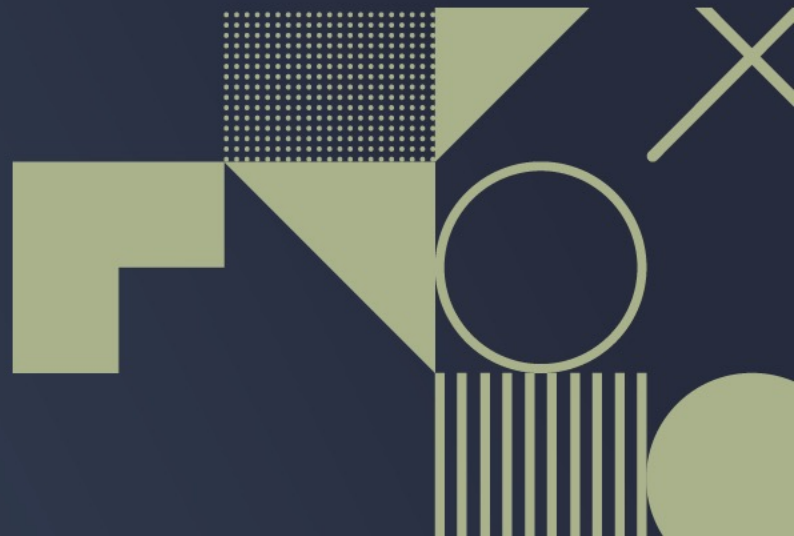
HatRPC: Hint-Accelerated Thrift RPC over RDMA

Tianxi Li (li.9443@buckeyemail.osu.edu),

Haiyang Shi (shi.876@buckeyemail.osu.edu) *and*

Xiaoyi Lu (xiaoyi.lu@ucmerced.edu)

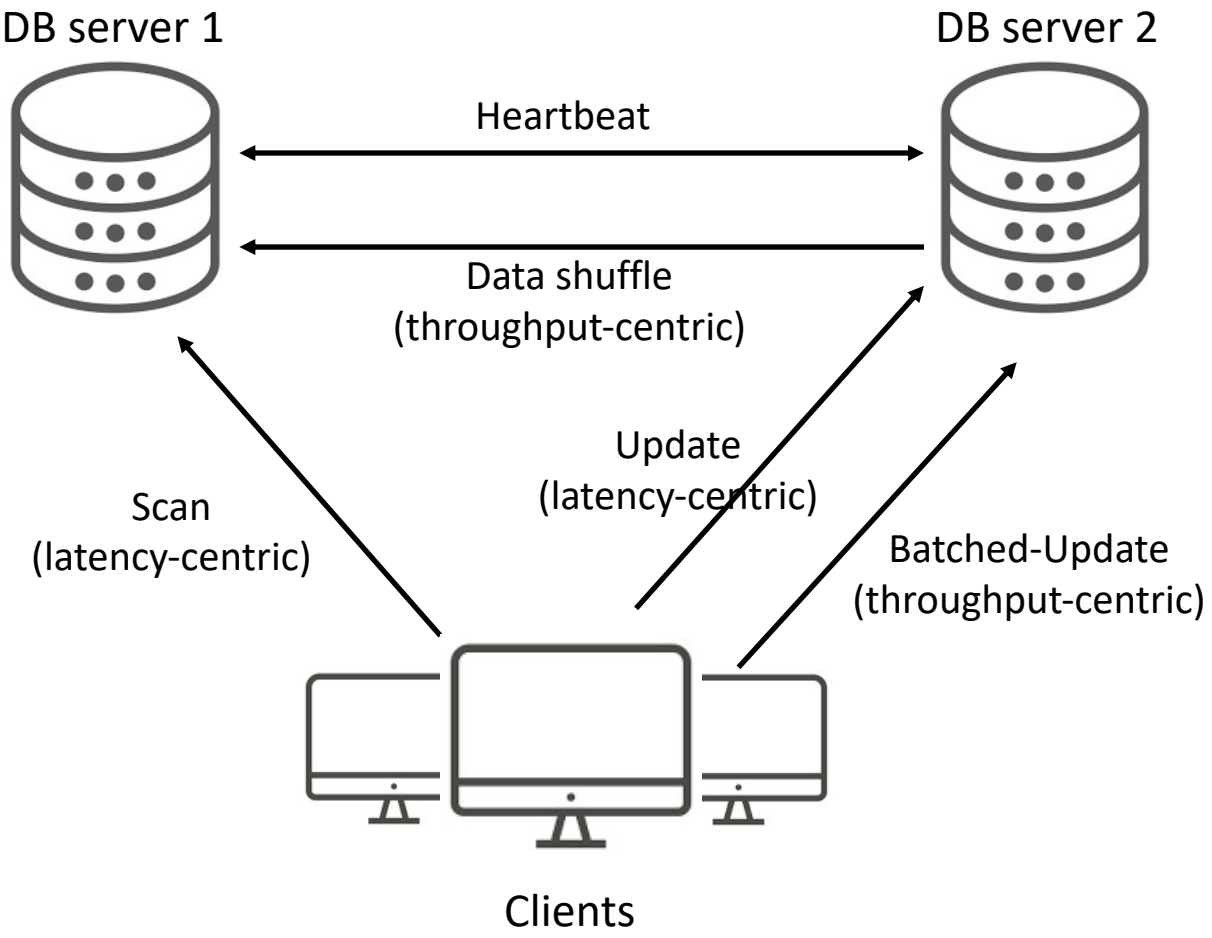


Overview

- Introduction
- Motivation and Problem Statements
- HatRPC Design
- Evaluation
- Conclusion and Future Work
- Acknowledgement

Communication Challenges in Applications

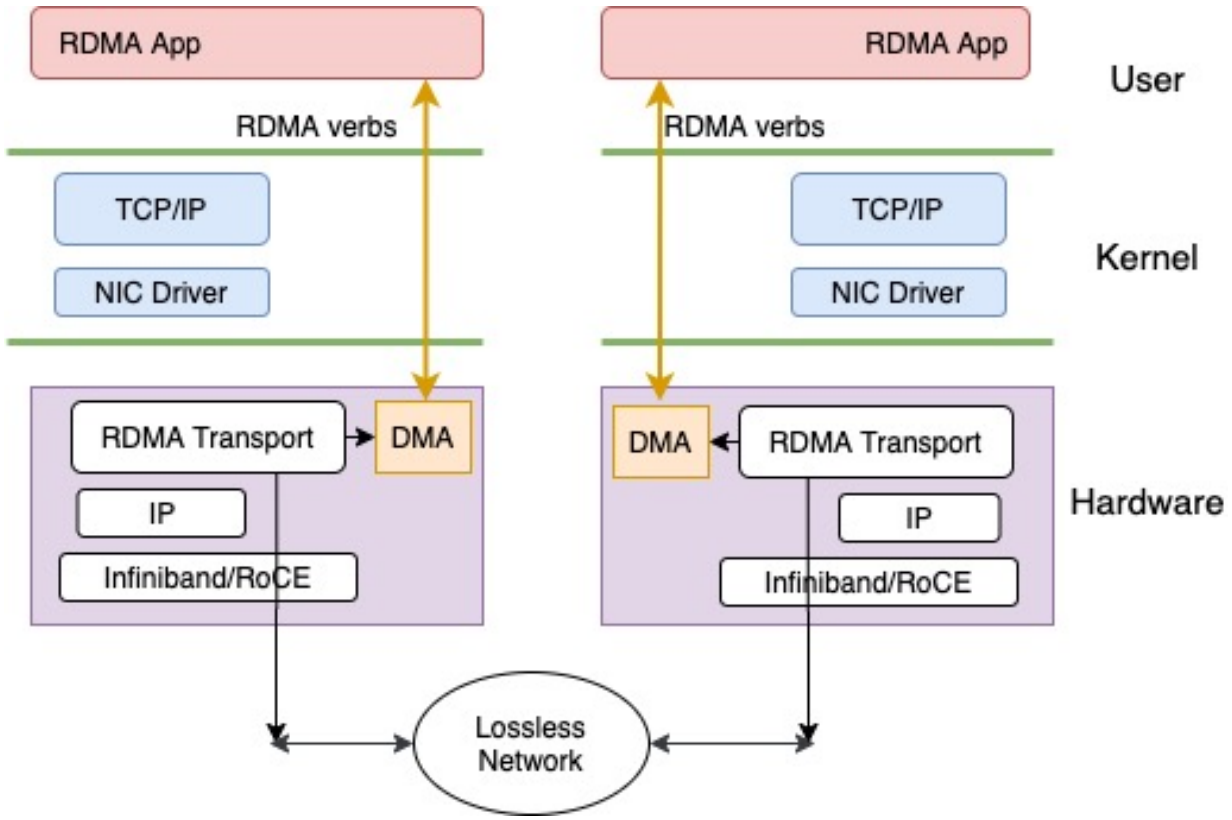
- Modern datacenters and clouds have many heterogeneous applications running simultaneously
- Different services and functions within the same server have different communication characteristics, communication patterns, and requirements



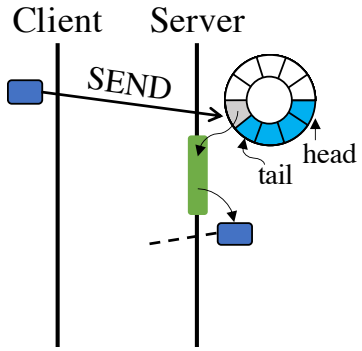
Database with Heterogeneous Applications

