IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON SMART CACHING, COMMUNICATIONS, COMPUTING AND CYBERSECURITY FOR INFORMATION-CENTRIC INTERNET OF THINGS

Received May 30, 2018, accepted July 18, 2018, date of publication August 2, 2018, date of current version September 5, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2862425

ACA-SDS: Adaptive Crypto Acceleration for Secure Data Storage in Big Data

CHUNHUA XIAO[1], PENGDA LI[1], LEI ZHANG[1], WEICHEN LIU[2], (Member, IEEE), AND NEIL BERGMANN[3], (Member, IEEE)

[1]Department of Computer Science, Chongqing University, Chongqing 400044, China
[2]School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798
[3]School of Information Technology and Electrical Engineering, The University of Queensland, Brisbane, QLD 4072, Australia

Corresponding author: Chunhua Xiao (xiaochunhua@cqu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61502061, in part by the Chongqing Application Foundation and Research in Cutting-Edge Technologies under Grant cstc2015jcyjA40016, in part by the Fundamental Research Funds for the Central Universities under Grant 106112017CDJXY180004, and in part by the Program of China Scholarship Council under Grant 201706055029.

ABSTRACT In the era of big data, the demand for secure data storage is rapidly increasing. To accelerate the complex encryption computation, both specific instructions and hardware accelerators are adopted in a large number of scenarios. However, the hardware accelerators are not so effective especially for small volume data due to the induced invocation costs, while the AES-NI (Intel®advanced encryption standard new instructions) is not so energy efficiency for big data. To satisfy the diversity performance/energy requirements for intensive data encryptions, a collaborative solution is proposed in this paper. We proposed a feasible hardware-software co-design methodology based on the stack file system eCryptfs, with quick assist technology, which is named adaptive crypto acceleration for secure data storage (ACA-SDS). ACA-SDS is able to choose the optimal encryption solution dynamically according to file operation modes and request characters. Adjustable parameters, such as α, β, and M are provided in our scheme to provide a better adaptability and tradeoff choices for encryption computation. Our evaluation shows that ACA-SDS can get 15%–25% performance improvement for big-data blocks compared with only software or hardware accelerations. Furthermore, our methodology provides a wide range of practical design concepts for the further research in this field.

INDEX TERMS Hardware-software co-design, eCryptfs, encryption, data-intensive, QAT.

I. INTRODUCTION

The widespread growth of data usage and cloud computing technology has resulted in surging security demand for data transmission and storage. The demand for intensive-data encryption is also increasing year by year, and more encryption technology are applied in many other fields. Thus, the security, efficiency, and performance of data-intensive encryption are key factors in the mass data management. For data-intensive crypto computation, specific instruction acceleration (such as AES-NI) and hardware accelerators (such as intel QAT) are widely adopted to ease the performance slowdown induced by complex encryption algorithms. However, either way has its own benefits and drawbacks.

In terms of computer instruction set encryption, the CPU dependency might be a bottleneck for performance improvement. The development of hardware manufacturing has reached the stage when the transistor speed almost reaches the limit. The maximum speed of its switch is basically stable at about 4G, which matches the CPU speed of the current Intel i7 series. If the overclocking strategy is adopted, it can only be achieved by increasing power consumption (exponential power increase) or lowering the temperature. However, both methods are highly expensive. At the same time, a software encryption acceleration algorithm cannot reduce the quantity of computation involved and instruction set encryption consumes a lot of CPU resources. Therefore, hardware acceleration with specify crypto engines is a predominant alternative for better computation off-loading.

Hardware acceleration focus on the complex algorithm computations in a more energy-efficiency manner, while

only management resource needed from CPUs. With hardware accelerators, CPUs are able to handle other necessary tasks/services besides processing management and scheduling. If there are multiple CPUs and multiple accelerators, it is easy to get a high throughput through parallel crypto computations. The most representative technique of hardware accelerators is QAT [16], [17]. QAT provides typical computation accelerations including data compression and encryptions, which has been applied widely for server-based encryption, network transmission and data storage, etc. However, due to the generality of hardware accelerators, it is hard to take full utilization of them in the working flow of a specific application scenario without optimization. Our experiments showed the crypto acceleration is not efficiency for certain file operations in data storage; similarly, it cannot work efficiently for small crypto data blocks.

Based on the above analysis, encryption operation through single hardware or software has its characteristic strengths and weaknesses. Therefore, this paper proposes a software-hardware co-design method for data-intensive encryption, named Adaptive Crypto Acceleration for Secure Data Storage (ACA-SDS). The goal of this system is to achieve a better crypto performance through better resource utilization.

The contributions of this paper include:

- We did a comprehensive analysis and comparison for the existing crypto acceleration methodologies from different respective. We revealed the cons and pros of two predominant crypto solutions, which can be referred for a better strategy decision for security data storage scenarios.
- We proposed an adaptive Hardware/Software co-scheduling strategy, which is able to choose the most reasonable crypto methodology dynamically according to the operation types, request characters, and CPU load. Adjustable parameters and threshold are also provided along with the scheme, so the users can customize the trade-off between performance and cost. Further, we discussed scheduling schemes under multiple CPUs. Our experiment showed that our proposed solution can get 15%-25% performance improvement for big-data blocks compared with existing crypto acceleration methods.
- An efficient management and scheduling scheme is proposed for hardware crypto accelerates. In the actual system, it will be dynamically adjusted according to the demand characteristics of the encryption task, current system operating conditions and external environmental factors. Encryption operations become more efficient and robust by setting the appropriate index parameters in the system.

The rest of this paper is organized as follows. Section II introduces our research motivation. Section III shows the design ideas and scheme. Section IV presents the analysis of the experimental data. The related work is discussed in Section V. Finally, the paper is concluded in Section VI.

II. RESEARCH MOTIVATION

In order to protect sensitive data, documents are encrypted before storing them on devices. As result, many file systems are proposed to transparently handle encryption and decryption when accessing data. These file systems are either built on FUSE [6], such as goCryptfs [7], CryFS [8], or integrated into the Linux kernel, such as LUKS [10] and eCryptfs [11]. All the file systems provide common strategies to handle security metadata, such as File Encryption. However, one fundamental challenge is not addressed by state-of-the-art cryptographic file systems: how to avoid the performance degradation caused by encryption operations. The widely-used ciphers, such as AES-CBC, AES-GCM and 3DES, are all time-consuming, they require considerable computational power to manipulate byte streams. Take eCryptfs as the example, eCryptfs based security data storage showed a poor performance for write operation, which is 16 times worse in bandwidth than original file system without any crypto operations. While the cost of reading operation is not too large, only a drop of about 20% -30%. Since the Page Cache stores only the plaintext, subsequent read operations do not have the overhead after the first read of the data requires a decryption operation; when x bytes of data are written, the encryption operation requires $((x-1)\ /\ extent\_size + 1) * extent\_size$ (the minimum size of the data storage unit) data bytes, which results in a relatively large overhead [1].

As we discussed in the section I, both of the crypto acceleration methodologies (AES-NI and hardware accelerators) have serious limitations to ease the performance slowdown for secure data storage with eCryptfs. To target the problem of existing crypto solutions, we did lots experiments and comparisons to analyze the performance (operation delay) and CPU idle in different situations (different file operations, different data blocks, and different concurrency).

We tested the crypto delay with standard fio tool for AES-NI and QAT separately. If the concurrency level of 12 is selected, the evaluation data for a single processing is 500, and so file size is 6000M in total. If the concurrency level is 32, the data size for file evaluation is 16000 totally.

