Frontmatter

Editors

SIGAPP FY’17 Quarterly Report J. Hong

Selected Research Articles


Write-Aware Memory Management for Hybrid SLC-MLC PCM Memory Systems C. Ho, Y. Chang, Y. Chang, H. Chen, and T. Kuo 16

Large Scale Document Inversion using a Multi-threaded Computing System S. Jung, D. Chang, and J. Park 27

An Extracting Method of Movie Genre Similarity using Aspect-Based Approach in Social Media Y. Han and Y. Kim 36
SIGAPP FY’17 Quarterly Report

April 2017 – June 2017
Jiman Hong

Mission

To further the interests of the computing professionals engaged in the development of new computing applications and to transfer the capabilities of computing technology to new problem domains.

Officers

Chair Jiman Hong
Soongsil University, South Korea
Vice Chair Tei-Wei Kuo
National Taiwan University, Taiwan
Secretary Maria Lencastre
University of Pernambuco, Brazil
Treasurer John Kim
Utica College, USA
Webmaster Hisham Haddad
Kennesaw State University, USA
Program Coordinator Irene Frawley
ACM HQ, USA

Notice to Contributing Authors

By submitting your article for distribution in this Special Interest Group publication, you hereby grant to ACM the following non-exclusive, perpetual, worldwide rights:

- to publish in print on condition of acceptance by the editor
- to digitize and post your article in the electronic version of this publication
- to include the article in the ACM Digital Library and in any Digital Library related services
- to allow users to make a personal copy of the article for noncommercial, educational or research purposes

However, as a contributing author, you retain copyright to your article and ACM will refer requests for republication directly to you.

Next Issue

The planned release for the next issue of ACR is September 2017.
ABSTRACT
Presently, most of the software development organizations are adopting the phenomena of Global Software Development (GSD), mainly because of the significant return on investment it produces. However, GSD is a complex phenomenon and there are many challenges associated with it, especially that related to Software Process Improvement (SPI). The aim of this work is to identify humans’ related success factors and barriers that could impact the SPI process in GSD organizations and proposed a theoretical framework of the factors in relation to SPI implementation. We have adopted the Systematic Literature Review (SLR) method in order to investigate the success factors and barriers. Using the SLR approach, total ten success factors and eight barriers were identified. The paper also reported the Critical Success Factors (CSFs) and Critical Barriers (CBs) for SPI implementation following the criteria of the factors having a frequency ≥ 50% as critical. Our results reveal that five out of ten factors are critical for SPI program. Moreover, total three barriers were ranked as the most critical barriers. Based on the analysis of the identified factors, we have presented a theoretical framework that has highlighted an association between the identified factors and the implementation of the SPI program in GSD environment.

CCS Concepts
Software and Its engineering → Software creation and management → Software development process management → Software development methods → Capability Maturity Model

Keywords
Software Process Improvement; Systematic Literature Review; Human; Global Software Development

1. INTRODUCTION
Software process improvement (SPI) provides an organization with a commanding means of evaluating their abilities to improve the software development process; accordingly, this practice allows organizations to identify their strengths and weaknesses. Zahran [1] defined SPI as “the discipline of defining, characterizing, improving, and measuring software management, better product innovation, faster cycle times, greater product quality, and reduced development costs simultaneously.” Different standards and models for SPI have been developed, each of which assists organizations in optimizing their software processes through evaluation of process efficiency. Khan and Keung [2] mentioned that the failure rate of SP programs is 70%, and one of the ground reasons of this failure is the inadequate attention given to process improvement problems.

Various process improvement models and standards have been developed to assist software organizations effectively manage the software development processes. Capability maturity model integration (CMMI) is one such model consists of structured and methodical practices for process evaluation and improvement [3]. The International Organization for Standardization (ISO) has also developed standards and recommendations for SPI: for example, ISO 9000 is used to assess the quality of established systems in an organization [4], whereas ISO/IEC 15504 is used for process improvement under the Software Process Improvement and Capability Determination (SPICE) [5]. SPICE was developed to test and advertise process improvement standards and models [5]. The ISO/IEC 15504 standard has since evolved into more advanced process assessment and improvement standards, i.e., ISO/IEC 330XX [6]. The ISO/IEC 330XX family covers the assessment of processes deployed in an organization, including their maintenance, change management, delivery, and improvement [6]. These models and techniques can assist an organization to develop a quality product, reduce development cost and time, and promote user satisfaction [2, 7, 8, 9].

However, little attention has been paid to developing process improvement models and standards in the context of GSD, which has hence yielded limited success for process improvement efforts [7]. GSD is the plan of action in which the
software development is performed beyond the geographical, cultural and temporal boundaries [2]. Ramasubbu [9] reported that, most organizations currently outsource software development activities to attain various benefits, and it is significant that process improvement experts have complete understanding of SPI program. However, process improvement challenges in GSD environment are different and SPI practitioners should explicitly address those challenges [2, 8, 9]. The implementation of SPI activities is considerably more complex in the GSD environment than in collocated development [9]. The literature on SPI has not examined the distributed nature of GSD organizations in sufficient detail [8].

Less attention has been given to successfully execute the process improvement activities in GSD environment, and even few studies have been done to develop models and frameworks in general and factors that are critical for the efficient execution of process improvement programs in particular [2, 7, 8, 9]. There is a pressing need for some techniques and methods that could guide the GSD organizations to assess and implement the process improvement programs [2, 8]. These techniques and methods should have the potential to reduce the time, cost, and failure risk of SPI implementation. In this regard, we propose a model that can assist SPI practitioners to successfully assess, measure, and improve their process improvement activities related to humans. Therefore, we have been motivated to develop a software process improvement implementation and management model (SPIIMM), that could help the SPI practitioners to assess, implement and manage the humans’ related aspects of SPI. According to Komiyama [10] “all personnel must be interested in the process, as the success of software projects depends on processes”. Organizations were believed to be mostly dependent on human factors as they usually relied on key individuals [10]. Therefore, it is significant for organizations to properly manage the success factors and barriers reported in the domain of human category.

In this paper, we have discussed the primary step to develop the proposed model (SPIIMM). We have reported the success factors and barriers that could have a positive or negative impact on humans’ related aspects of the SPI program in GSD environment. These factors will help to develop the factors component of SPIIMM.

Systematic Literature Review (SLR) approach was adopted to have a better understanding of the concern factors for SPI. The SLR approach enabled us to assess and understand the available literature [11] in order to extract the most relevant success factors and barriers of SPI. The reported success factors and barriers will assist the theorist and SPI practitioners to address the key areas of SPI program related to the humans. For this reason, we have developed the following research questions.

RQ1: What are the humans related success factors, as identified in the literature that could have a positive impact on the SPI implementation in GSD?

RQ1.1: What are the most critical humans related success factors identified in the literature?

RQ2: What are the humans’ related barriers to SPI implementation in GSD environments are identified in the literature?

RQ2.1: What are the most critical humans related barriers reported in the literature?

RQ3: How to develop a theoretical framework for human factors of SPI?

2. RESEARCH METHODOLOGY

Systematic literature review (SLR) method was selected in order to conduct this research study. An SLR is a secondary study that reviews all primary studies (i.e., those that explore a specific research area), identifying, analyzing, and exploring all evidence related to research questions in an unbiased and iterative way [11]. Kitchenham and Charters [11] classified SLR into three main phases: planning the review, conducting the review and reporting the review.

2.1 Planning the Review

In the first phase of the review, the SLR protocol was developed. SLR protocol provided the complete guideline to develop the research questions, search process, inclusion and exclusion criteria, articles selection, publications quality assessment and data extraction [11].

2.1.1 Research Questions

The research questions developed for this study are discussed in section 1.

2.1.2 Search Process

A comprehensive search process was conducted in order to classify all the available research articles relevant to our questions of interest.

For the search process, the keywords and their alternatives were chosen based on the available literature in the domain of SPI [2, 7, 8, 9, 12, 17]. The major keywords and their alternate words were concatenated with the help of “OR” and “AND” operators that were later used in different databases to identify the relevant primary studies.

The following search strings were developed and used in several databases in order to identify the relevant articles.

("Factors" OR “Aspects” OR “Items” OR “Elements” OR “Drivers” OR “Motivators” OR “Variables” OR “Characteristics” OR “Parameters”) AND (“barriers” OR “obstacles” OR “hurdles” OR “difficulties” OR “impediments” OR “hindrance”) AND (“GSD” OR “global software development” OR “global software engineering” OR “distributed software development” OR “software outsourcing” OR “offshore software development” OR “information technology outsourcing” OR “software contracting-out” OR “IT outsourcing”) AND (“SPI” OR “software process improvement” OR “software process enhancement” OR “CMM” OR “CMMI” OR “SPICE” OR “software process enrichment” OR “software process evaluation” OR “software process assessment” OR “software process appraisal”)

Databases were chosen based on prior research experience, literature background and suggestions provided by other researchers [12]. Total five databases were selected, i.e., IEEE, Scopus, Google Scholar, Science Direct and ISI Web of Science.
2.1.3 Inclusion and Exclusion Criteria
The selected primary studies must be available in English language. Each primary study article must be conference, journal or book chapter. We emphasized on the articles that have discussed process improvement programs in the GSD environment. More focus was on those papers that discussed success factors and barriers along their categories.

We have excluded those papers which have not reported the software process improvement success factors and barriers. Those articles were also excluded that did not provide the detail information regarding the process improvement standards and models. The duplicated articles were also not considered. Additionally, those articles were also excluded that were reported in other languages except English.

2.1.4 Study Quality Assessment
The Quality Assessment (QA) of the selected articles was performed concurrently with the data extraction phase. A checklist was developed to assess the quality of the selected research articles. The guideline provided by [2, 8, 13] was followed in the design of this checklist (Table 1).

The quality assessment checklist consists of five QA questions (Q1-Q5). For each given item (QA1-QA5), the evaluation was done as follows:

- The articles containing answers to the checklist questions were assigned 1 point.
- The articles containing partial answers to the checklist question were assigned 0.5 points.
- The articles not containing any answers to the checklist questions were assigned 0 points.

<table>
<thead>
<tr>
<th>QA Questions</th>
<th>Checklist Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>QA1</td>
<td>“Do the adopted research methods address the research questions?”</td>
</tr>
<tr>
<td>QA2</td>
<td>“Does the study discuss any factors of SPI?”</td>
</tr>
<tr>
<td>QA3</td>
<td>“Does the study discuss SPI implementation standards and models?”</td>
</tr>
<tr>
<td>QA4</td>
<td>“Is the collected data related to SPI?”</td>
</tr>
<tr>
<td>QA5</td>
<td>“Are the identified results related to justification of the research questions?”</td>
</tr>
</tbody>
</table>

We have also evaluated the quality of the selected primary studies against each quality assessment question discussed in Table 1. The QA score for each primary study is shown in Table 2.

Table 2: QA score for each selected primary study

<table>
<thead>
<tr>
<th>ID</th>
<th>QA1</th>
<th>QA2</th>
<th>QA3</th>
<th>QA4</th>
<th>QA5</th>
<th>Final Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>[LT1]</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>[LT2]</td>
<td>0.5</td>
<td>1</td>
<td>0.5</td>
<td>1</td>
<td>0.5</td>
<td>3.5</td>
</tr>
<tr>
<td>[LT3]</td>
<td>0.5</td>
<td>1</td>
<td>0.5</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>[LT4]</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>1</td>
<td>0.5</td>
<td>2.5</td>
</tr>
</tbody>
</table>

2.2 Planning the Review
2.2.1 Articles Selection
The SLR protocol search strings (Sect. 2.1.2) were applied and a total of 2174 articles extracted from all the selected databases. The tollgate approach proposed by Afzal [14] was used in the articles selection process. Tollgate approach [14] led to a short list of the 22 articles to be considered in the primary studies selection. Tollgate techniques have five phases and the selected digital repositories were exclusively assessed using these phases (Table 3)

- St-1: Explore using the search terms
- St-2: Title and abstract based inclusion/exclusion
- St-3: Introduction and conclusion based inclusion/exclusion
- St-4: Full text based inclusion/exclusion
- St-5: Final selection of the articles to be included in the primary studies selection

After applying the tollgate approach, we have identified total 19 articles that focused on SPI in GSD.

Table 3: Tollgate Approach

<table>
<thead>
<tr>
<th>SD</th>
<th>St-1</th>
<th>St-2</th>
<th>St-3</th>
<th>St-4</th>
<th>St-5</th>
<th>% of final selected articles (n=19)</th>
</tr>
</thead>
<tbody>
<tr>
<td>WI</td>
<td>25</td>
<td>390</td>
<td>135</td>
<td>76</td>
<td>5</td>
<td>21</td>
</tr>
<tr>
<td>IE</td>
<td>461</td>
<td>390</td>
<td>135</td>
<td>76</td>
<td>5</td>
<td>26</td>
</tr>
<tr>
<td>AC</td>
<td>560</td>
<td>511</td>
<td>194</td>
<td>117</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>SP</td>
<td>146</td>
<td>112</td>
<td>37</td>
<td>25</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>ScD</td>
<td>502</td>
<td>673</td>
<td>232</td>
<td>154</td>
<td>4</td>
<td>21</td>
</tr>
<tr>
<td>GS</td>
<td>480</td>
<td>430</td>
<td>176</td>
<td>86</td>
<td>2</td>
<td>11</td>
</tr>
<tr>
<td>TL</td>
<td>2174</td>
<td>2140</td>
<td>791</td>
<td>469</td>
<td>19</td>
<td>100</td>
</tr>
</tbody>
</table>


2.2.2 Data Extraction
We have reported the following data of each selected primary study article in order to address the research questions: paper
title and QA score. A complete list of the selected 19 articles is given in the Appendix.

3. REPORTING THE RESULTS
Using SLR approach, we have identified total ten success factors and eight barriers in the domain of human category of SPI. Grounded theory-based coding approach [15] was adopted which provides qualitative and analytical approach in which concepts (success factors and barriers) are labeled and categorized through the close examination of qualitative data. We have used grounded theory-based coding approach [15] to review the selected 19 primary studies, conceptualize the underpinning factors and assign a label (name) to each factor. Similar or related factors are semantically compared and grouped in the human category. Therefore, applying the grounded theory-based coding approach enabled us to identify humans’ related category of SPI and classified the identified success factors and barriers across this category.

This section reports the results of the identified factors in relation to each of the research questions.

3.1 RQ1
The following are the identified ten success factors as shown in Figure 1, Table 4 and briefly discussed thereafter. In Table 4, (n=19) is the total number of primary studies selected during the SLR study.

![Figure 1: Frequency analysis of the identified success factors](image)

<table>
<thead>
<tr>
<th>S.No</th>
<th>Success Factors</th>
<th>Frequency (n=19)</th>
<th>% of occurrence</th>
</tr>
</thead>
<tbody>
<tr>
<td>SF1</td>
<td>Management Commitment</td>
<td>16</td>
<td>84</td>
</tr>
<tr>
<td>SF2</td>
<td>Staff Involvement</td>
<td>14</td>
<td>73</td>
</tr>
<tr>
<td>SF3</td>
<td>Strong Relationship</td>
<td>9</td>
<td>47</td>
</tr>
<tr>
<td>SF4</td>
<td>Information Sharing</td>
<td>8</td>
<td>42</td>
</tr>
<tr>
<td>SF5</td>
<td>SPI Expertise</td>
<td>6</td>
<td>32</td>
</tr>
<tr>
<td>SF6</td>
<td>Roles and Responsibilities</td>
<td>11</td>
<td>58</td>
</tr>
<tr>
<td>SF7</td>
<td>SPI Leadership</td>
<td>10</td>
<td>53</td>
</tr>
<tr>
<td>SF8</td>
<td>SPI Awareness</td>
<td>7</td>
<td>37</td>
</tr>
<tr>
<td>SF9</td>
<td>Skilled Human Resources</td>
<td>12</td>
<td>63</td>
</tr>
<tr>
<td>SF10</td>
<td>3C’s (Communication, coordination and control)</td>
<td>5</td>
<td>26</td>
</tr>
</tbody>
</table>

Table 4: Identified Success Factors

3.1.1 Management Commitment (SF1)
Most of the SPI literature focused on the importance of organizational management support for SPI [LT1]. The organizational management commitment plays a significant role in order to successfully execute the SPI activities. Sulayman et al [LT2] define management commitment as “the degree to which the higher and lower level management in an organization financially support, realize and involved in SPI activities”. Hall et al [LT1] demonstrate the importance of management commitment by conducting an empirical study. In particular, they reported that when the SPI manager who was highly committed to SPI was replaced by someone with a lower commitment, all prior SPI effort was lost. Therefore, based on the above discussion, we have developed the following hypothesis.

H1: Management commitment has a positive relationship with the SPI implementation in GSD.

3.1.2 Staff Involvement (SF2)
Sulayman et al [LT2] define staff involvement as the extent to which organizational staff own, believe, attain and participate in the process improvement activities. Bayona et al [LT3] highlighted that the staff involvement during the process definition of SPI activities is important to make them feel part of the program. Ramasubbu [LT4] reported the results of an industrial empirical study and pointed out that staff involvement in SPI activities reduce the resistance to the deployment of new processes. As a result of the given discussion, the following hypothesis is developed.

H2: Staff Involvement has a positive relationship with the SPI implementation in GSD.

