CSSE 220 Day 29

Performance with Threads

Checkout *SumArrayInParallel* project from SVN

We Used Threads For:

- We have used threads for achieving more than one “thing” at a time
  - Animation
  - WebpageMonitor
  - etc.
- What about performance?
  - Could we not get better performance by creating enough threads to divide them among different processor cores?

Java Performance

- We may not see the performance gains in Java that we can see in other languages, but there are some gains to be had…

Conceptually

- The concept is pretty straightforward:
  - If we have a large task and write a serial program, that program runs on one core, doing one thing at a time
  - Running a program in one core on our machines would be roughly as “fast” as running the same program on a processor from 10 years ago!
  - Modern processors have multiple cores
    - HOW DO WE TAKE ADVANTAGE OF MULTIPLE CORES??

Modern Operating Systems

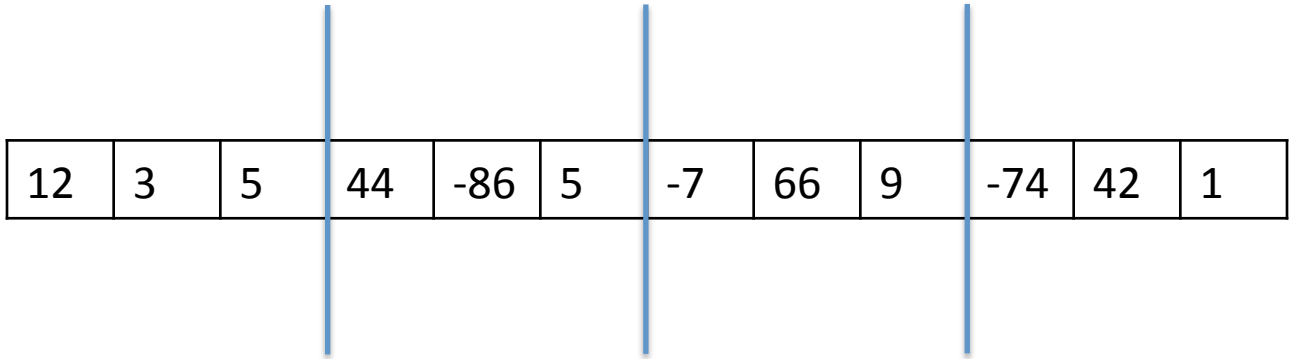
- Woo Hoo!
- Modern operating systems automatically (more-or-less) send waiting threads to a processor core that is waiting for work
- If we write the program to allow the operating system to assign threads to separate cores, then our task (in this class) is just splitting up the work into different threads!

Our Task Today

- We want to sum a huge array of integers
- Serially, we just add each array element to the current sum and then return the sum when finished
- With threads, we can split up the work very easily because of the associative law of addition

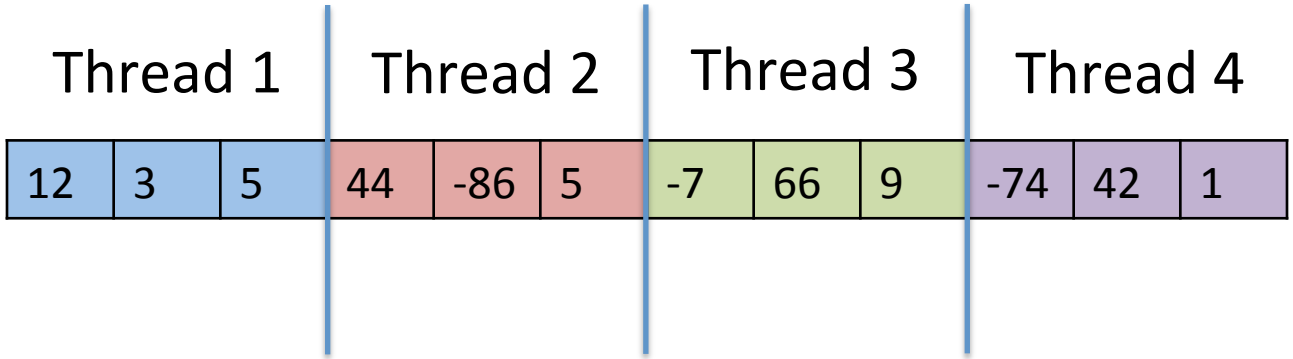
The idea

- When a very large task can be split into pieces
  - Assign a thread to one piece and let that thread return its result



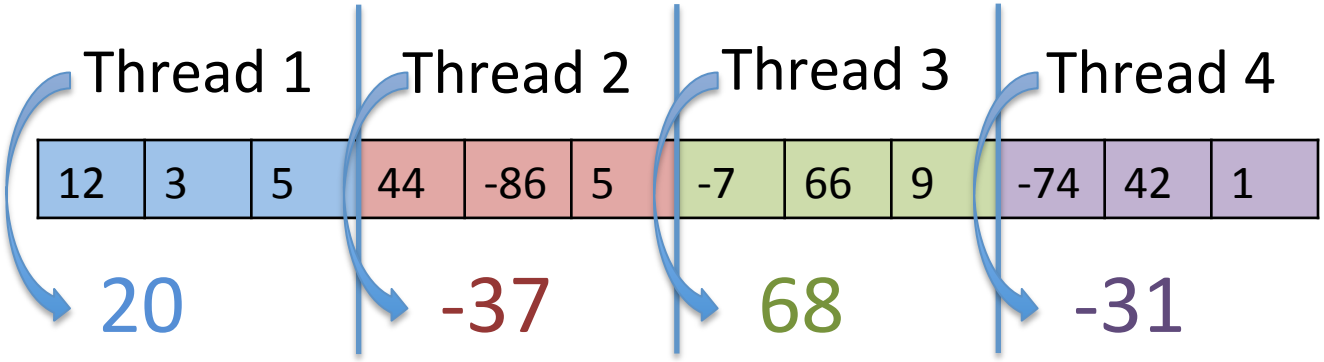
The idea

- When a very large task can be split into pieces
  - Assign a thread to one piece and let that thread return its result



The idea

- When a very large task can be split into pieces
  - Assign a thread to one piece and let that thread return its result



Add individual portions and return result: **20**

The Difference

- In our previous example, we can conceptually see that one core adding 12 numbers is “more work” than 4 cores adding 3 numbers, then one of the cores finishing by adding 4 numbers to get the result
- IN REALITY, we need to sum a very large array to see the performance gains in Java since the threads are so heavyweight
  - We’ll use about 200,000,000 integers in an array!

Matrix Multiplication

- We have a running example of matrix multiplication and how that is split into different threads

Work time

PRESENTATION IS TOMORROW!!!

TEAM PROJECT