Fachhochschule Nordwestschweiz
Hochschule für Technik

Introduction to OpenCL

Concurrent Programming

OpenCL

Inhalt

- Types of parallelism
- OpenCL platform
- Programming model
  - Extensions

Types of parallelism

- Task parallelism
  - Client connection handler
  - Web crawler
  - ...
- Data parallelism
  - Image manipulation
  - Vector arithmetics
  - ....

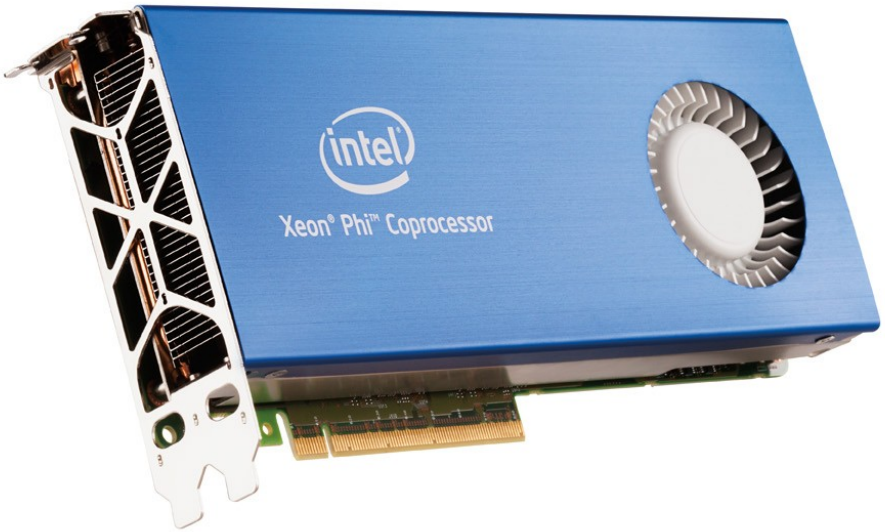
Types of parallelism – Task parallel hardware

- Modern CPUs
  - Multiple, full-blown CPU cores
  - Branch-prediciting
  - Caching
  - Independent from each-other (but share infrastructure)

Types of parallelism – Data parallel hardware

- Modern GPUs
  - SIMD – Single instruction multiple data
  - „stupid“ processors
  - One GPU processor can execute several threads at once

Types of parallelism – Parallel hardware



Types of parallelism – Parallel hardware

Intel® Xeon Phi™ Coprocessor Family Reference Table

SKU #	Form Factor, Thermal	Peak Double Precision	Max # of Cores	Clock Speed (GHz)	GDDR5 Memory Speeds (GT/s)	Peak Memory BW	Memory Capacity (GB)	Total Cache (MB)	Board TDP (Watts)	Process
SE10P (special edition)	PCIe Card, Passively Cooled	1073 GF	61	1.1	5.5	352	8	30.5	300	22nm
SE10X (special edition)	PCIe Card, No Thermal Solution	1073 GF	61	1.1	5.5	352	8	30.5	300	
5110P	PCIe Card, Passively Cooled	1011 GF	60	1.053	5.0	320	8	30	225	
3100 Series	PCIe Card, Actively Cooled	>1 TF	Disclosed at 3100 series launch (H1'13)		5.0	240	6	28.5	300	
	PCIe Card, Passively Cooled	> 1 TF			5.0	240	6	28.5	300	

