SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

April 2018, Vol. 61 042102:1–042102:11
https://doi.org/10.1007/s11432-017-9221-0

Customizing the HPL for China accelerator

Xinbiao GAN[1,2,3*], Yikun HU[4], Jie LIU[1], Lihua CHI[5], Han XU[1], Chunye GONG[1], Shengguo LI[1] & Yihui YAN[1]

[1]*School of Computer, National University of Defense Technology, Changsha* 410073*, China;*
[2]*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing* 210093*, China;*
[3]*State Key Laboratory of Space Weather, Chinese Academy of Sciences, Beijing* 100190*, China;*
[4]*College of Information Science and Engineering, Hunan University, Changsha* 410073*, China;*
[5]*Institutes of Advanced Science and Technology, Hunan Institute of Traffic Engineering, Hengyang* 414600*, China*

Received 17 January 2017/Revised 24 July 2017/Accepted 29 August 2017/Published online 6 March 2018

Abstract HPL is a Linpack benchmark package widely used in high-performance computing tests. Customizing the HPL is crucial for a heterogeneous system equipped with CPU and the China accelerator because of the complexity of the China accelerator and the specified interface on matrix multiplication built in the China accelerator. Therefore, it is advisable to use delicate partition and encapsulation on matrix (DPEM) to expose a friendly testing configuration. More importantly, we propose the orchestrating algorithm for matrix multiplication (OAMM) to enhance the efficiency of the heterogeneous system composed of CPU and China accelerator. Furthermore, optimization at vectorization (OPTVEC) is applied to shield the architectural details of the vector processing element (VPE) equipped in the China accelerator. The experimental results validate DPEM, OPTVEC and OAMM. OPTVEC optimizations would speed up matrix multiplication more than twofold, moreover OAMM would improve productivity by up to 10% compared to the traditional HPL tested in a heterogeneous system.

Keywords HPL, China accelerator, DPEM, OAMM, OPTVEC

Citation Gan X B, Hu Y K, Liu J, et al. Customizing the HPL for the China accelerator. Sci China Inf Sci, 2018, 61(4): 042102, https://doi.org/10.1007/s11432-017-9221-0

1 Introduction

High-performance computing is an important indicator of the progress made by national science and technology and has been widely applied to numerical calculation, weapons, and equipment simulation. The China accelerator is a high-performance domestic accelerator that will be equipped in Tianhe-2 for updation [1].

Linpack (linear system package) is used to evaluate the floating point performance measured in a high-performance computer by solving a one-variable linear system of order N using Gauss elimination. Linpack is divided into Linpack100, Linpack1000, and high-performance Linpack (HPL) based on problem size and optimal policy [2]. The HPL is popularly used in testing the high-performance computer floating-point performance because of the variable problem size.

Given that the problem size is N, the number of floating point calculation would be $\frac{2}{3}N^3 + \frac{2}{3}N^2$, and the calculation time is t. Hence, the measured floating-point performance should be $(\frac{2}{3}N^3 + \frac{2}{3}N^2)/t$, which is a crucial basis for ranking within the top 500.

* Corresponding author (email: xinbiaogan@163.com)

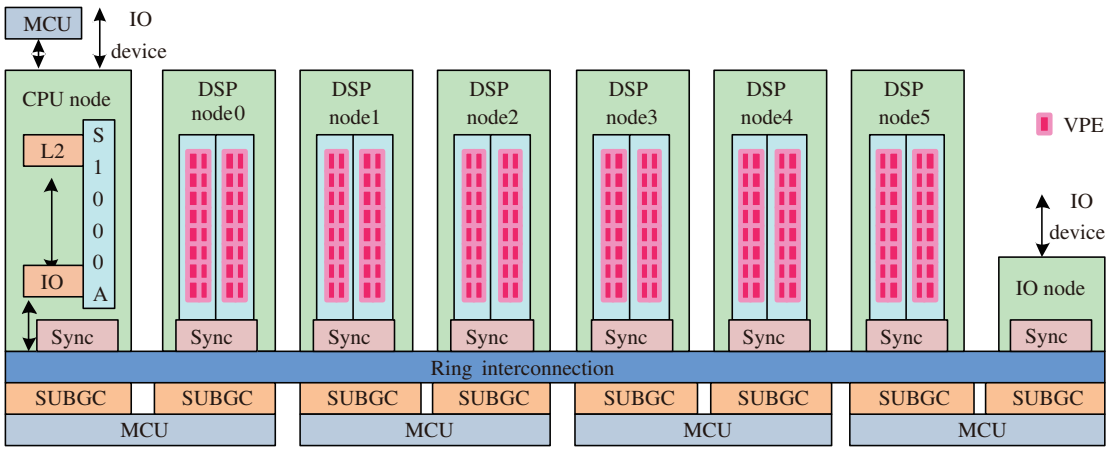

Figure 1 (Color online) Architecture of the China accelerator.

