NEW CUDA TOOLKIT PACKAGES FOR CONDA

RICK RATZEL, THOMSON COMER, JOHN KIRKHAM

Agenda

- What is CUDA and the CUDA Toolkit?
- Conda, channels, and how they're used to create a CUDA Toolkit environment
- The initial CUDA Toolkit conda packaging
  - ...and some of its problems
- The new CUDA Toolkit conda packaging!
  - ...and how you can take advantage of it as a package maintainer, end user, etc.
- Current status
- Next steps

What is CUDA and the CUDA Toolkit?

- Compute Unified Device Architecture (CUDA)
  - System drivers for programmable NVIDIA GPUs
- CUDA Toolkit (CTK)
  - GPU Accelerated Scientific Programming Libraries
  - NVCC (Compiler)
  - Cuda-GDB (Debugger)
  - Debug tools (Nsight, profiles, etc.)
  - cuBLAS, other libs, etc.

Conda

- Package manager (particularly in the Python & Data Science space)
- Language agnostic (so easy to glue C/C++ libraries, Python, Rust, Go, etc. together)
- User friendly (can install in user's home directory or elsewhere without issues)

CONDA

Conda channels

- Way to distribute a collection of packages that work together
- Similar to repos in Enterprise Linux
- Couple major ones and a few domain specific ones

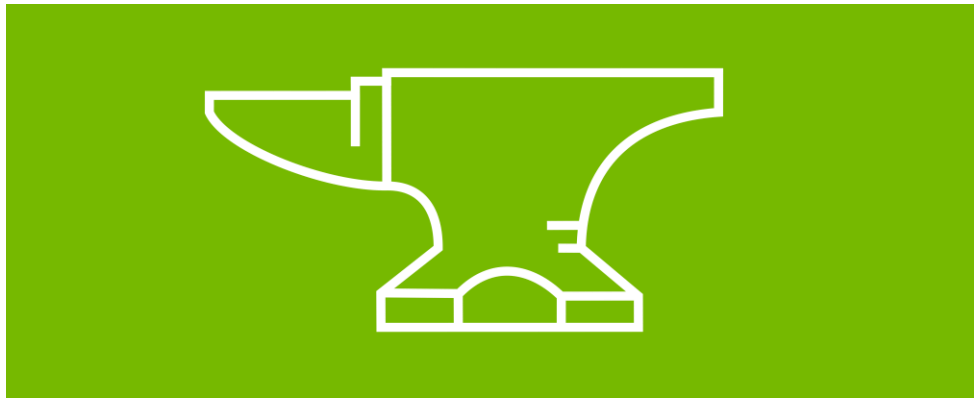
- A few common Conda channels are…
  - defaults
  - conda-forge
  - nvidia
- Some domain specific ones:
  - rapidsai (data science libraries)
  - numba (particular for RC / nightlies of Numba)
  - dask (also for RC / nightlies)
  - pytorch (channel for pytorch + friends)

```
$ conda install --channel conda-forge scipy
```

Conda Channel Tiers

- Defaults
  - Professionally curated, self-contained, tested set of packages

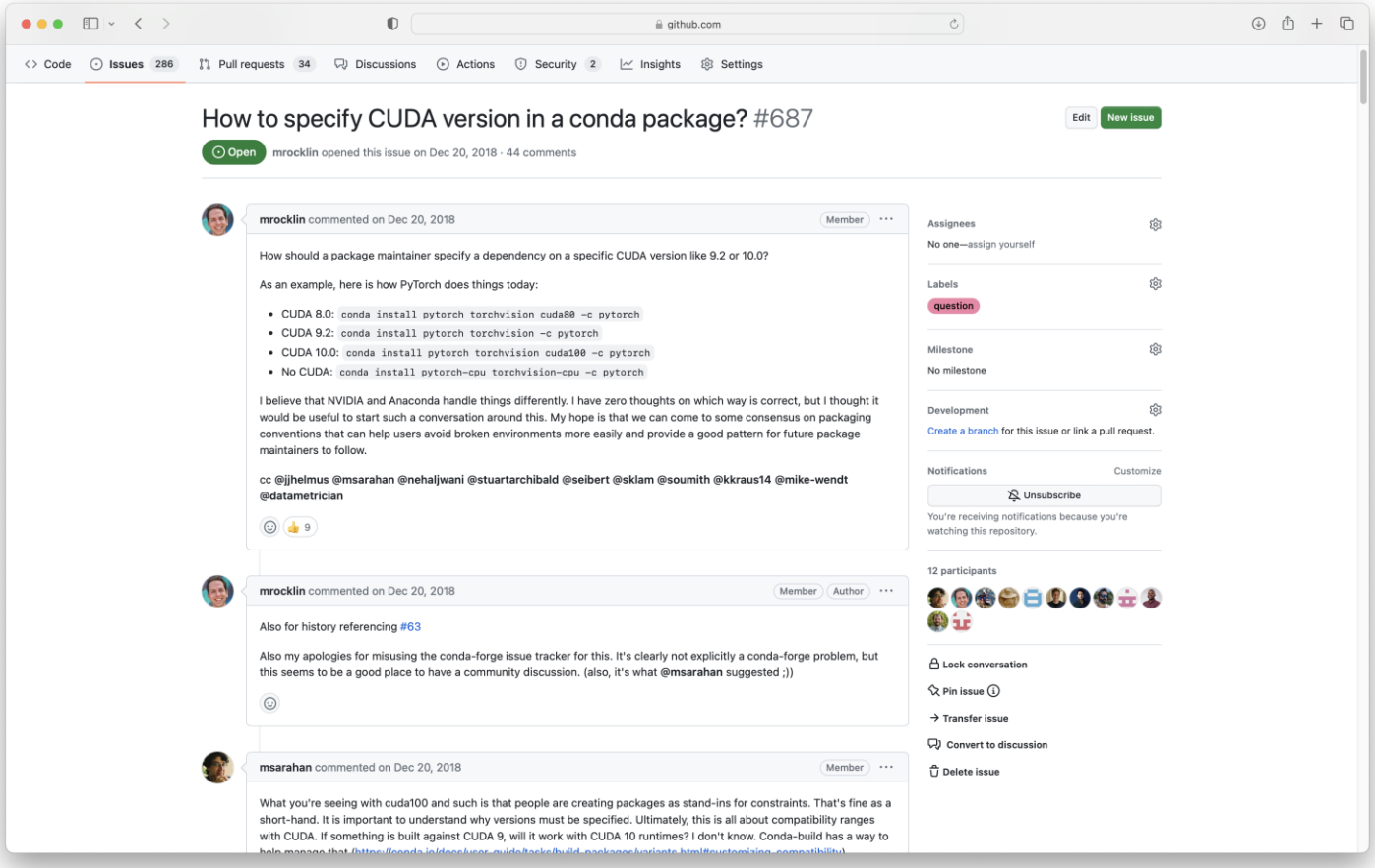
- Conda-forge
  - Maintained by a community of packaging enthusiasts
  - Consumed by a broad range of users
  - Free to use / contribute to
  - Easily accessible / used without restrictions
  - Self-contained
  - Has lightweight installers for getting started



- Nvidia
  - Not quite as big as defaults & much smaller than conda-forge
  - Primarily CTK packages
  - Other packages: NVTabular, Merlin, cuQuantum, & some RAPIDS primitive libraries

Initial versions and early cross-channel CUDA coordination

https://github.com/conda-forge/conda-forge.github.io/issues/687



github.com

Code | Issues 286 | Pull requests 34 | Discussions | Actions | Security 2 | Insights | Settings

How to specify CUDA version in a conda package? #687

Edit | New issue

Open | mrocklin opened this issue on Dec 20, 2018 · 44 comments

mrocklin commented on Dec 20, 2018 — Member

How should a package maintainer specify a dependency on a specific CUDA version like 9.2 or 10.0?

