JÜLICH
FORSCHUNGSZENTRUM

panda

Using CUDA from PandaRoot

PANDA Collaboration Meeting 3-2013 (Bochum)

11. September 2013 | Andreas Herten

CUDA

- NVIDIA‘s GPU programming language
- Additions to C++
- Host (CPU) code: **gcc** (/…)
- Device (GPU) code: **nvcc**

CUDA @ PandaRoot

CUDA @ PandaRoot

- Idea based on Mohammad‘s work, see
  https://subversionion.gsi.de/trac/fairroot/browser/fairbase/release/cuda
  - Not used for long time
  - Did not work, when tried to compile
  - *Updated*

CUDA @ PandaRoot

- Idea based on Mohammad's work, see https://subversionion.gsi.de/trac/fairroot/browser/fairbase/release/cuda
  - Not used for long time
  - Did not work, when tried to compile
  - *Updated*
- Updated to fit new CMakeLists.txt structure with `rootmaps` *(thanks, Florian)*

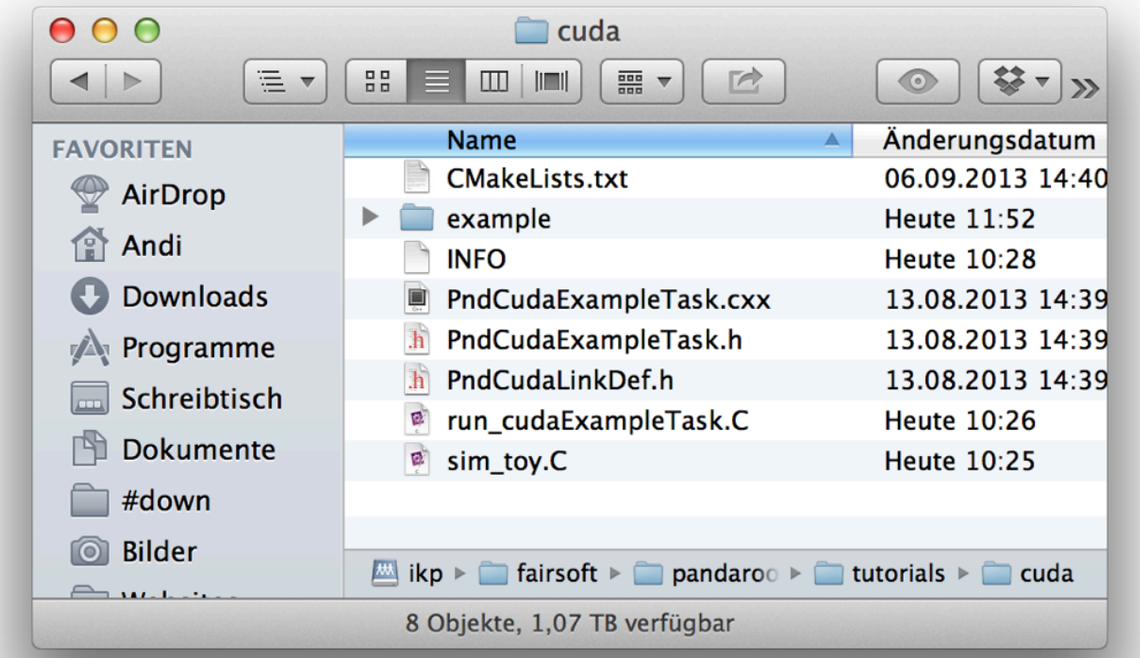
CUDA @ PandaRoot

- Idea based on Mohammad's work, see https://subversionion.gsi.de/trac/fairroot/browser/fairbase/release/cuda
  - Not used for long time
  - Did not work, when tried to compile
  - *Updated*
- Updated to fit new CMakeLists.txt structure with `rootmaps` *(thanks, Florian)*
- Parts:
  - C++/CUDA necessities
  - CMakeLists.txt configuration

CUDA @ PandaRoot

- Idea based on Mohammad's work, see https://subversionion.gsi.de/trac/fairroot/browser/fairbase/release/cuda
  - Not used for long time
  - Did not work, when tried to compile
  - *Updated*
- Updated to fit new CMakeLists.txt structure with `rootmaps` *(thanks, Florian)*
- Parts:
  - C++/CUDA necessities
  - CMakeLists.txt configuration

→ **Exemplary project** at `trunk/tutorials/cuda`



CUDA @ PandaRoot — C++

- Define device functions as `extern`, write host **wrapper**
- `example/cudaExample.cu`:

```
__global__ void square_array(double *a, int N) {
  int idx = blockIdx.x * blockDim.x + threadIdx.x;
  if (idx<N) a[idx] = a[idx] * a[idx];
  printf("idx = %d, a = %f\n", idx, a[idx]);
}


extern "C" void someOperation() {
  …
  cudaMalloc((void **) &a_d, size);
  cudaMemcpy(a_d, localFloat, size, cudaMemcpyHostToDevice);

  …
  square_array <<<n_blocks, block_size>>> (a_d, localN);
  …
}
```

CUDA @ PandaRoot — C++

- Define device functions as `extern`, write host **wrapper**
- `example/cudaExample.cu`:

```
extern "C" void someOperation() {}
```

