Aparapi Quick Reference Guide

Create A Kernel by extending com.amd.aparapi.Kernel

Kernel overrides method.

Kernel run() and run-reachable methods can read/write elements of single dimension array primitive final and non-final fields.

Kernel can read primitive final and non-final fields

Create an anonymous inner class extending com.amd.aparapi.Kernel

Kernel must override method.

Kernel run() and run-reachable methods can read and write elements of captured single dimension arrays of primitive from the call-site

Kernel can read captured final primitive values from the call-site.

This code sets each element of data[] to its index.

Executing your kernel over a given range (0..<range>)

Use Kernel.execute(int <range>) to execute over the range 0..<range>

Each Kernel execution will receive a unique value (0..<range>) via kernel.getGlobalId().

This code each element of data[] to its index.

Compiling your application

Add aparapi.jar to your classpath

Turn on debugging information for compiled classes (javac –g option)

Linux® users use : as a path separator instead of ;

Running your application

Add aparapi.jar to your classpath

Add aparapi shared library to java.library.path

Modes of execution

Mode	Method Of Execution	OpenCL™
JTP	Java Thread Pool. One Thread per available core	
SEQ	Single sequential Java loop.	
GPU	Convert bytecode to OpenCL execute on first available GPU device	✓
CPU	Convert bytecode to OpenCL execute on available CPU cores.	✓

Default to GPU if OpenCL is available

Falls back to JTP mode if Kernel cannot be converted or OpenCL unavailable.

Setting default execution mode

Developer request mode prior to first call to kernel.execute(<range>)using kernel.setExecutionMode().

Alternatively set the requested mode (for all Kernels) via com.amd.aparapi.ExecutionMode property

Determining the execution mode

Determine the actual execution mode after Kernel.execute(<range>) returns via Kernel.getExecutionMode().

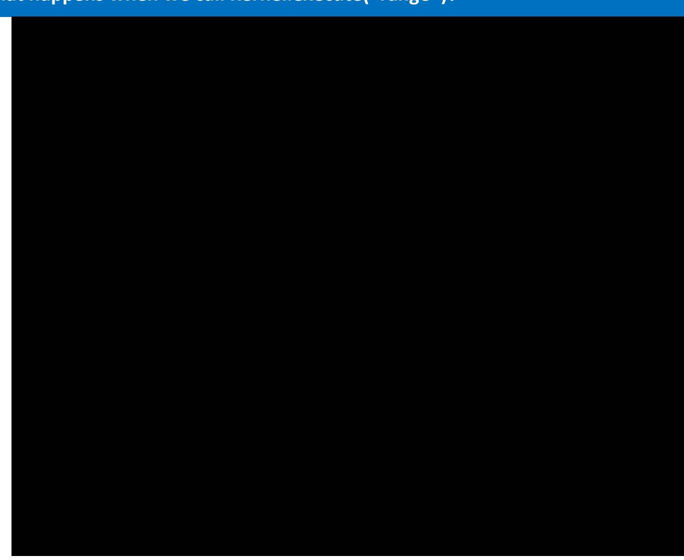
Alternatively set com.amd.aparapi.enableReportKernelExecutionMode property to true when application is launched.

Aparapi properties to control runtime behavior

Properties	Options (default)
com.amd.aparapi.executionMode	SEQ|JTP|CPU|GPU
com.amd.aparapi.enableExecutionModeReporting	true|false
com.amd.aparapi.enableShowGeneratedOpenCL	true|false
com.amd.aparapi.logLevel	FINEST|FINE|INFO|WARNING|SEVERE
com.amd.aparapi.enableProfiling	true|false
com.amd.aparapi.enableVerboseJNI	true|false
com.amd.aparapi.instructionListenerClass	Name of a class implements Config.InstructionListener

For example:

What happens when we call Kernel.execute(<range>)?



Explicit buffer management

For some applications where Kernel.execute calls are executed multiple times (loops) and where the Java code between Kernel.execute calls does not access buffers (primitive arrays) Aparapi can cause unnecessary buffer copies.

For performance reasons we allow the developer to take explicit control of transfers.

Kernel.execute(range), Kernel.put() and Kernel.get() support the fluent style of API (return Kernel instance) so that calls can be chained.

Kernel methods to determine execution identity

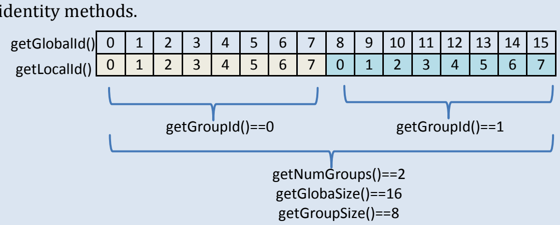
Kernel.execute(range, passCount) will execute the Kernel.run() method 0..<range> for each iteration through an outer the loop 1..<passCount>.

