Received: 31 January 2017 | Revised: 3 May 2018 | Accepted: 31 May 2018
DOI: 10.1002/cpe.4758

WILEY

SPECIAL ISSUE PAPER

Implementation and optimization of a data protecting model on the Sunway TaihuLight supercomputer with heterogeneous many-core processors

Yuedan Chen[1] | Kenli Li[1,2] | Xiongwei Fei[1] | Zhe Quan[1] | Keqin Li[1,2,3]

[1]College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China
[2]National Supercomputing Center in Changsha, Changsha 410082, China
[3]Department of Computer Science, State University of New York, New York, NY 12561, USA

Correspondence
Kenli Li, College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China.
Email: lkl@hnu.edu.cn

Funding information
National Key R&D Program of China, Grant/Award Number: 2016YT80201900; Key Program of National Natural Science Foundation of China, Grant/Award Number: 61432005; National Outstanding Youth Science Program of National Natural Science Foundation of China, Grant/Award Number: 61625202; Open Fund of the State Key Laboratory of Mathematical Engineering and Advanced Computing, Grant/Award Number: 898209040; Youth Program of National Natural Science Foundation of China, Grant/Award Number: 61602166

Summary

With the rapid development of information technology, the security of massive amounts of digital data has attracted huge attention in recent years. The Advanced Encryption Standard (AES) algorithm and the Security Hash Algorithm 3 (SHA3) are extensively used as cryptographic algorithms for protecting the security of information. The Sunway TaihuLight, with massive heterogeneous many-core SW26010 processors, has the peak performance of over 100 PFlops. To achieve high efficiency of data encryption/decryption and guarantee the data integrity for large-scale applications, this paper proposes a fast and secure data protecting model using the parallel AES algorithm and the SHA3 on the Sunway TaihuLight. According to the particular computing architecture and memory hierarchy of the Sunway TaihuLight, we propose a fine-grained software design for the data protecting model to fully exploit the parallelism and properly arranges the data on the Sunway. Furthermore, we propose optimization strategies for the parallel AES algorithm of the data protecting model. It is proved that our data protecting model has high security, good scalability, and excellent encryption/decryption efficiency on the Sunway. Our data protecting model achieves a high throughput of 269.95 Gbits/s, and the optimized parallel AES algorithm achieves 511.28 Gbits/s on the Sunway.

KEYWORDS

advanced encryption standard (AES), heterogeneous many-core processor, parallelism, security hash algorithm 3 (SHA3), Sunway TaihuLight supercomputer

1 | INTRODUCTION

1.1 | Motivation

In recent years, many network platforms, such as micro-blog, forums, and e-commerce sites, have become increasingly popular. Large amount of digital data is generated and stored in the internet. As the activity of internet users progressively increases, we are faced with the risk of security and confidentiality of massive digital information. In addition, the growing number of incidences of data leakage leads serious impact and losses to the internet users, which makes the data security become the hot point and a major concern of the society. It is extremely urgent to figure out a way to provide a reliable and safe environment for internet users.

Using the cryptographical technology is one of the key methods to ensure the security of digital data. In cryptography, the block cipher has strong applicability and security. It executes encryption and decryption on a set of fixed-size data blocks using a symmetric key. The Advanced Encryption Standard (AES)[1] is a block cipher extensively used by enterprises and organizations for protecting the security of information. However, the traditional AES encryption/decryption is a computationally intensive task, which leads to the inefficiency of encrypting/decrypting large-scale digital data. In addition, the traditional AES algorithm is unable to meet the demand of protecting massive data efficiently. Users cannot accept too much time spent on data encryption/decryption. Normally, the AES algorithm takes a 16-byte data block as input and performs several rounds of transformations to encrypt/decrypt data blocks. Thus, there is no data dependency in the process of AES encryption/decryption, and the AES algorithm is naturally amenable to parallelism. In order to overcome the disadvantages of the traditional AES algorithm, we endeavor to implement efficient AES algorithms using parallelism based on high-performance computing platforms.

The SHA, a cryptographic standard, is also widely used to ensure the security of digital data. The SHA is one of the cryptographic hash algorithms that are widely used in many information security applications, such as digital signatures and message authentication codes (MACs). The SHA can efficiently map any-sized data to a fixed-size bit string. It is a one-way function, which means that it is unworkable to invert the hash results back to the input data. The SHA3 is the latest one in the family of secure hash algorithms. Therefore, using the SHA3 to detect the data corruption and guarantee the data integrity is an effective method to ensure the security of digital data.

The Sunway TaihuLight, a Chinese supercomputer, has been the world's fastest supercomputer ranked in the TOP500 list since June 2016.[2] It is also the third most energy-saving supercomputer in the TOP500 list. The Sunway TaihuLight is developed based on the heterogeneous many-core SW26010 processor instead of NVIDIA GPGPU or Intel Xeon Phi. There are 40,960 China-designed SW26010 processor chips in the Sunway TaihuLight system. A heterogeneous many-core SW26010 processor chip includes 4 management processing element (MPE) cores and 256 computing processing element (CPE) cores. Thus, each processor chip contains 260 cores, and the whole system has 10,649,600 cores. How to achieve the maximum utilization of supercomputers' supercomputing power in high performance applications is one of the main challenges nowadays.[3,4]

Employing the world's fastest supercomputer, the Sunway TaihuLight supercomputer, this paper implements a fast and secure data protecting model using the parallel AES algorithm and the SHA3 to achieve high efficiency of data encryption/decryption and guarantee the data integrity. Furthermore, this paper proposes optimization strategies for the parallel AES algorithm to get better performance for our data protecting model. The Sunway TaihuLight supercomputer can dramatically enhance the overall performance of computationally intensive tasks, and AES encryption is a compute-intensive problem with complex processes. Therefore, the parallel AES algorithm can get greatly performance improvement by fully exploiting the parallelism of the Sunway. Especially, each CPE core of a SW26010 processor is equipped with a Scratch Pad Memory (SPM) instead of cache. The SPM is efficient and effective for temporary storage, and it would not need to always access to the main memory. Hence, we design a fine-grained software design based on the special computing architecture and memory hierarchy of the Sunway and optimize the mechanism of memory access in our method. The parallel AES algorithm provides high efficiency for the data protecting model, and the SHA3 guarantees the data integrity for the data protecting model on the Sunway. The proposed work demonstrates great significance in the research fields of information security and high-performance computing and provides insights to solve the aforementioned problems.

1.2 | Our contributions

In this paper, we propose a fast and secure data protecting model using the parallel AES algorithm and the SHA3 on the Sunway TaihuLight to gain high efficiency of data encryption/decryption and guarantee the data integrity. According to our detailed understanding of the computing architecture and communication mechanism of the Sunway TaihuLight system, a fine-grained software design is presented to implement the the data protecting model on the Sunway. In addition, we further optimize the parallel AES algorithm to achieve more excellent performance of data encryption/decryption for the data protecting model. Our main contributions are summarized as follows.

(i) We propose a fast and secure data protecting model using the parallel AES algorithm and the SHA3 on the Sunway TaihuLight. The data protecting model not only gains high efficiency of data encryption/decryption but also guarantees the security and integrity of data for many large-scale information security applications.

(ii) We present a fine-grained software design to implement the data protecting model according to the special computing architecture and memory hierarchy of the Sunway TaihuLight. The fine-grained software design for the data protecting model fully exploits the parallelism and properly arranges the data on the Sunway. The performance of the parallel AES algorithm of our data protecting model on the Sunway is analyzed in detail. Both theory analysis and experiments prove that our data protecting model based on the Sunway has good scalability, high efficiency, and high security.

(iii) We further propose optimization strategies for the parallel AES algorithm to get better performance of data encryption/decryption for proposed data protecting model on the Sunway TaihuLight. We also analyze the performance of the optimized parallel AES algorithm. Both theory analysis and experiments prove that the performance of data encryption/decryption of the data protecting model is optimized by applying the proposed optimization strategies.

A primary version of this paper presented implementation and optimization of the AES algorithm on the Sunway TaihuLight.[5] This extension paper explores another cryptographic algorithm, the SHA3, and proposes a data protecting model using the parallel AES algorithm and the SHA3 on the Sunway TaihuLight. In addition, new experimental results of the data protecting model are presented and analyzed.

The rest of this paper is organized as follows. Related works about efficient AES algorithms and the SHA are summarized in Section 2. Section 3 demonstrates the cryptographic algorithms, ie, the AES algorithm and the SHA3, involved in this paper. Section 4 has a detailed description about the Sunway TaihuLight supercomputer and the SW26010 processor. Section 5 presents the implementation and optimization of our data protecting using the parallel AES algorithm and the SHA3 on the Sunway TaihuLight. Section 6 provides experimental results and analysis of the single-CG and multi-CG experiments. Finally, in Section 7, we conclude this paper.

2 | RELATED WORK

With the improvement of high-performance computing technology and high requirement of data security, several methods are proposed to improve the efficiency of the AES algorithm.

Applying hardware is an effective way to gain the performance improvement of AES. Hodjat and Verbauwhede[6] presented area-efficient architectures for fully pipelined high speed AES processors; they can provide an encryption throughput of 30 to 70 Gbits/s for a 0.18-m CMOS ASIC technology. Wang et al[7] designed a configurable architecture with a group of AES processors as its major building blocks. The multi-core architecture made high throughput rate and more cost effective. Mathew et al[8] designed an on-die reconfigurable AES encrypt/decrypt hardware accelerator fabricated in 45nm CMOS for for content-protection in high-performance microprocessors. Desai et al[9] proposed an optimized architecture of the parallel AES algorithm for disk encryption, suitable to the multicore environment. Patel et al[10] developed a low-cost parallel architecture for the fast AES algorithm using VHDL. Parmar and Kadam[11] designed a pipelined architecture for S-Box of the AES algorithm. The experimental results demonstrate that the AES algorithm, with the pipelined S-Box, performs well based on FPGA, and they can get higher throughput and less delay. Hodjat and Verbauwhede[12] proposed an AES encryption processor fully utilizing the pipeline technique on a single chip FPGA. Thulasimani and Madheswaran[13] proposed an efficient hardware architecture of the AES encryption based on FPGA and VHDL, which showed high efficiency and energy saving. Wang and Ha[14] presented the FPGA-based masked AES with area optimization for Storage Area Network (SAN), and in another work of the aforementioned authors,[15] they developed a high throughput and resource efficient AES encryption/decryption on FPGA for SAN to fully exploit the dedicated resources of modern FPGAs. Rahimunnisa et al[16] presented a folded architecture of the AES algorithm with parallel operations, and they achieved a high throughput of the AES algorithm.

