



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

Zhongyue Nah, Intel

Srivatsan Krishnan, Intel

Legal Disclaimer

- INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.
- A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.
- Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.
- The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.
- The code names presented in this document are only for use by Intel to identify products, technologies, or services in development, that have not been made commercially available to the public, i.e., announced, launched or shipped. They are not "commercial" names for products or services and are not intended to function as trademarks.
- Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.
- Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.
- Copies of documents which have an order number and are referenced in this document, or other Intel literature may be obtained by calling 1-800-548-4725 or by visiting Intel's website at <http://www.intel.com/design/literature.htm>.
- Intel is a trademark of Intel Corporation in the US and other countries.
- Copyright © 2017 Intel Corporation. All rights reserved.
- * Other brands and names may be claimed as the property of others.

- 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

What is FPGA

1000s of hard DSPs (floating-point units)

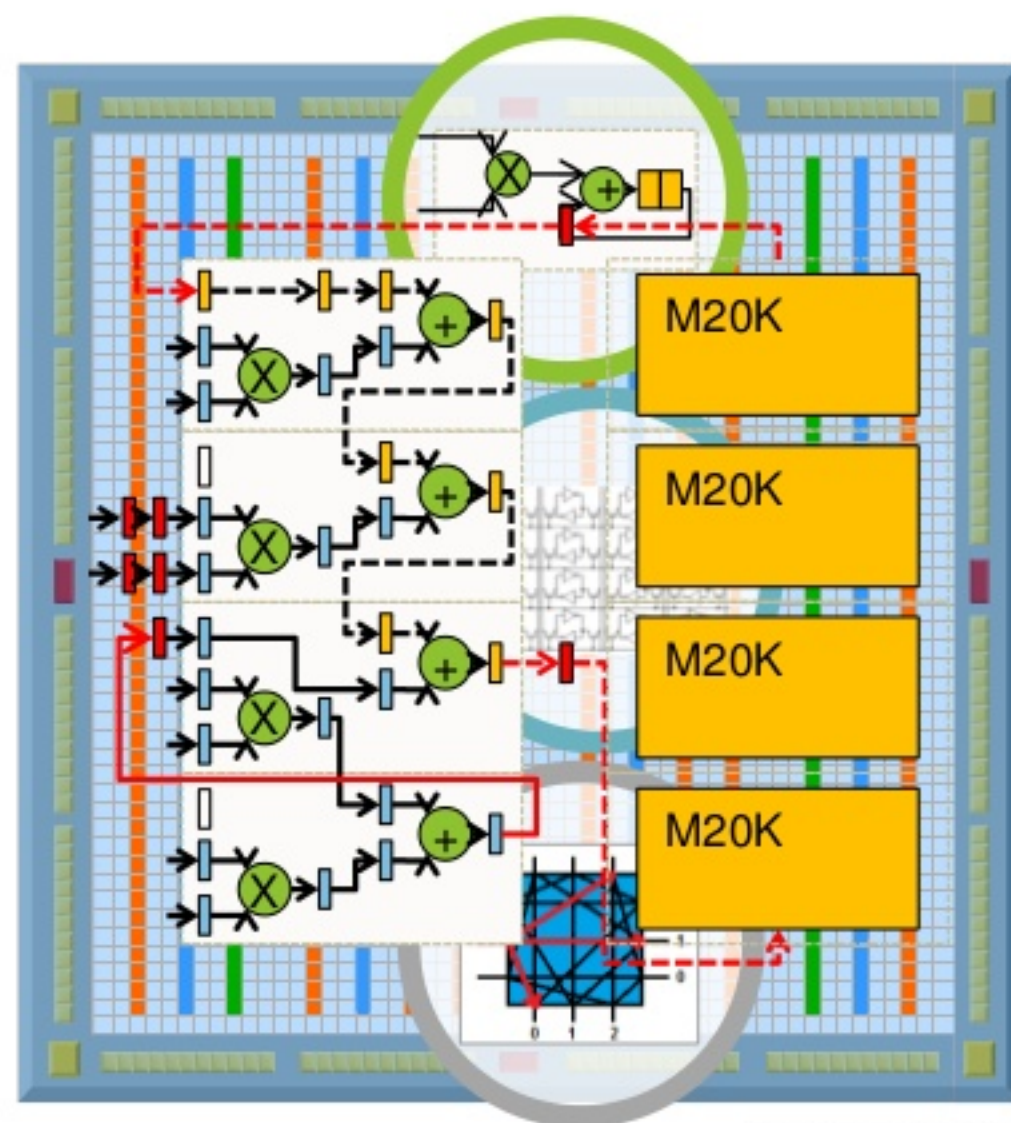
1000s of Hard “M20K” SRAMs (2.5KB/SRAM)

