Enabling GPU support for the COMPSs-Mobile framework

Francesc Lordan, Rosa M Badia and Wen-Mei Hwu

Nov 13, 2017

4th Workshop on Accelerator Programming Using Directives
COMPSs-Mobile infrastructure

Collects data from sensors

Offloads computation to Cloud

PAN

LAN

WAN

Cloud

Database
Programming Model: COMPSs

- Programming
  - Sequential code
  - Standard programming languages with no APIs
  - Infrastructure agnostic
  - Task-based
    - An annotated interface contains the declaration of those methods (CE) selected to become tasks with information about the accesses to data

- Execution managed by a runtime system
  - Replaces CE invocations by asynchronous tasks
  - Monitors data-dependencies among tasks
  - Orchestrates the execution on the underlying infrastructure
    - Transfers data values
    - Submits task executions
Programming Model Example

Matmul.java

```java
public class Matmul {
    public static void main(String[] args) {
        int[][] A;
        int[][] B
        int[][] C;
        ...
        C = multiply(A, B);
        ...
    }
    public static int[][] multiply(int[][] A, int[][] B) {
        // Matrix multiplication code
        // C = AB
        ...
        return C;
    }
}
```

CEI.java

```java
public interface CEI {
    @Method(declaringClass="es.bsc.compss.matmul.Matmul")
    int[][] multiply(
        @Parameter(direction = IN) int[][] A,
        @Parameter(direction = IN) int[][] B
    );
}
```
Enabling GPU support on COMPSs-Mobile
Programming Model Extension

• Add the OpenCL kernel codes in application resource files

• Indicate on the CEI the CE implementation as an OpenCL kernel using the @OpenCL annotation indicating:
  • Resource containing the code implementation (kernel)
  • Number of work-items (globalWorkSize)
  • Work-items ID offset (offset)
  • Work groups size (localWorkSize)
  • Result Size (result)
Programming Model Extension Example

Matmul.java

```java
public class Matmul {
    public static void main(String[] args) {
        int[][] A;
        int[][] B
        int[][] C;
        ...
        C = multiply(A, B);
        ...
    }
    public static int[][] multiply (int[][] A, int[][] B) {
        // Matrix multiplication code
        // C = AB
        ...
        return C;
    }
}
```

Matmul.cl

```c
__kernel void multiply (  
  __global const int *a,
  __global const int *b,
  __global int *c) {
    // Matrix multiplication code
    // C = AB
    ...
}
```

CEI.java

```java
public interface CEI {  
    @OpenCL(kernel="matmul.cl", globalWorkSize="par0.x,par1.y", resultSize="par0.x,par1.y")
    @Method(declaringClass="es.bsc.compss.matmul.Matmul")
    int[][] multiply (  
        @Parameter(direction = IN)  
        int[][] A,
        @Parameter(direction = IN)  
        int[][] B
    );
}
```
Kernel Execution Lifecycle

[0 .- Fetch input values from a remote node]

1 .- Create a cl_buffer for each parameter and the result
   
   ```c
   cl_mem par0Dev = clCreateBuffer(...);
   ```

2 .- Fill the cl_buffers corresponding to the input values
   
   ```c
   clEnqueueWriteBuffer(..., par0Dev, ...);
   ```

3 .- Run the kernel
   
   ```c
   cl_kernel task1 = clCreateKernel (...);
   clSetKernelArgument(task1, 0, par0Dev);
   clNDRangeKernel(..., task1, ...)
   ```

4 .- Bring back the modified/result values
   
   ```c
   clEnqueueReadBuffer(..., par0Dev, ...)
   ```

5 .- Notify the existence of the created values to enable data-dependent tasks executions.
Kernel Execution Lifecycle Management

• Management is twofold:
  • The runtime library directly controls the creation of OpenCL structures and Host memory management
    • Fetching data from remote workers
    • Allocating the necessary device memory space to host the buffers
    • Creating the Kernel object and setting up its arguments
    • Publishing the value existence to release data dependencies

• It delegates on the out-of-order mode of OpenCL the execution of OpenCL commands:
  • clEnqueueWriteBuffer
  • clNDRangeKernelExecution
  • clEnqueueReadBuffer
Optimizations

- Keep values on the device memory to reuse them
- Keep track of the events generating the values

\[ C = \text{matmul}(A,B) \]
\[ E = \text{matmul}(C,D) \]
Executing Platform Selection

• Each task is profiled to measure the execution time and energy consumption

• Using historical values of previous executions the runtime forecasts:
  • The execution time on the resource
  • The energy consumption of the execution
  • The waiting time before the execution start

• According to a heuristic defined by the application user and based on the forecasts, the runtime picks one platform
Performance Evaluation
Digits Recognition

- CNN recognizing 512 handwritten digits
- Sequence of 8 steps (tasks) processing all 512 images
- GPU speeds up task calculation by 6.5x
- Optimizations reduce the 1,531 ms overhead to 5 ms
- Energy consumption falls from 30 J to 8J
Canny Edge detection

• Detects the edges in 30 frames of a video
• Each frame goes through 4 stages (4 tasks)

• Developers normally try to balance the load according to:
  • Data Partitioning: assign the processing of a whole frame to a resource
  • Task Partitioning: run the two first tasks of every frame on the GPU and the last two on the CPU

• 2 dynamic policies
  • DynPerf: minimizes the execution time
  • DynEn: slows down 1 ms the execution if that saves 5 mJ
Canny Edge detection

- GPU speeds up 12x the task computation
- GPU reduces the energy consumption from 9.89 J to 1.34 J
- Data Partitioning achieves lower execution times but consumes more energy than Task Partitioning
- DynPerf achieves the lowest execution time (375 ms)
- DynEn gets lower energy consumptions while its performance compares to Task Partitioning
Conclusions

• The programming model allows developers to code parallel applications that run on heterogeneous systems being unaware of the parallelization and collaboration issues.

• Using GPUs reduces drastically the timespan and the energy consumption of the application.

• Dynamic policies provide applications with
  • Portability
  • Flexibility

• Code is available at
  • http://compssdev.bsc.es/gitlab/flordan/WACCPD
  • http://compssdev.bsc.es/gitlab/flordan/COMPSs-Mobile
Future Work

• Enable the usage of the GPUs to compute tasks on remote nodes

• Improve the forecasts for the energy and end time

• Allow the dynamic policies to change an initial decision on which resources the task will run
Thank you!

francesc.lordan@bsc.es