Performance Contracts for Software Network Functions

Rishabh Iyer, Luis Pedrosa, Arseniy Zaostrovnykh, Solal Pirelli, Katerina Argyraki, George Candea
Software Network Functions – Pros and Cons

- Increased flexibility ✓
- Reduced capital and operating expenses ✓
- Programming errors ×
- Unexpected performance behaviour ×
Dealing with unexpected NF performance

- **Goal**: Comprehensive understanding of NF’s performance profile
  - Operators – capacity planning and anticipate attacks
  - Developers – informed development decisions

- **Previous work** [NSDI’12, NSDI’18, SIGCOMM’18]
  - Focus on narrow subset of input workloads
  - Offer few completeness guarantees
Performance Contracts for NFs

- Abstraction for users to parameterize arbitrary input workloads
- Predict performance for workload spec without running NF
- Performance predicted as function of **Performance Critical Variables (PCVs)**
- Per-packet metrics: Instruction count, memory accesses, latency (cycles)
Outline

- What is a performance contract?
- How does Bolt generate contracts?
- Evaluation & Use-Case
void MAC_bridge(pkt* p, port in_port) {
    if (invalid_hdr(p)) {
        DROP(p);
        return;
    }
    MACtable_put(p->src_mac, &in_port);

    if (MACtable_get(p->dst_mac, &out_port))
        FORWARD(p, out_port);
    else
        BROADCAST(p, in_port);
}
void MAC_bridge(pkt* p, port in_port) {

    if (invalid_hdr(p)) {
        DROP(p);
        return;
    }

    MACtable_put(p->src_mac, &in_port);

    if (MACtable_get(p->dst_mac, &out_port))
        FORWARD(p, out_port);
    else
        BROADCAST(p, in_port);
}
```c
void MAC_bridge(pkt* p, port in_port) {

    if (invalid_hdr(p)) {
        DROP(p);
        return;
    }

    MACtable_put(p->src_mac, &in_port);

    if (MACtable_get(p->dst_mac, &out_port))
        FORWARD(p, out_port);
    else
        BROADCAST(p, in_port);
}
```
void MAC_bridge(pkt* p, port in_port) {

    if (invalid_hdr(p)) {
        DROP(p);
        return;
    }

    MACtable_put(p->src_mac,&in_port);

    if (MACtable_get(p->dst_mac,&out_port))
        FORWARD(p,out_port);
    else
        BROADCAST(p,in_port);
}
void MAC_bridge(pkt* p, port in_port) {
    if (invalid_hdr(p)) {
        DROP(p);
        return;
    }
    MACtable_put(p->src_mac, &in_port);
    if (MACtable_get(p->dst_mac, &out_port))
        FORWARD(p, out_port);
    else
        BROADCAST(p, in_port);
}
Performance Contracts Example

```c
void MAC_bridge(pkt* p, port in_port) {
    if (invalid_hdr(p)) {
        DROP(p);
        return;
    }
    MACtable_put(p->src_mac, &in_port);
    if (MACtable_get(p->dst_mac, &out_port))
        FORWARD(p, out_port);
    else
        BROADCAST(p, in_port);
}
```

Performance Contract for MAC_bridge
Metric: Lines of pseudo-code

<table>
<thead>
<tr>
<th>Traffic Class</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Performance Contracts Example

```c
void MAC_bridge(pkt* p, port in_port) {
    if (invalid_hdr(p)) {
        DROP(p);
        return;
    }
    MACtable_put(p->src_mac,&in_port);
    if (MACtable_get(p->dst_mac,&out_port))
        FORWARD(p,out_port);
    else
        BROADCAST(p,in_port);
}
```

Performance Contract for MAC_bridge
Metric: Lines of pseudo-code

<table>
<thead>
<tr>
<th>Traffic Class</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invalid Header</td>
<td></td>
</tr>
<tr>
<td>Valid, DestMAC known</td>
<td></td>
</tr>
<tr>
<td>Valid, DestMAC unknown</td>
<td></td>
</tr>
</tbody>
</table>
Performance Contracts Example

```c
void MAC_bridge(pkt* p, port in_port) {
    if (invalid_hdr(p)) {
        DROP(p);
        return;
    }
    MACtable_put(p->src_mac,&in_port);

    if (MACtable_get(p->dst_mac,&out_port))
        FORWARD(p,out_port);
    else
        BROADCAST(p,in_port);
}
```

