TBB

TBB parallel for

tbb::parallel_for(partitioner,function_class)

Partitioner-A function that tells each thread what work to do
Function_class - C++ class with an operator object that takes the result of the partition as its only argument

Function class

```
20 class ArraySummer {
 21
 22   int * p_array_a;
 23   int * p_array_b;
 24   int * p_array_sum;
 25
 26 public:
 27   // This empty constructor with an initialization list is used to setup calls to the function
 28   ArraySummer(int * p_a, int * p_b, int * p_sum) : p_array_a(p_a), p_array_b(p_b), p_array_sum(p_sum) { }
 29
 30   /*-------------------------------------------------------+
 31    | Here is the actual body, that will be called in parallel |
 32    | by the TBB runtime. You MUST put this code inside the    |
 33    | class definition, since the compiler will be expanding   |
 34    | and inlining this code as part of the template process.  |
 35    |                                                          |
 36    | The blocked_range<int> is something like a list of       |
 37    | indexes corresponding to each invocation of the function |
 38    +-------------------------------------------------------*/
 39
 40   void operator() ( const blocked_range<int>& r ) const {
 41     for ( int i = r.begin(); i != r.end(); i++ ) { // iterates over the entire chunk
 42       p_array_sum[i] = p_array_a[i] + p_array_b[i];
 43     }
 44   }
 45
 46 };
```

```
tbb::parallel_for(partitioner,
ArraySummer(p_A,p_B,P_SUM_TBB));
```

Function class

tbb::parallel_for(tbb::blocked_range<int>(0,nelements),
ArraySummer(p_A,p_B,P_SUM_TBB));

Are loop is over an integer range from 0, nelements-1

Break that loop in a blocked fashion

Function class

```
tbb::parallel_for(tbb::blocked_range<int>(0,nelements),
                  ArraySummer(p_A,p_B,P_SUM_TBB));
```

```
tbb::parallel_for(tbb::blocked_range<int>(0,nelements),
                  [&], const blocked_range<int>& r){
```

Capture by reference

Function argument is a blocked range object

Function class

```
tbb::parallel_for(tbb::blocked_range<int>(0,nelements),
                  ArraySummer(p_A,p_B,P_SUM_TBB));

tbb::parallel_for(tbb::blocked_range<int>(0,nelements),
                  [&], const blocked_range<int>& r){

for ( int i = r.begin(); i != r.end(); i++ ) { // iterates over the
entire chunk
      p_array_sum[i] = p_array_a[i] + p_array_b[i];
}}
```