Star clusters and black holes in galaxies across cosmic time
Proceedings IAU Symposium No. 312, 2014
Y. Meiron, S. Li, F.-K. Liu & R. Spurzem, eds.

doi:10.1017/S1743921315007966

Performance evaluation of the Hermite scheme on many-core accelerators

Naohito Nakasato

Department of Computer Science and Engineering, University of Aizu,
Aizu-Wakamatsu, Fukushima, 965-8580, Japan
email: nakasato@u-aizu.ac.jp

Abstract. We are developing a software library to calculate gravitational interaction for the Hermite scheme on parallel computing systems supported by OpenCL API. Our library is partly compatible with a standard GRAPE-6A interface and is easily usable in existing N-body codes. Since our library is based on OpenCL standard API, our library is working on many parallel computing systems such as a multi-core CPU, a GPU, and a many-core architecture. We report the performance evaluation of our library on computing platforms from various vendors.

Keywords. methods: N-body simulations

1. Introduction

Recent development of parallel computing architecture enable us to use multi/many-core architecture such as CPUs, GPUs and Xeon Phi(MIC) for modeling star clusters (e.g. Gaburov *et al.* 2009; Tanikawa *et al.* 2012; Nitadori & Aarseth 2012; Wang *et al.* 2015). In this paper, we report a new library to speed-up the Hermite scheme (Makino & Aarseth 1992) on the emerging parallel architectures. Our library utilizes the OpenCL API that is a standard API working on many parallel architectures while previous techniques/libraries rely on a specific computing platform. We make our library compatible to the GRAPE-6A library (Fukushige *et al.* 2005) so that it is usable as a replacement to the GRAPE library. In the following section, we present optimization techniques used in our library to gain high performance on GPUs and report the performance evaluation on different GPU architectures.

2. Performance optimization

Fundamentally, there are I- and J- parallelism in GRAPE-like evaluation of N-body kernels. "I-particle" is a particle where we want to compute the force. "J-particles" are particles that exert the force on the I-particle. In our current implementation, each thread running on a GPU first loads N_i I-particles and loop-over all other particles (J-particles) to compute particle interaction such as potential energy, acceleration and jerk for the I-particles (Makino & Aarseth 1992). In this loop, the thread loads N_j particles at a time. A simplest case is $(N_i,\ N_j) = (1,\ 1)$ where each thread compute one-to-one interaction on every iteration.

A combination of $(N_i,\ N_j)$ is a key to gain optimal performance on GPUs since GPU architectures favor more floating-point operations per memory access operations. Larger N_i and N_j in an N-body kernel, the number of arithmetic operations per unit memory access is larger so that the performance of the kernel is better. In Figure 1, we compare the performance in GFLOPS with a different combination of $(N_i,\ N_j)$. For this comparison,

Table 1. Evaluated Architectures and Optimal (N_i, N_j)

Architecture	SP peak	DP peak	Kernel D	Kernel DS
FirePro W9100	5337	2619	(1,2)/987	(2,2)/1300
GeForce TITAN	5376	1570	(1,2)/333	(1,1)/793
Radeon 7970	3789	947	(1,1)/626	(2,2)/1063
MIC 5110P	1888	994	(1,2)/188	(2,2)/255

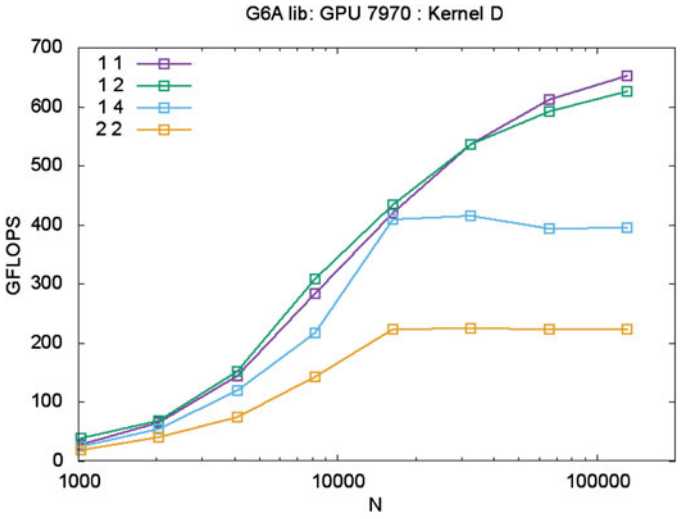

Figure 1. Performance of our library as a function of N with different combinations of (N_i, N_j)

