Explaining parallel computing at Middle School

What Grown-Ups Do Not Understand About Parallel Programming: The Peanut Butter and Jelly Story

Computers are machines engineered to operate by themselves ('automatically') based on computer programs. These programs work on data, like numbers and text. For example, given a text (Word) file, a program can find all occurrences of the word 'jelly' in it. We can understand computer programs by thinking about things we can touch with our hands.

Let's think about the following problem. You have two jars. Due to a mistake, one is imprinted with the words 'peanut butter' but has jelly and the other is imprinted with the word 'jelly' but has peanut butter. You need to swap the peanut butter and jelly (move the peanut butter to jar that says 'peanut butter' and the jelly to the jar that says 'jelly') as fast as possible. We can solve this problem in 3 steps by using one additional (empty) jar. Step 1: Move the peanut butter to the new jar. Step 2: move the jelly to the jar that says 'jelly'. Step 3: move the peanut butter to the jar that says 'peanut butter'. Each step takes one time unit and the whole program takes 3 time units.

Next, think about 1,000 peanut butter jars arranged in a row and 1,000 jelly jars arranged in a second row, where the first peanut butter jar is next to the first jelly jar and so on. Due to a mistake, all the peanut butter jars are imprinted with the word 'jelly' and all the jelly jars are imprinted with the words 'peanut butter'. You need to swap the peanut butter in the first peanut butter jar with jelly in the first jelly jar. You need to do the same for the second peanut butter jar and the second jelly jar and so on. A basic rule of parallel programming allows you to tell the computer that it can do any number of operations ('tasks') at same time ('concurrently'), if these operations do not interfere with one another. The question now is: how to solve the new swapping problem in the shortest time.

The solution is a 'parallel' program that uses, **at the same time**, 1,000 'copies' of the above 3-step program. We simply use 1,000 additional (empty) jars, one for each swap. This 'parallel program' still needs a total of only 3 time units. The reason is that each of the 3-step copies of the program works at the same time. The parallel program uses 1,000 new jars instead of one new jar, as before. The parallel program has the machine do 1,000 operations per time unit, for a total of 3,000 operations.

Traditional computers solve the 1000-jar-swap problem very differently. Using only one new jar, their program first swaps contents of the first peanut butter jar with the first jelly jar. Second, their program uses the same new jar to swap the peanut butter and the jelly in the second pair of jars. Completing the 1000 swaps takes a total of 3,000 time units instead of 3, so the parallel computer will work 1,000 times faster! This example demonstrates the power of parallel computers. They can run much faster. The total number of operations will still be 3,000; however the parallel program uses 1,000 extra jars instead one extra jar in the traditional program.

John von Neumann faced many constraints when he helped to design the traditional computer. Because of these constraints, programs written for these computers seem to be 'fitting a thread through the eye of a needle'. This means that the one-new-jar solution, which uses the same single jar for every three-

step swap and performs operations one at a time, is preferred for this computer. However, technology has advanced so much that the hardware cost of using many more new jars now makes more sense.

The problem that grown-up programmers have with this is that they have become so used to thinking about programs fitting through the eye of a needle that the slow, 3,000-time-unit solution that was good for old computers looks easy to them, while the quick 3-time-unit solution looks too difficult.