Deep Learning Inference with Intel PAC on Dell EMC Infrastructure – Part II

Tech Note by

David Ojika
(University of Florida)
Bhavesh Patel
(Dell EMC)
Shawn Slockers
(Intel)

Summary

This is the second part of a three-part series on deep learning inferencing on FPGAs. In part 1, we presented the Intel PAC with Intel Acceleration Stack integrated with a Dell EMC PowerEdge server running image classification

Here, in part 2, we demonstrate the performance of the newly improved ResNet-50 image classification model.

Recap

Before we dive into the details, let's briefly go over what is deep learning inference and why we may use a field-programmable gate-array (FPGA) accelerator like the Intel Programmable Accelerator Card (PAC) to speed up the process.

Deep learning is a class of machine learning that learns a neural network model from sample data sets over a series of training iterations and loss function [1]. The output of this phase, the learned model, is then used to make predictions on new data. While this model-learning process lends itself well to single-instruction multiple data (SIMD) type computing with coarse-grain architectures like many-core CPUs and GPUs, the inferencing process is much more amenable to irregular, fine-grain architectures like FPGAs, which allow for greater architectural flexibility to meet specific application requirements: latency, throughput, power, etc. Inferencing is the stage where most enterprises realize the business value of their AI investments.

To accelerate inferencing in resource-constrained servers, the PCIe-based Intel PAC with Arria 10 GX FPGA supports up to 1.5 teraFLOPS (1 trillion floating-point instructions per second) performance within a thermal dissipation power of only 60 Watts. This makes the Intel PAC particularly suited for datacenters and edge computing environments. Combined with the Intel Acceleration Stack and the Intel distribution of OpenVINO, developers can deploy models on the Intel PAC while leveraging unique, built-in hardware features of the Intel PAC including direct I/O and networking.

Figure 1a: Intel PAC with Arria 10 GX FPGA

Figure 1b: Dell EMC R740 PowerEdge server

Recent Optimizations

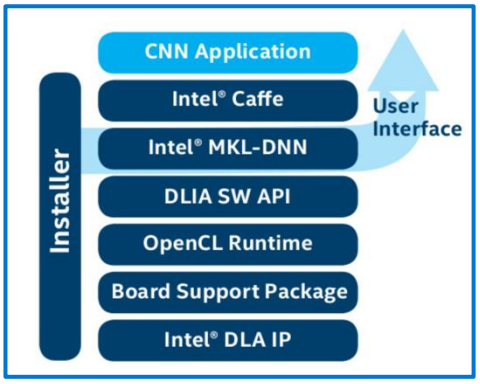

Figure 2: System stack

In part 1 of this blog, we presented the Intel Acceleration Stack and the Intel distribution of OpenVINO both of which are part of the system stack for the Intel PAC. Within the PAC, the Deep Learning Accelerator (DLA), part of the PAC hardware stack, accelerates specific deep learning models, e.g., AlexNet, GoogleNet, ResNet, etc.

Intel has released an updated ResNet-50 binary [2], and also released new versions of OPAE and OpenVINO. With these new improvements, developers can derive increased acceleration in their deep-learning inference workloads with minimal changes to the system stack (Fig 2).

New ResNet-50 Binary

ResNet-50 is a deep learning model for image classification, allowing applications to describe an image with only 3.57% error. Like any other deep neural network, ResNet-50 has input, output, and hidden layers which describe its underlying algorithm through a network of interconnected neurons that propagate information from one layer to the next (Fig 3.)

As new research in deep learning architectures and models continue to evolve, FPGAs are uniquely positioned to incorporate these research advancements down into the hardware without necessitating a new silicon spin. Intel has further optimized the ResNet-50 model for low-bit (FP11) precision inferencing, enabling increased performance of vision applications. Further, this new model optimization at the hardware level, combined with the many software optimizations, provides seamless application integration while significantly increasing performance.

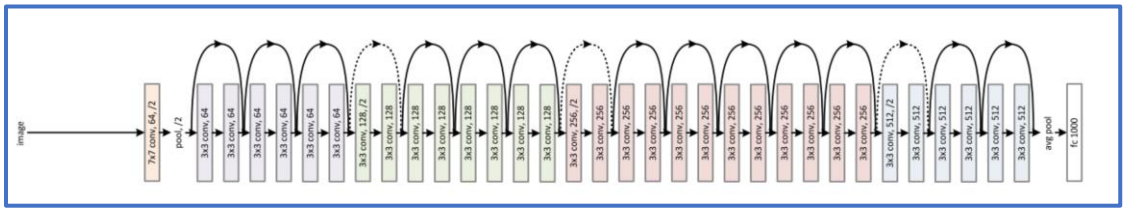

Figure 3: ResNet-50 model architecture.

The ResNet-50 model has 150,528 input neurons; 1,000 output neurons and 50 layers, totaling 3.8 billion operations. With recent improvements in the OpenVINO SDK, the Intel PAC with Arria 10 FPGA can comfortably run this ResNet-50 model at increased performance compared to previously published performance numbers.

