REMCOM® Electromagnetic Simulation Solutions

XFdtd® EM Simulation Software

MPI - GPU Acceleration Performance Report

XFdtd supports simulations with virtually an unlimited number of cells -- limits are introduced only by the available hardware and its scalability. To make such large problems truly tractable, the simulation must be able to utilize high-performance hardware on multiple physical nodes. Remcom has long been a leader in both MPI and GPU technologies. XFdtd brings these two technologies together to provide unparalleled simulation performance.

We often are asked “How can I allocate resources to obtain the fastest simulation?” or “What are the performance advantages of hardware X over hardware Y?” This report attempts to quantify the performance profile of XFdtd’s GPU and MPI technologies.

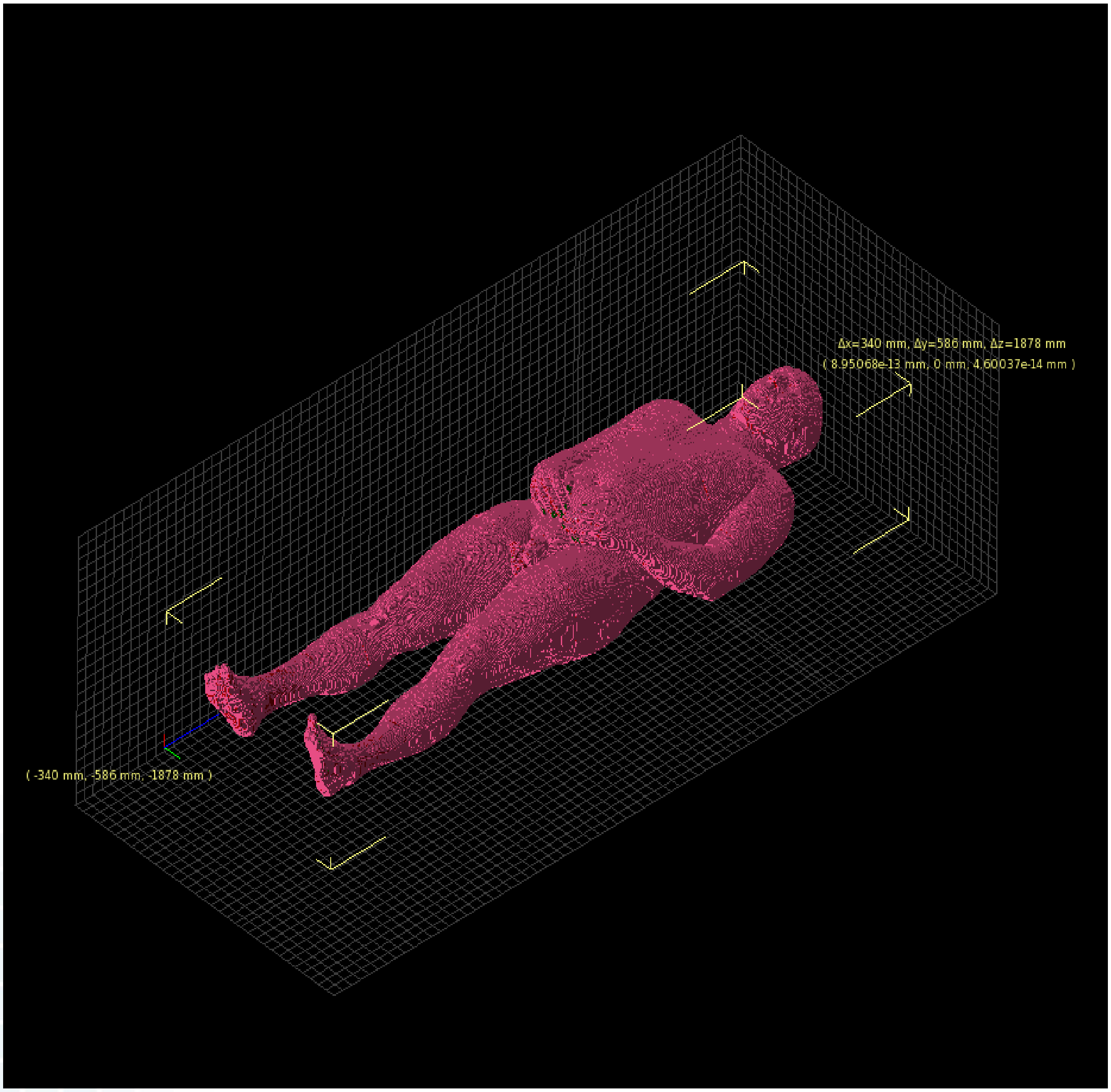
All data in this report was collected by running a series of simulations using NVIDIA’s PSG Cluster, access to which was graciously provided by NVIDIA Corporation. This report will summarize and highlight the most useful findings from the approximately 160 simulations that were run.

Contents

Test Project Description

The project use to create the series of simulations for this report was the VariPose® man with a patch antenna embedded in his chest, found at http://www.remcom.com/examples/patch-antenna-in-body.html

This project was selected because large simulations could be created from it without making the cell sizes artificially small relative to the smallest geometry and it is representative of many real-world projects in terms of materials, aspect ratio and excitation. The image below shows the full project to give perspective of the physical size extent of the simulation space.


Δx=340 mm, Δy=586 mm, Δz=1878 mm
(8.95068e-13 mm, 0 mm, 4.60037e-14 mm)
(-340 mm, -586 mm, -1878 mm)

Remcom XFdtd MPI/GPU Performance

A suite of 12 simulations of different sizes was created using this project by keeping the bounding box of the simulation constant while changing the base cell size. The simulation was configured to run for 20k time steps, which was chosen such that the fastest expected simulation would take no less than four minutes. The table below describes the simulation specifications in detail. The memory requirements are specified for XStream® GPU Acceleration RAM, not system RAM, and were verified by inspecting the log of the actual memory allocated.

