Using the PGAS Programming Paradigm for Biological Sequence Alignment on a Chip Multi-Threading Architecture

OUTLINE

- The context
- Related work
- PGAS Parallel Local Sequence Alignment
- Experimental results
- Conclusion and future work

The context

- The massive computational resources required for searching and comparing sequences is a big challenge for biologists today.
- Many parallel computation methods have been developed on high performance computing systems.
- Most of these methods have been developed on distributed memory architectures using the message-passing paradigm.
- Recently, new parallel machines and processor architectures have been introduced for through-put computing (e.g. T2000).

The context

- Most of the efforts have gone into benchmarking the real world performance of enterprise I/O-bound applications, but there are a few examples of compute-centric application analysis for this architecture
- There are no such applications implemented using PGAS languages optimized for these machines.
- Evaluate a parallelization technique for implementing a local DNA sequence alignment algorithm using a PGAS based language, UPC on a chip multi-threading architecture, the UltraSPARCT1.

Related work

- Sequential Algorithms:
  - The global alignment introduced by the Needleman and Wunsch
    - computes a similarity score between two sequences as the sum of all individual elementary similarities
  - The local alignment algorithm introduced by Smith and Waterman
    - Find the most similar subsequence of two sequences
- The local algorithm is based on DPT on two phases:
  - Building phase: computes the total score that indicates the degree of the similarity.
    - The similarity matrix C is a two dimensional table of size and each element C(I,j) can be computed as follows:

$$C(i,j) = Max \begin{cases} C(i-1,j-1) + sbt(X(i),Y(j)) \\ C(i,j-1) + \sigma \\ C(i-1,j) + \sigma \\ 0 \end{cases}$$

  - Back-tracing phase: identifies the corresponding alignment by tracing back along the score matrix.

Related work

- Example:

(sigma= -1, sbt(X(i),Y(j))= + 2 if X(i),Y(j)) are identical, -1 otherwise)

		G	A	T	C	G	G	A	A	T	A	G
	0	1	2	3	4	5	6	7	8	9	10	11
0	0	0	0	0	0	0	0	0	0	0	0	0
G 1	0	**2**	1	0	0	2	2	1	0	0	0	2
A 2	0	1	**4**	**3**	2	1	1	4	3	2	2	1
C 3	0	0	3	3	**5**	4	3	3	3	2	1	1
G 4	0	2	2	2	4	**7**	6	5	4	3	2	3
G 5	0	2	1	1	3	6	**9**	8	7	6	5	4
A 6	0	1	4	3	2	5	8	**11**	10	9	8	7
T 7	0	0	3	6	5	4	7	10	**10**	12	11	10
T 8	0	0	2	5	5	4	6	9	9	**12**	11	10
A 9	0	0	2	4	4	4	5	8	11	11	**14**	13

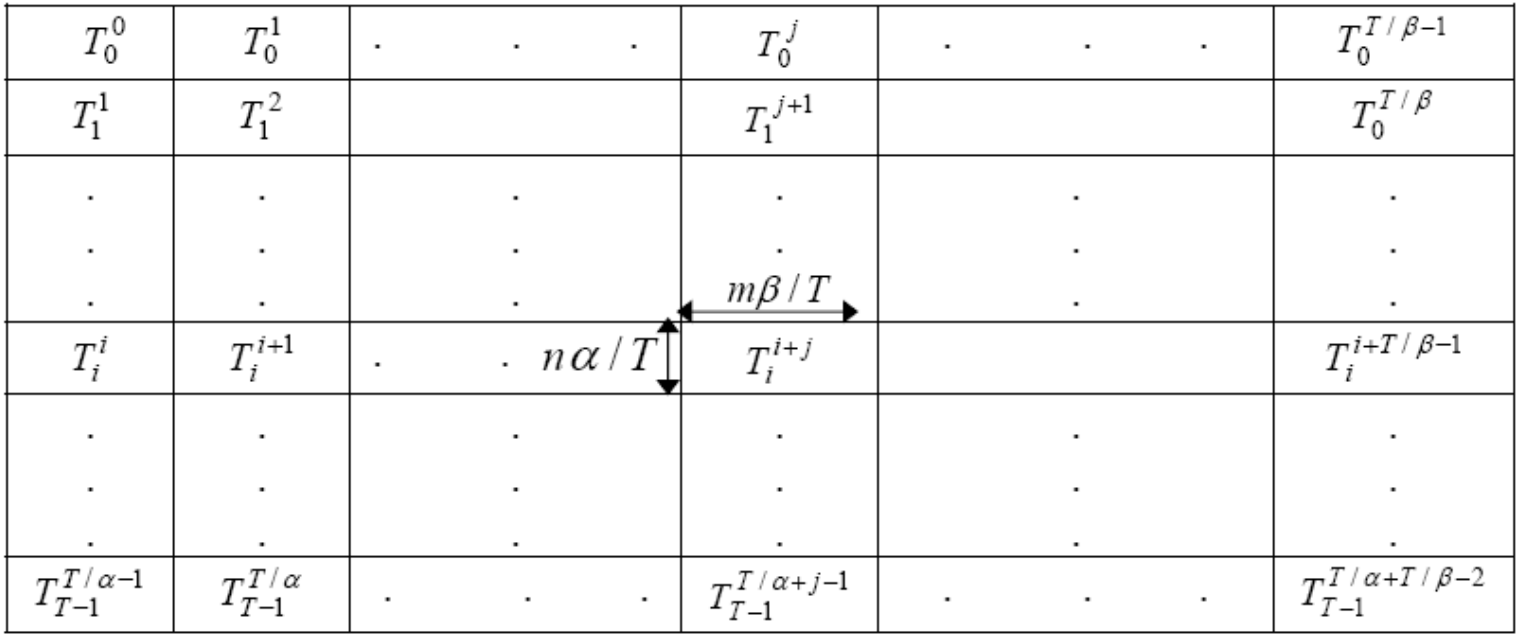
Table 1. The score matrix of X and Y, where the highest one is 14