However, the hardware implementations of AES have poor scalability and flexibility. Many research studies are focused on parallelizing the AES algorithm on GPUs or/and CPUs and improve the efficiency of AES encryption and decryption for large-scale data. Harrison and Waldron[17] proposed some new methods for the implementation of the AES algorithm on GPUs, and they found that the GPU could be used effectively as a coprocessor. Tran et al[18] proposed the parallelism of AES-CTR algorithm with extended block size by using a general-purpose many-core processor and a GPU. Particularly, the performance improvement on GPU is dramatic because of the “multi-core” nature of the GPU architecture. Luo et al[19] designed a side-channel power analysis methodology to extract all of the last round key bytes of the AES algorithm implemented by CUDA on GPU. They found that GPU is highly vulnerable target to power-based side-channel attacks and needs to be hardened against side-channel threats. Patchappen et al[20] implemented the batch execution of multi-variant AES encryption on GPU, performing up to 3 times and 1.6 times faster than single-core and multi-core CPU, respectively. Adeshina and Hashim[21] proposed a safe radiology-diagnostic data framework for connected health network with the accelerative AES algorithm based on GPU. Khan et al[22] executed the AES-128 ECB encryption on two GPUs and revealed the effects of input plaintext patterns. Guo et al[23] implemented the fast AES algorithm using the AES-NI extended instruction sets based on CUDA. Moreover, Niewiadomska-Szynkiewicz et al[24] obtained high performance improvement of encrypting massive data by using both multi-core CPUs and many-core GPUs. Marks and Niewiadomska-Szynkiewicz[25] solved the problems associated with distributed computational systems and the application of hybrid GPU and CPU technology to data intensive computation.

In this paper, we implement parallel AES algorithm on the Sunway TaihuLight supercomputer. The implementations of the AES algorithm based on different architectures vary greatly. The customized software design for the AES algorithm that exploits the powerful parallel processing ability of the Sunway TaihuLight is unique. As far as we know, we take the first step to implement the efficient AES algorithm on the Sunway TaihuLight.

In addition, the SHA is widely used in many information security applications to further ensure the confidentiality and integrity of data. Bayat-Sarmadi et al[26] proposed an efficient concurrent error detection scheme for the SHA-3 algorithm and provided reliable architectures for this algorithm. Moreira et al[27] focused on the use of AES-128 and new SHA-3 based new Message Authentication Code (MAC) algorithms for securing Precision Time Protocol (PTP) messages. Arshad et al[28] presented a compact design of SHA-3 on FPGA. The design is logically optimized for area efficiency by merging Rho, Pi, and Chi steps of algorithm into a single step. Ahmed and Farag[29] presented a novel design of a dynamically configurable hardware accelerator for SHA-3 standard. Xiao et al[30] proposed a security management, IMSet-SHA3-Tree, which guarantees data integrity strongly through an integrity checking module based on SHA3-TAH and incremental multiset hash tree with a reduced overhead in area, memory footprint, and performance. Fei et al[31] proposed a safe and efficient file protecting system using the parallel AES algorithm and the SHA3 on GPUs or/and CPUs.

Therefore, we design a fast and secure data protecting model for large-scale data using the combination of AES algorithm and the SHA3 based on the Sunway architecture. The proposed model guarantees not only the high speed of AES encryption and decryption but also the security and integrity of large-scale data on the Sunway TaihuLight. To the best of our acknowledgement, we also take the first step to implement a secure and efficient data protecting system on the Sunway TaihuLight.

3 | CRYPTOGRAPHIC ALGORITHMS

3.1 | The AES algorithm

The AES algorithm, also named Rijndael, is a symmetric-key block cipher algorithm. It is an encryption standard of encrypting data established by the U.S. National Institute of Standards and Technology (NIST) in 2001.[1] The AES algorithm encrypts fixed data block size of 16 bytes using a key with size of 128, 192, or 256 bits.

Each 16-byte data block is represented as a 4 × 4 matrix, called a state. The input data is known as the plaintext, and the output data is called the ciphertext. The AES encryption procedure consists of four basic transformations, ie, SubBytes, ShiftRows, MixColumns, and AddRoundKey. The encryption procedure repeats transformation rounds with key to convert the plaintext into the ciphertext. The number of repetitions depends on the length of the key, specifically 10 times repetitions for 128-byte keys, 12 times repetitions for 192-byte keys, and 14 times repetitions for 256-byte keys.

The aforementioned key (called initial key) is entered by users. However, the AES uses different 128-bit round key, generated by initial key using Rijndael's key schedule (called KeyExpansion) in each round. These generated keys and the initial key are collectively called subkeys.

In encryption process, there is only one AddRoundKey operation in the initial round firstly. In this round, each byte of the state is transformed with a block of initial key using the bitwise XOR operation.

Then, the transformation steps in subsequent rounds are implemented as the following order.

(i) SubBytes: This is a non-linear substitution operation. In this step, each byte of the input is substituted by the element of substitution box (called the S-box).
(ii) ShiftRows: This is a transposition operation. In this step, the last three rows of the state are shifted with several rounds. The first row remains constant, whereas the second row cyclically rotates one byte to the left, the third row cyclically rotates two bytes to the left, and the fourth row cyclically rotates three bytes to the left.
(iii) MixColumns: This is a mixing operation. In order to mix each column of the state adequately, this step uses linear transformations to combine four bytes in each column.
(iv) AddRoundKey: This step performs the bitwise XOR operation to generate a subkey with each byte of the state and the subkey for this round.

The last round does not have the MixColumns step, and the transformation steps are implemented in the order of SubBytes, ShiftRows, and AddRoundKey.

After each plaintext block is encrypted sequentially, these result ciphertext blocks are merged into the final ciphertext.

3.2 | The SHA3

The SHA3 is one of the cryptographic algorithms in the family of Secure Hash Algorithms (SHA), released by NIST on August 5, 2015. The SHA is a group of cryptographic hash functions that maps any-sized data to a fixed-size bit string. Particularly, it is a one-way function, which means that it is unworkable to invert the hash results back to the input data.

There are several main characters of cryptographic hash functions. First, the same input data has the same hash result. It is impossible that two different input data has the same hash value. Second, the function is efficient. Third, it is unworkable to convert the hash results back to the input data. Fourth, a small change of the input data will result in a new hash value, and there is no correlation between the old and new hash values. Therefore, cryptographic hash functions are applied in many applications of the information security to detect data corruption and guarantee the security of data storage and transmission.

The SHA3 is the newest hash function in SHA. According to different output sizes, there are SHA3-224, SHA3-256, SHA3-384, SHA3-512, etc, whose size of the hash value respectively is 224, 256, 384, and 512 bits. In this paper, we use SHA3-256 to compute hash values.

4 | THE SUNWAY TAIHULIGHT SUPERCOMPUTER

The Sunway TaihuLight was designed by the National Research Center of Parallel Computer Engineering and Technology (NRCPC) and installed at the National Supercomputing Center in Wuxi (a city in Jiangsu province in China).[32] It was named as the fastest supercomputer at the International Supercomputing Conference in July 2016. With processing capacity of 125.436 PFlops, the Sunway TaihuLight is the first supercomputer in the world achieving speeds in excess of 100 PFlops.

As shown in Figure 1, the system architecture of the Sunway TaihuLight is mainly composed of the High-speed Computing System and the Auxiliary Computing System. The two computing systems are connected with the network system. Besides, there are Online Storage System (corresponding with the High-speed Computing System, 10 PB, 32 GB/s bandwidth) and Near-line Storage System (corresponding with the Auxiliary Computing System, 10 PB, 288 GB/s bandwidth). These four parts, adding numerous software systems, such as many-core software, the parallel operating systems, and parallel languages, form the entire system architecture.

In particular, the complete computing system has forty cabinets; a cabinet includes four super nodes, and each super node consists of 256 compute nodes.[32] A compute node corresponds to a SW26010 processor, and it is the primary element of the computing system.

The entire network system includes three layers, ie, the resource-sharing network, the super node network, and the central switching network. More specifically, the central switching network takes care of the connections and communications among super nodes. The super node network supports the communications among all the compute nodes in each super node (256 compute nodes). In addition, the resource-sharing network is responsible for enabling super nodes to access to the sharing resource.

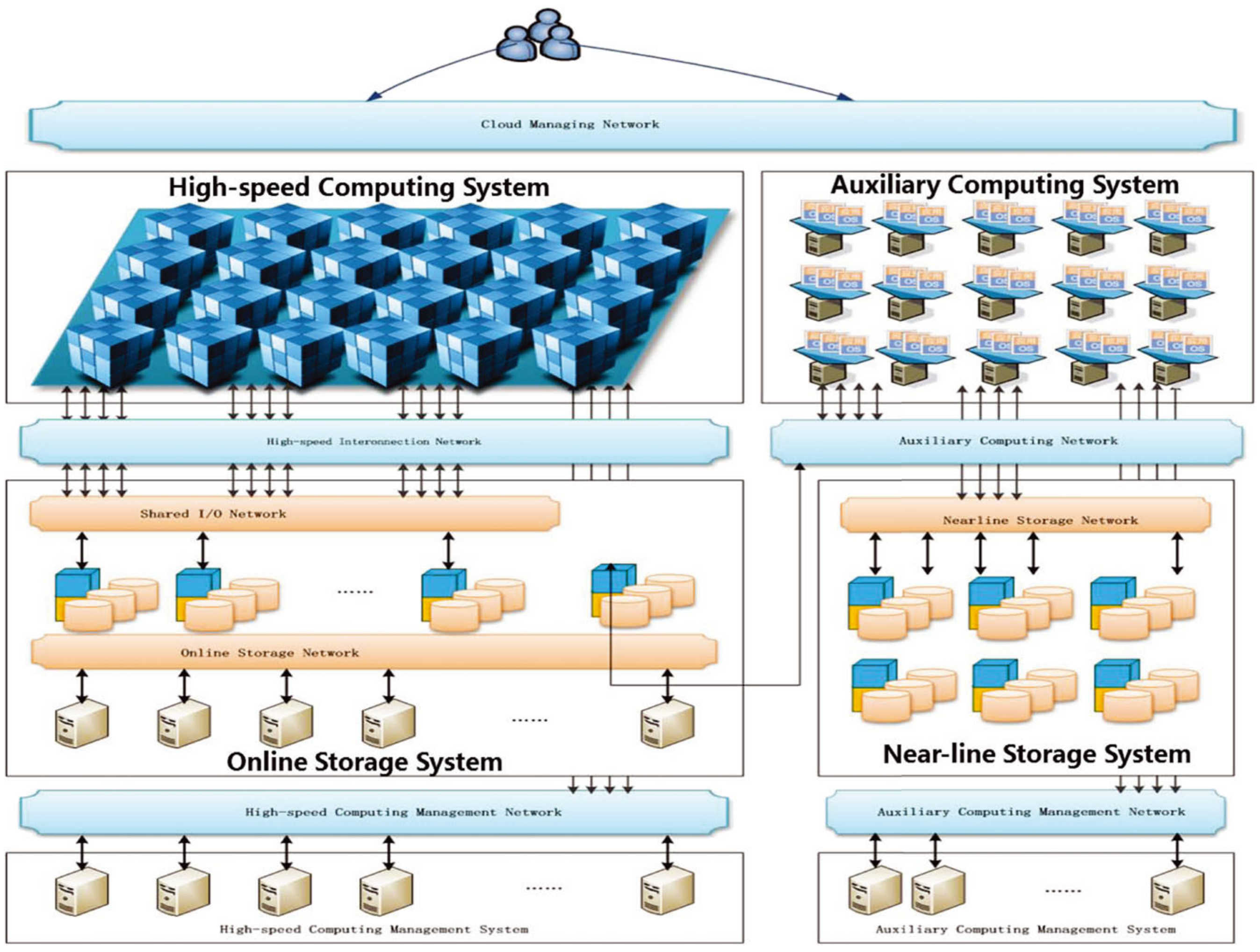
Cloud Managing Network
High-speed Computing System
Auxiliary Computing System
High-speed Interconnection Network
Auxiliary Computing Network
Shared I/O Network
Nearline Storage Network
Online Storage Network
Online Storage System
Near-line Storage System
High-speed Computing Management Network
Auxiliary Computing Management Network
High-speed Computing Management System
Auxiliary Computing Management System

