Networking Session

Wednesday 10:50 - 11:50

Simon Peter (UT Austin)
Networking = Dealing with Scale

- We leverage networks to “scale out” our systems
  - Use more hardware/software to stem a bigger workload

- Examples
  - Networks on Chip (10s - 1K nodes, RTT: nanoseconds)
  - Supercomputers (“small” scale, 1K - 10K nodes, RTT: 10 us)
  - Datacenters (“hyperscalers”, 100K - 1M nodes, RTT: 100 us)
  - Internet (global scale, billions of nodes, RTT: milliseconds)
Networking: Fundamental Properties

- Some fundamental properties of networked systems
  - Latency
  - Bandwidth
  - Failures
  - Heterogeneity
Networking: Perennial Problems

- Minimize/stabilize latency
  - Fast remote procedure calls [Birrell’84, lots of recent RDMA work]
  - “Tail at scale” [Dean’13]
- Share available bandwidth
  - Congestion control (lots and lots of work - no one size fits all)
- Quality of service when things fail or are heterogeneous [MPEG-DASH’11]
- Programming with heterogeneity [XDR’87]
  - Recently: Accelerators [U-Net’95, PacketShader’10, Click-NP’16]
Networking: Perennial Problems

- Minimize/stabilize latency
  - Fast remote procedure calls [Birrell’84, lots of recent RDMA work]
  - “Tail at scale” [Dean’13]
- Share available bandwidth
  - Congestion control (lots and lots of work - no one size fits all)
- Quality of service when things fail or are heterogeneous [MPEG-DASH’11]
- Programming with heterogeneity [XDR’87]
  - Recently: Accelerators [U-Net’95, PacketShader’10, Click-NP’16]
Each RPC has its RTT - long latency for many messages

Can we reduce round trips by shipping code to KV stores?

KV stores are shared among 1000s of users. How to provide low-cost isolation?
- Use language-level protection (Rust), cooperative scheduling
Neural Adaptive Content-aware Internet Video Delivery
Hyunho Yeo, Youngmok Jung, Jaehong Kim, Jinwoo Shin, Dongsu Han (KAIST)

- Internet video streaming under heterogeneous conditions
  - Quality of experience suffers when network conditions deteriorate

High bandwidth

![High bandwidth video](image1)

Low bandwidth

![Low bandwidth video](image2)
Neural Adaptive Content-aware Internet Video Delivery
Hyunho Yeo, Youngmok Jung, Jaehong Kim, Jinwoo Shin, Dongsu Han (KAIST)

- Use machine learning to enhance video quality at the client
- Use scalable DNNs to provide prediction on heterogeneous hardware
Floem: A Programming System for NIC-Accelerated Network Applications
Phitchaya Mangpo Phothilimthana (UC Berkeley), Ming Liu, Antoine Kaufmann (UW), Simon Peter (UT Austin), Rastislav Bodik, Thomas Anderson (UW)

● Programming with accelerators is difficult
  ○ Complex, heterogeneous hardware/interconnect architecture
  ○ No good performance models
  ○ Require many design/implement/test iterations to find well-performing offload
  ○ Iterations involve non-trivial changes to programs

● What abstractions will help the programmer?
  ○ Observe expert developers do their work and generalize from there

● Abstract communication details
  ○ Programmers can easily move program components
  ○ Compiler infers what data to send, how to keep it consistent

● Integrate well with legacy code
Conclusion

● **Come to the networking session at 10:50 - 11:50!**

● **Papers:**
  ○ KV store acceleration via scalable function shipping
  ○ Video quality improvements via ML
  ○ Programming system to simplify NIC offload for networked applications

● **How does each paper deal with our four fundamental properties?**
  ○ Latency
  ○ Bandwidth
  ○ Failures
  ○ Heterogeneity
Backup
Problems and key solutions:

- **What abstractions will help the programmer?**
  - Observe expert developers do their work and generalize from there

- **Program needs to be componentized** to explore offload design space
  - Data-flow helps express communication among components

- **Use bandwidth** between NIC and CPU efficiently
  - Compiler can infer what data needs to be sent

- **Different offloads require different communication strategies**
  - Can be expressed via virtual-to-physical queue mappings
Splinter: Bare-Metal Extensions for Multi-Tenant Low-Latency Storage
Chinmay Kulkarni, Sara Moore, Mazhar Naqvi, Tian Zhang, Robert Ricci, Ryan Stutsman (University of Utah)

Problems and key solutions:

- KV stores are shared among 1000s of users. **How to provide low-cost protection?**
  - Use language-level protection (Rust), rather than hardware
- **How to provide performance isolation?**
  - Use run-to-completion scheduling, with preemption as a fallback
- To provide more benefit to shipping code, can we **eliminate node-local overheads**?
  - Eliminate copies wherever possible
Key ideas:

- Use language-level protection (Rust) for zero-cost protection
  - Type safety, memory safety
- Minimize copies in KV store API to minimize local overheads
  - Operate on buffers directly, use reference counts
- Use run-to-completion scheduling, with preemption as a fallback for performance isolation
  - Cooperative scheduling via yield statement
- Adaptive multi-core request routing to maintain locality when sharing data
  - Use Intel Flow director
Neural Adaptive Content-aware Internet Video Delivery
Hyunho Yeo, Youngmok Jung, Jaehong Kim, Jinwoo Shin, Dongsu Han (KAIST)

- **Problems:**
  - Do clients have enough compute bandwidth?
  - CDN servers need to provide DNN model. One model does not fit all content.
  - DNN-based quality enhancement must work in real-time on all types of devices
  - Existing ABR needs to take DNNs into account when selecting bitrate

- **Assumption:** Clients have enough compute bandwidth

- **Key ideas:**
  - Train DNNs for each video separately, at server
  - Use **multiple scalable DNNs** to provide anytime prediction on heterogeneous hardware
    - Scalable DNNs work, even when partially downloaded
  - Devise a **DNN-aware ABR algorithm** for QoE optimization
Floem: A Programming System for NIC-Accelerated Network Applications
Phitchaya Mangpo Phothilimthana (UC Berkeley), Ming Liu, Antoine Kaufmann (UW), Simon Peter (UT Austin), Rastislav Bodik, Thomas Anderson (UW)

Main insights:

- Leverage **data-flow** based programming language
  - Easy to express communication and parallelism
- **Logical queue** abstraction
  - Can be mapped to physical queues without code change
- **Packet state** abstraction
  - Compiler infers minimum dataset to be transferred across components
- **Caching construct**
  - Compiler infers how to provide cache consistency
- Easy **integration with existing code**