Distributed GPGPU Computing

Martin Stumpf

Ste||ar Group, Louisiana State University

September 17, 2014

Table of Contents

Outline

CPU vs GPU

"<IMAGE GPU vs CPU>"

Why GPGPU?

The theoretical calculation power of a GPU is much higher than a CPU.

Example

CPU (Intel Xeon E5-2670 v3):

- 12 Cores, 2.3 GHz, 32 FLOPS/cycle
    - 884 GFLOPS
- Prize: $\sim$ 1500 $

GPU (NVidia Tesla K40):

- 2880 Cores, 745 MHz, 2 FLOPS/cycle
    - 4291 GFLOPS
- Prize: $\sim$ 4000 $

So, what computational tasks are actually suitable for GPGPU?

Problems suitable for GPGPU

Every problem that fits the SPMD programming scheme, can benefit greatly from GPGPU.

Examples:

- Fluid Simulations
- Mathematical Vector Operations
- Image Processing
- Stencil Based Simulations

SPMD based Programming Languages:

- CUDA (NVidia)
- OpenCL (Platform independent)

OpenCL

"<Image OpenCL host-device(kernel+buffer)>"

OpenCL

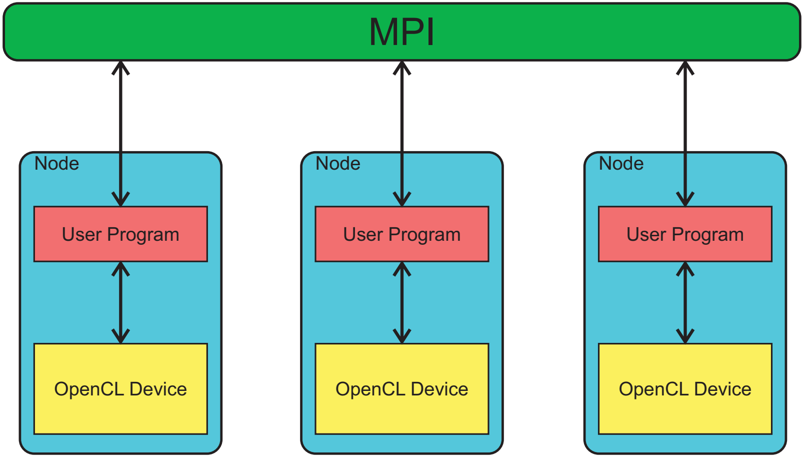
- An OpenCL device is split in two components:
    - The Buffer: Represents memory on the device
    - The Kernel: A C-style function that modifies one or multiple elements of a buffer
- Kernel source code stays plain text and gets compiled at runtime

$\Longrightarrow$ OpenCL programs are device independent

- Kernel executions on the device run asynchronous to the host program

Outline

Distributed OpenCL with MPI



Distributed OpenCL with MPI

Disadvantages:

- MPI and OpenCL are independent from each other

  $\Longrightarrow$ Connection between computation and data exchange has to be implemented manually

- Every OpenCL device can only be accessed within its own node
- If no further methodes are used, the whole cluster will run in lockstep

Outline

HPX

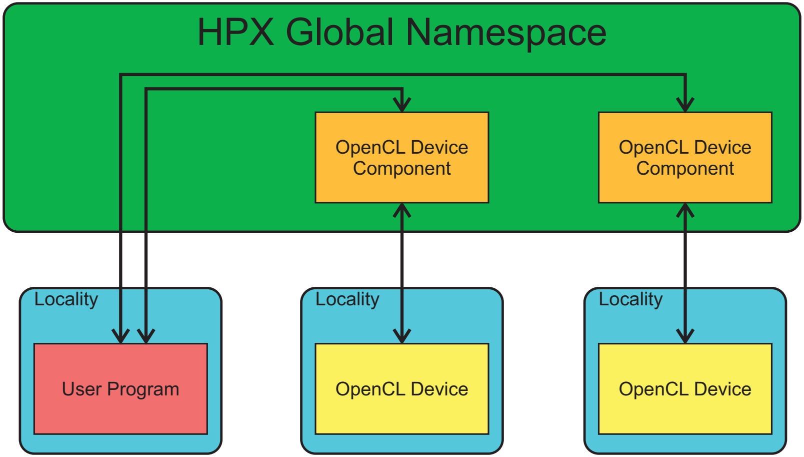
What is HPX?

- A scaling C++ runtime system for parallel and distributed applications
- Based on the ParalleX model

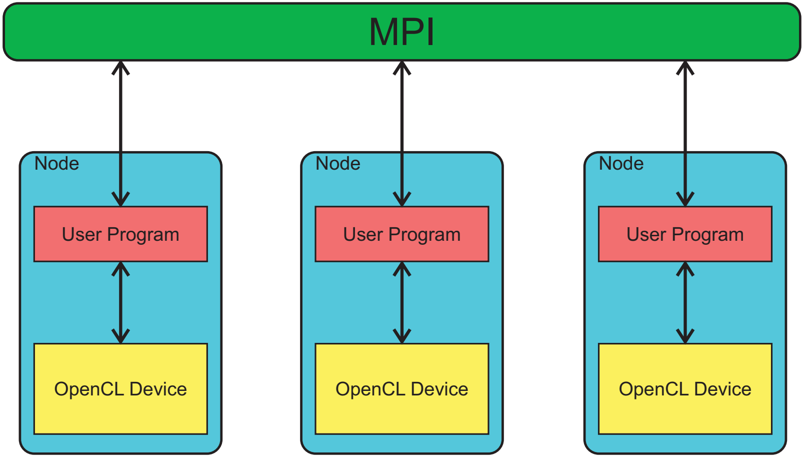
Advantages for distributed OpenCL:

- Global Namespace
- Cluster as "one large machine" (MPI: every Node is autonomous)
- Data dependencies (futures) (MPI: Send-Wait)

Distributed OpenCL with HPX



Distributed OpenCL with MPI



Affect on distributed GPGPU programming

- Abstracting the whole cluster as one machine
- Simpler, not need to think in a distributed way
- Data dependencies
  - faster due to prevention of lockstep
  - possible to apply synchronization techniques of standard opencl
- Seamless integration of more opencl nodes into the system
- Possibility to run heterogeneous nodes/devices in one system
- Easy to port non-distributed code to distributed opencl whilst maintaining descent scaling

Outline

Layout

- Is an implementation of that concept.
- Wraps every OpenCL datastructure in a component:

OpenCL	HPXCL
cl_device	hpx::opencl::device
cl_program	hpx::opencl::program
cl_kernel	hpx::opencl::kernel
cl_mem	hpx::opencl::buffer
cl_event	hpx::opencl::event
	(soon: hpx::future)

Hello, World! - Getting devices

Hello, World! - Writing data to the device

Hello, World! - Creating a kernel

Hello, World! - Executing the kernel

Hello, World! - Reading the result

Outline

bla

Scaling

Parallel Efficiency