Improving the Performance of an AstroPhi Code for Massively Parallel Supercomputers Using Roofline Analysis

Boris Glinsky, Igor Kulikov, Igor Chernykh✉

Institute of Computational Mathematics and Mathematical Geophysics SB RAS, Lavrentjeva ave. 6, 630090 Novosibirsk, Russia
gbm@sscc.ru, kulikov@ssd.sscc.ru, chernykh@ssd.sscc.ru

Abstract. Astrophysics is the branch of astronomy that employs the principles of physics and chemistry "to ascertain the nature of the heavenly bodies, rather than their positions or motions in space". Numerical modeling plays a key role in modern astrophysics. It is the main tool for the research of nonlinear processes and provides communication between the theory and observational data. New massive parallel supercomputers provide an opportunity to simulate these kinds of problems in high details. Our astrophysics code AstroPhi was written for new massive parallel supercomputers based Intel Xeon Phi architecture. The original numerical method based on the combination of the Godunov method, operator splitting approach and piecewise-parabolic method on local stencil was used for numerical solution of the hyperbolic equations. The piecewise-parabolic method on local stencil provides the high-precision order. After the transition of AstroPhi to KNL architecture, we obtained abnormally low performance of solver on KNL cores. In this paper, we will show the roofline analysis using Intel Advisor application and the results of the AstroPhi optimizations.

Keywords: Massively Parallel Supercomputers · Astrophysics · Roofline Analysis.

1 Introduction

Numerical simulation in astrophysics allows to research many important problems such as the collision and evolution of galaxies, chemical evolution of stars, identification of dark matter and more. Modern supercomputers have given us the possibility of detailed astrophysics modeling that considers different physical effects such as magnetohydrodynamics, chemical kinetics, cooling/heating, and more. One of the most interesting developments in supercomputer technology at this moment is massively parallel supercomputers. The main concept of this technology is based on the possibility of massive usage of computational cores of CPUs or GPUs. Recently the scientific community is widely discussing the transition to exascale supercomputers. The main global challenge is the development of algorithms that can consider the massive ex-

ascale level parallelism. One of the problems is the difficulty of debugging and optimization of massively-parallel codes. The difficulties of debugging of parallel code connect with the problem of a greater propensity for race conditions, asynchronous events, and the general difficulty of trying to understand N processes simultaneously executing. Modern parallel debugging tools such as Eclipse Parallel Tools Platform [1], Intel Debugger [2], Nvidia nsight [3] helps in this problem. The difficulties of massively-parallel code optimizations are based on array dependence analysis, pointer alias analysis, loop transformations, adaptive profile-directed optimizations, and dynamic compilation [4]. Last years the roofline model [5] became a very popular tool for application performance analysis and optimization. The Roofline model is an intuitive visual performance model used to provide performance estimates of a given compute kernel or application running on multicore, many-core, or accelerator processor architectures, by showing inherent hardware limitations, and potential benefit and priority of optimizations. By combining locality, bandwidth, and different parallelization paradigms into a single performance figure, the model can be an effective alternative to assess the quality of attained performance instead of using simple percent-of-peak estimates, as it provides insights on both the implementation and inherent performance limitations [6]. The main idea of roofline model based on visualizing of application performance as a function of arithmetic intensity, where the application performance is the number of floating point operations per second (FLOPS) and the arithmetic intensity is the ratio of application performance to the memory traffic created by the application. Modern roofline analysis tools such as Intel Advisor [7] shows this data for each loop in the application and visualize the machine peak performance and machine peak memory bandwidth for target CPU architecture. Fig. 1 showing the typical roofline chart [8]. We can see in this figure that some of the application’s loops using cache because the arithmetic intensity is higher than DRAM peak bandwidth. Fig. 2 helps to identify which kind of your application is: memory-bound, memory/compute-bound or compute-bound application. After classification of application, this chart helps to build the strategy for optimization. Vectorization of application will increase the performance of the compute-bound code. L1/L2 cache optimizations of code will increase the performance of the memory-bound code.

