Parallel Sorting Algorithms on Hybrid Heterogenous Servers

Table of Contents

Abstract

asdasdasd

Intro

Hetrogeneous servers featuring multicore CPU processors hosting multiple accelerators (GPUs and FPGAs) dominate the computing landscape due to their superior performance and energy efficiency. We use the term hybrid parallel program to signify a program that divides the workload heterogeneously between the multicore CPU processors and accelerators of a server to provide maximum resource utilization.

Sorting algorithms place the data elements in a list of certain order. Sorting is foundational to modern data and information services. Because of their importance, efficient sorting algorithms have been the subject of many computer science research efforts. Even with these efficient algorithms, sorting large data lists is still time consuming and can benefit from parallel execution. Parallelizing efficient sorting algorithms is challenging and requires efficient design.

CPUs

asdasdasd

GPUs

The graphics processing unit (GPU), first invented by NVIDIA in 1999, is the most common parallel processor to date. Unlike CPUs the GPU contains a much larger amount of smaller cores which are designed to run in parallel. It was originally designed for use in video game graphics but work to use GPUs for non graphical applications have been underway since 2003.

Today GPUs are one of the most important pieces of technology underpinning the current boom in AI and Data Centers and NVIDIA are ranked as the highest valued company by market capitalization.

Fermi and Tesla GPUs

To address the problems of GPGPUS NVIDIA introduced the Ge80 unified graphics and computer architecture and CUDA which is a software and hardware architecture designed to make GPUs programmable with a variety of high level programming languages. This paved the way for general use GPUs as we know them today where developers are able to write C programs with CUDA extensions to perform general purpose parallel processes.[1]

The complexity of the Fermi architecture is managed by a multi level programming model that allows software developers to focus on algorithm design rather than the details of how to map the algorithm to the hardware.[2]

Cuda Programming Model

The CUDA programming model assumes a heterogeneous computing system. The CPU and the memory directly connected to it are called the host and host memory respectively. The GPU and the memory directly connected to it are called the device and device memory respectively.

The code an application executes on the GPU is referred to as device code and a function that is invoked for launching the GPU is called a kernel.

When an application launches a kernel, it does so with many threads, often millions of threads. These threads are organized into blocks. A block of threads is referred to as a thread block. Thread blocks are organized into a grid. All threads in a grid have the same size and dimensions.

Heterogeneous systems have multiple physical memories where data can be stored. The host CPU has attached DRAM, and every every GPU in a system has its own attached DRAM. Performance is best when data is resident in the memory of the processor accessing it.

Shared memory is located on chip and as a result has a much higher bandwidth and lower latency than local and global memory. Shared memory enables cooperation between threads in a block, when multiple threads in a block use the same data from global memory, shared memory can also be used to access data the data from global memory only once.

Shared memory is only accessible by threads within the same block, this can become a problem when sorting large inputs that are greater than the size of a block. Currently Tesla GPUs only support blocks sizes up to 1024.

Sorting Algorithms

There are many things which make parallelizing sorting algorithms complected. One being that sorting algorithms often make use of recursion. Recursion is possible in CUDA using dynamic parallelism, however can introduce problems such as divergence.

An example of a sorting algorithm which does not use recursion is odd-even sort. This algorithm was orginally presented and shown to be efficent on parallel processors by Habermann in 1972.

```
while(){
  isSorted=true;
  for(){
    if(){
      swap();
      isSorted = false;
}
}
}
```

Parallel Sorting

asdasd

Parallel Merging

asdasdasd

Parallel Programming Models

asdasdasd

OpenMP

OpenACC

OpenH

[1]https://www.nvidia.com/content/pdf/fermi_white_papers/p.glaskowsky_nvidia's_fermi-the_first_complete_gpu_architecture.pdf

[2]https://www.nvidia.com/content/pdf/fermi_white_papers/p.glaskowsky_nvidia's_fermi-the_first_complete_gpu_architecture.pdf

https://www.cs.cmu.edu/afs/cs/academic/class/15869-f11/www/readings/lindholm08_tesla.pdf

https://arxiv.org/pdf/1511.03404