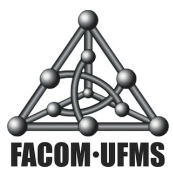
Efficient Parallel Implementations of Multiple Sequence Alignment Using BSP/CGM Model

Jucele F. A. Vasconcellos, Christiane Nishibe, Nalvo F. Almeida and Edson N. Cáceres

Faculdade de Computação
Universidade Federal de Mato Grosso do Sul
Campo Grande - MS
Brazil

15 de Fevereiro de 2014



Motivation and Goals

- Important tool in bioinformatics:
  - Extract biological similarities;
  - Predict protein structure;
  - Reconstruct phylogeny;
  - Illustrate mutations events;
  - Assess sequence conservations.
- Design an BSP/CGM algorithm and implement it in a manycore architecture;
- Compare with Message Passing implementation.

Definition

Five input sequences

s_1	A	T	T	G	C	C	A	T	T	
s_2	A	T	G	G	C	C	A	T	T	
s_3	A	T	C	C	A	A	T	T	T	T
s_4	A	T	C	T	T	C	T	T		
s_5	A	C	T	G	A	C	C			

A multiple sequence alignment

s_1	A	T	T	G	C	C	A	T	T	-	-
s_2	A	T	G	G	C	C	A	T	T	-	-
s_3	A	T	C	-	C	A	A	T	T	T	T
s_4	A	T	C	T	T	C	-	T	T	-	-
s_5	A	C	T	G	A	C	C	-	-	-	-

Approaches

- Exact Algorithms:
  - Carrillo-Lipman.
- Progressive and Iterative Algorithms:
  - ClustalW;
  - Muscle;
  - T-Coffee;
  - Gusfield.
  - FFTNSI;
- Consistency Based Algorithms:
  - CBA.

Building one alignment

S = ACTTCCAGA

T = AGTTCCGGAGG

$$M_{i,j} = max \begin{cases} M_{i-1,j-1} + p[S_i, T_j] \\ M_{i-1,j} + gap \\ M_{i,j-1} + gap \end{cases}$$

S_i S_i -

T_j - T_j

		A	G	T	T	C	C	G	G	A	G	G
	0	-2	-4	-6	-8	-10	-12	-14	-16	-18	-20	-22
A	-2	1	-1	-3	-5	-7	-9	-11	-13	-15	-17	-19
C	-4	-1	0	-2	-4	-4	-6	-8	-10	-12	-14	-16
T	-6	-3	-2	1	-1	-3	-5	-7	-9	-11	-13	-15
T	-8	-5	-4	-1	2	0	-2	-4	-6	-8	-10	-12
C	-10	-7	-6	-3	0	3	1	-1	-3	-5	-7	-9
C	-12	-9	-8	-5	-2	1	4	2	0	-2	-4	-6
A	-14	-11	-10	-7	-4	-1	2	3	1	1	-1	-3
G	-16	-13	-10	-9	-6	-3	0	3	4	2	2	0
A	-18	-15	-12	-11	-8	-5	-2	1	2	5	3	1

S =	A	C	T	T	C	C	-	-	A	G	A	
T =	A	G	T	T	C	C	G	G	A	G	G	
	+1	-1	+1	+1	+1	+1	-2	-2	+1	+1	-1	= +1

Calculate the pairwise alignments

$s_1 =$ A T T G C C A T T
$s_2 =$ A T G G C C A T T
$s_3 =$ A T C C A A T T T T
$s_4 =$ A T C T T C T T
$s_5 =$ A C T G A C C

$$\frac{k(k-1)}{2}$$

$s_1 =$ A T T G C C A T T $s_2 =$ A T G G C C A T T	$s_1 =$ A T T G C C A - - T T $s_3 =$ A - T C C A A T T T T
$s_1 =$ A T T G C C A T T $s_4 =$ A T C T T C - T T	$s_1 =$ A T T G C C A T T $s_5 =$ A C T G - - A C C
$s_2 =$ A T G G C C A - - T T $s_3 =$ A T - C C A A T T T T	$s_2 =$ A T G G C C A T T $s_4 =$ A T C T T C - T T
$s_2 =$ A - T G G C C A T T $s_5 =$ A C T G A C C - - -	$s_3 =$ A T C C A A T T T T $s_4 =$ A T - C - T T C T T
$s_3 =$ A T C C A A T T T T $s_5 =$ A - C T G A - - C C	$s_4 =$ A T C T T C T T $s_5 =$ A - C T G A C C