To illustrate the performance clearly, we concluded the definition of major testing parameters as below:

- *slat* represents the delay time from the data submission to the time when data is processed by the kernel;
- *clat* represents the time for encryption/decryption completion;
- *lat* represents total time taken from task submission to encryption/decryption completion.

In relation, $lat = slat + clat$.

The testing results are illustrated as table 1, in which microsecond (μs) is adopted for delay description. We varied the testing data size from 1K to 64K to see the performance for different data blocks. Both AES-NI and QAT based crypto acceleration methodologies are tested with 12 concurrency level and 32 concurrency level separately.

We compared the performance difference with AES-NI and QAT for read and write operations, the influence of data-block size and concurrency level, and also the CPU idle.

Discussion 1: The Read. As we can see from the table 1, for both of the methodologies, the *slat* time majorly depends on *clat* time, and the *clat* time is increasing along with the size increasing. However, the AES-NI performs better for all of the testing with different size of data blocks. The *clat* time for QAT increases along with the data block size from 8μs to 440μs, while the AES-NI only increases from 4μs to 160μs, which makes *clat* time of QAT be more than twice of AES-NI. Through this testing, we can conclude that QAT is not efficiency for read operation compared with AES-NI. In another words, if we need to do decryption, it is better to choose software acceleration with AES-NI.

TABLE 1. Delay time of data operations.

Block Size(K) Unit(μsec)	QAT 12(read)	AES-NI 12(read)	QAT 12(write)	AES-NI 12(write)	QAT 32(write)	AES-NI 32(write)
slat (1K)	0.05	0.1	0.06	0.18	0.13	0.37
clat (1K)	7.95	3.52	35.16	29.04	72.63	79.26
lat (1K)	8	3.62	35.22	29.22	72.76	79.63
slat (2K)	0.05	0.14	0.07	0.20	0.13	0.44
clat (2K)	14.09	6.25	41.60	31.94	85.76	87.08
lat (2K)	14.14	6.39	41.67	32.14	85.89	87.52
slat (4K)	0.05	0.15	0.09	0.18	0.18	0.47
clat (4K)	27.36	11.14	28.77	30.63	69.86	83.8
lat (4K)	27.41	11.29	28.86	30.81	70.04	84.27
slat (8K)	0.05	0.14	0.12	0.26	0.29	0.86
clat (8K)	55.92	21.34	61.21	64.85	131.09	161.05
lat (8K)	55.97	21.48	61.33	65.11	131.38	161.91
slat (16K)	0.06	0.07	0.26	0.35	0.55	1.11
clat (16K)	111.53	40.21	114.36	118.23	251.83	320.34
lat (16k)	111.59	40.28	114.62	118.58	252.38	321.45
slat (32K)	0.06	0.13	0.51	0.63	1.02	2.19
clat (32K)	221.72	81.2	229.01	235.15	506.02	636.42
lat (32K)	221.78	81.33	229.52	235.78	507.04	638.61
slat (64K)	0.06	0.05	1.13	1.78	2.43	3.77
clat (64K)	440.49	159.02	456.91	467.71	979.54	1245.04
lat (64K)	440.55	159.07	458.04	469.49	981.97	1248.81

Discussion 2: The Write. As we can see from the table 1, the change trend and percentage of *clat* time is the same as the read operation in the 12-concurrency level. However, the QAT performance is better when testing larger size of data blocks. The *clat* time of QAT gradually increases with the data blocks ranging from 36μs to 458μs, and the AES-NI increases from 30μs to 468μs. At 4k, The QAT's *clat* time starts to fall below AES-NI. And this trend has been maintained to the last data block of 64K. By this comparison, we can get the following conclusions that QAT has advantages over the AES-NI with larger block sizes in writing. Therefore, if we need to do encryption with large blocks, it is better to choose hardware acceleration with QAT.

Discussion 3: The Concurrency Level & Block Size. As can be seen from the comparison of write between 12 concurrency level and 32 concurrency level in Table 1. With larger concurrency and data blocks, QAT has greater performance advantages compared with 12 concurrency level. The total *slat* time is also small, and the *lat* time of QAT ranges from 73μs to 982μs at 32 concurrencies, while the AES-NI ranges from 80μs to 1,249μs. With the data block increases, the QAT's range-difference ratio and relative time increases than AES-NI. At the same time, the ratio of AES-NI and QAT based on *lat* time is about 1.3 times in 32 concurrency level, while the ratio is only about 1.02 times in 12 concurrency level. So, we know that QAT's encryption performance advantage is more obvious under the big block and high concurrency. Therefore, the hardware and software calls can be dynamically adjusted according to the actual block size and number of concurrencies in writing.

Discussion 4: The CPU Idle Resources. As shown in Figure 1, when the data is encrypted, QAT is still a portion of idle CPU resources compared to the AES-NI. In the trend of data blocks from small to large, QAT has 3%-12% of the remaining CPU resources, while AES-NI's idle CPU resources are always zero. This is a flash point for QAT. For this part of the excess resources available, we can make more use of it to make the system more efficient. Therefore, according to the actual CPU idle condition, we can decide whether to call the AES-NI continues to encrypt or complete other tasks.

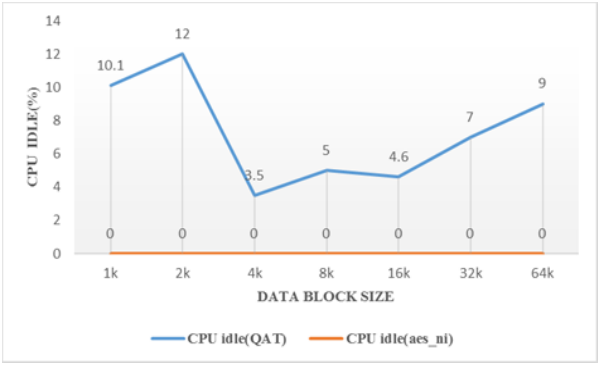


FIGURE 1. CPU idle status for 16-process in writing.

As we discussed above, in term of secure data storage with AES-NI and QAT in eCryptfs, there is a shortage of hardware in secure reading, and a weakness of software in secure writing. Besides, the size of data blocs and the concurrency level play an important role to data encryption.

Based on these interesting findings, we propose Adaptive Crypto Acceleration for Secure Data Storage (ACA-SDS),

