02-saxpy-basics

July 19, 2021

1 Notebook #2 - Programming Basics with SAXPY

Initial steps: We must import specific environment variables to point to the user's notebook code director and to the Lucata tools.

[1]:
```
import os
os.environ["USER_NOTEBOOK_CODE"]=os.environ["HOME"]+"/pearc-tutorial/code"
os.environ["PATH"]=os.pathsep.join(["/tools/emu/pathfinder-sw/21.06/bin",os.
 ↪environ["PATH"]])
```

This notebook goes along with the Lucata programming basics slides, so please follow along with the slides for a supplemental resource.

1.1 SAXPY with Cilk Spawn

Our first example shows an example of Single-precision AX Plus Y (SAXPY), a basic linear algebra kernel that combines scalar multiplication and vector addition. As shown in the saxpy kernel, the output, `y`, is equal to the sum of the constant `a` multiplied by the elements of a vector `x`.

This first example shows how to implement SAXPY using the most basic Cilk functions, `cilk_spawn` and `cilk_sync`. Note that the Lucata architecture operates on a particular "grain size", which is specified by the number of threads (argument 1).

*For examples of SAXPY in other parallel languages please check out this NVIDIA developer blog on SAXPY.

```
#include <stdio.h>
#include <stdlib.h>
#include <cilk/cilk.h>

void saxpy(long n, float a, float *x, float *y)
{
  for (long i = 0; i < n; i++)
    y[i] += a * x[i];
}

int main(int argc, char **argv)
{
  long nth = atol(argv[1]); // number threads
  long size = atol(argv[2]); // array size
  float aval = atof(argv[3]); // constant
  float *x = malloc(size * sizeof(*x));
  float *y = malloc(size * sizeof(*y));
  for (long i = 0; i < size; i++) {
    x[i] = i; y[i] = 0;
  }

  long grain = size / nth; // elements per thread

  for (long i = 0, j = 0; i < nth; i++, j += grain)
    cilk_spawn saxpy(grain, aval, &x[j], &y[j]);
  cilk_sync;
}
```

We'll test compiling and running this example with a 4 and 8 threads, an array with 32 or 128 elements, and a constant value `a` of 5.0. Then we will check the .cdc and .vsf files to get some high-level statistics on the execution.

[2]:
```
%%bash
emu-cc -o saxpy.mwx saxpy.c
emusim.x -- saxpy.mwx 4 32 5.0
```

```
Start untimed simulation with local date and time= Mon Jul 19 00:34:01 2021

End untimed simulation with local date and time= Mon Jul 19 00:34:01 2021



        SystemC 2.3.3-Accellera --- Jun 22 2021 17:09:43
        Copyright (c) 1996-2018 by all Contributors,
        ALL RIGHTS RESERVED
```

[3]:
```
!more saxpy.cdc
!more saxpy.vsf
```

```
************************************************
Program Name/Arguments:
saxpy.mwx
4
32
5.0
************************************************
Simulator Version: 21.6.23
************************************************
Configuration Details:
Ring Model = Stratix: 3 GC Clusters, 8 MSPs
Number of Nodes=1
Total Memory (in GiB)=64
Logical MSPs per Node=1
```

```
Log2 Memory Size per MSP=36
GC Clusters per Node=3
GCs per Cluster=8
************************************************
************************************************
Simulator wall clock time (seconds): 0
Node ID: Outbound Migrations, Threads Created, Threads Died, Spawn Fails
0: 0, 4, 5, 0

NodeID: num_reads, num_writes, num_rmws
0: 17, 2, 1
```

[4]:
```
%%bash
emusim.x -- saxpy.mwx 8 128 5.0
```

```
Start untimed simulation with local date and time= Mon Jul 19 00:34:01 2021

End untimed simulation with local date and time= Mon Jul 19 00:34:01 2021



        SystemC 2.3.3-Accellera --- Jun 22 2021 17:09:43
        Copyright (c) 1996-2018 by all Contributors,
        ALL RIGHTS RESERVED
```