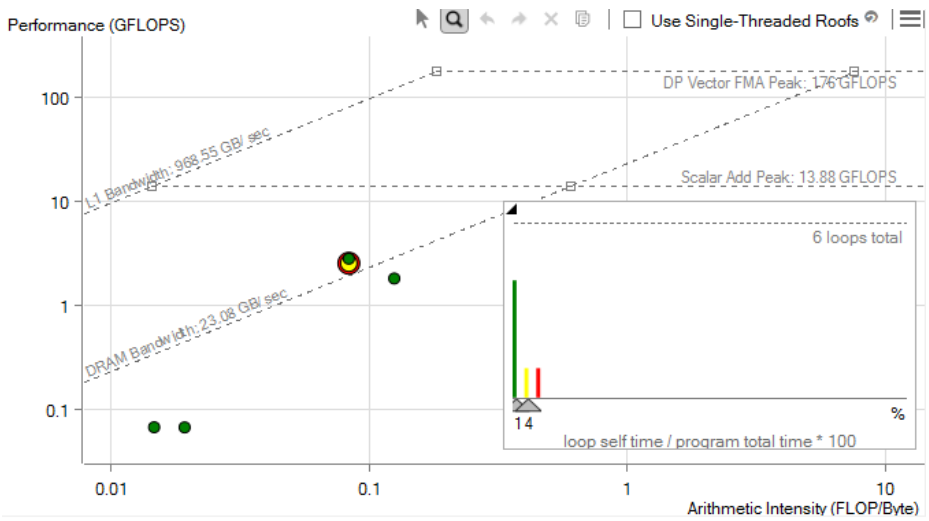

Fig. 1. Typical roofline chart with memory bandwidth and peak performance data of the target architecture [8].

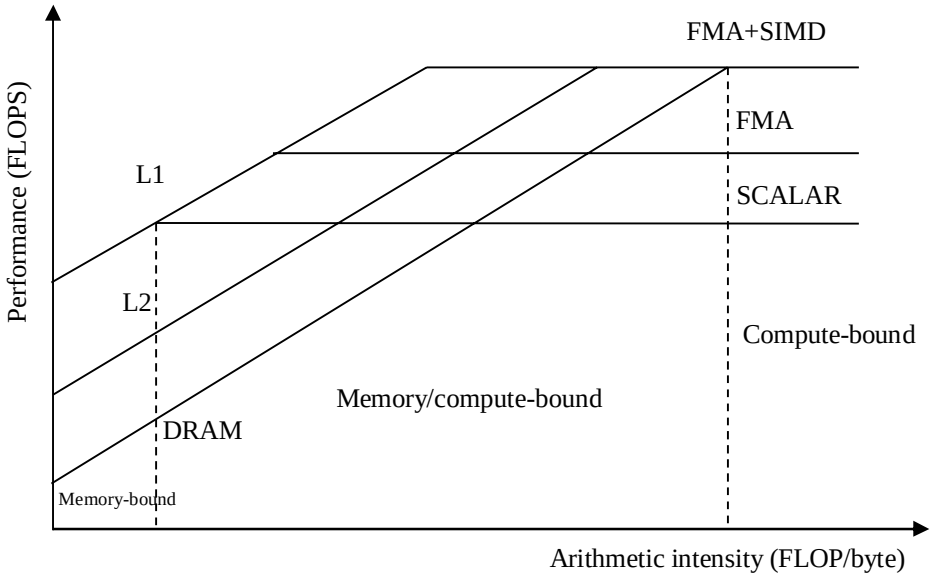

Fig. 2. Classification of application with roofline analysis.

2 Mathematical Model and Numerical Method

In our work, we use a multicomponent hydrodynamic model of galaxies considering the chemodynamics of molecular hydrogen and cooling in the following form:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \left(\rho \vec{u} \right) = 0 ,$$

$$\frac{\partial \rho_{H_2}}{\partial t} + \nabla \cdot \left(\rho_{H_2} \vec{u} \right) = S\left(\rho, \rho_H, \rho_{H_2} \right) ,$$

$$\frac{\partial \rho_H}{\partial t} + \nabla \cdot \left(\rho_H \vec{u} \right) = -S\left(\rho, \rho_H, \rho_{H_2} \right) ,$$

$$\frac{\partial \rho \vec{u}}{\partial t} + \nabla \cdot \left(\rho \vec{u} \vec{u} \right) = -\nabla p - \rho \nabla \Phi ,$$

$$\frac{\partial \varepsilon}{\partial t} + \nabla \cdot \left(\varepsilon \vec{u} \right) = -\left(\gamma - 1 \right) \varepsilon \nabla \cdot \left(\vec{u} \right) - Q ,$$

$$\frac{\partial E}{\partial t} + \nabla \cdot \left(E \vec{u} \right) = -\nabla \cdot \left(p \vec{u} \right) - \left(\rho \nabla \Phi, \vec{u} \right) - Q ,$$

$$\Delta \Phi = 4 \pi G \rho ,$$

$$E = \varepsilon + \frac{\rho \vec{u}}{2} ,$$

$$p = \left(\gamma - 1 \right) \varepsilon ,$$

where ρ is density, ρ_H is atomic hydrogen density, ρ_{H_2} is molecular hydrogen density, $\vec{u}$ is the velocity vector, ε is internal energy, p is pressure, E is total energy, γ is the ratio of specific heats, Φ is gravity, G is the gravitational constant, S is the formation rate of molecular hydrogen, and Q is a cooling function. A detailed description of this model can be found in [9].

The formation of molecular hydrogen is described by an ordinary differential equation [10]:

$$\frac{dn_{H_2}}{dt} = R_{gr}\left(T \right) n_H \left(n_H + 2 n_{H_2} \right) - \left(\xi_H + \xi_{diss} \right) n_{H_2} ,$$

where n_H is the concentration of atomic hydrogen, n_{H_2} is the concentration of molecular hydrogen, and T is temperature. Detailed descriptions of the H_2 formation rate R_{gr} and the photodissociation ξ_H, ξ_{diss} of molecular hydrogen, can be found in [11,12]. Chemical kinetics was don with using of CHEMPAK tool [13,14]

The original numerical method based on the combination of the Godunov method, operator splitting approach and piecewise-parabolic method on local stencil was used for numerical solution of the hyperbolic equations [15]. The piecewise-parabolic

method on local stencil provides the high-precision order. The equation system is solved in two stages: at the Eulerian stage, the equations are solved without advective terms and at the Lagrangian stage, the advection transport is being performed. At the Eulerian stage, the hydrodynamic equations for both components are written in the non-conservative form and the advection terms are excluded. As the result, such a system has an analytical solution on the two-cell interface. This analytical solution is used to evaluate the flux through the two-cell interface. In order to improve the precision order, the piecewise-parabolic method on the local stencil (PPML) is used. The method is the construction of local parabolas inside the cells for each hydrodynamic quantity. The main difference of the PPML from the classical PPM method is the use of the local stencil for computation. It facilitates the parallel implementation by using only one layer for subdomain overlapping. It simplifies the implementation of the boundary conditions and decreases the number of communications thus improving the scalability. The detailed description of this method can be found in [16]. The same approach is used for the Lagrangian stage. Now the Poisson equation solution is based on Fast Fourier Transform method. This is because the Poisson equation solution takes several percents of the total computation time. After the Poisson equation solution, the hydrodynamic equation system solution is corrected. It should be noticed here that the system is over defined. The correction is performed by means of the original procedure for the full energy conservation and the guaranteed entropy nondecrease. The procedure includes the renormalization of the velocity vector length, its direction remaining the same (on boundary gas-vacuum) and the entropy (or internal energy) and dispersion velocity tensor correction. Such a modification of the method keeps the detailed energy balance and guaranteed non-decrease of entropy.

3 Roofline Analysis

Roofline analysis with using of Intel Advisor consists of 3 steps: survey collection, trip count collection, visualization and/or extraction of the collected data into a report. Before analysis, the application should be compiled in debug mode.

1. Survey collection by command line with advisor:

```
mpirun -n <number of KNL nodes> advixe-cl -collect survey
–-trace-mpi -- ./<app_name>
```

2. Trip count collection by command line with advisor:

```
mpirun -n <number of KNL nodes> advixe-cl -collect
tripcounts -flops-and-masks –-trace-mpi -- ./<app_name>
```

3. Extraction of the data in a report:

```
advixe-cl –report survey –show-all-columns –-format=text
-– report-output report.txt
```

In our research, we used RSC Tornado-F [17] experimental node with Intel Xeon Phi 7250 (16GB MCDRAM) processor. We used all 68 cores for the tests. Fig. 3 shows roofline chart for AstroPhi application before optimizations. We can see that main loop of the application has very low arithmetic intensity (less than 0.1 FLOP/byte) and very low performance (less than 1 GFLOPS).

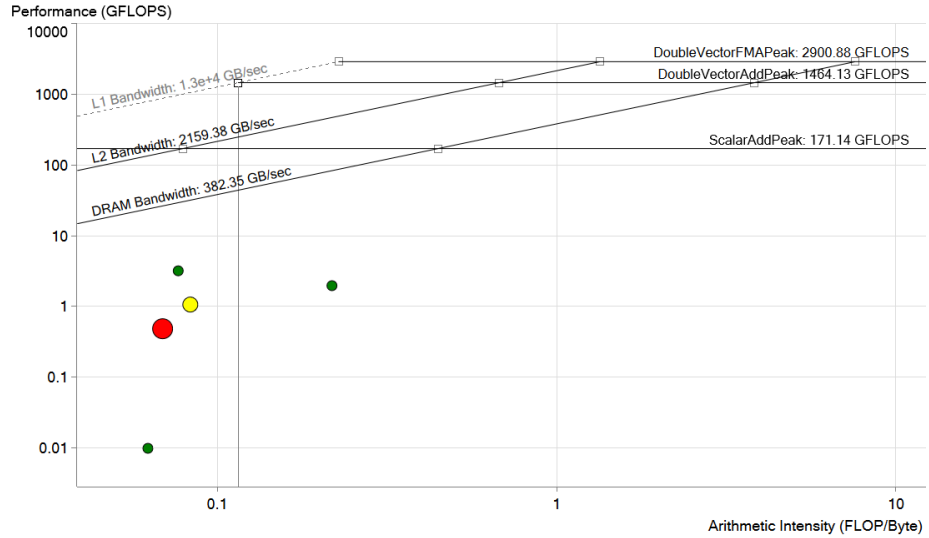

Fig. 3. Roofline chart for AstroPhi code before optimizations. The red dot is the main loop of the application.

Intel Advisor proposed some optimizations for gaining performance. The main of these optimizations are to remove the vector dependencies, to optimize memory access patterns, to move source loop iterations from peeled/ remainder loops to the loop body.

After the optimization roofline analysis was repeated on the same hardware with the same analysis steps. Fig. 4 shows roofline chart for AstroPhi application after optimizations. After all improvements in AstroPhi application, we achieved 190GFLOPS performance and 0.3 FLOP/byte arithmetic intensity with 100% mask utilization and 573 GB/s memory bandwidth.

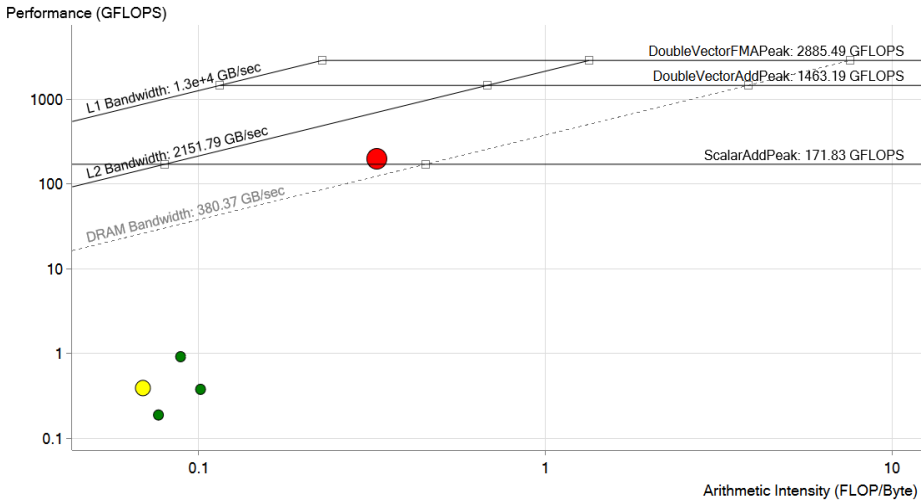

Fig. 4. Roofline chart for AstroPhi code after optimizations. The red dot is the main loop of the application.

