International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 08 Issue: 07 | July - 2024 SJIF Rating: 8.448 ISSN: 2582-3930

AI Hardware Resource Monitoring in the Data Center Environment

Nanduri Vijaya Saradhi (*Author*)
Solution Architect, DXC Technology
Building # 4, Mindspace IT Park,
Madhapur, Hyderabad, Telangana 500081, India
Email: vnanduri@dxc.com

1.0. Executive Summary

Deploying an AI (Artificial Intelligence) model in the data center initiates more responsibilities to the backend services such as Monitoring. It is required to monitor the performance of AI systems regularly to ensure that they meet the requirements and will not encounter any system performance issues.

This whitepaper focuses on the importance of monitoring AI systems, the monitoring model, how to measure the performance of the system hardware resources such as CPU, Memory, disk and GPU, and tools to be used to monitor the system resources. Organisations can take necessary proactive maintenance actions before an incident is caused due to performance bottlenecks in the AI systems, proving the importance of monitoring the AI system.

The goal of continuous monitoring of AI systems is to ensure the effective operation of AI systems throughout their lifecycle to meet several objectives such as performance, anomaly detection, security monitoring, data compliance and continuous improvements.

Performance measurement of critical resources such as GPU, Memory and Storage by using suitable tools and configuring the alerts when the thresholds are reached on the identified resource threads. These measurements will be utilized to strengthen the AI system that will be stable for any performance bottlenecks.

2.0. Importance of AI Systems Monitoring

AI models require high computing and high-performance resources such as GPU (Graphical Processing Unit), Memory, and Storage. Real-time monitoring of these computational resources ensures the AI services are continuously running without any performance bottlenecks and outages. By continuously monitoring the AI system, Organizations can detect issues early before they escalate into larger problems. This proactive approach reduces downtime, improves reliability, and enhances user satisfaction.

Monitoring the hardware resource and its power consumption helps to reduce environmental emissions. The power efficiency can be optimized by monitoring the GPU usage that is used extensively in AI systems for their parallel processing capabilities. By identifying resource-intensive tasks or inefficient algorithms, the overall power consumption can be optimized leading to a greener environmental footprint.

3.0. Monitoring Model for AI System Resources

Monitoring the AI system resource components such as CPU, Memory, Storage and GPUs is straight forward solution by using System monitoring tools. Monitoring the AI model requires mainly four components.

- AI Server Pool, containing hardware resources to be monitored
- Monitoring tool

- Performance statistics for end-user analysis
- Automated alert generated by the ticketing tool

