1 Methods and Techniques

The bag of tasks implementation created for this report uses a *pull* protocol: Once a client starts, it first sends a request to the server for a new task. The server sends tasks in predefined sized *chunks*. It includes a kill signal for the client to exit in the chunk once it has run out of tasks to send.

This report will present an analysis of process speedup given various chunk sizes and processor counts. The times were measured from when the server process starts until it exits. All timings were measured using the functions defined in the *utilities.h* file on the Ptolemy cluster.

2 Analysis

A rudimentary but useful way for predicting parallel program speedup is Amdahl's Law, which provides the following formula for speedup.

$$S(p) = \frac{1}{1 - f + \frac{f}{p}} \tag{1}$$

In this case, f is the portion of the program that can be parallelized and p is the number of processors.

Predicting the exact percentage of the program that is parallelizable involves a lot of consideration. For example, the server has to perform relatively slow IO operations every time it sends a new chunk to a client, and it has to do more IO when it logs solutions it receives. The depth first searching used to find solutions is also inherently serial.

Considering these overheads, say only 70% of the program is truly parallelizable. In this case, $f = 0.7$ making the maximum possible speedup $S = 3$.

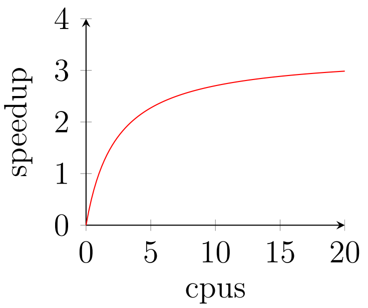

Figure 1: Program speedup prediction using Amdahl's Law

2.1 Chunk Size Consideration

Another overhead that will effect the speedup of the program is the communication overhead. This implementation does not use non-blocking communication functions, so time spent waiting for a communication to finish is idle time.

$$t_{comm} = t_s + mt_w \tag{2}$$

3 Results

4 Synthesis