Sequential, Parallel, and Distributed Computing

Sequential Computing

Each of these different styles of computing dictates how a computer goes about finishing its tasks. We'll take a look at sequential computing first.

With this style of computing, each task is taken one at a time, in the order that they were fed to the processor.

Sequential Computing

Let's say our computer has 4 tasks to perform.

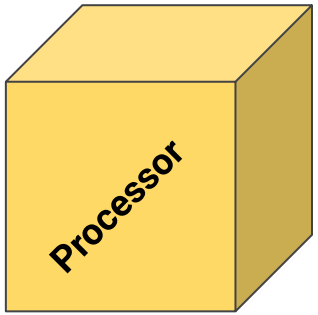
Sequential Computing

Let's say our computer has 4 tasks to perform.

Sequential Computing

Let's say our computer has 4 tasks to perform.

Task 1 | Task 2 | Task 3 | Task 4



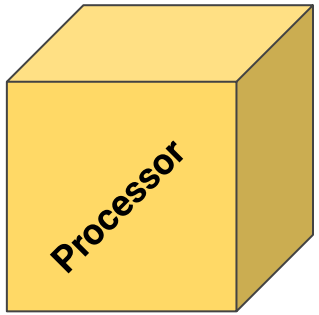
Result

Result

Sequential Computing

If we use sequential computing, we're going to accomplish 1 task at a time!

Task 1 | Task 2 | Task 3 | Task 4

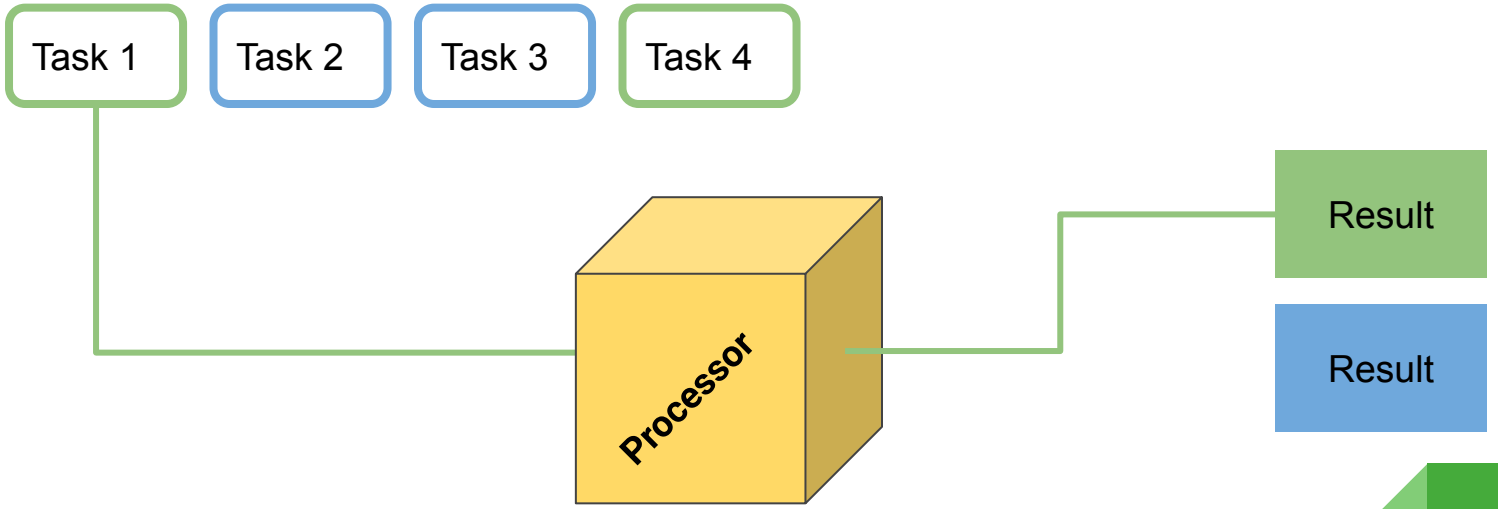


Result

Result

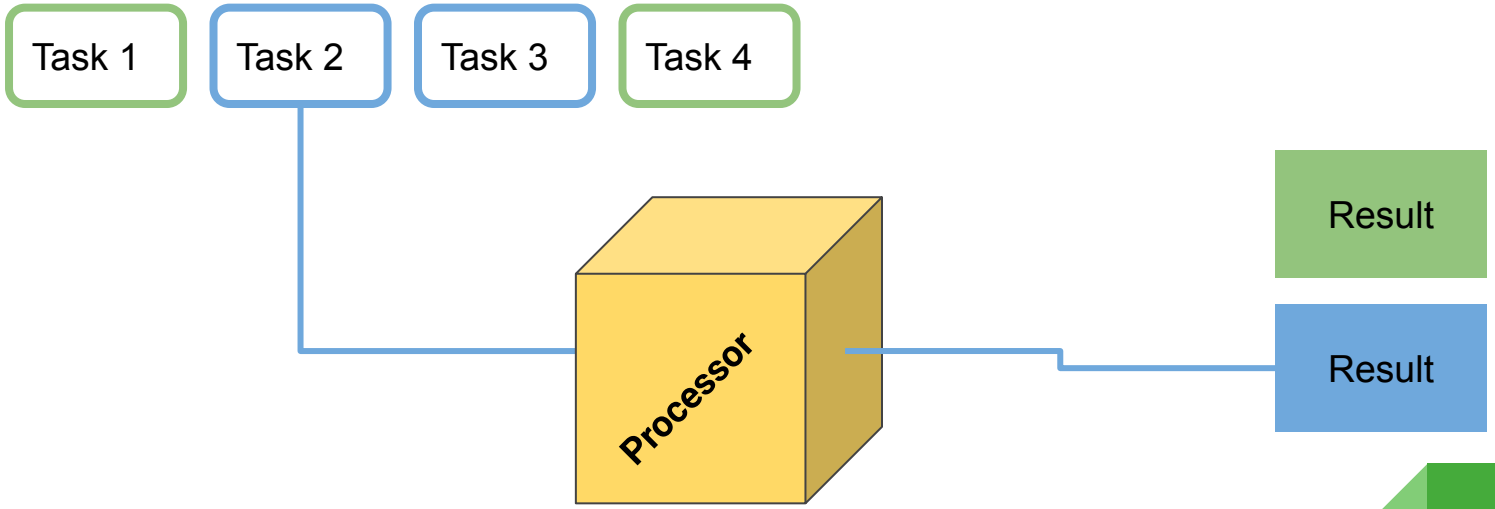
Sequential Computing

If we use sequential computing, we're going to accomplish 1 task at a time!



