Switches for HIRE: Resource Scheduling for Data Center In-Network Computing

[ACM ASPLOS 2021]

Marcel Blöcher¹, Lin Wang¹,², Patrick Eugster¹,³, Max Schmidt¹

¹TU Darmstadt  ²VU Amsterdam  ³USI Lugano

@ NetSys 2021
September 15, 2021
Software-defined networking and programmable data plane

Software Defined Networking (SDN)

Controller

Specifying the match-action entries

Defining packet headers, match-action tables

Compiler

Switching ASIC

Program

Programmable Data Plane (e.g., with P4)

Beyond packet forwarding: in-network computing

In-network computing: performing application-specific computations “in the network” on the path between data sources and sinks, leveraging modern programmable switches
### In-network computing examples

**NetCache: Balancing Key-Value Stores with Fast In-Network Caching**

<table>
<thead>
<tr>
<th>Authors</th>
<th>Conference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xin Jin, Xiaosong Li, Haoyu Zhang, Robert Szewczyk, Jeongseon Lee, Nate Foster, Eugene Shih</td>
<td>SOSP'17</td>
</tr>
</tbody>
</table>

**NetCache** is a protocol for balancing key-value stores in-network. It uses a combination of in-network caching and data replication to improve performance and reduce latency. Key features include:
- **Efficient data replication** through replicated data blocks (DRBs), allowing clients to locally cache data blocks for faster access.
- **Flexible deployment** through a hybrid approach that can be deployed in a variety of network environments, from small enterprises to large-scale data centers.
- **Scalable performance** through adaptive replication strategies that adjust to changing data access patterns.

**NetChain (NSDI'18)**

<table>
<thead>
<tr>
<th>Authors</th>
<th>Conference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhenan Zhu, Jasjeet Singh, Daniel Hartley, Haoyu Zhang, Robert Szewczyk</td>
<td>NSDI'18</td>
</tr>
</tbody>
</table>

**NetChain** is a network-based distributed ledger protocol that enables secure and efficient data sharing in a distributed network. Key features include:
- **Decentralized data storage** through a blockchain-like structure that allows for secure and transparent data sharing.
- **Efficient data retrieval** through a distributed hash table (DHT) that allows for fast and reliable data lookup.
- **Scalability** through a dynamic network topology that adapts to changes in network conditions.

**DistCache (FAST'19)**

<table>
<thead>
<tr>
<th>Authors</th>
<th>Conference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhenan Zhu, Jasjeet Singh, Daniel Hartley, Haoyu Zhang, Robert Szewczyk</td>
<td>FAST'19</td>
</tr>
</tbody>
</table>

**DistCache** is a distributed caching system that leverages the benefits of blockchain technology to improve data availability and reduce latency. Key features include:
- **Efficient data retrieval** through a distributed hash table (DHT) that allows for fast and reliable data lookup.
- **Scalability** through a dynamic network topology that adapts to changes in network conditions.
- **Secure data sharing** through a blockchain-like structure that allows for secure and transparent data sharing.

**NetLock (SIGCOMM'20)**

<table>
<thead>
<tr>
<th>Authors</th>
<th>Conference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhenan Zhu, Jasjeet Singh, Daniel Hartley, Haoyu Zhang, Robert Szewczyk, Xinyu Liu, Haoyu Zhang</td>
<td>SIGCOMM'20</td>
</tr>
</tbody>
</table>

**NetLock** is a protocol for centralized lock management using programmable switches. Key features include:
- **Flexible lock management** through a combination of software and hardware, allowing for efficient and scalable lock management.
- **Scalability** through a distributed lock management system that can be deployed in a variety of network environments.
- **Efficiency** through a combination of software and hardware, allowing for efficient lock management.

**HoverRaft (EuroSys'20)**

<table>
<thead>
<tr>
<th>Authors</th>
<th>Conference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhenan Zhu, Jasjeet Singh, Daniel Hartley, Haoyu Zhang, Robert Szewczyk</td>
<td>EuroSys'20</td>
</tr>
</tbody>
</table>

**HoverRaft** is a network-based distributed consensus protocol for in-network computing. Key features include:
- **Scalability** through a distributed consensus protocol that can be deployed in a variety of network environments.
- **Efficiency** through a combination of software and hardware, allowing for efficient consensus.
- **Security** through a combination of software and hardware, allowing for secure consensus.