DPEM (delicate partition and encapsulation on matrix) is recommended to expose a friendly testing configuration for HPL because of the specified interface on the matrix multiplication built in the China accelerator. In addition, we propose the OAMM (orchestrating algorithm for matrix multiplication) to enhance the efficiency of a heterogeneous system composed of a CPU and a China accelerator. Furthermore, OPTVEC (optimization at vectorization) is advised to shield the architectural details of the VPE (vector processing element) equipped in the China accelerator.

2 Orchestrating HPL between the CPU and the China accelerator

Traditionally, matrix multiplication in HPL would be divided and distributed into all CPUs [3–4]. However, this method is inefficient for a heterogeneous system equipped with a CPU and a China accelerator. To increase the speed of HPL, customizing the HPL for a heterogeneous system equipped with a CPU and the China accelerator is advisable.

No unnecessary communication exits when T_{cpu} and $T_{\text{China-accelerator}}$ satisfy (1) as follows:

$$\begin{cases} T_{\text{cpu}}(M, K, N_2) = T_{\text{China-accelerator}}(M, K, N_1), \\ N_2 + N_1 = N, \end{cases} \tag{1}$$

where $A(M, K)$, $B(K, N)$, $C(M, N)$, and T_{cpu} as well as $T_{\text{China-accelerator}}$ denote updating time for matrix multiplication in the CPU and the China accelerator, respectively.

The result matrix $C(M, N)$ would be divided into CPU(M, K, N_2) in the CPU and the China-accelerator(M, K, N_1) for the China accelerator, respectively, to avoid unnecessary waiting between the CPU and the China accelerator, then reduced into $C(M, N)$. Moreover, the size of N_1, N_2 should satisfy $N_1 + N_2 = N$, and the CPU and the China accelerator would simultaneously finish the matrix multiplication.

2.1 DPEM for the China accelerator

The China accelerator is a self-controlled high-performance accelerator created by the National University of Defense Technology of China for counterattacking Chip-Restricted Order from the USA. The China accelerator is a high-performance accelerator that would simultaneously operate on a large quantity of data in the VPE (Figure 1).

As illustrated in Figure 1, the China accelerator consists of one CPU node, six DSP nodes, one IO node, a global cache (GC) partitioned into each node, and four memory control units (MCU). All nodes are connected by ring interconnection. Each DSP node is composed of two computing core, one SUB-global cache (SUBGC), and Sync between GC for synchronization. Each computing core contains 16 VPEs. Every two SUBGCs are connected with one MCU.

A specified matrix multiplication interface, such as $A(\mathrm{FT}\_m, K) \times B(K, N)$, is advised to speed up the HPL testing and maximize the efficiency of the China accelerator, in which FT_m must be a multiple of 576 and greater than or equal to 576×6. K and N are also natural numbers. However, the specified interface could result in an unfriendly testing configuration for the HPL. Hence, the DPEM is advised to expose a friendly testing configuration for the HPL. The DPEM would encapsulate a specified matrix multiplication interface into an ordinary matrix multiplication interface, such as $A(M, K) \times B(K, N)$, in which M, K and N denotes arbitrary positive integers.

Accordingly, matrix multiplication would be delicately partitioned and distributed into a heterogeneous system composed of the CPU and the China accelerator using the DPEM. More importantly, the OAMM is proposed to enhance the efficiency of the heterogeneous system composed of the CPU and the China accelerator.

2.2 OAMM between CPU and the China accelerator

Two kinds of dispatching algorithm on matrix multiplication acceleration are used to speed up the heterogeneous HPL, namely the static dispatch strategy (SDS) and the dynamic schedule dispatch (DSD).

SDS has been employed in the Tianhe-1A heterogeneous supercomputer equipped with a CPU and a GPU [5–6]. In SDS, matrix scheduling should be given priority and would explore the optimal matrix division ratio for the GPU accelerator, as shown in

$$k = \frac{T_{\mathrm{gpu}}}{T_{\mathrm{cpu}} + T_{\mathrm{gpu}}} \times 100\%, \tag{2}$$

where T_{gpu} and T_{cpu} represent updating time for matrix multiplication in the CPU and the GPU, respectively.

The sub-matrix multiplication in Tianhe-1A is decided in advance according to (2), in which the data transfer between the CPU and the GPU would be overlapped with matrix multiplication to minimize communication and maximize the efficiency of Tianhe-1A [6–8]. Similarly, meaningful studies are being referred to herein on the parallelization and optimizations for the GPU [9–14].

Different from the Tianhe-1A supercomputer, the DSD based on the queue buffer has been successfully applied to Tianhe-2, which is composed of CPU and MIC [15–18]. In DSD, matrix multiplication distributed into the CPU or MIC would depend on the status of the computing mission queue.