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

Fixed Function ACCELERATION



and/
or

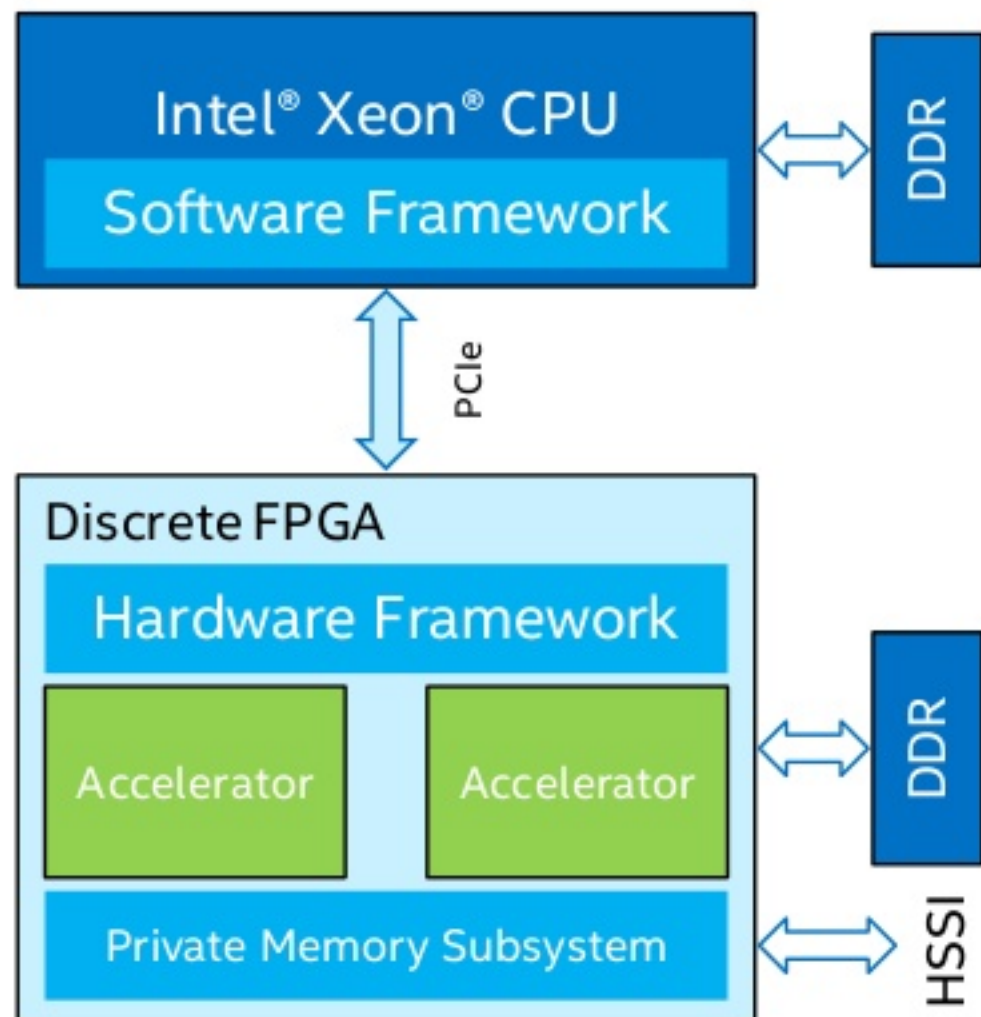
Standard
Workloads

Built-in Standard
platform acceleration,
Highly Optimized

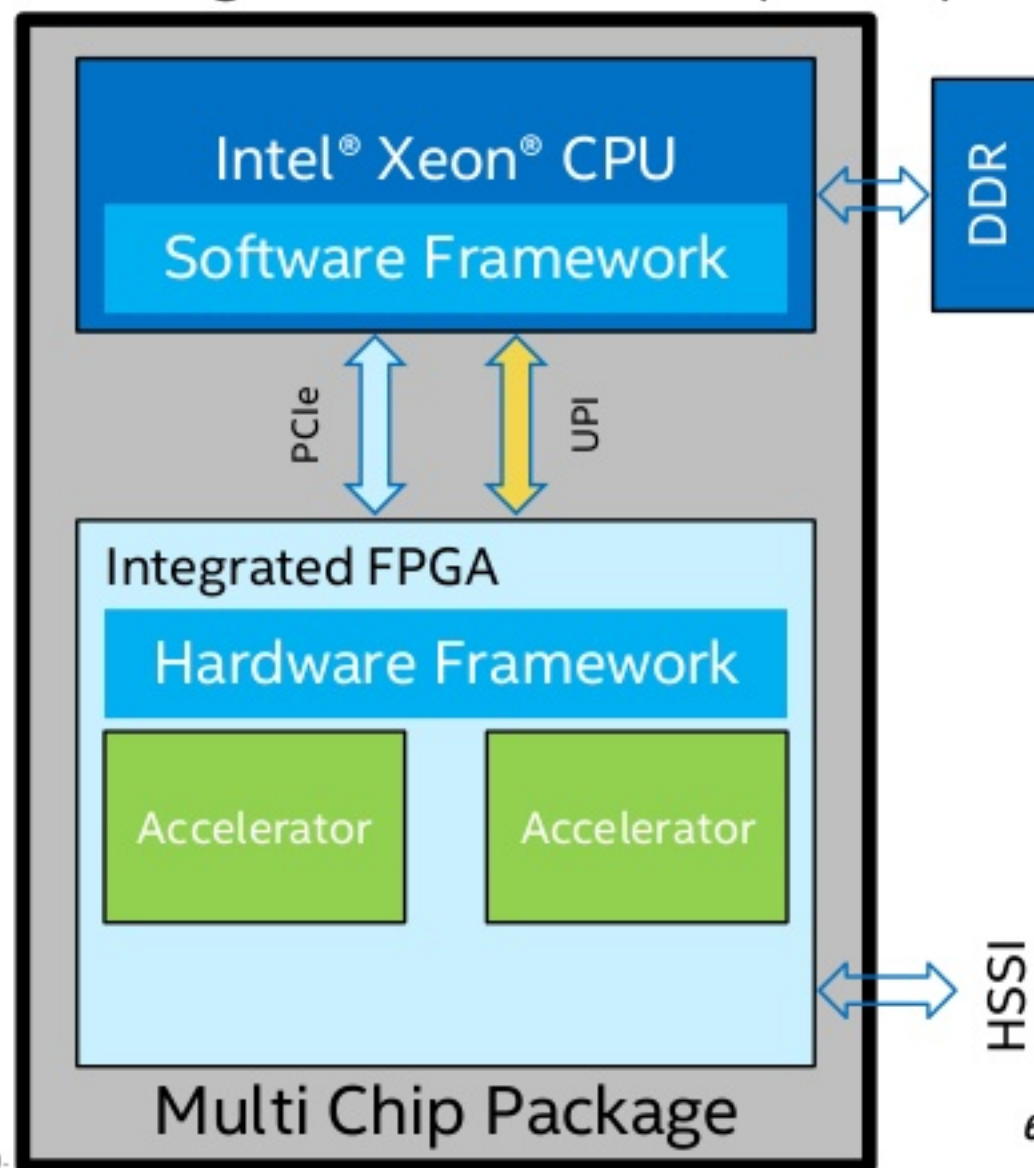
Fixed
HW Acceleration

Discrete and Integrated FPGA platforms

Discrete Platform (DCP)

