Performance Modeling of Stencil Computation on SW26010 Processors

Yao Liu[1], Li Liu[1], Mengtao Hu[1], Wei Wang[1], Wei Xue[2], and Qingting Zhu[1](✉)

[1] School of Data Science and Engineering, East China Normal University, Shanghai, China
{liuyao,qtzhu}@cc.ecnu.edu.cn, bran96@163.com, izrail@163.com, wwang@dase.ecnu.edu.cn

[2] Department of Computer Science and Technology, Tsinghua University, Beijing, China
xuewei@tsinghua.edu.cn

Abstract. Stencil computation is a basic part in a large variety of scientific computing programs, especially for those containing partial differential equations. Due to the limited memory bandwidth, it is a challenge to improve the parallel efficiency of stencil computation on modern supercomputers. Performance modeling is the most common method of performance analysis. In this paper, we propose the generic performance model based on Sunway TaihuLight which is powered by SW26010 heterogeneous many-core processors. The generic model indicates the interaction between the programs and the computing platform from the architecture perspective, and points out the performance bottlenecks of the programs from the optimization perspective. Furthermore, we propose the specific performance model of stencil computation on SW26010 processors, and optimize the performance of stencil computation under the guidance of the model. The experimental results show that the performance models proposed in this paper are effective—the average error ratio of the predicted performance is less than 7%. Guided by the specific model, the optimized stencil computation achieves better performance than the unoptimized many-core version by 154.71% on 4096 cores.

Keywords: Stencil computation · Performance modeling · Sunway TaihuLight · Heterogeneous many-core processors

1 Introduction

Large-scale scientific computing programs such as materials science [24], atmospheric modeling [7], seismic prediction [4], molecular dynamics [9], play important roles in the national core science and technology fields. Stencil computation is an essential part of many scientific computing programs, especially in the programs containing a large number of partial differential equations [14]. Due to the particularity and importance of stencil computation, how to improve its computing efficiency has been widely concerned.

M. Qiu (Ed.): ICA3PP 2020, LNCS 12452, pp. 386–400, 2020.
https://doi.org/10.1007/978-3-030-60245-1_27

Considering the complexity and high time consumption of stencil computation, high-performance computing platforms are used to accelerate computing. However, with the increase of computing resources, the parallel efficiency of the programs is often unsatisfactory. One of the key reasons is that the computing capability of modern microprocessors is not fully utilized because of their complicated architecture. Therefore, improving the match between the platform and the program, analyzing and tuning the program design, and exploring the potential concurrency have become the main methods to improve the performance of the program.

Sunway TaihuLight supercomputer reached the peak performance of more than 100 PFLOPS in 2016, which provides a favorable guarantee for large-scale scientific computing [11]. Nonhydrostatic atmospheric dynamics simulation on the Sunway TaihuLight system won the Gordon Bell Award [21]. However, Sunway TaihuLight has unique architecture and programming features, it's a big challenge to make full use of its high computing capability. Generally speaking, it takes a lot of time and effort to transplant scientific computing programs to Sunway TaihuLight. The factors that affect the performance of a program include the program's computing scale, computing intensity, and programming habits. Analysis of the program performance could point out the performance bottlenecks of the programs, improve the match between the platform and the program, and achieve the purpose of performance optimization.

Performance modeling is one of the most commonly used methods for performance analysis [6]. It abstracts the various behaviors and running characteristics of the program through mathematical formulas, and can effectively identify the bottlenecks. The performance model can also be used to predict the scalability of the performance, providing the guidance for performance optimization. There have been some research on performance modeling on the GPU and Intel platforms [5,13,17,22]. How to design the performance model on Sunway TaihuLight has gradually become a research hotspot. Ao et al. [1] proposed atmospheric model performance modeling, which mainly focused on the optimization methods and the scalability of stencil computation. Zhang et al. [23] addressed 3D stencil performance modeling on GPU, which gave the best data blocking scheme on multiple levels of storage. However, these studies rarely involve quantitative modeling, and the model parameters are not rich enough to fully indicate the features of the algorithms and architecture.

In this paper, based on the architecture features of SW26010, we propose the generic performance model of scientific computing program on Sunway TaihuLight, which provides guidance for optimizing the performance of scientific computing programs. Using this generic model, we propose the performance model of stencil computation on SW26010 processors, which considers the features of architecture and program. The model quantitatively analyzes the characteristics of the program and the running status of each program part, which provides a favorable guidance for the performance optimization of the stencil computation on Sunway TaihuLight. Additionally, the model can predict the performance of the program under different blocking and optimizing schemes. In summary, this paper makes the following contributions.

