Lecture 2-3: Scan

Advanced Programming for HPC

Professor Lilian Aveneau

1 SCAN pattern

- Very important pattern for parallelism on vector machine
- And by extension, on hybrid machine with many (vector) cores ...
  - Like GPUs
- Several variants
  - Inclusive scan
  - Exclusive scan
  - Segmented versions

1.1 Inclusive version

- Let the input be:
  - An array of values $\{X_i\}$
  - An associative operator $\bigoplus$
- It computes the following array of values $\{Y_i\}$

$$Y_i = \bigoplus_{k=1}^{i} X_i$$

- Examples
  - $\{1,1,1,1,1,1,1,1\}$ and addition over $\mathbb{Z}$: $\{1,2,3,4,5,6,7,8\}$
  - $\{1,2,3,4,5,6,7,8\}$ and multiplication over $\mathbb{Z}$: $\{1,2,6,24,120,720,5040,40320\}$

1.1.1 PRAM version

Inclusive SCAN pattern, EREW indexed version

```
{ copy in O(1) from X to Y }
FOR each PE i ∈ [1 … n] in parallel:
    Y[i] ← X[i]
END FOR
{ loop in O(log n): pointer jumping! }
j ← 1
WHILE j < n:
    FOR each PE i ∈ [1 … n − j] in parallel:
        aux[i] ← Y[i]
        aux[i] ← aux[i] ⊕ Y[i+j] { Take care to order }
        Y[i+j] ← aux[i]
    END FOR
    j ← j × 2
END FOR
```

1.1.2 Version with p processors

- Less EP, so more sequential code
  - Complexity is increasing, but so is efficiency
- Use associativity!

$$Y_i = \bigoplus_{j=1}^{i} X_j = \left\{ \bigoplus_{j=1}^{k} X_j \right\} \oplus \left\{ \bigoplus_{j=k+1}^{i} X_j \right\}$$

- Method: example with $\{1,1,1,1,1,1,1,1\}$ and $p = 2$ EP

1. Make an inclusive SCAN by PE (on data block): $\{1,2,3,4\}$ et $\{1,2,3,4\}$
2. Then add Y_4 to the values in the second block: $Y_4 \oplus \{1,2,3,4\} = \{5,6,7,8\}$

1.1.2 Version with p processors

- Efficiency?
  - One processor $\Rightarrow$ $7/(1 \times 7)$ so 1 ☺
  - Two processors $\Rightarrow \frac{7}{2\times(3+2)} = \frac{7}{10}$
  - Eight processors $\Rightarrow \frac{7}{8\times(1+1+1)} \sim \frac{1}{3}$
- With $p = 4$ it gives
  - Per EP: $\{1,2\}$ and $\{1,2\}$ and $\{1,2\}$ and $\{1,2\}$
  - Then you must calculate a value per block!? It is a SCAN
    - p is small, so sequential SCAN $\Rightarrow$ 2 operations
    - This gives $\{2,4,6,8\}$ or $\{2,4,6\}$ because the last value is useless
  - At last, the addition over $p - 1$ blocks: 2 parallel operations per EP
    - $\{3,4\}$ et $\{5,6\}$ et $\{7,8\}$
  - Efficiency $\Rightarrow \frac{7}{4\times(1+2+2)} = \frac{7}{20}$

1.1.2 Version with p processors

Inclusive SCAN pattern, p-processor version

```
{ Block size to handle per EP }
blockSize ← (n + p − 1)/p

{ First step: sequential per block SCAN }
FOR EACH PE i ∈ [1 … p] IN PARALLEL:
    blockStart ← 1 +blockSize× (i − 1) { 1+ -> or X'First }
    Y[blockStart] ← X[blockStart]
    FOR EACH j ∈ [1 …blockSize-1]:
        IF blockStart+j ≤ n THEN
            Y[blockStart+j] ←
                         Y[blockStart+j-1] ⊕ X[blockStart+j]
        END IF
    END FOR
END FOR
```

1.1.2 Version with p processors

Inclusive SCAN pattern, p-processor version

```
{ Second step : SCAN on the last values of the p-1
  first blocks }

aux[1] ← Y[blockSize] { 1 -> or X'First }

FOR EACH j ∈ [2 ...p−1]: { In sequential, e.g. with 1st EP }
    aux[j] ← aux[j-1] ⊕ Y[1+blockSize*j-1]
END FOR
```