[5]:
```
!more saxpy.cdc
!more saxpy.vsf
```

```
************************************************
Program Name/Arguments:
saxpy.mwx
8
128
5.0
************************************************
Simulator Version: 21.6.23
************************************************
Configuration Details:
Ring Model = Stratix: 3 GC Clusters, 8 MSPs
Number of Nodes=1
Total Memory (in GiB)=64
Logical MSPs per Node=1
Log2 Memory Size per MSP=36
GC Clusters per Node=3
GCs per Cluster=8
************************************************
************************************************
```

```
Simulator wall clock time (seconds): 0
Node ID: Outbound Migrations, Threads Created, Threads Died, Spawn Fails
0: 0, 8, 9, 0

NodeID: num_reads, num_writes, num_rmws
0: 18, 2, 1
```

1.2 SAXPY with Cilk For

`cilk_for` can be used to launch one thread per loop iteration in a fashion similar to traditional OpenMP pragma-based `omp parallel for` loops. Note here that the programmer must explicitly specify a grainsize to partition up the input array.

```c
#include <stdio.h>
#include <stdlib.h>
#include <cilk/cilk.h>

int main(int argc, char **argv)
{
  long size = atol(argv[1]); // array size
  float aval = atof(argv[2]); // constant

  float *x = malloc(size * sizeof(*x));
  float *y = malloc(size * sizeof(*y));

  for (long i = 0; i < size; i++) {
    x[i] = i; y[i] = 0;
  }

  #pragma cilk grainsize = 8
  cilk_for (long i = 0; i < size; i++) {
    y[i] += aval * x[i];
  }
}
```

[6]:
```
%%bash
emu-cc -o saxpy-for.mwx saxpy-for.c
emusim.x -- saxpy-for.mwx 8 128 5.0
```

```
Start untimed simulation with local date and time= Mon Jul 19 00:34:05 2021

End untimed simulation with local date and time= Mon Jul 19 00:34:05 2021


        SystemC 2.3.3-Accellera --- Jun 22 2021 17:09:43
        Copyright (c) 1996-2018 by all Contributors,
        ALL RIGHTS RESERVED
```

[7]:
```
!more saxpy-for.cdc
!more saxpy-for.vsf
```

```
************************************************
Program Name/Arguments:
saxpy-for.mwx
8
128
5.0
************************************************
Simulator Version: 21.6.23
************************************************
Configuration Details:
Ring Model = Stratix: 3 GC Clusters, 8 MSPs
Number of Nodes=1
Total Memory (in GiB)=64
Logical MSPs per Node=1
Log2 Memory Size per MSP=36
GC Clusters per Node=3
GCs per Cluster=8
************************************************
************************************************
Simulator wall clock time (seconds): 0
Node ID: Outbound Migrations, Threads Created, Threads Died, Spawn Fails
0: 0, 0, 1, 0

NodeID: num_reads, num_writes, num_rmws
0: 18, 8, 1
```

1.3 SAXPY with Distributed Allocation (1D)

In this example, `memoryweb.h` is included for Lucata-specific distributed allocation strategies while `mw_malloc1dlong` is used to distribute data across different nodes within the system.

```
#include <stdlib.h>
#include <cilk/cilk.h>
#include <memoryweb.h>
void saxpy(long n, long a, long *x, long *y)
{
  for (long i = 0; i < n; i++)
    y[i] += a * x[i];
}

int main(int argc, char **argv)
{
  long nth = atol(argv[1]); // number threads
  long size = atol(argv[2]); // array size
  long aval = atol(argv[3]); // constant
  long *x = mw_malloc1dlong(size);
  long *y = mw_malloc1dlong(size);

  for (long i = 0; i < size; i++) {
    x[i] = i; y[i] = 0;
  }

  long grain = size / nth; // elts per thread

  for (long i = 0, j = 0; i < nth; i++, j += grain)
    cilk_spawn saxpy(grain, aval, &x[j], &y[j]);
  cilk_sync;
}
```