PGAS Parallel Local Sequence Alignment

- Issue:
  - Hard to be fully parallelized.
  - To compute the value *C(i,j)*, the value of the upper cell *C(i-1,j)*, the left *C(i,j-1)* and the upper-left *C(i-1,j-1)* should be checked.
- Approach proposed in the literature: Wave-front technique
  - Proposed mainly for string editing problem
  - Uses the BSP (Bulk Synchronous Parallel) model
  - Uses a parameterized scheduling scheme to make a compromise between the workload of each processor and the number of communication rounds required
- Unlike this technique, we use the transpose of this matrix (n*m) and we add another factor that allows the reduction of the load imbalance between threads.

PGAS Parallel Local Sequence Alignment

- A general distribution scheme of the workload between threads.



PGAS Parallel Local Sequence Alignment

		G 1	A 2	T 3	C 4	G 5	G 6	A 7	A 8	T 9	A 10	G 11
0	0	0	0	0	0	0	0	0	0	0	0	0
G 1	0	2	1	0	0	2	2	1	0	0	0	2
A 2	0	1	4	3	2	1	1	4	3	2	2	1
C 3	0	0	3	3	5	4	3	3	3	2	1	1
G 4	0	2	2	2	4	7	6	5	4	3	2	3
G 5	0	2	1	1	3	6	9	8	7	6	5	4
A 6	0	1	4	3	2	5	8	11	10	9	8	7
T 7	0	0	3	6	5	4	7	10	10	12	11	10
T 8	0	0	2	5	5	4	6	9	9	12	11	10
A 9	0	0	2	4	4	4	5	8	11	11	14	13

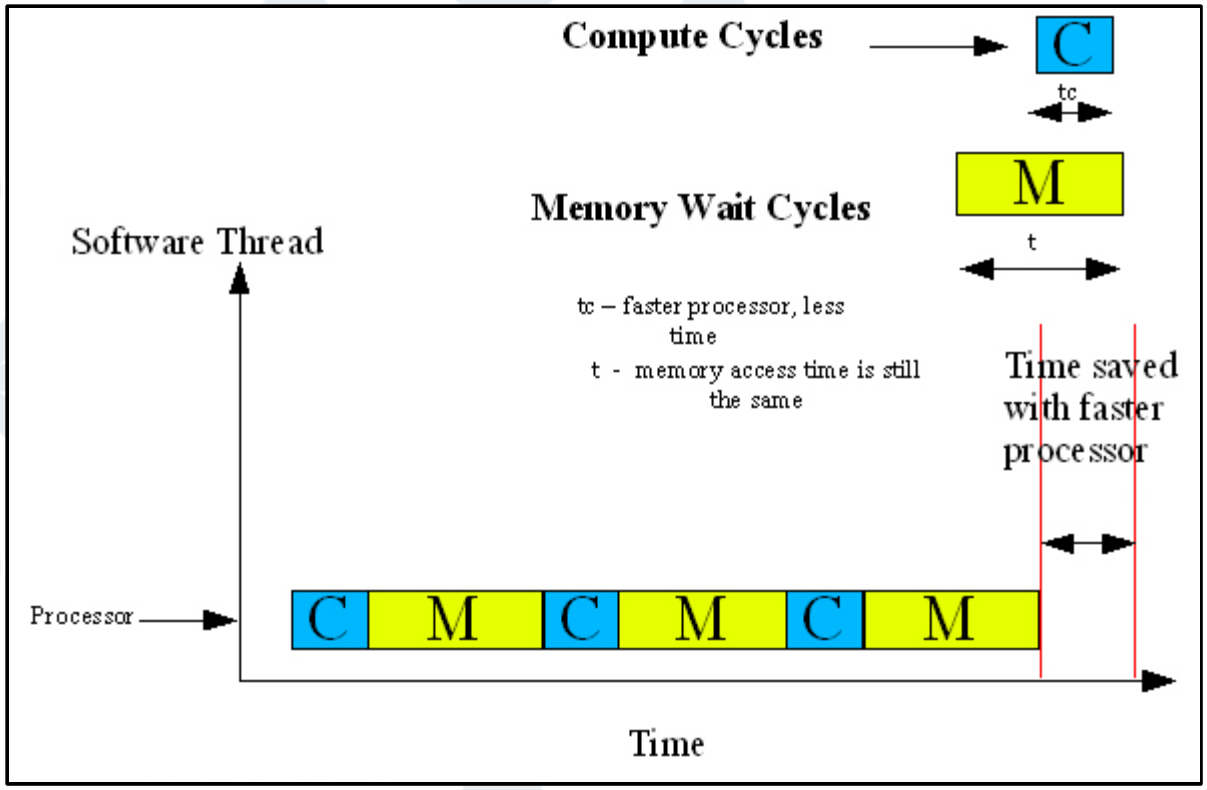
Table 1: the score matrix of X and Y, where the highest score is 14.

