CATEGORY: DATA CENTER, CLOUD COMPUTING & HPC - DC02

POSTER P5257 | CONTACT NAME Mark Staveley: MarkStav@microsoft.com

GPU TECHNOLOGY CONFERENCE

GPU based HPC within a Windows Environment - Operational Best Practices and Lessons Learned

Mark S. Staveley[1] and Jim Jernigan[1]

[1]Grand Central Data, Microsoft Research *{MarkStav, Jim} @microsoft.com*

Microsoft Research · GRAND CENTRAL DATA · HPC Pack 2012 R2 · NVIDIA

Abstract:

In order to better support the use of Windows Servers running GPU workloads, the Engineering Team that supports Microsoft Research have combined various in-box and custom solutions that deliver a more consistent and reliable Windows Server + GPU experience for our Researchers. This poster presents an overview of various in-box and custom software solutions and operational choices that have been essential with regards to managing and scheduling research workloads on GPUs running in a Windows-based environment.

Motivation:

Microsoft Research has more than 1,000 world-class scientists and engineers working across research areas in labs worldwide. Their job: make significant product contributions and address some of society's toughest challenges. Increasingly, Microsoft engineers and researchers are turning to GPUs to help with their tasks. A growing percentage of their work is focused on machine-learning projects – one of the hottest fields in computer science.

The goal of machine learning is to let computers make better predictions by capturing the wisdom of historical data. But history shows that promise is far easier to describe than to achieve. Three trends are driving a resurgence in machine learning. First, data of all kinds is growing exponentially. Second, researchers have made big improvements in the mathematical models used for machine learning. Finally, GPUs have emerged as a critical computational platform for machine learning research. These drivers are resulting in game-changing improvements in the accuracy of these models. That's because GPUs allow researchers to train these models with more data – much more data – than was possible before.

Microsoft Research has just deployed a computer system packed with NVIDIA GPUs - part of a large initiative called Grand Central Data. This poster presents a collection of architectural and design choices in the areas of system management and workload scheduling as well as a summary of outcomes resulting from these choices that are at the heart of this GPU ecosystem.

Architecture and Design:

In order to provide some context as to the system management and workload scheduling design choices that have been made, it becomes relevant to mention GPU Cluster Architecture and Design.

Throughout our design process, we have focused on flexibility as we want to create a GPU ecosystem that can not only support large amounts of Single-GPU workloads, but at the same time it is able to handle some of the most challenging Multi-GPU workloads within Microsoft. Table 3.1 highlights some of the general design decisions around the hardware configurations of our GPU machines. Table 3.2 gives examples of both core and domain specific software packages that are required for our GPU machines.

During the design of these systems we were very aware of industry standards / best practices (e.g. FDR Infiniband intra-cluster interconnect) and at the same time we also wanted to be mindful of the different directions that research is headed in. Examples of this include specialized networks for data ingestion, and software components that allow for interaction with large scale data sources (e.g. SQL and Hadoop).

GPU Node Hardware *(General Guidelines)*	
Server Class	HP SL Series Gen8 *(or comparable)*
CPU Guidelines	2x Xeon E5-2600 Series *(minimum)*
Memory	128GB DDR3 RAM
Storage	4x 400GB SSD Local Scratch *(minimum)*
Networking	40 Gbps Ethernet ***and*** 56 Gbps FDR IB
GPU	2-4 NVIDIA Tesla K20x or K40 GPUs

Table 3.1: General description of design choices made regarding hardware configurations for our GPU machines.

GPU Node Software *(General Guidelines)*	
Core	Windows Server 2012 R2 *(DataCenter Edition)* Windows HPC Pack 2012 R2
Domain Specific	NVidia Nsight • Matlab MCR Runtime support • Intel MKL and Compiler Runtime support • HDFS Client / Hadoop Tools (HDP 2.x) • Visual Studio Runtime support • CUDA, cuDNN

Table 3.2: Examples of Core and Domain Specific software packages that are required for our GPU machines.

System Management:

When building the Grand Central Data GPU ecosystem, we considered the following operational areas; OS Deployment, Cluster Management, Hardware Monitoring, Software / Firmware Version Control, OS Monitoring / Reporting and Security Compliance Monitoring and Control. Table 4.1 provides an overview as to the different choices that were made in these areas.

In general, the Microsoft Research operations engineering team is continuously working to ensure a balance between reliability, best-practices, and time-to-innovation with minimal complexity. Furthermore, by utilizing automation systems (both in-house and external) we are able to ensure that our GPU environment is consistent and repeatable.