Remote Direct Memory Access (RDMA)

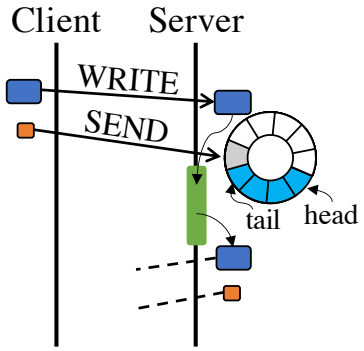
- Remote Direct Memory Access (RDMA) can bypass CPU in transferring data across network
- By reducing CPU involvement, RDMA delivers excellent performance in latency, bandwidth etc
- RDMA programming is not easy with verbs.
  - ~600 LOC for verbs
  - ~500 LOC for UCX



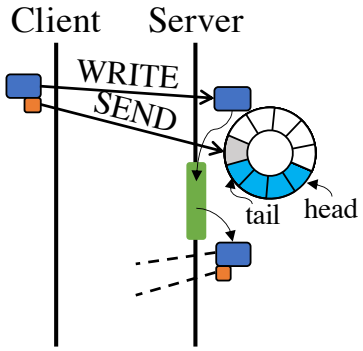
RDMA Communication Protocols



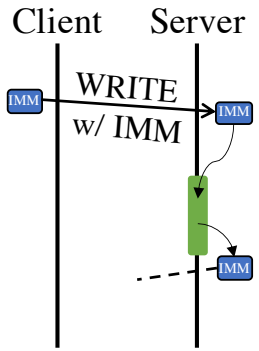
Eager-SendRecv



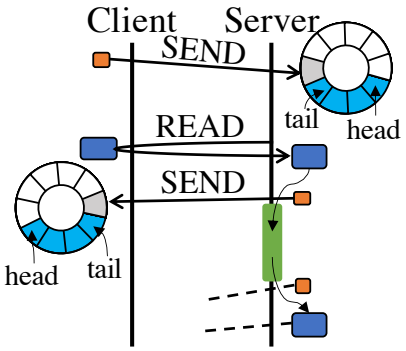
Direct-Write-Send



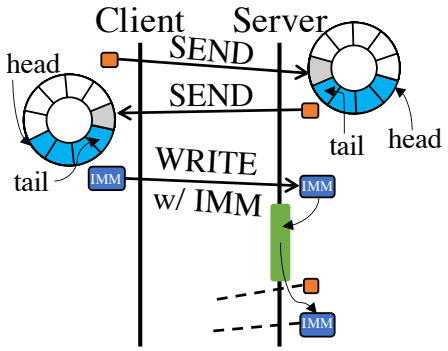
Chained-Write-Send



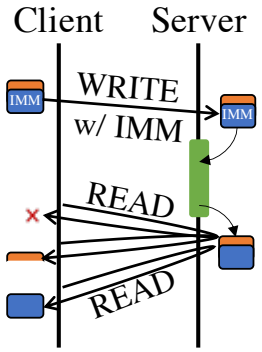
Direct-WriteIMM



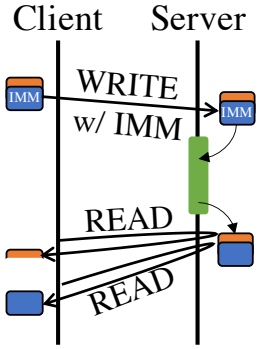
Read-RNDV



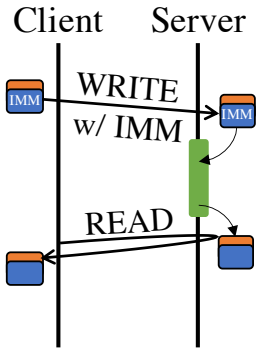
Write-RNDV



Pilaf



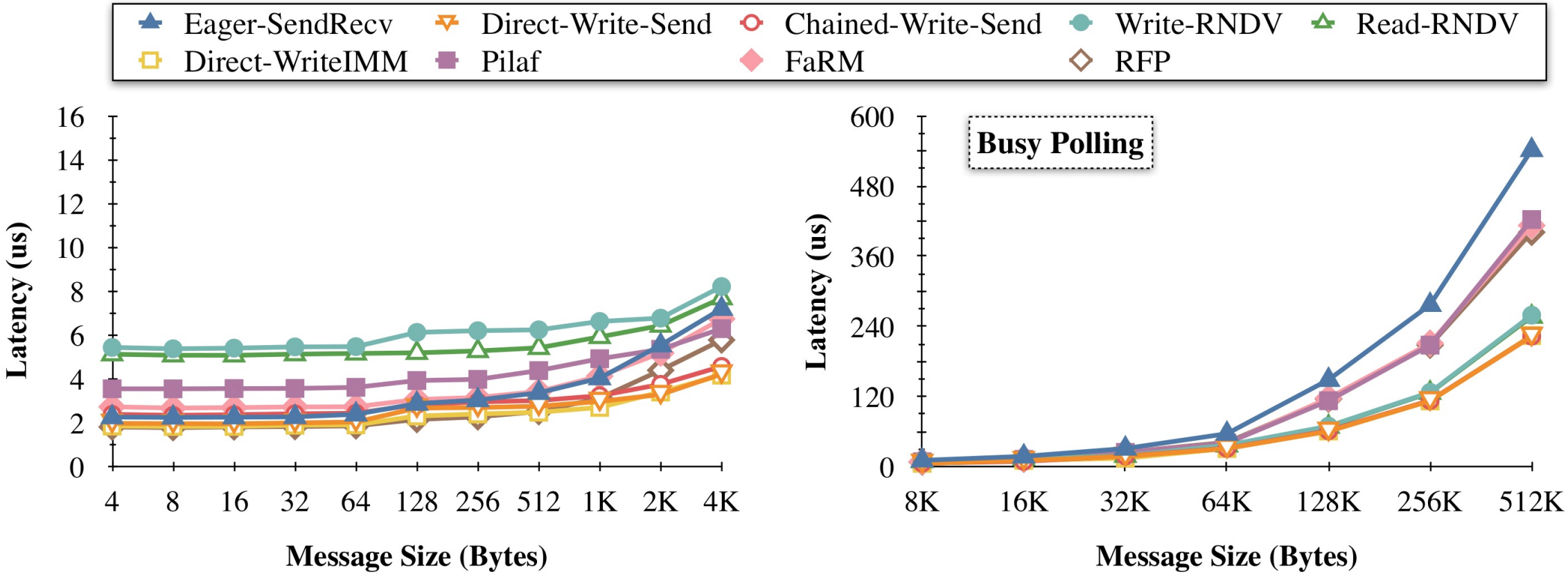
FaRM



RFP (Best Case)

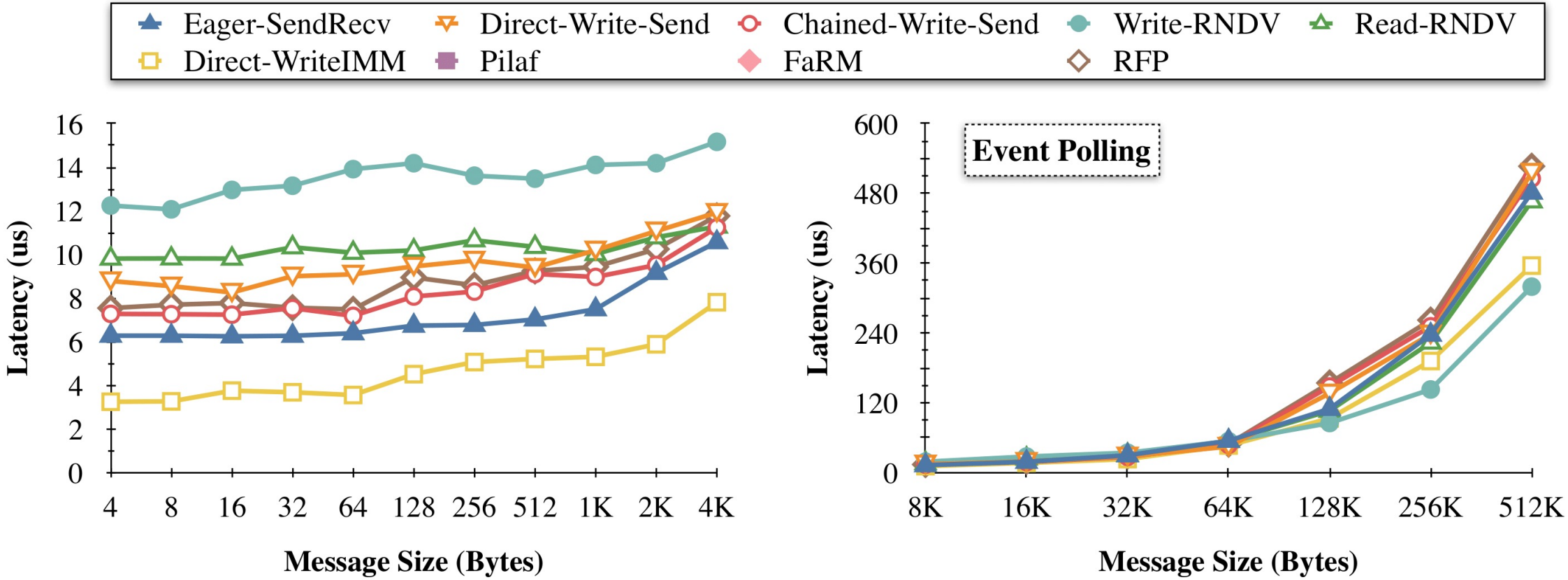
- Various RDMA communication protocols from previous works