FIGURE 1 The Sunway TaihuLight system architecture

4.1 | The SW26010 processor

The SW26010 processor is the main element of the Sunway TaihuLight system and provides a tremendous amount of computing power. It is designed by the Shanghai High Performance IC Design Center. The heterogeneous many-core processor chip makes a significant breakthrough in China-made chips.

As presented in Figure 2, a SW26010 processor consists of four Core Groups (CGs). Each of CGs contains a Management Processing Element (MPE) core, a Computing Processing Element (CPE) cluster with 64 CPE cores (arranged in an 8×8 array), the interface for the CGs, and a Memory Controller (MC).[32] The MC provides the interconnection between the memory space and MPE/CPE cores of the CG. The four CGs of a processor achieve memory sharing and communication through the network on chip (NoC). In addition, the system interface (SI) of a processor develops the interface between other outside devices.

Each MPE core has a dual pipeline with performance of 8 flops per cycle per pipeline, and each CPE core has a single floating point pipeline with performance of 8 flops per cycle per pipeline.[32] Both MPE and CPE cores have a frequency of 1.45 GHz. Therefore, each SW26010 processor has a peak performance of about 3 TFlop/s ($4\ core \times 16\ Flops/cycle \times 1.45\ GHz + 256\ cores \times 8\ Flops/cycle \times 1.45\ GHz$).

The SW26010 processor also supports a more efficient storage structure. Each MPE core is equipped with a 32-KB L1 cache for storing instruction, another 32-KB L1 cache used for data storage, and a 256-KB L2 cache for storing both instruction and data.[32] Unlike MPEs, each CPE core contains a 16-KB L1 cache for storing instruction and a 64-KB Scratch Pad Memory (SPM) for storing data.[32] Particularly, the SPM has been studied a lot in multiprocessor system-on-chip for embedded systems to reduce power consumption and/or improve efficiency, but it is the first time for SPM to be applied on a supercomputer. The SPM is mostly proper for storing temporary data that would not access to the main memory frequently.

It is well known to us that a SW26010 processor represents a compute node. Each SW26010 node is composed of four CGs. More specifically, four MPE cores, four CPE clusters with 256 CPE cores, four Memory Controllers (MC), a Network on Chip (NoC), and a System Interface (SI) comprise a SW26010 node. Each CG corresponds to an MPI process, and each CPE corresponds to a thread. In addition, the amount of memory that each CG can access is 8 GB. In total, the Sunway TaihuLight system has 40,960 SW26010 nodes with 1.31 PB memory and 10,649,600 cores.[32]

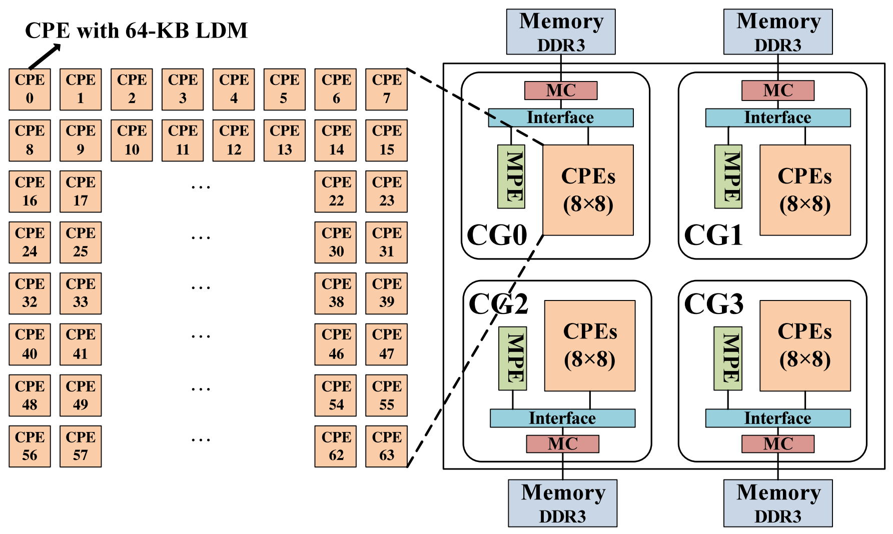

FIGURE 2 The SW26010 processor architecture

4.2 | Programming languages and environments

The programming languages and environments supported by the Sunway TaihuLight system are as follows.

(i) Basic programming languages: The system supports several main basic programming languages, including C, C++, and Fortran.
(ii) Parallel programming languages/interfaces: The system supports international parallel programming standards, such as MPI 3.0, OpenMP 3.1, Pthreads, and OpenACC 2.0. Meanwhile, the system supports a programming interface of the self-designed accelerated thread library.
(iii) User environment: The usage environment of users is provided that assembles editing, compiling, debugging, and performance monitoring with graphical interface.
(iv) Basic programming environment: The system provides basic programming languages, MPE-CPE heterogeneous programming, basic function libraries, and auto-vectorization/parallelization. The system also provides abundant and efficient functions of compiling optimization.

4.3 | Parallel modes

According to the architecture of the Sunway TaihuLight with SW26010 processors, the CPE cluster of each CG performs the role of providing parallel computing capability. To fully utilize the accelerated parallel superiority of the CPE cluster and obtain better performance on the Sunway TaihuLight, it is necessary to coordinate MPE core and CPE cores. There are four parallel modes provided to collaborate with the MPE and CPEs of each CG on the Sunway TaihuLight as follows.

(i) The accelerative parallel mode: It is the most common mode. As shown in Figure 3, the MPE is in charge of communications and some computations that cannot be implemented in parallel. Nevertheless, when CPEs are busy in parallel computing, the MPEs are waiting.
(ii) The collaborative parallel mode: As shown in Figure 4, in this mode, the MPE collaborates with CPEs as peers on parallel computations. They load workloads, respectively, according to their computing capabilities.

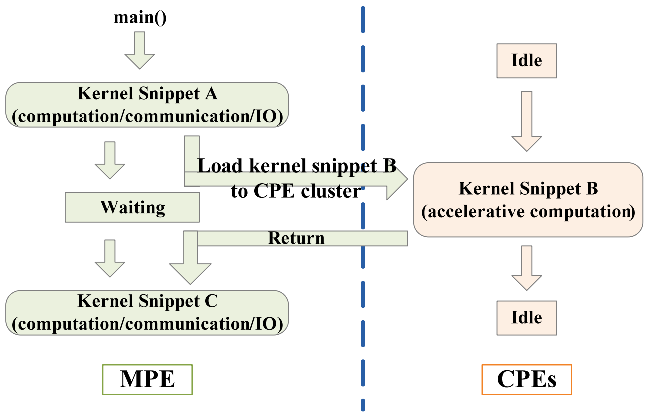
main()
Kernel Snippet A (computation/communication/IO)
Load kernel snippet B to CPE cluster
Waiting
Return
Kernel Snippet C (computation/communication/IO)
MPE
Idle
Kernel Snippet B (accelerative computation)
Idle
CPEs

FIGURE 3 The accelerative parallel mode

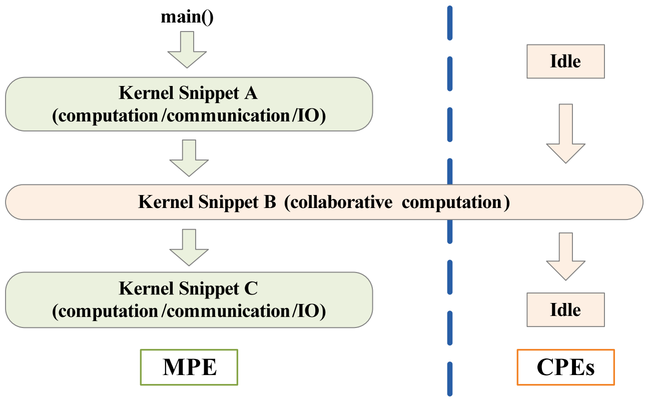
main()
Kernel Snippet A (computation /communication /IO)
Idle
Kernel Snippet B (collaborative computation)
Kernel Snippet C (computation /communication /IO)
Idle
MPE
CPEs

FIGURE 4 The collaborative parallel mode

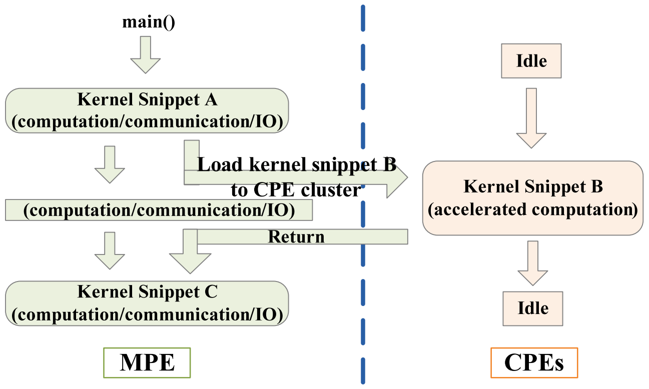
main()
Kernel Snippet A (computation/communication/IO)
Idle
Load kernel snippet B to CPE cluster
Kernel Snippet B (accelerated computation)
(computation/communication/IO)
Return
Kernel Snippet C (computation/communication/IO)
Idle
MPE
CPEs

FIGURE 5 The asynchronous parallel mode

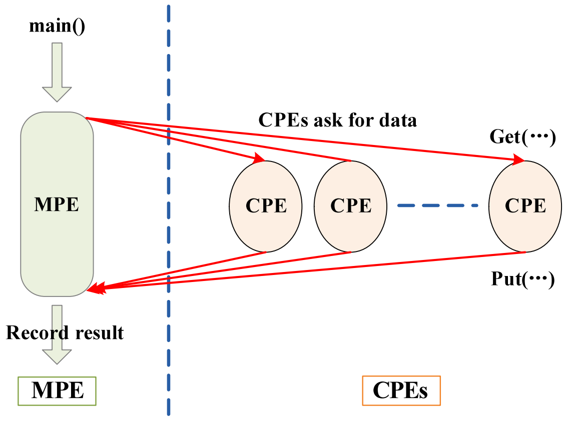

FIGURE 6 The dynamic parallel mode

(iii) The asynchronous parallel mode: As shown in Figure 5, in this mode, the MPE is completing other operations (such as other computations, communications, or I/O) while CPEs are accelerating calculation. This mode tremendously improves the efficiency of the collaboration between the MPE and CPEs.

