# Requirement

Before excute ansible, the following files need to be set appropriate path.

* cuda 10.2 deb file and patch deb files to /ansible/roles/cuda/files.
* gromacs tar gz file from <https://manual.gromacs.org/documentation/> to each gromacs files directory.
* slurm tar bz2 file from <https://download.schedmd.com/slurm/slurm-21.08.1.tar.bz2> to /ansible/roles/slurm/files.
* home directory of ansible shoud be /srv/ansible

# Version info

ansible : 2.13.7

## /srv/ansible/admin.yml の内容

# /srv/ansible/admin.yml

- hosts: localhost

connection: local

become: yes

roles:

- role: apt

- role: admin-ubuntu

- role: nfs

- role: nis-server

- role: gfortran

- role: gromacs-2022.4

- role: slurm

services:

- slurmctld.service

- role: admin-alphafold

## /srv/ansible/nodes.yml の内容

# /srv/ansible/nodes.yml

- hosts: nodes

become: yes

roles:

- role: child-ubuntu

- role: apt

- role: nis

- role: nfs

- role: gromacs-2020.3

- role: gromacs-2022.4-child

- role: tensorflow

- role: lm-sensors

- role: slurm

services:

- slurmd.service

- role: child-alphafold

## /srv/ansible/production の内容

# /srv/ansible/production

[nodes]

golgi01

golgi02

golgi03

golgi04

golgi05

golgi06

golgi07

golgi08

golgi09

golgi10

golgi11

golgi12

golgi13

[nodes:vars]

ansible\_ssh\_user=ansible

ansible\_ssh\_pass=ansible

ansible\_python\_interpreter=/usr/bin/python3

## 以下は、./roles(/srv/ansible/roles/)ディレクトリ以下にある、YAML形式(.yml)のテキストファイルの内容をそのテキストファイルがある場所と共にすべて書き下したもの。(コマンド”””for FILE in $(find ./roles -type f -name '\*.yml'); do echo "==== $FILE ===="; cat "$FILE"; echo; done”””で出力。)

==== ./roles\_before\_202502/admin-alphafold/tasks/main.yml ====

- name: create directory for alphafold

file:

path: /opt/alphafold

state: directory

owner: alphafold

- name: mount /dev/sdb1 to /opt/alphafold

mount:

path: "/opt/alphafold"

src: "/dev/sdb1"

fstype: ext4

state: mounted

- name: install nfs-kernel-server

apt:

name: nfs-kernel-server

state: present

update\_cache: yes

- name: set export directory

lineinfile:

dest: /etc/exports

line: '/opt/alphafold 192.168.2.0/255.255.255.0(rw,sync,no\_subtree\_check,no\_root\_squash)'

insertbefore: EOF

state: present

==== ./roles\_before\_202502/admin-ubuntu/tasks/main.yml ====

- name: set timezone to Asia/Tokyo

timezone:

name: Asia/Tokyo

- name: disable suspend mode

file:

src: /dev/null

dest: "{{ item }}"

state: link

with\_items:

- /etc/systemd/system/sleep.target

- /etc/systemd/system/suspend.target

- /etc/systemd/system/hibernate.target

- /etc/systemd/system/hybrid-sleep.target

- name: configure /etc/hosts

blockinfile:

path: /etc/hosts

block: |

192.168.2.200 GolgiAdmin GolgiAdmin.golgi

192.168.2.201 GolgiFS

192.168.2.1 Golgi01 golgi01

192.168.2.2 Golgi02 golgi02

192.168.2.3 Golgi03 golgi03

192.168.2.4 Golgi04 golgi04

192.168.2.5 Golgi05 golgi05

192.168.2.6 Golgi06 golgi06

192.168.2.7 Golgi07 golgi07

192.168.2.8 Golgi08 golgi08

192.168.2.9 Golgi09 golgi09

192.168.2.10 Golgi10 golgi10

192.168.2.11 Golgi11 golgi11

192.168.2.12 Golgi12 golgi12

192.168.2.13 Golgi13 golgi13

192.168.2.14 Golgi14 golgi14

192.168.2.15 Golgi15 golgi15

- name: copy script for admin

copy:

src: "{{ item }}"

dest: "/root/sbin/{{ item }}"

owner: root

group: root

with\_items:

- do\_all

- shutdown\_all

- name: IP forwarding setup

replace:

path: /etc/sysctl.conf

regexp: "^#net.ipv4.ip\_forward=1"

replace: "net.ipv4.ip\_forward=1"