Cell Size (mm)	XStream RAM (GB)	Cell Counts x	Cell Counts y	Cell Counts z	PML Layers	Total Cells
3.00	2	235	316	747	7	62,009,920
2.50	4	282	379	897	7	105,232,400
2.30	5	306	412	974	7	133,812,525
2.10	6	334	452	1068	7	174,424,755
1.90	8	370	499	1179	7	233,746,432
1.60	13	439	593	1399	7	386,763,744
1.40	19	501	678	1600	7	572,895,662
1.20	30	585	789	1866	7	901,160,884
1.10	40	638	861	2036	7	1,165,827,726
0.96	58	731	987	2332	7	1,744,680,000
0.90	70	780	1052	2488	7	2,112,207,045
0.86	79	816	1102	2604	7	2,418,984,695

Total cells were calculated as number of user space cells and padding cells. Simulation throughput in this report is calculated using this number of cells per time step. Simulation time was measured as the amount of time spent during the time stepping phase for the purposes of throughput computation. XFdtd supports much larger simulations sizes than shown here (virtually unlimited size), but could not have been tested on the PSG cluster due to hardware resource limitations.

Computational Cluster

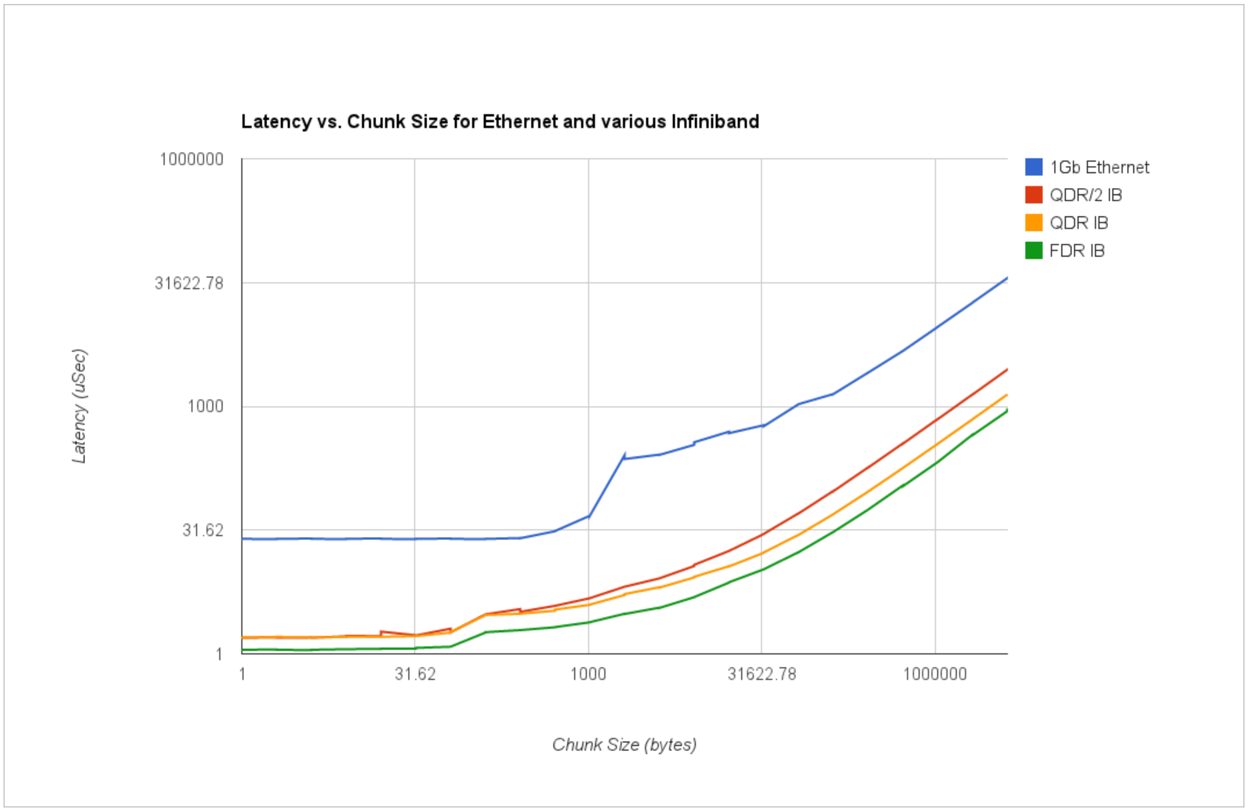
At the time of testing, the PSG cluster consisted of Westmere- and Sandy Bridge-based computers populated with one to eight GPUs (depending upon machine architecture) of various models ranging from M2050 through K20X. The nodes were interconnected with Gigabit ethernet and one of three different forms of Infiniband: QDR connected at one half bandwidth, QDR full bandwidth, and FDR full bandwidth.

