Optimized Scheduling Algorithms for Heterogeneous Microarchitectures in Vision Systems

Bhuvanesh S K

Department of Electrical and Computer Engineering, University of Wisconsin–Madison, WI, USA

Paper ID: VP802

Scan for paper & contacts

visionschedule.netlify.app

Abstract

Real-time vision systems face growing computational demands that homogeneous CPU/GPU architectures struggle to meet under Size, Weight, and Power (SWaP) constraints. This work presents a **microarchitecture-aware scheduling framework** that partitions PyTorch vision pipelines across CPU, GPU, and FPGA based on profiled compute throughput, memory bandwidth, and interconnect latency. GPUDirect RDMA and Xilinx PCIe peer-to-peer eliminate host-mediated data movement. Across three representative pipelines, the heterogeneous platform achieves up to **1.11× faster kernel execution** and **1.08× better energy efficiency** than the best standalone accelerator, with classification accuracy preserved within 0.5%.

Methods

Hardware. AMD Ryzen 9 5900X (12C, 4.8 GHz) + NVIDIA RTX A2000 (3328 CUDA, 12 GB GDDR6) + Xilinx Alveo U50 (872K LUT, 8 GB HBM). PCIe Gen3 interconnect with GPUDirect RDMA and Xilinx P2P for device-to-device DMA.

Software. PyTorch 2.4 (GPU), Vitis Vision Library via Vitis 2022.2 (FPGA), OpenCV (CPU). FP16 throughout. Energy via RAPL / nvidia-smi / xbutil. Dataset: LIU4K-v2 (2,000 images @ 3840×2160).

Methodology. Each operation is routed to CPU, GPU, or FPGA using a profiling-based cost model (or an explicit `#GPU`/`#FPGA` pragma). DMA transfers are inserted automatically at every device boundary; FIFO ordering preserves the original pipeline semantics.

Scheduler Architecture

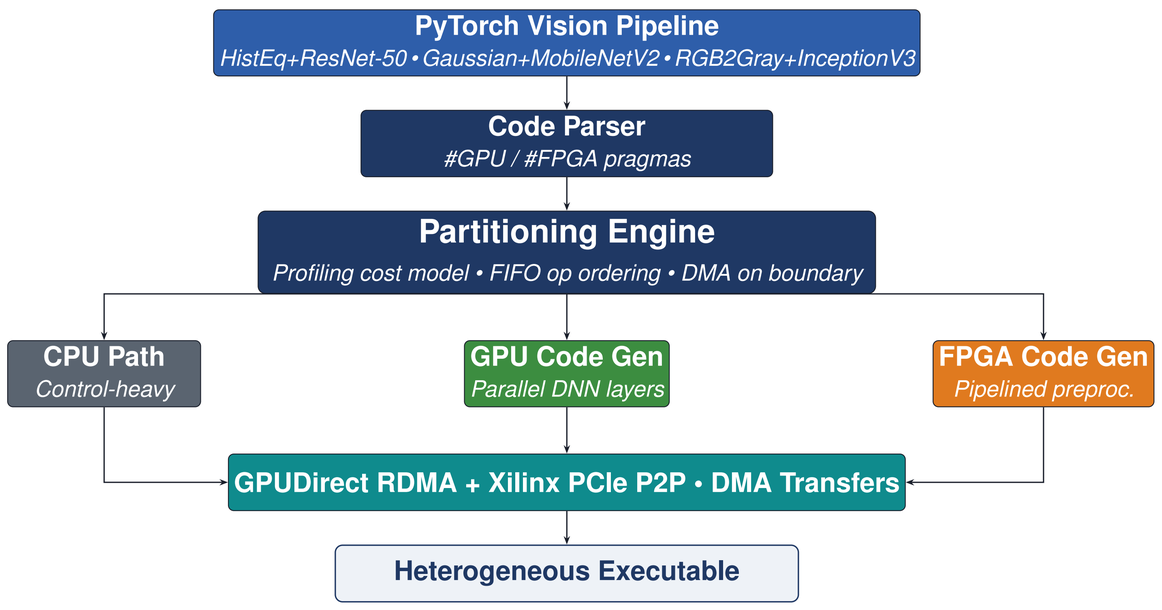

Fig. 1. Scheduler architecture. Parallel ops → GPU, pipelined ops → FPGA, control ops → CPU, with peer-to-peer DMA between devices.

Results Table & Key Findings

Pipeline	Kernel Time (ms)	Kernel Energy (J)	FPS	Acc. (%)
HistEq + ResNet-50	7.11	0.27	140.65	87
Gaussian + MobileNetV2	5.22	0.17	191.57	82
RGB2Gray + InceptionV3	8.44	0.35	118.48	84

Table 1. Kernel performance on the heterogeneous platform (FP16, LIU4K-v2 at 3840×2160).

Key Findings.

