White Paper

**Memory Class Storage for AI**

# Getting Back Up; Coupling AI and Memory Class Storage Saves in a Big Way

## By Bill Gervasi

**Abstract**: The hockey stick in adoption of artificial intelligence applications is well under way and custom architectures including CPUs, GPUs, FPGAs, and Neural Network Processors are claiming their place in the cloud. One of the weaknesses of the massively parallel architectures for AI is sensitivity to data loss on power failure, resulting in high costs for these data centers. The world's only Memory Class Storage architecture, Nantero NRAM™ provides a high performance persistent memory for AI applications, enabling a new level of uptime, lower power, and higher profits for data centers.

**About the Author:** Mr. Gervasi is Principal Systems Architect at Nantero, Inc. He has been working with memory devices and subsystems since 1Kb DRAM and EPROM were the leading edge of technology. He has been a JEDEC chairman since 1996 and responsible for key introductions including DDR SDRAM, the integrated Registering Clock Driver and RDIMM architecture, the formation of the JEDEC committee on SSDs, and actively involved in the definition of NVDIMM protocols.

# Introduction

The rapid rise in the adoption of artificial intelligence as well as deep learning algorithms is changing the profile of modern data centers. Applications including voice recognition, augmented reality, medical data mining, and thousands of others and are capturing more and more data cycles as the cloud becomes the universal resource for information. The simplistic term AI as I use it here implies the collection of many types of applications, some with wildly different technical requirements.

The cloud is a major source of revenue for most businesses as well, and uptime has become a dominant requirement. Losses from system downtime are pegged at an average of $22,000 per minute, and with an average of 14.1 hours of downtime annually, an average annual loss of $18M. Companies wholly dependent on the internet could easily see multiples of that overhead cost.

Clearly there is a rapid return on investment for a solution that minimizes down time, and speeds recovery from power failures. Introducing Memory Class Storage, a term defined below, into the data centers and especially into AI processor subsystems is a key performance enhancement that increases the value of a data center.

# Common AI Applications

AI is not new, though mass deployment of AI is a recent trend. There is no one architecture for an AI processing center, however a number of common trends have emerged in the deployment of AI. Generally speaking, AI uses a large array of relatively simple processing units operating in parallel on a very wide data set, with each processing element operating on a subset of the data.

To help visualize how this would work, imagine a JPEG picture processing algorithm with a small processing unit running the discrete cosine transform algorithm (DCT) for an 8x8 array of pixels. Since typical raw pictures are far greater than 8x8 pixels, an AI engine can include many processing units operating in parallel, each running the same DCT algorithm, and then have one master controller that feeds 8x8 parts of the overall picture to the processing units. The master controller feeds the results from each processing unit back into an output cache for the results, such as the data for an encoded JPEG file.

These parallel processing elements may also choose to share data between them, for example to manage the edges between adjacent input data blocks. One approach to this solution is the Recurrent Neural Network (RNN) as shown in Figure 1.

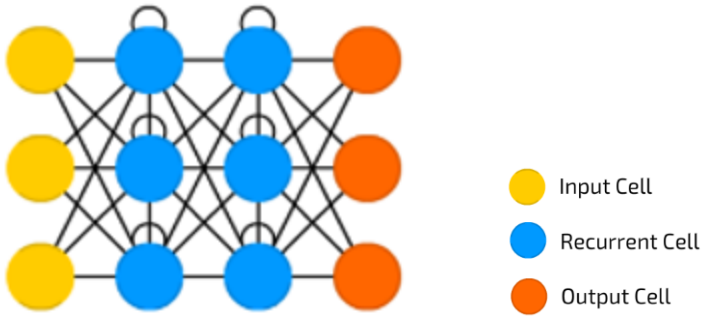


*Figure 1: Recurrent Neural Network Processing*

Relating the use of AI to the cost of power loss, the simple RNN application above shows the inherent overhead when the input data in yellow and the output data in orange is stored in dynamic memory. On data loss, the input data must be reloaded and the RNN algorithms rerun to restore the output data.

Deep learning is another variation of AI that exhibits greater sensitivity to data throughput. Deep learning algorithms load a seed pattern into the AI memory that represents a pre-calculated estimate of expected possible answers. Examples of deep learning include voice recognition, where the seed data may represent generic voice patterns. However, deep learning can modify the seed pattern once input data starts streaming past the processing units. With voice recognition, this could include tuning the recognition engine for a specific user's accent. Deep Convolutional Networks are one mechanism for adding recognition, context, and modified seeds to enhance the quality and accuracy of the algorithm.