[8]:
```
%%bash
emu-cc -o saxpy-1d.mwx saxpy-1d.c
emusim.x -- saxpy-1d.mwx 8 128 5.0
```

```
Start untimed simulation with local date and time= Mon Jul 19 00:34:09 2021

End untimed simulation with local date and time= Mon Jul 19 00:34:09 2021



        SystemC 2.3.3-Accellera --- Jun 22 2021 17:09:43
        Copyright (c) 1996-2018 by all Contributors,
        ALL RIGHTS RESERVED
```

[9]:
```
!more saxpy-1d.cdc
!more saxpy-1d.vsf
```

```
************************************************
Program Name/Arguments:
saxpy-1d.mwx
8
128
5.0
************************************************
Simulator Version: 21.6.23
************************************************
Configuration Details:
Ring Model = Stratix: 3 GC Clusters, 8 MSPs
Number of Nodes=1
Total Memory (in GiB)=64
Logical MSPs per Node=1
Log2 Memory Size per MSP=36
GC Clusters per Node=3
GCs per Cluster=8
```

```
************************************************
************************************************
Simulator wall clock time (seconds): 0
Node ID: Outbound Migrations, Threads Created, Threads Died, Spawn Fails
0: 0, 8, 9, 0

NodeID: num_reads, num_writes, num_rmws
0: 49, 39, 5
```

1.4 SAXPY Distributed Spawn with migrate_hint

The `migrate_hint` allows the programmer to pass a pointer that is then used by the next `cilk_spawn` operation to efficiently jump to a specific part of a distributed array. Here the migration hint is specifying a "directed spawn" to the location where `y[j]` is located. Note that `cilk_spawn_at` provides a similar purpose by combining a spawn and migration hint operation into one call.

```c
#include <stdlib.h>
#include <cilk/cilk.h>
#include <memoryweb.h>

void saxpy(long n, long a, long *x, long *y)
{
  for (long i = 0; i < n; i++)
    y[i] += a * x[i];
}

int main(int argc, char **argv)
{
  long nth = atol(argv[1]); // number threads
  long size = atol(argv[2]); // array size
  long aval = atol(argv[3]); // constant
  long *x = mw_malloc1dlong(size);
  long *y = mw_malloc1dlong(size);

  for (long i = 0; i < size; i++) {
    x[i] = i; y[i] = 0;
  }

  long grain = size / nth; // elts per thread

  for (long i = 0, j = 0; i < nth; i++, j += grain) {
    cilk_migrate_hint(&y[j]);
    cilk_spawn saxpy(grain, aval, &x[j], &y[j]);
  } cilk_sync;
}
```

[10]:
```
%%bash
emu-cc -o saxpy-1d-hint.mwx saxpy-1d-hint.c
emusim.x -- saxpy-1d-hint.mwx 8 128 5.0
```

```
Start untimed simulation with local date and time= Mon Jul 19 00:34:13 2021

End untimed simulation with local date and time= Mon Jul 19 00:34:13 2021



        SystemC 2.3.3-Accellera --- Jun 22 2021 17:09:43
        Copyright (c) 1996-2018 by all Contributors,
        ALL RIGHTS RESERVED
```

[11]:
```
!more saxpy-1d-hint.cdc
!more saxpy-1d-hint.vsf
```

```
************************************************
Program Name/Arguments:
saxpy-1d-hint.mwx
8
128
5.0
************************************************
Simulator Version: 21.6.23
************************************************
Configuration Details:
Ring Model = Stratix: 3 GC Clusters, 8 MSPs
Number of Nodes=1
Total Memory (in GiB)=64
Logical MSPs per Node=1
Log2 Memory Size per MSP=36
GC Clusters per Node=3
GCs per Cluster=8
************************************************
************************************************
Simulator wall clock time (seconds): 0
Node ID: Outbound Migrations, Threads Created, Threads Died, Spawn Fails
0: 0, 8, 9, 0

NodeID: num_reads, num_writes, num_rmws
0: 49, 39, 5
```