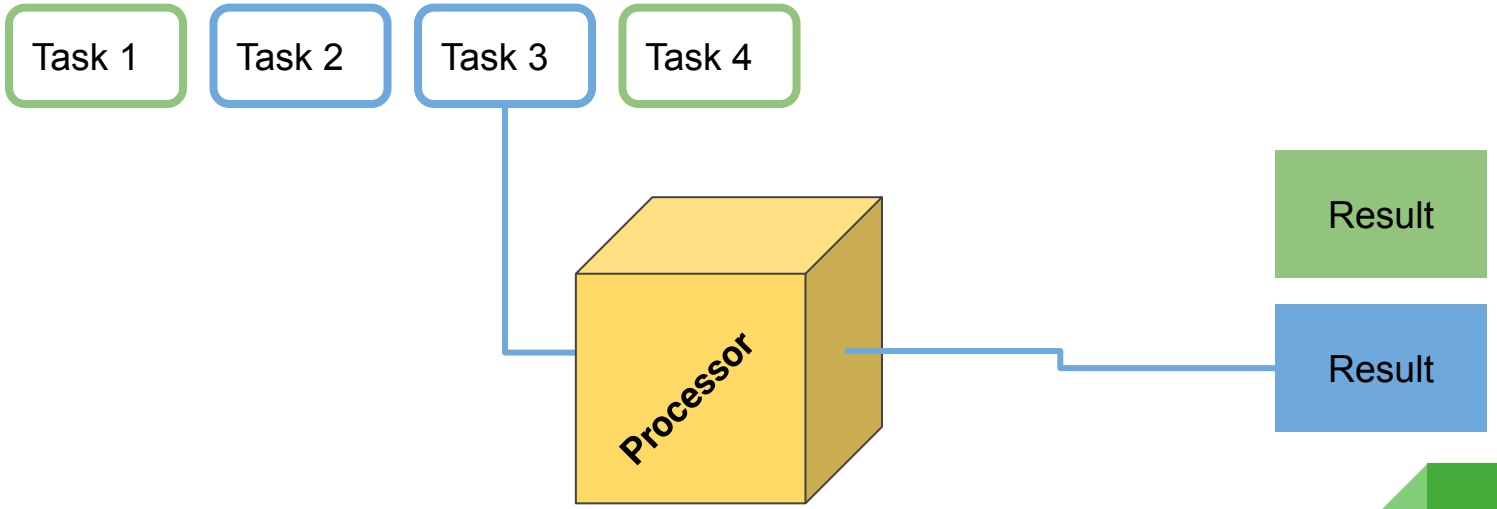
Sequential Computing

If we use sequential computing, we're going to accomplish 1 task at a time!



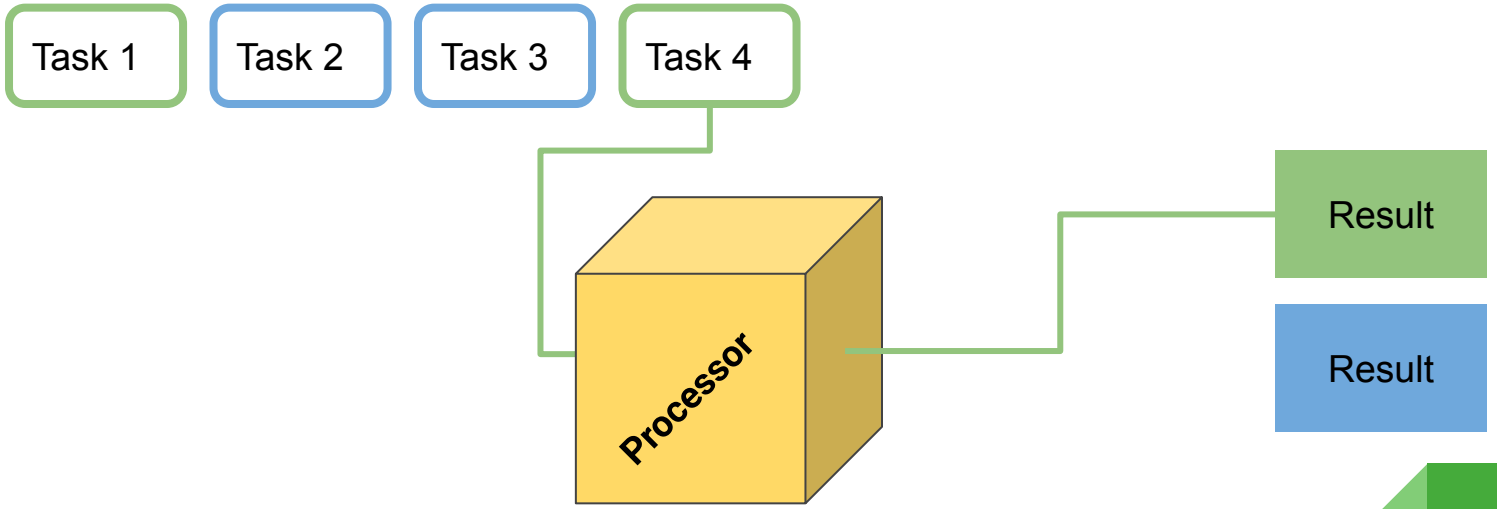
Sequential Computing

If we use sequential computing, we're going to accomplish 1 task at a time!



Sequential Computing

If we use sequential computing, we're going to accomplish 1 task at a time!



Sequential Computing

Depending on the number of tasks, this works just fine. As the number grows larger, though, the computer could take a while to accomplish all of its tasks! Imagine if we needed to accomplish over 100 or 1000 tasks - this could definitely cause some slowdown.

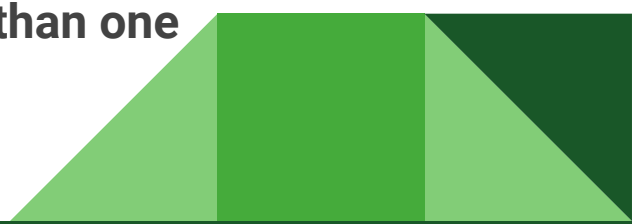
Spreading the Load

This problem is why solutions such as **Parallel** and **Distributed Computing** exist!

With these two different styles, we divide up the tasks that need to be completed, then assign different processors smaller sub-sections of the workload!

