IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 32, NO. 11, NOVEMBER 2021 2631

Critique of “Planetary Normal Mode Computation: Parallel Algorithms, Performance, and Reproducibility” by SCC Team From Tsinghua University

Chen Zhang, Chenggang Zhao, Jiaao He, Shengqi Chen, Liyan Zheng, Kezhao Huang, Wentao Han, and Jidong Zhai

Abstract—In this article we present our results from the SC19 Student Cluster Competition Reproducibility Challenge. The challenge entails reproducing the algorithm entitled “Computing Planetary Interior Normal Modes with A Highly Parallel Polynomial Filtering Eigensolver” presented at SC’18, which proposes a parallel polynomial filtered Lanczos algorithm to directly calculate the planetary normal modes of heterogeneous planets. The proposed algorithm showed excellent performance with relatively low memory consumption and high parallel efficiency. In this work, we reproduce the scaling tests in that article on a cluster using Intel Cascade Lake architecture and use the proposed algorithm to illustrate specific normal modes of Mars. We compare the results obtained on our cluster with those in the original article. We also design a new metric to better analyze the results. In addition, we use the profiling tool Intel VTune Amplifier to explain our discoveries. Our results demonstrate that the given models show great scalability, which is similar to the original article. The required normal modes of Mars are also successfully calculated and visualized.

Index Terms—Reproducibility, scalability, geophysics, eigenvalues, eigenfunctions, profiling, student cluster competition

1 INTRODUCTION

Planetary normal modes are important for the study of planets. They provide abundant information about the inner structure of the planets and are useful in earthquake characterization, wave computation, and meteorite impact analysis [1],[2]. However, traditional approaches of computing normal modes are overly simplified to be used on strongly heterogeneous models. Also, the commonly used eigensolver algorithms are not sufficiently efficient for computation of large planetary normal modes.

Shi et al. propose a method to directly calculate the normal modes of 3-D strongly heterogeneous planets in their SC’18 paper entitled “Computing Planetary Interior Normal Modes with A Highly Parallel Polynomial Filtering Eigensolver” [3], which is referred to in our paper as the “NormalModes” paper. They use a mixed finite element method [4] to model the elastic-gravitational system of a planet into a Generalized Eigenvalue Problem (GEP) and apply a polynomial filtered Lanczos algorithm [5],[6] to solve the GEP efficiently. In their work, the bulk of the computations are converted into Sparse Matrix-Vector multiplication (SpMV) operations, which are amenable to parallel systems. The proposed algorithm achieves an approximately speedup and reduces the memory footprint by 98 percent, compared with the traditional shift-and-invert method. It also scales to 256 nodes with 75 percent parallel efficiency.

In this paper, we reproduce the scaling tests in the “NormalModes” paper, including weak scaling of SpMV, scaling with model size, and strong scaling on our 4-node competition cluster as part of the SC19

Student Cluster Competition (SCC) [7] Reproducibility Challenge. Our results, described within this paper, are similar to the “NormalModes” paper. We also calculate and visualize the normal modes of the Mars dataset required for the SCC reproducibility challenge.

There are 4 computing nodes in our competition cluster, thus we use 1, 2, and 4 nodes for the scaling studies. We first verify the weak scaling of SpMV used in the “NormalModes” paper. We then collect run details and performance data and design a new metric to further explain performance differences observed when scaling with model size. Later we keep the input data constant and use Intel VTune Amplifier to analyze the strong scaling for different numbers of processes. Finally, we use a MATLAB script to visualize results for verification.

The rest of the paper is organized as follows. Section 2 describes both the hardware and software configuration of our cluster. Section 3 introduces the details of compiling and running. We then present and analyze our results in Section 4. Conclusions are presented in Section 5.

2 EXPERIMENTAL SETUP

2.1 Hardware Configuration

The hardware of our cluster used to perform the experiments is listed in Table 1. Each node contains 2 Intel Xeon Cascade Lake Platinum 8280 processors, which has 28 cores per socket, 32 KB L1d/i cache, 1 MB L2 cache per core, and a 38.5 MB shared L3 cache.

