Calculation of Maximal Overlaps

A look at a parallel algorithm and its running time

John Rivera
Prof. Russ Miller

May 9, 2019

CSE 633 – Parallel Algorithms
SUNY The University at Buffalo

Motivation

Cis-Regulatory Modules (CRMs) are segments of an organism's DNA that regulates the expression of certain developmental traits during that organism's lifecycle.

The REDfly project (redfly.ccr.buffalo.edu), among other things, contains the largest database of fruit fly (Drosophila melanogaster) CRMs.

As you probably guessed, discovering CRMs is hard, especially when we do not know where to look. Inferred CRMs (iCRMs), which are inferred from known CRMs (hence the name), are "guesses" at where potential CRMs may be located. Scientists use them to help narrow down the segments of the DNA to focus on during experiments.

Motivation (cont'd)

iCRMs are the regions where two or more CRMs overlap (actually, there is a bit more to it, but this description is sufficient for our purposes today).

As an example, the following is a (randomized) nucleic acid sequence of length 25, and five (fictional) CRMs:

```
gtgtccctgggctgctgcacaggag
gtgtccc          caca
    ccctggg     gcacaggag
        gggctgctgcaca
```

The maximal overlaps would be `ccc`, `ggg` and `caca`; those are the iCRMs (more or less).

The Problem in More Formal Terms

Given a set $S = \{\langle l_0, r_0\rangle, \langle l_1, r_1\rangle, \ldots, \langle l_n, r_n\rangle\}$ of n segments, find a set of segments that maximally overlaps all segments $\in S$.

As an example derived from the previous slide, let the set of intervals be $\{\langle 0, 6\rangle, \langle 4, 10\rangle, \langle 8, 20\rangle\ \langle 16, 24\rangle, \langle 17, 20\rangle\}$. From this set, the expected output would be $\{\langle 4, 6\rangle, \langle 8, 10\rangle, \langle 17, 20\rangle\}$.

The Algorithm at a High Level

The problem, at a high level, is a slightly more complicated version of the Maximal Overlapping Point problem (described in *Algorithms Sequential and Parallel: A Unified Approach*, 3rd Ed., by Russ Miller & Laurence Boxer, pp. 195 - 196).

Assume that we are given the endpoints as two arrays: $E = \{e_1, e_2, \dots, e_n\}$ containing all endpoints sorted in ascending order, and $O = \{o_1, o_2, \dots, o_n\}$, where $o_i \in \{1, -1\}$, consisting of '*operands*' representing the orientations $\forall e \in E$ – that is, operand o_i is the orientation of endpoint e_i and equals 1 if e_i is a starting endpoint, or -1 otherwise. The algorithm consists of two parts:

1. Calculate the prefix sums $\forall o \in O$; let this be an array $P = \{p_1, p_2, \dots, p_n\}$.
2. Find all local maximums $\in P$.

If p_i is a local maximum, then segment $\langle e_i, e_{i+1} \rangle$ is a maximal overlap.

The Algorithm at a High Level (cont'd)

An example:

endpoint	operand	prefix sum	
0	1	1	
4	1	2	$\longleftarrow$
6	-1	1	
8	1	2	$\longleftarrow$
10	-1	1	
16	1	2	
17	1	3	$\longleftarrow$
20	-1	2	
20	-1	1	
24	-1	0	

From the above table, we can see that segments $\langle 4, 6\rangle$, $\langle 8, 10\rangle$ and $\langle 17, 20\rangle$ are the maximal overlaps.

The Sequential Algorithm

Prefix Sum

```
$p_1 \leftarrow o_1$
for $i \leftarrow 2$ to $n$ do
    $p_i \leftarrow p_{i-1} + o_i$
end for
```

Local Maximums

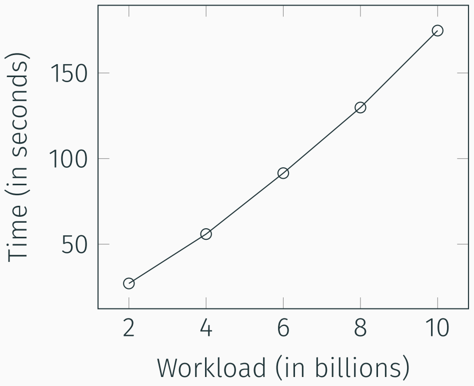
```
for $i \leftarrow 2$ to $n-1$ do
    if $p_{i-1} < p_i > p_{i+1}$ then
        $p_i$ is a local maximum
    end if
end for
```

The running time is $\Theta(n) + \Theta(n) \equiv \Theta(n)$.

Sequential Algorithm Running Times

$p = 1$

n	t
2×10^9	27.06
4×10^9	55.90
6×10^9	91.49
8×10^9	129.82
10×10^9	174.76



The Parallel Algorithm

The parallel prefix sum algorithm is described in *Sequential and Parallel Algorithms*, pp. 175 - 178.

The local maximums algorithm makes use of the result of the prefix sum algorithm and is similarly calculated in parallel. The first and last elements of each processor's partition is gathered and scattered so that each processor receives its predecessor's last element and its successor's first element; this is so the processor can determine whether its first and/or last element is a local maximum.

Intuitively, the running time is $\Theta(n/p) + \Theta(n/p) \equiv \Theta(n/p)$, where p is the number of processors.

Alternatively, the running time is $\Theta(n/\lg n)$, when $p = \lg n$.