4 Conclusion

Numerical modeling plays a key role in modern astrophysics. It is the main tool for the research of nonlinear processes and provides communication between the theory and observational data. Numerical simulation in astrophysics allows detailed investigation of the collision and evolution of galaxies. Author's astrophysics code was written for new massive parallel supercomputers based Intel Xeon Phi architecture. The original numerical method based on the combination of the Godunov method, operator splitting approach and piecewise-parabolic method on local stencil was used for numerical solution of the hyperbolic equations. The piecewise-parabolic method on local stencil provides the high-precision order. After the transition of AstroPhi to Intel Xeon Phi KNL architecture, we obtained abnormally low usage of KNL's cores. The roofline analysis of our code with using of Intel Advisor showed that main loop has very low arithmetic intensity (less than 0.1 FLOP/byte) and very low performance (less than 1 GFLOPS). Due to recommendations of Intel Advisor, vector dependencies were removed, memory operations were optimized, and arrays sizes were adapted for KNL architecture. After these improvements, we achieved 190GFLOPS performance and 0.3 FLOP/byte arithmetic intensity with 100% mask utilization and 573 GB/s memory bandwidth. This arithmetic intensity is standard for this kind of algorithms.

Acknowledgments. This work was partially supported by the Grant of the President of Russian Federation for the support of young scientists number MK – 1445.2017.9, RFBR grant 15-01-00508, 16-29-15120 and 16-07-00434.

References

1. Eclipse Parallel Tools Platform. http://www.eclipse.org/ptp/
2. Intel Parallel Studio. https://software.intel.com/en-us/intel-parallel-studio-xe
3. Nvidia Nsight. http://www.nvidia.com/object/nsight.html
4. Sarkar V.: Challenges in Code Optimization of Parallel Programs. CC 2009. Lecture Notes in Computer Science, 5501 (2009)
5. Ofenbeck, G.; Steinmann, R.; Caparros, V.; Spampinato, D. G.; Püschel, M.: Applying the roofline model. 2014 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 76–85 (2014)
6. Da Qi Ren: Algorithm level power efficiency optimization for CPU–GPU processing element in data intensive SIMD/SPMD computing. J. Parallel Distrib. Comput., 71, 245–253 (2011)
7. Intel Advisor. https://software.intel.com/en-us/intel-advisor-xe
8. Understanding the roofline chart. https://software.intel.com/en-us/intel-advisor-2017-user-guide-linux-understanding-the-roofline-chart?language=fr
9. Vshivkov V.A., Lazareva G.G., Snytnikov A.V., Kulikov I.M., Tutukov A.V.: Hydrodynamical code for numerical simulation of the gas components of colliding galaxies. Astrophysical Journal. Supplement Series, 194(47), 1-12 (2011)
10. Bergin E.A., Hartmann L.W., Raymond J.C., Ballesteros-Paredes J.: Molecular cloud formation behind shock waves. Astrophys. J.,612, 921-939 (2004)
11. Khoperskov S.A., Vasiliev E.O., Sobolev A.M., Khoperskov A.V.: The simulation of molecular clouds formation in the Milky Way. Monthly Notices of the Royal Astronomical Society, 428 (3), 2311-2320 (2013)
12. Glover S., Mac Low M.: Simulating the formation of molecular clouds. I. Slow formation by gravitational collapse from static initial conditions. Astrophysical Journal. Supplement Series, 169, 239-268 (2006)
13. Chernykh, I., Stoyanovskaya, O., Zasypkina, O.: ChemPAK software package as an environment for kinetics scheme evaluation. Chemical Product and Process Modeling, 4(4) (2009)
14. Snytnikov, V.N., Mischenko, T.I., Snytnikov, V., Chernykh, I.G.: Physicochemical processes in a flow reactor using laser radiation energy for heating reactants. Chemical Engineering Research and Design, 90(11), 1918-1922 (2012)
15. Godunov S.K., Kulikov I.M.: Computation of discontinuous solutions of fluid dynamics equations with entropy nondecrease guarantee. Computational Mathematics and Mathematical Physics, 54, 1012–1024 (2014)
16. Kulikov I., Vorobyov E.: Using the PPML approach for constructing a low-dissipation, operator-splitting scheme for numerical simulations of hydrodynamic flows. J. Comput. Phys., 317, 316—346 (2016)
17. RSC Tornado. http://www.rscgroup.ru/en/our-technologies/267-rsc-tornado-cluster-architecture