An exact algorithm for energy-efficient acceleration of task trees on CPU/GPU architectures.

Mark Silberstein
marks@cs.technion.ac.il
Electrical Engineering Department
Technion – Israel Institute of Technology

CCIT Report #783
January 2011

ABSTRACT

We consider the problem of *energy-efficient acceleration* of applications comprising multiple interdependent tasks forming a dependency tree, on a hypothetical CPU/GPU system where both a CPU and a GPU can be powered off when idle. Each task in the tree can be invoked on both a GPU or a CPU, but the performance may vary: some run faster on a GPU, others prefer a CPU, making the choice of the lowest-energy processor input dependent. Furthermore, greedily minimizing the energy consumption for each task is sub-optimal because of the additional energy required for the communication between the tasks executed on different processors.

We propose an efficient algorithm, which accounts for the energy consumption of a CPU and a GPU for each task, as well as for *the communication costs of data transfers between them*, and constructs an *optimal* acceleration schedule with *provably minimal* total consumed energy.

We evaluate the algorithm in the context of a real application having task dependency tree structure, and show up to 2.5-fold improvement in the expected energy consumption versus CPU only or GPU only schedule, and up to 50% improvement over the communication unaware schedule on real inputs. We also show another application of this algorithm which allows to achieve up to a 2-fold speedup in real CPU/GPU systems.

1. INTRODUCTION

Energy efficiency has become one of the central goals in contemporary hardware designs, in particular for embedded processors and SoCs, often at the expense of the peak performance. To this end, many systems already implement software-controlled dynamic power management, sometimes allowing to completely shut down idle components, quickly turning them back on when necessary. For example, NVIDIA Optimus technology [1] enables dynamic switching between power-hungry high-performance discreet GPU to the integrated low-power GPU to extend the battery life.

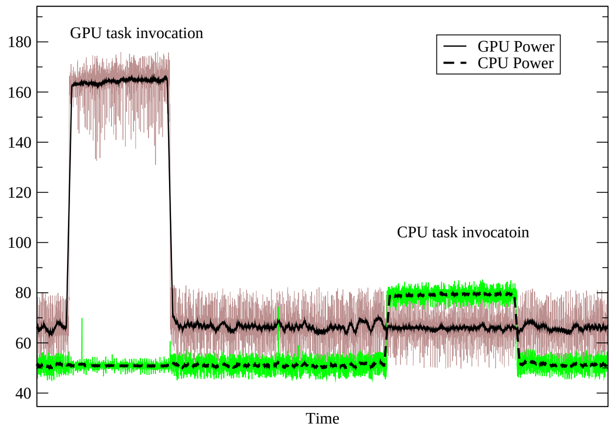

Figure 1: An example of power consumed by CPU and GPU when running the same task.

We believe that similar capabilities will be also available in the GP-GPU world, allowing for almost complete power off and zero-overhead power up of both a CPU and a GPU to enable prolonged battery life in mobile platforms.

However, without appropriate software support, energy-efficient hardware by itself will not allow energy-efficient execution. Consider an application which comprises two independent tasks. Both can be executed on a CPU or a GPU, but their performance vary: while the first runs faster on a GPU, the second does not benefit from the massive parallelism. In fact, it is easy to minimize makespan of the application by executing tasks where they perform best.

In order to achieve energy-efficient execution we also need to know the actual power dissipated by each processor. To illustrate the difference between the CPU and GPU power consumption today, Figure 1 shows the power consumed when executing the same task on all 4 cores of AMD Phenom 9500 Quad-core processor and a GTX285 NVIDIA GPU. While the GPU requires slightly less time, the CPU consumes about 50% less energy in total. Predicting GPU power consumption by statistical methods or via modeling has been investigated before [7], and can be used by the application scheduler to assign the tasks accordingly.