2.2 Software Configuration

The software configuration of our cluster to run the experiments, including operating system, kernel version, compilers and libraries is listed in Table 2.

We use the latest versions of OS, Linux kernel, compilers and libraries to maximize the performance of the Intel Cascade Lake architecture.

3 COMPILATION AND RUN

3.1 Compilation

We get the source code of “NormalModes” [8] and pEVSL [8]. We then install compilers and libraries in Table 2 and carefully follow the instructions given by the authors to compile and link to reproduce their results.

3.2 Running

All the experiments use the environment variable setting of OMP_NUM_THREADS=1 to ensure no thread is forked by MKL. Also, the stack size limit is set to unlimited to avoid crashing from stack overflow errors. Then we run the experiments with one process per physical core.

3.3 Dataset

We use the Mars dataset provided by the SCC Committee. The dataset consists of 6 models, labeled from M1 to M6. Each model has a different number of elements, scaling from 591,301 to 15,283,899. The details of the models are shown in Table 3.

4 REPRODUCTION WORK

The SCC reproducibility challenge requires teams to reproduce Tables VI, Table VII, Figure 7 and a similar visual output of the Mars in the “NormalModes” paper.

1. https://github.com/p1019/NormalModes/commit/7ad0c92
2. https://github.com/eigs/pEVSL/commit/d996ba, a parallel eigenvalue slicing library depended by “NormalModes” from the authors’ repositories
In this section, we review results from the original paper and present our efforts to reproduce the work through the following challenges:

**Weak Scaling of SpMV.** To reproduce Figure 7 in the “NormalModes” paper, we divide the models provided into 2 groups and test on 1, 2, and 4 nodes of our cluster to study the weak scaling performance of each group.

**Scaling With Model Size.** We further study the relationship between total time and model size using the data collected in the weak scaling experiments. We propose a new metric to analyze results and compare it with Table VI in the “NormalModes” paper.

**Strong Scaling.** To compare with the results of Table VII in the “NormalModes” paper, we run M2 and M3 on 1, 2, and 4 nodes of our cluster respectively and analyze the strong scaling performance of each group.

**Visualization.** We study Fig. 5 of the “NormalModes” paper carefully and use Paraview to visualize the required normal modes.

### 4.1 Weak Scaling of SpMV

The “NormalModes” paper uses a GEP $Ax = \lambda Mx$ to model the elastic-gravitational system of a non-rotating planet, where $A$ and $M$ are sparse matrices representing the planet system. The GEP is solved by a polynomial filtered Lanczo algorithm. The main operations of this algorithm are the SpMVs, with $A$ and $M$ act as the “matrix”. We call these operations $Av$ and $Mv$, respectively.

In the weak scaling experiment of the “NormalModes” paper, the numbers of nodes used are doubled when the elements of the models are doubled. However, for the dataset provided, the number of elements is not doubled from M1 to M6, but approximately doubled in 2 groups from M1 to M3 and M4 to M6, just as Table 3 shows.

In order to provide consistent results, the following two strategies may be used:

- **Strategy A:** Change the numbers of cores used into the same ratio of the numbers of 6 models regardless of the communication topology between the ranks.

- **Strategy B:** Divide the models into 2 groups: M1 to M3 and M4 to M6. Then use 1, 2, and 4 nodes for relatively close workload per processor in each group.

First, there is little difference between the two strategies with respective to core hour utilization. And Strategy B is of a similar configuration with the original experiments, which is also in accord with the definition of weak scaling. However, Strategy A which scales by cores is more difficult to describe the weak scaling, because the MPI communication behavior between ranks is more complicated due to a non-fixed number of ranks per node.

So we choose Strategy B, dividing the models into 2 groups and keep the number of ranks used per node constant.

