Parallel Prefix Sum (Scan)

Klaus Mueller

Computer Science Department

Stony Brook University

Code examples: GPU Gems Chapter 39. Parallel Prefix Sum (Scan) with CUDA
CPU code

```plaintext
out[0] := 0
for k := 1 to n do
    out[k] := in[k-1] + out[k-1]
```

- \(O(n)\)
- we want our parallel code to also be \(O(n)\) but parallel
Algorithm:

1: for $d = 1$ to $\log_2 n$ do
2:    for all $k$ in parallel do
3:       if $k \geq 2^d$ then
4:          $x[k] = x[k - 2^{d-1}] + x[k]$

but this is not safe since not all threads run simultaneously
• $x[k]$ is read twice and the first write might overwrite for the 2\textsuperscript{nd} read
• need double buffering
Double-Buffered Algorithm

```c
__global__ void scan(float *g_odata, float *g_idata, int n)
```

```c
eextern __shared__ float temp[]; // allocated on invocation
int thid = threadIdx.x;
int1 pout = 0, pin = 1;
// Load input into shared memory.
// This is exclusive scan, so shift right by one
// and set first element to 0
temp[pout*n + thid] = (thid > 0) ? g_idata[thid-1] : 0;
__syncthreads();
for (int offset = 1; offset < n; offset *= 2)
{
    pout = 1 - pout; // swap double buffer indices
    pin = 1 - pout;
    if (thid >= offset)
        temp[pout*n+thid] += temp[pin*n+thid - offset];
    else
        temp[pout*n+thid] = temp[pin*n+thid];
    __syncthreads();
}
g_odata[thid] = temp[pout*n+thid]; // write output
```
Better Code

The naive code has $O(n \lg(n))$ work

• actually more work than sequential algorithm (but parallel)
• will not do well when arrays are large
• want to get an algorithm that is parallel but still only does $O(n)$ work

Better code

• has to two phases
• up-sweep and down-sweep
• $O(n)$ work
Better Parallel Code

up-sweep (reduce)

performs $O(n)$ operations ($2 \times (n - 1)$ adds and $n - 1$ swaps)
Be Aware of Bank Conflicts

Addressing Without Padding

Offset = 1: Address (ai) stride is 2, resulting in 2-way bank conflicts

Bank
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

ai
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30

thid
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Offset = 2: Address (ai) stride is 4, resulting in 4-way bank conflicts

Bank
0 1 2 3 4 5 6 7 8 9 10 11

ai
1 5 9 13 17 21 25 29 33 37 41 45

thid
0 1 2 3 4 5 6 7 8 9 10 11

Addressing With Padding

Offset = 1: Padding addresses every 16 elements removes bank conflicts

Bank
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

ai
0 2 4 6 8 10 12 14 17 19 21 23 25 27 29 31

thid
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Offset = 2: Padding addresses every 16 elements removes bank conflicts

Bank
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

ai
1 5 9 13 18 22 26 30 35 39 43 47 52 56 60 64

thid
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Padding Increment: 0 1 2 3
Be Aware of Bank Conflicts

Addressing Without Padding

Offset = 1: Address \( a_i \) stride is 2, resulting in 2-way bank conflicts

```
int ai = offset*(2*thid+1)-1;
int bi = offset*(2*thid+2)-1;
temp[bi] += temp[ai]
```

Bank

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
</tbody>
</table>

\( ai \)  | \( thid \)
---|---
0 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 | 22 | 24 | 26 | 28 | 30

Offset = 2: Address \( a_i \) stride is 4, resulting in 4-way bank conflicts

```
int ai = offset*(2*thid+1)-1;
int bi = offset*(2*thid+2)-1;
ai += ai / NUM_BANKS;
bi += bi / NUM_BANKS;
temp[bi] += temp[ai]
```

Bank

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
</tbody>
</table>

\( ai \)  | \( thid \)
---|---
0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

Padding Increment: 0 1 2 3

Ldng

Addresses every 16 elements removes bank conflicts

<table>
<thead>
<tr>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
</table>

Offset = 2: Padding addresses every 16 elements removes bank conflicts

Bank

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
</tbody>
</table>

\( ai \)  | \( thid \)
---|---
1 | 5 | 9 | 13 | 17 | 21 | 25 | 29 | 33 | 37 | 41 | 45 | 49 | 53 | 57 | 61

Padding Increment: 0 1 2 3
GPU Gems 3:
  • chapter 39. Parallel Prefix Sum (Scan) with CUDA