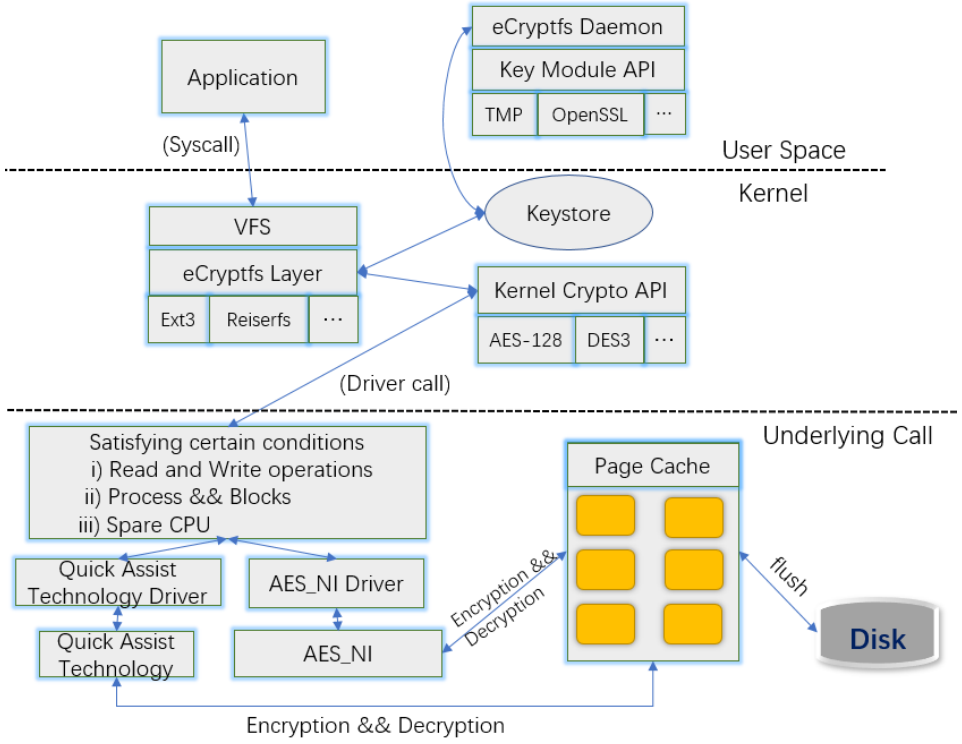


FIGURE 2. Adaptive Encryption Scheduling System (ACA-SDS).

which can make best use of the advantages and bypass the disadvantages of both AES-NI and hardware accelerators (QAT).

III. DESIGN IDEAS AND SCHEME

A. ARCHITECTURE OVERVIEW OF ACA-SDS

Section II compares and analyzes the encryption of AES-NI and QAT, which shows that each encryption method has its characteristic advantages and disadvantages. Based on this analysis, we propose a software-hardware collaborative design scheme based on the eCryptfs stack encryption file system. The architecture overview of ACA-SDS is shown as figure 2.

In the upper part of ACA-SDS, eCryptfs is a stack-encrypted file system used in the Linux file system. It belongs to the middle tier of the file model and lies between the virtual file system (VFS) and the universal file system (UFS). During the actual data file encryption and decryption operation access process, the VFS layer interface is invoked on the application layer to perform data access operations, and the UFS interface is uniformly invoked to store and retrieve the data at the system layer. The eCryptfs is only needed to implement data encryption and decryption operations. And it uses symmetric key encryption algorithm to encrypt the data. The commonly used encryption algorithms are DES, AES128, etc. The file encryption key (FEK) is randomly generated and applied to the encryption of the current file data. Subsequently, the Key Encryption Key (KEK) is used to process and generate FEK ciphertext and to place it in the encrypted file header for storage and dispatch.

In the lower part of ACA-SDS, for lower-tier driver calls, the API (Application Programming Interface) file chooses whether to call hardware encryption (QAT) or instruction set encryption (AES-NI) based on whether the encryption environment meets certain criteria for read and write operations, processes and blocks, and CPU idle. After encrypted by AES-NI or QAT, the ciphertext is written into Cache. Then the dirty page in Cache is written to disk. For decryption, the ciphertext is read into Cache from disk. After decrypting the data, the plaintext is returned to the user.

The proposed method is implemented in the kernel. QAT has a separate driver, which is called by eCryptfs in file read and write operations. The driver implements encryption and decryption functions. Data is stored in the kernel's page cache for temporary scheduling decisions and written back to disk only when the contents are encrypted.

B. SCHEDULING DESIGN

1) READ/WRITE OPERATION SCHEDULING

Decryption of read data and encryption of write data are the basic functions of eCryptfs. Those operations have different operational complexity for the eCryptfs file system. From the preceding information, the performance of the write operation is relatively poor, as it takes time about 16 times longer than an unencrypted write operation. The read operation

overhead is not as large, the performance is decreased by 20% -30% only. Therefore, QAT is highly suitable for write encryption in actual operation, to take advantage of its speed and efficiency. Detailed data delay time for read/write operation are shown in Table 1, QAT is hardware-based encryption, while AES-NI is based on instruction set encryption.

From the data analysis, the following Hardware-Software co-design program for data read/write operation can be made. When the data is read and decryption operations, the CPU calls the AES-NI; when the data is written and encryption operations, the CPU calls the QAT.

$$E = \begin{cases} S & (Read) \\ H & (Write) \end{cases} \tag{1}$$

As in Formula (1), E represents the performance of the system, S represents the software AES-NI operation, and H represents the hardware QAT operation. Based on this, we use a combination of hardware and software encryption strategy. When the read operation request is issued, the system calls the instruction set to work; and when the write operation request is issued, the system calls the hardware QAT to work.

2) SCHEDULING BASED ON BLOCK SIZE & CONCURRENCY LEVEL

The number of the concurrency in the task process may affect the encryption performance of AES-NI and QAT. Furthermore, there is also the possibility that block size affects the performance of the AES-NI and QAT for data encryption and decryption operation. Based on the size of the data block and concurrency level, a suitable combination of hardware and software programs can be selected. After changing the concurrency level and data blocks in the test, the hardware and software calls should also be changed to achieve the best performance.

As seen in the Figure 3., in the 32 processes, the bandwidth of QAT and AES-NI gradually expands along with the data block increasing for encryption. And the advantage of QAT's encryption bandwidth value is more and more obvious compared to AES-NI with the big blocks.

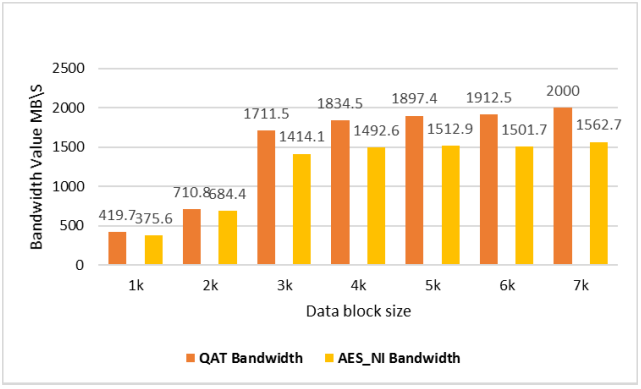


FIGURE 3. 32 concurrency level in writing.

Based on this analysis, a software and hardware combination scheme can be made as follows.

$$E = \begin{cases} H\ (process > \alpha\ AND\ blocks > \beta) \\ S\ (others) \end{cases} \tag{2}$$

In Formula (2), E represents the performance of the system, H represents the hardware QAT operation, and S represents the software AES-NI operation. While α and β are the adaptive scheduling values. The AES-NI is called when the number of tasks is less than α or the size of data block is less than β. QAT is called when the number of tasks and the page data block is large. Thus, we can conclude that QAT is applied for more heavy tasks in the work environment. Such an arrangement enables the system to make better use of the resources and maximize its own work efficiency.

This section that the data read and write operations analysis shows that AES-NI has a greater advantage in reading and decrypting, while QAT is superior in writing and encrypting (in the actual situation, writing and encrypting are more important because it takes more time). So, we design related strategies to take full advantage of the unique advantages of the QAT and AES-NI.