- **FPGA wins preprocessing:** 1.81× faster than GPU, 1.79× faster than CPU thanks to custom datapaths and HBM bandwidth.
- **GPU wins DNN inference:** 4.21× over CPU, 1.41× over FPGA on convolutional layers.
- **Heterogeneous scheduler:** 1.12× faster than GPU and 1.42× faster than FPGA; 1.34× less energy than GPU and 1.27× less than FPGA.
- **Peer-to-peer DMA** keeps inter-device communication to 8–12% of runtime; accuracy preserved within 0.5%.

Kernel Runtime Comparison

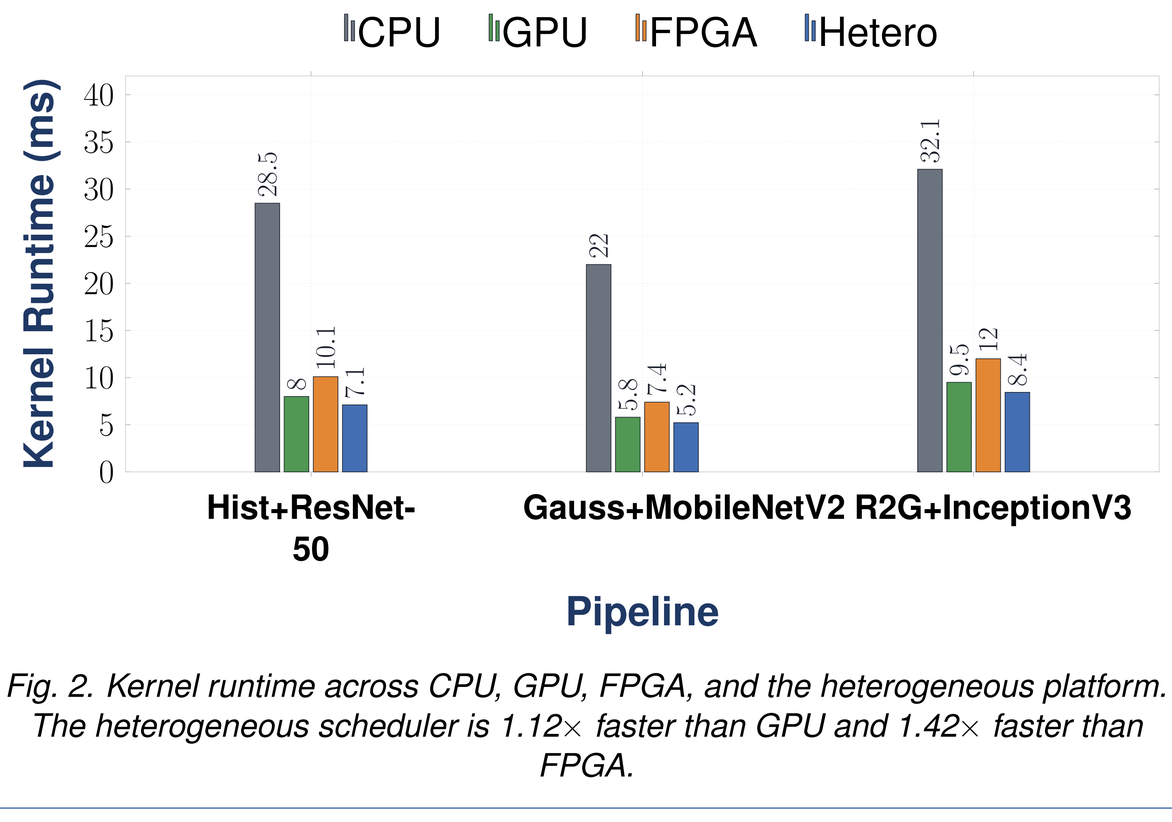

Fig. 2. Kernel runtime across CPU, GPU, FPGA, and the heterogeneous platform. The heterogeneous scheduler is 1.12× faster than GPU and 1.42× faster than FPGA.

Scheduling Algorithm

Require: Vision pipeline $P = \{op_1, \ldots, op_n\}$

1: **for** each $op_i \in P$ **do**
2: **if** op_i has pragma **then** $target \leftarrow$ pragma(op_i)
3: **else** $target \leftarrow \arg\min_{acc}$ profile$(op_i, acc).\{time \mid energy\}$
4: **end if**
5: Append op_i to P_{target}; if $target \neq prev$, insert DMA transfer
6: **end for**
7: **return** Partitioned code for CPU/GPU/FPGA + DMA config

Algorithm 1. Pragmas take precedence; otherwise a profiled cost model picks the minimum-time (or minimum-energy) accelerator per op, with DMA inserted at every device boundary.

Conclusions & Future Work

Microarchitecture-aware scheduling unlocks gains homogeneous execution cannot: routing preprocessing to FPGA, DNN inference to GPU, and control to CPU with peer-to-peer DMA delivers **1.11× faster kernel execution** and **1.08× better energy efficiency** than the best standalone accelerator, with accuracy preserved within 0.5%.

Future work.

- Dynamic load balancing via ML-guided partitioning.
- Layer-by-layer DNN partitioning (finer granularity).
- A DSL for vision-pipeline specification.
- NPU and neuromorphic-processor integration.