ARMY RESEARCH LABORATORY

Graphics Processing Unit (GPU) Performance on an *N*-Body Problem

by Pat Collins

ARL-CR-0629 **August 2009**

prepared by

Lockheed Martin Corporation
PMB-203
939-I Beards Hill Rd.
Aberdeen, MD 21001

under contract

GS04T08DBC0020

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
Aberdeen Proving Ground, MD 21005

ARL-CR-0629 **August 2009**

Graphics Processing Unit (GPU) Performance on an *N*-Body Problem

Pat Collins
Computational and Information Sciences Directorate, ARL

prepared by

Lockheed Martin Corporation
PMB-203
939-I Beards Hill Rd.
Aberdeen, MD 21001

under contract

GS04T08DBC0020

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
August 2009	Final	January to March 2009

4. TITLE AND SUBTITLE
Graphics Processing Unit (GPU) Performance on an *N*-Body Problem

5a. CONTRACT NUMBER
GS04T08DBC0020

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Pat Collins

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Lockheed Martin Corporation
PMB-203
939-I Beards Hill Rd.
Aberdeen, MD 21001

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-CR-0629

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
ATTN: RDRL-CIH-M
Aberdeen Proving Ground, MD 21005

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The objective of this study is to evaluate the performance of clusters of Nvidia graphics processing units on an *N*-body problem derived from the computation of vector potentials. Two clusters are used for this purpose. The first is a 2-node, Intel Xeon system with a single Tesla S870 system cross connected to each node. The second is a 20-node Opteron system with one Quadro FX 5600 GPU per node. The results show a significant increase in performance when GPUs accelerate the computation. With 16 GPUs and a sufficiently large problem, an estimated 3 teraflops is achieved.

15. SUBJECT TERMS
Graphics Processing Units, *N*-body, Vector Potential, CUDA

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Pat Collins
Unclassified	Unclassified	Unclassified	UU	26	19b. TELEPHONE NUMBER (*Include area code*) (410) 278-5061

Contents

List of Figures

List of Tables

Acknowledgments

This work was sponsored by the U.S. Army Research Laboratory's Advanced Computing and Computational Sciences Division in coordination with the Advanced Computing Strategic Technology Initiative.

INTENTIONALLY LEFT BLANK.

Executive Summary

This study measures the performance of clusters of Nvidia Graphics Processing Units (GPUs) on an N-body problem that computes the curl of a vector potential. Two clusters are used in this study. The first is a 2-node, Intel Xeon system with a single Tesla S870 system cross connected to each node. The second is a 20-node, Advanced Micro Devices (AMD) Opteron system with one Quadro FX 5600 GPU per node. The results from both of these systems show a significant increase in performance when GPUs are used to accelerate the computation. With 16 GPUs and a sufficiently large problem, an estimated 3 teraflops was achieved.

INTENTIONALLY LEFT BLANK.

1. Introduction

The objective of this study is to evaluate the performance of clusters of Nvidia Graphics Processing Units (GPUs) on an N-body problem using the direct method, or $O(N^2)$ algorithm. Computations of vector potentials were chosen because they have applications in many different fields. For instance, in computational fluid dynamics the curl of a vector potential relates vorticity to velocity, and in computational electromagnetics it relates current density to the magnetic field. Such computations generally require the solution of an N-body problem. Even with today's fast computers, solving this with an $O(N^2)$ algorithm places a significant limit on the size of the problem that can be solved in a reasonable amount of time. Fast multipole methods (FMM) can reduced this to $O(N \log N)$; however, the FMM still relies on an $O(N^2)$ algorithm for all near field interactions so enhancing the performance of the direct method is still relevant. Significant improvements in the performance of an FMM algorithm could be achieved when combined with a fast $O(N^2)$ algorithm, such as one implemented on a GPU, resulting in performance gains for a wide class of problems.

This problem maps well to the Nvidia GPU architecture because the computation requires $O(N^2)$ operations, but properly arranged, the communication requires only $O(N)$. The algorithm used to map this problem to the GPU is based on the one described in chapter 31 of reference 1 and implemented on a single GPU in the Nvidia Compute Unified Device Architecture (CUDA) Software Development Kit (SDK) code (*2*).

The rest of this report is arranged as follows: the N-body problem and the algorithm are described in sections 2 and 3, respectively. Section 4 gives the results and section 5 the conclusions.

