NSight

How to diagnose nanosecond network latencies in rich end-host stacks

Roni Haecki
Radhika Niranjan Mysore
Lalith Suresh
Gerd Zellweger
Bo Gan
Timothy Merrifield
Sujata Banerjee
Timothy Roscoe
What causes network latencies *within* the end-host stack?
What causes network latencies within the end-host stack?
An ideal tool shows **what** is delaying messages and **how often**
An ideal tool shows \textit{what} is delaying messages and \textit{how often}.

A \textbf{holistic} picture of sources of latencies anywhere in the stack.

This talk.
31 fixes to network latency problems in end-host stacks ...
... built in the dark without a holistic profiling tool

<table>
<thead>
<tr>
<th>netmap</th>
<th>Arrakis</th>
<th>StackMap</th>
<th>TAS</th>
<th>Snap (Pony Express)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Affinity-Accept</td>
<td>VMA</td>
<td>Tales of the Tail</td>
<td>FlexNIC</td>
<td>eRPC</td>
</tr>
<tr>
<td>Chronos</td>
<td>DPDK</td>
<td>Sandstorm</td>
<td>FaSST</td>
<td>NeBuLa</td>
</tr>
<tr>
<td>MegaPipe</td>
<td>Tail at Scale</td>
<td>IX</td>
<td>Killer microseconds</td>
<td>BMC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Manycore NI</td>
<td>F-stack</td>
<td>隐身</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fastsocket</td>
<td>PerfISO</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Parties</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Caladan</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Perséphone</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Network latencies sources within the end-host stack
Network latencies sources within the end-host stack
Network latencies sources within the end-host stack
Network latencies sources within the end-host stack
Network latencies sources within the end-host stack
Network latencies sources within the end-host stack
Network latencies sources within the end-host stack

NIC -> PCIE -> DRIVER -> NETWORK STACK -> KERNEL CORE -> INTERFERENC -> APPLICATION CORE

END-HOST
Network latencies sources within the end-host stack
Essential to examine the **entire processing pipeline** at once
Essential to examine the *entire processing pipeline* at once
Essential to examine the *entire processing pipeline* at once

Imagine what we can do if we have a tool that shows *all sources of latency all at once.*
Essential to examine the **entire processing pipeline** at once

Imagine what we can do if we have a tool that shows **all sources of latency all at once**.

Goal 1: It must **directly and precisely measure** the network-latency impact of end-hosts
Goal 2: Its **measurement overhead** must be **tiny** so we can measure the **entire stack**
NSight: How to build a holistic latency diagnosis tool

<table>
<thead>
<tr>
<th>1. Profiling (online)</th>
<th>2. Analysis (offline)</th>
<th>3. Visualization (offline)</th>
</tr>
</thead>
</table>

19
### NSight: How to build a holistic latency diagnosis tool

<table>
<thead>
<tr>
<th>1. Profiling (online)</th>
<th>2. Analysis (offline)</th>
<th>3. Visualization (offline)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>P1.1:</strong> Examine the entire pipeline</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>P1.2:</strong> Minimize latency perturbation</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>P1.3:</strong> Construct a single timeline</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>P1.4:</strong> Track messages across cores</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Problem 1.1: Examine the entire pipeline
Existing tools miss parts of the pipeline
Problem 1.1: Examine the entire pipeline

Existing tools miss parts of the pipeline

TOOLS THAT USE KPROBES, UPROBES, TRACEPOINTS, SOFTWARE TRACING (e.g., eBPF, Ftrace)
Problem 1.1: Examine the entire pipeline

Solution: Profile entire *message-lifetimes*

Track *all system activity* messages encounter between NIC and application
Problem 1.1: Examine the entire pipeline

Solution: Profile entire *message-lifetimes*

- NIC
- MESSAGE LIFETIME
- $T_N$
- END-HOST
- TIME
Problem 1.1: Examine the entire pipeline

Solution: Profile entire *message-lifetimes*

---

![Diagram showing NIC, PCIE, and KERNEL CORE stages with MESSAGE LIFETIME and GAP annotations.](image-url)
Problem 1.1: Examine the entire pipeline
Solution: Profile entire *message-lifetimes*

MESSAGE LIFETIME

\[ T_N \]  

MESSAGE LATENCY
Problem 1.1: Examine the entire pipeline

Solution: Profile entire message-lifetimes

MESSAGE LIFETIME

MESSAGE LATENCY

TIME

END-HOST
Problem 1.1: Examine the entire pipeline  
Solution: Profile entire *message-lifetimes*

---

**MESSAGE LIFETIME**

- **GAP**
- **TIME**
- **MESSAGE LATENCY**

- **T_N**
- **T_H**
Problem 1.1: Examine the entire pipeline

Solution: Profile entire **message-lifetimes**

**MESSAGE LIFETIME**

\[ T_N \quad \text{GAP} \quad T_H \]

