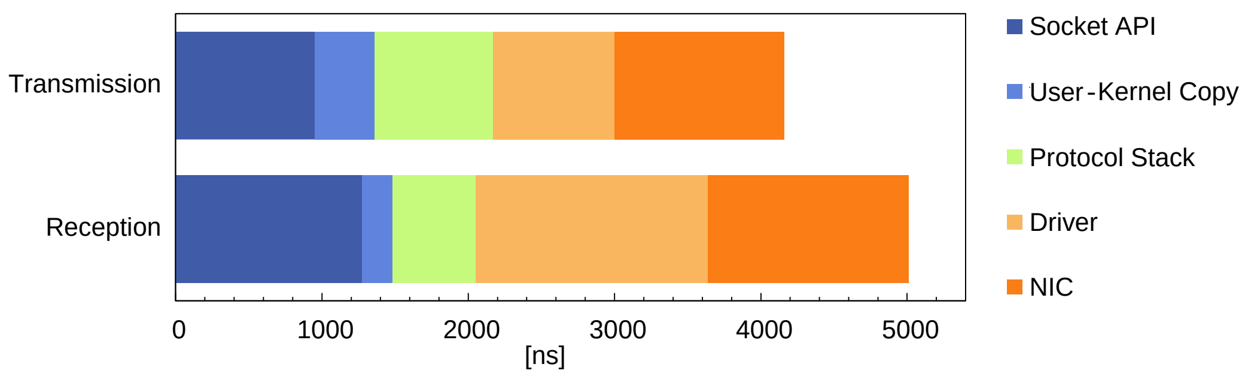
It's Time to Combine Network Advances and Databases

