Shiyi Cao, Salvatore Di Girolamo, Torsten Hoefler

# Accelerating Data Serialization/Deserialization Protocols with In-Network Compute

# Content

- **Introduction**

- **Background**
  - Data (De)serialization
  - In-network Compute

- **Network Accelerated SerDes**
  - Design
  - Evaluation

# Content

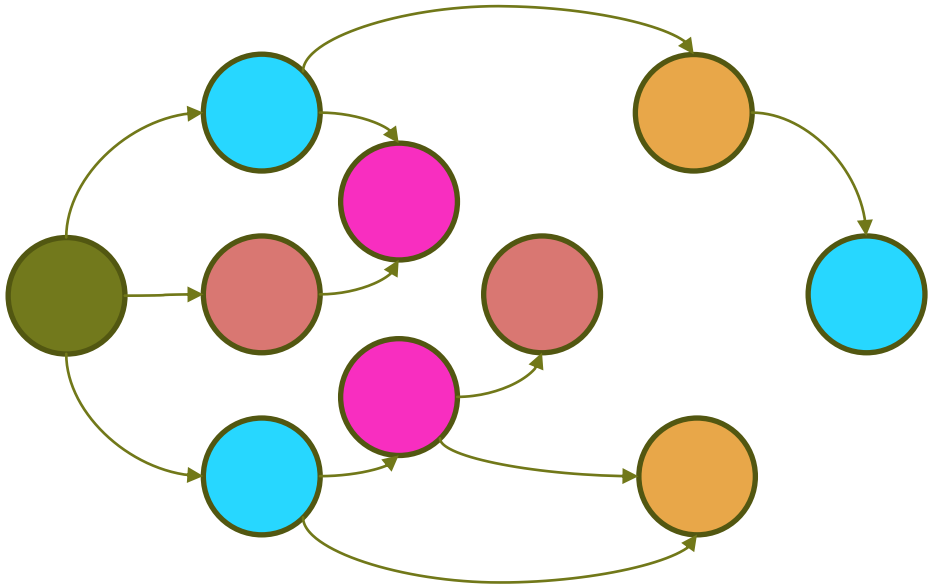
- **Introduction**

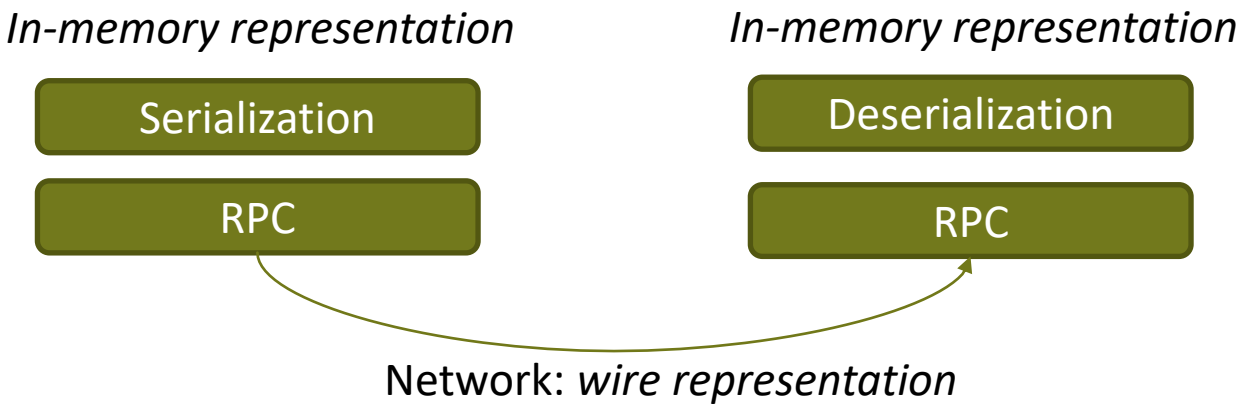
- **Background**
  - Data (De)serialization
  - In-network Compute

- **Network Accelerated SerDes**
  - Design
  - Evaluation

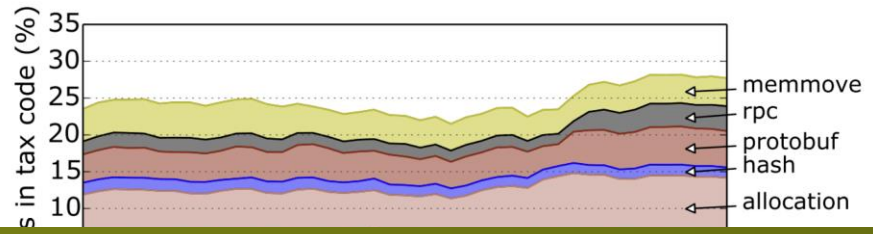
# Data SerDes in distributed applications

