Adhishwar Singh Mittal (19300379) – CS7NS1 (Data Science Strand)

Paper 1: *"A GPU-aware Parallel Index for Processing High-dimensional Big Data"*

DOI – "*10.1109/TC.2018.2823760*" **|** (Published in: "*IEEE Transactions on Computers*")

Conclusions and Findings

- Developed an efficient way to process/search through high dimensional datasets – G-Tree
- Space cost of a G-tree is half of that of traditional framework - R-tree
- G-tree surpasses R-tree as well as its variants X-tree and SR-tree for higher dimensionality data sets
- Fanout of G-tree, if designed as multiple of threads, maximises GPU performance/parallelism

Technological Insights

- CPU based indexing is inefficient because of uncoalesced readings due to scattering of nodes in non-continuous memory space
- Traditional algorithms using MBRs and bounding spheres work poorly with increasing dimensionality
- Dimensionality reduction techniques do not work well with subspace queries and metric based queries (KNN etc.)
- Access constraints of stacks and queues data structure prevent optimal use of GPU parallelism

Insights on Processing Scalability

- Infrastructural setting could be optimised as per the data structure (eg. SoA instead of AoS, with fanout of GPU designed as a multiple of threads)
- The traversal order could be optimised to suit infrastructural setting (eg. BFS with parallel processing)
- Selective dimensional filtering could speed the process by quickly identifying totally irrelevant data objects
- We can rearrange the process to minimize space cost and time complexity and optimise for available resource and even get linear performance regardless of increasing dimensionality like in the case of G-Tree

Paper 2: *"Scalable GPU Virtualization with Dynamic Sharing of Graphics Memory Space"*

DOI – "*10.1109/TPDS.2018.2789883*" **|** (Published in: "*IEEE Transactions on Parallel and Distributed Systems*")

Conclusions and Findings

- A scalable GPU virtualisation solution by introducing partition and sharing to overcome limitations of global memory space in gVirt
- The system can host 4x number of virtual instances as compared to gVirt
- There is a performance loss due to copying of shadow GTT into physical GTT while switching context which is solved by predictive GTT copying
- Near native performance by predictive copy aware scheduling, 96% performance of gVirt with multiple instances

Technological Insights

- gVirt suffers in scalability due to the static resource partition since each VM can only access the memory assigned to it
- gVirt is less flexible because CPUs for VMs cannot access host's global graphics memory
- gVirt is an open source algo for Intel graphics chipset only. gScale needs to be validated over other chips such as Nvidia and AMD which are not open source
- The process may not yield similar results for other graphics drivers as they may run on dedicated graphics memory but needs to be validated empirically

Insights on Processing Scalability

- Cost effective scalable systems can be achieved by virtualisation of processing capabilities
- GPU virtualization aims at making graphical calculations on a host server and rendering results on VMs. For eg. Amazon EC2
- With sharable vGPU memory space we can optimally assign limited memory to dynamic requirements of each VM
- Fence registers may point VMs CPU to the wrong memory space and must be taken care of if we want to bypass global memory or else it will reduce flexibility of resource utilisation and hence restrict scalability