RDMA Protocols in RPC - Latency



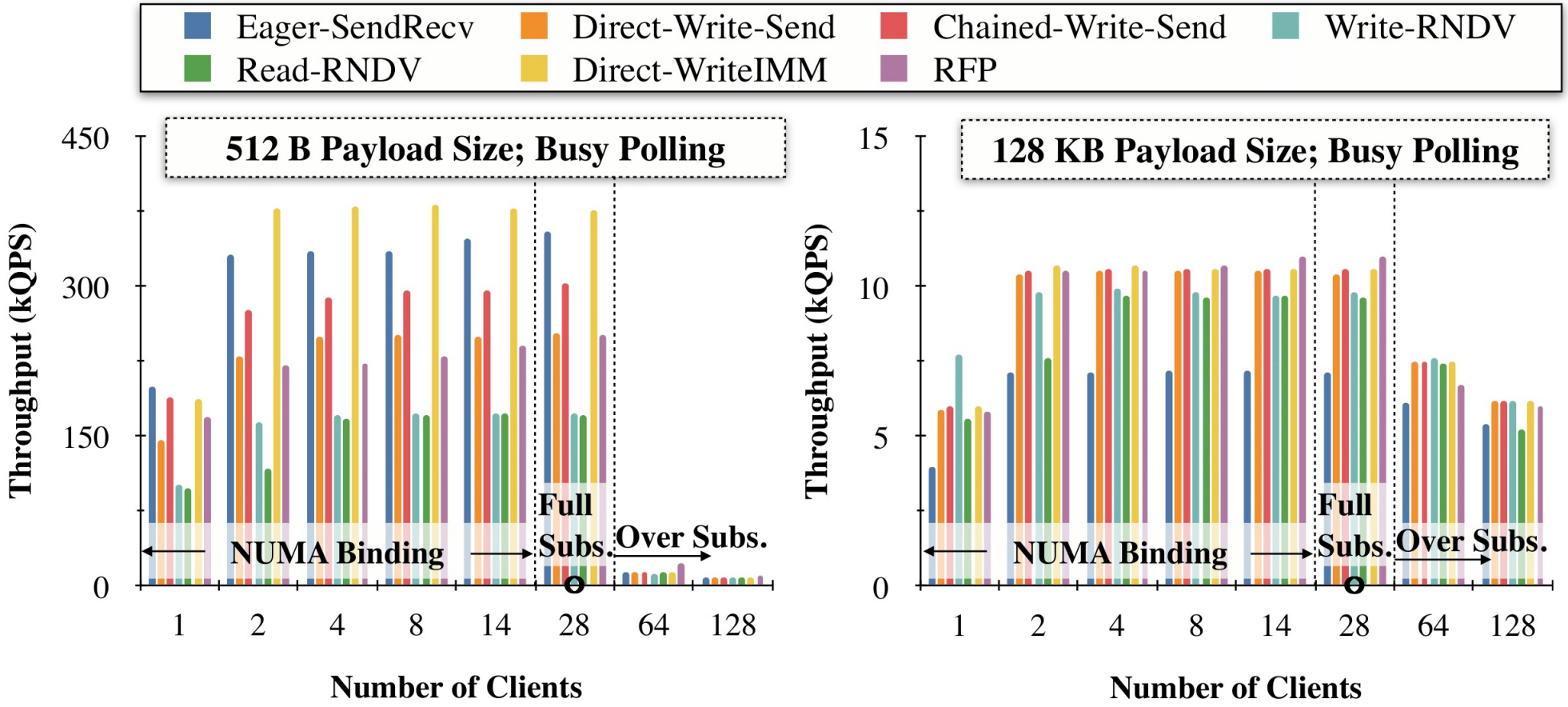
- Direct-WriteIMM provides the best performance in busy polling

RDMA Protocols in RPC - Latency



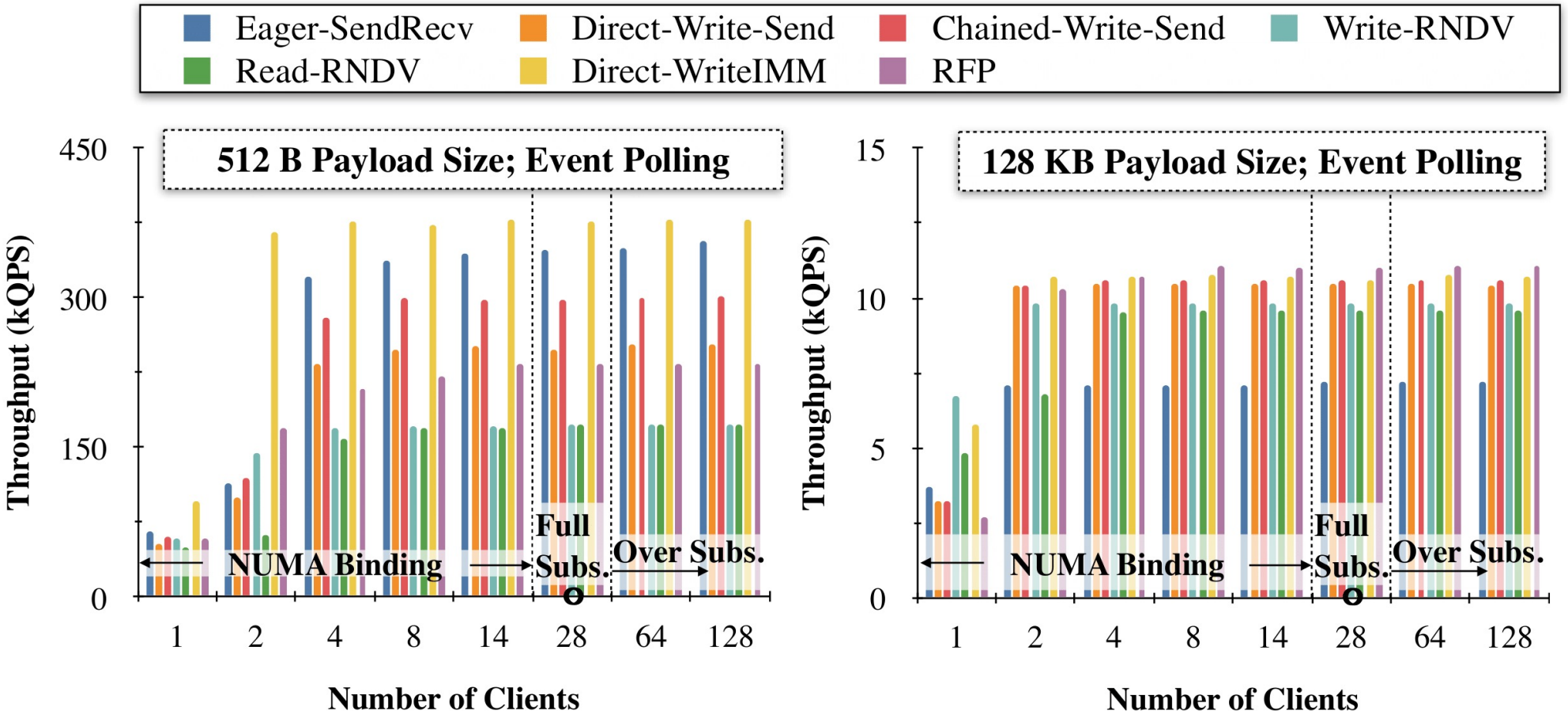
- Compared with busy polling results, event polling is not suitable for latency critical applications

RDMA Protocols in RPC - Throughput



- When used for small payload size and under subscription, Direct-WriteIMM has good throughput in busy polling

RDMA Protocols in RPC - Throughput



- Direct-WriteIMM with event polling is suitable for small payloads
- RFP with event polling is suitable for full- and over-subscription for large payloads