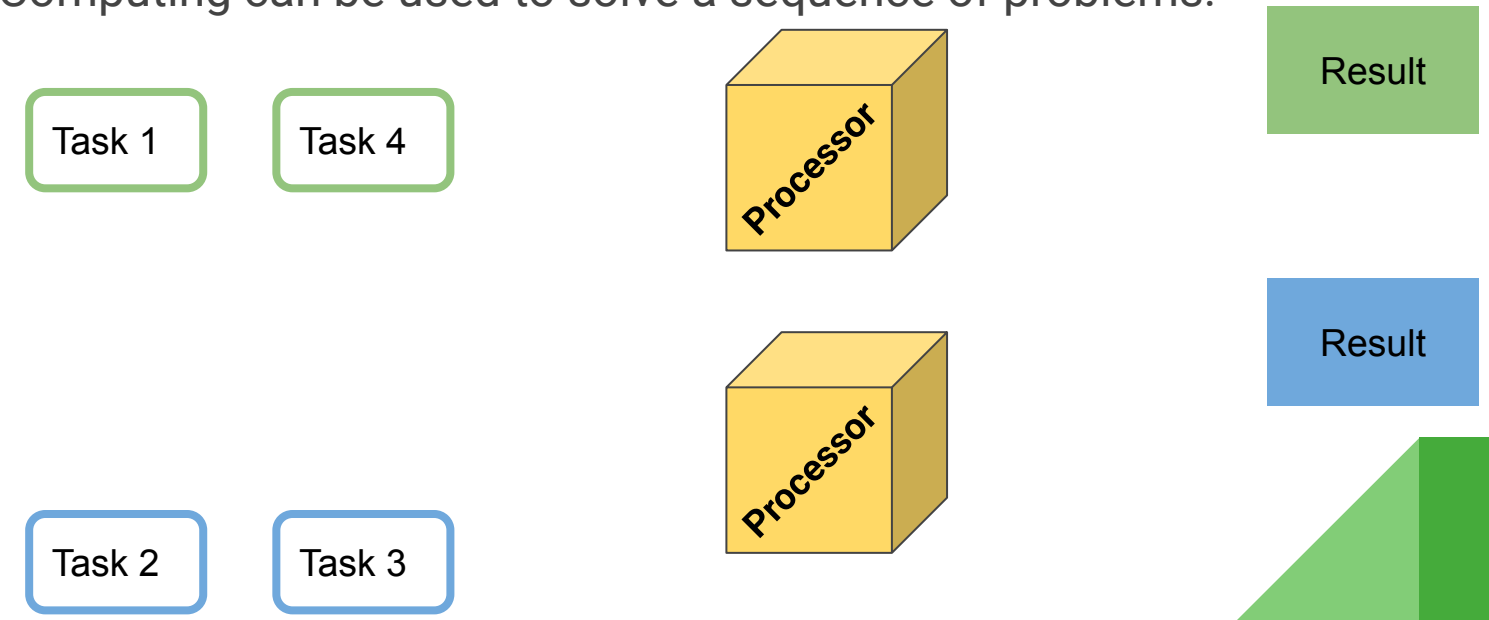
Parallel computing is most often completed within the same computer, using shared memory. Distributed computing is most often accomplished with multiple different computers, which would need to communicate with one another.

The most important thing is that both of them use **more than one processor** to accomplish their tasks.



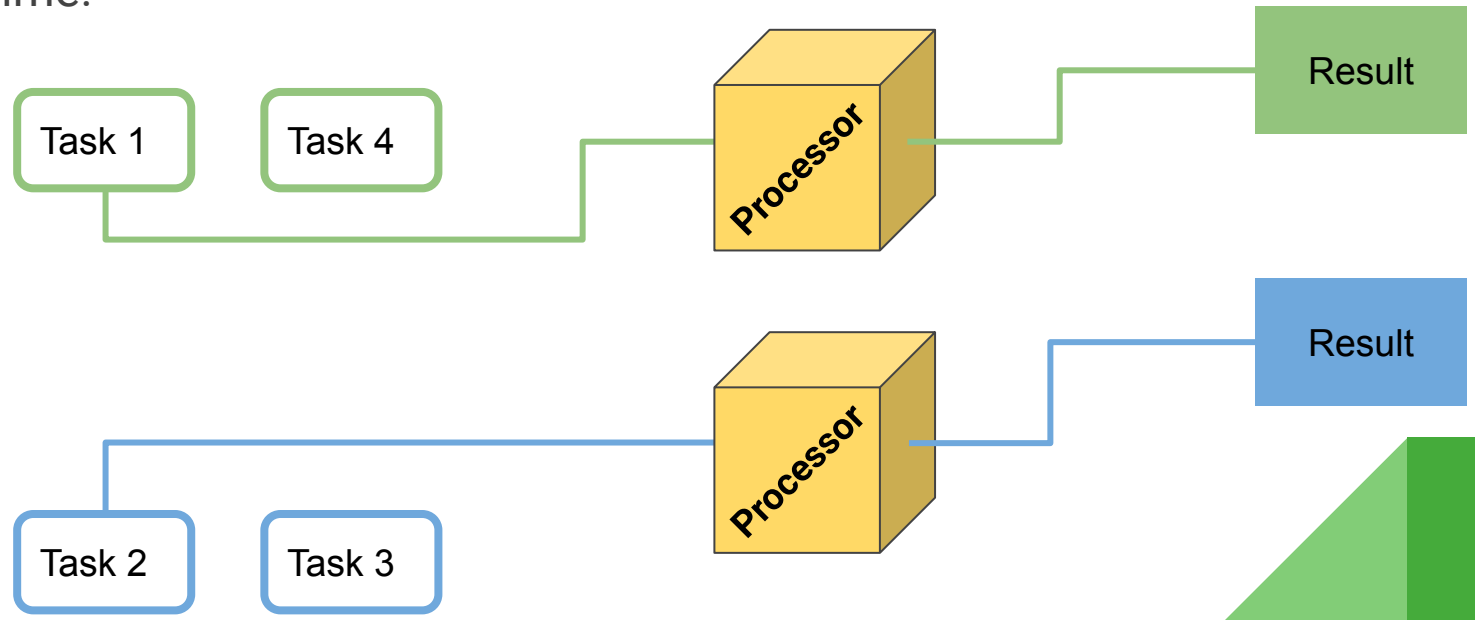
Parallel Computing

If we look back at our example computer, we can very generally see how Parallel Computing can be used to solve a sequence of problems.