- name: set iptables.rules

copy:

dest: /etc/iptables.rules

src: iptables.rules

- name: enable to restore iptables setting durint start-up

copy:

dest: /etc/network/if-pre-up.d/iptables-restore

src: iptables-restore

mode: '755'

- name: install ifupdown for iptables setting during start-up

apt:

name: ifupdown

state: present

- name: install zsh

apt:

name: zsh

state: present

- name: install packages for golgi temperature

apt:

name: ["parallel", "moreutils"]

state: present

==== ./roles\_before\_202502/apt/tasks/main.yml ====

- name: change archive servers to those in Japan

replace:

dest: /etc/apt/sources.list

regexp: 'deb https?://[^(security)]\S\* (.\*)'

replace: 'deb http://jp.archive.ubuntu.com/ubuntu \1'

- command: grep -e "\<universe\>" /etc/apt/sources.list

register: check\_universe\_repo

check\_mode: no

ignore\_errors: yes

changed\_when: no

- name: add the universe repository

command: add-apt-repository universe && apt-get update

when: check\_universe\_repo.rc != 0

- name: disable unattended upgrades

lineinfile:

path: "/etc/apt/apt.conf.d/20auto-upgrades"

regexp: "{{ item.regexp }}"

line: "{{ item.line}}"

with\_items:

- { regexp: 'Update-Package-Lists', line: 'APT::Periodic::Update-Package-Lists "0";' }

- { regexp: 'Unattended-Upgrade', line: 'APT::Periodic::Unattended-Upgrade "0";' }

==== ./roles\_before\_202502/child-alphafold/defaults/main.yml ====

install\_prefix: /opt/alphafold

ubuntu\_version: Ubuntu20.04

gpgkey\_path: gpgkey.gpg

ncr\_list: nvidia-container-runtime.list

==== ./roles\_before\_202502/child-alphafold/tasks/main.yml ====

- name: create directory for alphafold

file:

path: "{{ install\_prefix }}"

state: directory

- name: mount admin /opt/alphafold

mount:

path: "{{ install\_prefix }}"

src: "GolgiAdmin:/opt/alphafold"

fstype: nfs

state: mounted

- name: install docker for alphafold

apt:

pkg:

- docker-ce

- docker-ce-cli

update\_cache: yes

state: latest

- name: pull docker image for nvidia/cuda:11.4-base

command:

cmd: docker pull nvidia/cuda:11.4.0-base

- name: copy gpgkey to tmp

copy:

dest: "/tmp/{{ gpgkey\_path }}"

src: "{{ gpgkey\_path }}"

- name: add gpg key for nvidia-docker

command:

cmd: "sudo apt-key add /tmp/{{ gpgkey\_path }}"

- name: copy nvidia-container-runtime list to /etc/apt/sources.list.d/

copy:

dest: "/etc/apt/sources.list.d/nvidia-container-runtime.list"

src: "{{ ncr\_list }}"

- name: update apt

apt:

update\_cache: yes

- name: restart docker

service:

name: docker

state: restarted

- name: confirm or build nvidia/cuda:11.4.0-base

command:

cmd: docker run --rm --gpus all nvidia/cuda:11.4.0-base nvidia-smi

- name: build docker

command:

cmd: "docker build -f {{ install\_prefix }}/alphafold/docker/Dockerfile -t alphafold {{ install\_prefix }}/alphafold"

- name: install python3-pip

apt:

name: python3-pip

state: present

- name: install docker requirements by pip

command:

cmd: "python3 -m pip install -r {{ install\_prefix }}/alphafold/docker/requirements.txt"

==== ./roles\_before\_202502/child-ubuntu/tasks/main.yml ====

- name: set timezone to Asia/Tokyo

timezone:

name: Asia/Tokyo

- name: configure /etc/hosts

blockinfile:

path: /etc/hosts

block: |

192.168.2.200 GolgiAdmin GolgiAdmin.golgi

192.168.2.201 GolgiFS

- name: link python to python3

file:

src: /usr/bin/python3

dest: /usr/bin/python

state: link

- name: disable suspend mode

command: "sudo systemctl mask sleep.target suspend.target. hibernate.target hybrid-sleep.target"