(iv) The dynamic parallel mode: Figure 6 presents the mechanism of this mode. The MPE is responsible for assigning tasks to CPEs, while CPEs are in charge of receiving, finishing these tasks, and returning the results to the MPE.

5 | DATA PROTECTING MODEL ON THE SUNWAY TAIHULIGHT

5.1 | Data protecting model

To gain high efficiency of encryption/decryption for large-scale data and guarantee the security and integrity of storage and transmission of large-scale data, we propose a fast and secure data protecting model using the parallel AES algorithm and SHA3-256 on the Sunway TaihuLight. There are two parts of the data protecting model, ie, the encryption part and the decryption part.

5.1.1 | The encryption part

For the encryption part, there are three steps to encrypt data.

Step 1: plaintext padding. The plaintext is padded to ensure that the total length of padded plaintext, and its hash value is an integral multiple of 16 bytes. The hash value of SHA3-256 has a fixed size of 32 bytes. The plaintext is padded with bytes. All the values of the added bytes are the same, ie, the total number of added bytes. Particularly, if the length of the plaintext is an integral multiple of 16 bytes and the last byte of the plaintext is 1, it is difficult to figure out whether it has been padded or not. Therefore, although the plaintext is an integral multiple of 16 bytes, it is necessary to pad an extra 16 bytes and the value of added bytes is 16.

Step 2: SHA3-256 processing. SHA3-256 processes the padded plaintext into a hash value H with a fixed size of 32 bytes. H is added behind the padded plaintext. The total length of the padded plaintext with a hash value H is an integral multiple of 16 bytes.

Step 3: parallel AES encryption. The length of the padded plaintext with H is an integral multiple of 16 bytes, which meets the criteria of AES encryption/decryption. Therefore, this step encrypts the padded plaintext with H into ciphertext using the parallel AES algorithm.

5.1.2 | The decryption part

The decryption part is the reverse process of the encryption part, and there are also three steps to decrypt data.

Step 1: parallel AES decryption. The ciphertext is first decrypted using the parallel AES algorithm. The decryption results contain two parts, ie, the padded plaintext and the hash value H.

Step 2: detecting data corruption. The decryption result is separated into the padded plaintext and the hash value H. The last 32 bytes is the hash value, and the rest part is the padded plaintext. Then, the SHA3-256 processes the padded plaintext into a new hash value H'. According to the characters of the SHA described in Section 3.2, if H' is the same with H, it proves that the padded plaintext is uncorrupted and keeps confidential during the storage or transmission. Otherwise, it means that the padded plaintext has been corrupted and altered.

Step 3: getting the plaintext. If the padded plaintext is proved to be uncorrupted and confidential during the storage or transmission in **Step 2**, the plaintext is taken out of the padded plaintext.

5.2 | A fine-grained software design

As described in Section 5.1, there are both three steps in the encryption and decryption parts of the data protecting model. For the AES encryption and decryption, a fixed 16-byte plaintext block is processed each time. It is clear that there is no data dependency among these data blocks and these blocks can be accessed continuously, which means that the AES algorithm is naturally amenable to parallelism. Therefore, **Step 3** of the encryption part and **Step 1** of the decryption part are processed on CPEs of each CG. The rest steps cannot be implemented in parallel, so that they are processed on the MPE of each CG. In addition, the three steps in the encryption/decryption part of the data protecting model must be done in order.

Therefore, we adopt the combination of the accelerative parallel mode and the dynamic parallel mode, described in Section 4.3, to implement the data protecting model on the Sunway TaihuLight according to its workflow. For the encryption part, **Step 1** and **Step 2** are executed on the MPE and get the padded plaintext with its hash value. **Step 3** is executed on CPEs in parallel to AES encrypt the padded plaintext with its hash value into ciphertext. For the decryption part, **Step 1** is executed on CPEs in parallel to AES decrypt the ciphertext into the padded plaintext with a hash value. **Step 2** and **Step 3** are executed on the MPE to detect data corruption and get the original plaintext if there is no data corruption. Particularly, the MPE should wait when CPEs are running to ensure all the steps are done in order on each CG.

According to the particular computing architecture of Sunway TaihuLight, each CG corresponds to an MPI process and each CPE core in a CG corresponds to a thread. It is important to fully exploit the two-level parallelism (the CG-level and the CPE-level) of the Sunway to gain significant performance improvement for the data protecting model. In addition, the memory hierarchy of the Sunway TaihuLight is also special. Each CPE is equipped with a 64-KB SPM instead of cache, which requires further fine-grained data blocking to properly manage data for the data protecting model on the Sunway.

Therefore, we apply a fine-grained method for three-level data blocking, shown in Figures 7 and 8, to implement the efficient and large-scale data protecting model based on the Sunway architecture.

First Level: blocking the plaintext/ciphertext for CGs. Define the number of CGs used on the Sunway TaihuLight be N_g. The plaintext/ciphertext is evenly divided into N_g CG-level blocks. Each CG processes a CG-level block.

Second Level: further blocking for CPEs. Define the number of CPEs used in each CG be N_e. The CG-level block on each CG is further evenly divided into N_e CPE-level blocks. Each CPE processes a CPE-level block. Particularly, for the encryption part, the CG-level block on each CG should be padded according to **Step 1** and **Step 2** before being blocked for CPEs.

Third Level: further blocking to fit in the 64-KB SPM of each CPE. The storage capacity of SPM on each CPE is 64 KB, which means that each CPE can only get and process M-byte data each time to complete parallel AES encryption and decryption, where M is less than 64 KB. Therefore, each CPE-level block is further evenly divided into several fine-grained blocks to ensure the size of each fine-grained block fit in the SPM. Each CPE loads a fine-grained block, executes AES encryption/decryption, and returns the result back in each round until the complete CPE-level block data on the CPE is encrypted/decrypted through several rounds.

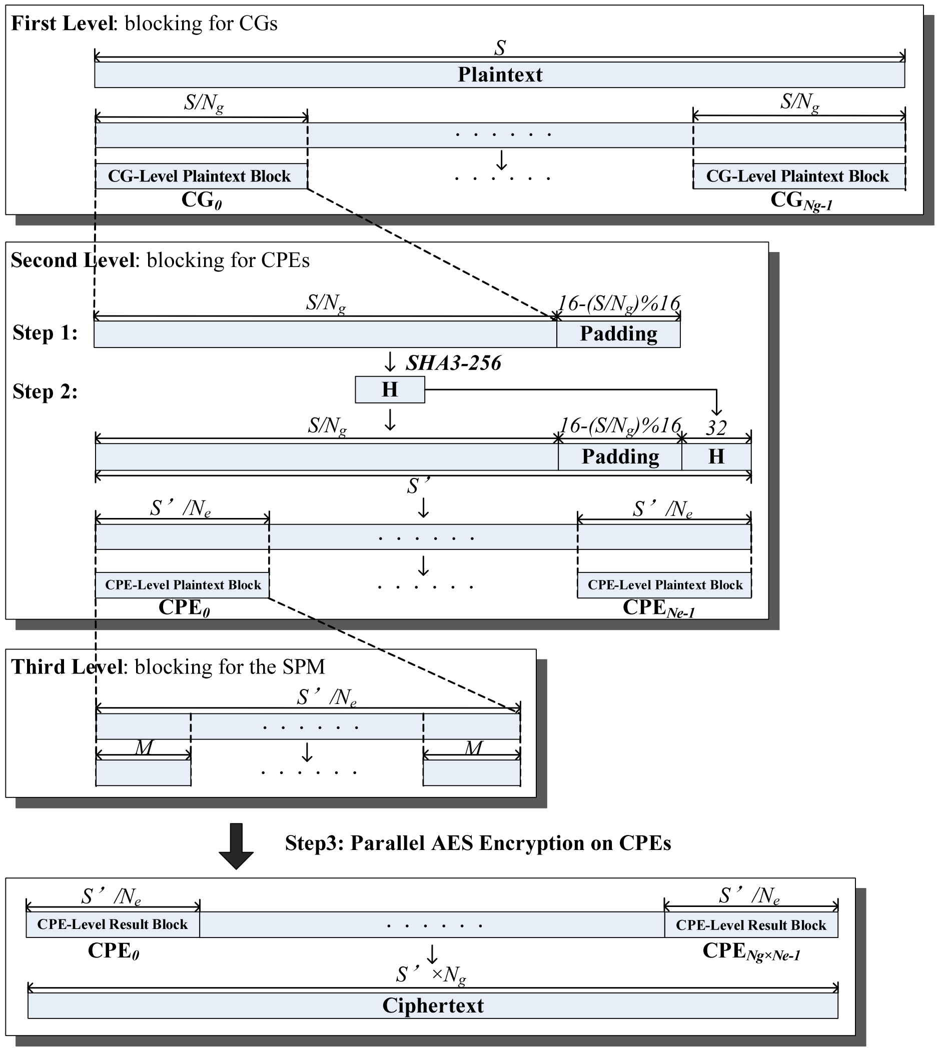

FIGURE 7 The fine-grained data blocking for the encryption part of data protecting model on the Sunway TaihuLight

Algorithms 1 to 3 demonstrate the data protecting model based on the SHA3-256 and the parallel AES algorithm on the Sunway TaihuLight.

5.3 | Performance analysis

In this section, we analyze the performance of the parallel AES algorithm. The following theorem gives the speedup of our parallel implementation.

Theorem 1. *The speedup of the parallel AES algorithm on the Sunway is*

$$Speedup = \frac{T}{T/(N_g \times N_e) + P \times N}. \tag{1}$$

Proof. In our implementation of parallel AES algorithm on the Sunway, there are two major parts including computation part and communication part. The computation part represents the part of executing the parallel AES encryption/decryption on the Sunway. The communication part mainly represents the memory access part of receiving and sending data from MPE.