However, the China accelerator is different from traditional accelerators, such as GPU and MIC. Hence, neither SDS nor DSD is suitable for the heterogeneous system equipped with CPU and the China accelerator. The OAMM is, therefore, proposed to orchestrate the matrix multiplication between the CPU and the China accelerator and speed up the HPL running on the CPU and the China accelerator with coordination.

The OAMM would take advantage of the SDS and the DSD, and maximize the efficiency of the China accelerator to speed up the HPL. Theoretically, matrix multiplication in OAMM is first divided according to the specified interface on matrix multiplication built in the China accelerator (Figure 2), queued, then dynamically deployed into the CPU or the China accelerator according to the character of the deployed matrix multiplication and the status of the computing mission queue, as listed in Algorithm 1.

3 OPTVEC based on the China accelerator

Programming the China accelerator is challenging, especially when taking full advantage of the VPE equipped in the China accelerator. Accordingly, the OPTVEC is employed to shield the details in the VPE and unwittingly simplify the vector optimization. The OPTVEC includes two kinds of techniques, namely recuperative loop unrolling (RLU) and packing assembler for vectorization (PASMVEC).

The OPTVEC is a theoretical foundation for vector optimization on the VPE equipped in the China accelerator. The RLU is a basis on encapsulating vector assembly instructions in the PASMVEC, while optimization evaluation profiling from the PASMVEC would instruct the RLU with feedback information.

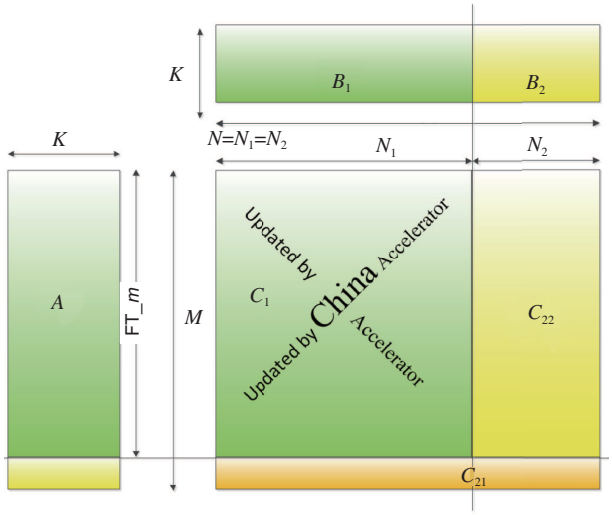

Figure 2 (Color online) Matrix multiplication updated by the CPU and the China accelerator (FT_m must be multiple of 576 and greater than or equal to 576×6, and M, K, N denote arbitrary positive integers).

Algorithm 1 Pseudocode on OAMM

```
Init CQ for CPU queue;
Init AQ for China accelerator queue;
Set current matrix multiplication A(M, K) × B(K, N);
Get architecture parameter FT_m;
Divide B(K, N) into B1(K, N1) + B2(K, N2) reference to (1) and (2);
if M >FT_m then
    divide A(M, K) into A1(FT_m, K) + A2(M−FT_m, K);
else
    queue A × B into CQ;
end if
//queue A1 × B1;
if (AQ is not full) then
    AQ ⇐ A1 × B1;// queue AQ;
else if (CQ is empty) then
    AQ ⇐ A1 × B1;// queue CQ;
    end if
else
    waiting;
end if
// queue A1 × B2;
if (CQ is not full) then
    AQ ⇐ A1 × B2;// queue CQ;
else
    waiting;
end if
Set A2 × B into current matrix multiplication A × B;
goto step 6;
// matrix multiplication updating;
while (AQ is not empty) do
    get head from AQ and then call specified interface on matrix multiplication built in the China accelerator;
end while
while (CQ is not empty) do
    get head from CQ and then call ordinary interface on matrix multiplication on CPU;
end while
```

3.1 Recuperative loop unrolling

Loops might usually take up the majority of the time of the entire running program. Hence, automatic loop unrolling is advised to reduce the running time and guide further optimizations, including

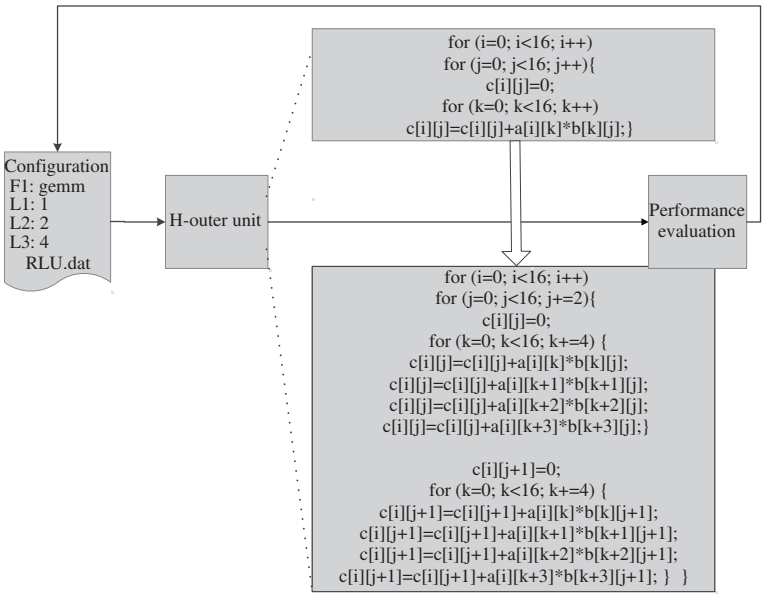

Figure 3 Structure of the RLU.

vectorization [19–22].