PCIe Card, Actively Cooled

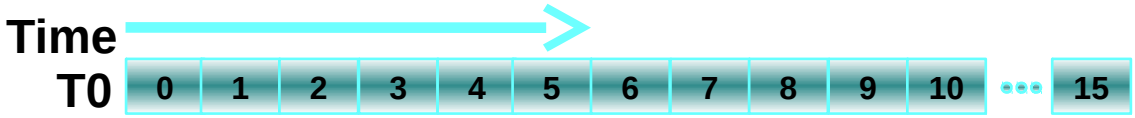
PCIe Card, Passively Cooled

Types of parallelism – Parallel hardware

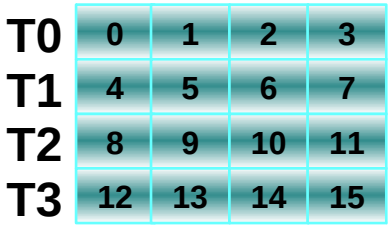
TECHNICAL SPECIFICATIONS	TESLA K10[a]	TESLA K20	TESLA K20X
Peak double precision floating point performance (board)	0.19 teraflops	1.17 teraflops	1.31 teraflops
Peak single precision floating point performance (board)	4.58 teraflops	3.52 teraflops	3.95 teraflops
Number of GPUs	2 x GK104s	1 x GK110	
Number of CUDA cores	2 x 1536	2496	2688
Memory size per board (GDDR5)	8 GB	5 GB	6 GB
Memory bandwidth for board (ECC off)[b]	320 GBytes/sec	208 GBytes/sec	250 GBytes/sec
GPU computing applications	Seismic, image, signal processing, video analytics	CFD, CAE, financial computing, computational chemistry and physics, data analytics, satellite imaging, weather modeling	
Architecture features	SMX	SMX, Dynamic Parallelism, Hyper-Q	
System	Servers only	Servers and Workstations	Servers only

Overview of work splitting

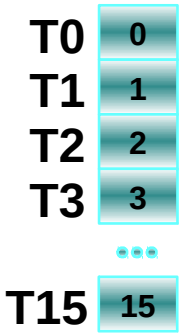
- Serial Execution



- Multi-Threading (CPU)



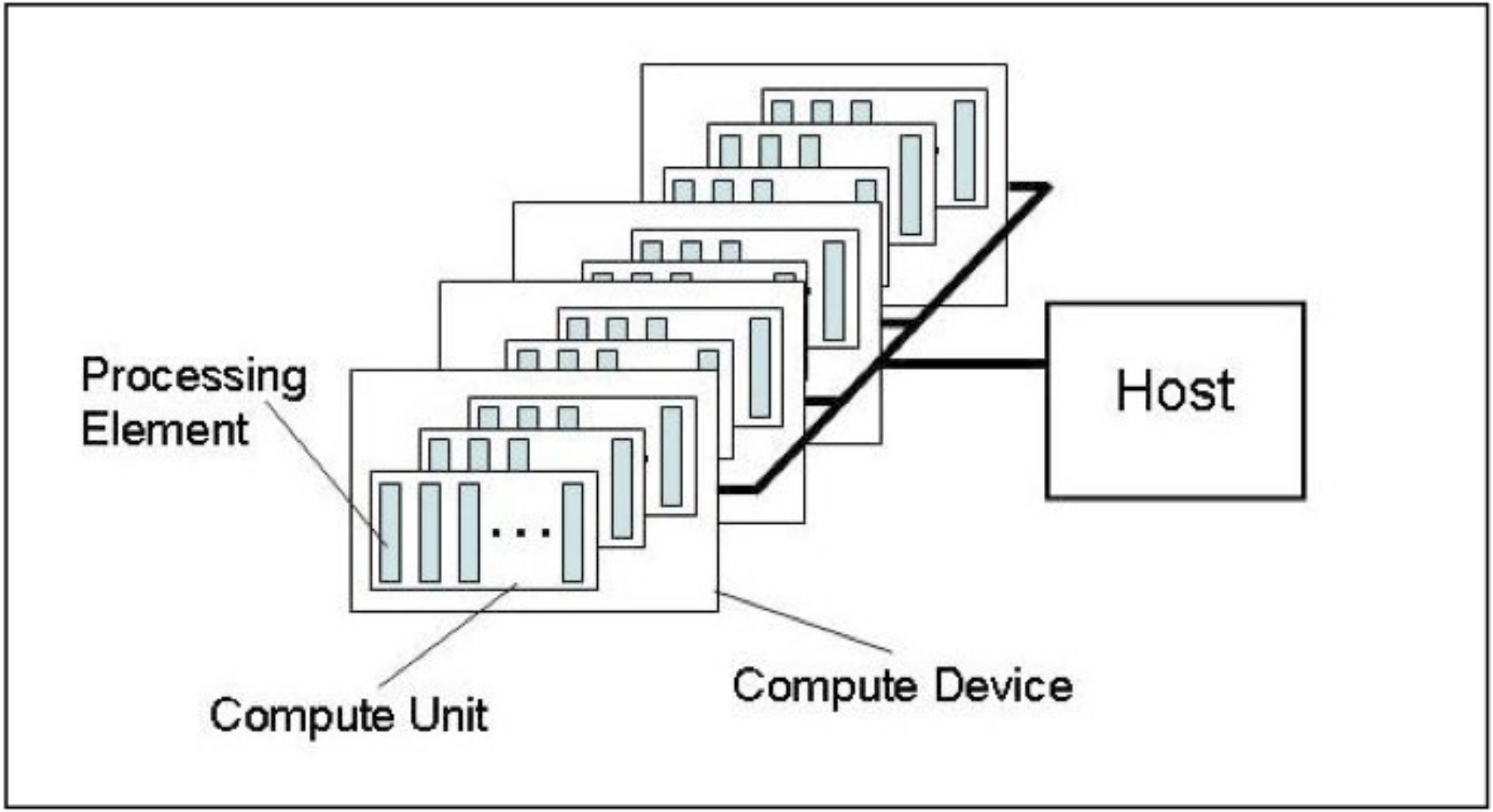
- Massivly parallel (GPU)