2. N-Body Problem

The vector potential $\mathbf{A}$, as defined in Karamcheti (*3*), is

$$\mathbf{A}(\mathbf{r}, \mathbf{t}) = \frac{\mathbf{1}}{\mathbf{4}\pi} \int \int_{\mathbf{R}} \int \frac{\mathbf{\Omega}(\mathbf{s}, \mathbf{t})}{\mid \mathbf{r} - \mathbf{s} \mid} \, d\tau_{\mathbf{s}}, \tag{1}$$

where $\mathbf{\Omega}$ is the source distribution vector. The curl of $\mathbf{A}$, the objective of the computation and defined here as $\mathbf{V}$, is

$$\mathbf{V}(\mathbf{r}, \mathbf{t}) = \nabla_{\mathbf{r}} \times \mathbf{A}(\mathbf{r}, \mathbf{t}) = \frac{\mathbf{1}}{\mathbf{4}\pi} \int \int_{\mathbf{R}} \int \nabla_{\mathbf{r}} \times \frac{\mathbf{\Omega}(\mathbf{s}, \mathbf{t})}{\mid \mathbf{r} - \mathbf{s} \mid} \, d\tau_{\mathbf{s}} \tag{2}$$

where subscript $\mathbf{r}$ represents differentiation with respect to the coordinates of the evaluation point, or field point. The coordinates of the source points are represented as $\mathbf{s}$. The assumption of point sources of $\mathbf{\Omega}$ greatly simplifies the integral. Given an infinitesimally small region region of the domain, say $\delta\tau$, with source $\mathbf{\Omega}$ located at $\mathbf{s}$, the influence at $\mathbf{r}$ is approximated by

$$\delta\mathbf{V}(\mathbf{r},\mathbf{t}) = \frac{\mathbf{1}}{\mathbf{4}\pi}\nabla_{\mathbf{r}} \times \frac{\mathbf{\Omega}(\mathbf{s},\mathbf{t})}{\mid \mathbf{r}-\mathbf{s} \mid +\epsilon}\ \delta\tau_{\mathbf{s}}. \tag{3}$$

where ϵ is some small number introduced to prevent division by zero. The singularity is a result of the point source assumption; it integrates out in the full volume integral. In practice, a smoothing function is applied to remove the singularity. A complicated smoothing function would require additional logic that would impact GPU performance. This was not done. Therefore, with the simple smoothing function of adding ϵ to the denominator, the performance results from this study should be considered an upper bound.

Summing the influence of all source points on all field points is the N-body problem to be solved. The set consisting of the field points may or may not be equivalent to the set of field points. For this study, the sets of source and field points are chosen randomly and are different.

3. Algorithm

The algorithm used is based on the one described in chapter 31 of reference 1 and implemented in the "nbody" Nvidia CUDA SDK code (*2*). It is generalized to compute the vector potential and to handle cases where the number of sources and field points are different, and modified to run on multiple GPUs. For ease of programming, however, N and M are assumed to be powers of 2. Although the code was verified for cases where $N \neq M$, to avoid the additional complexity of comparing results, this study will present data only for $M = N$.

The influence of source points on field points can be represented as a matrix whose rows are associated with field points and columns with source points. If $\{a_{ij}\}$ is that matrix, then element a_{ij} is the influence of the j^{th} source point on the i^{th} field point. Summing the i^{th} row gives the influence of all source points on the i^{th} field point. The algorithm described here maps this computation efficiently to the GPU hardware. To use a GPU to its fullest, it strives to perform the following:

- create a sufficient number of thread blocks to mask memory access latencies with computations,

- maximize the computational work per memory access,
- minimize access to the device global memory (i.e., use shared memory where possible), and
- coalesce device global memory access to avoid conflicts.

The algorithm divides the matrix into sub-matrices, assigning the computational work accordingly while paying close attention to memory access patterns. These sub-matrices are referred to as computational tiles. Figure 1 shows an example of an influence matrix with $N = M = 16$. Each colored 4×4 sub-matrix represents a computational tile and each row of the matrix is handled by a single thread; hence, there is a one-to-one correspondences between threads and field points. In this example, the threads are grouped into four blocks of four threads each, where each thread block is assigned four computational tiles, see the left side of figure 1. Each tile operates on four source points.

