CCF Transactions on High Performance Computing (2023) 5:322–333
https://doi.org/10.1007/s42514-023-00147-x

REGULAR PAPER

Adapting combined tiling to stencil optimizations on sunway processor

Biao Sun[1] · Mingzhen Li[1] · Hailong Yang[1] · Jun Xu[2] · Zhongzhi Luan[1] · Depei Qian[1]

Received: 5 March 2023 / Accepted: 18 April 2023 / Published online: 17 May 2023

Abstract
Stencil is one of the indispensable computation patterns in scientific applications, which is a long-standing optimization target in the field of high performance computing (HPC). The Sunway processor adopted in Sunway TaihuLight supercomputer has demonstrated its performance potential with unique heterogeneous many-core architecture. Although a large number of optimization methods have been proposed, the memory-bound nature of stencil computation and the limited bandwidth of Sunway processor make it challenging to adapt stencil computation efficiently on Sunway processor. To better use the computation capability of Sunway processor, we propose a combined tiling optimization of stencil computation tailored for the architectural features. In addition, we implement double buffering, vectorization, and register communication to further accelerate stencil computation on Sunway processor. We evaluate our method on six stencil benchmarks with different orders and shapes (thus different memory access patterns and computation intensities). The experimental results show that our implementation can achieve 1.97× speedup on average compared to the state-of-the-art stencil implementation on Sunway.

Keywords Stencil computation · Sunway processor · Performance optimization · Combined tiling

1 Introduction

Stencil is an important and indispensable building block of modern scientific applications. It is widely used in the fields of weather prediction (Powers et al. 2017), earthquake simulation (Fu et al. 2017), fluid dynamic (Dongarra et al. 2008) and etc. Therefore, performance optimization of stencil computation has been a long-standing research topic in the field of HPC (High Performance Computing) ever since. Although tremendous research efforts have been devoted, stencil computation is challenging for performance optimization due to its memory-bound nature.

There are already a large number of studies on the performance optimization of stencil computation on CPUs and GPUs (Bertolacci et al. 2015; Guo et al. 2009; Habich et al. 2009; Matsumura et al. 2020; Micikevicius 2009; Mostafazadeh et al. 2018; Nguyen et al. 2010; Rawat et al. 2018, 2019; Rivera and Tseng 2000; Wellein et al. 2009). Although the computation capability of CPUs and GPUs has been increasing rapidly, their limited memory bandwidth makes it difficult for stencil computation to fully utilize the computation resources. Therefore, various tilling techniques (Bertolacci et al. 2015; Frigo and Strumpen 2005; Guo et al. 2009; Habich et al. 2009; Nguyen et al. 2010; Rivera and Tseng 2000; Wellein et al. 2009) have been proposed to exploit the data locality of stencil computation.

Meanwhile, Sunway TaihuLight is the first supercomputer with a peak performance of over 100 PFlops, ranking first in the TOP500 list from 2016 to 2017. The Sunway SW26010 processor adopted in Sunway TaihuLight can deliver promising performance with unique heterogeneous many-core

✉ Hailong Yang
hailong.yang@buaa.edu.cn

Biao Sun
biaosun@buaa.edu.cn

Mingzhen Li
lmzhhh@buaa.edu.cn

Jun Xu
xujun711@sina.com

Zhongzhi Luan
07680@buaa.edu.cn

Depei Qian
depeiq@buaa.edu.cn

[1] School of Computer Science and Engineering, Beihang University, Beijing 100191, China

[2] Science and Technology on Special System Simulation Laboratory Beijing Simulation Center, Beijing 100854, China

architecture. However, the performance gap between the computation capability and memory bandwidth of Sunway processor is even larger than CPUs and GPUs (Xu et al. 2017). Therefore, the stencil implementation without in-depth optimization on Sunway will suffer from the extremely low utilization of computation capability. In order to run stencil computation efficiently, we propose a combined tilling method to alleviate the memory bandwidth bottleneck and perform several architecture-tailored optimizations for performance acceleration.

Specifically, the contributions of this paper are as follows:

- We propose a combined (spatial, streaming, and temporal) tiling method for stencil computation tailored for Sunway processor. After carefully leveraging the special architectural features of Sunway processor, the proposed method can exploit the spatial and temporal locality of stencil computation to alleviate the memory bandwidth bottleneck of Sunway processor, which improves the performance of stencil computation.
- We utilize double buffering to overlap the DMA (Direct Memory Access) transfer with stencil computation, propose a vectorization method to eliminate unaligned and overlapped SIMD (Single Instruction Multiple Data) loads, and integrate collaborative memory accessing through register communication to eliminate redundant memory access and increase DMA bandwidth, which can further improve the performance of stencil computation.
- We evaluate our implementation on six stencil benchmarks with various memory access patterns and computation intensities on Sunway, and demonstrate its effectiveness with 1.97× speedup on average compared to the state-of-the-art stencil implementation. Following that, we present sufficient performance analysis experiments on the effects of different optimizations, which can serve as a basis for a comprehensive understanding of stencil optimization on Sunway processor.

The rest of this paper is organized as follows. Section 2 introduces the background of stencil computation and the Sunway many-core architecture. Section 3 presents the related work about the optimization of stencil computation. Section 4 describes the detailed design and implementation of our stencil optimizations on Sunway. Section 5 provides the evaluation results and performance analysis, and Sect. 6 concludes the paper.

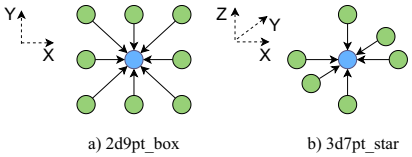


Fig. 1 The *2d9pt_box* and *3d7pt_star* stencil operators

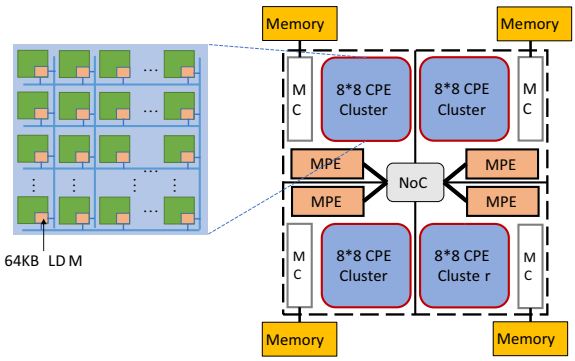


Fig. 2 The architecture of Sunway SW26010 processor

2 Background

2.1 Stencil computation

Stencil computation is one of the most important computation patterns in scientific computing applications. A stencil operator sweeps through the input grid and updates the value of each element by reading neighboring elements based on a specific computation pattern. A stencil operator can be time-iterated, sweeping through the entire grid multiple times. Stencil operators can be divided into different types according to dimension, points, shape, radius, timestep, etc. Figure 1 is the visualization of 2D 9-point stencil operator (*2d9pt_box*) with a shape of box and radius of 1 and 3D 7-point stencil operator (*3d7pt_star*) with a shape of star and radius of 1.