		0	G 1	A 2	C 3	G 4	G 5	A 6	T 7	T 8	A 9
	0	0	0	0	0	0	0	0	0	0	0
T0	G 1	0	**2**	1	0	2	2	1	0	0	
	A 2	0	1	**4**	3	2	1	4	3	2	2
	T 3	0	0	*3*	3	2	1	3	6	5	4
T1	C 4	0	0	2	**5**	4	3	2	5	5	4
	G 5	0	2	1	4	7	6	5	4	4	4
	G 6	0	2	1	3	6	**9**	8	7	6	5
T0	A 7	0	1	4	3	5	8	**11**	10	9	8
	A 8	0	0	3	3	4	7	10	*10*	9	11
	T 9	0	0	2	2	3	6	9	12	**12**	11
T1	A 10	0	0	2	1	2	5	8	11	11	**14**
	G 11	0	2	1	1	3	4	7	10	10	13

Table 2: the score matrix of X and Y where the highest score is 14 constructed by two threads, where $\alpha = 2$ and $\beta = 1$.

Chip Multithreading Architecture

Chip multithreading architectures implement concurrency and parallelism through multiple hardware states (sharing of functional units) and multiple cores.



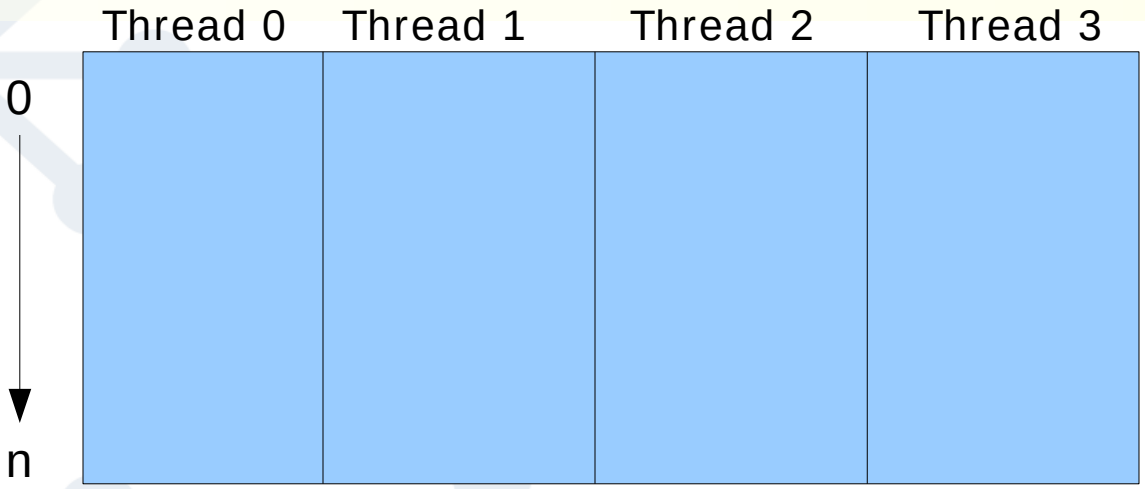
Chip Multithreading Architecture

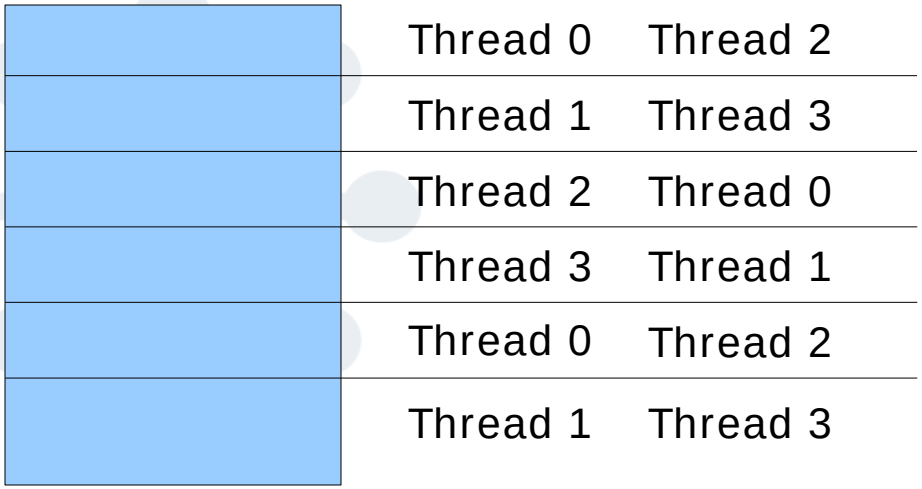
Smith Waterman algorithms are a fair share of both I/O (memory accesses included) and computation (scoring process). It is a perfect map to CMT architectures such as the Sun Niagara family of processors.

The Problem

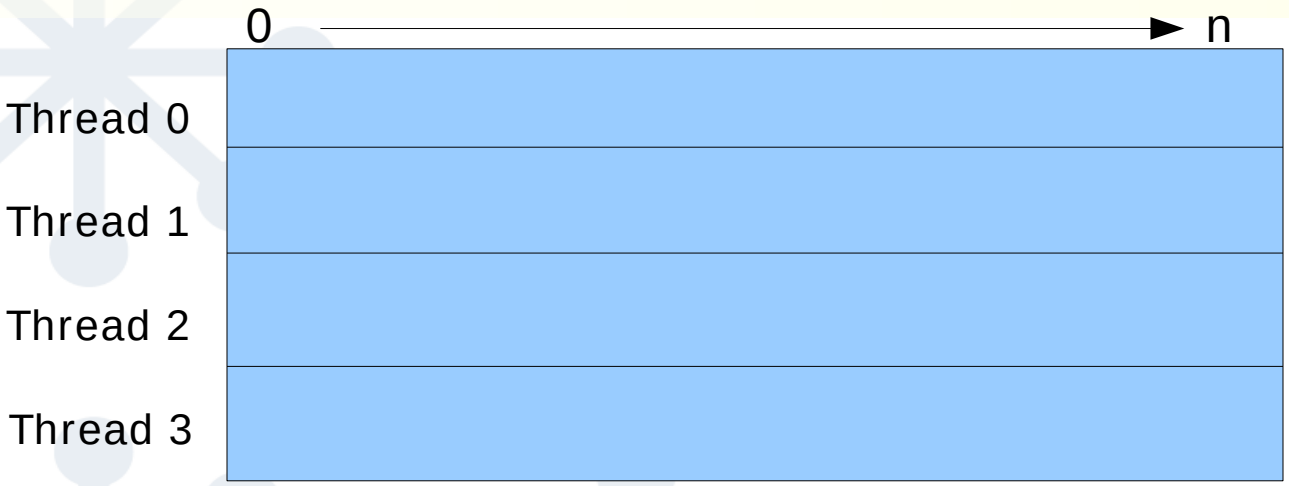
The traditional method of workload and data distribution of parallel smith waterman with wave front does distribution column by column. This actually makes a sizable impact on cache utilization. **This can lead to significantly worse performance on shared cache architectures such as T1/Niagara (15%).**