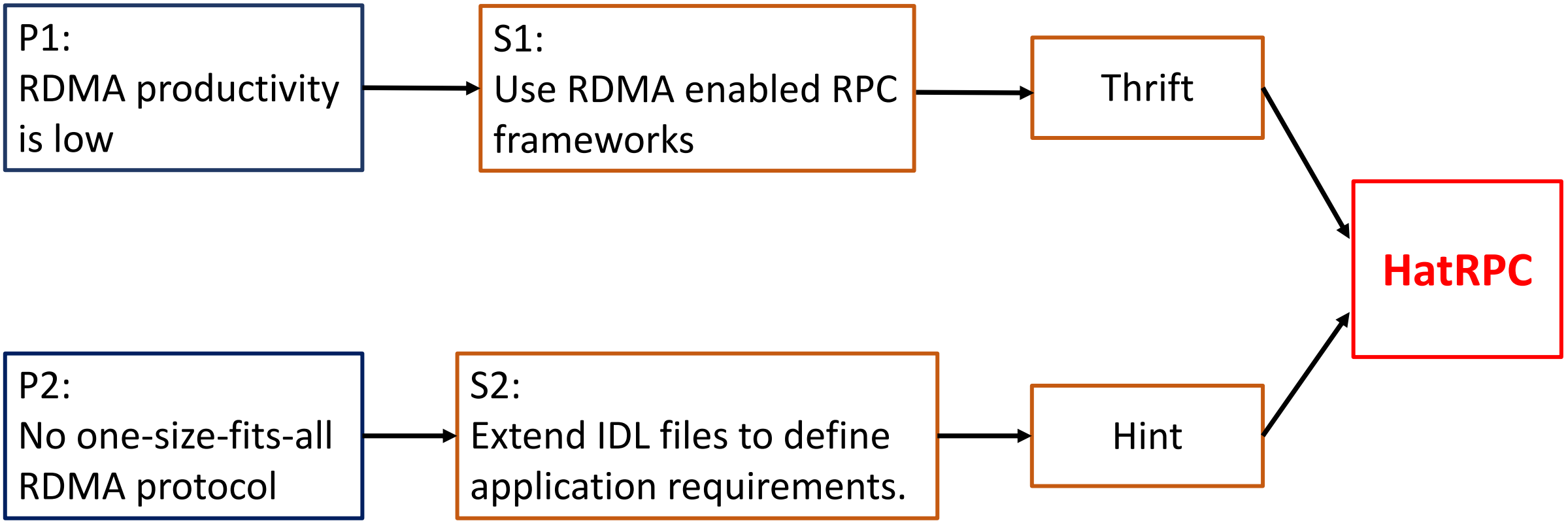
Problem Statements

- **RDMA productivity is low**
  - Can we design an approach to automatically generate efficient RDMA-based communication substrates for data center applications?
- **No one-size-fits-all RDMA protocol**
  - How can the proposed approach satisfy different communication requirements on various RDMA protocols in datacenter applications?
  - How can we guarantee the effectiveness and efficiency of the generated RDMA-based communication protocols for heterogeneous applications?

Overview

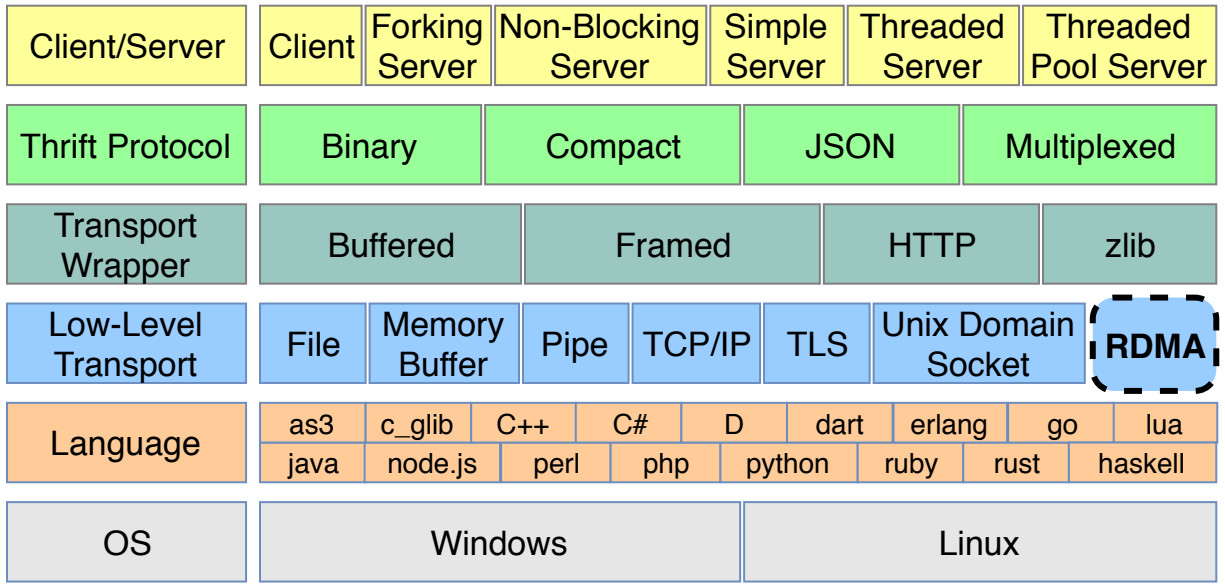
- Introduction
- Motivation and Problem Statements
- HatRPC Design
- Evaluation
- Conclusion and Future Work
- Acknowledgement

Key Ideas of HatRPC



Overview of RPC Framework – Apache Thrift

- A communication framework that hides platform and hardware details for users
- Typically provide user-friendly Interface Definition Language for different services and applications
- Apache Thrift is a widely used RPC framework that adopts a hierarchical architecture and provides an IDL code generator

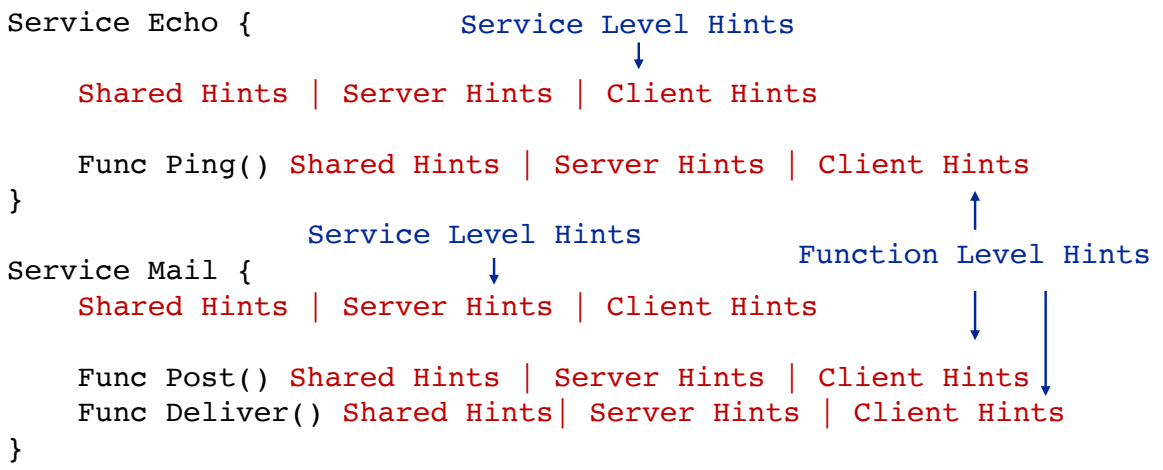


Apache Thrift Architecture

Proposed Hints in Thrift IDL

```
Service      ::= 'service' Identifier ( 'extends' Identifier )?
                         '{' HintGroup* Function* '}'
Function     ::= 'oneway'? FunctionType Identifier '(' Field* ')'
                         Throws? ListSeparator? FunctionHint?
FunctionHint ::= '[' HintGroup* ']'
HintGroup    ::= 'hint' ':' HintList ';'
                         | 'c_hint' ':' HintList ';'
                         | 's_hint' ':' HintList ';'
HintList     ::= Hint ',' HintList | Hint
Hint         ::= key '=' value
```

HatRPC IDL Abstract Syntax Structure

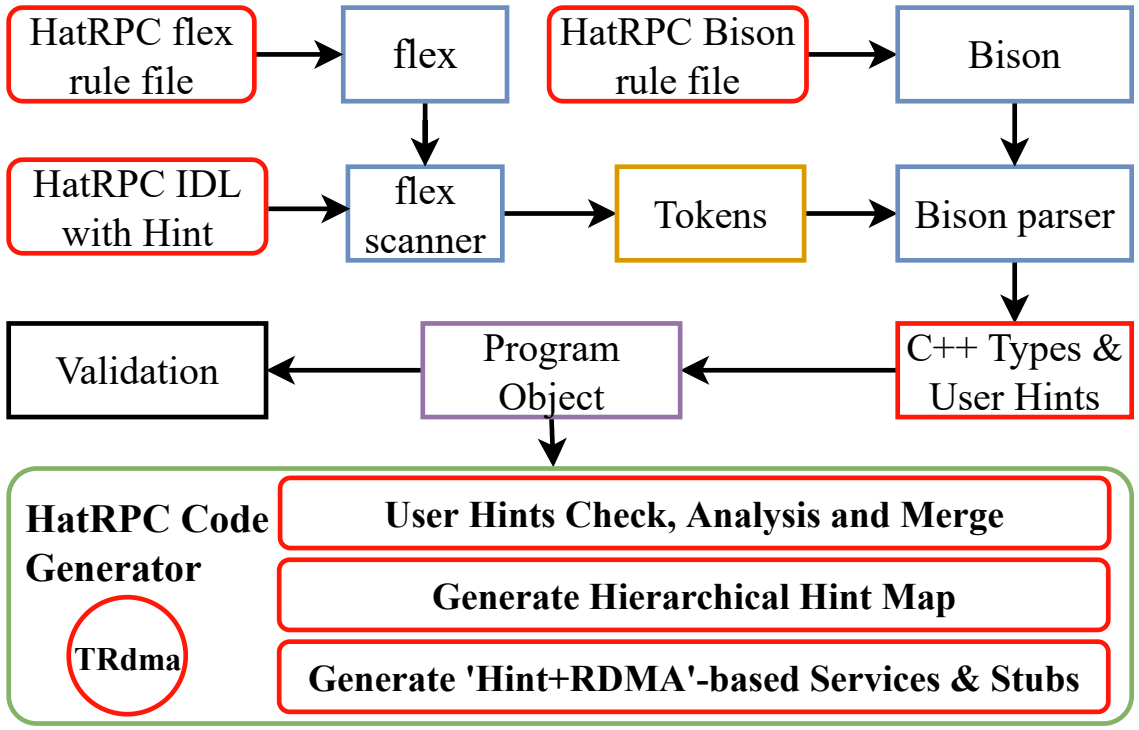


HatRPC Hint Hierarchy

- Extend from Thrift's IDL syntax and allow users to insert key-value pairs as hints in IDL files
- Adopt a hierarchical architecture to specify hints in different services or functions. Use lateral partitioning to differentiate the client and server side