Assuming the cost of computation part is T when the AES algorithm is serially run on only one core, the cost is $T/(N_g \times N_e)$ when the AES algorithm is run on $N_g \times N_e$ CPE cores in parallel. Assuming the communication cost of the parallel AES algorithm in each round is P, the total cost is $P \times N$ for N rounds. Therefore, the sequential execution time of the AES algorithm is T.

The parallel execution time includes two components, ie, computation time and communication time. The computation time is $T/(N_g \times N_e)$, and the communication time is $P \times N$. Thus, the the parallel execution time of the AES algorithm is $T/(N_g \times N_e) + P \times N$.

Accordingly, we can calculate the speedup of the parallel AES algorithm on the Sunway as follows:

$$\begin{aligned} Speedup &= \frac{SerialTime}{ParallelTime} \\ &= \frac{T}{T/(N_g \times N_e) + P \times N}. \end{aligned}$$

This completes the proof of the theorem. □

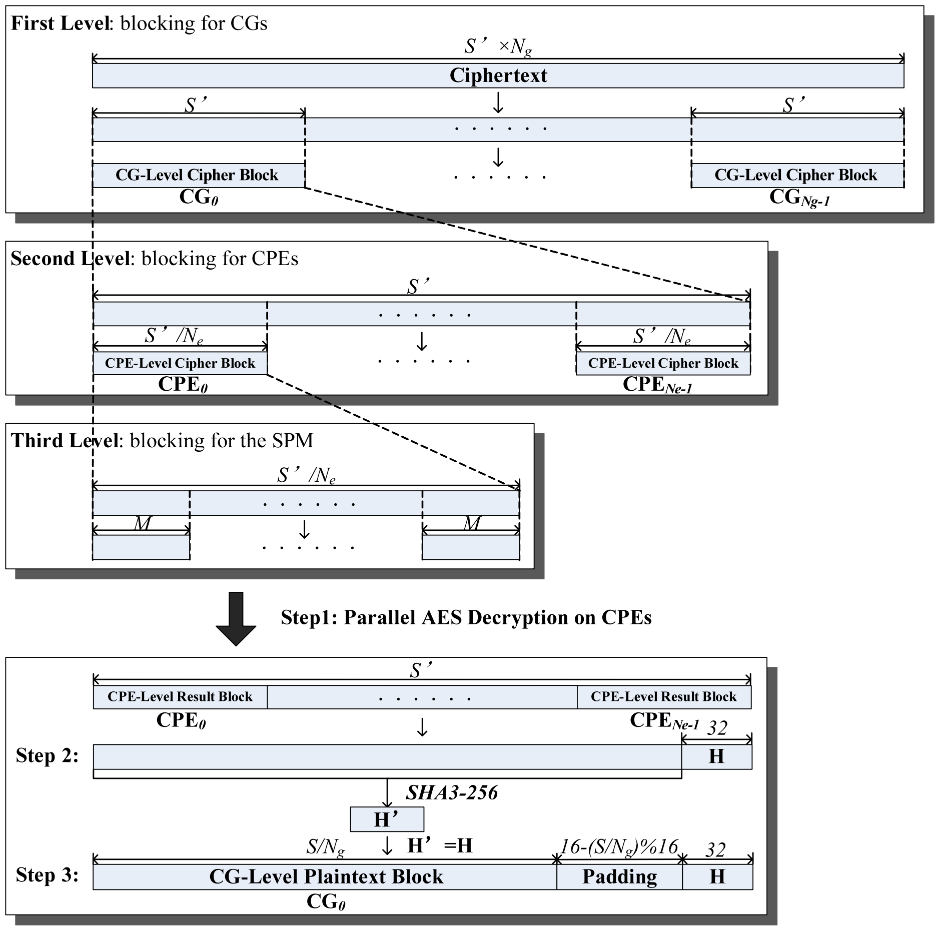

FIGURE 8 The fine-grained data blocking for the decryption part of data protecting model on the Sunway TaihuLight

Algorithm 1 The encryption part of data protecting model on the MPE of each CG

```
Require: the number of CGs used, $N_g$;
    the number of CPEs applied in each CG, $N_e$;
    the plaintext data, plaintext;
    the size of plaintext data, $S$.
Ensure: ciphertext.
 1: Initiate $l, S', H$;
 2: $l \leftarrow 16 - (S/N_g)\%16$;
 3: $S \leftarrow S/N_g + l + 32$;
 4: Initiate $\boldsymbol{PT}[S'], \boldsymbol{CT}[S']$;
 5: Load a block of plaintext with the size of $S/N_g$ into PT;
 6: Padding(PT, l, l); //pad the plaintext with l bytes whose value is l
 7: $H \leftarrow$ SHA3-256(PT); //get the hash value
 8: Padding(PT, H, 32); //add the hash value behind the padded plaintext
 9: athread_init();
10: athread_set_num_threads($N_e$);
11: athread_spawn(PT, CT, $N_e$); //invoke Algorithm 3 to AES encrypt PT in parallel on the CPE
    cluster
12: athread_halt();
13: return CT. //the ciphertext is stored in CT
```

It can be seen from Equation (1) that the speedup will improve as the number of cores increases. To achieve better performance, we can ensure that there is no core being in idle state in each round. Besides, we can also get higher speedup of CPE cluster by reducing the cost of memory access. The optimization of memory access will be presented in the following section.

The following theorem gives the throughput of our parallel AES algorithm on the Sunway.

Algorithm 2 The decryption part of data protecting model on the MPE of each CG

```
Require: the number of CGs used, $N_g$;
    the number of CPEs applied in each CG, $N_e$;
    the ciphertext data, ciphertext;
    the size of ciphertext data, $S' \times N_g$.
Ensure: plaintext.
 1: Initiate PT[S'], CT[S'];
 2: Load a block of ciphertext with the size of S' into CT;
 3: athread_init();
 4: athread_set_num_threads($N_e$);
 5: athread_spawn(CT, PT, $N_e$); //invoke Algorithm 3 to AES decrypt CT in parallel on the CPE
    cluster
 6: athread_halt();
 7: Initiate $H, H', l$;
 8: unPadding(PT, H, 32); //separate the result of decryption into the padded plaintext and the
    hash value H
 9: $H' \leftarrow$ SHA3-256(PT); //get the hash value of the padded plaintext //data corruption detection
10: if $H = H'$ then
11:     unPadding(PT, l, l); //the plaintext is safe and take it out of the padded plaintext
12: end if
13: return PT. //the plaintext is stored in PT
```

Algorithm 3 Parallel AES algorithm on the CPE cluster of each CG

```
Require: the number of CPEs applied in each CG, $N_e$;
    the identification number of each CPE in cluster, core_id;
    the plaintext/ciphertext data stored in the MPE, PT[S'] or CT[S'];
    the size of data transferred to each CPE in each round, M.
Ensure: the ciphertext/plaintext data, CT[S'] or PT[S'].
 1: Initiate count, block, i, j;
 2: $count \leftarrow N_e \times M$; //the total problem size of the CPE cluster for each round
 3: $block \leftarrow M/16$; //the number of 16-byte blocks in each round
 4: Allocate SPM memory for loading PT[S'] or CT[S'] into Input[M] each round;
 5: //cycles the process of encrypting/decrypting M-length data
 6: for $j \leftarrow 0; j < block; j \leftarrow j + 1$ do
 7:     Load M-length plaintext/ciphertext from PT[core_id × M] or CT[core_id × M] into Input;
 8:     for $j \leftarrow 0; j < block; j \leftarrow j + 1$ do
 9:         Execute AES encryption/decryption on a 16-byte block from Input[$j * 16$];
10:     end for
11:     Send M-length ciphertext/plaintext from Input to CT[core_id × M] or PT[core_id × M];
12: end for
```

Theorem 2. *The throughput of the parallel AES algorithm is*

$$Throughput = \frac{M}{T}. \tag{2}$$

Proof. Assuming the input size of the parallel AES encryption is M bytes, and the execution time is T. Therefore, the throughput of the parallel AES algorithm can be calculated as follows:

$$Throughput = \frac{InputSize}{ExecutionTime} = \frac{M}{T}.$$

This completes the proof of the theorem. □

5.4 | Performance optimization

5.4.1 | Memory access optimization

As has been mentioned, each CPE core is equipped with a user-controlled SPM rather than caches. Compared with the hardware design of auto-cache, using SPM is low power consumption and more initiative for programmers. However, we find that it brings more programming complexity, and all memory access operations should be finished manually. Therefore, it costs too much time for SPM to frequently access the data stored in main memory, which seriously affects the parallel processing efficiency of CPE cluster.

As a result, the compiling system of the Sunway TaihuLight provides the DMA intrinsic to achieve high speed and efficiency to transmit data between the main memory of MPE and the SPM in CPE. The DMA commands can be issued only by CPEs. When the data in main memory is aligned with 128 B and the amount of data transmitted is a multiple of 128 B, the DMA intrinsic can reach peak performance. There are five DMA modes to transmit data as follows.

(i) The single-CPE DMA mode: In this mode, each SPM exchanges data with main memory individually.
(ii) The broadcast DMA mode: In this mode, data in the main memory is broadcasted to CPEs.
(iii) The single-row DMA mode: In this mode, each row of SPMs (containing 8 SPMs) transfers data with main memory. Data is recurrently distributed on 8 SPMs in each row.
(iv) The row-broadcast DMA mode: In this mode, data in the main memory is broadcasted to several rows of SPMs. Data is recurrently distributed in each row.
(v) The array DMA mode: In this mode, the array of SPMs (containing 64 SPMs) in a CPE cluster exchanges data with main memory. Data is recurrently distributed on 64 SPMs in each array.

In this paper, we adopt the single-CPE DMA mode to optimize memory access. In particular, for each round, each SPM receives a fixed-length plaintext block from main memory using DMA first. Next, each CPE core will, respectively, complete the AES encryption. Finally, the CPE cores will send the ciphertext blocks back to main memory also using DMA. The characteristic of our implementation process is suitable for adopting the single-CPE DMA mode. Besides, the amount of data transferred must be a multiple of 128 B.

According to Equation (1), since the cost of memory access in each round can be reduced by using DMA intrinsic, we can get higher speedup of CPE cluster and obtain optimal performance using DMA intrinsic.

5.4.2 | The Double-buffering mechanism

Even though the cost of memory access can be reduced by the DMA intrinsic, it still greatly restricts the parallel efficiency and scalability of the algorithm. As we described before, in each AES encryption round, there are 2 DMA operations, ie, a DMA read operation to get data before encryption and a DMA write operation to send ciphertext after encryption. With the amount of plaintext increasing, the number of AES encryption rounds increases, and the number of DMA read and write operations increases as well. In order to obtain more optimized results, we propose and adopt the double-buffering mechanism and overlap the cost of computation and communication.