- **Microservices**
  - Exchange data in language- and architecture-independent manner (e.g., Protocol Buffers, Apache Thrift)





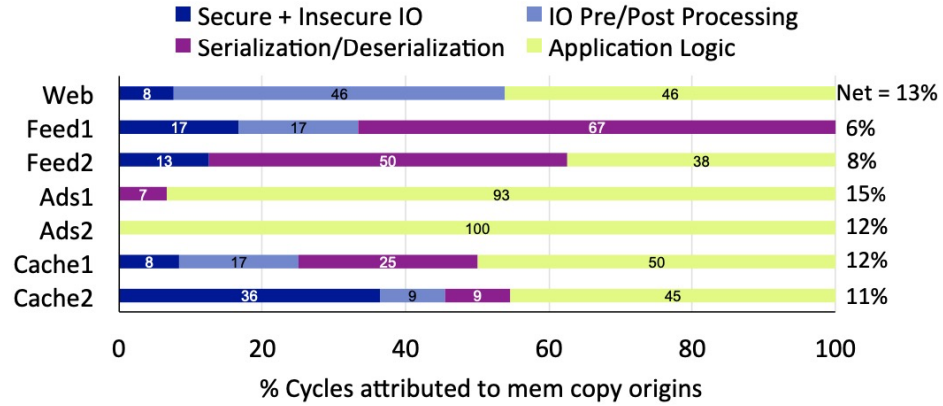
# Performance Bottleneck



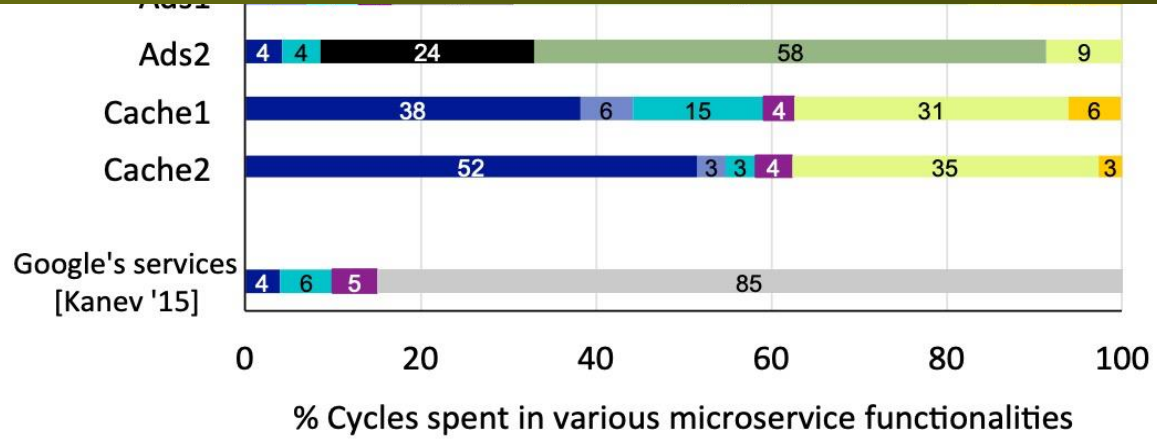


**Need a way to free up CPU cycles**

1. https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/44271.pdf (ISCA'15)



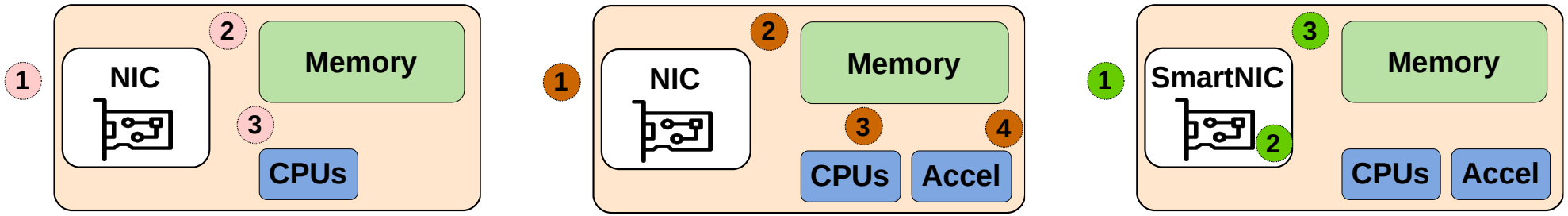

2. https://dl.acm.org/doi/pdf/10.1145/3373376.3378450 (ASPLOS'20)