1.5 SAXPY with 2D Distributed Allocation

This example shows the usage of `cilk_spawn_at` and 2D block allocation of data across the Lucata nodes. In this case, the number of threads matches the number of blocks and the work done by each thread is the block size.

```c
#include <stdlib.h>
#include <cilk/cilk.h>
#include <memoryweb.h>

void saxpy(long n, float a, float *x, float *y)
{
  for (long i = 0; i < n; i++)
    y[i] += a * x[i];
}

int main(int argc, char **argv)
{
  long num = atol(argv[1]); // number blocks
  long size = atol(argv[2]); // block size
  float aval = atof(argv[3]); // constant
  float **x = mw_malloc2d(num, size * sizeof(*x));
  float **y = mw_malloc2d(num, size * sizeof(*y));

  for (long j = 0; j < num; j++) {
    for (long i = 0; i < size; i++) {
      x[j][i] = j * size + i; y[j][i] = 0;
    }
  }

  for (long i = 0; i < num; i++) {
    cilk_spawn_at (y[i]) saxpy(size, aval, x[i], y[i]);
  }
  cilk_sync;
}
```

[12]:
```
%%bash
emu-cc -o saxpy-2d-spawn-at.mwx saxpy-2d-spawn-at.c
emusim.x -- saxpy-2d-spawn-at.mwx 8 128 5.0
```

```
Start untimed simulation with local date and time= Mon Jul 19 00:34:16 2021

End untimed simulation with local date and time= Mon Jul 19 00:34:16 2021



        SystemC 2.3.3-Accellera --- Jun 22 2021 17:09:43
        Copyright (c) 1996-2018 by all Contributors,
        ALL RIGHTS RESERVED
```

[13]:
```
!more saxpy-2d-spawn-at.cdc
!more saxpy-2d-spawn-at.vsf
```

```
************************************************
Program Name/Arguments:
saxpy-2d-spawn-at.mwx
8
128
5.0
************************************************
Simulator Version: 21.6.23
************************************************
Configuration Details:
Ring Model = Stratix: 3 GC Clusters, 8 MSPs
Number of Nodes=1
Total Memory (in GiB)=64
Logical MSPs per Node=1
Log2 Memory Size per MSP=36
GC Clusters per Node=3
GCs per Cluster=8
************************************************
************************************************
Simulator wall clock time (seconds): 0
Node ID: Outbound Migrations, Threads Created, Threads Died, Spawn Fails
0: 0, 8, 9, 0

NodeID: num_reads, num_writes, num_rmws
0: 348, 410, 3
```

1.6 SAXPY with Local Allocation

This example shows a variation of the previous 2D code with a local allocation for the output. You will notice that the local allocation for the output (as opposed to 2D allocation) results in more migrations overall.

```c
#include <stdlib.h>
#include <cilk/cilk.h>
#include <memoryweb.h>

void saxpy4(long n, float a, float *x, float *y)
{
  for (long i = 0; i < n; i++)
    y[i] += a * x[i];
}

int main(int argc, char **argv)
{
  long num = atol(argv[1]); // number blocks
  long size = atol(argv[2]); // block size
  float aval = atof(argv[3]); // constant
  float **x = mw_malloc2d(num, size * sizeof(*x));
  float *y = mw_localmalloc(num * size * sizeof(*y), x[0]);

  for (long j = 0; j < num; j++) {
    for (long i = 0; i < size; i++) {
      x[j][i] = j * size + i; y[j * size + i] = 0;
    }
  }

  for (long i = 0; i < num; i++) {
    cilk_spawn_at (x[i]) saxpy4(size, aval, x[i], &y[i * size]);
  }
  cilk_sync;
}
```

[14]:
```
%%bash
emu-cc -o saxpy-local-spawn-at.mwx saxpy-local-spawn-at.c
emusim.x -- saxpy-local-spawn-at.mwx 8 128 5.0
```

```
Start untimed simulation with local date and time= Mon Jul 19 00:34:18 2021

End untimed simulation with local date and time= Mon Jul 19 00:34:18 2021



        SystemC 2.3.3-Accellera --- Jun 22 2021 17:09:43
        Copyright (c) 1996-2018 by all Contributors,
        ALL RIGHTS RESERVED
```

