Руководство пользователя

раздела МВС-10П МПЗ А100

ВЕРСИЯ ОТ 07.02.2024

МОСКВА 2024

Оглавление

1. Конфигурация подсистемы A100

Подсистема суперкомпьютера МВС-10П МП3 A100 (далее – a100) состоит из вычислительных узлов со следующими характеристиками:

- 2 процессора Intel Xeon Platinum 8368Q (Ice Lake) 2.6 GHz, 38 ядер, 512 ГБ оперативной памяти;

- 2 ускорителя NVIDIA A100, 40 ГБ.

2. Обращение к подсистеме A100

Для пользования подсистемой `a100` необходимо подключиться к серверу доступа `mvs10q.jscc.ru` по протоколу `ssh` (`putty` – для ОС MS Windows). После соединения с сервером доступа необходимо загрузить модуль для работы с СУППЗ:

```
module load suppz
```

Далее при обращении к подсистеме `a100` необходимо указывать в командах СУППЗ ключ

```
-s a100
```

Например, просмотр очереди осуществляется по команде

```
mqinfo -s a100
```

3. Возможности, предоставляемые в подсистеме A100

В подсистеме `a100` установлена и функционирует система контейнеризации Singularity (https://apptainer.org). Singularity – это контейнерная платформа с открытым исходным кодом, предназначенная для использования в средах высокопроизводительных вычислений. Платформа Singularity характеризуется простотой, быстротой и безопасностью. Подробную документацию по платформе можно найти здесь: https://docs.sylabs.io/guides/3.5/user-guide/introduction.html

Для пользователя подсистема `a100` с платформой Singularity предоставляет следующие возможности.

1. Сборка и выполнение программ, разработанных на CUDA, с применением установленного в МСЦ РАН контейнера по умолчанию.

2. Сборка и выполнение CUDA-программ с применением контейнера пользователя.

3. Загрузка, сборка и выполнение пользовательских контейнеров.

Контейнеры Singularity, установленные в МСЦ РАН, расположены в каталоге `/common/runmvs/CUDA`. Контейнером по умолчанию является файл с именем `cuda_11.5.2-devel-ubi8.sif`, представляющий собой официальный образ CUDA версии 11.5.2. Этот образ включает:

- среду выполнения CUDA (cudart);
- математические библиотеки CUDA, NCCL и base;
- заголовки и инструменты разработки для создания программ и образов CUDA.

В файле `cuda_11.5.2-devel-ubi8.sif` используется дистрибутив ubi8 (Universal base image), созданный компанией RedHat, который может служить базовым образом (основой) для создания других образов. Отметим, что список доступных официальных образов представлен на https://hub.docker.com/r/nvidia/cuda/tags.

4. Сборка и выполнение CUDA-программ с применением контейнера по умолчанию

Порядок работы предусматривает три этапа:

1. Сборка CUDA-программы с применением контейнера `cuda_11.5.2-devel-ubi8.sif`.
2. Формирование сценария запуска.
3. Запуск CUDA-программы на выполнение.

1. Сборка CUDA-программы с применением контейнера `cuda_11.5.2-devel-ubi8.sif`

Сборка CUDA-программ осуществляется на сервере доступа `mvs10q.jscc.ru` в проектном каталоге пользователя. Пусть код CUDA-

программы размещен в файле mycuda.cu. Тогда сборка с применением контейнера по умолчанию осуществляется командой

```
singularity exec /common/runmvs/CUDA/cuda_11.5.2-devel-ubi8.sif nvcc –o mycuda mycuda.cu
```

Здесь ключ –o mycuda задает имя исполняемого файла CUDA-программы.

