L12 Cluster Monitor v1.2 03/29/2023

Chenyang Li chenyangl@supermicro.com

Introduction

L12 Cluster Monitor (L12CM) is an easy deploy program based on Redfish and Intelligent Platform Management Interface aims to monitor the system status on super sever remotely. It is under development, currently have following functions:

- real-time device status monitor
- cluster hardware software summary
- firmawre update and debug
- BIOS settings comparison
- benchmark module
- system report generator

Required Docker Images

L12CM is based on Docker, which requires following images:

- **neonadia/rackobserver** Only one per deployment. It is the webpage of racks view. Not necessary for L12 Cluster Monitor (L12CM) core functionality.
- **neonadia/udpserver** One per ".csv" file. Used to send/receive short messages with UDP clients. Necessary for UDP functions and benchmark functions. Not necessary for L12CM core functionality.
- **neonadia/sysmonitorfrontend** One per ".csv" file. It is the webpage of systems view, display sensor reading and advanced features. Necessary for L12CM core functionality.
- **neonadia/sysmonitorsensor** One per ".csv" file. It is the backend of systems view, responsible for periodic sensor reading. Necessary for L12CM core functionality.
- **mongo** Database for sysmonitorsensor and sysmonitorbackend, one per ".csv" file. Necessary for L12CM core functionality. Database for UDP clients, responsible for distributing ports' numbers for UDP clients. The port number for this

Database is 8888. Necessary for UDP functions and benchmark functions. Noe necessary for L12CM core functionality.
- **neonadia/sysmonitorbackend** One per ".csv" file. It is the backend of systems view, responsible for one-time hardware/firmware specs reading. Necessary for L12CM core functionality. Will exit after finishing its job.
- **neonadia/autostart** One per deployment. Pre-deployment set up for L12CM. Necessary for L12CM core functionality. Will exit after finishing its job.
- **willfarrell/autoheal** One per deployment. It can autoheal the unhealthy container by restarting them.

Background Information

L12CM includes the following parts:

- **Auto Start Module**
  - Only run once. **Need to be fnished without any errors to make L12CM functional**.
  - Consume csv file and parse it to pwd*.txt and node_names*.txt.
  - Generates environmental files (port*.env) for all the containers.
  - Creates mapping bettween UDP ports and system mac address and stored it in a MongoDB database with port number 8888.
- **System Reading Module**
  - Only run once. **Need to be fnished without any errors to make L12CM functional**.
  - Most issues are mainly due to: wrong IPMI password, wrong Redfish API or lost connection.
  - Using Redfish API to read hardware and firmware information.
  - Using SQL to query IPMI password from database based on system MAC address.
- **Sensor Reading Module**
  - Periodically running. **At least one set of data needed for L12CM to functional**.
  - Can be pasued/stopped after L12CM functionning.
  - Using Redfish API to read temperatures, voltage, power and fan speed.
  - Using ipmitool to read DCMI power reading and events.
  - Using SMCIPMITOOL to get CPLD version, BMC version and BIOS version.
  - Updating the OS IP based on the host.json file generated by UDP server.
- **MongoDB Database**

    - There are two kinds of MongoDB in L12CM. One for UDP ports mapping and one for storing system data.
    - Each deployment only has one database for UDP ports mapping. Once UDP client get its port number from the mapping, it is no longer needed.
    - Each deployment could have more than one database for storing system data. The number of database is same as number of input csv files.
    - There are at most 5 collections in the system data database:
        - servers: storing data from **System Reading Module**.
        - monitor: storing data from **Sensor Reading Module**.
        - udp: storing benchmark results.
        - cmd: storing UDP client command line outputs.
        - hw_data: storing hardware information loading from the folders.
    - **MongoDB for storing system data is necessary to make L12CM functional**.
- **Webpages and Frontend Module**
    - GUI interface. **Index page will be displayed correctly once L12CM is functionning.**
    - PDF report generator.
    - SUM toolbox for system maintainence.
    - IPMITOOL and Redfish API for system maintainence.
- **UDP Server**
    - UDP Server is not necessarily for L12CM to function.
    - Get port mapping from MongoDB database.
    - Communicate with UDP clients to check connectivity, get benchmark results and commandline outputs.
- **UDP Client**
    - UDP Client is not necessarily for L12CM to function.
    - Handle the message from server and run benchmarks or commands.
    - Parse benchmark results and commandline outputs. Insert them to MongoDB. Sent signal to UDP Server to inform the completion.
    - Can only do one thing at a time.
    - Get port mapping from MongoDB database.