Performance Contract for MAC_bridge

Metric: Lines of pseudo-code

<table>
<thead>
<tr>
<th>Traffic Class</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invalid Header</td>
<td>3</td>
</tr>
<tr>
<td>Valid, DestMAC known</td>
<td>3C + 20</td>
</tr>
<tr>
<td>Valid, DestMAC unknown</td>
<td>3C + 100</td>
</tr>
</tbody>
</table>

C = Number of hash collisions
Using performance contracts

Spec 1: Unconstrained traffic

Performance Contract for MAC_bridge
Metric: Lines of pseudo-code

<table>
<thead>
<tr>
<th>Traffic Class</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invalid Header</td>
<td>3</td>
</tr>
<tr>
<td>Valid, DestMAC known</td>
<td>$3C + 20$</td>
</tr>
<tr>
<td>Valid, DestMAC unknown</td>
<td>$3C + 100$</td>
</tr>
</tbody>
</table>

$C = \text{Number of hash collisions}$
Using performance contracts

Spec 1: Unconstrained traffic

\[ C = \text{max}_{\text{collisions}} \]

Predicted performance:
\[ 3(\text{max}_{\text{collisions}}) + 100 \]

Performance Contract for MAC\_bridge

Metric: Lines of pseudo-code

<table>
<thead>
<tr>
<th>Traffic Class</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invalid Header</td>
<td>3</td>
</tr>
<tr>
<td>Valid, DestMAC known</td>
<td>3(C + 20)</td>
</tr>
<tr>
<td>Valid, DestMAC unknown</td>
<td>3(C + 100)</td>
</tr>
</tbody>
</table>

\( C = \) Number of hash collisions
Using performance contracts

Spec 2: No hash collisions

⇒ $C = 0$

Predicted performance: 100

Performance Contract for MAC_bridge
Metric: Lines of pseudo-code

<table>
<thead>
<tr>
<th>Traffic Class</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invalid Header</td>
<td>3</td>
</tr>
<tr>
<td>Valid, DestMAC known</td>
<td>$3C + 20$</td>
</tr>
<tr>
<td>Valid, DestMAC unknown</td>
<td>$3C + 100$</td>
</tr>
</tbody>
</table>

$C =$ Number of hash collisions
Using performance contracts

Spec 3: Valid, no collisions, DestMAC known

\[ C = 0 \]

Predicted performance:
20

Performance Contract for MAC_bridge
Metric: Lines of pseudo-code

<table>
<thead>
<tr>
<th>Traffic Class</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invalid Header</td>
<td>3</td>
</tr>
<tr>
<td>Valid, DestMAC known</td>
<td>3C + 20</td>
</tr>
<tr>
<td>Valid, DestMAC unknown</td>
<td>3C + 100</td>
</tr>
</tbody>
</table>

\[ C = \text{Number of hash collisions} \]
Using performance contracts

Spec 3: Valid, no collisions, DestMAC known

\[ C = 0 \]

Predicted performance: 20

Performance Contract for MAC_bridge

Metric: Lines of pseudo-code

<table>
<thead>
<tr>
<th>Traffic Class</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invalid Header</td>
<td>3</td>
</tr>
<tr>
<td>Valid, DestMAC known</td>
<td>3C + 20</td>
</tr>
<tr>
<td>Valid, DestMAC unknown</td>
<td>3C + 100</td>
</tr>
</tbody>
</table>

\[ C = \text{Number of hash collisions} \]

Contracts quantify performance for all traffic classes of the NF.
Users query contract for performance of specific input workloads.
Outline

- What is a performance contract?
- How does Bolt generate contracts?
- Evaluation & Use-Case
Generating performance contracts recursively

NF chain

Firewall + NAT + Bridge =

[Diagram showing the interconnection of Firewall, NAT, and Bridge with performance contracts]

(20)
Generating performance contracts recursively

Individual NF

Stateless Code

```c
void MAC_bridge(pkt x, port in_port) {
    if (invalid_hdr(p)) {
        DROP(p);
        return;
    }
    MACtable_put(p->src_mac, &in_port);
    if (MACtable_get(p->dst_mac, &out_port))
        FORWARD(p, out_port);
    else
        BROADCAST(p, in_port);
}
```

Stateful NF data structures

MACtable put(), get()
Generating performance contracts recursively

Individual NF

Stateless Code
(Simple to analyze)*

Stateful NF
(data structures
(Hard to analyze)*

MACtable
put(), get()