- Facing the challenges of heterogeneous architecture and hybrid programing models, we propose the generic performance model on Sunway TaihuLight which provides guidance for modeling and optimizing scientific computing programs.
- We propose the stencil computation performance model on SW26010 processors. The model quantitatively analyzes the performance under different blocking schemes. The average error ratio of the predicted performance is less than 7%.
- Under the optimization guidance provided by the performance model, the stencil computation could be optimized significantly on Sunway TaihuLight. The optimized stencil computation achieves better performance than the unoptimized many-core version by 154.71% on 4096 cores.

The paper is organized as follows. Section 2 presents related work about performance modeling and stencil computation. Section 3 illustrates the details of performance modeling on SW26010 processors. Section 4 presents the experimental results and analysis. Finally, Sect. 5 concludes this paper and points out the future work.

2 Related Work

2.1 Performance Modeling

Scientific computing programs include two parts: the computation part and the communication part. For the computing part, Samuel Williams et al. [19] proposed the Roofline Model, which is a basic performance model. It is simple and easy to measure, but the performance events are coarse-grained. Barnes et al. [2] use formula to fit model. They use performance tools to directly measure the performance of parallel programs, and perform linear or nonlinear fitting. Martin et al. [3] proposed PerfExpert, to analyze the performance bottlenecks of program comprehensively.

For the communication part, Torsten et al. [12] established an analytical communication model for the communication part of the MILC program. They divided the communication behavior into point-to-point communication and collective communication. The analytical communication model is complete, but the cost is high, and the process of modeling requires domain expert knowledge. Other communication models such as PRAM, BSP, Hockney model and LogP family are classic communication models [8].

2.2 Stencil Computation

Stencil computation is a kind of computing kernels that updates the value in a certain way by steps. For example, solving the heat conduction equation in a uniform anisotropic heat-conducting medium, we usually use a regular grid to represent the temperature distributed in three-dimensional space, and the temperature of each grid point will change by steps. Then the temperature of a point at the current moment is computing from the temperature of several points around the point at the previous moment. If it depends on the temperature of 7 or 27 nearby points, it is considered as a 7-point or 27-point stencil computation problem, as shown in Fig. 1.

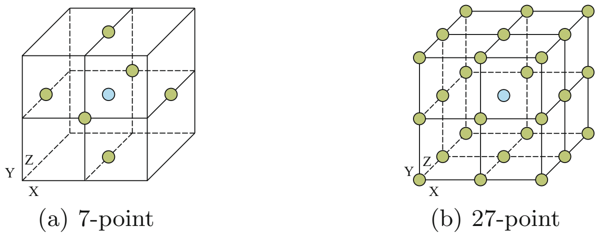


Fig. 1. Visualization of the 3D stencil.

2.3 Blocking Schemes

Due to the limitation of on-chip storage, for the GPU platform, the input data are divided into multiple small blocks, which are allocated to threads in order to optimize the stencil computation. The current work shows that there are three typical spatial blocking schemes, namely: 2D, 2.5D, and 3D blocking, as shown in Fig. 2. Vizitiu et al. [18] addressed that the effect of 3D blocking is better than the other two methods. However, the blocking schemes are flexible and should be considered in fact. Performance model proposed in this paper combines above blocking schemes, and analyzes the performance of stencil computation using different blocking schemes on Sunway TaihuLight.

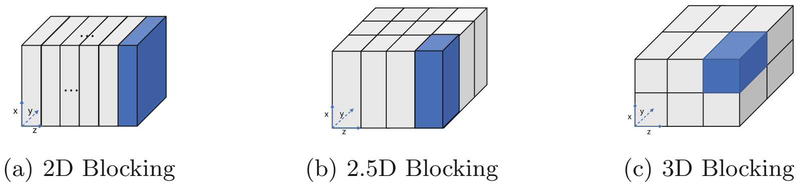


Fig. 2. Blocking schemes.

3 Performance Modeling on SW26010 Processors

3.1 Overview of Sunway TaihuLight

Sunway TaihuLight system, a supercomputer independently developed by China, is the first supercomputer with a peak performance of over 125 PFlops in the world [16]. It consists of 40960 SW26010 heterogeneous many-core processors. The general architecture of the processor is shown in Fig. 3. Each processor contains 4 core groups (CGs) connected via the network on chip (NoC). Each CG includes one management processing element (MPE) and one cluster of 8×8 computing processing elements (CPEs). Sunway TaihuLight supports three major programming models, namely: MPI, MPI+OpenACC, MPI+Athread [1]. Hybrid programing model—MPI+Athread is most used by researchers. Athread is a light-weight effective thread library that is used to exploit the parallelism of CPEs. The heterogeneous architecture of the processor and the hybrid programing model increase the difficulty of performance modeling, that is, how to integrate the architecture and the hybrid programming model in the performance model.

