Using the GPGPU for Scaling Up Mining Software Repositories

Rina Nagano†, Hiroki Nakamura†, Yasutaka Kamei†, Bram Adams††, Kenji Hisazumi†, Naoyasu Ubayashi†, Akira Fukuda†
†*Kyushu University, 744 Motooka, Nishi-ku, Fukuoka 819-0395, Japan*
††*MCIS, École Polytechnique de Montréal, Canada*
{*rina, hiroki*}*@f.ait.kyushu-u.ac.jp,* {*kamei, ubayashi, fukuda*}*@ait.kyushu-u.ac.jp,*
bram.adams@polymtl.ca , nel@slrc.kyushu-u.ac.jp

***Abstract*—The Mining Software Repositories (MSR) field integrates and analyzes data stored in repositories such as source control and bug repositories to support practitioners. Given the abundance of repository data, scaling up MSR analyses has become a major challenge. Recently, researchers have experimented with conventional techniques like a supercomputer or cloud computing, but these are either too expensive or too hard to configure. This paper proposes to scale up MSR analysis using "general-purpose computing on graphics processing units" (GPGPU) on off-the-shelf video cards. In a representative MSR case study to measure co-change on version history of the Eclipse project, we find that the GPU approach is up to a factor of 43.9 faster than a CPU-only approach.**

I. Introduction

The Mining Software Repositories (MSR) field analyzes the data in software repositories (e.g., source control repositories and bug repositories) to help practitioners make strategic decisions about their projects [1]. For example, Shihab *et al.* [2] show that recent bug fixes to large files are good indicators of breakage bugs (old bugs that re-surface), hence such files should be tested more carefully.

As pointed out by Hassan [1] and Shang *et al.* [3], one of the current challenges in the MSR field is how to deal with large scale data, since increasingly more kinds of repositories are put to use and each of them continues to grow in size. Mockus [4] shows that the source code history by itself of all open source software systems available online already contains TBs of data. Therefore, for the field to progress, new approaches need to be sought to scale up MSR analyses.

Currently, the two main approaches to scale up MSR analyses use 1) a supercomputer, or 2) a cloud-based environment like Hadoop and Pig [3]. Although such approaches were shown to be successful in some contexts, they also have their drawbacks. Both approaches require changes to the implementation of MSR analyses, and a supercomputer is also expensive. While the cost of installing Hadoop and Pig is less than the supercomputer, they still require significant configuration and tweaking to function optimally [5].

In this paper, we focus on an approach that exploits off-the-shelf graphical processing units (GPU), which sport a massively parallel architecture, in the field of MSR. One of the advantages is that we can easily set up the environment, since we just need to install the GPU and its driver. Although GPUs also require changes to the implementation of analyses, their ease of use already has led to numerous applications in domains like computational biology and cryptography [6], [7]. To our knowledge, there is no work to date on using the GPU for software engineering research, specifically in the MSR field.

This paper provides the following contributions:

- A case study on the Eclipse project that shows that the GPU approach outperforms a conventional approach (i.e., CPU-only) to measure co-change.
- We provide our CUDA source code[1] to help other researchers learn how to make use of GPGPU in the field of MSR.

II. GPGPU for Mining Software Repositories

The essential idea behind GPGPU is to split a task into parallel tasks, each of which operates on a separate array of data. This eliminates costly loops in a script. We show an example of this in the context of measuring co-change, i.e., the number of changes to files that co-changed with a given file [2]. This requires the following two steps:

Step 1 Load and parse the change history (i.e., revision logs) from source control repositories.

Step 2 Measure co-change from the parsed change history.

Since the GPU is only suited for calculation, not for data loading (limited memory), we did not use it in Step 1. The CUDA pseudo-code for Step 2 is shown in Figure 1. The major difference between a GPU-based and CPU-only approach is Line 1, where the script identifies the file that each array works on to split the task into parallel arrays. Each box (i.e., a core) runs and repeats the lines 2-6 in

[1]The CUDA scripts are available at http://www.f.ait.kyushu-u.ac.jp/~rina/mine/gpgpu.zip

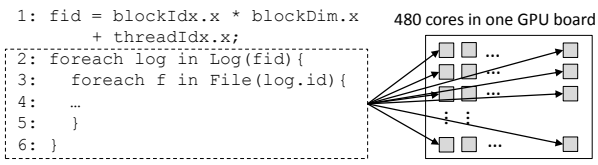

Figure 1. Example CUDA pseudo-code to measure co-change.

