Abstract

Entity Resolution (ER), the process of finding identical entities across databases, is critical to many information integration applications. As sizes of databases explode, it becomes computationally expensive to recognize identical entities for all records with variations allowed. Profiling results show that approximate matching is the primary bottleneck. Micron’s Automata Processor (AP), an efficient and scalable semiconductor architecture for parallel automata processing, provides a new opportunity for hardware acceleration for ER. We propose an AP-accelerated ER solution, which accelerates the performance bottleneck of fuzzy matching for similar but potentially inexactly-matched names, and use a real-world application to illustrate its effectiveness. Results show 9.5x to 400x speedups for matching one record, with 9.2% more correct pairs and 43% less generalized merge distance (GMD) cost over Apache Lucene. 2.8x to 23.2x speedups are achieved compared with the sorting-based method with 18.1% more correct pairs and 51% less GMD cost.

1 Introduction

Entity Resolution (ER) refers to finding records which describe the same entity within a single database or across different databases [1]. ER is an important kernel of a large number of information integration applications [2]. For example, the Social Networks and Archival Context (SNAC) project collects records from different databases to provide an integrated platform to search historical collections [3].

In such applications, the records of the same person may be stored with slight differences, because documents may come from different sources, with different naming conventions, transcription conventions, etc. SNAC needs to find the records referring to the same entity with different representations, and merge these records. Digital Bibliographic Library Browser (DBLP) encounters similar problems as its database increases.

Determining whether two records represent the same entity is usually computationally expensive. For example, the time complexity of the naive method is $O(N^2)$, where N is the number of records. The process will consume a long time in the context of big data, where one needs to compare a huge number of records. Prior work has proposed different algorithms and computation models to improve the performance [4] [5]. However, the performance is still unsatisfying [6].

Micron’s Automata Processor is an efficient and scalable semiconductor architecture for parallel automata processing [7]. The AP is a hardware implementation of non-deterministic finite automata (NFA) and is capable of matching a large number of complex patterns in parallel. Therefore, we propose a hardware acceleration solution to ER using the AP. This paper focuses on solutions for string-based ER.

To illustrate how the AP can accelerate ER, we present a framework and performance evaluation of several real-world ER applications. The problem is to identify and combine the records with the same identity stored in the SNAC database.

In summary, we make the following contributions.

1. We propose a novel AP-based hardware acceleration framework to solve the Entity Resolution problem.
2. We present an automata design which can process string-based ER, e.g. fuzzy name matching.
3. We compare the proposed methods with state-of-the-art technologies (Apache Lucene and a sorting-based method). On average, it works 9.5x to 400x faster than Lucene for matching one name and works 2.8x to 23.2x faster than sorting-based method. 9.2% and 18.1% more correct pairs are found, and 42% and 51% less GMD cost is needed than Lucene and the sorting-based method respectively.

2 Related Work

Many methods have been proposed to solve ER. One is a domain-independent algorithm for detecting approximately duplicate database records [4]. The paper proposed first computing the minimum edit-distance to recognize pairs of possible duplicate records, and then using union/find algorithm to keep track of duplicate records incrementally. Another method
sorts the records and checks whether the neighboring records are the same [1]. In this paper, we implemented this sorting-based method and evaluated the performance. Apache Lucene is a high-performance state-of-art search engine and it uses a similar method [8]. The difference lies in that Lucene calculates the score of a document based on the query, and sorts documents instead of every individual record. However, we are not aware of any implementations of these algorithms using accelerators. Therefore, we propose an AP-based approach of the Hamming distance based method and evaluate the proposed method.

3 Name Matching using the AP

In this paper, we use a real-world ER problem in SNAC to illustrate how the proposed AP-accelerated method works. When building the SNAC platform, the same person’s name may not always be consistent from one record to the next because of typos, misspellings, different versions of abbreviation, etc. These differences lead to three major problems. 1) One may miss some correct results when querying a record; 2) multiple entries for one entity waste storage space; 3) the duplicated items also slow down the speed of searching. Therefore, SNAC needs to identify and combine potentially similar names and we refer it as the Name Matching problem.

3.1 Workflow

The workflow using the AP is shown in Figure 1. The CPU first reads the file and extracts the names from the original document. Then the AP stores all these names on board. The input file is then streamed into the AP and the AP compares the input name with the names stored on board. If the AP finds a match, it will report back to the CPU; if not, the AP continues to compare the next name. Based on the reporting STE id and the offset of reporting time, the CPU can tell which name has found a match and combine these records. A reconfiguration phase is needed if the number of names exceeds the capacity of the AP hardware.

3.2 Name Formats

One name is usually composed of several sub-names, like family name, middle name and first name. In this paper, we only consider family name and first name, because this is sufficient to evaluate the suitability of the proposed AP-accelerated solution. We choose a subset from the whole database randomly as a basis to extract a representative set of formats for first name and family name respectively. Table 1 shows common variants of family names. Some family names cannot be represented by these formats, e.g. “Colan Lulah Johnston Durr” has more than three parts in it. If this is the case, we treat it as a failure (not match). Refinement of these rare special cases is left for future work. First name formats can be found in [9].

3.3 Automata for Last/First Name

Figure 2 shows the exact-matching automaton design to recognize family names. A fuzzy macro refers to the macro executing a fuzzy match. One fuzzy macro example is shown in Figure 3. All macros in this paper adopt this structure, but with different sequence lengths. Furthermore, the macro structures are not limited to Hamming distance. For example, we also have macro designs for insertions and general edit distance. Details of these designs can be found in [9].