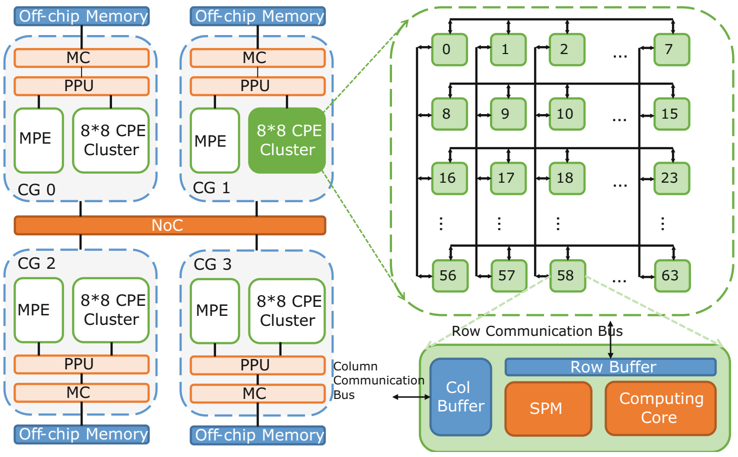


Fig. 3. Architecture of SW26010.

3.2 Generic Performance Model on Sunway TaihuLight

Most scientific computing programs use a master-slave accelerated parallel method on Sunway TaihuLight. Computing tasks are assigned to each core group. MPE (one process per MPE) is responsible for data communication between core groups, a small part of serial computation, I/O, etc., and CPEs (one thread per CPE) are responsible for computing.

We conclude from Sect. 2 that most scientific computing programs consist of two basic phases, computation phase and communication phase [8]. The computation phase consists of computations and memory accesses. The total time the program runs can be indicated as Eq. 1, where T_{comp} means the computation time, T_{mem} for the memory accesses time, T_{comm} for the communication time and T_{others} for the rest time (almost negligible) such as program initialization. Actually, the computation phase and the communication phase could be hidden from each other. Furthermore, in the computation phase, the computations and memory accesses could be hidden from each other. We use RF, RC to represent the hidden ratio, respectively.

$$T_{model} = (T_{comp} + RF \cdot T_{mem}) + RC \cdot T_{comm} + T_{others} \tag{1}$$

Computation Phase Modeling. According to the heterogeneous architecture, we extend the computation phase model. The model is indicated as Eq. 2. Each model item represents the behavior of MPE and CPEs in the computation phase.

$$\begin{aligned} T_{comp} + RF \cdot T_{mem} = T_{MPE\_comp} + RF_{MPE} \cdot T_{mem\_MPE} \\ + T_{CPE\_comp} + RF_{CPE} \cdot T_{mem\_CPE} \end{aligned} \tag{2}$$

Since the floating-point computation performance of CPEs is about 32× of MPE in one CG [11], all the computing tasks are assigned to CPEs in this section. We focus on performance modeling of CPEs computation, the performance modeling of MPE computation is the same as CPEs. The CPEs computation model is defined as Eq. 3, where $Dsize$ means the amount of computed data, P for the number of MPE, t_{comp} for minimum computation unit overhead.

$$T_{comp\_CPE} = \frac{Dsize}{P} \cdot t_{comp} \tag{3}$$

We can use timing functions to measure t_{comp} [8]. According to different types of scientific computing programs, we could choose a loop or a function as a minimum computation unit.

The speed of CPEs accessing its local device memory (LDM) is very fast, and the access delay is only 4 cycles [15]. However, due to the limited size of LDM (64 KB), CPE is required to continuously copy data from the main memory to its LDM, compute and copy the results back to the main memory finally. Although all CPEs independently complete their assigned computing tasks, the CPEs have to compete for the limited bandwidth between the main memory and LDMs. Due to the memory bound, memory access optimization is the effective method of performance optimization on SW26010 processors. Since the computing tasks are assigned in CPEs, memory access time of CPEs T_{mem_CPE} becomes an key component of the performance model. The SW26010 processor provides two ways for CPEs to access the main memory. The CPEs can directly access the main memory through the gld/gst method discretely, or can access the main memory in batches through the DMA method, and there are also mixed use methods.

T_{mem_CPE} can be indicated as Eq. 4, where $T_{gld/gst}$ means time consumed by gld/gst, T_{DMA} for time consumed by DMA.

$$T_{mem\_CPE} = T_{gld/gst} + T_{DMA} \tag{4}$$

The delay of one gld/gst is about 220 cycles [10], while the delay of one DMA is only 25 cycles [15]. We should avoid using gld/gst as much as possible in practical applications. In general, CPEs use DMA to access the main memory. When modeling performance of memory access, we mainly focus on DMA performance modeling. The modeling method of gld/gst is the same as DMA.