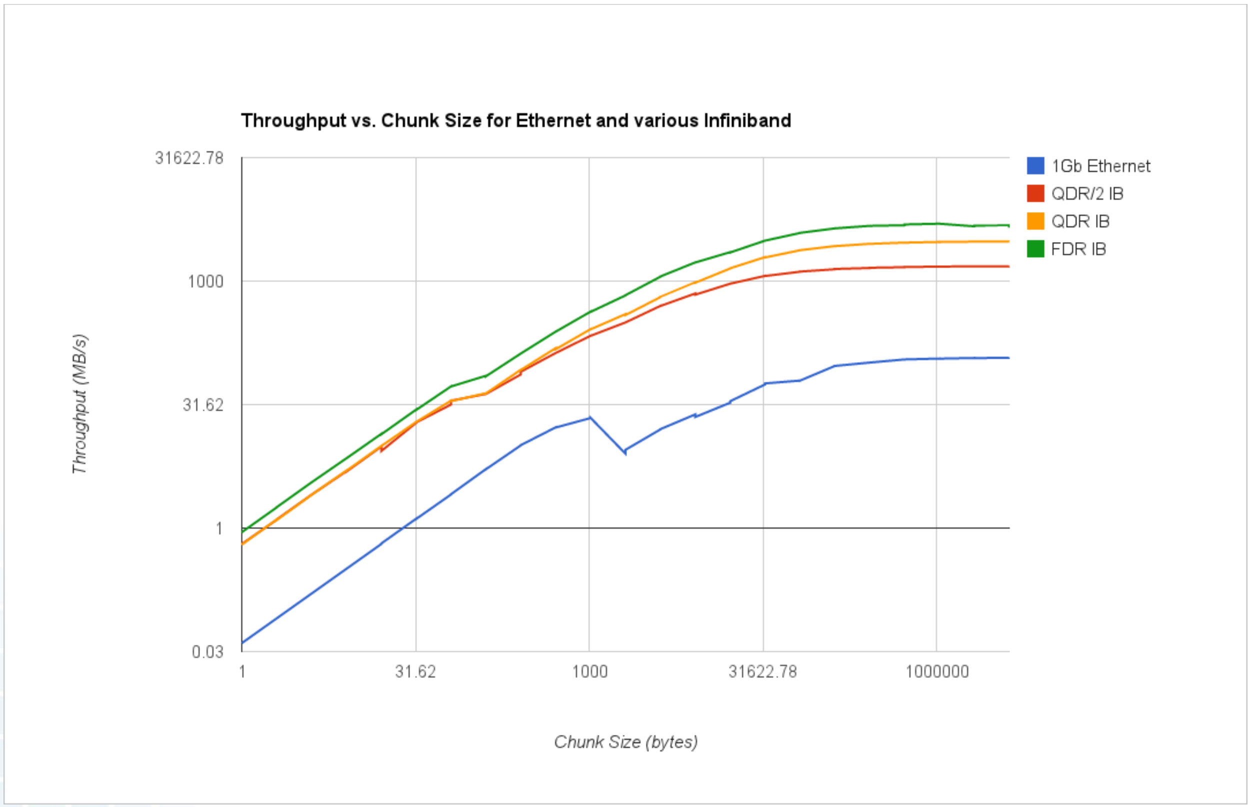
Some terms that are used throughout this report are defined here:

> *Node*: A single, complete computer that may have more than one GPU
>
> *SMP*: Symmetric Multiprocessing, meaning that a single instance of the program is run on a single node, possibly utilizing multiple GPUs, to compute a single simulation
>
> *MPI*: Message Passing Interface, meaning that multiple instances of the program are run on one or more nodes, possibly utilizing multiple GPUs, and those instances communicate with each other to compute a single simulation
>
> *Rank*: One instance of the program when using MPI

The MPI testing tool “pingpong” was run to assess the network performance of the various interconnects available on the cluster. This tool determines both *latency*, which is a measure of the delay in time between initiating the sending of data from one node to another and when the transfer actually begins, and *throughput*, which is how fast data is transferred after the transfer begins. These quantities are measured for different “chunk” sizes, which is the amount of data being transferred. The tool was run between three different sets of four machines using both the 1Gb ethernet and the available Infiniband interface. As can be seen in the charts below, ethernet performance was basically equivalent between all of the machines, whereas Infiniband performance was increasingly lower latency and higher throughput for QDR/2 (QDR half bandwidth), QDR (QDR full bandwidth) and FDR (FDR full bandwidth), respectively. Note that the vertical scales in these graphs is logarithmic, so the differences between these four interfaces is significant.

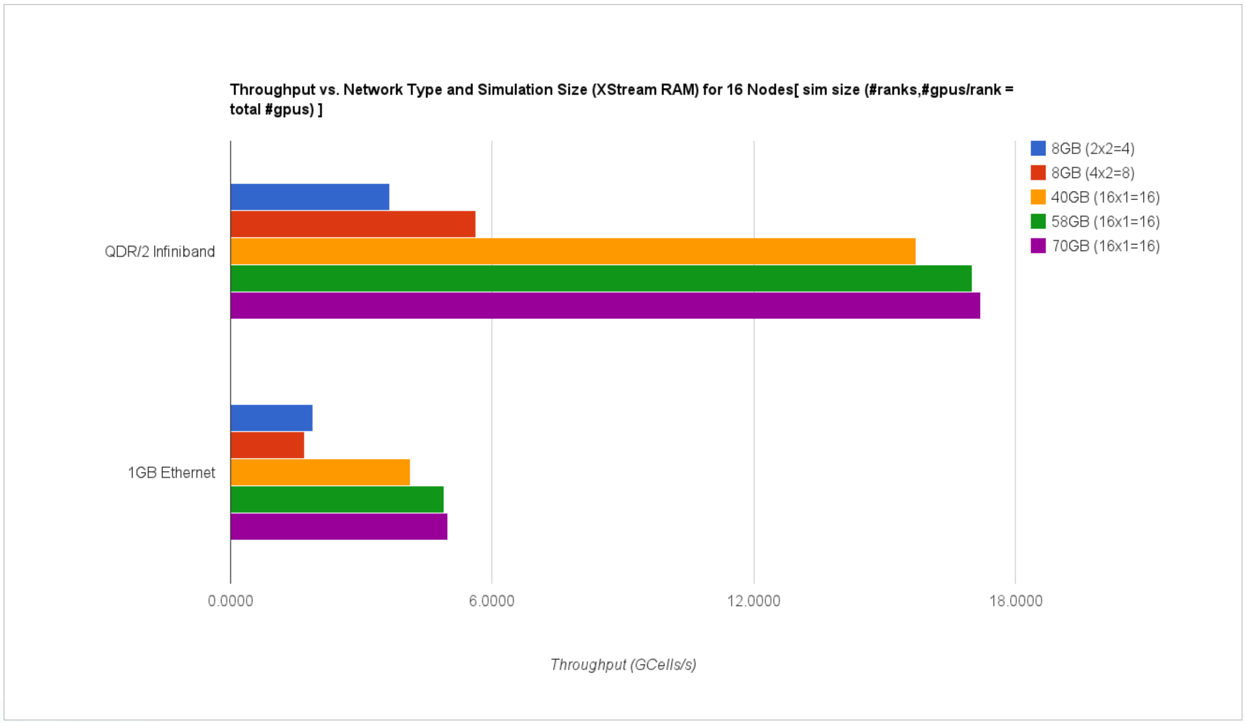
Remcom XFdtd MPI/GPU Performance





Impact of Network Interconnect on Simulation Performance

To assess how the type of network interconnect affects simulation speed, several simulations were selected to be run with MPI over both ethernet and QDR/2 interfaces. Ideally, all different interface types would have been tested, but the only homogeneous selection of machines large enough to perform the tests was interconnected with QDR/2. The figure below shows a comparison of simulation throughput (measured in gigacells per second) for different configurations. The notation in the legend is “{simulation size} (#ranks, #gpus/rank = total #gpus)”. As expected, simulations using the Infiniband interface outperformed those using ethernet. The difference is expected to be even larger if QDR or FDR is used, though estimating how much different would be purely speculative given that the speedup going from ethernet to QDR/2 is not the same ratio as that between the latency or throughput of ethernet and QDR/2 as shown above. The figure shows a “worst-case” speedup going from ethernet to Infiniband; any installation with Infiniband should perform as well, or better, than shown.