C. HARDWARE-SOFTWARE DYNAMIC SCHEDULING SCHEME

According to the analysis in Section B, we can come up with two optimization schemes. In addition to the above optimization scheme, QAT also has CPU idle resources compared to AES-NI, because there is a hardware interrupt when the CPU calls the QAT. For existing idle CPU resources, we can shift its free CPU resources to relative performance.

$$E = \begin{cases} H\ (Spare\ CPU \leq M) \\ S + H\ (others) \end{cases} \tag{3}$$

As in Formula (3), E represents the efficiency of the system, H represents the hardware QAT operation, and S represents the software AES-NI operation. While M represents the system adaptive scheduling values When the system calls the QAT to encrypt data, the current CPU utilization of the system can be monitored. If the percentage of idle CPU resource is less than M (adaptive scheduling value), the system runs other waiting sequence commands. If the idle CPU is greater than the M value, the system can use the remaining CPU resources to invoke the software for encryption and decryption. This method of combining software and hardware can maximize the CPU utilization and improve performance for data encryption.

The above combination of hardware and software scheme can be represented in the following flow chart for data-intensive encryption shown in the Figure 4. below, where M, α and ß are the adaptive scheduling values.

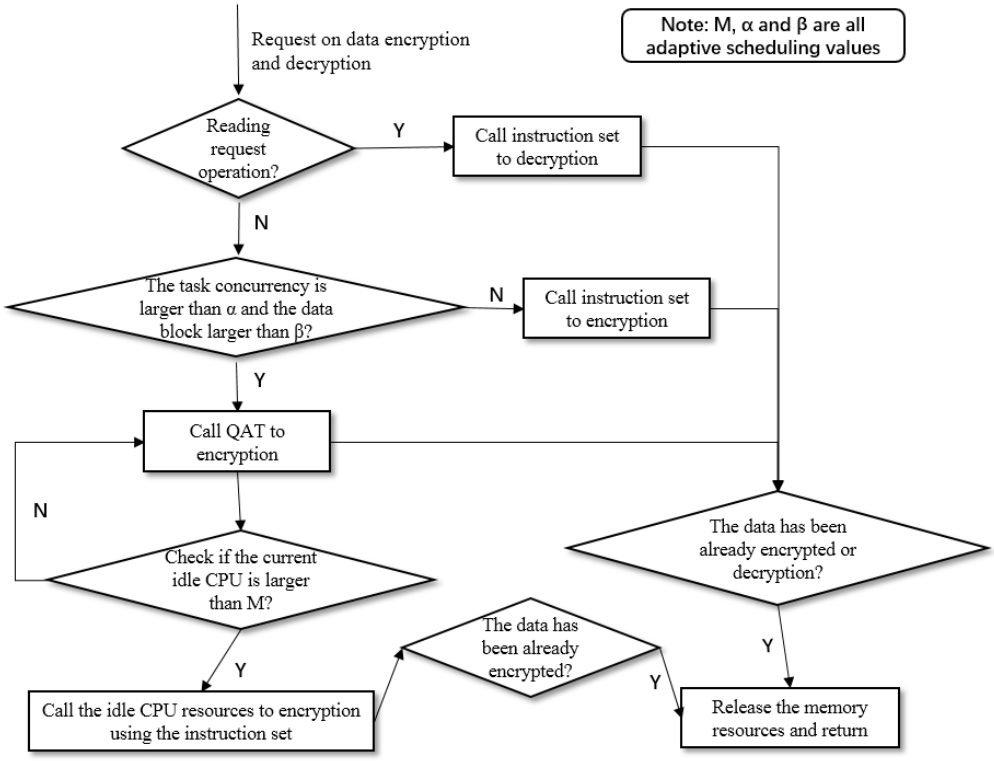


FIGURE 4. Flow chart of Hardware-Software Co-design.

At the same time, the related pseudo-code for data encryption and decryption completed by the hardware/software co-designed is shown as algorithm1.

This section proposes an adaptive software/hardware co-scheduling scheme. Based on this, through the observation and comprehensive consideration of the CPU idle resource in the hardware, a coordinated scheduling of spare CPU resources is proposed, and the related structure/flow chart are drawn.

D. ALLOCATION AND MANAGEMENT OF MULTI-CPU

1) THE QAT PERFORMANCE WITH NUMBER OF CPUS

For the hardware acceleration engine QAT, when the system calls the QAT to perform eCryptfs file encryption and decryption operations, it usually generates a hardware interrupt and allows the CPU more time to deal with other tasks. Only when QAT's encryption and decryption tasks for the requested data are completed, the system will wait for the return of information, after which the CPU can continue to call QAT. Therefore, the bandwidth performance of QAT encryption and decryption depends on the capabilities of the QAT hardware itself.

From the above, QAT's accelerate bandwidth must exist a limit value. When the QAT bandwidth reaches or closes the limit, despite the increase in the number of CPUs, the QAT hardware acceleration bandwidth is saturated and cannot further expand bandwidth. Therefore, the optimal number of CPUs to call the QAT hardware acceleration engine can be chosen at the design stage, and the hardware-software combination design can be optimized by determining the CPU threshold for QAT.

$$P = Max(bw/n) \tag{4}$$

In Formula (4), bw represents the data bandwidth, n represents the number of CPUs, and P represents the bandwidth for a single CPU. This formula measures the average distribution performance of the CPU and is one of the most important performance evaluation indicators. In addition, CPU utilization is another related parameter that can used to evaluate whether the QAT is in a saturated state. Both reference indicators can be combined to test experimental data.

In measuring the number of CPUs for QAT, the test conditions set are as follows: single data thread size is 500M, 32K data block size, 32 concurrency levels and sequential write operations.

In fact, a QAT has twelve acceleration modules. As can be seen from Figure 5., the number of CPUs required to manage the QAT hardware acceleration with maximum efficiency is 4 or 5. When the number of CPUs exceeds 5, the value of the QAT encryption bandwidth is maintained at an interval value and the system needs more CPU management resources. Further comparison and analysis show that the P bandwidth value for 4 CPUs is 464.5 MB/(S*single), and the P value for 5 CPUs is 416.4 MB/(S*single). Therefore, in the case of multi-CPU encrypted data, setting the number of CPUs that call QAT to 4 is most appropriate. This can

Algorithm 1 Encryption Scheduling Algorithm

Input: *Req: The request for data encryption and decryption*
Key: symmetric key
Output: *IS_encryption(P, key): plaintext pages are encrypted by instruction set operations*
IS_decryption(C, key): ciphertext pages are decrypted by instruction set operations
HW_encryption(P, key): plaintext pages are encryptedby Hardware Acceleration operations
HW_decryption(C, key): ciphertext pages are decrypted by Hardware Acceleration operations
Note: M, α and β are all adaptive scheduling values.
Accept Req from the Application or Disk
If *Rec is* ***Read***
then *do IS_decryption(C, key)*
else if *(process* $>$ *α && blocks* $>$ *β)*
then *do HW_encryption(P, key)*
While*Spare CPU* $>$ *M && Res exists*
do *IS_encryption(P, key)*
End While
else
then *do IS_encryption(P, key)*
End if
End

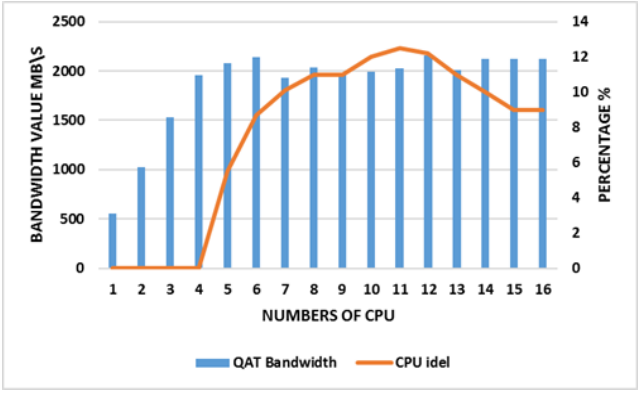


FIGURE 5. Bandwidth and CPU idle during QAT call.

get QAT encryption performance with minimal CPU resource utilization.