2. https://dl.acm.org/doi/pdf/10.1145/3373376.3378450 (ASPLOS'20)

# In-Net SerDes

- **Contributions**
  - **Fully expressed in software**
    - Transferring and manipulating complex pointer-based data structures between different memory spaces
    - Keep original semantics of the data SerDes protocols
  - **Free up CPU cycles**
    - 4.8x deserialization throughput than a single CPU
    - Up to 60% E2E throughput improvement
  - **SmartNIC specialties**




CPU-based | Near-core Accelerator | Our approach

# Content

- Introduction
- **Background**
  - Data (De)serialization
  - In-network Compute
- **Network Accelerated SerDes**
  - Design
  - Evaluation

# Protobuf Example

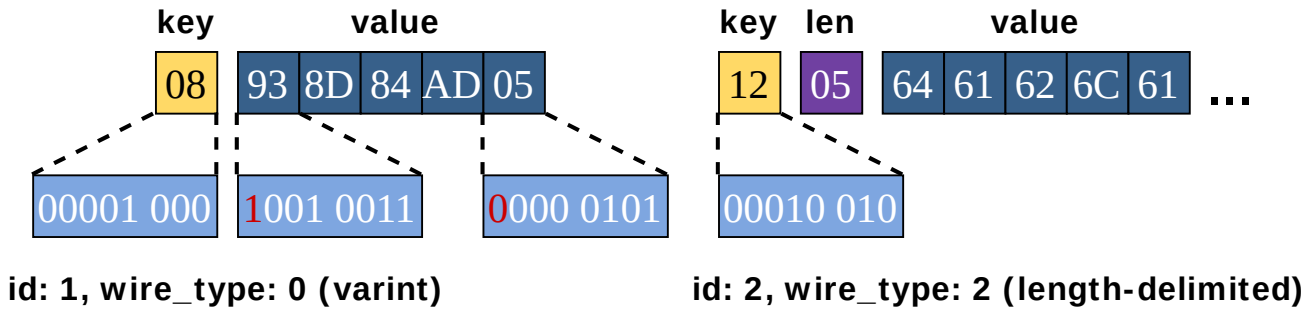
- **Message Definitions:**
  - Field Type (data_type): uint64_t, bool, string, …
  - Field Name
  - Field Number (field_number)
  - Field Rule (Optional, Repeated, …)

```
message Student {
  uint64_t id = 1;
  string name = 2;
  repeated Advisor advisors = 3;
}

message Advisor {
  uint64_t id = 1;
  string name = 2;
}
```

# Protobuf Example

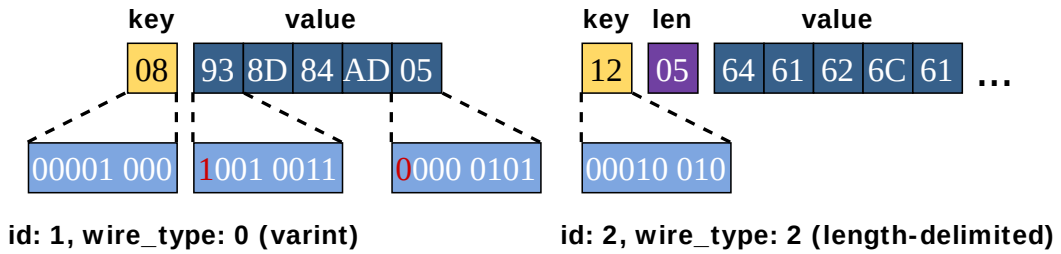
- **Key-Value pairs:**
  - [field_id << 3 | wire_type] as key
- **Wire Types:**
  - Varint (uint32/64, bool): ***chain-style encoding***, using msb of each byte to indicate if there are more bytes
  - Length-delimited (string, bytes, embedded messages)
  - Fixed-length (fixed64, double)



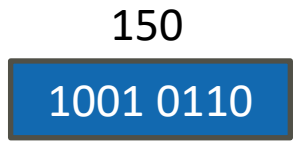
# Inefficient processing in general-purpose CPUs

- **Varint Encoding**
- **Sequential Processing**
- **Excessive *mem_alloc***

```
unsigned max_rv = len > 5 ? 5 : len;
for (rv = 1; rv < max_rv; rv++) {
  if (data[rv] & 0x80) {
    tag |= (data[rv] & 0x7f) << shift;
    shift += 7;
  } else {
    tag |= data[rv] << shift;
    *tag_out = tag;
    return rv + 1;
  }
}
```





