01, 1.02, 1.00

Tasks: 374 total, 1 running, 373 sleeping, 0 stopped, 0 zombie

Cpu(s): 0.8%us, 1.6%sy, 0.0%ni, 96.0%id, 0.0%wa, 0.0%hi, 1.6%si, 0.0%st

Mem: 8045544k total, 7529984k used, 515560k free, 321100k buffers

Swap: 6495228k total, 3919288k used, 2575940k free, 4740468k cached

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
34031	bob	20	0	59356	1912	484	D	8.6	0.0	2872:19	_mproapsv

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
45296	roy	20	0	19532	1608	1044	R	0.7	0.0	0:00.04	top
1762	root	20	0	209m	7320	1212	S	0.3	0.1	109:30.56	vmtoolsd
2554	qpidd	20	0	481m	1168	788	S	0.3	0.0	11:19.23	qpidd
49314	roy	20	0	98.2m	2676	1516	S	0.3	0.0	0:02.95	sshd
62073	root	20	0	4783m	732m	19m	S	0.3	9.3	48:45.89	java
90709	bob	20	0	4486m	293m	18m	S	0.3	3.7	0:35.71	java
117103	bette	20	0	206m	3920	932	S	0.3	0.0	80:07.48	vmtoolsd
123863	root	20	0	4134m	478m	21m	S	0.3	6.1	3:17.90	java

Explanation of TOP data from above:

TOP Data from above	Explanation of data	What to look for that describes bottlenecks
Load Average 0.0,0.0,0.0	Last 5, 10, 15 minutes	When load average equals # of CPU it is considered 100% loaded
Cpu(s) %us %sys %id %wait	User percentage of all CPUs System percentage of all CPUs Idle percentage of all CPUs Wait percentage of all CPUs	Most time should be _mproapsv Too much here? Swapping? 0% means 100% Load Time waiting? Swapping?
Memory	Total and used	
Swap	Memory swapped to disk	Always want 0 memory in swap
COMMAND _mproapsv java	Should be using most CPU Should be in the top few (#2)	Remember this is 100% times # of CPUs

Sample Scripts for Gathering System Information

Monitoring with OEManager REST API

The following “jget” commands gather information from the OEManager REST API. The command will get the agent status, which includes session memory size. The metrics command retrieves PAS for OpenEdge instance information, including: requests, reads, writes and errors.

Gather System and PAS for OpenEdge Information Sample Script

The following example pseudo-code script loops through top and OE JMX to create system information files periodically. It is based off of a script we ran during the performance and sizing tests during the ATM Test application run. It can be used as a basis for creating a script for your environment.

```
#!/bin/sh
# this file:  perf-loop.sh
# loop periodically and
#     - gather system information and write to output file
#     - gather PASOE information and write to output file
#
# this is pseudo code, it must be modified to work on your system
#
# create a loop, run myLoopTotal times with a mySleep delay between
# this will run for 10 minutes
#
myLoopTotal=40
mySleep=15
myLoopCount=1
while [ $myLoopCount -le $myLoopTotal ]
do
# gather operating system information
#
# get current date in number format for unique file identifier
#
myDate=`date +%y%m%d%H%M%S`
# run top in batch mode and send to file
top -b -n 2 > top-$myDate.txt
# run oejmx query
<instance-dir>/bin/oejmx.sh -Q <path-to-query-file>/<query-file-name> -O
$WRKDIR/queries-out-$myDate.txt
# please find the queries.txt for an example PASOE query
# increment counter
myLoopCount=`expr 1 + $myLoopCount`
sleep $mySleep
# end loop
done
# end of script
```

Note: The Gather System and PAS for OpenEdge Information Sample script is only for demonstration purposes and must be edited to suit your application environment.