Results are given in Table 3 and Fig. 1. We can see that the scaling of M4 to M6 is good while M1 to M3 does not scale very well, as the problem size of M1 to M3 is smaller so that both computation and communication overhead are relatively larger than M4 to M6.

Compared to the results of Figure 7 in the “NormalModes” paper, we find our parallel efficiency is much lower. There are 2 main reasons as follows.

First, in terms of problem size (number of elements), the Mars one is smaller, leading to more overhead.

Second, we find the polynomial degree varies greatly in the Mars models, while that of solid and Earth models in the “NormalModes” paper is very stable. Taking this fact into consideration, we need to further unify all the runtime details and use a new metric to analyze the weak scaling. We present a discussion of this problem in Section 4.2.

### 4.2 Scaling With Model Size

It is a common scenario to compute all eigenvalues in a specified interval of the spectral interval. In Table 4 and Fig. 2, we report the performance on our cluster for models M1 to M6 (as described in Table 3). From Table 2, we can observe a constant number of iterations is not doubled from M1 to M6, but approximately doubled in 2 groups from M1 to M3 and M4 to M6, just as Table 3 shows.

In order to provide consistent results, the following two strategies may be used:

- **Strategy A:** Change the numbers of cores used into the same ratio of the numbers of 6 models regardless of the communication topology between the ranks.

- **Strategy B:** Divide the models into 2 groups: M1 to M3 and M4 to M6. Then use 1, 2, and 4 nodes for relatively close workload per processor in each group.

First, there is little difference between the two strategies with respective to core hour utilization. And Strategy B is of a similar configuration with the original experiments, which is also in accord with the definition of weak scaling. However, Strategy A which scales by cores is more difficult to describe the weak scaling, because the MPI communication behavior between ranks is more complicated due to a non-fixed number of ranks per node.

So we choose Strategy B, dividing the models into 2 groups and keep the number of ranks used per node constant.

Results are given in Table 3 and Fig. 1. We can see that the scaling of M4 to M6 is good while M1 to M3 does not scale very well, as the problem size of M1 to M3 is smaller so that both computation and communication overhead are relatively larger than M4 to M6.

Compared to the results of Figure 7 in the “NormalModes” paper, we find our parallel efficiency is much lower. There are 2 main reasons as follows.

First, in terms of problem size (number of elements), the Mars one is smaller, leading to more overhead.

Second, we find the polynomial degree varies greatly in the Mars models, while that of solid and Earth models in the “NormalModes” paper is very stable. Taking this fact into consideration, we need to further unify all the runtime details and use a new metric to analyze the weak scaling. We present a discussion of this problem in Section 4.2.

### Table 1: Hardware Configuration of Our Cluster

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Supermicro SuperServer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4029GP-TR-T \times 2</td>
</tr>
<tr>
<td></td>
<td>2029GP-TR \times 2</td>
</tr>
<tr>
<td>(including the head node)</td>
<td></td>
</tr>
<tr>
<td>CPU (per node)</td>
<td>Intel Xeon Platinum 8280 \times 2</td>
</tr>
<tr>
<td>Memory (per node)</td>
<td>DDR4 2933 MHz 32 GB \times 12</td>
</tr>
<tr>
<td>Accelerators (4029GP)</td>
<td>Tesla V100 32 GB</td>
</tr>
<tr>
<td>Storage (head node)</td>
<td>Intel SSD DC S4600 960 GB</td>
</tr>
<tr>
<td>Storage (other nodes)</td>
<td>Intel SSD DC S3610 100 GB</td>
</tr>
<tr>
<td>Infiniband HCA card</td>
<td>Mellanox HDR100 adapter</td>
</tr>
<tr>
<td>Ethernet card</td>
<td>1000MBase-T ethernet adapter</td>
</tr>
</tbody>
</table>

### Table 2: Software Configuration of the Cluster

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Debian 9.11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel</td>
<td>Linux kernel 4.19.67</td>
</tr>
<tr>
<td>Compilers</td>
<td>Intel C++ and Fortran Compilers 2019</td>
</tr>
<tr>
<td>MPI</td>
<td>Intel MPI Library 2019</td>
</tr>
<tr>
<td>MKL</td>
<td>Intel Math Kernel Library 2019</td>
</tr>
<tr>
<td>ParMETIS</td>
<td>ParMETIS 4.0.3</td>
</tr>
</tbody>
</table>

### Table 3: Mars Models With Different Sizes

<table>
<thead>
<tr>
<th>Exp</th>
<th>nn/np</th>
<th># of elm.</th>
<th>size of $A_d$</th>
<th>$T - Av(s)$</th>
<th>$T - Mv(s)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>1/56</td>
<td>591,301</td>
<td>281,784</td>
<td>0.001641</td>
<td>0.000157</td>
</tr>
<tr>
<td>M2</td>
<td>2/112</td>
<td>1,294,290</td>
<td>610,355</td>
<td>0.002768</td>
<td>0.000262</td>
</tr>
<tr>
<td>M3</td>
<td>4/224</td>
<td>2,229,825</td>
<td>1,208,013</td>
<td>0.003333</td>
<td>0.000353</td>
</tr>
<tr>
<td>M4</td>
<td>1/56</td>
<td>3,706,428</td>
<td>1,786,566</td>
<td>0.009920</td>
<td>0.001894</td>
</tr>
<tr>
<td>M5</td>
<td>2/112</td>
<td>7,532,466</td>
<td>3,569,283</td>
<td>0.012335</td>
<td>0.002162</td>
</tr>
<tr>
<td>M6</td>
<td>4/224</td>
<td>15,283,899</td>
<td>7,182,987</td>
<td>0.014089</td>
<td>0.002409</td>
</tr>
</tbody>
</table>

Fig. 1. Weak scaling tests of M1 to M3 and M4 to M6. The parallel efficiency is calculated as $\frac{\text{SpMV}}{\text{av}}$. 

Fig. 3a shows that the “unified time” is almost constant for solid models and slightly increasing for the Earth models. And from Fig. 3b, we can see that the pattern of M1 to M6 is more like the Earth model, with a slow rising unified time caused by fluid regions.
For a detailed relationship between the variables, both Table VI in “NormalModes” paper and our result show: in most cases, when the problem size grows larger, the spectrum becomes wider and the required degree increases while the number of iterations remains almost unchanged, so the total time grows proportionately to the polynomial degree.

4.3 Strong Scaling

Fig. 4a shows the \( A_v \) time and total time of C3 and E3, using data from Table VII of the “NormalModes” paper. Most data show strong scaling, and the only exception is \( A_v \) time of E3, which can be seen in Fig. 4b. The reason is that the Earth model E3 has fluid regions, leading to a more complex pattern of \( A_v \).

But as \( A_v \) does not take much time, it has little impact on the overall scaling.

We test the strong scaling of the proposed algorithm by computing the 36 eigenvalues in interval \((-9.8e-7, 1.15)\) of \( M_2 \) and \( M_3 \), which is the same in Table 4 except the number of nodes. From Fig. 4c, we can see a good scaling of M3, while the scaling of M2 is not that ideal due to a smaller problem size. Fig. 4d shows that the \( A_v \) time of the two Mars models do not scale well due to the fluid regions like E3 in the “NormalModes” paper.

Note that the parallel efficiency in M3 is greater than 1.0 from using 1 node to 2. To understand the reasons, we use Intel VTune Amplifier to trace the hardware counters. We find that L2 bound reduces from 26 percent (1 node) to 12 percent (2 nodes), indicating that L2 Cache is more likely to hold the data of matrices with more processors.

4.4 Visualization

After completing the runs, we use the MATLAB scripts in the repository\(^3\) to convert the data of required normal modes into VTK format which can be opened by ParaView.

We only obtain 27 out of the 36 modes of M6, so our “20th mode” may not match the “20th mode of M6” that SCC Committee wants us to visualize. Therefore, we put both the 20th mode of M5 and M6 in Fig. 5, showing the toroidal mode (M5) and the spheroidal mode (M6). The colors present the radial components of the displacement field.

The outputs look reasonable and similar with the toroidal and spheroidal modes in Fig. 5 of the “NormalModes” paper, which also verifies our results on Cascade Lake processors.

5 CONCLUSIONS

In this paper, we reproduce the experiments of weak scaling of SpMV, scaling with model size, and strong scaling on a 4-node Intel Cascade Lake Architecture cluster.

| Table 4 | Computation of all the Eigenvalues of Models With Different Sizes in Fixed Intervals |
|------------------------|---------------------------------|-------------------------------|----------------------|-----------------------|
| Exp | \( (\lambda_{min}, \lambda_{max}) \) | \((\xi, \eta)\) | (deg, #it) | eigs | total (s) |
|------------------------|---------------------------------|-------------------------------|----------------------|-----------------------|
| M1 | \((-9.8e-7, 1.15)\) | \((9.9e-6, 3.9e-5)\) | (531, 172) | 36 | 515.13 |
| M2 | \((-1.4e-7, 1.15)\) | \((9.9e-6, 3.9e-5)\) | (517, 172) | 36 | 839.64 |
| M3 | \((-7.0e-7, 2.62)\) | \((9.9e-6, 3.9e-5)\) | (792, 172) | 36 | 1656.54 |
| M4 | \((-1.9e-7, 1.35)\) | \((9.9e-6, 3.9e-5)\) | (559, 172) | 36 | 5646.79 |
| M5 | \((-2.1e-7, 2.12)\) | \((9.9e-6, 3.9e-5)\) | (702, 172) | 36 | 8112.10 |
| M6 | \((-3.0e-8, 3.44)\) | \((9.9e-6, 3.9e-5)\) | (889, 172) | 27 | 11270.69 |

The proposed algorithm finds 36 eigenvalues of M6, but 9 of them do not meet the precision requirement and are discarded by pEVSL.

For a detailed relationship between the variables, both Table VI in “NormalModes” paper and our result show: in most cases, when the problem size grows larger, the spectrum becomes wider and the required degree increases while the number of iterations remains almost unchanged, so the total time grows proportionately to the polynomial degree.

\[^3\] https://github.com/js1019/PlanetaryModels

Fig. 2. Required degrees and total time of models with different sizes in fixed intervals.

Fig. 3. Unified time of computing all the eigenvalues in a fixed interval, defined as \( \frac{T_{total}}{\text{deg}/C2\#it} \).

Fig. 4. Strong scaling test for problem C3, E3 from the “NormalModes” paper and M2, M3 on our cluster. The parallel efficiency is calculated as \( \frac{n_0T_{np}}{n_pT_{np}} \).

Fig. 5. Visualization of M5’s and M6’s 20th normal modes.
In our experiments, we observe degraded weak scaling due to the non-constant polynomial degree. Later we define a new “unified time” for a consistent weak scaling comparison. In terms of “unified time”, the results show a great weak scaling as mentioned in the “NormalModes” paper. For strong scaling tests, we find that a larger-problem-size and solid input scales better than the smaller-problem-size input with fluid regions. We also use profiling tools to explain that the phenomenon of super-linear speedup is caused by better cache usage.

In conclusion, our results are similar to the “NormalModes” paper.

ACKNOWLEDGMENTS

The authors would like to thank the SCC Committee of SC19 for organizing such a great competition and sparing no effort to help us know the field better. They would also like to show their gratitude to the committee member Hua Huang from Georgia Tech and the author of the “NormalModes” article Jia Shi from Rice University for sharing their ideas towards our work. This work was partially supported by the National Key R&D Program of China under Grant 2016YFB0200100, Tsinghua University Initiative Scientific Research Program under Grant 20191080594. Chen Zhang and Chenggang Zhao contributed equally to this work.

REFERENCES


For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.