Motivation

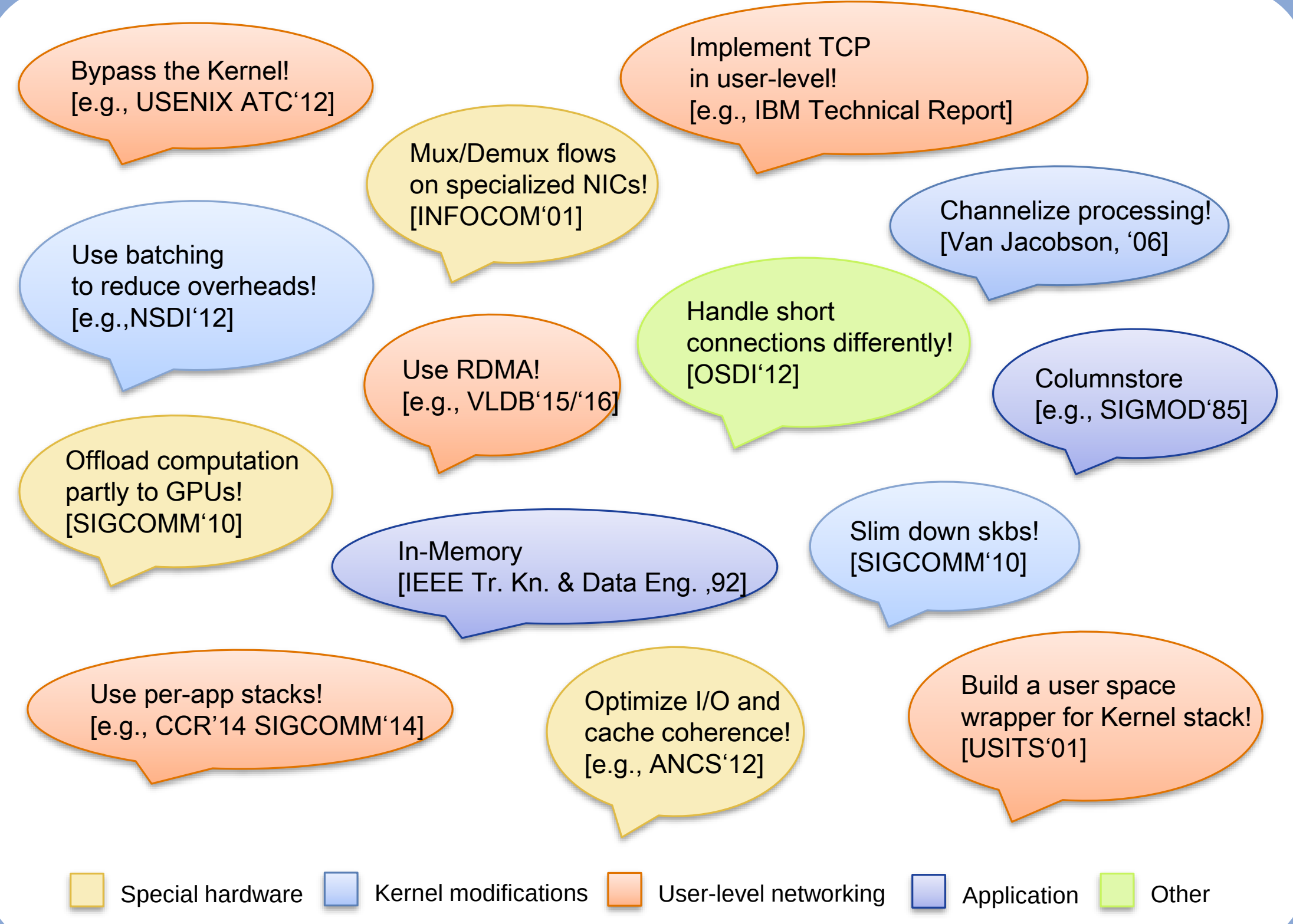
The Quest for Speed

- **Ever increasing database performance demands**
  - Better hardware (faster/multiple CPUs, more RAM, SSDs)
  - Better software (caching, in-memory, …)
  - Scaling out (distribution, scheduling, partitioning, …)
- **Classical networking has significant overheads**
  - Memory allocations and copy operations
  - System calls and context switches



Source: Larsen et al., "Architectural Breakdown of End-to-End Latency in a TCP/IP network, J Parallel Prog, 37:6, (2009)

Related Work



Bypass the Kernel! [e.g., USENIX ATC'12]

Implement TCP in user-level! [e.g., IBM Technical Report]

Mux/Demux flows on specialized NICs! [INFOCOM'01]

Channelize processing! [Van Jacobson, '06]

Use batching to reduce overheads! [e.g.,NSDI'12]

Handle short connections differently! [OSDI'12]

Use RDMA! [e.g., VLDB'15/'16]

Columnstore [e.g., SIGMOD'85]

Offload computation partly to GPUs! [SIGCOMM'10]

In-Memory [IEEE Tr. Kn. & Data Eng. ,92]

Slim down skbs! [SIGCOMM'10]

Use per-app stacks! [e.g., CCR'14 SIGCOMM'14]

Optimize I/O and cache coherence! [e.g., ANCS'12]

Build a user space wrapper for Kernel stack! [USITS'01]

Special hardware · Kernel modifications · User-level networking · Application · Other

