SPARK SUMMIT 2017

Accelerating SparkML Workloads on the Intel® Xeon®+FPGA Platform

Zhongyue Nah, Intel

Srivatsan Krishnan, Intel

Legal Disclaimer

Outline

- What is an FPGA
- Intel® Accelerator Portfolio
- Intel® FPGA Hardware Platform
- Intel® FPGA Software
- Intel® Accelerator Abstraction Layer Enabling
  - Spark
  - YARN
  - Spark MLlib
- Spark-perf Performance Test Results
  - Netlib vs. Intel® DAAL
- Intel® FPGA Acceleration Demo

SPARK SUMMIT 2017

What is FPGA

1000s of hard DSPs (floating-point units)

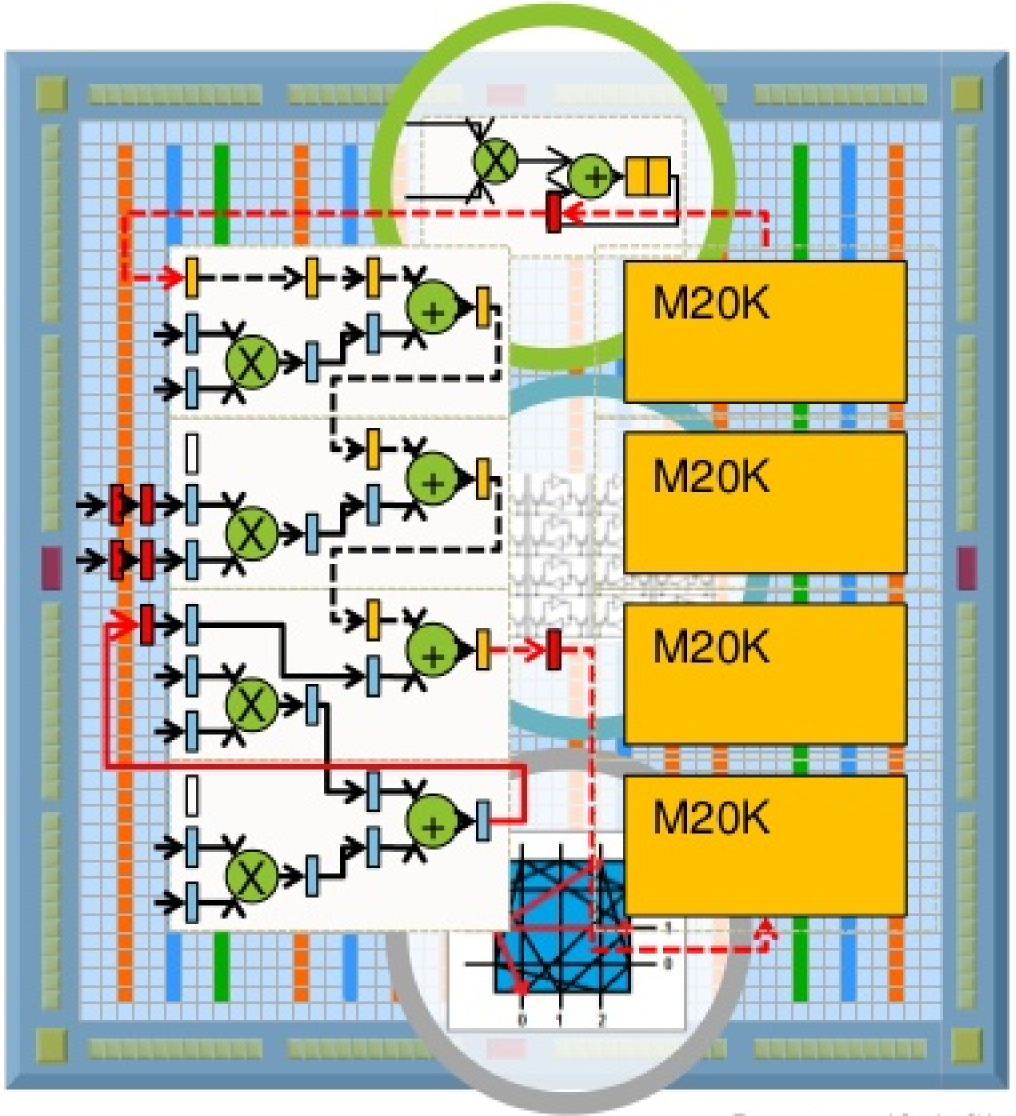
1000s of Hard “M20K” SRAMs (2.5KB/SRAM)

Sea of Programmable Logic and Routing

Extreme degree of customizations

Arbitrary bitwidth, mix bitwidths, etc

Arbitrary SRAMs compositions (spad, $, fifo, ..)



Figures courtesy of Gordon Chiu

FPGAs well positioned for High performance and providing flexibility

Intel® Portfolio of Accelerator Options

IA PROCESSORs

+

Software Flexibility

FPGA

Arria 10 Stratix 10

Flexible Workloads

Integrated FPGA
CPU socket compatible access to FPGA capabilities

Discrete FPGA
Scalable range of FPGA options (I/O, TDP, Price, Mem, Features)