**ATP (NSDI'21)**

<table>
<thead>
<tr>
<th>Authors</th>
<th>Conference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhenan Zhu, Jasjeet Singh, Daniel Hartley, Haoyu Zhang, Robert Szewczyk</td>
<td>NSDI'21</td>
</tr>
</tbody>
</table>

**ATP** is a network-based distributed data management system for in-network computing. Key features include:
- **Scalability** through a distributed data management system that can be deployed in a variety of network environments.
- **Efficiency** through a combination of software and hardware, allowing for efficient data management.
- **Security** through a combination of software and hardware, allowing for secure data management.

**NetSki (NSDI'19)**

<table>
<thead>
<tr>
<th>Authors</th>
<th>Conference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhenan Zhu, Jasjeet Singh, Daniel Hartley, Haoyu Zhang, Robert Szewczyk</td>
<td>NSDI'19</td>
</tr>
</tbody>
</table>

**NetSki** is a network-based data management system for in-network computing. Key features include:
- **Scalability** through a distributed data management system that can be deployed in a variety of network environments.
- **Efficiency** through a combination of software and hardware, allowing for efficient data management.
- **Security** through a combination of software and hardware, allowing for secure data management.

---

**NetSki** is a protocol for data management in-network. It uses a combination of in-network caching and data replication to improve performance and reduce latency. Key features include:
- **Efficient data retrieval** through a distributed hash table (DHT) that allows for fast and reliable data lookup.
- **Scalability** through a dynamic network topology that adapts to changes in network conditions.
- **Security** through a combination of software and hardware, allowing for secure data management.

**DistCache** is a distributed caching system that leverages the benefits of blockchain technology to improve data availability and reduce latency. Key features include:
- **Efficient data retrieval** through a distributed hash table (DHT) that allows for fast and reliable data lookup.
- **Scalability** through a dynamic network topology that adapts to changes in network conditions.
- **Secure data sharing** through a blockchain-like structure that allows for secure and transparent data sharing.

**HoverRaft** is a network-based distributed consensus protocol for in-network computing. Key features include:
- **Scalability** through a distributed consensus protocol that can be deployed in a variety of network environments.
- **Efficiency** through a combination of software and hardware, allowing for efficient consensus.
- **Security** through a combination of software and hardware, allowing for secure consensus.

---

**NetLock** is a protocol for centralized lock management using programmable switches. Key features include:
- **Flexible lock management** through a combination of software and hardware, allowing for efficient and scalable lock management.
- **Scalability** through a distributed lock management system that can be deployed in a variety of network environments.
- **Efficiency** through a combination of software and hardware, allowing for efficient lock management.

---

**ATP** is a network-based distributed data management system for in-network computing. Key features include:
- **Scalability** through a distributed data management system that can be deployed in a variety of network environments.
- **Efficiency** through a combination of software and hardware, allowing for efficient data management.
- **Security** through a combination of software and hardware, allowing for secure data management.

---

**NetSki** is a protocol for data management in-network. It uses a combination of in-network caching and data replication to improve performance and reduce latency. Key features include:
- **Efficient data retrieval** through a distributed hash table (DHT) that allows for fast and reliable data lookup.
- **Scalability** through a dynamic network topology that adapts to changes in network conditions.
- **Security** through a combination of software and hardware, allowing for secure data management.
## In-network computing examples

### NetCache: Balancing Key-Value Stores with Fast In-Network Caching

<table>
<thead>
<tr>
<th>X. Jin</th>
<th>Xiaoming Li</th>
<th>Haoyou Zhang</th>
<th>Robert Sadler</th>
<th>Jinghong Liu</th>
</tr>
</thead>
</table>

### NetChain: Scale-Free Sub-RTT Coordination

| X. Jin | E. Kwon | H. Zhang | N. Foster | J. Hwang | L. S. Cao | D. Kim | J. Wang | D. Song |

### DistCache: Provably Load Balancing for Large-Scale Storage Systems with Distributed Caching


### NetLock: Fast, Centralized Lock Management Using Programmable Switches

| Z. Yang | X. Ji | L. Hu | Y. Zeng | B. R. Shulam |

### HoverRaft: Achieving Scalability and Fault-tolerance for micro-scale Datacenter Service

| M. Kherani | P. A. O. | E. S. H. | A. Parab | D. K. J. |

### ATP: In-network Aggregation for Multi-tenant Learning

| C. Lam | L. J. | Y. Yang | A. M. | Y. Chen | W. Wu | A. K. | M. Ballard |

### Best Paper Award

- NetCache: Balancing Key-Value Stores with Fast In-Network Caching
- NetLock: Fast, Centralized Lock Management Using Programmable Switches
- HoverRaft: Achieving Scalability and Fault-tolerance for micro-scale Datacenter Service

### Best Student Paper Award

- NetLock: Fast, Centralized Lock Management Using Programmable Switches

### Keywords

- In-network computing
- Key-value stores
- Programmable switches
- Caching
- Load balancing
- Scalability
- Fault-tolerance
- Datacenter services
The Achilles’ heel of in-network computing: sharing

Resource scheduling on switches: no existing solutions at this moment!