2) MULTI-CPU ALLOCATION STRATEGY

For multi-CPU-based server data encryption storage, if the server has n CPUs ($n > 4$), you can use the software-hardware combination scheme to encrypt the data. When the server receives a large number of data encryption requests, it can use four CPUs to manage QAT for encryption, and the rest $(n-4)$ use the instruction set for encryption. In this way, data encryption performance can be effectively improved compared to using hardware or software alone to complete encryption.

From the theoretical derivation and the analysis of specific data in this section, we propose the scheduling allocation of the hardware and software in reading-writing operations and data factors of the concurrent number and block size. Based on the above two points, we further propose the effective use of idle CPU resources in the hardware engine encryption process. Finally, from the analysis of the hardware and software implementation methods, we propose a feasible scheme that obtains the maximum hardware acceleration efficiency with minimal resource management and coordinates scheduling hardware-software resources in the case of multiple CPUs.

IV. EXPERIMENT AND ANALYSIS

A. EXPERIMENTAL ENVIRONMENT

1) SOFTWARE ENVIRONMENT

The software used in this experiment is the eCryptfs file system and the fio test tool. The system uses symmetric key encryption. Table 2 shows the specifications and versions of the software in the experimental setup.

TABLE 2. Software information.

Name	Version
OS Kernel	Linux 4.10.10
Operating system	Ubuntu 17.0
Hardware drive Instruction set drive	QAT_dh895xcc.ko aesni_intel.ko
eCryptfs	ecryptfs_utils_111
fio	fio 2.0.7

2) HARDWARE ENVIRONMENT

The main hardware used in this experiment is CPU, RAM, hard disk and QAT hardware acceleration engine. The QAT hardware encryption engine has 12 built-in encryption acceleration modules. Table 3 lists the relevant hardware versions.

TABLE 3. Hardware information.

Name:	Configuration:
CPU	Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz, quad core
RAM	128GB
Disc space	1TB
QAT	12 acceleration modules

B. CONFIGURATION AND TESTING FLOW OF ECRYPTFS

In the experiment, eCryptfs uses the AES-128 algorithm as the encryption algorithm. FEK is an encryption key, which consists of 16 bytes. When eCryptfs is mounted, all data placed in the specified folder must be encrypted. The file name is not encrypted by default. After installing the eCryptfs file system in the experiment, we configure the related parameters for mounting. The specific operation is shown in Figure 6. -eCryptfs mounting information.

This experiment uses fio to test IO bandwidth and iops values. The iops value is used as a performance indicator for

```
Passphrase:
Select cipher:
 1) aes: blocksize = 16; min keysize = 16; max keysize = 32
 2) blowfish: blocksize = 8; min keysize = 16; max keysize = 56
 3) des3_ede: blocksize = 8; min keysize = 24; max keysize = 24
 4) twofish: blocksize = 16; min keysize = 16; max keysize = 32
 5) cast6: blocksize = 16; min keysize = 16; max keysize = 32
 6) cast5: blocksize = 8; min keysize = 5; max keysize = 16
Selection [aes]:
Select key bytes:
 1) 16
 2) 32
 3) 24
Selection [16]:
Enable plaintext passthrough (y/n) [n]:
Enable filename encryption (y/n) [n]:
Attempting to mount with the following options:
  ecryptfs_unlink_sigs
  ecryptfs_key_bytes=16
  ecryptfs_cipher=aes
  ecryptfs_sig=c0f0403c08e48b1d
Mounted eCryptfs
```

FIGURE 6. eCryptfs mounting information.

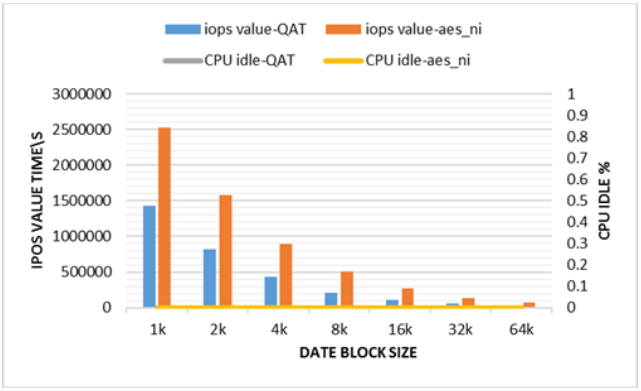


FIGURE 7. IOPS value for 12-process concurrency level read operation.

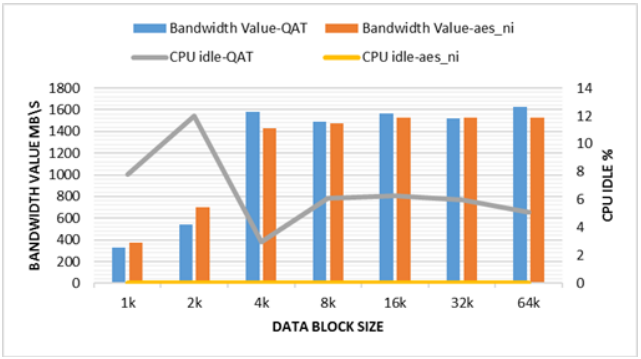


FIGURE 8. Bandwidth value for 12-process concurrency level write operation.

reading and decrypting, and the bandwidth value is used as a performance indicator for writing and encryption. In the experiment, we obtained performance data for certain aspects of software and hardware by setting related parameters and testing data. See the column chart for details.

This experiment uses the Linux command 'top' to get the current CPU utilization. 'top' is a common performance analysis tool for Linux systems. It can display real-time information about the process being executed in the system, including process ID, memory usage, and CPU usage, etc. You can use the top tool to view the following CPU parameters. The following information is a CPU-side parameter index displayed by using the top command.

Command: top

- *us* represents the CPU occupied by user space;
- *sy* represents the CPU occupied by kernel space;
- *ni* represents the CPU occupied by processes which adjust the prioritization;
- *id* represents the idle CPU (CPU's important parameters);
- *wa* represents the CPU occupied by IO waiting;
- *hi* represents the CPU occupied due to hardware interrupt;
- *si* represents the CPU occupied due to software interrupt.

Note: Amongst the relevant parameters of the 'top' command, this experiment focuses on the percentage of idle CPU.

In order to obtain valid test data, this experiment sets corresponding test parameter variables for encrypting and decrypting data. In the test script, there are seven IO block sizes (1K /2K /4K /8K /16K /32K /64K) and eight concurrency levels (1 /4 /8 /12 /16 /32 /48 /64). The single thread data is configured to 500M and the files created under the test directory are cleared each time to keep the testing data reliable. At the same time, the CPU utilization is measured with the 'top' command at a set interval of 0.25s during the test.

C. DATA ANALYSIS

1) READ AND WRITE OPERATIONS

After a large number of data testing and comparison, the sequential read and write operations of 12-process concurrency level were selected. Single process file size remains 500M. The value of the iops parameters is a measure of the read operation, while the bandwidth (*bw*) represents the metric for the write operation. See Figure 7. and Figure 8. for details.

