GROMACS - Feature #1720
Manage GROMACS JIT caching
04/28/2015 06:02 PM - Mark Abraham

Description
Just-in-time compilation seems like part of the present and future of HPC (CUDA uses it behind the scenes already, OpenCL will also use it in GROMACS 5.1, GCC 5 has early support for it). However, it generates binary lumps that need managing. The CUDA approach is documented e.g. at [http://devblogs.nvidia.com/parallelforall/cuda-pro-tip-understand-fat-binaries-jit-caching/](http://devblogs.nvidia.com/parallelforall/cuda-pro-tip-understand-fat-binaries-jit-caching/). There is nothing in the OpenCL standard (or AMD support of it) that helps here.

So, I will copy the CUDA approach in a generic way, and then we can use that for the OpenCL. Roughly, that means supporting using `~/.gromacs/jit_cache/<code_version>/<device_identifier>/<kernel_name>.bin`

Features we need for 5.1:
- cache compiled kernels (to re-use in successive runs)
- work on *nix, windows, mac (along the lines of CUDA, see above link)
- provide a mechanism to prevent misuse of cached kernels compiled for another device or version of GROMACS (e.g. the version and identifiers in the above; discussion below)
- provide CMake-time configuration of the location to use for caching (so sysadmins can choose somewhere to which user programs can write)

Features that are merely nice
- provide run-time environment variable to override the cache location (so users can make things work without admin support, or perhaps devs can re-direct the cache while experimenting without trashing; this will be easy to implement, so may as well have it for 5.1)

Features we probably don't need (or don't need yet)
- support for disabling caching in general (sometimes convenient when developing, and there's already such functionality for the OpenCL case)

Features we can't do
- secure the cached binary against change (mdrun has no way to check anything, except by re-compilation)

Preventing misuse of binary lumps is tricky. We need to anticipate multiple devices, replacement of devices, installed versions of GROMACS. Also developers iterating on multiple code versions, and differences in base compilers, optimization flags, etc. Getting a device ID string at run time is easy, and we should use that for part. We can already use the version string generated in src/gromacs/utility/baseversion-gen.cmake in for part. We could md5 the binary(+libgromacs) as part of the build process, but that md5 then has to be fetched at run time from the build or install tree, since it can't be in the binary (else circular definition). Not great, but I can't think of anything better, unless anyone else can? (CUDA example is no help, since everything under my `~/.nv/ComputeCache` is a binary lump.)

Related issues:
- Related to GROMACS - Feature #1597: OpenCL port - Closed
- Related to GROMACS - Task #1938: work around broken NVIDIA JIT caching - Closed

Associated revisions
Revision b08ab12c - 05/29/2016 01:31 PM - Mark Abraham
- Simplified and updated OpenCL compilation
- Moved into gmx and new ocl namespace, updated variable naming, updated string handling, treated many more error conditions, also with
exceptions, used more RAII, used more of the standard GROMACS utility infrastructure.

Removed some string databases functions that existed merely to be looked up once.

Changed to write OpenCL build log to file pointer provided by the caller, if needed, rather than a separate file. This currently uses stderr, so can't yet work well with multiple ranks, but neither did the old approach. We need a proper MPI-aware logging module, first.

Separated the caching functionality into its own source file. Changed the naming of binary cache to reflect the name of the kernel source file whose binary is being cached. Noted further requirements if we would re-activate caching at some point, but since it is still de-activated, this is not worth further effort now.

Removed the requirement that we must be able to read source code, if instead a binary cache is available.

Required that compileProgram compile kernels for the vendor of the target device. This was always the behaviour, but there is no reason to be able to select alternative things there.

Simplified the passing of preprocessor defines required by the caller of compileProgram to the JIT compilation.

Removed use of GMX_OCL_FORCE_CPU in log file coordination, as CPU OpenCL devices are not supported.

