Arvados - Story #17240

Scoping/grooming GPU support work

01/06/2021 06:01 PM - Peter Amstutz

Status:	In Progress	**Start date:**	04/07/2021
Priority:	Normal	**Due date:**	
Assigned To:	Peter Amstutz	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:	2021-06-09 sprint		

Description

https://docs.nvidia.com/deploy/cuda-compatibility/index.html

Nvidia says

The CUDA software environment consists of three parts:

- CUDA Toolkit (libraries, CUDA runtime and developer tools) - User-mode SDK used to build CUDA applications
- CUDA driver - User-mode driver component used to run CUDA applications (such as libcuda.so on Linux systems)
- NVIDIA GPU device driver - Kernel-mode driver component for NVIDIA GPUs

On Linux systems, the CUDA driver and kernel mode components are delivered together in the NVIDIA display driver package. This is shown in Figure 1.

...

1.3. Binary Compatibility

We define binary compatibility as a set of guarantees provided by the library, where an application targeting the said library will continue to work when dynamically linked against a different version of the library.

The CUDA Driver API has a versioned C-style ABI, which guarantees that applications that were running against an older driver (for example CUDA 3.2) will still run and function correctly against a modern driver (for example one shipped with CUDA 11.0). This is a stronger contract than an API guarantee - an application might need to change its source when recompiling against a newer SDK, but replacing the driver with a newer version will always work.

The CUDA Driver API thus is binary-compatible (the OS loader can pick up a newer version and the application continues to work) but not source-compatible (rebuilding your application against a newer SDK might require source changes). In addition, the binary-compatibility is in one direction: backwards.

...

Each version of the CUDA Toolkit (and runtime) requires a minimum version of the NVIDIA driver. The CUDA driver (libcuda.so on Linux for example) included in the NVIDIA driver package, provides binary backward compatibility. For example, an application built against the CUDA 3.2 SDK will continue to function even on today's driver stack. On the other hand, the CUDA runtime has not provided either source or binary compatibility guarantees. Newer major and minor versions of the CUDA runtime have frequently changed the exported symbols, including their version or even their availability, and the dynamic form of the library has its shared object name (.SONAME in Linux-based systems) change every minor version.

Notes

Inside the container: must include the correct nvidia runtime (if dynamically linked) or the application must be statically linked.

- runtime requires a minimum version of the driver -- this should be declared as a requirement
- nvidia-smi tells us some stuff?
- cubins (programs compiled directly for a GPU) target a specific "compute" capability and are only backwards compatible across minor revisions.
- required libraries: libcuda.so.* - the CUDA Driver
- required libraries: libnvidia-ptxjitcompiler.so.*

Apparently nvidia also offers "driver containers" where it actually installs the kernel driver (???) on the fly and a persistent daemon, instead of relying on drivers being installed on the host.

https://docs.nvidia.com/datacenter/cloud-native/driver-containers/overview.html

- to use docker, need to install nvidia-container-runtime but it's confusing because there are a whole bunch of container related projects on github:
    - https://github.com/NVIDIA/nvidia-docker
    - https://github.com/nvidia/nvidia-container-runtime
    - https://github.com/NVIDIA/libnvidia-container
    - https://github.com/NVIDIA/nvidia-container-toolkit

- https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/user-guide.html

- Singularity support for bind mounting the nvidia driver exists. It apparently requires nvidia-container-cli
  https://github.com/NVIDIA/libnvidia-container
  Seems like you can use this tool to interrogate the system to find out which libraries need to be bind mounted.

node needs to declare, for each device:

- driver version
- hardware capability

container request needs to specify

- minimum driver version → select nodes with >= minimum driver version
- cubin hardware capability (or none) → select nodes with = major revision, >= minor revision hardware capability
    - can compile for multiple targets, so this should be a list
- PTX hardware capability → select nodes with >= PTX hardware capability

It seems each SDK release has a new driver, so often the SDK version is printed instead of the underlying driver version. There's a table of which driver corresponds to which SDK revision.

With version 11 of the SDK it seems that the userspace part of the driver can be upgraded without upgrading the kernel driver.

What docker --gpus does

https://github.com/docker/cli/blob/88c6089300a82d3373892adf6845a4fed1a4ba8d/docs/reference/commandline/run.md

https://docs.docker.com/config/containers/resource_constraints/

https://github.com/docker/cli/blob/88c6089300a82d3373892adf6845a4fed1a4ba8d/opts/gpus.go

https://github.com/moby/moby/blob/46cdcd206c56172b95ba5c77b827a722dab426c5/daemon/nvidia_linux.go

Using the API, the simplest valid request is DeviceRequest with:

- DeviceRequest.Count = 1
- DeviceRequest.Capabilities = [["gpu"]]

A better one is probably:

- DeviceRequest.Driver = "nvidia"
- DeviceRequest.Count = -1 (request all GPUs)
- DeviceRequest.Capabilities = [["gpu", "nvidia", "compute"]]

Docker sets NVIDIA_VISIBLE_DEVICES and NVIDIA_DRIVER_CAPABILITIES

Driver capabilities and options:

https://github.com/nvidia/nvidia-container-runtime#supported-driver-capabilities

NVIDIA_REQUIRE_CUDA is a thing, this takes the SDK version and checks the driver version.