As can be seen in Figure 7. and Figure 8., it can be seen that the QAT performance in the read operation is obviously insufficient compared to the AES-NI operation, with its CPU idle rate being basically zero. However, QAT shows higher performance during data writing and encryption, as well as some CPU idle resources. Changes in CPU idleness and encryption trends show that 4k is a turning point, because the eCryptfs file system uses 4k page size for encryption and decryption each time by default.

2) CONCURRENT LEVEL & BLOCK SIZE

Similarly, after mass data analysis, the sequential write operation of 4-process and 40-process concurrency levels is selected. The file size of each process is kept as 500M. See Figure 9. for details.

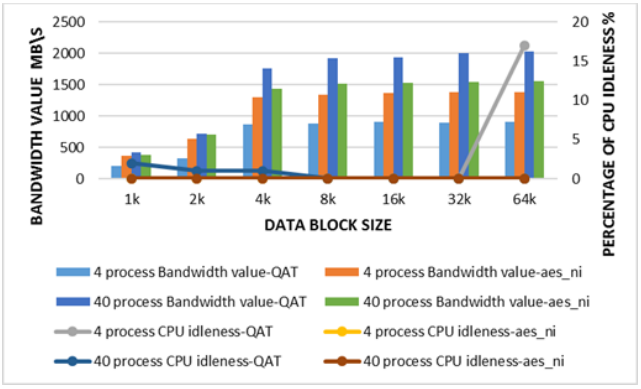


FIGURE 9. Sequential write with 4-process and 40-process concurrency level.

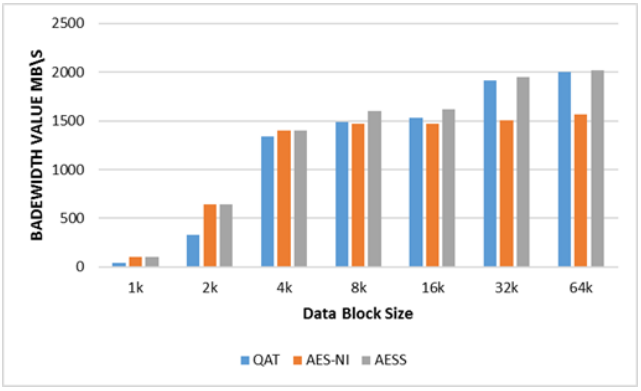


FIGURE 10. Comparison of performance with different schemes in 4-CPU situation.

By analyzing Figure 9., it can be concluded that the performance of instruction set encryption is better when the task concurrency level (the number of processes) is small and the data block size written each time is small. The obvious advantages of the QAT performance when the task concurrency level and the data block size written is larger each time.

In this section, we analyze the iops of reading and decrypted, *bw* of writing and encrypted, and the percentage of idle CPU (comparison between the instruction set and QAT). Based on the experimental data tested, in terms of writing and encryption, QAT provides greater performance improvement over the instruction set at large data blocks and high concurrent numbers, and leaves 8%-15% free CPU resources. In terms of reading and decryption, QAT has limited or no improvement in performance compared to the instruction set, and there are essentially no idle CPU resources.

The experimental results show that the scheme proposed in this paper is suitable and effective.

D. PERFORMANCE OPTIMIZATION ANALYSIS

As shown in Figure 10, this column comparison shows the ACA-SDS optimized performance data for 4 CPUs compared to QAT and AES-NI. During the entire request from a small block to a large block, ACA-SDS always maintains the highest bandwidth performance compared to QAT and AES-NI. Moreover, because of the use of the remaining CPU, the ACA-SDS also has improved performance in higher block encryption. In short, this method keeps the encryption of the entire system always maintaining the highest encryption performance.

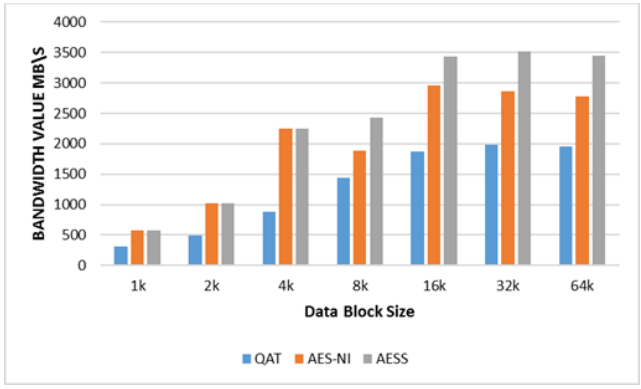


FIGURE 11. Comparison of performance with different schemes in Multi-CPU situation (8 CPUs).

As shown in Figure 11, the following optimization data can be obtained by multiple CPUs simultaneously scheduling hardware and software. When the data block is small and the bandwidth value is low, the encryption performance of ACA-SDS is basically the same as that of AES-NI. Afterwards, ACA-SDS has a performance improvement of around 15%-25% with large data blocks and higher bandwidth values. ACA-SDS has clear advantages compared with AES-NI or QAT when encrypting high-intensity data.

V. RELATED WORK

In the field of cryptography, researchers have proposed an encryption platform architecture that integrates security policy, traditional key management and public key management [8]. Some researchers introduced the eCryptfs stack file system into enterprise private cloud platforms and developed viable eCryptfs-based enterprise file encryption solutions for private clouds [2]; others have proposed an eCryptfs based multi-user encrypted file system to ensure that only users with legal keys can access ciphertext data [4], [9], [34], [38], [39].

In terms of file encryption, researchers have proposed a file-based information storage and translation are coordinated with packet alignment using the IRP (Input / output request packet) method, enabling the use of the versatile encryption and decryption filter driven algorithm [3]. Due to the high sensitivity of the chaotic system to the initial value, a method for encrypting the file using the state of the chaotic system as the key signal has been proposed [5]. Designing a multi-path network with the ability to calculate probability of attack enables the path encryption level to be proactively adjusted, it adjusts the encryption strength before the attack sequence completes its last step. By using statistical learning techniques, researchers have obtained data on the specific attack signatures for each network path and inferred suitable

information security level for each path through the analysis of these possibilities [10]. Certain researchers have worked on searchable encryption techniques for use in cloud computing. The paper proposes a hybrid model of searchable encryption and attribute-based encryption, which supports personalized and secure multi-user access to outsourcing data with high search performance [11]. In the field of improving encryption algorithms, researchers have proposed that a symmetric cipher algorithm based on polarization characteristics has been proposed to decompose the vector and the inner product, in which certain amount of computational complexity and security are guaranteed by a series of key combinations [12]. Some researchers introduce a new type of Identity-Based Encryption (IBE) scheme which is called Fuzzy Identity-Based Encryption [28]. And present a new methodology for realizing Ciphertext-Policy Attribute Encryption (CP-ABE) under concrete and noninteractive cryptographic assumptions in the standard model [29]. There are other aspects of encryption applications [30], [31], [33], [35]–[37].

In terms of hardware acceleration, researchers have proposed a genomic data processing involving CAST (Complexity Analysis of Sequence Tracts) algorithm based on parallel processed and hardware accelerated architecture, specially designed CAST accelerator architecture and proven FPGA prototypes, which is used in the complexity analysis of protein sequences encoded in genomic data [13]. Research on parallel algorithm-based processors and the development of a suitable processor in hardware has led to the development of a high-performance multi-core processor system for software implementation and a high-end programmable gate array system for hardware implementation [14]. The use of hardware acceleration technology has been proposed for use in radio resource scheduling in 5G mobile telecommunication systems. The acceleration technology multiplies the efficiency 60 times faster than the extant communication speed by calculating the throughput of each UE at the same time [15]. The experimenter also proposed other methods [21], [22], [24], [26].