Quick deployment steps

- Install Docker, docker-compose and ncat:
    - Docker: https://docs.docker.com/engine/install/centos/
    - Docker-compose: https://docs.docker.com/compose/install/
    - Ncat:
- `apt install ncat`

or

```
yum install nmap-ncat.x86_64
```

- Configure the default bridge, create “/etc/docker/daemon.json” file and restart docker:

```
[root@bgfs1 ~]# cat /etc/docker/daemon.json
{
  "default-address-pools":
    [
      {"base":"10.10.0.0/16","size":24}
    ]
}
```

- Necessary input files for deployment:
  - "cluster1.csv" is the system access input file.
    - pwd column and name column are optional.
    - OS_IP column is necessary and it can be filled with '0.0.0.0' or 'none' to denote unknown.
    - All other columns are mendatory.
    - Pure number is not allowed in any columns.
  - "auto.env" is the L12CM configuration file. NOTE: INPUTFOLDER must be in /root/

```
[root@localhost ~]# ls INPUTFOLDER/
auto.env    cluster1.csv
[root@localhost ~]# cat cluster1.csv # pwd column and name column are optional.
Node XX,IPMI_IP,OS_IP,eth0,pwd,name
none,172.27.xx.xx,172.27.xx.xx,ac:1f:6b:99:xx:xx,ADMIN,name1
none,172.27.xx.xx,172.27.xx.xx,ac:1f:6b:99:xx:xx,ADMIN,name2
none,172.27.xx.xx,172.27.xx.xx,ac:1f:6b:99:xx:xx,ADMIN,name3

[root@localhost ~]# cat auto.env
IOPATH=INPUTFOLDER   # need to be the same as input folder name
PORTNUM=10000 # note that not only 10000 will be used
PWDSERVER=10.2.1.136 # server for password query
PWDUSERNAME=rackteam # user name for password query
PWDPWD=sMcraCK@17 # password for password query
UDPMONGOPORT=8888 # port number of mongo DB provide UDP client their UDP server port
MACNAME=eth0 # mac address used to identify UDP client and find password
SENSORRATE=HIGH # 30 sec per reading
POWERDISP=ON # please set this to OFF for archive
UIDDISP=ON # please set this to OFF for archive
TIMEZONE=Etc/UTC # timezone for container
DEBUG_MODE=False # flask debug mode and debug toolbar
```

NOTE: Remove all comments from auto.env file and confirm the server ip and credentials you are using

- Optional input files for deployment:
  - "SMCIPMITOOL"
    folder: https://www.supermicro.com/en/support/resources/downloadcenter/smsdownload?category=IPMI.
  - Make sure to rename the folder containing SMCIPMITOOL to SMCIPMITOOL not the name it starts with. And move this folder into INPUTFOLDER.
  - "hw_data" folder. Used for manually upload hardware information. The folder structure can be generated by the 'load hardware data' on hardware_output page.
  - "benchmark_config" folder.

```
├── benchmark_HPL.json
├── benchmark_long_log_test.json
├── benchmark_ROCM_RBT.json
├── benchmark_ROCM_SGEMM.json
├── benchmark_sat.json
├── benchmark_stress-ng-non-num.json
└── benchmark_stress-ng-num.json
```

    Example of config file:

```
{
"category":"benchmark",
"exe":"./stressapptest",
"prefix":"",
"config":"-s 10 -i 2 -c 2 -W --cc_test",
"log":"SAT",
"walltime":3600,
"parseResultLog":1,
"selfLog":0,
"selfLogPath":"",
"numOfResults":3,
"keywords":[["Stats","Memory Copy"],["Stats","Data
Check"],["Stats","Invert Data"]],
"addRow":[0,0,0],
"dfs":["None","None","None"],
"index":[-1,-1,-1],
"unit":["MB/s","KB/s","TB/s"],
"resultName":["Memory Copy","Data Check","Invert Data"],
"criteriaType":["numericType","numericType","numericType"],
"criteria":[[53000,60000],[2200,9000],[800,9000]]
}
```