2. Формирование сценария запуска

Для запуска собранной программы пользователю необходимо в отдельном файле подготовить командный сценарий (скрипт) с командой запуска. Пусть имя файла сценария mycuda.sh. Тогда его содержимое в самом простом случае будет иметь следующий примерный вид:

```
#!/bin/bash
singularity run --nv /common/runmvs/CUDA/cuda_11.5.2-devel-ubi8.sif ./mycuda
```

Рассмотрим опцию выбора ускорителя. При помощи переменной окружения SINGULARITY_CUDA_VISIBLE_DEVICES=N можно выбирать определенные ускорители, где N – это номер ускорителя (0 или 1 в случае узла с двумя ускорителями). Например, чтобы использовать ускоритель с номером 1, в сценарии запуска mycuda.sh необходимо указать:

```
#!/bin/bash
SINGULARITY_CUDA_VISIBLE_DEVICES=1 singularity run --nv /common/runmvs/CUDA/cuda_11.5.2-devel-ubi8.sif ./mycuda
```

3. Запуск CUDA-программы на выполнение

Обращение к СУППЗ для запуска CUDA-программы осуществляется при помощи стандартной команды СУППЗ mpirun, например

```
mpirun –s a100 –np 1 –maxtime 100 mycuda.sh
```

Здесь через СУППЗ осуществляется обращение к подсистеме a100 (ключ -s a100) для запуска одного (ключ –np 1) экземпляра сценария mycuda.sh на время, не превышающее 100 минут (ключ –maxtime 100).

Команда mpirun направит сценарий в очередь подсистемы a100, сформировав задание с именем a100.mycuda.sh.1, где 1 – порядковый номер задания СУППЗ. Одновременно будет сформирован каталог задания с именем a100.mycuda.sh.1, где в файлах output и errors будет помещены стандартные потоки вывода и ошибок соответственно. Для сформированного задания доступны все команды СУППЗ, такие как просмотр очереди, завершение задания, получение статуса задания и др.

5. Сборка и выполнение CUDA-программ с применением контейнера пользователя

Вместо контейнера по умолчанию /common/runmvs/CUDA/cuda_11.5.2-devel-ubi8.sif можно применить любой контейнер Singularity, собранный пользователем, либо полученный им из доступных репозиториев. Для использования пользовательского контейнера его необходимо скопировать (загрузить) в проектный каталог пользователя на сервере доступа mvs10q.jscc.ru.

Важное замечание!!! Пусть у пользователя есть персональный компьютер (ноутбук) под управлением ОС Linux, и на этом компьютере установлены видеокарта Nvidia, среда CUDA и платформа Singularity. В этом случае пользователь может на этом компьютере собрать собственный образ CUDA в виде контейнера Singularity, разработать и отладить CUDA-программу, после чего на сервере доступа mvs10q.jscc.ru загрузить собственный образ CUDA и программу в проектный каталог и выполнить их в подсистеме a100.

Пусть контейнер-образ CUDA размещен в файле с именем mycontainer.sif. Тогда сборку CUDA-программы можно произвести командой

```
singularity exec mycontainer.sif nvcc –o mycuda mycuda.cu
```

Для запуска необходимо в отдельном файле подготовить командный сценарий (скрипт) с командой запуска. Пусть имя файла сценария mycuda.sh. Для запуска в

контейнере команды по умолчанию необходимо указать в сценарии mycuda.sh следующую команду:

```
#!/bin/bash
singularity run --nv mycontainer.sif
```

Для запуска собранной CUDA-программы с именем mycuda в пользовательском контейнере содержимое сценария mycuda.sh должно быть следующим:

```
#!/bin/bash
singularity run --nv mycontainer.sif ./mycuda
```

Так же, как и в случае запуска в контейнере по умолчанию, через переменную окружения SINGULARITY_CUDA_VISIBLE_DEVICES доступна опция выбора ускорителя.