In hardware-software co-design, there are collaborative design applications based on hardware and software platforms, which use a fast hardware-software platform to calculate irreducible tests [18]. Some researchers have proposed an effective and adjustable approach to hardware and software co-design in embedded software protection. By coupling protective compiler technology along with reconfigurable hardware support, it is possible to achieve a higher level of complexity compared with traditional mainstream hardware or software methods, within the range of safety performance [18]. Some researchers propose a CPU-based design scheme based on hardware / software collaboration along with a systematic method to compile and translate CPU core data path and control path through the instruction sequence described by the C language [20]. It plans to reduce the number of data transmission paths by selecting paths to bypass rarely used ones, and to select the optimal data path cost and control path cost among candidate CPU kernels [20]. Someone designed a humanoid robot platform - the iCub - to support collaborative research in cognitive development through autonomous exploration and social interaction [23]. Some researches build a persistent virtual visualization facility linked by ultra-high-speed optical networks to enable the comprehensive and synergistic research and development of the necessary hardware, software and interaction techniques [25]. Here are also some other aspects of the combination of hardware and software [27], [40], [41].

About research in QAT, a live migration scenario has been proposed that optimizes VNF with paravirtualization drivers and QAT technology. A solution called PV-QAT has been proposed to speed up the migration of common VNFs to prevent the degradation of VNF service caused by unavoidable service outages and the migration of large amounts of data over long periods of time [16]. PV-QAT uses paravirtualization (PV) drivers to filter out useless memory pages during migration and fast compression of low-overhead memory pages by using QAT technology (QAT), allowing PV-QAT to significantly reduce downtime by 77.5% and 80.5 % of the total migration time [16].

In terms of review, the papers discuss possible directions for computer architecture research and one or more examples of cross-layer research advocated, including architecture as infrastructure, energy first, impact of new technologies, and cross-layer opportunities [32].

Comparatively speaking, the research is comparatively rare on description and implementation of eCryptfs file system based on the combination of AES-NI and QAT. This highlights the practicality and value of this paper. Based on the research and exploration of the software and hardware co-design of eCryptfs file system, this paper proposes a working method that makes eCryptfs data encryption and decryption tasks more high-performance, more energy-efficient, and more valuable.

VI. CONCLUSION, LIMITATION, AND PROSPECTS

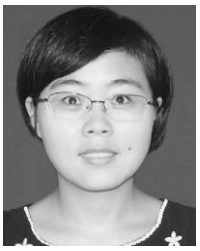
Based on a large amount of secure data storage, this paper proposes an ACA-SDS scheme based on read and write operations, different tasks and block sizes, and idle CPU resources. After this scheme, the paper further proposes to achieve the maximization of QAT efficiency gains by using the minimizing CPU resource management and simultaneous co-scheduling of software and hardware under multiple CPU conditions. Through the analysis of specific experimental data, it demonstrates the rationality of the designed program, and provides an effective software and hardware co-design method for enhancing the performance of data-intensive encryption. Through the simultaneous scheduling of hardware and software by the 2) Multi-CPU, ACA-SDS is able to get 15%-25% performance boost in intensive big data block encryption compared with the pure hardware and instruction set.

In fact, the default page data block in the eCryptfs file system is 4K when the data is encrypted by calling QAT and

AES-NI. For further optimization of the encryption performance, we can try to modify the source code of the eCryptfs file system so that the data blocks of encrypted pages are larger, such as 16K, 32K, 128K, 1M, 4M, 8M, 1G, and so on. This can increase the size of encrypted data blocks per page and reduce the number of CPU calls by hardware engines in data-intensive encryption requests.

