Shoal: A Network Architecture for Disaggregated Racks

Vishal Shrivastav (Cornell University)
Asaf Valadarsky (Hebrew University of Jerusalem)
Hitesh Ballani, Paolo Costa (Microsoft Research)
Ki Suh Lee (Waltz Networks)
Han Wang (Barefoot Networks)
Rachit Agarwal, Hakim Weatherspoon (Cornell University)
Traditional racks in datacenters
Disaggregated racks in datacenters

Prior works [OSDI’16] [HPCA’12] [Keeton’15]
- High compute density
- Fine-grained resource pooling and provisioning
- Seamless scaling and independent evolution of resources
Prior works [OSDI’16] [HPCA’12] [Keeton’15]
• High compute density
• Fine-grained resource pooling and provisioning
• Seamless scaling and independent evolution of resources

Disaggregated racks in datacenters

Inter-rack DC Network

Intra-rack Network

I/O controllers
- CPU
- Memory
- NIC

Acclerators (FPGA, GPU, TPU)

NVMe

Storage

SoCs
Challenges for disaggregated rack network

- Connect as many as an order of magnitude more nodes than traditional racks

- Network
  - ~15KW power budget [NSDI’16]

Intra-rack Network

- Be high performant
  - low latency / high throughput

- Be power efficient
  - to enable high compute density
Challenges for disaggregated rack network

- Connect as many as an order of magnitude more nodes than traditional racks

- Intra-rack Network

  - Be high performant
    - low latency / high throughput

  - Be power efficient
    - to enable high compute density

- ~15KW power budget [NSDI’16]
Challenges for disaggregated rack network

- Connect as many as an order of magnitude more nodes than traditional racks

- \(~15\text{KW power budget [NSDI'16]}\)

- Be high performant
  - low latency / high throughput

- Be power efficient
  - to enable high compute density
Potential disaggregated rack network designs

<table>
<thead>
<tr>
<th></th>
<th>Low Power consumption</th>
<th>High Performance (low latency / high throughput)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Packet-switched</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Networks</td>
<td><img src="image" alt="ToR chassis switch" /></td>
<td><img src="image" alt="Network of switches" /></td>
</tr>
<tr>
<td></td>
<td><strong>X</strong></td>
<td><strong>✓</strong></td>
</tr>
<tr>
<td><strong>Direct-connect</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Networks</td>
<td><img src="image" alt="Network of switches" /></td>
<td><img src="image" alt="Network of switches" /></td>
</tr>
<tr>
<td></td>
<td><strong>✓</strong></td>
<td><strong>X</strong></td>
</tr>
</tbody>
</table>
Shoal is a network stack and fabric for disaggregated racks that is both low power and high performance (low latency, high throughput)

Key feature:
Shoal network fabric comprises purely fast circuit switches that can reconfigure within nanoseconds
Shoal is a network stack and fabric for disaggregated racks that is both **low power** and **high performance** (low latency, high throughput)

**Key feature:**
Shoal network fabric comprises purely *fast circuit switches* that can reconfigure within nanoseconds.
Goal 1: Low power consumption

Circuit switches
- No buffering
- No packet processing
- No serialization/de-serialization

Consumes significantly less power than packet switches
Goal 2: High network performance

Key Challenge:
Need to explicitly set up circuits (reconfigure) before sending packets

- **Traditional circuit-switched networks**
  - Uses switches with high reconfiguration delay, up to milliseconds
  - Uses a central controller to decide the circuits (reconfiguration algorithm)
  - Not suitable for low latency traffic

- **Shoal**
  - Leverages circuit switches with nanosecond reconfiguration delay

Key Design Idea:
De-centralized, traffic agnostic reconfiguration algorithm
  - Inspired from LB monolithic packet switches [Comp Comm’02]
Shoal for a single circuit switch network

A permutation of connections

N-1 time slots (an epoch)

Static pre-defined schedule (a cyclic permutation)

Each node has N-1 queues (one per dst)

Arbitrary traffic pattern

Uniformly load-balanced traffic

100% throughput

in worst-case
Extending Shoal to a network of circuit switches
Extending Shoal to a network of circuit switches

Requires very tight network-wide synchronization

- DTP [Sigcomm’16] + WhiteRabbit can achieve sub-nanosecond synchronization precision

A non-blocking topology of circuit switches
### Congestion in Shoal

#### Time slot

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>G</td>
<td>H</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>G</td>
<td>H</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>G</td>
<td>H</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>D</td>
<td>E</td>
<td>F</td>
<td>G</td>
<td>H</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>E</td>
<td>F</td>
<td>G</td>
<td>H</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>F</td>
<td>G</td>
<td>H</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>G</td>
<td>H</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
</tr>
<tr>
<td>H</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>G</td>
</tr>
</tbody>
</table>

#### Diagram

- **Flow to H**
  - **A** -> **H**
  - **B** -> **H**
- **Flow to H**
  - **C** -> **H**
  - **D** -> **H**
  - **E** -> **H**
  - **F** -> **H**
  - **G** -> **H**
  - **H**

- **A -> H**

- **B**
- **C**
- **D**
- **E**
- **F**
- **G**
- **H**
Congestion control in Shoal

Each per-destination queue $Q_i$ corresponding to destination $i$ is bounded!

\[ \text{len}(Q_i) \leq 1 + \text{incast\_degree}(i) \text{ packets} \]
Key properties of Shoal

- No central controller for reconfiguration
  - Fully de-centralized, traffic agnostic reconfiguration logic
  - Allows circuit switches to reconfigure at nanosecond timescales

- Each per-destination queue in the network is bounded

- Each packet traverses the network *at most* twice
  - Worst-case 50% throughput compared to an ideal packet-switched network
  - Can be compensated by allocating 2X bandwidth per node
  - Cost (Shoal) $\leq$ Cost (packet-switched network with $\frac{1}{2}$ bandwidth of Shoal)
Implementation

- Stratix V FPGA
  - Bluespec System Verilog
- Implemented custom NIC and circuit switch on FPGA

Circuit switch implementation can reconfigure in < 6.4ns

Verified the queuing and throughput properties of Shoal on a 8-node testbed
Evaluation

- **Power consumption**

For a 512-node rack

- Packet-switched network comprises 24 64x50 Gbps packet switches
- Shoal comprises 48 64x50 Gbps circuit switches

<table>
<thead>
<tr>
<th>Network</th>
<th>Power Consumption (KW)</th>
<th>Rack Budget Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet-switched Network</td>
<td>8.72</td>
<td>(58%)</td>
</tr>
<tr>
<td>Shoal</td>
<td>2.55</td>
<td>(17%)</td>
</tr>
</tbody>
</table>

- Shoal consumes 3.5x less power than packet-switched network!
Evaluation

Network performance

- Packet-level simulator in C
- 512-node rack
- 5 disaggregated workload traces [OSDI’16]
- Shoal has 2X bandwidth (with comparable cost)

- Shoal performs comparable or better than several recent designs for packet-switched networks!
Conclusion

<table>
<thead>
<tr>
<th></th>
<th>Low Power consumption</th>
<th>High Performance (low latency / high throughput)</th>
</tr>
</thead>
</table>
| Packet-switched  |                       | ![Packet-switched](image)
| Networks         | ![ToR chasis switch](image) | ![Network of switches](image) | ![x](image) | ![green check](image) |
| Direct-connect    |                       | ![Direct-connect](image) | ![green check](image) | ![x](image) |
| Networks         |                       | ![Shoal (circuit-switched)](image) | ![green check](image) | ![green check](image) |
Thank you!

Shoal FPGA prototype and simulator code is available at:
https://github.com/vishal1303/Shoal