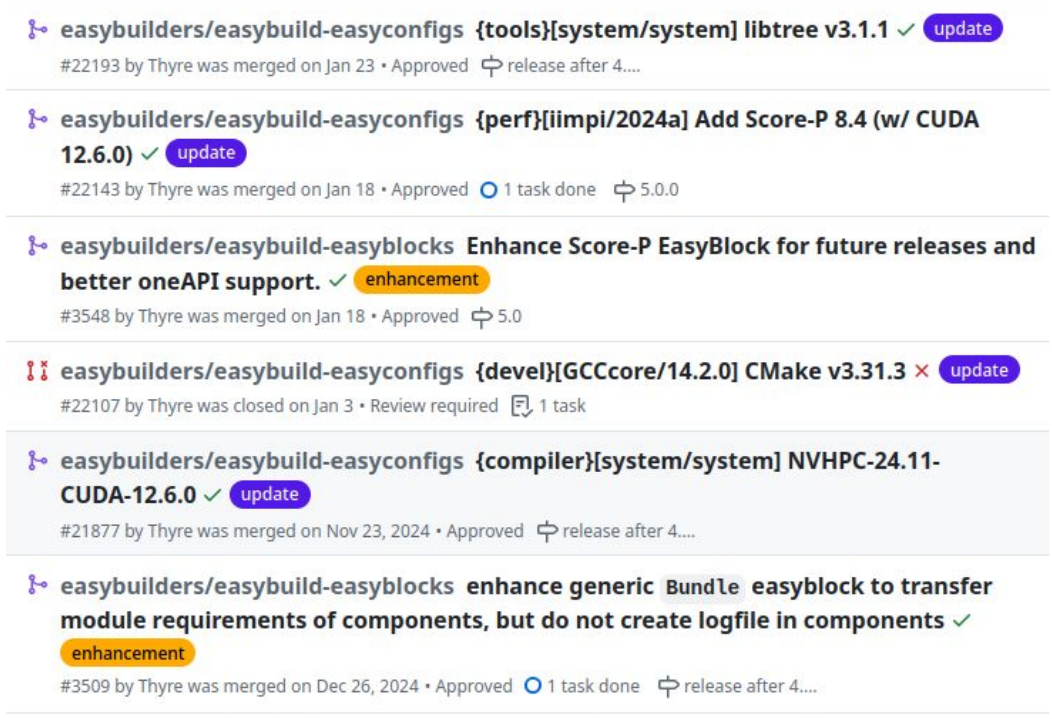
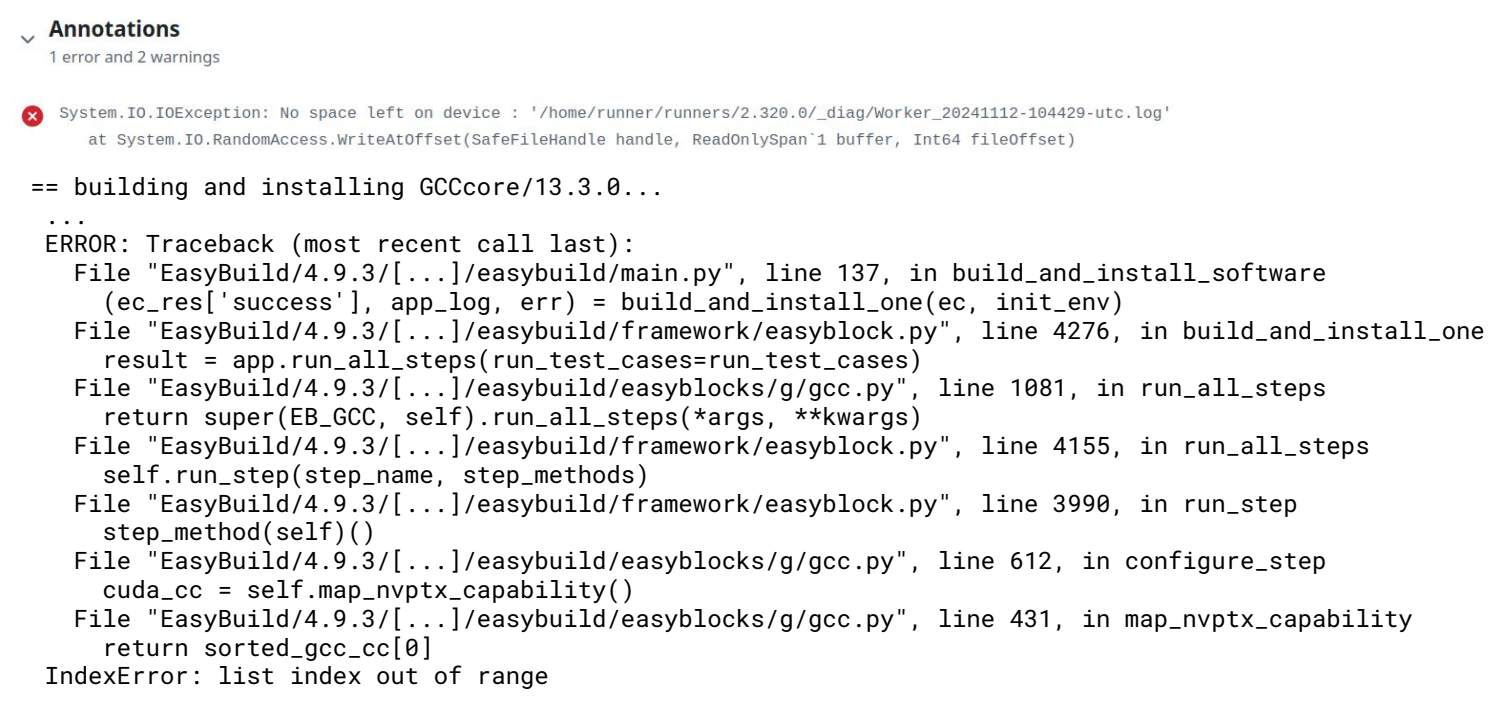




10th EasyBuild User Meeting

Getting Started as EasyBuild Contributor

2025-03-25 I Jan André Reuter

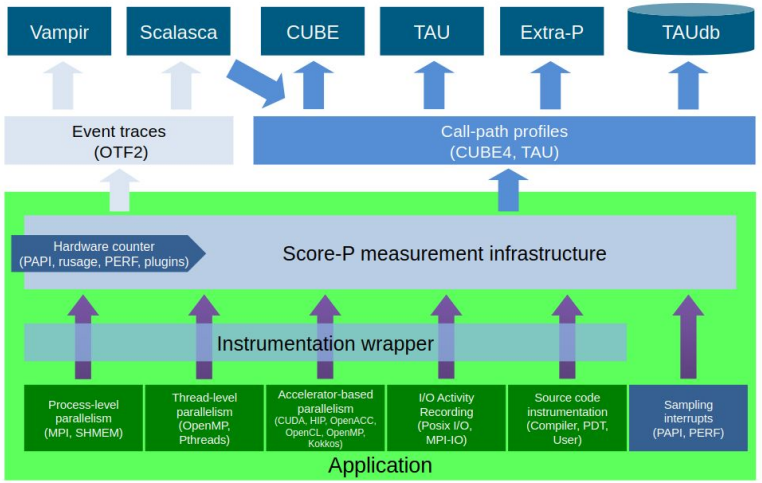
EASYBUILD.io
building software with ease

WHY DO I USE EASYBUILD?