3.1.3 Strong Relationship (SF3)
According to Niazi et al [LT5], “strong relationship is the degree to which the team members can effectively coordinate and communicate to implement the SPI program”. Strong relationship is the key element to effectively perform the SPI related activities and tasks [LT6]. Moreover, it leads to better team management, decision making, risk management and staff coordination [LT6]. Niazi et al, [LT5] reported that GSD teams might not be able to accomplish SPI activities within time and budget due to the lack of trust and support among the team members. Higher team management should motivate the team members to participate in SPI related activities in order to build long term positive and trustworthy relationship with different other distributed teams [LT5]. Consequently, we argue that strong relationship is a key factor to successfully implement the SPI program in GSD.

H3: Strong relationship has a positive association with the SPI implementation in GSD.

3.1.4 Information Sharing (SF4)
Sulayman et al [LT2] define information sharing as “the degree to which the distributed team members coordinate and communicate to share the information to involve in process improvement program”. Ramasubbu [LT4] reported that proper information sharing among the geographically distributed sites could assist the team members to positively execute the SPI activities. Khan et al [LT7] concluded that team members can properly participate in the process improvement activities based
we have developed the following hypothesis.

H4: Information sharing has a positive association with the SPI implementation in GSD.

3.1.5 SPI Expertise (SF5)
Bayona et al [LT3] define SPI expertise as the extent to which individuals can achieve specific SPI goals [LT3]. According to Hall [LT1], the success of SPI implementation programs depends on the expertise of team members. More specifically, if the management staff of SPI program has appropriate process improvement knowledge and awareness, then there is more chance of process improvement success and adoption in the best practice of domain [LT8]. Otherwise, the deployment of the process improvement program may end with the failure and frustration [LT3].

H5: SPI Expertise has a positive relationship with the SPI implementation in GSD.

3.1.6 Roles and Responsibilities (SF6)
Bayona et al [LT3] suggests that the assignment of specific roles and responsibilities to team members can improve the SPI implementation process. Roles and responsibilities should be clearly defined; otherwise confusion may occur during the implementation of SPI activities [LT3]. Therefore, we hypothesize that

H6: Roles and Responsibilities have a positive relationship with the SPI implementation in GSD.

3.1.7 SPI Leadership (SF7)
Hall et al [LT1] reported in their empirical study that most of the respondents highlighted the importance of the managers' leadership. They specified that the leadership of SPI team is vital for the successful execution of the SPI program [LT1]. The management should have the ability to direct and encourage team members to achieve the specific goals [LT3]. The success and acceptance of the SPI program mostly depend on the skillful top management individuals. If they have ample knowledge and deep understanding of SPI program, then they could accomplish their concern objectives [LT8]. Lack of SPI leadership could undermine the effectiveness of SPI program and the organization might not be able to actualize the key objectives of the business [LT1]. Hence, we argue that SPI leadership is a core factor for the SPI implementation.

H7: SPI leadership has a positive association with the SPI implementation in GSD.

3.1.8 SPI Awareness (SF8)
Sulayman et al [LT2] explained SPI awareness as the degree to which the top management takes the initiatives of SPI certification and provides team members with the training opportunities. Deployment of SPI program is the practice of implementation of new processes in an organization [LT5]. It is important to motivate the team members to conduct and participate in the awareness sessions relating to the implementation of process improvement programs. Therefore, we have developed the following hypothesis.

H8: SPI Awareness has a positive association with the SPI implementation in GSD.

3.1.9 Skilled Human Resources (SF9)
The results reveal that success factor ‘SF8: skilled human resources’ can play a significant role in SPI implementation program. Various researchers have discussed the significance of skilled humans for SPI activities. Bayona et al, [LT8] emphasized on the deployment of the qualified workers having professional certification and experience in the field of computer science, management and other relevant fields. They consider the skilled staff as the backbone of the GSD industry. Consequently, we hypothesize that:

H9: Skilled Human Resources have a positive association with the SPI implementation in GSD.

3.1.10 3C’s (Communication, coordination and control) (SF10)
Khan et al, [LT9] define 3C’s as the process of knowledge transmission between the distributed team members and the mode of transfer they adopt to enhance this interaction. Effective communication channels are claimed to assist process improvement. Coordination refers to the contribution of different people working together on a task for a specific goal [LT9]. Control refers to “the process of adhering to goals, policies, standards or quality levels” [LT9]. The Control manages the basic structures needed for SPI implementation i.e. on time, in budget and desired quality software development. Control and coordination are correlated with each other and both of them depend on communication [LT10]. Therefore, for successful implementation of SPI activities, it’s significant to properly manage the 3C’s issues.

H10: 3C’s (Communication, coordination and control) have a positive association with the SPI implementation in GSD.

3.2 RQ1.1
According to Niazi [LT11], the critical factors are presenting the key areas where the organizational management must focus to achieve the specific business goals. Niazi [LT11] and Khan et al. [LT12] shed light that limited attention given to those key areas can undermine the business performance. Critical factors may differ from person to person as it depends on the position of individual’s hold in an organization. Critical success factors (CSFs) depend on the geographical regions of the managers and may change with the passage of time [LT11].

In this research study, we have used the following criteria in order to conclude the criticality of the identified 10 success factors:

If a factor has frequency ≥50% of the selected primary studies, then the factor is considered as a critical factor. The same criteria have been used by other researchers in different other domains [LT11, LT7].

By using the above criteria, the identified CSFs are: SF1: Management commitment (84%), SF2: staff involvement (73%), SF6: roles and responsibilities (58%), SF7: SPI leadership (53%) and SF9: skilled human resources (63%). The identified CSFs are labelled as the most critical for SPI because their frequency ≥50% of all the other success factors.
3.3 RQ2

To address RQ2, Table 5 and Figure 2 presents the list of the identified barriers along their frequencies and percentage of occurrence in the selected primary studies (n=19).

Table 5: Identified Barriers

<table>
<thead>
<tr>
<th>S. No</th>
<th>Success Factors</th>
<th>Frequency (n=19)</th>
<th>% of occurrence</th>
</tr>
</thead>
<tbody>
<tr>
<td>BA1</td>
<td>Inexperienced staff</td>
<td>15</td>
<td>79</td>
</tr>
<tr>
<td>BA2</td>
<td>Staff turnover</td>
<td>13</td>
<td>68</td>
</tr>
<tr>
<td>BA3</td>
<td>Cultural differences</td>
<td>9</td>
<td>47</td>
</tr>
<tr>
<td>BA4</td>
<td>Time pressure</td>
<td>13</td>
<td>68</td>
</tr>
<tr>
<td>BA5</td>
<td>Lack of process improvement knowledge</td>
<td>8</td>
<td>42</td>
</tr>
<tr>
<td>BA6</td>
<td>Lack of training</td>
<td>5</td>
<td>26</td>
</tr>
<tr>
<td>BA7</td>
<td>Lack of trust</td>
<td>7</td>
<td>37</td>
</tr>
<tr>
<td>BA8</td>
<td>Lack of communication</td>
<td>8</td>
<td>42</td>
</tr>
</tbody>
</table>

Figure 2: Frequency analysis of the identified barriers

In this section, we have thoroughly discussed the relationship between the independent variables (identified barriers) and dependent variable (humans related SPI Implementation in GSD).

3.3.1 Inexperienced Staff (BA1)

The results presented in Table 5 revealed that (79%) of the selected primary studies indicated BA1 (inexperienced staff) as a barrier to process improvement activities. Hall et al. [LT1] discussed the results of empirical study and reported the challenges of SPI activities, including the lack of knowledge regarding the process improvement program. Niazi et al. [LT6] reported in their empirical study that those organizations which have successfully addressed the process improvement challenges had skillful and experienced SPI team members. We have proposed the following hypothesis based on the above discussion.

H11: Inexperienced staff negatively impacts the process improvement activities in GSD.

3.3.2 Staff Turnover (BA2)

BA2 (staff turnover) was cited as a critical barrier by (68%) of the primary studies. Staff turnover indicates the percentage of the individuals who leave the organization or replaced by new staff [LT12]. Rainer and Hall [LT19] mentioned that the frequent staff turnover could significantly increase the cost of the process improvement program. Staff turnover is related to the direct cost of employing new staff members as well the loss of skillful and experienced individuals [LT19]. Hall et al. [LT1] conducted empirical study with 45 focus groups and they suggested that boosting the motivation levels of staff members could decrease the staff turnover.

H112: Frequent staff turnover could negatively impact the process improvement activities in GSD.

3.3.3 Cultural Differences (BA3)

BA3 (cultural differences) was mentioned in the (47%) of the reported primary studies. In distributed environment, the team members might be culturally and geographically distinct [LT9]. It is significant to successfully overcome the cultural challenges faced by the team members. For instance, it is possible that the distributed team members may not be able to communicate with each other using their native languages [LT9]. Furthermore, cultural difference may cause misunderstanding, which can bring confusion among the team members [LT9]. Therefore, we hypothesized that:

H13: Cultural differences have negative association with SPI programs in GSD.

3.3.4 Time Pressure (BA4)

The results given in Table 5 shows that (68%) of the articles considered BA4 (time pressure) as the critical barrier of SPI implementation. Niazi et al. [LT6] highlighted that time pressure was one of the critical challenges for organizations who initiated the process improvement program. Due to time pressure, the practitioners might make quick decisions in order to stay on schedule, but those decisions may not be in the favor of SPI program [LT1, LT6]. Empirical study conducted by Hall et al. [LT1] reported that majority of the practitioners considered time pressure as the demotivation factor for software process improvement. Based on the given discussion, we have developed the following hypothesis.

H14: Time pressure could negatively affect the process improvement programs in GSD.

3.3.5 Lack of Process Improvement Knowledge (BA5)

BA5 (lack of process improvement knowledge) was reported by (42%) of the selected primary studies. The lack of adequate process improvement knowledge and expertise could be a serious threat for the SPI programs [LT16]. Bayona et al. [LT8] focused on the experience and expertise of SPI practitioners for process improvement programs. They suggested that process improvement practitioners should have complete knowledge and understanding of SPI activities. Niazi et al. [LT6] highlighted that lack of process improvement knowledge could undermine the entire SPI program. Based on the above discussion, the following hypothesis is developed.

H15: Lack of SPI knowledge could be a serious challenge for SPI activities in GSD.
3.3.6 **Lack of Training (BA6)**
BA5 (lack of training) was cited in (26%) of the selected primary studies as the barrier to process improvement. The process improvement program could not be effective if the GSD organizations did not provide proper training to SPI practitioners [LT17]. Due to lack of training, the process improvement team members might not be able to assess the real need of process improvement change [LT17]. It is significant for the SPI practitioners to have strong understanding of process improvement standards, models and techniques such as CMM, CMMI and ISO/IEC 330XX [LT17].

**H16:** Lack of proper training decrease the success rate of SPI implementation in GSD.

3.3.7 **Lack of Trust (BA7)**
It is challenging to develop trust and confidence between the SPI practitioners working in GSD environment [LT9, LT17]. It is shown in Table 5 that (37%) of the selected primary studies considered lack of trust as the barriers for SPI. Promoting team building activities could strengthen the inter-team communication that could increase the trust level of team members [LT9].

**H17:** Lack of trust has negative relationship with SPI programs in GSD.

3.3.8 **Lack of Communication (BA8)**
The SLR results given in Table 5 indicate that BA8 (lack of communication) is cited by (42%) of the primary studies. The geographically distributed nature of GSD organizations makes communication a major issue for SPI practitioners [LT9]. Lack of communication could decrease the level of trust and frequent feedback. Weak communication could create misunderstanding, lack of coordination and of lack of control over SPI activities [LT9].

**H18:** Lack of communication has negative relationship with SPI programs in GSD.

3.4 RQ2.1
We have followed the criteria discussed in section 3.2 in order to categorize the critical barriers (CBs). Three barriers were ranked as CBs by using the criteria of barriers having frequency >50% of the selected primary studies. The identified CBs are: BA1: inexperienced staff (79%), BA2: staff turnover (68%) and BA4: time pressure (68%). The CBs are presenting some of the crucial areas of process improvement program where organization need much focus in order to successfully execute the SPI activities [LT17].

![Figure 3: Proposed Theoretical Framework](image-url)
3.5 RQ3

The theoretical research framework of the reported success factors and barriers is presented in Figure 3. The framework is based on the identified success factors, barriers and their relationship with the SPI implementation as discussed in sections 3.1 and 3.3. The framework consists of 18 independent variables (i.e. the reported success factors, barriers) and single dependent variable (i.e. humans’ related SPI implementation in GSD). We have also identified one moderating variable (i.e. organizational size). It has been discussed in software engineering studies that small, medium and large size organizations have various operational differences.

Majority of the large size organizations adopts formal standards and models, while small and medium focus on informal processes [11]. Small and medium size organizations have concerns about the budget need to acquire and sustain the formal processes [8]. In this study, we have investigated that whether small, medium and large organizations implement process improvement practices differently [12]. It is important to investigate the significance of organizational size in the context of SPI implementation.

The following hypothesis has been developed to investigate the relationship of organizational size and the SPI success. 

H19: The reported success factors and barriers influence the SPI implementation activities for a large organization to a greater extent than that of small and medium organization.

It is shown in Figure 3, that each success factor has a positive relationship with the dependent variable and identified barriers have negative association. The framework is presenting a relational view of the independent, moderating and dependent variables. The relationship of the variables is presented based on the proposed hypotheses.

4. THREATS TO VALIDITY

During the SLR process, most of the data were extracted by the first author of the article. However, we tried to alleviate this threat by observing any unclear problems and discussing them together, still there exists a higher risk that a single researcher could be biased and continually extract the wrong data. However, the co-authors were involved to randomly check different steps in the SLR.

One possible threat to internal validity is that for any selected SLR article, the reported factors might not have assuredly described their primary causes. It might possible that in some articles there may have been a trend to report specific types of factors. Most of the selected articles reported findings of empirical studies, experience reports and case studies which might be subject to the publication bias.

In addition, most of the authors of the selected 19 primary studies are from academia, which might lack a deep knowledge about the current trends of SPI programs in the industry. In order to mitigate this threat, we planned to perform an industrial empirical study to assess the validity of the identified success factors and barriers.

5. CONCLUSIONS AND FUTURE WORK

Presently majority of the software development firms are adopting the phenomena of global software development. The fast growing nature of GSD motivated us to identify humans related success factors that can positively affect the SPI program. We have followed SLR approach and 19 research articles related to humans oriented SPI and GSD were selected. We have identified a total of ten success factors and eight barriers from the selected research articles. In the identified ten success factors, total five factors were categorized as the most critical factors. Furthermore, three barriers were ranked as the most critical barriers. The identified critical success factors and critical barriers can act as a guide for the SPI practitioners to effectively implement the SPI activities in the domain of GSD. The critical factors are presenting the key areas which have much more impact on process improvement implementation as compared to the other identified factors. We have developed the theoretical framework of the identified success factors, barriers and the SPI implementation based on their relationship discussed in the literature. We believe that the findings of this research work can possibly result into tackling the problems associated with the humans related process improvement activities, which is a key to the success and progression of GSD organizations.

The following points highlight the future directions of this study.

- Empirically validate the identified factors in the context of humans related software process improvement.
- Investigate the best practices to address the identified success factors and barriers.
- Conduct industrial empirical study with the SPI practitioners to identify additional success factors, barriers and their practices.
- Empirically assess the theoretical framework of this study in order to test the hypotheses.

The concluding aim of this study is to develop a model that could address the humans’ related aspects of software process improvement in GSD. This model will assist the GSD organizations to assess and measure their process improvement readiness prior to SPI implementation.

6. ACKNOWLEDGMENT

This research is supported by the City University of Hong Kong research funds (Project No. 7004683, 7004474 and 7200354).

7. REFERENCES


**APPENDIX**

Selected primary studies using SLR

<table>
<thead>
<tr>
<th>ID</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference</td>
<td>Abstract</td>
</tr>
<tr>
<td>-----------</td>
<td>----------</td>
</tr>
</tbody>
</table>
ABOUT THE AUTHORS:

Arif Ali Khan received his BS in Software Engineering from University of Science and Technology Bannu, Pakistan in 2010. Arif has MSc by research in Information Technology from Universiti Teknologi PETRONAS, Malaysia. He is currently pursuing PhD degree with the department of computer science, City University of Hong Kong. He is an active researcher in the field of empirical software engineering. He is interested in Software Process Improvement, 3C's (Communication, Coordination, Control), Global Software Development and Evidence-based Software Engineering. He has participated in and managed several software Engineering related research projects. He is a student member of IEEE and ACM.

Jacky Keung Jacky Keung received his B.Sc. (Hons) in Computer Science from the University of Sydney, and his Ph.D. in Software Engineering from the University of New South Wales, Australia. He is Assistant Professor in the Department of Computer Science, City University of Hong Kong. His main research area is in software effort and cost estimation, empirical modeling and evaluation of complex systems, and intensive data mining for software engineering datasets. He has published papers in prestigious journals including IEEE-TSE, IEEE-SOFTWARE, EMSE and many other leading journals and conferences.

Mahmood Niazi is an active researcher in the field of empirical software engineering. He has published over 100 articles. He is interested in developing sustainable processes in order to develop systems which are reliable, secure and fulfils customer needs. His research interests are: Evidence-based Software Engineering, Requirements Engineering, Sustainable, Reliable and Secure Software Engineering Processes, Global and Distributed Software Engineering, Software Process Improvement and Software Engineering Project Management. Dr Niazi has got PhD from University of Technology Sydney Australia.

Shahid Hussain received his MSc (Computer Science) and MS (Software Engineering) degrees from Gomal University, DIK and City University of Science and Information Technology (CUSIT), Peshawar, Pakistan. Currently, he is pursuing PhD (Software Engineering) with the department of computer science, City University of Hong Kong. His research interests include the Software Design patterns and metrics, text mining, empirical studies, and software defect prediction. Mr. Shahid Hussain is the student member of ACM and IEEE.

Muhammad Manzoor Ilahi Tamimy is currently an Associate Professor at CIIT, Islamabad. He received his PhD from GSCAS, Beijing, China. The Ph.D Thesis was Title: “Detecting Outliers for Data Stream under Limited Resources” Supervised By Prof. Hongan Wang, Intelligence Engineering Laboratory Institute of Software, Chinese Academy of Sciences. Dr. Tamimy major areas of interest are Mining Data Streams, Ubiquitous Data Mining, Intrusion Detection, Distributed Data Mining, Machine learning and data mining, Real-Time system; Publish/Subscribe system; Wireless Sensor Networks, Active databases ECA Rules.
Write-Aware Memory Management for Hybrid SLC-MLC PCM Memory Systems

Chien-Chung Ho  
Academia Sinica  
kib65060@gmail.com

Yu-Ming Chang  
National Taiwan University  
d00922001@csie.ntu.edu.tw

Yuan-Hao Chang  
Academia Sinica  
johnson@iis.sinica.edu.tw

Hsiu-Chang Chen  
National Taiwan University  
r01922044@csie.ntu.edu.tw

Tei-Wei Kuo  
National Taiwan University  
ktw@csie.ntu.edu.tw

ABSTRACT
In recent years, phase-change memory (PCM) has generated a great deal of interest because of its byte addressability and non-volatility properties. It is regarded as a good alternative storage medium that can reduce the performance gap between the main memory and the secondary storage in computing systems. To further reduce the bit cost of PCM, the development trend of PCM goes from single-level-cell (SLC) to multi-level-cell (MLC) technology. However, the worse endurance and the intolerable long write latency hinder a MLC PCM from being used as the main memory of computing systems. In this work, we propose a write-aware memory management design to facilitate enabling the use of hybrid PCM as main memory to achieve a better trade-off between the cost and the performance of PCM-based computing systems, where the hybrid PCM is composed of SLC PCM and MLC PCM. In particular, the proposed design can be seamlessly integrated into the inherent memory management of modern operation systems without additional hardware components. The capability of the proposed design was evaluated by a series of experiments, for which it was shown that the proposed design could greatly improve the read and write performance of hybrid PCM memory system up to 30%. At the same time, our proposed idea can significantly extend the lifetime of the investigated hybrid PCM architecture up to 1174 times, compared to existing approaches.

CCS Concepts
- Hardware → Non-volatile memory;  
  Software and its engineering → Memory management;  
  Computer systems organization → Embedded systems;

Keywords
Memory Management; Phase Change Memory; Endurance; Wear Leveling; SLC; MLC; MMU

1. INTRODUCTION
Due to the shrinking barrier of DRAM and the heavy burden of power consumption, modern computing systems are thirsting for a better alternative of working memory. Recently, phase change memory (PCM), which has the characteristics of non-volatility, byte-addressability, and good scalability, has drawn massive attention and is deemed to be a possible promising candidate. In order to further seek for a low-cost environment, multi-level-cell (MLC) technology is a must solution to double the PCM capacity by storing two-bit information in each memory cell. However, the use of the MLC technology degrades the access performance and the endurance of PCM significantly. To achieve a good compromise between cost and performance, it is desirable to have a new system design that adopts hybrid PCM, i.e., SLC PCM and MLC PCM, as main memory. With such a new system design, new memory management designs in operating systems are needed. Therefore, we are motivated to enable the use of hybrid PCM with a software solution, which can obtain the first-hand OS knowledge to facilitate the management of hybrid PCM to take advantage of the high-performance SLC PCM and the high-capacity MLC PCM simultaneously.

Today, PCM has a good potential in replacing traditional DRAM; however, it has a fundamental endurance problem. That is, each memory cell has a limited number of write cycles. To deal with this problem, existing researches focus on the solutions to improve the effective lifetime of PCM-based systems. In general, the solutions for the systems that adopt PCM as main memory can be categorized into three types. The first type aims to eliminate the data traffic written to PCM [7, 9, 19]. The second type tries to reuse worn-out pages in which only a small number of bits are broken so that the effective lifetime of the pages is extended [14, 17]. The last type is to spread write operations over the entire PCM space as evenly as possible to prevent wearing out any PCM cell excessively [11, 15, 19]. In addition, it was also proposed to prolong the system lifetime with a optimized file-system design when PCM is used as a storage medium [3].

In addition to the endurance problem, some research directions focus on resolving the PCM’s slow-write issue. That is, the latency of a write operation is several times longer than that of a read operation. Thus, some researchers proposed to optimize the design of caches with STT-RAM and SRAM to improve both the overall system performance and the PCM lifetime [2, 8, 13]. Some other researchers also proposed to exploit compiler-assisted techniques to derive an optimized data allocation to improve the system performance with a software-controlled cache [6, 16]. Qureshi et al. proposed to dynamically switch PCM cells between MLC and SLC modes for different workloads; however, this solution shall rely on additional hardware components to support the switch op.
DRAM SLC PCM MLC PCM

<table>
<thead>
<tr>
<th>Attributes</th>
<th>DRAM</th>
<th>SLC PCM</th>
<th>MLC PCM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte Addressability</td>
<td>Yes</td>
<td>Yes</td>
<td>YES</td>
</tr>
<tr>
<td>Read Latency</td>
<td>&lt; 10ns</td>
<td>10ns</td>
<td>44ns</td>
</tr>
<tr>
<td>Write Latency</td>
<td>&lt; 10ns</td>
<td>100ns</td>
<td>395ns</td>
</tr>
<tr>
<td>Endurance</td>
<td>&gt; $10^{15}$</td>
<td>$10^6$</td>
<td>$10^6$</td>
</tr>
</tbody>
</table>

A high resistance state (referred to as amorphous phase) can be obtained to represent “1” by a SET operation that heats up the material above the melting temperature and then quenches the heating process quickly. Note that PCM has the asymmetric read/write property, where the latency of a write operation, i.e., SET/RESET, is longer than that of a read operation. In addition, the write operation requires an iterative heating process, and it will lead to higher power consumption.

To further reduce the bit cost of PCM by increasing the bit density, multi-level-cell (MLC) technology is a popular approach that can store two-bit information in a PCM cell by introducing two additional intermediate resistance states between the crystalline state and the amorphous state. Note that, four distinct resistance states could represent “11”, “10”, “00”, and “01”. As a result, comparing with charging-based memory like DRAM, PCM has better density. However, such a technology will lengthen the overall latency in programming PCM cells since it is necessary to perform more write-and-verify operations to form a narrower resistance distribution. In addition, even though PCM can provide a access speed comparable to that of DRAM, it suffers the endurance problem, which does not exist in DRAM. That is, a PCM cell can only sustain a limited number of write cycles because of the degradation of the electrode-storage contact [9]. In particular, PCM with MLC technology will further exacerbate the endurance problem because a larger number of writes is required on programming a PCM cell. As shown in Table 1 [5, 9], the endurance gap of single-level-cell (SLC) PCM (in which a cell can store one-bit information) and MLC PCM (in which a cell can store two-bit information) is up to two orders of magnitude. In this work, we consider to adopt a hybrid PCM, composed of SLC PCM and MLC PCM, as the main memory of the considered computer system, where the high-cost SLC PCM can deliver high performance/endurance and the low-cost MLC PCM could provide high capacity.

### 2.2 Memory Management

In most modern computer systems, a program execution should rely on the support of memory management techniques. Typically, the technique is realized with a hardware component, memory management unit (MMU), and software components, page manager and free space manager, as shown in Figure 1. The MMU built in the CPU is responsible for translating the virtual addresses issued by running processes to the corresponding physical addresses of a byte-addressable memory, which is hybrid PCM in this work. The free space manager is to allocate free space for serving memory allocation requests. The page manager concentrates on the deallocation or the reclamation of in-used physical pages (referred to as...
In order to preserve as many as possible frequently-accessed pages on main memory, the page reclaiming management in the page manager should be able to identify and evict the pages that are less-likely to be accessed recently. In a Linux-based OS, the page reclaiming management is typically implemented with two lists, i.e., active list and inactive list [1]. The active (resp. inactive) list maintains the in-used pages with high (resp. low) access frequency in recent time, and both lists are manipulated in least-recently-used (LRU) fashion. Each in-used page is associated with two one-bit flags, i.e., \( P_{\text{active}} \) and \( P_{\text{referenced}} \), as depicted in Figure 2 [1]. The \( P_{\text{active}} \) flag is used to indicate whether the page is in the active list or in the inactive list, and the \( P_{\text{referenced}} \) flag is exploited to implement the second-chance strategy to determine the movement of pages between the active and inactive lists. For example, if a page with its \( P_{\text{referenced}} \) flags cleared, its \( P_{\text{referenced}} \) flag is set when it is accessed; if a page is accessed with its \( P_{\text{active}} \) flag cleared and its \( P_{\text{referenced}} \) flag set, it is moved to the active list from the inactive list with its \( P_{\text{active}} \) flag set and its \( P_{\text{referenced}} \) flag cleared, as shown in the “accessed” arrow of Figure 2. If a page with its \( P_{\text{referenced}} \) flag set has not been accessed for a while, its \( P_{\text{referenced}} \) flag will be cleared by the periodic page scanner of the page reclaiming manager, as shown in the “timeout” arrow of Figure 2. If the number of free pages is getting too low, all the pages in the active list are possibly demoted for reclaiming more pages, as shown in the “refill” arrow of Figure 2. As a result, the pages in the inactive list with their \( P_{\text{referenced}} \) flags cleared will be the best candidate for page reclamion (or deallocation).

### 2.3 Motivation
To investigate the practicability of using a hybrid PCM as main memory, we are motivated to propose a software solution without extensive modifications on hardware components. In particular, we are interested in the software design of memory management with minimized modifications to typical memory systems. Our goal is to maximize the access performance by utilizing (1) the asymmetric read/write property of PCM as well as (2) the performance gap between SLC PCM and MLC PCM in a hybrid PCM. Especially, the reduction of write operations is our main focus since the write latency of PCM is far longer than that of DRAM. At the same time, we shall also consider the endurance problem, which is a crucial issue in the design of PCM-based memory systems. In contrast to the past works that focus on wear leveling designs for lifetime extension by evenly distributing write cycles over all the PCM cells, we seek for a lightweight software solution to simultaneously enhance the access performance and extend the endurance of hybrid PCM-based memory systems. The technical challenges fall on (1) how to keep frequently accessed pages (resp. infrequently accessed pages) on SLC PCM (resp. MLC PCM) precisely and efficiently for performance enhancement and (2) how to avoid allocating old pages, i.e., pages with high write cycles, for lifetime enhancement. In the meantime, the compatibility issue will be considered to minimize the modification to existing memory management designs.

### 3. HYBRID-PCM-AWARE MEMORY MANAGEMENT

#### 3.1 Overview
In this section, we present a new memory management design that can be incorporated into the existing memory management of operating systems to improve the write performance and the effective lifetime of a hybrid PCM-based memory system. To better take advantage of the underlying hybrid PCM, our main idea is to distribute most of the memory writes to the high-performance SLC PCM rather low-performance MLC PCM and to avoid allocating old pages. To this end, we design a write-aware page manager and a lifetime-aware free space manager into the existing memory management of operating systems. Note that, the hybrid PCM is assumed to consist of a smaller capacity of SLC PCM and a larger capacity of MLC PCM for the cost consideration. The write-aware page manager tries to reallocate data of different access frequencies between SLC PCM and MLC PCM, and to swap the mapping between virtual addresses and the physical addresses by modifying the corresponding entries in the page table (see Section 3.2). The lifetime-aware free space manager aims to allocate young enough free pages with limited or nearly-zero searching or sorting costs (see Section 3.3). In the most popular operating system, i.e., Linux, the page manager is used to periodically reclaim (or release) in-used pages and the free space manager is invoked to allocate free pages by using the buddy memory allocator. To demonstrate the practicality of the proposed design, this work will focus our discussion on how to realize our idea in the two managers (shown in Figure 3) in the following sections (see Sections 3.2 and 3.3).
3.2 Write-Aware Page Manager

To minimize the number of frequently-written pages on SLC PCM and to manage all in-used pages on MLC PCM, the proposed write-aware page manager will relocate in-used pages with the proposed data structure introduced in Section 3.2.1. With consideration of the existing design in Linux kernel, the proposed page manager is further divided into parts. The first part is the management of page accesses that is to relocate data between SLC PCM and MLC PCM to minimize the average write latency (see Section 3.2.2). The second part is the management of page reclamation that is to manage and reclaim infrequently-accessed pages when there are not enough free pages or free memory space (see Section 3.2.3).

3.2.1 Data Structure

The proposed write-aware page manager maintains three LRU lists, namely write active list, read active list, and inactive list, to manage all in-used pages that have different access behaviors. Each list manages its own pages in the least-recently-used (LRU) manner. The pages in the write active list is expected to receive more write operations, while the pages in the read active list should be more likely read. The pages that are less likely written and read would be put in the inactive list. With such a classification, the in-used pages in the write active list should be allocated from the SLC PCM to offset the adverse impact on performance because of the long write latency of PCM, especially MLC PCM. For the pages in the read active list and the inactive list, we proposed to let them be allocated from MLC PCM that has a large space and a moderate read performance, as shown in Figure 4. The rationale behind this design is two-fold. First, the read performance of MLC PCM is comparable with that of SLC PCM such that the frequently-read pages in MLC PCM could still sustain an acceptable read performance. Second, typical access patterns on main memory are considerably skew. This phenomenon implies that most page data are seldom accessed and therefore are suitable to be stored in MLC PCM. To manage the location of each in-used page among the three LRU lists, three one-bit flags, i.e., (PG_wActive, PG_rActive, PG_referred), are introduced and included in each page’s descriptor to identify the access behavior of each page. Note that, A page descriptor is a data structure defined in the implementation of Linux’s memory management to describe a page’s status such as “dirty” and “locked”. The proposed three flags are stored in the reserved locations of the descriptor. In addition, the PG_wActive flag (resp. PG_rActive flag) indicates whether its corresponding page is in the write (resp. read) active list or not, and each page in the inactive list is with these two flags cleared. The remaining flag, i.e., PG_referred, is used to implement the second-chance strategy.

3.2.2 The Management of Page Access

Figure 5 shows an example to illustrate the management design for a page accessed at different conditions. The design aims to find out frequently-accessed pages and put them in the write active list, where the pages in this list shall be allocated from the SLC PCM to have a better performance. As the example in Figure 5 shows, each step in Steps 1–4 has a corresponding “Access” arrow. When a new page is needed and allocated (Step 1), it should be allocated from MLC PCM, inserted into the inactive list with the three flags (PG_wActive, PG_rActive, PG_referred) cleared as (0, 0, 0). If this page is accessed again, its PG_referred flag will be set to make the three flags become (0, 0, 1), and this page will still stay in the same list, i.e., inactive list (Step 2). Such a management design is able to avoid the unnecessary page data movement among different lists or different types of PCM. This is because many applications such as video streaming usually have a one-time-access property; more specifically, many pages allocated for this kind of applications are only accessed once before it is reclaimed/released. Therefore, these pages should not be mistakenly identified as frequently-accessed pages, and should not be moved to the high-cost, small-capacity SLC PCM. Note that, the page moving between the inactive list and the read active list only involves the modification of pointers and flags because pages of both the two lists are in the same memory, i.e., MLC PCM. On the contrary, moving a page out of the write active list needs an extra operation to move the page data from an SLC PCM page to an MLC PCM page, and moving a page into the write active list needs an extra operation to relocate the page data from an MLC PCM page to an SLC PCM page. Note that a page allocated from SLC PCM is called an SLC PCM page and a page allocated from MLC PCM is referred to as MLC PCM page.

If an accessed page in the inactive list is accessed again, this page
more likely contains hot data and should be moved to one of the read/write active lists (Step 3). If the access is a write operation, the data on the accessed page should be copied to a free SLC PCM page, i.e., a free page in the SLC PCM space, allocated by the free space manager. Then, the page is inserted into the write active list, and the corresponding flags are cleared except the \( PG\_{\text{wActive}} \) flag, i.e. \((1, 0, 0)\). This is because the latency of a write operation is relatively longer than that of a read operation so that the operation should receive a high priority to use SLC PCM for the performance consideration. Meanwhile, the reason to clear the \( PG\_{\text{referenced}} \) flag is that we could move some page data into the SLC PCM space (or the write active list) but these data could become (or could be in fact) infrequently accessed; thus, we make them have higher chances to be swapped to MLC PCM since the pages with the \( PG\_{\text{referenced}} \) flag cleared will be reclaimed first. The rationale behind this is that this accessed data could be written several times in the near future according to the prediction from the \( PG\_{\text{referenced}} \) flag, i.e., an accessed page in the inactive list is written. On the contrary, if the access is a read operation, the moving destination depends on the number of free pages in SLC PCM. If the number of free SLC PCM pages is sufficient, the page data can be directly moved to a free SLC PCM page, which is then inserted into the write active list, so as to further improve the read performance. However, if there are not enough free SLC PCM pages, the page data should remain in its original MLC PCM page, which is then moved to the read active list. Note that any page in the read active list or in the write active list will be make its \( PG\_{\text{referenced}} \) flag set if its flag was not set (Step 4).

**Algorithm 1: MARK-PAGE-ACCESSED**

**Input:** page, type

**Output:**

If \( page.\) \( PG\_{\text{referenced}} = \) FALSE then

if isNot (page, \( PG\_{\text{wActive}} = \) TRUE and type = READ) then

\( page.\) \( PG\_{\text{referenced}} \leftarrow \) TRUE

else /* \( PG\_{\text{referenced}} \) is set */

if page.\( PG\_{\text{wActive}} = \) TRUE then

\( \text{return;} \)

if type = Write then /* A write operation */

\( \text{MOVE-TO-WRITEACTIVELIST}(page) \)

else /* A read operation */

if \( \text{IS-FREE-SPACE-SUFFICIENT-SLC-PCM}() = \) TRUE then

\( \text{MOVE-TO-WRITEACTIVELIST}(page) \)

else if page.\( PG\_{rActive} = \) FALSE then

\( \text{MOVE-TO-READACTIVELIST}(page) \)

Algorithm 1 shows procedure MARK-PAGE-ACCESSED, which is invoked on each access to the hybrid PCM. The algorithm realizes the management design of page accesses, as shown in Figure 5, where the input page stores the address of the accessed page and the input type stores the operation type, e.g., read and write. When a page with the \( PG\_{\text{referenced}} \) flag cleared is accessed, the page remains in the same list and its \( PG\_{\text{referenced}} \) flag shall be set except that this page is in the write active list and the type of the access operation is a write operation (Lines 1-1). However, if a page with the \( PG\_{\text{referenced}} \) flag set is accessed, the page data are most likely hot and is good to be stored in SLC PCM space. However, if this page has already maintained in the write active list and allocated from SLC PCM space, this page should be kept in the same state without further operations (Lines 1-1). If this page is not in the write active list and the access type is write, this page should be moved to the write active list directly (Lines 1-1). On the other hand, if the access type is read and the free space of the SLC PCM space is sufficient, the page is moved to the write active list as well (Lines 1-1). Otherwise, the page should remain in the same memory space, i.e., MLC PCM space, and is moved to the read active list if this page is originally in the inactive list (Lines 1-1).

### 3.2.3 The Management of Page Reclamation

![Figure 6: An example of page reclamation flow](image)

Figure 6 shows an example to illustrate the management design of the page reclamation. It is to explain how victim pages are selected and reclaimed by the page scanner, where victim pages are the pages that are selected for space reclamation. The page scanner is a program that is periodically invoked to circularly scan pages in the lists. In each invocation, it selects one proper page and adjust the flags of the selected page; based on the flags of the selected page, this scanner could be possibly reclaim the selected page as well. The main focus of the page scanner is (1) to find out the least-recently-used page data and reclaim their corresponding pages to release more space for accommodating new data and (2) to move the data that has become cold from SLC PCM pages (maintained in the write active list) to MLC PCM pages, so as to avoid the unnecessary space occupancy of SLC PCM. In the example of Figure 6, each step of Steps 1–4 has a corresponding “Timeout” arrow. If a page with its \( PG\_{\text{referenced}} \) flag set is scanned and selected by the page scanner, this flag will be cleared immediately (Step 1 or Step 3) no matter which list the page stays. Because if the page data stored in the scanned page is still hot (or frequently accessed), the \( PG\_{\text{referenced}} \) flag will be set back in the near future. Conversely, if the page data of the scanned page become cold (or infrequently accessed), this page will have a higher probability to be moved to MLC PCM or reclaimed/deallocated. Therefore, if a page, in either the write active list or the read active list, has its \( PG\_{\text{referenced}} \) cleared and selected/scanned by the page scanner again, it should be moved to the inactive list (Step 2). When a page is just swapped out from one of the two active lists, its \( PG\_{\text{referenced}} \) flag is set to give it a second chance and to avoid being reclaimed immediately. This is because when a page is moved to another list, it could be
scanned by the page scanner immediately because the page scanner scan each list individually and circularly. Finally, if a page with all flags cleared, i.e., (0, 0, 0), is scanned and selected, this page is reclaimed and its page data are moved to the swap space to return a free page (Step 4).

Algorithm 2: PAGE-REFERENCED

| Input: page |
| Output: |
| if page.PG_referenced = TRUE then |
| else /* Page is with PG_referenced clear */ |
| if page.PG_wActive = TRUE or page.PG_rActive = TRUE then |
| RECLAIM (page) |
| else |
| MOVE-TO-INACTIVE-LIST (page) |

Algorithm 2 shows the procedure of PAGE-REFERENCED(), and it is invoked on each execution of the page scanner. The algorithm realizes the proposed design illustrated in Figure 6. The design logic is quite simple such that it can be integrated in the page scanner of existing Linux with minimized modification and only incurs a negligible overhead. Note that this algorithm is integrated into the page scanner of Linux in our implementation; nonetheless, this algorithm can be also implemented as an independent function called right after the execution of the page scanner. In the algorithm, a page with its PG_referenced flag set is cleared immediately when it is selected by the page scanner (Lines 2-2). However, if a page with its PG_referenced flag cleared, it is moved to the inactive list no matter whether it is in either the write active list or the read active list (Lines 2-2). However, if this page is in the inactive list, this page shall be reclaimed and swapped out from the PCM main memory (Lines 2-2).

3.3 Lifetime-Aware Free Space Manager

PCM suffers from the endurance problem, so that each PCM page can only be written with a limited number of times. Thus, in this section, a lifetime-aware free space manager is presented to extend the effective lifetime of the hybrid PCM. The objective is to preferentially allocate young pages and avoid the allocation of old pages, where young (resp. old) means that a page undergoes a small (resp. large) number of write times. Our main idea is to let young pages be close to and old pages be farther from the (free) location of page allocation. To this end, a lifetime-aware heap tree for the free space manager is proposed to manage page ages and support the query for page ages, so as to facilitate the performance on page allocation, e.g., Step 1 of Figure 5, and page deallocation, e.g., Step 4 of Figure 6 (see Section 3.3.1).

3.3.1 A Lifetime-aware Heap Tree

The lifetime-aware heap tree is to manage page ages and support the query for page ages, where page age is the number of write cycles of a page. As shown in Figure 7, this heap tree is able to inquire the oldest page among a group of continuous pages, where the number of pages in a group ranges from \(2^{0}\) to \(2^{N}\). The \(N\) is a positive integer, and \(2^{N}\) is the upper bound of the number of pages in the hybrid PCM. The node at Level \(N\) of the heap tree stores the lifetime (or the number of write cycles) of the oldest page for a group of \(2^{N}\) continuous pages. The heap tree is a complete binary tree and the value stored in each node must be either greater than or equal to that of its child nodes. For example, the second node in Level 1 of the heap tree manages \(2^1\) continuous pages, i.e., page 2 and page 3, and its stored value 5 is obtained from its child node for page 2 since page 2 is older than page 3. Similarly, the node in the top level of the heap tree stores the largest value of lifetime for the entire memory space. Thus, the proposed heap tree is able to easily compare the lifetime of two groups with any number of continuous pages. Moreover, the time complexity to inquire the lifetime value for a group of continuous pages is \(O(1)\) because any group of continuous pages has a corresponding node at a fixed location. Note that, since the underlying device is a hybrid PCM, both two memory spaces, i.e., SLC PCM and MLC PCM, shall have its lifetime-aware heap tree respectively.

In addition, to greatly reduce the maintenance cost, e.g., frequent update due to the intensive writes on PCM pages, for the heap tree, a new definition of lifetime is proposed to record how old a page is. As shown in the right part of Figure 7, the value 0 represents that the corresponding page has at most \(\frac{1}{2}\) of lifetime to write, the value 1 represents that the corresponding page has at most \(\frac{1}{4}\) of lifetime to write, and so on. The larger the value is, the older the page will be. The rationale behind this design is that it can mitigate the management overhead for frequently-updated young pages and more precisely observe the increase of writes for old pages, especially when an old page is close to be worn out. Even though this design might need to cascade updating the value from the leaf node to the root node when the lifetime value in a leaf node is changed, the new definition of lifetime can avoid the cascading updates most of the time. This is because our design aims to allocate as many young pages as possible and avoid the allocation of old pages, and the lifetime of young (resp. old) pages are managed in a coarse-grained (resp. fine-grained) manner.

3.3.2 The Lifetime-aware Management for Free Page Allocation and In-used Page Deallocation

In this section, a lifetime-aware management for free page allocation and in-used page deallocation is presented with the cooperation of the lifetime-aware heap tree. This management is integrated with the most popular memory allocator, i.e., buddy memory allocator.
to demonstrate the practicability of the proposed main idea, which lets young pages be closer to and old pages be farther from the location of memory/page allocation. In this management, our idea is not only limited to one specific memory allocator but also can be applied to memory allocators, because the proposed heap tree manages the lifetime information for the group with any number of continuous pages. The page (or memory) allocation from MLC PCM might occur when a new page is needed, and the page allocation from SLC PCM occurs when a page is moved to the write active list. The page deallocation might occur on MLC PCM when a page in the inactive list is reclaimed and might occur on SLC PCM when a page with its PG_referred flag cleared is moved to the inactive list. Note that we refer to memory allocation/deallocation and page allocation/deallocation interchangeably when there is no ambiguity.

![Lifetime-aware Heap Tree](image)

**Figure 8: Free page allocation and in-used page deallocation**

Figure 8 illustrates the free page allocation and the in-used page deallocation with the proposed lifetime-aware heap tree. In the proposed free space manager, the free space is allocated in the unit of $2^n$ continuous pages, where $n$ is an integer ranging from 0 to a specified upper limit. For example, the pointer marked with “Order 0” links a number of groups having $2^n$ free pages, e.g., the group with page 99, the pointer marked with “Order 1” links a number of groups having 2$^{2n}$ continuous free pages, e.g., the group with pages 6 and 7, and so on. In a traditional memory management design, its memory allocation usually return the first group in the linked list with a specified order, and its memory deallocation just inserts the group of free pages into the tail of the linked list with the same order. In the proposed memory deallocation, a page identified as an old page should be inserted to the tail of the corresponding linked list; conversely, a page identified as a young one should be inserted to the head of the corresponding list. In this way, when the proposed memory allocation retrieves the first group of free pages from the head of the linked list, it more likely (resp. less likely) gets young (resp. old) pages. In this example, if a memory deallocation occurs and will return a group with two continuous pages, e.g., pages 2 and 3, to the free space manager, the returned position depends on the lifetime value inquired from the lifetime-aware heap tree. If the lifetime value, i.e., 5, is greater than a predefined threshold, e.g., 3, this group of pages should be inserted to the tail of the linked list of “Order 1”. Thus, the allocation of this group of pages, which is identified as old, can be postponed effectively. Note that, the time complexity of inserting a group of pages to either the head or the tail in a linked list is $O(1)$ so that the extra overhead is negligible as well.

### 4. PERFORMANCE EVALUATION

#### 4.1 Experiment Setup

In this section, a series of experiments were carried out to demonstrate the practicability, the capability and the effectiveness of the proposed memory management for a hybrid PCM based memory system. Our objective is to simultaneously improve the access latency and extend the system lifetime. Therefore, the experiments are evaluated in terms of the performance and the lifetime. The performance of our proposed idea is evaluated based on the average read/write latency, number of total executed read/write operations. On the other hand, the lifetime is evaluated based on the maximum write count. In our experiments, we used four hybrid PCM schemes evaluated with adopting the different sizes of SLC PCM on the setting of main memory. The total size of main memory (SLC PCM and MLC PCM) for all of the configurations are set with 4GB, but the size of SLC PCM of each configuration is various (64MB, 128MB, 256MB and 512MB). To investigate the possible performance loss of using the hybrid PCM, the baseline is compared with the setting using a 4GB SLC PCM which theoretically delivers the best performance but has the highest cost (referred to as “Baseline”). In the same time, the above mentioned configurations are also compared with the approach proposed by Zhao et al. in 2014 since the work is also designed for the hybrid PCM based memory system and the main objective of it is improve the endurance (referred to as “DAC work”) [18].

#### Table 2: Hardware Parameters

<table>
<thead>
<tr>
<th></th>
<th>Intel Core 2 Duo Processor (4M cache, 2.4GHz)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td></td>
<td>8KB</td>
</tr>
<tr>
<td>Cache Size</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory Size</td>
<td>Comparison 1: All SLC PCM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Comparison 2: Hybrid PCM with 64MB SLC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Comparison 3: Hybrid PCM with 128MB SLC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Comparison 4: Hybrid PCM with 256MB SLC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Comparison 5: DAC work [18]</td>
<td></td>
</tr>
<tr>
<td>PCM latency</td>
<td>SLC PCM: 10/40 (r/w) ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MLC PCM: 100/400 (r/w) ns</td>
<td></td>
</tr>
<tr>
<td>Endurance</td>
<td>SLC PCM: 10$^9$ times</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MLC PCM: 10$^9$ times</td>
<td></td>
</tr>
</tbody>
</table>

In our experiments, each configuration is conducted with a widely used virtual machine called QEMU [10]. The QEMU is additionally integrated with the proposed memory management and used to collect memory traces during the system execution [4]. In addition, the system is executed under four representative benchmarks, i.e., perlbench, bzip2, gcc, and libquantum, chosen from SPEC 2006 to investigate the results. The benchmark perlbench has 23.3% of branch instructions, 23.9% of load instructions and 11.5% of store instructions, bzip2 has 15.3% of branch instructions, 26.4% of load instructions and 8.9% of store instructions, gcc has 21.9% of branch instructions, 25.6% of load instructions and 13.1% of store instructions, libquantum has 27.3% of branch instructions, 14.4% of load instructions and 5.0% of store instructions. To ensure a fair comparison, the execution time of each benchmark are the same for each configuration. Also, each configuration uses the same hardware parameters as listed in Table 2 except for the memory. The read and write latency of SLC PCM are 10ns and 40ns respectively. read and write latency of MLC PCM are 100ns and 400ns respectively [5,9].
The endurance of SLC PCM and MLC PCM are $10^8$ times and $10^6$ times respectively.

4.2 Experiment Results

4.2.1 Performance Results

Figure 9 shows the results of normalized average read/write latency under various benchmarks, i.e., perlbench, bzip2, gcc, and libquantum. Our design is evaluated with adopting the different configurations for the size of SLC PCM. All the experimental results are normalized to the performance result of the baseline approach, which only adopts 4GB SLC PCM as the main memory. The experimental results show that our proposed design adopting a hybrid PCM can achieve nearly the same performance with the setting adopting a high-performance SLC PCM under all benchmarks. This is because the memory access has a very skew pattern that most of the writes concentrates on a small area of the memory space. As a result, the working set of pages are more likely the same and could be easily caught with the proposed write-aware page manager by leaving such kind of pages on the write active list of SLC PCM to sustain a good performance. On the other hand, it can also be found that the performance of the proposed design can also achieve the similar performance of the previous work proposed by Zhao et al. Especially, our write-aware management strategy can efficiently handle the benchmark which consists of large amount of branch and load instructions, like gcc. These results imply that replacing a small portion of MLC PCM with an SLC PCM can greatly enhance the overall performance with the proposed write-aware memory management.

Figure 10 shows the results of read/write distributions over the different approaches and configurations. As shown in Figure 10(a), it is observed that our proposed design can gather the majority of write operations in the SLC PCM area so as to prevent MLC PCM area from being frequently accessed so that the performance of our proposed design can be guaranteed. In addition, it is also observed that our proposed scheme can use MLC PCM to absorb large amount of read operations, as shown in Figure 10(b). Although the read latency of MLC PCM is longer than that of SLC PCM, it can still help on improving the system performance since the larger amount of sequential reads can efficiently reduce the impact caused by the read latency of MLC PCM. Our proposed scheme can take advantages of SLC PCM and MLC PCM; it exploits the high performance and high endurance properties of SLC PCM to absorb large amount of write requests so as to maintain the write performance, and it also exploit the larger capacity property of MLC PCM for the infrequently accessed data to reduce the accumulation of P/E cycles.

4.2.2 Lifetime Results

Figure 11 shows the results for the lifetime evaluation under different benchmarks. The y-axis denotes the number of instructions that main memory can serve, and it is normalized to the results of baseline approaches. The experimental results are also compared with the existing approach proposed by Zhao et al. It can be easily observed that the lifetime of our proposed schemes with adopting the different configuration of SLC PCM size can all significantly improve the lifetime of hybrid PCM based memory architecture. This is because that our proposed write-aware management design can exploit the high endurance property of SLC PCM to reduce the opportunities of serve hot data with MLC PCM. As a result, the wear-out speed of MLC PCM can be efficiently slowed down so that the overall lifetime of the hybrid PCM can be extended.
It can be easily observed that the curve of write count distribution with the proposed life-aware free space manager is more even than that without any proper management. The reason is that the proposed design tries to avoid the allocation of old pages by inserting them into the tail of the allocation linked lists. In addition, we can find that the maximal write count which determines the system failure time are greatly reduced on average by 32.78x under all benchmarks. Especially, when running the benchmark bzip, the maximal write count under the proposed design is 73.7x as small as that under a default memory management, as shown in Figure 12(b). These results show that the system failure time is greatly postponed to extend the system lifetime effectively. On the other hand, the results also show that our design does not impose much overhead on the SLC PCM are even we use SLC PCM to absorb large amounts of write requests so as to extend the lifetime of MLC PCM. Our proposed scheme can generate a graceful wear distributions over SLC and MLC PCM on the same device.

Figure 12: Wear distribution of top 1000 pages

5. CONCLUSION
The development trend of low cost and low energy consumption compels system designers to seek for a better replacement of DRAM.

6. ACKNOWLEDGEMENTS
This work was partially supported by the Ministry of Science and Technology under MOST Nos. 105-3111-Y-001-041, 105-2221-E-001-004-MY2 and 105-2221-E-001-013-MY3.

7. REFERENCES


ABOUT THE AUTHORS:

Chien-Chung Ho received his Ph.D. degree in Computer Science and Information Engineering from National Taiwan University in June 2016. Previously, he received his B.S. degree in Department of Computer Science from National Chiao Tung University in June 2010. His primary research interests primarily cover storage systems, file systems, operating systems, embedded systems, and computer system architecture. Up to now, he has 14 papers accepted/published by international conferences or journals.

Yu-Ming Chang received his Ph.D. degree in Computer Science and Information Engineering from National Taiwan University in June 2015. Previously, he received his M.S. degree in Networking and Multimedia of Computer Science and Information Engineering from National Taiwan University, Taipei, Taiwan, in June 2011, and he received his B.S. degree in Computer Science and Information Engineering from Chiao Tung University in June 2009. He was also a chief engineer in Emerging System Lab, a division of Macronix International Co., Ltd., Hsinchu, Taiwan. In addition, he was elected as a member of the Phi Tau Phi Scholastic Honor Society at M.S. degree in 2011. His primary research interests include operating systems, storage systems, and embedded systems.

Yuan-Hao Chang received his Ph.D. degree in Networking and Multimedia of Computer Science and Information Engineering from National Taiwan University, Taipei, Taiwan, in 2009. Now he is an assistant research fellow at the Institute of Information Science, Academia Sinica, Taipei, Taiwan, since Aug. 2011. Previously, he was an assistant professor at the Department of Electronic Engineering, National Taipei University of Technology, Taipei, Taiwan, between Feb. 2010 and Jul. 2011. His research interests include storage systems, embedded systems, and computer system architecture. He is a member of IEEE.

Hsiu-Chang Chen received the M.S. degree in computer science and information engineering from National Taiwan University in 2014. Now he is an software engineer in Infortrend Technology Inc.

Tei-Wei Kuo received his M.S. and Ph.D. degrees in Computer Sciences from the University of Texas at Austin in 1990 and 1994, respectively. He is the Executive Vice President for Academics and Research of the National Taiwan University, Taipei, Taiwan, and a distinguished professor of the Department of Computer Science and Information Engineering, where he was promoted to a distinguished professor in 2009. He was a distinguished research fellow and the Director of the Research Center for Information Technology Innovation, Academia Sinica between January 20, 2015, and July 31, 2016. His research areas include embedded systems, real-time systems, and non-volatile memory. He is a fellow of the ACM.
Large Scale Document Inversion using a Multi-threaded Computing System

Sungbo Jung  
University of Louisville  
Computer Engineering and Computer Science  
Louisville, KY 40292  
1-502-852-0467  
sungbo.jung@louisville.edu

Dar-Jen Chang  
University of Louisville  
Computer Engineering and Computer Science  
Louisville, KY 40292  
1-502-852-0472  
djchan01@louisville.edu

Juw Won Park  
University of Louisville  
Computer Engineering and Computer Science  
Louisville, KY 40292  
1-502-852-6307  
juw.park@louisville.edu

ABSTRACT

Current microprocessor architecture is moving towards multi-core/multi-threaded systems. This trend has led to a surge of interest in using multi-threaded computing devices, such as the Graphics Processing Unit (GPU), for general purpose computing. We can utilize the GPU in computation as a massive parallel co-processor because the GPU consists of multiple cores. The GPU is also an affordable, attractive, and user-programmable commodity. Nowadays a lot of information has been flooded into the digital domain around the world. Huge volume of data, such as digital libraries, social networking services, e-commerce product data, and reviews, etc., is produced or collected every moment with dramatic growth in size. Although the inverted index is a useful data structure that can be used for full text searches or document retrieval, a large number of documents will require a tremendous amount of time to create the index. The performance of document inversion can be improved by multi-thread or multi-core GPU. Our approach is to implement a linear-time, hash-based, single program multiple data (SPMD), document inversion algorithm on the NVIDIA GPU/CUDA programming platform utilizing the huge computational power of the GPU, to develop high performance solutions for document indexing. Our proposed parallel document inversion system shows 2-3 times faster performance than a sequential system on two different test datasets from PubMed abstract and e-commerce product reviews.

CCS Concepts

• Information systems→Information retrieval  
• Computing methodologies→Massively parallel and high-performance simulations.

Keywords

Graphics Processing Unit; GPU; high-performance computing; document inversion

1. INTRODUCTION

The rapid growth of network and computer technology leads to the immense production of information globally. This has resulted in a large number of text data in the digital domain, leading to the need for more efficient methods for searching databases. For example, PubMed, one of the largest collections of bioinformatics and biomedical publications, currently has over 26 million articles, and over 1,000,000 articles are added annually. With databases of this size, researchers must rely on automatic information retrieval systems that can quickly find articles relevant to their work. The demand for fast information retrieval has motivated the development of efficient indexing and searching techniques for any kind of data expressed as text.

An index is a list of important terms, including the locations where those terms appear in a document. If the document is divided into pages, the index also provides page numbers and line numbers for the terms. An index facilitates content-based searching by locating the topic word or phrase in the index and returning the corresponding locations. Thus, indexing plays a very important role in searching and retrieving documents. During the past decade, indexing is regarded an excellent text processing technique. An index links each term in a document to the list of all its occurrences. It allows efficient retrieval of all the occurrences of a term. This method usually requires a large amount of space, which can be up to several times the size of the original text. It also requires space for storing the text itself, because the text is hardly reproduced by the index. The process of generating an index of a document is called document inversion to emphasize that an index is the inverse function of a document. It transforms the sequence of symbols in the document into an index. A document can be construed as a mapping from the positions in the document to the terms that appear there.

Recently a trend in microprocessor design is to include multiple cores in one chip. Because the heat generated in a microprocessor is roughly quadratic of the clock rate, it becomes a barrier to increasing processor speed. With improvement of manufacturing technology, a processor can be made with smaller size circuits, from 90nm to 65nm down to 14nm, so a processor chipset can hold more circuits in the same size die. For these reasons, most of microprocessor vendors, such as Intel, AMD, and IBM, turned to multi-core architecture instead of high clock-speed single-core processors. The multi-core trend affects not only CPU chipset, but also graphics processing units. NVIDIA, one of the major GPU chipset vendors, launched a multi-core GPU chipset called G80 about 10 years ago. It is equipped with 128 streaming processors with 768MB DDR3 memory. Currently, the top of the line
NVIDIA chipssets (GP100) have 3584 streaming processors with 16GB HBM2 memory [5]. These NVIDIA GPU chipssets support Computing Unified Device Architecture (CUDA), which is an extended C language environment, to implement general purpose applications in the GPU. CUDA simplifies the development process to such a degree that many researchers now convert massive computation problems to the GPU platform. Many applications, such as molecular dynamics, N-body from physics, and the DNA-folding in bioinformatics, show a dramatic speed-up in GPU solutions.

2. RELATED WORK

Some research in the literature demonstrates that multi-core platforms achieve good performance for document indexing. A. Narang, et al. [12], uses a distributed indexing algorithm on IBM BlueGene/L supercomputer platform. Their system focused on high-throughput data handling, real-time scalability with increased data size, indexing latency, and distributed search across 8 to 512 nodes with 2GB-8GB size data. It shows 3-7 times improvement in indexing throughput and 10 times better indexing latency. D. P. Scarpazza [17] implemented a document inversion algorithm on the IBM cell broadband engine blade, which has 18 processing cores (16 synergistic processor elements and 2 power processor elements). He adopted a single instruction multiple data (SIMD) blocked hash-based inversion (BHBII) algorithm for the IBM cell blade. It is 200 times faster than the single pass in-memory indexing algorithm. Another scalable index construction approach was performed on multi-core CPUs by H. Yamada and M. Toyama [21]. They implemented multiple in-memory index and on-disk merging methods on two quad-core Xeon CPU equipped system with 30GB web-based document collection. N. Sophoclis, et al. presented an Arabic language indexing approach on a GPU with OpenCL [20]. Their experiment shows that a GPU Arabic indexer is 2.5 times faster than CPU one on overall performance. M. Frumkin presented a real time indexing method on a GPU [4]. The presenter uses Tokenizer(Sequential), Splitter, BucketSort, and Reduce implementation on a GPU. The system was tested with 4200 documents in a literature collection and 7M documents of Wikipedia web data, and was 3.1 times faster than a CPU searching the literature collection and 2.2 times faster in searching the Wikipedia data.

3. GPU AND CUDA

3.1 CUDA Programming

NVIDIA CUDA is a general purpose scalable parallelized programming model for highly parallel processing applications [13]. It is an abstract high-level language that has distinctive abstractions, such as a hierarchy of thread blocks, barrier synchronization, and shared memory structures. CUDA is well-suited for programming on multiple threaded multi-core GPUs. Many researchers and developers attempt to use CUDA for demanding computational applications in order to achieve dramatic improvements in speed. In earlier GPGPU designs, general purpose applications on the GPU must be mapped through the Application Programming Interface (API) because traditional GPUs had highly specialized pipeline designs. This structural property made it necessary for a programmer to write the programs to fit the graphics API. Sometimes the programmer needed to rework all the programs. The GPU has a global memory that can be addressed directly from the multiple sets of processor cores. The global memory makes the GPU architecture a more flexible and general programming model than previous GPGPU models. The global memory also allows programmers to implement data-parallelized kernels for the GPU easily. A processor core in the GPU can share the same traits with other processor cores. Thus, multiple processor cores can run independent threads in parallel at the same time.

3.2 SPMD Design

The GPGPU application systems use the GPU as a group of fast multiple coprocessors that execute data-parallelized kernel code, allowing the programmers to access the GPU cores via a single source code encompassing both CPU and GPU code. A kernel function operates in a Single Program Multiple Data (SPMD) fashion [1]. The SPMD concept extends Single Instruction Multiple Data (SIMD) by executing several instructions for each piece of data. A kernel function can be executed by the threads in order to run the data-parallelized operations. It is very efficient to utilize many threads in one operation. For full utilization of the GPU, fine-grained decomposition of work is required, which might cause redundant instructions in the threads. However, there are several restrictions in using the kernel functions. A CUDA kernel function cannot be recursive and it cannot use static variables. Kernel functions also need a non-variable type of parameters. The host (CPU) code can copy data between the CPUs memory and the GPUs global memory via API calls.

3.3 Thread, Block, and Grid

Thread execution on the GPU architecture consists of a three-level hierarchy, grid, block, and thread. The grid is the highest level. A block is a part of the grid, and there can be a maximum of 216 –1 blocks in the grid, organized in a one or two dimensional array. A thread is a part of a block, and there can be up to 512 threads in a block, organized in a one, two, or three dimensional array. Threads and blocks have their unique location numbers as threadID and blockIdx. The threads in the same block share the data through the shared memory. In CUDA, the function syncthreads() performs the barrier synchronization, which is the only synchronization method in CUDA. Additionally, the threads can be grouped into warps of up to 32 threads.

3.4 CUDA Memory Architecture

The GPU has several specially designed types of memory that have different latencies and different limitations [8]. The registers are fast and very limited size read-write per-thread memory in each SP. The local memory is slow, not cached, limited size read-write per-thread memory. The shared memory is a low-latency, fast, very limited size, read-write per-block memory in each SM. Shared memory is useful for data among the threads in a block. The global memory is a large, long-latency, slow, non-cached, read/write per-grid memory. It is used for communication between CPU and GPU as a default storage location. Constant and texture memory is used for only one-sided communication. The constant memory is slow, cached, limited size, read-only per-grid memory. The texture memory is slow, cached, large size, read-only per-grid memory. Table 1 indicates the CUDA memory types [16].

3.5 CUDA Toolkit and SDK

NVIDIA provides the CUDA toolkit and CUDA SDK as the interface and the examples for the CUDA programming environment. They are supported on Windows XP, Windows Vista, Mac OS X, and Linux platforms. The CUDA toolkit contains several libraries and the CUDA compiler NVCC. A CUDA program is written as a C/C++ program. NVCC separates the code
for the host (a regular C/C++ code) and the device (a CUDA native code). The CUDA toolkit also supports a simulation mode of CUDA which allows for debugging. The CUDA toolkit and SDK also provide GPU management functions, memory management functions, external graphical API supported functions, and some useful sample codes.

Table 1. Various CUDA memory type

<table>
<thead>
<tr>
<th>Memory</th>
<th>Location</th>
<th>Cached</th>
<th>Access</th>
<th>Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register</td>
<td>On-Chip</td>
<td>No</td>
<td>Read/Write</td>
<td>One thread</td>
</tr>
<tr>
<td>Local</td>
<td>Off-Chip</td>
<td>No</td>
<td>Read/Write</td>
<td>One thread</td>
</tr>
<tr>
<td>Shared</td>
<td>On-Chip</td>
<td>-</td>
<td>Read/Write</td>
<td>One thread</td>
</tr>
<tr>
<td>Global</td>
<td>Off-Chip</td>
<td>No</td>
<td>Read/Write</td>
<td>All grid</td>
</tr>
<tr>
<td>Constant</td>
<td>Off-Chip</td>
<td>Yes</td>
<td>Read only</td>
<td>All grid</td>
</tr>
<tr>
<td>Texture</td>
<td>Off-Chip</td>
<td>Yes</td>
<td>Read only</td>
<td>All grid</td>
</tr>
</tbody>
</table>

4. DOCUMENT INVERSION

4.1 Documents as Data

Documents come in a variety of forms. They may contain not only texts and graphics, but also audio, video, and multi-media data. In information retrieval, traditionally, information of a document can be represented by words or phrases. A word is the minimum component of a document, and it can be used for query and retrieval. A list of words is the simplistic representation of a document.

4.2 Text Preprocessing

Currently, various document file formats are used in the digital environment. From a simple text file with an extension .txt to a more complicated binary file with an extension .pdf, several document formats are used by different applications: such as simple text editors, Microsoft Word, Adobe Acrobat, web browser, and so on. These document formats have their own structures. In order to create the index from a document file, extracting text contents is required. Each document format requires a different extraction method. Before indexing, the extracted texts are processed to isolate the words or the terms. The term extraction process includes four main stages: tokenization, term generation, stopword removal, and stemming (See Figure 1).

4.3 Inverted Index Construction

In order to efficiently search and retrieve documents matching a query from the document collection, the list of terms and their postings are converted to an inverted index. There are many algorithms that construct an inverted index, such as sort-based inversion, memory-based inversion, and block-based inversion. Their performance varies by memory usage, storage types, processor types, and the number of passes through the data. Conventional document inversion algorithms process a document collection in the system memory. Both the document collection and the resulting dictionary are stored in the system memory. Thus the size of the document collection that can be processed by a conventional algorithm is restricted by the size of the system memory. If the document collection is too large to fit into the memory at once, document inversion cannot be performed with these methods. To deal with large document collections, contemporary algorithms process the documents in blocks of manageable sizes.

Figure 1. Document inversion process

4.3.1 Blocked Sort-based Inversion

Blocked sort-based inversion (BSBI) addresses the issue of insufficient system memory for a large document collection. It maps a term to a unique identifier (termID) so that the data to be sorted later have uniform sizes. Furthermore, it uses external storage to store the intermediate results. This inversion algorithm is a variant of sort-based inversion. To invert a large document collection, the collection is divided into blocks of equal sizes. The blocks are inverted individually. After the termIDs and the associated postings are extracted from the block, the algorithm sorts the list of termIDs and postings in the system memory. Then, it outputs the sorted list of the block to the external storage. After all blocks are inverted individually, the sorted lists of all the blocks are merged and sorted into a final inverted index. The algorithm shows excellent scaling properties for increasing sizes of the document collections, and the time complexity is $O(n \log n)$ [7]. However, blocked sort-based inversion is not suitable for a multi-core system with small local memory for each core, because the algorithm needs to maintain a large table that maps terms to termIDs.

4.3.2 Single-pass in Memory Inversion

For a very large document collection, the data structure, used in term-termID mapping, of the BSBI algorithm does not fit the system memory. Single-pass in memory inversion (SPIMI) is a more scalable algorithm to solve this problem by storing the terms directly rather than storing the termIDs and maintaining a huge table of the term-termID mapping [7]. Because it uses the dynamic allocation of memory when processing a block of documents, the sizes of terms and the associated posting lists can be expanded dynamically. When terms and postings are extracted from a document, the SPIMI algorithm adds the term and posting into the memory directly without sorting. The first occurrence of a term is added to the dictionary, and a new posting list is created. If multiple occurrences of a term appear, only the postings are added to the posting list. Often a hash table is used to store the terms. Specifically, a term is associated with a dynamic linked list of postings. Because the space for posting lists is dynamic, if the space is full, SPIMI can double the space of the posting lists. If the memory for the hash table is full, SPIMI saves its output block to an external storage and starts a new empty block. Once all the blocks are processed, they are merged and sorted into a final inverted index. Because sorting is not required during the first pass, this algorithm is faster than the BSBI algorithm.
4.3.3 Blocked Hash-based Inversion

Scarpazza [17] proposed the blocked hash-based inversion (BHBI) algorithm on the multi-core processor, the IBM cell broadband engine, for document inversion. The BHBI algorithm is proposed to solve several issues of the BSBI and the SPIMI with multi-core processor platforms. In a multi-core processor, local memory and registers of each processor are very useful for computation because they have very low access latency. Although they are very fast, the sizes of local memory and registers are too small to store large document collection. Thus, the BHBI is optimized for using small local memory in a multi-core processor system. To avoid dynamic allocation and overflows, input and output buffers for a block are set to the size of local memory. After reading a term from the input stream, instead of using the term directly in a dictionary, the BHBI uses a hash value of the term as its identifier. The term itself is not stored at all. Thus BHBI hashes the term, and adds to the output block an entry of the hash value, a document identifier, and a location. Multiple occurrences of a term will result in multiple entries in the output block. If the output block is full, all the entries in the block are sorted by their hash values. Then BHBI writes the output block to the global memory. After finishing all the blocks, BHBI performs a global merge sort that combines all the index blocks together in a global index. It is important to note that there is no hash table in BHBI. The hash function is used to generate identifiers of a uniform width. The width of the hash values must be chosen carefully. Because the hash values are used as the unique identifiers in the output block, a collision of hash values will lead to mistaking two different terms as the same term.

4.4 Sequential Document Inversion

The document inversion system consists of five components: input module, tokenizer, stopword remover, stemmer, and index maker. First, there are several file formats determined by the applications that created the files. Pure text files are preferred in text processing due to easy handling. However, most of the journal papers and articles are available in the PDF, PS, RTF, and DOC formats, or various marked-up file formats. In addition, these document files are converted into texts using file converters, and unnecessary structures and additional information are ignored. Second, after obtaining pure texts from the documents, the system performs tokenization. The tokenizer reads a document sequentially, and it separates a word, the minimum unit of the document, as a token from the input stream. If the input stream contains a blank space or a line separator, the tokenizer creates a new token after that point. In addition, the system keeps track of the location where the token occurs, and this information will be used later in document inversion. Third, after obtaining a token, the tokenizer passes it to the stopword remover. The stopword remover eliminates all stop words in the stopword list. The stopword list can have only one word per line with no leading or trailing spaces. Any character after the first word on a line is ignored. The apostrophe characters (') are stripped out, and letters before and after a apostrophe are concatenated. The underscore (_) and pound sign (#) characters are treated as normal characters. Hyphen (-), semicolon (;), and colon (:) characters are not allowed. These characters have the effect of breaking words, which is equivalent to two words on the same line. A minimal perfect hash function for stopwords is generated by the minimal perfect hash function generator, implemented by Zbigniew J. Czech [3]. It produces a minimal perfect hash function from the selected 641 stopwords. After a token is found by the tokenizer, the token is hashed with the minimal perfect hash function of stopwords. If a token is matched to a stopword, the token is discarded. If the token is not a stopword, it is passed to the stemming code. Fourth, the stemming algorithm creates the base (or stem) of each term. The system uses the Porter stemming algorithm [15] for a stemmer. In this stage, a stemmer removes the prefixes or suffixes by the defined rules in order to restore the bases. Lastly, after passing all the steps above, the refined terms, which will be used in indexing, are stored into the hash table. The system uses the separate chaining type hash table. The base of a term is used as a key in the hash table, and the value of docID and location, called a posting, is stored in the hash table. When the memory space for the index is full, the index is stored as a block file. Then the indexing system starts a new block. After finishing all the documents, all the index block files are merged and sorted using merge sort by the keys. The finalized index can be stored as the bag of words format, which contains a pair of the dictionary file and the index file.

4.5 GPU Document Inversion

There are two main design issues of GPU computing. One issue is to design the thread-blocks and how to feed the data to the thread-blocks efficiently. Each document can be processed with one thread, one block of threads, or one grid of blocks of threads by the GPU. The thread-block design follows the SPMD programming. The other issue is to use the GPU memory (global memory, constant memory, and shared memory) efficiently.

4.5.1 Thread Design

The sizes of documents, the numbers of terms in them, and the lengths of terms, are all different. The numbers of stopwords in the documents are also different. These variations make it difficult to predict the number of operations and the size of storage, and to divide the data in the SPMD programming design. After reading the documents from external storage, the documents are copied to the GPU global memory, which is large but slow. The shared memory has fast access cycles and all the threads in the same block can access the shared memory. The CUDA architecture gives a programmer the full flexibility to control the use of shared memory. However, the size of shared memory in each streaming multiprocessor (SM) of the CUDA device is 16KB, which is fairly small to store common data structures. Therefore, adequate distribution of data is required to use full benefits from the limited size of shared memory. During the preliminary study, the abstracts/reviews as documents are found have fewer than 4,000 characters, which is less than 4KB using the ASCII code. Thus it is proposed to store one document at a time in the shared memory, and to use one block of threads to process the document. Initially, one block will contain 128 threads, and up to 512 threads can be used. The optimal number of threads in a block will be chosen experimentally. There will be 65,535 thread blocks in a 1-dimensional grid. That is, the GPU may process 65,355 documents with one launch, and each document will be stored in the shared memory to be processed by one block of threads. For the thread design and memory usage, see Figure 2.

4.5.2 Tokenization

The document is copied from the global memory to the shared memory as an array of characters. As shown in Figure 3, we use 128 threads in a block. These threads will examine the first 128 characters in the array, and mark the locations of token breaking characters (blank spaces, or non-alphabet symbols). The beginning
and the length of a token are stored. Then the next 128 characters are examined in the same fashion until the end of the document. After finding the locations of the tokens, the information is stored in the shared memory as an array, the token array.

4.5.3 Stopword Removal
To remove stopwords, each token in the token array must be compared to the list of 641 stopwords. To reduce the computation time, the 641 stopwords are stored in a hash table, and a minimal perfect hash function [14] is used. That is, these 641 stopwords are stored in an array of 641 entries without collision. For every token in the token array, the minimal perfect hash function \( h() \) is used to compute its hash value, and the token is compared to the corresponding entry in the hash table. Thus these stopwords can be removed from the tokens. The GPU device has 64KB of constant memory, which is cached and is much faster than the global memory. The stopword hash table will be constructed by the CPU and copied to the GPU constant memory, which cannot be modified by the GPU threads. The tokens will be hashed by the minimal perfect hash function. If the hashed value of the token is between 0 and 640, the token is matched to the corresponding stopword in the hash table. Then the token is the stopword, it is removed from the token array.

4.5.4 Stemming
The Porter stemming algorithm will also be implemented in CUDA. The original implementation of the algorithm had highly branched code. Most of the branches in the algorithm are checking the presence of a suffix or prefix for removal. However, the CUDA programming platform does not support branch prediction, and any branching will lead to divergent execution, which is serialized on the GPU device. Thus, the implementation of the Porter stemmer will use a finite state automata [11]. The state transition table is used for lookup for each state and each input character. During the stemming, a state of each token is determined by the location of the suffix or prefix. The number of stemming states is small enough to be stored in the GPU constant memory. Then a token is stemmed in place; that is, the location and the length are modified to reflect the stemming. The code does not use additional storage because it overwrites its input with output in the shared memory directly (Figure 4). Using the constant memory, the state transition table, and overwriting, the Porter stemming algorithm can be performed on the GPU.

4.5.5 Generating the Index of One Document
As shown in Figure 5, the document inversion follows the BHBI algorithm. First, the hash value of a token will be computed. The choice of a hash function is important. The clock cycles needed for operations in the GPU are different from those for the CPU architecture. For example, the integer multiplication and division take 16 cycles in the GPU. It is four times longer than the floating...
point multiplication, which takes only four cycles. The operations of addition and subtraction take four cycles in both cases. Therefore, elimination of integer multiplication/division or conversion from integer to floating point calculation are recommended to reduce the computing time. The SDBM hash function, which uses addition, subtraction, and bit shift, is used in the CUDA code. SDBM hash function is an extended version of NDBM, a part of the Berkely DB Library [18]. The hash values of terms are stored in the shared memory of a block of the threads.

Figure 5. Every token will be hashed as its Term Hash

4.5.6 Merging the Indices of All Documents

After a thread block has finished indexing a document, the indices in the index block are sorted by the hash values of terms using a parallel bubble sort method. Generally, the index block from a document is fewer than the number of the original tokens. Thus, if the number of block indices is fewer than the number of threads, it will be sorted with the threads. After sorting, the index blocks are transferred into GPU global memory. Then, the next documents are indexed in the same fashion until the end of the document in the current work block. When all of the available memory is full, the index blocks in the memory need to be merged together. In a global merge step, American flag sort [10] is used to finalize the index in the work blocks. American flag sort consists of three steps. In the first step, the number of indices in each bucket is counted. In the second step, the starting position in the bucket array is computed. Lastly, indices are cyclically permuted to their proper bucket. There is no collection stage because the buckets are already stored in order. If the work block is sorted, the system copies the sorted block to the CPU and makes an index file. Then the GPU will process the next work block in the same fashion until the end of the work blocks. The finalized index is generated by merging and sorting the work block files in the CPU.

5. RESULT AND DISCUSSION

5.1 Dataset

Abstracts of PubMed and e-commerce product reviews were used for this research. The machine learning repository at University of California at Irvine (UCI) provides PubMed abstracts of 8,200,000 articles published before 2008 in the bag of word format as well as texts [6]. J. McAuley, et al. at University of California at San Diego (UCSD) used Amazon product data with 142.8 million reviews during May 1996 - July 2014 [9]. They released smaller dataset of certain categories as JSON data type. For the dataset, abstracts of PubMed articles and product reviews of e-commerce on-line stores were directly collected from each web site. A total of 143,201 abstracts of PubMed articles and a total of 229,115 reviews in digital camera category were downloaded. Most abstracts and product reviews had less than 3,000 words. Therefore, most data files are less than 4KB in size.

5.1.1 Prediction of Term Size

For the experiment, preprocessing was performed with 8 sets of different numbers of abstracts from the collected abstract data. The numbers of abstracts in the sets are 1,000, 2,000, 4,000, 8,000, 16,000, 32,000, 64,000, and 128,000. After preprocessing, the results show the number of terms used in each set. 10,343 terms are used in 1,000 abstracts, and 146,062 terms are used in 128,000 abstracts. When the number of abstracts is doubled, the number of terms is increased by an average 1.46 times.

Figure 6. Estimated number of unique terms

In order to reduce the collision in hash values of the terms, the width of hash values should be wide enough. Thus, the expected number of unique terms from a certain number of documents is needed to choose the width of hash values. The expected number of unique terms can be estimated by Heaps’ law, which describes the portion of a vocabulary which is represented by certain documents consisting of terms chosen from the vocabulary [2]. Let M be the expected number of unique terms, and let n be the number of documents. K and β are parameters to be determined by empirical data. Heaps’ law states

\[ M = Kn^\beta \]

For English text, the range of K is between 10 and 100, and β is between 0.4 and 0.6. For PubMed abstracts, the parameters K = 19 and β = 0.55 are the best-fit for the preliminary data. Calculating with Heaps’ law, if PubMed contains 16 million abstracts, the expected number of unique terms is around 2.2 million. Figure 6 shows that the preliminary data follows Heap’s law.

A width W (the number of bits) of the hash values must be chosen so that the expected number of collision among M unique terms is less than one. Under this constraint, W and M are related by the following inequality:

\[ M < \frac{1 + \sqrt{1 + 8 \cdot (2^W - 1)}}{2} \approx \frac{1}{2} + 2^W \cdot \frac{1}{2} \]
A 26-bit hash function is expected to produce no collision for up to 11,585 terms. The abstracts from 1,000 articles have 10,343 unique terms. Thus, a 26-bit hash function is wide enough for 1,000 abstracts. A 41-bit hash function can safely hold up to 2.09 million terms, which can cover the 2.07 million terms from the estimated number of terms in 16 million PubMed abstracts. Currently, the PubMed has over 20 million articles and around 26% of the articles do not have abstracts. Therefore, a 41-bit hash function is enough for all PubMed abstracts without producing a collision. Since the average length of product reviews is shorter than one of abstracts, product reviews have a smaller number of terms. 8,931 terms are used in 1,000 product reviews and 81,590 terms are used in 128,000 product reviews. Thus, a hash width of abstracts is enough to hold the terms, used in the same number of product reviews, without any collision.

5.2 Discussion of GPU Document Inversion
The CPU used for the experiments was a 3.4 GHz AMD Phenom II X4 965 processor with 4 cores, 8 GB total memory and 512 KB cache. The GPU used for the implementation was NVIDIA Tesla C2050 device with 14 multiprocessors and 32 cores per multiprocessor [19]. It has a GPU clock speed of 1.15 GHz and 3 GB GDDR5 global memory. The CUDA Toolkit 5.0 based on CentOS 6.5 was used.

The performance of document inversion using GPU is affected by the number of thread and the data transfer time between the host (CPU) and the device (GPU). For the comparison of speedup, 4 sets the number of thread and the data transfer time between the host (CPU) and the device (GPU). For the comparison of speedup, 4 sets of abstracts, 16,000, 32,000, 64,000, and 128,000 are selected from PubMed abstract collection and 4 sets of reviews, 50,000, 100,000, 150,000 and 200,000 are selected from product review collection. Each document is less than 4KB size and at least 800MB in global memory is required to hold up to 200,000 documents.

Table 2 shows the variation in total run time for CPU and GPU for each set of abstract collection from PubMed and product review. Using 128 threads per one block design, the GPU performed 1.97-2.97 times as fast as CPU on average. Speed was dependent on the number of documents and on the number of threads. The data transfer from host to device and from device to host took a significant amount of time. Thus, the small number of abstracts showed less speedup than large size of data.

Table 2. Document inversion of CPU vs GPU

<table>
<thead>
<tr>
<th>Document</th>
<th>CPU (sec)</th>
<th>GPU (sec)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>PubMed Abstract</td>
<td>16000</td>
<td>23.59</td>
<td>11.98</td>
</tr>
<tr>
<td></td>
<td>32000</td>
<td>45.98</td>
<td>15.44</td>
</tr>
<tr>
<td></td>
<td>64000</td>
<td>89.21</td>
<td>31.18</td>
</tr>
<tr>
<td></td>
<td>128000</td>
<td>178.33</td>
<td>62.43</td>
</tr>
<tr>
<td>Product Review</td>
<td>50000</td>
<td>63.41</td>
<td>22.48</td>
</tr>
<tr>
<td></td>
<td>100000</td>
<td>112.67</td>
<td>40.38</td>
</tr>
<tr>
<td></td>
<td>150000</td>
<td>183.85</td>
<td>64.96</td>
</tr>
<tr>
<td></td>
<td>200000</td>
<td>242.22</td>
<td>84.99</td>
</tr>
</tbody>
</table>

6. Conclusions
With the advent of CUDA and GPU, several attempts have been made to parallelize the existing algorithms as well as to develop other new algorithms that work best with CUDA architecture. In this work, we implement the parallel document inversion on high throughput document data using massively parallel computational device. We conducted preliminary experiments and found that the parallel document inversion on the GPU is 1.97 times to 2.97 times faster than the same method on the CPU. The performance of implementation is limited by the amount of global memory and shared memory available on the GPU, as it requires storing all the documents in the global memory. It is likely that even better performance may be achieved by employing a more sophisticated arrangement of threads, blocks, and the grid. With newer generation GPU devices and the latest version of CUDA toolkit, it could lead to improvements in speed. These findings indicate that this approach has potential benefits for large-scale document collection, and could be easily applied to other similar problems.

7. ACKNOWLEDGMENTS
We would like to thank Julia Chariker for insightful discussions and comments. Funding was provided by a grant from the National Institutes of Health (NIH), National Institute for General Medical Science (NIGMS) grant P20GM103436 (Nigel Cooper, PI). The contents of this manuscript are solely the responsibility of the authors and do not represent the official views of NIH and NIGMS.

8. REFERENCES

REFERENCES


ABOUT THE AUTHORS:

Mr. Sungbo Jung received his Bachelor's degree from the school of Journalism and Mass Communication at the Korea University, Seoul, South Korea and Master's degree from Computer Engineering and Computer Science at University of Louisville, Louisville, KY, USA. Currently, he is a Ph.D. candidate in Computer Engineering and Computer Science at University of Louisville. He is broadly interested in the research area of High Performance Computing(HPC), Bioinformatics, and Information Retrieval. Contact him at sungbo.jung@louisville.edu.

Dr. Dar-jen Chang is an Associate Professor of the Department of Computer Engineering and Computer Science (CECS) at the University of Louisville. He received an M.S. in Computer Engineering and a Ph.D. in Mathematics from the University of Michigan, Ann Arbor. His current research interests include computer graphics, 3D modelling, computer games, GPU computing, and compiler design with LLVM. He can be contacted at djchan01@louisville.edu.

Dr. Juw Won Park is an Assistant Professor at the Department of Computer Engineering and Computer Science, University of Louisville. His research interests include the analysis of alternative mRNA splicing and its regulation in eukaryotic cells using high-throughput sequencing along with related genomic technologies. He has a Ph.D. in computer science from the University of Iowa. He can be contacted at juw.park@louisville.edu.
An Extracting Method of Movie Genre Similarity using Aspect-Based Approach in Social Media

Youngsub Han  
Department of Computer and Information Sciences  
Towson University  
7800 York Road, Towson, MD, USA  
yhan3@students.towson.edu

Yanggon Kim  
Department of Computer and Information Sciences  
Towson University  
7800 York Road, Towson, MD, USA  
ykim@towson.edu

ABSTRACT
In the movie industry, a movie recommendation is a manner of advertisement or promotion. To recommend movies, we used movie reviews and YouTube comments because user-generated data containing people’s opinions. To analyze the data, we used MSP model, which is one of aspect-based approaches and it guarantees relatively higher accuracy than existing approaches. To discover a genre similarity, we proposed two methods, which are “TDF-IDF” and “Genre Score”. The “TDF-IDF” is designed to extract genre specific keywords and the “Genre Score” indicates a degree of correlation between a movie and genres. Then, the system recommends movies based on results of K-means and K-NN.

CCS Concepts
• Information systems → Data mining  
• Information systems → Recommender systems  
• Information systems → Information extraction

Keywords
Data Mining; Aspect Based Analysis; Clustering; Movie Recommendation; Social Media

1. INTRODUCTION
In 2015, filmmakers launched 2,087 movies around the world. This number is a 26% growth from 2014 according to the theatrical market statistics in 2015 released by MPAA (Motion Picture Association of America). Moreover, online video streaming is quickly growing through streaming services such as Netflix, Hulu Plus, and Amazon. Accordingly, many studies have been conducted to recommend movies because a movie recommendation is a manner of advertisement or promotion that target customers in the movie industry [1, 2].

Recommendation system is one of the information filtering techniques to indicate information items such as movies, music, web sites, news that are likely of interest to the user. Various companies such as Netflix, Google and Amazon uses recommendation system for the success of their business. To recommend movies, various information can be used as features such as movie reviews, comments, view histories, ratings, actors, producers, writers, running time, box office, and genres [2, 3, 4, 27]. In this research, we decide to use movie reviews and YouTube comments to discover characteristics of movies in terms of people’s perspectives because user-generated data contains people’s opinions including unpleasant or dissatisfied experiences [5]. To analyze the data, we used text-based approaches, which are commonly used on video classification [20]. The advantage of text-based approach is utilization of a large body of research conducted on text classification, which helps to understand user’s perspectives [20, 21].

Orellana-Rodriguez et al. proposed a text-based approach that automatically extracts affective context from user comments associated with short films available on YouTube to explore the role of an emotional state as an alternative to explicit human annotations [3]. All extracted emotional keywords would be categorized into four representative categories: joy-sadness, anger-fear, trust-disgust, and anticipation-surprise [3]. The results showed how the affective context can be influenced for emotion-aware film recommendation [3]. Diao et al. proposed a movie recommendation model using an aspect-based approach from online movie reviews and ratings [4]. The aspect-based approach provides more in-depth analysis than the traditional lexicon-based approach [6, 7, 8]. Thus, we use this approach to analyze movie reviews and YouTube comments.

In this paper, we proposed a system including two main methods, which are “TDF-IDF” and “Genre Score” for movie recommendation. The TDF-IDF is designed to finds genre representative keywords and the Genre Score is designed to discover a similarity of movies with consideration for genres. We also used K-means and K-NN to find similar movies based on results of the methods.

2. RELATED WORKS
2.1 Aspect-based Analysis
The aspect based sentiment analysis is the lexicon-based approach because the lexicon is used as a measurement. The aspect-based approach has a strength, which is a more in-depth sentiment analysis by categorizing all results into each aspect. In this sense, each aspect is paired with each sentimental expression.

For example, if an object is a mobile phone, its aspects are “display,” “size,” “price,” “camera,” or “battery.” In this case, aspects seem the attributes of the objects to describe more detail. Thus, expected results are “display-clean,” “price-good,” or “camera-awesome” [6, 7, 8]. Therefore, we decided to apply this approach for more in-depth analysis.
2.2 Morphological Sentence Pattern Model

In this paper, we used MSP (Morphological Sentence Pattern) model, which was proposed in our previous research [9]. This model was developed for building aspect-based sentimental lexicon for sentiment analysis. In this model, the recognizer extracts which Part of Speeches (POS) are surrounding aspects or expressions. We used the “Stanford Core NLP” made by the Stanford Natural Language Processing Group to recognize patterns and extract lexicon. This tool provides refined and sophisticated results from textual data based on English grammar such as the base forms of words, the parts of speech (POS), and the structure of sentences [10]. For a diversity of extraction, this method considered the N-gram model for matching patterns. To analyze social media data including YouTube and Twitter, the system built morphological sentence patterns for each media separately because people share their opinions and emotions differently depending on the sources [23]. This approach showed relatively higher F-score (79.64) than existing approaches without any of the human coding process. Therefore, we decided to use this model to extract aspects and expressions from movie reviews and YouTube comments.

2.3 TF-IDF

The TF-IDF (Term Frequency-Inverse Document Frequency) was designed for calculating the importance of keywords in corpus. In this method, TF (Term Frequency) is a measurement of how frequently a keyword occurs in a document and IDF (Inverse Document Frequency) means how each keyword is less-commonly used in the corpus. IDF is the number of documents divided by the document’s frequency of a keyword in all documents. Therefore, the TF-IDF is TF multiplied by the IDF and it indicates a degree of a document specific keywords in corpus. In this research, we used a normalized TF-IDF using the logarithmically scaled frequency to prevent a bias towards longer documents [11, 12]. The base form of the equation is:

$$tfidf_{t,d} = (1 + \log(tfr_{t,d})) \times (\log \frac{N}{df_{t}})$$  \hspace{1cm} (1)

To extract genre specific aspects and expressions, we modified the equation (1). We describe the method in detail in section 3.2.

2.4 K-Means Clustering

Clustering is an unsupervised learning algorithm for dividing a set of objects into smaller sets [24]. K-Means clustering algorithm was originally proposed in 1965 by Forgy [13] and in 1967 by MacQueen [14]. It is still one of the most popular clustering algorithms in various fields such as data mining, artificial intelligence and computer graphics. This algorithm aims to partition n observations into k clusters. Each observation belongs to the cluster with the nearest mean in the cluster [15]. Two main features of K-Means are the Euclidean distance as a metric for measuring the distances between the points, and the number of clusters k, which is given as an input parameter to the algorithm [16]. In this research, we used this method to find groups of movies based on their similarity to recommend movies.

To determine the number of clusters (k), we used the Elbow method [16]. This is a method of interpretation and validation of consistency within cluster analysis designed to find the appropriate number of clusters (k). In the Elbow method, the percentage of variance is explained as a function of the number of clusters: One should choose a number of clusters so that adding another cluster does not give much better modeling of the data. More precisely, if one plots the percentage of variance explained by the clusters against the number of clusters, the first clusters will add much information such as variances, but at some point the marginal gain will drop, giving an angle in the graph. The number of clusters is chosen at this point [16].

2.5 K-Nearest Neighbor (KNN)

Collaborative filtering is a filtering process for information or patterns on collaboration of objects such as users or data sources. The k-Nearest Neighbors algorithm (KNN) is used for the collaborative filtering-based recommender system [27].

The KNN finds similar objects such as movies by a majority vote of its neighbors in order to determine their similarity. If k = 1, then the object is simply assigned to the class of that single nearest neighbor [25, 26]. In this research, we also used this method to find similar movies based on their similarity to recommend movies.

3. SYSTEM AND IMPLEMENTATION

Figure 1 shows the system architecture and flow. The system consists of four main phases; collecting data, extracting aspects and expressions for building trainset, calculating genre scores, and clustering for recommendation. At the first phase, the crawler collects movie reviews and YouTube comments using keywords as seeds such as movie names. Then, the system extracts aspects and expressions using MSP model [9]. Then, the system calculates an importance of aspects and expressions based on TDF-IDF, which is developed based on TF-IDF [11, 12]. These aspects and expressions are used as genre representative keywords to calculate a genre similarity. The system then calculates the genre score, which is developed for finding correlations between genres and movies. The score indicates a degree of relations between four representative genres, which are “action,” “animation,” “comedy,” and “horror”. Then, this system recommends movies based on K-means and KNN results using R Studio [17, 18, 19].

3.1 Data Collecting

To collect movie reviews from IMDb, Rotten Tomatoes, and Metacritic, we used a crawler developed in our previous research [9]. The crawler uses the jsoup HTML parser, which is an open-source Java library of methods designed and developed to extract information stored in HTML based documents by Jonathan.
To collect YouTube comments, we used a YouTube collecting tool, which was developed by Lee et al [22]. YouTube provides APIs to collect data such as video information, user profiles, and comments written by users. The crawler collects comments posted on movies by keywords that are related to the target objects such as companies, products, politicians or movies chosen as seeds. It also collects the data repeatedly within scheduled time based on user requests. We selected official movie trailers to collect comments for experiments.

3.2 Extracting Aspects and Expressions

As we mentioned in Section 2.2, we used the MSP model to extract keywords [9, 23]. In this model, there is a preprocessing module to extract aspects and expressions accurately from movie reviews and YouTube comments [23]. The module filters out a sentence not including either a noun, an adjective, or a verb because the sentence is considered less meaningful in the linguistic approach [23]. This module also transforms and filters out reserved words such as hash tags (#), accounts (@) or URL formats (http://) by service providers because these words cause errors when analyzing the data.

Once preprocessing is complete, the system extracts aspects and expressions using morphological sentence patterns. Through the model, the system generated 4,190 patterns for extracting aspects and 1,325 patterns for extracting expressions from movie reviews, and 1,099 patterns for extracting aspects and 595 patterns for extracting expressions from YouTube comments. The system calculates an importance of aspects and expressions based on TDF-IDF (Term Frequency Inversed Document Frequency).

3.3 Discovering Genre Similarity

After extracting aspects and expressions from movie reviews and YouTube comments using the MSP model, the system retrieves genre representative aspects and expressions based on the TDF-IDF (see section 3.2.1). This method helps to filter out commonly used aspects and expressions, which occur frequently in every genres. To calculate genre similarity, the system selects the top 100 aspects and expressions as features. Then, the system calculates genre scores (see section 3.2.2) to find a correlation between relevant genres and irrelevant genres.

Figure 2 shows our factorized method in terms of datasets and flows. A given \( a \) is extracted aspects and expressions from relevant movie reviews \( r \), irrelevant movie reviews \( i \) and target movie reviews \( t \). Therefore, \( a_r \) is a set of aspects from relevant movies and \( a_t \) is a set of aspects from irrelevant movies and \( a_i \) is a set of aspects from target movies. \( s_r \) is a set of intersection between \( a_r \) and \( a_t \). We used this set as stopwords to filter out commonly used aspects. \( t_r \) and \( t_i \) are a set of \( a_r - a_t \) and \( a_t - a_r \). These two sets are representatives of relevant movies and irrelevant movies.

Finally, the system extracts \( W_{tr} \), which is a set of intersection between \( t_r \) and \( a_t \), and \( W_{ti} \), which is a set of intersection between \( t_i \) and \( a_t \). \( W_{tr} \) is a set of representative aspects from relevant movies and \( W_{ti} \) is a set of representative aspects from irrelevant movies compared with target movies’ representative aspects. The numbers of these two sets of aspects would be used to calculate the genre score.

\[
\frac{\text{feature}}{\text{top100}}(\text{tfidf}_f,d) = \text{Top100}(\text{ indef}_f,d) \tag{4}
\]

\[
tdf_{f,d} = (1 + \log(tfd_{f,d})) \times df_{fedt} \tag{2}
\]

\[
tdfidf_{f,d} = tdf_{f,d} \times \log \left( \frac{N}{df_f} \right) \tag{3}
\]

3.3.1 TDF-IDF

At the beginning of this research, we tried to use the TF-IDF to discover genre representative keywords because TF-IDF is broadly used to calculate the importance of a keyword. However, this method was not sufficient to use for this research. For example, when a keyword has a higher number of the TF-IDF score, the aspect is a movie-specific keyword such as a character name, actor or location, rather than a genre-specific keyword.

Therefore, we proposed a method named ‘TDF-IDF’ based on the TF-IDF to solve this problem. In this method, the TDF is the weighted TF with a document frequency of all relevant movies (2). An aspect can be a genre or a group of movies’ representative aspect chosen by the TDF. Then, the TDF is multiplied by IDF as shown in the equation (3). After calculating TDF-IDF, the system selects the Top 100 aspects and expressions as features to calculate the genre score as shown in the equation (4).

3.3.2 Genre Score

The genre score is designed to discover a correlation between a target movie and movie genres using all extracted features based on their TDF-IDF scores. As shown in the equation (5), \( \mathcal{G}_p \) is all the accumulated TDF-IDF scores when a set of target movie’s aspects

---

appears in $T_p$, which is a set of Top 100 aspects are extracted from a relevant genre’s reviews. As shown in the equation (6), $G_q$ is all of the accumulated TDF-IDF scores when a set of target movie’s aspects $t$ appears in $T_q$. This is the top 100 aspects extracted from other genre’s reviews. Therefore, $G_p$ is how often a target movie’s aspect occurs in relevant movie reviews and the $G_q$ is how often a target movie’s aspect occurs in irrelevant movie reviews. Finally, the system calculates a gap of $G_p$ and $G_q$ as a degree of the correlation between the relevant movie and the irrelevant movies (7) as the genre score. The analyzer calculates the genre score by the aspect and expression separately since some movies have more than one genre and could not be categorized into either one of them. In section 4.2, we examined why both aspects and expressions should be used in this research based on an experiment. Finally, the Genre Score$_{all}$ is calculated as shown in equation (8).

\[
G_p = \sum_{t \in T_p} tfidf(t) \quad (5)
\]

\[
G_q = \sum_{t \in T_q} tfidf(t) \quad (6)
\]

\[
\text{Genre Score} = (G_p - G_q) + (G_p + G_q) \quad (7)
\]

\[
\text{Genre Score}_{all} = \text{Genre Score}_{aspect} + \text{Genre Score}_{expression} \quad (8)
\]

4. EXPERIMENT

To verify our method, we selected the Top 100 movies listed on the box office mojo released from January 2015 to May 2016. Then, we divided the 100 movies into 3 groups; 16 movies for train-set, 4 movies for test-set, and the remaining 80 movies for recommendation.

4.1 Train-set

As shown in the Table 1, we further categorize 16 movies into 4 movie genres, which were “action,” “animation,” “comedy,” and “horror” for the train-set. These movies are generally co-labeled with other genres. For example, action movies are labeled with the “adventure,” “fantasy,” or “Sci-fi,” animation movies are labeled with “family,” comedy movies are labeled with “drama,” and horror movies are labeled with “thriller,” or “mystery.” We assumed that these movies were representative movies for each selected genre. Our system calculated the genre scores based on these groups to be used as train-set.

4.1.1 Movie Reviews

We collected 100 movie reviews for each movie from IMDb, Rotten Tomatoes, and Metacritic. Then, the system extracted aspects and expressions using the MSP model. Figure 3 shows examples of extracted aspects and expressions with the TDF-IDF score by the genres from movie reviews. The TDF-IDF score indicates a degree of an importance of a genre. Then, the analyzer calculated the genre score using the top 100 aspects and expressions for each movie.

<table>
<thead>
<tr>
<th>Group</th>
<th>Movies</th>
<th>Labeled Genres</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Star Wars: The Force Awakens</td>
<td>Action, Adventure, Fantasy</td>
<td>1st</td>
</tr>
<tr>
<td></td>
<td>Jurassic World</td>
<td>Action, Adventure, Sci-Fi</td>
<td>2nd</td>
</tr>
<tr>
<td></td>
<td>Avengers: Age of Ultron</td>
<td>Action, Adventure, Sci-Fi</td>
<td>3rd</td>
</tr>
<tr>
<td></td>
<td>Deadpool</td>
<td>Action, Adventure, Comedy</td>
<td>4th</td>
</tr>
<tr>
<td></td>
<td>Inside Out</td>
<td>Animation, Adventure, Comedy</td>
<td>5th</td>
</tr>
<tr>
<td></td>
<td>Minions</td>
<td>Animation, Comedy, Family</td>
<td>8th</td>
</tr>
<tr>
<td></td>
<td>Zootopia</td>
<td>Animation, Action, Adventure</td>
<td>10th</td>
</tr>
<tr>
<td></td>
<td>Home</td>
<td>Animation, Adventure, Comedy</td>
<td>20th</td>
</tr>
<tr>
<td>Comedy</td>
<td>Daddy’s Home</td>
<td>Comedy</td>
<td>27th</td>
</tr>
<tr>
<td></td>
<td>Trainwreck</td>
<td>Comedy, Romance</td>
<td>34th</td>
</tr>
<tr>
<td></td>
<td>Get Hard</td>
<td>Comedy, Crime</td>
<td>38th</td>
</tr>
<tr>
<td></td>
<td>Sisters</td>
<td>Comedy</td>
<td>41st</td>
</tr>
<tr>
<td>Horror</td>
<td>10 Cloverfield Lane</td>
<td>Drama, Horror, Mystery</td>
<td>50th</td>
</tr>
<tr>
<td></td>
<td>Insidious: Chapter 3</td>
<td>Fantasy, Horror, Thriller</td>
<td>71st</td>
</tr>
<tr>
<td></td>
<td>Poltergeist (2015)</td>
<td>Horror, Thriller</td>
<td>72nd</td>
</tr>
<tr>
<td></td>
<td>The Boy</td>
<td>Horror, Mystery, Thriller</td>
<td>89th</td>
</tr>
</tbody>
</table>

2 http://www.boxofficemojo.com, box office mojo

Figure 4 shows the results of calculated genre scores to see how the results are distinct between its own genre and other genres based on the aspect and expression.