Hardware Flexibility

and/ or

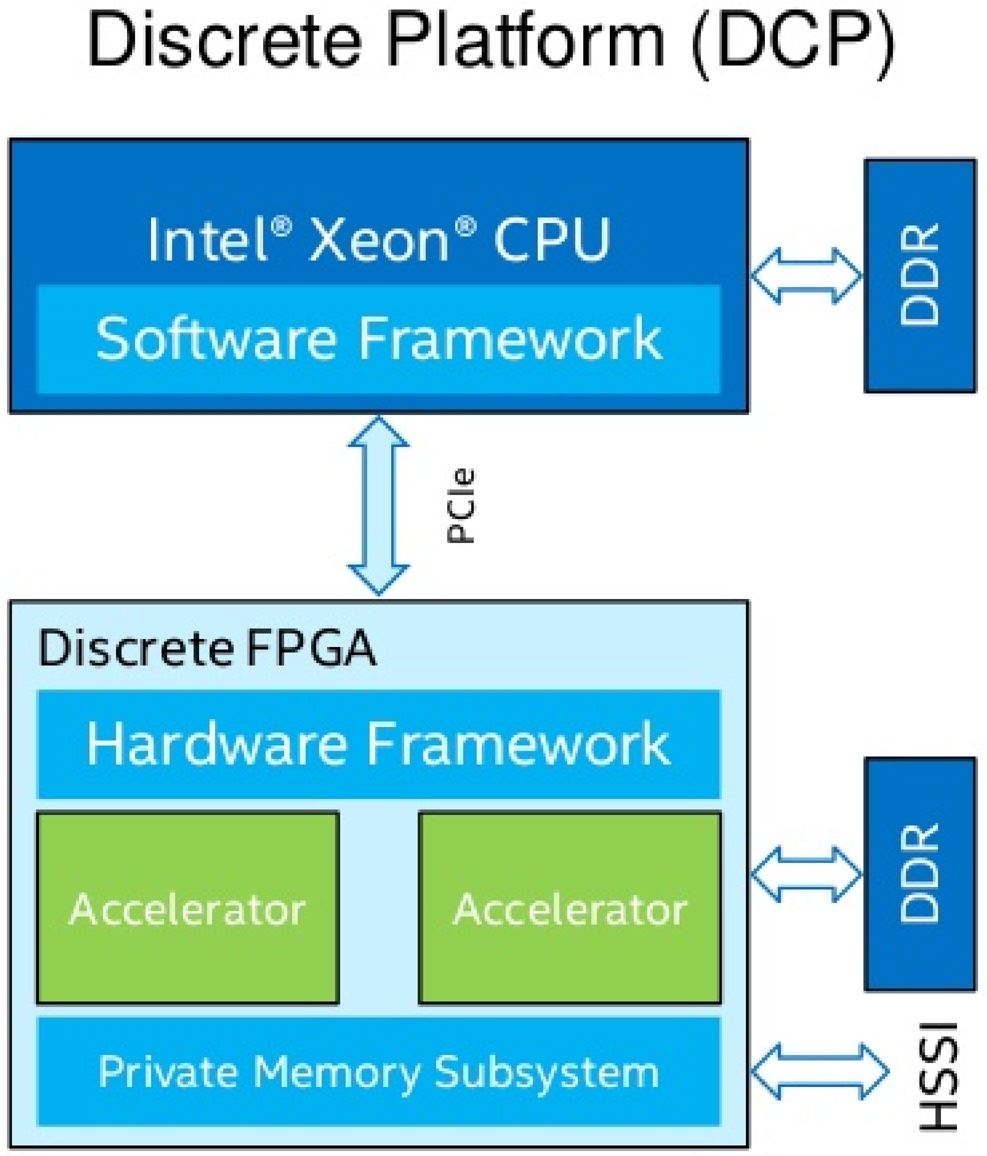
Fixed Function ACCELERATION

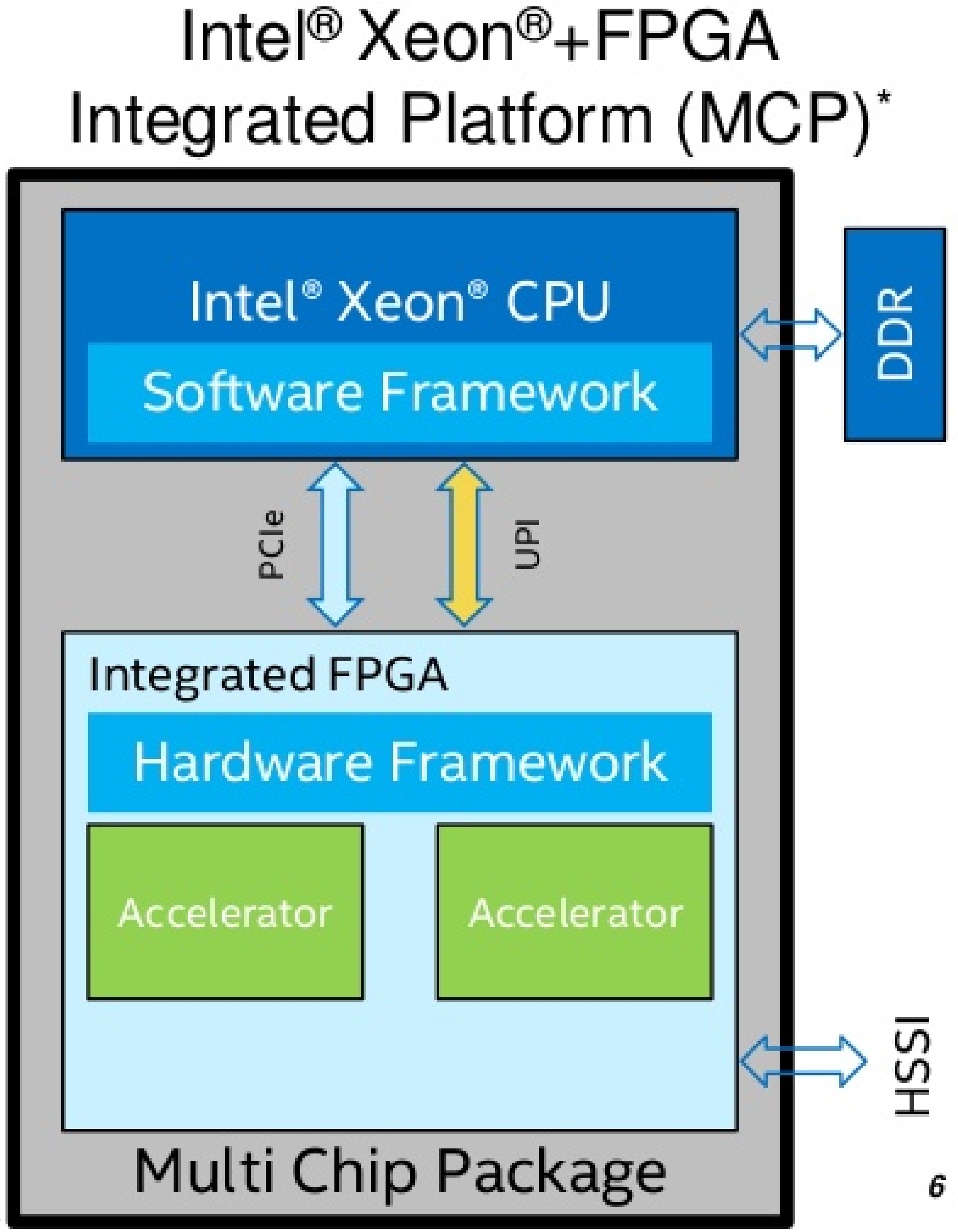
Standard Workloads

Built-in Standard platform acceleration, Highly Optimized

Fixed HW Acceleration

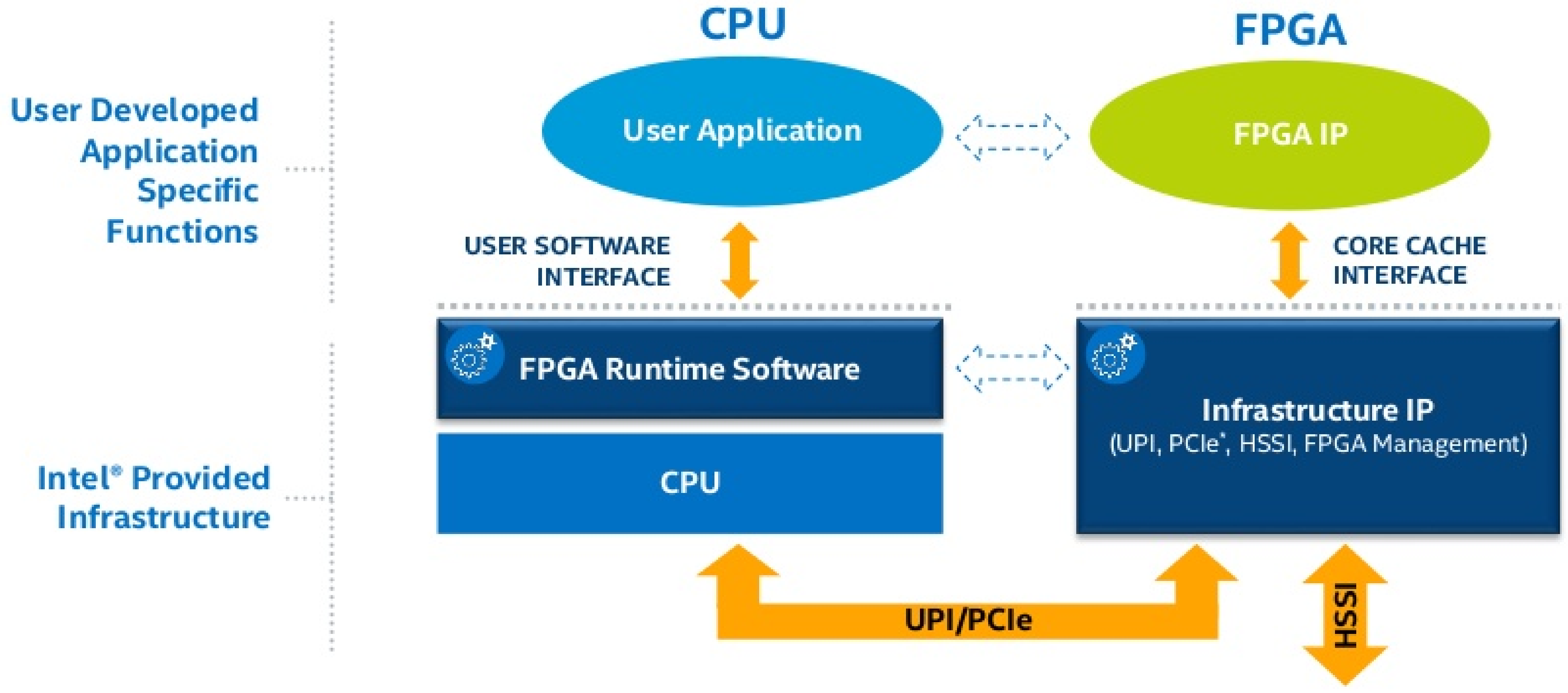
Discrete and Integrated FPGA platforms





* Intel® Broadwell + FPGA System (HARPv2 System).

