Name: Keaton Whitehead

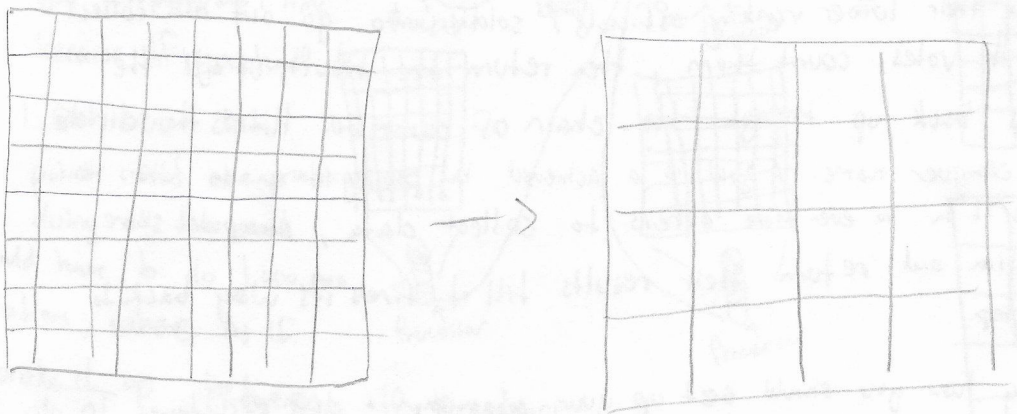
ID: 104 6068 391

CSCI 3104, Algorithms **Profs. Grochow & Layer**
Explain-It-Back 3 **Spring 2019, CU-Boulder**

You colleagues in the meteorology department need your help scaling their weather prediction capabilities to a wider area. They explain that weather prediction involves dividing the forecast region into a grid, computing a short-term prediction for each cell in the grid based on conditions from weather sensors, then combining those results for region-wide forecast. Right now they start in the top left cell of the grid, then proceed right cell by cell, then row by row, until a prediction has been made in every cell. They then go back over the grid and consolidate the forecasts in a 2x2 cell square starting in the top left and moving right across the grid two cells at a time (i.e., the squares do not overlap). This is then repeated several more times considering ever larger squares, until they have one final prediction. Their department has invested in a new large multi-processor computer to help them make predictions over larger areas, but they do not know how to utilize all of these new processors since the computation of individual cells, or groups of cells, cannot be efficiently computed by multiple CPUs. Help them understand how a new divide-and-conquer strategy may help them better leverage the power of their new hardware without adding extra work to the overall solution.

NOTE: For simplicity, assume that there is no direct processor-to-processor communication. A CPU gathers data stored in a cell or group of cells, runs a program on that data, and writes the result back to the same cell/cells which can then be read by some other processor. There are many real-world strategies for coordinating work among many processors. The one we assume here is similar to what is used in the map-reduce/Hadoop framework.

- How to divide up the work to do it in parallel
- Divide + Conquer



Dear Friends,

I am here to help you. It is really good that you were able to get the new processors because this will improve your run time for very large sets of data. All you have to do is implimen the divide and conqour structure into your algorithm. What is the divide and conqour method you might ask?

Well, the divide and conqour method is a data structure design to make your algorithms and processors as efficient as possible. How does it do this? In order to understand how it works I have to define what recursion is.

- Recursion is a function that may or may not have given inputs, but it will call itself either directly or indirectly. There is a certain point or limit that the recursive function will reach where it will end the self calling cycle and return your calculated result. Each time the recursive function calls itself however it is able split up a task into smaller peices in such a way that multiple chunks of data can be processed very quickly because of these constant break downs and iteration through the loops. If this still doesnt make sense think of A recursive function as the Roman Empire. The function has a goal so we can say Rome has a goal. Let it be to collect votes of an election from citizens of rome to elect an new official. So the leader of Rome orders this to happen (obviously he himself will not go about every house hold collecting them all so he deligates). He orders his messengers to go out to the different cities of the Roman Empire to instruct the Military generals of each city to order their lower ranking officials / soldiers to go out and collect all the votes, count them, then return the results of all the votes back up through the chain of command. Hints the divide and conquer name. A goal is a achieved by passing it on to others to pass it on to even more others to collect data, compute some function and return their results till it finds its way back to the top.