  - "benchmark_logs" folder.

```
├── benchmark_52_logs_ac-1f-6b-99-05-06
│   ├── 3c-ec-ef-7f-be-50_sat-run_01-11-2022-15-03-57.out
│   ├── bandwidth_rvs_3c-ec-ef-7f-be-50_01-11-2022-16-44-57.log
```

```
    ├── log.3c-ec-ef-7f-be-50_HPL_01-11-2022-15-16-06
    ├── sgemm_3c-ec-ef-7f-be-50_01-11-2022-16-24-59.log
    ├── stress_01-11-2022_16-27-14_3c-ec-ef-7f-be-50.rvs
    ├── stress-ng-non-num-172.27.28.12.log
    ├── stress-ng-num-172.27.28.12.log
    ├── TEST1-172.27.28.12-stress-ng.log
    └── TEST1-172.27.28.14-stress-ng-2.log
└── benchmark_53_logs_ac-1f-6b-99-06-9e
    ├── 3c-ec-ef-7f-be-50_sat-run_01-11-2022-15-03-57.out
    ├── bandwidth_rvs_3c-ec-ef-7f-be-50_01-11-2022-16-44-57.log
    ├── log.3c-ec-ef-7f-be-50_HPL_01-11-2022-15-16-06
    ├── sgemm_3c-ec-ef-7f-be-50_01-11-2022-16-24-59.log
    ├── stress_01-11-2022_16-27-14_3c-ec-ef-7f-be-50.rvs
    ├── TEST1-172.27.28.13-stress-ng-2.log
    └── TEST1-172.27.28.13-stress-ng.log
```

- Build images: (Only necessary if you are doing a git clone method of installing or modifying the code on the server)
- `docker build -t neonadia/<imagename> .`
- Instead of building, you can just pull all the images:
- `docker pull neonadia/<imagename>`
- Start the delpolyment, select **"n"** for **"Do you wish to update the LCM docker images?"** if you are testing your own image:
- `./docker_deploy.sh FOLDERNAME`
- **Optional** Stop firewall:
- `systemctl stop firewalld`
- If the ./docker_deploy.sh command goes through without failure. Open a browser and navigate to: :PORTNUM

Deployment tips

MUST READ

- Some projects are not using unique password, you need to add a column named “PWD” inside “cluster1.csv” file to specify the password. Otherwise, L12CM will look for password from the database.
- “PORTNUM” inside "auto.env" file is the starting number of all ports’ numbers: Rackobserver webpage = PORTNUM Sensor/System mongo DB = PORTNUM + n UDP mongo DB = 8888 Sysmonitorfrontent webpage = PORTNUM + 1000 + n UDP port = PORTNUM + 2000 + n [Where n is the number of clusters]
- **OOB** license is necessary for the basic function of L12CM. **DCMS** license is necessary for redfish advanced features.
- Multiple ".csv" files are allowed, each ".csv" will be treated as one cluster/rack, and file name will be used as cluster/rack name.
- Configure default bridge can help avoid IP confliction.

- L12CM can be deployed before or after UDP clients starts. UDP clients will obtained its port number once the L12CM UDP mongo database has been initialized.
- "Power Control" is power consumption.
- Please do not include any pure number data in csv file, the pandas will read it as float or int.