Our Solution

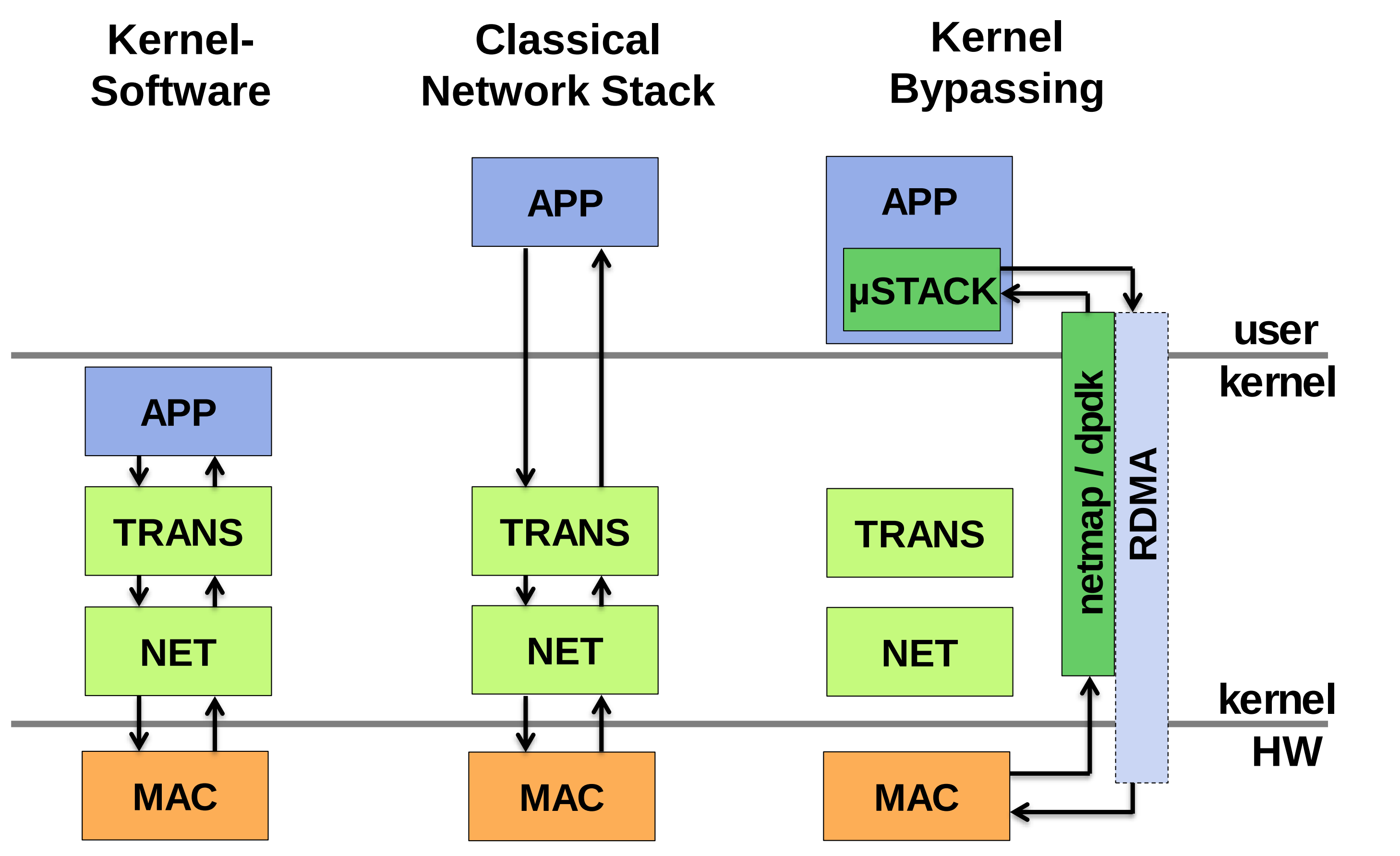
Current Trend: Remote Direct Memory Access

- Drastic performance improvements by bypassing the kernel & CPU; user-level networking
- Highly specialized hardware, infrastructure & software

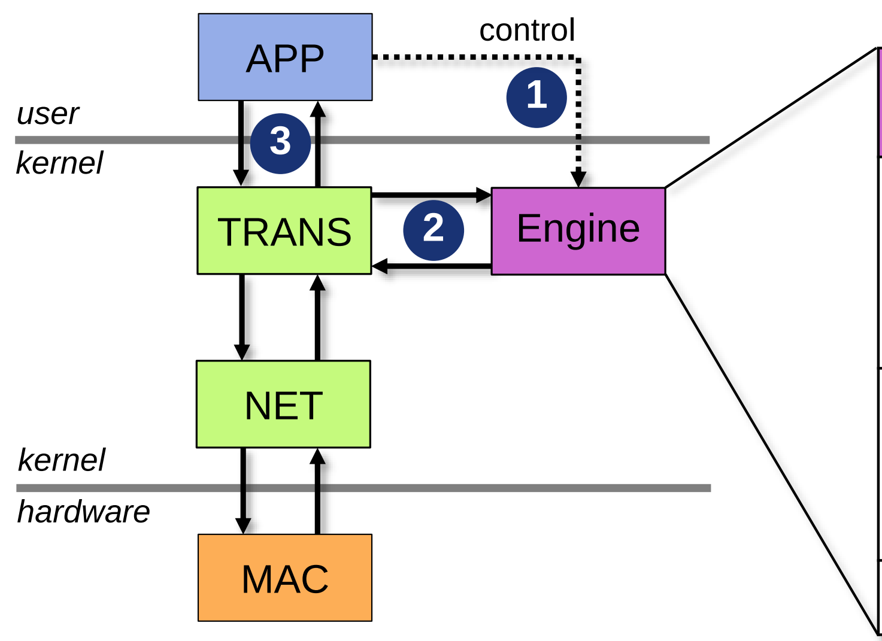
Our Approach:

- **Propose middle ground:**
  - Move parts of application logic into the stack
  - Benefits from well-established TCP/IP network stack
- **Solution: application agnostic offloading scheme**
  - Generic rule execution engine
  - Allows applications to install custom rules (condition/processors) for common requests
  - If condition matches, engine replies using processors instead of application

Other Solutions



Santa Architecture



Rule #	Condition	Reply-Processors
1	Offset: 0 Content: "GET key\r\n END\r\n"	Offset: 0 Content: "VALUE key 0 5\r\n value\r\n END"
2	Offset: 12 Content: "SELECT * FROM table"	Offset: 0 Copy: bytes 0–1 Offset: 2 Content: "all table rows"
…	…	…

Santa Setup

1. Application inserts rule

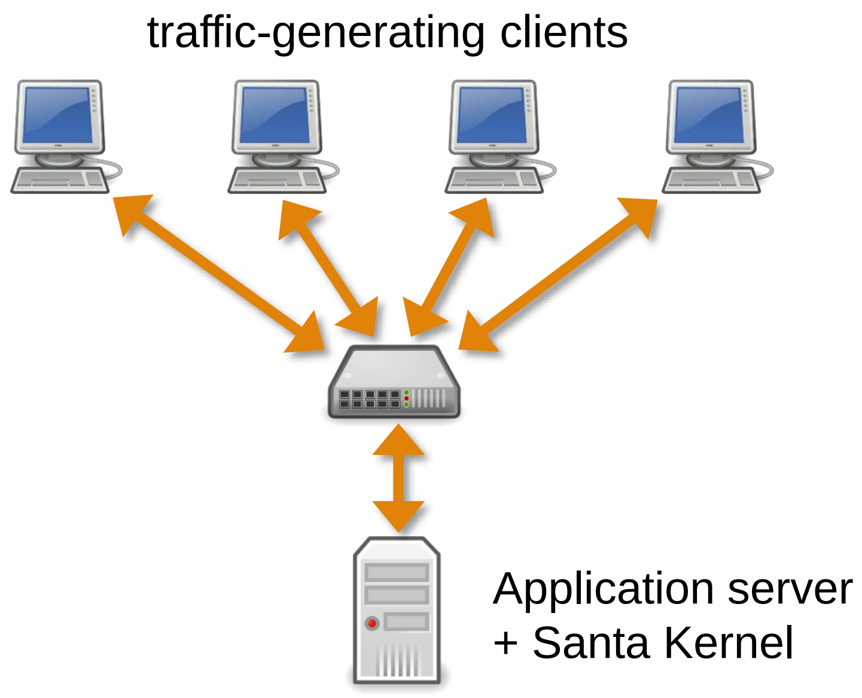
On Receive:

2. If packet matches rule, send pre-determined reply
3. If not, application handles the packet as usual

Engine

- Condition as byte-pattern
- Recipe for reply using processors, e.g.:
  - Write string into packet
  - Copy data from input packet to output packet

Testbed



- 4 traffic generating clients
  - Unmodified Linux kernel
  - Generate workload
- 1 Server
  - Different applications
  - Santa Linux kernel
- 10 Gbit/s Ethernet links

Memcached (KV-Store)

- UDP transport
- Modified multi-socket server
- Pre-populated keys (16 Byte)
- Varying value size
- Modified memaslap
- Four clients

MySQL (RDBMS)

