A Systolic Array for Rapid String Comparison

Article · January 1985

CITATIONS
113

READS
151

2 authors, including:

Daniel Lopresti
Lehigh University
192 PUBLICATIONS **3,055** CITATIONS

SEE PROFILE

A Systolic Array for Rapid String Comparison

Richard J. Lipton and Daniel Lopresti
Department of Electrical Engineering and Computer Science
Princeton University, New Jersey

ABSTRACT

This paper presents a linear systolic array for quantifying the similarity between two strings over a given alphabet. The architecture is a parallel realization of a standard dynamic programming algorithm. Also introduced is a novel encoding scheme which minimizes the number of bits required to represent a state in the computation, significantly reducing the size of a processor. An nMOS prototype, to be used in searching genetic databases for DNA strands which closely match a target sequence, is being implemented. Preliminary results indicate that it will perform hundreds to thousands of times faster than a minicomputer.

1. Introduction

String comparison is an important operation in many disciplines; a particularly interesting application comes from the field of molecular biology. After isolating and "sequencing" a chain of DNA, which may consist of from tens to thousands of the four bases Adenine, Cytosine, Guanine, and Thymine (abbreviated A, C, G, and T), biologists often want to search a database of known DNA for close matches. In doing so they hope that previous results will help them draw conclusions about their new strand. One such database, maintained by the National Institutes of Health, contains millions of bases, so such searches can take hours of mainframe time. Hence, molecular biologists resort to heuristics; they only search against the portion of the database which they consider relevant and they use sub-optimal algorithms which trim the search. In doing so, they may miss an unexpected, but important, homology. Still, these searches require significant computational resources and time [1], [2], [3].

† This work was supported in part by DARPA Contract N00014-82-K-0549.

Presented in this paper is a linear (i.e., one-dimensional) systolic array for the string comparison problem in general, as illustrated by the DNA comparison problem in particular. It is a parallelization of an optimal dynamic programming algorithm, which promises to run thousands of times faster than the same algorithm run on a serial computer. As is characteristic of systolic arrays (an architecture first described by H.T. Kung and associates at Carnegie-Mellon University [4]) the machine is composed of a large number of simple processors which are highly regular and have only local communication requirements; hence it is ideal for implementation in VLSI.

Perhaps even more significantly, a technique is introduced which greatly simplifies the processing elements, allowing a relatively large number to be fit on a single chip. This observation, which minimizes the amount of data which must flow between adjacent processors, may be applicable to other systolic implementations.

2. Quantifying String Similarity

An intuitive metric for quantifying the similarity of two strings is their "edit" distance [5]. That is, a count of the number of basic editing operations:

i) insert
ii) delete
iii) substitute

needed to transform one string, the "source," into the other, the "target."
For example:

to transform ACG into TGG

ACG -*delete A*→ CG -*delete C*→ G -*insert G*→ GG -*insert T*→ TGG

which is two deletions and two insertions, so the difference between ACG and TGG is four. A substitution has cost two, as it is equivalent to a deletion and an insertion:

ACG -*substitute T for A*→ TCG -*substitute G for C*→ TGG

The edit distance between two strings can be calculated using a dynamic programming algorithm which would, in the above case, build the following table:

		T	G	G
	0	1	2	3
A	1	2	3	4
C	2	3	4	5
G	3	4	3	4

The difference, four, is the entry in the lower right corner of the matrix.

In general, the value d in the 2×2 table fragment:

		t_j
		...
	a	b
s_i	...	c	d

is determined by the rule:

$$d = \min \begin{cases} b + 1 \\ c + 1 \\ \begin{cases} a & \text{if } s_i = t_j \\ a + 2 & \text{if } s_i \neq t_j \end{cases} \end{cases}$$

It is known that dynamic programming algorithms map well onto systolic arrays [6]. In particular, there is a tremendous potential for concurrency in the construction of an edit distance table; the entries on a given 45 degree diagonal can be calculated simultaneously because

they depend only on values up and to the left. This parallelism can be realized with a systolic array. All of the values on 45 degree diagonals will be calculated at the same time and all of the values on -45 degree diagonals will be calculated in the same processor. In this way the quadratic time serial algorithm becomes a linear time parallel algorithm requiring a linear number of processors.

3. The Systolic Architecture

A rhythmic pumping of data through simple processors distinguishes systolic arrays. Figure 1 demonstrates the data flow through one implementation of a linear comparison array using the strings of the earlier example. The source and target are shifted in simultaneously from the left and right respectively. Interleaved with the characters are the data values from the first row and column of the dynamic programming matrix. When two non-null characters enter a processor from opposite directions a comparison is performed (indicated by the darkened circles in the figure). On the next clock tick the characters shift out and the values following them shift in. This processor now determines a new state based on the result of the comparison, the two values just shifted in, and its previous state value, using the same rule as in the dynamic programming algorithm. When the strings are shifted out they carry with them the last row and column of the dynamic programming matrix, and hence the answer.

Strings too long to be compared in one pass through the array can be handled by making multiple passes. In this case, the initializing data values shifted in with the characters reflect a matrix row or column at an intermediate point in the calculation.

4. Minimizing Processor Size

Using the above scheme, comparing two strings of length n in one pass requires approximately $2n$ processing elements. Since DNA sequences of interest can be several thousand bases long it is necessary to fit a large number of processors onto a single chip. The greatest influence on the size of a single processor is the appearance that it must add and compare relatively large data values, on the order of $\log(n)$ bits. Even being optimistic, such adders and comparators require significant silicon area; it is unlikely that a naive implementation would fit more than a couple of processors onto one chip. There is, however, an important property of this algorithm which makes manipulating such large values unnecessary; a *constant* two bits will suffice for *any* length string comparison. The key observation is that

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.