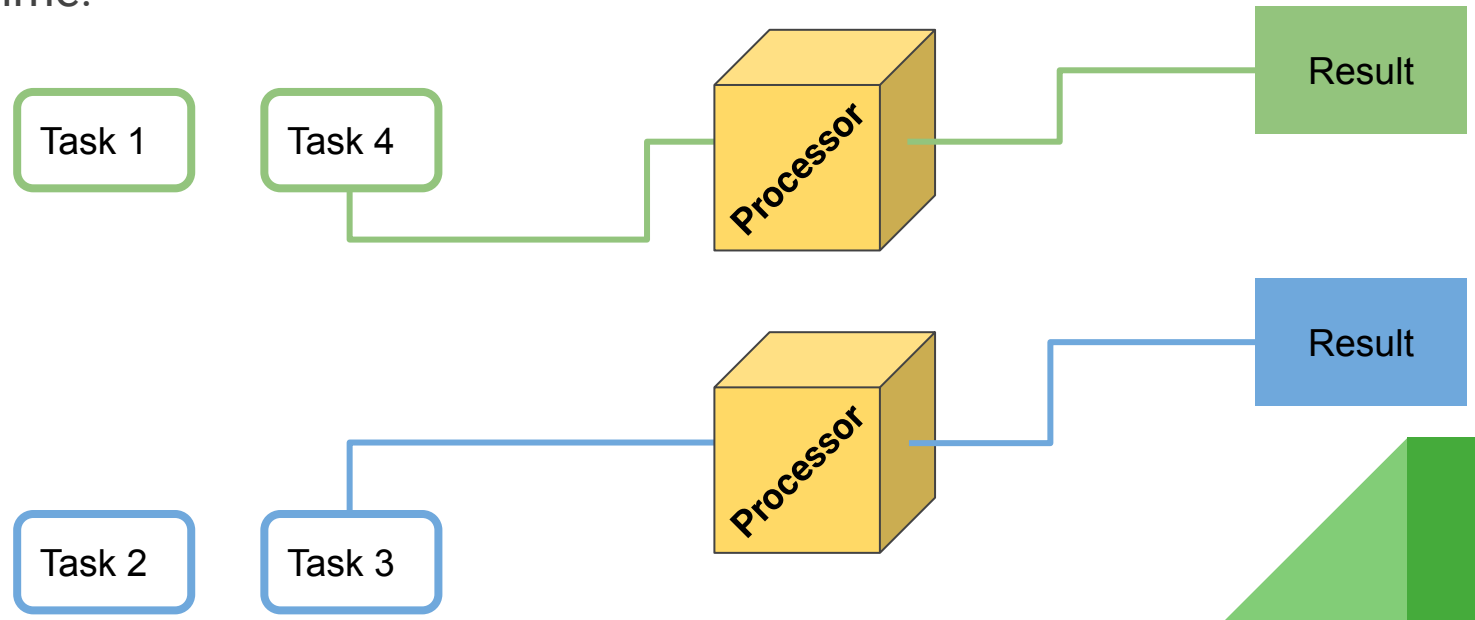
Parallel Computing

Since we have access to 2 processors now, we can complete 2 tasks at the same time!



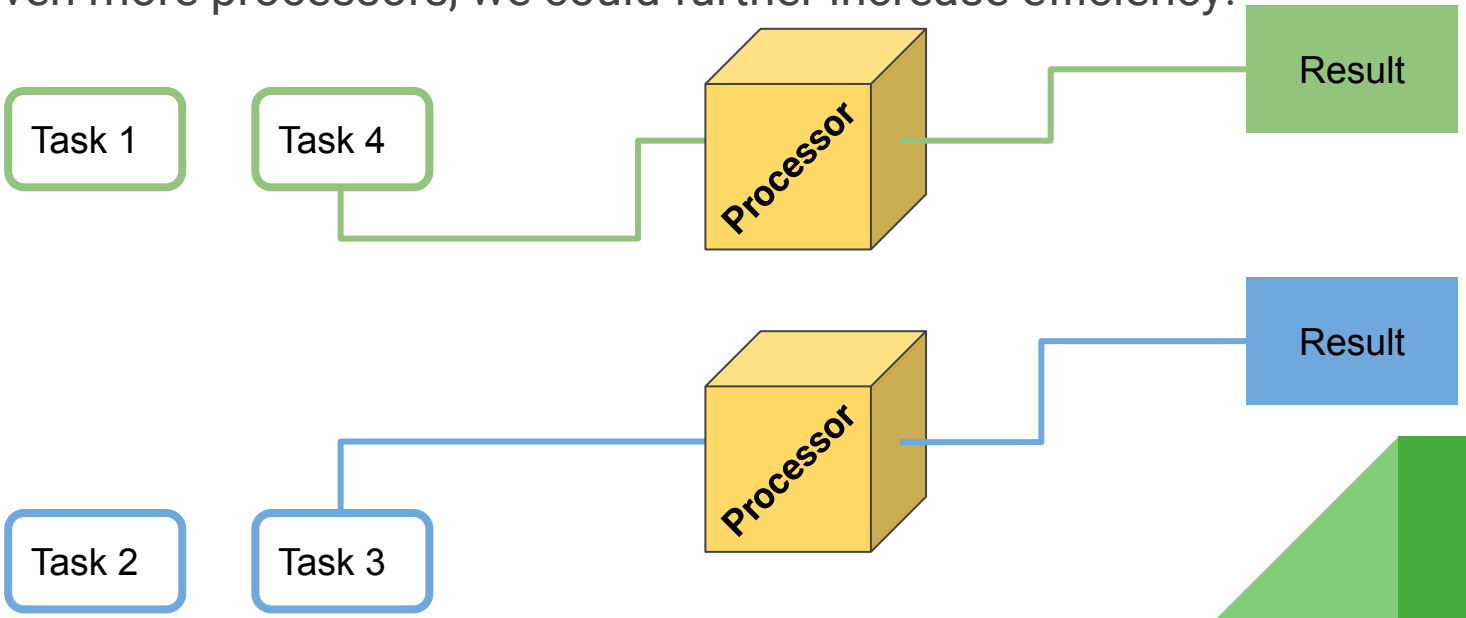
Parallel Computing

Since we have access to 2 processors now, we can complete 2 tasks at the same time!