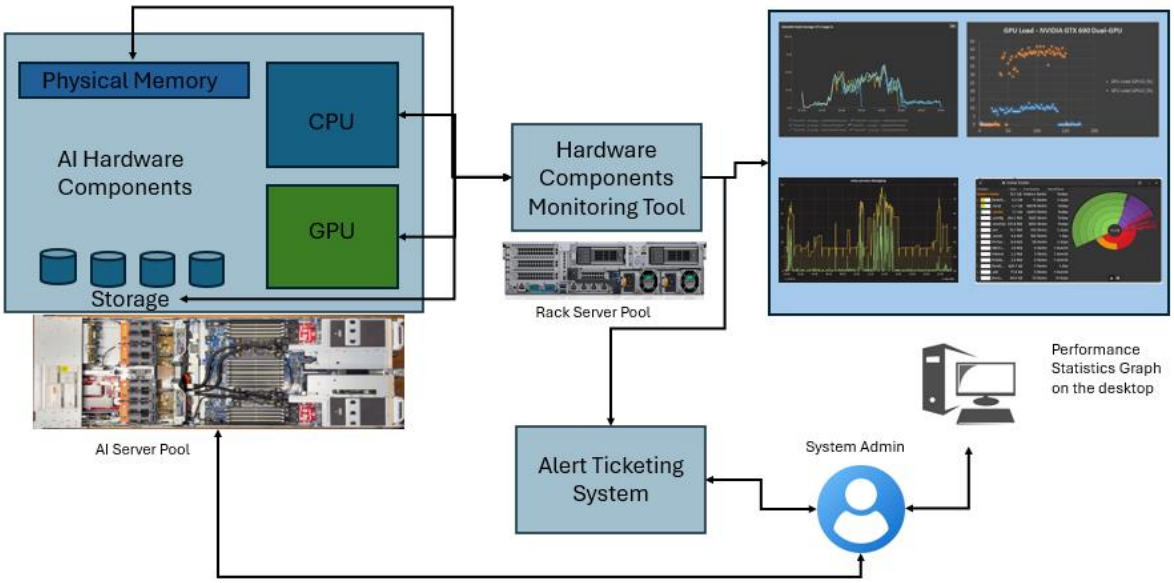

Figure 1:Monitoring model for AI system resources

These monitoring tools contain agents installed on the AI system and are part of the standard SOE (Standard Operating Environment) component. Some of the monitoring tools are agentless and directly monitor the resources. The monitoring patterns are displayed in the frontend tool where the system admin can directly monitor and analyse the real-time data. The monitoring tool can also generate an alert if the specific counter reaches its threshold. The tool even generates a ticket for the support teams.

4.0. Performance measurement of CPU, Memory, GPU and Storage

Any computational system contains the basic hardware components such as CPU, Memory and Storage. In AI systems, GPUs are the main computational system resource that performs multiple and simultaneous computations and move large amounts of data between CPU and GPU.

The following are the important performance counters to be considered while monitoring the hardware resources for an AI system.

4.1. CPU

AI chips are important components in AI hardware systems due to their ability to efficiently handle the computational demands of AI algorithms. . These chips are made for specific purposes such as "Graphics Processing Units (GPUs) for parallel processing, Field-Programmable Gate Arrays (FPGAs) for customizable programmed chips, Application-Specific Integrated Circuits (ASICs) are custom-designed for specific applications and Neural Processing Units (NPUs) are designed to handle neural network computations efficiently.

The following is the list of the new AI chips available in the market.

Vendor	Selected AI Chip
NVIDIA	GH200
Intel	Gaudi 3
AMD	MI350
Apple	M4
Meta	Artemis
Microsoft Azure	Maia 100
AWS	Trainum2
IBM	NorthPole

The following is a list of some of the important performance counters that need to be considered while monitoring the CPUs used in AI hardware systems.

4.1.1. Instructions per Cycle

This counter measures the number of instructions a processor can execute for each cycle of its clock. This indicates the CPU's efficiency and the higher value indicates that the CPU is capable of handling more operations per clock. This value helps in identifying the bottlenecks in the CPU's execution pipeline and can guide optimization efforts.

4.1.2. Branch Miss Predictions

This counter measures the number of times the processor's branch prediction unit incorrectly guesses the outcome of a branch instruction. The branch prediction is a mechanism that attempts to guess the direction of branch instructions before they are computed to maintain efficient instruction pipeline flow. The speculative execution guesses the path of a program and executes instructions ahead of time which significantly improves the performance. High-rate value of branch miss predictions can lead to performance bottlenecks.

4.1.3. % Processor Time

This is one of the important and primary counters to be enabled while monitoring the CPU in the monitoring toolset. It is defined as the percentage of elapsed time that the processor spends to execute a non-Idle thread. It measures the percentage of time the processor is busy executing non-idle threads. High utilization of this counter indicates that the CPU is a bottleneck.

4.1.4. Process% Processor Time

This counter is particularly useful for identifying processes that are using a high amount of CPU resources. This counter measures the percentage of elapsed time that all threads of a process use the processor to execute instructions. An instruction is the basic unit of execution in the system, and this counter reflects the total time the CPU was actively executing the process's threads.

4.1.5. System\Context Switches/sec

This counter measures the combined rate at which all processors on the computer are switched from one thread to another thread. It's a system-wide metric that indicates how often the operating system had to switch the CPU from executing one thread to another. A high rate of context switches can indicate several things such as High contention, Inefficient threading, and Processor saturation.

