Co-Design with DPUs

BUILDING SYSTEM SOFTWARE WITH HARDWARE DESIGN CONSIDERATIONS – FIRST STEPS

PRESENTER: DR. RYAN E. GRANT

STUDENT CREDIT: TINOTENDA MATSIKA

CAESAR
COMPUTING AT EXTREME-SCALE ADVANCED RESEARCH LAB

Queen's UNIVERSITY

DPUs: What are they good for?

- DPUs: relatively new, still finding their “killer” app
- Hardware designs pre-date software
  - Opposite of usual co-design
  - Designs more generalized – hard to specialize on applications that are not settled
- Apps are not ready, so what do we do?

Don’t use DPUs for applications! (yet)

Applications

- Already heavy use on CPUs and GPUs
  - SmartNIC hardware is wimpy in comparison
- Finding things for the DPU to do for the application is challenging
  - Requires deep application expertise combined with hardware architecture expertise to leverage the DPU resources
- Co-design: logical conclusion -> work on broadly applicable de-coupled solutions for software for foreseeable future

System Software to the Rescue

- DPU cycles can be useful for helping to run system software
- Resource management layers (see RaDD runtimes)
  - Help with dynamic resource allocation/management
  - Create new layer of management running solely on DPU
  - Manage reliability solutions asynchronously (checkpoint/restart)
  - Build expected shared resource schedules to avoid conflicts
    - Same node and multinode

Grant, Ryan E., Whit Schonbein, and Scott Levy. "RaDD runtimes: Radical and different distributed runtimes with smartnics." In *2020 IEEE/ACM Fourth Annual Workshop on Emerging Parallel and Distributed Runtime Systems and Middleware (IPDRM)*, pp. 17-24. IEEE, 2020.

System Software - Pitfalls

Just like applications – need to make sure that DPU helps and doesn't hurt

System software priority < application priority

But! The network is a shared resource

So how do we reliably get out of the way of the application?

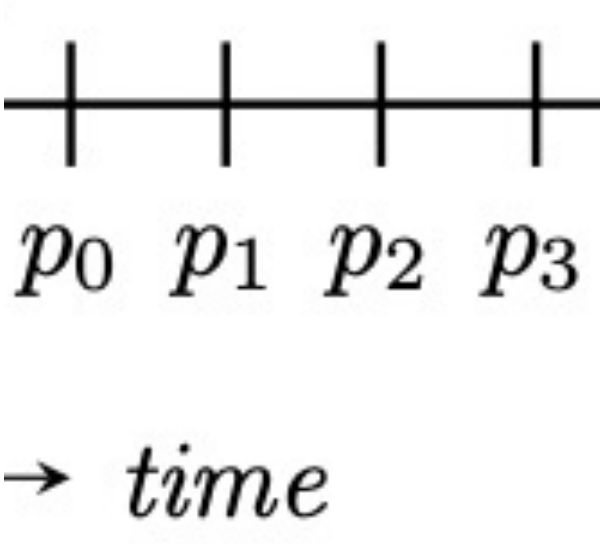
Luckily SmartNIC/DPU resources are well located to have the knowledge/data to get out of the application's way

Getting out of the way is hard

- Easier said than done
- AI/ML doesn't save the day
  - Remember we have limited compute resources
- What do we need to do to "cause no harm"?
  - Predict when the network (shared resource) is needed by the application and avoid using it with system software