- name: install zsh

apt:

name: zsh

state: present

==== ./roles\_before\_202502/cuda-10.2/defaults/main.yml ====

deb\_version: ubuntu1804-10-2-local-10.2.89-440.33.01\_1.0-1

apt\_key\_version: 10-2-local-10.2.89-440.33.01

patch\_prefix: ubuntu1804-10-2-local\_10.2

==== ./roles\_before\_202502/cuda-10.2/handlers/main.yml ====

- name: reboot the machine

reboot:

==== ./roles\_before\_202502/cuda-10.2/tasks/main.yml ====

- name: copy a .deb files

copy:

dest: /tmp/cuda.deb

src: "cuda-repo-{{ deb\_version }}\_amd64.deb"

- name: install a .deb package

apt:

deb: "/tmp/cuda.deb"

- name: add an apt key

apt\_key:

file: "/var/cuda-repo-{{ apt\_key\_version }}/7fa2af80.pub"

- name: install cuda

apt:

name: cuda

state: present

update\_cache: yes

notify: reboot the machine

- name: copy patches .deb files

copy:

dest: "/tmp/{{ item }}"

src: "{{ item }}"

with\_items:

- "cuda-repo-{{ patch\_prefix }}.1-1\_amd64.deb"

- "cuda-repo-{{ patch\_prefix }}.2-1\_amd64.deb"

- name: install patches .deb packages

apt:

deb: "/tmp/{{ item }}"

force: yes

with\_items:

- "cuda-repo-{{ patch\_prefix }}.1-1\_amd64.deb"

- "cuda-repo-{{ patch\_prefix }}.2-1\_amd64.deb"

- name: upgrade cuda

apt:

name: cuda

state: latest

update\_cache: yes

notify: reboot the machine

- meta: flush\_handlers

==== ./roles\_before\_202502/cuda-11.4/defaults/main.yml ====

deb\_version: ubuntu2004-11-4-local\_11.4.0-470.42.01-1

cuda\_version: ubuntu2004-11-4-local

==== ./roles\_before\_202502/cuda-11.4/handlers/main.yml ====

- name: reboot the machine

reboot:

==== ./roles\_before\_202502/cuda-11.4/tasks/main.yml ====

- name: copy a .deb files

copy:

dest: /tmp/cuda.deb

src: "cuda-repo-{{ deb\_version }}\_amd64.deb"

- name: install a .deb package

apt:

deb: "/tmp/cuda.deb"

- name: add an apt key

apt\_key:

file: "/var/cuda-repo-{{ cuda\_version }}/7fa2af80.pub"

- name: install cuda

apt:

name: cuda-11-4

state: present

update\_cache: yes

notify: reboot the machine

#- name: upgrade cuda

# apt:

# name: cuda

# state: latest

# update\_cache: yes

# notify: reboot the machine

- meta: flush\_handlers

==== ./roles\_before\_202502/cuda-11.4-admin/defaults/main.yml ====

deb\_version: ubuntu2004-11-4-local\_11.4.0-470.42.01-1

cuda\_version: ubuntu2004-11-4-local

==== ./roles\_before\_202502/cuda-11.4-admin/handlers/main.yml ====

- name: reboot the machine

reboot:

==== ./roles\_before\_202502/cuda-11.4-admin/tasks/main.yml ====

- name: copy a .deb files

copy:

dest: /tmp/cuda.deb

src: "cuda-repo-{{ deb\_version }}\_amd64.deb"

- name: install a .deb package

apt:

deb: "/tmp/cuda.deb"

- name: add an apt key

apt\_key:

file: "/var/cuda-repo-{{ cuda\_version }}/7fa2af80.pub"

- name: install cuda

apt:

name: cuda

state: present

update\_cache: yes

notify: reboot the machine

- name: upgrade cuda

apt:

name: cuda

state: latest

update\_cache: yes

notify: reboot the machine

- meta: flush\_handlers

==== ./roles\_before\_202502/cuda-12.0/defaults/main.yml ====

deb\_version: ubuntu2204-12-0-local\_12.0.0-525.60.13-1

cuda\_version: ubuntu2204-12-0-local

key\_version: 825BBB4F

==== ./roles\_before\_202502/cuda-12.0/handlers/main.yml ====

- name: reboot the machine

reboot:

==== ./roles\_before\_202502/cuda-12.0/tasks/main.yml ====

- name: copy a .deb files

copy:

dest: /tmp/cuda.deb

src: "cuda-repo-{{ deb\_version }}\_amd64.deb"

- name: install a .deb package

apt:

deb: "/tmp/cuda.deb"

