NetDAM: Network Direct Attached Memory with Programmable In-Memory Computing ISA

Kevin Fang
Cisco Systems
Shanghai, China
zhiyfang@cisco.com

David Peng
Cisco Systems
Shanghai, China
yupeng@cisco.com

ABSTRACT

Data-intensive applications like distributed AI-training may require multi-terabytes memory capacity with multi-terabits bandwidth. We directly attach the memory to the ethernet controller with some programable logic to design an efficient hardware "template" for *Memory pooling* and *in-memory / in-network computing*.

We built an FPGA prototype of the NetDAM, and we demonstrate MPI-Allreduce communication case, the NetDAM can be used as a software and hardware friendly programmable architeture with high performance alternative for RDMA.

CCS CONCEPTS

• **Computer systems organization → Cloud computing**; • **Networks → Transport protocols**; • **Hardware → Networking hardware**.

KEYWORDS

Mempool, In-Memory Computing,In-Network Computing, Programmable ISA.

ACM Reference Format:
Kevin Fang and David Peng. 2021. NetDAM: Network Direct Attached Memory with Programmable In-Memory Computing ISA. In *draft*. ACM, New York, NY, USA, 5 pages. https://doi.org/xx

1 INTRODUCTION

In a conventional computer, the processing and memory units are physically separated. Consequently, significant data need to be moved during computation, which creates a performance bottleneck commonly called "von Neumann Bottleneck". Domain Specific ASICs(DSA) start to address this problem, however, ultra-high bandwidth and ultra-low latency communication is still required for DSA accelerator and CPU.

1.1 Intra-host vs Inter-host communication

PCIe is widely used for intra-host communication, direct memory access (DMA) is used for inter-chip communication. RDMA [11] over Ethernet simply extend the DMA operation to inter-host network. The go-back-N method require lossless Ethernet, DCQCN [14] is designed for congestion control and reliable data delivery. PFC based congestion control mechanism introduce significant latency, even more it may not work well at large scale and cloud based virtual private network environment, some vendors like fungible [3] use TCP Offload Engine on NIC with iWARP [7], HPCC [9] NDP [5] are designed to eliminate the PFC but require special Ethernet switch to support In-network-telemetry(INT) or NDP. Simultaneously, [10] shows that PCIe, alongside its interaction with the root complex and device drivers, can significantly impact the performance of end host networking. GenZ, CCIX, CXL are designed to address such problem.

Therefore,we re-examine the *intra-host* and *inter-host* communication protocols,Intra-host protocol often use *shared memory mode*, the inter-host protocol often use *message passing mode*, the design principal has significant difference:

- **topology:** intra-host communication has dedicated topology which could easily use fixed algorithm(eg. DFS) to encoding devices address. message routing is very simple. inter-host communication are topology independent, multi-path and overlay transport may cause message routing more complicated.
- **latency:**The round-trip-time between intra-host chips only sub-nanoseconds, inter-host communication may
- **loss:** intra-host communication has dedicated topology with lossless circuits, meanwhile congestion or interim node failure may cause packet drop on inter-host communication
- **coherency:** The round-trip-time between intra-host chips only sub-nanoseconds, so coherency mechanism implementation is much easier and more efficient than inter-host communication.
- **ordering:** Intra-host communications are strictly ordered to keep the cache coherency. Inter-host communication may cause Out-Of-Order under multi-path scenario.
- **flit size:** coherency and other real-time communication scenario require small flit size, many intra-host protocols(eg. CXL) are designed to use single flit just carry one cacheline(64B). The flit size in inter-host protocol(eg. Ethernet) is much larger(1500B).

Directly extend *intra-host* protocol to *inter-host* (eg. RDMA)need carefully process the congestion control and coherence issue, however the host PCIe link and cache coherence processing may introduce high latency and unpredictable jitters. congestion control module may use more buffer to absorb jitters but it may cause more delay.

Directly extend *inter-host* protocol to *intra-host* (eg. NanoPU [6]) require processor attached to the Ethernet controller, packet directly

draft, Oct 03–05, 2021, Shanghai,China

ACM ISBN XXX-X-XXXX-XXXX-X/XX/XX…$15.00
https://doi.org/xx

send and operate at register level which may loss programming flexibility and increase complexity when application need to use other sophisticated processors.

Therefore, "Dam" is required at the barrier of host to provide unified memory access to absorb the bursts and smoothly transition the flit size, "batch-mode" with large flit size for *message passing interface* during inter-host communication, while small flit size to implement cache coherency for *shared memory mode*.