1.1.2 Version with p processors

Inclusive SCAN pattern, p-processor version

```
{ Last step: add the partial sums to the last blocks }

FOR EACH PE i ∈ [1 … p − 1] IN PARALLEL: { p-1 processors }
    blockStart ← 1 + blockSize × i
    FOR EACH j ∈ [0 … blockSize − 1]:
        IF blockStart+j ≤ n THEN
            Y[blockStart+j] ← aux[i] ⊕ Y[blockStart+j]
        END IF
    END FOR
END FOR

{ Warning: during labwork, index start at 0 }
```

1.2 Exclusive version

- Same as the inclusive version, except for the end index:

$$\left\{ Y_i = \bigoplus_{j=1}^{i-1} X_j \right\}$$

$\Rightarrow$ In other words, X_i is excluded

- Example: $X = \{1,1,1,1,1,1,1,1\}$
  - Um … what is Y_1 worth?
  - Obligation to specify the null value!
    - $Y_i = \bigoplus_{j=1}^{i-1} X_j = \emptyset$ if $i \leq j$
    - Thus, taking 0 for the addition, $Y = \{0,1,2,3,4,5,6,7\}$

1.2.1 PRAM version

Exclusive SCAN pattern, EREW version

```
FOR each PE i ∈ [1 … n] in parallel: { O(1), Y ← nil element }
    IF i = 1 THEN Y[i] ← ⊥
    ELSE Y[i] ← X[i-1]
    END IF
END FOR
j ← 1
WHILE j < n:
    FOR each PE i ∈ [1 … n − j] in parallel:
        aux[i] ← Y[i]
        aux[i] ← aux[i] ⊕ Y[i+j]
        Y[i+j] ← aux[i]
    END FOR
    j ← j × 2
END FOR
```

1.2.1 PRAM version

Example with $X = \{1,2,3,4,5,6,7,8\}$ and $\oplus$ being multiplication over $\mathbb{Z}$

- Initialisation: $Y = \{1,1,2,3,4,5,6,7\}$ and 1 as nil element
- For $j = 1$ there are 7 EP that compute
$$Y = \{1,1,2,6,12,20,30,42\}$$
- For $j = 2$ there are 6 EP that compute
$$Y = \{1,1,2,6,24,120,360,840\}$$
- For $j = 4$ there are 4 EP that compute
$$Y = \{1,1,2,6,24,120,720,5040\}$$

1.2.2 Version with p processors

- Easy: resume inclusive version?
- Yes, but beware of step 2: you need an inclusive SCAN
  - To understand, let's go back to the partitioning

$$Y_i = \left\{ \bigoplus_{k=0}^{j} X_k \right\} \oplus \left\{ \bigoplus_{k=j+1}^{i-1} X_k \right\}.$$

  - Look at the first sum!
  - So, you must add the missing term:

$$Y_i = \left\{ \bigoplus_{k=0}^{j-1} X_k \right\} \oplus \mathrm{X_j} \oplus \left\{ \bigoplus_{k=j+1}^{i-1} X_k \right\}.$$

  - Respecting the order (associativity $\neq$ commutativity)

1.2.2 Version with p processors

Exclusive SCAN pattern, p-processor version

```
{size of a block to be processed by each EP }
blockSize ← (n + p − 1)/p

{ First step: per block EXCLUSIVE SCAN}
FOR EACH PE i ∈ [1 … p] IN PARALLEL:
    blockStart ← 1 +blockSize× (i − 1)
    Y[blockStart] ← ⊥
    FOR EACH j ∈ [1 …blockSize−1]: { Sequential! }
        Y[blockStart+j] ←
                    Y[blockStart+j-1] ⊕ X[blockStart+j]
    END FOR
END FOR
```

1.2.2 Version with p processors

Exclusive SCAN pattern, p-processor version

```
{ Step 2: INCLUDING SCAN on last block values }
aux[1] ← Y[blockSize] ⊕ X[blockSize] { missing value }
FOR EACH i ∈ [2 ...p−1]: { Sequentially, e.g. on EP 1 }
    aux[j] ← aux[j-1] ⊕ Y[blockSize×j] ⊕ X[blockSize×j]
END FOR

{ Last step: add partial sums }
FOR EACH PE i ∈ [2 ... p] IN PARALLEL:
    blockStart ← 1 +blockSize× (i − 1)
    FOR EACH j ∈ [0 ...blockSize-1]:
        IF blockStart+j ≤ n THEN
            Y[blockStart+j] ← aux[i-1] ⊕ Y[blockStart+j]
        END IF
    END FOR
END FOR
```