4.1.6. Processor (_Total)\Interrupts/sec)

This counter measures the rate at which the processor receives and responds to hardware interrupts. When an interrupt occurs, the processor pauses its current tasks, saves its state, and executes a function called an interrupt handler to handle the event. A high value of this counter indicates that a larger number of devices are making requests of the processor, which could be a sign of hardware issues, such as failing components or conflicts between devices.

4.1.7. Processor (_Total)% User Time

This counter represents the percentage of elapsed time the processor spends executing user-mode applications. This counter is a measure of the processor's workload that is being used for user processes. % User Time indicates the portion of the processor's time dedicated to running user-mode processes. _Total signifies that the measurement is an aggregate of all the CPU cores on the system.

4.1.8. Processor(_Total)% Privileged Time

This measures the percentage of time the processor spends executing in privileged (kernel) mode. This includes operations that require direct access to hardware and memory, such as I/O operations, disk access, and system services. % Privileged Time shows some part of the total CPU time that is spent on kernel mode operations. _Total indicates an aggregate across all CPU cores in the system.

4.1.9. Processor Queue Length

This counter measures the number of threads waiting in the processor queue to be executed. It reflects the number of ready threads that are not currently running but are queued up and waiting for a processor core to become available. The Processor Queue is where threads wait for their turn to be executed by the CPU. The Queue Length indicates the number of threads in the queue. A sustained processor queue length greater than two generally indicates processor congestion and could be a sign of a CPU bottleneck.

4.2. Memory

Memory is one of the critical hardware components of the AI system. There are multiple types of memory in AI systems.

On-Chip Memory (Cache Memory) – This memory is directly integrated into the processor used for storing temporary data that the processor needs immediate access to.

High-Bandwidth Memory – This memory is usually a RAM module in the hardware system.

GDDR Memory – Graphic double data rate memory is used in graphics cards and is used in AI hardware systems.

The following is the important list of performance counters that need to be considered while monitoring the Memory used in AI hardware systems.

4.2.1. Memory Core Frequency

This counter refers to the operating frequency of the memory cores. This counter is relevant in systems with GPUs or any AI accelerators that support dynamic voltage and frequency scaling (DVFS). This influences the rate at which data can be transferred between the memory and the processing units. The computational performance and energy consumption can be balanced by adjusting the core and memory frequency. This is very useful for deep learning workloads where the computational requirements are high.

4.2.2. Cache Misses

This counter measures the number of times the systems cache does not contain the requested data, forcing the CPU to fetch the data from a slower level of the memory hierarchy from the main memory. This event is called Cache Miss. Modern processors have multiple levels of cache such as L1, L2, and L3. If the value is larger, then significantly it affects the performance of AI applications because of an increase in the memory access latency.

4.2.3. GDDR (Graphics Double Data Rate) data rate

This counter measures the speed at which data can be transferred per pin on the GDDR memory interface. It measures the throughput capabilities of the memory in AI systems. The high data rates are essential to processing large datasets in AI and ML. The GDDR7 achieves 32 GT/s (giga-transfers per second) and is expected to reach up to 48 GT/s. The best performance of the AI system is dependent on this data rate as it directly implies moving data to and from the GPU.

4.2.4. \Memory\Available Mbytes

This performance counter measures the amount of physical memory, in megabytes, that is immediately available for allocation to a process. This metric assesses whether a system has enough memory or if it requires additional memory. If this value consistently drops below 10 percent of the total physical memory indicates the system is running low on available memory and could benefit from an upgrade.

4.2.5. \Memory\Page Faults/Se

This counter measures the rate at which page faults occur in the main memory. A page fault happens when a program requests data that is not currently in the system's main memory (RAM), prompting the system to retrieve the data from a hard disk or a shared memory area. There are two types of faults. The first one is hard page faults occur when the data is not in physical memory and must be retrieved from disk. This causes delays due to disk access. The second one is soft page faults occur when the data is already in physical memory but not in the expected location.