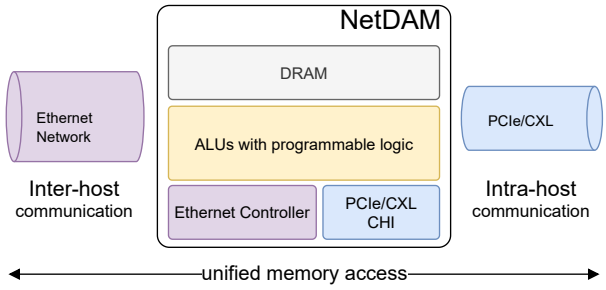

Figure 1: NetDAM function block

NetDAM is designed to bridge the *intra-host* and *inter-host* protocols by directly share memory with additional instruction level support for *in-memory* and *in-network* computing. With this architecture, CPU / Domain Specific Accelerator / other storage component could directly attach to netDAM via AXI or CHI or PCIe/CXL bus and share the unified memory pool.

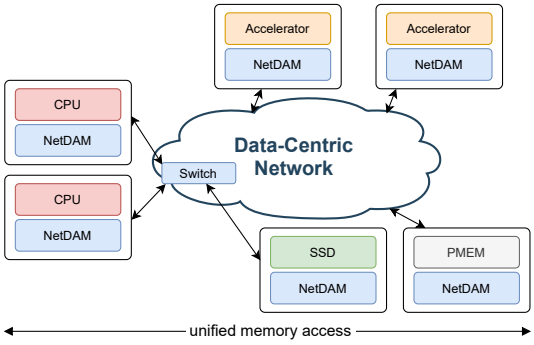

Figure 2: NetDAM interconnection system

Paper Organization: We present the system architecture in Section 2. We demonstrate MPI-allreduce case by add reduce-scatter and all-gather instruction based on NetDAM programmable ISA in Section 3.We conclude in Section 4.

2 SYSTEM ARCHITECTURE

In this section, we present the details about system architecture and design trade-off.

2.1 NetDAM packet format

NetDAM is a packet based protocol which combines the instructions and data.

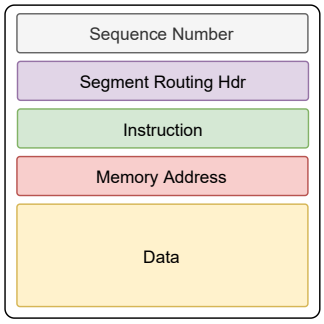

Figure 3: NetDAM packet format

- **Sequence:** is used for packet ordering and reliable transmit.
- **Segment Routing Header:** is used for leverage multi-path to avoid datacenter network congestion. Meanwhile, it could be used for dataflow computation. Many distributed systems use Directed acyclic graph(DAG) to abstract the computation job, Segment Routing Header could be a chaining function to processing packet on different node. Detailed information could be found in subsection 2.2.
- **Instruction:** NetDAM provide basic *READ*,*WRITE*,*CAS* and *MEMCOPY* instructions which operate in SIMD mode. NetDAM support programmable ISA extension, we reserve multiple bits in this field, user could define their own instructions for different computation jobs, detailed information could be found in subsection 2.3.
- **Address:** The instruction operate data memory address. IOMMU may implement on NetDAM for Virtual Address and Physical Address translation. Remote Memory could also mapping to local Virtual Address by this IOMMU, detailed information could be found in subsection 2.4.
- **Data:** has variable length based on instruction definition, many instruction could be operated in SIMD mode, especially for inter-host communication. Data length could be 9000B which means SIMD could leverage multiple ALUs on NetDAM to operate ≈ 2048 x float32 in parallel.

2.2 Transport Layer

NetDAM design principal is providing ultra-low latency and ultra-high bandwidth transport with necessary hardware dependencies, IP/UDP over Ethernet is used to carry NetDAM packet for inter-host communication.

Deterministic Latency: NetDAM has fixed pipeline to processing packet by eliminate PCIe DMA and bypass snoop for cache coherency. Packet could be ACK with deterministic latency, we test wire-to-wire SIMD read 32 x float32 data from DRAM, average latency is 618 nanoseconds, jitter is 39 nanoseconds, max latency is only 920 nanoseconds, which is much faster than RoCE. Latency is the key indicator for congestion control (eg. SWIFT [8]), We need to emphasize isolate the intra-host DMA to implement inter-host

deterministic latency communication is very useful to simplify the congestion control algorithm.

Reliable Transmit: is optional. One reason is lossless Ethernet with virtualization or container overlay support will introduce significant overhead. Another reason is many distribute applications could design *idempotent interface*, simply re-transmit does not impact the result, *atomic instruction* could be added to implement *idempotent operators*.