Find the center sequence S_c

s_1 = A T T G C C A T T s_2 = A T G G C C A T T = 7	s_1 = A T T G C C A - - T T s_3 = A - T C C A A T T T T = -2
s_1 = A T T G C C A T T s_4 = A T C T T C - T T = 0	s_1 = A T T G C C A T T s_5 = A C T G - - A C C = -3
s_2 = A T G G C C A - - T T s_3 = A T - C C A A T T T T = -2	s_2 = A T G G C C A T T s_4 = A T C T T C - T T = 0
s_2 = A - T G G C C A T T s_5 = A C T G A C C - - - = -4	s_3 = A T C C A A T T T T s_4 = A T - C - T T C T T = 0
s_3 = A T C C A A T T T T s_5 = A - C T G A - - C C = -7	s_4 = A T C T T C T T s_5 = A - C T G A C C = -3

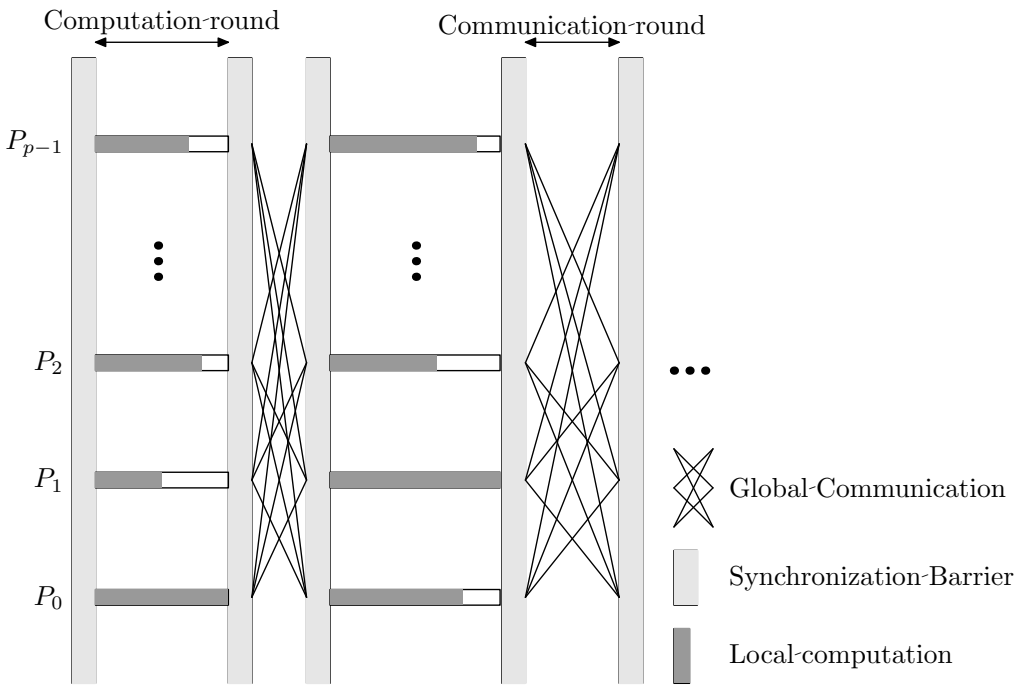
	s_1	s_2	s_3	s_4	s_5	$\sum aln(s_i, s_j)$
s_1		7	-2	0	-3	2
s_2	7		-2	0	-4	1
s_3	-2	-2		0	-7	-11
s_4	0	0	0		-3	-3
s_5	-3	-4	-7	-3		-17

Construct the alignment and add the alignment to the MSA

$s_1 =$ A T T G C C A T T $s_2 =$ A T G G C C A T T	$s_1 =$ A T T G C C A - - T T $s_3 =$ A - T C C A A T T T T
$s_1 =$ A T T G C C A T T $s_4 =$ A T C T T C - T T	$s_1 =$ A T T G C C A T T $s_5 =$ A C T G - - A C C

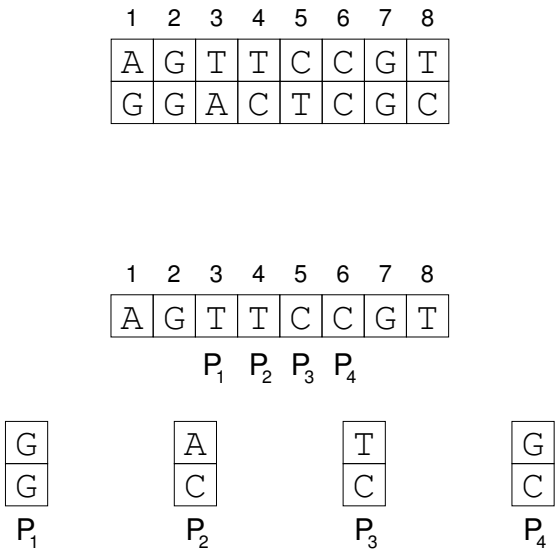
$s_1 =$ A T T G C C A - - T T
$s_2 =$ A T G G C C A - - T T
$s_3 =$ A - T C C A A T T T T
$s_4 =$ A T C T T C - - - T T
$s_5 =$ A C T G - - A - - C C

