Prefix scan

Gauthier Castro, David Beley, Hoang Bich Ngoc

7/12/2018

Section 1

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:

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

Simple Example With Cumsum

```
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
```

Section 2

Applications

Polynomial Calculation

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

Exclusive Prefix Scan With Product Operator

$$\underset{s_0 = x^0}{x} \quad \underset{s_1 = x^1}{x} \quad \underset{s_2 = x^2}{x} \quad \underset{s_3 = x^3}{x} \quad \underset{s_4 = x^4}{x}$$

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

```
res
```

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

```
sum(res)
```

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

Section 3

Parallelization Methods

Algorithm: Log-Based Method

for i $\leftarrow$ 0 to $\lceil \log_2 n \rceil$ - 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^~
```

Illustration

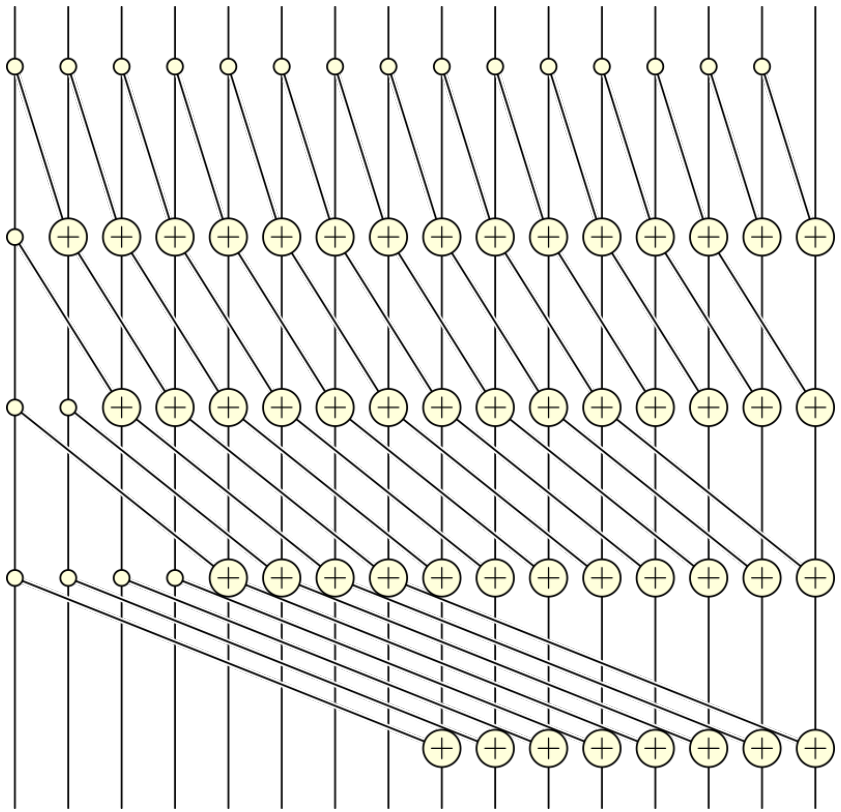
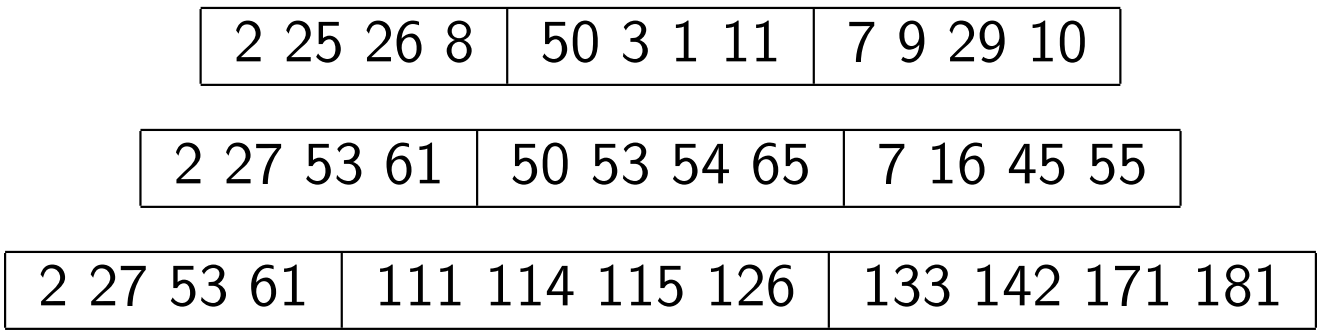


Figure 1: log-based Illustration

Algorithm: Chunk-Based