For one CPE, T_{DMA} can be defined as Eq. 5, where $count_{DMA}$ represents the number of DMA requests, t_{DMA} means time consumed by each DMA.

$$T_{DMA} = count_{DMA} \cdot t_{DMA} \tag{5}$$

t_{DMA} is closely related to the data size required for each DMA—$Dsize_{DMA}$ because $Dsize_{DMA}$ determines the DMA bandwidth. Therefore, we introduce the effective DMA bandwidth $effBW_{DMA}$ to represent the actual bandwidth used. t_{DMA} can be indicated as Eq. 6, where $count_{CPE}$ means the number of CPEs participating in the computation, which is another main factor that affects DMA bandwidth [10]. $effBW_{DMA}$ can be found in [15,16,20].

$$t_{DMA} = \frac{count_{CPE} \cdot Dsize_{DMA}}{effBW_{DMA}} \tag{6}$$

In the computation phase, there are cases where computations and memory accesses are hidden from each other, and how to hide them from each other as much as possible is the key to improving performance of programs. RF is related to the programmer's programming ability, the computation and memory characteristics of different problems and the system's own optimization. $RF = 0$ proves that computations and memory accesses have reached perfect hiding, and $RF = 1$ means that computations and memory accesses are not hidden from each other.

In summary, the computation phase performance model M_{comp} can be represented as Eq. 7.

$$M_{comp} = \frac{D_{size}}{P} \cdot t_{comp} + RF_{CPE} \cdot (count_{DMA} \cdot \frac{count_{CPE} \cdot Dsize_{DMA}}{effBW_{DMA}}) \tag{7}$$

Communication Phase Modeling. Programs such as partial differential equation solving usually assign grid computing tasks to each CG. When the updating of grid points require adjacent data, it needs to communicate with the surrounding grid points, so there is a need for data communication between MPEs. The communication phase can usually be divided into collective communication and point-to-point communication [8]. Communication time can be defined as Eq. 8, where T_{col} means time spent on collective communication, t_{p2p} for time consumed by point-to-point communication.

$$T_{comm} = T_{col} + T_{p2p} \tag{8}$$

We take collective communication as an example, assuming that the global reduction uses the k-tree reduction method. P means the number of MPEs, n means the number of the communication starts, $Csize$ is total communication data, t_{comm} means minimum communication unit overhead, T_{col} can be represented as Eq. 9.

$$T_{col} = n \cdot log_k(P) + \frac{Csize}{P} \cdot t_{comm} \tag{9}$$

In practical applications, we often use point-to-point communication, therefore communication model can be defined as Eq. 10, where t_{start} means communication start time. We can also measure t_{start} and t_{comm} using the timing function.

$$T_{comm} = n \cdot t_{start} + \frac{Csize}{P} \cdot t_{comm} \tag{10}$$

In general, we purpose the generic performance model on Sunway TaihuLight. The model provides a useful guide for the performance modeling of programs on Sunway TaihuLight.

3.3 Performance Modeling of Stencil Computation

When we have the generic performance model, we only need to extend the model according to the characteristics of the target programs to achieve rapid modeling.

Stencil Computation Description. Stencil computation includes computation phase and communication phase introduced in Sect. 2.2. When stencil computation runs on SW26010 processors by steps, the phases in this process include updating halo, fetching data to LDM, computing stencil, and writing data back to main memory.

Computation Phase Modeling. From the description, we found that stencil computation has a large number of memory access operations. Since LDM size is only 64 KB, it is impossible to fetch data to CPEs at once, so the memory accesses become more frequent. We use the 3D 7-point stencil computation as an example. When updating a grid point, it needs to use its top, bottom, front, back, left and right grid points as shown in Fig. 1. In this process, there is a lot of data redundancy. A lot of data are read repeatedly, so how to improve data utilization becomes the focus of the computation phase. Blocking is an effective method to improve the efficiency of memory access. Redundant memory accessing can be defined as Eq. 11, where $block_y$ means length of y-axis after blocking, $block_z$ means length of z-axis after blocking. The larger r is, the less redundant memory access.

$$r = \frac{block_y \cdot block_z}{count_{DMA}} \tag{11}$$

Since LDM size is fixed, $Dsize_{DMA}$ is inversely related to $count_{DMA}$. If the halo size is 1, $count_{DMA}$ can be defined as Eq. 12. If we take a 2.5D or 3D blocking scheme, r becomes larger, but $Dsize_{DMA}$ becomes smaller. Therefore, balancing these two indicators is the key to improving performance during the computation phase.