Intel® Accelerator Abstraction Layer (AAL)

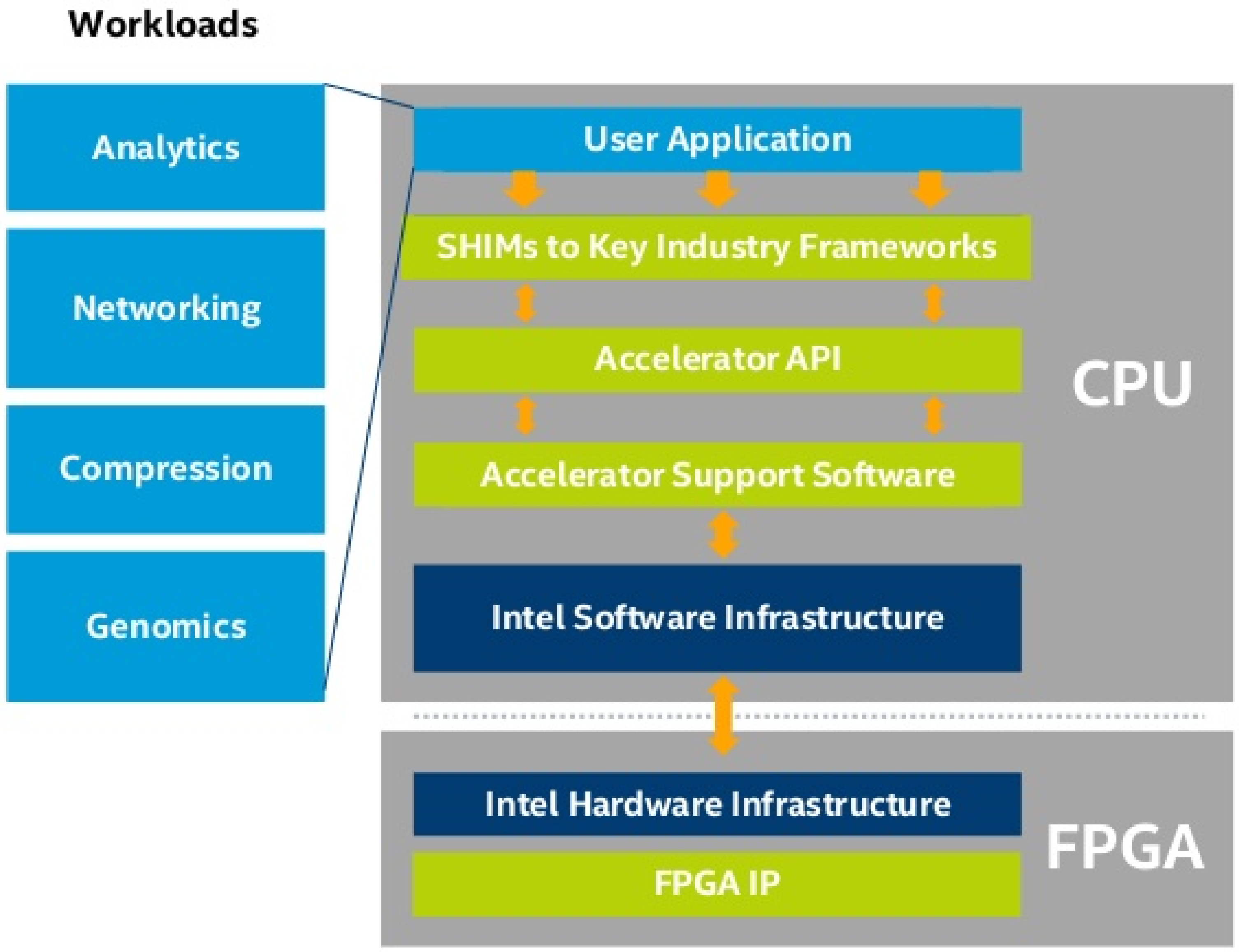


= New blocks that simplify code development.

Single Node Software Stack

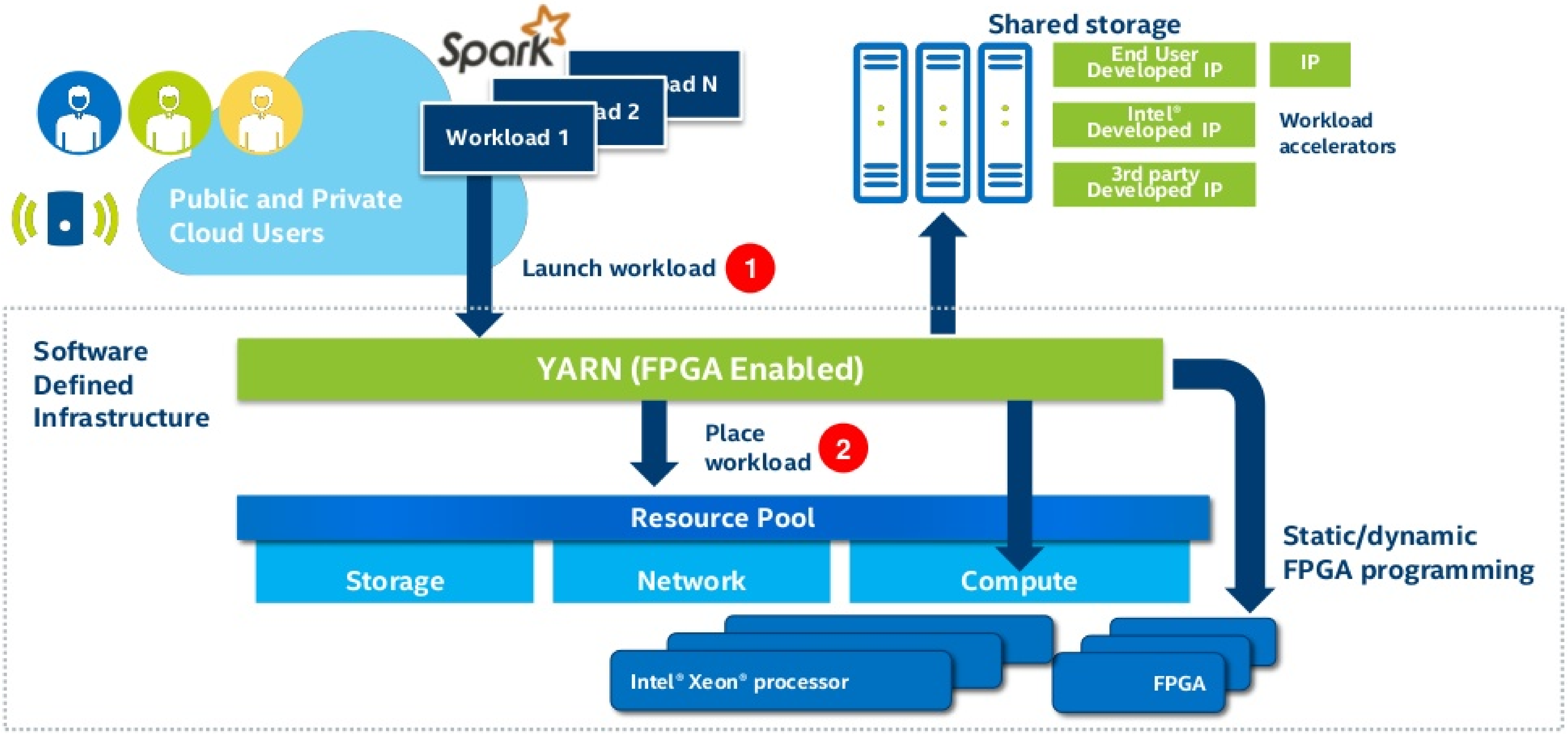
Running a FPGA accelerated app

1. Prepare host with required software and libraries
2. Write/modify application which calls the accelerator API through the shim layer
3. Use FPGA Runtime CLI to flash FPGA device with FPGA IP corresponding to the accelerator API
4. Run application



Workloads
Analytics
Networking
Compression
Genomics
User Application
SHIMs to Key Industry Frameworks
Accelerator API
Accelerator Support Software
Intel Software Infrastructure
CPU
Intel Hardware Infrastructure
FPGA IP
FPGA

Vision of FPGAs in the Data Center



New Options to Spark User Interface

```
spark-submit
   ...
   --conf spark.executor.fpga.type=MCP | DCP
   --conf spark.executor.fpga.ip=AFU1_UUID:AFU1_CNT
   --conf spark.executor.fpga.ip=AFU2_UUID:AFU2_CNT
   ...
   --executor 10
```