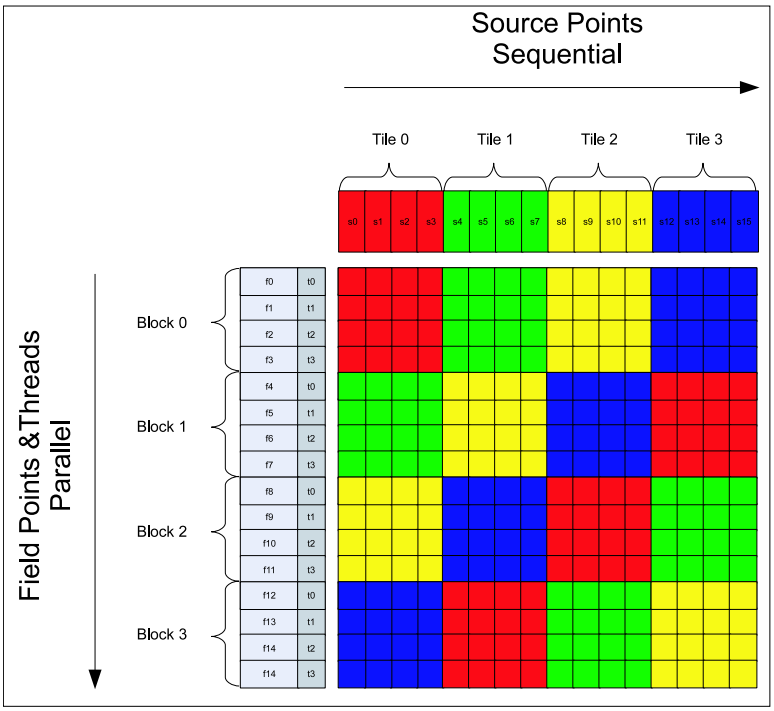

Figure 1. Computational tile structure for a 16×16 influence matrix.

The computational tiles are color coded to show data accessed patterns. At the top of figure 1, the source point data, as it resides in device global memory, is shown in red, green, yellow, and blue. As the computation proceeds serially, i.e., from left to right in figure 1, each block accesses a different chunk of device global memory to avoid access conflicts. For example, while working on tile 0, thread block 0 operates on red source data, block 2 on green, etc. As is obvious from the figure, each subsequent tile is arranged so that global

memory bank conflicts are avoided. (Displaying the temporal progression using the color scheme as in figure 1 takes liberties with the matrix analogy but hopefully without confusion.) Processing of each computational tile starts with an efficient copy of source point data from device global memory to processor shared memory. Each thread then accesses all the source point data residing in shared memory for its assigned computations.

The high level algorithm is as follows:

- Proc 0 reads input data and the precomputed solution.
- Proc 0 sends all source data to all processors.
- Proc 0 evenly distributes the field points to the processors.
- For each processor, do the following:
  - Initialize the assigned GPU device.
  - Start the compute timer.
  - Allocate the device global memory on the GPU for the source data, field points, and results.
  - Copy the source data and field points from the central processing unit (CPU) to the GPU.
  - Map the computation to the GPU hardware (i.e., define thread blocks).
  - For each thread block do the following:
    - Assign one thread per field point.
    - For each thread, do the following:
      - Copy a portion of the computational tile's assigned source data from device global memory to shared memory.
      - Compute the influence of all the tile's source points on the field points associated with the threads.
      - Write the results to the GPU global memory.
  - Copy the results from GPU to CPU.
  - Stop the compute timer.
- Proc 0 gathers all GPU compute times and reports the maximum time.
- Proc 0 gathers the results.
- Proc 0 compares the GPU solution to the precomputed solution.

This algorithm is written in C/C++ using CUDA (*4*) and open Message Passing Interface (MPI).

4. Results

4.1 Test Systems

The test systems used are referred to as the tesla and aspen clusters. The tesla cluster is a 2-node system where each node has a dual 3 GHz quad-core Intel Xeon processor. Connected to this system is an Nvidia S870 GPU computing system with four C870 GPUs. Two GPUs are connected to each of the nodes via a Peripheral Component Interconect (PCI) Express connector. The nodes are connected via a 1 Gbps Ethernet network. The aspen cluster is a 20-node system where each node consists of dual quad-core AMD Opteron 2350 processors and one Nvidia Quadro FX 5600 GPU. The first two nodes have two GPUs each; however, the second GPU was not used during the tests. Table 1 summarizes these properties and table 2 gives additional specifications on the GPUs. Except where noted, the data in table 2 applies to both the Tesla C870 (tesla cluster) and the Quadro FX 5600 (aspen cluster).

Table 1. Test systems.

	Aspen	**Tesla**
CPU	Dual Quad-Core AMD Opteron 2350	Dual Quad-Core Intel Xeon E5450
GPU	1 Quadro FX 5600 / node	2 Tesla C890 / node
Node Interconnect	Myri-10G Myrinet	Gigabit Ethernet
GPU Interconnect	MCP55 PCI Express bridge (33 MHz); Internal Connection	5400 Chipset PCI Express (33 MHz); External Connection w/ switch

Table 2. GPU properties.