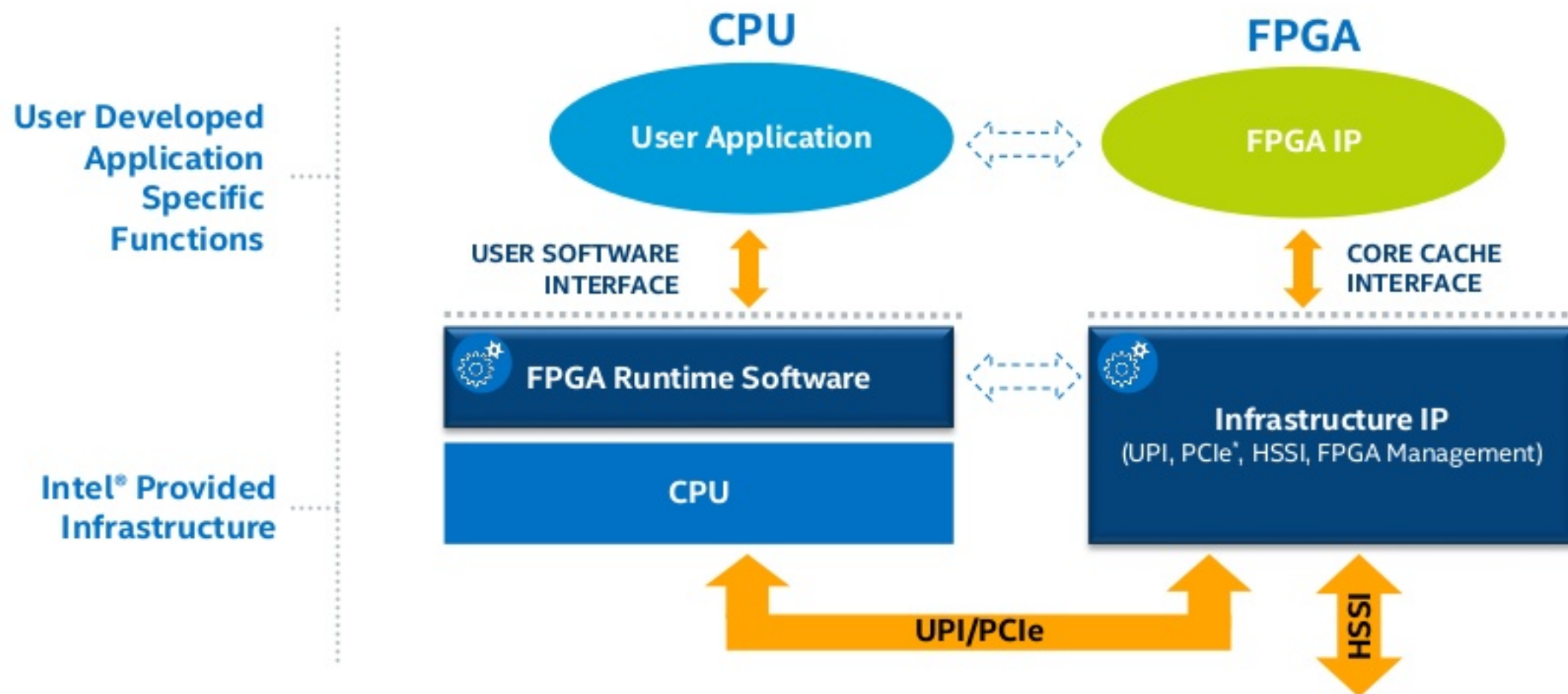


Intel® Xeon®+FPGA Integrated Platform (MCP)*



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

Intel® Accelerator Abstraction Layer (AAL)

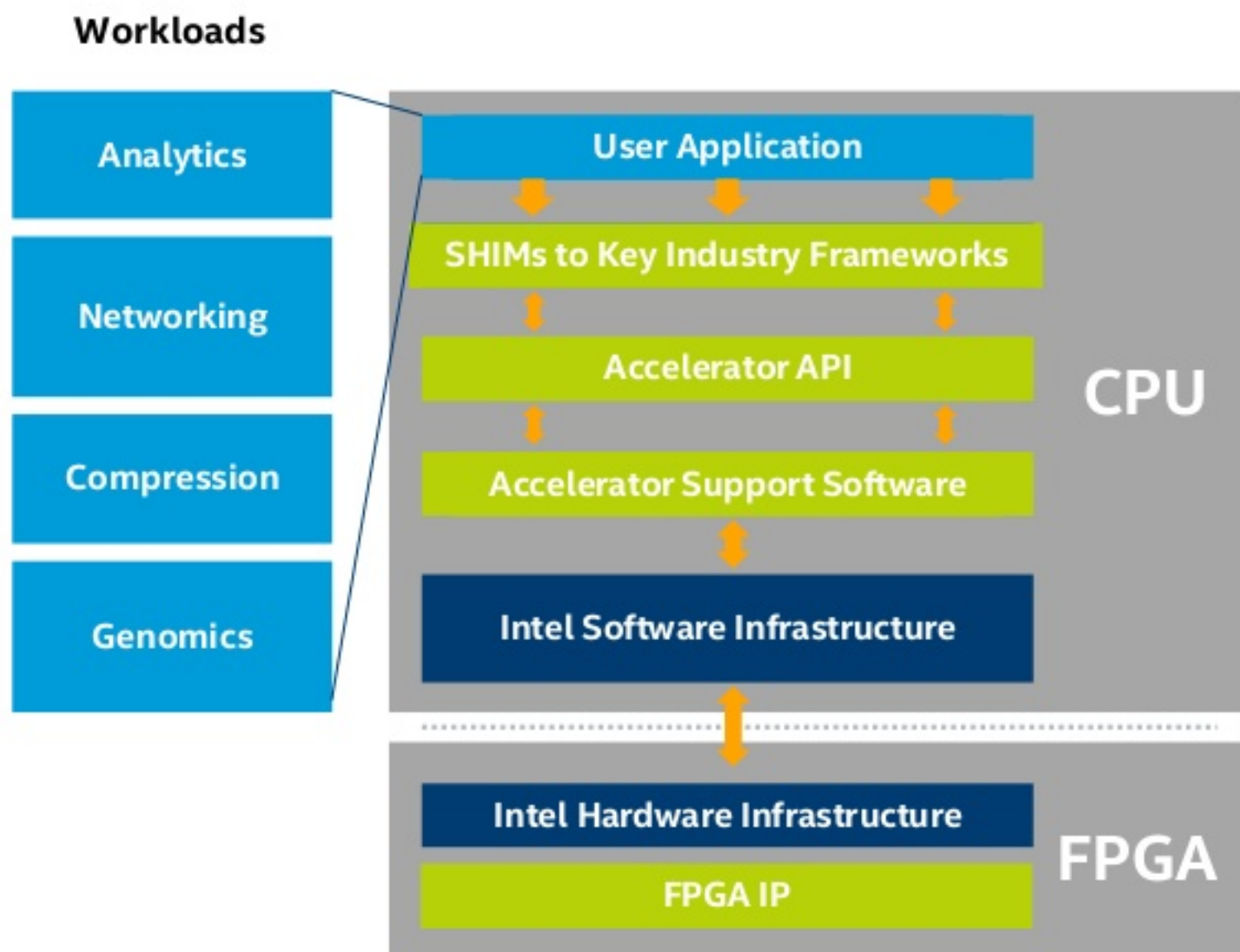


 = New blocks that simplify code development.