# Content

- Introduction
- **Background**
  - Data (De)serialization
  - In-network Compute
- Network Accelerated SerDes
  - Design
  - Evaluation

# Network Acceleration: SmartNIC Specialties

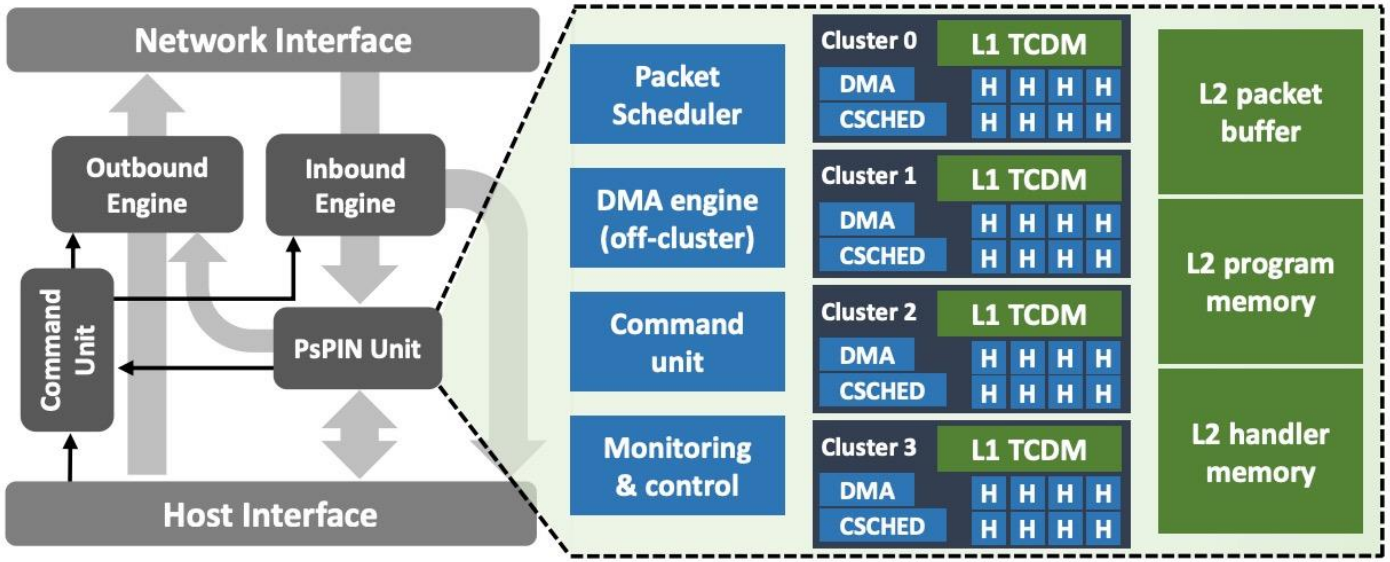
- **User-level**
  - Without violating isolation principles
  - Not satisfied by many in-network compute solutions like DPDK

- **Flexibility and High Programmability**
  - Many in-network compute solutions impose constraints on the code

# Network Acceleration

- **sPIN**
  - Packet Handlers
  - Execution Context

- **PsPIN**
  - 32 RISC-V cores (4 clusters x 8 cores/cluster)
  - 1MiB L1 scratchpad memory for each cluster
  - 4MiB slower off-cluster memory