The RLU is a configurable automatic loop unrolling technique, including homo-polar INNER (HPINNER) and hierarchy OUTER (HOUTER). The HPINNER is a traditional loop unrolling technique applied to classical compiler optimization, which acts on the most inner loop or single level loop body. The HOUTER is an updatation of the HPINNER and would support multi-level loop containing both inner and outer loop bodies.

In the HOUTER, if loop body L_l in function F_f might be unrolled U_u times (e.g., triple nested loop body in matrix multiplication function F_1), the most outer loop L_1 might be unrolled for $U_1 = 1$ times. The second outer loop L_2 might be unrolled for $U_2 = 2$ times, and the most inner loop L_3 might be unrolled for $U_3 = 4$ times. The HOUTER should obtain a pre-configuration and collect profiling from performance evaluation to adjust the loop unrolling frequency for probing optimal configuration (Figure 3).

In practice, all features and acts included in the HPINNER are built in the HOUTER. The HPINNER could be deemed as an instance or sub-set of the HOUTER.

3.2 PASMVEC based on the China accelerator

The OPTVEC is proposed and pre-compiled as a milestone component to simplify the optimizations on the China accelerator and vectorize a continuous vector array with minimum architectural details on the VPE. In OPTVEC, the RLU is a basis of PASMVEC, and PASMVEC is a further optimization on the China accelerator. The HPL running on the China accelerator would be transformed and pruned by the RLU based on the OPTVEC. The PASMVEC would then automatically encapsulate the vector assembly instruction for further optimizations.

The PASMVEC would load a continuous vector array, then pack the assembler with vector data for vectorization. The PASMVEC focuses on a multiple-level nested loop in matrix multiplication.

The PASMVEC would collect the loop characteristic, analyze data dependence from the innermost to the outermost in the compound loop, and then automatically pack the assembler with a continuous vector array for vectorization. At the same time, PASMVEC would quantitate the effect of a specified level on vectorization for instructing and adjusting the RLU.

Taking triple nested loops in matrix multiplication as an instance, traditional compiler optimization could support automatic PASMVEC for HPINNER, in which PASMVEC would pack the assembler with a continuous vector array based on the loop invariant k, as demonstrated in Figure 4(a). Different from HPINNER, PASMVEC for HOUTER would pack the assembler with a continuous vector array based

```
for (i=0; i<M; i++)
for (j=0; j<M; j++){
    c[i][j]=0;
    for (k=0; k<M; k++)
c[i][j]=c[i][j]+a[i][k]*b[k][j];}
```

```
for (i=0; i<M; i++){
for (j=0; j<M; j++){
      vsum=0;
   for (k=0; k<M; k+=2){
      //load
asm("VLDW *a[i][k], VR1"
    : "=VR1"(v1));
      //store
asm("VSTW VR2, *b[k][j]"
    :"=VR2"(v2));
      vsum+=v1*v2;}
   vstore(vsum, sum[0]);
 c[i][j]=sum[0]+sum[1];}
```

(a)

```
for (i=0; i<M; i++){
for (j=0; j<M; j+=2){
      vsum={0, 0};
   for (k=0; k<M; k+=2){
asm("VLDW *b[k][j], VR1"
    :"=VR0"(va));
asm(" VLDW *a[k][j], VR0"
    :"=VR2"(v2));
   asm(" VADDW VR1, VR0, VR2"
    :"=VR1"(v1));
      vsum+=v2*va;
asm(" VADDW VR0, VR0, VR2"
    :"=VR0"(v0));
      vsum+=v0;}
   asm("VSTW VR0, *c[i][j]"
    :"=VR0"(v0));
      vsum=v0;}
```

(b)

Figure 4 Different strategies on the PASMVEC: (a) HPINNER; (b) HOUTER.