2.2 Architecture of Sunway processor

Sunway TaihuLight achieves a theoretical peak performance of 125 PFlops, integrating 40,960 Sunway heterogeneous many-core processors. The architecture of the Sunway processor is shown in Fig. 2. Sunway SW26010 processor contains four core groups (CG), and each CG contains one management processing element (MPE) and 64 computing processing elements (CPE). Each CPE is equipped with a 256-bit vector unit. For memory hierarchy, each CG is attached to 8 GB DDR3 memory. The

MPE has 32 KB L1 instruction cache, 32 KB L1 data cache and 256 KB L2 cache for both instruction and data. Each CPE has its own 16 KB L1 instruction cache and 64 KB local device memory (LDM) which is explicitly managed by the programmer. For main memory access, two approaches are provided, including global load/store (gld/gst) instructions and direct memory access (DMA). And the bandwidth of DMA is much greater than gld/gst. Besides, Sunway processor supports low-latency on-chip register data communication mechanism between CPEs in the same row/column.

3 Related work

3.1 Stencil optimization on CPUs and GPUs

There are quite a few works trying to improve the performance of stencil computation on CPUs and GPUs. Spatial tiling is a common tiling method to exploit the parallelism and locality of stencil computation in spatial dimensions (Micikevicius 2009; Mostafazadeh et al. 2018). However, spatial tiling only achieves data reuse within a single time step. Therefore, temporal tiling was further introduced to exploit the temporal locality (Habich et al. 2009; Wellein et al. 2009). 3.5D blocking (Nguyen et al. 2010) was presented to exploit both spatial and temporal data locality of stencil computation. Other tiling methods with different tiling shapes were also proposed (Bertolacci et al. 2015; Frigo and Strumpen 2005; Guo et al. 2009; Rivera and Tseng 2000). There are also efficient vectorization schemes (Yount et al. 2016; Yuan et al. 2021; Li et al. 2022) and advanced algorithmic techniques such as folding (Li et al. 2021), tessellating (Yuan et al. 2017) and fast fourier transforms (Ahmad et al. 2021) designed to further accelerate stencil computation on CPUs. In addition, stencil domain-specific languages (DSLs), such as STENCILGEN (Rawat et al. 2018), AN5D (Matsumura et al. 2020), and Artemis (Rawat et al. 2019) focus on automatic code generation for stencil computation, which can greatly reduce the burden of implementing high-performance stencils on CPUs and GPUs. Moreover, performance auto-tuning frameworks (Garvey and Abdelrahman 2015; Sun et al. 2021) have been proposed to better adapt the stencil computation patterns to the processor architectures.

3.2 Stencil optimization on Sunway processor

Although a large number of stencil optimizations have been proposed on CPUs and GPUs (Bertolacci et al. 2015; Guo et al. 2009; Habich et al. 2009; Matsumura et al. 2020; Micikevicius 2009; Mostafazadeh et al. 2018; Nguyen et al. 2010; Rawat et al. 2018, 2019; Rivera and Tseng 2000; Wellein et al. 2009), there are few works optimizing stencil computation on Sunway processors. Meanwhile, the optimizations of numerical operators such as GEMM (Jiang et al. 2017), SpGEMM (Chen et al. 2019), SpMV (Liu et al. 2018), SpTV (Chen et al. 2020) have been well studied on Sunway. Moreover, the existing works (Ao et al. 2017; Cai et al. 2018; Fu et al. 2017; Yang et al. 2016) on Sunway mainly focus on accelerating the large-scale applications that contain particular stencil computation, other than optimizing stencil computation with various patterns. For example, the optimizations for earthquake simulation (Fu et al. 2017) and atmospheric modeling (Ao et al. 2017) presented customized parallelization schemes to accelerate particular stencil patterns such as *3d13pt_star*. *MSC* (Li et al. 2021) is a new DSL designed to generate optimized stencil codes on Sunway. However, *MSC* only applies spatial tiling optimization tailored for Sunway architecture. In addition, there are other works that adopted the temporal tiling method time skewing (Tang et al. 2020) and proposed performance models to guide stencil optimization on Sunway (Liu et al. 2020). In general, the above works only provide a partial glimpse and fail to provide an in-depth optimization study of stencil computation with various patterns on Sunway processor. Particularly, except (Tang et al. 2020), all above works fail to exploit temporal tiling, and thus the data locality of stencil computation in the temporal dimension. Furthermore, without carefully addressing the unaligned and overlapped SIMD loads, all the above works fail to achieve the desired performance benefit of vectorization for optimizing stencil computation on Sunway.

4 Stencil optimizations on Sunway

4.1 Spatial tiling

There is no data dependency within a single timestep in stencil computation, which means that stencil computation has parallelism in the spatial dimension. Spatial tiling is a tiling method of stencil computation based on the spatial dimension, and it splits the computational grid into multiple tiles. Each CG of a Suwnay processor contains 1 MPE and 64 CPEs, thus we need to parallelize the stencil computation to 64 CPEs in order to utilize the LDM and computation resources of CPEs. Particularly, we use spatial tiling to split the computational grid into $64 * k$ tiles ($k = 1, 2, 3 \ldots$) as shown in Fig. 3. Each tile is loaded into the LDM of the corresponding CPE through DMA in order to utilize the memory bandwidth. After that, each CPE completes the stencil computation tasks with data in LDM and writes back the results to the main memory through DMA. Since stencil computation requires reading neighboring elements within radius R, each tile should contain both the computation

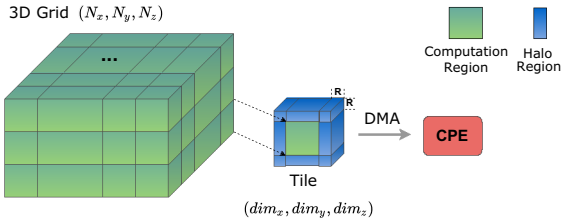


Fig. 3 Spatial tiling on Sunway processor

region and the extra halo region. We assume the size of each tile is: (dim_x, dim_y, dim_z), and the size of each grid element is denoted as ϵ. Because we need to load the whole tile into the LDM of CPE, the size of a tile cannot exceed the size of LDM, which requires: $\epsilon(dim_x dim_y dim_z) < 64KB$. Since the computation region of each tile is non-overlapped, there is overlapping between the halo region of tiles that belong to adjacent CPEs, which means that the elements in the halo region are accessed several times by adjacent CPEs. We use the ratio of redundant bandwidth to represent the cost: $((1 - 2R/dim_x)(1 - 2R/dim_y)(1 - 2R/dim_z))^{-1}$, where R is the radius of stencil operator (Nguyen et al. 2010). When $dim_x = dim_y = dim_z$, the ratio of redundant bandwidth gets its minimum. However, since we use DMA as the data transfer approach, according to the data transfer characteristic of DMA, the bandwidth of DMA increases when the data transfer size namely dim_x gets larger, so the tile size for optimal performance must be obtained through experiments (Li et al. 2018). This optimal tile size is a trade-off between the ratio of redundant bandwidth and DMA bandwidth, which we should pay attention to when using spatial tiling on Sunway.