Score-P
Scalable performance measurement infrastructure for parallel codes

- Software Developer for the performance measurement infrastructure **Score-P**
  - Main work focus: OpenMP, accelerators & compiler instrumentation

- Developing involves a lot of testing for different software versions, e.g.:
  - LLVM IR changes
  - New CUDA features
  - Changes in the OpenMP Tools Interface
  - …



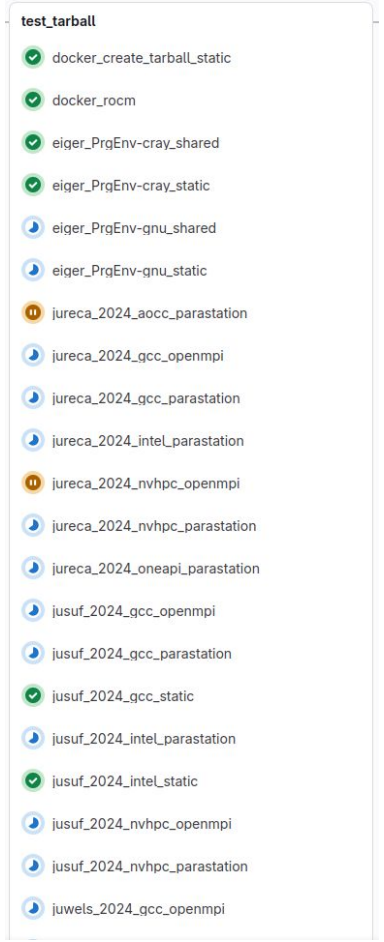
WHY DO I USE EASYBUILD?

Score-P
Scalable performance measurement infrastructure for parallel codes

- Software Developer for the performance measurement infrastructure **Score-P**
  - Main work focus: OpenMP, accelerators & compiler instrumentation

- Developing involves a lot of testing for different software versions, e.g.:
  - LLVM IR changes
  - New CUDA features
  - Changes in the OpenMP Tools Interface
  - …

- We have a CI for that, but…
  - HPC systems do have a limited amount of versions available
  - Doesn't really help for local development



JÜLICH Forschungszentrum

WHY DO I USE EASYBUILD?

Score-P
Scalable performance measurement infrastructure for parallel codes

- Software Developer for the performance measurement infrastructure **Score-P**
  - Main work focus: OpenMP, accelerators & compiler instrumentation

- **Solution: Have software locally available**
  - **EasyBuild:** Flexible, easily reproducible, and similar to software on HPC systems
  - In case where no EasyConfigs exist: Manual build…

```
------------------------
 CUDA:
------------------------
    Versions:
       CUDA/11.7.0
       CUDA/11.8.0
       CUDA/12.0.0
       CUDA/12.1.1
       CUDA/12.2.2
       CUDA/12.3.2
       CUDA/12.4.0
       CUDA/12.5.0
       CUDA/12.6.0
       CUDA/12.8.0
```

```
------------ Core Compilers ------------
  AOCC/5.0                  ROCm-LLVM/5.7.0
  Clang/trunk               ROCm-LLVM/6.0.2
  Clang/11.1.0              ROCm-LLVM/6.0.3
  Clang/12.0.1              ROCm-LLVM/6.1.2
  Clang/13.0.1              ROCm-LLVM/6.2.0
  Clang/14.0.6              ROCm-LLVM/6.3.0
  Clang/15.0.7              aomp/18.0-0
  Clang/16.0.6              aomp/18.0-1
  Clang/17.0.6              aomp/19.0-0
  Clang/18.1.8              aomp/19.0-2
  Clang/19.1.0              aomp/19.0-3
  Clang/20.1.1              aomp/20.0-0
```

NAIVE TRIES TO USE EASYBUILD…

- First tries to use EasyBuild on non-HPC system went… not well
- Unconventional OS (Arch Linux) not expected
- OS packages are picked up sometimes
  - Containers *can* help
- EasyBuild tutorial helped a lot for using it:
  - https://tutorial.easybuild.io/
- Started digging through EasyBuild & EasyConfigs to work around issues I ran into
- Lead to contributing fixes & local EasyConfigs

Some `GCCcore` versions failing when building `nvptx`

```
make[1]: Entering directory '/data/EasyBuild-develop/build/GCCcore/12.3.0/system-system/gcc-12.3.0/build-nvptx-gcc'
mkdir -p -- ./fixincludes
mkdir -p -- ./libiberty
mkdir -p -- ./intl
mkdir -p -- ./gmp
mkdir -p -- ./zlib
Configuring in ./fixinc (took 28 mins 0 secs)
== Results of the build can be found in the log file(s) /tmp/eb-bbqft5x3/easybuild-GCCcore-12.3.0-20250227.184558.pgzBE.log

ERROR: Build of /data/EasyBuild/easybuild/easyconfigs/g/GCCcore/GCCcore-12.3.0.eb failed (err: 'build failed (first 300 chars): cmd " make
  -j 8 " exited with exit code 2 and output:\nmake[1]: Entering directory \'/data/EasyBuild-develop/build/GCCcore/12.3.0/system-system/gcc
-12.3.0/build-nvptx-gcc\'\nmkdir -p -- ./fixincludes\nmkdir -p -- ./libiberty\nmkdir -p -- ./intl\nmkdir -p -- ./gmp\nmkdir -p -- ./zlib\n
Configuring in ./fixinc')
```

