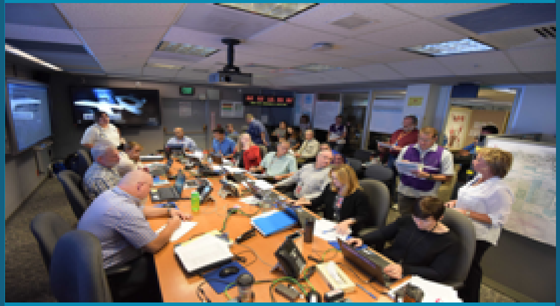
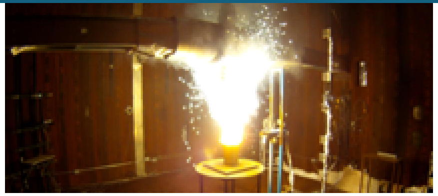
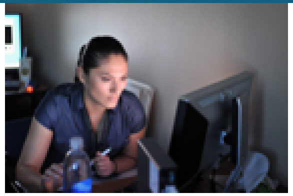
Sandia National Laboratories

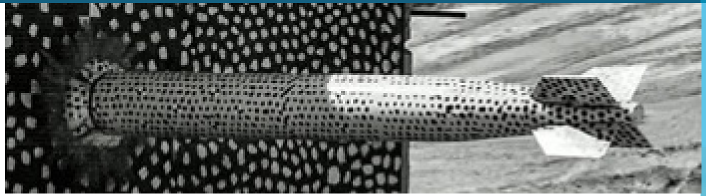
SAND2021-5924C

RVMA: Remote Virtual Memory Access









PRESENTED BY

Ryan E. Grant
Center for Computing Research
Sandia National Laboratories
regrant@sandia.gov

CCR Center for Computing Research

U.S. DEPARTMENT OF ENERGY NNSA

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Outline

- What is RVMA?
- Why RVMA?
- How can RVMA help me?
- Results
- Conclusions

What is RVMA?