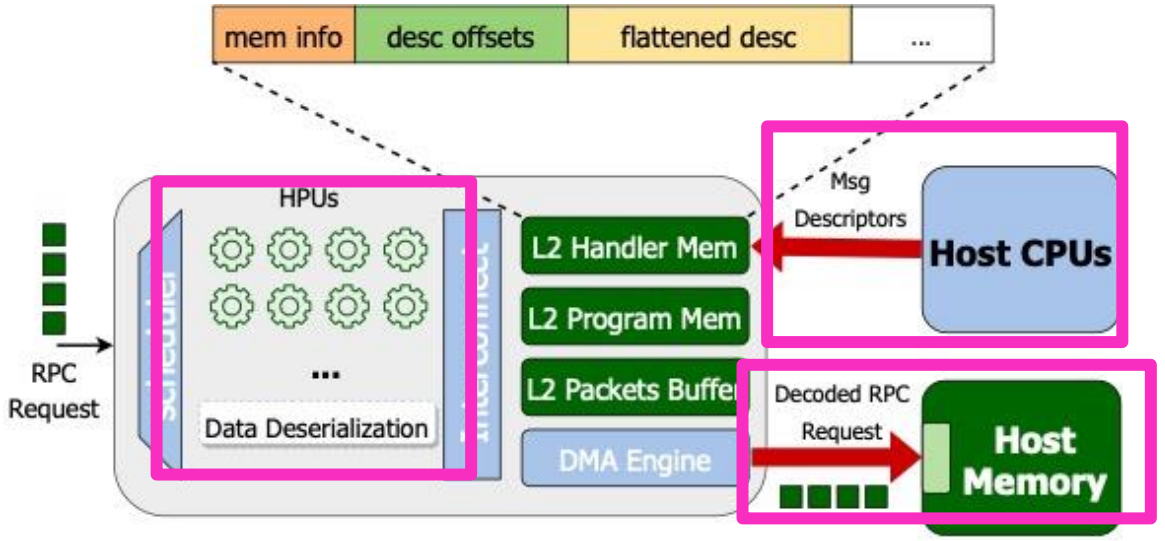


# Content

- Introduction
- Background
  - Data (De)serialization
  - In-network Compute
- **Network Accelerated SerDes**
  - Design
  - Evaluation

# Offloading Overview

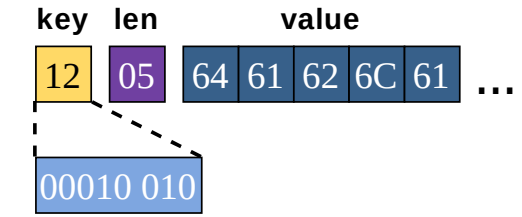
- **Workflow**
  - Descriptors Installing
  - Deserialization according to message type



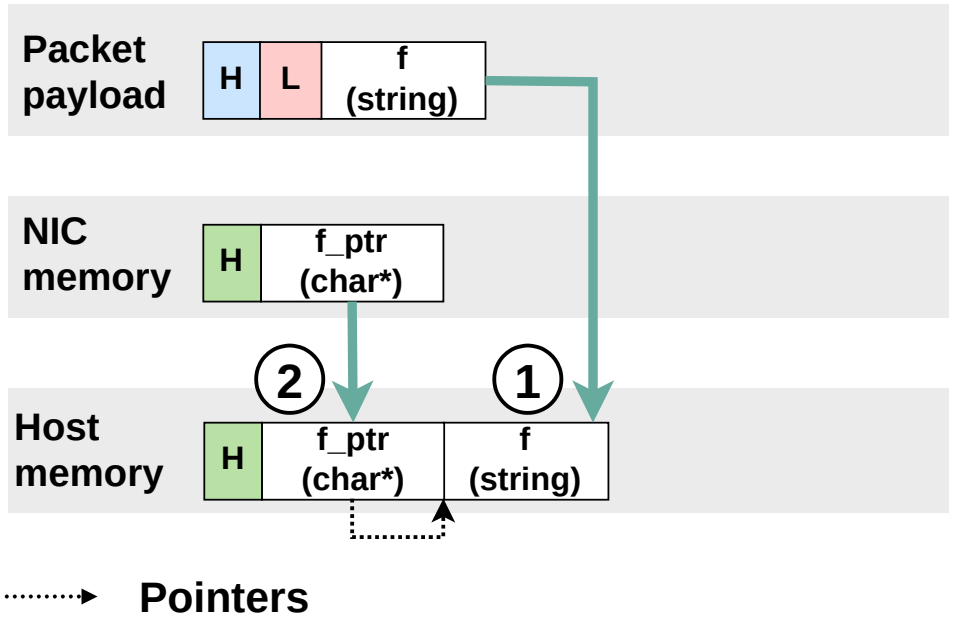

- **Memory Management**
  - Host Memory
    - *Disjoint regions in pinned memory are assigned to each cluster*
  - NIC Memory
    - *Some reserved space for shared information within a cluster*
    - *Disjoint partitions of remaining L1 scratchpad memory are assigned to each HPU*

# Decoding on the NIC (some examples)

- **Messages containing *String* field**




id: 2, wire_type: 2 (length-delimited)