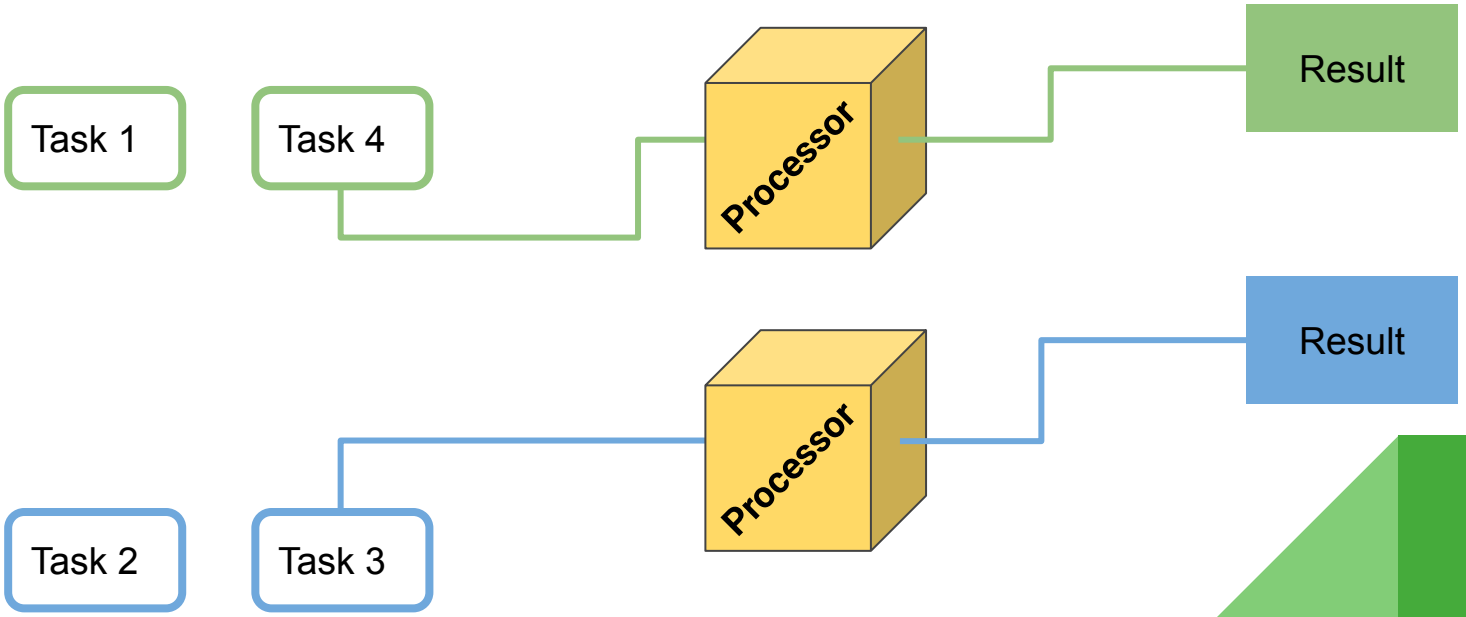
Parallel Computing

With parallel computing, we've completed our tasks in twice the speed - if we added even more processors, we could further increase efficiency!



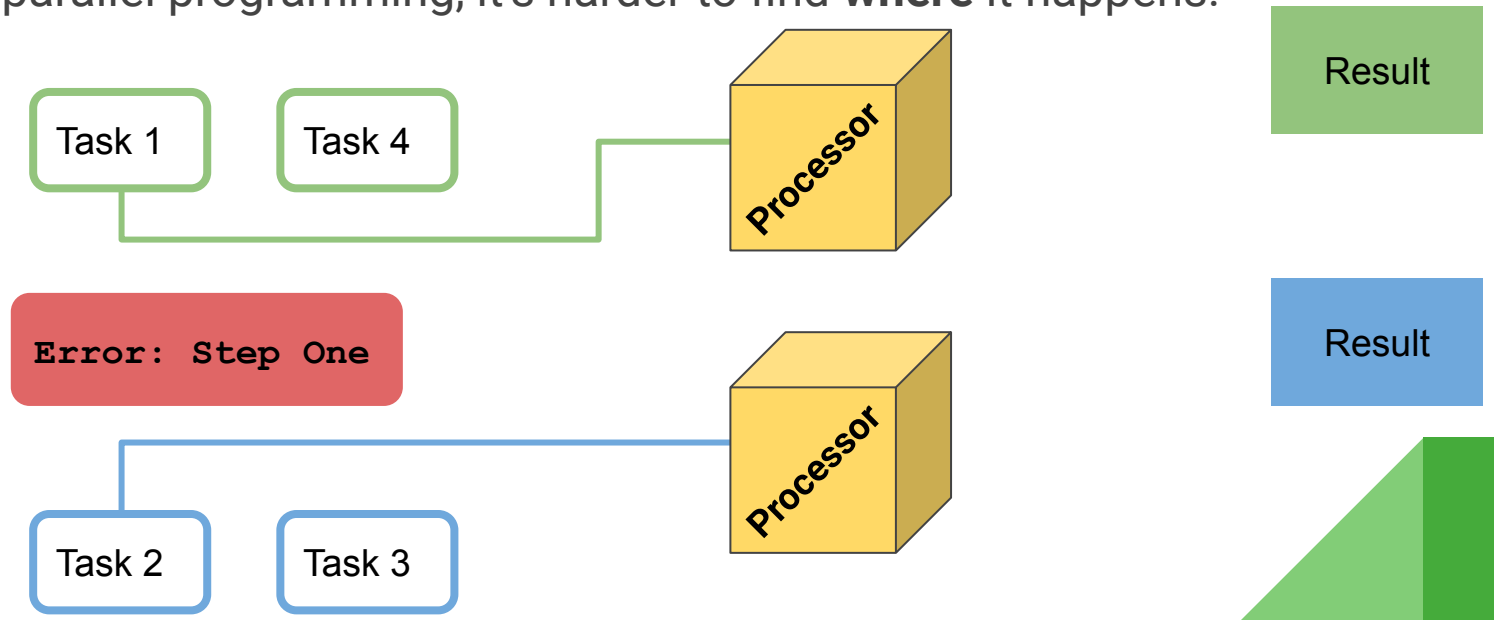
Parallel Computing

There are some challenges that arise when we use parallel computing though.