System dependencies not detected, as Arch Linux is not handled

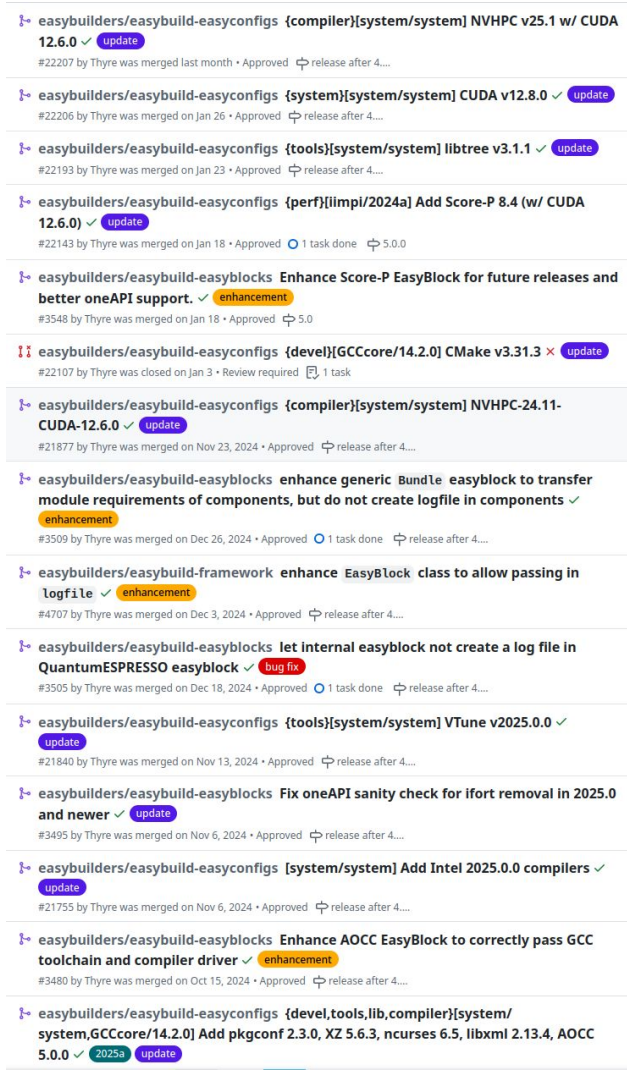
```
 jreuter@Linux  ~  eb --rebuild --installpath /tmp UCX-1.16.0-GCCcore-13.3.0.eb
== Temporary log file in case of crash /tmp/eb-zbvb8tif/easybuild-zyx96w6v.log
ERROR: Failed to process easyconfig /data/EasyBuild/easybuild/easyconfigs/u/UCX/UCX-1.16.0-GCCcore-13.3.0.eb: One or more OS dependencies
were not found: [('libibverbs-dev', 'libibverbs-devel', 'rdma-core-devel')]
 ✗ jreuter@Linux  ~  pacman -Ss libibverbs
cachyos-extra-znver4/rdma-core 56.0-1.1 [installed]
    RDMA core userspace libraries and daemons
extra/rdma-core 56.0-1 [installed: 56.0-1.1]
    RDMA core userspace libraries and daemons
 jreuter@Linux  ~  find /usr -name "libibverbs.so"
/usr/lib/libibverbs.so
```

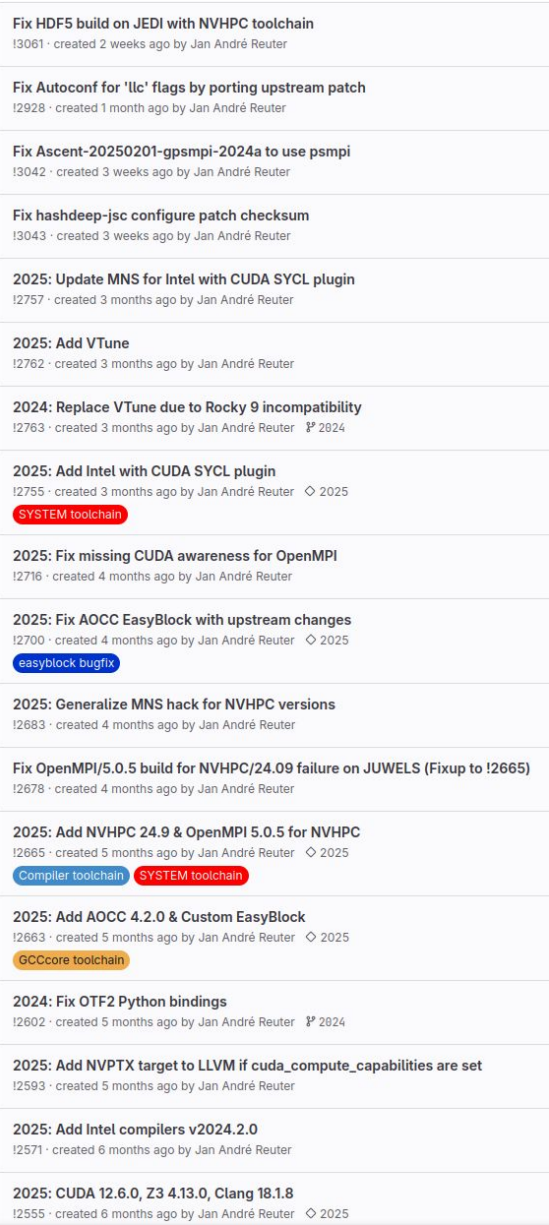
Smaller contributions (Upstream & JSC)

- Main focus & interests:
  - Compilers (GCC, LLVM, AOCC, …)
  - General accelerator support
  - Score-P, Scalasca and other performance tools

- For new software:
  - Used `eb --search` to find if ECs for dependencies exist (also checking PRs)
  - Create EasyConfig step by step

- For updating software:
  - Manual update, or
  - Use of `--try-toolchain --experimental --try-update-deps`





THE FIRST MAJOR CONTRIBUTION

easybuild-easyblocks#3396 : enhance custom easyblock for GCC to use `with-arch` option for nvptx with 13.1+

GCC 12.3.0 on JURECA-DC (AMD EPYC + NVIDIA A100):

```
[reuter1@jrlogin07 ~]$ gcc -fopenmp saxpy.c
ptxas fatal   : Value 'sm_35' is not defined for option 'gpu-name'
nvptx-as: ptxas returned 255 exit status
mkoffload: fatal error: x86_64-pc-linux-gnu-accel-nvptx-none-gcc returned 1 exit status
compilation terminated.
lto-wrapper: fatal error: /p/software/fs/jurecadc/stages/2024/software/GCCcore/12.3.0/bin/../libexec/gcc/x86_64-pc-linux-gnu/12.3.0//accel/nvptx-none/mkoffload returned 1 exit status
compilation terminated.
/p/software/jurecadc/stages/2024/software/binutils/2.40-GCCcore-12.3.0/bin/ld: error: lto-wrapper failed
collect2: error: ld returned 1 exit status
```

- Build issues when trying to use offloading in GCC
- Found release entry for GCC 13:

  The default value for the -march option can be now changed when building GCC using the --with-arch= configure option. GCC's target libraries are then build both with sm_30 and the specified target architecture. If not specified, GCC defaults to sm_30.