The process of double-buffering mechanism is as follows. Since there are several rounds of DMA read and write operations, the SPM requires a memory space to store data buffered by both MPE and CPE. The size of the memory space in SPM is twice as much as the amount of transferred data. Except the first round of receiving data and the final round of sending data, CPEs not only complete computations in this round but also receive (send) data for next round (last round). The details of our optimized parallel AES algorithm applying the double-buffering mechanism are presented in Algorithm 4.

5.5 | Optimized performance analysis

The following theorem gives the speedup of our optimized parallel implementation.

Theorem 3. *With the optimization strategies proposed above, the speedup of the optimized parallel AES algorithm on the Sunway is*

$$Speedup = \frac{T}{max[T/(N_g \times N_e), P \times (N-1)] + P}. \tag{3}$$

Proof. Assuming the cost of computation part is T when the AES algorithm is serially run on only one core, the cost is $T/(N_g \times N_e)$ when the AES algorithm is run on $(N_g \times N_e)$ CPEs in parallel. With the optimization strategies, the cost of memory access contains overlapping part and non-overlapping part. Assuming the communication cost of parallel AES encryption is P in each round, the cost of non-overlapping part includes the cost of data transmission in the first and the last round, which is P. Moreover, the cost of overlapping part is $P \times (N-1)$.

Therefore, the sequential execution time of the AES algorithm is T. The parallel execution time includes two components, ie, the overlapping time and non-overlapping time. The overlapping time is $max[T/(N_g \times N_e), P \times (N-1)]$, and the non-overlapping time is P. Thus, the the parallel execution time of the AES algorithm is $max[T/(N_g \times N_e), P \times (N-1)] + P$.

Accordingly, we can calculate the speedup of the optimized parallel AES algorithm on the Sunway as follows:

$$\begin{aligned} Speedup &= \frac{SerialTime}{ParallelTime} \\ &= \frac{T}{max[T/(N_g \times N_e), P \times (N-1)] + P}. \end{aligned}$$

The theorem is proved. □

Algorithm 4 Optimized parallel AES algorithm on a CPE cluster of each CG

```
Require the number of CPEs applied in each CG, $N_e$;
    the identification number of each CPE in cluster, $core\_id$;
    the plaintext/ciphertext data stored in the MPE, $\boldsymbol{PT}[S']$ or $\boldsymbol{CT}[S']$;
    the size of data transferred to each CPE in each round, $M$.
Ensure the ciphertext/plaintext data, $\boldsymbol{CT}[S']$ or $\boldsymbol{PT}[S']$.
 1: Initiate $count, block, flag, i, j$;
 2: $count \leftarrow N_e \times M$;
 3: $block \leftarrow M/16$;
 4: Allocate double the SPM memory needed for loading $\boldsymbol{PT}[S']$ or $\boldsymbol{CT}[S']$ into $\boldsymbol{Input}[2][M]$ each time;
 5: $flag \leftarrow 0$;
 6: $DMA\_get(\boldsymbol{PT}[core\_id \times M], \boldsymbol{Input}[0][0], M)$ or $DMA\_get(\boldsymbol{CT}[core\_id \times M], \boldsymbol{Input}[0][0], M)$;
    //get $M$-length data for the first round
 7: for $i \leftarrow 0; i < length; i \leftarrow i + count$ do
 8:     $flag \leftarrow flag+1$;
 9:     //get $M$-length data for next round
10:     $DMA\_get(\boldsymbol{PT}[core\_id \times M], \boldsymbol{Input}[flag][0], M)$ or $DMA\_get(\boldsymbol{CT}[core\_id \times M], \boldsymbol{Input}[flag][0], M)$;
11:     for $j \leftarrow 0; j < block; j \leftarrow j + 1$ do
12:         Execute AES encryption/decryption on a 16-byte block from $\boldsymbol{Input}[(flag-1)\%2][j * 16]$;
13:     end for
14:     //send $M$-length ciphertext/plaintext back for this round
15:     $DMA\_put(\boldsymbol{Input}[(flag-1)\%2][0], \boldsymbol{CT}[core\_id \times M], M)$ or $DMA\_put(\boldsymbol{Input}[(flag-1)\%2][0], \boldsymbol{PT}[core\_id \times M], M)$;
16: end for
17: for $j \leftarrow 0; j < block; j \leftarrow j + 1$ do
18:     Execute AES encryption/decryption on a 16-byte block from $\boldsymbol{Input}[flag\%2][j * 16]$ for the last round;
19: end for
20: //send $M$-length ciphertext/plaintext back for the last round
21: $DMA\_put(\boldsymbol{Input}[flag\%2][0], \boldsymbol{CT}[core\_id \times M], M)$or $DMA\_put(\boldsymbol{Input}[flag\%2][0], \boldsymbol{PT}[core\_id \times M], M)$;
```

We would like to mention that the speedup in Equation (3) is greater than that in Equation (1). In our optimized parallel AES encryption/decyption process; the cost of computation is far greater than the cost of communication. Equation (3) indicates that, when *P* is barely measurable and the cost of computation is overlapped completely, the speedup on the Sunway is nearly $(N_g \times N_e)$. In other words, we adequately utilize the advantage of the CPEs cluster and gain better optimization results with the DMA intrinsic and the double-buffering mechanism.

6 | EXPERIMENTAL RESULTS

6.1 | Experimental environment

We perform all the experiments on the Sunway TaihuLight. Table 1 shows the configuration of the Sunway TaihuLight system. Single-CG experiments are ran on a CG of the Sunway, and multi-CG experiments are ran on multiple CGs of the Sunway. All the programs are coded in C language, and the multi-CG parallel implementation is programmed by MPI 3.0.

TABLE 1 The Sunway TaihuLight System Configuration

CPU	SW26010 processor
Processor Node	4 CGs (4 MPEs and 256 CPEs)
OS	Sunway Raise OS 2.0.5 (based on Linux)
Instruction set	Sunway-64 Instruction Set
Compile language	C, C++, Fortran
Parallel programming interface	MPI 3.0, OpenMP 3.1, OpenACC 2.0

Since the performance of the encryption part of the data protecting model is similar to the decryption part, we only present the encryption performance. We perform these experiments with different sizes of plaintext. The plaintext was randomly generated by MPEs. In our experiments, we set that the MPE in each CG sends a CPE-level data block with the size of 1 KB to each CPE in each process round, where 1 KB refers to 1024 bytes. We present the execution time and throughput of the data protecting model to show its efficiency on the Sunway TaihuLight.

6.2 | Single-CG performance results

In order to prove the efficiency of our soft design and optimization strategies, we perform several experiments of encrypting plaintext with different sizes using the proposed data protecting model on a single CG.

Figure 9 presents the throughput of encrypting 1-GB plaintext using the data protecting model on a CG. As shown in Figure 9, the data protecting model achieves the optimized performance with all concurrency on the CG. Since the SHA3-256 is executed on the MPE of the CG and the AES algorithm is executed on the CPE cluster of the CG, the execution time of SHA3 will be unchanged with the different number of CPEs used in the CG. The performance improvement of data protecting model shown in Figure 9 demonstrates the parallel execution time of the AES algorithm is improved with the increasing number of CPEs used in the CG.

Figure 10 presents the throughput of data protecting model varying the input size on a CG. The lines in Figure 10 respectively show the results varying the number of CPEs on the CG. Each line is nearly parallel to the x-axis and the throughput keeps constant varying the input size, which demonstrates that the execution time of the data protecting model increases linearly with the input size. The data protecting model achieves the highest throughput, around 34 MB/s, when all the 64 CPEs are used in the CG.

In particular, the parallel AES algorithm guarantees the encryption efficiency for the data protecting model. Therefore, we execute the optimized parallel AES algorithm on a CG and present its throughput on the CG with different number of CPEs used. In addition, we execute the serial AES algorithm on the MPE of a CG and the parallel AES algorithm as well as optimized parallel AES algorithm on a CG with all concurrency to compare the performance among the serial, parallel, and optimized parallel implementation on a CG.

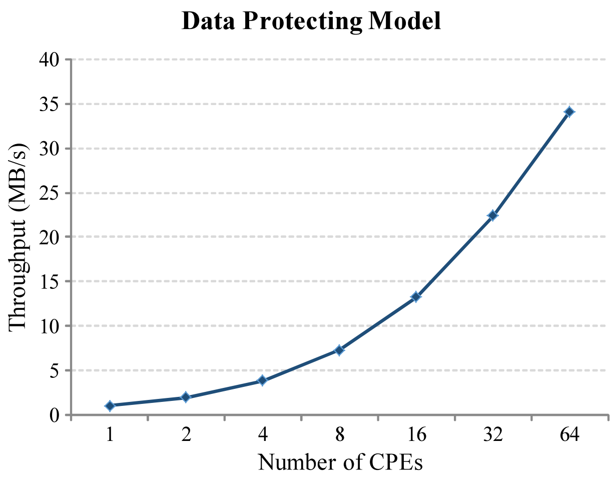

FIGURE 9 The throughput (MB/s) of encrypting 1-GB plaintext using the data protecting model on a CG

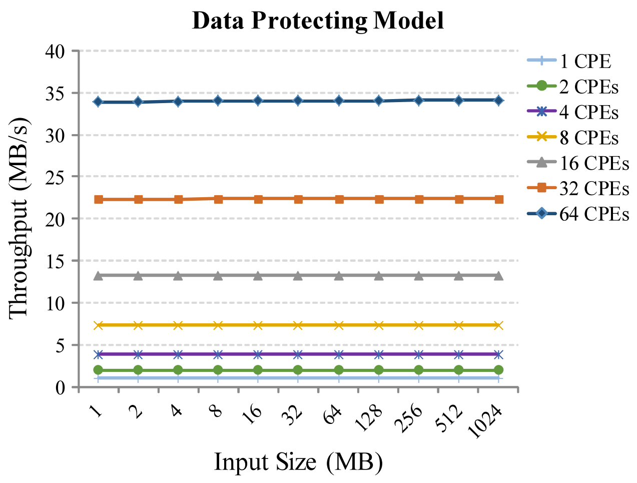

FIGURE 10 The throughput (MB/s) of data protecting model varying the input size on a CG