As an example, here is how PyTorch does things today:

- CUDA 8.0: `conda install pytorch torchvision cuda80 -c pytorch`
- CUDA 9.2: `conda install pytorch torchvision -c pytorch`
- CUDA 10.0: `conda install pytorch torchvision cuda100 -c pytorch`
- No CUDA: `conda install pytorch-cpu torchvision-cpu -c pytorch`

I believe that NVIDIA and Anaconda handle things differently. I have zero thoughts on which way is correct, but I thought it would be useful to start such a conversation around this. My hope is that we can come to some consensus on packaging conventions that can help users avoid broken environments more easily and provide a good pattern for future package maintainers to follow.

cc **@jjhelmus @msarahan @nehaljwani @stuartarchibald @seibert @sklam @soumith @kkraus14 @mike-wendt @datametrician**

9

mrocklin commented on Dec 20, 2018 — Member, Author

Also for history referencing #63

Also my apologies for misusing the conda-forge issue tracker for this. It's clearly not explicitly a conda-forge problem, but this seems to be a good place to have a community discussion. (also, it's what **@msarahan** suggested ;))

msarahan commented on Dec 20, 2018 — Member

What you're seeing with cuda100 and such is that people are creating packages as stand-ins for constraints. That's fine as a short-hand. It is important to understand why versions must be specified. Ultimately, this is all about compatibility ranges with CUDA. If something is built against CUDA 9, will it work with CUDA 10 runtimes? I don't know. Conda-build has a way to

Assignees: No one—assign yourself

Labels: question

Milestone: No milestone

Development: Create a branch for this issue or link a pull request.

Notifications: Customize — Unsubscribe — You're receiving notifications because you're watching this repository.

12 participants

Lock conversation | Pin issue | Transfer issue | Convert to discussion | Delete issue

CUDA Toolkit alignment

For versions from 2018-present

- Align on cudatoolkit package for configuring CUDA version used by packages
- All CUDA packages depend on cudatoolkit
- Add __cuda virtual package to Conda for CUDA driver support version detection
- Constrain cudatoolkit on __cuda
- Recipe authors only need to think about adding `{{ compiler("cuda") }}` in `requirements/build`

nvcc-feedstock meta.yaml

```
    run_exports:
      strong:
{% if cuda_major_minor < (11, 2) %}
        - cudatoolkit {{ cuda_compiler_version }}|{{ cuda_compiler_version }}.*
{% else %}
        - cudatoolkit >={{ cuda_compiler_version }},<{{ cuda_major + 1 }}
{% endif %}
```

cudatoolkit meta.yaml

```
  run_constrained:
{% if major_version < 11 %}
    - __cuda >={{ major_minor }}
{% else %}
    - __cuda >={{ major_version }}
{% endif %}
    - arm-variant * {{ arm_variant_type }}
```

CUDA Toolkit contents

For versions from 2018-present in conda-forge

```
cudatoolkit
└── lib
  ├── libcublasLt.so -> libcublasLt.so.11.11.3.6
  ├── libcublasLt.so.11 -> libcublasLt.so.11.11.3.6
  ├── libcublasLt.so.11.11.3.6
  ├── libcublas.so -> libcublas.so.11.11.3.6
  ├── libcublas.so.11 -> libcublas.so.11.11.3.6
  ├── libcublas.so.11.11.3.6
  ├── libcudadevrt.a
  ├── libcudart.so -> libcudart.so.11.8.89
  ├── libcudart.so.11.0 -> libcudart.so.11.8.89
  ├── libcudart.so.11.8.89
  ├── libcufft.so -> libcufft.so.10.9.0.58
  ├── libcufft.so.10 -> libcufft.so.10.9.0.58
  ├── libcufft.so.10.9.0.58
  ├── libcufftw.so -> libcufftw.so.10.9.0.58
  ├── libcufftw.so.10 -> libcufftw.so.10.9.0.58
  ├── libcufftw.so.10.9.0.58
  ├── libcupti.so -> libcupti.so.2022.3.0
  ├── libcupti.so.11.8 -> libcupti.so.2022.3.0
  ├── libcupti.so.2022.3.0
  ├── libcurand.so -> libcurand.so.10.3.0.86
  ├── libcurand.so.10 -> libcurand.so.10.3.0.86
  ├── libcurand.so.10.3.0.86
  ├── libcusolverMg.so -> libcusolverMg.so.11.4.1.48
  ├── libcusolverMg.so.11 -> libcusolverMg.so.11.4.1.48
  ├── libcusolverMg.so.11.4.1.48
  ├── libcusolver.so -> libcusolver.so.11.4.1.48
  ├── libcusolver.so.11 -> libcusolver.so.11.4.1.48
  ├── libcusolver.so.11.4.1.48
  ├── libcusparse.so -> libcusparse.so.11.7.5.86
  ├── libcusparse.so.11 -> libcusparse.so.11.7.5.86
  ├── libcusparse.so.11.7.5.86
  ├── libdevice.10.bc
  ├── libnppc.so -> libnppc.so.11.8.0.86
  ├── libnppc.so.11 -> libnppc.so.11.8.0.86
  ├── libnppc.so.11.8.0.86
  ├── libnppial.so -> libnppial.so.11.8.0.86
  ├── libnppial.so.11 -> libnppial.so.11.8.0.86
  ├── libnppial.so.11.8.0.86
  ├── libnppicc.so -> libnppicc.so.11.8.0.86
  ├── libnppicc.so.11 -> libnppicc.so.11.8.0.86
  ├── libnppicc.so.11.8.0.86
  ├── libnppidei.so -> libnppidei.so.11.8.0.86
  ├── libnppidei.so.11 -> libnppidei.so.11.8.0.86
  ├── libnppidei.so.11.8.0.86
  ├── libnppif.so -> libnppif.so.11.8.0.86
  ├── libnppif.so.11 -> libnppif.so.11.8.0.86
  ├── libnppif.so.11.8.0.86
  ├── libnppig.so -> libnppig.so.11.8.0.86
  ├── libnppig.so.11 -> libnppig.so.11.8.0.86
  ├── libnppig.so.11.8.0.86
  ├── libnppim.so -> libnppim.so.11.8.0.86
  ├── libnppim.so.11 -> libnppim.so.11.8.0.86
  ├── libnppim.so.11.8.0.86
  ├── libnppist.so -> libnppist.so.11.8.0.86
  ├── libnppist.so.11 -> libnppist.so.11.8.0.86
  ├── libnppist.so.11.8.0.86
  ├── libnppisu.so -> libnppisu.so.11.8.0.86
  ├── libnppisu.so.11 -> libnppisu.so.11.8.0.86
  ├── libnppisu.so.11.8.0.86
  ├── libnppitc.so -> libnppitc.so.11.8.0.86
  ├── libnppitc.so.11 -> libnppitc.so.11.8.0.86
  ├── libnppitc.so.11.8.0.86
  ├── libnpps.so -> libnpps.so.11.8.0.86
  ├── libnpps.so.11 -> libnpps.so.11.8.0.86
  ├── libnpps.so.11.8.0.86
  ├── libnvblas.so -> libnvblas.so.11.11.3.6
  ├── libnvblas.so.11 -> libnvblas.so.11.11.3.6
  ├── libnvblas.so.11.11.3.6
  ├── libnvjpeg.so -> libnvjpeg.so.11.9.0.86
  ├── libnvjpeg.so.11 -> libnvjpeg.so.11.9.0.86
  ├── libnvjpeg.so.11.9.0.86
  ├── libnvrtc-builtins.so -> libnvrtc-builtins.so.11.8.89
  ├── libnvrtc-builtins.so.11.8 -> libnvrtc-builtins.so.11.8.89
  ├── libnvrtc-builtins.so.11.8.89
  ├── libnvrtc.so -> libnvrtc.so.11.8.89
  ├── libnvrtc.so.11.2 -> libnvrtc.so.11.8.89
  ├── libnvrtc.so.11.8.89
  ├── libnvToolsExt.so -> libnvToolsExt.so.1.0.0
  ├── libnvToolsExt.so.1 -> libnvToolsExt.so.1.0.0
  ├── libnvToolsExt.so.1.0.0
  ├── libnvvm.so -> libnvvm.so.4.0.0
  ├── libnvvm.so.4 -> libnvvm.so.4.0.0
  └── libnvvm.so.4.0.0
```

