Performance Guide

IPS PAGER

telisca Applications for
Cisco Unified Communications

Directory Solutions
IPS Global Directory
Web Directory
IPS Popup
Personal Directory
ClickNDial
Provisioning
Corporate Speed Dials
IP Phone Applications
IPS Manager Assistant
IPS Lock
IPS Phone Config
IPS Alarm Callback
IPS Pager
IPS Reservation
Audio Applications
Busy Alerter Callback
TAnnounce
Wake-up Call
Voice Alert
Voice Callback
Recording Notification

Jabber Applications
Conference tab
Directory tab
Manager Assistant
User Settings
Pin & Password Manager
Productivity
Extension Mobility Single
Sign-on
FrontDesk Attendant Console
Conference Center
Missed Calls Email Alert
Admin Tools
Phone Robot
Delog-Relog
PIN & Password Manager
CUCM Query
User to Phone Deployment
Contact Center
Callback Server
Desktop Popup
MediaSense Extensions
Silent Monitoring

Version: 1.0.0

Summary

1 Document Overview

This document provides a guideline for IPS PAGER server's sizing.

IPS Pager is an alerting solution for Cisco IP Phones which sends text, graphical and sound message. The alert can be triggered from a phone, a web interface or an API.

Please find the product's datasheet on:

http://telisca.com/docs/IPSPAGER_EN.pdf

Or the admin guide:

http://telisca.com/docs/IPSPAGER_ADMIN_EN.pdf .

The goal of the document is to provide information on IPS Pager performances depending on the number of phones to alert.

2 Study

2.1 Message push description

This chapter describes the different steps needed to send a text message by HTTP to a Cisco phone. HTTP push mode is advised for massive push as it provides the best performances and minimize CUCM used resources. We advise to configure our authentication proxy (see Administration guide). In this case, only audio notifications may still require some resources from CUCM TFTP server.
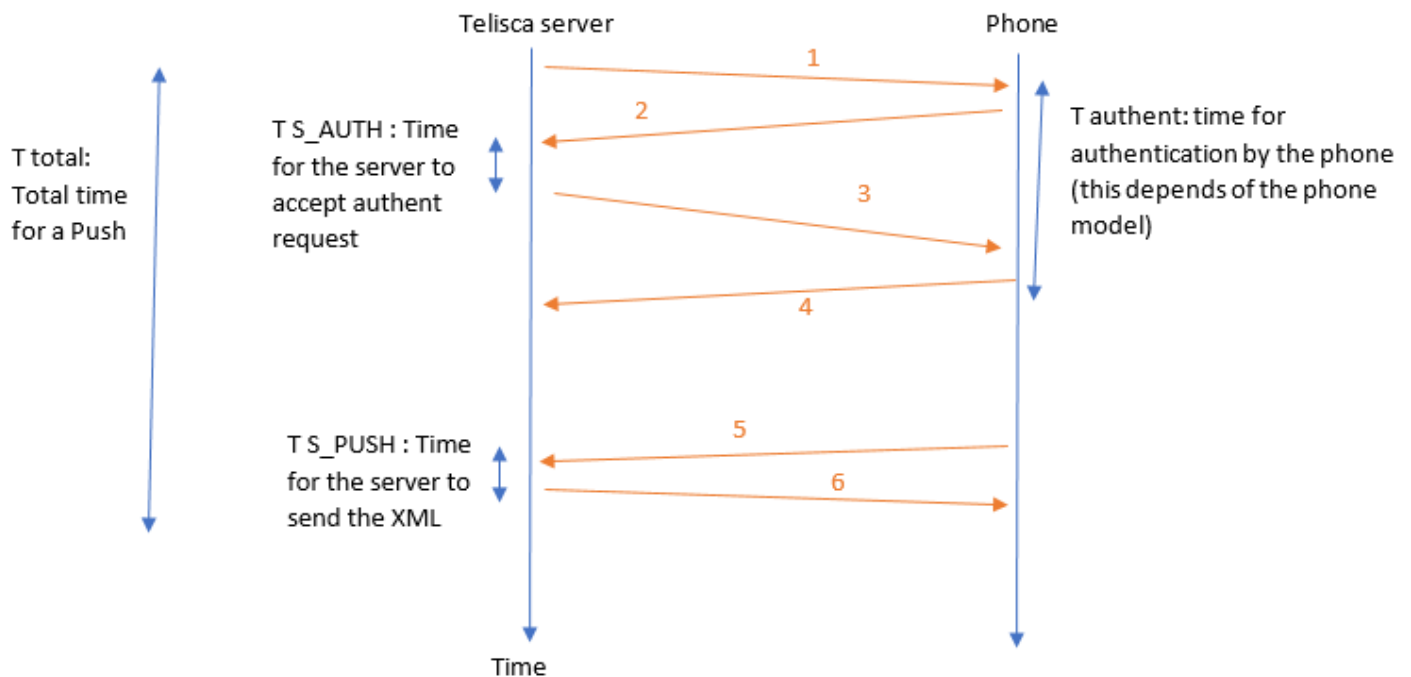
Messages description:

1. PUSH Server.

```
Push2Phone2;Push to => 10.2.111.100 :
<CiscoIPPhoneExecute>
<ExecuteItem Priority="0"
URL="http://10.1.1.236:80/IPSPUSH/user/Default.aspx?grp=1&amp,lst=14GM75&amp,msg=14GM7
U&amp,snd=192.168.0.135&amp,si=1&amp,li=1&amp,qe=1&amp,user=%3f%3f%3f%3f&amp,pn=SEP
B000B4BA4E0C"/>
</CiscoIPPhoneExecute>, loginPwd=X
```