*A.Zaostrovnykh, S.Pirelli, L.Pedrosa, K.Argyraki, G.Candea “A Formally Verified NAT” SIGCOMM 2017
Generating performance contracts recursively

- Well defined separation between stateful and stateless NF code*
  - NFs typically have well defined, isolated state

- Encapsulate NF state using a library of data structures

- Stateful data structures – Base case of recursive process
  - Analyze once, reuse across NFs

* A.Zaostrovnykh, S.Pirelli, L.Pedrosa, K.Argyraki, G.Candea “A Formally Verified NAT” SIGCOMM 2017
Analyzing stateful data structures

\[ \text{Performance}_{NF} = f(\text{input packet}, NF \text{ state}, \text{config}, \ldots) \]

- Cannot account for all possible packet histories -> Path explosion
- BUT, performance of MACtable depends **ONLY** on number of hash collisions
Performance Critical Variables (PCVs)

- Abstract away NF state specificities
- Succinctly summarize impact of packet history, configuration on performance
- Tailor legibility and detail to audience

<table>
<thead>
<tr>
<th>Traffic Class</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unconstrained</td>
<td>$1C + 2$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Traffic Class</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key present</td>
<td>$2C + 12$</td>
</tr>
<tr>
<td>Key absent</td>
<td>$2C + 7$</td>
</tr>
</tbody>
</table>

$C = \text{Number of hash collisions}$

Only PCV required to summarize perf in terms of lines of pseudo-code
Generating Performance Contracts for NFs

- Symbolically execute stateless code to traverse all execution paths

- While traversing each path
  - Keep track of performance metrics for stateless code
  - Plug in contracts for stateful code using path constraints
void MAC_bridge(pkt* p, port in_port) {
    if (invalid_hdr(p)) {
        DROP(p);
        return;
    }
    MACtable_put(p->src_mac, &in_port);

    if (MACtable_get(p->dst_mac, &out_port))
        FORWARD(p, out_port);
    else
        BROADCAST(p, in_port);
}
Generating Performance Contracts for NFs

```c
void MAC_bridge(pkt* p, port in_port) {
    if (invalid_hdr(p)) {
        DROP(p);
        return;
    }
    MACtable_put(p->src_mac,&in_port);

    if (MACtable_get(p->dst_mac,&out_port))
        FORWARD(p,out_port);
    else
        BROADCAST(p,in_port);
}
```

<table>
<thead>
<tr>
<th>Traffic Class</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unconstrained</td>
<td>$1C + 2$</td>
</tr>
</tbody>
</table>

Contract for MACtable_put
Generating Performance Contracts for NFs

```c
void MAC_bridge(pkt* p, port in_port) {
    if (invalid_hdr(p)) {
        DROP(p);
        return;
    }
    MACtable_put(p->src_mac, &in_port);
    if (MACtable_get(p->dst_mac, &out_port))
        FORWARD(p, out_port);
    else
        BROADCAST(p, in_port);
}
```

### Traffic Class and Performance

<table>
<thead>
<tr>
<th>Traffic Class</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key present</td>
<td>$2C + 12$</td>
</tr>
<tr>
<td>Key absent</td>
<td>$2C + 7$</td>
</tr>
</tbody>
</table>

- **Invalid:** $3$
- **Valid:** $1$
- **DestMAC known:** $2C + 12$
- **DestMAC unknown:** $2C + 7$
Generating Performance Contracts for NFs

```c
void MAC_bridge(pkt* p, port in_port) {
    if (invalid_hdr(p)) {
        DROP(p);
        return;
    }
    MACtable_put(p->src_mac, &in_port);
    if (MACtable_get(p->dst_mac, &out_port))
        FORWARD(p, out_port);
    else
        BROADCAST(p, in_port);
}
```

![Decision tree diagram]

- Invalid: 3
- DestMAC known: 2C + 12
- DestMAC unknown: 2C + 7
- Valid: 1
  - 1C + 2
  - 5
  - 90
Generating Performance Contracts for NFs

Performance Contract for MAC_bridge

<table>
<thead>
<tr>
<th>Traffic Class</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invalid Header</td>
<td>3</td>
</tr>
<tr>
<td>Valid, DestMAC known</td>
<td>$3C + 20$</td>
</tr>
<tr>
<td>Valid, DestMAC unknown</td>
<td>$3C + 100$</td>
</tr>
</tbody>
</table>