we have used Radeon 7970 GPU and N-body kernels are implemented with double-precision (DP) operations. In this particular case, $(N_i, N_j) = (1,1)$ and $(1,2)$ show the comparable performance. We did similar evaluations on other parallel architectures and found an optimal combination for a given architecture as shown in Table 1. In this table, each row shows GPUs and MIC. The second and third columns show the peak GFLOPS in single-precision (SP) and DP operations, respectively. The forth and fifth columns present the optimal combination of (N_i, N_j) and the performance in GFLOPS for $N = 131072$. We have tested two variants of kernels as explained below. Roughly speaking, the performance of "Kernel D" that uses only DP operations is proportional to the peak performance in DP operations except GeForce TITAN. TITAN GPU favors a proprietary CUDA programming API and other authors have reported much better performance than our result on CUDA-enabled GPUs (Gaburov *et al.* 2009; Nitadori & Aarseth 2012).

Also, we have appropriately used SP and DP operations in particle interactions since GPUs are much faster on SP operations as shown in Table 1. Accordingly, we have two variant of N-body kernels as Kernel D and Kernel DS. "Kernel D" uses only DP operations while "Kernel DS" adopts a mixed-precision technique that is similar to Gaburov *et al.* (2009). Note that we use actual DP operations instead of emulated DP operations by combining 2 SP variables to represent emulated DP variable. In addition, we use vector type variables defined in OpenCL language such as float4, double2 to explicitly express vectorized evaluation of the particle interaction. In Figure 2, we present the performance of our library as a function of N. "Kernel DS" shows better performance

https://doi.org/10.1017/S1743921315007966 Published online by Cambridge University Press