978-1-4673-1067-3/12/$31.00 © 2012 IEEE

Table I
EXPERIMENTAL SETUP

	Proposed method	Conventional method
CPU†	Intel(R) Core(TM) i7 CPU 930 (2.80GHZ), 4 cores	4 × Intel(R) Core(TM) i7 CPU 930 (2.80GHZ)
Memory	16 GBs	16 GBs
OS	Ubuntu 11.10	Ubuntu 11.10
Disk type	SCSI 2TB	SCSI 2TB
GPU†	NVIDIA GeForce GTX 480 (1.40GHZ), 480 cores	-

Table II
STATISTICS OF THE USED DATASETS

	Item A	Item B
# of weeks of history	12	24
# of commits	7,000	14,000
# of modified files	6,362	17,517

parallel. Therefore, the GPGPU approach eliminates the loop across all files, reducing the calculation from $O(n^3)$ to $O(n^2)$.

III. CASE STUDY

A. Overview

To explore the potential of GPGPU, we measure the processing time for the co-change example, which is a representative example of an MSR analysis [2], and compare the processing time using GPGPU with the time using CPU-only. In addition, we control our evaluation for the size of the dataset.

Table I shows our experimental setup. The GPGPU setup is identical to the CPU-only setup, except for the use of the GPU. We use one GPU board with 480 cores, which costs less than 500 US dollar as of Feb. 1, 2012. We use NVIDIA's CUDA 4.0 computing engine to implement the scripts on the GPGPU. CUDA scripts are written in C.

B. Studied Projects

We applied the co-change case study on the revision logs of the well-known Eclipse CVS repository provided by the MSR Mining Challenge 2008[2]. To see the effect of scaling up, we used two data sets for our analysis. Table II summarizes the statistics of the used data sets.

C. Results and Discussion

Table III compares the experimental results of the GPGPU and CPU-only approaches. The Step 1 processing time is identical for both, since we could not use GPGPU for loading and parsing the version history. "All" is the sum of the time taken by Step 1 and Step 2.

The results show that GPGPU requires 7.7 (= 61.1/7.9 [sec]) times less time than the CPU-only approach for Item A and 43.9 (= 1177.3/26.7 [sec]) times for Item B. Especially in the larger Item B data set, the speed-up of the GPGPU approach is apparent. We believe that this is a consequence of the $O(n^3)$ complexity of the CPU-only computation (with n the number of modified files).

Table III
EXPERIMENTAL RESULTS FOR GPGPU VS. CPU

Item A	Step 1	Step 2	All [sec]	Improvement
GPGPU	2.690	5.210	7.900	7.7 times
CPU-only	2.690	58.470	61.160	-
Item B	Step 1	Step 2	All [sec]	Improvement
GPGPU	14.980	11.810	26.790	43.9 times
CPU-only	14.980	1162.330	1177.310	-

IV. CONCLUSIONS AND FUTURE WORK

In this paper, we explored the use of the GPGPU to scale up Mining Software Repositories analyses. A case study on the Eclipse Project demonstrated that GPGPUs could be used to effectively scale up co-change analysis. Since the results are promising, we hope that our experiences will help to encourage further research of the use of GPGPU in the MSR field.

We performed only one case study on only one data set (with 2 different sizes). To generalize our results, we need to analyze other datasets and representative case studies. Furthermore, we need to compare the performance of the GPGPUs to the performance of a supercomputer and MapReduce (e.g., Hadoop).

Acknowledgments: This research was conducted as part of the Core Research for Evolutional Science and Technology (CREST) Program, "Software development for post petascale supercomputing — Modularity for supercomputing", 2011 by the Japan Science and Technology Agency.

REFERENCES

[1] A. E. Hassan, "The road ahead for mining software repositories," in *Proc. Frontiers of Software Maintenance (FoSM)*, 2008, pp. 48–57.

[2] E. Shihab, A. Mockus, Y. Kamei, B. Adams, and A. E. Hassan, "High-impact defects: A study of breakage and surprise defects," in *Proc. European Softw. Eng. Conf. and Symp. on the Foundations of Softw. Eng. (ESEC/FSE)*, 2011, pp. 300–310.

[3] W. Shang, B. Adams, and A. E. Hassan, "Using pig as a data preparation language for large-scale mining software repositories studies: An experience report," *Journal of Systems and Software*, 2011(Online).

[4] A. Mockus, "Amassing and indexing a large sample of version control systems: Towards the census of public source code history," in *Proc. Int'l Working Conf. on Mining Software Repositories (MSR)*, 2009, pp. 11–20.

[5] W. Shang, Z. M. Jiang, B. Adams, and A. E. Hassan, "Mapreduce as a general framework to support research in mining software repositories," in *Proc. Int'l Working Conf. on Mining Software Repositories (MSR)*, 2009, pp. 21–30.

[6] S. A. Manavski, "CUDA compatible GPU as an efficient hardware accelerator for AES encryption," in *Proc. Int'l Conf. on Signal Processing and Communications (ICSPC)*, 2007, pp. 24–27.

[7] C. Trapnell and M. C. Schatz, "Optimizing data intensive GPGPU computations for DNA sequence alignment," *Parallel Computing*, vol. 35, no. 8-9, pp. 429–440, 2009.

[2]http://msr.uwaterloo.ca/msr2008/challenge/index.html