To support different representations of the same name, such fuzzy macro is required. The fuzzy-matching (Figure 4) automaton is similar to the exact match automaton. The major difference is that we use the fuzzy macro in Figure 4 to support fuzzy match. The design for the first name can be found in [9].

We use a hybrid version (Figure 5) of automata for family name and first name in experiments. We match the first name and the family name using one single automaton to conserve STEs.

4 Evaluation

4.1 Experiment Setup

The experiments are executed on a server with 16 AMD Opteron 4386 Cores (3100MHz). We use an AP simulator to derive the execution time for the AP until the real hardware is available. All the data is sampled from the SNAC database.

<table>
<thead>
<tr>
<th>Formats</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abc (basic)</td>
<td>Aachen</td>
</tr>
<tr>
<td>Abc Bcd</td>
<td>Abad Santillan</td>
</tr>
<tr>
<td>Abc Bcd Cde</td>
<td>Abascal Yea Sousa</td>
</tr>
<tr>
<td>Abc II</td>
<td>Abdulhamid II</td>
</tr>
</tbody>
</table>

Table 1: Family name formats.
4.2 Performance Evaluation

4.2.1 Matching Time vs. Lucene

We first compare proposed method with Apache Lucene used by SNAC. Though Lucene is not designed to solve ER specifically; it is widely used in text searching and supports many advanced query types, like proximity queries and range queries, which enables us to solve the ER problem with it [8].

Instead of comparing total running time, we focus on matching time, the kernel we aim to accelerate. Total running time involves some other overhead not related to the matching process, such as index building for Lucene, and compiling for the AP, which are static costs that can be amortized over many executions.

The results are shown in Figure 6. Matching time for Lucene grows roughly linearly as the database size increases. The AP yields 400x speedup on average when the number of names is 14,000. When the number of names exceeds 14,000, we reconfigure the symbols in the STEs and re-stream the input file. The average matching time increases after 14,000 and stays constant until another reconfiguration phase is executed. If the number of names are fewer than 14,000, the speedup increases almost linearly as the number of names increases. After 14,000, the speedup drops from 400x to 200x and continues to increase afterwards until another reconfiguration phase is executed.

We also compare our proposed method with Lucene with its multiple search function. Although searching multiple queries helps to reduce the matching time, the proposed AP-accelerated method still works better. Compared to searching 50 names in parallel, the proposed method achieves 9.5x to 28.2x speedup.

4.2.2 Total running time vs. Sorting-based Method

We also implement a sorting-based method suggested in [1] and work on a much larger dataset from SNAC(from 14,000 to 168,000), so that we can study how multiple reconfiguration phases affect the performance. We collect the total running time in Figure 7. The running time of the AP mainly includes matching time and reconfiguration time. As a whole, we can see even when total number exceeds the capacity of the current hardware and multiple reconfiguration phases are needed, the proposed method still runs faster than the sorting-based method at least 2.8x faster. The worst speedup is achieved when the number is 98,000 when the configuration time is still dominant. After 98,000, the speedup increases as the number of names increases, because the matching time starts to dominate the total running time.

4.3 Accuracy

To evaluate ER results accuracy, we use a subset of names so that we can calculate the correct results manually. We use the manual results as ground truth for checking the accuracy of the AP, Lucene and sorting-based methods. The results are listed in Table 2. For all three different metrics, the AP-accelerated method always works better. This is due to the ability of the AP to better support fuzziness of matching. More details of performance and accuracy results are discussed in [9].

Note that the AP board we use in this paper is the first generation. Technology scaling projections suggest
Figure 6: Performance comparison for Multiple Search Lucene. (X axis represents the database size, ranging from 1,000 names to 20,000 names. Y axis represents the average matching time for one name and speedup respectively. The solid black line represents the speedup against searching 50 simultaneously using Lucene.)

Figure 7: Performance comparison with sorting-based method. (X axis represents the database size, ranging from 14,000 names to 168,000 names. Y axis represents the total running time and speedup respectively.)

<table>
<thead>
<tr>
<th>Compression</th>
<th>Correct Pairs</th>
<th>GMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lucene</td>
<td>65.3%</td>
<td>262</td>
</tr>
<tr>
<td>Sorting</td>
<td>71.4%</td>
<td>233</td>
</tr>
<tr>
<td>AP</td>
<td>57.2%</td>
<td>292</td>
</tr>
<tr>
<td>Manual</td>
<td>47.4%</td>
<td>325</td>
</tr>
</tbody>
</table>

Table 2: Accuracy Results.

that, in the future, we may have a larger capacity or higher frequency, which could lead to even better performance.

5 Conclusions and Future Work

In this paper, we propose using an accelerator-based method to speed up ER using Micron’s AP. The proposed method can make full use of the massive parallelism of the AP and search for up to 14,000 names simultaneously for current hardware. We evaluate both performance and accuracy using real data sets from SNAC. The proposed method achieved both higher performance and better accuracy.

Future work includes using the AP to process larger datasets and rare special cases, solve other string-based ER instances, and compare our proposed method with other existing methods (both software and hardware acceleration).

6 Acknowledgements

This work was supported in part by C-FAR, one of the six SRC STARnet Centers, sponsored by MARCO and DARPA. We would like to thank Micron Technology for their support with the Automata Processor research.

References