3. Authentication request. The phone accepts the request. The phone uses an ad hoc user name temporary accepted by the server:

```
M;cPush;Push2Phone2;IP Phone : 10.2.111.100 =>
CiscoIPPhoneResponse =
<?xml version="1.0" encoding="utf-8"?>
<CiscoIPPhoneResponse>
<ResponseItem
URL="http://10.1.1.236:80/IPSPUSH/user/Default.aspx?grp=1&amp,lst=14GM75&amp,msg=14GM7
U&amp,snd=192.168.0.135&amp,si=1&amp,li=1&amp,qe=1&amp,user=%3f%3f%3f%3f&amp,pn=SEP
B000B4BA4E0C" Data="Success" Status="0" />
</CiscoIPPhoneResponse>
```

4. Authentication accepted by the server

> 170411;11:02:57.138;72;;methods;default.aspx;Page_Load;(10.2.86.99) One Time Pwd : AUTHORIZED

5. Authentication acknowledgment

> M;cPush;Push2Phone2;Push 10.2.111.100 : OK
> (URL=http://10.1.1.236:80/IPSPUSH/user/Default.aspx?grp=1&lst=14GM75&msg=14GM7U&snd=19
> 2.168.0.135&si=1&li=1&qe=1&user=%3f%3f%3f%3f&pn=SEPB000B4BA4E0C)

6. Message loading by the phone

> SEPB000B4BA4E0C;T;Default.aspx;Page_Load;
> http://10.1.1.236:80/IPSPUSH/user/Default.aspx?grp=1&lst=14GM75&msg=14GM7U&snd=192.168.
> 0.135&si=1&li=1&qe=1&user=%3f%3f%3f%3f&pn=SEPB000B4BA4E0C FROM 10.2.111.100,
> phoneName=SEPB000B4BA4E0C

7. Send XML to the phone

> CCiscoIPPhone;SendXmlResponse;REFRESH_HEADER=60,
> url=http://10.1.1.236:80/IPSPUSH/user/Default.aspx?grp=1&msg=14GM7U&quit=1&pn=SEP544A00
> 37389A
> XML=<?xml version="1.0" encoding="UTF-8"?>
> <CiscoIPPhoneText><Title>Test</Title><Prompt>test_thomas</Prompt><Text>
> Test this is a
> test</Text><SoftKeyItem><Name>Quit</Name><URL>http://10.1.1.236:80/IPSPUSH/user/Default.
> aspx?grp=1&amp,msg=14GM7U&amp,quit=1</URL><Position>3</Position></SoftKeyItem></Cisco
> IPPhoneText>

T S_AUTH and T_PUSH depend on a server processing. So, these times may increase depending on the server load (number of push per second).

2.2 Multi-threading

The application manages multi-threading to send message to several phones almost at the same time. The figure bellow describes the behavior of the server for 3 threads and 4 messages pushed on 4 different phones.
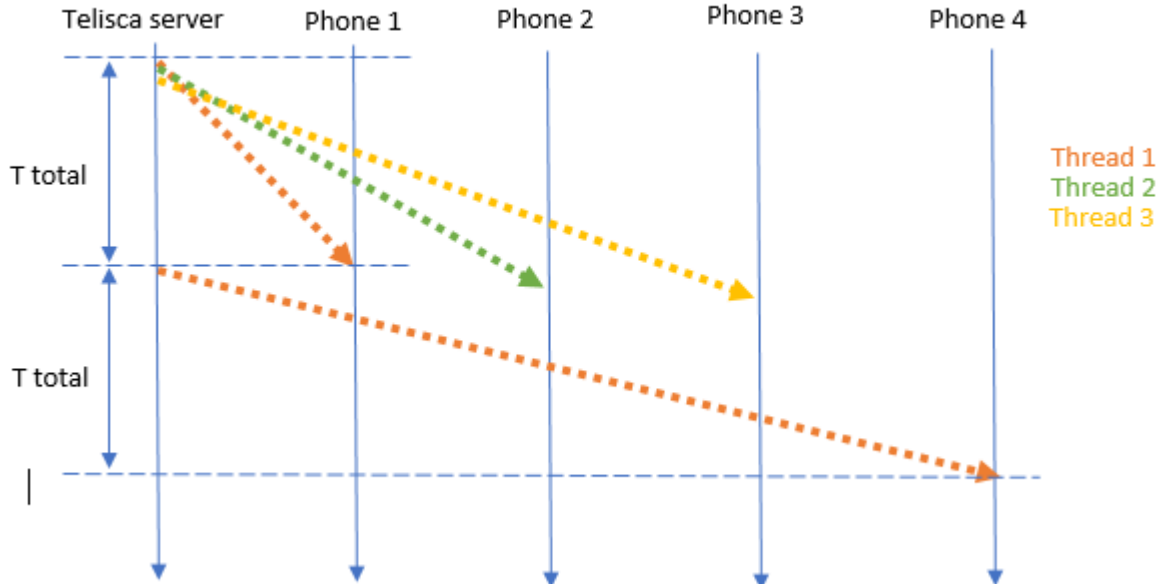
Requests from different threads are sent almost at the same time. But a single thread will wait for the end of his current push before sending a new push to another phone.