Impact of GPU Model on Simulation Performance

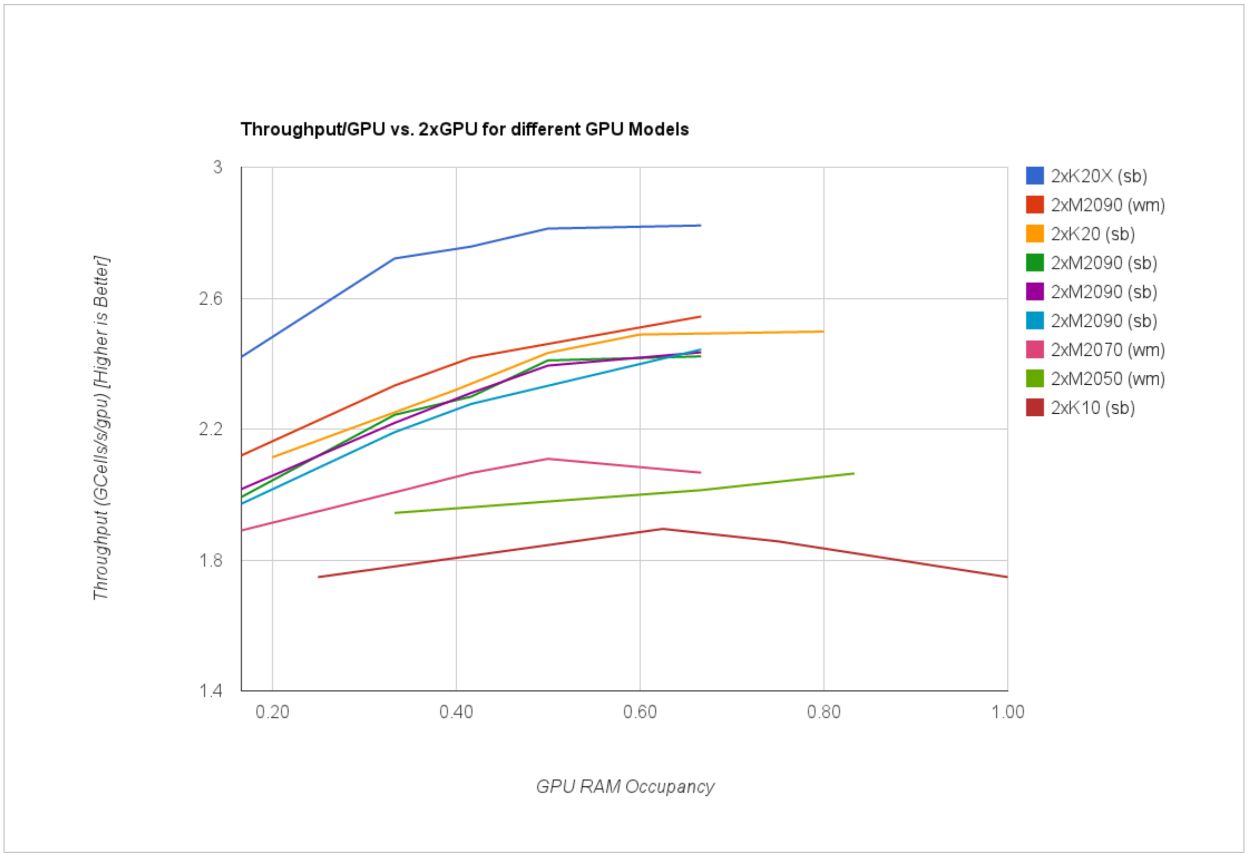
The PSG cluster made the NVIDIA GPU models listed in the table below available for testing.

GPU Model	Total RAM (GB) [ECC Disabled]	# cores	Peak Single Precision (GFLOPS)	BW (GB/sec) [ECC Disabled]	GPU Architecture
M2050	3	448	1030	148	Fermi
M2070	6	448	1030	150	Fermi
M2090	6	512	1331	177	Fermi
K10[1]	4	1536	2288	160	Kepler
K20	6	2496	3520	208	Kepler
K20X	6	2688	3950	250	Kepler

At least one machine was available with two of each model. This allowed a straightforward, head-to-head comparison between the different models for use with XFdtd. Additionally, the M2090 was available in machines of both the Westmere and Sandy Bridge architectures. The architecture is indicated with “wm” or “sb” in the figures below.