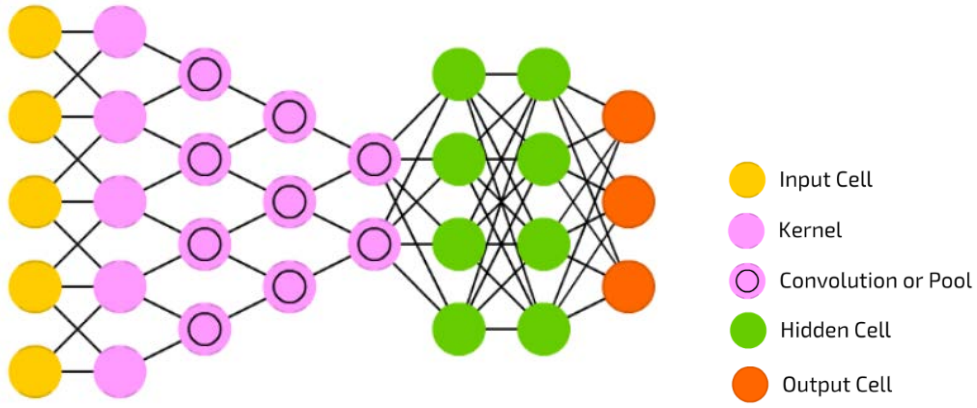


*Figure 2: Deep Convolutional Network Processing*

One challenge for deep learning algorithms is to avoid retraining. Imagine the reaction of the customer who had to retrain a voice recognition device every time they moved the device to a different room. For deep learning, the modified seed models need to be checkpointed periodically so that the learned data may be restored after power failure. In Figure 2, the green circles represent these models, and if this data is lost, any updates to the model must be relearned. The pooling layers, in pink double circles, also contain information to accelerate the processing and must be saved or reconstructed.

# Common AI Architectures

As stated previously, there are trends in the industry regarding hardware implementations of AI architectures. These trends are: 1) feeding large amounts of data into a wide array of processing elements, 2) each processor operating on a relatively simple algorithm to 3) produce output data which is recombined into a desired result. 4) For deep learning, these intermediate results can modify an initial input data set which needs to be extracted periodically.

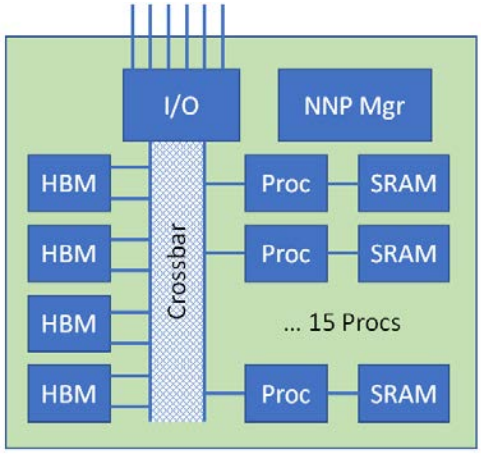


*Figure 3: Example AI Processor*

A fairly typical AI device is shown in Figure 3 wherein an array of 15 processing elements are tied through an internal memory crossbar to a large data resource (e.g., 32 GB) using high bandwidth memory. HBM provides a 1024-bit wide interface running at 2.4 Gbps per data pin, and with four internal HBM buses the processing elements can be fed at a combined throughput of roughly 1.2 TB/s.

In addition, each AI device provides a number of high speed serial I/O interfaces (e.g., 1 Tbps) to expand the architecture to large numbers of processors. These serial buses are interconnected in a somewhat application specific pattern, often a hypercube or toroid arrangement, so that data can be distributed through the array for processing in parallel by many AI devices. Figure 4 is an example of a toroid connection scheme to route data through the array.

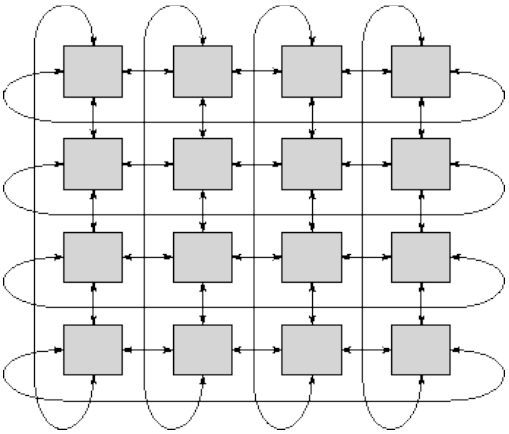

*Figure 4: Toroid Connection of Multiple AI Devices*

At 1 Tbps, these serial links are reasonably fast, but the number of links are still limited and data must make several hops through the array to reach the destination. Even at fairly high bus efficiency, it can take a multiple seconds to fill the HBM content of each AI device. One solution would be to increase the number of serial input feeds from the supporting hardware, but at 26 GB/s for a full DDR4 interface, it takes four channels of DDR4 to feed one input pipe, so the external support hardware quickly becomes the bottleneck.

