ISSN:0975-9646

Parallel OCR Error Correction

G. Narender[#1], Dr.Meda Srinivasa Rao[*2]

[#]*Research Scholar, Associate Professor, Department of CSE, Keshav Memorial Institute of Technology, Hyderabad, India.*

[*]*Professor&Director, School of Information Technology, JNTUH, Hyderabad*

Abstract— **Optical Character Recognition or OCR is the process that recognizes alpha numerical characters on printed pages and converts them to a machine-readable text file. OCR makes it possible to digitize books and other printed materials. Our current research work is investigating ways to speed up implementation of digital libraries. Multicore systems are now becoming common on desktops, servers and even laptops. In order to take full advantage of such multicore systems, current research is looking at ways to make parallel programming main stream. One such effort is the Intel Cilk Plus extensions to C and C++ from Intel Corporation that offer a quick, easy and reliable way to improve the performance of programs on multicore processors. In this paper, we present the results from our work using the Cilk Plus extensions to parallelize OCR error correction.**

Keywords— **OCR, digital Library.**

I. INTRODUCTION

A digital library is a library where collections are stored in digital format as opposed to print, microform, or other media. The collections are accessible via computers [1]. The digital content may be stored locally, or accessed remotely via computer networks. A digital library is a type of information retrieval system. OCR plays an important role in digitizing books and other printed material. The effectiveness of OCR is dependent on the quality of the scanned image. OCR may also not be effective on hand written manuscripts, or when using cursive scripts. Once a OCR processing system digitizes a print document into a set of words, OCR error correction aims to correct "non-word errors". There are several approaches to OCR error correction and a brief survey of these methods can be found in [2]. Most OCR correction methods that involve language models work on a word level. OCR correction methods rely primarily on spell checkers to correct words that do not appear in the given lexicon [3].

[4] describes a statistical approach to OCR error correction by making use of character misrecognition probabilities with the aim of improving the accuracy of OCR error correction. Recent work in this area has tried to make use of Google Web 1T 5-gram data set as a dictionary of words to spell-check OCR text [5]. [5] also proposes parallelizing the algorithm for future work to take advantage of parallel architectures.

Work in the area of OCR error correction has also focused on the evaluation of error correction systems to measure an OCR error correction system performance. [6] focuses on a consistent set of metrics for evaluation of an OCR error correction system.

Word recognition errors can broadly be classified into two categories:

(1) Character mis-recognition errors – as an example the letter 't' the word 'catch' may be recognized as 'k' leading to the word error 'cakch'

(2) Character omission errors – some characters may be omitted at the beginning or the ending of a word. The omission of 'h' in the word 'reach' may be recognized as the word error 'reac'.

OCR error correction aims to correct the list of mis-recognized words by comparing them against known words in the dictionary. In this paper we have focused on correction of word recognition errors and parallelizing the error correction algorithms using Intel Cilk Plus. There has been some work done in the area of parallelizing OCR [7]. As far as we know, ours is the first effort in parallelizing OCR error correction using Cilk Plus.

Section 2 provides a brief introduction to Intel Cilk Plus. Section 3 describes our algorithm for the non-parallel OCR error correction for the two kinds of word errors. Section 4 describes our effort to parallelize these algorithms using Intel Cilk Plus. We give the results of our work in section 5 and conclusions and scope for future work in section 6.We give a sample of the error correction results in Appendix A.

II. INTRODUCTION TO CILK PLUS

Intel Cilk Plus provides extensions to C and C++ languages that offer an easy and reliable way to improve the performance of programs on multicore processors [8]. Three Intel Cilk Plus keywords provide a simple model for parallel programming. Intel Cilk Plus allows us to write parallel programs using a simple model with only three new keywords to learn. This allows C and C++ developers to move quickly into the parallel programming domain. The three new supported keywords are: _Cilk_for,_Cilk_spawn and _Cilk_sync. The header file <cilk/cilk.h> defines macros that provide names with simpler conventions (cilk_for, cilk_spawn and cilk_sync).

A cilk_for loop allows loop iterations to run in parallel and is a replacement for the normal C or C++ for loop. The general cilk_for syntax is:

```
cilk_for (declaration;
          conditional expression;
          increment expression)
```

As an example given the following code, the Cilk runtime system can choose to run different iterations of the loop in parallel.

```
cilk_for (int index = 0; index < 1000; index++)
        func(index);
```

The cilk_spawn keyword is used with a function call and is used to indicate to the Cilk runtime system that the function call can be run in parallel with the caller

As an example, in the following code, the call to function call1() can run in parallel with other code. The spawned function is usually referred to as the child and the caller is referred to as the parent.

```
void caller() {
      cilk_spawn call1();
      other_code;
}
```

The use of a cilk_sync statement in a function indicates that the current function cannot continue in parallel with its spawned children. The function needs to wait until all the spawned children complete execution before it can continue further.

As an example, in the following code, the parent function caller2 can only start executing the function call to call4 only after the spawned call to call2() completes execution.

```
void caller2() {
      cilk_spawn call2();
      call3();
      cilk_sync;
      call4();
}
```

III. OCR ERROR CORRECTION

In order to test the effectiveness of using the Cilk Plus keywords to parallelize OCR error correction, we started out by implementing two non-parallel or serial algorithms for fixing the two types of errors mentioned earlier.

