Performing Reductions in OpenCL

Here's the Problem We Are Trying to Solve

Like the first ccpp demo program, we are piecewise multiplying two arrays. Unlike the first demo program, we want to then add up all the products and return the sum.

\[ \sum \text{prods} \rightarrow C \]

After the array multiplication, we want each work-group to sum the products within that work-group, then return them to the host in an array for final summing.

To do this, we will not put the products into a large global device array, but into a local array that is local to each work-group.

Here's What You Would Change in your Host Program

```c
size_t numWorkGroups = numItems / LOCAL_SIZE;
float *hA = new float[numItems];
float *hB = new float[NUM_ELEMENTS];
float *hC = new float[numWorkGroups];
size_t cSize = numWorkGroups * sizeof(float);
float *hC = new float[numWorkGroups];

int wgNum = get_group_id(0);  // which work-group number this is in
prods[tNum] = dA[gid] * dB[gid];  // multiply the two arrays together
Thread #0: prods[0] = prods[1];
Thread #1: prods[1] = prods[2];
Thread #2: prods[2] = prods[3];
Thread #3: prods[3] = prods[4];
Thread #4: prods[4] = prods[5];
Thread #5: prods[5] = prods[6];
Thread #6: prods[6] = prods[7];
Thread #7: prods[7] = prods[8];
```

The Arguments to the Kernel

```c
status = clSetKernelArg(kernel, 0, sizeof(cl_mem), &dA);
status = clSetKernelArg(kernel, 1, sizeof(cl_mem), &dB);
status = clSetKernelArg(kernel, 2, LOCAL_SIZE * sizeof(float), NULL);
status = clSetKernelArg(kernel, 3, sizeof(cl_mem), &dC);
```

Reduction Takes Place in a Single Work-Group

If we had 8 work-items in a work-group, we would like the threads in each work-group to execute the following instructions...

...but in a more general way than writing them all out by hand.
Reduction Takes Place Within a Single Work-Group

Each work-item is run by a single thread

A work-group consisting of numItems work-items can be reduced to a sum in Log2(numItems) steps. In this example, numItems=8.

The final sum will end up in prods[0], which will then be copied into dC[wgNum].

Reduction Takes Place in a Single Work-Group

Each work-item is run by a single thread

And, Finally, in your Host Program

Wait(cmdQueue);

double time0 = omp_get_wtime();

status = clEnqueueNDRangeKernel(cmdQueue, kernel, 1, NULL, globalWorkSize, localWorkSize, 0, NULL, NULL);

if (status) PrintCLError(status, "clEnqueueNDRangeKernel failed: ");

Wait(cmdQueue);

double time1 = omp_get_wtime();

status = clEnqueueReadBuffer(cmdQueue, dC, CL_TRUE, 0, numWorkGroups*sizeof(float), hC, 0, NULL, NULL);

if (status) PrintCLError(status, "clEnqueueReadBuffer failed: ");

Wait(cmdQueue);

float sum = 0;

for (int i = 0; i < numWorkGroups/4; i++)
    sum += hC[i];