CISC 372: INTRODUCTION TO PARALLEL PROGRAMMING
Fall 2006
Team Programming Assignment 4
Due Date: start of class, Tuesday, October 24, 2006

1 Objectives

The objective of this assignment is to write an MPI program that attempts to achieve load balancing for a problem for which a static data distribution is not appropriate.

2 The Team

You may work in teams of 2 people on this project if you like. Each member should be taking an equal role in the work. Each team member will be completing a peer review when the assignment is handed in, and each team member's grade will be a percentage of the assigned grade. If both members worked fairly equally together, then each will receive the same grade, which is the assigned grade. If one member worked significantly more than the other, that member will receive the assigned grade and the other member will receive some percentage of that grade.

3 Project Description

Your project consists of creating a parallel MPI spelling checker that seeks to achieve load balancing through dynamic data(work) distribution. That is, there is no fixed static data distribution scheme, but tasks are distributed at runtime as a worker completes a task and needs more work to do.

Basic Spelling Checker Operation:
The spelling checker that you are to implement will also be very simple. Your spelling checker will ask the user for a word to check, read in a dictionary of words from a dictionary file, and then dispatch searching tasks to the processors. The program should report if the word is spelled correctly. If the word is incorrect, then words in the dictionary that are within one mismatch of the word or within one gap of the word should be reported.

For example:

```
TOOO = word to check
TOOL = word within one match
TOO = word within one gap
AOOO = word within one match
```

You are in luck! A serial version of the spelling checker is located on porsche in `pollock/372porsche06/public/seqdict.c`. A dictionary file of over 25K words is there also, called `dictionary`, in the same directory. Look at the sequential code, and compile and execute it to see how it works.

A Parallel Version:
For your parallel version:

- Give the query word to all worker processes. Note: You are only searching for a single word at a time, not trying to parallelize searches for different queries at the same time.
- Research parallel searching methods or develop one on your own to use the multiple processes to speed up the search for this word in the dictionary. This parallel algorithm part is left to you to think about! The key is to get some load balancing occurring at runtime when you don't know how big each task is ahead of time.

- Print out a single message (not one from each process) indicating that the word was in the dictionary, or up to 10 close words if the word was not found in the dictionary. For simplicity, if the word does not match, you only need to keep track of no more than 10 close words.

4 What to Hand In

1. **Cover Page:** Title, author, course number and semester, date.
2. **Project Summary:** In one paragraph, describe the algorithm that you implemented for the parallel version.
3. **Program code:** Hardcopy of your parallel program. Your program should be tested on 1, 2, 4, 8, and 16 processes.
4. **Electronic Version:** Your parallel program should be emailed to the ta at sprenkle@cis.udel.edu.
5. **Timings:** Perform several experiments with your code. Be sure not to include the input and output as part of your timings. Be sure to take several timings for the same data point, but you only need to present the minimum of them after you are convinced you have a stable number for that data point. Time your code for 1, 2, 4, and 8 processes for a fixed task size given to each process. Create a table or graph that shows how the timings change as number of processes changes.

   Time your code for 8 processes for several different task sizes. Plot a graph or present a table that shows how the timings change as the task size changes. Make sure you include some extreme cases like task size of 1 and some very high number.
6. **Analysis:** In English, describe what your timing results show, and try to explain why they are that way. Discuss what you believe to be a reasonable task size for your parallel dictionary search program, based on your results.
7. **Division of Effort:** Describe what each team member contributed to the project and how the team functioned as a team in this project.

SPECIAL NOTE: This assignment requires a dictionary file of considerable length. For this, create a link to pollock/public/dictionary by ln -s pollock/public/dictionary dictionary. This dictionary has over 20,000 words, one per line. In order to avoid excess network traffic (and getting me in trouble by slowing down the system!), be sure to have only the master process reading from the dictionary. Do not have the workers all reading from the dictionary file.

5 Assessment Criteria:

- Correctness of Code:
  (58) interact with user, distribute query word, dispatch search tasks with runtime scheduling of tasks, processes report result, stop reading dictionary and sending tasks when word found, report if word spelled correctly, If the word is incorrect, then words in the dictionary that are within one mismatch of the word or within one gap of the word should be reported.

  (6) Print single message (not one from each process) indicating that word was found, or up to 10 close words if the word not found. For simplicity, if the word does not match, you only need to keep track of no more than 10 close words.

  (4) Time your code for 1, 2, 4, and 8 processes for a fixed task size given to each process.

  (4) Time your code for 8 processes for several different task sizes. Make sure you include some extreme cases like task size of 1 and some very high number.

- Quality of Code:
  (2) Cleverness and simplicity of solution
  (2) Generality of solution
  (2) Time Efficiency of solution
  (2) Space Efficiency of solution

- (Total: 20) Your experimental report:
  (1) Cover Page: Title, author, course number and semester, date. (5) Project Summary:
  In one paragraph, summarize the activities of the lab, and in particular, describe the algorithm that you implemented for the parallel version.

  (6) Collected Timing Data:
  Present your experimental data, in a clear graphical form. Plot a graph that shows how the timings change as number of processes changes for fixed task size. Plot a graph that shows how the timings change as the task size changes for 8 processes.

  (5) Analysis:
  In English, describe what your timing results show, and try to explain why they are that way. Discuss what you believe to be a reasonable task size for your parallel dictionary search program, based on your results.

  (2) Division of Effort.
  (1) All Materials handed in as required.