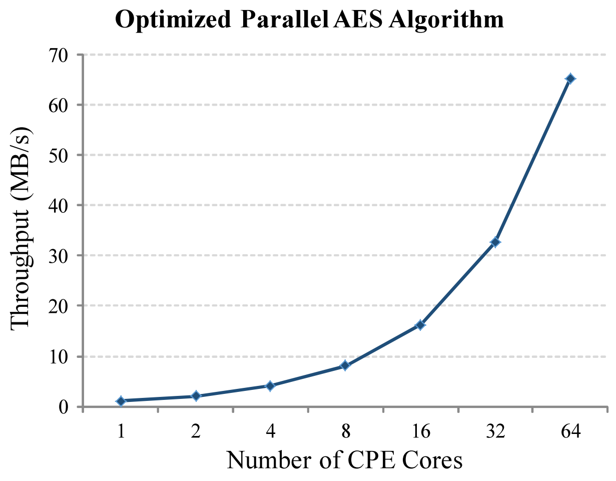

FIGURE 11 The throughput (MB/s) of encrypting 1-GB input data using the optimized parallel AES algorithm on a CG

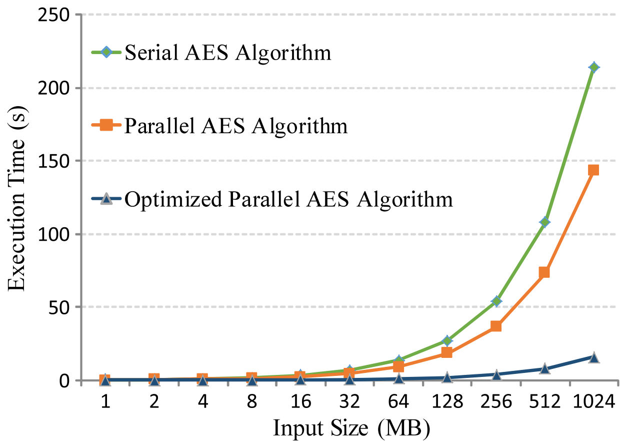

FIGURE 12 The comparison of execution time of serial, parallel, and optimized parallel AES algorithm on a CG

We encrypt 1-GB data using the optimized parallel AES algorithm on a CG varying the number of applied CPEs. As shown in Figure 11, the throughput of parallel encryption gradually increases as the number of CPE cores used in the CG increasing. In other words, we can achieve the possible maximum speed of the parallel AES algorithm and the data protecting model with all concurrency on a CG.

Figure 12 presents the comparison among the serial, parallel, and optimized parallel AES algorithm on a CG. As we can see from Figure 12, the parallel implementation of AES spends evidently less time than the serial implementation. What's more, the optimized parallel AES algorithm behaves the best performance with optimization. It is obvious that we get high speedup ratio and parallel efficiency of the optimized parallel AES algorithm on a CG, and the effective optimization strategy is available.

Figure 13 shows the throughput of the optimized parallel AES algorithm using all CPE cores on a CG. With the input size increasing from 1 MB to 1 GB, there is almost no change on the trend line of parallel AES encryption throughput. It is illustrated that the execution time of the optimized parallel AES algorithm increases linearly with the input size. Besides, we note that we can achieve the throughput of about 65.18 MB/s on a CG.

6.3 | Multi-CG performance results

The Sunway TaihuLight supports MPI parallel programs, so we also perform scalability experiments of the data protecting model. We perform multi-node tests with MPI based on multiple CGs. In this paper, all the compute nodes used in experiments are responsible for the workloads with same size. That is, in *N*-CG experiments, the problem size of each CG is *M* MB, so the total problem size of the experiments is $N \times M$ MB.

Figure 14 presents the throughput of encrypting 1-GB plaintext using the data protecting model on multiple CGs. It is clear that the data protecting model achieves higher throughput and better performance with the number of CG increasing, which proves that the data protecting model has good scalability on the Sunway TaihuLight.

Figure 15 shows the throughput of the data protecting model varying the input size on 256, 512, and 1024 CGs, respectively. With the input size increasing gradually, the tend-lines of throughput create a very fast rising trend on multiple CGs. Especially, the throughput on 1024 CGs grows the fastest. The data protecting model achieves the throughput of about 35 GBytes/s (269.95 Gbits/s) on 1024 CGs, which proves that our data protecting model has high efficiency on the Sunway TaihuLight.

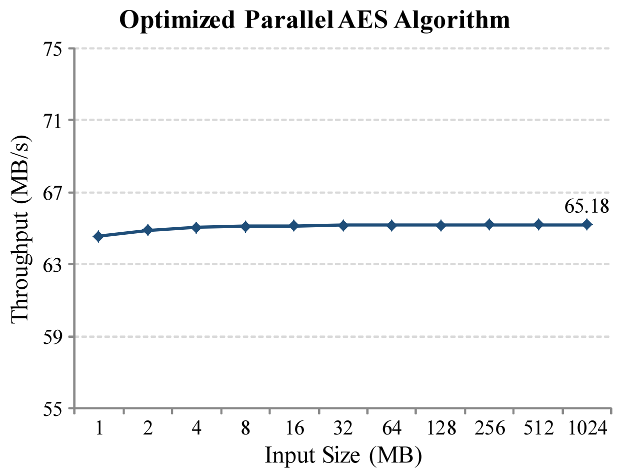

FIGURE 13 The throughput (MB/s) of the optimized parallel AES algorithm on input data with different sizes on a CG

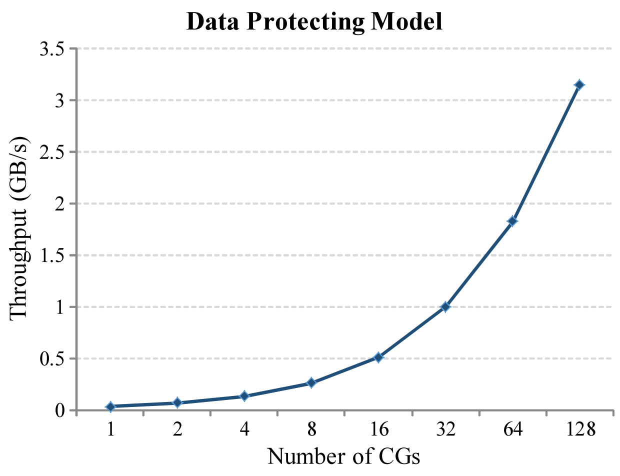

FIGURE 14 The throughput (GB/s) of encrypting 1-GB plaintext using the data protecting model on multiple CGs

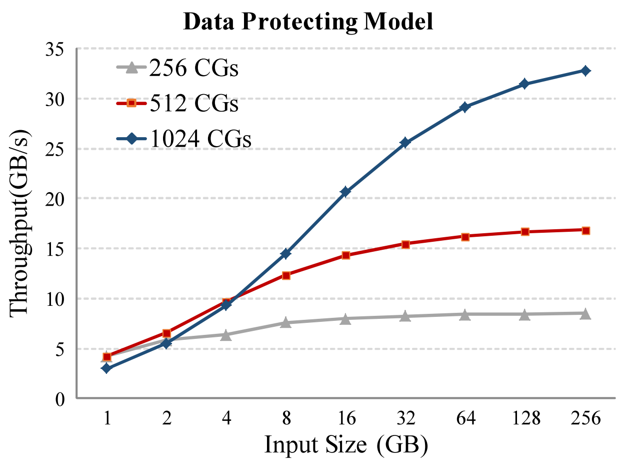

FIGURE 15 The throughput (GB/s) of the data protecting model varying the input size on multiple CGs

We also test the optimized parallel AES algorithm of the data protecting model on multiple CGs. Figure 16 demonstrates the throughput of encrypting 1-GB input data using the optimized parallel AES algorithm on multiple CGs. The changing tendency of the throughput has a sharp rise with the number of CGs increasing from 1 to 128. Accordingly, employing more CGs, the optimized parallel AES algorithm can get better performance on the Sunway TaihuLight system.

In Figure 17, we fix the problem size of each CG as 64 MB, 128 MB, 256 MB, 512 MB, and 1024 MB, and the trend-line of the execution time of the optimized parallel AES encryption in each case has little changes with the number of CGs increasing from 1 to 1024. If the problem size of each

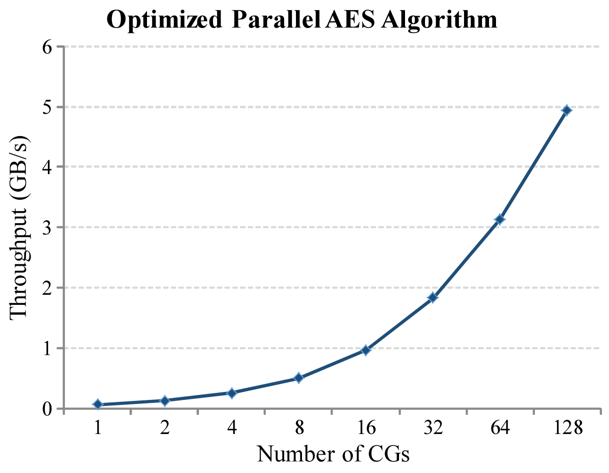

FIGURE 16 The throughput (GB/s) of encrypting 1-GB input data using the optimized parallel AES algorithm on multiple CGs

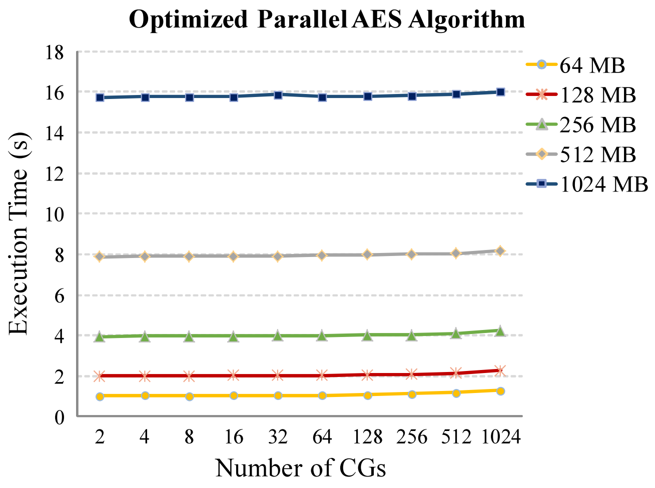

FIGURE 17 The execution time of the optimized parallel AES algorithm on multiple CGs; in particular, the problem size of each CG is fixed as 64 MB, 128 MB, 256 MB, 512 MB, and 1024 MB

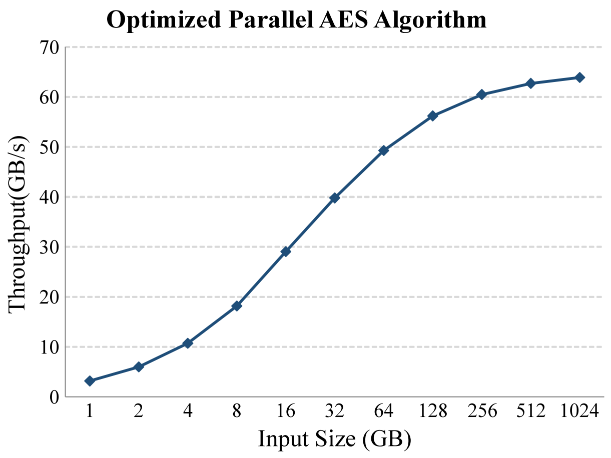

FIGURE 18 The throughput (GB/s) of the optimized parallel AES algorithm on 1024 CGs

CG is constant, the total amount of problem size for *N* nodes is multiplied with the number of CGs increasing. We can infer from Figure 17 that the communication overhead among CGs is very small. With the workload of each CG unchanged, applying more CGs can deliver higher throughput and greater efficiency.