Queen's University

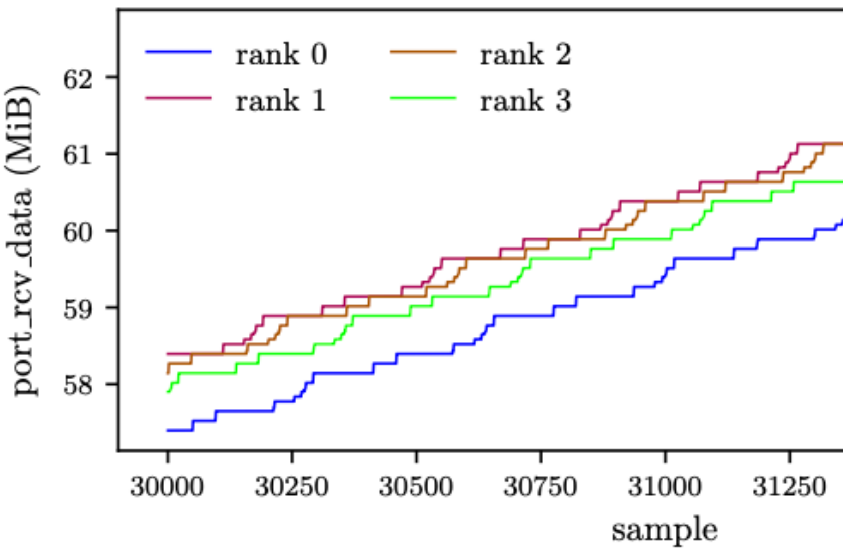
Predicting Traffic



- Predictions need to be designed for DPU architectures
- Prediction time slot granularity matters!
- Time of predictions from P1->P2 gives us the “opportunity” time to avoid conflicts. Smaller time windows for predictions give more opportunities!

Queen's University

Predicting Traffic



- Want to fit into the right timeslots to avoid interfering
- First let's try the easy approach
  - Simple ML Random forest, trained on counter data collected during application runs
- Use a production App – LAMMPS with rhodo problem
  - Bluefield 1 ***inference*** time: 17.05ms on average
  - Training takes a lot longer, but we can do that elsewhere

Timeslot period – why it matters

- At ~17ms per prediction, we can run at a maximum 58 Hz
  - Good but not great, not many opportunities to send small messages
  - ~215 MB per prediction slot of bandwidth on a 100Gbps link

- Another problem – inference with classic RFC on full data set is expensive computation for Bluefield
- But the results were pretty good: 0.97 accuracy and 0.78 recall

Getting to a better result

- Want a faster method but better accuracy/recall too!
- Approach:
  - Combine sophisticated linear regression (LR) with a second layer RFC that tells us if the LR should be trusted
- Why?
  - Linear regression does pretty well but not as well as RFC
  - Fundamental challenge – can't predict when slope 0 -> another value well

Faster, Better Predictions

- Tried a bunch of ML methods and a plethora of apps and workloads
- Condensed version here with LAMMPS-rhodo
  - Got accuracy up to 0.97 and recall up to 0.96
  - So accuracy == RFC and recall > RFC
- But does it run faster?
  - Yes!
  - Inference time now 1.1ms on Bluefield
  - Prediction frequency > 900Hz

Lightweight Prediction

So 15.5X faster inference with a better result!