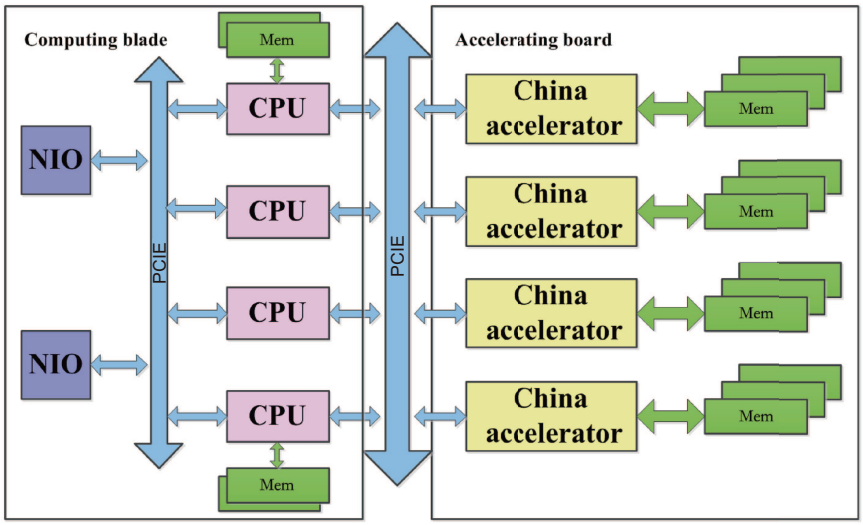

Figure 5 (Color online) Architecture of a computing node.

on the loop invariant k, j and i respectively, and obtain higher speed-up than that of HPINNER, as demonstrated in Figure 4(b).

The PASMVEC on the HOUTER could gain a higher benefit than that of the HPINNER for compound loops, especially for triple nested loops in matrix multiplication, because no dependencies can be found for vectorization based on loop invariant$j$, $a[i][k]$ access is discontinuous for the j index. In addition, there are dependencies for vectorization based on the loop invariant k. Hence, introducing special transformation and redundant operations for vectorization based on loop invariant k is unavoidable. Therefore, it is inadvisable to vectorize based on loop invariant i, j, k, respectively. However, there are unexpected factors in vectorization based on loop invariant j along with k using HOUTER model.

4 Experiments and analysis

4.1 Validation system based on the China accelerator

A validation system has been built with an array of computing nodes (CN), and each CN contains a computing blade, an accelerating board, and a peripheral component interconnect express (PCIE) bus, as demonstrated in Figure 5. In the validation system, the computing blade and accelerating board could be attached and detached on demand. There are four China accelerators connected with the computing

Table 1 Details on computing node

System	Component	Attribute	Number
Hardware	CPU	Intel(R) Xeon(R) CPU E5-2692 v2 @ 2.20 GHz	4
	The China accelerator	FT-GPDSP 2000b @1.25 GHz	4
	Memory sub-system	8 X Samsung 8 G DDR3 1333 MHz	4
Software	OS	Linux kylin-phytium+	—
	Compiler	Lintel icc 15.0.0 + Phytium Compiler	—
	BLAS	MKL + FTBLAS	—

Table 2 The HPL testing on DPEM

Parameter	Accelerated by China accelerator without DPEM	Accelerated by China accelerator with DPEM	Comments
$N < 3456$	×	×	The HPL running on CPU only because of N less than 576×6
$N = 3456$	√	√	The HPL running on both CPU and China accelerator with coordination
$N > 3456$ && $N\%576 \doteq 0$	√	√	The HPL running on both CPU and China accelerator with coordination
$N > 3456$ && $N\%576 \neq 0$	×	√	The HPL would run both CPU and China accelerator with coordination assisted by DPEM

blade using PCIE in an accelerating board, and there are four CPUs in a computing blade interconnected with high express intranet by network input and output (NIO) card.

Based on the validation system, DPEM, OAMM and OPTVEC including LRU and PASMVEC are validated by customizing the HPL for the China accelerator.

Experimentally, the whole validation system is composed of 16 CNs, and details on figuration of CN are listed in Table 1.

4.2 Validation on DPEM

A specified matrix multiplication interface, such as $A(\mathrm{FT}\_m, K) \times B(K, N)$, is required to call by the HPL accelerated using the China accelerator because of the specified matrix multiplication interface built in the accelerator, in which FT_m is an integer, must be multiple of 576, and greater than or equal to 576×6; and K and N are natural numbers, which result in an unfriendly testing configuration for the HPL.

The DPEM proposed would encapsulate the specified matrix multiplication interface into an ordinary interface, such as $A(M, K) \times B(K, N)$, in which M, K, and N are arbitrary positive integers, which would transform an unfriendly testing configuration into an ordinary one.

Table 2 lists the HPL testing comparisons on the DPEM. Accelerating the HPL using the China accelerator would be impossible when the HPL input parameter N is less than $576 \times 6 = 3456$. Unfortunately, even though N was greater than or equal to 3456, the HPL was still barely accelerated by the China accelerator without the DPEM because of the strong requirements of N being greater than or equal to 3456 and N being also in multiple of 576. However, with the DPEM, testers could test the HPL as usual, and the HPL would then be accelerated by the China accelerator when N is greater than or equal to 3456, which is a rather weaker requirement for acceleration using the China accelerator compared to that of a non-DPEM.

4.3 Performance on the matrix transferred

A specified matrix multiplication interface, such as $A(\mathrm{FT}\_m, K) \times B(K, N)$, is advised to speed up the HPL testing and maximize the efficiency of the China accelerator, in which FT_m must be a multiple of 576 and greater than or equal to 576×6. In practice, the size of the matrix transferred would affect

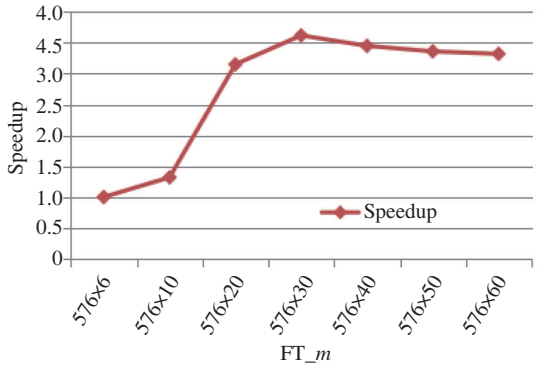

Figure 6 (Color online) Performance comparisons on the matrix transferred.

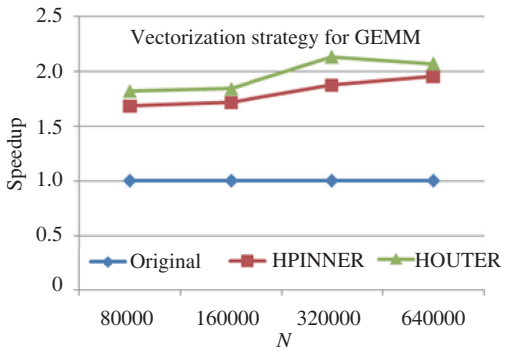

Figure 7 (Color online) Performance comparisons on the OPTVEC.

the efficiency of the matrix multiplication in the HPL. Hence, the size of the matrix transferred, then accelerated is evaluated as shown in Figure 6.

Aside from its unfriendly configuration, the architectural parameter FT_m in the specified matrix multiplication interface could affect the HPL testing performance. The transferred matrix size with 576×6 was marked as a reference point to probe an optimal FT_m. Moreover, different sizes of the matrix transferred, then accelerated by the China accelerator were experimented and compared to the reference point (Figure 6). The speedup of the matrix multiplication operations per second increased as the size of matrix transferred increased from 576×6 to 576×10, Furthermore, the speedup sharply increased as the size of the matrix transferred increased from 576×10 to 576×20. Meanwhile, the speedup increased at a peak point as the size of the matrix transferred increased from 576×20 to 576×30. The speedup on the matrix multiplication operations per second slowly descended as the size of the matrix transferred further increased because the PCIE bandwidth of the data transferred was crucial for the performance when the size of the matrix transferred increased from 576×6 to 576×30. The PCIE bandwidth of the data transferred would be fully utilized as the size of the matrix transferred increased. However, the performance bottleneck was changed from the PCIE bandwidth to the size of a GC when the size of the matrix transferred increased from 576×30. Unfortunately, the GC could not afford the huge data volume of the matrix as the size of the matrix transferred increased. Hence, the speedup on the matrix multiplication operations per second should descend accordingly.

4.4 Performance analysis on the OPTVEC

The OPTVEC optimizations are the key techniques for the HPL running on both the CPU and the China accelerator with coordination. The RLU is a basis for PASMVEC, and the PASMVEC would instruct the RLU with feedback information on the cost and benefit of the general matrix-multiplication (GEMM).

Theoretically, loop unrolling in the RLU could reduce branches and improve the performance. The China accelerator is a high-performance processor and an array of VPE equipped to accelerate computations and operations can be used. Hence, the focus of the RLU in the OPTVEC for the China accelerator is to bring about an optimization space for further PASMVEC optimizations when taking advantages of the VPE equipped in the China accelerator. Figure 7 shows the performance comparisons on the OPTVEC strategy.

As demonstrated in Figure 7, the original version of the matrix multiplication without the OPTVEC is marked as the baseline. The proposed OPTVEC strategies, including the HOUTER and the HPINNER optimizations on matrix multiplication, were compared to the baseline. The HOUTER and HPINNER versions of the matrix multiplication with the OPTVEC would promote the efficiency of the matrix multiplication operations per second. Moreover, the speedup for the HOUTER version of the matrix multiplication with the OPTVEC is higher compared to the HPINNER version.

With regard to the HPINNER version, the speedup of the matrix multiplication steadily increased when the size of the matrix was increased from 80000 to 160000. Meanwhile, the speedup of the matrix

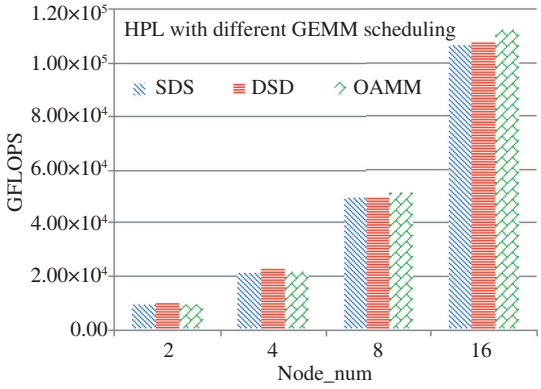

Figure 8 (Color online) Performance comparisons on OAMM and SDS/DSD.

multiplication sharply increased when the size of the matrix was increased from 160000 to 320000. The speedup slowly increased when the size of the matrix was increased from 320000 to 640000.

For the HOUTER version, the speedup steadily increased similar with the HPINNER when the size of the matrix was expanded from 80000 to 160000. Different from the HPINNER version, the speedup of the matrix multiplication sharply increased and reached a peak point when the size of the matrix was expanded from 160000 to 320000. However, the speedup did not increase when the size of the matrix was increased from 320000 to 640000 because the registers and the VPE were not able to afford the variable duplication for an automatic vectorization in the PASMVEC.

4.5 Performace evaluation on the HPL

In practice, scheduling strategies on the matrix multiplication play an important role on improving the performance of the HPL running on a heterogeneous system. Two famous scheduling strategies are used on matrix multiplication, namely SDS and DSD. SDS has been employed in the Tianhe-1A heterogeneous supercomputer equipped with CPU and GPU, while DSD has been successfully applied to the Tianhe-2 composed of CPU and MIC. However, either SDS nor DSD is suitable for the HPL running on a heterogeneous system composed of a CPU and a China accelerator because of the complexity and the specificity of the China accelerator. Instead of the DSD and the SDS, the OAMM is proposed to orchestrate the matrix multiplication between the CPU and the China accelerator with coordination.

Strategies, including SDS and DSD, are configured to compare with OAMM, speed up the HPL running on the heterogeneous system composed of the CPU and the China accelerator with coordination, and validate OAMM. Therefore, the HPL running on the heterogeneous system equipped with the CPU and the China accelerator could be configured and evaluated with SDS, DSD, and OAMM independently.

Figure 8 shows the measured performance on the scheduling strategies, including SDS, DSD, and OAMM on matrix multiplication. The testing performances were measured with the optimal matrix transferred and the OPTVEC optimizations discussed earlier.

Figure 8 presents that the proposed OAMM was clearly superior to both SDS and DSD when the HPL runs on the entire validation system with 16 CNs. Meanwhile, the performance of the SDS on the HPL running only on eight CNs was close to DSD, and OAMM was slightly better than DSD. However, the DSD was the best when the HPL run with two or four CNs, and SDS was close to OAMM. Therefore, the OAMM would scale to a huge heterogeneous system, especially for a heterogeneous system composed of a CPU and a China accelerator. Moreover, the OAMM would achieve an expected HPL performance, while the SDS and the DSD are secondarily advisable for the HPL running on a heterogeneous system compared to the OAMM. The DSD is only slightly better than the SDS.

In conclusion, the OAMM is the best scheduling strategy on matrix multiplication in the HPL running on the heterogeneous system composed of the CPU and the China accelerator with coordination.

5 Conclusion

The DPEM, OPTVEC and OAMM are detailed for the China accelerator without repetition because of the HPL configuration optimizations studied in many references.

The proposed DPEM would encapsulate a specified matrix multiplication interface into an ordinary interface and transform an unfriendly testing configuration into an ordinary one. The HPL testing experiences validated that the DPEM could shield the details of the specified matrix multiplication interface and transform an unfriendly testing configuration into a friendly one.

The OPTVEC optimizations, including RLU and PASMVEC, are the key techniques for the HPL running on both the CPU and the China accelerator with coordination. The RLU is a configurable automatic loop unrolling technique, including HPINNER and HOUTER. The HOUTER is an updatation of the HPINNER. The PASMVEC would load a continuous vector array, then pack the assembler with vector data for vectorization after the RLU. Two models were built in the PASMVEC to support the HOUTER and the HPINNER according to the RLU. The experimental results verified that the OPTVEC optimizations would simplify the China accelerator programming and sharply improve the HPL performance.

Scheduling strategies on matrix multiplication are very important in improving the performance of the HPL running on the heterogeneous system. The SDS has been employed in Tianhe-1A, while DSD has been successfully applied to Tianhe-2. However, they are unadvisable for a heterogeneous system equipped with a CPU and a China accelerator. Hence, instead of the SDS and the DSD, the OAMM is proposed to orchestrate matrix multiplication between the CPU and the China accelerator with coordination. The experimental results validated that the OAMM is the best scheduling strategy on matrix multiplication for the HPL running on a heterogeneous system composed of a CPU and a China accelerator with coordination.