4.2.6. \Memory\Committed Bytes

This counter reflects the current state of memory resources, indicating how much memory is committed and just not allocated. It's a key indicator of the system's memory usage and can help identify memory-related performance issues. If the value of this counter is consistently high, it suggests that the system is running low on available memory resources.

4.2.7. \Memory\Commit Limit

This counter represents the total amount of virtual memory that the system can commit to all processes without needing to increase the size of the paging file. This limit is a combination of the physical RAM available and the size of the system's paging file. If the limit approaches or exceeds the commit limit, then it is an indication that the system running low on virtual memory.

4.2.8. \Memory\Pages/sec

This counter measures the rate at which pages are read from or written to disk to resolve hard page faults. Hard page faults occur when a process requests a page in memory, but the system cannot find its location in physical memory (RAM) and is required to retrieve it from the paging file that is located on the storage disk. The pages/sec reflects the sum of the rate at which pages are read into physical memory from the storage disk and the rate at which pages are

written to the storage disk from the physical memory. The high rate of this value suggests that the paging activity is high due to insufficient physical memory.

4.2.9. \Memory\Cache Bytes

This counter measures the size of the file system cache. This indicates the amount of physical memory used for caching file system data that helps improve the performance of file access operations. The higher the value of this counter means more data is being cached for faster operations which leads to more memory utilization that directly impacts on the system performance.

4.2.10. \Memory\Pool Paged Bytes

This counter measures the size, in bytes of the paged pool. The paged pool is the space of physical memory used by the operating system for objects that can be written to the storage disk when not being used. This used memory can be paged out to the paging file to the Storage disk. If the value of the paged pool is high indicates that there is not enough physical memory available in the system. If the value is too small indicates there is no sufficient memory available for system drivers and kernel which leads to system instability.

4.2.11. \Memory\Pool Nonpaged Bytes

This counter measures the size of the non-paged pool in bytes. This is the portion of system memory that is used by the kernel and device drivers. The memory in the non-paged pool contains data that must remain in the physical memory and cannot be written to the storage disk paging file. The size of this memory is dynamic, and changes based on the memory needs. This counter helps in detect memory leaks within the kernel or device drivers. If the value of this counter is high and continuously grows, then it indicates a memory leak which leads to system instability.

4.2.12. Memory Bandwidth

This counter measures the rate at which the data can be transferred to and from the system's memory. This counter is necessary to monitor the systems that are running memory-intensive applications such as AI and ML workloads. There are sub-components to monitor this counter. Monitoring memory bandwidth is essential in AI hardware systems as the data transfer rates directly impact the speed and efficiency of algorithms and data processing tasks.

4.3. GPU (Graphical Processing Unit)

Graphics Processing Units, play an important role in AI hardware systems due to their ability to perform parallel processing that makes them highly efficient for the complex computations that are common in machine learning and deep learning tasks.

The following is the important list of performance counters that need to be considered while monitoring the GPUs used in AI hardware systems.

4.3.1. GPU Utilization

This counter measures the computational power that is actively being used by AI. The high GPU utilization indicates that the AI model is effectively leveraging the GPUs parallel processing capabilities. This metric helps to choose the most suitable GPU for specific AI computational requirements, performance finetuning, efficiency and cost.

4.3.2. Number of FLOPS

This counter measures the number of floating-point operations per second (FLOPS) that the GPU can handle. This metric helps in understanding the computational power and efficiency of the GPUs when running complex AI

algorithms and models. FLOPS is the indicator of GPU's ability to handle complex tasks such as training deep learning models. Higher FLPOS means the GPU can process more tasks in a specific time frame.

4.3.3. Cache Hits/Misses

This counter tracks the efficiency of the GPU's cache memory during data retrieval operations. The latency is caused by retrieving the data from the main memory due to cache misses slowing down the process as the GPU waits for data to be transferred from the main memory. Faster data access and improved performance are possible if the large datasets are taken from Cache memory which is Cache hit. These counters for cache hits and misses provide valuable insights into the GPU's memory access patterns and help identify bottlenecks.