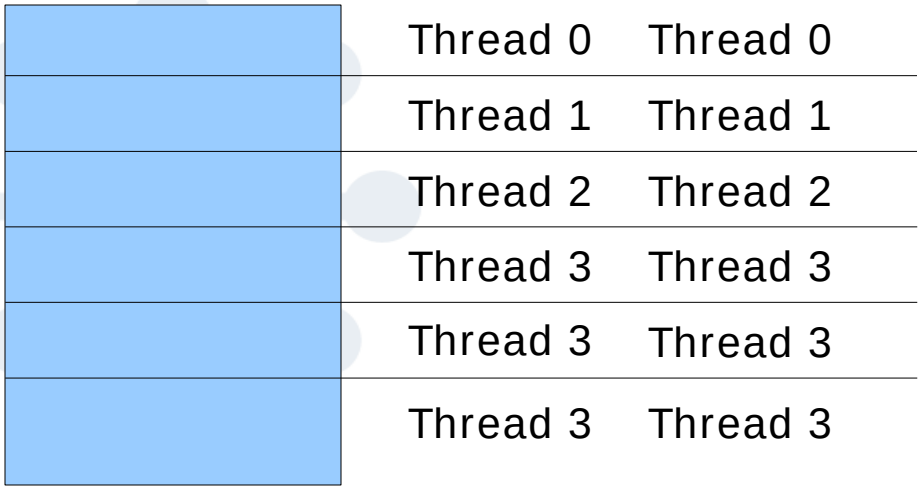
Chip Multithreading Architecture



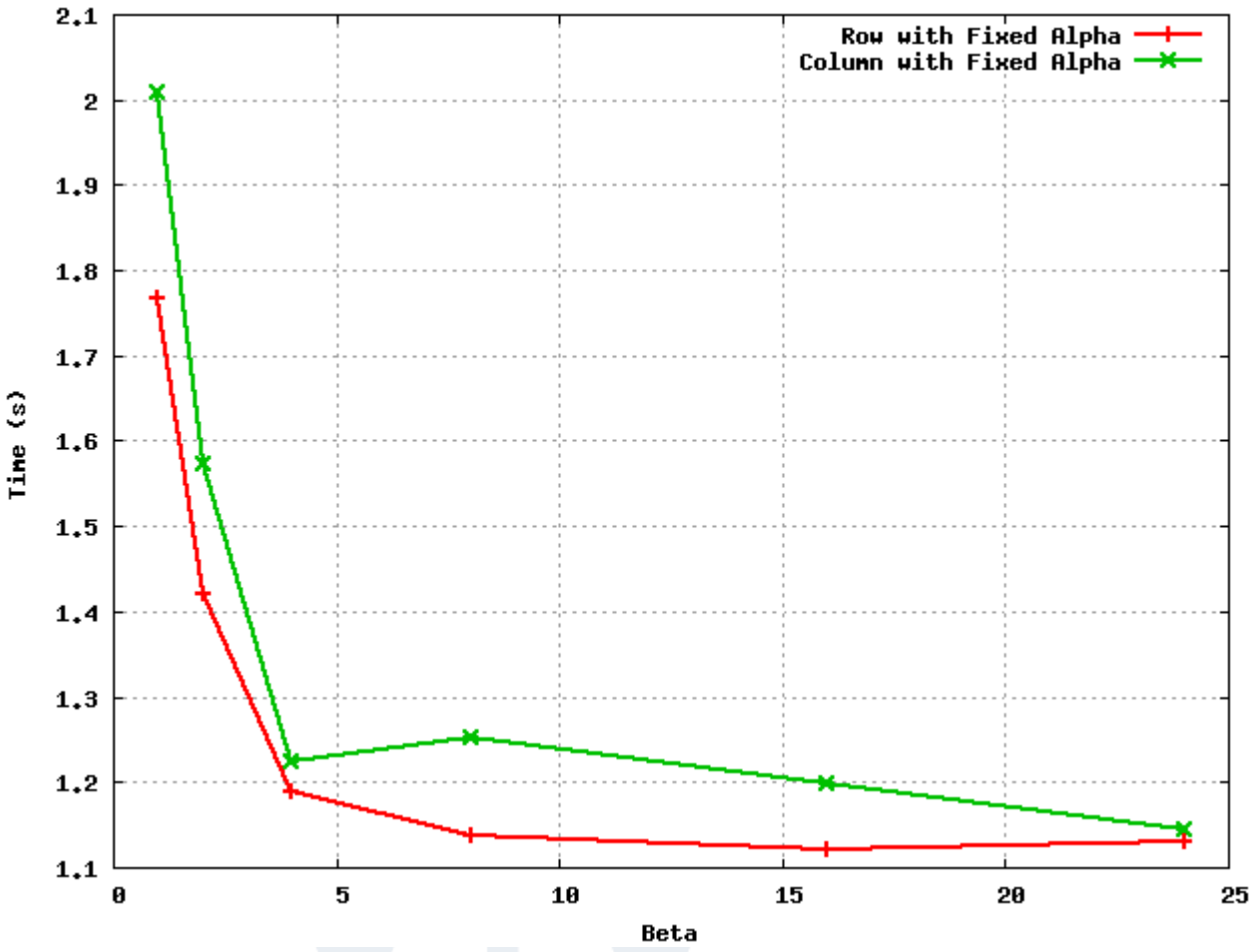


Chip Multithreading Architecture





Chip Multithreading Architecture



Chip Multithreading Architecture

Alpha	Beta	Time for Row	Time for Column
1	1	1.768271	2.008544
1	2	1.420448	1.572432
1	4	1.188866	1.225135
1	8	1.138515	1.252814
1	16	1.120819	1.198830
1	24	1.130519	1.144289
2	1	1.359638	1.580116
2	2	1.255556	1.420041
2	4	1.124641	1.215133
2	8	1.085519	1.258429
2	16	1.097943	1.192002