- Idea: Let's use it when building GCC 13.1+

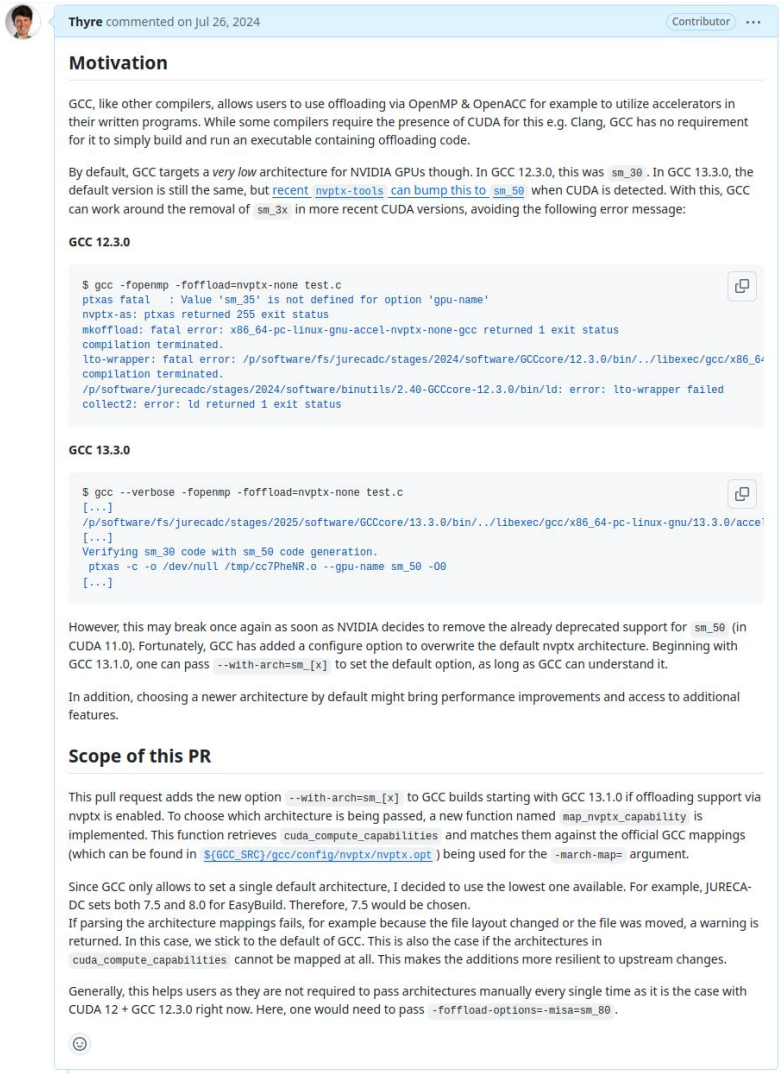
GCC 13.3.0 on JEDI (NVIDIA GH200):

```
[reuter1@jpblt-s01-01 jureca]$ gcc -fopenmp saxpy.c
lto1: internal compiler error: bytecode stream: string too long for the string table
0x6c35ff string_for_index
        ../../../gcc/data-streamer-in.cc:53
0x6c35ff bp_unpack_indexed_string(data_in*, bitpack_d*, unsigned int*)
        ../../../gcc/data-streamer-in.cc:97
0x935957 lto_input_mode_table(lto_file_decl_data*)
        ../../../gcc/lto-streamer-in.cc:2061
0x6380db lto_file_finalize
        ../../../gcc/lto/lto-common.cc:2275
0x6380db lto_create_files_from_ids
        ../../../gcc/lto/lto-common.cc:2298
0x6380db lto_file_read
        ../../../gcc/lto/lto-common.cc:2353
0x6380db read_cgraph_and_symbols(unsigned int, char const**)
        ../../../gcc/lto/lto-common.cc:2801
0x625403 lto_main()
        ../../../gcc/lto/lto.cc:654
Please submit a full bug report, with preprocessed source (by using -freport-bug).
Please include the complete backtrace with any bug report.
See <https://gcc.gnu.org/bugs/> for instructions.
nvptx mkoffload: fatal error: aarch64-unknown-linux-gnu-accel-nvptx-none-gcc returned 1 exit status
compilation terminated.
```

THE FIRST MAJOR CONTRIBUTION

easybuild-easyblocks#3396 : enhance custom easyblock for GCC to use `with-arch` option for nvptx with 13.1+

```
+    def map_nvptx_capability(self):
+        """
+        Convert PTX ISA architecture passed via EasyBuild configs to a version which is understood by GCC.
+        Valid architecture strings include 'sm_30', 'sm_35', 'sm_53', 'sm_70', 'sm_75', 'sm_80'.
+        (as of GCC 14.1.0). Some additional architectures may be mapped.
+        See: https://github.com/gcc-mirror/gcc/commit/de0ef04419e90eacf0d1ddb265552a1b08c18d4b
+
+        As this list is updated regularly, try to parse the GCC source file (gcc/config/nvptx/nvptx.opt)
+        and extract the supported architectures and their mapping. Based on the result, determine the lowest
+        architecture required to support all 'cuda_compute_capabilities' and return this value.
+        """
+        cuda_cc_list = build_option('cuda_compute_capabilities') or self.cfg['cuda_compute_capabilities']
+        architecture_mappings_file = os.path.join(self.cfg['start_dir'], 'gcc', 'config', 'nvptx', 'nvptx.opt')
+        architecture_mappings_flag = "march-map="
+        architecture_mappings_replacement = "misa=,"
+
+        # Determine which compute capabilities are configured. If there are none, return immediately.
+        if cuda_cc_list is None:
+            return None
+        cuda_sm_list = [f"sm_{cc.replace('.', '')}" for cc in cuda_cc_list]
+
+        if not os.path.exists(architecture_mappings_file):
+            warn_msg = f"Tried to parse nvptx.opt but file {architecture_mappings_file} was not found. " \
+                       "Please check the path and update the EasyBlock if necessary!"
+            self.log.warning(warn_msg)
+            return None
+
+        # We want to read the mappings found in the GCC sources and create a map for this.
+        # We're searching for the following pattern:
+        # march-map=sm_32
+        # Target RejectNegative Alias(misa=,sm_30)
+        gcc_architecture_mappings = {}
+        file_content = read_file(architecture_mappings_file).splitlines()
+        for line_idx, line in enumerate(file_content):
+            line = line.strip()
+            if line.startswith(architecture_mappings_flag):
+                key = line.split('=')[1]
+                # Mapped architecture can be found in the following line
+                line = file_content[line_idx + 1]
+                if architecture_mappings_replacement not in line:
+                    warn_msg = "Tried to parse nvptx.opt but failed to extract mapped architectures! " \
+                               f"Expected to find substring '{architecture_mappings_replacement}' in " \
+                               f"line {line_idx + 1} but found '{line}'. Choosing default of GCC."
+                    self.log.warning(warn_msg)
+                    # Bail out, since results of mapping cannot be trusted
+                    return None
+                value = line.split(architecture_mappings_replacement)[1].rstrip(')')
+                gcc_architecture_mappings[key] = value
+        self.log.info(f"Available architecture mappings in GCC {self.version}:\n {str(gcc_architecture_mappings)}")
+
+        # Map compute capabilities to GCC ones
+        # If no compute capability can be mapped, stick to default of GCC and return None
+        gcc_cc = [gcc_architecture_mappings[cc] if cc in gcc_architecture_mappings else None for cc in cuda_sm_list]
+        self.log.info(f"Mapped architectures: {str(cuda_sm_list)} -> {str(gcc_cc)}")
+        if any(cc is None for cc in gcc_cc):
+            self.log.info("At least one architecture could not be mapped. Choosing default of GCC.")
+            return None
+
+        # Get the lowest architecture mapping and return it
+        sorted_gcc_cc = sorted(gcc_cc)
+        self.log.info("Choosing first architecture in sorted list as default nvptx "
+                      f"architecture: {str(sorted_gcc_cc)}")
+        return sorted_gcc_cc[0]
+
```



Thyre commented on Jul 26, 2024 — Contributor

Motivation

GCC, like other compilers, allows users to use offloading via OpenMP & OpenACC for example to utilize accelerators in their written programs. While some compilers require the presence of CUDA for this e.g. Clang, GCC has no requirement for it to simply build and run an executable containing offloading code.

