AllDepOnTime

Proposed by George Chang and Lijie Yang.

Section 0 URL:

https://qiyouchang.github.io/AllDepOnTime/

Section 1 Summary:

we will implement Optimized time-based thread assignment on simulated human networks

Section 2: Background:

Man is by nature a social animal. Everyone may like or dislike another person and at the same time, experiences like or dislike from another person. Thus the relationship between humans can be simulated by a human network (an undirected weighted graph in CS terms).

Let's define the following rules for the network:

1. People are nodes and have weighted edges to another person showing how much they like/dislike the other person on a scale from -1 to 1.
2. Every once in a while, everyone will evaluate how healthy his/her living environment is by asking people within 2 degrees of separation to report their average like/dislikes rate and calculate based on this.
3. Upon getting the result of calculation in the last step, everyone will delete the edge with x friends based on their individual evaluation of how healthy the society is and add y new friends randomly. If the person thinks the society is healthy, then the person will drop x of his pessimistic friends. If the person thinks the society is not healthy, then the person will drop x of his optimistic friends. (x, y are tunable parameters)
4. After a constant number of iterations, we will gather the evaluation data (to another person or to the society) in general.

Speaking in technical terms, the system looks like this:

1. System is an undirected, weighted, randomly initialized graph where the values of edges are between -1 and 1.

2. In each iteration of simulation, communication between connected nodes are needed in order to send the average preference list. Then each processor/node will calculate the overall average and report to the master which stores all the evaluations.
3. When the calculation is done, x number of edges of each node are dropped and y number of new edges for each node are created. X,y are tunable parameters.
4. In the end all evaluations from each node are gathered to the master processor and the final result will be calculated.

This system will benefit a lot from parallelism because each node can calculate its evaluation independently. This system requires at least 2*(number of edges) messages to be sent between nodes so it will involve really heavy computation if no parallelism is used.

Section 3: Challenge

The problem is challenging because the network changes drastically as we add and drop new edges from each node and take 2 degrees of separation (friend and friend of a friend) so the calculation time may increase or decrease dramatically in two adjacent iterations. Thus, it would require us to reschedule which people are assigned to each processor to balance the workload. We will need to make predictions on how much work each processor will take (how many friends of a friend is not known) and that will be difficult and distinct from previous work we have done in this course. We want to learn more about MPI, and implement more efficient processor task assignments based on accurate predictions on the time cost of each node.

The workflow will look something like this:

1. The initial network is loaded to the master processor.
2. Nodes are assigned to processors based on our prediction algorithm on the time cost of each node. The respective nodes will be sent by the master processor to the other processors.
3. Each processor will communicate only with the processor recorded in the preference list to get their evaluation score.
4. Calculations of new evaluation score will be done on each processor in parallel
5. The edges of each node then change: drop some edges based on evaluation score and add some edges randomly to create some randomness.
6. Summarizing the workload of each processor to the master processor. If the workload is very imbalanced, repeat step 2. Else start with step 3 above.

Each processor is dependent on the information sent by the other processor and is dependent on the node list sent by the master processor. There is memory accessed by the master at the start and end of the program and there might be one master copy of nodes' preferences lists/ evaluation lists in the master process (but that will depend on the tradeoff between the number of communications and the size of each communication). The diverge execution happens in step 3,4 above where each processor separately makes evaluation.

As mentioned earlier, the most challenging part is that the workload of each processor is not completely known to the processor itself: the number of friends of friends is not known. Thus, we need prediction algorithms to predict how much nodes are connected to our neighbor nodes to avoid too much communication. The workload imbalance between processors and the reduction of communication cost are the two most important challenges we will face.

Section 4: Resources

We will start from scratch with no outside data/previous code. We may do research on the books/papers related to human network, computer network constructions, message passing, processor communication which we are unsure which exactly we will reference as of now. We will consult modules like MPI homepage and C++ help pages for syntax/function usage. We will also need to access PSC machines + GHC machines to benefit from the multiprocessors system that it has.