2.3 Experimental environment

A test machine simulates the requests sent to the server. Two types of requests have been simulated, an authentication request (on IPS Pager authentication proxy) and the page load request. The study is focused on these requests because they require server processing to be answered.

The tests will be divided in two parts, with a 1 vCPU server and a 2 vCPU-server.

The telisca server used for the tests has 4GB of RAM and is running Windows Server 2016.

2.4 Tests results
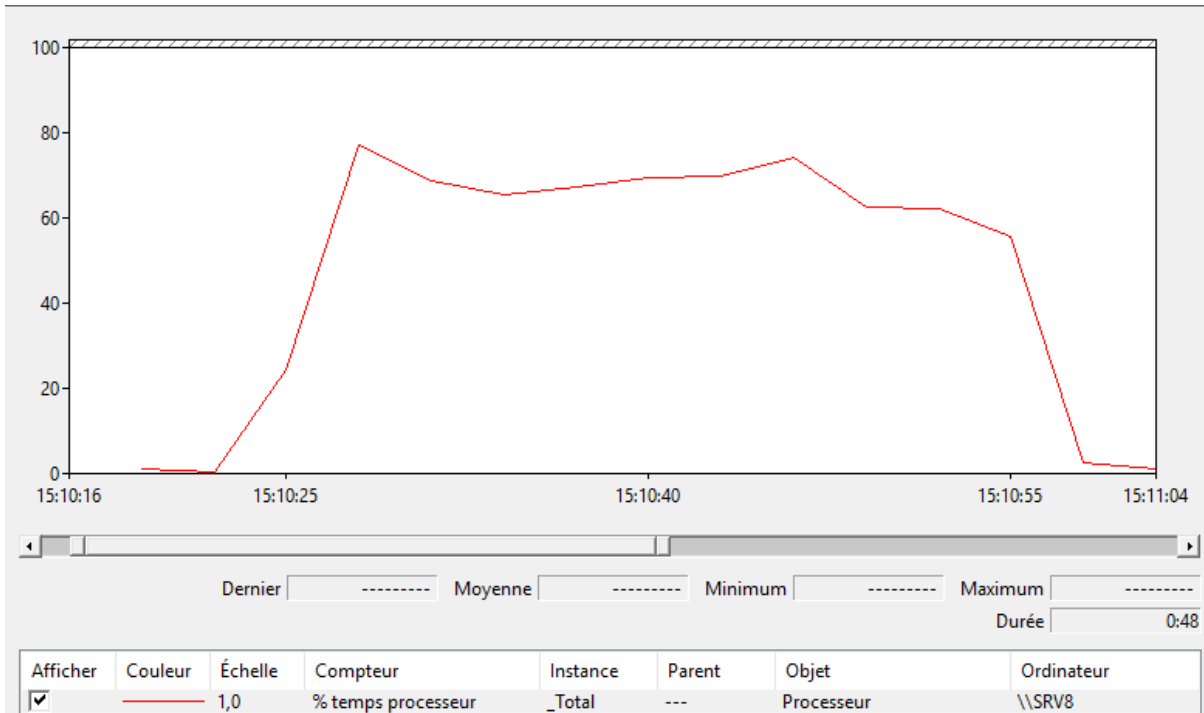
100 threads, 200ms delay for a 30s test
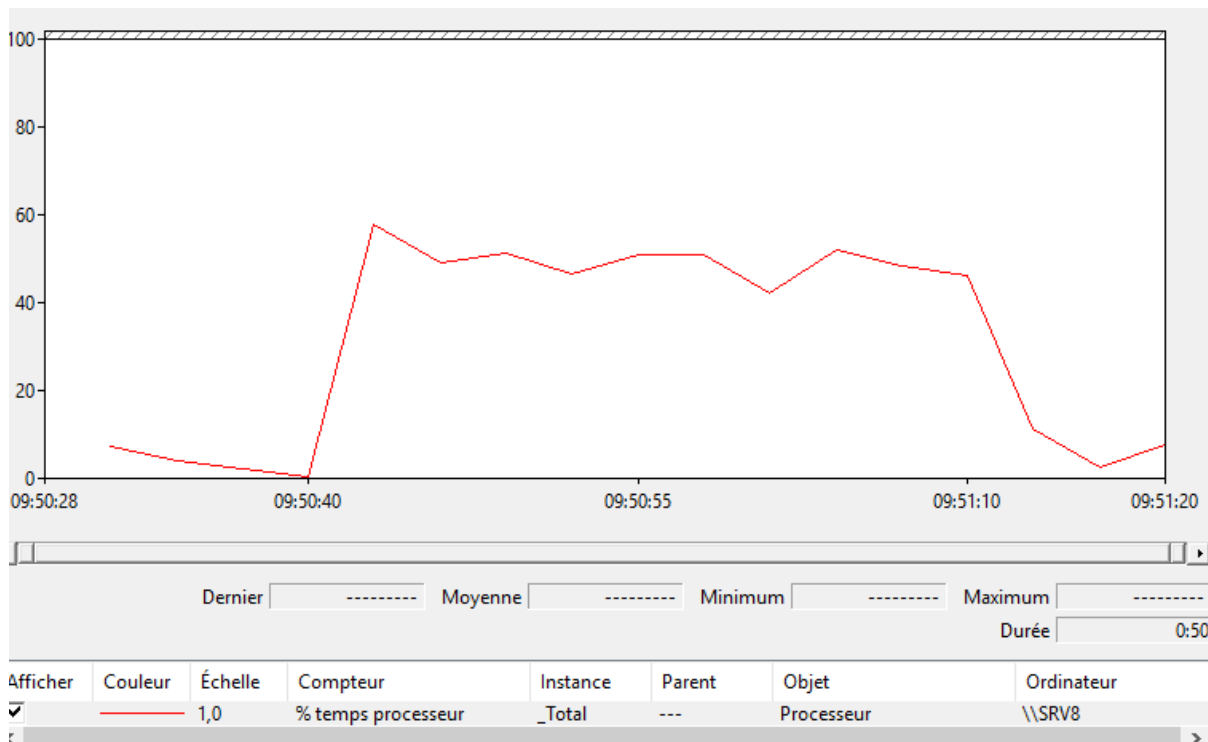