So we are done right?

Excerpt from an "Odd Lots" Podcast episode

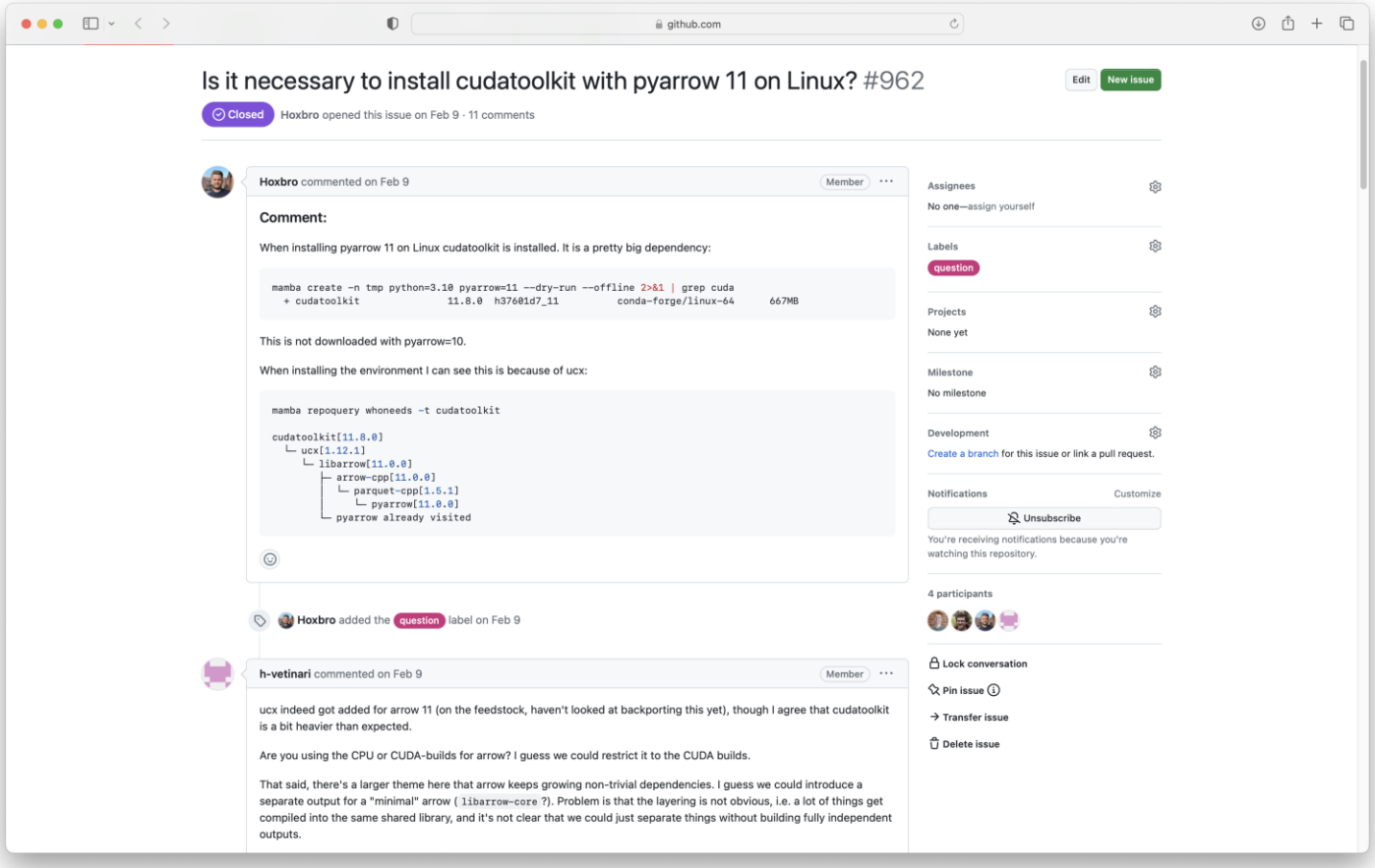
- (Tracy Alloway) It seems with these large-scale systems that there's always change. Something is always in motion. Something is always in flux. Why is that?

- (Patrick McKenzie) Well software engineers are working this week on a increasingly complex world where software is more leveraged than it had been even last week where there are increasing demands on the world etc.... Is there going to be a time where the last line of software is written? Probably not. There will never be a last bit of software written....Humans want more things out of the world and we have...infinite capacity for want at the margin.

https://www.youtube.com/watch?v=v6UQaXpzwQA&t=1933s

cudatoolkit is massive!

Is it necessary to install cudatoolkit with pyarrow 11 on Linux?



github.com

Is it necessary to install cudatoolkit with pyarrow 11 on Linux? #962

Edit | New issue

Closed | Hoxbro opened this issue on Feb 9 · 11 comments

Hoxbro commented on Feb 9 — Member

Comment:

When installing pyarrow 11 on Linux cudatoolkit is installed. It is a pretty big dependency:

```
mamba create -n tmp python=3.10 pyarrow=11 --dry-run --offline 2>&1 | grep cuda
  + cudatoolkit              11.8.0  h37601d7_11          conda-forge/linux-64      667MB
```

This is not downloaded with pyarrow=10.

When installing the environment I can see this is because of ucx:

```
mamba repoquery whoneeds -t cudatoolkit

cudatoolkit[11.8.0]
  └─ ucx[1.12.1]
     └─ libarrow[11.0.0]
        ├─ arrow-cpp[11.0.0]
        │  └─ parquet-cpp[1.5.1]
        │     └─ pyarrow[11.0.0]
        └─ pyarrow already visited
```

Hoxbro added the question label on Feb 9

h-vetinari commented on Feb 9 — Member

ucx indeed got added for arrow 11 (on the feedstock, haven't looked at backporting this yet), though I agree that cudatoolkit is a bit heavier than expected.

Are you using the CPU or CUDA-builds for arrow? I guess we could restrict it to the CUDA builds.