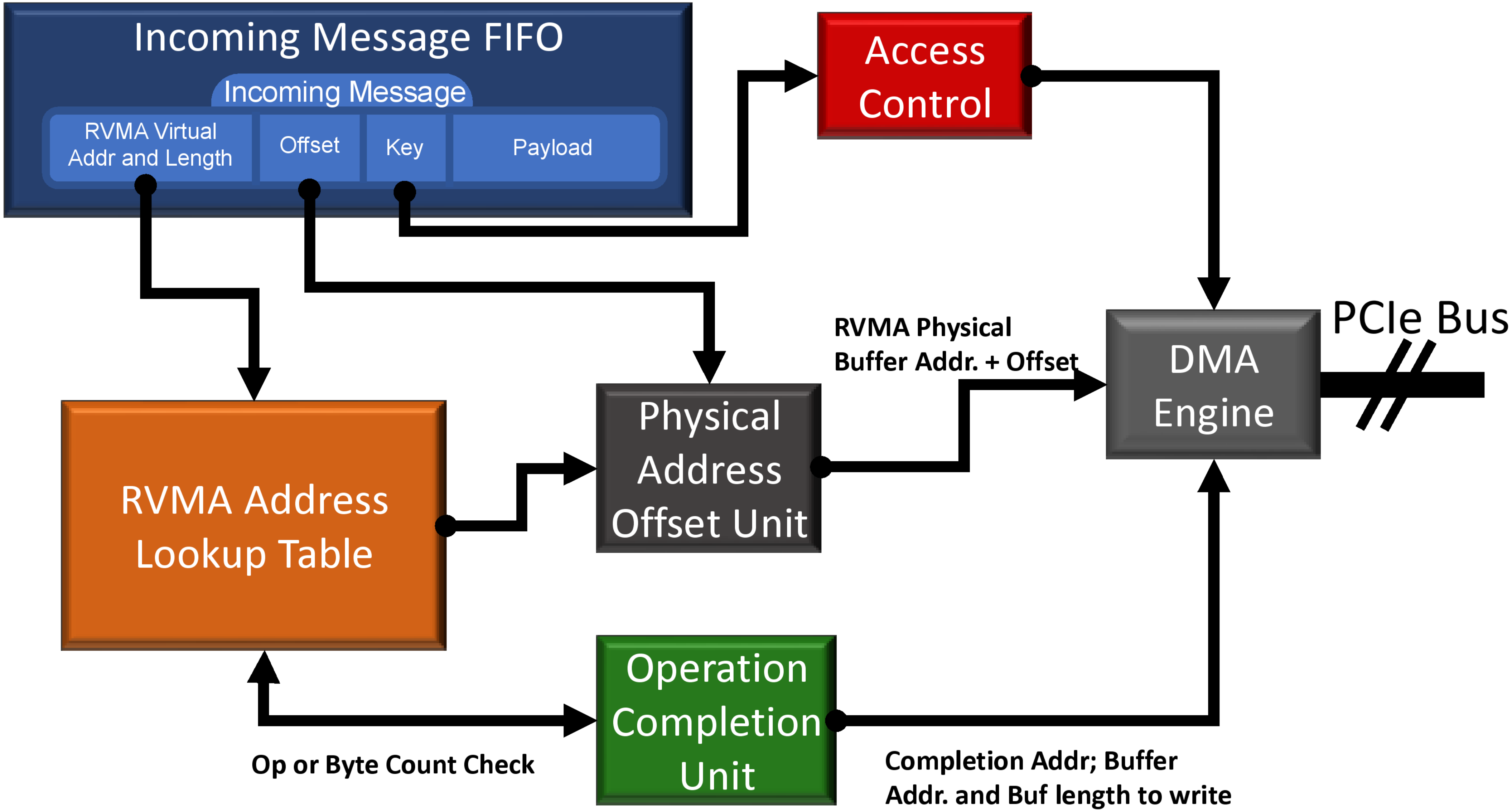
RVMA: Remote Virtual Memory Access

- Improvement to RDMA
  - Take physical addresses and provide abstraction
  - New completion mechanism – hardware notification on epochs
  - Design for adaptively routed networks

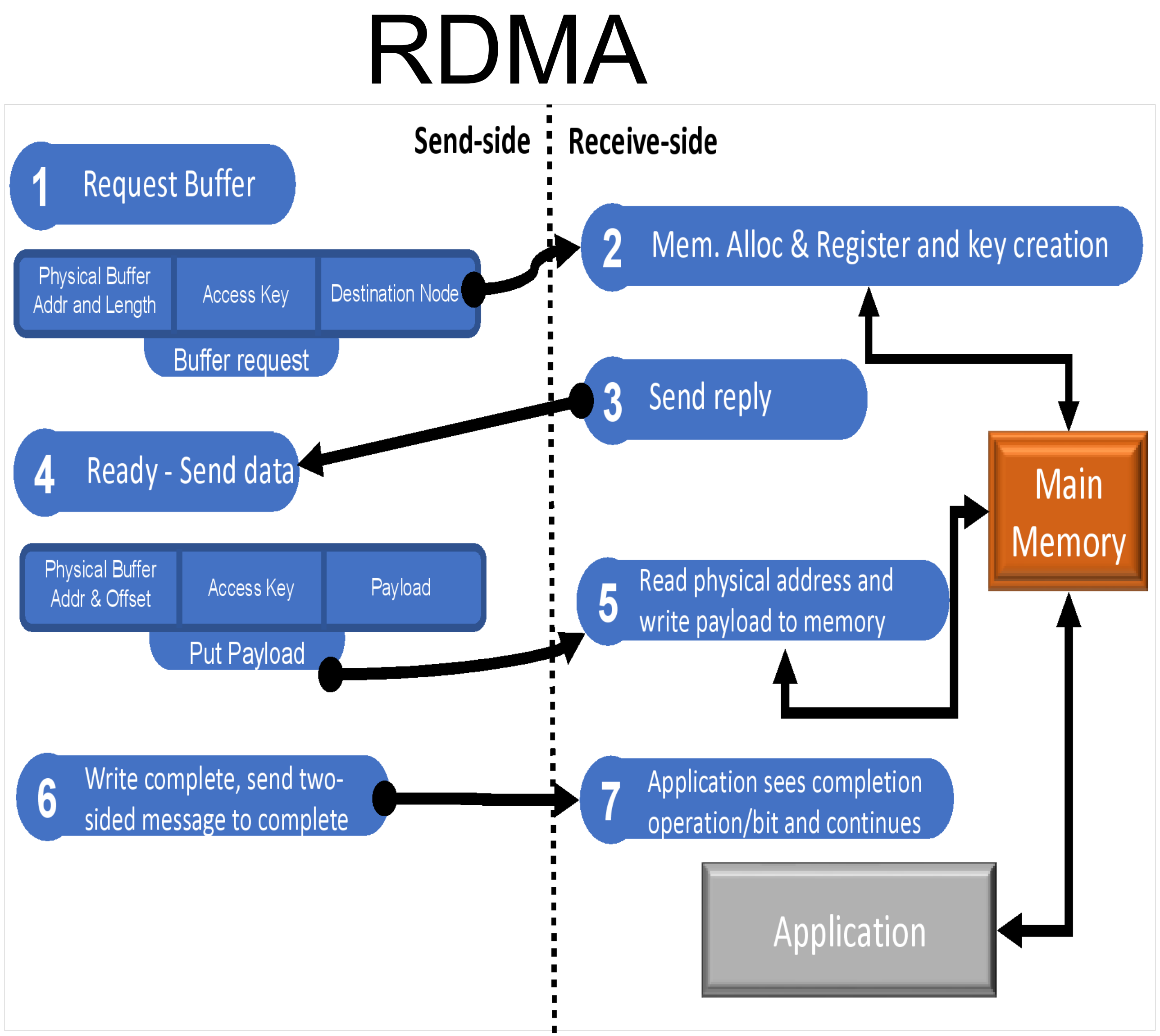
How?