4.3.4. Instruction throughput

This counter measures the rate at which instructions are processed by the GPU. This is an indicator of the GPU's efficiency in executing instructions related to AI workloads. High throughput means the GPU can handle a large number of instructions per second. If this counter is low it means the developers to look at their code to make better use of GPU capabilities.

4.3.5. Memory bandwidth

This counter measures the rate at which data can be transferred to and from the GPU's memory. High memory bandwidth allows faster data transfer between the GPU computation cores and GPU memory. This leads in improved performance in AI computations. If memory bandwidth is low, then it will become a bottleneck. This limits the overall performance of the AI system.

4.3.6. Kernel execution time

This counter measures the time it takes for a compute kernel to execute on the GPU. This performance counter includes the number of instructions executed and the time duration of the kernel execution. If the kernel has a high execution time and low resource utilization indicates the kernel is waiting for the memory operations.

4.3.7. Context Switching

This counter measures the frequency and efficiency with which GPU switches between different execution contexts. A context represents the state required for the GPU to execute certain tasks. GPU maintains the high throughput by switching the group of threads that are ready to execute rather than the group of threads waiting for data from memory. If the context switches are too high indicates issues in execution flow. The low switch rate suggests that the GPU is effectively managing its execution resources.

4.3.8. Pipeline Stalls

This counter measures the metric that tracks the number of times the GPU pipeline experiences a stall. A stall occurs when the pipeline cannot proceed with execution due to a lack of necessary resources. This is one of the critical counters that affects the AI system's performance. Reducing the pipeline stalls is one of the critical tasks to achieve optimal performance in GPU-accelerated AI applications.

4.3.9. Thermal Metrics

This counter measures the temperature of the GPU during processing. This is one of the crucial counters to measure to ensure the GPU does not overheat which can lead to thermal throttling or permanent damage to the hardware. Thermal metrics are GPU current temperature represents the immediate temperature of the GPU and GPU junction temperature represents the hottest temperature in degrees Celsius used to understand peak thermal load.

4.3.10. Power consumption

This counter measures the energy usage of the GPU. This counter provides insights of Total power usage drawn by the GPU, how the GPU effectively uses power and energy optimization possibilities. This helps in finetuning the GPU power consumption by setting its limits. This will significantly decrease the temperature and power draw which will improve the hardware lifespan.

4.4. Storage

Storage solutions must be designed to handle the high volume of data generated and processed by the AI systems. Local file storage, Network Attached Storage (NAS) and Storage Area Networks (SAN), Distributed File Systems (DFS), Object Storage systems, NVMe flash storage for faster access and Solid State Drives (SSDs) are the few storage types that can be considered for AI systems.

The following is the important list of performance counters that need to be considered while monitoring the storage used in AI hardware systems.

4.4.1. IOPS (Input Output Operations Per Second)

IOPS stands for Input/Output Operations Per Second. This counter measures the number of reads and write operations that a storage device can handle in one second. Each "operation" in IOPS refers to a single command to read from or write to a specific location on a storage medium. If IOPS is a higher value, then the better storage device ability to handle multiple concurrent read and write requests. By optimizing this read and write rate, the unnecessary delays caused by storage bottlenecks can be avoided in the AI system.

4.4.2. Storage Latency

This counter measures the time it takes for a data request to be processed and the corresponding actions to be completed at the storage devices. This metric reflects the delay between issuing a command and receiving a response from the storage device. High latency leads to low system performance as the system spends more time waiting for the data rather than compute. SSD type of storage is typically faster than HDDs that influences the storage latency. Even, the distance between the storage and the AI computing system is also one of the causes for storage latency.

4.4.3. Error Rates

This counter tracks the frequency of errors occurring during read and write operations on a storage device. This counter helps in identifying the health of the storage media. The error rates counter includes read errors, write errors, corrected errors and uncorrected errors. The high error rates indicate the storage device failure which causes data loss or system downtime. Keeping the error rates low ensures the smooth functioning of AI models.