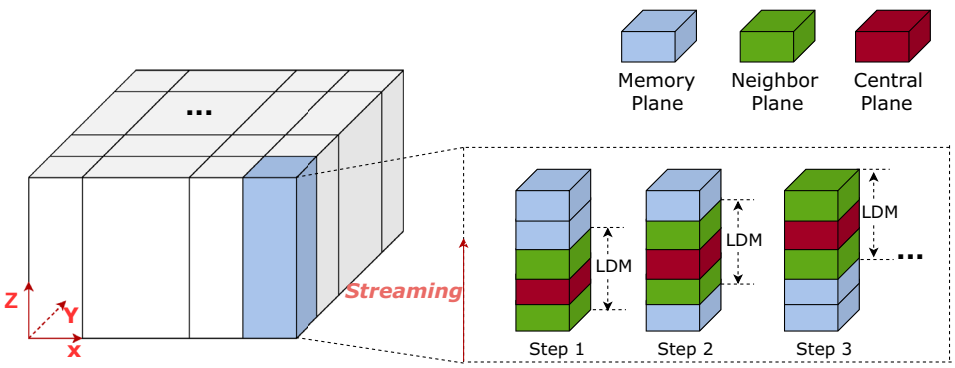
4.2 Streaming

Stencil computation has a fixed computation pattern. Only $(1 + 2R)$ planes need to be read when computing one plane. Therefore, an alternative is to use spatial tiling along two of the three spatial dimensions and stream along the third dimension. In this way, we no longer store the whole tile in LDM. Instead, as shown in Fig. 4, only $(1 + 2R)$ sub-planes along the streaming dimension need to reside in LDM. We use *Buffer* to represent the $(1 + 2R)$ XY sub-planes resident in LDM, the size of *Buffer* is $(dim_x, dim_y, 1 + 2R)$, which requires $\epsilon((1 + 2R)dim_x dim_y) < 64KB$. Since streaming eliminates redundant bandwidth along the streaming dimension, the ratio of redundant bandwidth is $((1 - 2R/dim_x)(1 - 2R/dim_y))^{-1}$ (Nguyen et al. 2010). We can find that streaming not only reduces redundant bandwidth but also reduces the LDM occupation. Smaller LDM occupation means that we can get larger dim_x, and therefore improve actual DMA bandwidth. Furthermore, by introducing streaming, we read one XY sub-plane at each Z iteration and then perform the corresponding stencil computation, rather than read the whole tile once and for all. This change makes DMA requests scattered over each Z iteration, which exposes more overlapped space between DMA and computation at each iteration. In-depth theoretical proof can be found in Xu et al. (2018). This positive impact of scattering DMA requests is rarely noticed in the related work of Sunway processor. We will illustrate this through the experiment in Sect. 5.

4.3 Combining spatial and temporal tiling

Temporal tiling realizes data reuse in the temporal dimension of stencil computation by computing consecutive dim_t timesteps without global memory access and therefore can reduce the amount of global memory access. The data dependency in the temporal dimension is resolved by redundant loading. As shown in Fig. 5, we apply temporal tiling (T dimension) combined with spatial tiling (XY dimension) and apply streaming along Z dimension. Therefore, we can perform consecutive dim_t timesteps of stencil computation without global memory access, and each timestep needs to buffer $(1 + 2R)$ XY sub-planes in LDM, which requires $\epsilon((1 + 2R)dim_t dim_x dim_y) < 64KB$. To resolve the data dependency in the temporal dimension for temporal tiling, we need to redundantly load $Rdim_t$ elements along

Fig. 4 Streaming on Sunway processor, when $R = 1$



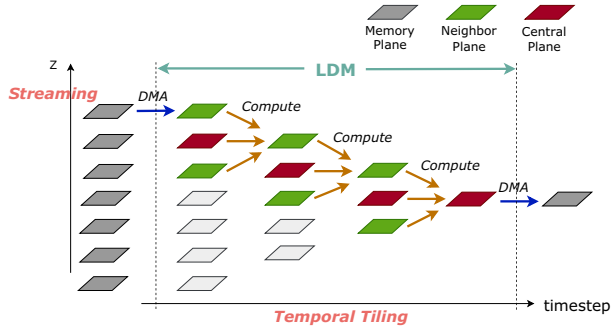


Fig. 5 Combined tiling on Sunway processor, when $dim_t = 3$ and $R = 1$

the X and Y dimension. So the ratio of redundant bandwidth is: $((1 - 2Rdim_t/dim_x)(1 - 2Rdim_t/dim_y))^{-1}$ (Nguyen et al. 2010). In addition, temporal tiling also introduces the overhead of redundant computation, whose computation formula is similar. In summary, by applying combined tiling, we exploit the locality of stencil computation in the temporal dimension and significantly increase the computation intensity due to the reduction of global memory access. However, noticeable overheads are introduced, including redundant bandwidth and redundant computation. Besides, the rapid increase of the LDM occupation makes dim_x smaller, which reduces the bandwidth of DMA. These factors limit the direct effectiveness of temporal tiling on Sunway processor. However, due to the significant increase of computation intensity, temporal tiling still plays an essential role in stencil optimizations on Sunway processor, which will be proved in Sect. 5.

4.4 Customized optimizations for Sunway architecture

4.4.1 Double buffering

To further exploit the architectural features of Sunway processor, we note that the DMA operation on Sunway processor is asynchronous, meaning that there is no need to wait for the data transfer to complete. Instead, the computation that does not depend on the data being transferred can be performed immediately. Therefore, we employ double buffering to designed optimization scheme. Each CPE allocates two data buffers for each DMA transfer, one for computation and the other for transfer, so that the DMA transfer and the computation are overlapped, as shown in Fig. 6. In our design, the combined tiling streams along the Z dimension and only initiates the DMA load at first timestep. After employing double buffering, we only need to store 1 more XY sub-plane in LDM at first timestep for the DMA load, which is quite small compared to the total LDM occupation

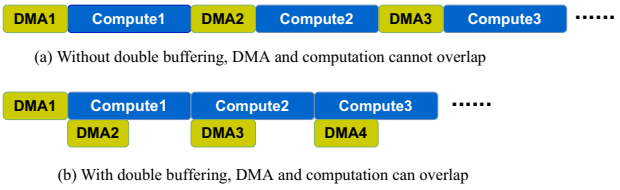


Fig. 6 Double buffering optimization

which is $(2 + 2R) + (dim_t - 1)(1 + 2R)$, thus it can improve the overall performance steadily.