OpenCL Platform

- Platform for data parallel applications
  - No support for task based parallelism
- Data parallelism
  - SPMD – Single program multiple data
- Programs are written as *Kernels*
  - One kernel executed on one Work Item

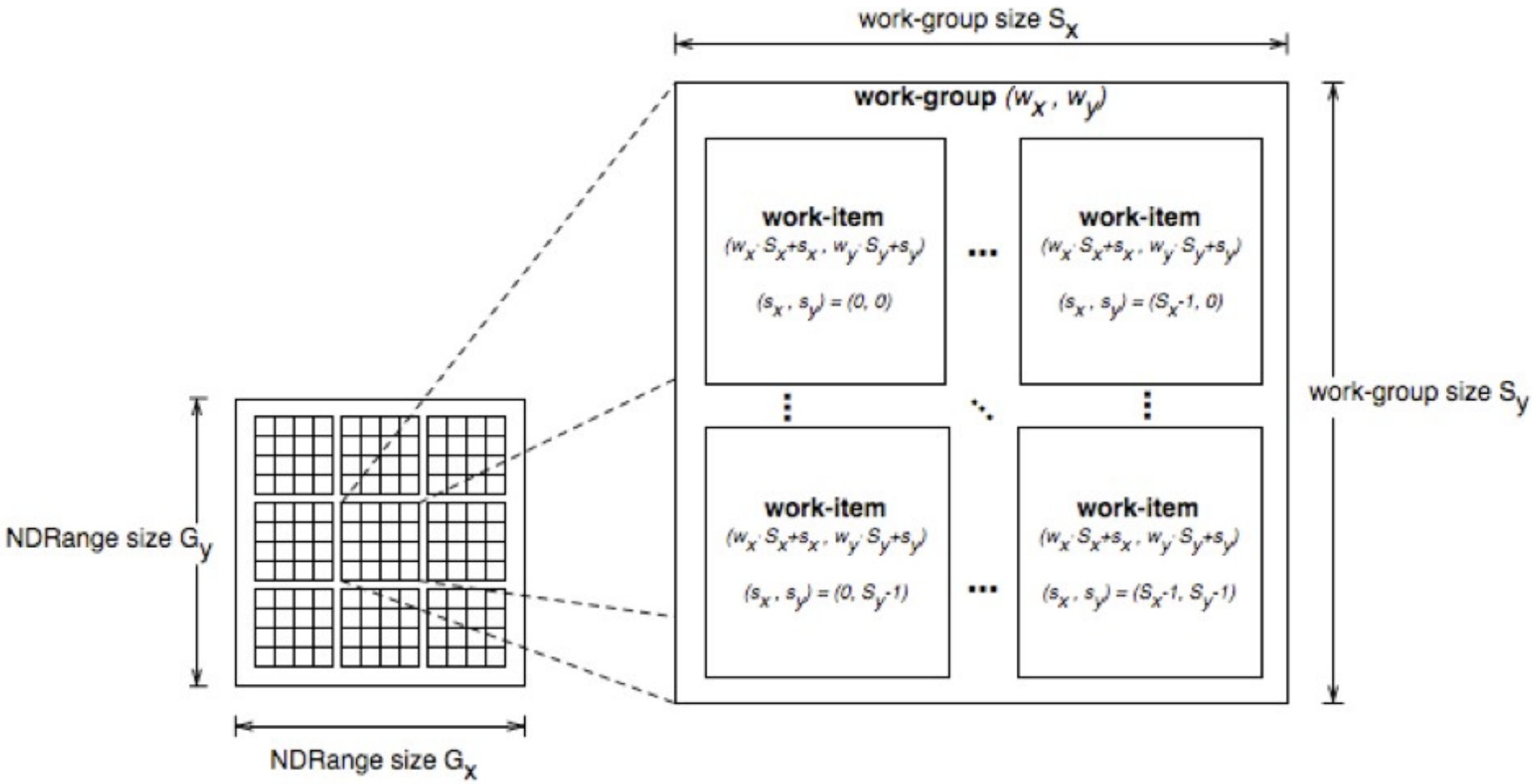
OpenCL Platform - Overview



OpenCL Platform

- Program consitsts of *Kernels*
- A *Kernel* is executed on a *Work Item*
- Wok Items are grouped to *Work Groups*