Section 5: Goals

PLAN to ACHIEVE:

1. Develop the system we have described in the earlier section (background/challenge section) complete with no error.
2. Expect a decent speedup from a sequential version of our system. Not sure how decent speedup will be because it will depend on the dataset we use and the number of edges each node will have. However, it will be indicative of the effectiveness of MPI in improving performance.

HOPE to ACHIEVE:

1. Extend the 2nd degree of separation to a higher degree of separation. This will require a much more sophisticated prediction on the time cost of each node and communication cost will be more. This will be hard because we will need to handle more message sending / synchronization of messages.
2. Or experience different communication strategies when the hyperparameters/dataset we created are different. This will be interesting because from testing it, we may be able to get a balance point to indicate when it will be good to have more communication and when it will be better to just do some extra work in each processor.

Demo planned at the poster session:
We will prepare a poster that includes the performance of our network model run on different numbers of cores with various iteration values. In the optimal case, our model will provide a significant speedup while parameters are changing.

Our background problem tends to be an optimization problem due to the fact that the network is almost completely randomized, the workload among different processors could be significantly imbalanced. Moreover, each processor is dependent on the information sent by the other processors and is dependent on the node list sent by the master processor. There is memory accessed by the master at the start and end of the program and there might be one master copy of nodes' preferences lists/ evaluation lists in the master process (but that will depend on the tradeoff between the number of communications and the size of each communication). Thus, we will construct our parallelized algorithm to predict how many nodes are connected to our neighbor nodes to avoid too much communication. The workload imbalance between processors and the reduction of communication cost are the two most important challenges we will face. In this case, hopefully our algorithm will achieve at least 35x compared to single-core sequential algorithm.

Section 6: Platform Choice:

GHC and PSC with multicores will be used as our machine to run the program since they can provide at maximum 128 cores, which provides us with the capability to measure the performance of the program on various configurations. We use C++ as main programming language since we need to MPI to apply multicore parallelism and C++ will best support those resources that we are going to use on our model

Section 7: Schedule

The course schedule is organized as:

—----------

Description Due When
Discuss Ideas with Instructors Mon/Wed, March 27/29, 9:30-11:00 am
Project Proposal Due Fri, March 31, 11:59pm
Milestone Report Due Wed, April 19, 9:00am
Final Report Due Thu, May 4, 11:59pm
Poster Session Fri, May 5, 8:30am or 1:00pm

—----------

Based on the course schedule, we decide to work on our final project following such timeline:

	Meeting 1 (Mon.)	Meeting 2 (Thurs.)	Instructor Communication (Wed.)
W1(Mar 31st to Apr. 7th)	Finish the final project proposal, decide main focus of the project, and assign work to each week	Search for resources (papers/ relevant open-source project information) and perform different ideations and decide which one to work on	Discuss various ideations with instructors during OH and ask for their advice (which model could be better to implement and if any other resources/APIs could be considered when implementing the model

W2(Apr.10th to Apr. 14th)	Implement basic structure of the model with available resources and discuss obstacles encountered, and prepare to implement the whole model and **start Milestone Report**	Done with the general model implementation as first prototype and recheck the defects of the model, think about possible ways of optimization while starting performance testing	Show the first prototype to instructors and ask for their advice and possible ways to optimize. Again, think about any more resources that can be added to the prototype during future optimization periods. Moreover, ask about any other aspects of program could be tested but not considered yet
W3(Apr. 17th to Apr. 21th)	Optimizing the first prototype and complete second version of model with performance test and brainstorm other possible ways of optimization and **submit Milestone Report**	Finish second prototype and refine the model to final version with advice from instructors and **start writing poster and final report**	Ask instructors for help on optimizing the second prototype and maybe more ways of testing and optimization
W4(Apr. 24th to Apr. 28th)	Test final performance/ improvements brought by our model and write the report and design poster, finishing it before meeting instructors	Refine final report based on advice given by instructors and redesign the poster according to final report of new version	Check details of report and poster with instructors and ask for advice on refining
W5(May 1st to May 5th)	Ready to **submit the final report**	**Finish the poster**	Final advice on poster and check if anything needs to be changed