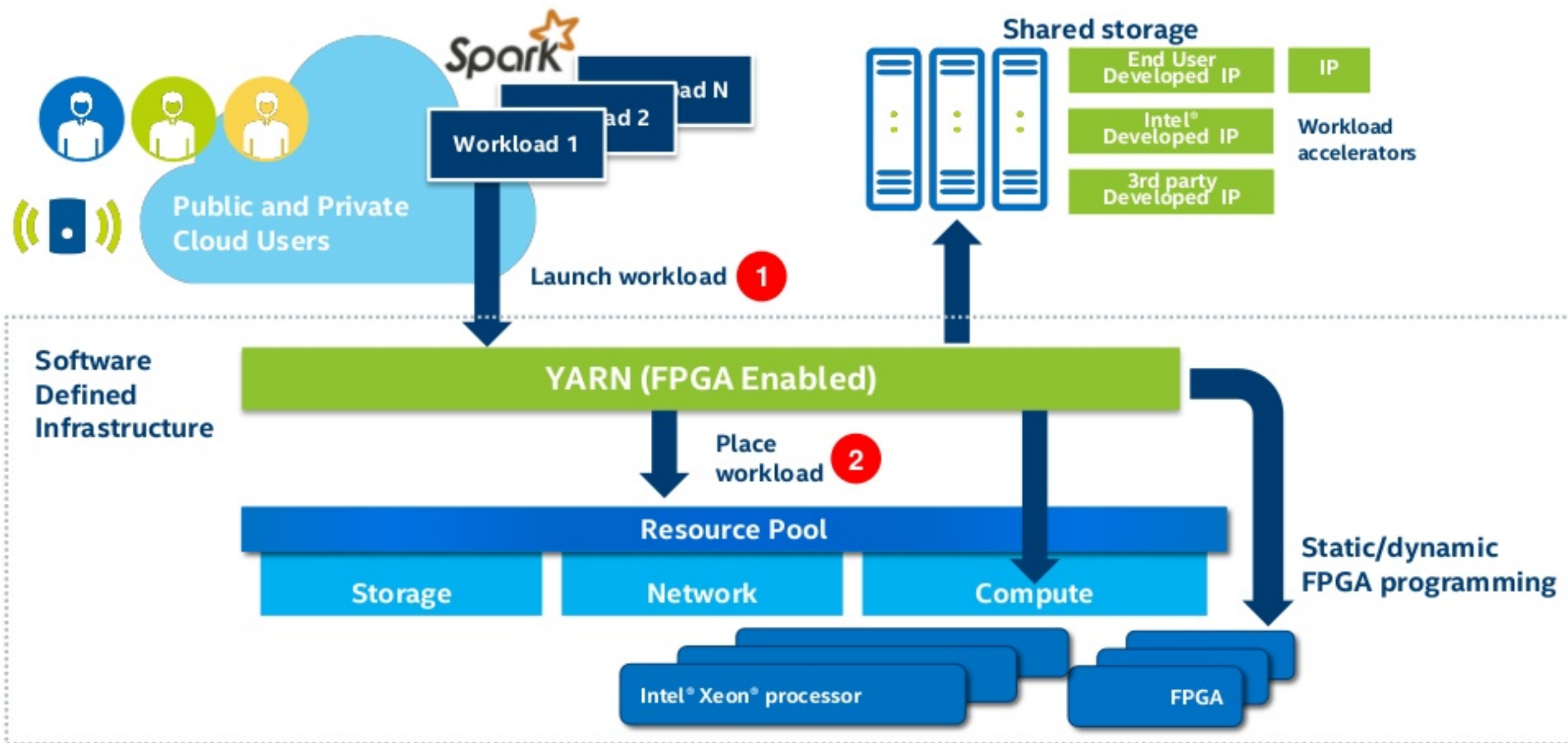
Single Node Software Stack

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



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

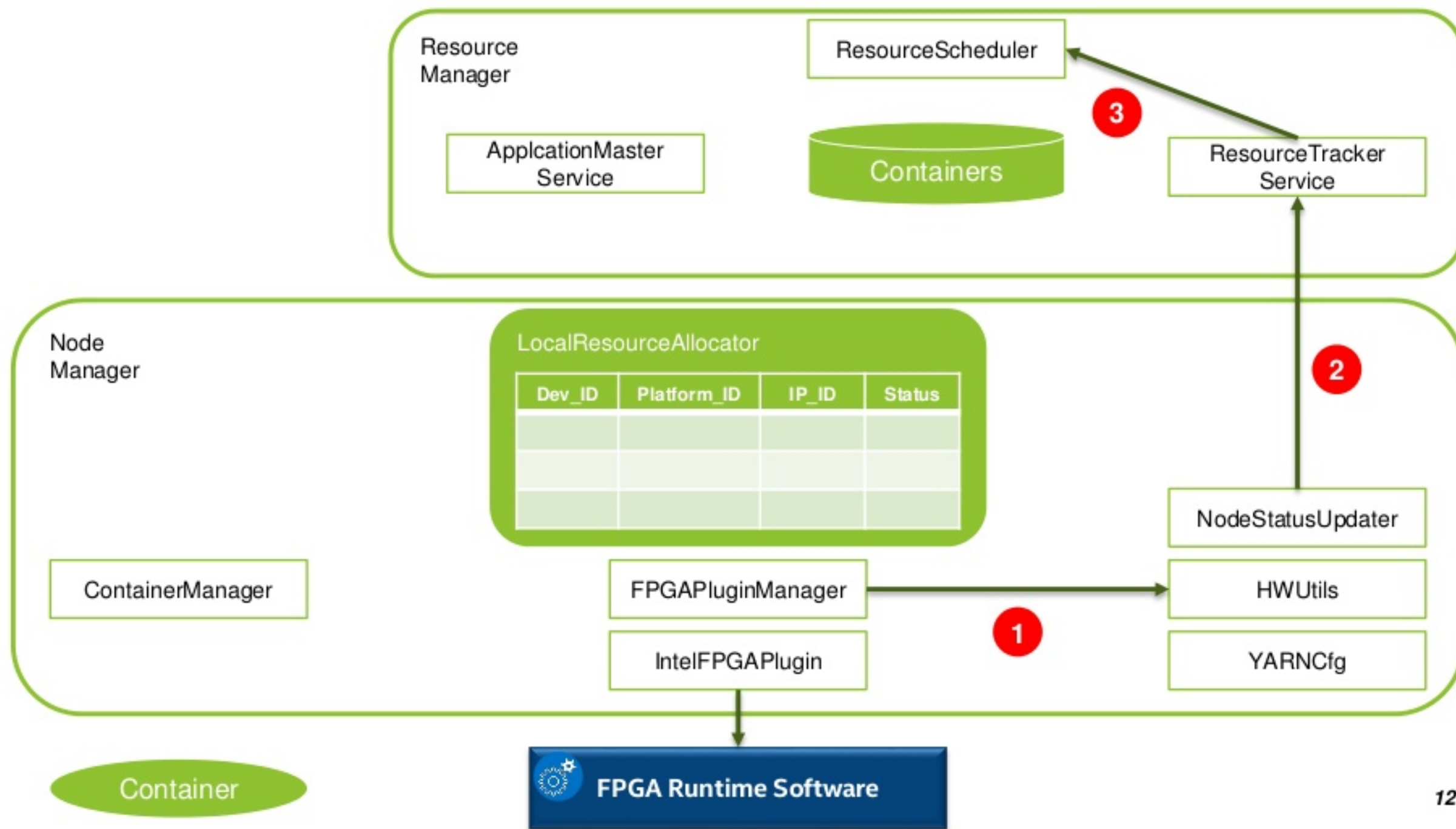


resourceCapability.setResourceValue("MCP", 2)

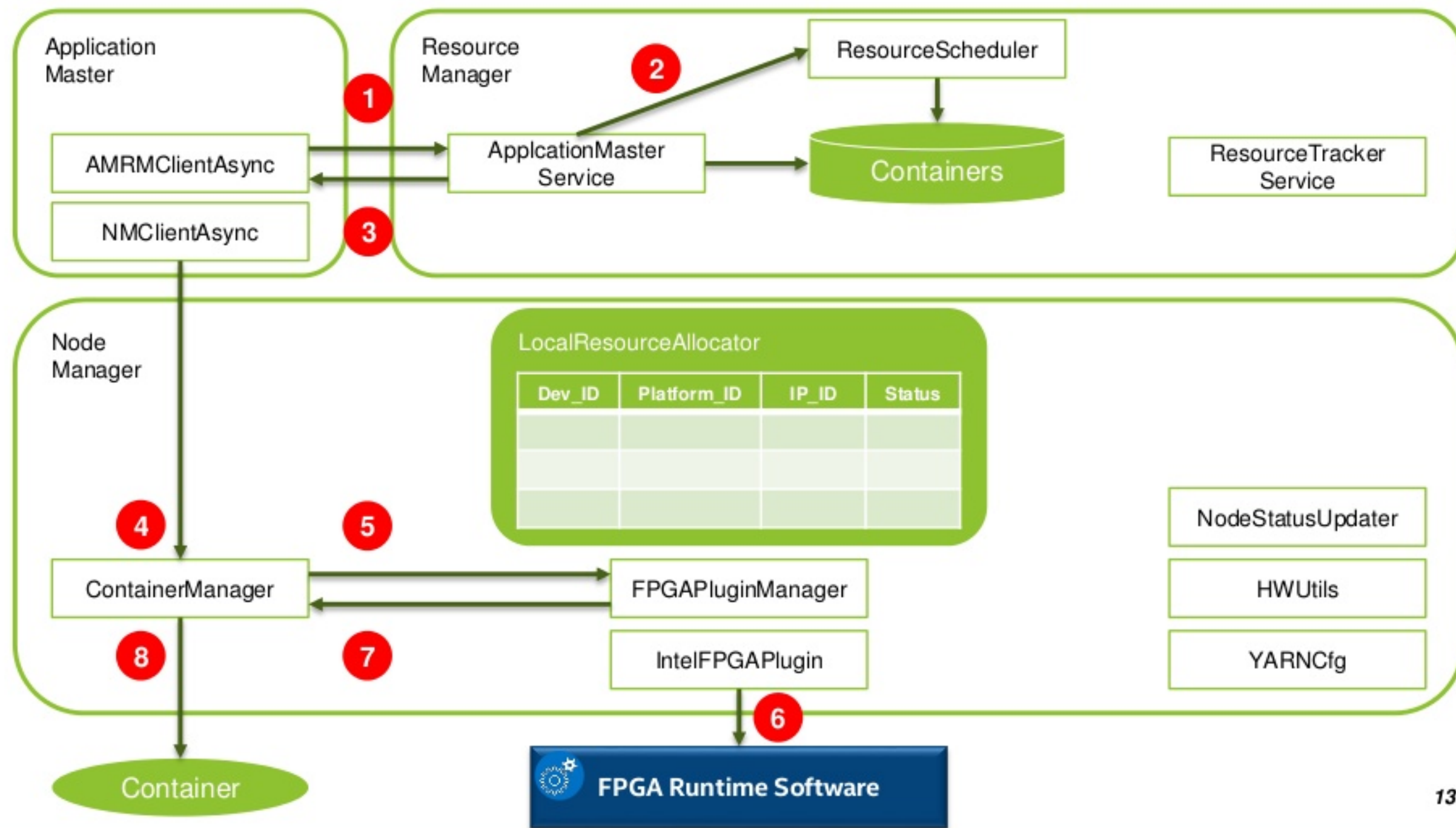
```
540 +  
541 + containerFpgaType = cliParser.getOptionValue("fpga_type", "");  
542 + containerFpgaIps = cliParser.getOptionValue("fpga_ips", "");  
543 +
```

```
1476 Resource resourceCapability =  
1477     Resource.newInstance(containerMemory, containerVirtualCores);  
1478 +  
1479 + if(containerFpgaType != null && !containerFpgaType.isEmpty()  
1480 +     && containerFpgaIps != null && !containerFpgaIps.isEmpty()) {  
1481 +     String[] ips = containerFpgaIps.split(",");  
1482 +     long count = 0;  
1483 +     for(String ip : ips) {  
1484 +         LOG.info("AFU_TYPE: " + containerFpgaType + "AFU_IP: " + ip);  
1485 +         count += Long.parseLong(ip.split(":")[1]);  
1486 +     }  
1487 +     resourceCapability.setResourceValue(containerFpgaType, count);  
1488 +     //resourceCapability.setResourceInformation(containerFpgaType,  
1489 +         ResourceInformation.newInstance(containerFpgaType, count));  
1490 + }
```