Relax Order: Commutative operations allows Out-Of-Order execution, especially we have memory address field in each packet to isolate operating memory space. However we provide sequence field in the packet, user could add optional reorder module in programming logic for ordering execution.

Multi-Path: Many datacenter network topology use fat-tree while some HPC cluster use 2D-Torus 3D-Torus. NetDAM design Segment Routing Header in UDP(SROU [1]) enable topology independent transport, source node could select dedicated path to avoid switch buffer overrun and fully utilize the fabric bandwidth. function callback could add in segment routing stack for chaining computations over multiple node, we will show a ring-allreduce demo in section3.

2.3 Programmable ISA

NetDAM instruction is more like a RPC(Remote Procedure Call) for software friendly development, it has dedicated memory space for Request and Complete Command Queue pairs. software could simply write the NetDAM packet to Request Queue memory address, and fetch from Complete Queue. For the inter-host communication case, software could simply use UDP socket send NetDAM packet to NetDAM device.

NetDAM "template" only define some basic memory access instructions(WRITE, READ, MEMCOPY, Atomic) and leaves multiple bits in instruction field for user defined behavior.

For the neural network case, user may define SIMD(ADD, SUB, MUL, XOR, MIN, MAX) and compute them directly near the memory, on-chip ALUs will be used for accelerate computation, HBM-PIM [12] could be attached to NetDAM for in-memory acceleration in the future.

This architecture also allows user defined your own circuit logic to build DSA IPCore and directly connect to NetDAM via AXI bus for adaptive computing. For MPI all-reduce communication case, Reduce-Scatter and All-Gather instruction could be added. For DPU offload case, compress, crypto, hash and longest prefix match instruction could be added.

2.4 Memory Addressing

Each NetDAM device has its own memory address space and mapping to intra-host network, additional IOMMU could be added for virtualization support and mapping remote memory to local host.

A special address pool could be used for NetDAM pkt Request Queue and Complete Queue.

NetDAM directed attached DRAM memory address could be mapping to host user-space(also support the VM/Container deployment), Shared Memory Packet Interface (memif) [2] could be implement on NetDAM for general Ethernet packet.

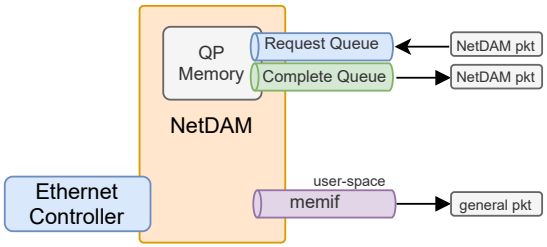

Figure 4: NetDAM memory interface

NetDAM device could be operated in standalone mode and directly attached to switch, multiple NetDAM device with switch construct a big memory pool with multi-terabytes memory capacity with multi-terabits bandwidth.

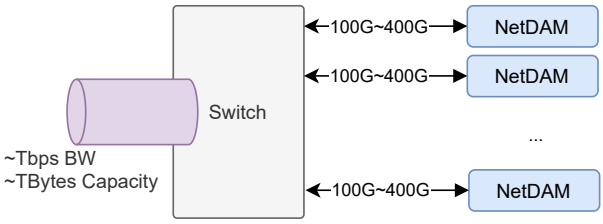

Figure 5: NetDAM memory pool

Each NetDAM could implement a local IOMMU to translate Global Virtual Address to NetDAM device IP address with NetDAM Local Address. Such IOMMU function could be implement on programmable switch, Virtual IP address cloud be configured on switch and provide address translation to NetDAM device.

Incast Avoidance The global memory pool could be operated in block interleaved mode, Thus many-to-one communication could be equally load balance to multiple NetDAM device, the receiving host could pull them back from global memory pool based sequencing and rate-limited READ command, the incast problem can be easily avoid without complex congestion control mechanism.

2.5 Security

Security problem is a concern when sharing memory and executing instruction remotely, SDN controller could act as a MMU to simply apply malloc/free request and translate request to access-control-list and apply to each NetDAM or in datacenter switch. *encryption-write* and *decryption-read* instruction could be added for secure computing.

3 MPI ALLREDUCE

MPI Allreduce communication is the key bottleneck in Distributed AI training platform. We implement MPI-Allreduce instruction on NetDAM and demonstrate it could be much faster than RoCE.

Ring-Allreduce[4] is a high efficient all-reduce algorithm designed by Baidu research, then widely used in Horovod library. The collective communication is based on ring topology which may fully utilize the network bandwidth and mitigate congestion.

