TORSTEN HOEFLER, ROBERT ROSS, TIMOTHY ROSCOE

Distributing the Data Plane for Remote Storage Access
My outing

- I’m an HPC (systems) guy
- Programming Models
- Performance Models
- Network (Models)
Memory systems become more complex!

Data movement is the new challenge!
<100 ns latency
>300 GB/s bandwidth

<1 us latency
>12 GB/s bandwidth

~100 ns latency
>400 GB/s bandwidth
... but software is lacking behind.

- **Our use-cases:**
  - Advanced parallel programming (e.g., MPI-IO [1])
  - Data analytics chains (e.g., DataPath [2], Niad [3])

- **File systems are served through the CPU**
  - *Access to remote persistent storage in a closely-coupled computer cluster is one of the main obstacles to scaling performance*
  - Huge overheads: *Energy, Time, Cost*

---

[2]: Arumugam, S. et al.. The DataPath System: A Data-centric Analytic Processing Engine for Large Data Warehouses. In SIGMOD’10
A (30-year-old) networking idea revived

- Separating data and control plane
  - Get the CPU/software out of the way!
  - Software-defined IO (SDIO, cf. NASD [1], Aerie [2])
  - *We set up a network route right into the device!*

- Key point is (direct) access to storage
  - Allocation
  - Read/Write
  - Protection
  - Caching
  - Consistency, coherence, durability

- Our central research question:
  - *How can we design a fast software/hardware data plane for safe, secure, direct remote access to persistent storage devices?*

[2]: Volos, H., et al.. Aerie: Flex-ible File-system Interfaces to Storage-class Memory, EuroSys’14
(1) Device allocations

- Exokernel-like filesystem library
  - Data path 100% in user-level
    *Manages metadata, data, and coordination*
  - Data is stored in allocations

- An allocation …
  - Is an area of main memory or on a storage device
    *Placed explicitly*
  - Can be created, opened, or closed
    *Using a central or distributed control plane*
  - Has a contiguous address space
    *Block translation implemented in the device*
  - Is the smallest unit of access control and sharing
    *Named in a global namespace*
  - Access through capabilities (e.g., IB PDs)
(2) Read/Write and protection

- Allocations are accessed …
  - Locally via MMU-mappings (cf. Aerie)
  - Remotely via IOMMUs or other address translations

- Not all devices are part of the physical address space
  - Mainly legacy …. two options:
    1. Software fallback (monitor RDMA regions and keep consistency)
    2. Use IOMMU logging schemes (cf. Active Access [1])

[1]: M. Besta, TH: Active Access: A Mechanism for High-Performance Distributed Data-Centric Computations, ACM ICS’15
(3) Caching

- **Software caching**
  - Explicit caching, similar to RMA programming
    
    *Allocate local cache and access it*
  - Can be application-specific or caching library (standard techniques)

- **Hardware caching**
  - Set up local memory as cache for remote allocations
    
    *Use standard (e.g., LRU) replacement policies*
  - Could be implemented by an extended IOMMU
    
    *Would allocate incoming transactions in cache*
(4) Consistency and coherence

- Current RDMA does not support consistent atomic access
  - At least not large enough
- We propose a weaker consistency model
  - All read/write accesses are nonblocking!
  - Arrange accesses into epochs separated by fence operations
    
    Modified data is only valid at the end of an epoch
  
  - The type determines the isolation level
    
    Shared: only consistency after epoch ends
    
    Exclusive: consistency + atomicity
    
    Persistent: consistency + durability
    
    Optimistic: consistency + atomicity but can fail
  
  - Types can be combined
    
    e.g., persistent + exclusive
- Implemented similar to other RMA programming models [1]
  - May require remote flushes (in the worst case RPCs)

Other filesystem requirements

- **Crash recovery**
  - Use transactional (optimistic, exclusive, persistent) epochs for metadata
  - Must ensure that locks time out if processes disappear

- **Scalability**
  - Scoping limits context of coherency/epochs (e.g., a shared file)
  - Integration with programming model (e.g., MPI-3 RMA)

- **Compatibility**
  - Provide standard library of user-level file systems
  - POSIX consistency with single-operation exclusive, persistent epochs
  - Magic byte in allocation allows automated “mounting” like files
Merge the network transparently into the file system.