Gather Stats Sample Script

The following example pseudo-code script runs against the created files from the system information script and aggregates information. It is based off of the scripts run during the performance and sizing tests during the ATM Test application run. It can be used as a basis for your own script.

```
Ga#!/bin/sh
# this file:  gather-stats.sh
# Reads all the top and oejmx output files and
# displays a summary of important values
# Files are expected to be in the same directory where  the script is run
#
# this is pseudo code, it must be modified to work on your system
#
# gather information and echo to screen
LOADAVG=`grep "load average" top*.txt | sed 's/:/,/g' | cut -d , -f 9 | sort | tail -1`
echo "max load average is " $LOADAVG
echo ""
USER=`grep Cpu top*.txt | sed 's/:/,/g' | cut -d , -f 3 | sort | tail -1`
echo "max system user cpu is " $USER
SYSTEM=`grep Cpu top*.txt | sed 's/:/,/g' | cut -d , -f 4 | sort | tail -1`
echo "max system cpu is " $SYSTEM
IDLE=`grep Cpu top*.txt | sed 's/:/,/g' | cut -d , -f 6 | sort -r | tail -1`
echo "max system idle is " $IDLE
WAIT=`grep Cpu top*.txt | sed 's/:/,/g' | cut -d , -f 7 | sort | tail -1`
echo "max system wait is " $WAIT
echo ""

MEM1=`grep Mem top*.txt | sed 's/:/,/g' | cut -d , -f 4 | sort | head -1`
echo "system memory to start " $MEM1
MEM3=`grep Mem top*.txt | sed 's/:/,/g' | cut -d , -f 4 | tail -1`
echo "systme memory at end " $MEM3
echo ""
SWAP=`grep Swap top*.txt | sed 's/:/,/g' | cut -d , -f 4 | tail -1`
echo "Swap memory at end" $SWAP
echo ""

JMEM1=`grep java top*.txt | sed 's/  */ /g' | cut -d ' ' -f 5 | head -1`
echo "PASOE memory to start " $JMEM1
JMEM2=`grep java top*.txt | sed 's/  */ /g' | cut -d ' ' -f 5 | tail -1`
echo "PASOE memory to end " $JMEM2
echo ""

MSAMEM1=`grep _mproapsv top*.txt | sed 's/  */ /g' | cut -d ' ' -f 5 | head -1`
echo "MSAGENT memory to start " $MSAMEM1
MSAMEM2=`grep _mproapsv top*.txt | sed 's/  */ /g' | cut -d ' ' -f 5 | head -4 | tail
-1`
echo "MSAGENT memory quickly to " $MSAMEM2
MSAMEM3=`grep _mproapsv top*.txt | sed 's/  */ /g' | cut -d ' ' -f 5 | tail -1`
echo "MSAGENT memory at end " $MSAMEM3
echo ""

MSACPU=`grep _mproapsv top*.txt | sed 's/  */ /g' | cut -d ' ' -f 9 | sort | tail -1`
echo "max MSAGENT cpu " $MSACPU
echo ""

echo "getMetrics output at end of test "
grep getMetrics `ls -tr queries-out*.txt | tail -1` | sed 's/,/,\n/g'
echo ""
echo "total requests in localhost_access_log"
cat $WRKDIR/pasoeatm/logs/localhost_access_log*.txt | wc
echo "total errors in log files"
grep -i error $WRKDIR/pasoeatm/logs/* | wc
```

Note: The Gather Stats Sample script is only for demonstration purposes and must be edited to suit your application environment.

OE JMX Sample Script

The following OE JMX sample script is a pseudo-code version of a script that was run during the performance testing on the ATM Test application for PAS for OpenEdge to gather system information.

The script runs a list of queries against PAS for OpenEdge using OE JMX:

```
# Tokens: APP_NAME, AGENT_ID, AGENT_PID, SESSION_ID
{"O":"java.lang:type=Memory","A":"HeapMemoryUsage"}
{"O":"java.lang:type=MemoryPool,name=PS Eden Space","A":"PeakUsage"}
{"O":"PASOE:type=OEManager,name=AgentManager","M":["getAgentMetrics","AGENT_ID"]}
{"O":"PASOE:type=OEManager,name=AgentManager","M":["getSessionMetrics","AGENT_ID"]}
{"O":"PASOE:type=OEManager,name=AgentManager","M":["getConnectionMetrics","AGENT_ID"]}
{"O":"PASOE:type=OEManager,name=SessionManager","M":["getMetrics","APP_NAME"]}
```

Note: The OE JMX Sample script is only for demonstration purposes and must be edited to suit your application environment.