This is essentially how you should set up your algorithms + data structures. To do as much deligation till you actually get to the point where you have to actually compute the values through your Function, i.e. when you want to actually collect, compute, or store the temperature readings at each location.

Name: Keaton Whitehead

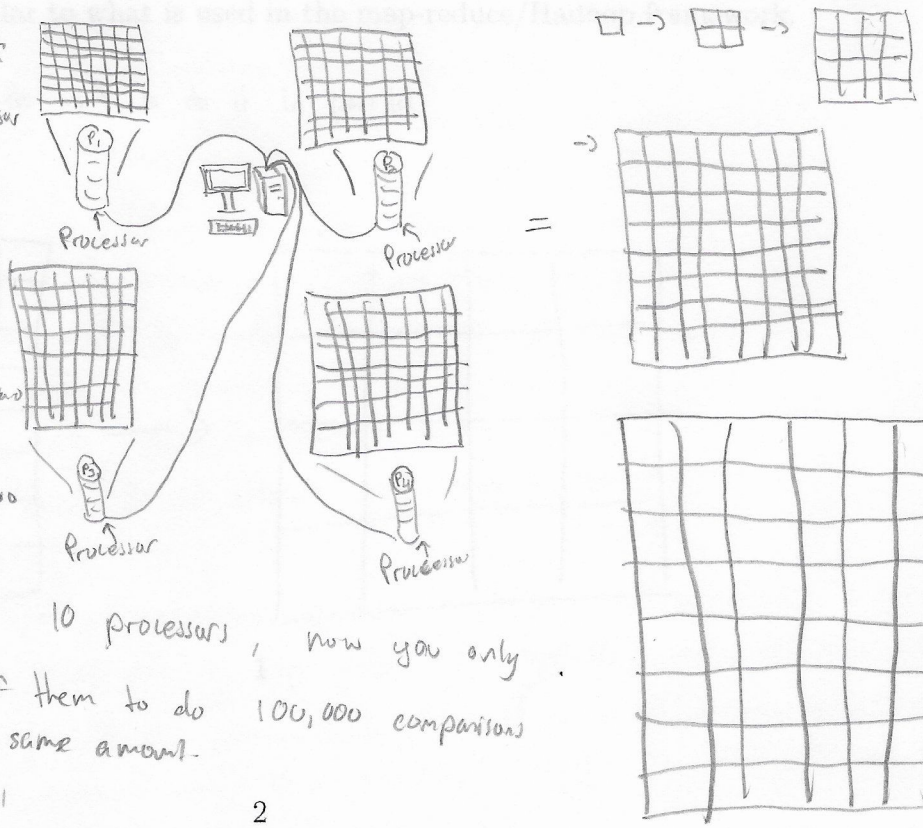
ID: 104 668 391

CSCI 3104, Algorithms
Explain-It-Back 3

Profs. Grochow & Layer
Spring 2019, CU-Boulder

This will be the most nested recursive loop. The next layer of the recursive loop will be the loop that takes the individual data and computes the 2×2 squares of data and that gets passed on upward through as many recursive parent functions that are needed until you finally compute the final average value.

Now since you have multiple processors you can have each one do all equivalent amount of work to calculate the final average but you only want that processor to get a fraction of all of the data. So say you only had 4 processors you would split all the data into four quadrents and assign each quadrent to a processor and it was on responsible for computing that 1/4 of total data available. Here is a diagram to help picture what I mean.



Deligating your work evenly to each processor will drastically improve your computational time because if you had 1,000,000 data points and one processor doing all the work it would have to do 1,000,000 calculations, where as if you broke it up between 10 processors, now you only have to wait for one of them to do 100,000 comparisons since they're is to doing the same amount.

I hope this finds you well!

Keaton Whitehead