Session 9: Keeping up with ever-increasing Performance Demands

Transcribed by: M. Taimoor Tariq, Ammar Tahir (UIUC)

Efficient Flow Scheduling in Distributed Deep Learning Training with Echelon Formation
Rui Pan *(Princeton University); Yiming Lei, Jialong Li (Max Planck Institute for Informatics); Zhiqiang Xie (Stanford University); Binhang Yuan (ETH Zurich); Yiting Xia (Max Planck Institute for Informatics)*

Paper makes the observation that semantically related ML flows do not take same amount of time and instead may have strict but different patterns. This observation allows for smarter flow scheduling decisions.

Questions:

1. **Would first in first out policy work?**
   FIFO may not work in all cases, our scheme is able to self correct if it errs.
2. **Does your model take of different implementations of parallelism**
   We only take care of g pipeline implementations.

Congestion Control in Machine Learning Clusters
Sudarsanan Rajasekaran*, Manya Ghobadi (MIT); Gautam Kumar (Google); Aditya Akella (UT Austin)*

Fair sharing the link may not be the best thing for congestion control in ML clusters. In fact, small amounts of unfairness can result in ML clusters converging to a state where compute and communication of tasks is phased out in such a way that congestion is minimized.

Questions:

1. **How the unfairness thing causes that, so do you have a notion of the two phases converging**
   They will eventually converge, but how long they take to converge, it depends on the unfairness between them.
2. **What is the definition of congestion that you have?**
   DCQCM is used in our experiments. Congestion is defined when more than one job uses up the same link. There could be different CC we can use.
3. **Do we need to use unfairness? Can we not use delays to induce this pattern?**
   Yes flow scheduling could be another way to do this
4. **In the non-compatible scenario, could you give the entire bandwidth to one flow? Can that help?**
   There can be partial compatibility and even in those cases there can be benefits. We are working on this.

Getting back what was lost in the era of high-speed software packet processing
Marcelo Abranches *(University of Colorado Boulder); Oliver Michel (Princeton University); Eric Keller (University of Colorado Boulder)*

To achieve faster speeds, today's kernel is often bypassed. However, the Linux kernel allowed for fast paced innovation because of the flexibility it offers. Can we achieve the best of both worlds?

Questions:

1. **How do you generate ebpf code?**
   We are not directly generation byte code, we use a python tool for that
2. **Linux primarily processes on single thread while eBPF is multi thread, did you face any replication issues?**
   Distributing our threads on multiple cpus. Did not face any such problems. Basically, with RSS you can get a key from your packet header and always send the packet from the same flow to be processed in the same cpu
3. **What kind of functionality did you implement in eBPF**
   We have in XDP some basic packet processing functions to help with correctness of processing, kernel function calls, state management, etc.

Understanding Host Interconnect Congestion
Saksham Agarwal*, Rachit Agarwal (Cornell University); Behnam Montazeri, Masoud Moshref, Khaled Elmeleegy, Luigi Rizzo, Marc de Kruijf, Gautam Kumar (Google); Sylvia Ratnasamy (Google and University of California Berkeley); David Culler, Amin Vahdat (Google)*

Presenter shows evidence of congestion in the host interconnect (IOMMU). This has interesting implications on how we think about and address congestion.

Questions:

1. **Do you have any vision about addressing cache behaviour (depending on traffic patterns)?**
   One way to reduce TLB miss rate is to reduce the number of active IOMMU pages (e.g. by reducing the active number of flows).
2. **Server memory bandwidth is usually 50-100 gbps, much higher than nic. Why is there memory contention then?**
   Per core limitation + Its not a realistic assumption that you will only be handling network traffic on your core.

End of Session Panel Discussion

1. **You assumed packets get delivered to memory via CPU? E.g. Can Swift address the congestion due to the host interconnect?**
   Using just delay (as in Swift) is not the right signal for host congestion, we need better signals.
2. **Are you assuming the network to be oversubscribed?**
   Due to ECMP collisions it is possible different flows end up sharing the same path even when the network is not oversubscribed.
3. **Do you have an insight on how many jobs typically share a link? Does it become harder with an increasing number of jobs?**
   Even when multiple jobs are shared, we can still find total compatibility. We are now also looking into partial compatibility, which can still result in better performance.
4. **Once you use ebpf, but you lose on some functionalities for example iptables?**
   We don't always need all the functionality so it is really a trade off.
5. **Suggestion: You can do congestion avoidance similar to your approach in video traffic as well.**
6. **Intel allows DDIO feature to send nic to send packet to L3 cache. Did you observe congestion on this DDIO machine? Maybe we should think about developing an isolation mechanism? This way we can make the network stack run faster.**
   We did have problems with DDIO on. Some recent work suggests that efficiency of DDIO gets worse as you increase the number of active flows and we saw the same thing. You might see less memory contention but in the real world it's no longer super effective. Secondly, if you do isolation for network, memory, bandwidth, cpu processing it will be helpful but then it becomes this balancing problem. The worst case memory bandwidth is not a trivial problem since there is so much variation so resource partitioning is hard due to unpredictability.
7. **There could be a scalability issue in Sudarsanan;s work since multiple works share the same link.**
   We have a lot of diversity in different ML models, and we can play with parameter models. Good direction to look at. How do different jobs interact when they share the same link.
8. **What happens if I stagger the start time at app level? Do we still get the separation of compute and communication time?**
   Yes the scheduler can handle that.