.proto

```
message M {
  string f = 1;
}
```

H — Field Header

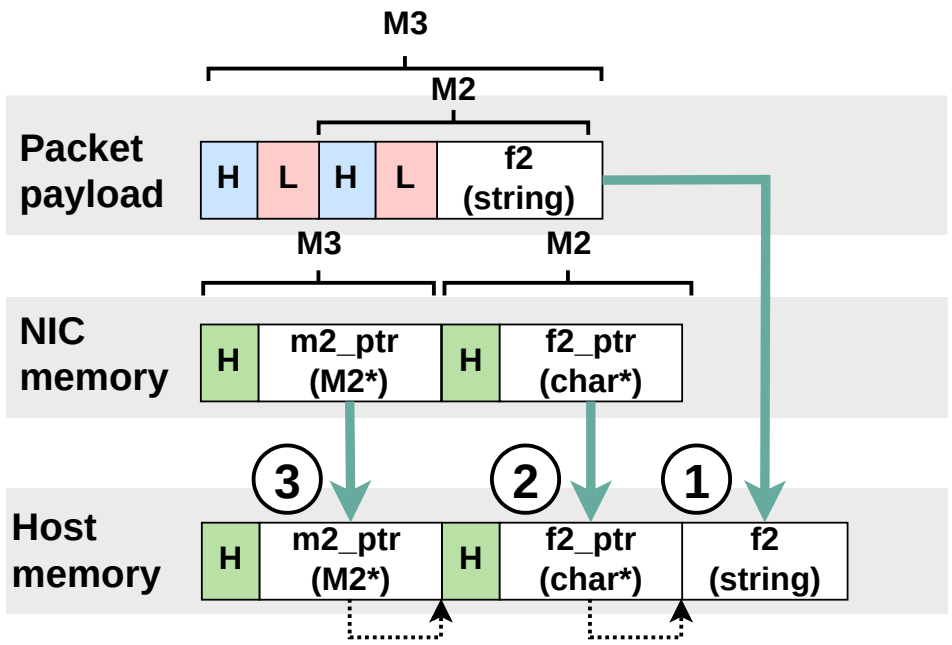
L — Field Length (varint)