4.4.4. Queue Depth

This counter measures the number of input/output (I/O) requests that are waiting to be processed by the storage device. This counter indicates the workload being processed on the storage system at any given point of time. The higher queue depth means that the storage system is under high load leads to increase in latency as each I/O request to wait longer. The lower queue depth indicates that the system is handling the incoming I/O requests efficiently without backlog.

5.0. Infrastructure Monitoring Tools

IT Infrastructure monitoring is essential to manage the continuous up and running of services. A wide variety of monitoring tools are available in the market to monitor the system resources. By considering the main features required for the environment, the following is the table of a few tools compared with their features, which provides a high-level view of the monitoring tools.

Feature	Dynatrace	Datadog	IBM Instana	LogicMonitor
Cloud Monitoring	AWS, Azure, Google Cloud, IBM Cloud, Oracle Cloud	AWS, Azure, Google Cloud, IBM Cloud, Oracle Cloud	AWS, Azure, Google Cloud, IBM Cloud, Oracle Cloud	AWS, Azure, Google Cloud, IBM Cloud, Oracle Cloud
Host Monitoring	CPU, Memory, Disk, Network, GPU	CPU, Memory, Disk, Network, GPU	CPU, Memory, Disk, Network, GPU	CPU, Memory, Disk, Network, GPU
Application Monitoring	Yes	Yes	Yes	Yes
Container Monitoring	Yes	Yes	Yes	Yes
Alerting	Yes	Yes	Yes	Yes
Dashboards	Customisable	Customisable	Customisable	Customisable
AI and Machine Learning	Yes	Yes	Yes	Yes
Log Monitoring	Yes	Yes	Yes	Yes
Integration with Other Tools	Extensive (e.g., Slack, PagerDuty, ServiceNow)	Extensive (e.g., Slack, PagerDuty, ServiceNow)	Extensive (e.g., Slack, PagerDuty, ServiceNow)	Extensive (e.g., Slack, PagerDuty, ServiceNow)
User Experience Monitoring	Yes	Yes	Yes	Yes
Pricing	Subscription-based	Subscription-based	Subscription-based	Subscription-based

Table 1: Monitoring Tools Comparison

6.0. Conclusion

Effective monitoring of Infrastructure components of an Artificial Intelligence system is required to ensure optimal performance, reliability, and security. This whitepaper has highlighted key strategies and considerations for implementing infrastructure monitoring solutions for AI systems. By continuously monitoring the hardware, software, and model performance metrics, Organizations can proactively detect issues, optimise resource utilisation, and mitigate the risks. Integrating the automated alerting system and leveraging advanced analytics will facilitate the support teams to respond quickly to anomalies and maintain the overall system health and efficiency of the AI infrastructure. Robust monitoring practices are essential for AI systems due to their systems evolving and scaling and to support their ongoing operations and innovations.

7.0. References

The following are the online references which we used to prepare this white paper on Monitoring AI hardware components.

- What is AI monitoring and why is it important?

  https://coralogix.com/blog/ai-monitoring/

- Infrastructure Monitoring

  https://www.datadoghq.com/product/infrastructure-monitoring/

- Unleash AI Power: A Guide to GPU Selection

  Unleash AI Power: A Guide to GPU Selection - Sesterce

- Demystifying FLOPS: A Beginner‘s Guide to Computing Performance

  Demystifying FLOPS: A Beginner‘s Guide to Computing Performance - 33rd Square

- What is instruction throughput and instruction latency?

  terminology - What is instruction throughput and instruction latency? - Computer Science Stack Exchange

- Performance and latency

  Azure OpenAI Service performance & latency - Azure OpenAI | Microsoft Learn

- How to measure the context switching overhead of a very large program?

  multithreading - How to measure the context switching overhead of a very large program? - Stack Overflow

- AI Storage: Optimized Storage for the AI Revolution

  https://cloudian.com/guides/data-lake/ai-storage-optimized-storage-for-the-ai-revolution/

- 11 Essential Cloud Metrics to Monitor for Optimal Performance

  11 Essential Cloud Metrics to Monitor for Optimal Performance | DigitalOcean