**MESSAGE LATENCY**
Problem 1.1: Examine the entire pipeline

Solution: Profile entire *message-lifetimes*

**MESSAGE LIFETIME**

- $T_N$: TIME
- $T_H$: TIME
- $T_A$: TIME

**MESSAGE LATENCY**
Problem 1.1: Examine the entire pipeline

Solution: Profile entire *message-lifetimes*

---

**MESSAGE LIFETIME**

- \( T_N \) (GAP)
- \( T_H \)
- \( T_A \)

**MESSAGE LATENCY**

- \( T_N \) to \( T_H \)
- \( T_H \) to \( T_A \)
Problem 1.1: Examine the entire pipeline

Solution: Profile entire *message-lifetimes*

---

**MESSAGE LIFETIME**

- **T_N**
- **GAP**
- **T_H**
- **T_A**

**MESSAGE LATENCY**

**DEVIAION**

---

NIC - PCIE - KERNEL CORE - APPLICATION - END-HOST
Problem 1.2: Minimize latency perturbation

Existing tools have high overheads and cannot be turned on everywhere at once

TOOLS THAT USE KPROBES, UPROBES, TRACEPOINTS, SOFTWARE TRACING (e.g., eBPF, Ftrace)
Problem 1.2: Minimize latency perturbation
Existing tools have high overheads and cannot be turned on everywhere at once
Problem 1.2: Minimize latency perturbation
Existing tools have high overheads and cannot be turned on everywhere at once
Problem 1.2: Minimize latency perturbation
Existing tools have high overheads and cannot be turned on everywhere at once
Problem 1.2: Minimize latency perturbation
Existing tools have high overheads and cannot be turned on everywhere at once.

Tools add *variable latencies* to profiled functions that look like true latency deviations.
Problem 1.2: Minimize latency perturbation
Existing tools have high overheads and cannot be turned on everywhere at once

Baseline and CPU hardware profiling

Fraction

memcached message handling latency [us]

Baseline
eBPF–1
Ftrace
Intel–PT
Problem 1.2: Minimize latency perturbation
Solution: Use *CPU hardware profiling* to profile end-host stack

Baseline and CPU hardware profiling and NSight
Problem 1.2: Minimize latency perturbation
Solution: Use *CPU hardware profiling* to profile end-host stack

Baseline and CPU hardware profiling and NSight

CPU profilers also track *deviations introduced* by the *profiling tool*. 
# NSight: How to build a holistic latency diagnosis tool

<table>
<thead>
<tr>
<th>1. Profiling (online)</th>
<th>2. Analysis (offline)</th>
<th>3. Visualization (offline)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hardware profiling</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Per-function elapsed time</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Problem 1.3: Create a single timeline

CPU PROFILER (e.g., INTEL PT)

NIC → PCIE

KERNEL CORE

DRIVER

NETWORK STACK

INTERFERENCE

SOCKET LAYER

APPLICATION

APPLICATION CORE

CPU PROFILES

TIME
Problem 1.3: Create a single timeline

- NIC
- PCIE
- DRIVER
- NETWORK STACK
- APPLICATION CORE
- INTERFERENCE
- SOCKET LAYER
- CPU PROFILER (e.g., INTEL PT)

CPU PROFILES

MESSAGE PROFILE

TIME

T_N

T_A
Problem 1.3: Create a single timeline

MESSAGE LIFETIME

TIME

GAP

T_N

T_A

NIC  PCIE  DRIVER  NETWORK STACK  KERNEL CORE  INTERFERENCE  SOCKET LAYER  APPLICATION  APPLICATION CORE

CPU PROFILER (e.g., INTEL PT)
Problem 1.3: Create a single timeline

Time synchronization protocols tamper with NIC and software clocks

CPU PROFILES

MESSAGE PROFILE

Tampered by PTP, NTP
Problem 1.4: Track messages across cores
No OS support to track messages as they transit cores
Problem 1.4: Track messages across cores

No OS support to track messages as they transit cores

Which cores processed the message? At what times?
Problem 1.4: Track messages across cores

Solution: Track cores at *message-processing boundaries*

- NIC
- PCIE
- DRIVER
- NETWORK STACK
- KERNEL CORE
- CPU PROFILER (e.g., INTEL PT)
- INTERFERENCE
- SOCKET LAYER
- APPLICATION

MESSAGE LIFETIME

- GAP
- $T_N$
- $T_H$
- $T_A$

TIME
Problem 1.4: Track messages across cores

Solution: Track cores at **message-processing boundaries**

Track **message hand-offs** across cores to track (1) system activity that impacts their latency and (2) **cross-message interference**.
Problem 1.4: Track messages across cores

Solution: Track cores at message-processing boundaries

All three timestamps collected per-message using socket library.
**NSight: How to build a holistic latency diagnosis tool**