H — Message Header

DMA to Host

# Decoding on the NIC (some examples)

- Messages containing *Embedded* field



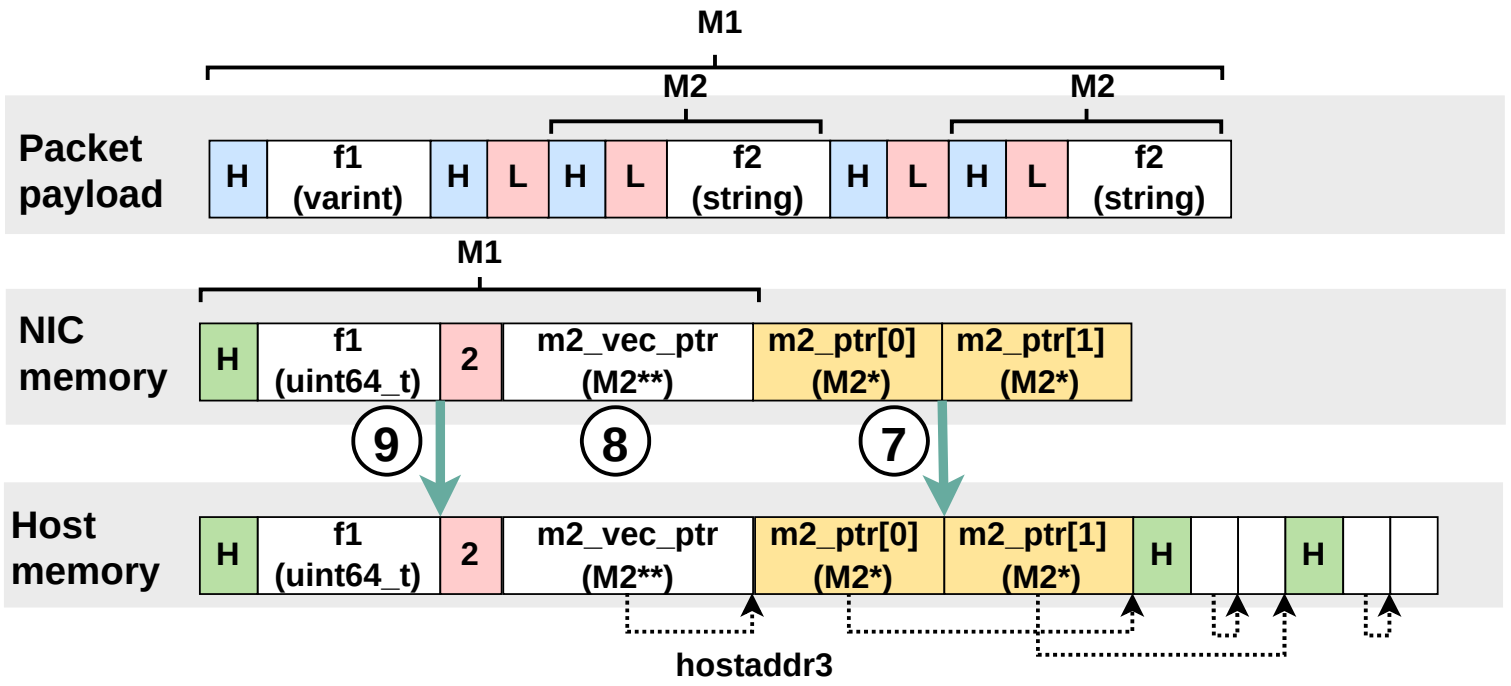

.proto

```
message M2 {
  string f2 = 1;
}

message M3 {
  M2 m2 = 1;
}
```

# Decoding on the NIC (some examples)

- Complex Messages (repeated embedded messages)




.proto

```
message M1 {
  uint64_t f1 = 1;
  repeated M2 m2 = 2;
}

message M2 {
  string f2 = 1;
}

message M3 {
  M2 m2 = 1;
}
```

Pointers

# Content

- Introduction
- Background
  - Data (De)serialization
  - In-network Compute
- **Network Accelerated SerDes**
  - Design
  - Evaluation

# Evaluation

- **Microservice Benchmark (DeathStarBench)**
  - User Service, Media Service, Post Service
- **Varying factors in messages of different services:**
  - Number and composition of fields
  - Depth of the nested message
  - Whether containing long string/bytes

TABLE I
STATISTICS FOR REQUESTS

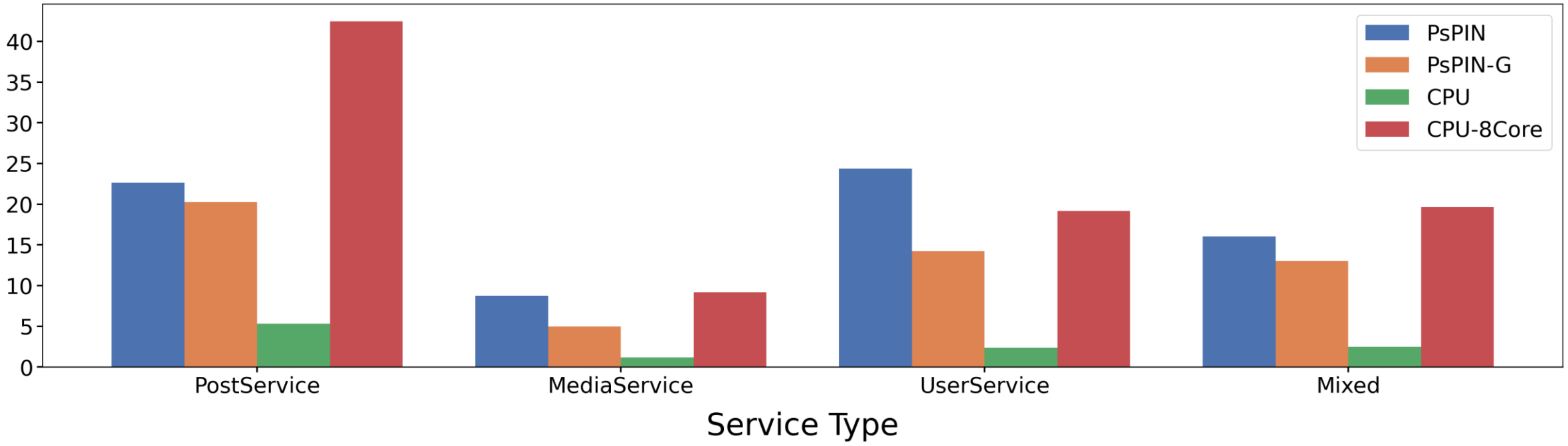
| Type | Instructions | Loads | Stores | Branches | L2 Accesses | L1 Accesses | Avg. Msg. Size (B) |
|---|---|---|---|---|---|---|---|
| StorePost | 3962 | 788 | 671 | 671 | 11 | 1279 | 427 |
| ComposePost | 1409 | 272 | 220 | 264 | 4 | 421 | 252 |
| ComposeMedia | 1615 | 315 | 248 | 295 | 5 | 497 | 76 |
| RegisterUser | 1008 | 182 | 167 | 196 | 1 | 281 | 103 |
| Login | 712 | 128 | 103 | 146 | 1 | 189 | 74 |

# Evaluation

- **Setting**
  - PsPIN: cycle-accurate simulation, 400Gbit/s network
  - CPU: AMD Ryzen 7 5700G CPUs (256KiB L1d and L1i cache, 4MiB L2 cache, and 16MiB L3 cache)