- name: copy the keyring file to the /usr/share/keyring directory

copy:

src: "{{ item }}"

dest: "/usr/share/keyrings/"

with\_fileglob:

- "/var/cuda-repo-{{ deb\_version }}-local/cuda-\*-keyring.gpg"

- name: install cuda

apt:

name: cuda

state: present

update\_cache: yes

notify: reboot the machine

- name: upgrade cuda

apt:

name: cuda

state: latest

update\_cache: yes

notify: reboot the machine

- meta: flush\_handlers

==== ./roles\_before\_202502/cuda-12.0-child/defaults/main.yml ====

deb\_version: ubuntu2004-12-0-local\_12.0.0-525.60.13-1

cuda\_version: ubuntu2004-12-0-local

key\_version: 5E22DB91

==== ./roles\_before\_202502/cuda-12.0-child/handlers/main.yml ====

- name: reboot the machine

reboot:

==== ./roles\_before\_202502/cuda-12.0-child/tasks/main.yml ====

- name: copy a .deb files

copy:

dest: /tmp/cuda.deb

src: "cuda-repo-{{ deb\_version }}\_amd64.deb"

- name: install a .deb package

apt:

deb: "/tmp/cuda.deb"

- name: copy the kering file to the /usr/share/keyring directory

copy:

src: "/var/cuda-repo-{{ cuda\_version }}/cuda-{{ key\_version }}-keyring.gpg"

dest: "/usr/share/keyrings/cuda-{{ key\_version }}-keyring.gpg"

remote\_src: yes

- name: install cuda

apt:

name: cuda

state: present

update\_cache: yes

notify: reboot the machine

- name: upgrade cuda

apt:

name: cuda

state: latest

update\_cache: yes

notify: reboot the machine

- meta: flush\_handlers

==== ./roles\_before\_202502/gfortran/tasks/main.yml ====

- name: install gfortran

apt:

name: gfortran

state: present

update\_cache: yes

==== ./roles\_before\_202502/gromacs/defaults/main.yml ====

version: 5.1.1

source\_directory: /opt/src

install\_prefix: /opt/gromacs

==== ./roles\_before\_202502/gromacs/meta/main.yml ====

---

dependencies:

- cuda

==== ./roles\_before\_202502/gromacs/tasks/main.yml ====

- name: install fftw and cmake

apt:

name: ["libfftw3-dev", "cmake"]

state: present

- stat:

path: "{{ install\_prefix }}-{{ version }}/bin/gmx"

register: gmx\_result

- name: ensure a source directory exists

file:

path: "{{ source\_directory }}"

state: directory

- name: unarchive source files

unarchive:

src: "gromacs-{{ version }}.tar.gz"

dest: "{{ source\_directory }}"

when: not gmx\_result.stat.exists

- name: fix cmake

lineinfile:

path: "{{ source\_directory }}/gromacs-{{ version }}/cmake/gmxManageNvccConfig.cmake"

regexp: '(.\*arch=compute\_20,code=sm\_20.\*)'

state: absent

when: not gmx\_result.stat.exists

- name: create builddir

tempfile:

state: directory

suffix: gromacs-build

register: builddir

when: not gmx\_result.stat.exists

- name: configure

command: "cmake {{ source\_directory }}/gromacs-{{ version }} -DGMX\_SIMD=AVX2\_256 -DGMX\_GPU=ON -DCUDA\_TOOLKIT\_ROOT\_DIR=/usr/local/cuda -DREGRESSIONTEST\_DOWNLOAD=ON -DCMAKE\_INSTALL\_PREFIX={{ install\_prefix }}-{{ version }}"

args:

chdir: "{{ builddir.path }}"

when: not gmx\_result.stat.exists

- name: compile, test and install

make:

target: "{{ item }}"

chdir: "{{ builddir.path }}"

with\_items:

- all

- check

- install

when: not gmx\_result.stat.exists

- name: clear builddir

file:

path: "{{ builddir.path }}"

state: absent

when: not gmx\_result.stat.exists

==== ./roles\_before\_202502/gromacs-2019.4/defaults/main.yml ====

version: 2019.4

source\_directory: /opt/src

install\_prefix: /opt/gromacs

==== ./roles\_before\_202502/gromacs-2019.4/meta/main.yml ====

---

dependencies:

- cuda-11.7

==== ./roles\_before\_202502/gromacs-2019.4/tasks/main.yml ====

- name: install fftw and cmake

apt:

name: ["libfftw3-dev", "cmake"]

state: present

- stat:

path: "{{ install\_prefix }}-{{ version }}/bin/gmx"

register: gmx\_result

- name: ensure a source directory exists

file:

path: "{{ source\_directory }}"

state: directory

- name: unarchive source files

unarchive:

src: "gromacs-{{ version }}.tar.gz"

dest: "{{ source\_directory }}"

when: not gmx\_result.stat.exists

- name: fix cmake

lineinfile:

path: "{{ source\_directory }}/gromacs-{{ version }}/cmake/gmxManageNvccConfig.cmake"

regexp: "{{ item }}"

state: absent

when: not gmx\_result.stat.exists

with\_items:

- '(.\*arch=compute\_20,code=sm\_20.\*)'

- '(.\*arch=compute\_30,code=sm\_30.\*)'

- name: create builddir

tempfile:

state: directory

suffix: gromacs-build

register: builddir

when: not gmx\_result.stat.exists

- name: unarchive regressiontest files

unarchive:

src: "regressiontests-{{ version }}.tar.gz"

dest: "{{ source\_directory }}"

when: not gmx\_result.stat.exists

- name: configure

command: "cmake {{ source\_directory }}/gromacs-{{ version }} -DGMX\_SIMD=AVX2\_256 -DGMX\_GPU=ON -DCUDA\_TOOLKIT\_ROOT\_DIR=/usr/local/cuda -DREGRESSIONTTEST\_PATH={{ source\_directory}}/regressiontests-{{ version }} -DCMAKE\_INSTALL\_PREFIX={{ install\_prefix }}-{{ version }}"

args:

chdir: "{{ builddir.path }}"

when: not gmx\_result.stat.exists

- name: compile, test and install

make:

target: "{{ item }}"

chdir: "{{ builddir.path }}"

with\_items:

- all

- check

- install

when: not gmx\_result.stat.exists

- name: clear builddir

file:

path: "{{ builddir.path }}"

state: absent

when: not gmx\_result.stat.exists

- name: make symbolic link

file:

src: "{{ install\_prefix }}-{{ version }}"

dest: "{{ install\_prefix }}"

state: link

==== ./roles\_before\_202502/gromacs-2020.3/defaults/main.yml ====

version: 2020.3

source\_directory: /opt/src

install\_prefix: /opt/gromacs

==== ./roles\_before\_202502/gromacs-2020.3/meta/main.yml ====

---

dependencies:

- cuda-10.2

==== ./roles\_before\_202502/gromacs-2020.3/tasks/main.yml ====

- name: install fftw and cmake

apt:

name: ["libfftw3-dev", "cmake"]

state: present

- name: install gcc-8 and g++-8 to use cmake in CUDA10.2

apt:

name: ["gcc-8", "g++-8"]

state: present

- stat:

path: "{{ install\_prefix }}-{{ version }}/bin/gmx"

register: gmx\_result

- name: ensure a source directory exists

file:

path: "{{ source\_directory }}"

state: directory

- name: unarchive source files

unarchive:

src: "gromacs-{{ version }}.tar.gz"

dest: "{{ source\_directory }}"

when: not gmx\_result.stat.exists

- name: fix cmake

lineinfile:

path: "{{ source\_directory }}/gromacs-{{ version }}/cmake/gmxManageNvccConfig.cmake"

regexp: "{{ item }}"

state: absent

when: not gmx\_result.stat.exists

with\_items:

- '(.\*arch=compute\_20,code=sm\_20.\*)'

- '(.\*arch=compute\_30,code=sm\_30.\*)'

- name: create builddir

tempfile:

state: directory

suffix: gromacs-build

register: builddir

when: not gmx\_result.stat.exists

- name: unarchive regressiontest files

unarchive:

src: "regressiontests-{{ version }}.tar.gz"

dest: "{{ source\_directory }}"

when: not gmx\_result.stat.exists

- name: configure

command: "cmake {{ source\_directory }}/gromacs-{{ version }} -DCMAKE\_C\_COMPILER=gcc-8 -DCMAKE\_CXX\_COMPILER=g++-8 -DGMX\_SIMD=AVX2\_256 -DGMX\_GPU=ON -DCUDA\_TOOLKIT\_ROOT\_DIR=/usr/local/cuda -DREGRESSIONTTEST\_PATH={{ source\_directory}}/regressiontests-{{ version }} -DCMAKE\_INSTALL\_PREFIX={{ install\_prefix }}-{{ version }}"

args:

chdir: "{{ builddir.path }}"

when: not gmx\_result.stat.exists

- name: compile, test and install

make:

target: "{{ item }}"

chdir: "{{ builddir.path }}"

with\_items:

- all

- check

- install

when: not gmx\_result.stat.exists

- name: clear builddir

file:

path: "{{ builddir.path }}"

state: absent

when: not gmx\_result.stat.exists

- name: make symbolic link

file:

src: "{{ install\_prefix }}-{{ version }}"

dest: "{{ install\_prefix }}"

state: link

==== ./roles\_before\_202502/gromacs-2022.4/defaults/main.yml ====

version: 2022.4

source\_directory: /opt/src

install\_prefix: /opt/gromacs

==== ./roles\_before\_202502/gromacs-2022.4/meta/main.yml ====

---

dependencies:

- cuda-12.0

==== ./roles\_before\_202502/gromacs-2022.4/tasks/main.yml ====

- name: install fftw and cmake

apt:

name: ["libfftw3-dev", "cmake"]

state: present

- stat:

path: "{{ install\_prefix }}-{{ version }}/bin/gmx"

register: gmx\_result

- name: ensure a source directory exists

file:

path: "{{ source\_directory }}"

state: directory

- name: unarchive source files

unarchive:

src: "gromacs-{{ version }}.tar.gz"

dest: "{{ source\_directory }}"

when: not gmx\_result.stat.exists

- name: create builddir

tempfile:

state: directory

suffix: gromacs-build

register: builddir

when: not gmx\_result.stat.exists

- name: configure

command: "cmake {{ source\_directory }}/gromacs-{{ version }} -DGMX\_SIMD=AVX2\_256 -DGMX\_GPU=CUDA -DCUDA\_TOOLKIT\_ROOT\_DIR=/usr/local/cuda -DREGRESSIONTEST\_DOWNLOAD=ON -DCMAKE\_INSTALL\_PREFIX={{ install\_prefix }}-{{ version }}"

args:

chdir: "{{ builddir.path }}"

when: not gmx\_result.stat.exists

- name: compile, test and install

make:

target: "{{ item }}"

chdir: "{{ builddir.path }}"

with\_items:

- all

- check

- install

when: not gmx\_result.stat.exists

- name: clear builddir

file:

path: "{{ builddir.path }}"

state: absent

when: not gmx\_result.stat.exists

==== ./roles\_before\_202502/gromacs-2022.4-child/defaults/main.yml ====

version: 2022.4

source\_directory: /opt/src

install\_prefix: /opt/gromacs

==== ./roles\_before\_202502/gromacs-2022.4-child/meta/main.yml ====

---

dependencies:

- cuda-11.4

==== ./roles\_before\_202502/gromacs-2022.4-child/tasks/main.yml ====

- name: install fftw and cmake

apt:

name: ["libfftw3-dev", "cmake"]

state: present

- stat:

path: "{{ install\_prefix }}-{{ version }}/bin/gmx"

register: gmx\_result

- name: ensure a source directory exists

file:

path: "{{ source\_directory }}"

state: directory

- name: unarchive source files

unarchive:

src: "gromacs-{{ version }}.tar.gz"

dest: "{{ source\_directory }}"

when: not gmx\_result.stat.exists

- name: create builddir

tempfile:

state: directory

suffix: gromacs-build

register: builddir

when: not gmx\_result.stat.exists

- name: configure

command: "cmake {{ source\_directory }}/gromacs-{{ version }} -DGMX\_SIMD=AVX2\_256 -DGMX\_GPU=CUDA -DCUDA\_TOOLKIT\_ROOT\_DIR=/usr/local/cuda -DREGRESSIONTEST\_DOWNLOAD=ON -DCMAKE\_INSTALL\_PREFIX={{ install\_prefix }}-{{ version }}"

args:

chdir: "{{ builddir.path }}"

when: not gmx\_result.stat.exists

- name: compile, test and install

make:

target: "{{ item }}"

chdir: "{{ builddir.path }}"

with\_items:

- all

- check

- install

when: not gmx\_result.stat.exists

- name: clear builddir

file:

path: "{{ builddir.path }}"

state: absent

when: not gmx\_result.stat.exists

==== ./roles\_before\_202502/lm-sensors/tasks/main.yml ====

- name: install lm-sensors

apt:

name: lm-sensors

state: present

update\_cache: yes

==== ./roles\_before\_202502/munge/handlers/main.yml ====

- name: restart munge

systemd:

name: munge.service

state: restarted

==== ./roles\_before\_202502/munge/tasks/main.yml ====

- name: install MUNGE

apt:

name: ["munge", "libmunge-dev"]

state: present

notify: restart munge

- name: copy munge.key

copy:

dest: /etc/munge/munge.key

src: munge.key

notify: restart munge

==== ./roles\_before\_202502/nfs/tasks/main.yml ====

- name: install nfs

apt:

name: nfs-common

state: present

- name: mount /home

mount:

path: /home

src: "GolgiFS:/volume1/homes"

fstype: nfs

state: mounted

==== ./roles\_before\_202502/nis/defaults/main.yml ====

domain: "GolgiAdmin.golgi"

server: "GolgiAdmin.golgi"

==== ./roles\_before\_202502/nis/handlers/main.yml ====

- name: restart nis

systemd:

name: "{{ item }}"

state: restarted

loop:

- rpcbind

- nis

==== ./roles\_before\_202502/nis/tasks/main.yml ====

- name: install nis package

apt:

name: nis

state: present

update\_cache: yes

notify: restart nis

- name: set defaultdomain

copy:

dest: /etc/defaultdomain

content: "{{ domain }}"

notify: restart nis

- name: set a domain name

lineinfile:

path: /etc/yp.conf

line: "domain {{ domain }} server {{ server }}"

notify: restart nis

- meta: flush\_handlers

- name: configure nsswitch.conf

replace:

dest: /etc/nsswitch.conf

regexp: '^(passwd|group|shadow):((?!.\*nis).\*)$'

replace: '\1:\2 nis'

- name: modify logind-systemd configuration for ssh

replace:

path: /lib/systemd/system/systemd-logind.service

regexp: '^IPAddressDeny=any'

replace: '#IPAddressDeny=any'

- name: modify MINGID in yp Makefile

replace:

path: /var/yp/Makefile

regexp: '^MINGID=.\*'

replace: 'MINGID=999'

- name: reload logind-systemd config

systemd:

daemon\_reload: yes

==== ./roles\_before\_202502/nis-server/defaults/main.yml ====

domain: "GolgiAdmin.golgi"

==== ./roles\_before\_202502/nis-server/handlers/main.yml ====

- name: restart nis

systemd:

name: "{{ item }}"

state: restarted

loop:

- rpcbind

- nis

==== ./roles\_before\_202502/nis-server/tasks/main.yml ====

- name: install nis package

apt:

name: nis

state: present

update\_cache: yes

notify: restart nis

- name: set defaultdomain

copy:

dest: /etc/defaultdomain

content: "{{ domain }}"

notify: restart nis

- name: configure /etc/default/nis

replace:

path: /etc/default/nis

regexp: '^NISSERVER=false'

replace: 'NISSERVER=master'

- name: remove all IP accept setting for admin node

replace:

path: /etc/ypserv.securenets

regexp: '^0.0.0.0 0.0.0.0'

replace: '#0.0.0.0 0.0.0.0'

notify: restart nis

- name: add IP range restriction for admin node

lineinfile:

path: /etc/ypserv.securenets

insertafter: '^#0.0.0.0'

line: '255.255.255.0 192.168.2.0'

notify: restart nis

- meta: flush\_handlers

- name: install pip3 to install pexpect

apt:

name: python3-pip

state: present

update\_cache: yes

- name: set domainname

command: "domainname {{ domain }}"

- name: install pexpect for ypinit

pip:

name: pexpect

become: yes

- name: excute ypinit

expect:

command: /usr/lib/yp/ypinit -m

responses:

"^.\*next host to add:": "\x04"

"^Is this correct?.\*[y/n: y]": "y"

- name: modify MINGID in yp Makefile

lineinfile:

path: /var/yp/Makefile

regexp: '^MINGID='

line: 'MINGID=999'

