




Paper information and file upload

Please complete the "Description of the paper" section and upload the paper in the "Upload files" section.

Help on how to fill out the form and upload papers can be found by clicking on the question mark at each section: .

Description of the paper

Contact person: **T. IITAKA, RIKEN**

Workshop: General purpose computation on graphics hardware (GPGPU): Methods, algorithms and applications

Author initials:

Author family name:

Author e-mail:

Author affiliation:

Co-authors: *None*

[Add co-author\(s\)](#)

Presenter initials:

Presenter family name:

Presenter e-mail:

Presenter affiliation:

Paper title:

Page count:

Abstract:

We report our experience of accelerating numerical calculations by Graphic Processing Unit (GPU) for the case of Molecular Dynamics simulation. The sustained performance of one GPU chip (GForce7800GTX) was found to be about 15 Gflops or 70 times of CPU calculation. This performance is compared with that of chips specially designed for molecular dynamics calculation. We show also how the algorithm for molecular dynamics calculation can be applied to solving integral equations.

Please don't forget to save the information by clicking on the "Save Paper Information" button **before** uploading files or doing anything else.

Save Paper Information

Themes

Below is the list of themes of which you indicated they best fit the paper. Themes can be assigned or unassigned by clicking on the icons at the top-right corner above the themes.

Assign  Unassign 

Themes for this paper

- Modelling and simulation in the natural sciences
- Hybrid computational methods
- Software engineering for computational science

Uploaded files 

All uploaded files are shown in this section. An uploaded file can be deleted from the server by clicking on the trashcan icon.

 mdgpu.pdf	111 kB, 19-Dec-2005 06:57	
mdgpu.tex	20 kB, 19-Dec-2005 06:57	

Upload new files 

Each paper should be uploaded in two versions - as **postscript** or **PDF** and as **source** (LaTeX or MS Word).

The submitted paper must be camera-ready and formatted according to the rules of LNCS. Full papers must not exceed eight pages in the LNCS format.

In order to upload files, just press the browse button and select a file on your local system.

Then press the "submit file(s)" button to send the files to our server. Depending on the size of the file and the bandwidth, this can take some time. You can compress (zip or gzip) files before uploading.

Postscript or PDF:

Source file:

We find that some authors have problems with uploading. This apparently affects especially larger files, and appears to be caused by the author's local proxy server.

Possible solutions are:

- *Compress (zip or gzip) the files before uploading*
- *Bypass your proxy server (in your browser preferences)*

Review results

Below are the results of the reviews of your paper as submitted by various reviewers.

	Very bad	Bad	Unsatisfactory	Barely acceptable	Acceptable	Good	Very good	Outstanding
Readability		✓						
Originality		✓						
Significance				✓				
Relevance				✓				
Overall		✓						

Reject	Marginal	Accept	Strong accept		Poster
✓					

Comments: First, I have to say that the presentation of the paper is very poor. The grammar is doubtful in places, and the paper needs some proper spellchecking.

That being said, the content is also poor. N-body simulations have been done before, and you even quote shader code from other authors which is not good scientific practice. You fail to properly explain why your code crashes, and in the end, your application is even slower than previously published results.

The comparison against fixed-purpose hardware is nice, but that alone does not make a good paper. I simply fail to see any novel contribution to the field of GPGPU.

	Very bad	Bad	Unsatisfactory	Barely acceptable	Acceptable	Good	Very good	Outstanding
Readability					✓			
Originality			✓					
Significance			✓					
Relevance						✓		
Overall			✓					

Reject	Marginal	Accept	Strong accept		Poster
✓					

Comments: This is a practical application of the N-body algorithm on the GPU but with little new substance. It would be a stronger paper if the performance and accuracy of the proposed schemes were evaluated in detail or the more general problem of integral equations had been treated.

Concerning the inconsistent behavior of the loop for large N , the author may want to try to replace 'uniform int' with 'uniform float' as ints are not native GPU types which may cause the problem. Note that the largest working size $128 \times 32 = 2^{12}$, where the fixed type is 12 bit wide.

Please use index ij for distance r , capitalize section titles and add explanatory comments to the Cg code.

	Very bad	Bad	Unsatisfactory	Barely acceptable	Acceptable	Good	Very good	Outstanding
Readability					✓			
Originality		✓						
Significance			✓					
Relevance			✓					
Overall			✓					

Reject	Marginal	Accept	Strong accept	Poster
✓				✓

Comments: *The paper's language needs some polish, as does the layout. All in all it is readable, but:*

- "i-particle" should read "the i th particle"
- "texture search" should read "texture fetch". The GPU knows quite precisely where its textures are.
- section 3 is totally obsolete. If the algorithm is presented elsewhere, there is no reason to put it into this paper as well.
- The sentence "Table 5 shows that the performance of each chip is compared" should read "Table 5 shows a comparison of the three chips".
- The "inert gas" should probably be an "idle gas".
- in the abstract, change "GForce7800GTX" to "GeForce 7800 GTX". This also fixes your overfull hbox.

My main concern is that the paper's contribution is rather marginal. You implement a somewhat different approach to the N -body problem than what was implemented by Mark Harris, trivially exploiting some of the newer GPUs' features. However, this does not result in any performance increase, so I do not see why I should use your method over Harris' method. Furthermore, the entire idea of solving N -body problems in $O(N^2)$ is dubious by itself. There exist some considerable amount of literature on approximating the problem in $O(N)$, called "fast N -body solvers". I would expect that such an $O(N)$ approximation would yield numerical results close to yours (due to limited GPU precision) in a far less amount of time on the CPU. By nature, these algorithms do not well port to the GPU, but of course you should be comparing fastest possible implementations against each other. These fast N -body algorithms are completely missing in your discussion. Also, as far as I understood, you are going to download the data from GPU to CPU to perform the $O(N)$ pass. What time requirements arise when doing so? Please include them in your GPU timings, as they are not present in a CPU solution.

The Lennard-Jones Potential is an approximation that is also valid for several non-ideal gases - with different coefficients.

You should state what kind of particles you are simulating - neglecting friction and inertia effects is not acceptable for any type of particles.

Please, do not provide shaders. They only contain marginal changes to previous ones, in table 1 they seemingly are even identical. Linearly approximating the Lennard-Jones Potential seems numerically dubious, as it is a highly non-linear function. Please discuss the resolution of the lookup textures as well as the internal format.

The systems simulated seem to be rather small from a physical point of view, and blocking

them into 2Kx2K textures has performance implications not discussed in the paper.

A log-step reduce-add is required to avoid round-off errors when summing up on the order of millions of values. This may explain why your first approach does not work with larger N.

The premise that one long for-loop is faster than a lot of small passes is not true in general. Your timings even show that the multi-pass approach is faster.

One reason why the MDGRAPE systems are successful might be their precision, that is yet unmatched by the GPU (and most CPU implementations). Please compare the error of your simulation to both CPU and MDGRAPE implementations.

Last but not least: What kind of CPU do you have in mind, when you claim that 15GFlops are 70x CPU performance? The pentium 4 3.0GHz processor yields a sustained performance of 3-4 GFlops, while its theoretical peak lies at 12GFlops. Please back up such claims with references. Linear interpolation is about the only task that the GPU can do about 70-100x faster than a CPU.

Summing up: Actually the paper shows more or less that the approach is not working too good. It is slower than previous approaches and it has numerical trouble (timing has nothing to do with it as long as you do not read and write into one target at the same time). Consequently I would suggest fixing the paper with the above comments and re-submitting it as a case study to another conference.