The Parallel Algorithm (cont'd)

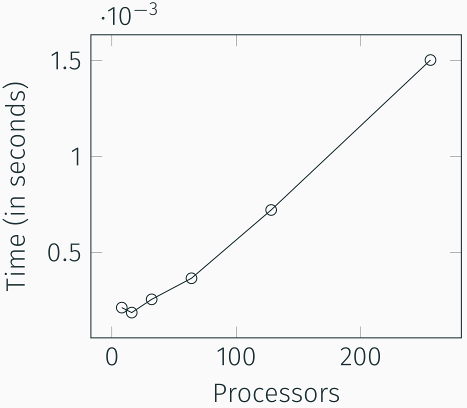
An example via a walk-through with 2 processors:

initial	1	1	-1	1	-1	1	1	-1	-1	-1
scatter	1	1	-1	1	-1	1	1	-1	-1	-1
prefix sum	1	2	1	2	1	1	2	1	0	-1
gather					1					-1
prefix sum					1					0
scatter					0					1
fix	1	2	1	2	1	2	3	2	1	0
gather	1				1	2				0
scatter	0				2	1				0
local maximums	1	2	1	2	1	2	3	2	1	0
		$\downarrow$		$\downarrow$			$\downarrow$			
	0	4	6	8	10	16	17	20	20	24

Parallel Algorithm Running Times

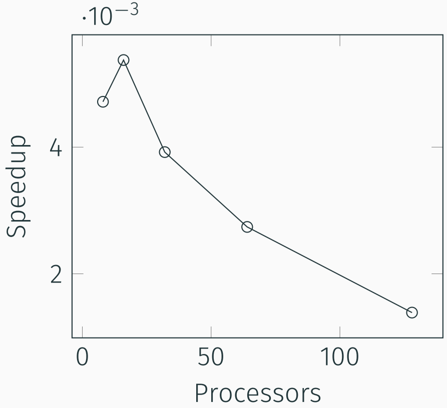
$n = 10 \times 10^3$

p	t
8	0.000212
16	0.000186
32	0.000255
64	0.000365
128	0.000721
256	0.001503



Parallel Algorithm Speedup

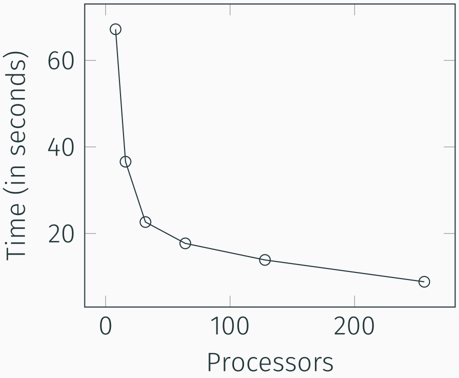
$n = 10 \times 10^3$	
p	s
8	0.004717
16	0.005376
32	0.003922
64	0.002740
128	0.001387
256	0.000665



Parallel Algorithm Running Times (cont'd)

$n = 10 \times 10^9$

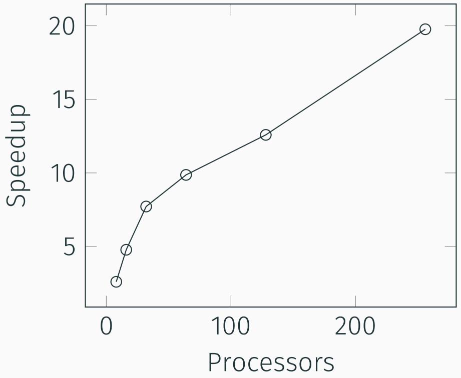
p	t
8	67.189068
16	36.584433
32	22.646106
64	17.715798
128	13.886721
256	8.842507



Parallel Algorithm Speedup (cont'd)

$n = 10 \times 10^9$

p	s
8	2.601018
16	4.776895
32	7.716999
64	9.864642
128	12.584684
256	19.763625



Lessons Learned

- Not All Nodes Are Created Equal: I initially set the minimum RAM per node to 24 GB. However, I ran into intermittent failures and strange timing spikes. Reviewing the cluster details on the CCR webpage, I noticed that the nodes with 24 GB of RAM are part of a older group of servers. Bumping the minimum RAM to 48 GB resolved my issues.
- Why I Started With 8 Nodes: I initially planned to run the parallel algorithm on 2 and 4 nodes as well, however, `MPI_Scatter(8)` and `MPI_Gather(8)` takes an *int* rather than a *size_t* as the size arguments. Oops.
- Integer Troubles: Or how I learned to stop worrying and love `stdint.h`. I was initially using *unsigned* for everything. An *unsigned* can hold up to approximately 4.2 billion values. What could go wrong? As is obvious in hindsight, the *size* of the dataset can be up to 10 billion!

Future Work

- The work presented today gets us only the maximal overlapping segments. We actually also want the component segments themselves. This is easy to do sequentially in $O(n^2)$ time, but I am confident that this is an area that would also benefit from parallelization.
- Remember when I said my description of an iCRM was sufficient for our purposes today? Well, an additional requirement for an iCRM is that the intersection of all overlapping CRMs' regulated expressions does not equal $\emptyset$. So I hope to be able to devise an efficient algorithm for finding all maximal overlaps which also allows for special conditions on whether two segments overlap. This is a much harder problem!

References

- Rivera, John, Keränen, Soile V. E., Gallo, Steven M. and Halfon, Marc S. (2018). *REDfly: the transcriptional regulatory element database for Drosophila*. Nucleic Acids Res.; doi:10.1093/nar/gky957
- Miller, Russ and Boxer, Lawrence. (2013). *Sequential and Parallel Algorithms: A Unified Approach, 3rd Ed*. Cengage Learning.
- *Academic Compute Cluster (UB-HPC)*. Center for Computational Research. http://www.buffalo.edu/ccr/support/research_facilities/general_compute.html