Nvidia Tesla C870 and Quadro FX 5600	
Number of multi-processors (MP)	16
Number of threads processors per MP	8
Total number of thread processors	128
Major revision number	1
Minor revision number	3 (tesla); 0 (aspen)
Total amount of global memory	1610350592 bytes
Total amount of constant memory	65536 bytes
Total amount of shared memory per block	16384 bytes
Total number of registers available per block	8192
Warp size	32
Maximum number of threads per block	512
Maximum sizes of each dimension of a block	512 x 512 x 64
Maximum sizes of each dimension of a grid	65535 x 65535 x 1
Maximum memory pitch	262144 bytes
Texture alignment	0 (tesla); 256 (aspen)
Clock rate	1.35 GHz

4.2 Test Results

The results show significant decrease in compute times when GPUs are used. Figures 2 and 3 show the solution times in seconds on the aspen and tesla clusters, respectively, for various problem sizes and numbers of GPUs. A thread block size of 256 is used for these calculations. The x-axis is the problem size and the y-axis is GPU compute times measured, as indicated in section 3. Compute times using a single CPU core on each cluster are also shown for comparison. The problem sizes run from $N = 2^{12}$ through 2^{18}. Figures 4 and 5 show the estimated gigaflops for each of the same runs. These estimates are based on 29 floating point operations per source point-field point interaction.

GPU performance on the aspen cluster exceeds the CPU performance even for the smallest problem size considered, $N = 2^{12}$, but good scalability is not achieved until $N > 2^{16}$. The tesla cluster shows similar performance for large N; however, for relatively small N, where communication latencies are more apparent, performance is reduced (see figures 2 and 3). On the tesla cluster, two C870 GPUs are connected to a node through a single PCI Express external interface and a switch. The aspen cluster has one GPU per node connected through an internal PCI connection. (Two nodes of the aspen cluster have two GPUs per node but no more then one was used during these tests.) The performance difference for small N between the aspen and tesla clusters with a single GPU is due to the different PCI communication latencies. The fact that performance further degrades for small N when more then one GPU is used on the tesla cluster, see figure 3, is because two GPUs share a single PCI interface through a switch.

The results are most dramatic when N is large. For $N = 2^{18}$, the solution time is 10 s on a single Nvidia Quadro FX 5600 GPU (200 gigaflops) versus 33 min (1 gigaflop) on the native Opteron processor. On the tesla cluster, it takes 11 s (174 gigaflops) when using a single Nvidia Tesla C870 GPU versus 33 min (1 gigaflop) on the native Xeon processor. Even more impressive is that it takes less then 1 s to solve such a large N-body problem on the aspen cluster using 16 GPUs, achieving almost 3 teraflops. It should be noted that with more effort the CPU performance may have been improved, especially if more than one core is used, but that was not the objective. Besides, the GPU performance far exceeds what would theoretically be possible on the CPUs, so there was little motivation for this comparison.

Theoretical peak gigaflops on each of these GPUs is advertised at 518 gigaflops; however, this counts Multiply ADD (MADD) instructions as two floating point operations. On the CUDA Zone Forum, some believe a more reasonable number is between 170 and 340 gigaflops. This is in line with the results obtained in this study.

Figures 6 and 7 show the speed-up achieved using multiple GPUs relative to a single GPU for various problem sizes. The data shows good scalability is achieved only for large problems. On the tesla cluster, the communication overhead for small problems relative to the computational work is high causing the speed-up to drop below 1.

Figure 8 shows the effect of thread block size on a single GPU of the tesla cluster. The data indicates that for block sizes greater than 32, the performance is essentially the same.

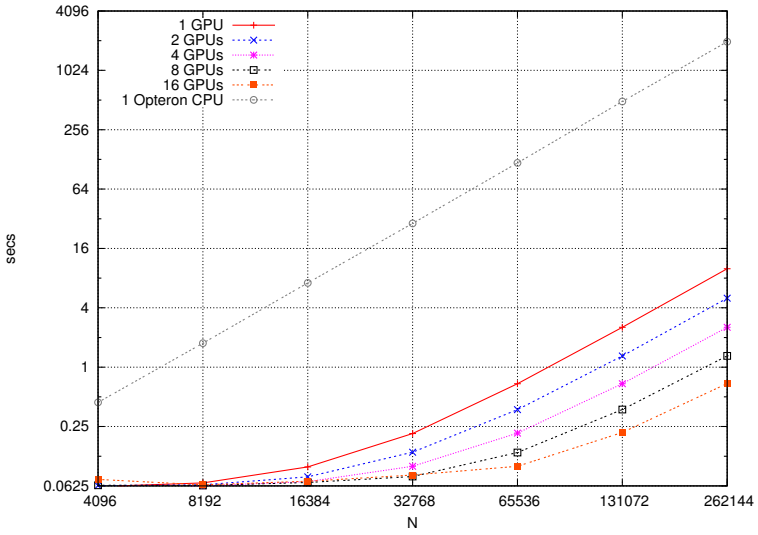

Figure 2. Compute times on the aspen cluster.

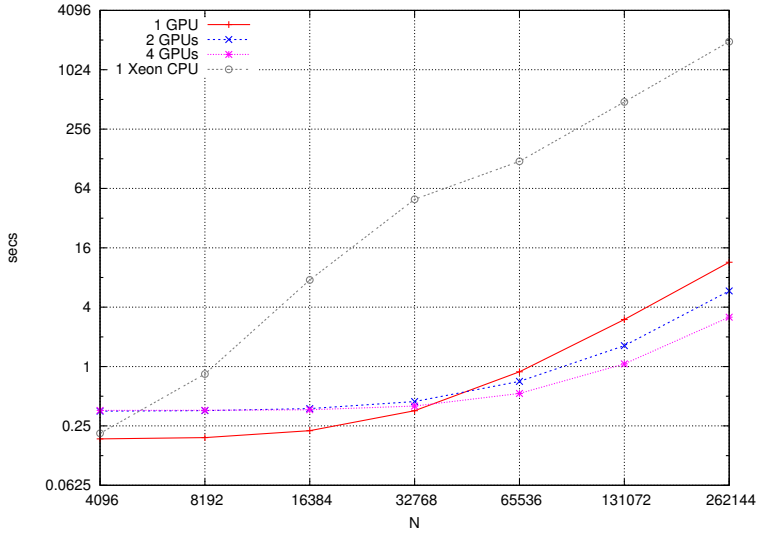

Figure 3. Compute times on the tesla cluster.

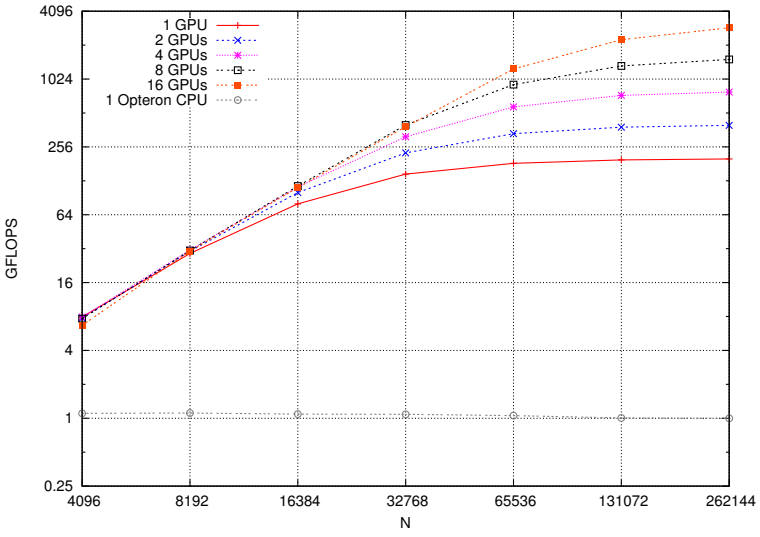

Figure 4. Approximate gigaflops on the aspen cluster.

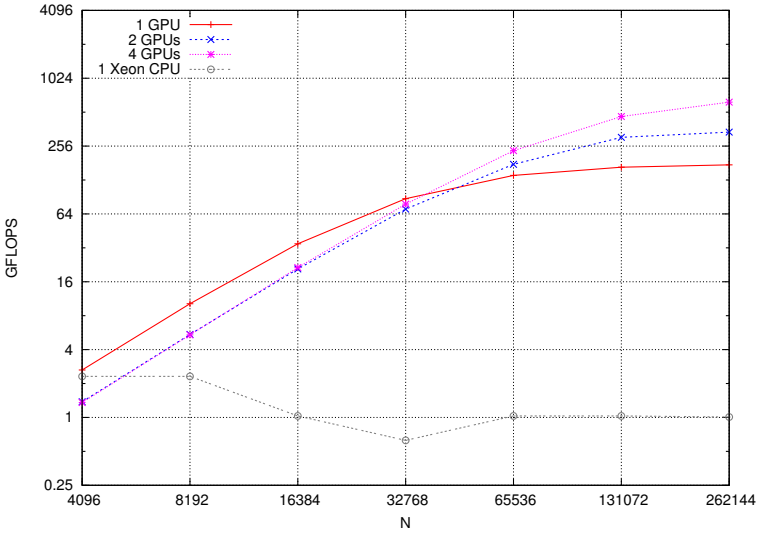

Figure 5. Approximate gigaflops on the tesla cluster.