Need super speed? LR can be tuned to as little as 0.9ns

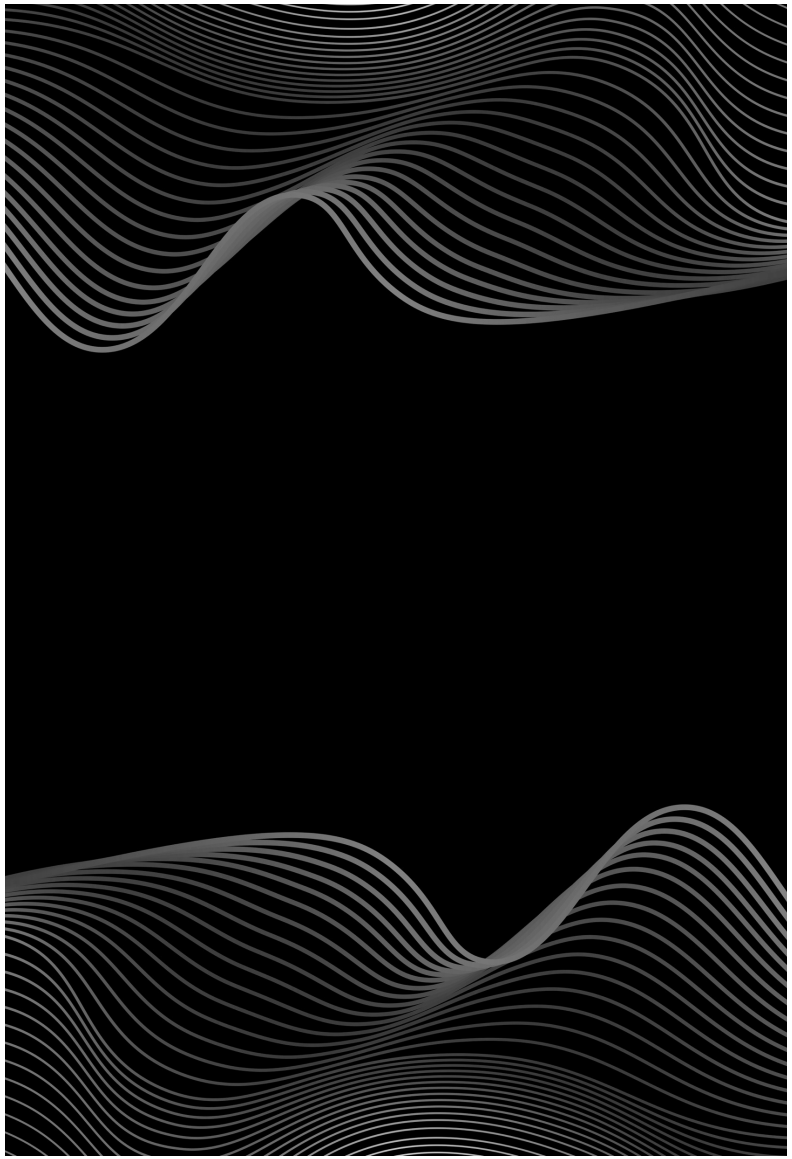
That much speed is not that useful, but can use the cycles for other tasks

But does it translate to other architectures?

Yes, V100 GPU results are 18.7X faster

So what? Why is this useful?

Queen's University



Benefits

- A foundation for building other useful systems software on DPUs
  - We can get out of the applications' way, so now we can fill up the DPUs with other useful work
- Do you need all 900 predictions per second?
  - Depends on the type/size of the messages you're trying to interleave with application traffic.
- Don't need them all?
  - Free up resources for other useful software on the DPU

Queen's University

Other exploration

- Tried a ton of different ML techniques and applications
  - Hyperparameter tuning, upsampling, downsampling, Neural Networks
  - Some have benefits but as expected random forest did a good job generally
    - Not sure that adding in this extra complexity to find and shift between methods is worth it
- Applications: HPCG, Laghos, LAMMPS, Lulesh, MILC, MiniAMR, MiniFE, MiniMD
  - Some apps are easier to predict that others, but none are particularly hard

Co-design Lessons Learned

- Co-design in reverse?
  - Designing software around hardware, much like just tuning software in the old days
- DPUs are resource limited
  - Motivates taking a lightweight design approach from the beginning of a project
- Danger in creating dependencies between work on the host and the DPU
  - Try not to get data/work stuck on DPU making everything wait

We now have a foundation for building the interesting system software

Thought this was a minor engineering task

- Ballooned into a much harder problem to solve

Let’s build proactive systems now

Manage and communicate resource usage between nearby nodes	CXL and disaggregated systems starting to make resource sharing viable

Now the fun begins…

Queen's UNIVERSITY

SmartNICs in the Future

- Vision towards intelligent self-learning resource management
- Application Assistance from DPU
  - Applications can be unaware of the help
  - Immediate return on investment for DPUs in the data center
  - No need for applications to change to benefit
- Results not specific to Bluefield
  - Can use other SmartNICs with same technique
  - Looking into applying results to INCA Portals SmartNIC designs

Thank You!

Questions?

Natural Sciences and Engineering Research Council of Canada
Conseil de recherches en sciences naturelles et en génie du Canada
Canada

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC).