

Parallel Sequence Comparison and Alignment

Richard Hughey
University of California, Santa Cruz

Abstract

Sequence comparison, a vital research tool in computational biology, is based on a simple $O(n^2)$ algorithm that easily maps to a linear array of processors. This paper reviews and compares high-performance sequence analysis on general-purpose supercomputers and single-purpose, reconfigurable, and programmable co-processors. The difficulty of comparing hardware from published performance figures is also noted.

1 Introduction

The vast databases produced by the Human Genome Project demand innovative tools for fast sequence and database analysis. Because sequence databases contain billions of characters, it is important to locate areas of interest within a database or genome quickly.

There are diverse sequence comparison methods, several of which are used by biologists to analyze RNA and DNA, which have 4-character nucleotide “alphabets”, and proteins, which have a 20-character amino acid “alphabet” [7]. The simplest is edit distance, the number of insertions and deletions to transform one sequence into another, which is appropriate for text comparison [16]. Affine gap comparison has similar structure but three times the computation [17], and is a preferred method of biologists as it allows insertion or deletion costs that are linear in the length of the gap plus an added penalty for starting the gap. Biologists also often want to know the alignment of two sequences, an operation that can require storing the dynamic programming matrix and tracing backward the path of minimizations. BLAST, a heuristic with simple dynamic programming, is popular on serial computers because of its speed [1].

High-performance sequence analysis relies on straightforward parallelization of the $O(n^2)$ dynamic programming. A common mapping is to assign one processing element (PE) to each character of the query string, and then to either shift the database through the linear chain of PEs (Figure 1). There are four approaches to high-performance sequence analysis.

Supercomputers (GS) General-purpose supercomputers are, when available, the most flexible means of fast sequence analysis. Unfortunately, supercomputers have high cost.

Single-purpose VLSI (SP) Single-purpose VLSI can achieve the highest performance on one single algorithm for, typically, prices in the low tens of thousands of dollars.

Reconfigurable Hardware (RH) Reconfigurable hardware includes systems based on field-programmable gate arrays (FPGAs) and other architectures such as MGAP. They tend to be slower and have a much lower PE density than SP, but can be faster than supercomputers. Creating and modifying programs for these systems can be tedious, though FPGA and related compiler technology is improving. RH machines generally cost somewhat more than SP machines.

This work was supported in part by funds granted by UCSC and by NSF grant MIP-9423985. The author may be reached at the Computer Engineering Board, University of California, Santa Cruz, 95064, or by email to rph@cse.ucsc.edu. Further information on this project can be obtained on the WWW from <http://www.cse.ucsc.edu/research/compbio/index.html>. The author gratefully acknowledges the contributions of the Kestrel team: Leslie Grate, J Alicia Grice, Kevin Karplus, Jeff Hirschberg, and Don Speck.

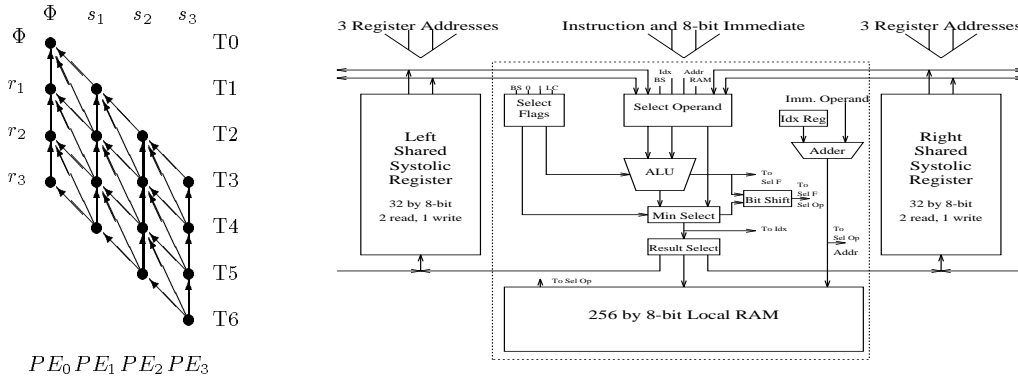


Figure 1. Sequence comparison mapping and Kestrel PE.

Programmable co-processors (PCP) Programmable co-processors strive for the algorithmic flexibility of reconfigurable systems and the speed and density of single-purpose systems. Cost and ease of programming generally fall between the other two classes. PIM falls in this class [6], as does Kestrel. Kestrel is a work in progress that draws from the B-SYS [10] approach. Rather than use special communication registers, a bank of Systolic Shared Registers is used for both communication and computation. Kestrel also incorporates circuitry to greatly reduce the instructions required for sequence comparison. A single cycle instruction can add two numbers, compare the result to a third from memory, select the minimum and store the choice in the bit shifter, and communicate to the next PE by writing the minimum to a shared register. Additionally, unlike other co-processors, it will perform sequence *alignment* as well as sequence comparison [8]. Although pieces of the design are still under consideration, the major components of Kestrel, as well as solid clock rate estimates, are stable, enabling justifiable performance calculations based on an achievable 33 MHz board clock. We will be fabricating prototype chips in early 1996.

2 Performance Comparison

Table 1 summarizes performance of several systems in millions of dynamic programming cell updates per second (MCUPS). For an $m \times n$ sequence comparison or alignment on an array of length l , the runtime in microseconds is approximated by $(1 + k \lfloor \frac{m}{7} \rfloor) ln / \text{MCUPS}$, where $k > 1$ indicates the co-processor's overhead in problem partitioning. For the systems without experimental times (projections are indicated with question marks), the I/O requirements of proposed implementations are included, as well as an MCUPS rating taking into account a typical disk rate of 3 MBytes per second. While high-performance storage systems can certainly sustain 12 MBytes/s, such systems can be expensive.

Figure 2 shows performance for protein sequence comparison and, for those machines able, alignment. The steps in several of the curves occur at multiples of the length of the array. In the Kestrel system, multiple characters are packed in each PE's local RAM with no additional overhead to the cell program length. In the SP machines, the computation must be externally partitioned into array-length segments, with values recycled from one partition to the next [12]. This overhead is estimated at one extra cycle per cell program.

Two observations can be made. First, PE density is critical, and something not achieved in RH systems (even for 4-character edit distance) or BISP. BISP's early lead in Figure 2 is quickly overcome by systems with more PEs. Second, complete system design is also important: fast chips are only as good as their boards.

On cost/performance, the MasPar costs \$2500 per affine gap MCUPS, and the Biocellator \$1000. We expect a 20-chip Kestrel system would cost \$30 000, or \$15 per MCUPS.

System	Ref	Type	Year	PEs/ Chip	PEs	System Performance, MCUPS				I/O MB/s	I/O MCUPS	Price k\$
						4-char edit dist	16-bit Blast	affine gap	Global Align			
Sparc 10/30		WS	1993	1	1	3	4	0.4	0.2		4	8
Maspar MP-2 ^a	[14]	GS	1992	32	4096	240	240?	120	15		240	300
PNAC ^b	[13]	SP	1986	30	9	1	×	×	×		1	1?
BioSCAN ^c	[15]	SP	1993	812	12992	×	25000	×	×		25000	20?
BioSCAN-AT		SP	1993	812	1624	×	3200?	×	×		2 3200	4?
BISP ^d	[4]	SP	1991	16	256	3200?	3200?	3200?	×		12 770	20?
Biocellator-80	[5]	RH	1994	?	?	68	68	68	×		68	49
Splash I ^e	[13]	RH	1992	24	746	370	?	?	?		370	40?
Splash II	[9]	RH	1993	24	64	3000?	?	?	?		12 190	50?
MGAP ^f	[3]	RH	1994	128	64x64	370?	?	?	×		6 190	30?
MGAP-2	[3]	RH	1996?	512	128x128	1480?	?	?	×		12 380	50?
B-SYS	[10]	PC	1991	47	470	18	1?	1?	×		< 1 18	8?
B-SYS* ^g	[10]	PC	—	47	1504	500?	33?	9?	×		< 1 500	30?
Kestrel ^h		PC	1996?	64	1280	3800?	3800?	2000?	550?		3 3800	30?

^aMPsrch_pp and MPsrch_ppa. ^bI/O bound; chips 10 times faster. ^cFull MCUPS only for long queries. ^dMCUPS for 16-chip system described in [4] rather than larger used in [4]’s performance estimate. [4] mentions 3 MB/s VME bandwidth not applied to its performance estimate. ^eConfiguration requires 0.47 s. ^f128x128 (MGAP2) pipeline works on 128 128-long comparisons at once. The I/O-efficient linear mapping would decrease I/O bounds with a small reduction in peak MCUPS. ^g4 MGAP cells per PE. ^h32 B-SYS chips with an on-board controller. ⁱ33 MHz board and 3Mchar/s I/O. Chips are expected to run up to 60–75 MHz for 10 edit distance, and 5 affine gap GCUPS, peak (i.e., meaningless).

Table 1. Sequence comparison hardware (‘×’ indicates ‘cannot perform’).

3 Conclusions

One of the most interesting parts of this study has been the varied and creative means of inflating performance numbers. Following the flavor of the classic “12 ways to fool the masses” [2], here are six of the original (with original numbers but modified to this domain) that came up in this study *several* times. 1. Quote only nucleotide edit distance performance results, not amino acid or affine performance. RH will have factors of 3–5 lower PE density for amino acids, and an additional 20–50 for affine costs. 2. Present performance figures for the inner loop, excluding configuration, data setup, I/O, or board times. 5. Quote performance results projected to a full system. 6. Compare your results against unoptimized code on another machine. 7. Compare with an old code on an obsolete system. 9. Quote performance in throughput rather than operations.

There are three additions to this list. A. Truncate other machines’ results down to a smaller size but not your own. B. Make outrageous scalability claims, such as “over 262,144 BISP chips can be connected together ...,” C. Be unclear about the exact status and size of the hardware. Note that this analysis compared future hardware to existing hardware, the power of which can be expected to grow as, for example, faster and larger FPGAs come on the market. The combination of programmability and SCP-speed, however, means that Kestrel will be able to keep pace with technology growth.

All architectures have strengths and weaknesses for sequence comparison and alignment and other applications. Supercomputers, where affordable, provide good performance coupled with applicability to a wealth of problems. For less expensive and faster solutions, suitable for repetitious use as network servers or HMM [11] trainers, specialized co-processors are an excellent solution. The reconfigurable, bit-oriented mesh designs are appropriate for sequence analysis, but are perhaps better suited to image processing and other applica-

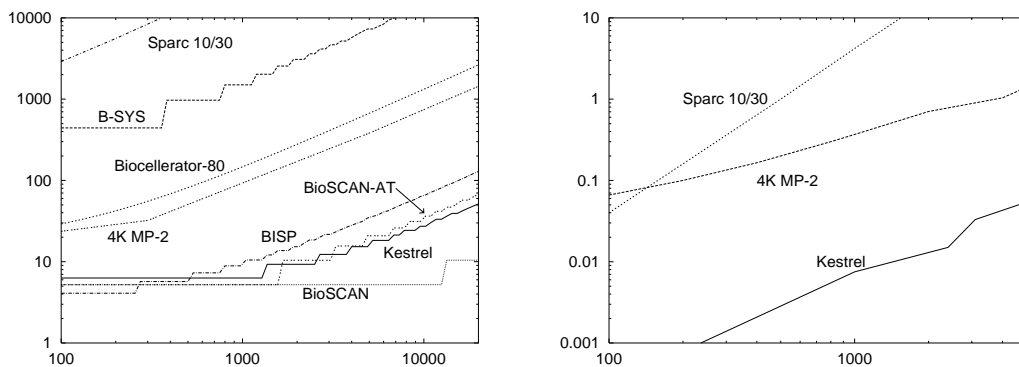


Figure 2. Seconds vs. length for search and alignment of proteins (affine or BLAST).

tions. Single-purpose hardware can be incredibly fast but is restricted to a single algorithm. Programmable co-processors attempt to mix the speed of single-purpose hardware and the programmability of more general approaches.