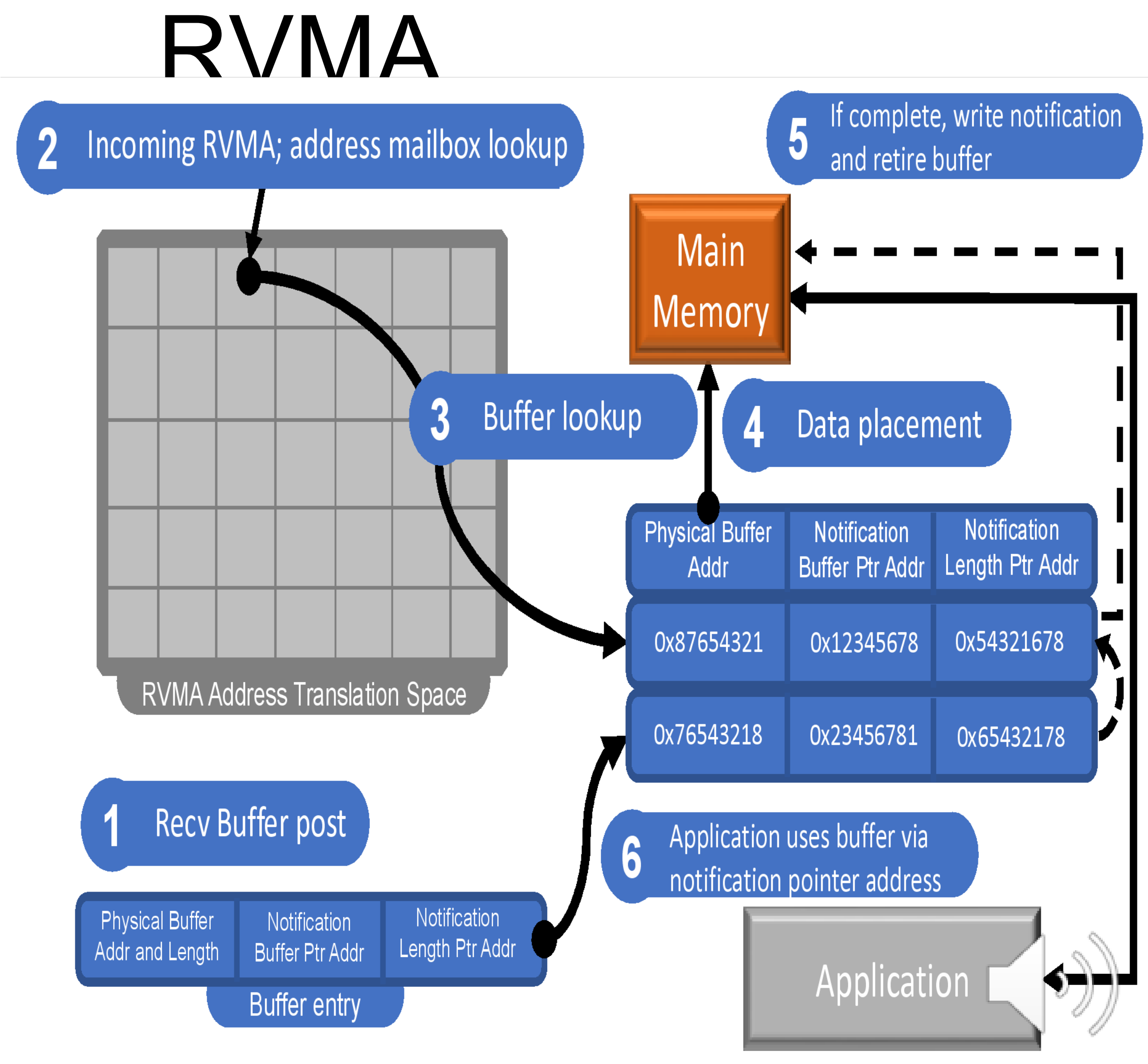
- Provides easy to use completion semantics
- Builds in knowledge about expected data
- Aligns with sockets interfaces
- Supports adaptively routed networks at native speed

RVMA Architecture



RDMA vs RVMA


RDMA
Send-side
Receive-side
1 Request Buffer
Physical Buffer Addr and Length
Access Key
Destination Node
Buffer request
2 Mem. Alloc & Register and key creation
3 Send reply
4 Ready - Send data
Physical Buffer Addr & Offset
Access Key
Payload
Put Payload
5 Read physical address and write payload to memory
Main Memory
6 Write complete, send two-sided message to complete
7 Application sees completion operation/bit and continues
Application


RVMA
2 Incoming RVMA; address mailbox lookup
5 If complete, write notification and retire buffer
Main Memory
3 Buffer lookup
4 Data placement
Physical Buffer Addr
Notification Buffer Ptr Addr
Notification Length Ptr Addr
0x87654321
0x12345678
0x54321678
0x76543218
0x23456781
0x65432178
RVMA Address Translation Space
1 Recv Buffer post
Physical Buffer Addr and Length
Notification Buffer Ptr Addr
Notification Length Ptr Addr
Buffer entry
6 Application uses buffer via notification pointer address
Application

Why RVMA?