4.4.2 Vectorization

Although the intuitive vectorization method is easy to implement for stencil computation, it faces the challenge of the unaligned and overlapped SIMD load instructions on Sunway processor. For example, a stencil operator with radius R has to read $(1 + 2R)$ elements in its central row. Assume that the datatype of the grid element is double and the central element x_c is 32 Bytes aligned in LDM. We emit SIMD load instructions at $x_{c\pm i}$ $(i \in \mathbb{N}, -R \le i \le R)$ respectively to get vectors $\vec{x_{c\pm i}}$ $(i \in \mathbb{N}, -R \le i \le R)$. However, the SIMD load instruction requires the data to be 32-Bytes aligned, and otherwise each load instruction will be split into more load/store instructions. Therefore, the overhead of unaligned SIMD load instruction is huge. Moreover, there is much overlap between the corresponding data region of SIMD load instructions, which further introduces redundancy and overhead. Therefore, we propose a stencil vectorization method based on the vector shuffle instruction of Sunway processor to address the challenges above. The SIMD shuffle instruction is one of the fastest SIMD instructions with a latency of one cycle. It can combine two vectors into a new one. It chooses two DP numbers in the first vector as the first two of the new vector and two DP numbers in the second vector as the last two of the new vector. Therefore, we can use SIMD load instruction respectively to get aligned vectors $\vec{x_{c\pm i}}$ $(i = 4k, k \in \mathbb{N}, -R \le i \le R)$, and obtain the remaining unaligned vectors $\vec{x_{c\pm i}}$ $(i \neq 4k, k \in \mathbb{N}, -R \le i \le R)$ through shuffle instructions. Figure 7 shows the implementation of stencil vectorization, and the radius of the stencil operator is

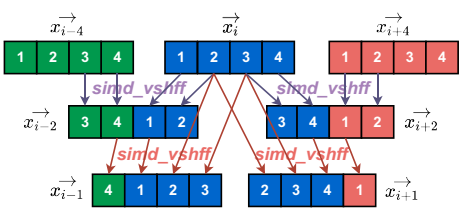


Fig. 7 The stencil vectorization method based on vector shuffle

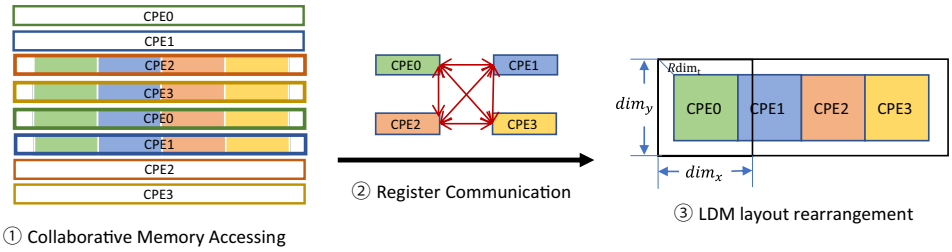


Fig. 8 The collaborative memory accessing scheme based on register communication

Table 1 The stencil benchmarks used in evaluation

Stencil	Dimension	Point	Shape	Radius	FLOPs/cell	Timestep
2d9pt_star	2D	9	Star	2	17	4
2d9pt_box	2D	9	Box	1	17	4
Gaussian	2D	25	Box	2	50	4
3d7pt_star	3D	7	Star	1	13	4
3d27pt_box	3D	27	Box	1	53	4
Helmholtz	3D	13	Star	2	17	4

2. The vectorization method for other stencil operators with different radius can be deduced based on the above proposed method and will not be elaborated here.

4.4.3 Register communication

After applying combined tiling, we exploit the data reuse in the temporal dimension. Meanwhile, the halo region is increased from R to $Rdimt$ and the tile size becomes small due to larger LDM occupation, which increases the amount of redundant memory access (i.e., redundant bandwidth) and decreases the bandwidth of DMA (i.e., unsaturated bandwidth). To resolve this issue, we integrate the collaborative memory accessing scheme (Ao et al. 2017) into our combined tiling algorithm. The number of CPEs in collaborative memory accessing groups should be 1, 2, 4, 8 due to the Sunway hardware implementation of CG and register communication. Based on our empirical study, the optimal number of CPEs within a group is 4, which is also adopted in Ao et al. (2017). Therefore, as shown in Fig. 8, 4 CPEs are bundled into a collaborative memory accessing group, and each CPE requests a chunk of continuous $4dim_x - 6Rdim_t$ data. The LDM layout of each CPE is then rearranged through exchanging data via register communication, and finally each CPE obtains its required data. With this scheme, redundant bandwidth is reduced from $((1-2Rdim_t/dim_x)(1-2Rdim_t/dim_y))^{-1}$ to $((1-2Rdim_t/(4dim_x-6Rdim_t))(1-2Rdim_t/dim_y))^{-1}$ and DMA bandwidth will increase significantly when the original dim_x is small. However, the overhead of on-chip register data communication and LDM layout rearrangement is also significant. Based on the above analysis, we can infer that this register communication approach is only effective when the redundant bandwidth is large and the DMA bandwidth is unsaturated, in other words, when the halo region is large and the tile size is small. We will further demonstrate it in Sect. 5.

Table 2 The parameter settings of *MSC*, *POS* and *CTS*

Stencil	Grid size	*MSC* tile size	*POS* tile size	*CTS* tile size
2d9pt_star	2048^2	(1,256,8)	(4,256,6)	(4,256,6)
2d9pt_box	2048^2	(1,256,8)	(4,512,4)	(4,512,4)
Gaussian	2048^2	(1,128,16)	(2,512,6)	(2,512,6)
3d7pt_star	128^3	(1,64,8,4)	(2,64,8,4)	(2,64,8,4)
3d27pt_box	128^3	(1,64,8,4)	(2,64,8,4)	(2,64,8,4)
Helmholtz	128^3	(1,64,8,4)	(2,32,8,6)	(2,32,8,6)

5 Evaluation

5.1 Experimental setup

In the experiments, we select six stencil operators as benchmarks, whose characteristics are listed in Table 1. We use Athread to implement the optimized version of benchmarks and set the number of CPE to 64. We compare our optimized implementation *CTS* with the state-of-the-art stencil DSL *MSC* (Li et al. 2021) and the state-of-the-art stencil implementation *POS* (Tang et al. 2020). Compared with *POS*, the stencil implementation for atmospheric modeling (Ao et al. 2017) did not apply time skewing but proposed a collaborative data accessing scheme based on register communication. However, it is not included in the performance comparison since the customized scheme only applies to 3D stencils and is only effective in limited cases. We will demonstrate it in the performance analysis experiment. Table 2

presents the optimal parameter settings of *MSC*, *POS*, and *CTS* across the benchmarks in experiments. For 2D stencils, the tile size parameters are (dim_t,dim_x,dim_y). For 3D stencils, the tile size parameters are (dim_t,dim_x, dim_y,dim_z). To measure the performance impact of six optimization methods, we conduct several performance analysis experiments and provide the roofline model analysis. Each experiment has been run ten times, with the average result reported. Note that, we focus on stencil optimization on a CG of Sunway SW26010 processor, which is orthogonal to large-scale optimization and is applicable in large-scale execution.