That said, there's a larger theme here that arrow keeps growing non-trivial dependencies. I guess we could introduce a separate output for a "minimal" arrow (`libarrow-core`?). Problem is that the layering is not obvious, i.e. a lot of things get compiled into the same shared library, and it's not clear that we could just separate things without building fully independent outputs.

Assignees: No one—assign yourself

Labels: question

Projects: None yet

Milestone: No milestone

Development: Create a branch for this issue or link a pull request.

Notifications: Customize — Unsubscribe — You're receiving notifications because you're watching this repository.

4 participants

Lock conversation

Pin issue

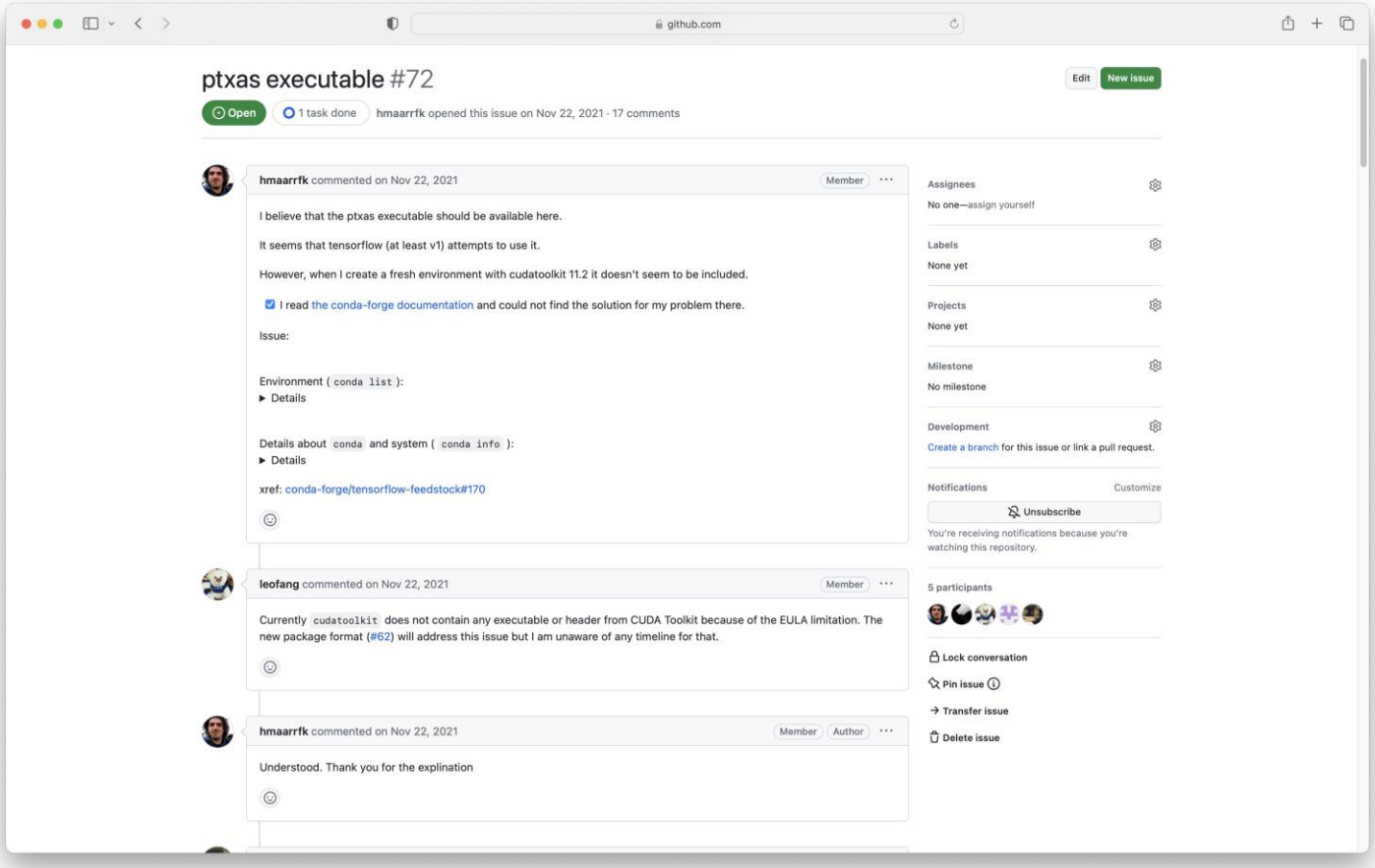
Transfer issue

Delete issue

https://github.com/conda-forge/arrow-cpp-feedstock/issues/962

And yet cudatoolkit is missing things?

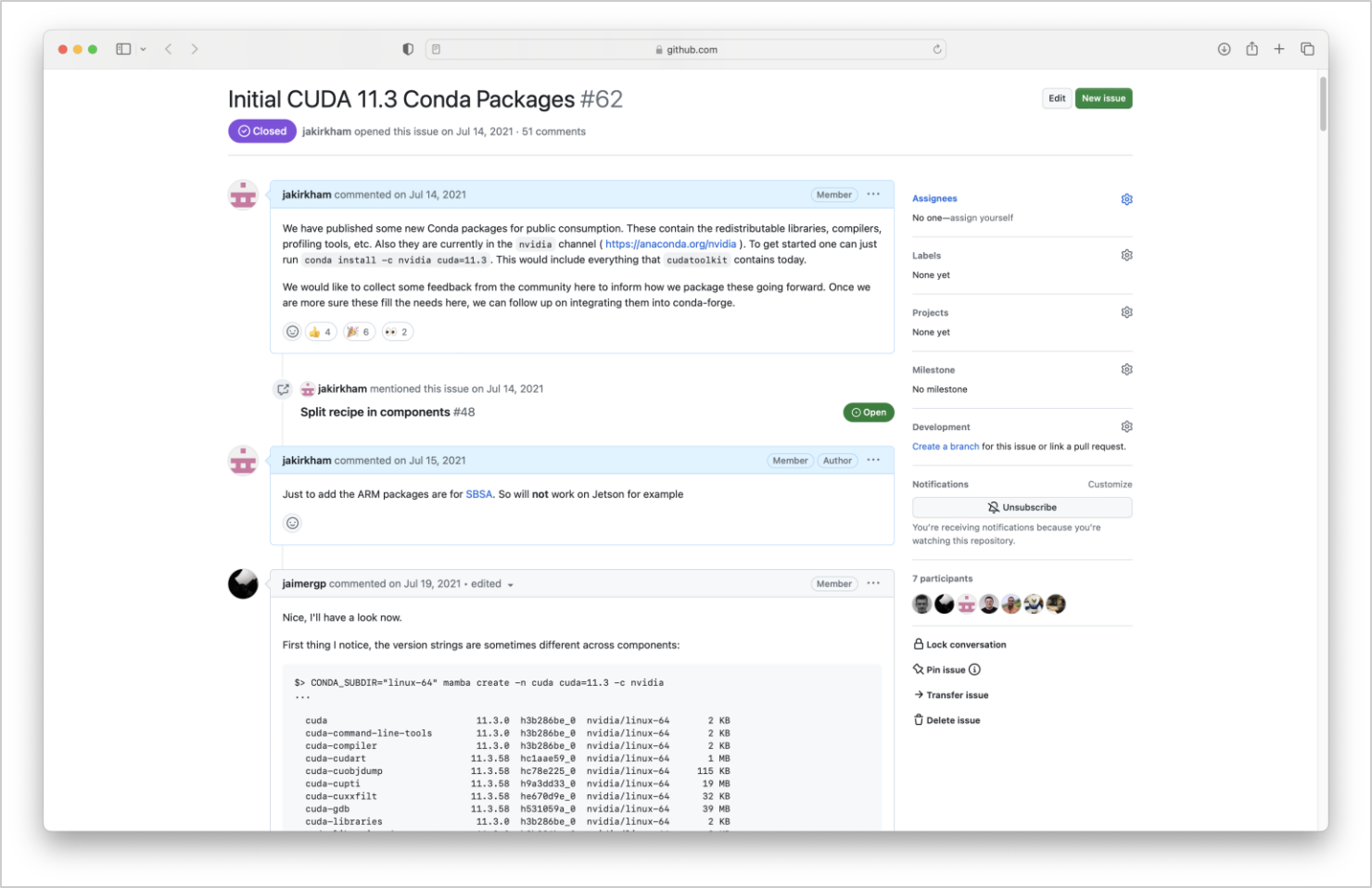
ptxas executable (needed for tensorflow)