$C = \text{Number of hash collisions}$

\[ \text{Invalid} \]
\[ \begin{align*}
3 \\
\end{align*} \]

\[ \text{DestMAC known} \]
\[ \begin{align*}
2C + 12 \\
5 \\
\end{align*} \]

\[ \text{Valid} \]
\[ \begin{align*}
1 \\
1C + 2 \\
\end{align*} \]

\[ \text{DestMAC unknown} \]
\[ \begin{align*}
2C + 7 \\
90 \\
\end{align*} \]
Performance Contracts for NF chains

- Generate performance contracts for individual NFs in chain
- Pair together traffic classes from communicating NFs
- For each pair - AND respective constraints together
  - Equate packet sent by first NF to packet received by second
Performance Contract for NF chains - Example

Firewall
- Drops packets with IP options
- Fast path – No IP options
- Slow path – IP options

Router

<table>
<thead>
<tr>
<th>Traffic Class</th>
<th>Perf</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP opt</td>
<td>20</td>
</tr>
<tr>
<td>No IP opt</td>
<td>50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Traffic Class</th>
<th>Perf</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP opt</td>
<td>500</td>
</tr>
<tr>
<td>No IP opt</td>
<td>60</td>
</tr>
</tbody>
</table>

NF chain

<table>
<thead>
<tr>
<th>Traffic Class</th>
<th>Perf</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP opt</td>
<td>20</td>
</tr>
<tr>
<td>No IP opt</td>
<td>110</td>
</tr>
</tbody>
</table>

<F1> <F2, R2>
Outline

- What is a performance contract?
- How does Bolt generate contracts?
- Evaluation and Use-Case
Evaluation setup & methodology

- 4 NFs - NAT, Maglev-like LB, MAC bridge, LPM router
  - Analyze NF logic + DPDK + NIC driver*

- Metrics – instructions executed, memory accesses, execution cycles

- Testbed - Intel Xeon E5-2667v2 3.3GHz, 82599ES 10Gb NICs

- Compare predicted vs measured performance for various packet classes

*S.Pirelli, A.Zaostrovnykh, G.Candea “A Formally Verified NAT Stack” KBNETS Workshop - SIGCOMM 2018
Predictions for Instruction Count, Memory Accesses

Results for Maglev-like Load Balancer

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LB1</td>
<td>Unconstrained traffic</td>
</tr>
<tr>
<td>LB2</td>
<td>Client packet, new flow</td>
</tr>
<tr>
<td>LB3</td>
<td>Client packet, existing flow, unresponsive backend</td>
</tr>
<tr>
<td>LB4</td>
<td>Client packet, existing flow, existing backend</td>
</tr>
<tr>
<td>LB5</td>
<td>Heartbeat packets</td>
</tr>
</tbody>
</table>

Max prediction gap – 7.5% (IC) and 7.6% (MA)
Why is there a prediction gap?

- Source 1: Trade-off between precision and legibility in PCVs
  - Can be overcome by exposing more detail

- Source 2: Differences between analyzed and production code
  - Disabled link time optimizations in analyzed code
Use Case – Informed cost-benefit analysis

- Example: Bridge with randomized hash table
  - Incorporates random key into hash function
  - Rehashes all entries with a new key when collisions greater than a threshold

- Question: Where to place threshold?
  - Avoid rehashing under normal operation
  - Should rehash under attack
Use Case – Informed cost-benefit analysis
Bolt allows operators to visualize the consequences of their decisions.

**Use Case – Informed cost-benefit analysis**

![Graph showing the relationship between the number of hash collisions and the CCDF and Predicted IC.](image)

**CCDF**

- Predicted IC

**Number of hash collisions**

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8

**Predicted IC**

- 10^3
- 10^4
- 10^5
- 10^6
Performance Contracts for NFs

- Abstraction for users to parameterize arbitrary input workloads
- Predict performance for workload spec without running NF
- Performance predicted as function of **Performance Critical Variables (PCVs)**

[QR Code: bolt-perf-contracts.github.io]
Backup Slides

- Distiller
- Results – IC, MA
- Results – NF chains
- Results – Latency
- Full Blown Contract
The Bolt Distiller

- Users need to know which traffic classes are likely
- Bolt is a static analysis tool, cannot know probabilities of each traffic class

