LETTER Special Section on Empirical Software Engineering

What Factors Affect the Performance of Software after Migration: A Case Study on Sunway TaihuLight Supercomputer

Jie TAN†(a), Jianmin PANG†, Nonmembers, and Cong LIU††(b), Member

SUMMARY Due to the rapid development of different processors, e.g., x86 and Sunway, software porting between different platforms is becoming more frequent. However, the migrated software’s execution efficiency on the target platform is different from that of the source platform, and most of the previous studies have investigated the improvement of the efficiency from the hardware perspective. To the best of our knowledge, this is the first paper to exclusively focus on studying what software factors can result in performance change after software migration. To perform our study, we used SonarQube to detect and measure five software factors, namely Duplicated Lines (DL), Code Smells Density (CSD), Big Functions (BF), Cyclomatic Complexity (CC), and Complex Functions (CF), from 13 selected projects of SPEC CPU2006 benchmark suite. Then, we measured the change of software performance by calculating the acceleration ratio of execution time before (x86) and after (Sunway) software migration. Finally, we performed a multiple linear regression model to analyze the relationship between the software performance change and the software factors. The results indicate that the performance change of software migration from the x86 platform to the Sunway platform is mainly affected by three software factors, i.e., Code Smell Density (CSD), Cyclomatic Complexity (CC), and Complex Functions (CF). The findings can benefit both researchers and practitioners.

key words: software migration, software performance, Sunway TaihuLight supercomputer

1. Introduction

Currently, processors with different instruction set architectures (ISA), such as x86, ARM, and Sunway [1], are developing rapidly. There are some software compatibility problems between different processors. For example, x86 applications cannot execute directly on the Sunway processor since the latter cannot recognize x86 binaries [2]. At this point, software porting becomes especially important when one wants to execute mature x86 programs on Sunway TaihuLight System [1]. The migrated software’s execution efficiency on the target platform is different from that of the source platform. Regarding the improvement of the migrated software’s execution efficiency, most of the existing studies have focused on the hardware [3]–[5]. For example, Ao et al. [6] provide a block multi-coloring approach for parallelization to achieve high performance for memory-bound applications on the Sunway TaihuLight system.

However, software factors (e.g., software architecture and source code) also affect the migrated software’s performance. For example, the way software is coded can affect performance due to the I/O hardware instructions it generates [7]. To the best of our knowledge, there is still a lack of research on investigating how software factors affect execution efficiency change after software migration. Thus, this is the first paper to exclusively focus on studying what software factors, e.g., complexity, duplication, and code smell, can result in significant performance change after software migration.

In this paper, we used SonarQube to detect and measure five software factors, namely Duplicated Lines (DL), Code Smells Density (CSD), Big Functions (BF), Cyclomatic Complexity (CC), and Complex Functions (CF), from 13 selected projects of SPEC CPU2006 benchmark suite. These five factors have been widely used in other software engineering research that focused on measuring software quality. In addition, we measured the change of software performance by calculating the acceleration ratio of execution time before (x86) and after (Sunway) software migration. The definition and the detailed experimental environment are explained in Sect. 2. Finally, we performed a multiple linear regression model to analyze the relationship between the software performance change and the software factors.

The results indicate that the performance change of software migration from the x86 platform to the Sunway platform is mainly affected by three software factors, i.e., Code Smell Density (CSD), Cyclomatic Complexity (CC), and Complex Functions (CF). In particular, the higher the Code Smell Density and Cyclomatic Complexity in the source code, the more significantly the software’s execution efficiency on the Sunway TaihuLight System will be reduced. However, the migrated software with a higher proportion of Complex Functions tends to be more effective on the Sunway platform.

The findings can benefit both researchers and practitioners. Current optimization methods are implemented after software migration, and our results can help researchers develop tools to predict post-migration performance from the perspective of software. Practitioners can also use this predictive result to consider whether the software is worth porting. Moreover, practitioners can take advantage of our findings to pre-process software to improve execution efficiency after migration. For example, they can minimize

Manuscript received February 26, 2021.
Manuscript revised July 7, 2021.
Manuscript publicized October 21, 2021.
†The authors are with State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450001, China.
††The author is with School of Computer Science and Technology, Shandong University of Technology, Zibo 255000, China.
a) E-mail: jessie_tanjie@hotmail.com
b) E-mail: liucongchina@sdu.edu.cn
DOI: 10.1587/transinf.2021MPL0003
the density of code smells and cyclomatic complexity before porting. Furthermore, as aforementioned, this is the first paper to exclusively focus on studying what software factors can result in significant performance change after software migration. Although the conclusion only applies to the Sunway platform, this paper provides a method to measure migration performance on different platforms from a software perspective.

The remainder of this paper is structured as follows. Section 2 reports on the study objectives and provides details regarding the data collection and analysis. Section 3 illustrates the results of our study and Sect. 5 presents the threats to the validity, followed by the related work in Sect. 4 and Sect. 6 concludes the paper.

2. Study Design

This study aims to investigate the software factors that influence the performance change after software transplantation. To achieve this goal, we describe the selected projects as the subject of our case study, the variables, and the process by which we collected each variable in this section.

2.1 Project Selection

To perform our study, we selected projects from the SPEC CPU2006 benchmark suite as subject systems. This suite provides a comparative measure of compute-intensive performance across the widest practical range of hardware using workloads developed from real user applications†. In addition, SPEC CPU2006 has been widely used in other studies to measure processor performance [8]–[10]. There are several programming languages involved in the SPEC CPU2006, e.g., C and Fortran. To measure the impact of software factors on execution performance effectively, we only focus on one programming language, i.e., C language, which is currently ranked as the most popular programming language††.

†http://www.spec.org/cpu2006/, visited in Feb 2021.
††According to the Tiobe Index, one of the best-known indices of programming languages popularity, https://www.tiobe.com/tiobe-index/, visited in Feb 2021.

Finally, we decide to choose 13 projects from the SPEC CPU2006 benchmark suite, and the largest number of files and source lines of code (SLOC) of these projects are written in C language. Table 1 presents the name, domain, and SLOC for the selected projects in the first three columns.

2.2 Variables and Data Collection

In this section, we describe the sets of variables necessary to answer the research questions, as well as the necessary tooling and the major steps of the data collection process.

To measure the impact of different software factors on the execution performance after software migration, as shown in Table 2, we select five well-known variables related to software quality. Among these five variables, Big Functions (BF) represents the percentage of big functions that contain more than 20 lines of code; Cyclomatic Complexity (CC) represents the number of code smells per LOC, which is commonly related to software quality. Complex Functions (CF) represents the number of complex functions that contain more than 15 code paths. All of these variables can be measured by SonarQube [11], which is being widely used in both industry††† and for research purposes [12]–[14]. The independent variables for each project are shown in Table 1.

To compare software performance before and after the migration, we need to record the software’s execution time on both platforms. We clarify that we recompile the source code of the benchmarks using the ported GCC on the Sunway platform without any modifications to the source code. In addition, this study aims to investigate the impact of code-level factors on the performance of software migration; thus, we do not perform any architecture-specific optimizations and parallel optimizations (e.g., multi-threading or SIMD)

†††https://www.sonarsource.com/customers/, visited in Feb 2021.

### Table 1 The selected projects from SPEC CPU2006

| Projects    | Domain                                      | SLOC   | Variables | Execution Time |
|-------------|---------------------------------------------|--------|-----------|----------------|
| 400.perlbench | Programming language                      | 78,848    | DL       | x86 (s)        |
| 401.bzip2   | Compression                                 | 4,632    | CSD      | Sunway (s)     |
| 403.gcc     | Compiler                                    | 335,561  | BF       | Ratio          |
| 429.mcf     | Combinatorial Optimization                   | 1,153    | CC       |                |
| 433.milc    | Physics: Quantum Chromodynamics             | 6,391    | CF       |                |
| 435.gromacs | Chemistry / Molecular Dynamics              | 56,599   |          |                |
| 445.gobmk   | Artificial Intelligence: go                 | 155,179  |          |                |
| 456.hmmer   | Search Gene Sequence                        | 17,856   |          |                |
| 458.sjeng   | Artificial Intelligence: Chess              | 10,146   |          |                |
| 462.libquantum | Physics: Quantum Computing                  | 2,255    |          |                |
| 464.h264ref | Video Compression                           | 33,165   |          |                |
| 470.hm      | Fluid Dynamics                              | 671      |          |                |
| 482.sphinx3 | Speech recognition                          | 11,387   |          |                |

### Table 2 Independent variables

| Variables | Definition                                                                 |
|-----------|-----------------------------------------------------------------------------|
| DL        | Number of lines involved in duplications                                   |
| CSD       | Average number of code smells per LOC                                      |
| BF        | Percentage of big functions                                                |
| CC        | Number of paths through the code                                           |
| CF        | Percentage of complex functions                                            |

In addition, this study aims to investigate the impact of code-level factors on the performance of software migration; thus, we do not perform any architecture-specific optimizations and parallel optimizations (e.g., multi-threading or SIMD).
instructions). Table 3 shows the detailed information for the source platform (i.e., x86 processor) and the target platform (i.e., Sunway processor), including the OS, CPU, compilers and compiler options. To ensure that software performance comparisons are strictly software-determined and accurate, we use the same hardware configuration when running these projects. In addition, to further enhance internal validity, we perform each project five times on both platforms and use the average value. Moreover, we measure the change of software performance by calculating the acceleration ratio of execution time before and after software migration, which is measured as:

\[
\text{Ratio} = \frac{\text{execution time}_{\text{x86}}}{\text{execution time}_{\text{Sunway}}}
\]

The results of all the analyzed projects are shown in Table 1.

In addition, we assume that the performance of the migrated software on the Sunway TaihuLight System will be affected by all of the variables mentioned above (see Table 2). To validate this hypothesis, we perform a multiple linear regression model for these variables. As a previous step, we use logarithmic transformation to normalize independent and dependent variables [15]. The initial regression model is measured as:

\[
\log(\text{Ratio}) = \beta_0 + \beta_1 \log(\text{DL}) + \beta_2 \log(\text{CS D}) \\
+ \beta_3 \log(\text{BF}) + \beta_4 \log(\text{CC}) + \beta_5 \log(\text{CF}) + \varepsilon
\]

3. Results

To perform our study, we have used the least-squares method [16] to construct the multiple linear regression model. In addition, we have obtained the high values of the R-square (0.93) and the adjusted R-square (0.88), which indicates that there may be multicollinearity problems between the independent variables. Figure 1 shows the correlation coefficients between five variables. The results present that there are two sets of variables, i.e., Duplicated Lines (DL) and Cyclomatic Complexity (CC), Big Functions (BF), and Complex Functions (CF), highly correlated. The collinearity between variables can be expected, e.g., the more complex functions (CF), the more large functions (BF).

To eliminate the influence of collinearity on the regression model, we have decided to remove the relevant variables and then carry out simulation fitting on the model. Since a variable in the initial model may be critical in other models where the variable has been removed, thus, we have used the Akaike’s criterion (AIC) [18] to the model before deciding to remove a variable. Finally, we have removed Duplicated Lines (DL) and Big Functions (BL). Thus, the adjusted regression model can be measured as:

\[
\log(\text{Ratio}) = \beta_0 + \beta_2 \log(\text{CS D}) + \beta_3 \log(\text{CF}) + \varepsilon
\]

The results are presented in Table 4, which includes the coefficients of the adjusted regression model (see Formula 3). In addition, the values of the R-square and the adjusted R-square are 0.68 and 0.58, respectively. It indicates that the independent variables in Table 2 can explain the 58% of the observed variation in the adjusted regression model. Moreover, all of the coefficients in Table 4 are significant since the p-values are less than 0.05. The variables corresponding to these values are the best choices obtained in the variable selection iteration process based on AIC.

Thus, the final multiple linear regression model is measured as:

\[
\log(\text{Ratio}) = -0.8706 + 0.5266 \log(\text{CS D}) \\
+ 0.1647 \log(\text{CC}) - 0.6650 \log(\text{CF}) + \varepsilon
\]

It indicates that the performance change of software migration from the x86 platform to the Sunway platform is mainly affected by three software factors, i.e., Code Smell Density

---

Table 3

| Source (x86) Platform | Target (Sunway) Platform |
|----------------------|--------------------------|
| OS       | Fedora 2.6.27.5-117.fc10.i686 | NeoKylin 4.0.0 |
| CPU      | Intel (R) Core (TM) 2 Quad CPU Q9500 2.83 GHz | SW26010 |
| Compiler | GCC-7.1.0 | GCC-7.1.0 |
| Compiler Options | GCC -O3 -mavx2 -fno-tree-vectorize | GCC -O3 -msimd -fno-tree-vectorize |

Table 4

| Variables | Coefficient (β) | Std. Error | t-value | p-value |
|-----------|-----------------|------------|---------|---------|
| Intercept (β₀) | -0.8706 | 0.3343 | -2.6041 | 0.0197 ** |
| CSD (β₁) | 0.5266 | 0.2142 | 2.4583 | 0.0299 ** |
| CC (β₂) | 0.1647 | 0.0640 | 2.5756 | 0.0363 ** |
| CF (β₃) | -0.6650 | 0.2350 | -2.8299 | 0.0197 ** |

---

*The correlation coefficient is greater than 0.7 [17].*
(CSD), Cyclomatic Complexity (CC), and Complex Functions (CF). In particular, the higher the Code Smell Density and Cyclomatic Complexity in the source code, the more significantly the software’s execution efficiency on the Sunway TaihuLight system will be reduced. However, the migrated software with a higher proportion of Complex Functions tends to be more effective on the Sunway platform.

4. Related Work

Sunway TaihuLight supercomputer is currently ranking #4 in the TOP500 list\(^1\). The previous studies that focus on optimizing the migrated software performance on that system are mainly in hardware\cite{6,20,9}. For example, Li et al.\cite{20} have taken full advantage of Sunway multi-core architecture and used a parallel framework to accelerate deep learning applications. However, this paper focuses on investigating the factors that lead to lower execution efficiency after software migration from the software perspective.

Moreover, the multiple linear regression model has been widely used in other software engineering research. For example, Fedotova et al.\cite{21} used this model to estimate software effort. In this paper, we use this model to measure which factors have a meaningful impact on software performance after migration.

5. Threads to Validity

To improve the internal validity, we used the source code of the standard benchmark suite and ensured that the source and target platforms’ hardware parameters were consistent during the analysis. Construct validity concerns the connection between the research questions and the objects of study. In this respect, we used SonarQube to detect the software factors in our study. Although SonarQube has been widely used, our interpretation is still limited to the capabilities of the tool. In principle, any used tool would be subject to similar threats. To address the threats related to the multiple linear regression, we also evaluated the impact of the correlations between different variables on the final model in a circular iteration. Regarding the external validity, despite the analyzed projects cover different domains and sizes, our results may not be fully representative for all C programs.

6. Conclusions

To the best of our knowledge, this is the first paper to exclusively focus on studying what software factors can result in significant performance change after software migration. The results indicate that the performance change of software migration from the x86 platform to the Sunway platform is mainly affected by three software factors, i.e., Code Smell Density (CSD), Cyclomatic Complexity (CC), and Complex Functions (CF). In particular, the higher the Code Smell Density and Cyclomatic complexity in the source code, the more significantly the software’s execution efficiency on the Sunway TaihuLight system will be reduced. The findings can benefit both researchers and practitioners. For example, practitioners can minimize the density of code smells and cyclomatic complexity before porting to improve execution efficiency. Although the conclusion only applies to the Sunway platform, this paper provides a method to measure migration performance on different platforms from a software perspective.

Acknowledgements

This work was partly supported by the National Natural Science Foundation of China under Grant 61902222, the Taishan Scholars Program of Shandong Province under Grant tsqn201909109, and the Natural Science Excellent Youth Foundation of Shandong Province under Grant ZR202102180934.

References

\hspace{1cm}[1] H. Fu, J. Liao, J. Yang, L. Wang, Z. Song, X. Huang, C. Yang, W. Xue, F. Liu, F. Qiao, W. Zhao, X. Yin, C. Hou, C. Zhang, W. Ge, J. Zhang, Y. Wang, C. Zhou, and G. Yang, “The sunway taihulight supercomputer: system and applications,” Science China Information Sciences, vol.59, no.7, pp.1–16, 2016.

\hspace{1cm}[2] J. Wang, J. Pang, X. Liu, F. Yue, J. Tan, and L. Fu, “Dynamic translation optimization method based on static pre-translation,” IEEE Access, vol.7, pp.21491–21501, 2019.

\hspace{1cm}[3] Y. Chen, K. Li, W. Yang, G. Xiao, X. Xie, and T. Li, “Performance-aware model for sparse matrix-matrix multiplication on the sunway taihulight supercomputer,” IEEE Transactions on Parallel and Distributed Systems, vol.30, no.4, pp.923–938, 2018.

\hspace{1cm}[4] H. Fu, J. Liao, W. Xue, L. Wang, D. Chen, L. Gu, J. Xu, N. Ding, X. Wang, C. He, S. Xu, Y. Liang, J. Fang, Y. Xu, W. Zheng, J. Xu, Z. Zheng, W. Wei, X. Ji, H. Zhang, B. Chen, K. Li, X. Huang, W. Chen, and G. Yang, “Refactoring and optimizing the community atmosphere model (cam) on the sunway taihulight supercomputer.” SC’16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp.969–980, IEEE, 2016.

\hspace{1cm}[5] G. Xiao, K. Li, Y. Chen, W. He, A.Y. Zomaya, and T. Li, “Caspnv: a customized and accelerative spmv framework for the sunway taihulight,” IEEE Transactions on Parallel and Distributed Systems, vol.32, no.1, pp.131–146, 2019.

\hspace{1cm}[6] Y. Ao, C. Yang, F. Liu, W. Yin, L. Jiang, and Q. Sun, “Performance optimization of the hpcg benchmark on the sunway taihulight supercomputer,” ACM Transactions on Architecture and Code Optimization (TACO), vol.15, no.1, pp.1–20, 2018.

\hspace{1cm}[7] A. Noureddine, R. Rouvoy, and L. Seinturier, “A review of energy measurement approaches,” SIGOPS Oper. Syst. Rev., vol.47, no.3, p.42–49, Nov. 2013.

\hspace{1cm}[8] S. Bird, A. Phansalkar, L.K. John, A. Mericas, and R. Indukuru, “Performance characterization of spec cpu benchmarks on intel’s core microarchitecture based processor,” SPEC Benchmark Workshop, pp.1–7, 2007.

\hspace{1cm}[9] Y. Wang, V. Lee, G.-Y. Wei, and D. Brooks, “Predicting new workload or cpu performance by analyzing public datasets,” ACM Transactions on Architecture and Code Optimization (TACO), vol.15, no.4, pp.1–21, 2019.

\hspace{1cm}[10] Y. Jeon, P. Biswas, S. Carr, B. Lee, and M. Payer, “Hextype: Efficient detection of type confusion errors for c++,” Proceedings of the

\hspace{1cm}[1]TOP500: https://www.top500.org/lists/top500/2020/11/, accessed Feb 2021.
(11) G.A. Campbell and P.P. Papapetrou, SonarQube in Action, 1st ed., Manning Publications Co., Greenwich, CT, USA, 2013.

(12) J. Tan, D. Feitosa, and P. Avgeriou, “An empirical study on self-fixed technical debt,” Proceedings of the Third International Conference on Technical Debt (TechDebt ’20), Seoul, South Korea, pp.11–20, ACM, 2020.

(13) J. Tan, D. Feitosa, P. Avgeriou, and M. Lungu, “Evolution of technical debt remediation in python: A case study on the apache software ecosystem,” Journal of Software: Evolution and Process, vol.33, no.4, 2020.

(14) D. Marcilio, R. Bonifácio, E. Monteiro, E. Canedo, W. Luz, and G. Pinto, “Are static analysis violations really fixed?: A closer look at realistic usage of sonarqube,” Proceedings of the 27th International Conference on Program Comprehension (ICPC ’19), Montreal, Quebec, Canada, pp.209–219, IEEE Press, 2019.

(15) I.G. Dambolena, S.E. Eriksen, and D.P. Kopcso, “Logarithmic transformations in regression: do you transform back correctly?,” Primus, vol.19, no.3, pp.280–295, 2009.

(16) E.P. Mansard and E. Funke, “The measurement of incident and reflected spectra using a least squares method,” Coastal Engineering 1980, pp.154–172, 1980.

(17) C.F. Dormann, J. Elith, S. Bacher, C. Buchmann, G. Carl, G. Carré, J.R.G. Marqués, B. Gruber, B. Lafourcade, P.J. Leitão, T. Münkemüller, C. McClean, P.E. Osborne, B. Reineke, B. Schröder, A.K. Skidmore, D. Zurell, and S. Lautenbach, “Collinearity: a review of methods to deal with it and a simulation study evaluating their performance,” Ecography, vol.36, no.1, pp.27–46, 2013.

(18) M.L. Thompson, “Selection of variables in multiple regression: Part i. a review and evaluation,” International Statistical Review/Revue Internationale de Statistique, vol.46, no.1, pp.1–19, 1978.

(19) X. Liu, J. Sun, L. Zheng, S. Wang, Y. Liu, and T. Wei, “Parallelization and optimization of nsga-ii on sunway taihulight system,” IEEE Transactions on Parallel and Distributed Systems, vol.32, no.4, pp.975–987, 2020.

(20) L. Li, J. Fang, H. Fu, J. Jiang, W. Zhao, C. He, X. You, and G. Yang, “Swcafe: A parallel framework for accelerating deep learning applications on sunway taihulight,” 2018 IEEE International Conference on Cluster Computing (CLUSTER), pp.413–422, IEEE, 2018.

(21) O. Fedotova, L. Teixeira, H. Alvelos, et al., “Software effort estimation with multiple linear regression: Review and practical application,” J. Inf. Sci. Eng., vol.29, no.5, pp.925–945, 2013.