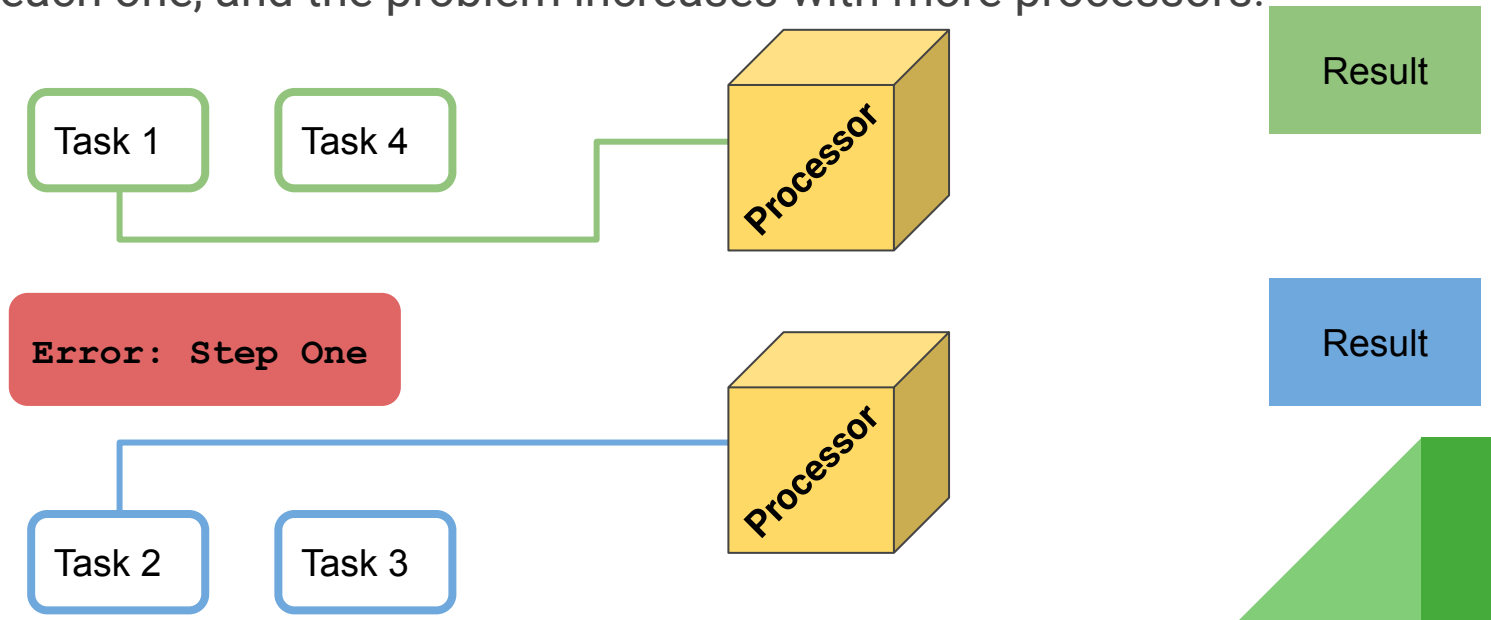
Parallel Computing

Let's say there is an error during Step One of the program. When we're using parallel programming, it's harder to find **where** it happens.



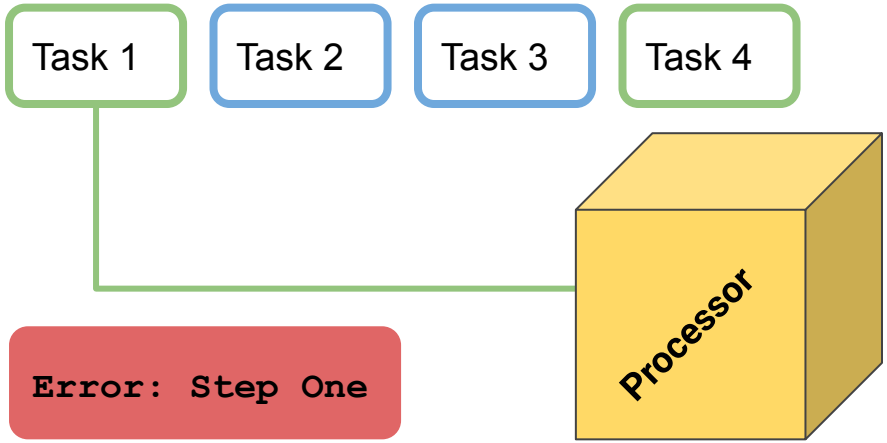
Parallel Computing

Was the error in the green step one, or was it in the blue step one? We need to check each one, and the problem increases with more processors!



Sequential Computing

With sequential computing, it's much easier to locate the error, since only 1 task is accomplished at the same time.

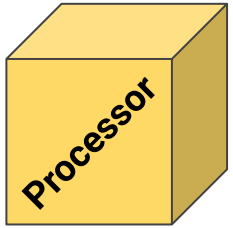


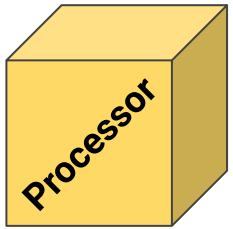
Parallel Computing

Another potential complication within parallel computing is that the architecture and setup is pretty complex. If there is a task that relies on another task, it can cause lockups!