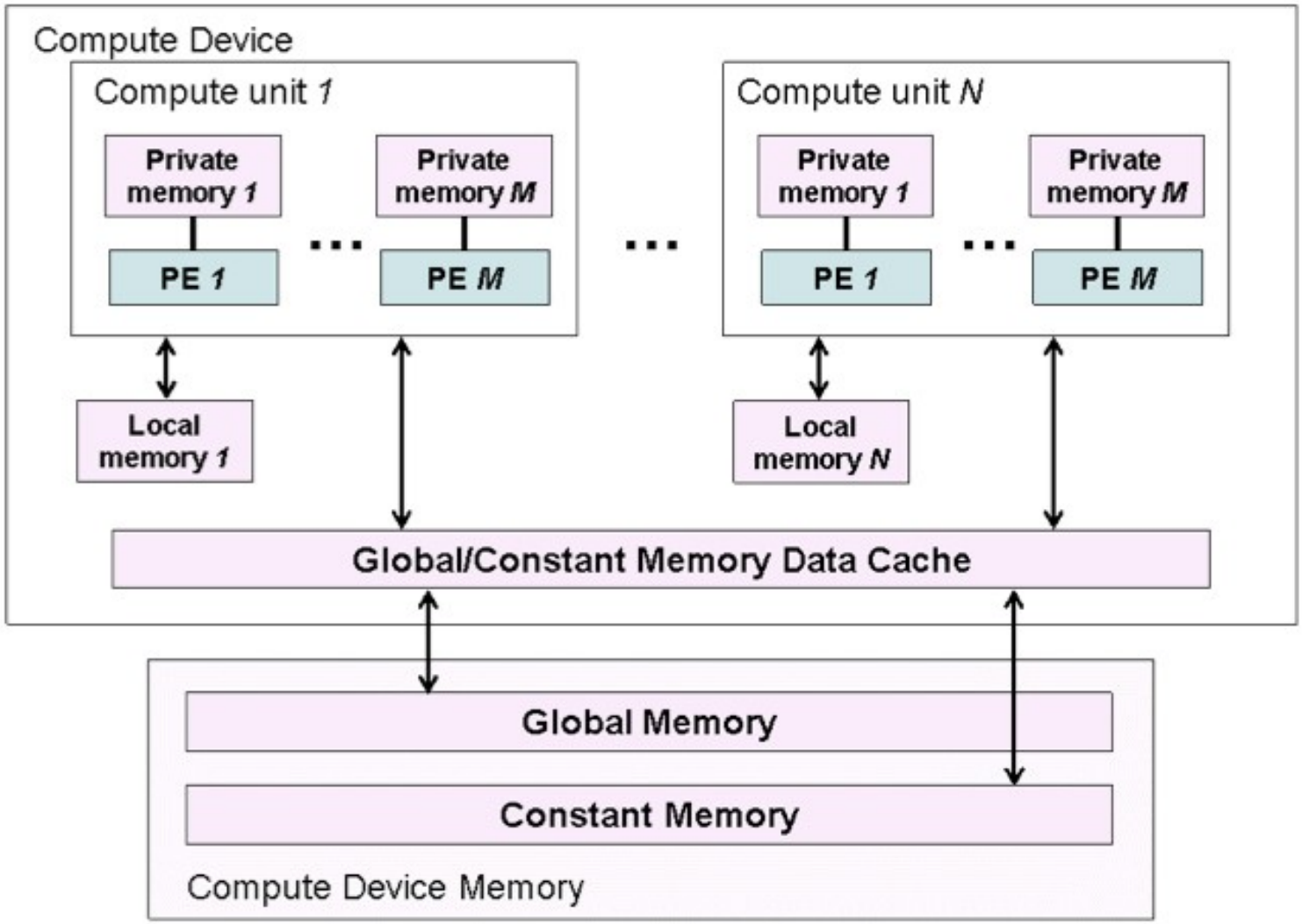
OpenCL Platform - Overview



OpenCL Platform - Memory

- Host memory
- Memory is separated to reflect the possibilties of GPUs
- Global memory (biggest, slowest) / Constant memory
  - All work items have access
  - On GPU: Streaming memory
- Local memory
  - All work items in same group share it
- Private memory (smallest, fastest)
  - Private to a work item

OpenCL Platform - Memory



OpenCL Platform - Memory

- Explicit memory management
  - You have to decide what to put to which memory location
- Be careful with optimizations
  - On CPUs no such thing as „local“ memory exists, only RAM