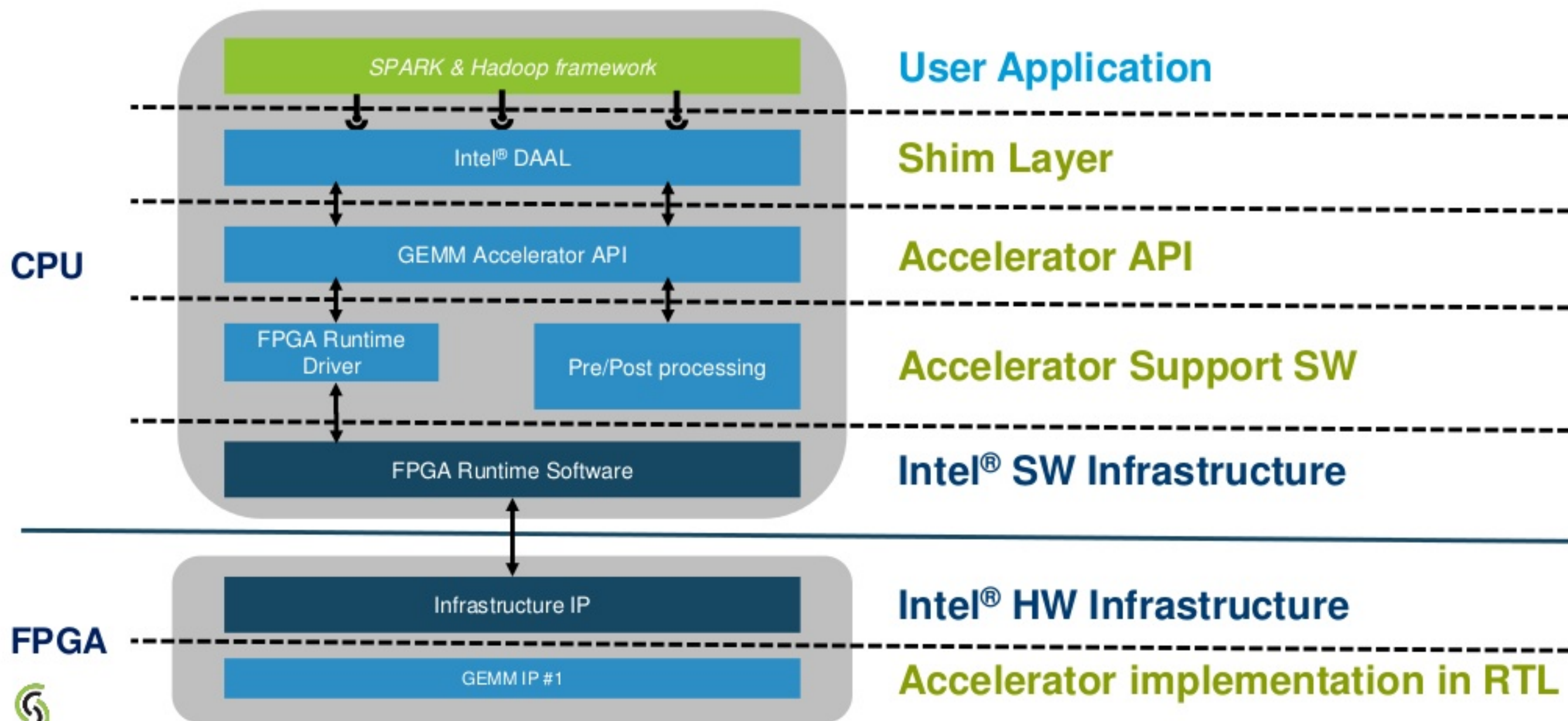

Intel FPGA Software Enabling on YARN – YARN-5983



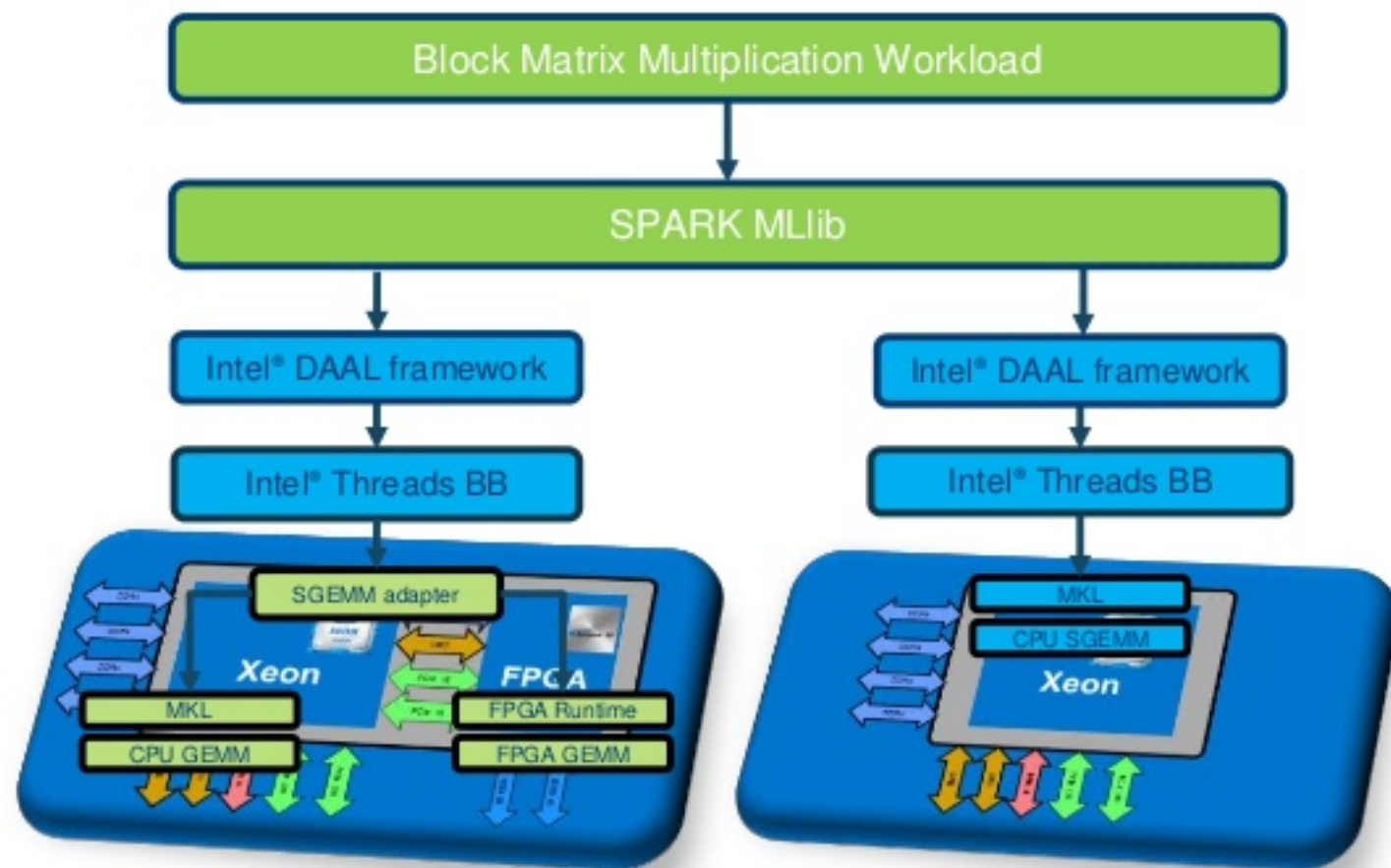
Launching and Placing Workloads



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 it 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

```
392 - nativeBLAS.dgemm(tAstr, tBstr, A.numRows, B.numCols, A.numCols, alpha, A.values, lda,
393 -   B.values, ldb, beta, C.values, C.numRows)
421 +
422 +   val context : DaalContext = new DaalContext()
423 +   val a : java.lang.Double = 0.0
424 +   val gemmAlgorithm : Batch = new Batch(context, a.getClass, Method.defaultDense)
425 +   val inputA:HomogenNumericTable = new HomogenNumericTable(
426 +     context,