Task 1a

Task 2b



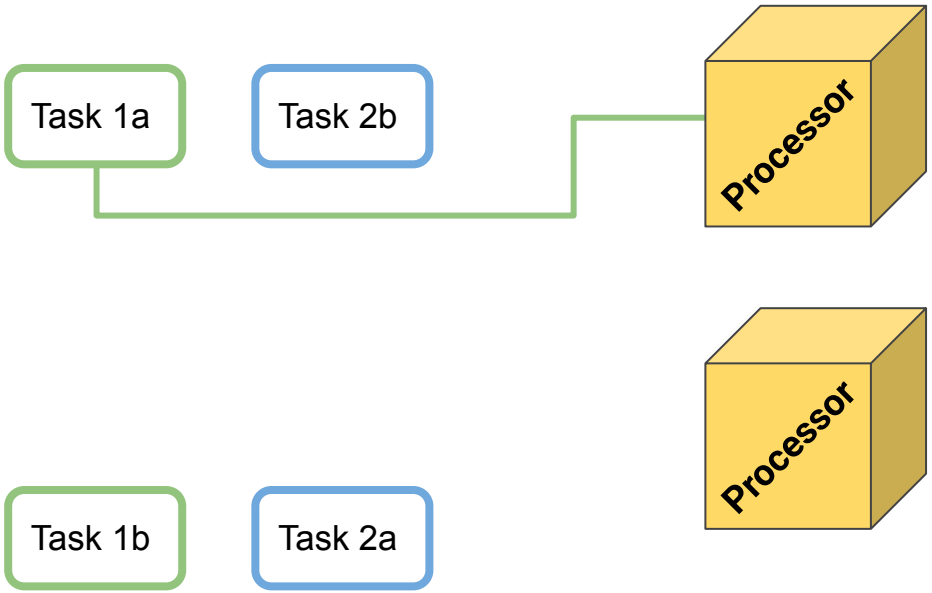


Task 1b

Task 2a

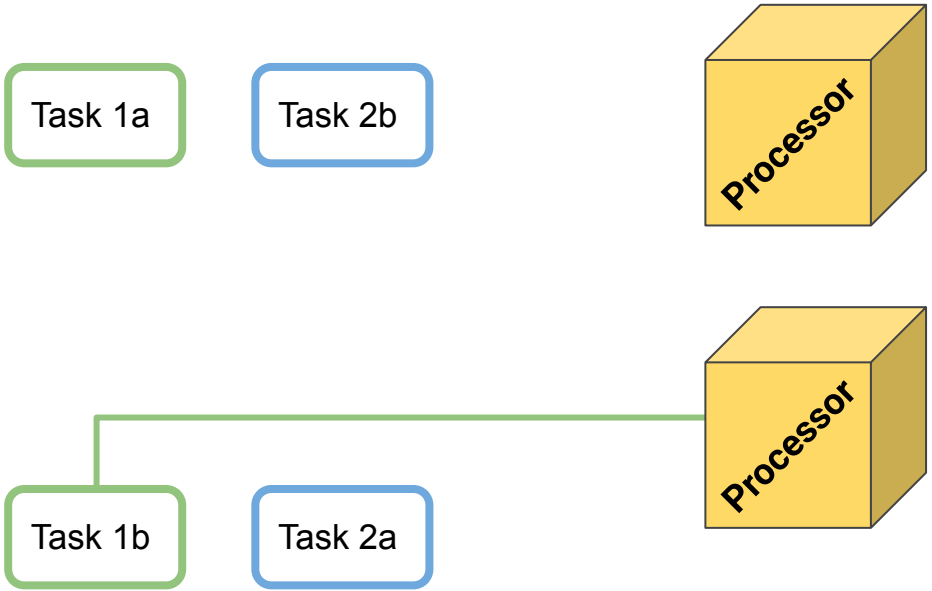
Parallel Computing

If Task 1b is waiting for Task 1a to complete, our system will be delayed!



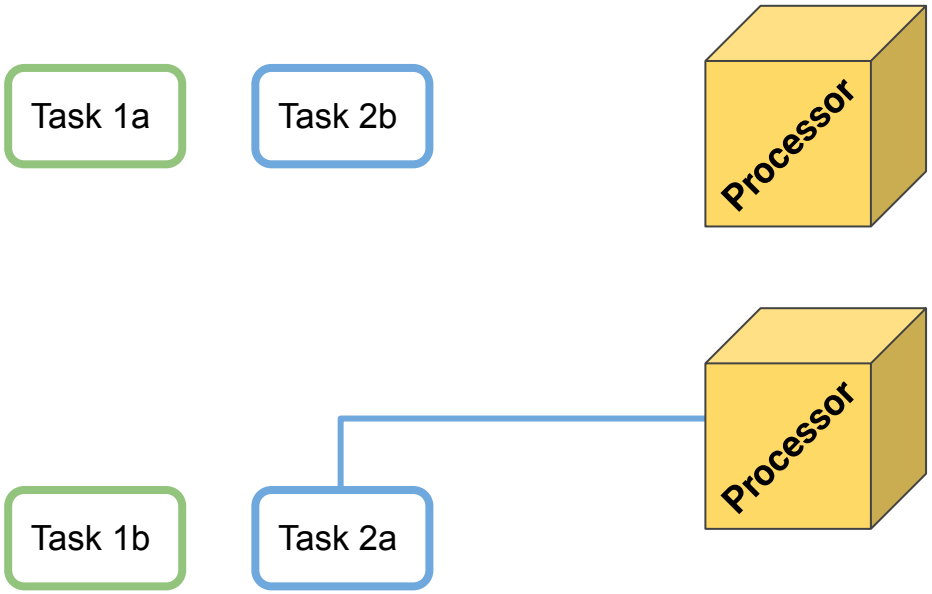
Parallel Computing

Now Task 1b can run, but Task 2b is locked behind Task 1a! That one can't run until Task 1b has finished, though!



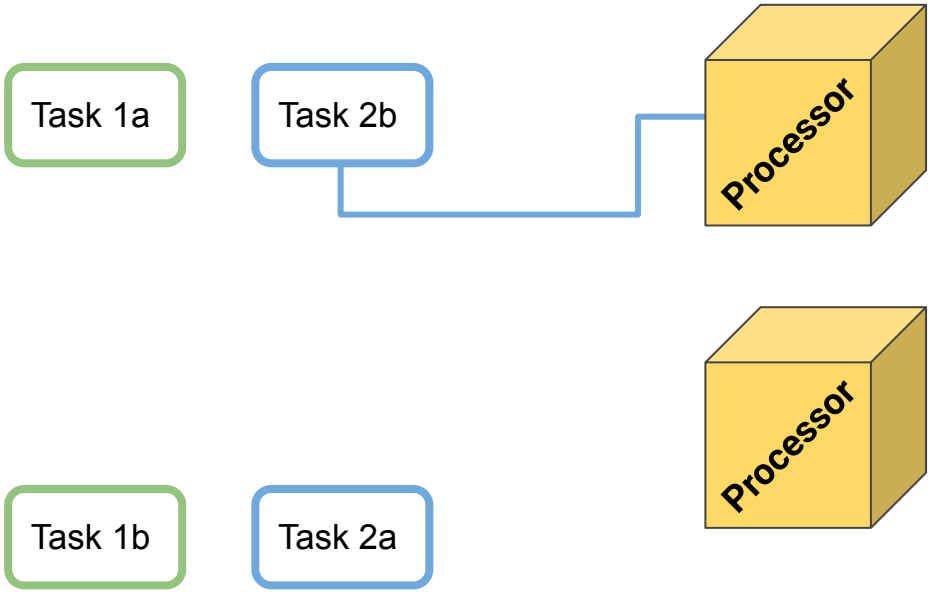
Parallel Computing

Now Task 2a can run!



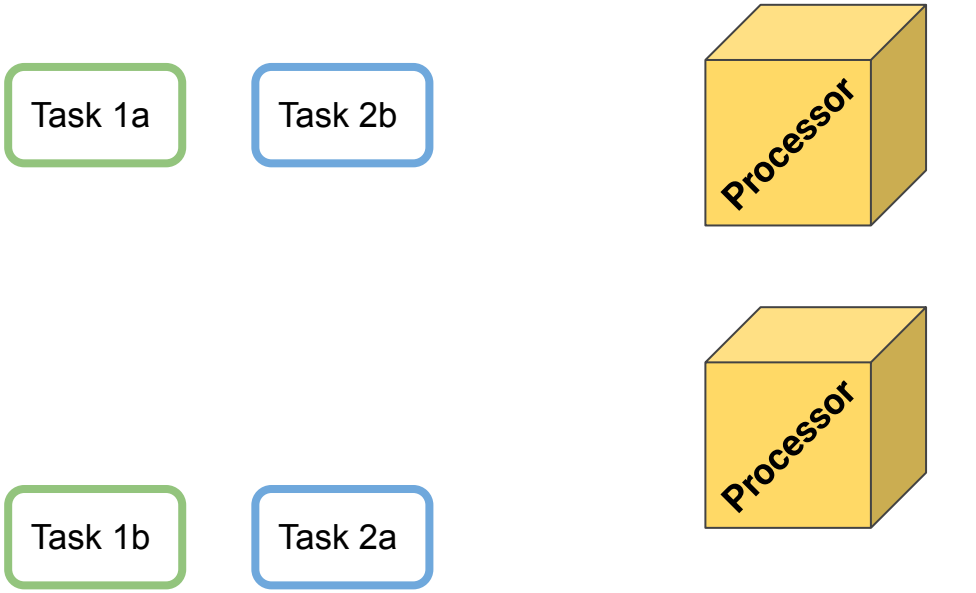
Parallel Computing

Finally, Task 2b can run. In this scenario, parallel computing didn't increase our efficiency at all - the tasks still ran one at a time!