- The Bolt Distiller
  - Input – A representative packet trace
  - Output - Execution path taken by each packet & values of PCVs
  - Users can then extrapolate the likelihood and query contract accordingly
Predictions for Instruction Count, Memory Accesses

Results for NAT, Bridge

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAT1</td>
<td>Unconstrained traffic</td>
</tr>
<tr>
<td>NAT2</td>
<td>Client packet, new flow</td>
</tr>
<tr>
<td>NAT3</td>
<td>Existing flow</td>
</tr>
<tr>
<td>NAT4</td>
<td>External, dropped packet</td>
</tr>
<tr>
<td>BR1</td>
<td>Unconstrained traffic</td>
</tr>
<tr>
<td>BR2</td>
<td>Broadcast traffic</td>
</tr>
<tr>
<td>BR3</td>
<td>Unicast traffic</td>
</tr>
</tbody>
</table>

Bolt predicts IC & MA accurately, irrespective of NF/Traffic Class
Predictions for NF chains

- NFs chained together
  - Firewall – drops packets with IP options
  - Router – Fast path (No IP options), Slow path (packets with IP options)

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>Packets with IP options</td>
</tr>
<tr>
<td>C2</td>
<td>Packets without IP options</td>
</tr>
</tbody>
</table>

![Graph showing Instruction Count (IC) and Memory Accesses (MA)]
# Predictions for Latency (Execution Cycles)

## Results for Maglev-like Load Balancer

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LB1</td>
<td>Unconstrained traffic</td>
</tr>
<tr>
<td>LB2</td>
<td>Client packet, new flow</td>
</tr>
<tr>
<td>LB3</td>
<td>Client packet, existing flow, unresponsive backend</td>
</tr>
<tr>
<td>LB4</td>
<td>Client packet, existing flow, existing backend</td>
</tr>
<tr>
<td>LB5</td>
<td>Heartbeat packets</td>
</tr>
</tbody>
</table>

9x for pathological traffic, 3x for typical traffic

![Over-estimation Ratio Graph](chart.png)
## Predictions for Execution Cycles

### Results for LB, NAT, Bridge, LPM

<table>
<thead>
<tr>
<th>NF+Class</th>
<th>Predicted Bound</th>
<th>Measured Cycles</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAT1</td>
<td>591,948,908,371</td>
<td>65,217,699,390</td>
<td>9.08</td>
</tr>
<tr>
<td>NAT2</td>
<td>7,401</td>
<td>2,376</td>
<td>3.11</td>
</tr>
<tr>
<td>NAT3</td>
<td>5,142</td>
<td>1,789</td>
<td>2.87</td>
</tr>
<tr>
<td>NAT4</td>
<td>2,956</td>
<td>884</td>
<td>3.34</td>
</tr>
<tr>
<td>Br1</td>
<td>295,984,939,878</td>
<td>32,383,472,634</td>
<td>9.14</td>
</tr>
<tr>
<td>Br2</td>
<td>7,329</td>
<td>2,013</td>
<td>3.64</td>
</tr>
<tr>
<td>Br3</td>
<td>7,383</td>
<td>1,808</td>
<td>4.08</td>
</tr>
<tr>
<td>LB1</td>
<td>591,969,879,756</td>
<td>66,062,284,173</td>
<td>8.96</td>
</tr>
<tr>
<td>LB2</td>
<td>5,299</td>
<td>2,386</td>
<td>2.22</td>
</tr>
<tr>
<td>LB3</td>
<td>8,108</td>
<td>2,541</td>
<td>3.19</td>
</tr>
<tr>
<td>LB4</td>
<td>4,300</td>
<td>2,310</td>
<td>1.86</td>
</tr>
<tr>
<td>LB5</td>
<td>4,837</td>
<td>2,079</td>
<td>2.33</td>
</tr>
<tr>
<td>LPM1</td>
<td>1,419</td>
<td>967</td>
<td>1.46</td>
</tr>
<tr>
<td>LPM2</td>
<td>1,015</td>
<td>545</td>
<td>1.86</td>
</tr>
</tbody>
</table>

Table 3: Accuracy of execution cycle performance contracts for multiple NFs and packet classes.
Table 6: VigNAT performance contract. Instructions are described as a function of the number of expired flows ($e$) and the number of hash collisions ($c$) and bucket traversals ($t$) incurred in the hash table.