Experimental results

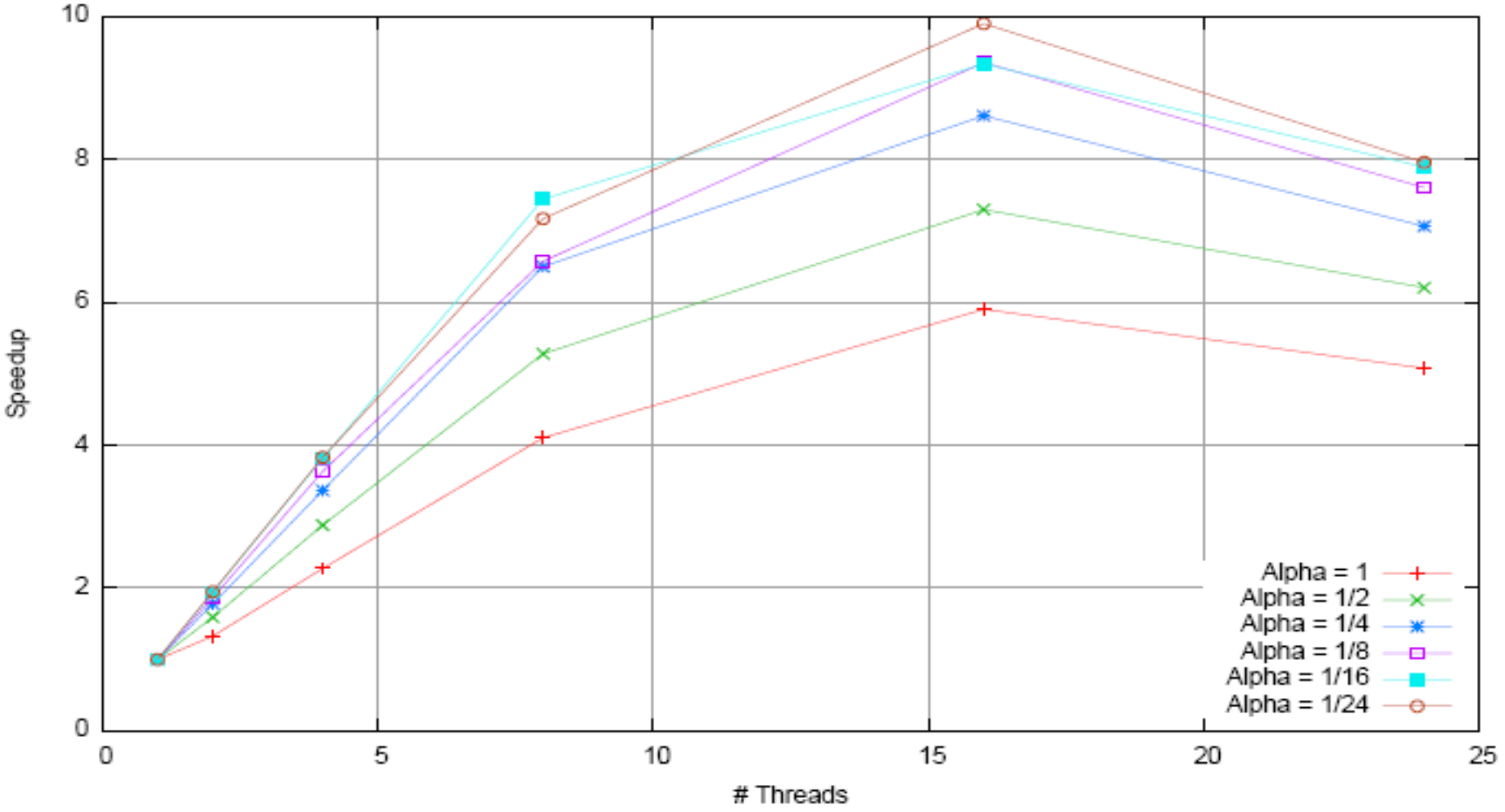


Fig. 1. The speedup with number of threads and varying the value of α.