--fpga.type: String to filter devices that qualify resource requirements

--fpga.ip: Pair of IP UUID and IP count, colon delimiter, repeatable

Spark AM Modifications

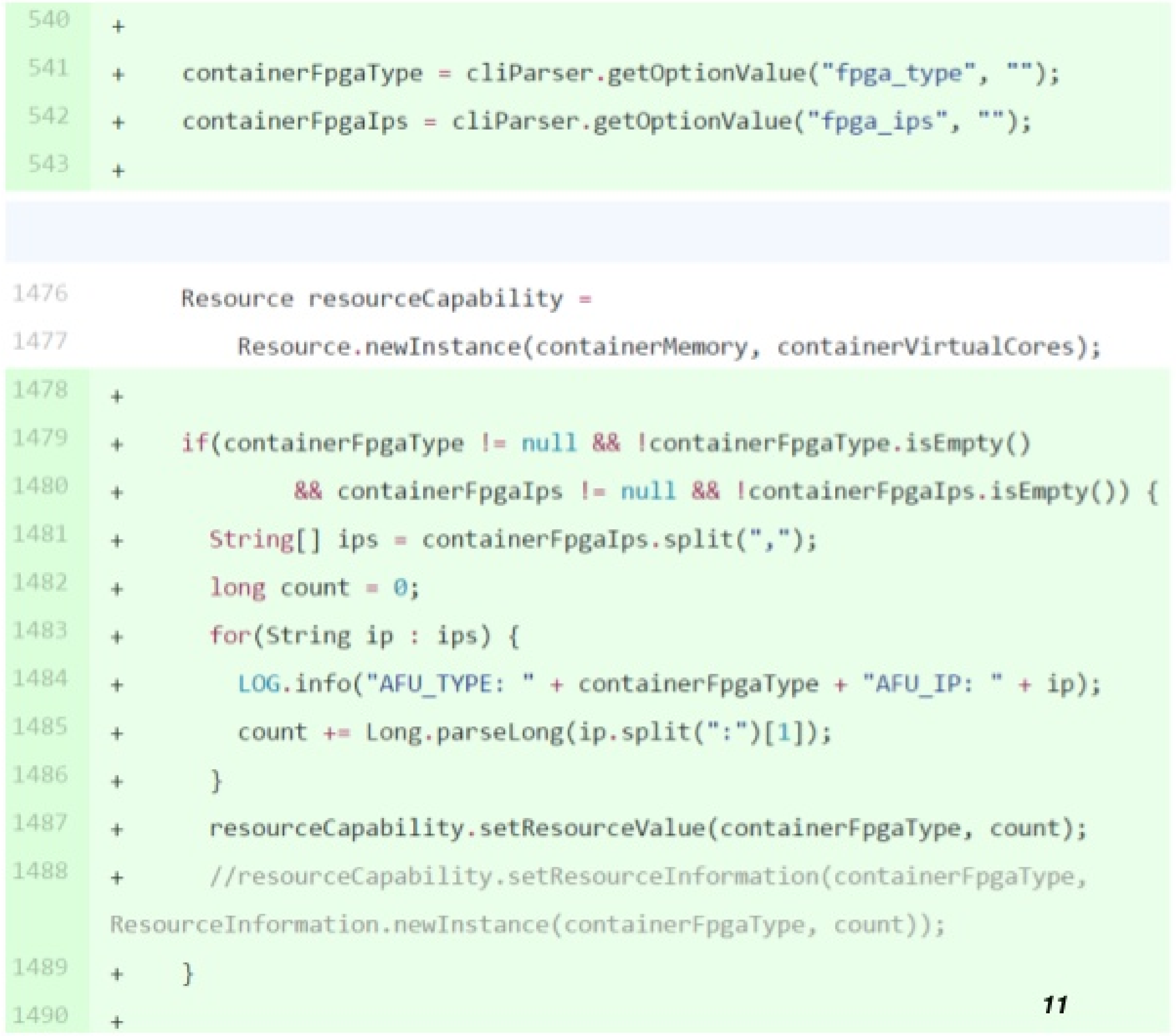
Set FPGA constraints based on YARN's New Resource model

- YARN-3926
- Example

spark-submit
  ...
  --conf spark.executor.fpga.type=MCP
  --conf spark.executor.fpga.ip=ANALYTICS:1
  --conf spark.executor.fpga.ip=GENOMICS:1
  ...
  --executor 10

fpga_type = MCP
fpga_ips = ANALYTICS:1,GENOMICS:1

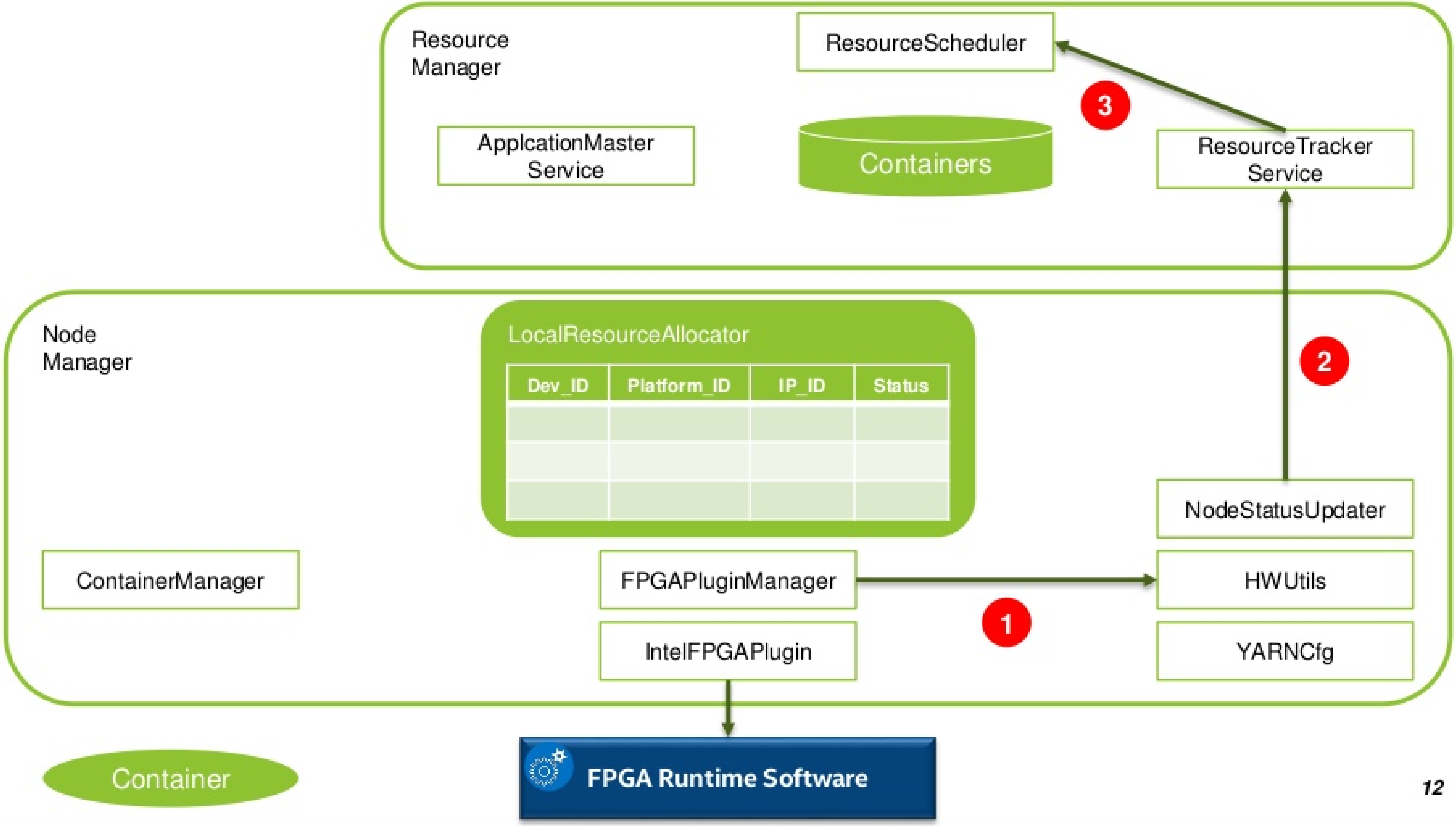
resourceCapability.setResourceValue("MCP", 2)