https://github.com/conda-forge/cudatoolkit-feedstock/issues/72

Restructuring CTK packaging (in nvidia channel to start)

Initial CUDA 11.3 nvidia channel conda packages



github.com

Initial CUDA 11.3 Conda Packages #62

Edit | New issue

Closed | jakirkham opened this issue on Jul 14, 2021 · 51 comments

jakirkham commented on Jul 14, 2021 — Member

We have published some new Conda packages for public consumption. These contain the redistributable libraries, compilers, profiling tools, etc. Also they are currently in the `nvidia` channel (https://anaconda.org/nvidia). To get started one can just run `conda install -c nvidia cuda=11.3` . This would include everything that `cudatoolkit` contains today.

We would like to collect some feedback from the community here to inform how we package these going forward. Once we are more sure these fill the needs here, we can follow up on integrating them into conda-forge.

4 | 6 | 2

jakirkham mentioned this issue on Jul 14, 2021

Split recipe in components #48 — Open

jakirkham commented on Jul 15, 2021 — Member, Author

Just to add the ARM packages are for SBSA. So will **not** work on Jetson for example

jaimergp commented on Jul 19, 2021 • edited — Member

Nice, I'll have a look now.

First thing I notice, the version strings are sometimes different across components:

```
$> CONDA_SUBDIR="linux-64" mamba create -n cuda cuda=11.3 -c nvidia
...

  cuda                          11.3.0  h3b286be_0  nvidia/linux-64      2 KB
  cuda-command-line-tools       11.3.0  h3b286be_0  nvidia/linux-64      2 KB
  cuda-compiler                 11.3.0  h3b286be_0  nvidia/linux-64      2 KB
  cuda-cudart                  11.3.58  hc1aae59_0  nvidia/linux-64      1 MB
  cuda-cuobjdump               11.3.58  hc78e225_0  nvidia/linux-64    115 KB
  cuda-cupti                   11.3.58  h9a3dd33_0  nvidia/linux-64     19 MB
  cuda-cuxxfilt                11.3.58  he670d9e_0  nvidia/linux-64     32 KB
  cuda-gdb                     11.3.58  h531059a_0  nvidia/linux-64     39 MB
  cuda-libraries                11.3.0  h3b286be_0  nvidia/linux-64      2 KB
```

Assignees: No one—assign yourself

Labels: None yet

Projects: None yet

Milestone: No milestone

Development: Create a branch for this issue or link a pull request.

Notifications — Customize

Unsubscribe

You're receiving notifications because you're watching this repository.

