The “Data uncertainty principle”

MEASUREMENTS, ANALYSIS AND AI FOR HIGH-SPEED NETWORKS

LEONARDO LINGUAGLOSSA – EC À TÉLECOM PARIS (IPP)
The ongoing process of softwarization

- Software-based networking tradeoff
- HW performance vs SW flexibility

Acceleration techniques can significantly reduce such gap: enabler for NFV
Network applications

- The Server owner rents her resources to Clients (e.g. service providers)
- Clients deploy their VNFs on the infrastructure
- VNFs are linked to provide Services (APPs) to users
- Service Level Agreement (SLA) regulates the clients/providers interactions
Behind the scenes

Applications use the low-level resources (CPU, NICs, RAM, …)
Need to **monitor the resource usage**
- For resource allocation
- For optimization of the infrastructure
Monitoring network traffic

**HW solutions**

- Complex
- Expensive

**SW solutions**

- Invasive (data alteration)
- Low accuracy (e.g., sampling, heavy hitters)
The **Data Uncertainty Principle**

**VNF**: Virtual Network Function

**Data tradeoff**:  
- impact on the system state  
- complexity VS cost  
- data availability

Limits which ML application could be deployed on SW routers
Thanks to the software nature of VNFs:
- The low-level CPU behavior reflects the high-level state of VNFs
- We can infer the current (and future) VNF’s state
- No need for complex monitoring infrastructure

*Low-level entities* | *High-level entities*
--- | ---
Easy to *Monitor*  
Hard to *Interpret*  
Data availability: *Huge*  | Hard to *Monitor*  
Easy to *Interpret*  
Data availability: *Low-to-medium*
1) Background on the *Data uncertainty principle*

2) Analysis and evaluation of high-speed measurement techniques

3) The case for AI in high-speed network contexts
NFV and software routers

Commodity server

NIC

Wire + transceiver

SW router
# Acceleration techniques for high-speed SW networks

## Part 2 – High speed NFV

<table>
<thead>
<tr>
<th></th>
<th>Poll</th>
<th>I/O Batch</th>
<th>Memory</th>
<th>Compute Batch</th>
<th>Threading</th>
<th>Coding</th>
<th>NIC-support</th>
<th>CPU-support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduce memory access</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Optimize memory allocation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Share overhead of processing</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Reduce interrupt pressure</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Horizontal scaling</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Exploit CPU cache locality</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Reduce CPU context switches</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Fill CPU pipeline</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Exploit HW computation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Simplify thread scheduling</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

High-speed packet processing in software

Packet-processing:
- no individual packet arrivals
- per-batch processing
- no individual packet departure
- Optimizing CPU usage
Batching and polling

1. Receive packets

SW router

Batch

Network Interface Card (NIC)

3. forward the processed packets
Input traffic and batch sizes

- RX and TX are done by the NIC → Focus on the CPU only
- Batch sizes depend on the input traffic
  - Low rate → small batches
  - High rate → big batches

Batch sizes depend on the input traffic.

- Low rate → small batches
- High rate → big batches
Translation into low-level instructions

**Program**

While(true):

  batch = getPkts(NIC)

  if (size(batch) > 0):
    doProcessing(batch)

  continue

**Assembler (ASM)**

getPkts:

  INSTR_1
  INSTR_2
  ...
  INSTR_n

doProcessing:

  INSTR_1
  ...
  INSTR_m
Depending on traffic and application characteristics, the CPU will show different patterns.
Methodology and experimental setup

- SW router executing a simple VNF
- Single CPU
  - both I/O and compute
- Input rate \( \in [0, 10] \text{ Gbps} \)
- Traffic pattern \( \in \{\text{Poisson, CBR, IMIX}\} \)

- **Perf tool**: capture CPU data
  - Sampling rate \( \in \{0.1, 1, 5\} \text{ s} \)

- **Data analysis**
  - Finding CPU/network correlation

- **Applications**
  - E.g., using the correlation to infer the VNF’s state
  - (More in the following)
The 4 components of the CPU behavior

- **Instructions and branches**
  - Reflect the complexity of the VNF code
  - Strongly correlate with input conditions
  - Application may affect the CPU efficiency

- **CPU caches**
  - Reflect the data (and instruction) similarity of input traffic
  - Correlate with spatial/temporal locality

- **Memory accesses**
  - Reflect the access to large data structures
  - E.g., IPv4 lookups, ACLs, …
  - Memory pattern may give insight on the computation performed by the VNF

- **Bus and storage**
  - Not really used in high-speed context
  - Usually the storage is not accessed for network-intensive apps
- We have different scenarios and combination of traffic
  - E.g., “IPv4 Poisson traffic of 60-byte packets at 5Gbps”
- We collect $m$ CPU measurements grouped into a $n$-dimensional vector $V$
  $V^1, \ldots, V^m$, where $V^i = \{\text{#instructions}, \text{#branches}, \text{#cache-hits}, \ldots\}$ and $|V^i| = n$
- We define a representative vector as the vector consisting of the average values
- Upon a new measurement: cosine similarity to compare w.r.t. each previous scenario
Accuracy/performance tradeoff

Assume a NN model
N hidden layers

Alternatives:
- Use simpler models
- Avoid per-packet operations
Conclusions

- Network softwarization: decoupling equipment from their function

- Complex measurement infrastructure can affect the performance

- Tackle the data uncertainty principle with our novel methodology
  
  - Analysis of the CPU behavior for different use cases
  - Correlation with several KPI, e.g. input rate, packet loss, …
  - Methodology successfully applied on different scenarios
Future work

- Focus on more complex scenarios
  - Multiple VNFs
  - Multicore

- Increase the number of applications
  - IPv4
  - Cryptographic

- From inference to prediction
  - Machine learning for predicting the future state of VNFs
Questions?

Contact:

linguaglossa@telecom-paris.fr
CPU behavior w.r.t. the input rate

- Computation features for *fastclick* executing a L2-fwd VNF
- We observe the polling/processing state dichotomy
- The number of misses reflects the code unpredictability → depends on the input rate
- More details on our ITC 33 Paper