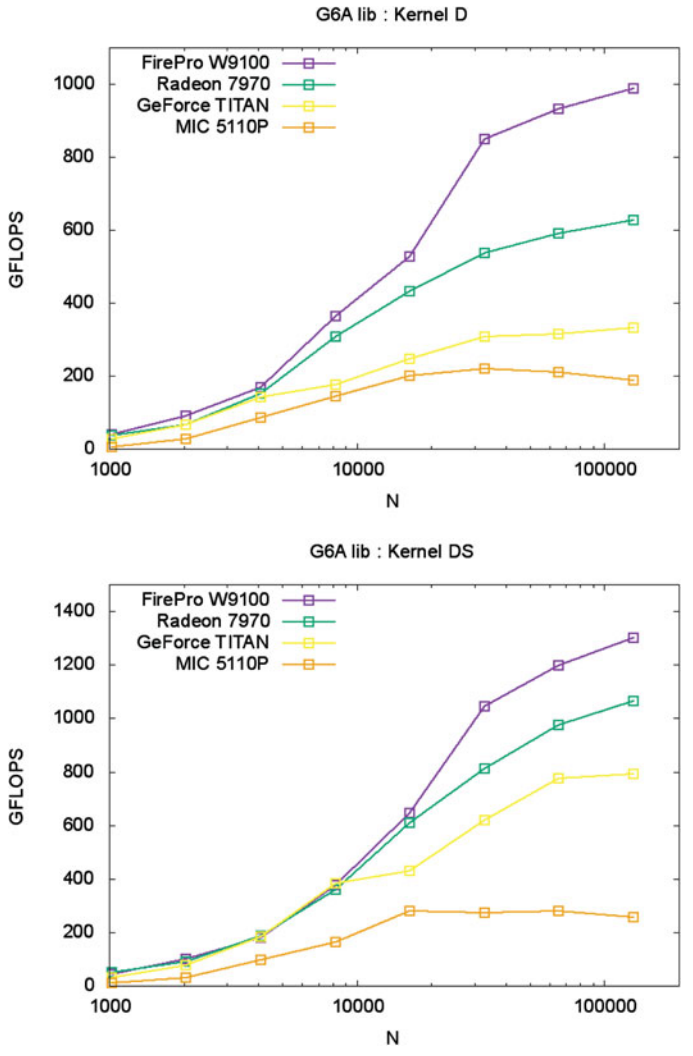

Figure 2. Performance of two variants of kernels as a function of N

than "Kernel D" over 1 TFLOPS on W9100 and 7970 GPUs. Our library also works on multi-core CPUs thanks to the standard OpenCL API.

Finally, we have further optimized Kernel DS by adopting different data storage. Performance of compute kernels on GPUs is sensitive to memory access pattern of the kernels. In "Kernel DS", we use separate arrays to store the position, velocity and mass of particles. This usage of data storage is a so-called structure of arrays (SoA). When a thread load the data of I- or J-particles, the memory access is stride access in SoA. Alternative way to store particles data is to use a structure for each particle that holds the the position, velocity and mass of the particle. This is a so-called array of structures (AoS). In Figure 3, we present the comparison of the performance with the data storage as SoA and AoS on 7970 GPU. The two integers are the combination of (N_i, N_j). AoS is 20 - 40 % better performance than SoA at $N = 131072$.

https://doi.org/10.1017/S1743921315007966 Published online by Cambridge University Press

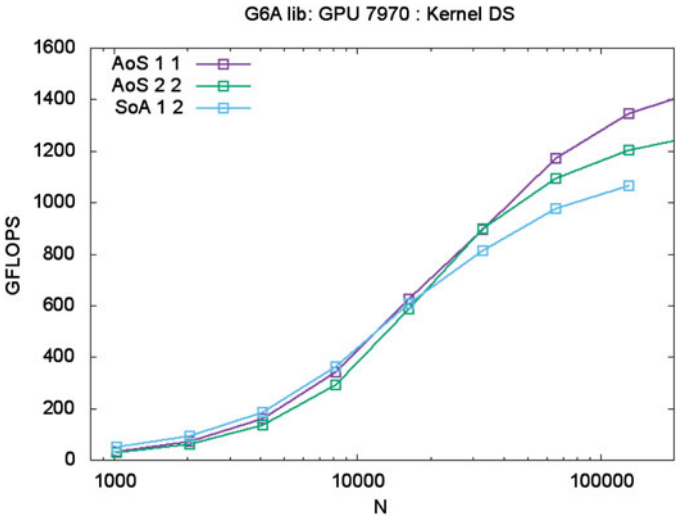

Figure 3. Performance of two variants of kernels as a function of N

3. Summary

Our GRAPE-6A compatible Hermite Scheme library shows fairly good performance on various GPU/MIC architectures. Since the OpenCL standard works on many platforms, our approach is a very effective way to implement a similar library for N-body integrations.

References

Toshiyuki, F., Junichiro, M., & Atsushi, K., 2005, *Publications of the Astronomical Society of Japan*, 57, 6, 1009–1021

Gaburov, E., Harfst, S., & Portegies Zwart, S., 2009, *New Astronomy*, 14, 7, 630–637

Makino, J., & Aarseth, S. J., 1992, *Publications of the Astronomical Society of Japan*, 44, 2, 141–151

Nitadori, K., & Aarseth, S. J., 2012, *Monthly Notices of the Royal Astronomical Society*, 424, 1, 545–552

Tanikawa, A., Yoshikawa, K., Okamoto, T., & Nitadori, K., 2012, *New Astronomy*, 17, 2, 82–92

Wang, L., Spurzem, R., Aarseth, S. J., Nitadori, K., Berczik, P., Kouwenhoven, M. B. N., & Naab, T., 2015, *Monthly Notices of the Royal Astronomical Society*, 450, 4, 4070–4080

https://doi.org/10.1017/S1743921315007966 Published online by Cambridge University Press