Because biologists' favored comparison and alignment algorithms are not fixed, programmable parallel solutions are required to speed these tasks. As an alternative to single-purpose systems, hard-to-program reconfigurable systems, and expensive general-purpose supercomputers, we advocate the use of specialized yet programmable hardware whose development is tuned to system speed.

References

- [1] S. F. Altschul *et al.*, "Basic local alignment search tool," *JMB*, vol. 215, pp. 403–410, 1990.
- [2] D. H. Bailey, "Twelve ways to fool the masses when giving performance results on parallel computers," Tech. Rep. RNR-91-020, NASA Ames Research Center, Moffett Field, CA, June 1991.
- [3] M. Borah, R. S. Bajwa, S. Hannenhalli, and M. J. Irwin, "A SIMD solution to the sequence comparison problem on the MGAP," in *ASAP*, IEEE CS, Aug. 1994.
- [4] E. Chow, T. Hunkapiller, J. Peterson, and M. S. Waterman, "Biological information signal processor," in *ASAP* (M. Valero *et al.*, eds.), pp. 144–160, IEEE CS, Sept. 1991.
- [5] Compugen Ltd., "Biocellerator information package." Obtained from compugen@datasrv.co.il, 1994.
- [6] M. Gokhale *et al.*, "Processing in memory: The Terasys massively parallel PIM array," *Computer*, vol. 28, pp. 23–31, Apr. 1995.
- [7] M. Gribskov and J. Devereux, eds., *Sequence Analysis Primer*. New York: Stockton Press, 1991.
- [8] J. A. Grice, R. Hughey, and D. Speck, "Parallel sequence alignment in limited space," in *ISMB* (R. Altman *et al.*, eds.), AAAI/MIT Press, 1995.
- [9] D. T. Hoang, "Searching genetic databases on Splash 2," in *Proc. IEEE Workshop on FPGAs for Custom Computing Machines* (D. A. Buell and K. L. Pocek, eds.), pp. 185–191, IEEE CS, Apr. 1993.
- [10] R. Hughey and D. P. Lopresti, "B-SYS: A 470-processor programmable systolic array," in *ICPP* (C. Wu, ed.), vol. 1, pp. 580–583, CRC Press, Aug. 1991.
- [11] A. Krogh, M. Brown, I. S. Mian, K. Sjölander, and D. Haussler, "Hidden Markov models in computational biology: Applications to protein modeling," *JMB*, vol. 235, pp. 1501–1531, Feb. 1994.
- [12] R. Lipton and D. Lopresti, "Comparing long strings on a short systolic array," in *Systolic Arrays* (W. Moore, A. McCabe, and R. Urquhart, eds.), pp. 363–376, Boston, MA: Adam Hilger, 1987.
- [13] D. P. Lopresti, "P-NAC: A Systolic Array for Comparing Nucleic Acid Sequences," *Computer*, vol. 20, pp. 98–99, July 1987.
- [14] J. R. Nickolls, "The design of the Maspar MP-1: A cost effective massively parallel computer," in *COMPCON Spring 1990*, pp. 25–28, IEEE CS, Feb. 1990.
- [15] R. Singh *et al.*, "A scalable systolic multiprocessor system for biosequence similarity analysis," in *Symp. Integrated Systems* (L. Snyder, ed.), pp. 169–181, Univ. Wash., Apr. 1993.
- [16] R. Wagner and M. Fischer, "The string-to-string correction problem," *JACM*, pp. 168–173, Jan. 1974.
- [17] M. S. Waterman, "Sequence alignments," in *Mathematical Methods for DNA Sequences*, ch. 3, pp. 53–92, Boca Raton, FL: CRC Press, 1989.