Experimental results

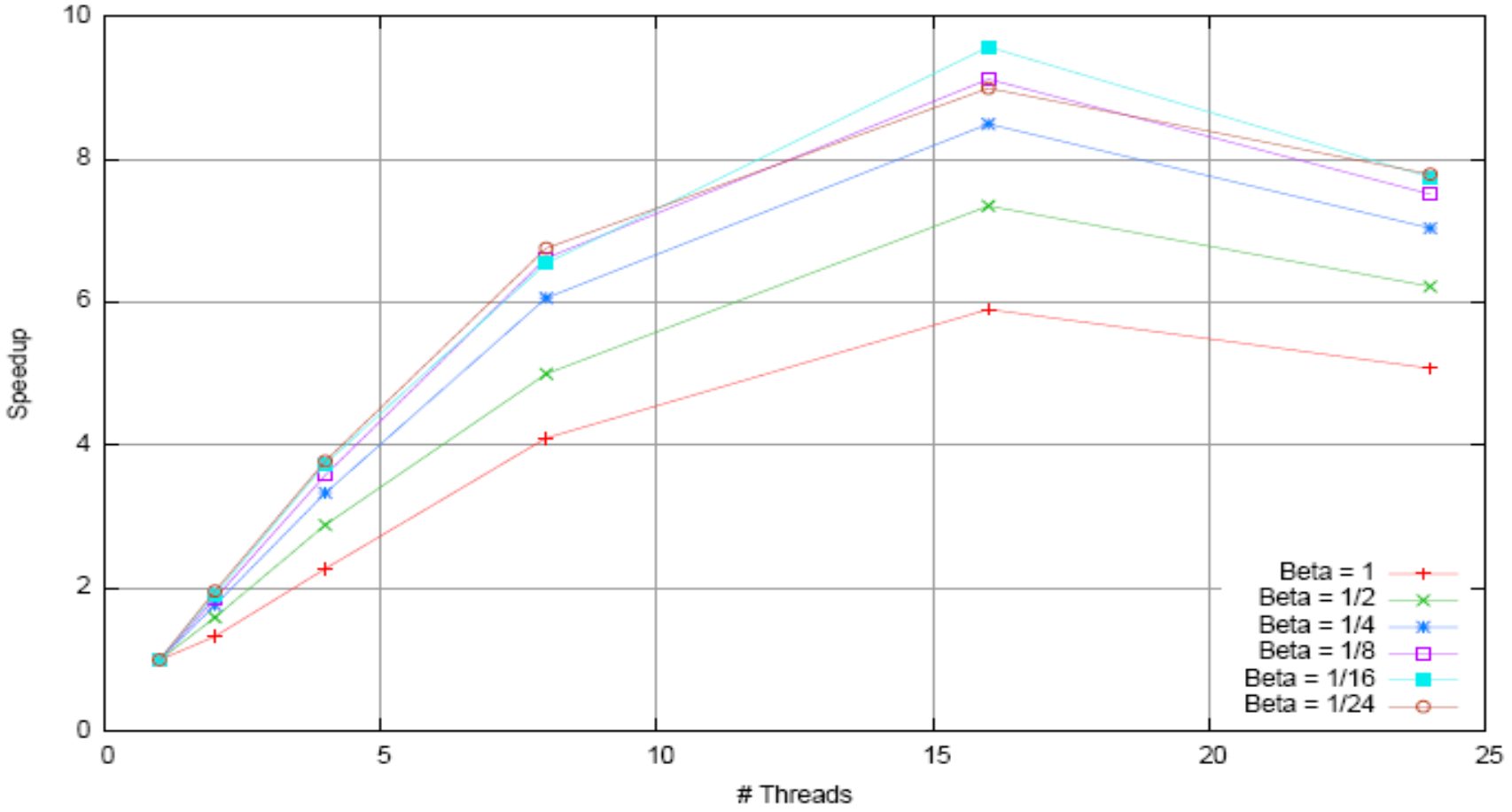


Fig. 2. The speedup with number of threads and varying the value of β.