Area	Software Package	Highlights
OS Deployment	Altiris Deployment Server HPC Server 2012 R2	Altiris can be a very powerful management tool and acts as a shared knowledge base for tasks. MS SCCM is similar but over-scoped for this simpler application. HPC Server 2012 R2 has excellent deployment performance and OS configuration abilities at scale.
Cluster Management	HPC Server 2012 R2	Very solid, mature cluster management tool. Built as part of the Windows ecosystem
Hardware Monitoring	HP System Insight Manager Microsoft SCOM HPC Server 2012 R2	Vendor provided (HP) monitoring tool with excellent information for preventative and reactive maintenance needs.
Software / Firmware Version Control	HP System Insight Manager HPC Server 2012 R2	Vendor provided (HP) monitoring tool with excellent information for preventative and reactive maintenance needs. HPC Server 2012 R2 provides mechanism for software standardization and version control of user applications and tools.
OS Monitoring and Reporting	Microsoft SCOM HPC Server 2012 R2	Monitors and alerts on OS-level issues like memory paging and drive utilization. HPC Server 2012 R2 provides an extensive reporting and diagnostic ecosystem.
Security Compliance	Microsoft WSUS HPC Server 2012 R2	One-stop reporting for patch compliance with Microsoft WSUS and HPC Server 2012 R2 provides automated tooling for the application of patches.

Table 4.1: Overview of System Management Tools employed by the Grand Central Data GPU ecosystem

Figure 3.1: Example GPU Systems – as architected by the Microsoft Research operations engineering team

Figure 5.1: Visualization of the distribution of workload runtimes as scheduled by our customized Windows HPC Scheduler.

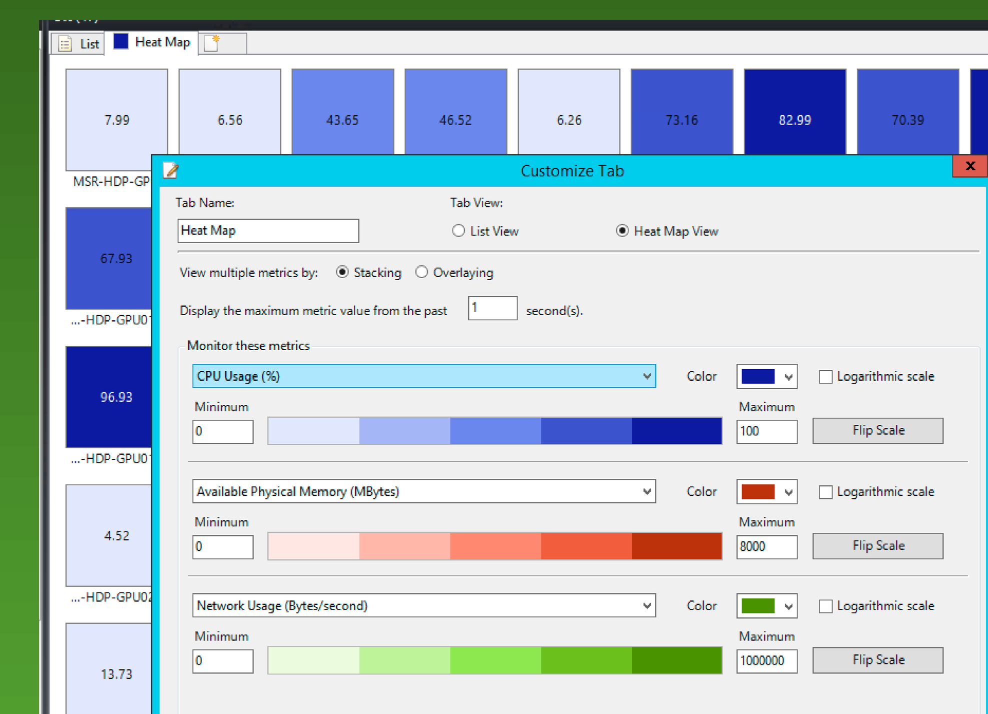

Figure 5.2: Screenshot of the Windows HPC Pack 2012 R2 Heat-map Interface

Acknowledgements

The authors would like to thank the Grand Central Data project team, Matej Ciesko (visualization tooling), and the Windows HPC Engineering team for their contributions to this effort. Also a special thank you to our partners at NVIDIA for their openness and collaboration while we work to make Windows a first-class GPU workload environment.

Scheduling and Job Management:

Our GPU environment relies on the Windows HPC Pack to provide customizable and robust Job Scheduling and Job Management tools for Windows servers. This collection of HPC tools has proven itself within the traditional CPU-based HPC space, but there are some issues with regards to how it operates in a mixed CPU/GPU ecosystem. In short, there are two main challenges; the first challenge is that the Job Scheduler is not GPU resource aware. The second challenge is that the Job Scheduler does not provide a built-in *Fair-Share* scheduling mechanism.