Backup data

- L12CM can automatically backup sensor readings and system information inside a created directory. The directory name should be looks like "[cluster1_MongoDB_2021-01-04-18-05-43]".
- After cleaning all the containers, the folder will not be removed.
- If you want to access the data, simply run a mongo DB mount with this folder:
- `docker run -d -p 20001:20001 -v ~/cluster1/cluster1_MongoDB_2021-01-04-18-05-43:/data/db --name cluster1backup mongo mongod --port 20001`
- If you want to visualize the data using frontend:
    - Copy input files into the archive folder: *.csv, flag*.txt, node_names*.txt and pwd*.txt.
    - Modify the *.csv name to N1-*.csv, and **MongoDB** to N1-**MongoDB**. (N1-*,N2-*,N3-*, etc)
    - Run generateDockercomposeForArchive.py.
- `python3 generateDockercomposeForArchive.py <PORT_NUMBER> <UNIQUE_ID>`
- `For example:`
- `python3 generateDockercomposeForArchive.py 20000 project_1`
    - Default port number is 20000.
    - Run startArchive.sh to start the containers.
- `bash startArchive.sh`
    - **IMPORTANT** If mongodb exit with error, try a certain mongo version such as: 4.4.14, 5.0.9, 6.0.3 or 6.0.4.
    - **IMPORTANT** Advanced Features are not working because there is no connection.

Redployment steps

Redeploy a new L12CM without completely turning off the previous one is possible by following the steps as below:

- Stop and remove UDP config mongo (port number 8888).
- Stop and remove the autoheal container.

- Change PORTNUM and MACNAME.
- Deploy a new L12CM.

Check sysmonitorsensor

Sometimes the sysmonitorsensor could be stoped or hanged due to ipmi ip not reachable. After fixing the network issue, you need to restart the sysmonitorsensor by:

```
docker start [sysmonitorsensor ID]
```

Session conflictions

To avoid potential conflictions, some sessions (pages) can only be opened by one user.

- SUM Pages:
  - sumtoolboxterminal.html
  - sumtoolboxupload.html
- Redfish Pages:
  - bioscomparisonoutput.html
  - biosupdate.html
  - biosupdaterack.html
  - bmcupdaterack.html
- Telemetry Page:
  - system_telemetry.html
- UDP Pages:
  - UDP_commandline.html
  - udpserverupload.html

UDP Server/Client Tips

Important

- The MAC address must be correctly set in the *.csv file to make UDP function working.
- Firewall has to be disabled for both UDP server and UDP clients.
- Need to set the UDP server IP address for every client.
- If UDP server deployed before UDP client started, the initialized will be done automatically.
- If UDP client deployed before UDP server, initialized is needed.

- If UDP server restarted with nodes mapping changed (csv file changed), you might need to restart UDP client to make it work.
- benchmark resutls are stored in UDP collecion of Redfish database.
- This function is still under development and need additional debugging.

Manually insert benchmark results and OS level hardware

Follow the steps below to insert benchmark results:

- Create a directory named "benchmark_logs" inside the same folder of input files.
- Put all the benchmark logs in the "benchmark_logs" folder: each node should have one log folder, the name should be similar as "benchmark_logs_ac-1f-6b-99-05-10".
- The sub log folder name should be ended with NIC MAC address.
- Create a directory named "benchmark_configs" inside the same folder of input files.
- Create corresponding benchmark config files. One example is shown as below:

```
{
"category":"benchmark",
"exe":"xhpl",
"prefix":"mpirun -n 32 --allow-run-as-root",
"config":"~/client/HPL.dat",
"log":"HPL",
"walltime":3600,
"parseResultLog":1,
"selfLog":0,
"selfLogPath":"",
"numOfResults":1,
"keywords":[["T/V","Time","Gflops"]],
"addRow":[2],
"dfs":["None"],
"index":[-1],
"unit":["Gflops"],
"criteriaType":["numericType"],
"criteria":[[20,100]]
}
```

- Open "udpoutput" page and click "LOAD BENCHMARK DATA" button to insert the results.

Follow the steps below to insert hardware data:

- Create a directory named "hw_data" inside the same folder of input files.
- Put all the hardware dat in the "hw_data" folder: each node should have one log folder, the name should be similar as "hw_info_ac-1f-6b-99-05-02".
- The sub log folder name should be ended with NIC MAC address.
- Create a hw.conf file inside "hw_data" folder. One example is shown as below:

```
OS=centos   # Possible options: ubuntu, centos, rocky linux, fedora
GPU=AMD # Possible options: nvidia, amd
```

- (Optional) put SN files inside "hw_data" folder.