Figure 1: 1vCPU - 100 Threads

Figure 2: 2vCPU - 100 Threads

The CPU is around 70% of usage for 1 vCPU and below 60% for 2vCPU.
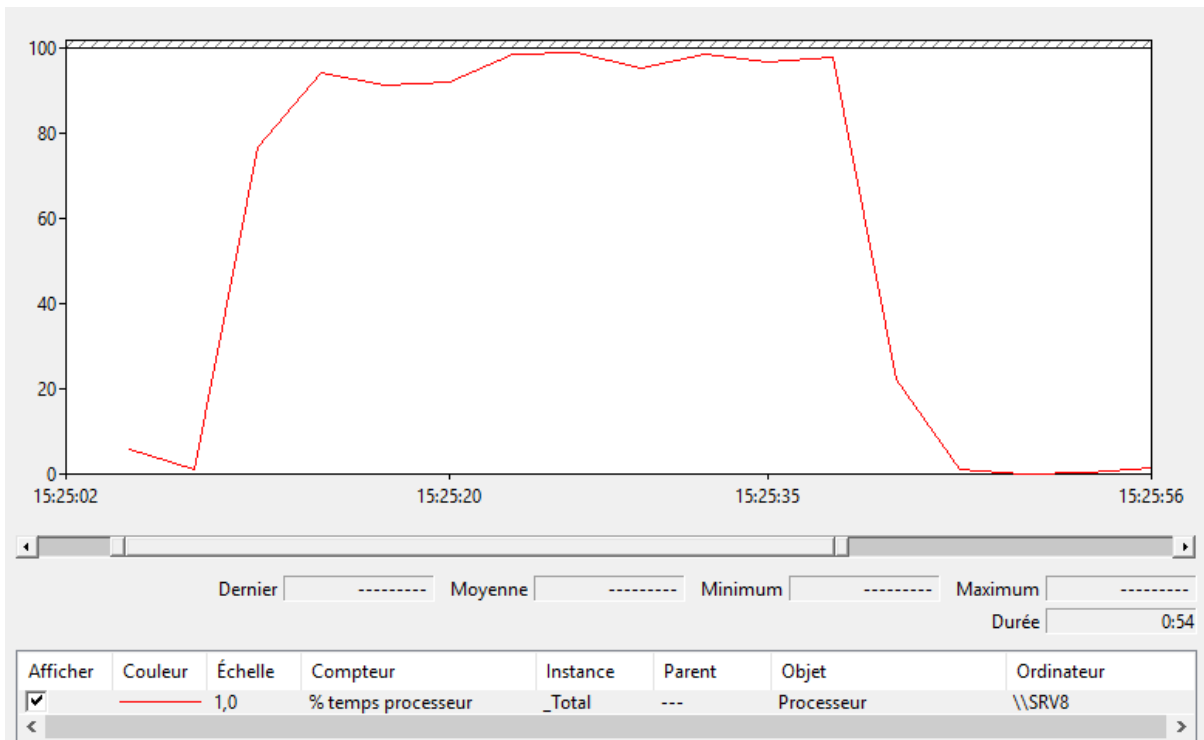
200 threads, 200 ms delay for a 30s test
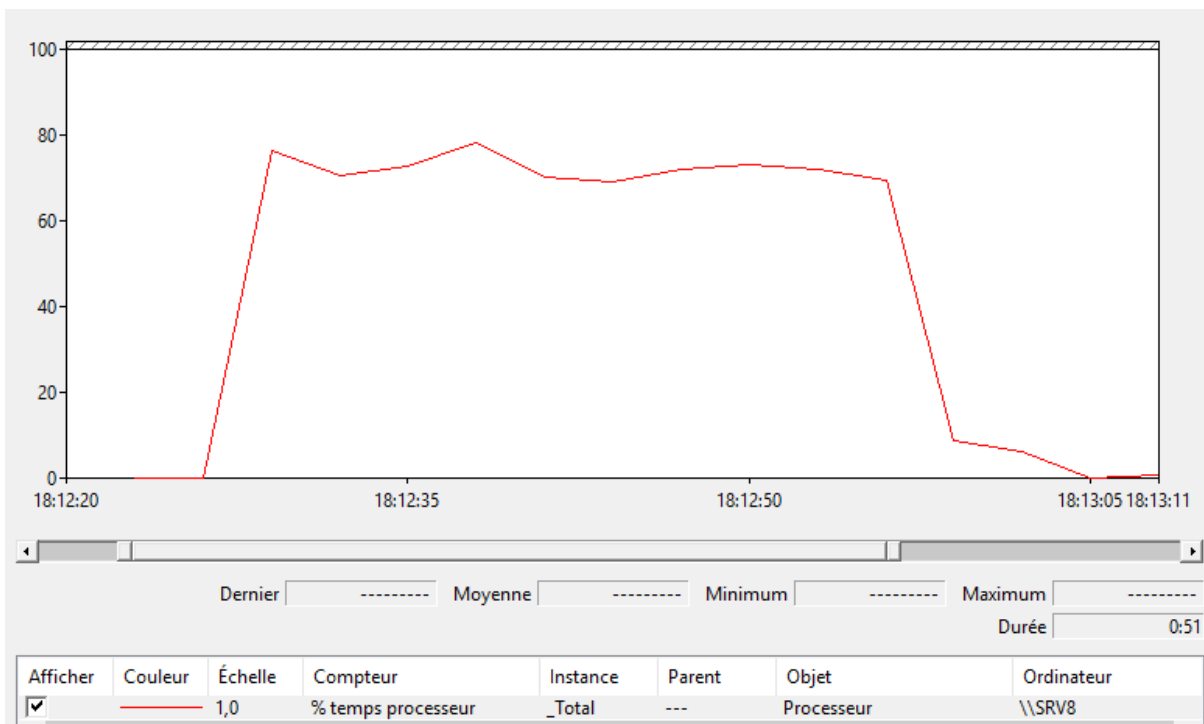

