**CPT S 411: Final White Paper (instructions and guidelines)**

| | |
|---|---|
| **When to submit?** | **11:59pm PDT, December 9, 2019 (Monday of Finals week)** |
| Where to submit? | Upload on Blackboard learn.wsu.edu dropbox |
| Will there be any extension? | **NO EXTENSION; this is a non-negotiable absolute deadline** |
| Assignment type: | Individual |
| Submission file type: | Each submission will be accepted only as a single ZIP file |

This course has a white paper required for all students, instead of a final exam. The paper is due at or before 11:59pm on December 9, 2019 (Monday of the Finals week).
This is an individual assignment (i..e., not team-based).

***How it works?***

- I give you a problem statement (see below).
- You design algorithms (consistent with the problem specification), provide implementations, design and perform experiments, analyze your results, and prepare a detailed report and a quad chart (adhering to the instructions below).
- You make the submission electronically by the deadline mentioned above.
- Go home and enjoy your Xmas break!

Collaboration rules:

- You are allowed to discuss with one another (or with me) during the design phase and experimental phase. Discussions should be restricted to understanding the problem parameters better including any assumptions, obtain a high level of understanding of the approach, and figuring out your experimental design/plan. Acknowledge these people in the Acknowledgments section of your white paper.
- You are *NOT* allowed to share or show any part of your solution (code, specific algorithms, report) to anyone in the class.
- You are *NOT* allowed to refer or use materials available online.
- The entire implementation and entire report should be 100% yours.
- If any deviation from the above rules is observed, it will be considered cheating, and an automatic F grade will be assigned.
- If you are unsure about something please discuss it with me prior to contacting anyone.

**Problem Statement: Parallel Pagerank Estimator**

In this project, you will design, implement and test a parallel estimator for PAGERANK of all nodes in a graph.

The Pagerank of a node (v) in a graph is an indicator of the node's importance. Take for instance, a webgraph, where nodes are webpages and edges are crosslinks (directional; incoming or outgoing) between two webpages. In this context, the Pagerank of a node is the relative importance of that webpage on the internet (higher the better).
The original Pagerank paper was written by Larry Page, Sergey Brin and others (@Stanford, and later Google):

*Page, L., Brin, S., Motwani, R. and Winograd, T., 1999. The PageRank citation ranking: Bringing order to the web. Stanford InfoLab.*

For the purpose of this white paper, you will design, implement and test a parallel estimator for Pagerank calculation.

**Design:**

Questions:
More specifically, in your white paper you are expected to answer these two questions:
1) Multithreaded version using OpenMP: For this you will design, implement and test.
2) Distributed memory version using MPI: For this, you will design an algorithm and present the pseudocode. I don't expect you to implement a code for this part. But I need your design details including your algorithm and related challenges.

High-level Approach:
The **input** is a directed graph G(V,E) with $n$ vertices (aka. "nodes") and $m$ edges. Vertices represent webpages. Edge $(u,v)$ means there is a hyperlink from webpage $u$ to webpage $v$ (and hence this edge is directed from $u$ to $v$).
The **output** should be the Pagerank that you estimate for every vertex.

The **approach** that you need to use to estimate Pagerank is as follows:

Starting from every node in the graph, perform a random walk of a user-specified length $K$. Keep track of how many times the random walks visit *each* node. The *Pagerank estimate* for a given node $u$ is then given by the number of times $u$ was visited (as part of any random walk) divided by the total visits across all vertices - i.e., in other words, the fraction of visits that occurred at node $u$.

To determine each step of a random walk, let us say a given random walk is at node $u$ at any given stage. And let the out-degree (i.e., the number of 1-hop neighbors) of $u$ be denoted by $d(u)$. Then to determine the *"target"* node to jump to for the next step, follow this procedure:
a. Toss a coin which has a probability $D$ of landing heads and a probability of 1-D of landing tails. The parameter $D$ is called Damping Ratio and is a user-specified probability term (obviously, D should be a term between 0 and 1).
b. IF the toss = tail
THEN target = pick a node among the pool of $u$'s neighbors with equal probability (i.e., $1 / (d(u)+1)$ )
ELSE (// i.e., toss = head): target = pick a random node among the entire set of $n$ nodes with equal probability (i.e., 1/n)

**Implementation:**

As pointed out above, you are required to implement and test only the OpenMP version of your algorithm.

**Testing:**

Input Data Sets:

Posted below are different real-world networks of varying sizes and shapes:
web-Google.txt
web-BerkStan.txt
web-Notredame.txt
facebook_combined.txt

These networks have been downloaded from the Stanford Large Network Dataset (SNAP). Descriptions about these networks can be found there. You are welcome to download and test with other networks from this site.