RDMA Challenge #1: Hard to use (like a non-cache coherent memory system)

RVMA solution:

- Build in completion semantics so the hardware can understand communication epochs
  - Gives users send/recv like completion but with put/get operation flexibility
  - Use operation counters or byte counters to determine when transfer epochs are complete at the receiver without send-side messages

RDMA Challenge #2

RDMA Challenge #2: Adaptively routed networks mean polling on the final bit/byte in an RDMA buffer is no longer possible to indicate completion

RVMA solution:

- Offsets place data in the right buffer location regardless of arrival order (true for RDMA too)
- Counters make byte ordering unnecessary for completion "cheats" as well as final completion handshakes or subsequent send/recv operations on the channel
- Due to this RVMA can take advantage of unordered delivery and multiple paths through an adaptively routed network with no additional completion overhead

RDMA Challenge #3

RDMA Challenge #3: Sockets are hard to implement and use over RDMA

RVMA Solution:

- Abstract elements of RDMA that make sockets hard to implement –
  - explicit buffer addresses, handshakes, state retention, translating sockets semantics to RDMA queue pairs
- Don’t use hardware addresses anymore – use virtual ones that can have well known values
  - Communicate without setting up explicit remote memory addresses that need to be stored at the sender -side
  - Use well known virtual addresses that translate to receive-side locally managed buffers
  - Also reduces setup overheads whenever a new buffer is needed with a remote host
- Allow buffers to be treated like streams

RDMA Challenge #4

RDMA makes it hard to recover from failures as addresses are physical memory addresses on hosts

RVMA Solution:

- Abstracted addresses allow for migration to other nodes without strict physical memory address requirements
- RVMA keeps rotating list of buffers so users can roll back to known good buffers if failures happen in a given epoch
  - RDMA typically uses the same buffers over and over again, scribbling over previous values

Does Middleware Have to Solve Everything?

No! We can provide assistance from hardware to make middleware’s job easier

- Provide failover capabilities in the switches
  - Re-route traffic from failed nodes to replacement nodes
  - Re-routing can be transparent to the application/middleware
    - Allow previous addresses to be forwarded, and notify nodes that use addresses of failed nodes to update their address information to the new node address
    - Essentially transparent to middleware, so finding replacement nodes is effortless, but getting replacements up to speed as useful compute nodes is hard
  - Hard problem – supporting this while still providing HPC switch latencies
    - Data center switches can do this today, but take significantly more time per switch hop
- Make hardware help in recovering from errors on nodes without failure
  - SNL working on hardware solutions that will allow communication roll-back to previous known good communication epochs
  - Allows for using alternative methods to checkpoint/restart

How Does Hardware Help?

Communication Epochs

- Need to be understood by hardware
  - RDMA is a good technique, but puts all of the control in the hands of the initiator
    - Target allows access to memory, then puts/gets anything requested into that memory from the initiator
    - No concept of what's being done to memory, communication epochs or patterns until after the communication is complete (initiator says it's done)
    - This makes fault tolerance very hard – MPI guarantees nothing when failures happen and RMA windows are open
      - Target never knows if its buffer is in a good state for computation or not
      - Buffers are overwritten constantly – memory buffer is almost always partially completed version of next epoch
- Understanding epochs at the target allows the target to 'checkpoint' the communication

The Hard Work is in the Restore

Using these hardware techniques:

- Solves many problems that are being explored today from system software
- Makes the hard job of re-establishing network communications in a consistent good group of compute nodes easy

There are still hard problems to solve:

- Hardware does not solve the problem of how to get a replacement node doing useful computation
- Middleware and application software need to be able to coordinate complicated approaches to restoring compute
- This may be application specific – need to support this with middleware on how to roll back computation on good nodes and/or get replacement nodes up to speed

