ProLUG – Baselines, load/stress testing, and building a test plan.

Required Materials

Putty

Rocky Server

Root or sudo command access

EXERCISES (Warmup to quickly run through your system and familiarize yourself)

1. mkdir lab_baseline
2. cd lab_baseline
3. which iostat
    a. If it's not there do this
        i. dns whatprovides iostat
            This should tell you it's sysstat
        ii. rpm -qa | grep -i sysstat
        iii. dnf install sysstat
        iv. rpm -qa | grep -i sysstat
4. which stress
    a. If it's not there do this
        i. dnf whatprovides stress
        ii. dnf install stress
        iii. rpm -qa | grep -i stress
        iv. rpm -qi stress
            Read the description of what this tool does.
5. which iperf3
    a) If it's not there do this
        1. dnf whatprovides iperf
        2. dnf install iperf
        3. rpm -qa | grep -i iperf
        4. rpm -qi iperf

LAB

There will be different scenarios during today's lab:

1. You have decided to take a basic system baseline of all your systems.

2. You will develop a test plan to execute in part 3.

3. You will execute your test plan and gather results.

Baseline information gathering:

The purpose of a baseline is not to find fault, load, or to take corrective action. A baseline simply determines what is. You must know what is so that you can test against that when you make a change to be able to objectively say there was or wasn't an improvement. You must know where you are at to be able to properly plan where you are going. A poor baseline assessment, because of inflated numbers or inaccurate testing, does a disservice to the rest of your project. You must accurately draw the first line and understand your system's performance.

Using SAR (cpu and memory data)

Some useful sar tracking commands. 10 minute increments
sar #by itself, this gives the last day in processing numbers
sar -r #gives memory statistics
sar -W #gives swapping statistics. Useful to see if system is out of physical memory often
ls /var/log/sa/
sar -f /var/log/sa/sa28 #view sar data from a recent day of the month.
For your later labs, you need to collect sar in real time to compare with the baseline data
systemctl cat sysstat-collect.timer #to view how the tool collects data every 10 minutes.
to do this use:
sar 2 10 #this will give sar info every 2 seconds 10 times sar -r 2 10 is the same thing

Using IOSTAT (cpu and device data)

iostat will give you either processing or device statistics for your system

iostat #by itself gives all information proc and device
iostat -c #will give only processor
iostat -d #will give only device information
iostat -c 1 #will give 1 second proc info until broken
iostat -c 1 5 #will give 1 second proc info 5 times

Using iperf3 (in the lab) to test network speeds

In the lab, red1 is the iperf3 server, so we can bounce connections off it (192.168.200.101)
time iperf3 -c 192.168.200.101 -n 1G -P 128 #TCP connection 128 connections
time iperf3 -c 192.168.200.101 -u -n 1G -P 128 #UDP connection 128 connections
How long do these take?

Using STRESS to generate load tests

stress will produce extra load on a system. It can run against proc, ram, and disk I/O.

```
stress   #by itself gives usage information
stress -c 1  #will not stop running against the cpu
stress --cpu 8 --io 4 --vm 2 --vm-bytes 128M -d 1 --timeout 10s #This will do a lot of things.
```

Read the usage to figure it out.

Developing a test plan

The company has decided we are going to add a new agent to all machines. Management has given this directive to you because of PCI compliance standards with no regard for what it may do to the system. You want to validate if there are any problems and be able to express your concerns as an engineer, if there are actual issues. No one cares what you think, they care what you can show, or prove.

1. Determine the right question to ask:

   Do we have system baselines to compare against?

   No? Make a baseline.

   ```
   iostat -xh 1 10
   ```

   Can we say that this system is not under heavy load?

   What does a system under no load look like performing tasks in our environment?

   Assuming our systems are running not under load, capture SAR and baseline stats.

   Perform some basic tasks and get their completion times.

   Writing/deleting 3000 empty files #modify as needed for your system speed ~10s

   ```
   time for i in `seq 1 3000`; do touch testfile$i; done
   ```

   removing them

   ```
   time for i in `seq 1 3000`; do rm -rf testfile$i; done
   ```

   writing large files

   ```
   for i in `seq 1 5`; do time dd if=/dev/zero
   of=/root/lab_baseline/sizetest$i bs=1024k count=1000; done
   ```

   removing the files

   ```
   for i in `seq 1 5`; do rm -rf sizetest$i ; done
   ```

   Testing processor speed

time $(i=0; while ((i < 999999)); do ((i ++)); done)
if this takes your system under 10 seconds add a 9

Alternate processor speed test

time dd if=/dev/urandom bs=1024k count=20 | bzip2 -9 >> /dev/null
This takes random numbers in blocks, zips them, and then throws them away. Tune to about ~10 seconds as needed

What is the difference between systems under load with and without the agent?

Run a load test (with stress) of what the agent is going to do against the system
While the load test is running, do your same functions and see if they perform differently.

Execute the plan and gather data.

Edit these as you see fit, add columns or rows to increase understanding of system performance. This is your chance to test and record these things.

System Baseline Tests:

System	Sar average load for past week	Iostat test 10 min	Iostat test 2 10 min	Disk write small files	Disk write small files test 2	Disk write large files	Processor test
Server 1							

You may baseline more than once, more data is rarely bad.

Make 3 different assumptions for how load may look on your system with the agent and design your stress commands around them (examples):

1. while true; do stress --cpu 2 --io 4 --vm 2 --vm-bytes 128M --timeout 30; done #I assume no load on hdd, light load on processors
2. while true; do stress --cpu 2-io 4 --vm 2 --vm-bytes 128M -d 1 --timeout 30; done #I assume low load on hdd, light load on processors

3. while true; do stress --cpu 4 --io 4 --vm 2 --vm-bytes 256M -d 4 --timeout 30; done #I just assume everything is high load and it's a mess

In one window start your load tests (YOU MUST REMEMBER TO STOP THESE AFTER YOU GATHER YOUR DATA):

In another window gather your data again, exactly as you did for your baseline with sar and iostat just for the time of the test.

System Tests while under significant load (put command you're using for load here):

System	Sar average load during test	Iostat test 10 min	Iostat test 2 10 min	Disk write small files	Disk write small files test 2	Disk write large files	Processor test
Server 1							

System Tests while under significant load (put command you're using for load here):

System	Sar average load for past week	Iostat test 10 min	Iostat test 2 10 min	Disk write small files	Disk write small files test 2	Disk write large files	Processor test
Server 1							

(Continue copying and pasting tables as needed)