NetBricks: Taking the V out of NFV

Aurojit Panda, Sangjin Han, Keon Jang, Melvin Walls, Sylvia Ratnasamy, Scott Shenker
UC Berkeley, Google, ICSI
What the heck is NFV?
A Short Introduction to NFV
A Short Introduction to NFV

Diagram showing a sequence of devices: Firewall, IDS, Cache, and LB.
A Short Introduction to NFV

Network Function Chain

Firewall → IDS → Cache → LB
Why NFV?

• Simplifies **adding new functionality**: Deploy new software.
Why NFV?

• Simplifies \textit{adding new functionality}: Deploy new software.

• Simplifies \textit{developing new functionality}: Write software vs design hardware
Why NFV?

• Simplifies **adding new functionality**: Deploy new software.

• Simplifies **developing new functionality**: Write software vs design hardware

• Reuse **management tools** from other domains.
Why NFV?

- Simplifies **adding new functionality**: Deploy new software.
- Simplifies **developing new functionality**: Write software vs design hardware
- Reuse **management tools** from other domains.
- **Consolidation**: Reduce number of hardware boxes in the network.
Challenges for NFV
Challenges for NFV

- Running NFs
  - Isolation and Performance
Challenges for NFV

- Running NFs
  - Isolation and Performance

- Building NFs
  - High-Level Programming and Performance
Running NFs
Isolation

- **Memory Isolation**: Each NF’s memory cannot be accessed by other NFs.
Isolation

- **Memory Isolation**: Each NF’s memory cannot be accessed by other NFs.

- **Packet Isolation**: When chained, each NF processes packets in isolation.
Isolation

- **Memory Isolation**: Each NF’s memory cannot be accessed by other NFs.

- **Packet Isolation**: When chained, each NF processes packets in isolation.
Isolation

- **Memory Isolation**: Each NF’s memory cannot be accessed by other NFs.

- **Packet Isolation**: When chained, each NF processes packets in isolation.

- **Performance Isolation**: One NF does not affect another’s performance.
Isolation

• **Memory Isolation**: Each NF’s memory cannot be accessed by other NFs.

• **Packet Isolation**: When chained, each NF processes packets in isolation.

• **Performance Isolation**: One NF does not affect another’s performance.
Current Solution

- vSwitch
- NIC
- VM/Container
- NIC
- VM/Container
- Memory Isolation
- Packet Isolation
- Performance
Current Solution

- vSwitch
- NIC...
- NIC

- VM/Container
- VM/Container
- VM/Container

Memory Isolation
Packet Isolation
Performance
Current Solution

vSwitch

NIC

... NIC

NIC

VM/Container

VM/Container

VM/Container

✔ Memory Isolation

Packet Isolation

Performance
Current Solution

- vSwitch
- NIC
- ... NIC
- VM/Container
- VM/Container
- VM/Container

- ✔ Memory Isolation
- Packet Isolation
- Performance
Current Solution

- vSwitch
- NIC
- VM/Container
- NIC
- VM/Container
- VM/Container

- Memory Isolation ✔
- Packet Isolation
- Performance
Current Solution

- Memory Isolation
- Packet Isolation
- Performance
Current Solution

- Memory Isolation
- Packet Isolation
- Performance

Diagram showing a vSwitch connected to NICs and VM/Containers.
Current Solution

- Memory Isolation
- Packet Isolation
- Performance
Current Solution

- NIC
- vSwitch
- Copy
- VM/Container

- Memory Isolation
- Packet Isolation
- Performance
Current Solution

Memory Isolation
Packet Isolation
Performance
Current Solution

- Memory Isolation ✔
- Packet Isolation ✔
- Performance
Current Solution

- Memory Isolation ✔
- Packet Isolation ✔
- Performance ❌
Isolation Costs Performance

![Graph showing isolation costs performance with a high processing rate (Mpps) for no isolation.]