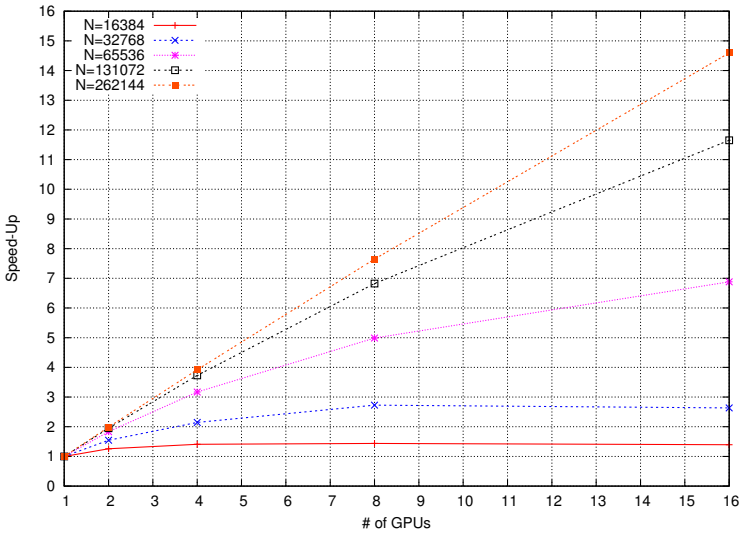

Figure 6. Scalability on the aspen cluster.

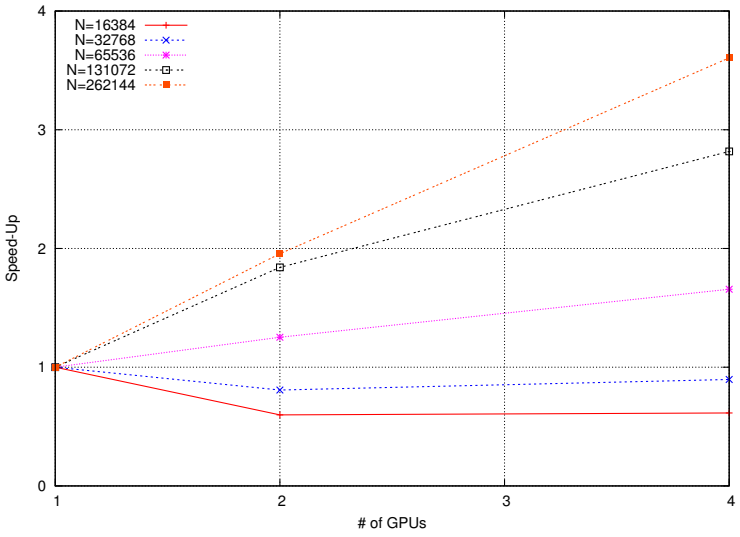

Figure 7. Scalability on the tesla cluster.

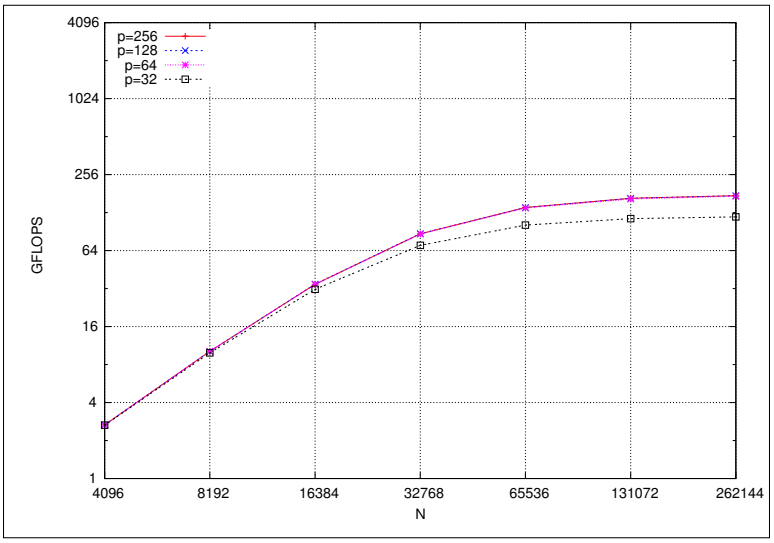

Figure 8. Single GPU performance for various thread block sizes.

