iPipe
Offloading Distributed Applications onto SmartNICs
What is a SmartNIC?
The problem

How do we utilize the **heterogeneous** compute capacity of a SmartNIC **efficiently**

- while exposing **simple programming** abstractions
- in the presence of **varying network traffic**
- for **complex** distributed applications?
The Characterization Study
<table>
<thead>
<tr>
<th>Observation</th>
<th>Design Implication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet size distribution impacts the availability of computing cycles.</td>
<td>Monitor request sizes to adaptively offload tasks to the SmartNIC core.</td>
</tr>
<tr>
<td>Wimpy processor on SmartNIC provides opportunity for cheap parallelism.</td>
<td>Offload processes with low IPC without affecting packet processing and execution latency.</td>
</tr>
<tr>
<td>SmartNICs perform poorly on tasks with working set exceeding the capacity of L2 cache.</td>
<td>Identify such tasks and migrate/schedule it on the host.</td>
</tr>
<tr>
<td>Accelerators are critical resources on a SmartNIC.</td>
<td>Tasks that benefit from domain specific acceleration must be batched and executed on the SmartNIC.</td>
</tr>
</tbody>
</table>
iPipe’s Design

- Actor Programming Model
- Actor Scheduler
- Distributed Memory Objects
Actor Programming Model

- Support compute heterogeneity and hardware parallelism.
- Actors have well-defined state and can be migrated between host and the NIC dynamically.
- Operations:
  - Initialize private state.
  - Trigger execution handler upon receiving a message.
  - Send messages to other actors.
Actor Scheduler

- **Runtime utilization, execution latency, thresholds and other statistics.**
- **Policy:** Queue buildup identified by increased mean latency of FCFS cores. Victim: Actor with highest load.
- **Policy:** High task dispersion identified by tail latency of FCFS cores. Victim: Actor with highest dispersion.
- **Autoscaling of FCFS and DRR CPU groups.**
Distributed Memory Objects

- Abstraction enabling migration of actor state between host and NIC.
- DMO has only one copy tracked by object tables on both sides.
- DMO moved along with actor upon migration.
- Object ID used instead of pointers to transparently migrate objects.
Applications developed using iPipe
Replicated KV Store
Distributed Transactions
Real Time Analytics

- Three Actors: Filter, Counter and Ranker
- Filter worker discards uninteresting rows.
- Counter uses a sliding window and periodically emits tuples to ranker.
- Ranker sorts incoming tuples and emits top-n data.
- Ranker has high load and is migrated to host when network load is high.
Evaluation
<table>
<thead>
<tr>
<th>Dimension</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host CPU savings when offloading to SmartNIC.</td>
<td>RKV - 3.1, DT - 2.6, RTA - 2.5 cores.</td>
</tr>
<tr>
<td>Execution latency savings when offloading to SmartNIC.</td>
<td>RKV - 5.4μs, DT - 28.0μs, RTA - 12.5μs.</td>
</tr>
<tr>
<td>iPipe scheduler performance for low and high dispersion workloads.</td>
<td>Low dispersion: Similar to FCFS, better than DRR, High dispersion: Better than both FCFS and DRR.</td>
</tr>
<tr>
<td>iPipe Framework overheads (DMO indirection and scheduler overheads).</td>
<td>iPipe consumes around 12% more CPU cycles for RKV leader.</td>
</tr>
<tr>
<td>Comparison with Floem, a similar SmartNIC programming framework.</td>
<td>Significant improvement.</td>
</tr>
</tbody>
</table>
Thank you!

Questions?