Code Generation

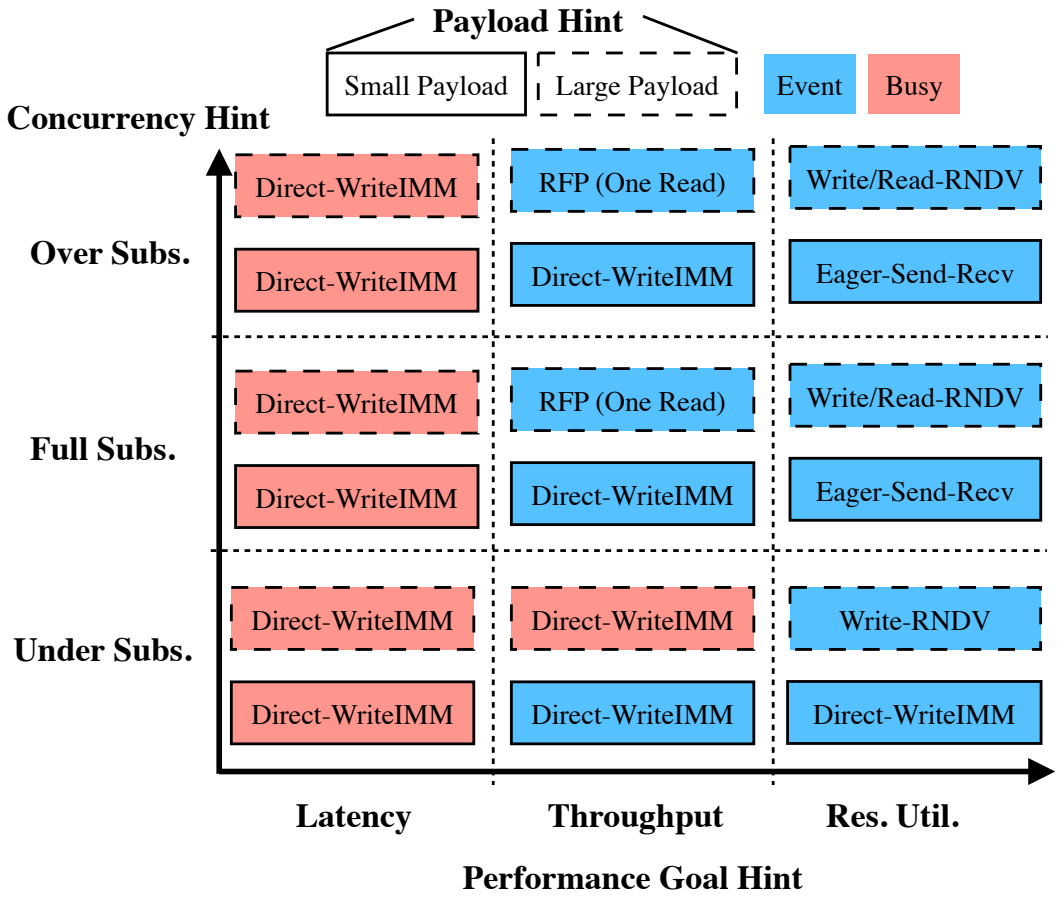
- Use flex to generate lexical analyser and Bison to generate parser for code generation
- Check hint validity and optimize hint map layout. Generate RDMA-based service and stub files with hints



HatRPC Code Generation

Hint Design Space and Decision

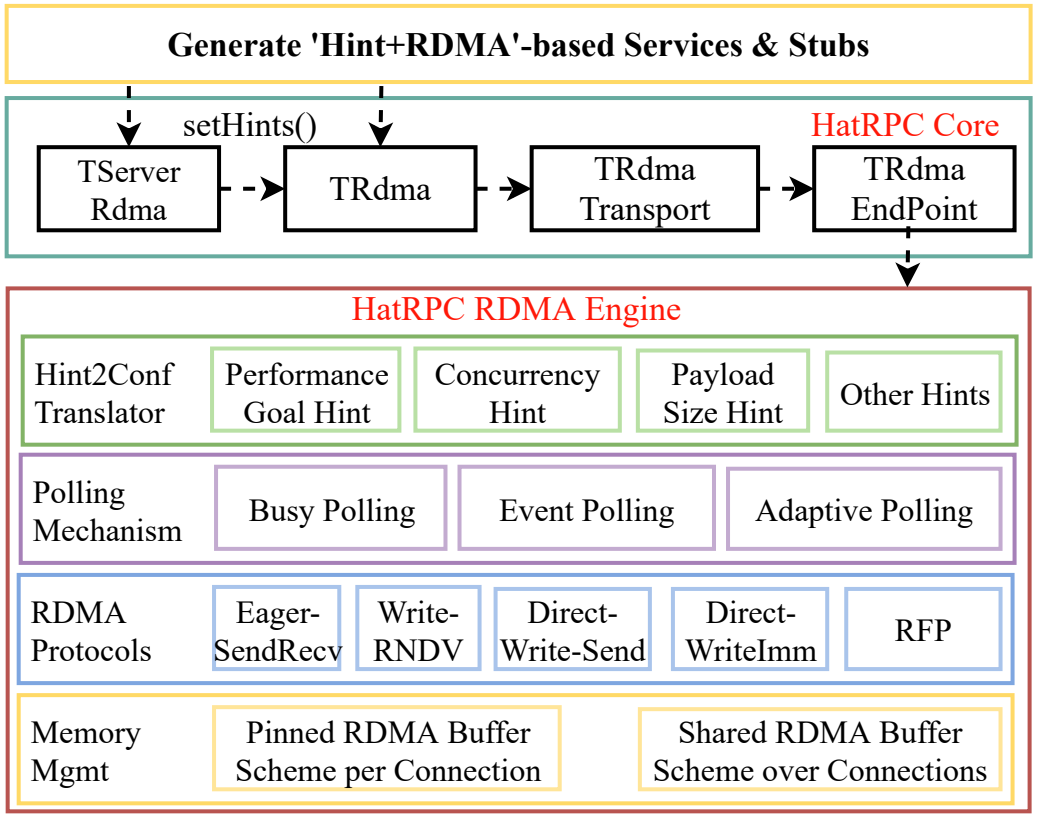
- Focus on performance perspective: latency, throughput or resource utilization
- Depending on the payload size, concurrency and performance goal etc, choose the optimal RDMA protocol and polling mode



HatRPC Hint Design Space and Decision

HatRPC Architecture

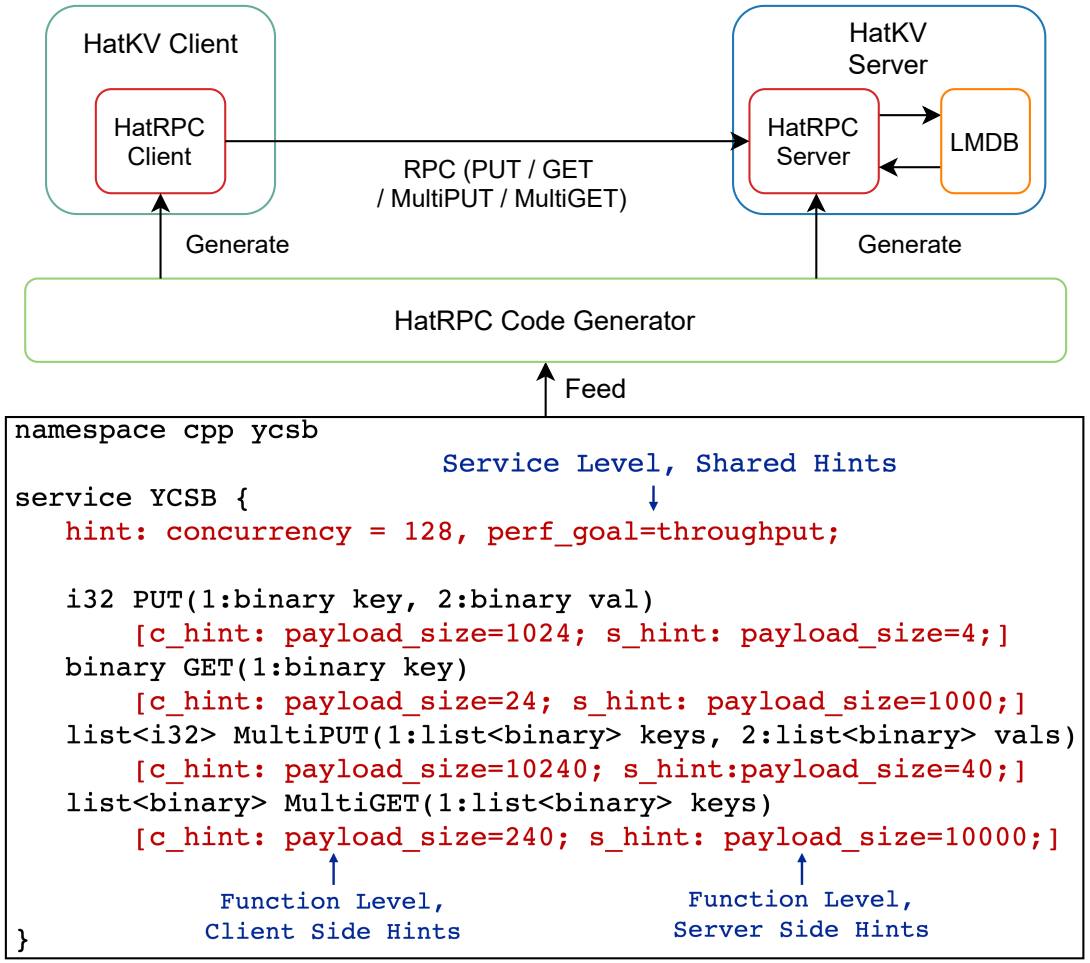
- Implement various RDMA communication protocols, different polling mechanisms and memory management strategies in HatRPC's RDMA engine
- TRdma layer is the counterpart for Thrift's TSocket to bridge Thrift with RDMA engine