Resource scheduling on servers: Mesos, Sparrow, YARN, Omega, Hydra, Kubernetes…
New challenges in resource scheduling for in-network computing

Resource scheduling in traditional data centers

Resource scheduling with in-network computing

Heterogeneity

Non-linear sharing

Alternatives

Locality

16 VMs
8 VMs + INC A
8 VMs + INC B
HIRE: Holistic INC Resource scheduler

1. Job submission with HIRE resource model
   - Jobs
   - CompReq

2. Job request transformation
   - PolyReq

3. Flow-network construction
   - Flow Network Manager
   - Server Resource Status
   - Task Schedule
   - Network Controller

4. Scheduling decision generation with MCMF solver
   - MCMF Solver
   - MCMF
   - INC Resource Status
HIRE main contributions

A new resource model based on the concept of composite

A flow-based scheduling algorithm with novel ways to build flow-networks
Evaluation results

**Baselines:** state-of-the-art schedulers retrofitted with HIRE resource model

![Graphs showing evaluation results]

- Serving >92% of INC demands
- 3x less traffic detour
- Low tail latency (<1s in 90% time)

https://github.com/mblo/hire-cluster-simulator
From sharing to caring: unified programming for in-network computing

Can we use in-network computing without worrying about network plumbing which is tedious and error-prone?

We propose Compute-Centric Communication (C3) based on a concept called network kernels, implemented with a domain-specific language called NCL and associated runtime libraries.

Takeaways

Computer networking has evolved into a new era
- Abstractions for networking: SDN and PDP with P4
- Fully programmable, software-based networking

In-network computing: the network can compute!
- Sharing is important!
- In-network computing demands a resource management system
- Our solution: holistic INC resource scheduler (HIRE)

Unified programming for in-network computing
- Caring is also important!
- In-network computing demands a higher-level programming model
- Our solution: Compute-Centric Communication (C3) and NCL

Contact: lin.wang@vu.nl
Backup slides
HIRE resource model

**Composite store (CompStore)**
- Holding templates for in-network computing solutions and their implementations (including resource demands)
- Extensible by application developers

**Composite resource request (CompReq)**
- **High-level APIs** for specifying job requests
- Hiding all the implementation-related details
- More friendly to users

**Polymorphic resource request (PolyReq)**
- Resource demands of the submitted composite requests
- Automatically generated from CompReq
- More friendly to resource scheduler

---

```scala
def setupSendCompositeRequest() {
  val c4 = Composite('c4', CompStore.lookup('Server', properties='{cpu:16, mem:8.5, instances:12}'))
  val coordi = CompStore.lookup('Coordinator', filterImpl=None, properties='{tp:50MQPS, ft:2}')
  coordi.impl.foreach(impl => { /* custom modify req. */})
  val c5 = Composite('c5', coordi)
  val composites = c4 :: c5 :: Nil
  val connections = Connect(c4, c5, Connect.Bidi) :: Nil
  val prio = Priority(requestPriority)
  ComReq(prio, composites, connections)
}
```

An example CompReq
Resource request transformation

CompReq (specified by users as input)

PolyReq (automatically generated for scheduling)

Expanding **composites** with specified **implementations** integrating resource demands information from the CompStore
Scheduling problem modeling

New challenges

- **Alternative selections**: late binding
- **Locality**: topology-awareness
- **Non-linearity**: processing stages may be reused by multiple jobs

Integer Program formulation

\[
\text{maximize } \sum_J y_J \quad \text{s.t.} \quad \\
\sum_Z \bar{e}_{Z,M} \otimes \bar{q}_{Z,M} \leq \bar{r}_M, \forall M \\
\prod_{G \in J} \prod_{T \in G} \sum_M s_{G \alpha T,M} = y_J, \forall J
\]

- Maximize the #jobs to be scheduled
- Respect resource constraints considering non-linearity
- Ensure atomicity for alternative selection

Resources reused among tasks from task groups of the same type
Flow-based scheduling

High-level idea

- Build a flow-network encoding all the costs and constraints
- Employ an MCMF solver to obtain the scheduling decision

Main innovations in building the flow-network

- A shadow network to encode INC resources and topology-aware locality
- Flavor and materialized components to reflect alternative selection: with super-flavor/flavor nodes
- Constraint propagation algorithms for INC-/server-dependencies and localities
- Caching strategies for efficient lookups