Reliable and Flexible Large Scale Memory Network

Zixuan Wang*, Xiao Liu*, Jongryool Kim[†], Hokyoon Lee[†], Jishen Zhao*

**University of California, San Diego* [†]*SOLAB, Memory System R&D, SK hynix, Korea*
**{zxwang, x1liu, jzhao}@eng.ucsd.edu,* [†]*{jongryool.com, hokyoon.lee}@sk.com*

ABSTRACT

The demand for server memory capacity and performance is rapidly increasing, due to the expanding working set size of modern applications. One promising technique to tackle these requirements is memory network, where a server memory system consists of multiple 3D die-stacked DRAMs and provides terabytes of memory. Previous works focus on DRAM-based memory networks. However, with new non-volatile memory technologies, such as Intel and Micron's 3D-Xpoint [1], we can improve the reliability of memory network in a more flexible way.

1. INTRODUCTION

Modern heavy workloads, such as big data analytics, in memory computing, are demanding significantly larger main memory than before. One way to meet this requirement is to add more CPU sockets to maintain additional memory channels [2]; yet another solution is to employ memory network, where multiple 3D die-stacked DRAM are interconnected and form a disaggregated memory pool, and shared by fewer CPU sockets.

Memory network has gained increasing attention from both academia and industry, yet the previous work mostly focuses on the DRAM-based networks. With the introduction of modern non-volatile RAM (NVRAM), we can easily achieve larger memory capacity, while improving the memory network reliability in a more flexible manner.

2. PRELIMINARY WORK

The prior work [3] String Figure designs a high performance, scalable, and flexible memory network architecture, which supports over 1,000 memory nodes shared by processors in a cloud server.

First, we design an easy-to-implement random network topology construction algorithm inspired by S2 [4]. This algorithm is used for initial construction or reconfiguration, and ensures the output topology is uniformly random and balanced. This algorithm also generates the shortcuts for each node; the shortcuts will be used to forward packets when the node is being removed from memory network.

To achieve high throughput, the routing protocol needs to be scalable, deadlock-free, while fully utilizing the bandwidth of topology. In conventional routing mechanisms on random topologies, each routing table contains k-shortest path routing with global information and size up to $O(NlogN)$ with N memory nodes. To maintain sub-linear increase of routing overhead, we adopt a hybrid compute+table routing protocol, where we divide the network into many sub-networks, and employ a greediest routing on the sub-networks. We also employ an adaptive routing to leverage the overhead of subnetwork crossing.

We design the dynamic and static network reconfiguration mechanisms in the String Figure. The dynamic reconfiguration automatically turns off the idle memory nodes to save power.And static reconfiguration adjusts the number of physical memory nodes in current network.

3. DESIGN OVERVIEW

In String Figure, we focus on flexibility and scalability of DRAM-based memory network. But we observe that modern NVRAM can provide much larger capacity [5] while maintaining non-volatile data. However, incorporating NVRAM with DRAM in memory network is challenging because of the performance and reliability gaps between DRAM and NVRAM. In this paper, we propose a reliable, flexible and scalable memory network design that incorporates NVRAM and DRAM nodes.

3.1 Flexibility

3.1.1 Network Topology Construction

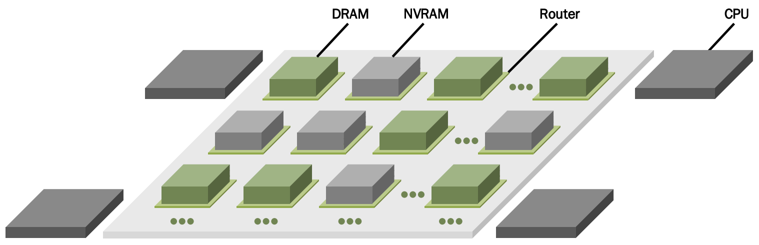
Prior studies in data center networks proved that sufficiently random regular graphs achieve throughput within several percents to the upper bound , at the scale of several thousand nodes. Due to the constraint of routing latency and router's capacity, directly adopting these random topologies in memory network can impose prohibitive scaling issues. In addition, the reliability of memory nodes varies among different types of memory (e.g. DRAM and NVRAM), across different workloads (e.g. hard faults accumulate over the memory device lifetime, the more stressed used memory nodes have higher hard error rates).