5.2 Performance comparison

The performance comparison results are shown in Fig. 9. The performance of our optimized implementation *CTS* achieves 3.03× speedup on average compared with the implementation generated by *MSC* and 1.97× speedup on average compared with the implementation of *POS*. This is because *MSC* only applies spatial tiling optimization and *POS* only applies spatial tiling, time skewing, and double buffering optimizations. In contrast, our implementation realizes data reuse in the spatial and temporal dimension of stencil computation through combined tiling and further leverages the computation resource of Sunway through double buffering, vectorization, and register communication. The significant performance improvement of our implementation demonstrates the importance of well-designed stencil optimizations tailored for Sunway processor.

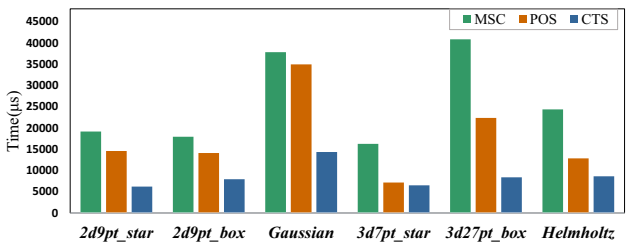


Fig. 9 Performance comparison between *MSC*, *POS*, and our implementation

5.3 Performance analysis

5.3.1 Ablation experiment results

Fig. 10 shows the results of ablation experiment. The baseline is the implementation generated by *MSC*. Among the 5 optimized implementations, the implementation applying only spatial tiling leads to 1.02× speedup over the baseline (*MSC*) on average. This is because although spatial tiling can parallelize the stencil computation to 64 CPEs, the performance is still bounded by the limited memory bandwidth of Sunway processor. By introducing streaming, the speedup increases to 1.44× on average, which proves that streaming can improve the overall performance considerably by reducing redundant bandwidth and increasing DMA bandwidth. The implementation which incrementally uses temporal tiling obtains a 1.43× speedup on average. Although temporal tiling fails to provide stable performance improvement due to its overhead, including redundant bandwidth, redundant computation, and DMA bandwidth reduction, however, it can increase computation intensity significantly and thus create more optimization space for subsequent optimizations. We will further explain it in the performance analysis experiment. By employing double buffering, the speedup increases to 1.61× on average. A further 1.84× speedup is achieved by enabling the vectorization. It provides a notable 2.31× performance improvement on stencil benchmarks with high computation intensity, including 2D stencils with high-degree temporal tiling (e.g., *2d9pt_star*, *2d9pt_box*, *Gaussian*), as well as high-order 3D stencils with box shape (e.g., *3d27pt_box*). These stencil benchmarks can be compute-bound without utilizing the computation capability of the vector units through vectorization. Register communication, an optional optimization scheme that only applies to 3D stencils and takes effect in limited cases, has performance promotion of 5.9% and 8.6% respectively on *3d7pt_star* and *Helmholtz*, while fails to work on *3d27pt_box*. The main reason is that the collaborative memory accessing scheme based on register communication only takes effect on memory-bound stencil benchmarks with high overheads of redundant bandwidth and unsaturated DMA bandwidth. We will

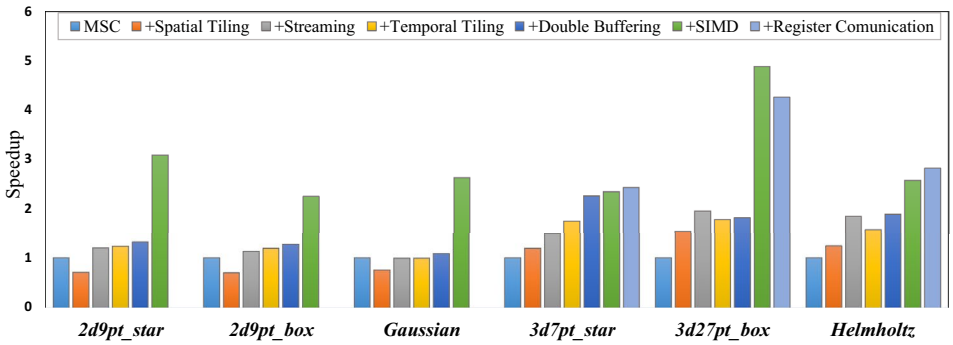


Fig. 10 Ablation experiment results, where *y*-axis indicates the speedup normalized to *MSC*

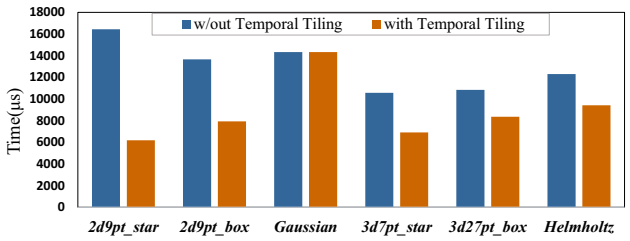


Fig. 11 The impact of temporal tiling

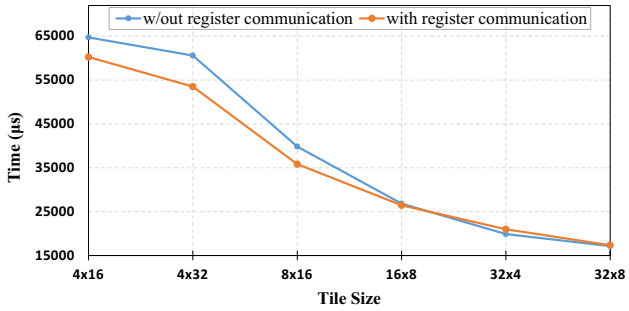


Fig. 12 The impact of collaborative memory accessing using register communication

further investigate it in the performance analysis experiment. And the final implementation using all six optimization methods achieves an impressive 3.03× speedup on average. It shows that after applying a series of optimization methods tailed for Sunway, we alleviate the bottleneck of memory bandwidth and fully exploit the architecture features of Sunway processor.

5.3.2 The impact of temporal tiling

To further verify the performance impact of temporal tiling, we compared the performance of the implementation using all the optimizations except temporal tiling with the final implementation using all the optimizations including temporal tiling, and the final experimental results are shown in Fig. 11. Compared with the version without temporal tiling, the version with temporal tiling achieves 1.59× speedup on average. For the four stencil benchmarks with low computation intensity (*2d9pt_star*, *2d9pt_box*, *3d7pt_star*, *Helmholtz*), it achieves 1.80× speedup on average. And in contrast, for the remaining two stencil benchmarks with high computation intensity (*Gassuian*, *3d27pt_box*), it achieves 1.15× speedup on average. This indicates that the increase of computation intensity is the main reason for the optimization effect of temporal tiling. Therefore, stencil benchmarks with low computation intensity obtain better optimization results due to the severe memory-bound nature, while stencil benchmarks with high computation intensity can still be compute-bound after enabling temporal tiling, and thus temporal tiling becomes less effective on these benchmarks.

