1 | What are distributed algorithms?

(https://stanford.edu/~rezab/dao/) • Algorithms designed to run on multi-core processors by splitting up into subroutines that simultaneously run on separate processors, saving time. • The general field of algorithm hardware optimization includes design algorithms to be run on GPUs and CPUs or even TPUs. – TPU stands for Tensor Processing Unit and is a chip developed by Google to aid systems running TensorFlow

2 | Simple example of distributed algorithm is addition

• Normally one steps down the list and adds to running total • Instead, with multiple cores, delegate pairs of numbers to each core and compute the sum, reducing the list to n/2. Recursively repeat this until the list is only one element to calculate the sum in O(log 2) time.

3 | Modelling

• Normally the model is of the Random Access Machine, where a processor is connected to memory and every operation takes a constant time step. • Not as straightforward for distributed algorithms

4 | Multiprocessor Models

• Random Access Machine model but with additional processors. • Three seperate submodels with different methods of accessing memory. – Local memory machine models each processor as having its own memory – Modular memory machine models each processor as being routed to m processors – PRAM models each processor as sharing memory with each other processor — Benefits include accessing memory in single step — Not widely believed that this model (in how it can access memory in single step) is realistic

5 | Techniques

5.1 | Divide and Conquer

• Divide and Conquer (or splitting the problem into easier subproblems) is useful in a sequential context, and extra useful in a parallel context. – Subproblems are usually independent of one another, and therefore can benefit from being calculated in parallel. • Merge sort is one good example, as its usual time complexity is O(n log n)

6 | In-class Lecture

• Around for ~25 years but more relevant in last 10 because many cores has become more of a reality. • Distribute tasks accross core

6.1 | Area 1: Distributing Algorithms

• Atomicity – Either all operations are run or none of them so as to prevent error • Leader election – Sometimes a core needs to be a ”leader” • Consensus – Processes must agreee

6.2 | Area 2: Designing Algorithms to be Distributed

6.2.1 | Map-Reduce

• Map: (inkey , invalue) -> list(outkey, intermediate value) – Write a function that gives each document to a different core, and count number of occurences in document. • Reduce: Collect responses and combine • Benefit is just structuring the problem in the framework