CPE419 - Lupo
Project 3 - report.pdf
Timothy Chu
Nathan Yee

Solution Methodology/Implementation Strategy:
Our implementation strategy for the TSP problem was to:

1. Parse all the required data into intermediate data structures (parallelized - see below).
2. Construct an adjacency matrix that indicated if there was a flight from one airport to another. The value inside the adjacency matrix was the distance between the airports, calculated by the Haversine formula (parallelized - see below).
3. Run a greedy algorithm to travel to all cities:
    a. Starting at SBP, select the lowest weight flight to next airport (in a city that we have not visited) and fly to that airport.
    b. If there was no flight available to a city we have not visited, we bent the constraints and just found an airport to a city we have not visited and “flew” there. We calculated the distance with the Haversine formula for these new routes.
    c. Run until all cities visited.

Parallelization Scheme/Tools:
Our parallelization scheme was to parallelize the haversine calculations, so that each thread would calculate the distances between the the airports. This worked out well but there was not a huge speed up, since most of the computation of this program was for the TSP algorithm. It was about 0.5 - 1 second of speedup. We also parallelized some sections of our code where we populated data structures.

We did not use any external libraries but we relied heavily on the use of maps and sets in C++. initially we wanted to implement our solution in C but found that C lacked a lot of key data structures that we wanted to use.

Personal Observations/Realizations:
We realized that the bulk of this program relied heavily on the TSP algorithm, which we could not find an easy way to parallelize. Our algorithm implemented a while loop that did not have a well defined number of iterations. This reminded us of the case talked about in class, where if you have 90% of the work that is not parallelizable, and 10% of the work is parallelizable, it doesn’t result in much speedup.

We also realized that if there are any problems with the pragma that it is very difficult to resolve unless we had more training on the subject. We were not able to offload our code to the Mic due to this error: “offload error: cannot find offload”. And after scanning the web and documentation we were able to find out that we could not offload C++ maps and sets to the Mic with the pragma and we had to change how our data was represented. After making these changes the

problem still existed. We spent a lot of time looking on the web and in the documentation, but we were unable to find a solution. This error still currently exists in our solution.

Difficulties:
The most difficult part was thinking of a good algorithm to choose the airports. We went with a greedy approach and this worked only up till there was a dead end where the airport that we were at had no outgoing flights to a city that we had not been to. We had some ideas to try and avoid these dead ends but ultimately the problem was not resolved. In the end we decided that we would blackmail the airline company and have them create routes to cities that we did not visit.

The other difficult we came across was finding a way to parallelize the TSP algorithm that we had wrote. We came to the decision that there was no easy way to do it since it was hard to keep track of which thread was at which airport. In addition, there was no defined loops that we could parallelize, since most of the work was done in a while loop.

The last difficulty that we came across was implementing the offload pragma so that the computation would be executed on the mic. The error that we continued to encounter was: “offload error: cannot find offload”. We were never able to resolve this error and in our current implementation it still exist. Removing this pragma line will not offload the code to the Mic but instead run the parallel code on the CPU. The code that we wanted to offload was the haversine calculations for all the routes but as I mentioned we ran into an error that we could not fix.

For Next Time:
For next time we may want to have more practice on implementing the offload pragma since we ran into errors trying to implement it and never figured out what these errors were and what caused them. There was also a lack of web resources since the pragma offload is intel icc specific and it is a licensed product so there is lack of user community.