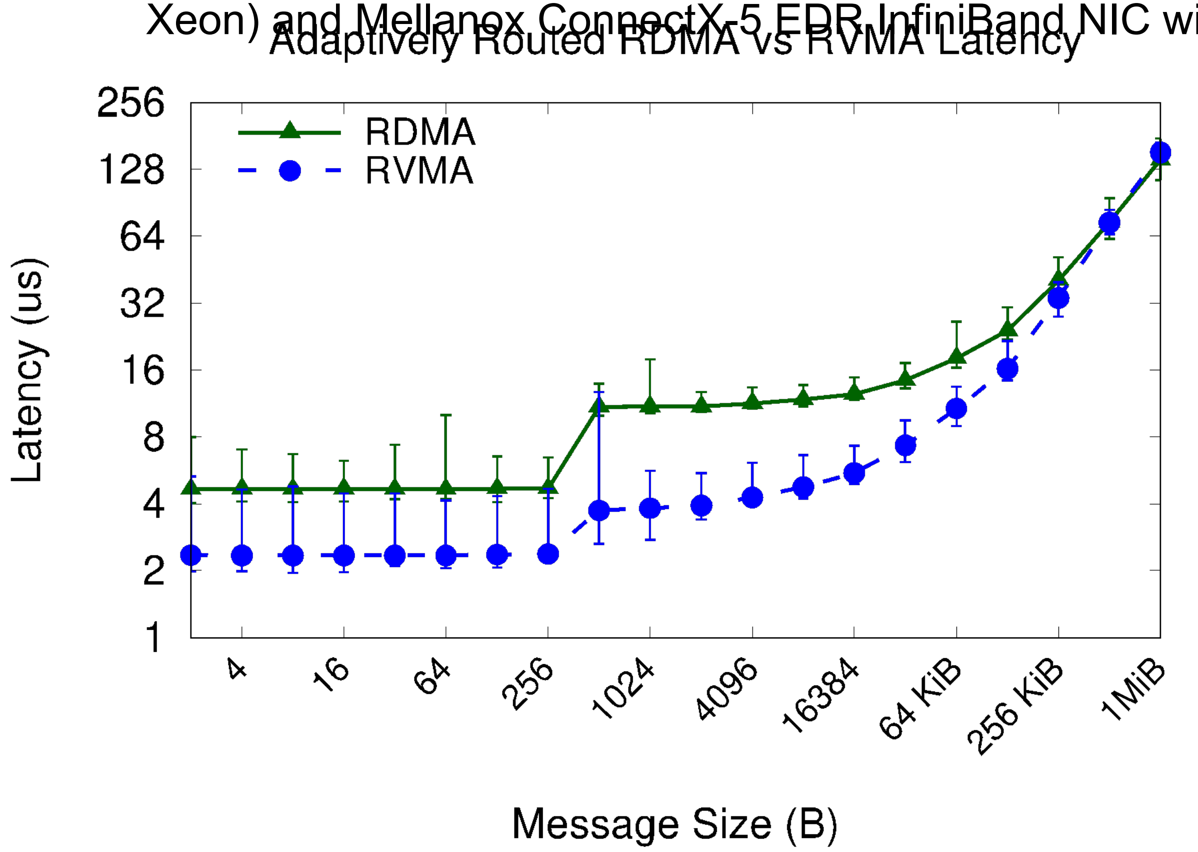
RVMA Performance

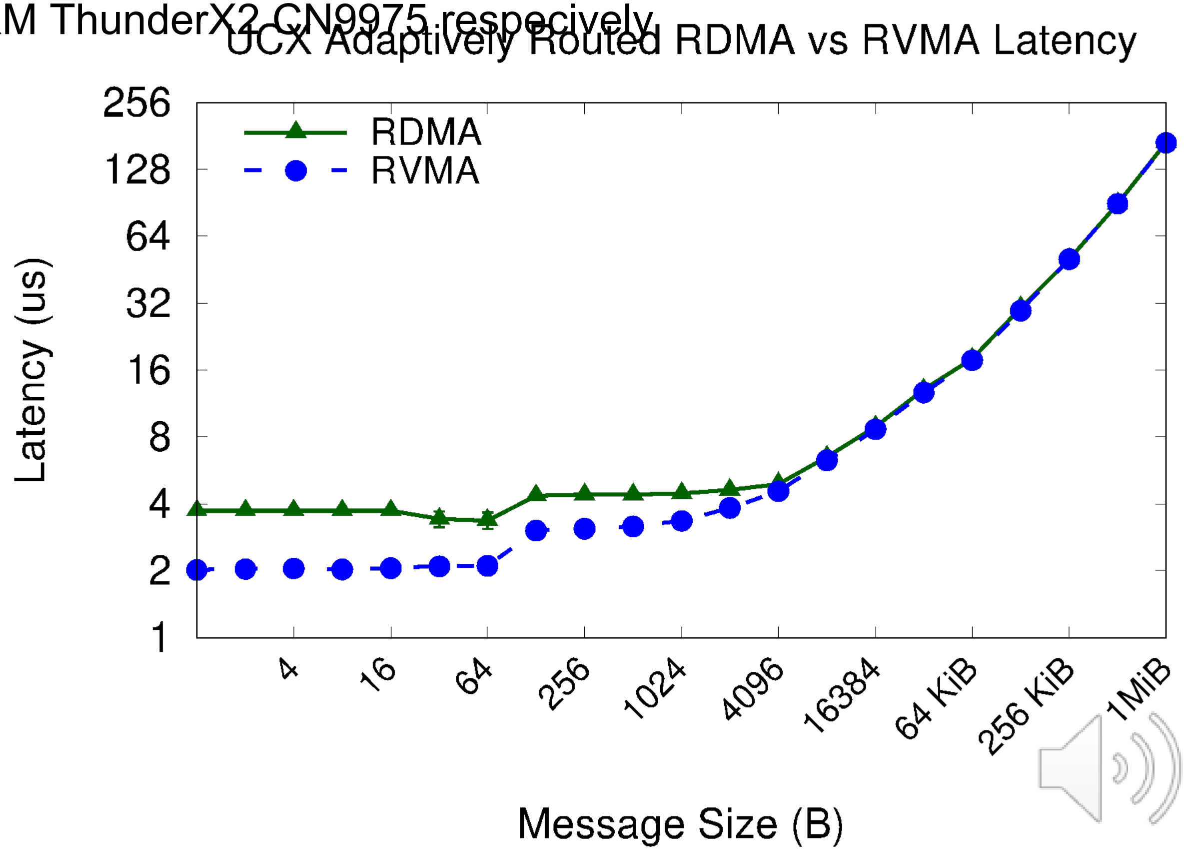
RDMA vs RVMA Emulation

Experiment: Take existing RDMA hardware and add emulation layer to it to evaluate RVMA performance

Approach: Assume adaptively routed network (no last byte polling) and allow RVMA to avoid handshakes and physical buffer address exchange needed in RDMA.