Figure 3: 1 vCPU - 200 Threads

Figure 4 : 2vCPU - 200 Threads

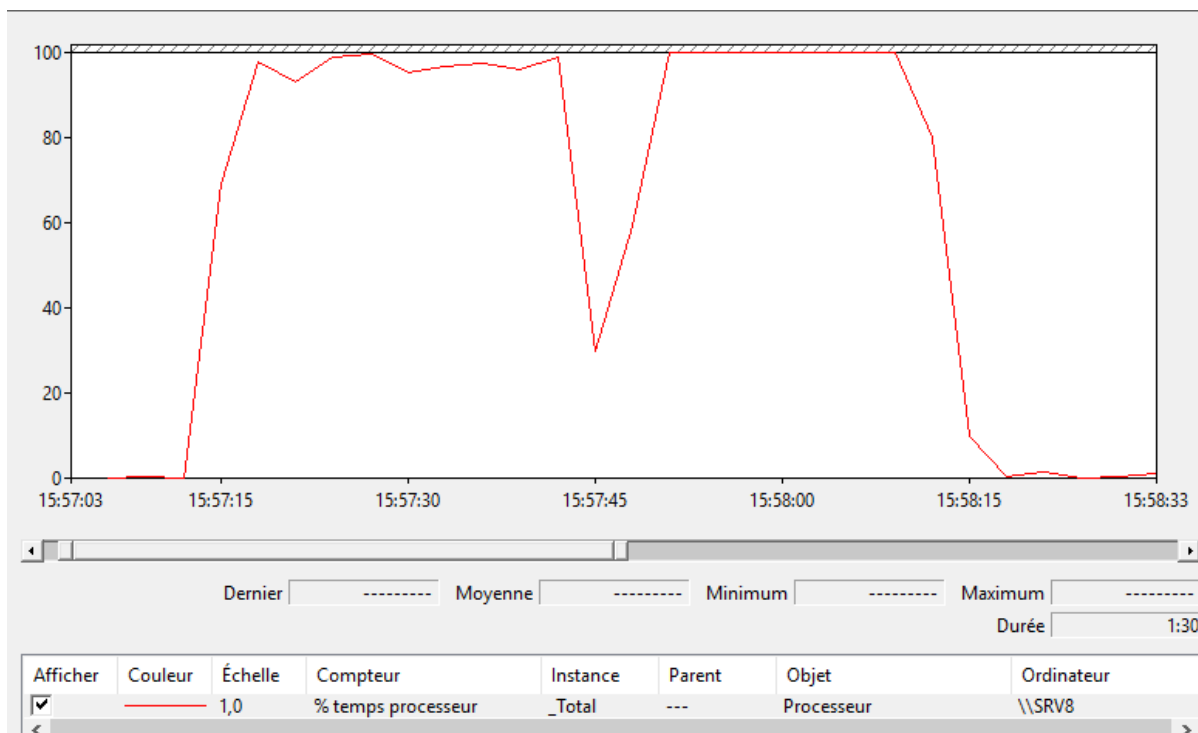The CPU is around 95% for 1 vCPU and 80% for 2 vCPU.
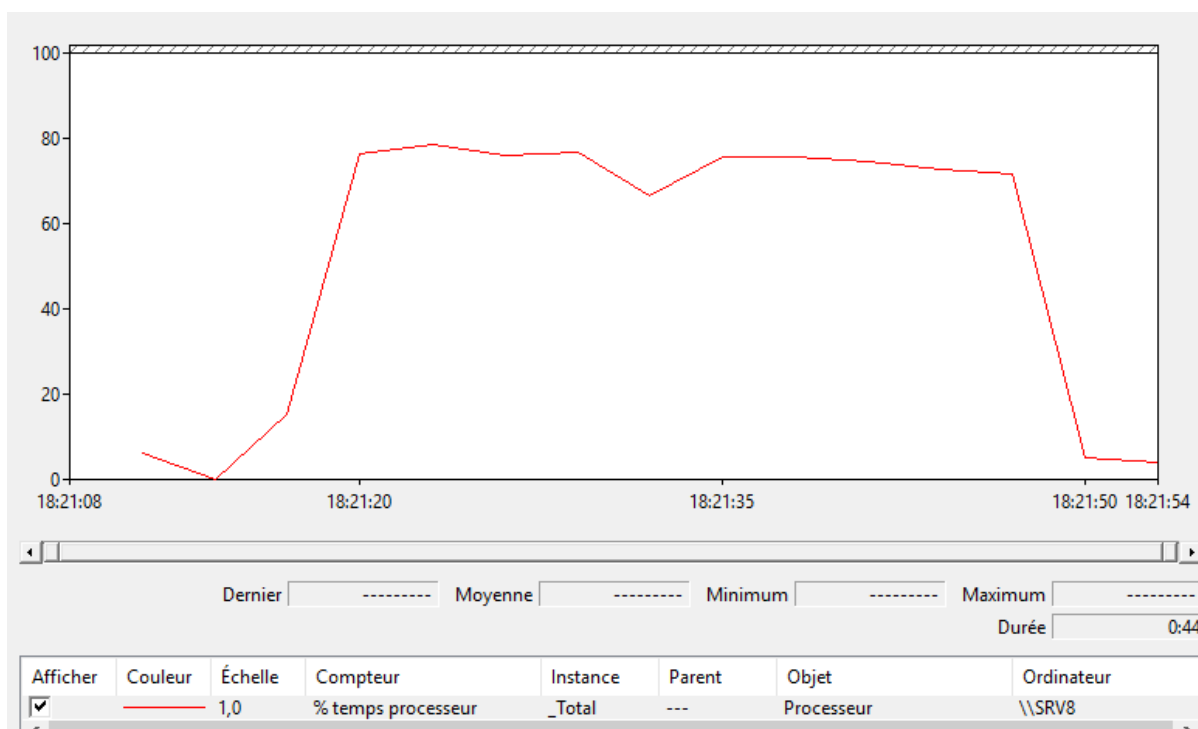
Figure 5 : 1 vCPU - 300 Threads