7 participants

Lock conversation

Pin issue

Transfer issue

Delete issue

https://github.com/conda-forge/cudatoolkit-feedstock/issues/62

Restructuring CTK packaging


```
cuda
├─ cuda-demo-suite
├─ cuda-runtime
│  └─ cuda-libraries
│     ├─ cuda-cudart
│     ├─ cuda-nvrtc
│     ├─ cuda-opencl
│     ├─ libcublas
│     ├─ libcufft
│     ├─ libcufile
│     ├─ libcurand
│     ├─ libcusolver
│     ├─ libcusparse
│     ├─ libnpp
│     ├─ libnvjitlink
│     └─ libnvjpeg
└─ cuda-toolkit
   ├─ cuda-compiler
   │  ├─ cuda-cuobjdump
   │  ├─ cuda-cuxxfilt
   │  ├─ cuda-nvcc
   │  └─ cuda-nvprune
   ├─ cuda-documentation
   ├─ cuda-libraries
   │  ├─ cuda-cudart
   │  ├─ cuda-nvrtc
   │  ├─ cuda-opencl
   │  ├─ libcublas
   │  ├─ libcufft
   │  ├─ libcufile
   │  ├─ libcurand
   │  ├─ libcusolver
   │  ├─ libcusparse
   │  ├─ libnpp
   │  ├─ libnvjitlink
   │  └─ libnvjpeg
   ├─ cuda-libraries-dev
   │  ├─ cuda-cccl
   │  ├─ cuda-cudart-dev
   │  │  ├─ cuda-cccl
   │  │  └─ cuda-cudart
   │  ├─ cuda-driver-dev
   │  ├─ cuda-nvrtc-dev
   │  │  └─ cuda-nvrtc
   │  ├─ cuda-opencl-dev
   │  │  └─ cuda-opencl
   │  ├─ cuda-profiler-api
   │  ├─ libcublas-dev
   │  │  └─ libcublas
   │  ├─ libcufft-dev
   │  │  └─ libcufft
   │  ├─ libcufile-dev
   │  │  └─ libcufile
   │  ├─ libcurand-dev
   │  │  └─ libcurand
   │  ├─ libcusolver-dev
   │  │  └─ libcusolver
   │  ├─ libcusparse-dev
   │  │  └─ libcusparse
   │  ├─ libnpp-dev
   │  │  └─ libnpp
   │  ├─ libnvjitlink-dev
   │  │  └─ libnvjitlink
   │  └─ libnvjpeg-dev
   │     └─ libnvjpeg
   ├─ cuda-libraries-static
   │  ├─ cuda-cudart-static
   │  │  └─ cuda-cudart-dev
   │  │     ├─ cuda-cccl
   │  │     └─ cuda-cudart
   │  ├─ cuda-cupti-static
   │  │  └─ cuda-cupti
   │  ├─ cuda-nvrtc-static
   │  │  └─ cuda-nvrtc-dev
   │  │     └─ cuda-nvrtc
   │  ├─ libcublas-static
   │  │  └─ libcublas-dev
   │  │     └─ libcublas
   │  ├─ libcufft-static
   │  │  └─ libcufft-dev
   │  │     └─ libcufft
   │  ├─ libcufile-static
   │  │  └─ libcufile-dev
   │  │     └─ libcufile
   │  ├─ libcurand-static
   │  │  └─ libcurand-dev
   │  │     └─ libcurand
   │  ├─ libcusolver-static
   │  │  └─ libcusolver-dev
   │  │     └─ libcusolver
   │  ├─ libcusparse-static
   │  │  └─ libcusparse-dev
   │  │     └─ libcusparse
   │  ├─ libnpp-static
   │  │  └─ libnpp-dev
   │  │     └─ libnpp
   │  └─ libnvjpeg-static
   │     └─ libnvjpeg-dev
   │        └─ libnvjpeg
   ├─ cuda-nvml-dev
   ├─ cuda-tools
   │  ├─ cuda-command-line-tools
   │  │  ├─ cuda-cupti
   │  │  ├─ cuda-gdb
   │  │  ├─ cuda-nvdisasm
   │  │  ├─ cuda-nvprof
   │  │  ├─ cuda-nvtx
   │  │  └─ cuda-sanitizer-api
   │  ├─ cuda-visual-tools
   │  │  ├─ cuda-libraries-dev
   │  │  │  ├─ cuda-cccl
   │  │  │  ├─ cuda-cudart-dev
   │  │  │  │  ├─ cuda-cccl
   │  │  │  │  └─ cuda-cudart
   │  │  │  ├─ cuda-driver-dev
   │  │  │  ├─ cuda-nvrtc-dev
   │  │  │  │  └─ cuda-nvrtc
   │  │  │  ├─ cuda-opencl-dev
   │  │  │  │  └─ cuda-opencl
   │  │  │  ├─ cuda-profiler-api
   │  │  │  ├─ libcublas-dev
   │  │  │  │  └─ libcublas
   │  │  │  ├─ libcufft-dev
   │  │  │  │  └─ libcufft
   │  │  │  ├─ libcufile-dev
   │  │  │  │  └─ libcufile
   │  │  │  ├─ libcurand-dev
   │  │  │  │  └─ libcurand
   │  │  │  ├─ libcusolver-dev
   │  │  │  │  └─ libcusolver
   │  │  │  ├─ libcusparse-dev
   │  │  │  │  └─ libcusparse
   │  │  │  ├─ libnpp-dev
   │  │  │  │  └─ libnpp
   │  │  │  ├─ libnvjitlink-dev
   │  │  │  │  └─ libnvjitlink
   │  │  │  └─ libnvjpeg-dev
   │  │  │     └─ libnvjpeg
   │  │  ├─ cuda-libraries-static
   │  │  │  ├─ cuda-cudart-static
   │  │  │  │  └─ cuda-cudart-dev
   │  │  │  │     ├─ cuda-cccl
   │  │  │  │     └─ cuda-cudart
   │  │  │  ├─ cuda-cupti-static
   │  │  │  │  └─ cuda-cupti
   │  │  │  ├─ cuda-nvrtc-static
   │  │  │  │  └─ cuda-nvrtc-dev
   │  │  │  │     └─ cuda-nvrtc
   │  │  │  ├─ libcublas-static
   │  │  │  │  └─ libcublas-dev
   │  │  │  │     └─ libcublas
   │  │  │  ├─ libcufft-static
   │  │  │  │  └─ libcufft-dev
   │  │  │  │     └─ libcufft
   │  │  │  ├─ libcufile-static
   │  │  │  │  └─ libcufile-dev
   │  │  │  │     └─ libcufile
   │  │  │  ├─ libcurand-static
   │  │  │  │  └─ libcurand-dev
   │  │  │  │     └─ libcurand
   │  │  │  ├─ libcusolver-static
   │  │  │  │  └─ libcusolver-dev
   │  │  │  │     └─ libcusolver
   │  │  │  ├─ libcusparse-static
   │  │  │  │  └─ libcusparse-dev
   │  │  │  │     └─ libcusparse
   │  │  │  ├─ libnpp-static
   │  │  │  │  └─ libnpp-dev
   │  │  │  │     └─ libnpp
   │  │  │  └─ libnvjpeg-static
   │  │  │     └─ libnvjpeg-dev
   │  │  │        └─ libnvjpeg
   │  │  ├─ cuda-nsight
   │  │  ├─ cuda-nsight-compute
   │  │  │  └─ nsight-compute
   │  │  ├─ cuda-nvml-dev
   │  │  └─ cuda-nvvp
   │  │     ├─ cuda-nvdisasm
   │  │     └─ cuda-nvprof
   │  └─ gds-tools
   │     └─ libcufile
   └─ libnvvm-samples
```

CUDA Toolkit alignment

For upcoming CUDA 12 versions and newer

- Align on cuda-version package for configuring CUDA version used by packages (supersedes cudatoolkit)
- All CUDA 12 built packages depend on cuda-version
- CUDA 11 built packages using cudatoolkit can be constrained via cuda-version
- The cuda-version package is constrained by __cuda virtual package (and cudatoolkit for legacy CUDA 11 support)
- Recipe authors still only need to think about adding `{{ compiler("cuda") }}` in `requirements/build`

CUDA Toolkit alignment

For upcoming CUDA 12 versions and newer

```diff
-  number: 1
-  skip: true  # [(not linux) or (cuda_compiler_version != "11.2")]
   script: {{ PYTHON }} -m pip install . --no-deps -vv
   missing_dso_whitelist:
     - '*/libcuda.*'  # [linux]
@@ -30,10 +30,15 @@ requirements:
     - pip
     - pybind11
     - python
+    - cuda-cudart-dev  # [(cuda_compiler_version or "").startswith("12")]
+    - libcublas-dev    # [(cuda_compiler_version or "").startswith("12")]
+    - libcusparse-dev  # [(cuda_compiler_version or "").startswith("12")]
```

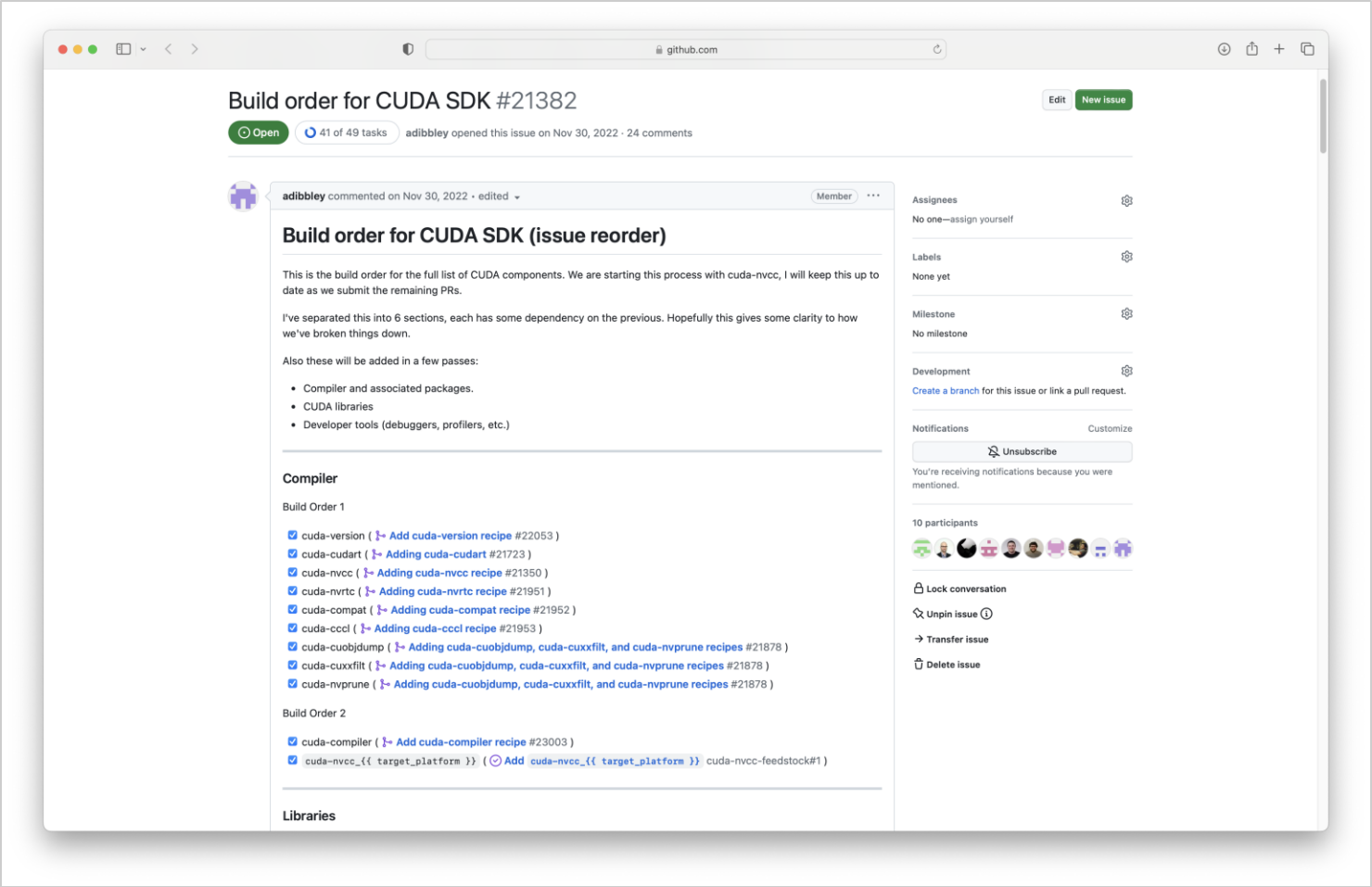
leofang marked this conversation as resolved. Show resolved

```diff
+    - cuda-version {{ cuda_compiler_version }}
   run:
     - {{ pin_compatible('custatevec', max_pin='x') }}
     - pennylane >=0.28
     - python
+    - {{ pin_compatible("cuda-version", min_pin="x", max_pin="x") }}
 test:
```