By default, GCC targets a *very low* architecture for NVIDIA GPUs though. In GCC 12.3.0, this was `sm_30`. In GCC 13.3.0, the default version is still the same, but recent `nvptx-tools` can bump this to `sm_50` when CUDA is detected. With this, GCC can work around the removal of `sm_3x` in more recent CUDA versions, avoiding the following error message:

GCC 12.3.0

```
$ gcc -fopenmp -foffload=nvptx-none test.c
ptxas fatal   : Value 'sm_35' is not defined for option 'gpu-name'
nvptx-as: ptxas returned 255 exit status
mkoffload: fatal error: x86_64-pc-linux-gnu-accel-nvptx-none-gcc returned 1 exit status
compilation terminated.
lto-wrapper: fatal error: /p/software/fs/jurecadc/stages/2024/software/GCCcore/12.3.0/bin/../libexec/gcc/x86_64
compilation terminated.
/p/software/jurecadc/stages/2024/software/binutils/2.40-GCCcore-12.3.0/bin/ld: error: lto-wrapper failed
collect2: error: ld returned 1 exit status
```

GCC 13.3.0

```
$ gcc --verbose -fopenmp -foffload=nvptx-none test.c
[...]
/p/software/fs/jurecadc/stages/2025/software/GCCcore/13.3.0/bin/../libexec/gcc/x86_64-pc-linux-gnu/13.3.0/acce
[...]
Verifying sm_30 code with sm_50 code generation.
 ptxas -c -o /dev/null /tmp/cc7PheNR.o --gpu-name sm_50 -O0
[...]
```

However, this may break once again as soon as NVIDIA decides to remove the already deprecated support for `sm_50` (in CUDA 11.0). Fortunately, GCC has added a configure option to overwrite the default nvptx architecture. Beginning with GCC 13.1.0, one can pass `--with-arch=sm_[x]` to set the default option, as long as GCC can understand it.

In addition, choosing a newer architecture by default might bring performance improvements and access to additional features.

Scope of this PR

This pull request adds the new option `--with-arch=sm_[x]` to GCC builds starting with GCC 13.1.0 if offloading support via nvptx is enabled. To choose which architecture is being passed, a new function named `map_nvptx_capability` is implemented. This function retrieves `cuda_compute_capabilities` and matches them against the official GCC mappings (which can be found in `${GCC_SRC}/gcc/config/nvptx/nvptx.opt`) being used for the `-march-map=` argument.

Since GCC only allows to set a single default architecture, I decided to use the lowest one available. For example, JURECA-DC sets both 7.5 and 8.0 for EasyBuild. Therefore, 7.5 would be chosen.
If parsing the architecture mappings fails, for example because the file layout changed or the file was moved, a warning is returned. In this case, we stick to the default of GCC. This is also the case if the architectures in `cuda_compute_capabilities` cannot be mapped at all. This makes the additions more resilient to upstream changes.

Generally, this helps users as they are not required to pass architectures manually every single time as it is the case with CUDA 12 + GCC 12.3.0 right now. Here, one would need to pass `-foffload-options=-misa=sm_80`.

- 1.5 months in review
- Many additional safety nets added
- Tested on jsc-zen3, other JSC machines & by reviewers
- Merged before EB 4.9.3

BREAKING AN EASYBUILD RELEASE

EasyBuild 4.9.3 came out on September 14th 2024.
There was one small issue though…

```
== building and installing GCCcore/13.3.0...
 ...
 ERROR: Traceback (most recent call last):
   File "EasyBuild/4.9.3/[...]/easybuild/main.py", line 137, in build_and_install_software
     (ec_res['success'], app_log, err) = build_and_install_one(ec, init_env)
   File "EasyBuild/4.9.3/[...]/easybuild/framework/easyblock.py", line 4276, in build_and_install_one
     result = app.run_all_steps(run_test_cases=run_test_cases)
   File "EasyBuild/4.9.3/[...]/easybuild/easyblocks/g/gcc.py", line 1081, in run_all_steps
     return super(EB_GCC, self).run_all_steps(*args, **kwargs)
   File "EasyBuild/4.9.3/[...]/easybuild/framework/easyblock.py", line 4155, in run_all_steps
     self.run_step(step_name, step_methods)
   File "EasyBuild/4.9.3/[...]/easybuild/framework/easyblock.py", line 3990, in run_step
     step_method(self)()
   File "EasyBuild/4.9.3/[...]/easybuild/easyblocks/g/gcc.py", line 612, in configure_step
     cuda_cc = self.map_nvptx_capability()
   File "EasyBuild/4.9.3/[...]/easybuild/easyblocks/g/gcc.py", line 431, in map_nvptx_capability
     return sorted_gcc_cc[0]
 IndexError: list index out of range
```

Overview of tested easyconfigs (in order)

- **SUCCESS** *GCCcore-10.2.0.eb*
- **SUCCESS** *GCCcore-12.3.0.eb*
- **SUCCESS** *GCCcore-14.2.0.eb*

EasyBuild info

- easybuild-framework version: 4.9.3.dev0-re0e04fb773a5cee26669211f563696da94455cfd
- easybuild-easyblocks version: 4.9.3.dev0-rc29c16dd9da0b9935a7fa4f041aacf0a76102453
- command line:

```
d-test-report --download-timeout=1000 GCCcore-10.2.0.eb GCCcore-12.3.0.eb GCCcore-14.2.0.eb --installpath /tmp
```

- full configuration (includes defaults):

```
--accept-eula-for='.*'
--accept-eula=''
--allow-loaded-modules='EasyBuild'
--buildpath='/tmp/boegelbot'
--check-ebroot-env-vars='warn'
--cleanup-builddir
--cleanup-easyconfigs
--cleanup-tmpdir
--color='auto'
--container-type='singularity'
--containerpath='/project/def-maintainers/boegelbot/rocky9/zen3/containers'
--cuda-compute-capabilities='8.0'
```

Building GCC 13.1+ failed without a CUDA architecture.
All tests were done with a CUDA architecture.