Therefore, we also test the throughput of the optimized parallel AES encryption varying the input size on 1024 CGs. As Figure 18 has shown, with input size increasing gradually, the trend-line of throughput creates a very fast rising trend on 1024 CGs. Comparing Figure 18 with Figure 13, we conclude that increasing the size of input data will allow a higher throughput but will also require more CGs.

TABLE 2 The comparison of throughput of the optimized parallel AES algorithm on single CG and multiple CGs

Number of CGs	Throughput(GB/s)
1	0.064
4	0.254
16	1.015
64	4.056
256	16.180
1024	63.910

Table 2 shows the performance comparison of the optimized parallel AES algorithm on a CG and multiple CGs. Especially, the problem size for each CG is a fixed-size of 1 GB in the experiments. This illustrates that better performance of parallel AES encryption can be achieved by using more CGs. We achieve the throughput of about 64 GBytes/s (511.28 Gbits/s) on 1024 CGs, which proves that our parallel implementation of AES algorithm on the Sunway TaihuLight has strong scalability.

All the experimental results in this paper indicate that the proposed data protecting model obtains high parallel efficiency on the Sunway TaihuLight.

7 | CONCLUSION

In this study, we have proposed a data protecting model using the parallel AES algorithm and the SHA3-256 on the Sunway TaihuLight. We propose a fine-gained software design and optimization strategies for the data protecting model based on the architecture of Sunway. Our data protecting model has high security and excellent efficiency of data encryption/decryption. The optimized parallel AES algorithm of our data protecting model achieves high throughput of 511.28 Gbits/s on the Sunway. The experimental results validate the effectiveness of our data protecting model on the Sunway TaihuLight. We provide solutions for companies and organizations to efficiently protect confidentiality and integrity of large-scale data.

In our future work, we will design our data protecting model based on other many-core architectures, such as the heterogeneous architecture of CPU-GPU/MIC.

ACKNOWLEDGMENTS

The authors would like to thank the two anonymous reviewers for their valuable and helpful comments on improving the manuscript. The research was partially funded by the National Key R&D Program of China (2016YT80201900), the Key Program of National Natural Science Foundation of China (61432005), the National Outstanding Youth Science Program of National Natural Science Foundation of China (61625202), the Open Fund of the State Key Laboratory of Mathematical Engineering and Advanced Computing (898209040), and the Youth Program of National Natural Science Foundation of China (61602166).

ORCID

Yuedan Chen http://orcid.org/0000-0001-5665-268X
Kenli Li http://orcid.org/0000-0002-2635-7716

REFERENCES

1. Daemen J, Rijmen V. *The Design of Rijndael: AES - the Advanced Encryption Standard*. Berlin, Germany: Springer-Verlag Berlin Heidelberg New York; 2002. Information Security and Cryptography.
2. The Top500 List. http://www.top500.org/. Accessed July 26, 2016.
3. Tao J, Blazewicz M, Brandt SR. Using GPU's to accelerate stencil-based computation kernels for the development of large scale scientific applications on heterogeneous systems. Paper presented at: PPoPP 2012 Proceedings of the 17th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming; 2012; New Orleans, LA.
4. Carretero J, García Blas J, Neytcheva MG. Introduction to the special section on "optimization of parallel scientific applications with accelerated high-performance computers". *Comput Electr Eng*. 2015;46:78-80.
5. Chen Y, Li K, Fei X, Quan Z, Li K. Implementation and optimization of AES algorithm on the Sunway TaihuLight. Paper presented at: 2016 17th International Conference on Parallel and Distributed Computing, Applications and Technologies; 2016; Guangzhou, China.
6. Hodjat A, Verbauwhede I. Area-throughput trade-offs for fully pipelined 30 to 70 Gbits/s AES processors. *IEEE Trans Comput*. 2006;55(4):366-372.
7. Wang M-Y, Su C-P, Horng C-L, Wu C-W, Huang C-T. Single- and multi-core configurable AES architectures for flexible security. *IEEE Trans Very Large Scale Integr (VLSI) Syst*. 2010;18(4):541-552.

8. Mathew SK, Sheikh F, Kounavis M, et al. 53 Gbps native GF$(2^4)^2$ composite-field AES-encrypt/decrypt accelerator for content-protection in 45 nm high-performance microprocessors. *J Solid-State Circuits*. 2011;46(4):767-776.
9. Desai A, Ankalgi K, Yamanur H, Navalgund SS. Parallelization of AES algorithm for disk encryption using CBC and ICBC modes. Paper presented at: 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT); 2013; Tiruchengode, India.
10. Patel RR, Kanjariya SD, Dabhi JB. Design of parallel advanced encryption standard (AES) algorithm. *Int J Res Comput Commun Technol*. 2015;4(3):219-222.
11. Parmar ND, Kadam P. Pipelined implementation of dynamic Rijndael S-box. *Int J Comput Appl*. 2015;111(10):36-38.
12. Hodjat A, Verbauwhede I. A 21.54 Gbits/s fully pipelined AES processor on FPGA. Paper presented at: 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines; 2004; Napa, CA.
13. Thulasimani L, Madheswaran M. A single chip design and implementation of AES-128/192/256 encryption algorithms. *Int J Eng Sci Technol*. 2010;2(5):1052-1059.
14. Wang Y, Ha Y. FPGA-based 40.9-Gbits/s masked AES with area optimization for storage area network. *IEEE Trans Circuits Syst II: Express Briefs*. 2013;60(1):36-40.
15. Wang Y, Ha Y. High throughput and resource efficient AES encryption/decryption for SANs. Paper presented at: 2016 IEEE International Symposium on Circuits and Systems (ISCAS); 2016; Montréal, Canada.
16. Rahimunnisa K, Karthigaikumar P, Rasheed S, Jayakumar J, Suresh Kumar S. FPGA Implementation of AES algorithm for high throughput using folded parallel architecture. *Secur Commun Netw*. 2014;7(11):2225-2236.
17. Harrison O, Waldron J. AES encryption implementation and analysis on commodity graphics processing units. In: Proceedings of the 9th International Workshop on Cryptographic Hardware and Embedded Systems CHES '07; 2007; Vienna, Austria.
18. Tran N-P, Lee M, Hong S, Lee S-J. Parallel execution Of AES-CTR algorithm using extended block size. Paper presented at: 2011 14th IEEE International Conference on Computational Science and Engineering; 2011; Dalian, China.
19. Luo C, Fei Y, Luo P, Mukherjee S, Kaeli DR. Side-channel power analysis of a GPU AES implementation. Paper presented at: 2015 33rd IEEE International Conference on Computer Design (ICCD); 2015; New York City, NY.
20. Patchappen M, Yassin YM, Karuppiah EK. Batch processing of multi-variant AES cipher with GPU. Paper presented at: 2015 Second International Conference on Computing Technology and Information Management (ICCTIM); 2015; Johor, Malaysia.
21. Adeshina AM, Hashim R. Computational approach for securing radiology-diagnostic data in connected health network using high-performance GPU-accelerated AES. *Interdiscip Sci Comput Life Sci*. 2016;9(1):140-152.
22. Khan AH, Al-Mouhamed MA, Almousa A, Fatayar A, Ibrahim AR, Siddiqui AJ. AES-128 ECB encryption on GPUs and effects of input plaintext patterns on performance. Paper presented at: 15th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD); 2014; Las Vegas, NV.
23. Guo G, Qian Q, Zhang R. Different implementations of AES cryptographic algorithm. Paper presented at: 2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems; 2015; New York, NY.
24. Niewiadomska-Szynkiewicz E, Marks M, Jantura J, Podbielski M. A hybrid CPU/GPU cluster for encryption and decryption of large amounts of data. *J Telecommun Inf Technol*. 2012;3:32-39.
25. Marks M, Niewiadomska-Szynkiewicz E. Hybrid CPU/GPU platform for high performance computing. Paper presented at: 28th European Conference on Modelling and Simulation (ECMS 2014); 2014; Brescia, Italy.
26. Bayat-Sarmadi S, Mozaffari-Kermani M, Reyhani-Masoleh A. Efficient and concurrent reliable realization of the secure cryptographic SHA-3 algorithm. *IEEE Trans Comput-Aided Des Integ Circuits Syst*. 2014;33(7):1105-1109.
27. Moreira N, Astarloa A, Kretzschmar U. SHA-3 based message authentication codes to secure IEEE 1588 synchronization systems. Paper presented at: IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society; 2013; Vienna, Austria.
28. Arshad A, Kundi DES, Aziz A. Compact implementation of SHA3-512 on FPGA. Paper presented at: 2014 Conference on Information Assurance and Cyber Security (CIACS); 2014; Rawalpindi, Pakistan.
29. Ahmed KE, Farag MM. Hardware/software co-design of a dynamically configurable SHA-3 System-on-Chip (SoC). Paper presented at: 2015 IEEE International Conference on Electronics, Circuits, and Systems (ICECS); 2015; Cairo, Egypt.
30. Xiao F, Luting C, Zibin D, Wei L. IMSet-SHA3-Tree: The efficient data integrity verification based on SHA3 and MSet-XOR-hash. Paper presented: 2015 IEEE 12th International Conference on Ubiquitous Intelligence and Computing, 2015 IEEE 12th International Conference on Autonomic and Trusted Computing, and 2015 IEEE 15th International Conference on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom); 2015; Beijing, China.
31. Fei X, Li K, Yang W, Li K. A secure and efficient file protecting system based on SHA3 and parallel AES. *Parallel Comput*. 2016;52:106-132.
32. Dongarra J. Report on the Sunway TaihuLight System. University of Tennessee Computer Science Technical Report UT-EECS-16-742. Knoxville, TN: The University of Tennessee - Department of Electrical Engineering & Computer Science; 2016. http://www.netlib.org/utk/people/JackDongarra/PAPERS/sunway-report-2016.pdf

How to cite this article: Chen Y, Li K, Fei X, Quan Z, Li K. Implementation and optimization of a data protecting model on the Sunway TaihuLight supercomputer with heterogeneous many-core processors. *Concurrency Computat Pract Exper*. 2018;e4758. https://doi.org/10.1002/cpe.4758