https://github.com/conda-forge/implicit-feedstock/pull/60

Current state: adding the new CTK CUDA 12.0 packages to conda-forge

Build order for CUDA SDK



Build order for CUDA SDK #21382

Open | 41 of 49 tasks | adibbley opened this issue on Nov 30, 2022 · 24 comments

Edit | New issue

adibbley commented on Nov 30, 2022 • edited | Member

Build order for CUDA SDK (issue reorder)

This is the build order for the full list of CUDA components. We are starting this process with cuda-nvcc, I will keep this up to date as we submit the remaining PRs.

I've separated this into 6 sections, each has some dependency on the previous. Hopefully this gives some clarity to how we've broken things down.

Also these will be added in a few passes:

- Compiler and associated packages.
- CUDA libraries
- Developer tools (debuggers, profilers, etc.)

Compiler

Build Order 1

- [x] cuda-version (Add cuda-version recipe #22053)
- [x] cuda-cudart (Adding cuda-cudart #21723)
- [x] cuda-nvcc (Adding cuda-nvcc recipe #21350)
- [x] cuda-nvrtc (Adding cuda-nvrtc recipe #21951)
- [x] cuda-compat (Adding cuda-compat recipe #21952)
- [x] cuda-cccl (Adding cuda-cccl recipe #21953)
- [x] cuda-cuobjdump (Adding cuda-cuobjdump, cuda-cuxxfilt, and cuda-nvprune recipes #21878)
- [x] cuda-cuxxfilt (Adding cuda-cuobjdump, cuda-cuxxfilt, and cuda-nvprune recipes #21878)
- [x] cuda-nvprune (Adding cuda-cuobjdump, cuda-cuxxfilt, and cuda-nvprune recipes #21878)

Build Order 2

- [x] cuda-compiler (Add cuda-compiler recipe #23003)
- [x] `cuda-nvcc_{{ target_platform }}` (Add `cuda-nvcc_{{ target_platform }}` cuda-nvcc-feedstock#1)

Libraries

Assignees: No one—assign yourself

Labels: None yet

Milestone: No milestone

Development: Create a branch for this issue or link a pull request.

Notifications: Customize

Unsubscribe

You're receiving notifications because you were mentioned.