BREAKING AN EASYBUILD RELEASE

2 easybuild/easyblocks/g/gcc.py

```
369 369     def map_nvptx_capability(self):
370 370         """
371 371         Convert PTX ISA architecture passed via EasyBuild configs to a version which is understood by GCC.
372 372         Valid architecture strings include 'sm_30', 'sm_35', 'sm_53', 'sm_70', 'sm_75', 'sm_80'.
373 373         (as of GCC 14.1.0). Some additional architectures may be mapped.
374 374         See: https://github.com/gcc-mirror/gcc/commit/de0ef04419e90eacf0d1ddb265552a1b08c18d4b
375 375
376 376         As this list is updated regularly, try to parse the GCC source file (gcc/config/nvptx/nvptx.opt)
377 377         and extract the supported architectures and their mapping. Based on the result, determine the lowest
378 378         architecture required to support all 'cuda_compute_capabilities' and return this value.
379 379         """
380 380         cuda_cc_list = build_option('cuda_compute_capabilities') or self.cfg['cuda_compute_capabilities']
381 381         architecture_mappings_file = os.path.join(self.cfg['start_dir'], 'gcc', 'config', 'nvptx', 'nvptx.opt')
382 382         architecture_mappings_flag = "march-map="
383 383         architecture_mappings_replacement = "misa=,"
384 384
385 385         # Determine which compute capabilities are configured. If there are none, return immediately.
386     -       if cuda_cc_list is None:
    386 +       if not cuda_cc_list:
387 387             return None
388 388         cuda_sm_list = [f"sm_{cc.replace('.', '')}" for cc in cuda_cc_list]
389 389
```

Thyre commented on Sep 16, 2024 Contributor

When no compute capability is set but NVPTX is enabled, trying to figure out the NVPTX architecture fails with the error:

```
File "easybuild/easyblocks/g/gcc.py", line 431, in map_nvptx_capability
  return sorted_gcc_cc[0]
IndexError: list index out of range
```

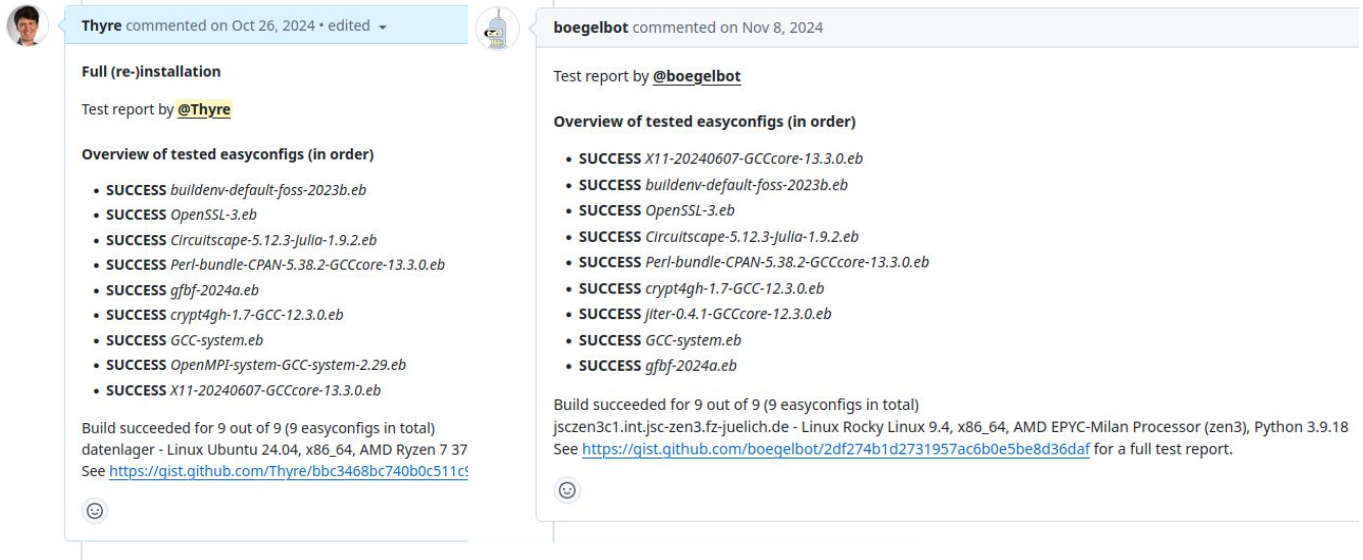
The error occurs because of an insufficient check for an unset CUDA compute capability. This commit changes the checked conditions, so that empty lists are also correctly handled.

Key takeaway:
Test EasyBlock changes with and without `cuda-compute-capabilities`

BREAKING THE EASYBUILD TEST SUITE

easybuild-easyblocks#3472 : Enhance generic `Bundle` easyblock to transfer module requirements of components

- **Idea:** Provide EasyConfig for Intel oneAPI with SYCL plug-in for NVIDIA GPUs (`easybuild-easyconfigs#21582`)
- Using Bundle EasyBlock failed: Additional module requirements not transferred…
- **Culprit:** `make_module_req_guess` was not implemented.
- **Solution:** Copy the requirements from all Bundle components.



Thyre commented on Oct 26, 2024 • edited

Full (re-)installation

Test report by **@Thyre**

Overview of tested easyconfigs (in order)

- **SUCCESS** *buildenv-default-foss-2023b.eb*
- **SUCCESS** *OpenSSL-3.eb*
- **SUCCESS** *Circuitscape-5.12.3-Julia-1.9.2.eb*
- **SUCCESS** *Perl-bundle-CPAN-5.38.2-GCCcore-13.3.0.eb*
- **SUCCESS** *gfbf-2024a.eb*
- **SUCCESS** *crypt4gh-1.7-GCC-12.3.0.eb*
- **SUCCESS** *GCC-system.eb*
- **SUCCESS** *OpenMPI-system-GCC-system-2.29.eb*
- **SUCCESS** *X11-20240607-GCCcore-13.3.0.eb*

Build succeeded for 9 out of 9 (9 easyconfigs in total)
datenlager - Linux Ubuntu 24.04, x86_64, AMD Ryzen 7 37
See https://qist.github.com/Thyre/bbc3468bc740b0c511c

boegelbot commented on Nov 8, 2024

Test report by **@boegelbot**

Overview of tested easyconfigs (in order)

- **SUCCESS** *X11-20240607-GCCcore-13.3.0.eb*
- **SUCCESS** *buildenv-default-foss-2023b.eb*
- **SUCCESS** *OpenSSL-3.eb*
- **SUCCESS** *Circuitscape-5.12.3-Julia-1.9.2.eb*
- **SUCCESS** *Perl-bundle-CPAN-5.38.2-GCCcore-13.3.0.eb*
- **SUCCESS** *crypt4gh-1.7-GCC-12.3.0.eb*
- **SUCCESS** *jiter-0.4.1-GCCcore-12.3.0.eb*
- **SUCCESS** *GCC-system.eb*
- **SUCCESS** *gfbf-2024a.eb*