Figure 6: 2 vCPU - 300 Threads

The CPU Usage is around 95% and 80% for 2 vCPU.

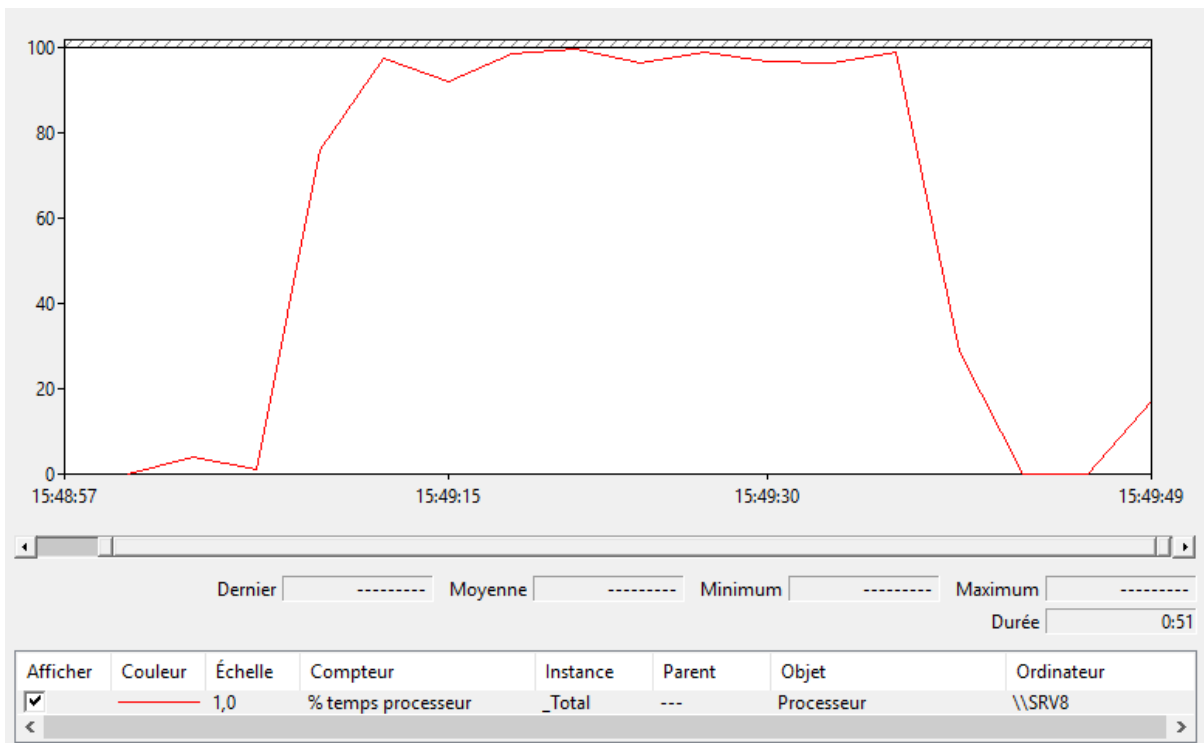400 threads, 200 ms delay for a 30s test
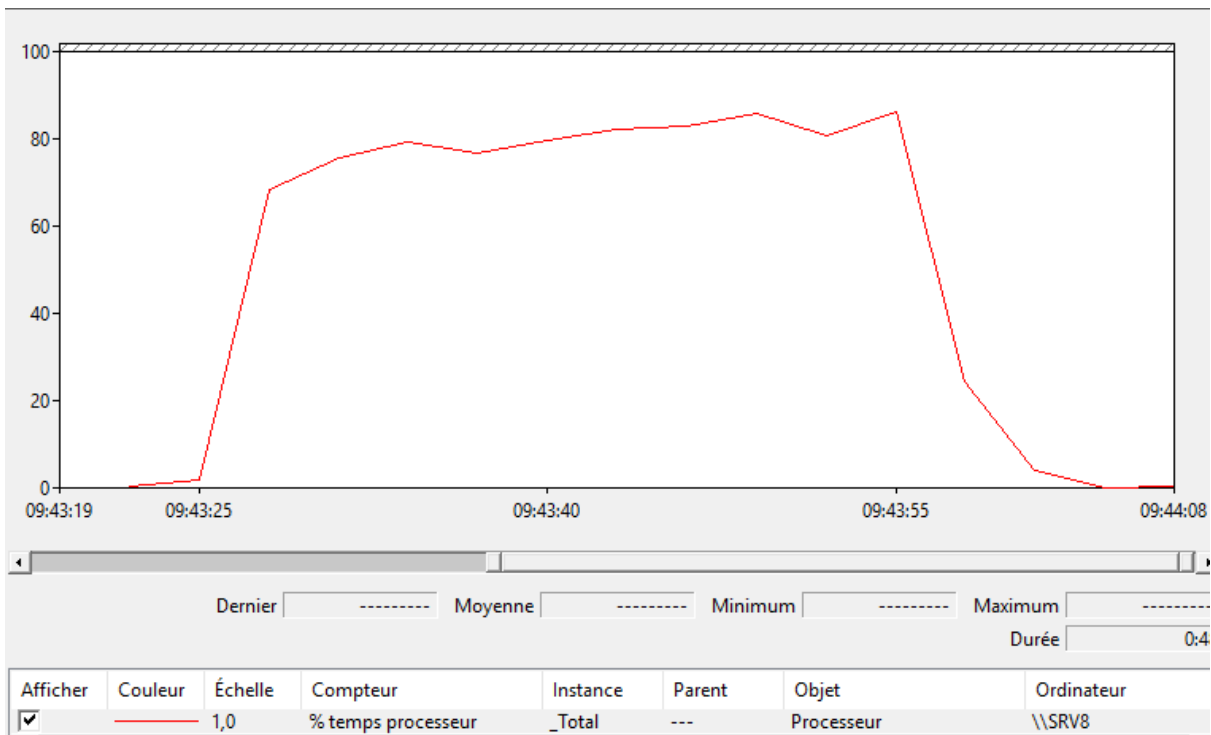

Figure 7: 1 vCPU - 400 Threads