10 participants

Lock conversation

Unpin issue

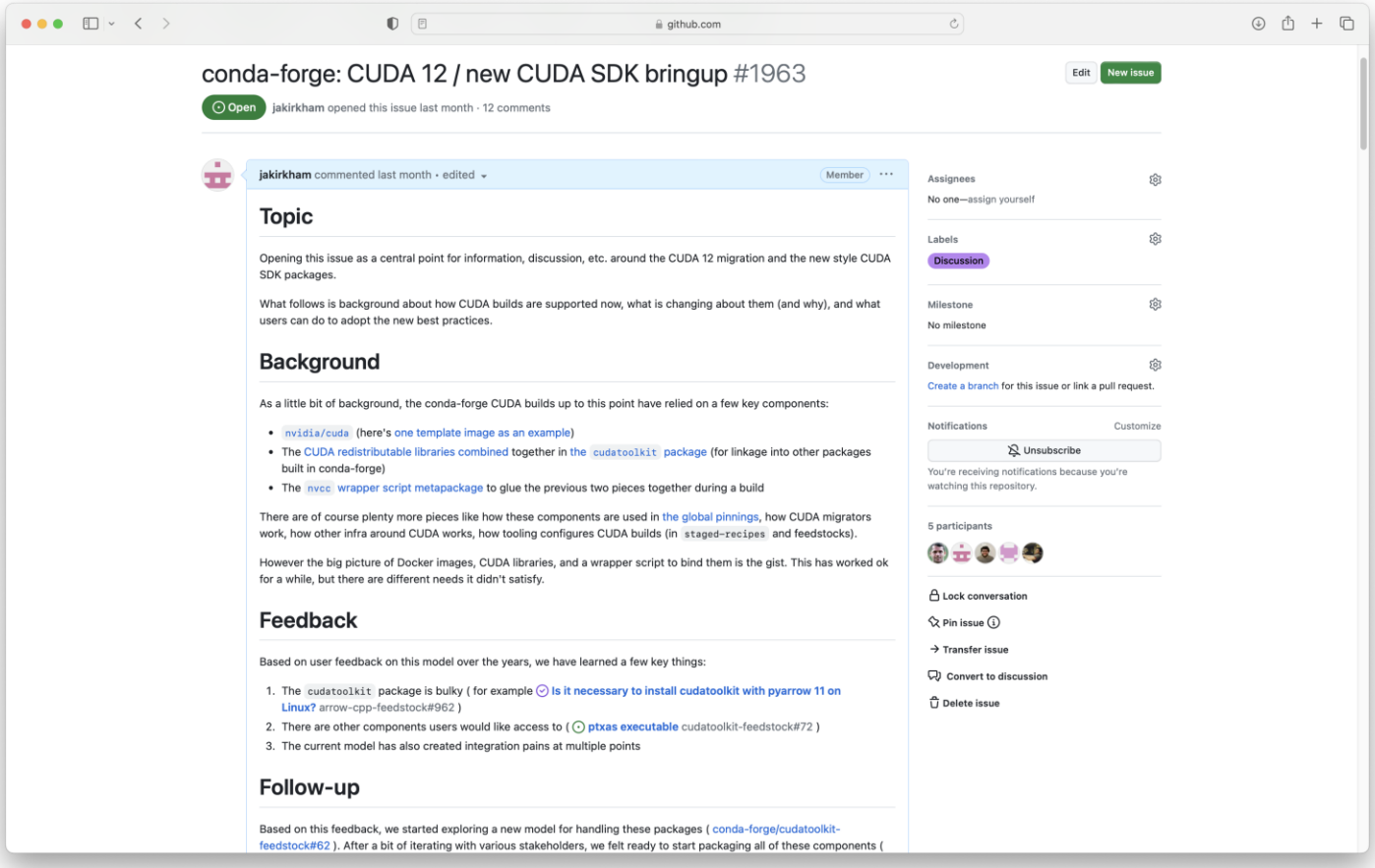
Transfer issue

Delete issue

https://github.com/conda-forge/staged-recipes/issues/21382

Rebuilding of conda-forge packages with new CTK

conda-forge: CUDA 12 / new SDK bringup



github.com

conda-forge: CUDA 12 / new CUDA SDK bringup #1963

Edit | New issue

Open | jakirkham opened this issue last month · 12 comments

jakirkham commented last month · edited | Member

Topic

Opening this issue as a central point for information, discussion, etc. around the CUDA 12 migration and the new style CUDA SDK packages.

What follows is background about how CUDA builds are supported now, what is changing about them (and why), and what users can do to adopt the new best practices.

Background

As a little bit of background, the conda-forge CUDA builds up to this point have relied on a few key components:

- `nvidia/cuda` (here's one template image as an example)
- The CUDA redistributable libraries combined together in the `cudatoolkit` package (for linkage into other packages built in conda-forge)
- The `nvcc` wrapper script metapackage to glue the previous two pieces together during a build

There are of course plenty more pieces like how these components are used in the global pinnings, how CUDA migrators work, how other infra around CUDA works, how tooling configures CUDA builds (in `staged-recipes` and feedstocks).

However the big picture of Docker images, CUDA libraries, and a wrapper script to bind them is the gist. This has worked ok for a while, but there are different needs it didn't satisfy.

Feedback

Based on user feedback on this model over the years, we have learned a few key things:

1. The `cudatoolkit` package is bulky (for example Is it necessary to install cudatoolkit with pyarrow 11 on Linux? arrow-cpp-feedstock#962)
2. There are other components users would like access to (ptxas executable cudatoolkit-feedstock#72)
3. The current model has also created integration pains at multiple points

Follow-up

Based on this feedback, we started exploring a new model for handling these packages (conda-forge/cudatoolkit-feedstock#62). After a bit of iterating with various stakeholders, we felt ready to start packaging all of these components (

Assignees: No one—assign yourself

Labels: Discussion

Milestone: No milestone

Development: Create a branch for this issue or link a pull request.

Notifications: Customize — Unsubscribe — You're receiving notifications because you're watching this repository.

5 participants

Lock conversation

Pin issue

Transfer issue

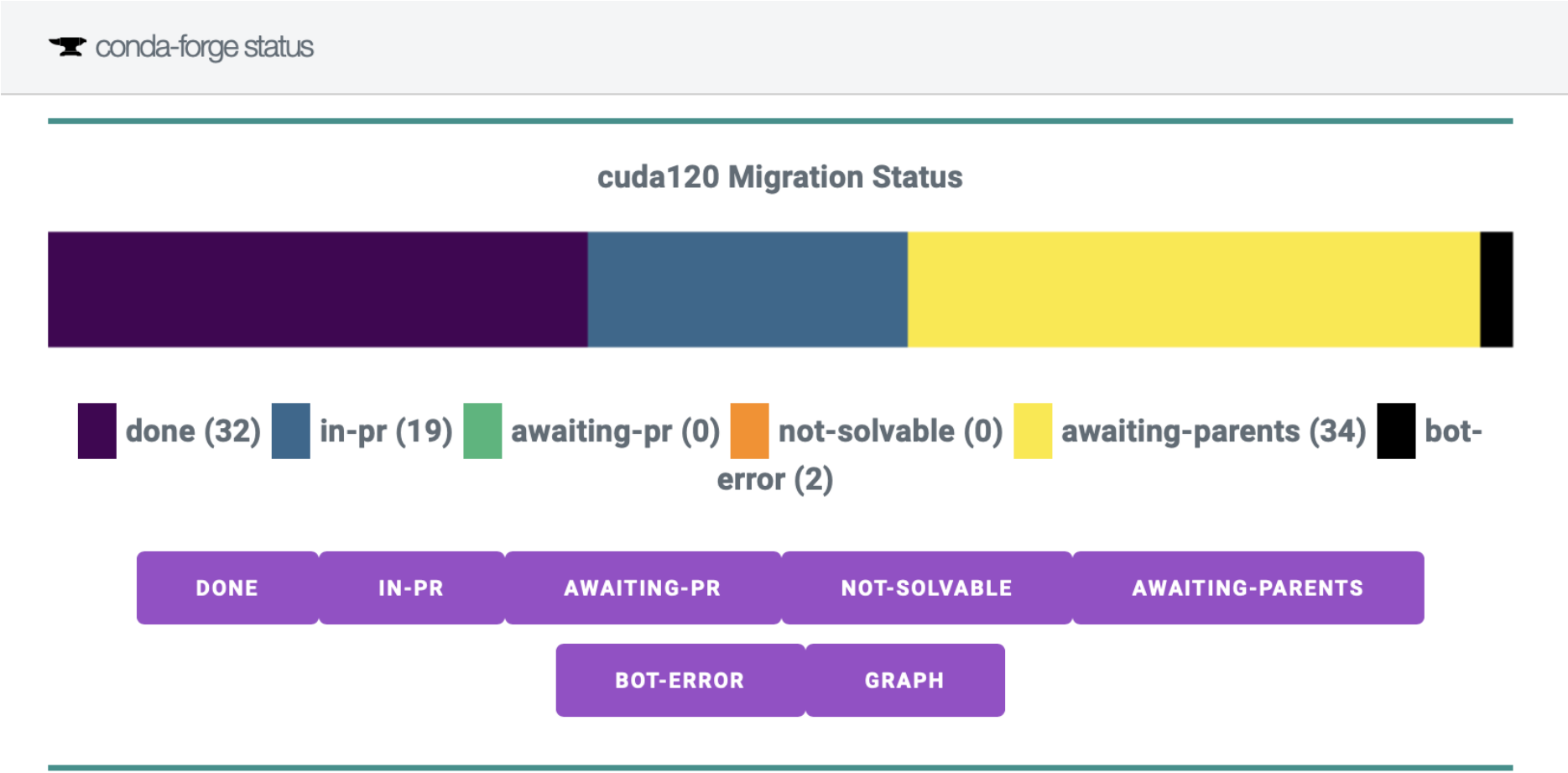
Convert to discussion

Delete issue

https://github.com/conda-forge/conda-forge.github.io/issues/1963

Conda-forge CUDA 12 migration

https://conda-forge.org/status/#cuda120



Next steps

- Close out CUDA 12 migration
- Add new conda-forge CUDA documentation
- Update CTK packages for CUDA 12.1+
- Collect user feedback
- Evaluate next steps

Thank you!

@rlratzel
@thomcom
@jakirkham