Supported OS:

- CentOS 7 & CentOS 8.
- CentOS Stream 8 & CentOS Stream 9
- Rocky Linux 8.5
- Ubuntu 20.04
- Ubuntu 22.04
- Fedora 35

Hardware tools usage

Usage: ./system_info_server.sh [pwd0.txt] [remote_username] [remote_password] [mac/sn] self_IP Options are: pwd0.txt example: 172.27.28.51,172.27.28.11,AC1F6B990512,0,ADMIN remote_username username of clients, assume all usernames are same remote_password password of clients, assuem all passwords are same mac/sn if mac, split it with '-', if sn do not change self_IP server IP address-pools

Benchmark tools usage

Automatically generate input files from IPMI authentication csv files, using Redfish API and SUM tool

- csv format: [IPMI_IP, password] or [IPMI_IP, username, password]

```
Usage: generate_input_files.py -i INPUTFOLDER [options]
  required:
    -i, --input         input folder with ipmi authentication csv files
  options:
    -s, --sumpath       path to SUM executable
```

Testing before the pull request

- Clone the latest code from gitea:

```
git clone http://172.27.21.102:3000/ChenyangL/LinuxClusterMonitor.git
```

- Inplace your changes to the latest code.
- Build the new images that contains your changes:

```
docker build -t neonadia/$IMAGE_NAME .
```

- Redeploy the L12CM to test the changes: **Important** the docker_clean.sh will stop and remove all the running containers.

```
./docker_clean.sh # clean all the running containers
./docker_deploy.sh $FOLDER_NAME # start the new L12CM
```

L12CM playground

- **172.27.28.15** L12CM development server 1
- **172.27.28.17** L12CM development server 2
- **172.31.32.198** L12CM development server and BLOG server on engineering network
- **10.33.10.36** L12CM server for Facebook Cluster monitoring.
- **10.33.10.36:20000** L12CM archive for Facebook project.

Possible errors and issues

Error: UDP server

```
Created new file: /app/RACK/alltestservers1-host.json
Exception in thread Thread-1:
Traceback (most recent call last):
  File "/usr/local/lib/python3.7/threading.py", line 926, in _bootstrap_inner
    self.run()
  File "udpserver-1.5.py", line 137, in run
    self.write_log('msg', data[1])
  File "udpserver-1.5.py", line 42, in write_log
    if load_json[self.mac]['ip'] == self.ip:
TypeError: 'NoneType' object is not subscriptable
```

Restart udpserver container will resolve this issue.

Issue: firewall

```
Failed to Setup IP tables: Unable to enable SKIP DNAT rule:
(iptables failed: iptables --wait -t nat -I DOCKER -i br-79c12f098a37 -j RETURN:
iptables: No chain/target/match by that name. (exit status 1))
```

Start the firewall will resolve the issue, after successfully deployed you can stop the firewall again.

Issue: IKVM

```
no response from RAKP 1 message
```

It happens sometime when redfish cannot get the IKVM address. This problem can be automatically resolved when L12CM running the next round data query. So, it is not critical.

Error: 500

```
Seems Boot Up Not Completed Yet (Error 500)
```

This error could be due to one of the following reasons:

- L12CM does not boot up completely.
- Wrong Redfish API. To resolve this error, you need to check the log information of the container.

Error: Redfish login failed

This error could be due to one of the following reasons:

- IPMI did not activated.
- Request confliction (Not critical). To activate IPMI, one can use sum to activate it, check "DCMS Single Key Activate Guide" for details. For example:

```
./sum -i 172.27.28.51 -u ADMIN -p ADMIN -c ActivateProductKey –key
xxxxxxxxxxxxxxxxxxxx
```

Error: Firmware update

```
Firmware updating cannot started, error code: 503
```

This error is due to something block the firmware update, please make sure the following pages have been closed:

- Correspond IPMI page.

- Correspond IKVM page.
- Correspond remote console. You need to cancel the update using redfish to resolve this issue.