<table>
<thead>
<tr>
<th>1. Profiling (online)</th>
<th>2. Analysis (offline)</th>
<th>3. Visualization (offline)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU Hardware profiling</strong></td>
<td><strong>Application-network interface interposition</strong></td>
<td></td>
</tr>
<tr>
<td>Per-function elapsed time</td>
<td>Per-message crossing times</td>
<td></td>
</tr>
</tbody>
</table>
NSight: How to build a holistic latency diagnosis tool

1. Profiling (online)
   - CPU Hardware profiling
   - Application-network interface interposition
   - Per-function elapsed time
   - Per-message crossing times

2. Analysis (offline)
   - Typical lifetimes
   - Network message lifetimes

3. Visualization (offline)
### NSight: How to build a holistic latency diagnosis tool

<table>
<thead>
<tr>
<th>1. Profiling (online)</th>
<th>2. Analysis (offline)</th>
<th>3. Visualization (offline)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU Hardware profiling</strong></td>
<td><strong>Detect anomalies</strong></td>
<td><strong>Typical lifetimes</strong></td>
</tr>
<tr>
<td>Per-function elapsed time</td>
<td></td>
<td>Anomalous lifetimes</td>
</tr>
<tr>
<td><strong>Application-network interface interposition</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Per-message crossing times</td>
<td>Network message lifetimes</td>
<td></td>
</tr>
</tbody>
</table>

**Per-function elapsed time**

- CPU Hardware profiling
  - Application-network interface interposition
  - Per-message crossing times

**Per-message crossing times**

- Detect anomalies
  - Typical lifetimes
  - Anomalous lifetimes

**Network message lifetimes**

- Detect anomalies
  - Typical lifetimes
  - Anomalous lifetimes
## NSight: How to build a holistic latency diagnosis tool

<table>
<thead>
<tr>
<th>1. Profiling (online)</th>
<th>2. Analysis (offline)</th>
<th>3. Visualization (offline)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU Hardware profiling</strong></td>
<td><strong>P2.1: Diagnose causes of deviations</strong></td>
<td></td>
</tr>
<tr>
<td>Per-function elapsed time</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Application-network interface interposition</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Per-message crossing times</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Process data**: 1. Profiling (online)  
  - CPU Hardware profiling
  - Per-function elapsed time
- **Analysis (offline)**: 2. Analysis (offline)
  - P2.1: Diagnose causes of deviations
- **Visualization (offline)**: 3. Visualization (offline)
Problem 2.1: Diagnose causes of deviations
Deep nesting of stack obfuscates root causes
Problem 2.1: Diagnose causes of deviations

Deep nesting of stack obfuscates root causes
Problem 2.1: Diagnose causes of deviations
Deep nesting of stack obfuscates root causes
Problem 2.1: Diagnose causes of deviations
Deep nesting of stack obfuscates root causes
Problem 2.1: Diagnose causes of deviations
Deep nesting of stack obfuscates root causes
Problem 2.1: Diagnose causes of deviations
Deep nesting of stack obfuscates root causes

Identify root causes in nested stacks by removing redundant anomalies (see paper).
NSight: How to build a holistic latency diagnosis tool

1. Profiling (online)
   - CPU Hardware profiling
   - Application-network interface interposition
   - Per-function elapsed time
   - Per-message crossing times

2. Analysis (offline)
   - Detect anomalies
     - Typical lifetimes
     - Anomalous lifetimes
     - Network message lifetimes

3. Visualization (offline)
NSight: How to build a holistic latency diagnosis tool

1. Profiling (online)
- CPU Hardware profiling
- Application-network interface interposition

2. Analysis (offline)
- Detect anomalies
  - Typical lifetimes
  - Anomalous lifetimes

3. Visualization (offline)

Per-function elapsed time
Per-message crossing times

Network message lifetimes

Source of Deviation [(A) = App Core, (R) = Recv Core]
Total Deviation [us]
NSight shows what is delaying messages and how often.
NSight shows **what** is delaying messages and **how often**
NSight matryoshka plot: head-of-line blocking in memcached
NSight matryoshka plot: head-of-line blocking in memcached
NSight matryoshka plot: head-of-line blocking in memcached
NSight matryoshka plot: **head-of-line blocking in memcached**

Message timeline

- Kernel core: $T_N$
- Application core: $T_H$
- Application send: $T_A$

**Head-of-line blocking**
NSight shows **what** is delaying messages and **how often**
NSight shows *what* is delaying messages and *how often*
NSight matryoshka plot: memcached connection-set up inefficient

Message timeline
NSight shows *what* is delaying messages and *how often*
NSight guides system design

Message handling latency [us]

Fraction

Initial Configuration
Second Configuration
Third Configuration
NSight guides system design

Message handling latency [us]
NSight shows resource usage inefficiencies

Initial iteration

Third iteration
NSight shows resource usage inefficiencies

Initial iteration

Third iteration

Idle task
NSight: A tool that shows all sources of network latency in end-hosts all at once.

rniranjan@vmware.com

With special thanks to Jon Howell for helping us tell the story