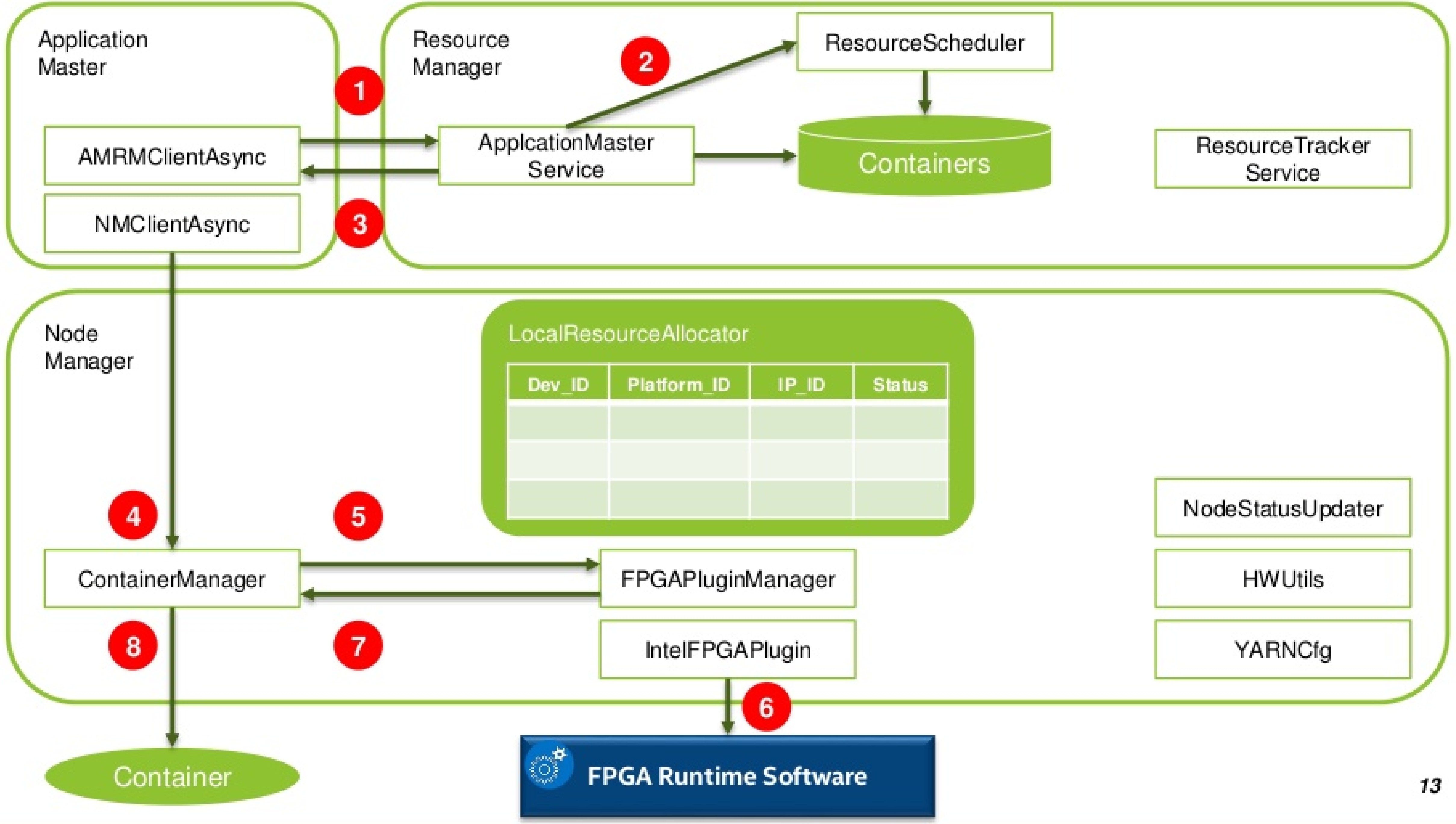
```
+
+    containerFpgaType = cliParser.getOptionValue("fpga_type", "");
+    containerFpgaIps = cliParser.getOptionValue("fpga_ips", "");
+
```

```
      Resource resourceCapability =
          Resource.newInstance(containerMemory, containerVirtualCores);
+
+    if(containerFpgaType != null && !containerFpgaType.isEmpty()
+            && containerFpgaIps != null && !containerFpgaIps.isEmpty()) {
+      String[] ips = containerFpgaIps.split(",");
+      long count = 0;
+      for(String ip : ips) {
+        LOG.info("AFU_TYPE: " + containerFpgaType + "AFU_IP: " + ip);
+        count += Long.parseLong(ip.split(":")[1]);
+      }
+      resourceCapability.setResourceValue(containerFpgaType, count);
+      //resourceCapability.setResourceInformation(containerFpgaType,
ResourceInformation.newInstance(containerFpgaType, count));
+    }
+
```

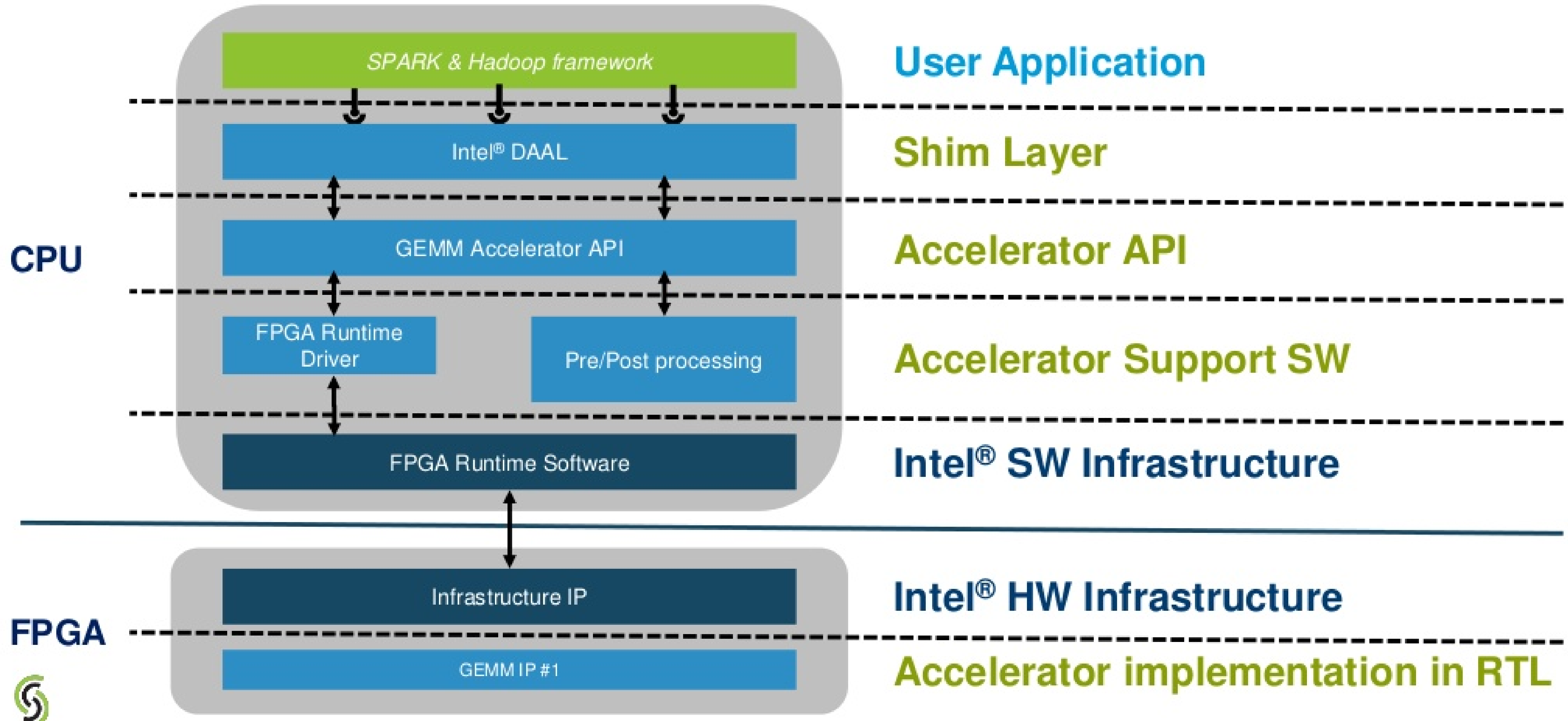
Intel FPGA Software Enabling on YARN – YARN-5983