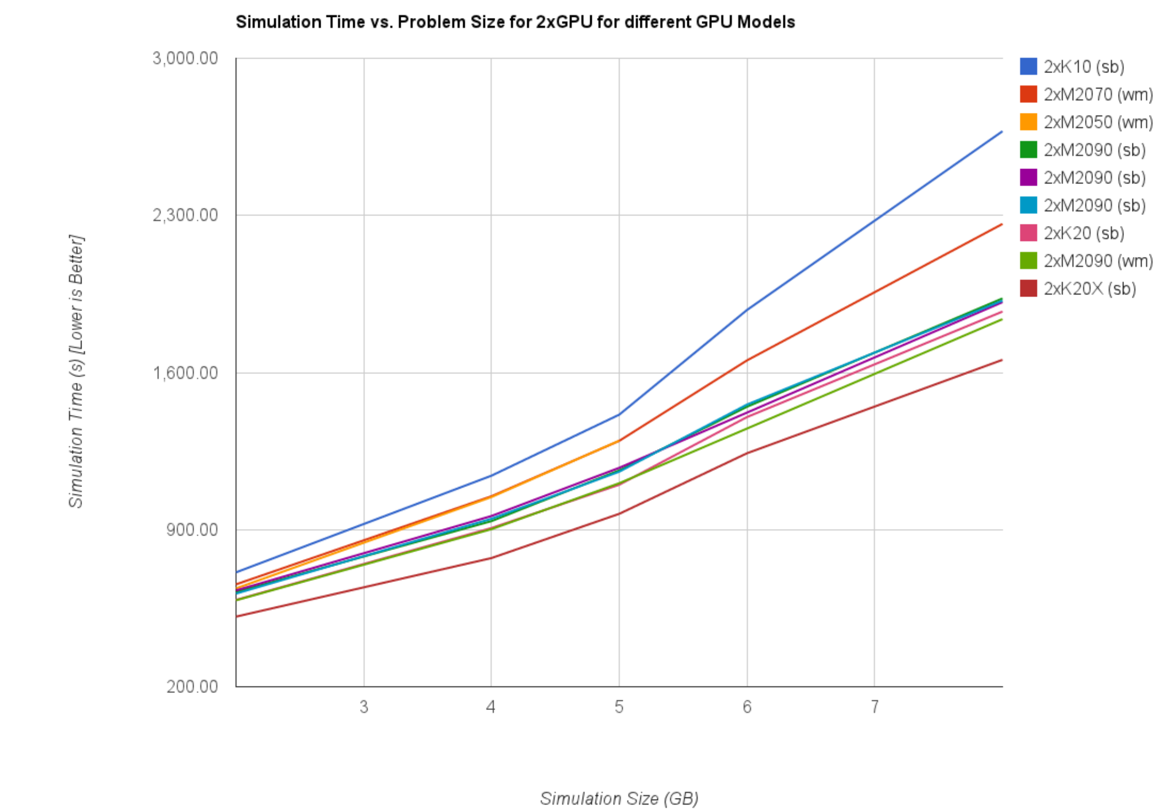
For the GPU model comparisons, the five simulation sizes that could be run on pairs of all models (2, 4, 5, 6 and 8GB) on a single machine (SMP) were chosen and executed. The figure below shows the throughput per GPU as a function of RAM Occupancy for each GPU model. RAM Occupancy is the proportion of the RAM on that GPU being used in the simulation. The two main factors that affect throughput are the number of GPU cores and the GPU bandwidth.   Increasing the number of GPUs would seem to have the obvious effect of increasing the throughput. Indeed, this can be seen in the chart since the models with more cores outperform models with fewer cores for any specific occupancy value. (The outlier in this statement is the K10, which needs more investigation.) However, it has been known for some time that the FDTD algorithm is bandwidth limited both on CPU and GPU architectures after reaching a certain number of processors.   This is seen in the graph by the fact that as the occupancy increases (and therefore more bandwidth consumed), throughput flattens out or even decreases slightly.

[1] The K10 card has two GPUs on it; the specifications provided here are per GPU.



This is interesting, but the end-user is probably more interested in the more direct measurement of simulation runtime, and throughput vs. RAM occupancy is a different type of measure. The figure below plots simulation runtime vs. simulation size for the same simulations used to generate the figure above.

Remcom XFdtd MPI/GPU Performance



In these tests, the K10's performance measure does not align with its specifications. The reason for this is unknown and needs to be investigated further. A different set of tests performed earlier on different hardware showed the K10 performing on-par with the M2070 (on a per-GPU basis).

The K20X is a clear performance winner in all cases. It is also the most expensive of all the models tested, as is shown in the table below which contains approximate current prices (as of February 2013).

GPU Model	Feb 2013 Price
M2050	$1610
M2070	$1840
M2090	$2530
K10	$3335
K20	$3323
K20X	$4370

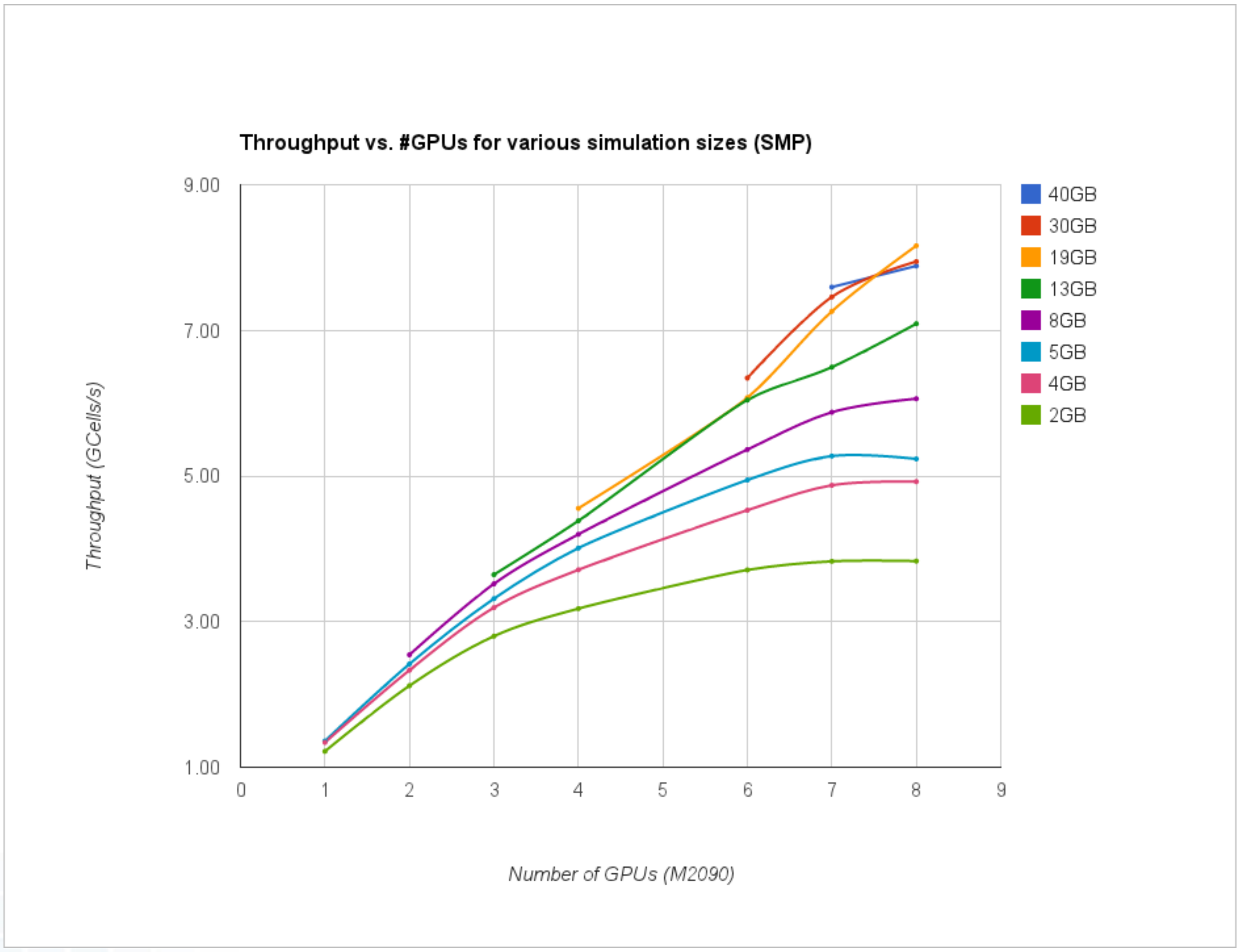
Note that the K10 has two GPUs whereas all the other models have one GPU. To understand what the best purchase would be, one would need to understand the types of problems that are to be solved so that the RAM requirements are known, and understand of the cost and performance of a simulation. For example, taking the 5GB simulation size above, we can create a table like the following to compare each of the models to the M2090.