Platform: OFED Perftest and UCX latency test on Intel OmniPath InfiniBand 100Gbps with Intel Sklake (platinum 8160 Xeon) and Mellanox ConnectX-5 EDR InfiniBand NIC with ARM ThunderX2 CN9975 respecively





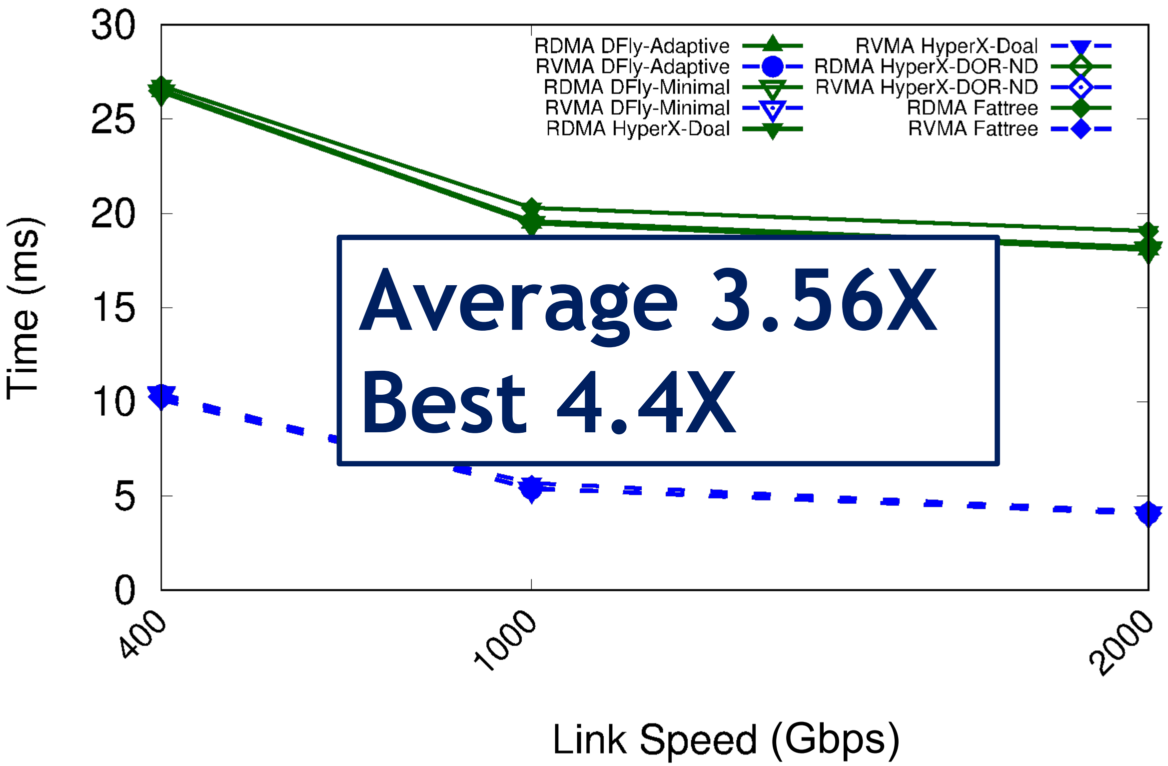
Large Scale Simulation

Experiment: Large scale simulation of RVMA with popular motifs (application proxies)