Build succeeded for 9 out of 9 (9 easyconfigs in total)
jsczen3c1.int.jsc-zen3.fz-juelich.de - Linux Rocky Linux 9.4, x86_64, AMD EPYC-Milan Processor (zen3), Python 3.9.18
See https://qist.github.com/boegelbot/2df274b1d2731957ac6b0e5be8d36daf for a full test report.

```
== Temporary log file in case of crash /tmp/eb-k6c_4924/easybuild-w2k6x1on.log
== processing EasyBuild easyconfig /tmp/eb-k6c_4924/files_pr21582/i/intel-compilers/intel-compi
lers-2024.2.0-CUDA-12.5.0.eb
== building and installing intel-compilers/2024.2.0-CUDA-12.5.0...
== fetching files...
== ... (took 2 secs)
== creating build dir, resetting environment...
== unpacking...
== ... (took 1 secs)
== patching...
== preparing...
== configuring...
== building...
== testing...
== installing...
== installing bundle component intel-compilers v2024.2.0 (1/2)...
== installing part 1/2 (l_dpcpp-cpp-compiler_p_2024.2.0.495_offline.sh)...
== installing part 2/2 (l_fortran-compiler_p_2024.2.0.426_offline.sh)...
== installing bundle component codeplay-oneapi-for-nvidia-gpus v2024.2.0 (2/2)...
== ... (took 55 secs)
== taking care of extensions...
== restore after iterating...
== postprocessing...
== sanity checking...
== FAILED: Installation ended unsuccessfully (build directory: /data/EasyBuild-develop/build/in
telcompilers/2024.2.0/system-system-CUDA-12.5.0): build failed (first 300 chars): Sanity check
failed: sanity check command icx --version exited with code 127 (output: /bin/bash: line 1:
icx: command not found
)
sanity check command icpx --version exited with code 127 (output: /bin/bash: line 1: icpx: comm
and not found
)
sanity check command ifx --version exited with code 127 (ou (took 59 secs)
== Results of the build can be found in the log file(s) /tmp/eb-k6c_4924/easybuild-intel-compil
ers-2024.2.0-20250310.094727.DcrBH.log
```

Custom module paths of oneAPI lost when using Bundle EasyBlock → Intel compilers not found

BREAKING THE EASYBUILD TEST SUITE

easybuild-easyblocks#3472 : Enhance generic `Bundle` easyblock to transfer module requirements of components

Annotations
1 error and 2 warnings

```
System.IO.IOException: No space left on device : '/home/runner/runners/2.320.0/_diag/Worker_20241112-104429-utc.log'
   at System.IO.RandomAccess.WriteAtOffset(SafeFileHandle handle, ReadOnlySpan`1 buffer, Int64 fileOffset)
   at System.IO.Strategies.BufferedFileStreamStrategy.FlushWrite()
   at System.IO.StreamWriter.Flush(Boolean flushStream, Boolean flushEncoder)
   at System.Diagnostics.TextWriterTraceListener.Flush()
   at GitHub.Runner.Common.HostTraceListener.WriteHeader(String source, TraceEventType eventType, Int32 id)
   at GitHub.Runner.Common.HostTraceListener.TraceEvent(TraceEventCache eventCache, String source, TraceEventType eventType, Int32 id, String message)
   at System.Diagnostics.TraceSource.TraceEvent(TraceEventType eventType, Int32 id, String message)
   at GitHub.Runner.Worker.Worker.RunAsync(String pipeIn, String pipeOut)
   at GitHub.Runner.Worker.Program.MainAsync(IHostContext context, String[] args)
System.IO.IOException: No space left on device : '/home/runner/runners/2.320.0/_diag/Worker_20241112-104429-utc.log'
   at System.IO.RandomAccess.WriteAtOffset(SafeFileHandle handle, ReadOnlySpan`1 buffer, Int64 fileOffset)
   at System.IO.Strategies.BufferedFileStreamStrategy.FlushWrite()
   at System.IO.StreamWriter.Flush(Boolean flushStream, Boolean flushEncoder)
   at System.Diagnostics.TextWriterTraceListener.Flush()
   at GitHub.Runner.Common.HostTraceListener.WriteHeader(String source, TraceEventType eventType, Int32 id)
   at GitHub.Runner.Common.HostTraceListener.TraceEvent(TraceEventCache eventCache, String source, TraceEventType eventType, Int32 id, String message)
   at System.Diagnostics.TraceSource.TraceEvent(TraceEventType eventType, Int32 id, String message)
   at GitHub.Runner.Common.Tracing.Error(Exception exception)
   at GitHub.Runner.Worker.Program.MainAsync(IHostContext context, String[] args)
Unhandled exception. System.IO.IOException: No space left on device : '/home/runner/runners/2.320.0/_diag/Worker_20241112-104429-utc.log'
   at System.IO.RandomAccess.WriteAtOffset(SafeFileHandle handle, ReadOnlySpan`1 buffer, Int64 fileOffset)
   at System.IO.Strategies.BufferedFileStreamStrategy.FlushWrite()
   at System.IO.StreamWriter.Flush(Boolean flushStream, Boolean flushEncoder)
   at System.Diagnostics.TextWriterTraceListener.Flush()
   at System.Diagnostics.TraceSource.Flush()
   at GitHub.Runner.Common.TraceManager.Dispose(Boolean disposing)
   at GitHub.Runner.Common.TraceManager.Dispose()
   at GitHub.Runner.Common.HostContext.Dispose(Boolean disposing)
   at GitHub.Runner.Common.HostContext.Dispose()
   at GitHub.Runner.Worker.Program.Main(String[] args)
```

Show less

- All CI jobs started failing due to disk space restriction on runners (14 GB)
- Affected `develop`, `5.x`, and all pull requests
- Changes were reverted quickly

revert changes from PR #3472, seems to be leading to an infinite loop in easyconfigs test suite? #3504

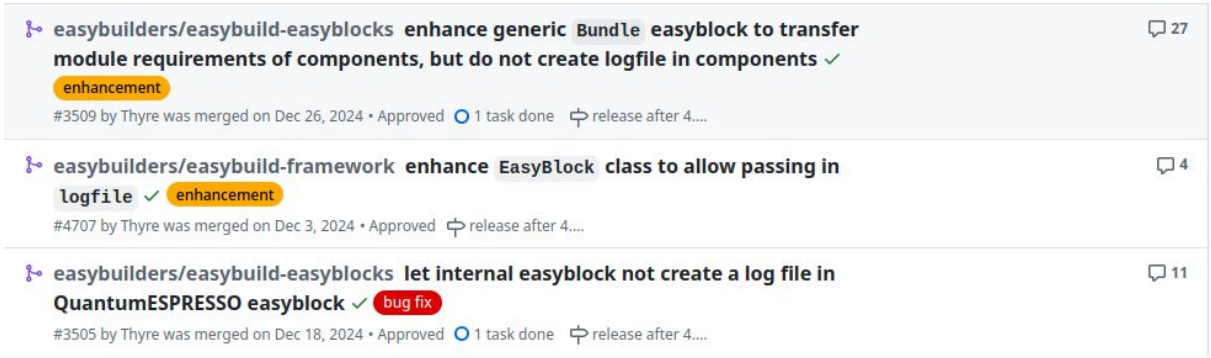
Merged jfgrimm merged 1 commit into easybuilders:develop from boegel:20241112184552_new_pr_bundle on Nov 12, 2024