[15]:
```
!more saxpy-local-spawn-at.cdc
!more saxpy-local-spawn-at.vsf
```

```
************************************************
Program Name/Arguments:
saxpy-local-spawn-at.mwx
8
128
5.0
************************************************
Simulator Version: 21.6.23
************************************************
Configuration Details:
Ring Model = Stratix: 3 GC Clusters, 8 MSPs
Number of Nodes=1
Total Memory (in GiB)=64
Logical MSPs per Node=1
Log2 Memory Size per MSP=36
GC Clusters per Node=3
GCs per Cluster=8
```

```
************************************************
************************************************
Simulator wall clock time (seconds): 0
Node ID: Outbound Migrations, Threads Created, Threads Died, Spawn Fails
0: 0, 8, 9, 0

NodeID: num_reads, num_writes, num_rmws
0: 380, 488, 11
```

1.7 SAXPY with Replicated Data Structures

Finally, we look at using replication to create copies of the constant variable, `a` across all the nodes. This prevents migrations to access this common variable if it were located only on a single node. Note that replication can be a powerful tool for optimized allocation but it should be used primarily with small data structures and variables that are read-only (for coherency reasons).

```c
#include <stdlib.h>
#include <cilk/cilk.h>
#include <memoryweb.h>

long aval;
replicated long a;

void saxpy(long n, long a, long *x, long *y) {
  for (long i = 0; i < n; i++) y[i] += a * x[i];
}

int main(int argc, char **argv)
{
  long nth = atol(argv[1]); // number threads
  long size = atol(argv[2]); // array size
  aval = atol(argv[3]); // constant
  mw_replicated_init(&a, aval);
  long *x = mw_malloc1dlong(size);
  long *y = mw_malloc1dlong(size);

  for (long i = 0; i < size; i++) {
    x[i] = i; y[i] = 0;
  }

  long grain = size / nth; // elts per thread

  for (long i = 0, j = 0; i < nth; i++, j += grain)
    cilk_spawn saxpy(grain, &x[j], &y[j]);
  cilk_sync;
}
```

[16]:
```
%%bash
emu-cc -o saxpy-1d-replicated.mwx saxpy-1d-replicated.c
emusim.x -- saxpy-1d-replicated.mwx 8 128 5
```

```
Start untimed simulation with local date and time= Mon Jul 19 00:34:22 2021

End untimed simulation with local date and time= Mon Jul 19 00:34:22 2021



        SystemC 2.3.3-Accellera --- Jun 22 2021 17:09:43
        Copyright (c) 1996-2018 by all Contributors,
        ALL RIGHTS RESERVED
```

[17]:
```
!more saxpy-1d-replicated.cdc
!more saxpy-1d-replicated.vsf
```

```
************************************************
Program Name/Arguments:
saxpy-1d-replicated.mwx
8
128
5
************************************************
Simulator Version: 21.6.23
************************************************
Configuration Details:
Ring Model = Stratix: 3 GC Clusters, 8 MSPs
Number of Nodes=1
Total Memory (in GiB)=64
Logical MSPs per Node=1
Log2 Memory Size per MSP=36
GC Clusters per Node=3
GCs per Cluster=8
************************************************
************************************************
Simulator wall clock time (seconds): 0
Node ID: Outbound Migrations, Threads Created, Threads Died, Spawn Fails
0: 0, 8, 9, 0

NodeID: num_reads, num_writes, num_rmws
0: 49, 39, 5
```

1.7.1 Postcript

Here we have investigated several different strategies for spawning threads and allocatin data with the Pathfinder's distributed layout. Note that the simulations we ran did not have fully accurate timing, but the .cdc and .vsf files do give some indication as to which strategies are more efficient for a particular input size and number of threads.

Once we've finished our testing, we can clean up some of the logfiles that we used for this example with `make clean`. Uncomment the following line to clean this directory.

[18]:
```
#!make clean
```