HatRPC Architecture

Co-designed HatKV and YCSB Example

- Build HatKV, a KV Store atop HatRPC with LMDB as the storage backend
- Set PUT and GET to be latency centric and the corresponding multi-operations to be throughput centric, set payload size and concurrency level accordingly
- Hints are also passed to LMDB to tune configurations



HatKV and IDL Example for YCSB Workloads

Overview

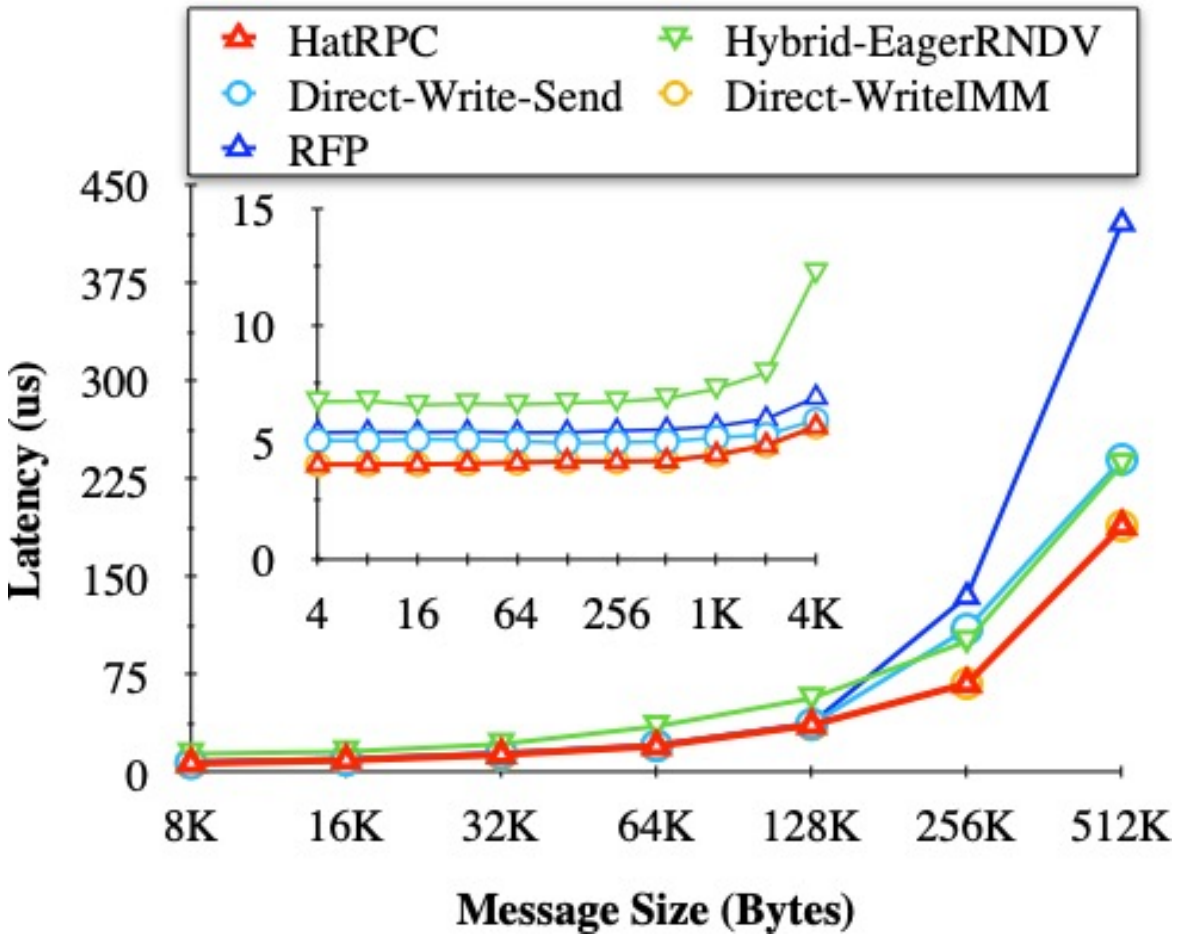
- Introduction
- Motivation and Problem Statements
- HatRPC Design
- Evaluation
- Conclusion and Future Work
- Acknowledgement

Experimental Setup

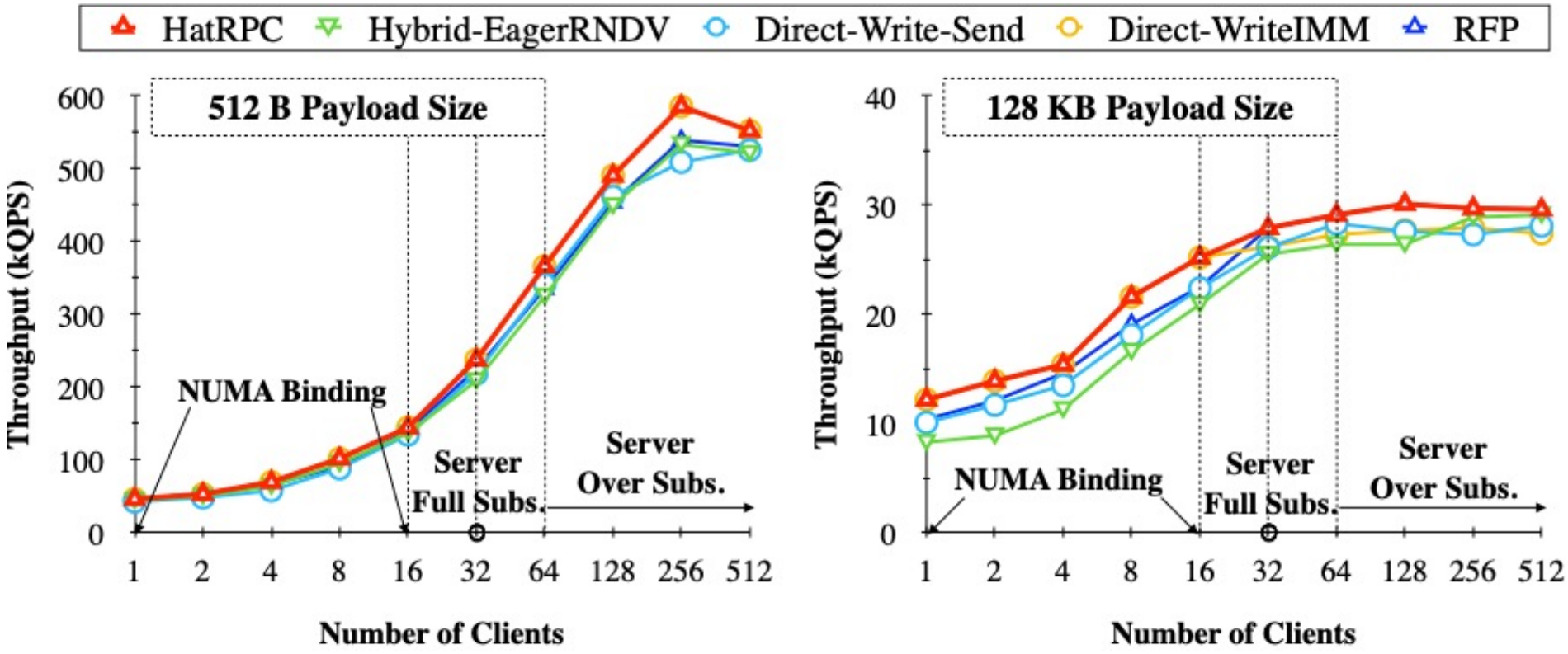
	OSU RI2 Cluster A
Processor	Intel Skylake Gold6132 (2.6 GHZ)
RAM (DDR)	192 GB
Storage	720 GB SSD
Interconnect	ConnectX-5 IB-EDR (100 Gbps)
OS	CentOS Linux 7.6.1810
OFED	OFED-5.0-2.1.8
Scale	10 nodes

