NetBricks: Taking the V out of NFV

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

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

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

• Simplifies **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.
NFV Requirements

- **High Packet Rates**: Must keep up with line rate which is >10MPPS
NFV Requirements

- **High Packet Rates**: Must keep up with line rate which is >10MPPS

- **Low Latency**: Used for applications like VoIP and video conferencing
NFV Requirements

- **High Packet Rates**: Must keep up with line rate which is >10MPPS
- **Low Latency**: Used for applications like VoIP and video conferencing
- **NF Chaining**: Packets go through sequence of NFs
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.

- **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

Memory Isolation
Packet Isolation
Performance

vSwitch
NI...
Current Solution

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

Memory Isolation
Packet Isolation
Performance
Current Solution

- Memory Isolation
- Packet Isolation
- Performance

Diagram:
- vSwitch
- NIC...
- VM/Container
- VM/Container
- VM/Container
Current Solution

- Memory Isolation
- Packet Isolation
- Performance
Current Solution

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

NIC

vSwitch

Copy

VM/Container

VM/Container

VM/Container

NIC

NIC

✔ Memory Isolation

Packet Isolation

Performance
Current Solution

NIC

Memory Isolation
Packet Isolation
Performance
Current Solution

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

✔ Memory Isolation
Packet Isolation
Performance
Current Solution

- Memory Isolation ✔
- Packet Isolation ✔
- Performance

Diagram:
- vSwitch
- Memory Isolation
- Packet Isolation
- Performance
Current Solution

- Memory Isolation: ✔
- Performance: ❌
- Packet Isolation: ✔

Diagram:
- vSwitch
- NIC → Copy
- VM/Container
- VM/Container
- VM/Container
- NIC → NIC
Isolation Costs Performance

- No Isolation
- OVS VM
Isolation Costs Performance

- No Isolation
- OVS VM
- BESS VM

Processing Rate (Mpps)
Isolation Costs Performance

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

Processing Rate (Mpps)
Isolation Costs Performance

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

ZCSI Scheduler

Single Process Space

DPDK Poll for I/O

NICs
NetBricks Runtime Architecture

Function Call

ZCSI Scheduler

Single Process Space

DPDK Poll for I/O

NICs
NetBricks Runtime Architecture

ZCSI Scheduler

NF A → NF B → NF C → NF D

NF Y → NF Z

DPDK Poll for I/O

Single Process Space

NF A → NF B → NF C → NF D

NF Y → NF Z

DPDK Poll for I/O

NICs
What about Isolation?
ZCSI: Zero Copy Soft Isolation

- VMs and containers impose cost on packets crossing isolation boundaries.
ZCSI: Zero Copy Soft Isolation

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

- **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 Isolation

• Provides memory and packet isolation.
Software Isolation

- Provides **memory** and **packet isolation**.
- Improved **consolidation**: multiple NFs can share a core.
Software Isolation

- Provides **memory** and **packet isolation**.

- Improved **consolidation**: multiple NFs can share a core.

- Function call to NF (~ few cycles) vs context switch (~1μs).
Software Isolation

- Provides **memory** and **packet isolation**.
- Improved **consolidation**: multiple NFs can share a core.
  - Function call to NF (~ few cycles) vs context switch (~1µs).
- Reduce **memory** and **cache pressure**.
Software Isolation

- Provides **memory** and **packet isolation**.
- Improved **consolidation**: multiple NFs can share a core.
  - Function call to NF (~ few cycles) vs context switch (~1μs).
- Reduce **memory** and **cache pressure**.
  - 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.
Abstractions

Packet Processing
- Parse/Deparse
- Transform
- Filter

Control Flow
- Group By
- Shuffle
- Merge

Byte Stream
- Window
- Packetize

State
- Bounded
- Consistency
## Abstractions

### Packet Processing
- Parse/Deparse
- Transform
- Filter

### Control Flow
- Group By
- Shuffle
- Merge

### Byte Stream
- Window
- Packetize

### State
- Bounded Consistency

### UDF
- Header
- UDF
- UDF
- UDF
- UDF
- UDF
Shuffle Abstraction

Spread packets across cores for scaling
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 time**.

- Comparable performance to optimized code
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

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

- Software isolation is necessary for high performance NFV.
- Type checking + bound checking + unique types.
- Performance is not anathema to high-level programming
- Abstract operators + UDF simplify development.

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

![Bar Chart]

- No Isolation
- 0-Copy Container
- BESS Container

Processing Rate (Mpps)