Checkpointing the contents of modified models in deep learning applications also takes place over these serial buses. Each AI device needs to communicate its contents to the support hardware for commitment to non-volatile resources, typically NVMe or SSD units. This checkpointing consumes valuable bandwidth on the communications links, time that could be better spent on data processing.

# The Cost of Power Failure

The bottleneck caused by the serial links highlights the problem faced by power failure. Once the models are lost, refilling a large array of AI devices can take a very long time. Reloading the applications into the execution units takes time as well, then restarting the algorithms and restoring lost work in progress can take a significant amount time as well. Meanwhile, the system is unavailable for end user data processing.

For deep learning environments, data loss may not be acceptable so complex mechanisms must be put in place to periodically checkpoint learned modification of seeded data models.

Depending on the system architecture, the speed of reloading will be limited by the non-volatile storage interface. Even with 10 GB/s NVMe backup, restoration of a large AI array could take many minutes. If this bottleneck defines the granularity of saved deep learning models, the likelihood of permanently lost data is assured. To avoid this data loss, uninterruptible backup systems are typically deployed, however the power requirements for these architectures tends to be quite high and the UPS relatively expensive.

# Memory Class Storage in AI

Nantero NRAM™ is a non-volatile memory technology that operates at full DRAM speed and with unlimited write endurance, hence the expression Memory Class Storage. NRAM uses electrostatic forces to switch arrays of carbon nanotubes for the storage of 1s and 0s. Coupling NRAM with an HBM interface, NRAM provides a great solution to the problem of power fail sensitivity of AI architectures.

The NRAM HBM drops into the AI device exactly like a DRAM HBM. It uses the same signals and timing, making integration exceptionally simple. Unlike Storage Class Memories, which are much slower than DRAM with very limited endurance and therefore still require support DRAM to achieve the desired performance, NRAM completely replaces all DRAM in the AI device with no additional support required. Since NRAM is inherently non-volatile, the AI controllers can also exploit its unique features, for example to turn off refresh and gain an additional 15% performance at the same clock frequency.

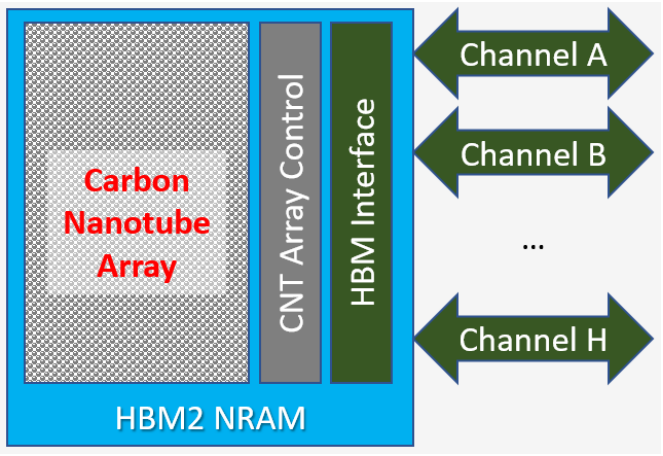


*Figure 5: HBM NRAM Architecture*

On power failure, the system sends a single signal to all AI devices warning of impending power failure. Each AI device responds by completing calculations in process and storing the results into the non-volatile NRAM, then shutting down. When power is restored, the execution unit code is restored from the NRAM to the local memory (such as SRAM) attached to each execution unit, temporary results are restored, and execution resumes in microseconds instead of minutes.

Since data is never lost, and the external serial links are not needed nor are external backup mechanisms such as NVMe or SSD, the incorporation of NRAM HBM into an AI architecture eliminates the need for expensive battery backup systems.

# Conclusion

In summary, use of NRAM HBM reimagines the computing infrastructure for artificial intelligence and deep learning applications. By providing inherent data persistence at all times, AI servers need not take the long delays associated with reloading models and other data. Checkpointing of modified model data is automatic, and does not consume bandwidth on the interconnects between processing elements and the support computing system.

Incorporating NRAM HBM into an AI architecture eliminates the cost, complexity, and reliability concerns of battery backup systems, and reduces power in the data center, while increasing performance. Backup and restore procedures turn mostly into NOPs… just turn back on and go!

Stay tuned… a carbon nanotube SRAM is in the future as well… we've only just begun.

--
bg