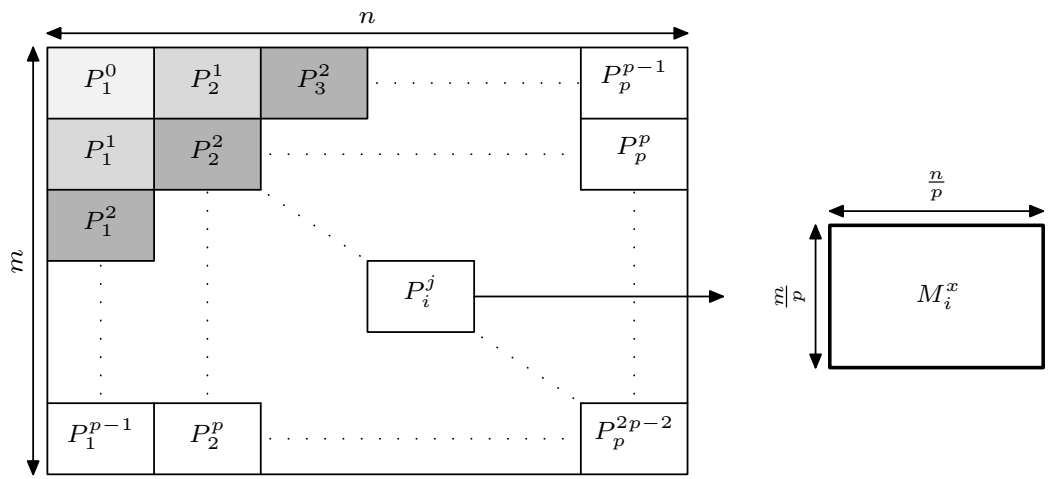
BSP/CGM Model



- $O(p)$ rounds of communication;
- $O(mn/p)$ local memory;

Wavefront Strategy



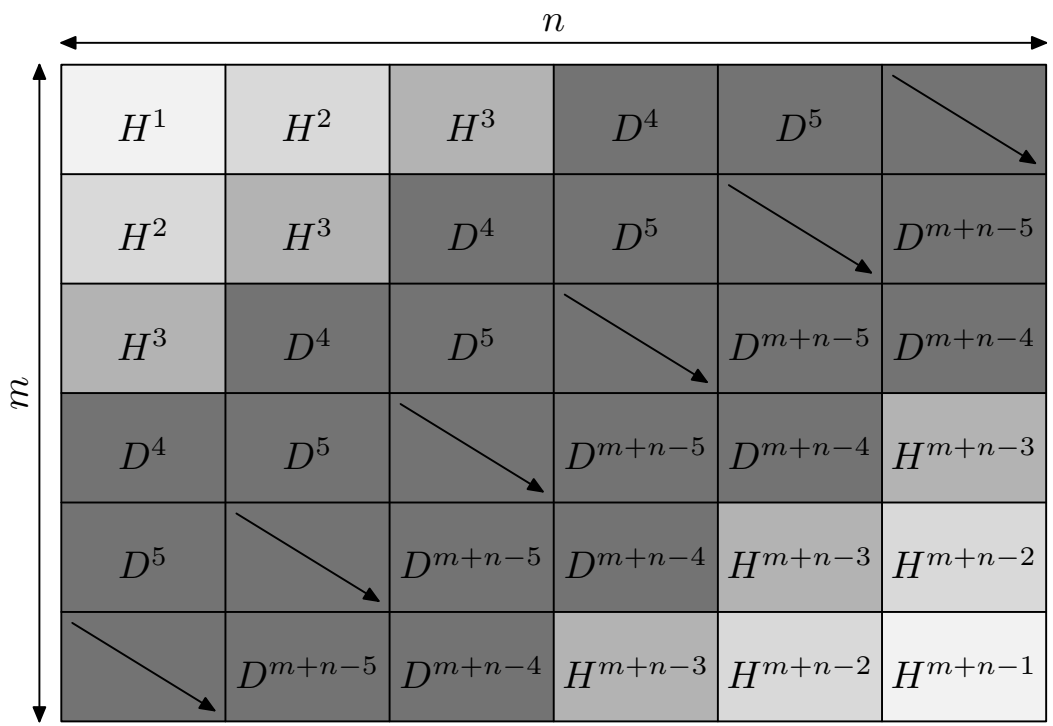


MPI Implementation

1. Calculate the pairwise alignment;
   - $\frac{k(k-1)}{2}$
2. Find the center sequence S_c;
3. Calculate the pairwise alignment between S_c and the other sequences;
4. Construct the alignment and add the alignment to the MSA;