Model	Avg Simulation time (s)	Performance Relative to M2090	Cost of Hardware Used	Cost Relative to M2090
M2050	1296.48	0.89	$3220	0.64
M2070	1295.58	0.89	$3680	0.73
M2090	1151.15	1.00	$5060	1.00
K10	1412.32	0.82	$3335	0.66
K20	1100.16	1.05	$6644	1.31
K20X	970.49	1.19	$8740	1.73

From this data, the lowest cost/performance is the M2050, but it also has the lowest amount of RAM. Taking into account available RAM, performance and cost, overall best purchase at this time might be the M2070, with the same RAM as the M2090, 11% less performance but 27% less expensive.

Impact of # of GPUs and Simulation Size (SMP)

One of the PSG cluster machines was equipped with eight M2090 GPUs. The figure below shows simulation performance while the number of GPUs used and simulation size were varied in this configuration (multiple GPUs in a single machine, SMP). For a fixed simulation size, increasing the number of GPUs results in diminishing returns, especially when the simulation size is small, since communications between the GPUs becomes a larger and larger percentage of the overall runtime due to underutilized GPU cores. The 5GB case seems to be an outlier since it dips slightly at eight GPUs, but the 2GB and 4GB cases are also basically flat moving from seven to eight GPUs. It is likely that if smaller simulations (1GB or smaller) were tested, a peak in performance would be seen for GPU numbers being less than eight.



In reviewing the chart and correlating each plot with RAM occupancy, it is interesting to note that in general there is nearly N-speedup with the number of GPUs as long as RAM occupancy of each GPU is roughly greater than 25%.

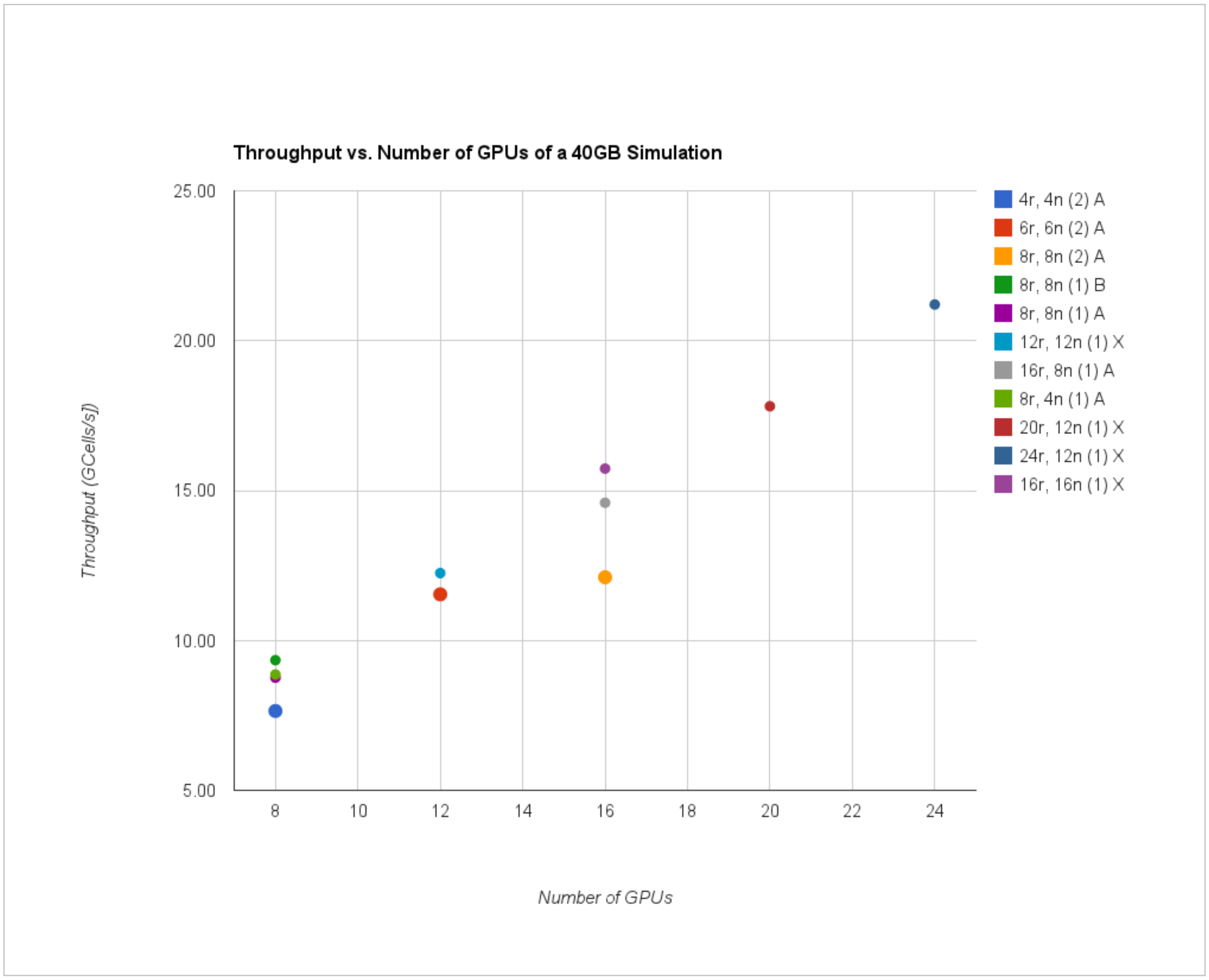
Another takeaway from this study is that except for extremely small (relative to total available GPU RAM) simulations, one should use all the GPUs to obtain the fastest simulation. However, if one wishes to obtain the best overall throughput on the available hardware, it would be better to run multiple simulations, each using only a subset of the available GPUs.

