Ethan Clark May 2, 2016

Text File	**Count of Numbers**	**Sum of Numbers**
1000numbers.txt	1000	523028
10000numbers.txt	10000	5066337
100000numbers.txt	100000	50074388
1000000numbers.txt	1000000	500869305

Java.1	The time using Threaded_Array with 1 thread is slower than the
Java.2	The times using Threaded_Array with T>1 threads are all slower
	The threads still have to wait for the previous threads to finish a
Java.3	When the machine has less cores than number threads being us
	I also think that multithreading is not very efficient in general at
	is what multithreading is very good at.

Text File	**Count of Numbers**	**Sum of Numbers**
1000numbers.txt	1000	523028
10000numbers.txt	10000	5066337
100000numbers.txt	100000	50074388
1000000numbers.txt	1000000	500869305

Ada.1	The times using tasked_threaded_array with 1 thread are faster
Ada.2	The times using tasked_threaded_array with T>1 threads are ge
	It seems that using more threads actually makes the time go fas
	Ada is actually passing each thread to a core and then the addit
Ada.3	My Ada times are faster comparitively than my Java times from

Text File	**Count of Numbers**	**Sum of Numbers**
1000numbers.txt	1000	523028
10000numbers.txt	10000	5066337
100000numbers.txt	100000	50074388

1000000numbers.txt	1000000	500869305

Ruby.1 The time using threaded_array_sum.rb with 1 thread is slower th

Ruby.2 The times using threaded_array_sum.rb with T>1 threads are al
During this concurrency, the threads still have to wait on one an

Ruby.3 Compared to my Java times above, Ruby seems to run a little bi

Ruby.4 Ruby does not manage processes well, and it seems that people
between the executing threads, rather than truely running them

	Java		
	Threaded_Array		
Time (ns)	**1**	**4**	**8**
13230	501908	763343	857090
133558	817108	855458	2554403
1209548	2154781	2748515	2822965
3278748	4351868	10161149	13742592

time using Array_Sum.
than only using a single thread.
nd then add up the results from each thread.
sed increases the time it takes to solve the problem.
adding up numbers, but other things like sorting

	Ada		
	Threaded_Array_Sum		
Time (sec)	**1**	**4**	**8**
0.000013	0.000028	0.000972	0.001519
0.000104	0.000029	0.000242	0.000753
0.000615	0.000679	0.000574	0.000567
0.002316	0.002535	0.001164	0.001157

than using array_sum (except for 1000numbers.txt).
nerally faster than the single thread.
ster (for the most part) unlike Java and Ruby.
ions don't slow it down at the end.
above.

	Ruby		
	Threaded_Array_Sum		
Time (sec)	**1**	**4**	**8**
0.000042874	0.000060718	0.000262104	0.000894468
0.000411116	0.000484068	0.000799303	0.001635749
0.004080664	0.005074965	0.005534808	0.005408567

0.041232391		0.048595014	0.074169112	0.075352471

an using array_sum.rb.
l slower than only using a single thread.
other to then do the additions.
t slower on average.
think that Ruby simply switches
concurrently.

16	24
1363646	1520970
2975171	5268328
2863792	3243180
11682759	12992519

16	24
0.001505	0.003820
0.001367	0.000921
0.001449	0.000924
0.001539	0.017810

16	24
0.001410068	0.001886627
0.002156343	0.002336938
0.009276609	0.010101899

0.084799907	0.088334993