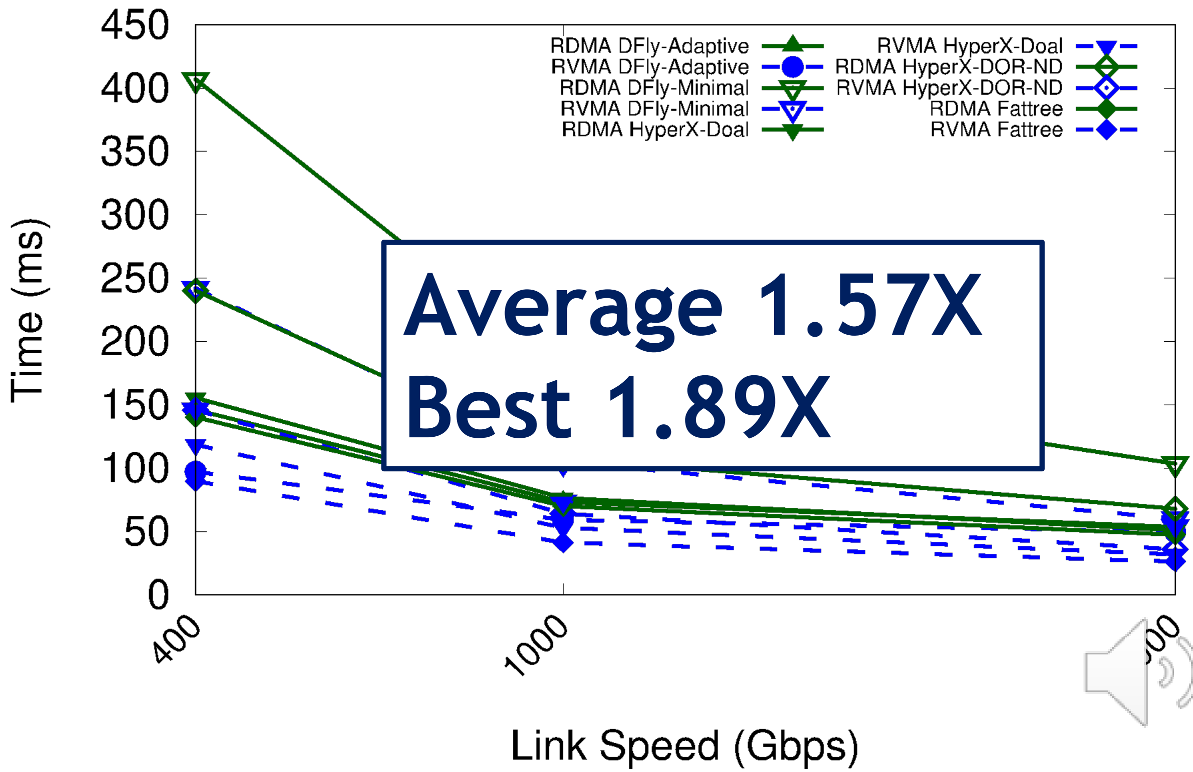
Approach: New RVMA component developed for the simulator, verified with emulation microbenchmarks.

Platform: SST simulator run in parallel on Cray XC40 simulating system with 262,144 cores split up as 8,192 nodes with 32 cores per node

RVMA Vs RDMA for Sweep3D Motif



RVMA Vs RDMA for Halo3D Motif



Where does the performance boost come from?

1. Known abstracted buffered addresses avoids physical memory setup/exchange

2. Better completion mechanism, no completion roundtrip

3. Reclaim byte-ordered last byte polling performance with support for adaptively routed networks

Conclusions

- ✓RVMA offers up to 45.8-65.8% latency reduction over RDMA in real world tests with emulation timings
- ✓RVMA offers an average 3.56X speedup on a sweep3D simulation and 1.57X better performance on Halo exchange than RDMA for large scale simulations (256K cores)
- ✓RVMA also provides improved interfaces for usability and compatibility with sockets,
- ✓RVMA allows users to take advantage of monitor/mwait type host side solutions for waking threads on RVMA completion

Questions?

regrant@sandia.gov

Acknowledgments:

This work was funded through the Computational Systems and Software Environment sub-program of the Advanced Simulation and Computing Program funded by the National Nuclear Security Administration

References

[1] B. W. Barrett, R. Brightwell, R. E. Grant, S. Hemmert, K. Pedretti, K. Wheeler, K. Underwood, R. Riesen, T. Hoefler, A. B. Maccabe, and T. Hudson, The Portals 4.2 Network Programming Interface, Sandia Na- tional Laboratories, November 2018, technical Report SAND2018-12790.

[2] J. A. Zounmevo, X. Zhao, P. Balaji, W. Gropp, and A. Afsahi, "Nonblocking epochs in MPI one-sided communication," in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE Press, 2014, pp. 475–486.

[3] J. Liu, J. Wu, and D. K. Panda, "High performance RDMA-based MPI implementation over InfiniBand," International Journal of Parallel Programming, vol. 32, no. 3, pp. 167–198, 2004.

[4] J. Jose, H. Subramoni, M. Luo, M. Zhang, J. Huang, M. Wasi-ur Rahman, N. S. Islam, X. Ouyang, H. Wang, S. Sur et al., "Memcached design on high performance RDMA capable interconnects," in 2011 International Conference on Parallel Processing. IEEE, 2011, pp. 743–752.

[5] N. S. Islam, M. W. Rahman, J. Jose, R. Rajachandrasekar, H. Wang, H. Subramoni, C. Murthy, and D. K. Panda, "High performance RDMA-based design of HDFS over InfiniBand," in SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. IEEE, 2012, pp. 1–12.

[6] T. Ma, T. Ma, Z. Song, J. Li, H. Chang, K. Chen, H. Jiang, and Y. Wu, "X-RDMA: Effective RDMA middleware in large-scale production environments," in 2019 IEEE International Conference on Cluster Computing (CLUSTER). IEEE, 2019, pp. 1–12.

[7] Z. Li, N. Liu, and J. Wu, "Toward a production-ready general-purpose RDMA-enabled RPC," in Proceedings of the ACM SIGCOMM 2019 Conference Posters and Demos, 2019, pp. 27–29.