Refs #1720

Change-Id: l25e78526f55715c779819e96d6bf6b52ad9394c6

History

#1 - 04/28/2015 06:02 PM - Mark Abraham
   - Parent task deleted (#1597)

#2 - 04/28/2015 06:03 PM - Mark Abraham
   - Related to Feature #1597: OpenCL port added

#3 - 04/28/2015 06:12 PM - Roland Schulz

What would be the purpose of securing the binary? Do you want a) to prevent malicious change or b) to prevent accidentally picking up the wrong version? If a) then I don't see the point. The main binary isn't protected either. And anyone who can write data to the home directory will have many other ways to get a user to execute code. If b) then wouldn't it be sufficient to write some data (i.e. the version (if git version including git hash)) into the binary and compare that data before using the cached binary?

#4 - 04/29/2015 12:06 AM - Szilárd Páll

Mark Abraham wrote:

Features we probably don't need (or don't need yet)

- support for disabling caching in general (sometimes convenient when developing, and there's already such functionality for the OpenCL case)

Note that the NVIDIA driver supports that for both CUDA and OpenCL (CUDA_CACHE_DISABLE).

Features we can't do

- secure the cached binary against change (mdrun has no way to check anything, except by re-compilation)

I assume you mean change in the compiled kernel code by some external entity. That does not sound too relevant, we don't check linked libraries either.

Having thought about it a bit, I came to the conclusion that the task is to predict that compiling the kernels would result in different binary code ("lump") without actually recompiling, and we'd like to avoid false negatives. Whether recompile is needed is influenced by the change in any of (at least) the following:

- gmx source (or binary?)
- embedded OpenCL kernel source - which is for the moment not changed on-the-fly (so this case is included in the former)
- GPU hardware
- OpenCL driver/runtime

AFAICT the last item is one that you did not mention and is probably the most difficult one, I'm not even sure it is possible to detect it reliably. Additionally, some kind of architecture version for identifying GPUs could be useful (like NVIDIA's "sm_XX"), but this is not of huge importance because users will not be swapping their GPUs with similar ones on a regular basis.

We could md5 the binary(+libgromacs) as part of the build process, but that md5 then has to be fetched at run time from the build or install tree, since it can't be in the binary (else circular definition). Not great, but I can't think of anything better, unless anyone else can?

I don't see a better alternative, it has to be either an external entity (like the NVIDIA GPU driver) that does the verification or mdrun itself has to trust an externally stored hash of its own binary and lbs.

#5 - 04/29/2015 12:22 AM - Mark Abraham
Roland Schulz wrote:

> What would be the purpose of securing the binary? Do you want a) to prevent malicious change or b) to prevent accidentally picking up the wrong version? If a) then I don't see the point. The main binary isn't protected either. And anyone who can write data to the home directory will have many other ways to get a user to execute code. If b) then wouldn't it be sufficient to write some data (i.e. the version (if git version including git hash)) into the binary and compare that data before using the cached binary?

I wasn't very clear, sorry. I don't care at all about securing the binary against some kind of malice, for the kinds of reasons you suggest. I just wanted to be clear that this was not a goal.

b) is a good suggestion to check that the lookup has found something correct. The simplest way to do that would be to find the cached binary, and run a kernel from it at setup time that just verifies that the embedded version string matches one passed in by the host. If it doesn't match, warn, back up the old cached binary, and start JIT. Not exactly cheap, though. Perhaps a good reason to write all JIT-ted code to the same binary? Will have to peruse the OpenCL API to see if there's a good way to do that. (Also, b doesn't help us find the right binary in the first place, of course.)

#6 - 04/29/2015 12:45 AM - Mark Abraham
Szilárd Páll wrote:

Mark Abraham wrote:

Features we probably don't need (or don't need yet)

- support for disabling caching in general (sometimes convenient when developing, and there's already such functionality for the OpenCL case)

Note that the NVIDIA driver supports that for both CUDA and OpenCL (CUDA_CACHE_DISABLE).

