Parallel Prefix Scan

Gauthier Castro, David Beley, Bich Ngoc Hoang

7/12/2018

1. Prefix Scan
2. Applications
3. Parallelization Methods
4. Benchmark

Prefix Scan

Algorithm

Input: $x_0, x_1, \ldots, x_{n-1}$

Output: $s_0, s_1, \ldots, s_{n-1}$

Operator: $\otimes$

Prefix scan general formulation:

$$
\begin{gathered}
s_0 = x_0, \\
s_1 = x_0 \otimes x_1, \\
s_2 = x_0 \otimes x_1 \otimes x_2, \\
\ldots, \\
s_{n-1} = x_0 \otimes x_1 \otimes \ldots \otimes x_{n-1}
\end{gathered}
$$

Simple Example With Cumsum

```r
x <- c(12, 5, 13)
cumsum(x)
```

```
## [1] 12 17 30
```

Inclusive Vs. Exclusive Scan

In inclusive scan, x_i is included in s_i. In exclusive prefix scan, x_i is not included.

Examples With Cumsum

```
x <- c(12, 5, 13)
cumsum_inclusive(x)
```

```
## [1] 12 17 30
```

```
x <- c(12, 5, 13)
cumsum_exclusive(x)
```

```
## [1]  0 12 17
```

Applications

Polynomial Calculation

$$P = 7 + 5x - 3x^2 - 6x^3 + 3x^4$$

Exclusive Prefix Scan With Product Operator

$$\begin{array}{ccccc} x & x & x & x & x \\ s_0 = x^0 & s_1 = x^1 & s_2 = x^2 & s_3 = x^3 & s_4 = x^4 \end{array}$$

Multiplication of the Input Vectors and the Coefficient Vectors

$$\begin{array}{ccccc} x^0 & x^1 & x^2 & x^3 & x^4 \\ * & * & * & * & * \\ 7 & 5 & -3 & -6 & 3 \end{array}$$

Calculation of Polynomial

$$x = 7$$
$$P = 7 + 5x - 3x^2 - 6x^3 + 3x^4$$

```
x
```

```
## [1] 7 7 7 7
```

```
c(1,cumprod(x))
```

```
## [1]    1    7   49  343 2401
```

```
coef
```

```
## [1]  7  5 -3 -6  3
```

Calculation of Polynomial

```r
res
```

```
## [1]      7     35   -147 -2058   7203
```

```r
sum(res)
```

```
## [1] 5040
```

Quick sort

```r
x
```

```
##  [1] 19 24 22 47 27  8 28 39  4 43 11 49 45 43  2 13
```

```r
x[x > 28]
```

```
## [1] 47 39 43 49 45 43
```

```r
as.integer(x>28)
```

```
##  [1] 0 0 0 1 0 0 0 1 0 1 0 1 1 1 0 0
```

```r
cumsum(as.integer(x>28))
```

```
##  [1] 0 0 0 1 1 1 1 2 2 3 3 4 5 6 6 6
```

Parallelization Methods

Algorithm: Log-Based Method

for $i \leftarrow 0$ to $[\log_2 n] - 1$ ***do***

- ***for*** $j \leftarrow 0$ to $n - 1$ ***do in parallel***
  - ***if*** $j < 2^i$ ***then***
    - $x_j^{i+1} \leftarrow x_j^i$
  - ***else***
    - $x_j^{i+1} \leftarrow x_j^i + x^i_{j-2^i}$

Algorithm: Log-Based Method - Illustration

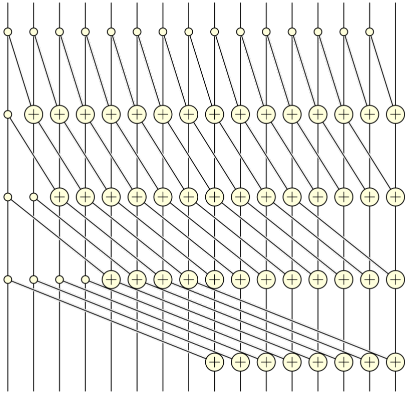

Figure 1: log-based Illustration

Algorithm: Chunk-Based

Ti: thread i

p : number of chunks

```
break the array into p blocks
parallel for i = 0,...,p-1
  Ti does scan of block i, resulting in Si
form new array G of rightmost elements of each Si
do parallel scan of G
parallel for i = 1,...,p-1
  Ti adds Gi to each element of block i+1
```

Algorithm: Chunk-Based - Illustration

Say we have the array and break it into 3 sections:

2 25 26 8	50 3 1 11	7 9 29 10

Apply a scan to each section:

2 27 53 61	50 53 54 65	7 16 45 55

The result:

2 27 53 61	111 114 115 126	133 142 171 181

Benchmark

Functions Implemented

- cs: sequential cumsum
- vcs: vectorized cumsum
- scs: “sapply” cumsum
- pscs: parallel “sapply” cumsum
- fcs: “foreach” cumsum
- pfcs: parallel “foreach” cumsum
- c-cs: compiled cs
- c-scs: compiled scs
- c-vcs: compiled vcs

Benchmark

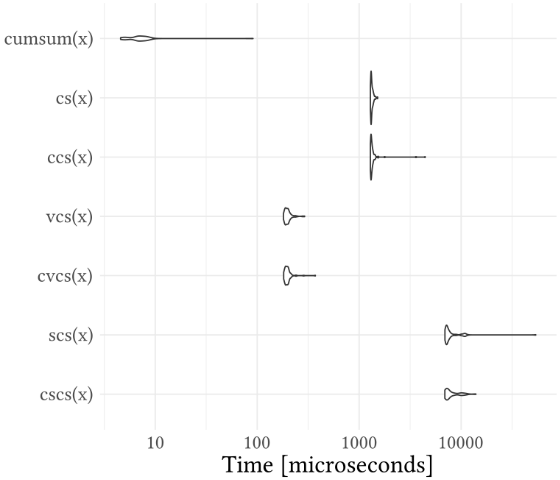

Figure 2: Cumsum functions comparison

Conclusion

To go Further

- Blelloch Algorithm (Work-Efficient Algorithm)

References

- Matloff N., Parallel Computing for Data Science, Chapter 11: Parallel Prefix Scan, 2016
- Blelloch G.E., Prefix Sums and Their Applications, 1990
- Sengupta S. et al., A Work-Efficient Step-Efficient Prefix-Sum Algorithm, 2006