5.3.3 The impact of register communication

As we mentioned in Sect. 4, through grouping several CPEs to access the main memory collaboratively, the register communication optimization can decrease redundant bandwidth and increase DMA bandwidth. To demonstrate this, we compared the optimized *Helmholtz* benchmark with the one without collaborative memory accessing, disabling temporal tiling to obtain clearer results. The experimental results are shown in Fig. 12. We can find that the register communication optimization can accelerate the benchmark in most tile size settings. Besides, the acceleration is more significant on settings with smaller tile size in total or in x-axis. However, the acceleration becomes minor or even negative when the tile size is 32×4 or 32×8. This is because when the tile size is large, DMA bandwidth is close to saturation due to the large DMA granularity, and the ratio of redundant bandwidth is relatively small. The performance improvement brought by the increase of DMA bandwidth and the decrease of redundant bandwidth is no longer significant. Therefore, the collaborative memory accessing scheme is only effective when the tile size is small and the halo region is large.

With the analysis above, we know that the register communication optimization reduces the amount of memory access and increases the memory access bandwidth, therefore applicable to memory-bound stencils. Meanwhile, although the overheads of register communication on *3d7pt_star* and *3d27pt_box* are similar due to the same radius and tile size, the optimized *3d27pt_box* can be compute-bound due to the higher computation intensity. Therefore, the overhead of on-chip register data communication and LDM layout rearrangement cannot be amortized, and hence the register communication optimization has side effects on the performance of *3d27pt_box*.

5.3.4 The impact of scattering DMA requests

In Sect. 4, we mention that streaming can scatter the DMA requests. To study the impact of scattering DMA requests, we take the implementation with only spatial tiling as the baseline *Spatial Tiling*. And we provide two implementations with streaming: *(1) Streaming* shares the same parameters with the baseline and therefore serves to scatter DMA requests, leaving other factors unchanged, *(2) Streaming-opt* uses the optimal parameters of tile size, which reduces redundant bandwidth and increases DMA bandwidth. The final experimental results of the performance comparison are shown in Fig. 13. The performance improvement of *Streaming* compared with *Spatial Tiling* only comes from

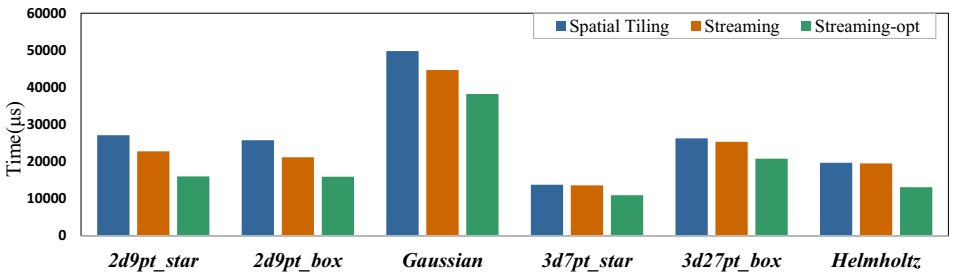


Fig. 13 The impact of scattering DMA requests

the scattering DMA requests. On the other hand, the performance improvement of *Streaming-opt* compared with *Streaming* comes from the optimal parameters of tile size. It can be seen that *Streaming* has a performance improvement of 8.3% on average compared with the baseline, which proves the positive performance effect of scattering DMA requests.

Unlike the baseline which requires initiating all DMA requests at once to read the whole tile, *Steaming* read one *XY* sub-plane at each *Z* iteration, which makes DMA requests scattered over each *Z* iteration. So after applying streaming, DMA and computation can be overlapped at each iteration, and thus the total overlapped time increases considerably, which improves the overall performance. In particular, the DMA request of 2D stencil is larger due to the larger dim_x of 2D tiles, which makes the improvement of scattering DMA requests more significant on 2D stencil benchmarks. The positive impact of scattering DMA requests has rarely been noticed in related optimization work on Sunway. And through the experiment, we emphasize that scattering DMA requests could improve the overall performance, especially when the transfer size of DMA is large.

5.4 Tile size sensitivity analysis

To better understand the performance sensitivity of tile size, we measure the performance under different settings of tile size on *Helmholtz*. Other stencil benchmarks show similar results as *Helmholtz*. The settings of $dim_t = 4$ are omitted as LDM is over-subscribed under most settings of $dim_t = 4$. In Fig. 14, $(dim_x - 2Rdim_t, dim_y - 2Rdim_t)$ represents the size of the inner region when the tile size is (dim_y, dim_x), and the pure white cell means that LDM is over-subscribed under the tile size setting. We can find that a larger tile size often leads to better performance due to the better exploitation of spatial and temporal locality. Although the redundant bandwidth reaches the minimum when $dim_x = dim_y$ as mentioned in Sect. 4, a larger dim_x can increase DMA bandwidth, therefore dim_x plays a much more significant role than dim_y in the performance of benchmark. When the setting turns to $dim_t = 2$ to increase the computation intensity, due to the doubled LDM occupation, the reachable maximum tile size is much smaller compared to when $dim_t = 1$, which can be

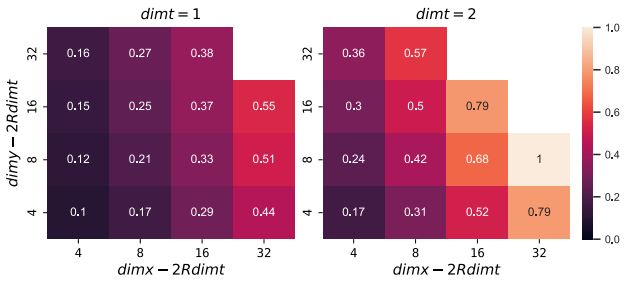


Fig. 14 Tile size sensitivity analysis of *Helmholtz*, where the value in each cell indicates the performance under the tile size, normalized to the best performance under optimal tile size

observed from the number of pure white cells in the figure. Therefore, the bandwidth of DMA becomes smaller when $dim_t = 2$.

With the analysis above, we can conclude that as long as the LDM occupation is smaller than 64 KB, the overall tile size should be as large as possible. The optimal value of dim_x and dim_y is a trade-off between redundant bandwidth and DMA bandwidth, while dim_x is much more significant. And the optimal value of dim_t is a trade-off between computation intensity and DMA bandwidth.