427 +     A.values,
428 +     A.numRows.toLong,
429 +     A.numCols.toLong
430 +   )
431 +   val inputB:HomogenNumericTable = new HomogenNumericTable(
432 +     context,
433 +     B.values,
434 +     B.numRows.toLong,
435 +     B.numCols.toLong
436 +   )
437 +   gemmAlgorithm.input.set(InputId.aMatrix, inputA)
438 +   gemmAlgorithm.input.set(InputId.bMatrix, inputB)
439 +   gemmAlgorithm.parameter.setTransposeA(false)
440 +   gemmAlgorithm.parameter.setTransposeB(false)
441 +   C.values = daalBLAS.dgemm(gemmAlgorithm)
442 +   context.dispose()
394 443 }
```

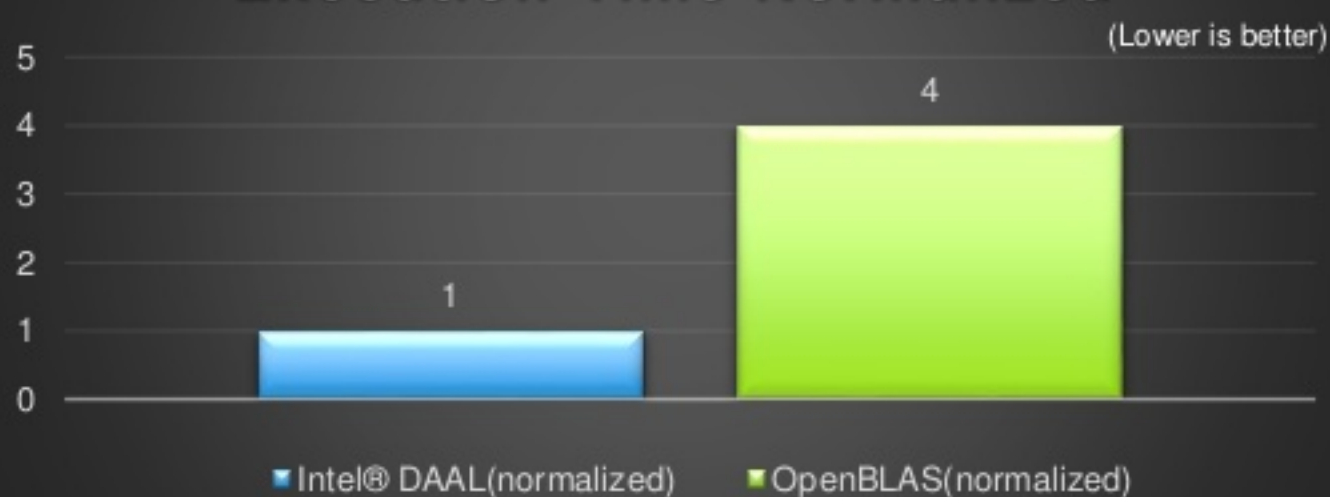
Performance test results

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 Micro Performance Benchmark