$$count_{DMA} = (block_y + 2) \cdot (block_z + 2) \tag{12}$$

Communication Phase Modeling. $Csize$ is a key factor affecting T_{comm}. From the description we found that the stencil computation requires swapping the halo data of the grid. For 3D stencil computation, the stencil computation intensity (5-point,7-point,27-point, etc.) determines $Csize$. $Csize$ can be defined as Eq. 13, where μ, ν are the flags that represent intensity, N for the data gird size, H for the halo size. Appropriate blocking scheme can effectively reduce $Csize$. We integrate the blocking scheme into the communication model. $Csize$ can be calculated by Eq. 14, where P_x, P_y, P_z mean the number of divisions on the x, y, z axis, and β_1, β_2, β_3 are the flags of the blocking schemes, as shown in Table 1.

Table 1. Blocking schemes

β_1	β_2	β_3	Blocking schemes
1	0	0	2D Blocking, split z-axis
1	1	0	2.5D Blocking, split y, z-axis
1	1	1	3D Blocking split x, y, z-axis

$$Csize_{3D} = 6 \cdot H \cdot N^2 + \mu \cdot 8 \cdot H^3 + \nu \cdot 12 \cdot N \cdot H^2 \tag{13}$$

$$\begin{aligned} Csize_{3D} = {} & 2 \cdot H \cdot (\beta_1 \cdot \frac{N^2}{P_x \cdot P_y} + \beta_2 \cdot \frac{N^2}{P_x \cdot P_z} + \beta_3 \cdot \frac{N^2}{P_y \cdot P_z}) \\ & + \mu \cdot 8 \cdot H^3 + \nu \cdot 4 \cdot (\beta_1 \cdot \frac{N}{P_x} + \beta_2 \cdot \frac{N}{P_y} + \beta_3 \cdot \frac{N}{P_z}) \cdot H^2 \end{aligned} \tag{14}$$

For example, $Csize$ of 3D 7-point stencil computation with halo size of 1 can calculated by Eq. 15. Different process blocking schemes will make the sending data discontinuous. Therefore, we need to pack and unpack the data in the communication phase, and the amount of discontinuous sending data determines the time consumed by the data packing and unpacking. In summary, the communication model of stencil computation can be defined as Eq. 16, where $t_{packunpack}$ means time spent on data packing and unpacking.

$$Csize_{3D7p} = 2 \cdot (\beta_1 \cdot \frac{N^2}{P_x \cdot P_y} + \beta_2 \cdot \frac{N^2}{P_x \cdot P_z} + \beta_3 \cdot \frac{N^2}{P_y \cdot P_z}) \tag{15}$$

$$\begin{aligned} T_{comm} = {} & n \cdot t_{start} + t_{packunpack} \\ & + 2 \cdot (\beta_1 \cdot \frac{N^2}{P_x \cdot P_y} + \beta_2 \cdot \frac{N^2}{P_x \cdot P_z} + \beta_3 \cdot \frac{N^2}{P_y \cdot P_z}) \cdot t_{comm} \end{aligned} \tag{16}$$

4 Experiments

In this section, we evaluate the proposed performance model from the perspective of model accuracy and performance optimization.

4.1 Experimental Setup and Metrics

We use standard MPI+Athread programming model to implement the parallelization of stencil computation on Sunway TaihuLight. The data sets are named in the form of N-$STENCIL$-P-$STEPS$, where N means the grid size, $STENCIL$ for the computation intensity, P for the number of processes, and $STEPS$ for the iteration number of stencil. To match the grid size, different numbers of processes are used. The four data sets used in the experiments have their own features, such as different N and $STEPS$. The grid data is placed in order of x-y-z, where x is the innermost direction.

Error Ratio. The model error ratio is used to evaluate the accuracy of the model. The smaller the model error ratio, the higher the model accuracy. Let Res_e be the average model result obtained from multiple experiments, and Res_m be the model results predicted by the performance model proposed in this paper, model error ratio can be defined as

$$R_{err} = |1 - \frac{Res_m}{Res_e}| \times 100\% \tag{17}$$

Speedup. The experiments use *speedup* to evaluate the acceleration effect of model-guided optimization. Let T_{mpe} be the running time of multi-process stencil computation, and $T_{manycore}$ be the running time after model-guided optimization, then *speedup* can be defined as

$$speedup = \frac{T_{mpe}}{T_{manycore}} \tag{18}$$

4.2 Error Ratio of Models

The experiments evaluate R_{err} of the computation model, the communication model, the computation model based on the blocking schemes, and the communication model based on the blocking schemes, respectively.