Experimental results

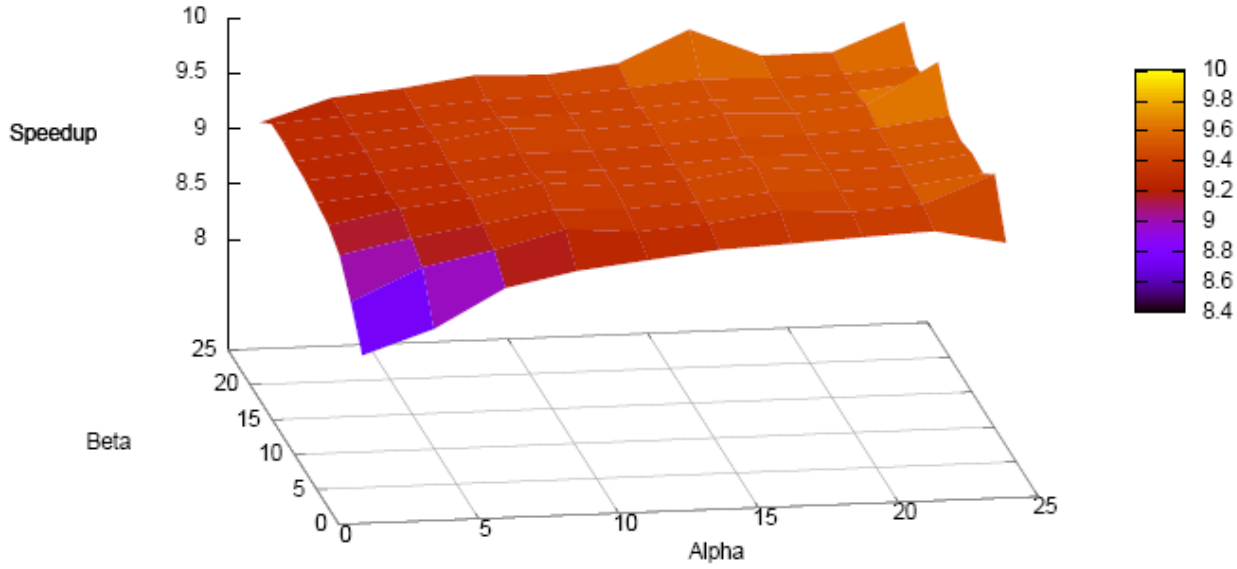


Fig. 3. Speedups for 16 threads by varying the values of α and β.

Experimental results

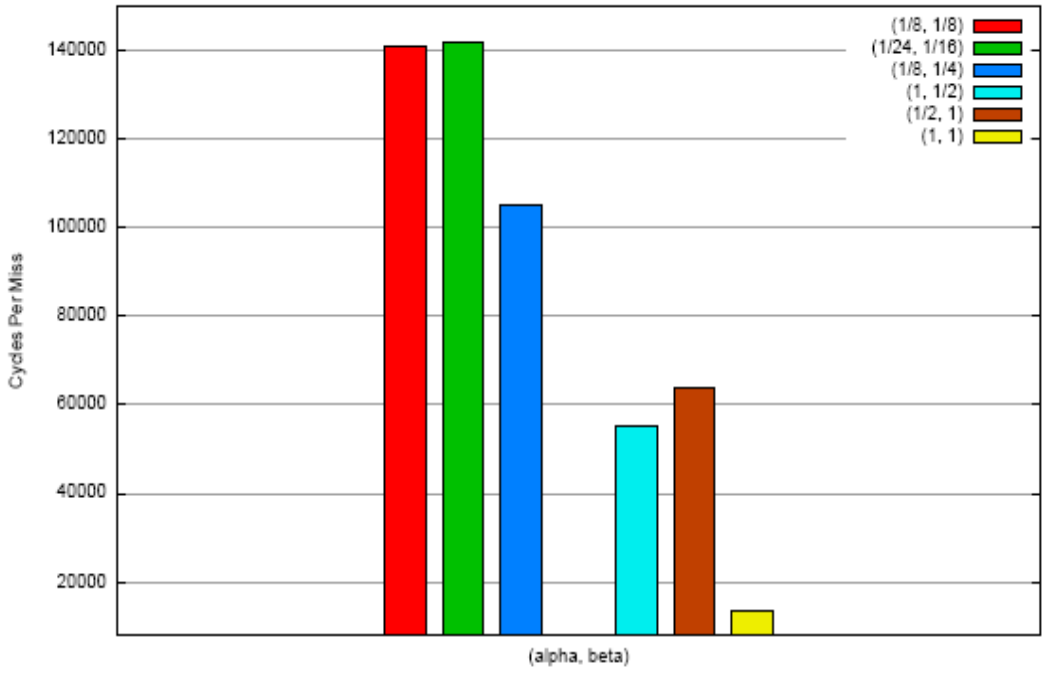


Fig. 4. Cycles Per L2 Cache Miss on Load for best and worst performing combination of α and β.

Conclusion

- A PGAS parallel algorithm for local sequence alignment on the T1 chip multi-threading architecture.
- The performance of the algorithm is based on two tuning parameters, alpha and beta (which ensure the optimal data and workload distributions).
- The programmer should be able to empirically select the best values for these parameters to simultaneously minimize the thread idle time while maximizing cache utilization.

Future Work

- Extending LCS algorithm for multiple sequence alignment.
- Make use of sparse matrix features to do block memory and computation in order to maximize concurrency on CMT architectures.
- Modify algorithm to do zero-filling so it is possible to work on arbitrary size datasets in order to maximize cache utilization.