OpenCL Programming Model

- Programs are writen in OpenCL C
  - Compiled at runtime by a specialized, device dependent compiler
  - Generates highly optimized code
  - Some overhead
- No fancy memory management units on GPU
  - No SIGSEGV
  - Display server can crash

OpenCL Synchronization

- Barrier
  - localBarrier(): All kernels of a work group
  - globalBarrier(): Every kernel
- Dead Lock awareness!
  - Every kernel has to have the same amount of calls to a barrier
- Atomics
  - cl_khr_global_int32_base_atomics

Aparapi

- AMD Open Source project
- Generates OpenCL C code from bytecode
  - Handles communication with devices
- OpenJDK Project Sumatra

Aparapi – Simple Example

- Sequential version

```
final float inA[] = .... // get a float arra
final float inB[] = .... // get a float arra
final float result = new float[inA.length];

for (int i=0; i<array.length; i++){
    result[i]=intA[i]+inB[i];
}
```

- OpenCL

```
Kernel kernel = new Kernel(){
   @Override public void run(){
      int i= getGlobalId();
      result[i]=intA[i]+inB[i];
   }
};
Range range = Range.create(result.length);
kernel.execute(range);
```

Aparapi – OpenCL Functions

- Defined on *com.amd.aparapi.Kernel*

- getGlobalId(int dimension)
- getLocalId(int dimension)
- getGlobalSize(int dimension)
- getLocalSize(int dimension)

Aparapi – OpenCL Math Functions

- Defined on *com.amd.aparapi.Kernel*
- Named like *java.lang.Math.**
  - But defined for floats as well

- abs
- exp
- sqrt
- ...

Aparapi – Device

- Device.best()
- Device.firstGPU()
- Device.firstCPU()
- List all OpenCL devices

```
@Test
public void printAvailableDevices() {
    OpenCLDevice.select(new DeviceSelector() {
        @Override
        public OpenCLDevice select(OpenCLDevice currentDevice) {
            System.out.println(currentDevice);
            return null;
        }
    });
```

Aparapi – Range

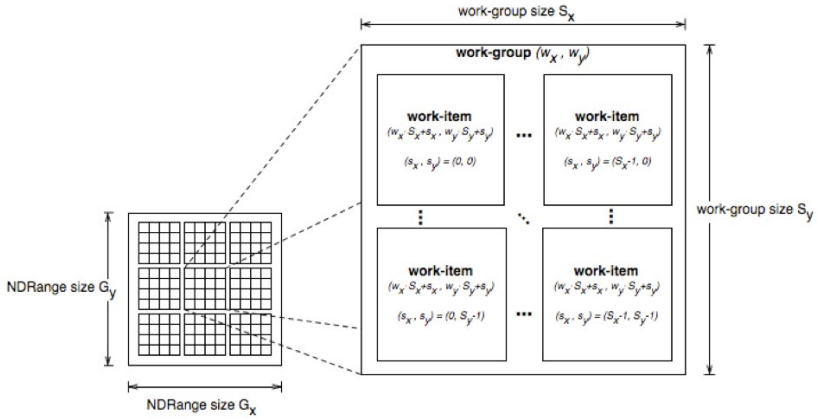
- Global Work Size has to be evenly devidable with groupSize

- Range.create(globalWorkSize)
- Range.create(Device, globalWorkSize, groupSize)
- Range.create2D(

  Device,

  globalSizeD1, globalSizeD2,

  groupSizeD1, groupSizeD2)



Aparapi – Memory Management

- Annotations

```
@Local private int[] densities;
@Constant private float[] dose;
```

- Name suffix

```
private int[] densities_$local$;
private float[] dose_$constant$;
```

Aparapi – What's supported

- Call functions on same class
- One dimensional arrays of primitive type
- Some khronos extensions
  - cl_khr_fp64
  - cl_khr_global_int32_base_atomics

Aparapi – When something breaks

- JTP fallback

```
Jun 10, 2013 7:07:17 PM com.amd.aparapi.KernelRunner warnFallBackAndExecute
WARNING: Reverting to Java Thread Pool (JTP) for class ch.fhnw.conpr.mandel.Ap
com.amd.aparapi.ClassParseException: Can't assign to two dimension array
```

Aparapi

+ JVM fallback
+ No memory management
+ Java
+ Debuggable

- Not extensible
- Code has to be rewritten

Links

- OpenCL Reference
  - http://www.khronos.org/registry/cl/sdk/1.2/docs/man/xhtml/
- Aparapi
  - https://code.google.com/p/aparapi/

Worksheet

- Task 1
  - Mandelbrot set
- Task 2
  - Experiment with different workloads