Figure 8: 2vCPU - 400 Threads

The CPU Usage is around 100% for 1 vCPU and 85% for 2 vCPU.

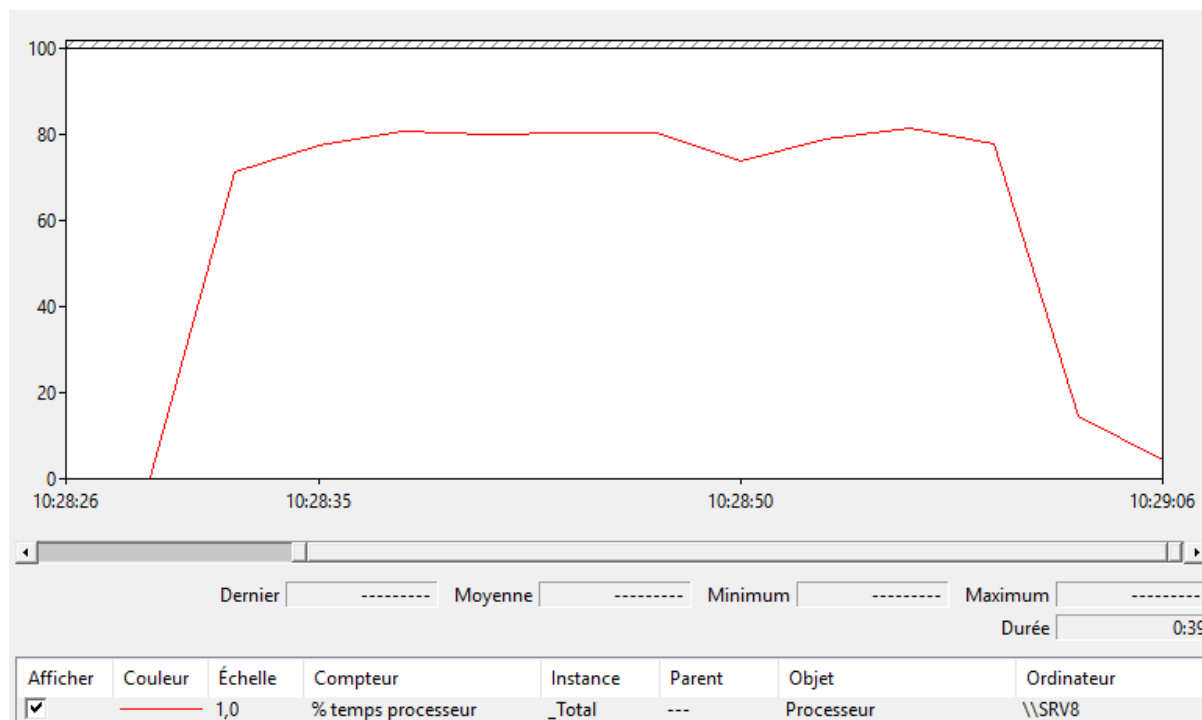500 threads, 200 ms delay for a 30s test

Figure 9: 2 vCPU - 500 Threads

The CPU is around 85% for 2 vCPU.