Thanks

Features we can't do

- secure the cached binary against change (mdrun has no way to check anything, except by re-compilation)

I assume you mean change in the compiled kernel code by some external entity. That does not sound too relevant, we don't check linked libraries either.

Ja

Having thought about it a bit, I came to the conclusion that the task is to predict that compiling the kernels would result in different binary code ("lump") without actually recompiling, and we'd like to avoid false negatives. Whether recompile is needed is influenced by the change in any of (at least) the following:

- gmx source (or binary?)
- embedded OpenCL kernel source - which is for the moment not changed on-the-fly (so this case is included in the former)
- GPU hardware
- OpenCL driver/runtime

Indeed. Also (base and wrapper) compiler version and options (e.g. switch Debug to Release from same build tree). For extra fun, environment variables that configure the behaviour of any of the wrapper compilers involved...

AFAICT the last item is one that you did not mention and is probably the most difficult one, I'm not even sure it is possible to detect it reliably.
Good point. I would expect the standard to include API calls that give that kind of information to the calling code, but I guess I'll find out when I look.

All considered, we can't anticipate all of the above different kinds of things in the general case, so we should pick a few things (such as GROMACS version, device type as already suggested) and then rely on somehow checking the md5 of the gmx that did the JIT matches that of the gmx trying to use it. If so, a user trying to benchmark (say) two versions of gcc (or GPU drivers, or OpenCL runtimes, etc.) and have JIT caching work is going to need to get involved in helping keep the caches separate (e.g. configuring those two builds to use different cache locations). That seems reasonable to me.

Additionally, some kind of architecture version for identifying GPUs could be useful.

Yeah, but I think we can leave that for later.

We could md5 the binary(+libgromacs) as part of the build process, but that md5 then has to be fetched at run time from the build or install tree, since it can't be in the binary (else circular definition). Not great, but I can't think of anything better, unless anyone else can?

I don't see a better alternative, it has to be either an external entity (like the NVIDIA GPU driver) that does the verification or mdrun itself has to trust an externally stored hash of its own binary and libs.

OK

#8 - 06/19/2015 09:00 AM - Erik Lindahl
- Target version changed from 5.1 to future

Given that we don't even have any commit for this yet, it's realistically not going to be in 5.1.

#9 - 06/22/2015 04:35 PM - Mark Abraham

Erik Lindahl wrote:

Given that we don't even have any commit for this yet, it's realistically not going to be in 5.1.

I had it half done a month ago, and am finishing minimal version of it now. This is a potential correctness issue - we can't blindly dump cached binary artifacts to users' working directories and then perhaps need to issue a bug-fix release that cannot work unless the user removes those files.

#10 - 06/22/2015 06:03 PM - Gerrit Code Review Bot
Gerrit received a related patchset '1' for Issue #1720.
Uploader: Mark Abraham (mark.j.abraham@gmail.com)
Change-Id: Ib3e462179376e202578490a793a22c13c27b6e05
Gerrit URL: https://gerrit.gromacs.org/4766

#11 - 06/30/2015 08:38 AM - Berk Hess
- Priority changed from High to Low

Changed priority from high to low, since we now JIT the required kernels only which take just a few seconds.

#12 - 02/08/2016 02:21 PM - Gerrit Code Review Bot
Gerrit received a related DRAFT patchset '1' for Issue #1720.
Uploader: Mark Abraham (mark.j.abraham@gmail.com)
Change-Id: I25e78526f55715c779819e9b6b6b52ad9394c6
Gerrit URL: https://gerrit.gromacs.org/5624

#13 - 08/03/2016 12:25 PM - Mark Abraham
- Status changed from New to Rejected
- Assignee deleted (Mark Abraham)

Caching is now disabled by default. I don't think we have a need to do anything, and I'd rather tackle any such problem by parallelising JIT with simple tasking.

#14 - 08/04/2016 01:36 AM - Mark Abraham
- Related to Task #1938: work around broken NVIDIA JIT caching added