DPU-Bench: A New Micro-Benchmark Suite to Measure the Offload Efficiency of SmartNICs

Ben Michalowicz

Presented at MUG 2023

michalowicz.2@osu.edu

(Paper: DPU-Bench: A Micro-Benchmark Suite to Measure Offload Efficiency Of SmartNICs
B. Michalowicz, K. Suresh, H. Subramoni, DK Panda, and S. Poole, Practice and Experience in Advanced Research Computing 23, Jul 2023)
Breakdown

• Introduction, Problem Statement, Motivation
• Design Choices
• Implementation
• Experimental Results
• Conclusion and Future Work
Problem Statement: We Need a New, DPU-Aware Microbenchmark Suite

• Already have plenty of Suites: OMB, IMB, mpiBench, OpenHPCA, etc.
  - NONE are DPU-Aware!

• Previous works: Utilize DPUs to offload in the context of an MPI-library, for specific apps, or in the context of Deep Learning, but no micro/benchmarks!

• SmartNICs are becoming popular: NVIDIA BlueField, AMD Pensando, Marvell etc.

• What algorithm is the “best” to offload to a DPU? Can we design a microbenchmark suite to measure the offload potential achieved from placing communication to the DPU?
Design Choice: Running at the IB-Verbs Level

- Running at the MPI level is bottlenecked by progress on the host
  - Naively offloading MPI-level ranks in, e.g., MPI_Ialltoall could create a bottleneck
  - Host issues MPI_Isends to a DPU process → needs to check message progress from host AND DPU sides

- Running over IB-Verbs:
  - Closer to the hardware
  - DPU issues all RDMA operations – host/CPU is not involved
    * No need for asynchronous progress on the host
General Breakdown of DPU-Bench benchmarks (1/2)

- At initialization:
  - Exchange of metadata for RDMA Read/Write ops
    - lkeys, rkeys, buffer addresses, QP numbers, etc.
- Runtime:
  - Step 1: Do Pure-Host execution to obtain reference time for dummy compute.
  - Step 2: Offload communication to DPUs while the host side does compute
  - Step 3: Measure Offload Efficiency

Listing 2. General Approach to Each Benchmark

```c
/* Setup - assume options are passed in through CLI */
MPI_Init(...);
/* Record-keeping struct */
global_struct g;
/* Every process makes its own memory region which makes the needed rkeys */
setup_ib_counters(&g, msg_size, num_workers, num_host_procs);
/* Exchange of rkeys between processes */
MPI_Allgather(...);
/* Exchange of RDMA Buffer addresses */
MPI_Allgather(...);
create_sends_and_recvs pure_host();
run pure host(); //gives reference time
create_sends_and_recv worker_procs();
if (proc_is_on_dpu()) {
    /* IB-level RDMA-operations */
    run benchmark();
} else {
    perform compute on host(ref_time);
}
MPI_Barrier(MPI_COMM_WORLD);
obtain max_of lat and comp();
compute overlap();
cleanup();
cleanup();
MPI_Finalize();
```
General Breakdown of DPU-Bench benchmarks (2/2)

- Assumptions made about MPI runtime
  - Use of a block-style hostfile – helps with organization of config files
  - Higher-numbered ranks are placed on the DPU
  - Multiple Program, Multiple Data (MPMD) mode in an MPI library – required for dealing with CPU/DPU configurations
Collective patterns in this benchmark suite

• Non-personalized one-to-all: Direct/Linear Broadcast
• Personalized all-to-one: Direct/Linear Gather
• Non-Personalized all-to-all: Direct/Linear Allgather with a single "root" worker
  - Later slides: Improve upon this to utilize multiple workers to demonstrate more efficient staging
• Will only show a subset here
Experiments Performed

• 8 nodes, 8 PPN on host
• Ranging from 1 worker (total) to 8 WPN on 8 DPUs (64 total workers)
  – Powers of 2: 1, 2, 4, 8...
• Study message sizes from 256K to 4 MB
• Offload Efficiency:
  \[(\text{reference}\_\text{time}/\max(\text{pure}\_\text{comm}, \text{compute})) \times 100\]
• Show Offload efficiency results for broadcast, gather, and allgather with both cyclic and block work
Offload Efficiency Results: Broadcast – Cyclic Work Assignment

Bcast Offload Efficiency (8 Nodes, 8 PPN) -- Cyclic Assignment

Bcast Offload Efficiency (8 Nodes, 8 PPN) -- Cyclic Assignment

- General takeaway: 1 WPN is a “sweet spot” for maximum efficiency
- 8 WPN: Incurs overhead from the BF-2’s single memory controller and limited cache sizes
Offload Efficiency Results: Gather – Cyclic Work Assignment

Gather Offload Efficiency (8 Nodes, 8 PPN) -- Cyclic Assignment

• 1 Worker: Adds overhead of an “intermediate” process to write to the root
• Similar trends to the “Broadcast” results
Offload Efficiency Results: Allgather (Single-Root Worker) – Cyclic Work Assignment

- Gather and Broadcast placed back-to-back incurs massive overhead
- Slight degradations with increase in message size
Offload Efficiency Results: Allgather (Efficient Use of Multiple Workers – Cyclic Work Assignment)

- 1 worker (total) to 1 WPN: 1.3-2X improvement in efficiency with the addition of workers
- 2 WPN – 6 WPN: Predictable trend of decreasing offload efficiency
Conclusion/Future Work

• Introduction of a new, DPU-Aware Microbenchmark Suite
• Explored three communication patterns/algorithms
• Initial Release planned for the near future
• Generalize the benchmarks to other programming models
• Generalize the work to other SmartNICs
Thank You!

michalowicz.2@osu.edu

Network-Based Computing Laboratory

Follow us on

https://twitter.com/mvapich

Network-Based Computing Laboratory

http://nowlab.cse.ohio-state.edu/

The High-Performance MPI/PGAS Project
http://mvapich.cse.ohio-state.edu/

The High-Performance Big Data Project
http://hibd.cse.ohio-state.edu/

The High-Performance Deep Learning Project
http://hidl.cse.ohio-state.edu/