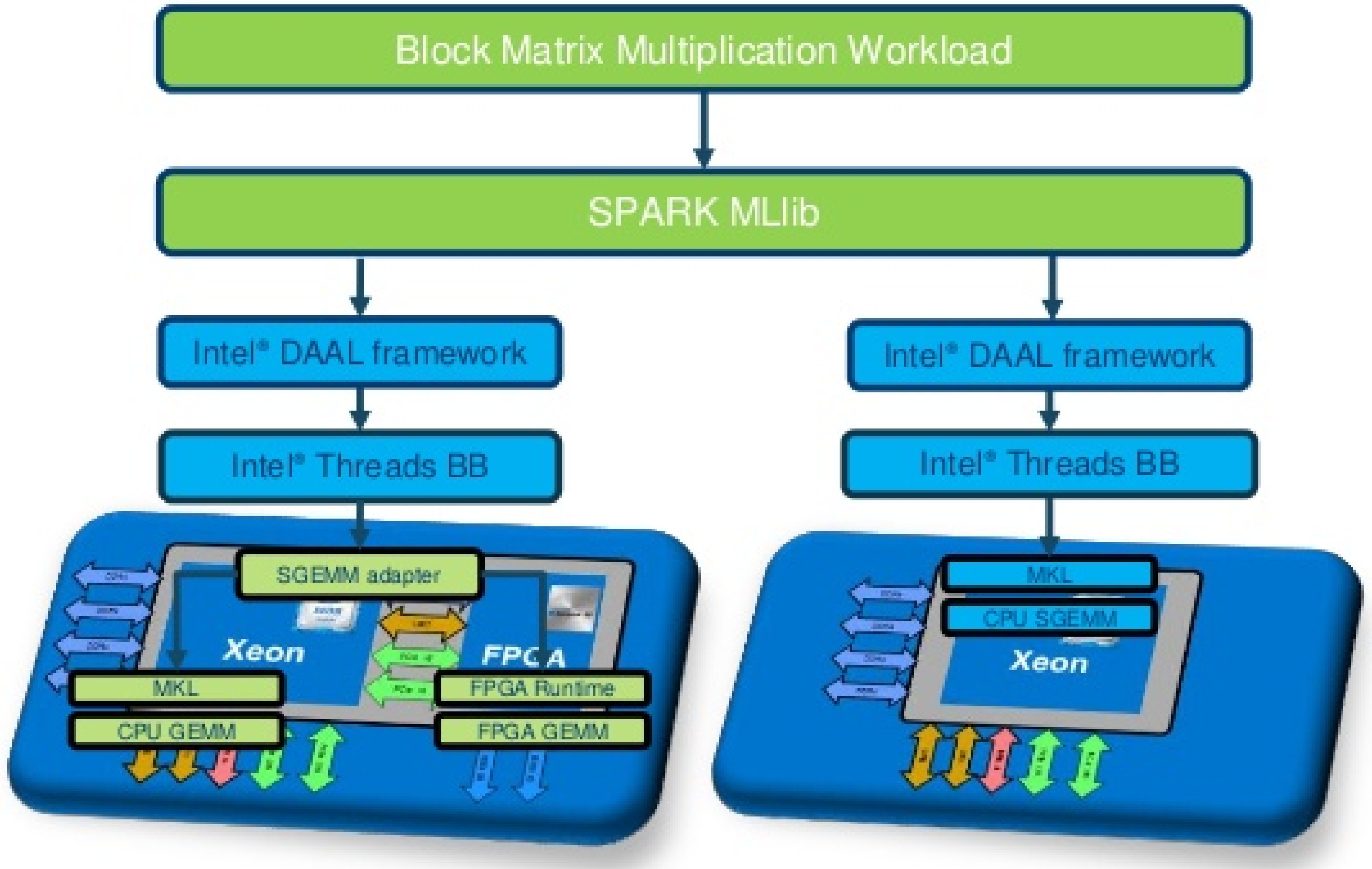
Launching and Placing Workloads



Example: Software Stack for GEMM offloading



Xeon®+FPGA with Intel® DAAL integration

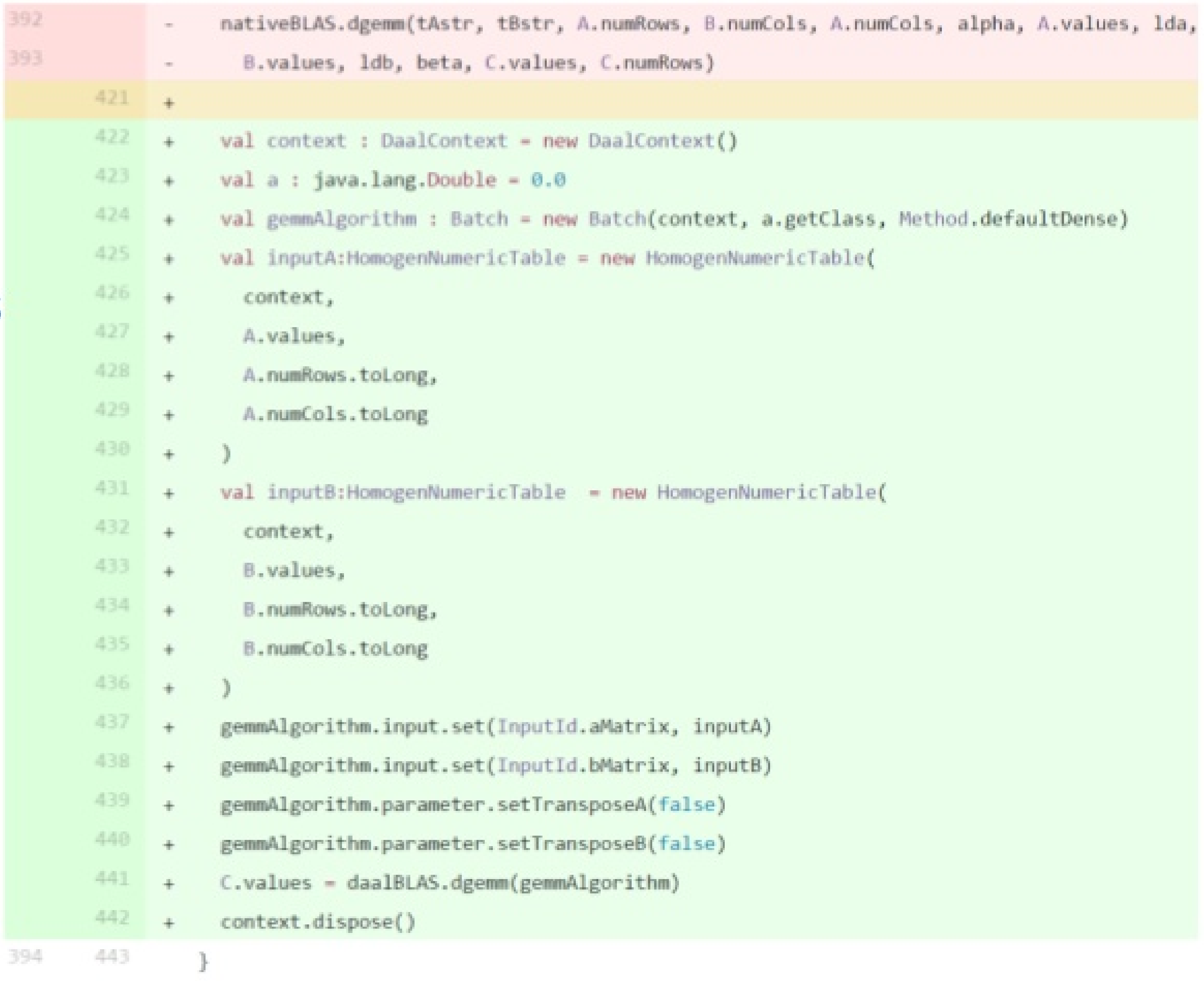


- Users/Developers completely agnostic about presence of FPGA on the server.
- Intel® DAAL framework is aware of the presence of the FPGA on the platform.
- SGEMM adapter schedules jobs on the FPGA if is available else call MKL GEMM on a separate thread.

Spark MLlib Modifications to offload GEMM

Accelerating linalg.BLAS.gemm()

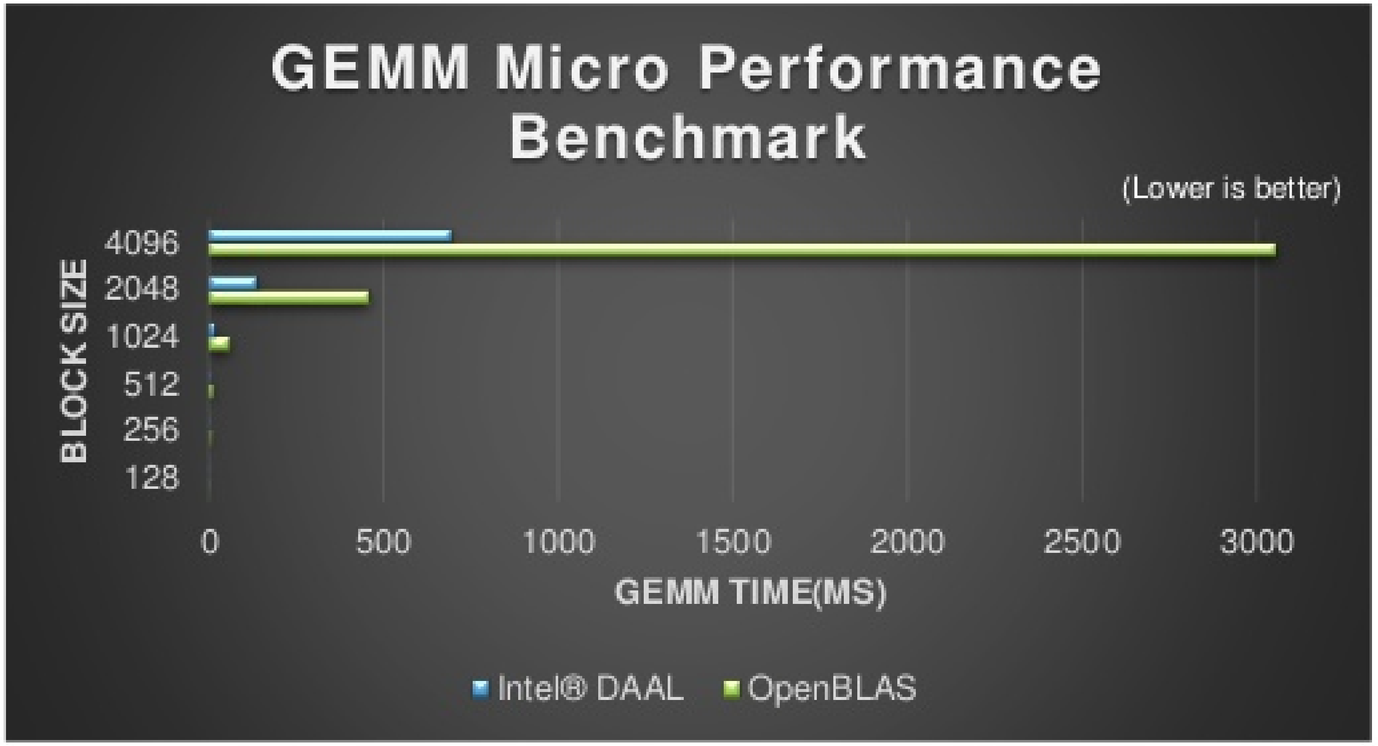
- Create DAAL context
- Instantiate Batch processing class for GEMM
- Set input matrices
- Execute GEMM through DAAL
- Retrieve and set GEMM result