Acknowledgements This work was partly supported by National Natural Science Foundation of China (Grant Nos. 61602495, 61402039, 91430218, 9130324, 11401580), Key Research and Development Program (Grant Nos. 2017YFB0202104, 2016YFB200401), Innovation Program from the National University of Defense Technology (Grant No. ZK16-03-06), partly supported by Specialized Research Fund for State Key Laboratories of Space Weather, Chinese Academy of Sciences, and partly supported by Open Research Fund of Key Laboratory of Spectral Imaging Technology, Chinese Academy of Sciences (Grant No. LIST201602D).

References

1 Lu Y T. The applications leveraging supercomputing systems. In: International Supercomputing Conference, Frankfurt, 2015
2 Dongarra J J, Luszczek P, Petitet A. The LINPACK benchmark: past, present and future. Concurr Computat-Pract Exper, 2003, 15: 803–820
3 Shi R, Potluri S, Hamidouche K, et al. A scalable and portable approach to accelerate hybrid the HPL on heterogeneous CPU-GPU clusters. In: Proceedings of IEEE International Conference on Cluster Computing (CLUSTER). Indianapolis: IEEE, 2014. 1–8
4 Wang Q, Ohmura J, Axida S, et al. Parallel matrix-matrix multiplication based on the HPL with a GPU-accelerated PC cluster. In: Proceedings of the International Conference on Networking and Computing. Higashi-Hiroshima: IEEE, 2010. 243–248
5 Yang X J, Liao X, Lu K, et al. The TianHe 1 a supercomputer, its hardware and software. J Comput Sci Tech, 2011, 26: 344–351
6 Du Y F, Yang C Q, Wang F, et al. Analysis and evaluation method for the Linpack benchmark. J Northeast Univ Nat Sci, 2014, 35: 102–107
7 Liu J, Gan X B, Chi L H, et al. A peak performance model for matrix multiplication on general-purpose DSP (in Chinese). J Hunan Univ Nat Sci, 2013, 40: 148–152
8 Chi L H, Liu J, Yan Y H, et al. FitenBLAS: high-performance BLAS for a massively multithreaded FT1000 processor (in Chinese). J Hunan Univ Nat Sci, 2015, 42: 100–106
9 Gong C Y, Bao W M, Tang G J, et al. An efficient parallel solution for Caputo fractional reaction-diffusion equation. J Supercomputing, 2014, 68: 1521–1537
10 Gong C, Bao W, Tang G. A parallel algorithm for the Riesz fractional reaction-diffusion equation with explicit finite difference method. Fract Calc Appl Anal, 2013, 16: 654–669

11 Gong C Y, Liu J, Chi L H, et al. GPU accelerated simulations of 3D deterministic particle transport using discrete ordinates method. J Comput Phys, 2011, 230: 6010–6022

12 Zhao X, Chen Y, Zhang H, et al. A new decomposition solver for complex electromagnetic problems. IEEE Antenn Propag Mag, 2017, 59: 131–140

13 Xie X L, Liang Y, Li X H, et al. Enabling coordinated register allocation and thread-level parallelism optimization for GPUs. In: Proceedings of the 48th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). New York: ACM, 2015. 395–406

14 Liang Y, Huynh H P, Rupnow K, et al. Efficient GPU spatial-temporal multitasking. IEEE Trans Parallel Distrib Syst, 2015, 26: 748–760

15 Chen C, Du Y F, Jiang H, et al. HPCG: preliminary evaluation and optimization on Tianhe-2 CPU-only nodes. In: Proceedings of Symposium on Computer Architecture and high-performance Computing. Jussieu: IEEE, 2014. 41–48

16 Ao Y L, Liu Y Q, Yang C, et al. Performance evaluation of HPGMG on tianhe-2: early experience. In: Proceedings of International Conference on Algorithms and Architectures for Parallel Processing. New York: Springer, 2015. 230–243

17 Liu Y Q, Yang C, Liu F F, et al. 623 Tflop/s HPCG run on Tianhe-2: leveraging millions of hybrid cores. Internat J High Perform Comput Appl, 2016, 30: 39–54

18 Li D, Xu C, Wang Y, et al. Parallelizing and optimizing large-scale 3D multi-phase flow simulations on the Tianhe-2 supercomputer. Concurr Computat-Pract Exper, 2016, 28: 1678–1692

19 Wei S, Zhao R C, Yao Y. Loop-nest auto-vectorizat ion based on SLP (in Chinese). J Softw, 2012, 23: 1717–1728

20 Zhao J, Zhao R C, Ding R, et al. Parallelism recognition technology based on nested loops classifying (in Chinese). J Softw, 2012, 23: 2695–2704

21 Gao W, Zhao R C, Han L, et al. Research on SIMD auto-vectorization compiling optimization (in Chinese). J Softw, 2015, 26: 1265–1284

22 Zhao J, Zhao R C, Han L, et al. An MPI backend for open64 compiler (in Chinese). J Softw, 2012, 23: 2695–2704