Tables 3 and 4 give the L_2 and L_∞ norms of the CPU solution, V, and GPU solution, W, and the norms of their difference and relative difference for various problem sizes on the aspen cluster with eight GPUs. These norms are computed as follows. If V_i^j is the CPU solution where j is the index of the field point and i the index of the solution vector at j, and if W_i^j is the same for the GPU computed solution, then the norms are given by

$$\|V\|_2 = \sqrt{\frac{\sum_{j=1}^{N}\sum_{i=1}^{3} {V_i^j}^2}{3N}}, \quad \|V\|_\infty = \max_{i,j} |V_i^j| \tag{4}$$

$$\|W-V\|_2 = \sqrt{\frac{\sum_{j=1}^{N}\sum_{i=1}^{3} \left(W_i^j - V_i^j\right)^2}{3N}}, \quad \|W-V\|_\infty = \max_{i,j} |W_i^j - V_i^j| \tag{5}$$

$$\|W-V\|_{2_r} = \sqrt{\frac{\sum_{j=1}^{N}\sum_{i=1}^{3} \left(\frac{W_i^j - V_i^j}{V_i^j}\right)^2}{3N}}, \quad \|W-V\|_{\infty_r} = \max_{i,j} |\frac{W_i^j - V_i^j}{V_i^j}| \tag{6}$$

with similar formulas for $\|W\|_2$ and $\|W\|_\infty$.

Table 3. L_2 norms of the solutions and their differences.

N	$\|V\|_2$ (CPU)	$\|W\|_2$ (GPU)	$\|W-V\|_2$	$\|W-V\|_{2_r}$
4096	7.36869e+00	7.36869e+00	1.94694e-06	5.96216e-05
8192	5.70731e+00	5.70731e+00	4.92994e-06	1.31579e-05
16384	1.34827e+01	1.34827e+01	2.66668e-05	2.55242e-05
32768	2.22855e+01	2.22855e+01	4.13682e-05	7.43773e-03
65536	5.22089e+01	5.22092e+01	5.15852e-04	1.55937e-04
131072	1.06966e+02	1.06966e+02	9.58753e-04	2.65740e-04
262144	1.68857e+02	1.68857e+02	9.53178e-04	3.86570e-04

Table 4. L_∞ norms of the solutions and their differences.

N	$\|V\|_\infty$ (CPU)	$\|W\|_\infty$ (GPU)	$\|W-V\|_\infty$	$\|W-V\|_{\infty_r}$
4096	5.55356e+02	5.55356e+02	9.15527e-05	3.09944e-06
8192	1.91733e+02	1.91733e+02	1.83105e-04	1.10865e-05
16384	1.08433e+03	1.08433e+03	4.76074e-03	4.19617e-05
32768	9.40967e+02	9.40967e+02	1.98364e-03	2.67029e-04
65536	1.15320e+04	1.15323e+04	2.22656e-01	5.07355e-04
131072	2.63124e+04	2.63122e+04	5.12695e-01	5.15580e-04
262144	8.35248e+03	8.35227e+03	2.07031e-01	4.88281e-03

Roundoff errors cause the solutions to be slightly different as indicated by the data. As N increases, the norms of the differences increase for two reasons. The errors accumulate because of the increased operation count and the absolute errors increase because the magnitude of the solution increases. This happens because the physical domain, which is chosen to be the unit cube, remains the same but the number of source points increases. Based on the data, the solutions are considered numerically the same within roundoff.

5. Conclusions

The results show that this N-body problem achieves a significant 200 gigaflops on a single GPU and almost 3 teraflops on a cluster of 16 GPUs. As a comparison, if one could achieve 3 gigaflops on a single 3 GHz CPU, then it would require at least 66 CPUs to equal 1 GPU. This represents a lower bound on the number of CPUs needed. One must, of course, pay a cost to enjoy such results. This cost is an increase in programming complexity. The N-body solution is straight forward and easy to program on a CPU. On the GPU, however, it has to be tailored to the architecture. In this case, the problem was broken up into computational tiles and much care was taken to avoid memory access conflicts. The measured performance gains, however, more than compensated for the increased complexity. Moreover, the Nvidia CUDA programming language greatly eased the programming task.

References

[1] Nguyen, H. *GPUGems 3*; Addison-Wesley Professional, August 2007.

[2] Nvidia Corporation, 2008. Nvidia CUDA SDK 10 Linux Version 1.10.1203.1115.

[3] Karamcheti, K. *Principles of Ideal-Fluid Aerodynamics*; Robert E. Krieger Publishing Company, 1966.

[4] Nvidia Corporation, 2009. http://www.nvidia.com (last accessed in March 2009).