==== ./roles\_before\_202502/slurm/defaults/main.yml ====

services: []

source\_directory: /opt/src

install\_prefix: /opt/slurm

slurm\_version: 22.05.7

user: slurm

spool\_dir: /var/spool/slurm.spool

state\_dir: /var/spool/slurm.state

accounting\_file: /var/spool/slurm.accounting

slurm\_logdir: /var/log/slurm

==== ./roles\_before\_202502/slurm/handlers/main.yml ====

- name: update library location

command: "ldconfig -n {{ install\_prefix }}/lib"

listen: update ldconfig

- name: restart SLURM services

systemd:

name: "{{ item }}"

state: restarted

daemon\_reload: yes

loop: "{{ services }}"

listen: restart services

==== ./roles\_before\_202502/slurm/meta/main.yml ====

---

dependencies:

- munge

==== ./roles\_before\_202502/slurm/tasks/main.yml ====

- name: install build-essential

apt:

name: build-essential

state: present

- name: make a source directory

file:

path: "{{ source\_directory }}"

state: directory

- name: unarchive source files

unarchive:

src: "slurm-{{ slurm\_version }}.tar.bz2"

dest: "{{ source\_directory }}"

- stat: path="{{ install\_prefix }}/sbin/slurmctld"

register: slurmctld

- name: configure SLURM

command: "./configure --prefix={{ install\_prefix }}"

args:

chdir: "{{ source\_directory }}/slurm-{{ slurm\_version }}"

register: configured

when: slurmctld.stat.executable is not defined

- name: "make {{ install\_prefix }}/etc"

file:

path: "{{ install\_prefix }}/etc"

state: directory

- name: create slurm.conf

template:

src: slurm2.conf.j2

dest: "{{ install\_prefix }}/etc/slurm.conf"

register: slurm\_conf

notify: restart services

- name: compile and install SLURM

make:

target: "{{ item }}"

chdir: "{{ source\_directory }}/slurm-{{ slurm\_version }}"

with\_items:

- all

- install

when: ((slurmctld.stat.executable is not defined) and (configured is success)) or

slurm\_conf.changed

- name: add a ldconf file for SLURM

copy:

dest: /etc/ld.so.conf.d/slurm.conf

content: "{{ install\_prefix }}/lib"

notify: update ldconfig

- name: make directories for SLURM

file:

path: "{{ item }}"

state: directory

recurse: yes

owner: "{{ user }}"

with\_items:

- "{{ state\_dir }}"

- "{{ spool\_dir }}"

- "{{ slurm\_logdir }}"

- stat:

path: "{{ accounting\_file }}"

register: acct\_result

- name: touch a file for accouting

file:

path: "{{ accounting\_file }}"

owner: "{{ user }}"

state: touch

when: not acct\_result.stat.exists

- name: copy gres.conf

copy:

src: gres.conf

dest: "{{ install\_prefix }}/etc"

- name: copy service files for SLURM

copy:

src: "{{ source\_directory }}/slurm-{{ slurm\_version }}/etc/{{ item }}"

dest: "/lib/systemd/system/{{ item }}"

remote\_src: yes

with\_items: "{{ services }}"

notify: restart services

- name: enable services

systemd:

name: "{{ item }}"

enabled: yes

daemon\_reload: yes

loop: "{{ services }}"

- name: set path setting script for slurm in admin

copy:

src: slurm-bin-path.sh

dest: "/etc/profile.d/slurm-bin-path.sh"

==== ./roles\_before\_202502/tensorflow/handlers/main.yml ====

- name: restart docker daemon

systemd:

name: docker

state: restarted

==== ./roles\_before\_202502/tensorflow/tasks/main.yml ====

- name: install some packages

apt:

name: ["apt-transport-https", "ca-certificates", "software-properties-common"]

state: present

update\_cache: yes

- name: add an apt key

apt\_key:

url: https://download.docker.com/linux/ubuntu/gpg

state: present

- name: add a docker stable repository

apt\_repository:

repo: "deb [arch=amd64] https://download.docker.com/linux/ubuntu {{ ansible\_distribution\_release }} stable"

state: present

- name: install docker

apt:

name: docker-ce

update\_cache: yes

- name: add an apt key for nvidia-docker

apt\_key:

url: https://nvidia.github.io/nvidia-docker/gpgkey

state: present

tags:

- nvidia-docker

- name: add apt repositories

get\_url:

url: "https://nvidia.github.io/nvidia-docker/ubuntu{{ ansible\_distribution\_version }}/nvidia-docker.list"

dest: /etc/apt/sources.list.d/nvidia-docker.list

tags:

- nvidia-docker

- name: install nvidia-docker

apt:

name: nvidia-docker2

update\_cache: yes

notify: restart docker daemon

tags:

- nvidia-docker