CS267 Assignment 0: Describe a Parallel Application

Anurag Khandelwal

January 29, 2016

1 About Me

I’m a third year graduate student at the AMPLab (EECS). My current research interests lie at the intersection of algorithms and big data systems. At the AMPLab, I’ve been working on designing data structures and algorithms that enable queries directly on compressed data, and building practical systems around them. My motivation in taking this class was to gain insights into both the low level system aspects of parallelization in modern architectures, as well as algorithmic approaches in achieving parallelism.

2 Parallelism for high throughput in Data Stores

The application I’m most excited about for this course is the careful use of parallelism to achieve high throughput in Data Stores. Modern data stores (e.g., key-value stores, document stores) typically hold huge collections of records, and are designed to provide extremely low latency and high throughput for access to these records. One of the key approaches in these data stores has been to aggressively exploit parallelism in modern CPUs — the operations are distributed across different cores, bringing in the age old challenges in designing parallel systems such as load balancing, data locality, data consistency, efficient parallelization of dependent operations, etc.

However, there are a few recent trends that make these problems particularly interesting:

- With the ubiquitous growth in the volume of data being generated and stored, it is critical to be able to serve these operations in main memory to meet the low latency and high throughput demands. While compression one way to push more data in memory, it imposes added burden on the CPUs due to increased computation (e.g., for compression/decompression). In such a setting, being able to effectively parallelize such computations becomes all the more interesting.

- The rate at which the data is being generated and needs to be stored is also unprecedented; data insertions and updates must be handled with the same (if not better) efficacy as data reads. In particular, read and write operations often occur concurrently on the same data, which presents an interesting challenge in terms of maintaining consistency while exploiting parallelism to scale to much higher read and write rates.