- Processing Rate (Mpps)
Isolation Costs Performance

Processing Rate (Mpps)

No Isolation

OVS VM
Isolation Costs Performance

- No Isolation
- OVS VM
- BESS VM

Processing Rate (Mpps)
Isolation Costs Performance

![Graph showing processing rate (Mpps) for different isolation methods: No Isolation, OVS VM, BESS VM, BESS Container. No Isolation has the highest processing rate, followed by BESS VM and OVS VM, with BESS Container having the lowest.](image-url)
Isolation Costs Performance

Processing Rate (Mpps)

- No Isolation
- Near Ideal
- OVS VM
- BESS VM
- BESS Container
Isolation Costs Performance

- No Isolation
- NetBricks
- OVS VM
- BESS VM
- BESS Container

Processing Rate (Mpps)
NetBricks Runtime Architecture

- NF D
- NF C
- NF B
- NF A
- NF Z
- NF Y
- NF X
- ZCSI Scheduler
- NICs
- DPDK Poll for I/O

Single Process Space

Poll for I/O
NetBricks Runtime Architecture

ZCSI Scheduler

DPDK Poll for I/O

Run to Completion Scheduling

Single Process Space

Poll for I/O

NICS
What about Isolation?
Provide Isolation through Software
ZCSI: Zero Copy Soft Isolation

• VMs and containers impose cost on packets crossing isolation boundaries.

• Frequent operation for many NFs which must support 10s of MPPS.
ZCSI: Zero Copy Soft Isolation

- VMs and containers impose cost on packets crossing isolation boundaries.
  - Frequent operation for many NFs which must support 10s of MPPS.
- Insight: Use type checking (compile time) and runtime checks for isolation.
  - Isolation costs largely paid at compile time (small runtime costs).
Our Approach

- Disallow pointer arithmetic in NF code: use safe subset of languages.
Our Approach

• Disallow pointer arithmetic in NF code: use safe subset of languages.

• Type checks + array bounds checking provide memory isolation.
Our Approach

• Disallow pointer arithmetic in NF code: use safe subset of languages.

• Type checks + array bounds checking provide memory isolation.

• Build on unique types for packet isolation.
Our Approach

• Disallow pointer arithmetic in NF code: use safe subset of languages.

• Type checks + array bounds checking provide memory isolation.

• Build on unique types for packet isolation.

  • Unique types ensure references destroyed after certain calls.
Our Approach

- Disallow pointer arithmetic in NF code: use safe subset of languages.
- Type checks + array bounds checking provide **memory isolation**.
- Build on unique types for **packet isolation**.
  - Unique types ensure references destroyed after certain calls.
  - Ensure only one NF has a reference to a packet.
Our Approach

• Disallow pointer arithmetic in NF code: use safe subset of languages.

• Type checks + array bounds checking provide memory isolation.

• Build on unique types for packet isolation.
  • Unique types ensure references destroyed after certain calls.
  • Ensure only one NF has a reference to a packet.
  • Enables zero copy packet I/O.
Our Approach

• Disallow pointer arithmetic in NF code: use safe subset of languages.

• Type checks + array bounds checking provide memory isolation.

• Build on unique types for packet isolation.
  • Unique types ensure references destroyed after certain calls.
  • Ensure only one NF has a reference to a packet.
  • Enables zero copy packet I/O.

• All of these features implemented on top of Rust.
Software can provide both Memory and Packet Isolation
Benefits of Software Isolation

• Enable better consolidation: multiple NFs can share a core.
Benefits of Software Isolation

• Enable better consolidation: multiple NFs can share a core.

• Normally hard because of context switch costs (~1μs).
Benefits of Software Isolation

• Enable better consolidation: multiple NFs can share a core.
  • Normally hard because of context switch costs (~1µs).
  • In our case just a function call (a few cycles at most).
Benefits of Software Isolation

• Enable better consolidation: multiple NFs can share a core.

• Normally hard because of context switch costs (~1μs).

• In our case just a function call (a few cycles at most).

• Reduce memory and cache pressure for NFV deployments.
Benefits of Software Isolation

• Enable better consolidation: multiple NFs can share a core.
  • Normally hard because of context switch costs (~1µs).
  • In our case just a function call (a few cycles at most).
• Reduce memory and cache pressure for NFV deployments.
  • Zero copy I/O => do not need to copy packets around.
Challenges for NFV

- Running NFs
- Isolation and Performance

- Building NFs
  - High-Level Programming and Performance
How to write NFs?

- **Current**: NF writers concerned about meeting performance targets
How to write NFs?