A. Correcting of OCR Character Misrecognition errors

The When OCR misrecognizes a character in a word, the number of characters in the word does not change. We take advantage of this fact by comparing only against known words in our dictionary that have the same length as the misrecognized word. If we assume that n character misrecognitions can occur, the algorithm works as follows:

```
  for (index = 0; index < number_of_misrecognized_words;
index++) {
        err_word = error_word[index];
        word_len = strlen(err_word);

        wordp = word_table_by_length[word_len];
        while (wordp) {
            tword = wordp->word;

            num_mismatch = 0;
            for (tindex = 0; tindex < word_len; tindex++) {
                if (err_word[tindex] != tword[tindex]) {
                    num_mismatch++;
                }
                if (num_mismatch > n) {
                    break;
                }
            }

            if (num_mismatch == n) {
                add tword as a possible correction choice for
err_word;
            }

            wordp = wordp->next;
        }
  }
```

B. Correction of OCR Character Omission Errors

When OCR misses recognizing some characters altogether at either the beginning or at the end of a word, we have character omission errors. If the recognized error word has length len and we assume *n* characters were omitted, we only need to compare against known words in our dictionary that have length len + n. The algorithm works as follows:

```
  for (index = 0; index <
number_of_misrecognized_words; index++) {
        err_word = error_word[index];
        word_len = strlen(err_word);

        wordp = word_table_by_length[word_len + n]
        while (wordp) {
            tword = wordp->word;

            if (strcmp(err_word, tword + n) == 0 ||
                strncmp(err_word, tword, word_len) == 0)
{
                add tword as a possible correction choice
for err_word;
            }

            wordp = wordp->next;
        }
  }
```

IV. PARALLEL OCR ERROR CORRECTION

For our initial work, we chose to parallelize OCR error correction by making use of the cilk_for keyword. The parallel OCR error correction implementation was a matter of simply replacing the for

loop that iterates over all the error words with cilk_for as shown below:

```
cilk_for(index=0;index<number_of_misrecognized_wor
ds; index++) {
        run ocr_character_misrecognition correction
    }


cilk_for(index=0;index<number_of_misrecognized_words;
index++) {
        run ocr_character_omission correction
    }
```

V. Results

In order to measure the speedup from the parallel implementation using cilk_for, we used the Intel C++ v13.0 beta compiler. We started out using a dictionary containing about 58000 words. We then built up a list of about 120000 error words by randomly picking a word in the dictionary and arbitrarily introducing character mis-recognition or omission errors. We then ran the serial and parallel versions of the error correction algorithms on a system with Intel core i5-2450 processor and 6 GB of memory which can run up to four threads in parallel. We ran the program five times and collected the average of the time in seconds it took to do the error correction. Our measurements showed an average speedup of 3 when running the parallel version by simply using the cilk_for keyword in place of the for keyword. This demonstrated the ease and effectiveness of using Cilk Plus in parallelizing applications.

VI. Conclusions And Future Work

Our work has demonstrated the usefulness of Intel Cilk Plus for parallelizing OCR error correction. We were able to demonstrate a significant speedup in the error correction algorithm by simply resorting to the use of the cilk_for keyword. For future work, we plan to investigate using the cilk_spawn/cilk_sync keywords to speedup OCR error correction. As an example, the character misrecognition and omission error correction can be run in parallel using cilk_spawn. We also plan to investigate using Cilk Plus extensions further to speed up other algorithms useful for the implementation of digital libraries.

References

[1] Greenstein, Daniel I., Thorin, Suzanne Elizabeth. The Digital Library: A Biography, published by Digital Library Federation (2002), Pages 1-76.

[2] Steven M. Beitzel, Eric C. Jensen, Davis A. Grossman. Retrieving OCR Text: A Survey of Current Approaches, Symposium on Document Image Understanding Technologies (2003).

[3] Eugene Borovikov, Ilya Zavorin, Mark Turner. A Filter Based Post-OCR Accuracy Boost System. Proceedings of workshop on Hardcopy Document Processing (2004), Pages 23-28.

[4] Xiang Tong, David A. Evans. A Statistical Approach to Automatic OCR Error Correction In Context. Proceedings of the Fourth Workshop on Very Large Corporations (1996), Pages 88-100.

[5] Youssef Bassil, Mohammad Alwani. OCR Context-Sensitive Error Correction Based on Google Web 1T 5-Gram Data Set. American Journal of Scientific Research, ISSN 1450-223X Issue 50 (2012).

[6] Martin Reynaert. All, and only, the errors: more complete and consistent spelling and OCR-error correction evaluation. The International Conference on Language Resources and Evaluation (2008), Pages 1867-1872.

[7] E. Montesinos, J. Kienhofer. Parallelizing OCR. Computers and Communication Conference (1991), Pages 46-52.

[8] http://software.intel.com/en-us/articles/intel-cilk-plus/

Appendix-A

Sample of Single Character-Misrecognition Error Correction

```
Correction choices for: hotpnate
     hotplate
Correction choices for: fawb
     fawn
Correction choices for: valsifying
     falsifying
Correction choices for: mobcters
     mobsters
Correction choices for: addicqiveness
     addictiveness
Correction choices for: clarsics
     classics
Correction choices for: helmkts
     helmets
Correction choices for: globrlar
     Globular
```

Sample of Single Character-Omission Error Correction

```
Correction choices for: anipulable
     manipulable
Correction choices for: oinery
     joinery
Correction choices for: xpressionless
     expressionless
Correction choices for: envoy
     envoys
Correction choices for: entrally
     centrally
     ventrally
Correction choices for: grovelle
     grovelled
     groveller
Correction choices for: lic
     lice
     lick
Correction choices for: eer
     beer
     deer
     eery
     jeer
     leer
     peer
     seer
     veer
```