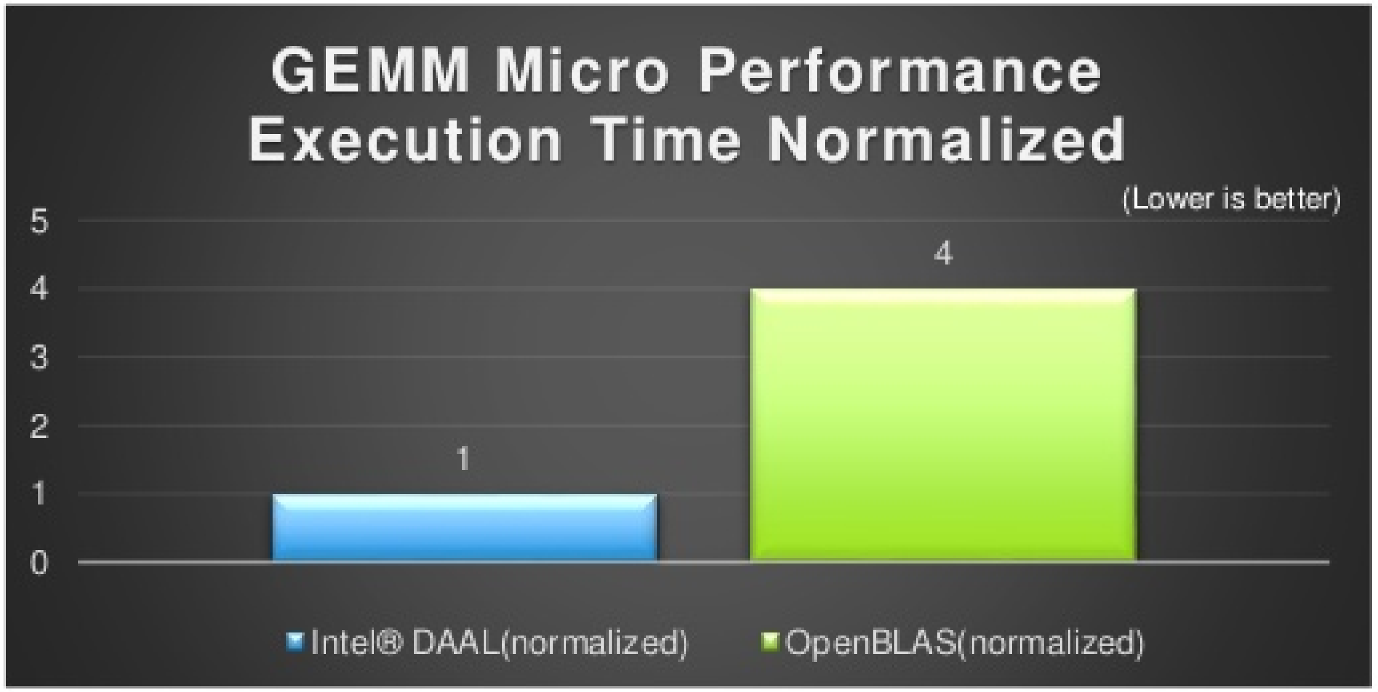


```
-   nativeBLAS.dgemm(tAstr, tBstr, A.numRows, B.numCols, A.numCols, alpha, A.values, lda,
-     B.values, ldb, beta, C.values, C.numRows)
+
+   val context : DaalContext = new DaalContext()
+   val a : java.lang.Double = 0.0
+   val gemmAlgorithm : Batch = new Batch(context, a.getClass, Method.defaultDense)
+   val inputA:HomogenNumericTable = new HomogenNumericTable(
+     context,
+     A.values,
+     A.numRows.toLong,
+     A.numCols.toLong
+   )
+   val inputB:HomogenNumericTable  = new HomogenNumericTable(
+     context,
+     B.values,
+     B.numRows.toLong,
+     B.numCols.toLong
+   )
+   gemmAlgorithm.input.set(InputId.aMatrix, inputA)
+   gemmAlgorithm.input.set(InputId.bMatrix, inputB)
+   gemmAlgorithm.parameter.setTransposeA(false)
+   gemmAlgorithm.parameter.setTransposeB(false)
+   C.values = daalBLAS.dgemm(gemmAlgorithm)
+   context.dispose()
  }
```

Performance test results

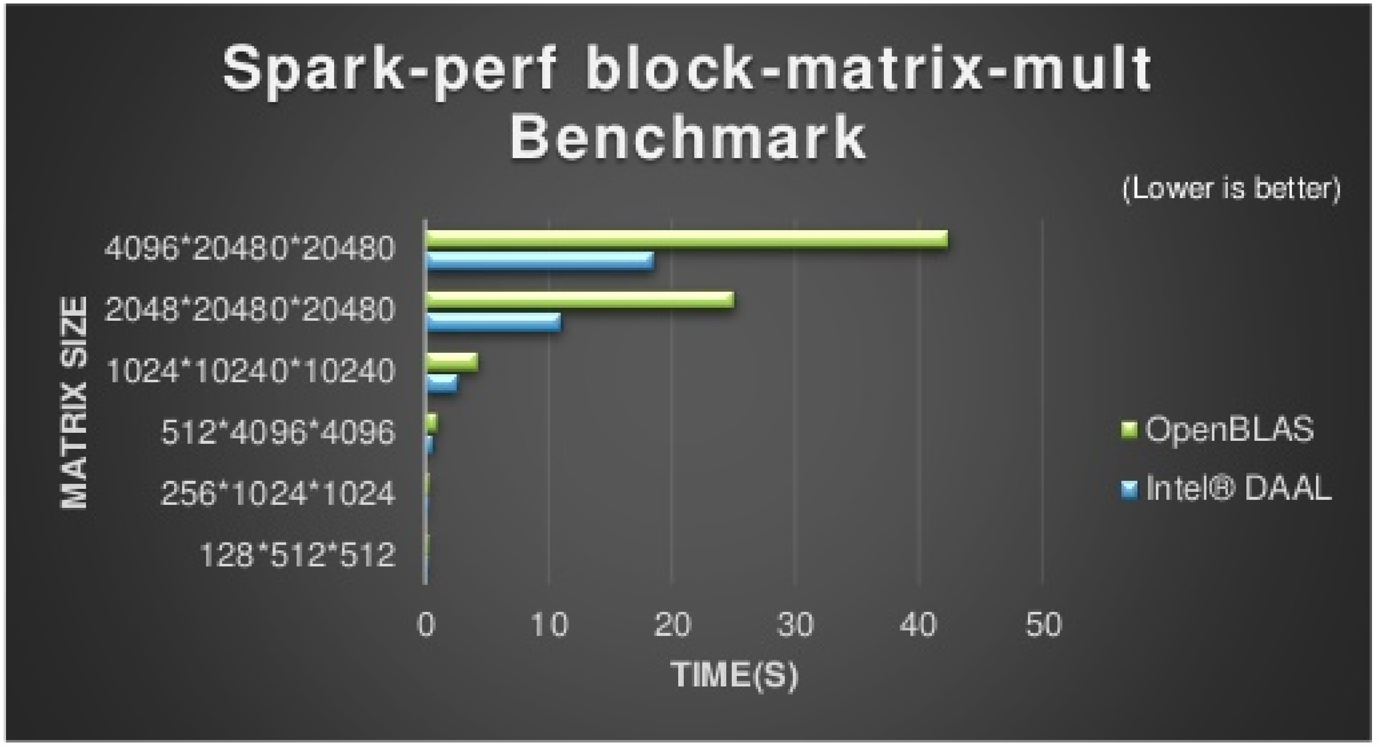
Hardware environment	
CPU model	Genuine Intel® CPU @ 2.40GHz
Memory	250G
FPGA model	BDX+MCP
Network	10Gigabit
Software environment	
Spark basic version	v1.6.3
Observation	Spark + Intel® DAAL (CPU only)
Baseline	Spark + OpenBLAS
Workload	Spark-perf
Executor total memory	160G
Executor total vcore	22
Executor number	1,2,4,8,16
Partition number	executor number *2
Matrix Size	128*512*512 256*1024*1024 512*4096*4096 1024*10240*10240 2048*20480*20480 4096*20480*20480
Block Size	128, 256, 512, 1024, 2048, 4096
Intel® DAAL version	v1.2
Intel® AAL version	v5.0.3