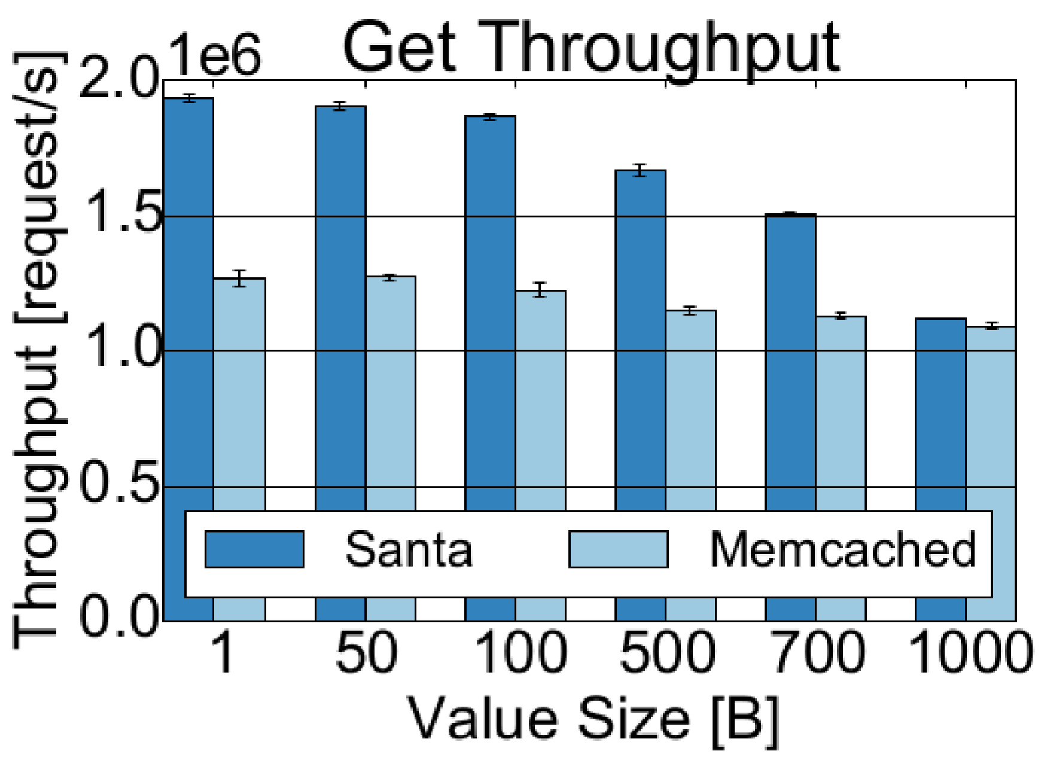
- TCP transport
- Single column indexed table
- Query cache on/off
- Varying payload size
- mysqlslap
- Single client

Acknowledgements

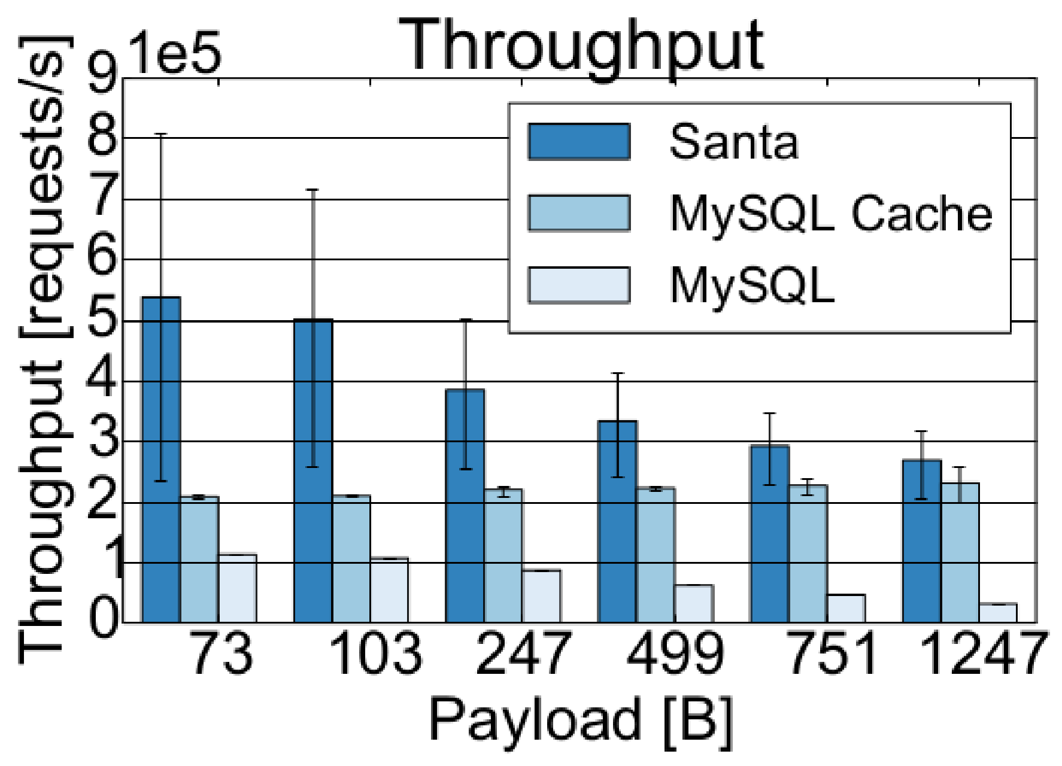
This work has received funding from the European Union's Horizon 2020 research and innovation program 2014–2018 under grant agreement No. 644866 ("SSICLOPS"). It reflects only the authors' views and the European Commission is not responsible for any use that may be made of the information it contains.