We implement **Ring Reduce-Scatter** and **Ring All-gather** instruction on NetDAM, The prototype use 2 x Xilinx Alveo U55N FPGA with build-in HBM, Each blade implement 2 independent NetDAM device. Each NetDAM device has one 100G Ethernet Port with 2GB HBM memory. All 4 NetDAM device connect to a Cisco Nexus 93180FX switch.

3.1 Ring Reduce-Scatter

Ring Reduce-Scatter is a chaining function to add the parameter node by node. Node1 send *A1* to Node2, Node2 calculate *A1+B1* then send it to Node3, Node3 send *A1+B1+C1* to Node4. Node4 write *A1 + B1 + C1 + D1* in local memory.

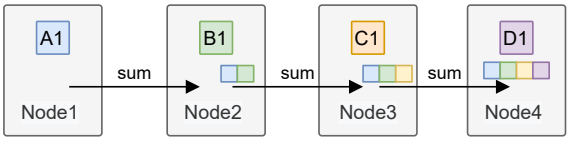

Figure 6: Ring Reduce-Scatter

RoCEv2 based implementation require multiple DMA and multiple load/store on CPU, the temporary sum A1 + B1 may require separated memory space to store to avoid side-effect. The sum operation need explicit instruction in each iterations, an explicit synchronization barrier required between iterations which may introduce more latency.

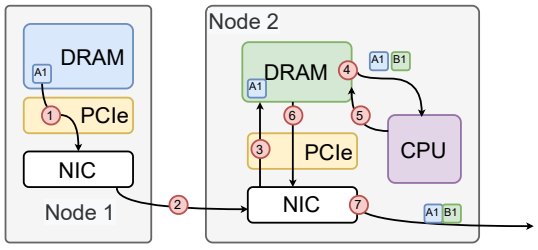

Figure 7: RoCE Reduce-Scatter

NetDAM implementation is much faster than RoCE, A1 directly fetch from DRAM on Node1 and send to Node2 packet buffer SRAM, Node 2 execute the instruction in ALU to add B1 and store the sum result in SRAM, then based on Segment Routing header in SRAM to self-route the packet to Node 3. Traditional CPU may only has AVX512 instruction support, each cycle may only support 32x float32 value add operation. NetDAM could leverage directly memory access and implement multiple ALUs to support 2048 x float32 add operation with single instruction.

Idempotent is important for error handling in distributed system, the reduce-scatter operator is idempotent in interim node(Node2 and Node3), because all computation is based on packet buffer modification, no side-effect on local memory, but the last node need to write to local memory which is not idempotent. we defined a block based hash algorithm to keep the last hop idempotent. *block-hash* instruction added to calculate block-hash, each blocks may contains 2048 x float32 data. The *Reduce-Scatter* could carry this hash value, when last hop local memory hash is same with the instruction, NetDAM write the packet data to local memory else drop the packet.

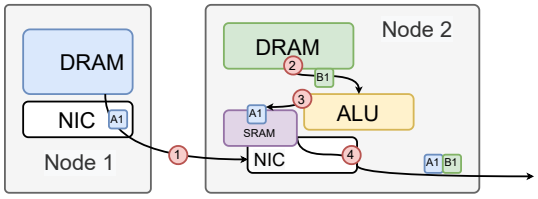

Figure 8: NetDAM Reduce-Scatter

3.2 Ring All-Gather

Ring All-Gather is simpler than Reduce-Scatter, it only require write the packet payload to each NetDAM device local memory, then use segment routing header route to next node, finally an ACK generated to send to the controller. Multicast/Broadcast may implement in the future to accelerate the all-gather procedure.

3.3 Evaluation

We build a 4xNodes RoCEv2 Platform to compare with 4xNodes NetDAM. RoCEv2 platform use Mellnanox CX516A NIC, 2x Intel Xeon Gold 6230R CPU, 12x DDR4 32G@2933Mhz DRAM. Mellanox HPC-X and OFED installed.

We execute 536,870,912 x float32 allreduce, the native MPI Allreduce takes 2.8 seconds, the ring-based allreduce use 2.1 seconds. NetDAM will only takes xxx microseconds.

4 CONCLUSION

We re-examine the intra-host communication bus(eg, PCIe, CXL, AXI, CHI..) and inter-host Ethernet network, there are significant differences in topology, latency, loss tolerance, coherency, ordering and flit size. We conclude that directly use one protocol for both communication network or use overlay technology does not efficient.