Parallel Computing

Since adding additional processors to a computer can increase the expense of a system, this would result in a higher cost of computing with no benefit whatsoever.



Parallel vs. Sequential

Sequential	Slower Completes tasks one at a time, in order	Easier to find bugs No additional setup needed
Parallel	*Usually* faster Completes tasks simultaneously	Difficult to find bugs Difficult to use and set up correctly

Distributed Computing

Parallel computing can be paired with distributed computing by using more than one computer.

Let’s say we have a ton of tasks to complete. We COULD try to build a supercomputer with high processing power, but gets expensive very quickly. There also are still limits to how fast one computer can run - this is mostly due to the heat that is generated when computers run.

Distributed Computing

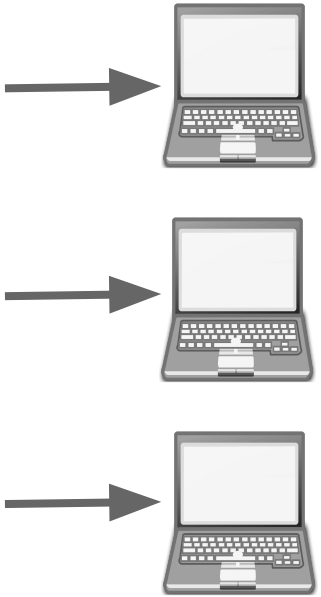
So instead of making one super powerful computer, we can instead split the workload between multiple different regular-strength computers. This still technically counts as parallel computing, because we're using more than one processor to accomplish our list of tasks!

There're many benefits to having a distributed system!

Distributed Computing

Let's say a single computer shuts down and can't keep processing.

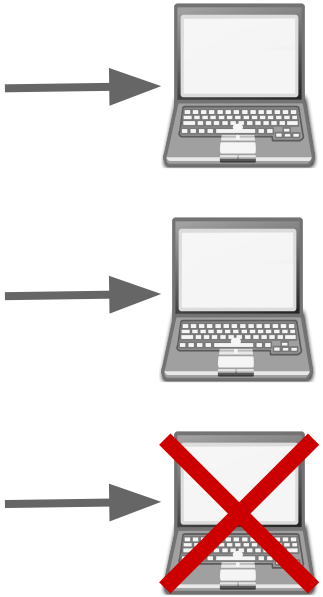
Tasks



Distributed Computing

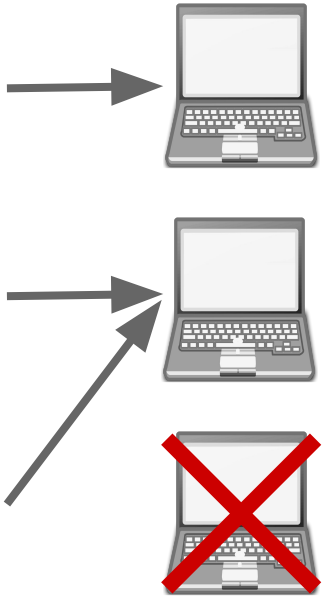
Let's say a single computer shuts down and can't keep processing.

Tasks



Distributed Computing

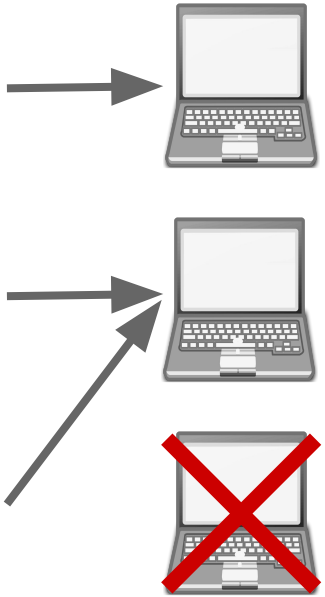
We can reroute the tasks that were assigned to that computer to one that's working!



Distributed Computing

We can reroute the tasks that were assigned to that computer to one that's working!

Tasks

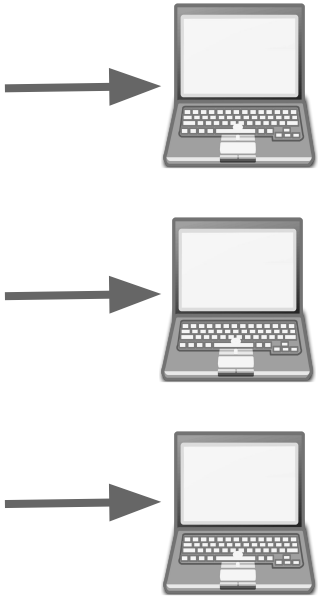


Distributed contributing has redundancy build in, just like routers on the Internet! This makes the system fault-tolerant!

Distributed Computing

We can also easily expand a distributed computing system if the workload increases!

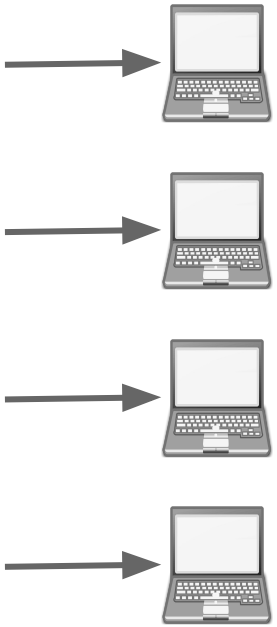
Tasks



Distributed Computing

We can also easily expand a distributed computing system if the workload increases!

Tasks



If we were working with a supercomputer, we would have to shut down the program to try to upgrade the computer to be able to handle the increasing task load. And again, there are limits to the processing power of one computer.

Distributed Computing

- Benefits
  - Scalable
  - Fault-tolerant
  - Reliable
- Challenges
  - Complex
  - Expensive to maintain