CWL support

```
CUDARequirement:
  minCUDADriverVersion: "10.0"              (required)
  minHardwareCapabilityCubins: ["7.0", "8.0"]   (optional, default null)
  minHardwardCapabilityPTX: "7.0"               (optional, default null)
  minDeviceCount: 1                           (optional, default 1)
  maxDeviceCount: 1                           (optional, default 1)
```

- minCUDADriverVersion: minimum driver version. CUDA SDK version (each SDK release mostly corresponds to a new driver revision). We *could* use the driver revision but that seems more likely to confuse people.
- minHardwareCapabilityCubins: supported nvidia architectures, if GPU programs are precompiled
- minHardwardCapabilityPTX: minimum hardware architecture, if using JIT compilation from PTX or C++
- min/maxDeviceCount: can request/require multiple devices, if program supports it.

At least one of cubinHardwareCapability or PTXHardwardCapability must be non-empty.

Arvados support

```
InstanceTypes:
  instanceWithGPU:
    ...
    CUDA:
      DriverVersion: "11.0"
      HardwareCapability: "9.0"
      DeviceCount: 1

runtime_constraints: {
  cuda_driver_version: "10.0"
  cuda_cubin_hardware_capability: ["9.0"]
  cuda_ptx_hardware_capability: "9.0"
  cuda_device_count: 1
}
```

Instance selection:

Select instance type which has

- InstanceType.DriverVersion >= cuda_driver_version
- InstanceType.HardwareCapability exists (>= with same major version) in cuda_cubin_hardware_capability, or null
- InstanceType.HardwareCapability >= cuda_ptx_hardware_capability, or null
- InstanceType.DevicesCount >= cuda_device_count

Subtasks:

Task # 17446: Group review	**In Progress**

Related issues:

Related to Arvados Epics - Story #15957: GPU support	**In Progress**	**01/01/2021**	**07/30/2021**

History

#1 - 01/06/2021 06:02 PM - Peter Amstutz

- Assigned To set to Peter Amstutz

#2 - 01/19/2021 06:44 PM - Peter Amstutz

- Target version changed from 2021-01-20 Sprint to 2021-02-03 Sprint

#3 - 02/02/2021 06:05 PM - Peter Amstutz

- Target version changed from 2021-02-03 Sprint to 2021-02-17 sprint

#4 - 02/18/2021 04:35 PM - Peter Amstutz

- Target version changed from 2021-02-17 sprint to 2021-03-03 sprint

#5 - 03/01/2021 06:27 PM - Peter Amstutz

- Target version changed from 2021-03-03 sprint to 2021-03-17 sprint

#6 - 03/04/2021 10:37 PM - Peter Amstutz

- Description updated

#7 - 03/04/2021 11:02 PM - Peter Amstutz

- Description updated

#8 - 03/05/2021 02:59 PM - Peter Amstutz

- Description updated

#9 - 03/05/2021 09:08 PM - Peter Amstutz

- Description updated

#10 - 03/05/2021 09:35 PM - Peter Amstutz

- Description updated

#11 - 03/05/2021 09:37 PM - Peter Amstutz

- Description updated

#12 - 03/05/2021 10:07 PM - Peter Amstutz

- Description updated

#13 - 03/05/2021 10:41 PM - Peter Amstutz

- Description updated

#14 - 03/05/2021 10:44 PM - Peter Amstutz

- Description updated

#15 - 03/05/2021 10:46 PM - Peter Amstutz

- Description updated

#16 - 03/08/2021 07:49 PM - Peter Amstutz

- Status changed from New to In Progress

#17 - 03/10/2021 04:01 PM - Peter Amstutz

- Related to Story #15957: GPU support added

#18 - 03/11/2021 03:01 PM - Peter Amstutz

- Target version changed from 2021-03-17 sprint to 2021-03-31 sprint

#19 - 03/31/2021 02:27 PM - Peter Amstutz

- Target version changed from 2021-03-31 sprint to 2021-04-14 sprint

#20 - 03/31/2021 03:43 PM - Peter Amstutz

- Target version changed from 2021-04-14 sprint to 2021-05-26 sprint

#21 - 04/07/2021 10:34 PM - Nico César

https://www.ibm.com/docs/en/spectrum-lsf/10.1.0?topic=lsfconf-lsb-gpu-req

#22 - 05/14/2021 02:29 PM - Peter Amstutz

- Target version changed from 2021-05-26 sprint to 2021-06-09 sprint