Аппаратные ускорители

NPU

Alauda Build of NPU Operator

О Alauda Build of Hami

О Alauda Build of Hami

Heterogeneous AI Computing Virtualization Middleware (HAMi) — это универсальный чарт для управления гетерогенными AI-вычислительными устройствами в кластере k8s.

О плагине устройства NVIDIA GPU сборки Alauda

О плагине устройства NVIDIA GPU сборки Alauda

Плагин устройства NVIDIA для Kubernetes, позволяющий автоматически управлять GPU в кластере.

NPU

Alauda Build of NPU Operator

Alauda Build of NPU Operator

Введение

Kubernetes предоставляет доступ к специальным аппаратным ресурсам (таким как Ascend NPU) через Device Plugins. Однако для настройки и управления узлом с такими аппаратными ресурсами требуется несколько программных компонентов (например, драйверы, контейнерные рантаймы или другие библиотеки). Установка этих компонентов сложна, трудоемка и подвержена ошибкам. NPU operator использует Operator Framework в Kubernetes для автоматического управления всеми программными компонентами, необходимыми для настройки устройств Ascend. Эти компоненты включают драйвер и прошивку Ascend (которые поддерживают весь процесс работы кластера), а также плагин устройства MindCluster (который поддерживает операции кластера, такие как планирование заданий, мониторинг O&M и восстановление после сбоев). Установив соответствующие компоненты, вы сможете управлять ресурсами NPU, оптимизировать планирование рабочих нагрузок и контейнеризировать задачи обучения и инференса, чтобы AI-задачи могли развертываться и выполняться на устройствах NPU в виде контейнеров.

Для получения дополнительной информации обратитесь к NPU Operator↗.

Установка

Содержание

Предварительные требования

Онлайн установка

1. **Версия ACP: v4.0 или выше**
2. **Доступ администратора к вашему ACP кластеру**
3. **Убедитесь, что в узле NPU установлен инструмент bash. В противном случае скрипт установки драйвера и прошивки может не распарситься.**

4. **Требования к операционной системе рабочих узлов**

   - Рабочие узлы (или группы узлов), на которых выполняются **NPU нагрузки**, должны использовать одну из следующих операционных систем (архитектура Arm):

     - `openEuler 22.03 LTS`
     - `Ubuntu 22.04`

   - Рабочие узлы, на которых выполняются **только CPU нагрузки**, могут использовать любую операционную систему, так как оператор NPU не выполняет конфигурацию на узлах без NPU нагрузок.

5. **Поддерживаемое NPU оборудование**

   - Узлы должны использовать поддерживаемые NPU:

     - `Ascend 910B`
     - `Ascend 310P`

   - Для подробной совместимости ОС и оборудования смотрите MindCluster Documentation ↗

6. **Должен быть установлен кластерный плагин `Alauda Build of Node Feature Discovery`.**

Офлайн установка

1. **Для офлайн установки требуются все предварительные условия онлайн установки, а также дополнительные подготовительные шаги.**
2. **Подготовьте пакет драйвера и прошивки, а также пакет MindIO SDK.** Скачайте следующие пакеты (если установка MindIO не требуется, скачивать пакет MindIO не нужно):

   - Для пакета драйвера и прошивки найдите файл `config.json` в репозитории GitCode npu-driver-installer ↗ и скачайте пакет, соответствующий выбранной версии, модели NPU и архитектуре ОС соответствующего узла по предоставленной ссылке.
   - Для пакета MindIO SDK найдите файл `config.json` в репозитории GitCode npu-node-provision ↗ и скачайте SDK пакет, соответствующий модели NPU и

архитектуре ОС соответствующего узла по предоставленной ссылке.