Experimental design:
Your code will need to accept two parameters {$K$: length of random walk} {$D$: damping ratio}

In your output, include the top 5 nodes (with the top 5 highest Pageranks).
(Note that given the stochasticity in the process, there is no one correct answer. But if you play with the parameter $K$ you are likely to achieve a majority or consensus across runs.)

The remaining parts of your experimental plan is something that I would like you to design. This is a critical part of your project.

**White paper: (to be followed exactly!)**
The white paper should be NOT EXCEED 8 pages including any required references. The document should use 1.5 line spacing, 11pt Times New Roman or 10pt Arial font, and 1" page margin on all sides. Adherence to this format is mandatory. Reports that do not adhere to this format will *not* be graded.

The report itself should be written like a short scientific paper or technical report. The recommended format is as follows (some deviations from this format will be tolerated):

Section 1) Introduction
- Provide problem background, with some motivation for parallelization as you see it.

Section 2) Problem Statement
- Problem definition (what is the input? what is the output?), use a formal definition to the extent possible. We used formal ways to define a problem in the class. Look at parallel prefix sum or other problems for examples.

Section 3) Key Challenges in Parallelization
- State briefly why parallelization poses challenges for this problem

Section 4) Proposed Approach(es)
- Propose your key ideas and approach elements here. It is encouraged that you describe your algorithms precisely in the form of a pseudocode. Use figures to help illustrate and articulate the main ideas clearly. This section should also do a complexity analysis of your algorithm - space and run-time complexity.

Section 5) Experimental Results and Discussion
- Please report your experimental results and your evaluation of those results in this section. Also add your discussion in an objective manner. This section must contain the main results as figures/charts/tables as you deem appropriate and all related discussion. I will be looking for what all performance and quality based evaluation you plan to include in this section. This is all part of your experimental plan. I am deliberately withholding information on that.

Section 6) Acknowledgments
- Acknowledge any people you would like to as appropriate.

Section 7) References
- This section should consist of citations to any literature that you have cited in the main text (any of the above sections). The bibliography format should be that of IEEE format (other similar ones okay), which means that each reference will be numbered and the numbered entries would appear like this:

[1] [author1_initial]. author1_Lastname, [author2_initial]. author2_Lastname. "Paper title", {Conference or Journal name}, {page and volume numbers}, {Year of publication}.

If you look up papers on Google Scholar there will be a CITE (") link next to each paper. If you click that it will show you different formats. You can use one of them.

**Quad chart:**

I would like you to use this Quad Chart (single slide) format and populate it for your solution. The idea of such a quad chart is to quickly convey the main highlights of everything that you have done (problem, solution, results/findings) in a single slide. It will be easy if you imagine giving a 60-second elevator pitch to a CS-literate person while preparing this Quad Chart.

**What you need to turn in?** (all bundled up in one ZIP file)

a) White paper (PDF)
b) QUAD chart (PPT or PDF)

c) Your source code folder. Please don't include test data sets (i.e., inputs). I have those. Just include your source code. If you used any additional inputs beyond the above list, please provide references to them in your report so that I know where to get them.

All the above parts are mandatory.

**My Rubric for Evaluation: *IMPORTANT***

In my evaluation of your white paper I will be looking for details related to the following set of broad questions (not meant to be exhaustive):

Design (for both versions):

1. Are the algorithms well described?
2. Are the challenges pertaining to the design of those algorithms adequately related to?
3. How well conceived are the solutions? Are the solutions technically sound and efficient?
4. Have any potential pitfalls or limitations been identified?
5. Are the assumptions (if any) spelled out clearly? Are there are any unwarranted/restrictive assumptions made?

Implementation and Testing (only for the OpenMP code):

1. Is the code consistent with the algorithm presented in the design section?
2. Are the results and observations adequately justified and/or explained?
3. Have performance characteristics of the multithreaded code appropriately analyzed?
4. Does the paper have an overall sound experimental design?
5. Have the effect of different scheduling schemes and synchronization schemes for the multithreaded code appropriately analyzed?
6. Has the effect of changing parameters like K and D appropriately analyzed?

Presentation and style:

1. Is the white paper easy to read and understand?
2. Is the writing style clear and scientific/technical?
3. Are there helpful figures and/or illustrations to facilitate reading of the paper?
4. Is there any sloppiness (e.g., typos, inconsistent formatting, etc.) in the writing?
5. Is the Quad Chart effective (e.g., does it do a good job of motivating the reader/listener to work or get more curious about the problem? is it effective in showing the main highlights of the work?)