Error: PDF report spacer

```
reportlab.platypus.doctemplate.LayoutError: Flowable <Spacer at 0xf01440
frame=normal>...(1 x 14.4) too large on page....
```

This error is due to the avalible height less than the spacer height, it is a known issue of reportlab library. Using the following code could resolve this issue:

```
class ConditionalSpacer(Spacer):

  def wrap(self, availWidth, availHeight):
      height = min(self.height, availHeight-1e-8)
      return (availWidth, height)
```

The code is referenced from: https://stackoverflow.com/questions/1842266/reportlab-layouterror-too-large-on-page

Known issue 1: SUMTOOLBOX and SMCIPMITOOL

- Manully stopped the sysmonitorsensor when sysmonitorsensor is fetching date will make the SUMTOOL BOX hanged forever.
- Since SMCIPMITOOL is conflicted with SUMTOOL BOX, they have been placed into different containers.
- Restart sysmonitorfrontend is a workaround.

Known issue 2: sysmonitorsensor can not use autobuild

using docker autobuild will result in SMCIPMITOOL not working. The script somehow can not make SMCIPMITOOL excutable.

Known issue 3: duplicate hardware folders could result in duplicate entries

For example, by creating the following two sub folders in the "hw_data" folder. Although they are the same mac addresses, but two entries will be created in the database.

```
hw_info_ac1f6b990512
hw_info_ac-1f-6b-99-05-12
```

User should always avoid additional folders/files inside the hw_data folder.

Known issue 4: DEBUG_MODE=True could result in frontend index page becomes slower overtime.

For production, set DEBUG_MODE=False.

Known issue 5: OS IP change could result in SN not showing in PDF report

Walk around: Reload the benchmark results could sync the OS IP to latest.

Known issue 6: Backend failed due to Decode Error

Caused by IPMI/REDFISH bug, need to run "ipmitool mc reset cold" and redeploy the L12CM to resolve this issue.

```
(172.27.28.51) Fetching Chassis Info......(172.27.28.53) Fetching Thermal Info......

Unhandled exception in thread started by <function server at 0x7fc2ac11b680>
Traceback (most recent call last):
  File "redfish_sys_docker_v1.py", line 93, in server
    chassis_info(redfish_obj, bmc_ip,
chassis_response.dict['Members'][i]['@odata.id'], sys_data['Chassis'][i + 1])
  File "redfish_sys_docker_v1.py", line 154, in chassis_info
    if chassis in chassis_response.dict.keys():
  File "/usr/local/lib/python3.7/site-packages/redfish/rest/v1.py", line 240, in dict
    return json.loads(self.text)
  File "/usr/local/lib/python3.7/json/__init__.py", line 348, in loads
    return _default_decoder.decode(s)
  File "/usr/local/lib/python3.7/json/decoder.py", line 337, in decode
    obj, end = self.raw_decode(s, idx=_w(s, 0).end())
  File "/usr/local/lib/python3.7/json/decoder.py", line 353, in raw_decode
    obj, end = self.scan_once(s, idx)
json.decoder.JSONDecodeError: Invalid control character at: line 1 column 285 (char
284)
```

Known issue 7: UDP Server Socket issue

```
Exception in thread Thread-1:
Traceback (most recent call last):
  File "/usr/local/lib/python3.7/threading.py", line 926, in _bootstrap_inner
    self.run()
  File "udpserver-1.5.py", line 136, in run
    self.ip, self.mac = self.get_ip_mac(header[1])
IndexError: list index out of range
```

This issue is found when running UDP initialization under Engineering Network. It seems doesn't affect any functions. No sultion yet.

SQL Databases

Located on 172.27.28.17 MSSQL Database (A small copy of unique password database)

- Check running status

```
systemctl status mssql-server
```

- Login command

```
sqlcmd -S localhost -U sa -P 'Iciy0220'
```

- Check all databases

```
select name from sys.databases;
go
```

- Check all tables

```
select * from information_schema.tables;
go
```

MYSQL Database

- Login command

```
mysql -uroot -ptest
```

Supported Browsers

- MICROSFOT EDGE (Most Recommand)
- GOOGLE CHROME
- Safari

Update local to the latest

Retrive latest changes from remote repository

```
git fetch
```

Merge the changes

```
git merge origin master
```