Contribution ID: **128** Type: **Plenary**

Application of Many-core Accelerators for Problems in Astronomy and Physics

Friday 26 February 2010 10:40 (40 minutes)

Recently, many-core accelerators are developing so fast that the computing devices attract researchers who are always demanding faster computers. Since many-core accelerators such as graphic processing unit (GPU) are nothing but parallel computers, we need to modify an existing application program with specific optimizations (mostly parallelization) for a given accelerator.
In this paper, we describe our problem-specific compiler system for many-core accelerators, specifically, GPU and GRAPE-DR. GRAPE-DR is another many-core accelerators device that is specially targeted scientific applications.
In our compiler, we focus a compute intensive problem expressed as two-nested loop.Recently, many-core accelerators are developing so fast that the computing devices attract researchers who are always demanding faster computers.
Since many-core accelerators such as graphic processing unit (GPU) are nothing but parallel computers, we need to modify an existing application program with specific optimizations (mostly parallelization) for a given accelerator.
In this paper, we describe our problem-specific compiler system for many-core accelerators, specifically, GPU and GRAPE-DR. GRAPE-DR is another many-core accelerators device that is specially targeted scientific applications.
In our compiler, we focus a compute intensive problem expressed as two-nested loop. Our compiler ask a user to write computations in the inner-most loop. All details related to parallelization and optimization techniques for a given accelerator are hidden from the user point of view. Our compiler successfully generates the fastest code ever for astronomical N-body simulations with the performance of 2600 GFLOPS (single precision) on a recent GPU.
However, this code that simply uses a brute-force $O(N2)$ algorithm is not practically useful for a system with $N > 100,000$. For more lager system, we need a sophisticated $O(N\log N)$ force evaluation algorithm, e.g., the oct-tree method. We also report our implementation of the oct-tree method on GPU. We successfully run a simulation of structure formation in the universe very efficiently using the oct-tree method. Another successful application on both GPU and GRAPE-DR is the evaluation of a multi-dimensional integral with quadruple precision. The program generated by our compiler runs at a speed of 5 - 7 GFLOPS on GPU and 3 - 5 on GRAPE-DR.
This computation speed is more than 50 times faster than a general purpose CPU.Recently, many-core accelerators are developing so fast that the computing devices attract researchers who are always demanding faster computers. Since many-core accelerators such as graphic processing unit (GPU) are nothing but parallel computers, we need to modify an existing application program with specific optimizations (mostly parallelization) for a given accelerator.
In this paper, we describe our problem-specific compiler system for many-core accelerators, specifically, GPU and GRAPE-DR. GRAPE-DR is another many-core accelerators device that is specially targeted scientific applications.
In our compiler, we focus a compute intensive problem expressed as two-nested loop. Our compiler ask a user to write computations in the inner-most loop.
All details related to parallelization and optimization techniques for a given accelerator are hidden from the user point of view. Our compiler successfully generates the fastest code ever for astronomical N-body simulations with the performance of 2600 GFLOPS (single precision) on a recent GPU.
However, this code that simply uses a brute-force $O(N2)$ algorithm is not practically useful for a system with $N > 100,000$. For more lager system, we need a sophisticated $O(N\log N)$ force evaluation algorithm, e.g., the oct-tree method.
We also report our implementation of the oct-tree method on GPU. We successfully run a simulation of structure formation in the universe very efficiently using the oct-tree method. Another successful application on both GPU and GRAPE-DR is the evaluation of a multi-dimensional integral with quadruple precision. The program generated by our compiler runs at a speed of 5 - 7 GFLOPS on GPU and 3 - 5 on GRAPE-DR. This computation speed is more than 50 times faster than a general purpose CPU.

Our compiler ask a user to write computations in the inner-most loop.
All details related to parallelization and optimization techniques for a given accelerator are hidden from the user point of view.
Our compiler successfully generates the fastest code ever for astronomical N-body simulations with the performance of 2600 GFLOPS (single precision) on a recent GPU.
However, this code that simply uses a brute-force $O(N2)$ algorithm is not practically useful for a system with $N > 100,000$. For more lager system, we need a sophisticated $O(N\log N)$ force evaluation algorithm, e.g., the oct-tree method. We also report our implementation of the oct-tree method on GPU. We successfully run a simulation of structure formation in the universe very efficiently using the oct-tree method. Another successful application on both GPU and GRAPE-DR is the evaluation of a multi-dimensional integral with quadruple precision.
The program generated by our compiler runs at a speed of 5 - 7 GFLOPS on GPU and 3 - 5 on GRAPE-DR.
This computation speed is more than 50 times faster than a general purpose CPU.

Author: NAKASATO, Naohito (University of Aizu)

Presenter: NAKASATO, Naohito (University of Aizu)

Session Classification: Friday, 26 February - Plenary Session

Track Classification: Methodology of Computations in Theoretical Physics