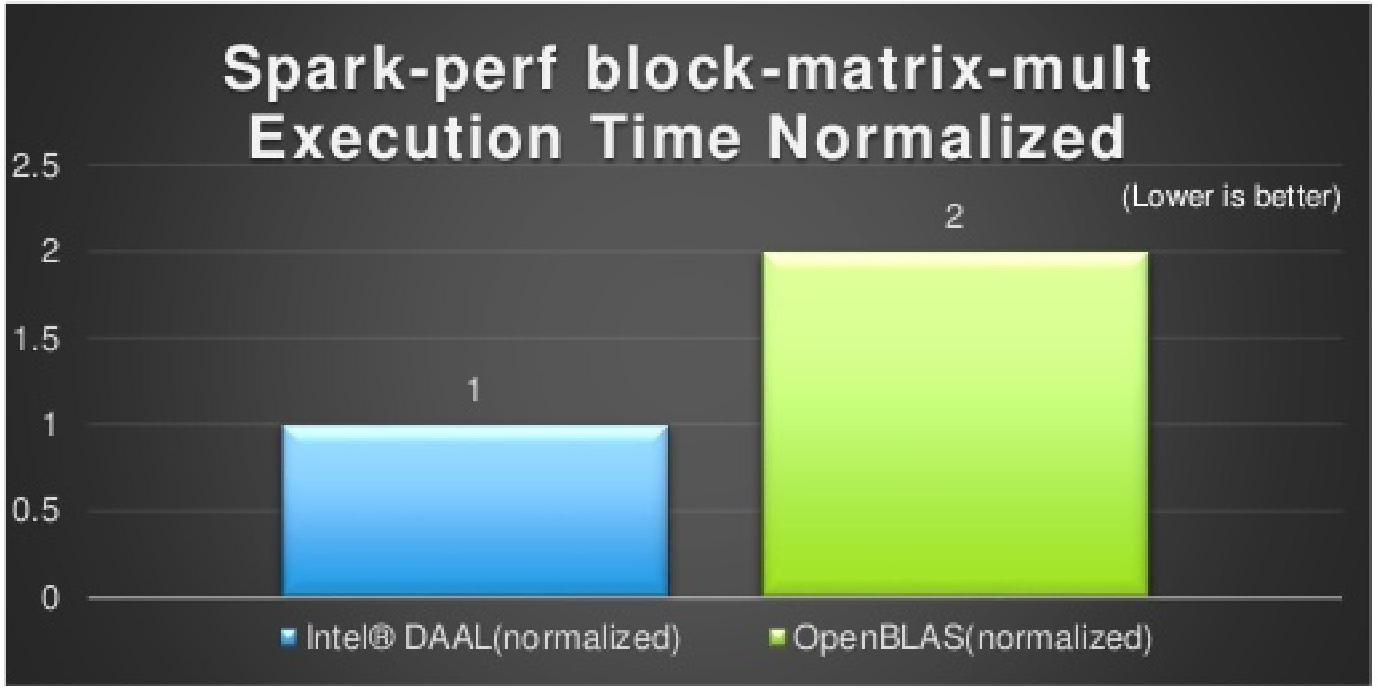




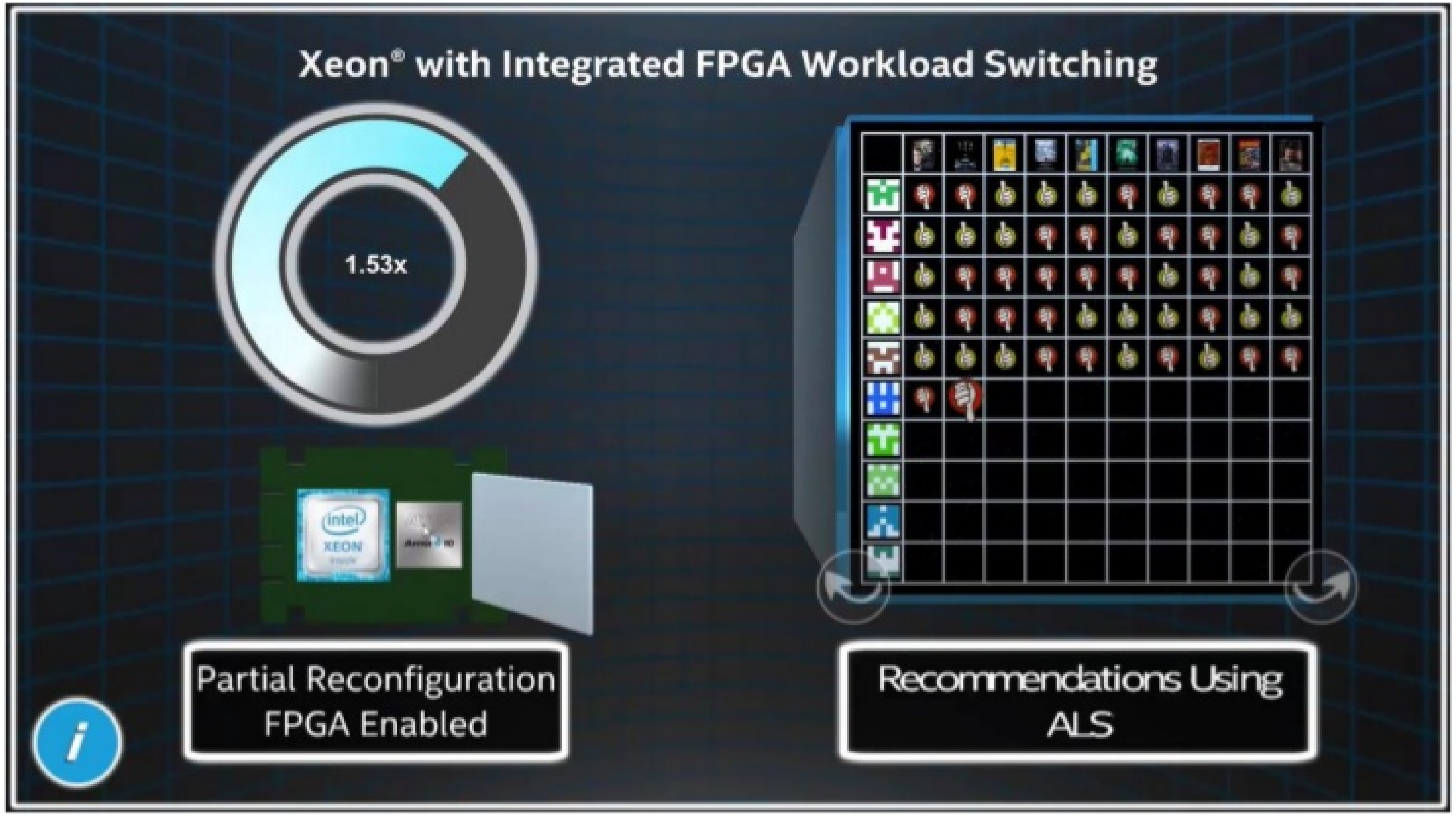
Performance test results

Hardware environment	
CPU model	Genuine Intel® CPU @ 2.40GHz
Memory	250G
FPGA model	BDX+MCP
Network	10Gigabit
Software environment	
Spark basic version	v1.6.3
Observation	Spark + Intel® DAAL (CPU only)
Baseline	Spark + OpenBLAS
Workload	Spark-perf
Executor total memory	160G
Executor total vcore	22
Executor number	1,2,4,8,16
Partition number	executor number *2
Matrix Size	128*512*512 256*1024*1024 512*4096*4096 1024*10240*10240 2048*20480*20480 4096*20480*20480
Block Size	128, 256, 512, 1024, 2048, 4096
Intel® DAAL version	v1.2
Intel® AAL version	v5.0.3





GEMM acceleration on Intel® DAAL with FPGA Acceleration


Xeon® with Integrated FPGA Workload Switching
1.53x
Partial Reconfiguration FPGA Enabled
Recommendations Using ALS

Conclusion

- Intel is leading the compute evolution with innovative products
- Intel also provides software stack solutions optimized to its hardware
- Using SparkML with Intel® DAAL/MKL shows better performance compared to OpenBLAS
- Additional performance improvement can be achieved on the same software stack by using the Intel® Xeon®+FPGA platform

SPARK SUMMIT 2017

SPARK SUMMIT 2017

Thank You

zhongyue.nah@intel.com
srivatsan.krishnan@intel.com