for $1 \leq x \leq k$ **do**
P_1 sends a subsequence of S_x, where $S_x \neq S_c$;
Algorithm Pairwise (p, i, S_c, S_x);
Each P_j constructs a part of the alignment between $S_c S_x$ and sends the alignment to P_1;
P_1 adds the alignment between $S_c S_x$ to MSA;
end for

Wavefront Strategy



CUDA Implementation

1. Calculate the pairwise alignment;
2. Find the center sequence S_c;
3. Calculate the pairwise alignment between S_c and the other sequences;
4. Construct the alignment and add the alignment to the MSA;

for $1 \leq x \leq k$ **do**
  Copy to device the sequence $S_x, S_x \neq S_c$;
  Host and device calculate the pairwise alignment between S_c and S_x;
  Host constructs the alignment $S_c S_x$;
  Host adds $S_c S_x$ to the MSA;
end for

Resources

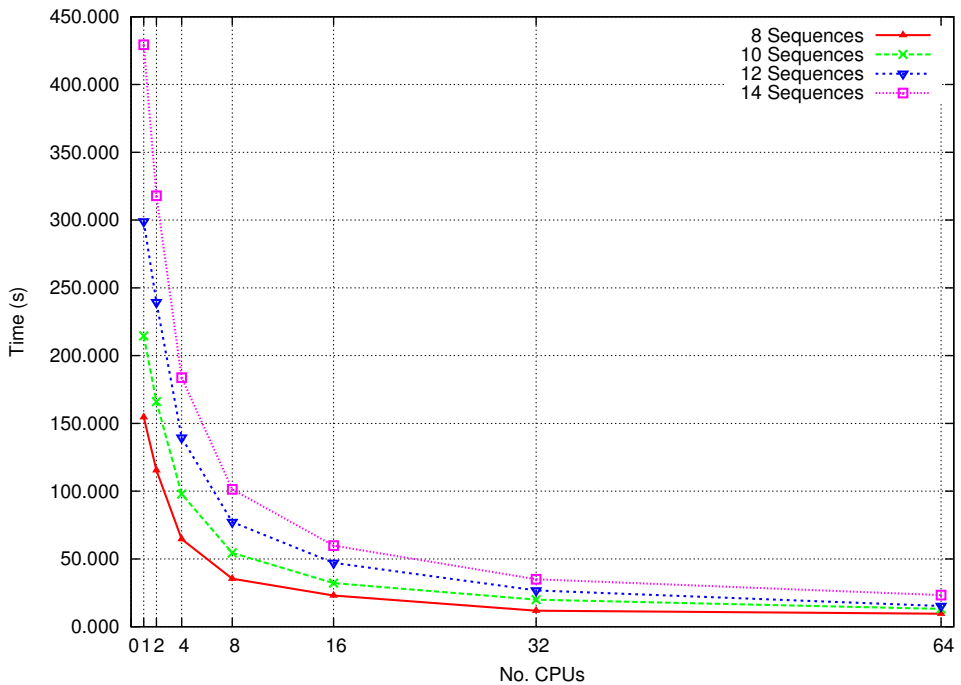
- Carleton Cluster
  - 64 Processor: AMD Opteron 2.2 GHz;
  - Cache: 1024 KB;
  - Memory: 8 GB.
- Desktop - CUDA
  - Processor: Intel Core 2 Quad 2.83 GHz;
  - Cache: 6144 KB;
  - Memory: 4 GB;
  - GeForce GTX 460:
    - 336 CUDA Cores;
    - GPU Clock rate: 1.50 GHz;
    - Global memory: 1024 MBytes.
  - Quadro FX 380:
    - 16 CUDA Cores;
    - GPU Clock rate: 1.10 GHz;
    - Global memory: 255 MBytes.

Input Data

- Number of sequences: 8, 10, 12 and 14;
- Length of sequences: 1024, 4096, 8192 and 16384.

