CS336 : Parallel & Distributed processing

Project 4 Report

Summary of tasks

For this project the main goal was to make a threaded version of the circadian simulation code. We simulated each network in parallel.

Tasks

1. I updated my folder as instructed
2. I updated the generateGaussian & generateGaussianPeriods methods. I ran sim_slow and I got a similar result as in project 3

```
[mggamedz@nscc project 5]$ ./sim_slow uncoupled_5.phs 5 0.0
setting vip strength 0.000000
ret 0.000000 0.000000
gets here
here as well
here
[mggamedz@nscc project 5]$
```

3. **& 4**. I parallelized the main loop in sim_stats_1s. I generated a unique seed for each thread function using the suggested clock method. Running my parallel & sequential methods I got this output result:

Sequential

```
final stats for trial 4:    4.722659
 4.511052 5.135605 5.246645 4.908796 4.722659
Ran in 0.189651 seconds
[mggamedz@nscc project 5]$ ./sim_stats_1s_sequential 10 100 0 5
```

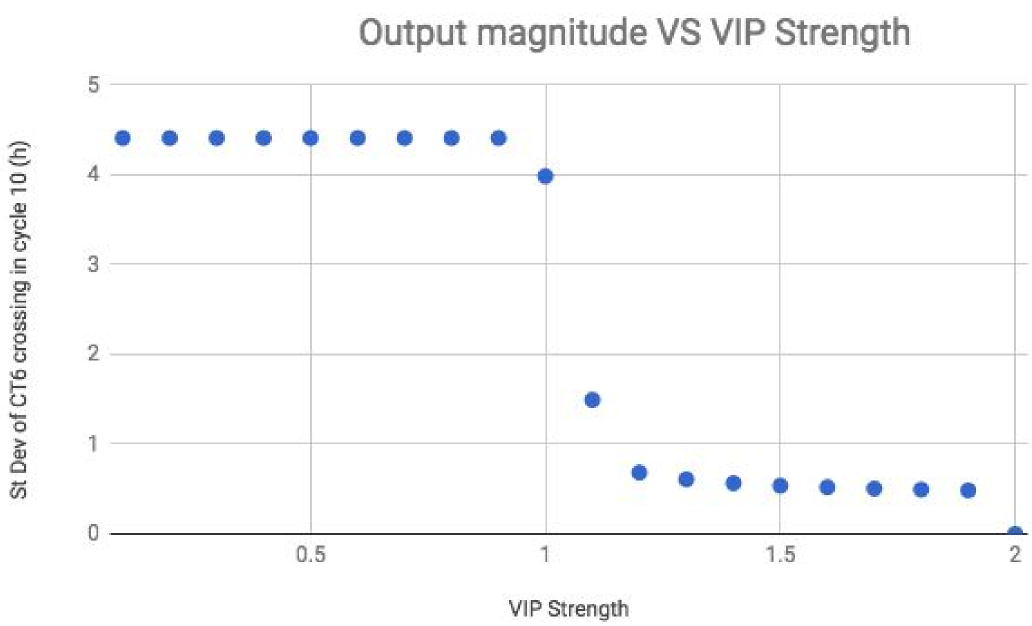
Parallel

```
 4.463455 3.660984 4.433151 5.004027 4.383939
Ran in 0.043269 seconds
[mggamedz@nscc project 5]$ ./sim_stats_1s 10 100 0 5
```

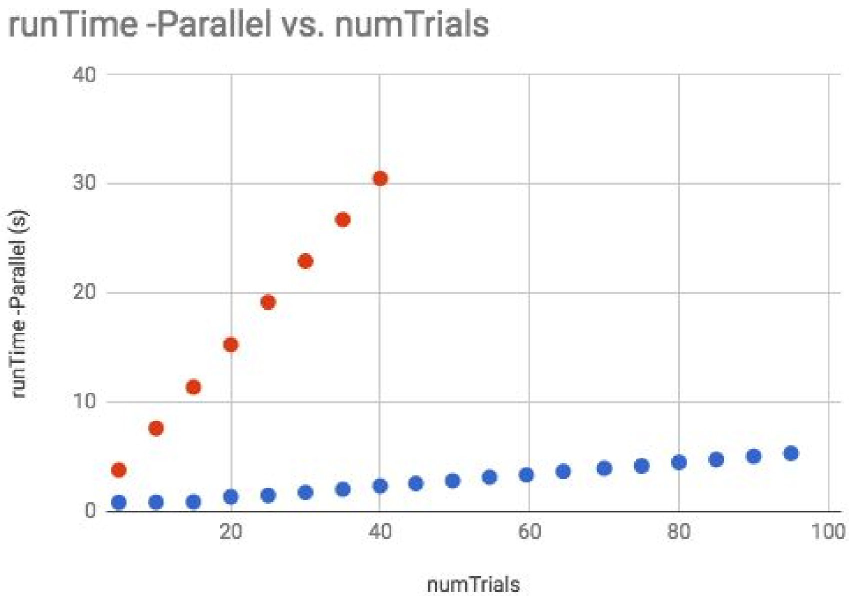
As exceptected the results are on the same ballpark

5 & 6. Using a similar method to that in sim_stats_1s, I parallelized sim_stats_ns, with numTrials determining the number of threads. Running the code with 20 trials and 20 VIP strengths, plotting the results I got this resulting table & graph:

Vip Strength	Trials					Average Output
0.1	4.453918	4.299678	4.677254	4.4509	4.159218	4.4081936
0.2	4.453918	4.299678	4.677254	4.4509	4.159218	4.4081936
0.3	4.453918	4.299678	4.677254	4.4509	4.159218	4.4081936
0.4	4.453918	4.299678	4.677254	4.4509	4.159218	4.4081936
0.5	4.453918	4.299678	4.677254	4.4509	4.159218	4.4081936
0.6	4.453918	4.299678	4.677254	4.4509	4.159218	4.4081936
0.7	4.453918	4.299678	4.677254	4.4509	4.159218	4.4081936
0.8	4.453918	4.299678	4.677254	4.4509	4.159218	4.4081936
0.9	4.453918	4.299678	4.677254	4.4509	4.159218	4.4081936
1	3.758079	3.758079	4.128345	4.292024	3.986008	3.984507
1.1	0.971656	0.957414	1.856431	2.784658	0.891112	1.4922542
1.2	0.724806	0.619914	0.670803	0.73971	0.656451	0.6823368
1.3	0.623456	0.564974	0.611548	0.641226	0.591668	0.6065744
1.4	0.584108	0.545379	0.561883	0.58517	0.545404	0.5643888
1.5	0.544462	0.499772	0.541942	0.562012	0.53213	0.5360636
1.6	0.540598	0.475861	0.531022	0.55255	0.498007	0.5196076
1.7	0.521153	0.465664	0.512357	0.534713	0.490594	0.5048962
1.8	0.514708	0.451792	0.503012	0.522885	0.477836	0.4940466
1.9	0.5056	0.451792	0.489464	0.510412	0.465149	0.4844834
2	0	0	0	0	0	0



7. Analyzing the runTime performance results when we increase the number of trials for the sequential and parallel sim_stats_ns.c. The result was expected, as the parallel version proved to be more efficient for multiple number of trials, as shown below:



I got the expected speedup, as the red represents the sequential version, whilst blue represents the parallel version. The runTimess for both versions exhibit a linear propagation, which indirectly correlates to the number of trials/processors in the parallel version.

8. According to my table of plots in tasks 5&6, as the VIP strength increases the standard deviations decrease in magnitude. This result shows that a larger VIP strength leads to better synchronization in this model.

Extensions

None

Collaborators

I worked alone on this project.