In terms of scheduling against the GPU systems, we had a number of options to choose from as the HPC Pack Job Scheduler recognizes cores, sockets and nodes. Through our investigations, we determined that two options were best suited for our environment. The first option is to schedule against nodes – whereby a user would ask for a particular number of nodes knowing how many GPUs were on each system. This works well provided that a single user job is able to use all of the GPU resources found on a given node. The second option is to undersubscribe the amount of cores on the machine so that the number of cores "seen" by the HPC Pack Job Scheduler matches the number of GPUs. In this case, we would be able to assign workloads to a specific GPU. However, there is more complexity involved on the side of the user when using this method for submission as they would need to clearly specify the number of GPUs they would need and if those GPUs need to be on the same physical machine. By using a combination of both methods we have been able to provide simple mechanisms for submitting jobs that can be easily applied to many different types of GPU workloads. However, even with the different node-based and undersubscription-based resource metrics, we still have the problem of no built-in *Fair-Share* scheduling mechanism.

Luckily, the Grand Central Data engineering team was able to overcome this shortfall by making use of a deep technical feature found within the HPC Pack Job Scheduler – activation filters. Within the HPC Job Scheduler there are 3 main states for workloads as they go from job submission to actually running on hardware. These states are *Submitted, Queued, and Running*. By creating an activation filter we are able to gate the transition between *Submitted* and *Queued.* Activation filters are able to take into account different types of information in order to best gate this transition. Examples of the information that can be leveraged by an activation filter includes information about quantity of resources being used by a given user, and the number of jobs already in either a submitted state or running state by a given user. The creation of an activation filter is not a trivial task. It requires extensive testing and monitoring to ensure that it is working as expected given the different types of workloads that are being submitted and that appropriate usage policies are being enforced. In order to assist with this task, internal tooling was developed by Microsoft Research so that workload job history patterns (including run time and wait time) could be visualized for analysis and monitoring purposes. Figure 5.1 shows sample visualization output from our in-house system that provides job profile information over time. Having this kind of visualization information was invaluable when trying to minimize wait times and identify areas where we had holes in our resource utilization / allocation. With this visualization information we were able to determine that an additional component was needed in order better optimize workload throughput. In addition to using activation filters and different GPU resource allocation metrics, we determined that we needed to implement two different job scheduling queues with dynamic resource pools.

Within our GPU ecosystem, our main goal is two-fold. We want our GPU ecosystem to not only be attractive to many types of user workloads, but we also want to be able to maximize job throughput and the fair-sharing of resources. In order to accommodate this, we set up two job queues (a long queue, and a short queue), each being assigned 30% of our available job resources. We then leveraged an additional feature HPC Pack Job Scheduler whereby we can allow the dynamic expansion of resources based on demand. In particular, we set up a shared resource pool that is made up of 40% of our capacity. This shared resource pool has been designed to provide dynamic capacity that can be leveraged either by the short or long queues. Work to better tune our job scheduling and resource allocation algorithm is ongoing, however we are quite pleased with results thus far. We wouldn't have been able to get to this stage without various visualization tools (examples of which can be seen in Figure 5.1 and Figure 5.2). Being able to see the workflow patterns across all of our GPU systems has been super-helpful and effective. We hope that we can continue to keep our researchers happy and maximize cluster usage. Currently our GPU cluster is running with a consistent utilization rate of >95% with minimal wait-times.

Future:

Although this poster addresses many of the challenges with running and supporting GPU workloads within a Windows ecosystem, there are still a number of areas where notable improvement and future work can be done. One of the main areas is the ability to track GPU usage directly and schedule against GPUs themselves rather than having to use some kind of substitute metrics (e.g. undersubscription of sockets or scheduling by node). We hope to be able to work with our partners from NVIDIA and the Windows HPC team to be able to solve this in the not to distant future. Additionally, we hope that additional developments will be made with regards as to how GPUs can be accessed programmatically within Windows. One specific example of this is the ability to reset a GPU programmatically. This feature is in existence today within a Linux ecosystem, but not within a Windows based ecosystem. This becomes particularly important when workloads cause a GPU to get "stuck" in some way and the only fix is a reset. This is a one line command in Linux, whereby in Windows a reboot of the system is typically required to get the same result. We are pleased however with regards to the level of parity between Linux and Windows and see no reason why these issues could not be addressed going forward, thereby closing the gap in functionality even more.