TABLE 1. SELECTED MOVIES FOR EXTRACTING GENRE SPECIFIC FEATURES

<table>
<thead>
<tr>
<th>Group</th>
<th>Movies</th>
<th>Labeled Genres</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Star Wars: The Force Awakens</td>
<td>Action, Adventure, Fantasy</td>
<td>1st</td>
</tr>
<tr>
<td></td>
<td>Jurassic World</td>
<td>Action, Adventure, Sci-Fi</td>
<td>2nd</td>
</tr>
<tr>
<td></td>
<td>Avengers: Age of Ultron</td>
<td>Action, Adventure, Sci-Fi</td>
<td>3rd</td>
</tr>
<tr>
<td></td>
<td>Deadpool</td>
<td>Action, Adventure, Comedy</td>
<td>4th</td>
</tr>
<tr>
<td></td>
<td>Inside Out</td>
<td>Animation, Adventure, Comedy</td>
<td>5th</td>
</tr>
<tr>
<td></td>
<td>Minions</td>
<td>Animation, Comedy, Family</td>
<td>8th</td>
</tr>
<tr>
<td></td>
<td>Zootopia</td>
<td>Animation, Action, Adventure</td>
<td>10th</td>
</tr>
<tr>
<td></td>
<td>Home</td>
<td>Animation, Adventure, Comedy</td>
<td>20th</td>
</tr>
<tr>
<td>Comedy</td>
<td>Daddy’s Home</td>
<td>Comedy</td>
<td>27th</td>
</tr>
<tr>
<td></td>
<td>Trainwreck</td>
<td>Comedy, Romance</td>
<td>34th</td>
</tr>
<tr>
<td></td>
<td>Get Hard</td>
<td>Comedy, Crime</td>
<td>38th</td>
</tr>
<tr>
<td></td>
<td>Sisters</td>
<td>Comedy</td>
<td>41st</td>
</tr>
<tr>
<td>Horror</td>
<td>10 Cloverfield Lane</td>
<td>Drama, Horror, Mystery</td>
<td>50th</td>
</tr>
<tr>
<td></td>
<td>Insidious: Chapter 3</td>
<td>Fantasy, Horror, Thriller</td>
<td>71st</td>
</tr>
<tr>
<td></td>
<td>Poltergeist (2015)</td>
<td>Horror, Thriller</td>
<td>72nd</td>
</tr>
<tr>
<td></td>
<td>The Boy</td>
<td>Horror, Mystery, Thriller</td>
<td>89th</td>
</tr>
</tbody>
</table>

Figure 3. Examples of feature words by genres from movie reviews.
on the distances of genre scores. In this figure, the gray line shows
the distance between the results of own-genres and other-genres.
In these results, horror is most distinguishable unlike comedy,
meaning that the features of comedy are more commonly used in
movie reviews.

![Figure 4. Results of the genre scores between own genres and
other genres from Movie reviews](image)

### 4.1.2 YouTube Comments

We collected 2,000 comments from YouTube on official movie
trailers for each movie. Then, the system extracted aspects and
expressions using the MSP model. Figure 5 shows examples of
extracted aspects and expressions with the TDF-IDF score by the
genres from YouTube comments. The TDF-IDF score indicates a
degree of importance of a genre. Then, the analyzer calculated the
genre score for each movie using the top 100 aspects and
expressions.

Also, Figure 6 shows the results of calculated genre scores between
its own genre and other genres as the result of YouTube comments.
In this case, comedy and horror are most clearly distinguishable
unlike animation, meaning that the features of animation are more
commonly used in YouTube comments.

![Figure 5. Examples of feature words from YouTube
Comments](image)

![Figure 6. Results of the genre scores between own genres and
other genres from YouTube comments](image)

### 4.2 Test-set

We selected a movie for each genre, which is “Batman v Superman:
Movie” for animation, and “The Visit” for horror for building test-
set.

### 4.2.1 Movie reviews

Table 2 shows genre scores of the movies compared with the four
genres by the aspect and expression. In this results, both results of
aspect and expression are positive scores in relevant genres because
the movies represent own genres. However, in the case of comedy
(“ted 2”), the result of aspects shows a lower polarity of the comedy
genre compared with the result of expression. This means that
people use more common aspects than expressions to share their
opinions about comedy movies on movie reviews. Accordingly, we
decided to use both genre scores of aspects and expressions.

<table>
<thead>
<tr>
<th>Movie</th>
<th>Genre</th>
<th>Aspect</th>
<th>Expression</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batman v Superman (Action)</td>
<td>Action</td>
<td>35</td>
<td>32</td>
<td>67</td>
</tr>
<tr>
<td></td>
<td>Animation</td>
<td>8</td>
<td>-35</td>
<td>-27</td>
</tr>
<tr>
<td></td>
<td>Comedy</td>
<td>-45</td>
<td>-6</td>
<td>-51</td>
</tr>
<tr>
<td></td>
<td>Horror</td>
<td>-2</td>
<td>-8</td>
<td>-10</td>
</tr>
<tr>
<td>The Peanuts Movie (Animation)</td>
<td>Action</td>
<td>17</td>
<td>4</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>Animation</td>
<td>51</td>
<td>50</td>
<td>101</td>
</tr>
<tr>
<td></td>
<td>Comedy</td>
<td>-8</td>
<td>-2</td>
<td>-10</td>
</tr>
<tr>
<td></td>
<td>Horror</td>
<td>-16</td>
<td>-25</td>
<td>-41</td>
</tr>
<tr>
<td>Ted 2 (Comedy)</td>
<td>Action</td>
<td>-11</td>
<td>10</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>Animation</td>
<td>58</td>
<td>-6</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td>Comedy</td>
<td>15</td>
<td>68</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td>Horror</td>
<td>-30</td>
<td>1</td>
<td>-29</td>
</tr>
<tr>
<td>The Visit (Horror)</td>
<td>Action</td>
<td>-36</td>
<td>19</td>
<td>-17</td>
</tr>
<tr>
<td></td>
<td>Animation</td>
<td>25</td>
<td>-65</td>
<td>-40</td>
</tr>
<tr>
<td></td>
<td>Comedy</td>
<td>-6</td>
<td>-6</td>
<td>-12</td>
</tr>
<tr>
<td></td>
<td>Horror</td>
<td>61</td>
<td>90</td>
<td>151</td>
</tr>
</tbody>
</table>
Figure 7 shows the results of the normalized genre score based on percentages (%). In these results, all movies are well categorized into their genres.

4.2.2 YouTube Comments

Table 3 shows results of the movies compared with the four genres by the aspect and expression as the results of movie reviews. The results are similar to the results of movie reviews because these movies also represent their genres. However, in the case of an animation movie (“The Peanuts Movie”), both the result of aspects and expressions are not distinct from other genres. This means that people use more common aspects and expressions to share their opinions about animation movies on YouTube comments.

Table 3. Results of genre score from YouTube comments

<table>
<thead>
<tr>
<th>Movie</th>
<th>Genre</th>
<th>Aspect</th>
<th>Expression</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batman v Superman (Action)</td>
<td>Action</td>
<td>54</td>
<td>66</td>
<td>120</td>
</tr>
<tr>
<td></td>
<td>Animation</td>
<td>-15</td>
<td>-9</td>
<td>-24</td>
</tr>
<tr>
<td></td>
<td>Comedy</td>
<td>-2</td>
<td>-18</td>
<td>-20</td>
</tr>
<tr>
<td></td>
<td>Horror</td>
<td>18</td>
<td>-3</td>
<td>15</td>
</tr>
<tr>
<td>The Peanuts Movie (Animation)</td>
<td>Action</td>
<td>6</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>Animation</td>
<td>15</td>
<td>11</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>Comedy</td>
<td>-33</td>
<td>25</td>
<td>-8</td>
</tr>
<tr>
<td></td>
<td>Horror</td>
<td>-13</td>
<td>-61</td>
<td>-74</td>
</tr>
<tr>
<td>Ted 2 (Comedy)</td>
<td>Action</td>
<td>-5</td>
<td>31</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>Animation</td>
<td>4</td>
<td>-27</td>
<td>-23</td>
</tr>
<tr>
<td></td>
<td>Comedy</td>
<td>37</td>
<td>42</td>
<td>79</td>
</tr>
<tr>
<td></td>
<td>Horror</td>
<td>9</td>
<td>-25</td>
<td>-16</td>
</tr>
<tr>
<td>The Visit (Horror)</td>
<td>Action</td>
<td>-19</td>
<td>-7</td>
<td>-26</td>
</tr>
<tr>
<td></td>
<td>Animation</td>
<td>16</td>
<td>-15</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Comedy</td>
<td>-28</td>
<td>0</td>
<td>-28</td>
</tr>
<tr>
<td></td>
<td>Horror</td>
<td>56</td>
<td>40</td>
<td>96</td>
</tr>
</tbody>
</table>


Table 4. Movie recommendations by K-means result using movie reviews

<table>
<thead>
<tr>
<th>Source Movie</th>
<th>1st Group, K=11 (Genre)</th>
<th>2nd Group, K=7 (Genre)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batman v Superman: Dawn of Justice (AC, AD, SF, F)</td>
<td>13 Hours: The Secret Soldiers of Benghazi (AC, D, T)</td>
<td>Ant-Man (AC, AD, C, SF, T)</td>
</tr>
<tr>
<td></td>
<td>American Sniper (AC, AD, D)</td>
<td>Furious 7 (AC, AD, CR, T)</td>
</tr>
<tr>
<td></td>
<td>Fantastic Four (AC, AD, SF)</td>
<td>Jupiter Ascending (AC, AD, SF, F)</td>
</tr>
<tr>
<td></td>
<td>London Has Fallen (AC, AD, CR, D)</td>
<td>Pan (AC, AD, FM, F)</td>
</tr>
<tr>
<td></td>
<td>The Divergent Series: Allegiant (AC, AD, SF, R, M)</td>
<td>Terminator Genys (AC, AD, SF)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The Huntsman: Winter's War (AC, AD, D, F)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The Jungle Book (AC, AD, D, FM)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The Martian (AC, AD, SF, D)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tomorrowsland (AC, AD, SF, FM)</td>
</tr>
<tr>
<td>The Peanuts Movie (AN, AD, C, FM)</td>
<td>Alvin and the Chipmunks: The Road Chip (AN, AD, C, FM)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Paddington (AN, C, FM)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The SpongeBob Movie: Sponge Out of Water (AN, AD, C, FM)</td>
<td></td>
</tr>
<tr>
<td>Ted 2 (C)</td>
<td></td>
<td>Max (AD, FM)</td>
</tr>
<tr>
<td></td>
<td>Dirty Grandpa (C)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Paul Blart: Mall Cop 2 (AC, C, CR)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Spy (AC, C, CR)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The Boss (C)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Barbershop: The Next Cut (C)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Brooklyn (D, C, R)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Concuassion (B, D, SP)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Creed (D, SP)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fifty Shades of Grey (D, R)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Focus (C, CR, D)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hot Pursuit (AC, C, CR)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hotel Transylvania 2 (AN, C, FM)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Joy (B, C, D)</td>
</tr>
<tr>
<td>The Visit (H, T)</td>
<td></td>
<td>Magic Mike XXL (C, D, MU)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>My Big Fat Greek Wedding 2 (C, R)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pitch Perfect 2 (C, MU)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ride Along 2 (AC, C)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Spotlight (B, D, HI, T)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The Big Short (B, C, D)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The Intern (C)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The Longest Ride (D, R)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Vacation (AD, C)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Southpaw (AC, D, SP, T)</td>
</tr>
</tbody>
</table>

Figure 8. Results of genre score (%) for test movies using YouTube comments

<table>
<thead>
<tr>
<th>Source Movie</th>
<th>1st Group, K=11 (Genre)</th>
<th>2nd Group, K=7 (Genre)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batman v Superman: Dawn of Justice</td>
<td>• Alvin and the Chipmunks: The Road Chip (AN, AD, C, FM)</td>
<td>• Creed (Drama, Sport)</td>
</tr>
<tr>
<td>(AC, AD, SF, F)</td>
<td>• Fantastic Four (AC, AD, SF)</td>
<td>• The Divergent Series: Insurgent (AC, AD, SF, R, M)</td>
</tr>
<tr>
<td></td>
<td>• Pan (AC, AD, FM, F)</td>
<td>• Mad Max: Fury Road (AC, AD, SF)</td>
</tr>
<tr>
<td></td>
<td>• The Divergent Series: Allegiant (AC, AD, SF, R, M)</td>
<td>• Terminator Genisys (AC, AD, SF)</td>
</tr>
<tr>
<td></td>
<td>• The Hateful Eight (Crime, Drama, Mystery)</td>
<td></td>
</tr>
<tr>
<td>The Peanuts Movie</td>
<td>• Kingsman: The Secret Service (AC, AD, C)</td>
<td>• Max (AD, FM)</td>
</tr>
<tr>
<td>(AN, AD, C, FM)</td>
<td>• The Good Dinosaur (AN, AD, C, FM)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Tomorrowland (AC, AD, SF, FM)</td>
<td></td>
</tr>
<tr>
<td>Ted 2 (C)</td>
<td>• American Sniper (AC, AD, D)</td>
<td>• Ride Along 2 (AC, C)</td>
</tr>
<tr>
<td></td>
<td>• Focus (C, CR, D)</td>
<td>• The Big Short (B, C, D)</td>
</tr>
<tr>
<td></td>
<td>• How to Be Single (C, D)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• McFarland, USA (B, D, SP)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Pitch Perfect 2 (C, MU)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Pixels (AN, AC, C)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• The Wedding Ringer (C)</td>
<td></td>
</tr>
<tr>
<td>The Visit (H, T)</td>
<td>• Krampus (C, F, H)</td>
<td>• Everest (AD, BIO, D)</td>
</tr>
<tr>
<td></td>
<td>• The Boy Next Door (M, T)</td>
<td>• Fifty Shades of Grey (D, RO, T)</td>
</tr>
<tr>
<td></td>
<td>• The Gift (M, T)</td>
<td>• Paul Blart Mall Cop 2 (AC, C, CR)</td>
</tr>
<tr>
<td></td>
<td>• Unfriended (H, M)</td>
<td>• The Perfect Guy (D, T)</td>
</tr>
</tbody>
</table>

Table 5. Recommended Movies by K-means results using YouTube comments

4.3 Recommendation

As described in Section 2, we used both the K-Means and the K-NN algorithms to find similar movies to recommend. Firstly, we use K-Means clustering algorithm to group 80 movies using their distances of genre scores. Then, the system recommends movies based on the results.

4.3.1 K-Means

In this experiment, we selected two k numbers which were k = 9 (69.7% of variance) and k = 11 (82.1% of variance) based on the elbow method for K-Means clustering [16]. The system firstly finds relevant movies based on k = 11 because it seems to have a stronger relation between items in a group than results of k = 9. Using R-Studio, we calculate and visualize the clustering results as shown in Figure 11 and 12. In these figures, we labeled the target movies and their groups. Table 4 and 5 show relevant movies by each target movie based on the clustering results. Figure 9 and Table 4 show the results of movie reviews, and Figure 10 and Table 5 show the results of YouTube comments.

In addition, the genre score of “comedy” in YouTube comments was a greater than the result of movie reviews and the genre score of “animation” in movie reviews is greater than the result of YouTube comments. This difference came from how the people express their opinions depending on the media.

Figure 8 shows the results of the normalized genre score based on percentages (%). In these results, all movies are well categorized into their genres like the results of movie reviews. The system recommends relevant movies based on these results.
In Table 4 and Table 5, we highlighted movies when a movie occurs both results of movie reviews and YouTube comments because we assume that these movies have a stronger relation. Especially, a half of horror movies ("Krampus," “The Boy Next Door," “The Gift,” and “Unfriended”) are overlapped. From these result, we can assumed that the horror-related aspects and expressions are more distinctive or unique than the others.

Through these results, we discover that the movie reviews contains more useful information to recommend movies than YouTube comments because the results of movie reviews seem more relevant compared with their original genres.

5. CONCLUSION

In this study, we propose an extracting method of movie genre similarity for movie recommendation using the morphological sentence pattern model and machine learning algorithms. Our method consists of two main methods, which are “TDF-IDF” and “Genre Score” for recommending movies. The “TDF-IDF” is designed to extract genre specific keywords and the “Genre Score” is designed to indicate a degree of correlation between a movie and genres. To experiment our methods, we selected the Top 100 movies listed on the box office released from January 2015 to May 2016 and we collected and analyzed 100 movie reviews and 2,000 YouTube comments for each movie. Through the experiments, we discovered that aspects were more commonly used over genres than expressions. Therefore, we decided to use both aspects and expressions separately as features to calculate the genre score. In addition, we verified that movie reviews and YouTube comments contain useful information to find characteristics of movie genres.

Thus, we can suggest relevant movies using the K-means clustering and K-NN. For the future work, we will add more features such as a movie’s running time, production company, producer, writer, actor, and music to calculate similarities and recommend movies. We can expect that these features are useful as well, when a movie has not enough reviews and comments.

6. REFERENCES


ABOUT THE AUTHORS:

Youngsub Han received M.S. degree in Computer Science from Towson University, Maryland, US in 2010. He is currently a doctoral candidate in Computer Sciences, Towson University. His current research interests include data mining and Big Data and social media analysis.

Yanggon Kim received his B.S. and M.S. degree from Seoul National University, Seoul, Korea in 1984 and 1986, respectively and Ph.D. degree in Computer Science from Pennsylvania State University, Pennsylvania, 1995. He is currently a professor in the Department of Computer and Information Sciences, Towson University and is a director of MS in CS program. His current research interests include Computer Network, Secure BGP network, distributed computing systems, Big Data and data mining in social networks.