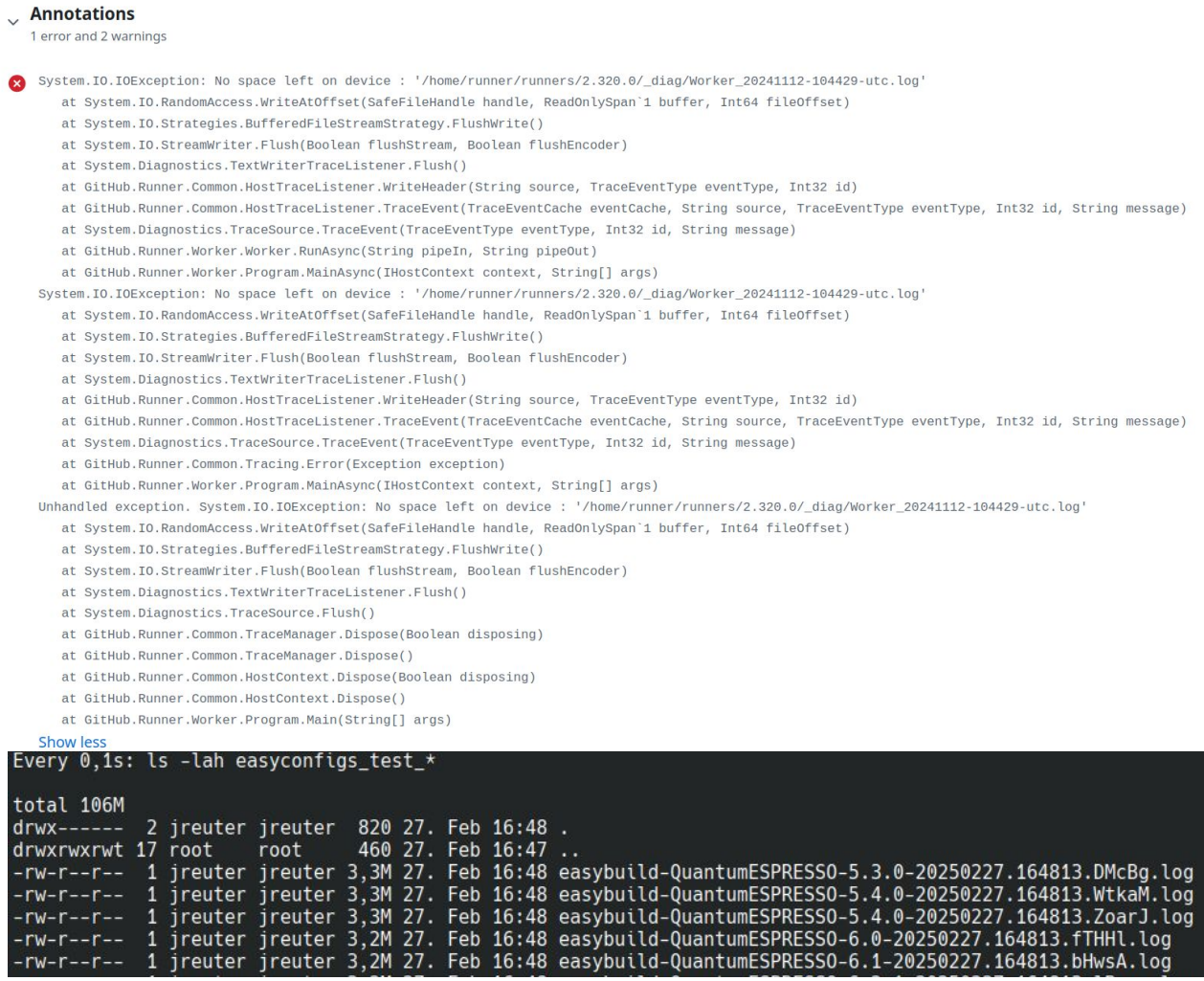
- But what happened?

WHAT HAPPENED?

Late night analysis of what was going wrong

- Bundle EasyBlock was creating multiple log files, but closed only one of them.
- Same issue was visible with QuantumESPRESSO EasyBlock
  - Fortunate enough that this issue was not triggered earlier
- Several follow-up PRs:





KEY TAKEAWAYS

The Good	The Bad
GitHub integration and testing `--new-pr, --upload-test-report`	Documentation, once working with EasyBlocks or Framework, e.g. how logging works
Creating / updating EasyConfigs is nicely documented, and often straight-forward.	Side-effects when changing EasyBlock / Framework. Additional tests necessary?
Great documentation to get started with using EasyBuild	(Long review times)
The large amount of software available!	

Tips:

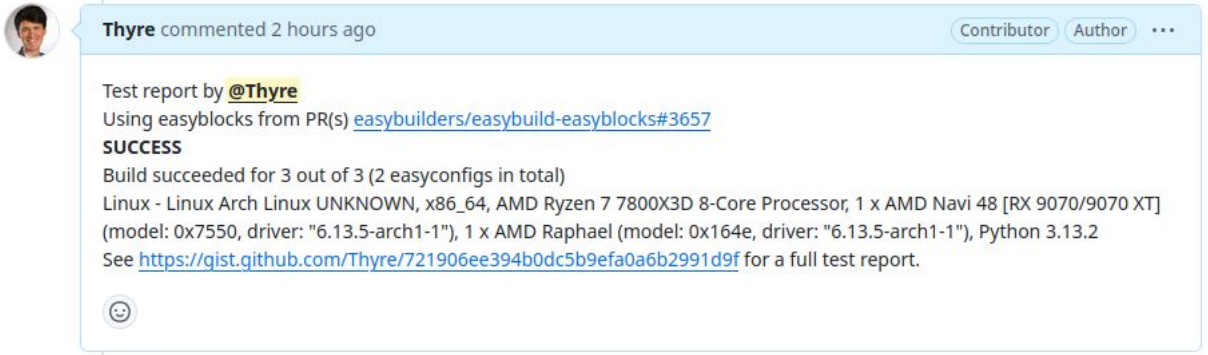
- Join EasyBuild conf. calls
- Don’t hesitate to ask if you’re running into issues!
- Look at logs to find out what failed

PERSONAL WISH LIST

1. Also test `aarch64`:
   a. Several build issues when setting up software stage on aarch64

2. Improved support for AMD GPUs / ROCm:
   a. `cuda-compute-capabilities` only for NVIDIA
   b. No (recent) ROCm packages
   c. Many packages only built for NVIDIA

3. Additional toolchain support:
   a. LLVM/Clang, AOCC, ROCm, …




```
class EB_Score_minus_P_minus_CPE(ConfigureMake):
    def configure_step(self, *args, **kwargs):
        comp_opts = {
            # assume that system toolchain uses a system-provided GCC
            toolchain.SYSTEM:    'gcc',
            toolchain.AMD:       'amdclang',
            toolchain.AOCC:      'aocc',
            toolchain.CCE:       'cray',
            toolchain.GCC:       'gcc',
            toolchain.IBMCOMP:   'ibm',
            toolchain.INTELCOMP: 'intel',
            toolchain.NVHPC:     'nvhpc',
            toolchain.PGI:       'pgi',
        }
```

THANKS FOR YOUR TIME