Based on our decade experience in Cisco Quantum Flow Processor [13], and from software/hardware programmable friendly perspective, we conclude directly attach memory to Ethernet controller and provide programmable instruction level memory access is the better way to isolate the intra-host and inter-host network, it could be smoothly transition the flit size, "batch-mode" with large flit size for *message passing interface* during inter-host communication, while small flit size to implement cache coherency for *shared memory mode*.

We also leverage the Segment Routing concept to support multi-path loadsharing in datacenter network to reduce congestion, and

the global memory pool with block interleaved addressing can be used to avoid incast problem.

We build a prototype on FPGA to verify this idea, memory access latency and jitter is much lower than RoCE. We demonstrate the programmable ISA based on MPI Allreduce case which is critical for hyper-scale AI training fabric, by simply adding few domain specific instruction, it shows incredible performance improvement than RoCE, we will add dynamic sparse matrix compression and implement in-memory optimizer in the future.

In the future, once we get the CXL based FPGA and CPU, we will implement high speed intra-host communication stack (eg. memif).

NetDAM architecture is not only for computation and memory acceleration, but also could accelerate storage network including NVMeOF and Persistent Memory support. Even more, cloud service provider(CSP) can use this infrastructure for serverless computing. We hope our results will encourage other researchers to push these ideas further.

REFERENCES

[1] Kevin Fang. 2020. Ruta. (2020). https://tools.ietf.org/html/draft-zartbot-sr-udp-00

[2] FD.io. 2021. Shared Memory Packet Interface (memif) Library. (2021). https://docs.fd.io/vpp/20.05/dc/dea/libmemif_doc.html

[3] fungible. 2018. https://www.fungible.com/. (2018).

[4] Andrew Gibiansky. 2017. Bringing HPC Techniques to Deep Learning. (2017). https://andrew.gibiansky.com/blog/machine-learning/baidu-allreduce/

[5] Mark Handley, Costin Raiciu, Alexandru Agache, Andrei Voinescu, Andrew W. Moore, Gianni Antichi, and Marcin Wójcik. 2017. Re-Architecting Datacenter Networks and Stacks for Low Latency and High Performance. , 14 pages. https://doi.org/10.1145/3098822.3098825

[6] Stephen Ibanez, Alex Mallery, Serhat Arslan, Theo Jepsen, Muhammad Shahbaz, Changhoon Kim, and Nick McKeown. 2021. The nanoPU: A Nanosecond Network Stack for Datacenters. (2021), 239–256. https://www.usenix.org/conference/osdi21/presentation/ibanez

[7] IWARP. 2018. https://en.wikipedia.org/wiki/IWARP. (2018).

[8] Gautam Kumar, Nandita Dukkipati, Keon Jang, Hassan M. G. Wassel, Xian Wu, Behnam Montazeri, Yaogong Wang, Kevin Springborn, Christopher Alfeld, Michael Ryan, David Wetherall, and Amin Vahdat. 2020. Swift: Delay is Simple and Effective for Congestion Control in the Datacenter. (2020), 514–528. https://doi.org/10.1145/3387514.3406591

[9] Yuliang Li, Rui Miao, Hongqiang Harry Liu, Yan Zhuang, Fei Feng, Lingbo Tang, Zheng Cao, Ming Zhang, Frank Kelly, Mohammad Alizadeh, and Minlan Yu. 2019. HPCC: High Precision Congestion Control. (2019), 44–58. https://doi.org/10.1145/3341302.3342085

[10] Rolf Neugebauer, Gianni Antichi, José Fernando Zazo, Yury Audzevich, Sergio López-Buedo, and Andrew W. Moore. 2018. Understanding PCIe Performance for End Host Networking. (2018), 327–341. https://doi.org/10.1145/3230543.3230560

[11] RDMA. 2021. wikipedia. https://en.wikipedia.org/wiki/Remote_direct_memory_access. (2021).

[12] Samsung. 2021. Processing in Memory. (2021). https://www.samsung.com/semiconductor/solutions/technology/processing-in-memory

[13] James Markevitch Will Eatherton, Don Steiss. 2008. Cisco Quantum Flow Processor. (2008). https://old.hotchips.org/wp-content/uploads/hc_archives/hc20/3_Tues/HC20.26.720.pdf

[14] Yibo Zhu, Haggai Eran, Daniel Firestone, Chuanxiong Guo, Marina Lipshteyn, Yehonatan Liron, Jitendra Padhye, Shachar Raindel, Mohamad Haj Yahia, and Ming Zhang. 2015. Congestion control for large-scale RDMA deployments. *ACM SIGCOMM Computer Communication Review* 45, 4 (2015), 523–536.