[8] S. Hefty, "Rsockets," in 2012 OpenFabris International Workshop, Monterey, CA, USA, 2012.

[9] R. E. Grant, M. J. Rashti, A. Afsahi, and P. Balaji, "RDMA capable iWARP over datagrams," in Parallel & Distributed Processing Symposium (IPDPS), 2011 IEEE International. IEEE, 2011, pp. 628–639.

[10] R. E. Grant, M. G. Dosanjh, M. J. Levenhagen, R. Brightwell, and A. Skjellum, "Finepoints: Partitioned multithreaded MPI communication," in International Conference on High Performance Computing. Springer, 2019, pp. 330–350.

[11] W. Schonbein, M. G. Dosanjh, R. E. Grant, and P. G. Bridges, "Measuring multithreaded message matching misery," in European Conference on Parallel Processing. Springer, 2018, pp. 480–491.

[12] A. A. Awan, C.-H. Chu, H. Subramoni, and D. K. Panda, "Optimized broadcast for deep learning workloads on dense-GPU InfiniBand clusters: MPI or NCCL?" in Proceedings of the 25th European MPI Users' Group Meeting, 2018, pp. 1–9.

[13] M. J. Rashti, R. E. Grant, A. Afsahi, and P. Balaji, "iWARP redefined: Scalable connectionless communication over high-speed ethernet," in High Performance Computing (HiPC), 2010 International Conference on. IEEE, 2010, pp. 1–10.

[14] R. Belli and T. Hoefler, "Notified access: Extending remote memory access programming models for producer-consumer synchronization," in 2015 IEEE International Parallel and Distributed Processing Symposium. IEEE, 2015, pp. 871–881.

[15] A. Hori, J. Lee, and M. Sato, "Audit: A new synchronization api for the get/put protocol," Journal of Parallel and Distributed Computing, vol. 72, no. 11, pp. 1464–1470, 2012.

[16] D. Kim, T. Yu, H. H. Liu, Y. Zhu, J. Padhye, S. Raindel, C. Guo, V. Sekar, and S. Seshan, “Freeflow: Software-based virtual RDMA networking for containerized clouds,” in 16th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 19), 2019, pp. 113–126.

[17] G. Shah, J. Nieplocha, J. Mirza, C. Kim, R. Harrison, R. K. Govindaraju, K. Gildea, P. DiNicola, and C. Bender, “Performance and experience with lapi-a new high-performance communication library for the ibm rs/6000 sp,” in Proceedings of the First Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing. IEEE, 1998, pp. 260–266.

[18] Barrett, B.W. and Brightwell, R. and Hemmert, S. and Pedretti, K. and Wheeler K. and Underwood, K.D. and Reisen, R. and Maccabe, A.B., and Hudson, T., The Portals 4.0 network programming interface, Sandia Na- tional Laboratories, November 2012, technical Report SAND2012-10087.

[19] T. Hoefler, S. Di Girolamo, K. Taranov, R. E. Grant, and R. Brightwell, “sPIN: High-performance streaming processing in the network,” in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2017, pp. 1–16.

[20] W. Schonbein, R. E. Grant, M. G. Dosanjh, and D. Arnold, “INCA: in-network compute assistance,” in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2019, pp. 1–13.

[21] R. E. Grant, P. Balaji, and A. Afsahi, “A study of hardware assisted IP over InfiniBand and its impact on enterprise data center performance,” in Performance Analysis of Systems & Software (ISPASS), 2010 IEEE International Symposium on. IEEE, 2010, pp. 144–153.

[22] A. Kalia, M. Kaminsky, and D. G. Andersen, “Design guidelines for high performance RDMA systems,” in 2016 USENIX Annual Technical Conference (USENIX ATC 16), 2016, pp. 437–450.

[23] R. E. Grant, M. J. Rashti, and A. Afsahi, “An analysis of QoS provisioning for Sockets Direct Protocol vs. IPoIB over modern InfiniBand networks,” in Parallel Processing-Workshops, 2008. ICPP-08. International Conference on. IEEE, 2008, pp. 79–86.

[24] “Myricom datagram bypass layer documents,” https://www. ariacybersecurity.com/support/, (Accessed on 10/14/2020).

[25] P. Balaji, S. Narravula, K. Vaidyanathan, S. Krishnamoorthy, J. Wu, and D. K. Panda, “Sockets Direct Protocol over InfiniBand in clusters: is it beneficial?” in IEEE International Symposium on-ISPASS Performance Analysis of Systems and Software, 2004. IEEE, 2004, pp. 28–35.

[26] B. Alverson, E. Froese, L. Kaplan, and D. Roweth, “Cray XC series network,” Cray Inc., White Paper WP-Aries01-1112, 2012.

[27] S. Derradji, T. Palfer-Sollier, J.-P. Panziera, A. Poudes, and F. W. Atos, “The BXI interconnect architecture,” in 2015 IEEE 23rd Annual Symposium on High-Performance Interconnects. IEEE, 2015, pp. 18–25.