For input items of the model, some of them are obtained through multiple measurements, for example, t_{comm}, t_{comp}, some are obtained by reviewing the paper [15,16,20], such as $effBW_{DMA}$, and the rest are obtained by calculation, such as $count_{DMA}$. We use the proposed performance model to predict the results, meanwhile the results of the experiment is obtained by instrumenting.

Evaluation of Computation Model. In this experiment, to utilize the DMA bandwidth, CPE take 650 double to LDM each time. Figure 4(a) shows the predicted results are basically consistent with the experimental results. With the increase of computation intensity, the proportion of T_{comp} in the total time increases. R_{err} is less than 5%, which proves the proposed computation model is effective.

Evaluation of Communication Model. The experiments use $MPI\_Sendrecv$ as the communication functions. In this experiment, we divide the grid along the z-axis, and the number of divisions is the number of processes. The results are shown in Fig. 4(b). $CSize$ is the key factor that determines pure communication time $T_{comm}(pure)$, and when $STEPS$ increases, T_{comm} also increases. The average R_{err} is satisfactory, only 6.08%.

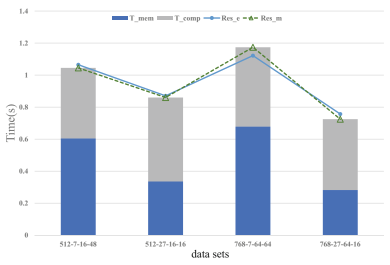
Evaluation of Computation Model Based on Blocking Schemes. We take 512-7-16-48 as an example for the experiment. Figure 4(c) shows the blocking schemes change T_{mem} obviously, because the schemes effect $count_{DMA}$ and $effBW_{DMA}$. The improvement of $effBW_{DMA}$ plays a key role in reducing T_{mem}, and $count_{DMA}$ is inversely related to $effBW_{DMA}$. The bigger and square yz-plane, the larger r. The R_{err} is about 5%, and the performance is obviously improved when $effBW_{DMA}$ and r are both large.

Evaluation of Communication Model Based on Blocking Schemes. We take 512-7-16-48 and 768-7-64-64 as examples for the experiment. Figure 4(d) shows 2.5D and 3D blocking schemes could effectively decrease $Csize$, and $T_{comm}(pure)$ decreases accordingly. However, the advantage brought by the decrease of $Csize$ is reduced due to the increase of $t_{packunpack}$. The average R_{err} is desirable, which is 5.09%. We found that in the case of several processes, the 2D blocking scheme along the z-axis is better because $t_{packunpack}$ can be omitted. In the case of a larger number of processes, the 2.5D blocking scheme is outstanding, especially the scheme with square yz-plane and more continuous dimensions. As for the 3D blocking scheme, it is unsatisfactory because of the massive data packing and unpacking.

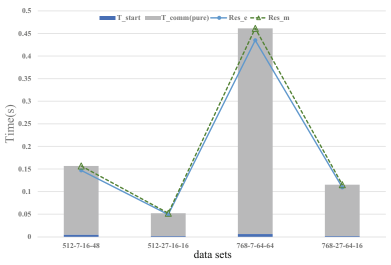
4.3 Model-Guided Performance Optimization

Process-Level Parallelism (PP). The computation tasks are assigned to each MPE, and each MPE is responsible for computation and communication.

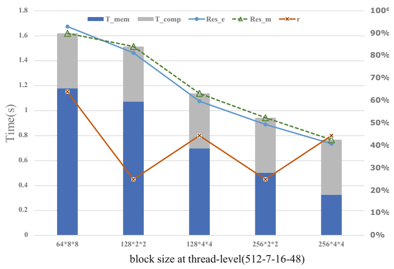
Multi-level Parallelism (MP). The computation tasks on each MPE are assigned to CPEs. CPEs are responsible for computation and each MPE is responsible for communication. Mutli-level parallelism is the most common parallel version of many-core on SW26010 processors.