3. **Сохраните ZIP-файл пакета драйвера и прошивки в путь `/tmp/driver_pkg/` на узле, где будет выполняться офлайн установка.**
4. **Сохраните ZIP-файл пакета MindIO в путь `/opt/openFuyao/mindio/` на узле, где будет выполняться офлайн установка. (Если установка MindIO не требуется, этот шаг пропустите.)**
5. **Проверьте, установлены ли на целевом узле следующие инструменты.**

   - Для систем с менеджером пакетов Yum необходимо установить пакет: "jq wget unzip which net-tools pciutils gcc make kernel-devel-$(uname -r) kernel-headers-$(uname -r) dkms".
   - Для систем с менеджером пакетов apt-get необходимо установить пакет: "jq wget unzip debianutils net-tools pciutils gcc make dkms linux-headers-$(uname -r)".
   - Для систем с менеджером пакетов DNF необходимо установить пакет: "jq wget unzip which net-tools pciutils gcc make kernel-devel-$(uname -r) kernel-headers-$(uname -r) dkms".

Процедура

Загрузка кластерного плагина

> **INFO**
>
> Вы можете скачать приложения `Alauda Build of NPU Operator` и `Alauda Build of Node Feature Discovery` из Marketplace на сайте Customer Portal.

Примечание: плагин кластера Volcano можно пока не устанавливать.

Загрузка кластерного плагина

Платформа предоставляет командный инструмент `violet` для загрузки пакетов, скачанных из Marketplace Custom Portal.

Подробности смотрите в разделе Upload Packages.

Установка Alauda Build of NPU Operator

1. Примените метку `masterselector=dls-master-node` ко всем мастер-узлам и метку `workerselector=dls-worker-node` ко всем рабочим узлам.

```
kubectl label nodes {master-node-id} masterselector=dls-master-node
kubectl label nodes {worker-node-id} workerselector=dls-worker-node
```

2. Перейдите на страницу `Administrator` -> `Marketplace` -> `Cluster Plugin`, переключитесь на целевой кластер и разверните кластерный плагин `Alauda Build of NPU Operator`.
   Описание параметров формы развертывания:

> **WARNING**
>
> Если компоненты, перечисленные в таблице ниже, уже установлены, обязательно отключите соответствующие кнопки при развертывании.

> **TIP**
>
> Ascend Operator, NodeD, ClusterD, Resilience Controller, MindIO TFT и MindIO ACP по умолчанию не развертываются. Разворачивайте их только при явной необходимости.

Компонент	По умолчанию	Описание
Driver	Включено	Устанавливать ли драйвер и прошивку.
Version	24.1.RC3	Версия драйвера и прошивки. Необходимо выбрать номер версии из каталога репозитория npu-driver-installer ↗.

Компонент	По умолчанию	Описание
Ascend Device Plugin	Включено	Устанавливать ли Ascend Device Plugin.
Ascend Docker Runtime	Включено	Устанавливать ли Ascend Docker Runtime.
NPU exporter	Включено	Устанавливать ли NPU exporter.
Ascend Operator	Отключено	Устанавливать ли Ascend Operator.
NodeD	Отключено	Устанавливать ли NodeD.
ClusterD	Отключено	Устанавливать ли ClusterD. Для этого сначала нужно установить плагин кластера Volcano.
Resilience Controller	Отключено	Устанавливать ли Resilience Controller.
MindIO TFT	Отключено	Устанавливать ли MindIO TFT.
MindIO ACP	Отключено	Устанавливать ли MindIO ACP.

Проверка

1. Сначала убедитесь, что на странице кластерного плагина `Alauda Build of NPU Operator` отображается статус "Installed".
2. Дождитесь, пока pod npu-driver перейдет в состояние running. Офлайн установка занимает около 10 минут, онлайн установка проходит значительно быстрее.

```
kubectl -n kube-system get pod -w | grep npu-driver
```

3. Перезагрузите все NPU узлы.

4. Выполните следующую команду на узле NPU.

```
npu-smi info
```

   Убедитесь, что вывод корректен.

5. Выполните следующую команду на мастер-узле.

```
kubectl get npuclusterpolicy cluster
```

   Убедитесь, что статус `npuclusterpolicy` — Ready.

6. Проверьте, есть ли доступные NPU ресурсы на узле NPU в управляющем узле бизнес-кластера. Выполните команду:

```
kubectl get node  ${nodeName} -o=jsonpath='{.status.allocatable}'
# Пример, вывод содержит: "huawei.com/Ascend310P":"1" (конкретное значение зависит от количества NPU карт)
```

7. Запустите тестовую нагрузку.