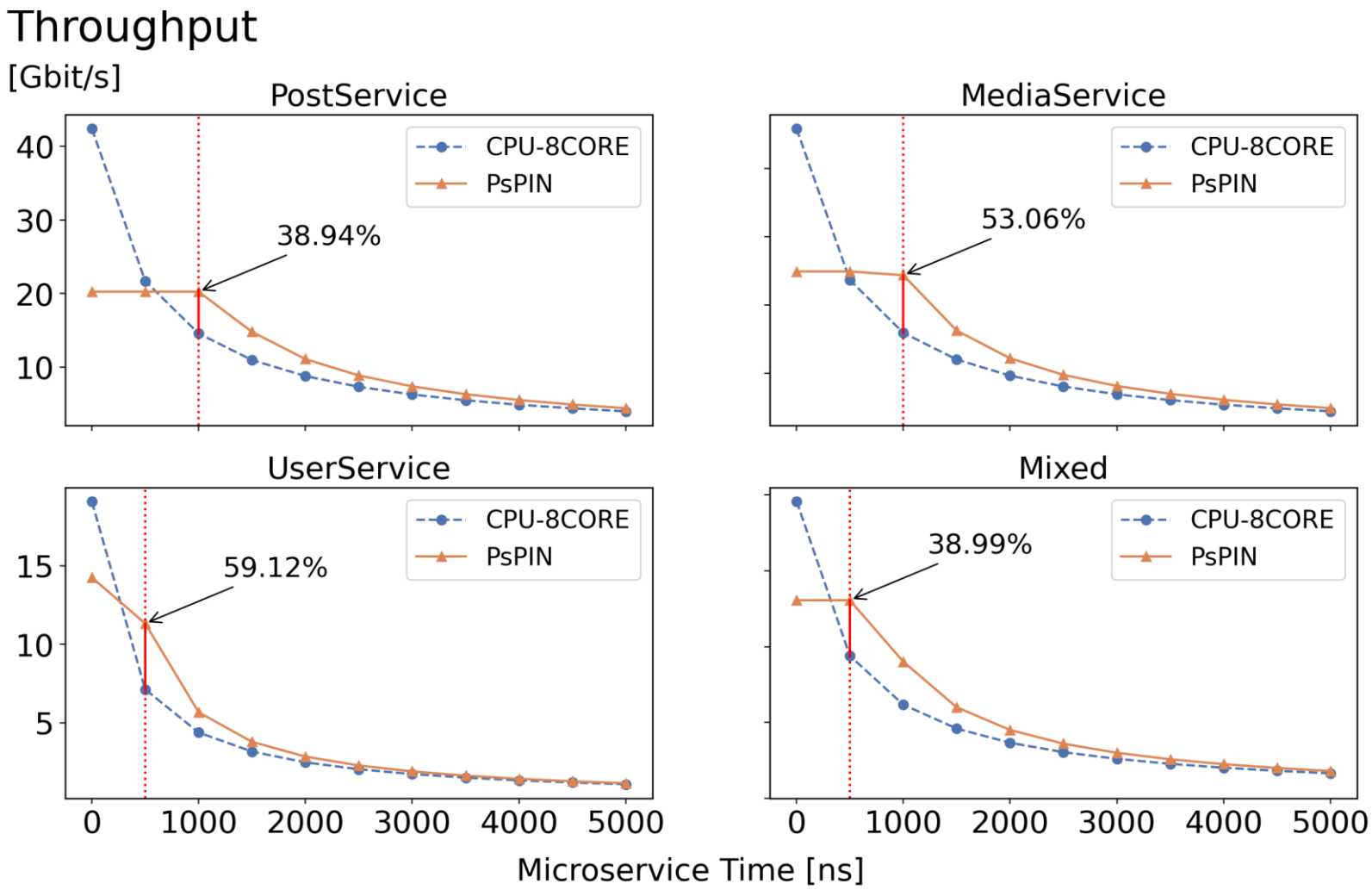
- **Workload Generation**
  - Single Service Mode
  - Mixed Service Mode

**Throughput**
[Gbit/s]




PsPIN-G achieves on average 4.8x higher throughput compared with a single CPU

# Microservice E2E Throughput Modeling



# Conclusion

- **Fully expressed in software**
  - Addresses the challenge of transferring and manipulating complex pointer-based data structures between different memory spaces
  - Keeps the original protocols' semantics
- **SmartNIC Specialties**
  - User-level
  - Flexibility and Programmability
- **Free up CPU cycles**
  - Up to 60% throughput improvement by pipelining data deserialization and actual application tasks