Impact of # of GPUs and Simulation Size (MPI)

The PSG cluster was configured with eight nodes containing one M2090 and eight nodes containing two M2090's, all interconnected with QDR/2 Infiniband. These machines were used for the bulk of the testing in this report since they offered a homogeneous platform with the use of up to 24 M2090 GPUs.

Because some of the machines had two M2090 cards, simulating with a specific number of GPUs could be accomplished with different MPI/GPU configurations. For example, using 16 GPUs could be accomplished by using 16 ranks using one GPU/rank on 16 different nodes (machines) or eight ranks using two GPUs/rank on eight different nodes. To understand how different configurations affected performance, the 40GB simulation was chosen and run in a number of ways as shown in the figure below. In the figure, the legend format is "{number of ranks}, {number of nodes} ({number of gpus/rank}) {T}", where T == A for using all nodes containing 2xM2090, T==B for using all nodes containing 1xM2090, and T=X for using a combination of A and B nodes.

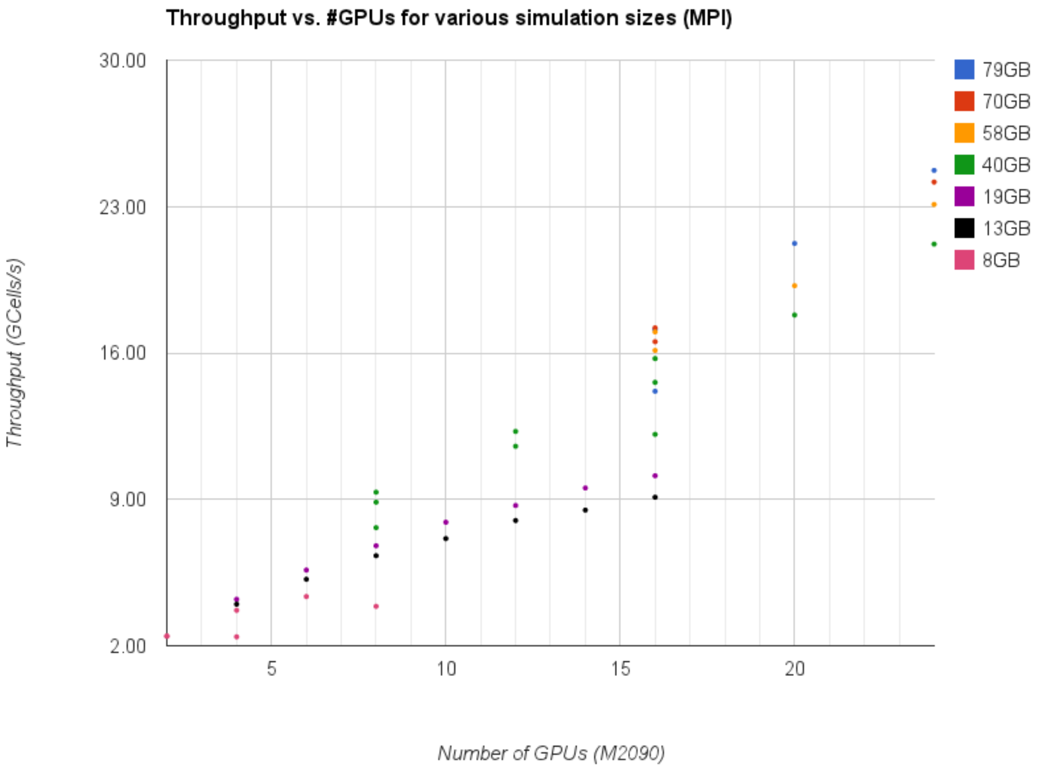
Remcom XFdtd MPI/GPU Performance



This study produced a very interesting result. It appears that better performance is achieved by using multiple ranks utilizing one GPU/rank multi-GPU node than by utilizing multiple GPUs per rank. This is likely due to GPU/CPU contention within the process of the latter. It is also clear from this chart that XFdtd scales very well, with nearly N speedup with the number of GPUs over the tested range at this problem size. A very interesting study would be to continue increasing the number of GPUs if a system with the required hardware ever becomes available in order to see where N speedup no longer holds.