Обращение к СУППЗ для запуска пользовательского контейнера и/или CUDA-программы осуществляется при помощи стандартной команды СУППЗ mpirun, например

```
mpirun –s a100 –np 1 –maxtime 100 mycuda.sh
```

Здесь через СУППЗ осуществляется обращение к подсистеме a100 (ключ –s a100) для запуска одного (ключ –np 1) экземпляра сценария mycuda.sh на время, не превышающее 100 минут (ключ –maxtime 100). Команда mpirun направит сценарий в очередь подсистемы a100, сформировав задание с именем a100.mycuda.sh.1, где 1 – номер задания СУППЗ. Одновременно будет сформирован каталог задания с именем a100.mycuda.sh.1, где в файлах output и errors будет помещены стандартные потоки вывода и ошибок соответственно. Для сформированного задания доступны все команды СУППЗ, такие как просмотр очереди, завершение задания, получение статуса задания и др.

6. Загрузка доступного образа CUDA в виде контейнера Singularity

Пользователь имеет возможность загрузить из доступных репозиториев необходимый ему образ CUDA в виде контейнера Singularity.

Например, для загрузки образа `cuda:11.5.2-devel-uib8` из репозитория DockerHub необходимо выполнить команду

```
singularity pull docker://nvidia/cuda:11.5.2-devel-uib8
```

Для загрузки образа `cuda11.4.3cudnn8` из репозитория Container Library необходимо выполнить команду

```
singularity pull library://ez82/dlc/cuda11.4.3cudnn8
```

7. Создание пользовательского образа CUDA в виде контейнера Singularity

Создание собственного образа возможно при помощи платформ Docker или Singularity. **Внимание!!!** Приведенные процедуры создания собственного образа **являются иллюстративными примерами**, выполнение которых возможно только на собственном компьютере (ноутбуке) пользователя, где он обладает правами администратора. Для освоения процедуры создания образа пользователю необходимо обратиться к документации платформ Docker и Singularity.

Рассмотрим пример сценария Dockerfile и команды для создания образа при помощи платформы Docker.

1. Создание нового образа из базового образа

```
FROM nvidia/cuda:10.1-cudnn7-runtime
```

2. Загрузка и установка необходимых пакетов

```
RUN apt update && \
    apt install --no-install-recommends -y build-
essential software-properties-common && \
    add-apt-repository -y ppa:deadsnakes/ppa && \
    apt install --no-install-recommends -y python3.8
python3-pip python3-setuptools python3-distutils && \
    apt clean && rm -rf /var/lib/apt/lists/*
```

3. Копирование файлов в образ

```
COPY req.txt /req.txt
COPY ./src /src
```

4. Загрузка и установка необходимых пакетов

```
RUN python3.8 -m pip install --upgrade pip && \
    python3.8 -m pip install --no-cache-dir -r /req.txt
```

5. Запуск приложения

```
CMD ['python3', '/src/app.py']
```

Чтобы собрать образ, необходимо воспользоваться командой:

```
sudo docker build <директория_с_Dockerfile> -t
<имя_образа>

sudo docker build . –t testuser:myapplication:1.0.0
```

Рассмотрим пример создания образа при помощи Singularity:

```
# Выбор агента начальной загрузки
Bootstrap: docker
# Создание нового образа из базового образа
From: nvidia/cuda:10.2-devel
# Загрузка и установка необходимых пакетов
%post
    apt-get update
    apt-get install -y git
    git clone https://github.com/NVIDIA/cuda-samples.git
/usr/local/cuda_samples
    cd /usr/local/cuda_samples
    git fetch origin --tags
    git checkout v10.2
    cd Samples/deviceQuery && make
# Запуск приложения
%runscript

/usr/local/cuda_samples/Samples/deviceQuery/deviceQuery

# singularity build <директория_для_сохранения_образа>
<директория_с_def_file>

# singularity build /home/singularity/test_image.sif
/home/singularity/test_image.def
```

8. Полезные ссылки

1. Официальные образы Nvidia CUDA на DockerHub https://hub.docker.com/r/nvidia/cuda#!
2. Образы из ресурса Container Library https://cloud.sylabs.io/library/search
3. Документация Docker https://docs.docker.com/
4. Документация SingularityCE https://docs.sylabs.io/guides/latest/user-guide/