REFERENCES

[1] L. Hao-Xiang and Q. Jun, "Detailed study of eCryptfs for the enterprise encryption file system," Tech. Rep., 2009, p. 21.
[2] X. Mao-zhi and L. Shuang, "High-performance network encryption platform," *Commun. Secur.*, no. 9, 2004.
[3] W. Quan-min, J. Hua-feng, W. Song, and Z. Li-yan, "File transparent encryption and decryption based on double-cache mechanism and its implementation," *Electron. Soft Sci.*, no. 5, 2011.
[4] Q. Xi-qin and C. Zhong-gui, "Research on encryption and decryption file system under NT architecture," *Human-Nature Sci. Stud.*, vol. 11, no. 2, 2012.
[5] D. Shao-Jiang and L. Xiao-Feng, "To realize text file encryption and decryption with chaotic system," *Comput. Sci.*, vol. 31, no. 8, 2004.
[6] L. Hai-Nan, X. Guo-Chun, and Z. Jian-Tao, "Hierarchical encrypted file system based on eCryptfs," *Comput. Eng. Des.*, Dec. 2016.
[7] X.-D. Tang, S.-L. Fu, and L.-Y. He, "Design and implementation of multi-user encryption file system based on eCryptfs," *J. Comput. Appl.*, vol. 30, no. 5, pp. 1236–1238, 2010.
[8] J.-H. Nah *et al.*, "T&I engine: Traversal and intersection engine for hardware accelerated ray tracing," *ACM Trans. Comput. Syst.*, vol. 30, no. 6, 2011, Art. no. 160.
[9] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 1592, J. Stern, Ed. Berlin, Germany: Springer-Verlag, May 1999, pp. 223–238.
[10] J. Obert, I. Pivkina, H. Huang, and H. Cao, "Proactively applied encryption in multipath networks," *Comput. Secur.*, vol. 58, pp. 106–124, May 2016.
[11] T. Bouabana-Tebibel and A. Kaci, "Parallel search over encrypted data under attribute based encryption on the cloud computing," *Comput. Secur.*, vol. 54, pp. 77–91, Oct. 2015.
[12] X. Xu and J. Feng, "Symmetric cryptographic algorithm based on polarization identity and implementation on file encryption," in *Proc. IEEE 3rd Int. Congr. Image Signal Process.*, vol 2, Oct. 2010, pp. 558–561.
[13] A. Papadopoulos, I. Kirmitzoglou, V. J. Promponas, and T. Theocharides, "FPGA-based hardware acceleration for local complexity analysis of massive genomic data," *Integr., VLSI J.*, vol. 46, no. 3, pp. 230–239, 2013.
[14] I. Damaj, M. Imdoukh, and R. Zantout, "Parallel hardware for faster morphological analysis," *J. King Saud Univ.-Comput. Inf. Sci.*, to be published.
[15] Y. Arikawa, T. Sakamoto, and S. Kimura, "Hardware acceleration technique for radio resource scheduling in 5G mobile systems," *Nippon Telegraph Telephone Corp*, vol. 15, no. 10, Oct. 2017.
[16] J. Zhang, L. Li, and D. Wang, "Optimizing VNF live migration via para-virtualization driver and QuickAssist technology," in *Proc. IEEE Int. Conf. Commun.*, May 2017, pp. 1–6.
[17] X. Shuai, L. Yao, and Z. Wang, "QAT: Evaluation of a dedicated hardware accelerator for high performance web service," in *Proc. 20th Int. Conf. Adv. Commun. Technol. (ICACT)*, Feb. 2008, p. 1.
[18] V. Rodríguez-Diez, J. F. Martínez-Trinidad, J. A. Carrasco-Ochoa, M. Lazo-Cortés, C. Feregrino-Uribe, and R. Cumplido, "A fast hardware software platform for computing irreducible testors," *Expert Syst. Appl.*, vol. 42, no. 24, pp. 9612–9619, 30 Dec. 2015.
[19] J. Zambreno, A. Choudhary, R. Simha, and B. Narahari, "Flexible software protection using hardware/software codesign techniques," in *Proc. Design, Automat. Test Eur. Conf. Exhib.*, 2004, p. 10636.
[20] Kunieda and Hiroaki, *CPU Core Generation for Hardware-Software Collaborative Design*. IEEE Press, 1996, pp. 306–309.
[21] R. Cowart, D. Coe, J. Kulick, and A. Milenkovi, "An implementation and experimental evaluation of hardware accelerated ciphers in all-programmable SoCs," in *Proc. South-East Conf.*, 2017, pp. 34–41.
[22] K. Romer and F. Mattern, "The design space of wireless sensor networks," *IEEE Wireless Commun.*, vol. 11, no. 6, pp. 54–61, Dec. 2004.
[23] G. Metta *et al.*, "The *iCub* humanoid robot: An open-systems platform for research in cognitive development," *Neural Netw.*, vol. 23, nos. 8–9, pp. 1125–1134, 2010.
[24] L. A. Rowe and J. Ramesh, "ACM SIGMM retreat report on future directions in multimedia research," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 1, no. 1, pp. 3–13, 2005.
[25] J. Leigh *et al.*, "The global lambda visualization facility: An international ultra-high-definition wide-area visualization collaboratory," *Future Gener. Comput. Syst.*, vol. 22, no. 8, pp. 964–971, 2006.
[26] S. Nooshabadi and J. Garside, "Modernization of teaching in embedded systems design-an international collaborative project," *IEEE Trans. Educ.*, vol. 49, no. 2, pp. 254–262, May 2006.
[27] J. C. Phillips *et al.*, "Scalable molecular dynamics with NAMD," *J. Comput. Chem.*, vol. 26, no. 16, pp. 1781–1802, 2005.
[28] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 3494, R. Cramer, Ed. Heidelberg, Germany: Springer-Verlag, 2005, pp. 457–473.
[29] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Public Key Cryptography—PKC* (Lecture Notes in Computer Science), vol. 6571, D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, Eds. Heidelberg, Germany: Springer, 2011, pp. 53–70.
[30] H. Cheng and X. Li, "Partial encryption of compressed images and videos," *IEEE Trans. Signal Process.*, vol. 48, no. 8, pp. 2439–2451, Aug. 2000.
[31] A. N. Khan, M. L. M. Kiah, M. Ali, S. Shamshirband, and A. U. R. Khan, "A cloud-manager-based re-encryption scheme for mobile users in cloud environment: A hybrid approach," *J. Grid Comput.*, vol. 13, no. 4, pp. 651–675, Dec. 2015.
[32] M. D. Hill, "21st century computer architecture," *ACM SIGPLAN Notices*, vol. 49, no. 8, pp. 1–2, 2014.
[33] J. Shu, Z. Shen, and W. Xue, "Shield: A stackable secure storage system for file sharing in public storage," *J. Parallel Distrib. Comput.*, vol. 74, no. 9, pp. 2872–2883, Sep. 2014.
[34] A. Khoje, K. A. Salih, and R. Moona, "TransCrypt: An enterprise encrypting file system over NFS," in *Proc. World Congr. Eng.*, 2009, pp. 1–6.
[35] A. Adya *et al.*, "Farsite: Federated, available, and reliable storage for an incompletely trusted environment," in *Proc. 5th Symp. Oper. Syst. Design Implement. (OSDI)*, 2002, pp. 1–14.
[36] M. Blaze, "A Cryptographic File System for UNIX," in *Proc. 1st ACM Conf. Comput. Commun. Secur.*, 1993, pp. 9–16.
[37] G. Cattaneo, L. Catuogno, A. Del Sorbo, and P. Persiano, "The design and implementation of a transparent cryptographic file system for UNIX," in *Proc. USENIX Annu. Tech. Conf.*, 2001, pp. 199–212.
[38] T. Dillon, C. Wu, and E. Chang, "Cloud computing: Issues and challenges," in *Proc. Int. Conf. Adv. Inf. Netw. Appl.*, Apr. 2010, pp. 27–33.
[39] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, "Plutus: Scalable secure file sharing on untrusted storage," in *Proc. 2nd USENIX Conf. File Storage Technol. (Fast)*, 2003, pp. 29–42.
[40] J. Coburn, J. L. Salmon, and I. Freeman, "Effectiveness of an immersive virtual environment for collaboration with gesture support using low-cost hardware," *J. Mech. Des.*, vol. 140, no. 4, p. 042001, 2018.
[41] J. Shalf, D. Quinlan, and C. Janssen, "Rethinking hardware-software codesign for exascale systems," *Computer*, vol. 44, no. 11, pp. 22–30, Nov. 2011.



CHUNHUA XIAO was born in 1987. She received the Ph.D. degree from the Beijing University of Technology, China. She did one year academic research as a joint-training Ph.D. student in the University of California at Los Angeles, Los Angeles, from 2011 to 2012. She is currently an Associate Professor with the School of Computer Science, Chongqing University, China.

He has authored and co-authored over 30 publications in peer-reviewed journals and conferences. Her research interests include MPSoCs, hardware and software codesign, and embedded systems. She has been an independent PI for a standard (2016–2018) National Nature Science Foundation of China grant, and also an independent PI (2016–2017) for a Crossing Research Projects with Huawei Technologies Co., Ltd. She was honored with the Science and Technology Progress Award from Beijing municipality in 2012.

PENGDA LI received the B.E. degree from the School of Information Engineering, Xiangtan University, Hunan, China, in 2017. He is currently pursuing the master's degree under the supervision of Dr. C. Xiao with the School of Computer Science, Chongqing University, China. His main research direction includes Internet of Things, hardware and software co-design, and embedded systems.

LEI ZHANG received the B.E. degree from the School of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing, China, in 2016, where she is currently pursuing the master's degree under the supervision of Dr. C. Xiao. Her current research interests include hardware security, hardware and software co-design, and energy-efficient computing and applications.

WEICHEN LIU (S'07–M'11) received the B.Eng. and M.Eng. degrees from the Harbin Institute of Technology, China, and the Ph.D. degree from The Hong Kong University of Science and Technology, Hong Kong. He is currently an Assistant Professor with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. He has authored and co-authored over 70 publications in peer-reviewed journals, conferences and books. His research interests include embedded and real-time systems, multiprocessor systems, and network-on-chip. He received the best paper candidate awards from ASP-DAC 2016, CASES 2015, CODES+ISSS 2009, the best poster awards from RTCSA 2017, AMD-TFE 2010, and the most popular poster award from ASP-DAC 2017.

NEIL BERGMANN (M'86) received the bachelor's degree in engineering, science and arts from The University of Queensland and the Ph.D. degree in computer science from the University of Edinburgh, U.K., in 1984. He has been a Professor of embedded systems with the School of ITEE, The University of Queensland, since 2001. His research interests are in computer systems, especially reconfigurable computing and wireless sensor networks. He is a fellow of the Institution of Engineers, Australia.