Remcom XFdtd MPI/GPU Performance

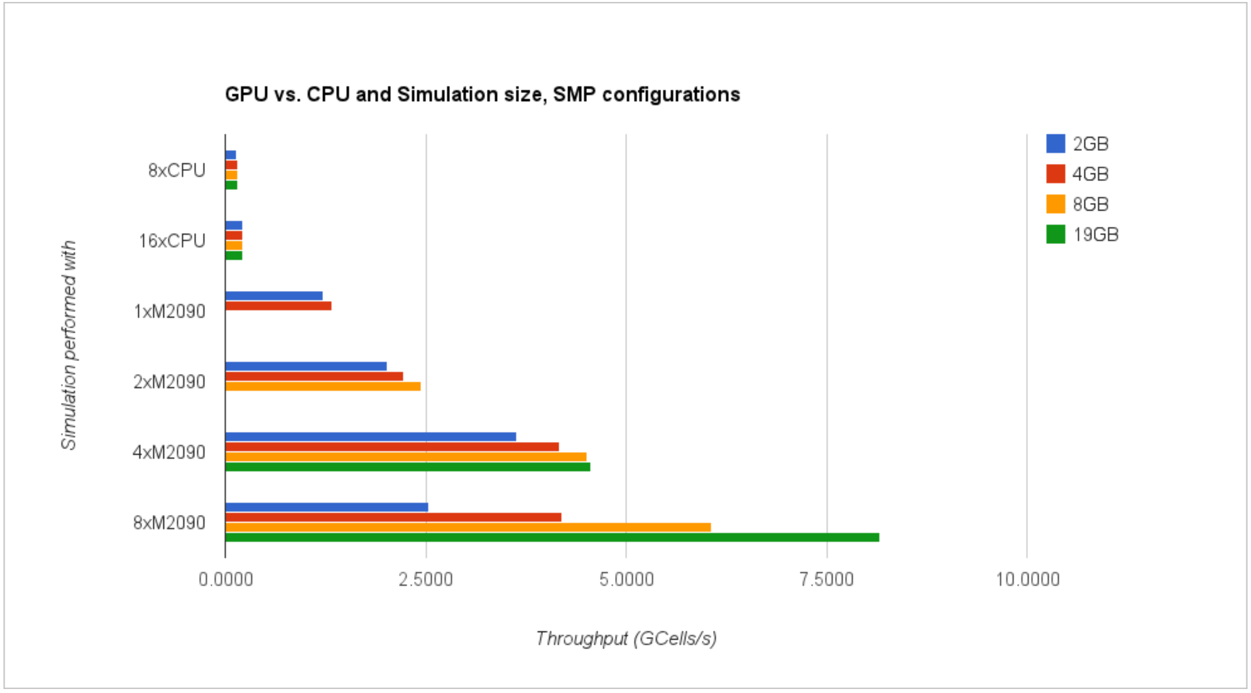
As in the previous section, a study of simulation performance while varying the number of GPUs *and* simulation size was performed using the same set of machines as above. The results of this study are shown in the figure below. Although not called out specifically in the figure, the 8GB, 12GB and 16GB simulation sizes were run with different configurations similar to above. Again, for simulation sizes 40GB and higher the speedup with increasing numbers of GPUs is nearly N. For smaller sizes, the amount of communication between ranks becomes a larger percentage of the overall runtime as the number of GPUs increases, since each GPU becomes less efficient due to decreasing workload.



As in the previous section, by correlating each plot with RAM occupancy, it is interesting to note that in general there is N-speedup with the number of GPUs as long as RAM occupancy of each GPU is roughly greater than 25%.

CPU vs. GPU

The PSG cluster included a node with two 8-core, Intel Xeon E5-2670 @ 2.6GHz processors based on the Sandy Bridge architecture. This provided the opportunity to make a small comparison between CPU and GPU performance. The figure below compares two simulation sizes for different numbers of CPU and GPUs. Clearly, GPUs provide high performance for their cost relative to CPUs.



Impact of ECC

NVIDIA Fermi and Kepler architecture GPUs have Error Correcting Code (ECC) capability built into them for detecting memory errors. This capability is enabled by default, but can be disabled. The data in this report was generated on GPUs with ECC disabled. One might ask, "Why would I disable error checking? Isn't it possible I could get bad results?" It is true that we have had numerous examples of NVIDIA cards failing in ways that allowed simulations to run but gave bad results with ECC disabled; however, it is unclear whether having ECC enabled would have detected the problem anyway. As for why to disable it, several simulations were run with ECC enabled. Comparing the runtimes, we find that enabling ECC reduces performance from anywhere between 22% to 33%. This is because the ECC computation is performed on the same processors on the GPU that the simulation uses. Additionally, ECC requires extra memory storage (one bit per byte), and therefore the amount of available RAM on the GPU is reduced by 1/8. Clearly, the extra peace of mind that may come with ECC turned on also incurs a steep performance penalty.

One technique that is used to check the health of the GPUs is to periodically run a known simulation and compare it with previous results. This works, but doesn't cover the entire memory space of the GPU and is also prone to false positives when XFdtd is upgraded and results change due to bug fixes or improvements. A better technique is to run a tool like cuda_memtest[2] periodically to check for errors.

Conclusions

More than 160 simulations of various sizes using XFdtd's XStream technology were run in various SMP and MPI configurations using recent and the very latest NVIDIA GPU hardware on NVIDIA's PSG Cluster. Performance of these simulations was analyzed. The following general conclusions were drawn:

- The NVIDIA K20X GPU currently offers the best performance in all cases, though it is expensive.
- The M2050 or M2070 currently offer best value in terms of cost/performance ratio.
- The intent of the user should be considered in determining how to distribute simulations on fixed resources. Except for extremely small (relative to total available GPU RAM) simulations, one should use all the GPUs to obtain the fastest simulation. On the other hand, if the intent is obtain the best overall throughput on available hardware, it is better to run multiple simulations, each using only a subset of the available GPUs.
- For the M2090, both MPI and SMP use have nearly N-speedup as long as the RAM occupancy on each CPU is greater than approximately 25%. Below this level, inter-rank communications becomes a bottleneck. Since these tests were performed using QDR half-bandwidth Infiniband, it is expected that QDR or FDR interconnections would extend N-speedup for an even higher number of nodes (or lower RAM occupancy level).
- One may obtain better performance by using multiple MPI ranks using one GPU each on multi-GPU machines rather than one rank using multiple GPUs.
- A dedicated high speed network interconnect should be used to obtain the best MPI performance, since it can increase performance a minimum of 4x even for the slowest Infiniband.

Remcom thanks NVIDIA Corporation for providing access to the PSG Cluster for performing the simulations used to generate this report, and Exxact Corporation for their assistance in gaining that access.

Contact Remcom for additional information: sales@remcom.com, www.remcom.com

[2] cuda_memtest is an open source project based on the well-known memtest86 program, available from http://sourceforge.net/projects/cudagpumemtest/