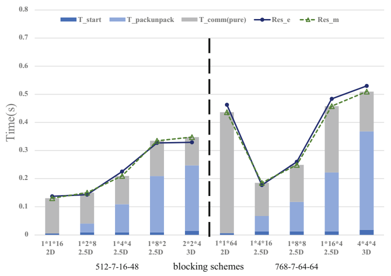
(a) CompModel



(b) CommModel

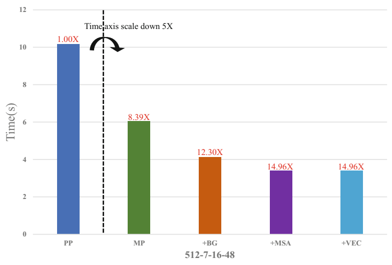


(c) CompModel based on blocking

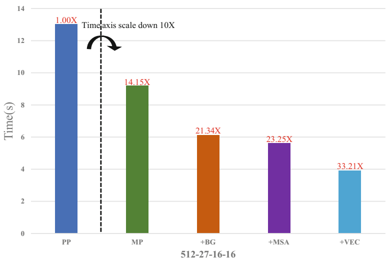


(d) CommModel based on blocking

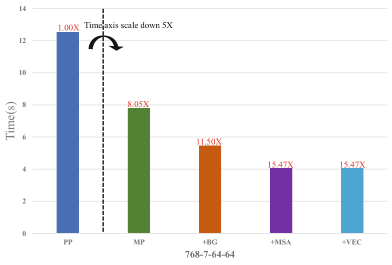
Fig. 4. Evaluation of the model.



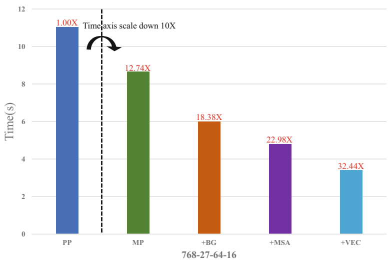
(a) Speedup of 512-7-16-48



(b) Speedup of 512-27-16-16



(c) Speedup of 768-7-64-64



(d) Speedup of 768-27-64-16

Fig. 5. Speedup by the model-guided optimization.

Blocking Guidance (BG). We use the proposed blocking-based model and experimental results to select the best blocking scheme. At the process level, the 2D blocking scheme is recommend for 16 processes, and the 2.5D blocking scheme is recommend for 64 processes. At the thread level, under the premise of ensuring the full utilization of LDM, the blocking scheme with larger $effBW_{DMA}$ and r is selected. From the results predicted by the model, we found that the memory access is time-consuming. Therefore, the i-th layer is being computed while the $i+2$ layer is being fetched under the blocking schemes, so that the computations and memory accesses could be hidden from each other.

Master-Slave Asynchronous (MSA). Under the guidance of the model, the computation phase and the communication phase could be hidden from each other. Considering the features of stencil computation that internal data do not participate in communication, we use CPEs to perform internal data computation while MPEs perform halo data communication.

Vectorization (VEC). When the computation intensity is high, we could use vectorization additionally to shorten the computing time.

The optimization results are shown in the Fig. 5. The maximum *speedup* of the experiments reaches 14.96×, 33.21× on 1024 cores and 15.47×, 32.44× on 4096 cores, respectively, which means model-guided optimization has achieved excellent results. Blocking guidance is significant for the performance improvement of the stencil computation on SW26010. Besides, vectorization is sensitive to the computation intensity, which notably reduces the cost of 27-point stencil as shown in Fig. 5(b) and Fig. 5(d). Guided by our model, the optimized stencil computation achieves better performance than the unoptimized many-core version by 78.24%, 134.74% on 1024 cores and 92.22%, 154.71% on 4096 cores, respectively.

5 Conclusion

In this paper, we propose the generic performance model of scientific computing program according to features of architecture and programming models on Sunway TaihuLight, which provides guidance for modeling and optimizing the performance of scientific computing programs. Especially, for the stencil computation, we propose the specific performance model according to the characteristics of SW26010 and the algorithm. The performance model quantitatively analyzes the cost of each program part, which provides a favorable guidance for the performance optimization of the stencil computation on Sunway Taihulight. The experimental results demonstrate that the proposed performance model has high accuracy and could effectively predict the performance of the program. The average error ratio predicted by the model is less than 7%. In addition, the performance optimization of the program guided by the model is also satisfactory. The optimized stencil computation achieves better performance than the unoptimized many-core version by 154.71% on 4096 cores.

In the future, we intend to conduct research on automatic performance modeling, and apply the performance model to larger scientific computing programs, which have higher computing scale and more computing kernels.