5.5 Roofline model analysis

To better understand the effects of our proposed optimizations for Sunway processor, we perform an analysis based on the roofline model. Due to the similar nature of stencil benchmarks, we only provide the roofline model analysis of *Helmholtz* for simplicity. As shown in Fig. 15, the computation intensity of the serial version on MPE is only 0.23 Flops/byte. After introducing spatial tiling, we parallelize the stencil computation to 64 CPEs, and each tile is stored in the LDM of the corresponding CPE, which greatly reduces the amount of memory access. As a result, the computation intensity increases to 0.67 Flops/Byte. Streaming improves the computation intensity through eliminating redundant memory access. Besides, Streaming increases the bandwidth of DMA due to the larger block size resulted by the reduction of LDM occupation. The computation intensity is increased by 2× after using temporal tiling, which creates

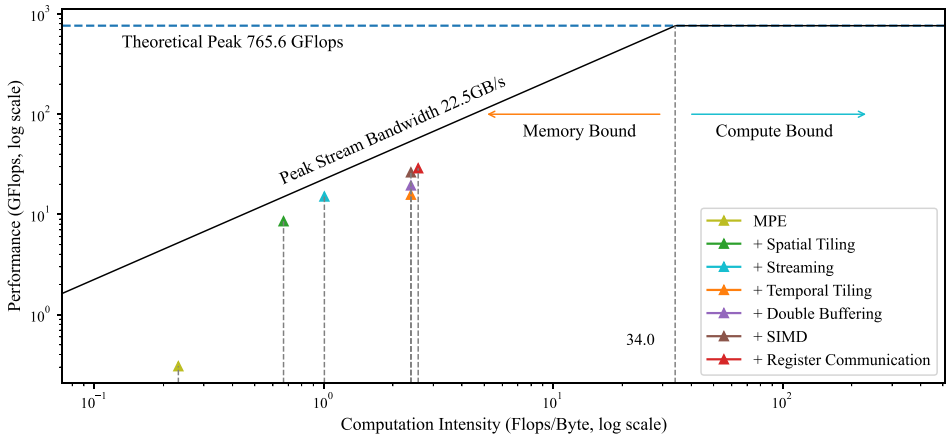


Fig. 15 The roofline model of *Helmholtz* stencil on Sunway

more optimization space for subsequent optimization. Double buffering and Vectorization further improve the performance while keeping the computation intensity unchanged. The computation instensity is further increased by the register communication optimization due to the decrease of redundant memory access. The register communication optimization also increases DMA bandwidth through the collaborative memory accessing.

The roofline model of a Sunway CG reveals that 33.84 Flops of calculation should be performed when accessing one-byte data in memory to fully utilize its performance. Compared with the computation intensity that we achieve, increasing the computation intensity is still the key to improving the performance of memory-bound stencil benchmarks, which is the main direction for our future work.

6 Conclusion

In this paper, we proposed a combined (spatial, streaming, and temporal) tiling method tailored for Sunway processor. Through effectively exploiting the massive parallelism and data locality of stencil computation, the proposed method mitigates the performance gap between the computation capability and the memory bandwidth of Sunway processor. Further performance improvements have been achieved by implementing double buffering, vectorization, and register communication. The experimental results on six stencil benchmarks demonstrate the effectiveness of our method, which achieves 1.97× speedup on average compared to the state-of-the-art stencil implementation on Sunway processor. Various performance analysis experiments are further presented, which can serve as a basis for the comprehensive understanding of stencil optimizations on Sunway processor.

Acknowledgements This work was supported by National Key Research and Development Program of China (No. 2022ZD0117805), National Natural Science Foundation of China (No. 62072018 and U22A2028), the Fundamental Research Funds for the Central Universities, and Iluvatar CoreX semiconductor Co., Ltd. Hailong Yang is the corresponding author.

Data availability The authors confirm that the data supporting the findings of this study are available within the article.

Declarations

Conflict of interest The authors declared that they have no conflicts of interest to this work.

References

Ahmad, Z., Chowdhury, R., Das, R., Ganapathi, P., Gregory, A., Zhu, Y.: Fast stencil computations using fast fourier transforms. In: Proceedings of the 33rd ACM Symposium on Parallelism in Algorithms and Architectures, pp. 8–21 (2021)

Ao, Y., Yang, C., Wang, X., Xue, W., Fu, H., Liu, F., Gan, L., Xu, P., Ma, W.: 26 pflops stencil computations for atmospheric modeling on sunway taihulight. In: 2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 535–544 (2017). https://doi.org/10.1109/IPDPS.2017.9

Bertolacci, I.J., Olschanowsky, C., Harshbarger, B., Chamberlain, B.L., Wonnacott, D.G., Strout, M.M.: Parameterized diamond tiling for stencil computations with chapel parallel iterators. In: Proceedings of the 29th ACM on International Conference on Supercomputing, pp. 197–206 (2015)

Cai, Y., Yang, C., Ma, W., Ao, Y.: Extreme-scale realistic stencil computations on sunway taihulight with ten million cores. In: 2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), pp. 566–571 (2018). https://doi.org/10.1109/CCGRID.2018.00086

Chen, Y., Li, K., Yang, W., Xiao, G., Xie, X., Li, T.: Performance-aware model for sparse matrix-matrix multiplication on the sunway taihulight supercomputer. IEEE Trans. Parallel Distrib. Syst. **30**(4), 923–938 (2019). https://doi.org/10.1109/TPDS.2018.2871189

Chen, Y., Xiao, G., Özsu, M.T., Liu, C., Zomaya, A.Y., Li, T.: AESPTV: an adaptive and efficient framework for sparse tensor-vector product kernel on a high-performance computing platform. IEEE Trans. Parallel Distrib. Syst. **31**(10), 2329–2345 (2020). https://doi.org/10.1109/TPDS.2020.2990429

Dongarra, J., Peterson, G., Tomov, S., Allred, J., Natoli, V., Richie, D.: Exploring new architectures in accelerating cfd for air force