• **Current**: NF writers concerned about meeting performance targets

• Low level abstractions (I/O, cache aware data structures) and low level code.
How to write NFs?

- **Current**: NF writers concerned about meeting performance targets
  
  - Low level abstractions (I/O, cache aware data structures) and low level code.
  
  - Spend lots of time optimizing how abstractions are used to get performance.
How to write NFs?

• **Current**: NF writers concerned about meeting performance targets

  • Low level abstractions (I/O, cache aware data structures) and low level code.

  • Spend lots of time optimizing how abstractions are used to get performance.

• **Observation**: NFs exhibit common patterns: abstract and optimize these.
How to write NFs?

• **Current:** NF writers concerned about meeting performance targets
  • Low level abstractions (I/O, cache aware data structures) and low level code.
  • Spend lots of time optimizing how abstractions are used to get performance.
• **Observation:** NFs exhibit common patterns: abstract and optimize these.
• What happened in other areas
How to write NFs?

- **Current**: NF writers concerned about meeting performance targets
  - Low level abstractions (I/O, cache aware data structures) and low level code.
  - Spend lots of time optimizing how abstractions are used to get performance.
- **Observation**: NFs exhibit common patterns: abstract and optimize these.
- What happened in other areas
  - MPI to Map Reduce, etc.
<table>
<thead>
<tr>
<th>Abstractions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Packet Processing Abstractions</strong></td>
</tr>
<tr>
<td>Parse/Deparse</td>
</tr>
<tr>
<td>Transform</td>
</tr>
<tr>
<td>Filter</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Byte Stream Processing Abstractions</strong></td>
</tr>
<tr>
<td>Window</td>
</tr>
<tr>
<td>Packetize</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Control Flow</strong></td>
</tr>
<tr>
<td>Group By</td>
</tr>
<tr>
<td>Shuffle</td>
</tr>
<tr>
<td>Merge</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>State Abstractions</strong></td>
</tr>
<tr>
<td>Bounded Consistency State</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Schedule Abstractions</strong></td>
</tr>
<tr>
<td>Invoke</td>
</tr>
</tbody>
</table>
Shuffle Abstraction

Spread packets across cores for scaling

Mux

Counter

Demux

Input

Output

Core 1

Core 2

Core 3

Core 4

Counters
Shuffle Abstraction

Spread packets across cores for scaling

Might even use hardware for this.
Example NF: Maglev

- **Maglev**: Load balancer from Google (NSDI’16).

- Main contribution: a **novel consistent hashing algorithm**.
  - Most of the work in common optimization: batching, scaling cross core.

- NetBricks implementation: **105 lines, 2 hours of grad student time**.

- Comparable performance to optimized code
Managing NFs

Building and Running NFs
Managing NFs

- E2 (SOSP’15)
- Stratos
- FTMB (SIGCOMM ’15)
- FlowTags (NSDI ’14)

Building and Running NFs
Managing NFs

E2 (SOSP’15)
Stratos
FTMB (SIGCOMM ’15)
FlowTags (NSDI ’14)

Building and Running NFs

No Isolation
CoMB (NSDI’12)
xOMB (ANCS’12)
Managing NFs

- E2 (SOSP’15)
- Stratos
- FTMB (SIGCOMM ’15)
- FlowTags (NSDI ’14)

Building and Running NFs

No Isolation

- CoMB (NSDI’12)
- xOMB (ANCS’12)

VM Isolation

- NetVM (IEEE TNSM)
- ClickOS (NSDI’14)
- HyperSwitch (ATC’13)
- mSwitch (SOSR’15)
Conclusion

- Performance demands for NFV require forwarding 10-100 MPPS.

- Requires **isolation** for consolidation.
  - Software isolation is necessary to meet performance requirements.

- Requires low level optimization, slowing down NF development.
  - Abstract operators + UDF can simplify development without sacrificing performance.
Conclusion

- Performance demands for NFV require forwarding 10-100 MPPS.

- Requires isolation for consolidation.
  
  - Software isolation is necessary to meet performance requirements.

- Requires low level optimization, slowing down NF development.
  
  - Abstract operators + UDF can simplify development without sacrificing performance.

Code available at http://netbricks.io/
Backup
Both Memory Isolation and I/O Induce Overheads