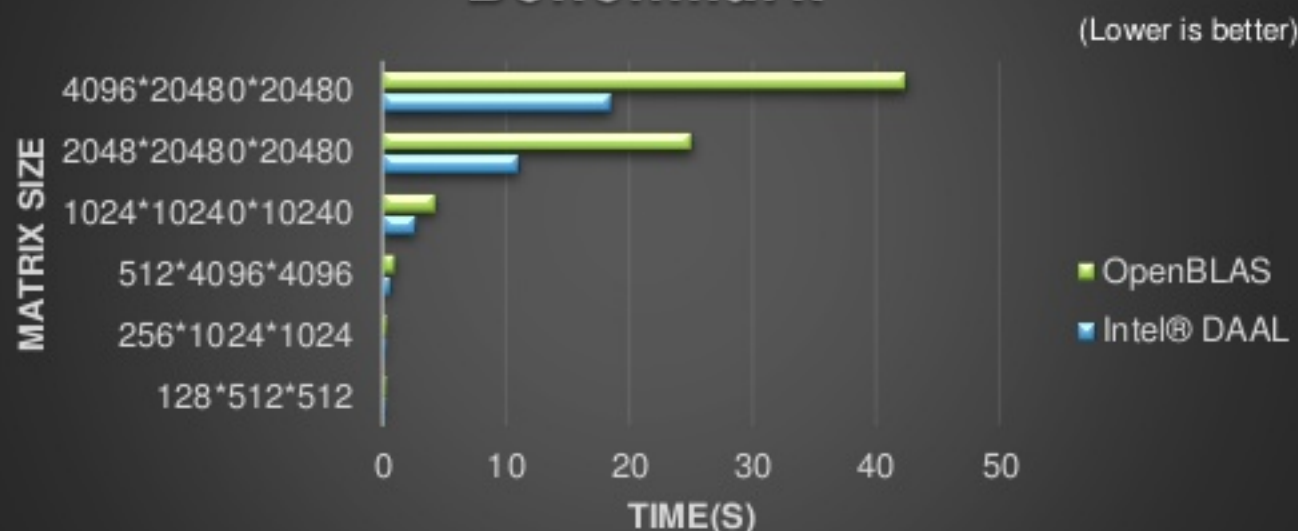
GEMM Micro Performance Execution Time Normalized



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

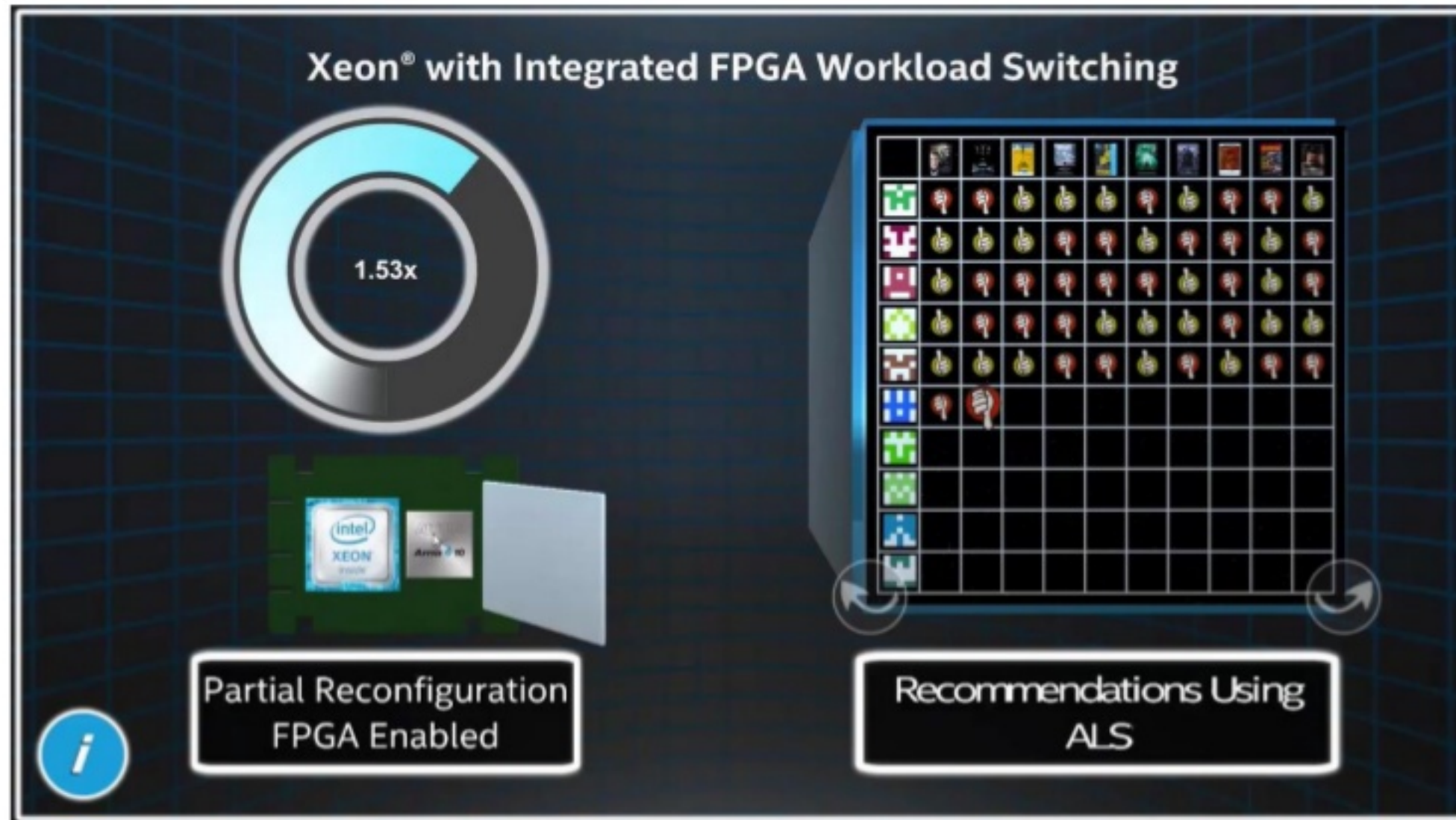
Spark-perf block-matrix-mult Benchmark



Spark-perf block-matrix-mult Execution Time Normalized



GEMM acceleration on Intel® DAAL with FPGA Acceleration



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



Thank You

zhongyue.nah@intel.com

srivatsan.krishnan@intel.com