applications. In: 2008 DoD HPCMP Users Group Conference, pp. 472–478. IEEE (2008)
Frigo, M., Strumpen, V.: Cache oblivious stencil computations. In: Proceedings of the 19th Annual International Conference on Supercomputing, pp. 361–366 (2005)
Fu, H., He, C., Chen, B., Yin, Z., Zhang, Z., Zhang, W., Zhang, T., Xue, W., Liu, W., Yin, W., et al.: 9-pflops nonlinear earthquake simulation on sunway taihulight: enabling depiction of 18-hz and 8-meter scenarios. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–12 (2017)
Garvey, J.D., Abdelrahman, T.S.: Automatic performance tuning of stencil computations on gpus. In: 2015 44th International Conference on Parallel Processing, pp. 300–309. IEEE (2015)
Guo, J., Bikshandi, G., Fraguela, B.B., Padua, D.: Writing productive stencil codes with overlapped tiling. Concurr. Comput. Pract. Exp. **21**(1), 25–39 (2009)
Habich, J., Zeiser, T., Hager, G., Wellein, G.: Enabling temporal blocking for a lattice Boltzmann flow solver through multicore-aware wavefront parallelization. In: 21st International Conference on Parallel Computational Fluid Dynamics, pp. 178–182 (2009)
Jiang, L., Yang, C., Ao, Y., Yin, W., Ma, W., Sun, Q., Liu, F., Lin, R., Zhang, P.: Towards highly efficient dgemm on the emerging sw26010 many-core processor. In: 2017 46th International Conference on Parallel Processing (ICPP), pp. 422–431 (2017). https://doi.org/10.1109/ICPP.2017.51
Li, L., Fang, J., Fu, H., Jiang, J., Zhao, W., He, C., You, X., Yang, G.: swcaffe: A parallel framework for accelerating deep learning applications on sunway taihulight. In: 2018 IEEE International Conference on Cluster Computing (CLUSTER), pp. 413–422 (2018). https://doi.org/10.1109/CLUSTER.2018.00087
Li, M., Liu, Y., Yang, H., Hu, Y., Sun, Q., Chen, B., You, X., Liu, X., Luan, Z., Qian, D.: Automatic code generation and optimization of large-scale stencil computation on many-core processors. In: 50th International Conference on Parallel Processing, pp. 1–12 (2021)
Li, K., Yuan, L., Zhang, Y., Yue, Y.: Reducing redundancy in data organization and arithmetic calculation for stencil computations. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–15 (2021)
Li, K., Yuan, L., Zhang, Y., Yue, Y., Cao, H.: An efficient vectorization scheme for stencil computation. In: 2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 650–660. IEEE (2022)
Liu, C., Xie, B., Liu, X., Xue, W., Yang, H., Liu, X.: Towards efficient spmv on sunway manycore architectures. In: Proceedings of the 2018 International Conference on Supercomputing. ICS '18, pp. 363–373. Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3205289.3205313
Liu, Y., Liu, L., Hu, M., Wang, W., Xue, W., Zhu, Q.: Performance modeling of stencil computation on sw26010 processors. In: Qiu, M. (ed.) Algorithms and Architectures for Parallel Processing, pp. 386–400. Springer, Cham (2020)
Matsumura, K., Zohouri, H.R., Wahib, M., Endo, T., Matsuoka, S.: An5d: automated stencil framework for high-degree temporal blocking on gpus. In: Proceedings of the 18th ACM/IEEE International Symposium on Code Generation and Optimization, pp. 199–211 (2020)
Micikevicius, P.: 3d finite difference computation on gpus using cuda. In: Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units, pp. 79–84 (2009)
Mostafazadeh, B., Marti, F., Liu, F., Chandramowlishwaran, A.: Roofline guided design and analysis of a multi-stencil cfd solver for multicore performance. In: 2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 753–762. IEEE (2018)
Nguyen, A., Satish, N., Chhugani, J., Kim, C., Dubey, P.: 3.5-d blocking optimization for stencil computations on modern cpus and gpus. In: SC '10: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–13 (2010). https://doi.org/10.1109/SC.2010.2
Powers, J.G., Klemp, J.B., Skamarock, W.C., Davis, C.A., Dudhia, J., Gill, D.O., Coen, J.L., Gochis, D.J., Ahmadov, R., Peckham, S.E., et al.: The weather research and forecasting model: overview, system efforts, and future directions. Bull. Am. Meteor. Soc. **98**(8), 1717–1737 (2017)
Rawat, P.S., Vaidya, M., Sukumaran-Rajam, A., Ravishankar, M., Grover, V., Rountev, A., Pouchet, L.-N., Sadayappan, P.: Domain-specific optimization and generation of high-performance gpu code for stencil computations. Proc. IEEE **106**(11), 1902–1920 (2018)
Rawat, P.S., Vaidya, M., Sukumaran-Rajam, A., Rountev, A., Pouchet, L.-N., Sadayappan, P.: On optimizing complex stencils on gpus. In: 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 641–652. IEEE (2019)
Rivera, G., Tseng, C.-W.: Tiling optimizations for 3d scientific computations. In: SC'00: Proceedings of the 2000 ACM/IEEE Conference on Supercomputing, p. 32. IEEE (2000)
Sun, Q., Liu, Y., Yang, H., Jiang, Z., Liu, X., Dun, M., Luan, Z., Qian, D.: cstuner: Scalable auto-tuning framework for complex stencil computation on gpus. In: 2021 IEEE International Conference on Cluster Computing (CLUSTER) (2021)
Tang, Y., Li, M., Chen, Z., Xue, C., Zhao, C., Yang, H.: Parallel optimization of stencil computation base on sunway taihulight. In: Sun, X., Wang, J., Bertino, E. (eds.) Artificial Intelligence and Security, pp. 141–152. Springer, Singapore (2020)
Wellein, G., Hager, G., Zeiser, T., Wittmann, M., Fehske, H.: Efficient temporal blocking for stencil computations by multicore-aware wavefront parallelization. In: 2009 33rd Annual IEEE International Computer Software and Applications Conference, vol. 1, pp. 579–586. IEEE (2009)
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 (2017). https://doi.org/10.1109/IPDPSW.2017.9
Xu, S., Xu, Y., Xue, W., Shen, X., Zheng, F., Huang, X., Yang, G.: Taming the" monster": Overcoming program optimization challenges on sw26010 through precise performance modeling. In: 2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 763–773. IEEE (2018)
Yang, C., Xue, W., Fu, H., You, H., Wang, X., Ao, Y., Liu, F., Gan, L., Xu, P., Wang, L., et al.: 10m-core scalable fully-implicit solver for nonhydrostatic atmospheric dynamics. In: SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 57–68. IEEE (2016)
Yount, C.R., Tobin, J., Breuer, A., Duran, A.: Yask-yet another stencil kernel: A framework for hpc stencil code-generation and tuning. 2016 Sixth International Workshop on Domain-Specific Languages and High-Level Frameworks for High Performance Computing (WOLFHPC), pp. 30–39 (2016)
Yuan, L., Zhang, Y., Guo, P., Huang, S.: Tessellating stencils. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–13 (2017)
Yuan, L., Cao, H., Zhang, Y., Li, K., Lu, P., Yue, Y.: Temporal vectorization for stencils. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. SC '21. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3458817.3476149

Biao Sun is working toward the graduate degree in the School of Computer Science and Engineering, Beihang University. He is currently working on performance optimization of scientific applications. His research interests include HPC, performance analysis and optimization.

Mingzhen Li is a PhD student in School of Computer Science and Engineering, Beihang University. He is currently working on identifying performance opportunities for scientific applications. His research interests include HPC, performance optimization, and code generation.

Hailong yang is an associate professor in School of Computer Science and Engineering, Beihang University. He received the Ph.D degree in the School of Computer Science and Engineering, Beihang University in 2014. His research interests include parallel and distributed computing, HPC, performance optimization and energy efficiency.

Jun Xu is a senior engineer in Beijing Simulation Center of the Second Institute of CASIC. She received the Ph.D degree of computer science and technology in Zhejiang University in 2011. Her research interest is modeling and simulation of weapon equipment system.

Zhongzhi Luan received the Ph.D. in the School of Computer Science of Xi'an Jiaotong University. He is an Associate Professor of Computer Science and Engineering, and Assistant Director of the Sino-German Joint Software Institute (JSI) Laboratory at Beihang University, China. Since 2003, His research interests including distributed computing, parallel computing, grid computing, HPC and the new generation of network technology.

Depei Qian is a professor at the Department of Computer Science and Engineering, Beihang University, China. He received his master degree from University of North Texas in 1984. He is currently serving as the chief scientist of China National High Technology Program (863 Program) on high productivity computer and service environment. He is also a fellow of China Computer Federation (CCF). His research interests include innovative technologies in distributed computing, high performance computing and computer architecture.