Note Kernel.execute(<range>) is equivalent to Kernel.execute(<range>, 1)

The 0..<range> execution is divided into one or more equal sized groups.

For example; Kernel.execute(16) may result in 1 group of 16, 2 groups of 8, 4 groups of 4 or 16 groups of 1

For example if Kernel.execute(16) resulted in 2 groups of 8 the following diagram shows typical values for each corresponding global id value we might expected for eacj pf the kernel identity methods.



Choose <range> carefully (powers of 2 ideal) to maximize group size.

Note if <range> is a prime number then group size will be 1 and inefficient

Method	Returns
getGlobaId()	The global id that this Kernel execution represents (0..<range>)
getGroupSize()	The size of each group. The same value returned for all Kernel executions resulting from one invocation of Kernel.execute(<range>) .
getGroupId()	The groupId of the currently executing Kernel (0..getNumGroups())
getLocalId()	The localId (from 0..<groupSize>) for this Kernel execution.
getGlobalSize()	The value of <range>
getPassId()	Determine the current passId (0..<passCount>).

There is no way to determine group size or count prior to executing Kernel.execute(<range>) and no guarantee that the same values will be used for subsequent invocations of Kernel.execute(<range>).

Extended forms of Kernel.execute()

If Kernel.execute() is sole statement in a loop whose iteration count is known on first iteration we can replace the loop with an extra 'pass count' arg to Kernel.execute(range, passCount).

Can be replaced with

Code is more compact and avoids unnecessary buffer transfers between iterations.

Kernel implementation can extract the value of the current pass using Kernel.getPassId().

Mapping of Aparapi Kernel math methods to Java and OpenCL equivalents

Kernel method	Java mapping	OpenCL mapping
abs(float) abs(double) abs(int) abs(long)	Math.abs(float) Math.abs(double) Math.abs(int) Math.abs(long)	fabs(float) fabs(double) abs(int) abs(long)
acos(float) acos(double)	Math.acos(float) Math.acos(double)	acos(float) acos(double)
asin(float) asin(double)	Math.asin(float) Math.asin(double)	asin(float) asin(double)
atan(float) atan(double)	Math.atan(float) Math.atan(double)	atan(float) atan(double)
atan2(float, float) atan2(double, double)	Math.atan2(float, float) Math.atan2(double, double)	atan2(float, float) atan2(double, double)
ceil(float) ceil(double)	Math.ceil(float) Math.ceil(double)	ceil(float) ceil(double)
cos(float) cos(double)	Math.cos(float) Math.cos(double)	cos(float) cos(double)
exp(float) exp(double)	Math.exp(float) Math.exp(double)	exp(float) exp(double)
floor(float) floor(double)	Math.floor(float) Math.floor(double)	floor(float) floor(double)
max(float,float) max(double,double) max(int,int) max(long,long)	Math.max(float,float) Math.max(double,double) Math.max(int,int) Math.max(long,int)	fmax(float,float) fmax(double,double) max(int,int) max(long,long)
min(float,float) min(double,double) min(int,int) min(long,long)	Math.min(float,float) Math.min(double,double) Math.min(int,int) Math.min(long,int)	fmin(float,float) fmin(double,double) min(int,int) min(long,long)
log(float) log(double)	Math.log(float) Math.log(double)	log(float) log(double)
native_sqrt(float) native_sqrt(double)	See method implementation	native_sqrt(float) native_sqrt(double)
pow(float, float) pow(double,double)	Math.pow(float, float) Math.pow(double, float)	pow(float, float) pow(double, float)
IEEEremainder(float, float) IEEEremainder(double,double)	Math.IEEEremainder(float, float) Math.IEEEremainder(double, double)	remainder(float, float) remainder(double, double)
rint(float) rint(double)	Math.rint(float) Math.rint(double)	rint(float) rint(double)
round(float) round(double)	Math.round(float) Math. round(double)	round(float) round(double)
rsqrt(float) rsqrt(double)	1f/Math.sqrt(float) 1.0/Math. round(double)	rsqrt(float) rsqrt(double)
sin(float) sin (double)	Math.sin(float) Math.sin(double)	sin(float) sin(double)
sqrt(float) sqrt(double)	Math.sqrt(float) Math.sqrt(double)	sqrt(float) sqrt(double)
tan(float) tan(double)	Math.tan(float) Math.tan(double)	tan(float) tan(double)
toRadians(float) toRadians(double)	Math.toRadians(float) Math.toRadians(double)	radians(float) radians(double)
toDegrees(float) toDegrees(double)	Math.toDegrees(float) Math.toDegrees(double)	degress(float) degrees(double)

Note the differences in precision between Java and OpenCL™ implementation of arithmetic functions to determine whether the difference in precision is acceptable.