Acknowledgement. This work is supported by the Ministry of Education's University-Industry Collaborative Education Program (No. 201902146019)

References

1. Ao, Y., et al.: 26 PFLOPS stencil computations for atmospheric modeling on Sunway TaihuLight. In: 2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 535–544. IEEE (2017)
2. Barnes, B.J., Rountree, B., Lowenthal, D.K., Reeves, J., De Supinski, B., Schulz, M.: A regression-based approach to scalability prediction. In: Proceedings of the 22nd Annual International Conference on Supercomputing, pp. 368–377 (2008)
3. Burtscher, M., Kim, B.D., Diamond, J., McCalpin, J., Koesterke, L., Browne, J.: Perfexpert: an easy-to-use performance diagnosis tool for HPC applications. In: SC 2010: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–11. IEEE (2010)
4. Chen, B., et al.: Simulating the Wenchuan earthquake with accurate surface topography on Sunway TaihuLight. In: SC18: International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 517–528. IEEE (2018)
5. Chen, G., Wu, B., Li, D., Shen, X.: Porple: an extensible optimizer for portable data placement on GPU. In: 2014 47th Annual IEEE/ACM International Symposium on Microarchitecture, pp. 88–100. IEEE (2014)
6. Datta, K., Kamil, S., Williams, S., Oliker, L., Shalf, J., Yelick, K.: Optimization and performance modeling of stencil computations on modern microprocessors. SIAM Rev. **51**(1), 129–159 (2009)
7. Dennis, J.M., et al.: Cam-se: a scalable spectral element dynamical core for the community atmosphere model. Int. J. High Perform. Comput. Appl. **26**(1), 74–89 (2012)
8. Ding, N., Xu, S., Song, Z., Zhang, B., Li, J., Zheng, Z.: Using hardware counter-based performance model to diagnose scaling issues of HPC applications. Neural Comput. Appl. **31**(5), 1563–1575 (2019). https://doi.org/10.1007/s00521-018-3496-z
9. Dong, W., Li, K., Kang, L., Quan, Z., Li, K.: Implementing molecular dynamics simulation on the Sunway TaihuLight system with heterogeneous many-core processors. Concurr. Comput.: Pract. Exp. **30**(16), e4468 (2018)
10. Fu, H., et al.: Refactoring and optimizing the community atmosphere model (CAM) on the Sunway TaihuLight supercomputer. In: SC 2016: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 969–980. IEEE (2016)
11. Fu, H., et al.: The Sunway TaihuLight supercomputer: system and applications. Sci. China Inf. Sci. **59**(7), 072001 (2016). https://doi.org/10.1007/s11432-016-5588-7
12. Hoefler, T., Gropp, W., Kramer, W., Snir, M.: Performance modeling for systematic performance tuning. In: SC 2011: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–12. IEEE (2011)

13. Hong, S., Kim, H.: An analytical model for a GPU architecture with memory-level and thread-level parallelism awareness. In: Proceedings of the 36th Annual International Symposium on Computer Architecture, pp. 152–163 (2009)
14. Langtangen, H.P.: Computational Partial Differential Equations: Numerical Methods and Diffpack Programming, vol. 2. Springer, Berlin (1999). https://doi.org/10.1007/978-3-662-01170-6
15. Li, L., et al.: swCaffe: a parallel framework for accelerating deep learning applications on Sunway TaihuLight. In: 2018 IEEE International Conference on Cluster Computing (CLUSTER), pp. 413–422. IEEE (2018)
16. Liu, Y., Liao, Q., Sun, J., Hu, M., Liu, L., Zheng, L.: A heterogeneous parallel genetic algorithm based on sw26010 processors. In: 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), pp. 54–61. IEEE (2019)
17. Shirako, J., et al.: Analytical bounds for optimal tile size selection. In: O'Boyle, Michael (ed.) CC 2012. LNCS, vol. 7210, pp. 101–121. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28652-0_6
18. Vizitiu, A., Itu, L., Niţă, C., Suciu, C.: Optimized three-dimensional stencil computation on Fermi and Kepler GPUs. In: 2014 IEEE High Performance Extreme Computing Conference (HPEC), pp. 1–6. IEEE (2014)
19. Williams, S., Waterman, A., Patterson, D.: Roofline: an insightful visual performance model for multicore architectures. Commun. ACM **52**(4), 65–76 (2009)
20. Xu, Z., Lin, J., Matsuoka, S.: Benchmarking sw26010 many-core processor. In: 2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pp. 743–752. IEEE (2017)
21. Yang, C., et al.: 10m-core scalable fully-implicit solver for nonhydrostatic atmospheric dynamics. In: SC 2016: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 57–68. IEEE (2016)
22. You, Y., et al.: Accelerating the 3D elastic wave forward modeling on GPU and MIC. In: 2013 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum, pp. 1088–1096. IEEE (2013)
23. Zhang, G., Zhao, Y.: Modeling the performance of 2.5 d blocking of 3D stencil code on GPUs. In: IEEE High Performance Extreme Computing Conference, HPEC (2016)
24. Zhang, J., et al.: Extreme-scale phase field simulations of coarsening dynamics on the Sunway TaihuLight supercomputer. In: SC 2016: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 34–45. IEEE (2016)