Results

We setup the Intel PAC on a Dell EMC R740 PowerEdge server with 2 Intel Xeon Gold 6130 CPU @ 2.10 GHz running CentOS Linux (release 7.6). We present the FPGA performance in comparison with the CPUs, noting the throughput, latency, and energy efficiency achieved across both devices.

Throughput

Throughput is a measure of how fast sets of images are processed every second. Here, we denote this measure as *frames per second* (FPS). Images can be processed in sets of 1 image or more, also referred to as *batch size*.

Fig 4 shows the throughput of the FPGA and CPU for batch sizes 1, 16, and 64 respectively, across different CPU configurations, i.e., number of threads. As indicated, the FPGA performance is consistent for a given batch size regardless of the thread count, indicating the FPGA provides deterministic performance. In practice, only 1 CPU thread (off-load) is necessary to achieve maximum throughput with the FPGA; the rest of the CPU threads are freed to perform other tasks. Conversely, as indicated by the dotted horizonal lines, only when the batch size is as large as 64 and the thread count is as much as 64 does the CPU surpass the FPGA in FPS. In short, doubling the CPU threads (from 32 to 64) drives throughput by a mere 11% increase. For brevity, we have considered thread counts in increasing power of 2. Curious readers are encouraged to try out finer thread counts to reach equivalency in CPU vs FPGA performance.

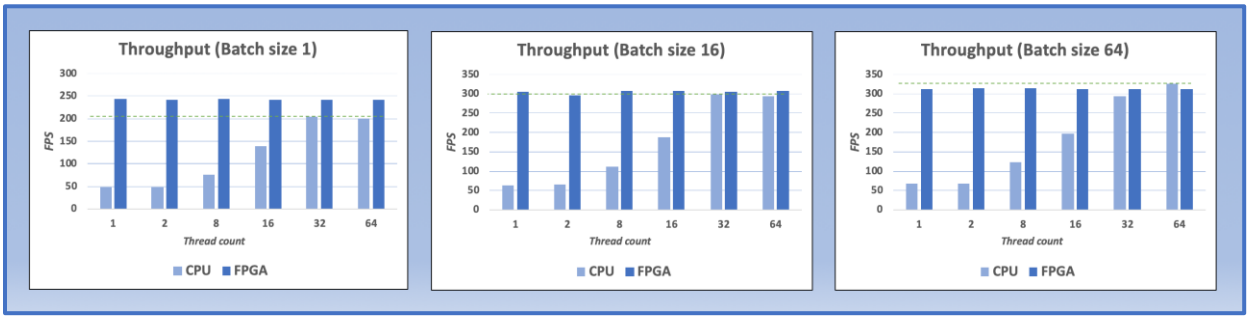

Figure 4: Throughput. In dynamic systems, where the number of available cores may be unknown, the FPGA will provide deterministic performance since only 1 thread is necessary.

Latency

Latency is the time it takes to process a request, i.e., inferencing. Here, a request has a batch size of 1, 16, or 64 as shown in Fig 5. A batch size of 1 is applicable to streaming applications where data must be processed on-the-fly as it is generated. In such a scenario, latency is more important than throughput.

At batch size 1, the latency with the FPGA is 4 ms compared to 21 ms with the CPU. For this comparison, we note that the CPU is configured to execute in single-thread mode, utilizing a single CPU core.

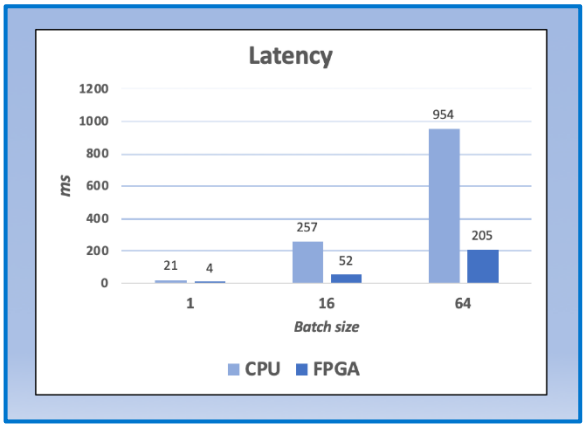

Figure 5: Latency. Compared to the CPU, the FPGA shows lower latency across different batch sizes.

CPU-onload

In this scenario, the CPU runs inference at a batch size that yields maximum throughput possible (from Fig 4, this batch size is 64). Here, we are interested in understanding the performance of an *inference-dedicated* CPU (in single-socket and dual-socket configurations) in comparison with the FPGA - the offload accelerator - with respect to the system power.

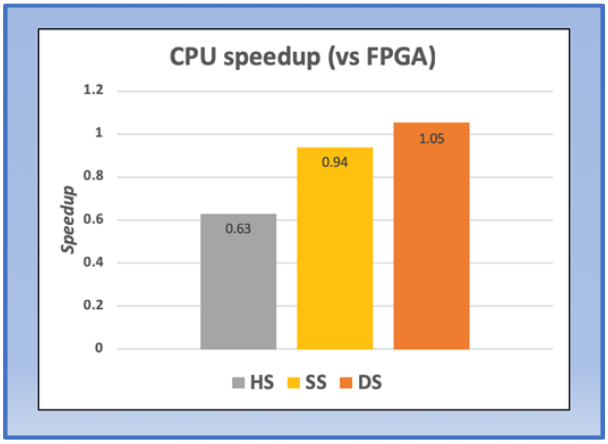

Figure 6: CPU-onload on inference-dedicated CPU(s) with half-socket, full/single-socket and dual-socket configurations.