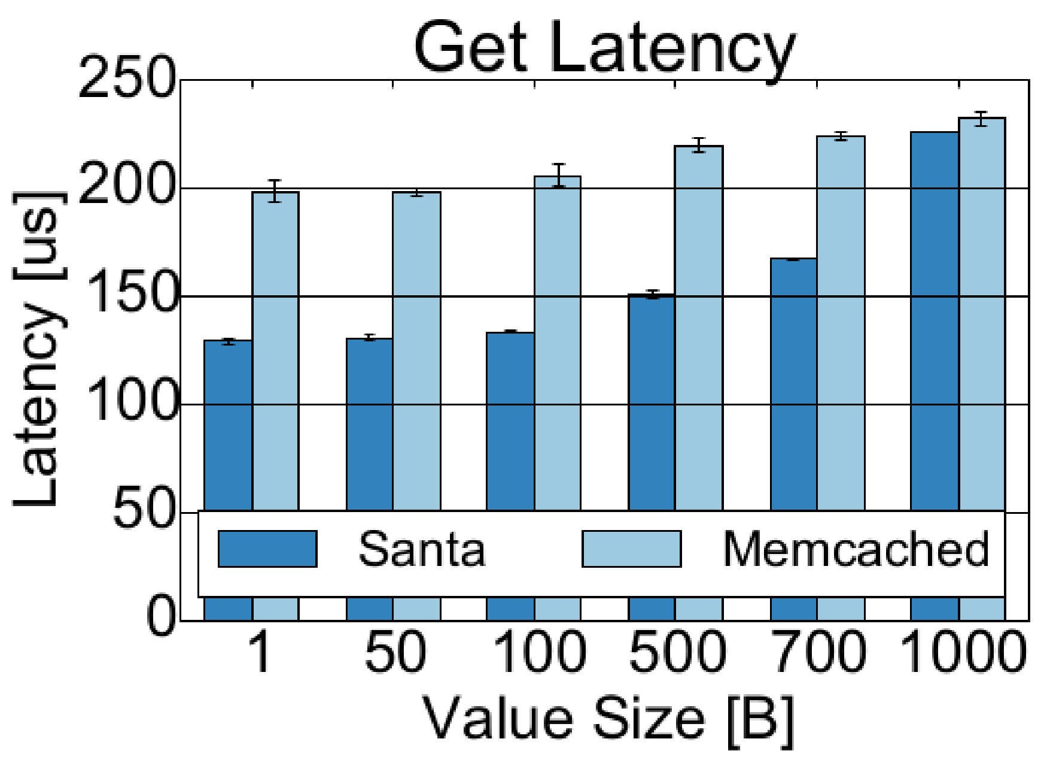
Evaluation

UDP: Memcached



TCP: MySQL





Conclusion

- Reduced computational expenses
- Santa increases throughput significantly
- Latency is significantly decreased
- At higher payloads, copy operations still become dominant
- Highly optimized user-space applications are slower (Memcached, MySQL query cache)
- Recent new networking techniques can speed up databases in general

COM SYS Communication and Distributed Systems

Jens Helge Reelfs, Oliver Hohlfeld, Klaus Wehrle

RWTH AACHEN UNIVERSITY