List of Symbols, Abbreviations, and Acronyms

AMD	advanced micro devices
CPU	central processing unit
CUDA	compute unified device architecture
FMM	fast multipole method
GPU	graphics processing unit
MADD	Multiply ADD
MPI	message passing interface
PCI	peripheral component interconnect
SDK	software development kit

NO. OF COPIES	ORGANIZATION
1 ELEC	ADMNSTR DEFNS TECHL INFO CTR ATTN DTIC OCP 8725 JOHN J KINGMAN RD STE 0944 FT BELVOIR VA 22060-6218
1 HC	DARPA ATTN IXO S WELBY 3701 N FAIRFAX DR ARLINGTON VA 22203-1714
1 CD	OFC OF THE SECY OF DEFNS ATTN ODDRE (R&AT) THE PENTAGON WASHINGTON DC 20301-3080
1 HC	US ARMY RSRCH DEV AND ENGRG CMND ARMAMENT RSRCH DEV AND ENGRG CTR ARMAMENT ENGRG AND TECHNLGY CTR ATTN AMSRD AAR AEF T J MATTS BLDG 305 ABERDEEN PROVING GROUND MD 21005-5001
1 HC	PM TIMS, PROFILER (MMS-P) AN/TMQ-52 ATTN B GRIFFIES BUILDING 563 FT MONMOUTH NJ 07703
1 HC	US ARMY INFO SYS ENGRG CMND ATTN AMSEL IE TD A RIVERA FT HUACHUCA AZ 85613-5300
1 HC	COMMANDER US ARMY RDECOM ATTN AMSRD AMR W C MCCORKLE 5400 FOWLER RD REDSTONE ARSENAL AL 35898-5000
1 HC	US GOVERNMENT PRINT OFF DEPOSITORY RECEIVING SECTION ATTN MAIL STOP IDAD J TATE 732 NORTH CAPITOL ST NW WASHINGTON DC 20402

NO. OF COPIES	ORGANIZATION
2 HCS	UNIV OF MARYLAND DEPT OF MECHANICAL ENGINEERING ATTN P BERNARD GLENN L MARTIN HALL COLLEGE PARK MD 20742-5121
5 HCS	HPCMO DOD HIGH PERFORMANCE COMPUTING MODERNIZATION PROGRAM OFFICE ATTN C HENRY ATTN DR L DAVIS ATTN DR R CAMPBELL ATTN B COMES ATTN DR A MARK 10501 FURNACE ROAD, SUITE 101 LORTON, VA 22079
2 HCS	US ARMY RDECOM-TARDEC ATTN AMSRD TAR R T CURRIER, MS157 D GORSICH, MS205 WARREN, MI 48397-5000
17 HCS	US ARMY RSRCH LAB ATTN RDRL CIH B SHEROKE C NIETUBICZ D THOMPSON ATTN RDRL CIH C B HENZ D RICHIE D SHIRES J CLARKE K KIRK M POTTS P CHUNG S DINAVAHI S PARK ATTN RDRL CIH M P COLLINS M KNOWLES M MOTSKO ATTN RDRL CIH S D BROWN T KENDALL ABERDEEN PROVING GROUND MD 21005

NO. OF COPIES	ORGANIZATION
1 HC	US ARMY RSRCH LAB ATTN RDRL VTU V S WILKERSON BLDG 390 ABERDEEN PROVING GROUND MD 21005
3 HC	US ARMY RSRCH LAB ATTN RDRL WMB C J SAHU ATTN RDRL WMB C K HEAVEY ATTN RDRL CIM G T LANDFRIED BLDG 4600 ABERDEEN PROVING GROUND MD 21005
1 HC	US ARMY RSRCH LAB ATTN RDRL WMB D M NUSCA ABERDEEN PROVING GROUND MD 21005-5056
4 HC	DIRECTOR US ARMY RSRCH OFFICE ATTN RDRL ROI M J MYERS DR J M COYLE ATTN RDRL ROI C DR C WANG, CHIEF ATTN RDRL ROE DR T DOLIGALSKI ACTING DIRECTOR PO BOX 12211 RESEARCH TRIANGLE PARK NC 27709
1 HC	US ARMY RSRCH LAB ATTN RDRL CIH C J ROSS ABERDEEN PROVING GROUND MD 21005
4 HCS	US ARMY RSRCH LAB ATTN RDRL CIM P TECHL PUB ATTN RDRL CIM L TECHL LIB ATTN RDRL CI R NAMBURU ATTN IMNE ALC HRR MAIL & RECORDS MGMT ADELPHI MD 20783-1197

TOTAL: 48 (1 PDF, 1 CD, 46 HCS)