- `PndCudaExampleTask.h/cxx`:

```
extern "C" void someOperation();
```

```
class PndCudaExampleTask : public FairTask
{
 public:
  …
  void callGpuStuff() {someOperation();};
};
```

```
void PndCudaExampleTask::Exec(Option_t* opt)
{
  callGpuStuff();
}
```

CUDA @ PandaRoot — CMake

- example/CMakeLists.txt:

```
find_package(CUDA)

CUDA_INCLUDE_DIRECTORIES(
  "${CMAKE_CURRENT_SOURCE_DIR}"
  )

set(CUDAEXAMPLE_SRCS
  cudaExample.cu
  )

list(APPEND CUDA_NVCC_FLAGS --gpu-architecture sm_20)

set(PNDCUDAEXAMPLESONAME "PndCudaExample")
set(PNDCUDAEXAMPLESONAME ${PNDCUDAEXAMPLESONAME} PARENT_SCOPE)

CUDA_ADD_LIBRARY(${PNDCUDAEXAMPLESONAME}
  ${CUDAEXAMPLE_SRCS}
   SHARED
  )

set_target_properties(${PNDCUDAEXAMPLESONAME}
  PROPERTIES
  ${FAIRROOT_LIBRARY_PROPERTIES}
  )
```

new

CUDA @ PandaRoot — CMake

- CMakeLists.txt:

```
add_subdirectory(${CMAKE_CURRENT_SOURCE_DIR}/example)

set(INCLUDE_DIRECTORIES
   ${ROOT_INCLUDE_DIR} ${CUDA_INCLUDE}
   ${BASE_INCLUDE_DIRECTORIES} ${CMAKE_CURRENT_SOURCE_DIR}
   ${CMAKE_SOURCE_DIR}/pnddata
  )
include_directories(${INCLUDE_DIRECTORIES})

link_directories(…)


set(SRCS
  PndCudaExampleTask.cxx
  )
set(LINKDEF PndCudaLinkDef.h)
set(LIBRARY_NAME PndCuda)
set(DEPENDENCIES Base ${PNDCUDAEXAMPLESONAME})

GENERATE_LIBRARY()
```

new

CUDA @ PandaRoot — CMake

- CMakeLists.txt:

```
add_subdirectory(${CMAKE_CURRENT_SOURCE_DIR}/example)

set(INCLUDE_DIRECTORIES
   ${ROOT_INCLUDE_DIR} ${CUDA_INCLUDE}
   ${BASE_INCLUDE_DIRECTORIES} ${CMAKE_CURRENT_SOURCE_DIR}
   ${CMAKE_SOURCE_DIR}/pnddata
  )
include_directories(${INCLUDE_DIRECTORIES})

link_directories(…)


set(SRCS
  PndCudaExampleTask.cxx
  )
set(LINKDEF PndCudaLinkDef.h)
set(LIBRARY_NAME PndCuda)
set(DEPENDENCIES Base ${PNDCUDAEXAMPLESONAME})

GENERATE_LIBRARY()
```

new

+Some overhead due to nvcc ↔ clang incompatibilities (*OS X?*):

```
if(NOT DEFINED
  CUDA_HOST_COMPILER AND CMAKE_C_COMPILER_ID
  STREQUAL ”Clang“
  ) …
```

new

CUDA @ PandaRoot — CMake

- Don‘t forget to…
  - Add to your `PndCudaLinkDef.h`:

```
#pragma link C++ class PndCudaExampleTask+;
```

  - Add to your `rootlogon.C`:

```
if(isLibrary("libPndCuda")) gSystem->Load("libPndCuda");
```

- Then call your Task as usual via:

```
PndCudaExampleTask * cudaDummy = new PndCudaExampleTask();
fRun->AddTask(cudaDummy);
```

- And be done:

```
============= PndCudaExampleTask:: START CUDA KERNEL CALL:
idx = 0, a = 0.000000
idx = 1, a = 1.000000
idx = 2, a = 4.000000
```

- You can also trigger on CUDA availability:

```
if (CUDA_FOUND) ...
```

Example project files:
- sim_toy.C
- **run_cudaExampleTask.C**

Try it!

CUDA @ PandaRoot — CMake

- Don‘t forget to…
  - Add to your `PndCudaLinkDef.h`:

```
#pragma link C++ class PndCudaExampleTask+;
```

  - Add to your `rootlogon.C`:

new

`rootmaps` are generated, no need for `rootlogon.C`

- Then call your Task as usual via:

```
PndCudaExampleTask * cudaDummy = new PndCudaExampleTask();
fRun->AddTask(cudaDummy);
```

Example project files:
- sim_toy.C
- **run_cudaExampleTask.C**

Try it!

- And be done:

```
============= PndCudaExampleTask:: START CUDA KERNEL CALL:
idx = 0, a = 0.000000
idx = 1, a = 1.000000
idx = 2, a = 4.000000
```

- You can also trigger on CUDA availability:

```
if (CUDA_FOUND) ...
```

Conclusion

- Possible to run CUDA from PandaRoot
- Updated to new CMake structure
- Example project & necessary files at `pandaroot/trunk/tutorials/cuda`
- Use it, give feedback!

Conclusion

- Possible to run CUDA from PandaRoot
- Updated to new CMake structure
- Example project & necessary files at `pandaroot/trunk/tutorials/cuda`
- Use it, give feedback!

Resources Used in This Talk

- [2] NVIDIA vector logo from Wikipedia
- [3] My screenshot