CS267 ASSIGNMENT 3: PARALLELIZE KNAPSACK

CHUN MING CHIN, CHRIS MELGAARD, VIRAJ KULKARNI

1. Comparing Answers

To ensure we get the correct answer, like the provided serial and fixed parallel codes, we save the outputs of the used, weight, value and total arrays and compare the results with the original serial version. The save function is obtained by modifying source code from homework 2.

In order to compare answers accurately, we changed the srand function to have a constant seed (i.e. stand48(1000) so we can use our serial results to compare with our ups results. In addition, for the initialization stage in the knapsack.upc implementation, we use a for loop on MYTHREAD==1 instead of the upc_forall loop to iterate on the lrand48() operations for the weight and value arrays.

We do blocking operations on rows of T (i.e. operating on contiguous blocks of memory) so as to increase cache hits. This would minimize communication time, hence making the code run faster.

Next, we also considered pipelining our computations, because an object, j, is only dependent on its previous object j-1.

2. Results

The time required for serial implementation is 0.031931 The time required for upc implementation is 11.4287, 10.9262 Time required is reduced to 10.2737 when 2 upc_forall loops are replaced by for loops

When the number of threads is 4 for a capacity of 1000, the block size is 250 When the number of threads is 8 for a capacity of 1000, the block size is 125 When the number of threads is 16 for a capacity of 1000, the block size is 63 When the number of threads is 32 for a capacity of 1000, the block size is 31