```
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



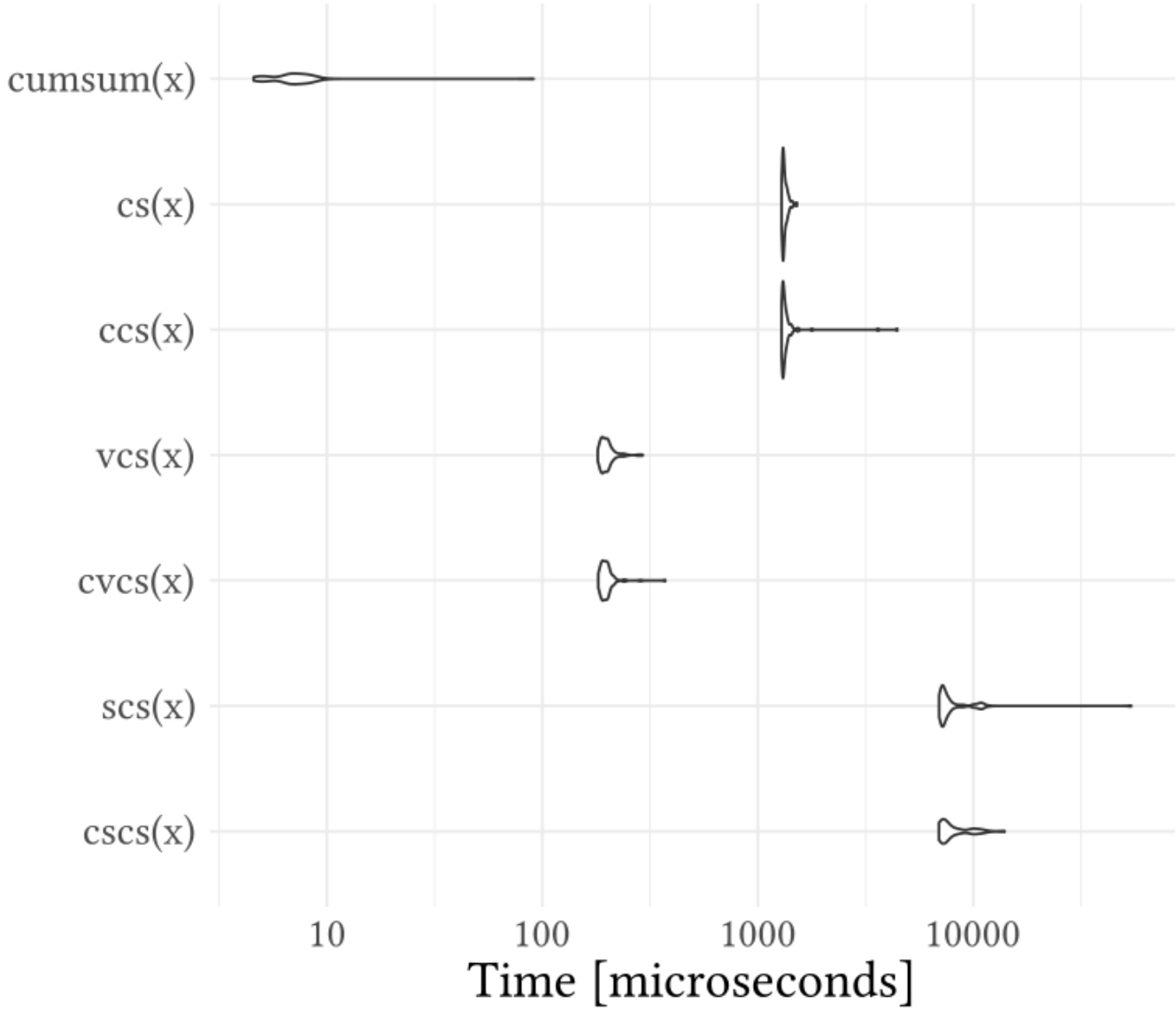
Section 4

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


cumsum(x)
cs(x)
ccs(x)
vcs(x)
cvcs(x)
scs(x)
cscs(x)
10
100
1000
10000
Time [microseconds]

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