To begin, we consider “half-socket” CPU, full / single-socket CPU, and dual-socket CPU configurations –denoted as *HS, SS,* and *DS* respectively – with the number of active threads equal to, respectively, the number of physical CPU cores (16 threads), number of virtual CPU cores (32 threads, in the case of hyperthreading), and 64 threads, and the total number of virtual threads on our dual-socket Skylake-based server. As indicated in Fig 6, our single PAC-based FPGA achieves the same performance as the dedicated dual-socket server. Next, we examine their performance efficiency with respect to power.

Efficiency

Performance efficiency is the throughput achieved, normalized by power consumed. As we saw in Fig 4, doubling the number of threads from 32 (i.e., SS) to 64 (i.e., DS) does not have as much impact as going from HS to SS and results in throughput increase by only 11%. The price of this rather sublinear performance increase is reduced performance efficiency: by a factor of 0.4 (Fig 7). The measured system power of the CPU execution reached a maximum 416 W. With a measured board power of 50W on the PAC, the FPGA achieves a much higher performance efficiency - a factor of 6x - compared to the CPU, with only 20% increase to the server' power budget.

Figure 7: Performance efficiency

Scale-up

We scaled up the number of PACs from 1 to 4. As shown in Fig 8, the FPGAs achieve a linear speedup of 1,251 FPS with a consistent Perf/Watt. Although increasing the thread count beyond 64 would cause little bump in the FPS (as we saw in Fig 4) we were curious to see the CPU performance at the same scale factor as the FPGA. In this scenario, we migrated our experiment from the dual-socket PowerEdge R740 server to a quad-socket (QS) PowerEdge R840 server with 4 CPUs. The QS configuration achieved a mere 330 FPS at 60% reduction in Perf/Watt efficiency. The aggregate system power for the QS and 4xFPGA (with SS) configurations were 615W and 405W respectively, indicating the FPGAs yield best performance both in terms of Perf/Watt and overall system-power budget.

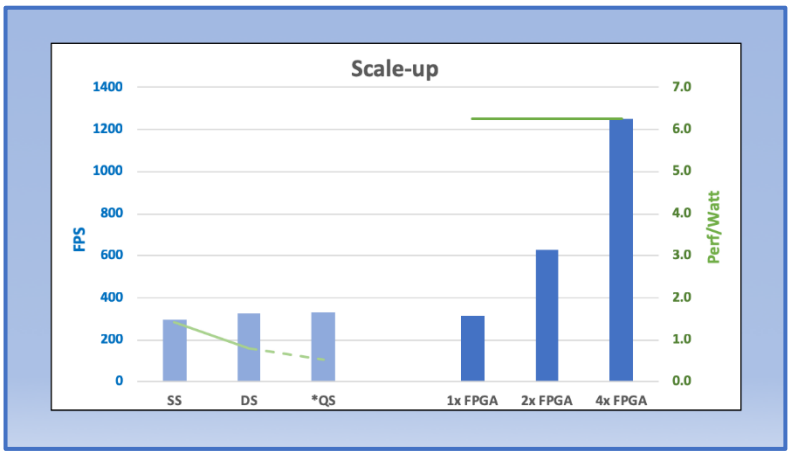

Figure 8: The FPGAs scale linearly with a consistent Perf/Watt. *In QS, batch size was increased to 96 to commensurate with the increased core count.

Conclusions

We presented the Intel Programmable Accelerator Card (PAC) with Arria 10 FPGA for deep-learning inference.

We showed that, using the Intel PAC on an x86-based Dell EMC PowerEdge server, we achieved improved performance of ResNet-50 compared to previously released ResNet-50 model. Specifically, we examined the throughput, latency, as well as CPU-onload performance with half-socket, full-socket, dual-socket, and quad-socket configurations, and scaled up the number of PACs to 4. While the full-fledged, quad-socket CPU configuration achieved 330 FPS at 0.79 FPS/Watt (a 60% efficiency reduction compared to the dual-socket counterpart), the FPGAs achieved 1,251 FPS at 6 FPS/Watt with 20% power increase per PAC to the server power budget. These performance numbers are expected to continue to improve with ongoing optimizations to the hardware and software system stacks.

Acknowledgements

Center for Space High-Performance and Resilient Computing (SHREC), University of Florida.

References

[1] A. Singh and J. C. Príncipe, "A loss function for classification based on a robust similarity metric," The 2010 International Joint Conference on Neural Networks (IJCNN), Barcelona, 2010

[2] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016.

PowerEdge DfD Repository
For more technical learning

Contact Us
For feedback and requests

Follow Us
For PowerEdge news