MPI Results

No. of Seqs	$P = 1$	$P = 2$	$P = 4$	$P = 8$	$P = 16$	$P = 32$	$P = 64$
8	154.527	115.399	64.645	35.362	22.970	11.817	9.513
10	214.526	166.007	97.864	54.472	32.158	19.957	13.173
12	299.001	239.347	139.494	77.189	47.121	26.733	15.295
14	429.346	317.902	183.771	101.305	59.804	35.033	23.248



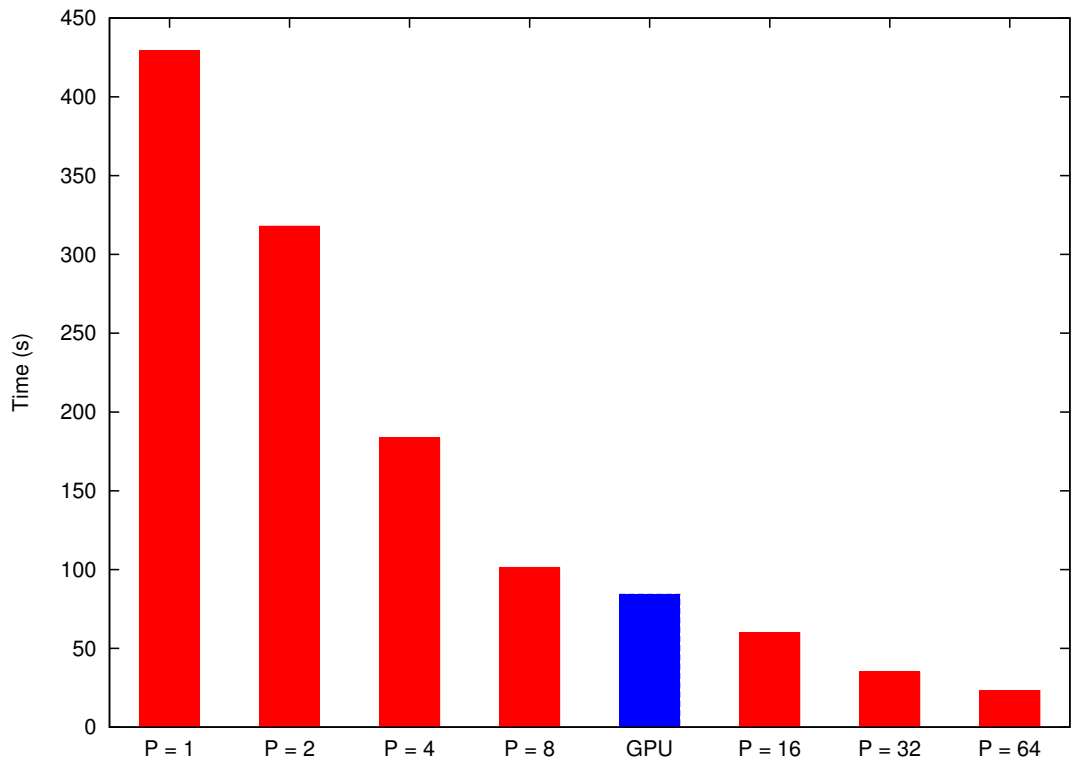
CUDA Results

No. of Seqs	1024	4096	8192	16384
8	0.344	4.837	21.195	33.288
10	0.527	7.448	32.586	47.956
12	0.755	10.588	46.418	64.895
14	1.010	14.245	62.551	84.254

No. of Seqs	$P = 1$	$P = 2$	$P = 4$	$P = 8$	$P = 16$	$P = 32$	$P = 64$
8	1.779	1.331	0.822	0.463	0.269	0.196	0.803
10	2.715	2.141	1.276	0.717	0.405	0.300	0.398
12	3.943	2.982	1.866	1.019	0.588	0.471	0.487
14	5.735	3.935	2.481	1.402	0.796	0.577	0.665

No. of Seqs	$P = 1$	$P = 2$	$P = 4$	$P = 8$	$P = 16$	$P = 32$	$P = 64$
8	154.527	115.399	64.645	35.362	22.970	11.817	9.513
10	214.526	166.007	97.864	54.472	32.158	19.957	13.173
12	299.001	239.347	139.494	77.189	47.121	26.733	15.295
14	429.346	317.902	183.771	101.305	59.804	35.033	23.248

14 sequences, 16384 characters



Conclusions

- Scalable implementations;
- Use different architectures;
- CUDA/GPGPU is suitable for BSP/CGM algorithms.

Future Work

- Improve memory utilization in CUDA;
- Improve the Threads/Kernels/SM's utilization;
- Comparison the results with other approaches.

Thank you!