2.5 Study's Results

Depending of phone models, the authentication response can take between 100ms and 300ms (most of them are under 200ms). To simulate this delay, all the series of requests are sent with a 200ms interval.

The testing has been done with a different number of threads (from 1 to 400).

Each thread sends an authentication plus a push requests and wait for the responses. A 200ms delay is applied after the response is received, before sending a new request.

Table 1: Results for 1 vCPU server

# Thread	1	5	10	25	50	100	200	300	400
Requests sent (~30s)	270	1387	2789	6830	12938	23155	37991	41578	41882
Req/s	4,543703	23,21883	46,63959	114,1568	216,5335	381,1966	635,1573	694,2312	694,761
% CPU	15%	15%	20%	30%	50%	70%	95%	95%	95%

# Thread	10	25	50	100	200	300	400	500
Requests sent (~30s)	2751	6902	13802	25591	37248	43491	45976	47794
Req/s	45,93267	115,6038	231,2913	428,0588	629,0307	722,3901	772,5451	789,5449
% CPU	15%	15%	40%	60%	80%	80%	85%	85%

Table 2: Experimental results for a 2 vCPU server
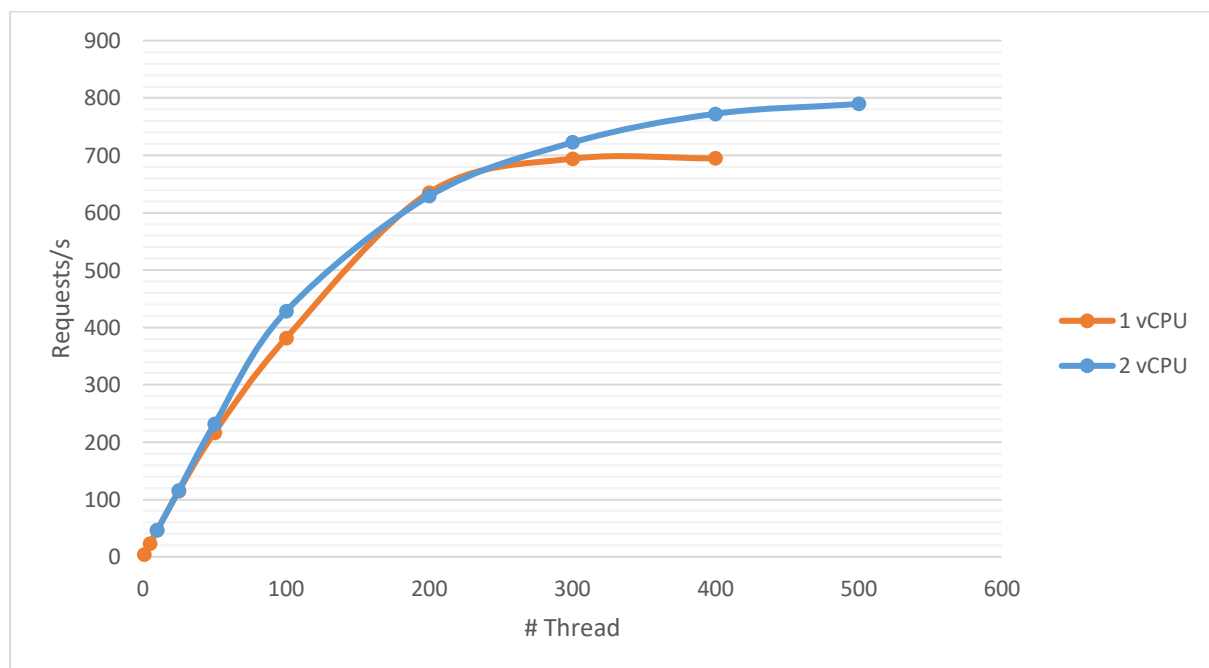
- For 1 vCPU : During the 200 threads test, the server CPU load reach 95%. For this configuration, the server threat more than 600 requests/s.
- For 2 vCPU : During the 400 threads test, the server is able to manage more than 750 request/s.

3 Recommendations for IPS Pager

These tests have been done in a laboratory, with a single application server and the first push request hasn't been simulated. So, the recommendations included a 20% marge to handle all unexpected events on a production system.

Also, the results are calculated based on fixed phone authentication delay (200ms). The number of alerted phones per second depends of this delay and may vary depending of the cisco phone's model used.

With a **1 vCPU** server, the number of threads configured must be under 200. With 200 threads, we can alert up to **480 phones per seconds**.

With a **2 vCPU** server, the maximum number of threads configured in the admin must be 400. With 400 threads, the system can alert up to **600 phones per second**.