Microbenchmark - Latency

- HatRPC can select the best protocol, Direct-WriteIMM for the latency goal, achieving up to 54% improvement over other protocols

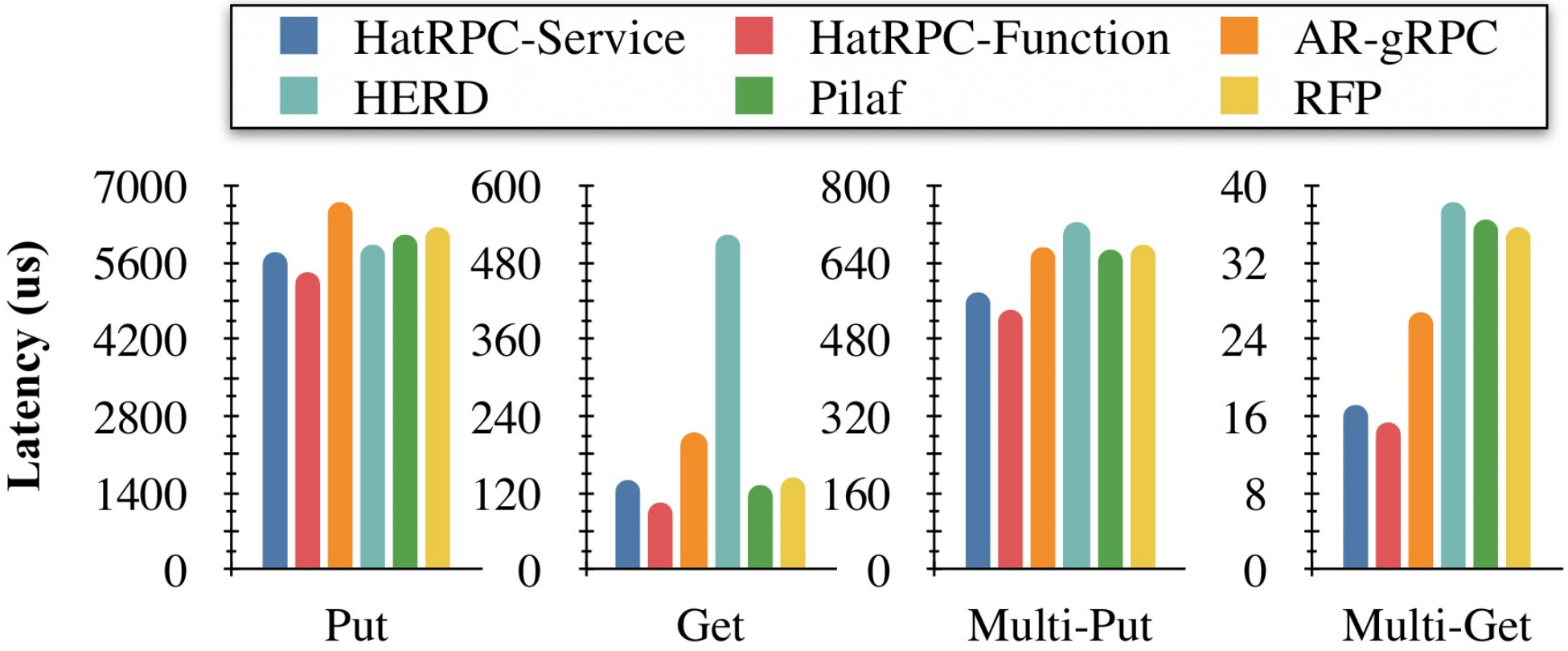


Microbenchmark - Throughput



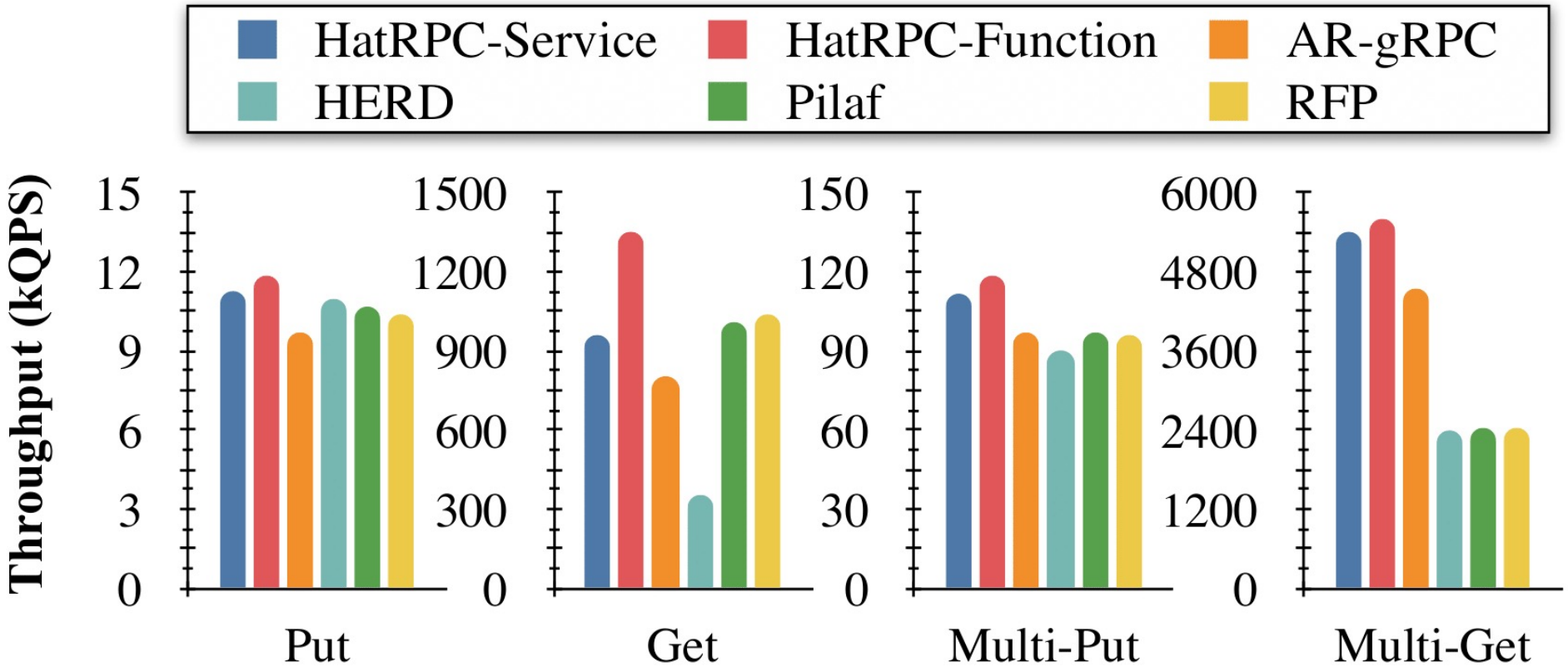
- HatRPC switches from Direct-WriteIMM to RFP when over-subscription (32), yielding up to 20% improvement for 512 B messages and up to 56% for 128 KB messages

YCSB Workload A Evaluation - Latency



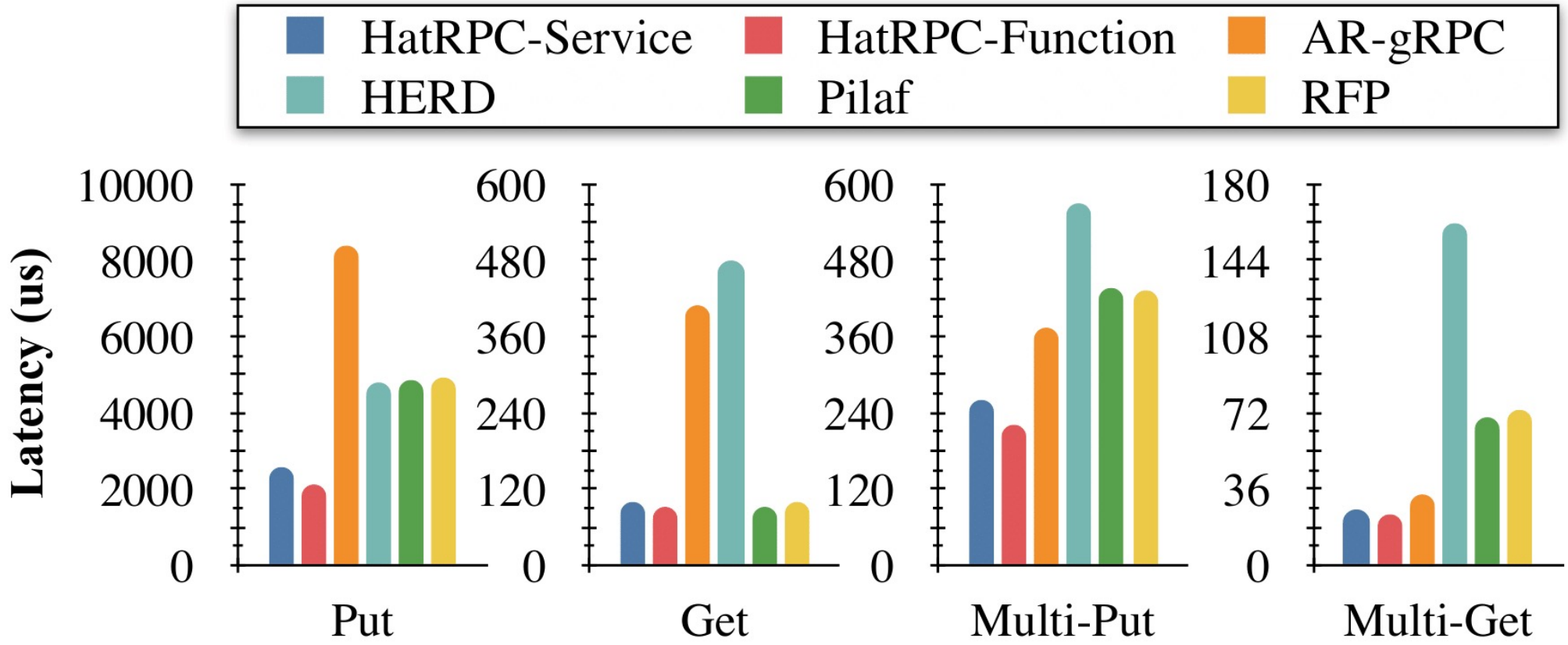
- For YCSB workload A, HatRPC-Service and HatRPC-Function reduce latency by up to 73% and 80%, respectively