To address these challenges, we propose a weight-balanced random topology generation algorithm, which focuses on *a*) Randomness: ensure the generated networks are uniformly random; *b*) Balance: ensure the weight-balanced connections, which takes reliability into account. Our approach consists of following steps:

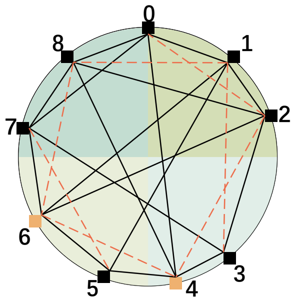
1. Calculation of the adaptive weight w for each memory node, e.g. for node N_i, the adaptive weight

$$w_i = \frac{1}{T_{ACC_i} \times (1 + k \times FIT_i)}$$

Where *a*) T_{ACC_i} represents the average access time of N_i, measured at manufacturing period; *b*) fit_i stands for



(a)



(b)

Figure 1: An example of memory network design. (a) The design includes heterogeneous memory nodes, which are interconnected and shared by multiple CPU sockets, on a single die. (b) An example of adaptive topology.

failures in time, measured at runtime by error detector (Section 3.2), k is a constant determined by reconfiguration policies.

2. Constructing L virtual spaces where each router has p ports, $L = \left\lfloor \frac{p}{2} \right\rfloor$, as each pair of ports are connected to one virtual space.
3. Distributing all the memory nodes in each virtual space with a random order, by employing the adaptive weighted distance. E.g. the adaptive distance between N_i and N_j is

$$D_{ij} = min\left(\left|x_i - x_j\right| \times w_i, 1 - \left|x_j - x_i\right| \times w_j\right)$$

4. Interconnecting the adjacent memory nodes in each virtual space respectively.
5. Interconnecting pairs of memory nodes with free ports.

Figure 1b shows an example topology generated by this algorithm. The solid lines are direct routing paths, and dashed lines represent shortcuts. Node 4 and 6 are NVRAMs with different weight from DRAM nodes.

3.1.2 *Network Reconfiguration*

For dynamic reconfiguration, we turn off the idle memory nodes by powering off the router when the system is running. When a node is sleeping, all packets to this node will be forwarded by shortcuts, which is an extra pair of ports of the switch. By adopting shortcuts, we maintain high throughput without waking up the node.

For static reconfiguration, where users need to physically adjust the memory nodes, we allow the memory network topology reconstruction after system boot up without changing the wire or fabric. Adding new memory nodes is challenging, which requires a larger routing table for each router. We suggest the producer leave free slots for further upgrade, which naturally expands the routing tables at manufacturing time.

For reliability reconfiguration when the system is running or rebooting, we allow fast access to the nodes with high FIT, to handle the errors faster; or migrate defective node's data to an idle node and perform a normal reconfiguration. As our topology construction algorithm allows an arbitrary number of nodes, losing a node does not affect the reconfiguration.

3.2 Reliability

One critical challenge in memory network with hybrid memory technologies is to ensure the reliability at a low performance cost. To address this challenge, we propose an adaptive fault tolerant mechanism:

For each DRAM node, we periodically write its snapshot with program state to the the NVRAM with lowest adaptive distance. When a node encounters errors:

1. The system first detects if its backup snapshot is outdated: *a*) If it's outdated, the system recovers the snapshot with corresponding program state from NVRAM, then roll back to this point and resume execution. *b*) If the backup is up to date, then system recovers this snapshot to DRAM, and resume the program.
2. After each recovery, the memory network updates the defective DRAM node's FIT.

We employ the NVRAM that's nearest to DRAM, as a backup node. By doing this, the network uniformly distributes the DRAM snapshots to multiple NVRAMs.

According to the performance gap between DRAM and NVRAM, the system may arrange multiple DRAM nodes as caches of an NVRAM node. When this NVRAM generates an error, the system can write back its correlated DRAM cache block, or perform an error correction on it if this part is not cached. Again, the network updates this NVRAM's FIT after recovery.

In this recover mechanism, each node's weight changes dynamically. If a node's FIT is higher than the threshold predefined by maintainer, its data will be migrated to an idle node, and it will be powered off. If the FIT is acceptable, then during the next reconfiguration, this node will be rearranged in the topology, to allow faster access from CPU and therefore reduce the error recovery overhead.

4. REFERENCES

[1] Micron Technology, Inc., *3D XPoint Technology*, 2018.

[2] J. Barr, *Now Available, Amazon EC2 High Memory Instances with 6, 9, and 12 TB of Memory, Perfect for SAP HANA*, 2018.

[3] M. A. Ogleari, Y. Yu, C. Qian, E. L. Miller, and J. Zhao, "String figure: A scalable and elastic memory network architecture," in *High Performance Computer Architecture (HPCA), 2019 IEEE International Symposium on*, IEEE, 2019.

[4] Y. Yu and C. Qian, "Space shuffle: A scalable, flexible, and high-performance data center network," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 11, pp. 3351–3365, 2016.

[5] Intel Coporation and SAP, *Persistent Memory and SAP HANA Database Start Times*.