   NOTE

   Бизнес-приложения должны вручную указывать поле `runtimeClassName` со значением `ascend`.

   Создайте файл спецификации:

```
key="huawei.com/Ascend310P" # Для 310P
cat <<EOF > deploy-npu.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ascend-pytorch
spec:
  replicas: 1
  selector:
    matchLabels:
      service.cpaas.io/name: deployment-ascend-pytorch
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
    type: RollingUpdate
  template:
    metadata:
      labels:
        service.cpaas.io/name: deployment-ascend-pytorch
    spec:
      affinity: {}
      containers:
        - args:
            - |
              sleep infinity
          command:
            - /bin/bash
            - -c
          image: ascendai/pytorch:ubuntu-python3.8-cann8.0.rc1.beta1-py
torch2.1.0
          imagePullPolicy: Always
          name: ascend-pytorch
          resources:
            limits:
              cpu: 500m
              $key: "1"
              memory: 2Gi
            requests:
              cpu: 500m
              memory: 2Gi
      runtimeClassName: ascend
EOF
```

Примените спецификацию:

```
kubectl apply -f deploy-npu.yaml
```

```
kubectl exec -it  deploy/ascend-pytorch -- bash
```

Затем выполните в контейнере:

```
npu-smi info
```

Убедитесь, что вывод корректен.

Установка мониторинга

Если при установке Alauda Build of NPU Operator был развернут компонент NPU exporter, выполните следующие шаги для создания панели мониторинга.

1. Выполните команды на **управляющем узле** кластера.

```
cat << EOF | kubectl apply -f -
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    prometheus: kube-prometheus
    serviceMonitorSelector: prometheus
  name: npu-exporter-ai
  namespace: monitoring
spec:
  endpoints:
  - interval: 10s
    path: /metrics
    port: http
    targetPort: 8082
  namespaceSelector:
    matchNames:
    - npu-exporter
  selector:
    matchLabels:
      app: npu-exporter-svc
EOF
```

2. Вы можете импортировать JSON-файл дашборда Grafana, следуя инструкции Import Dashboard, чтобы преобразовать его в панель мониторинга для отображения. JSON-файл доступен в репозитории ascend-npu-dashboard ↗.

NOTE

Теги в JSON-файле дашборда Grafana не могут содержать китайские символы и должны быть удалены вручную. Пример:

```
{
  "tags": [
    "ascend",
    "昇腾"
  ]
}
```

После правки:

```
{
  "tags": [
    "ascend"
  ]
}
```

FAQ

На что обратить внимание при удалении Alauda Build of NPU Operator?

Даже после удаления Alauda Build of NPU Operator драйвер может остаться на хост-машине. На узле NPU выполните следующую команду для удаления драйвера:

```
/usr/local/Ascend/driver/script/uninstall.sh
```

About Alauda Build of Hami

Heterogeneous AI Computing Virtualization Middleware (HAMi), ранее известный как k8s-vGPU-scheduler, представляет собой универсальный чарт, предназначенный для управления гетерогенными AI-вычислительными устройствами в кластере k8s. Он обеспечивает возможность совместного использования гетерогенных AI-устройств между задачами.

Note

Поскольку выпуски Alauda Build of Hami осуществляются в ином режиме, чем у Alauda Container Platform, документация Alauda Build of Hami теперь доступна в виде отдельного набора по адресу Alauda Build of Hami ↗。

О плагине устройства NVIDIA GPU сборки Alauda

Плагин устройства NVIDIA для Kubernetes — это Daemonset, который позволяет вам автоматически:

- Отображать количество GPU на каждом узле вашего кластера
- Отслеживать состояние здоровья ваших GPU
- Запускать контейнеры с поддержкой GPU в вашем Kubernetes кластере.

Note

Поскольку выпуски Alauda Build of NVIDIA GPU Device Plugin осуществляются в ином режиме, чем у Alauda Container Platform, документация Alauda Build of NVIDIA GPU Device Plugin теперь доступна в виде отдельного набора по адресу Alauda Build of NVIDIA GPU Device Plugin ↗。