YCSB Workload A Evaluation - Throughput



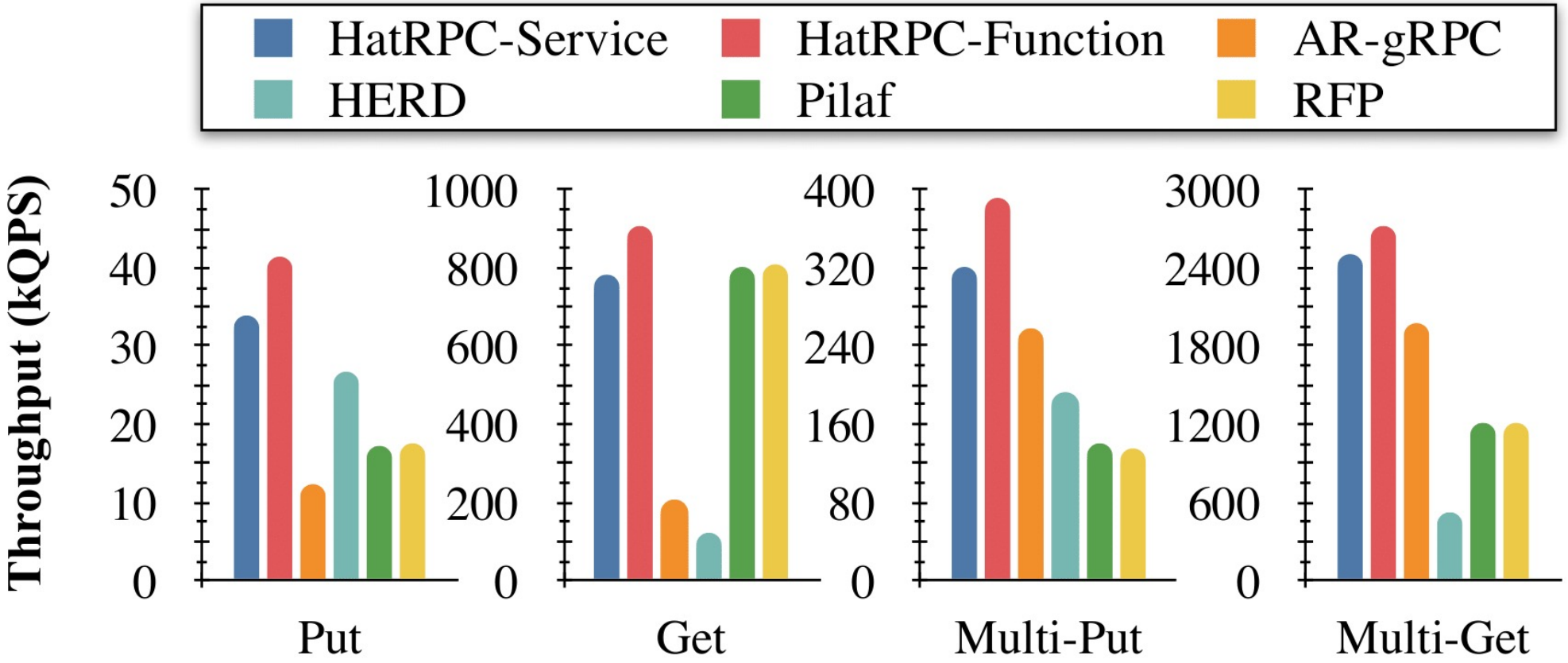
- For YCSB workload A, HatRPC-Service and HatRPC-Function gain a speedup of 2.7x and 3.8x, respectively

YCSB Workload B Evaluation - Latency



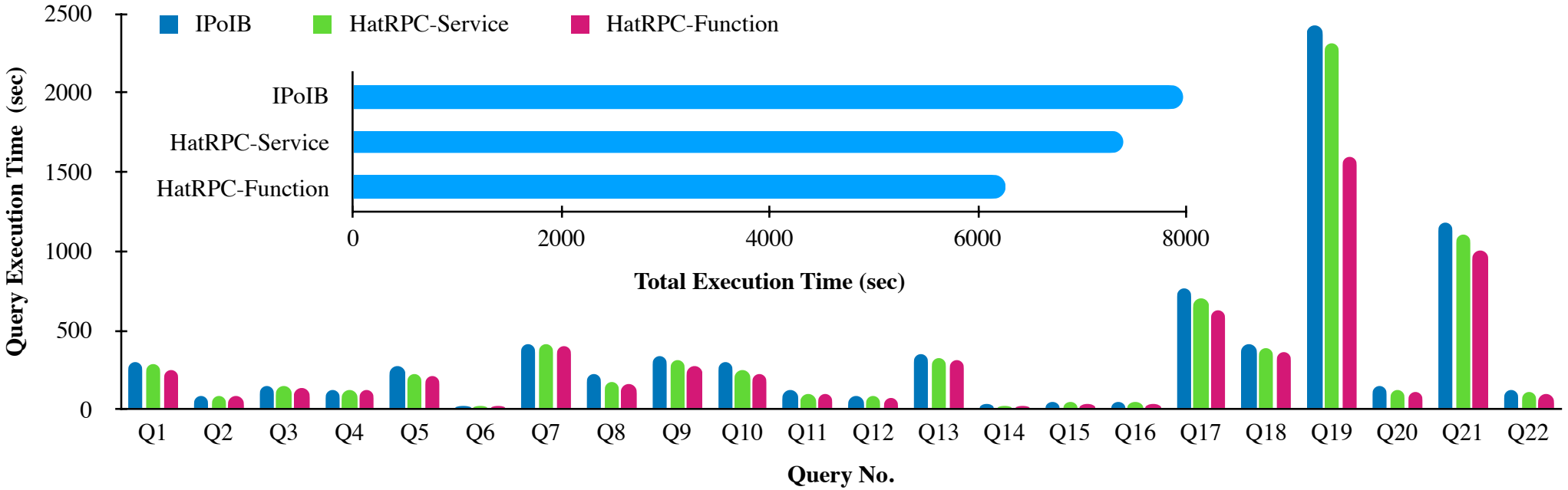
- For YCSB workload B, HatRPC-Service and HatRPC-Function improve the performance by up to 84% and 85%, respectively

YCSB Workload B Evaluation - Throughput



- For YCSB workload B, HatRPC-Service and HatRPC-Function can be up to 6.4x and 7.4x faster than other protocols

TPC-H Evaluation



	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22	Total
IPoIB	297	90	153	128	272	14	419	222	342	294	119	85	349	30	46	48	771	415	2420	151	1184	121	7970
HatRPC-Service	289	84	149	123	221	13	416	180	314	244	104	81	329	23	44	43	705	382	2312	119	1109	115	7399
HatRPC-Function	246	83	137	122	215	12	400	167	274	222	99	74	313	22	42	41	626	358	1599	108	1001	97	6258

- HatRPC-Function is 1.27x and 1.18x faster than IPoIB and HatRPC-Service in terms of total execution time, respectively

Overview

- Introduction
- Motivation and Problem Statements
- HatRPC Design
- Evaluation
- Conclusion and Future Work
- Acknowledgement

Conclusions and Future Work

- Re-examine existing RDMA protocols and their performance in RPC systems
- Propose HatRPC, a hint-accelerated RPC based on Apache Thrift and leverage hints to improve the performance in heterogeneous environments and applications
- Co-design a HatRPC-based key-value store (HatKV) with LMDB as the backend
- HatRPC gains up to 1.27x speedup for TPC-H workloads and HatKV can achieve up to 85% improvement for YCSB workloads
- Future Work: QoS support in HatRPC

Overview

- Introduction
- Motivation and Problem Statements
- HatRPC Design
- Evaluation
- Conclusion and Future Work
- Acknowledgement

Acknowledgement

This work is supported in part by National Science Foundation grant #CCF-1822987 and #CCF-2132049. We would like to sincerely thank Yujie Hui from The Ohio State University for his help in conducting some of the experiments. We also thank the anonymous reviewers for their precious feedback.

Thank You!

- Questions?