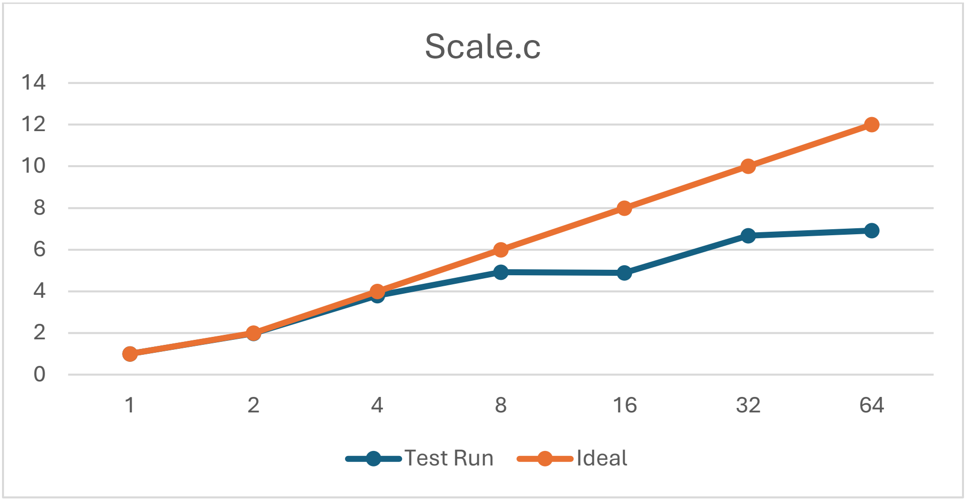
Hector Guzman – ole10

HW1: Performance Evaluation – Digging Deeper

1. Graph with max threads @ 64 | Baseline = 758.21ms

   Threads: (Given 1 Thread is at Baseline)

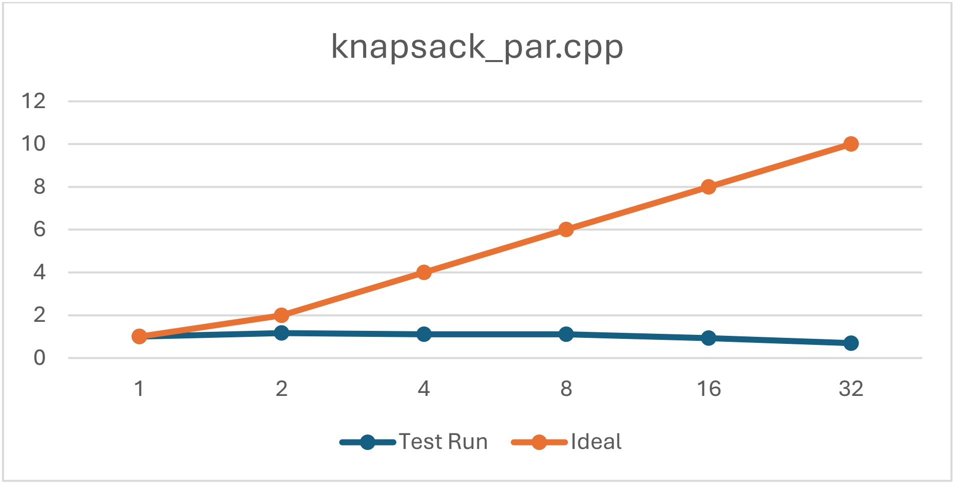
@ 2:	383.85ms	@ 4:	199.39ms	@ 8:	154.37ms
@ 16:	154.98ms	@32:	113.59ms	@64:	109.66ms



Graph with max threads @ 32 | Baseline = 8.608s

Threads: (Given 1 Thread is at Baseline)

@2:	6.433s	@4:	5.514s	@8:	5.804s
@16:	6.865s	@32:	9.213s		



Hector Guzman – ole10

a. The performance scalability of “scale.c” has a good trend following the ideal perfect scalability. However, the performance scalability of the knapsack with parallelism has a negative scalability.
b. To do parallelism properly, the program needs to inherently be built to handle it. Simply throwing parallelism even in a remedial attempt does nothing if the base code is not fast enough. An analogy for this would be that delegating too many tasks when there are not enough tasks to properly delegate causes more harm than good. This is the difference between “knapsack_par” and “scale.c”.

2. Time

	Real	User	System
Baseline	7.51967	7.33467	0.184
OPT	1.591	1.411	0.17867
OPT_MAX	1.09367	0.91567	0.17767

Instructions

	Instructions	Cache-ref	Branches	LLC-Loads	LLC-Stores
Baseline	38,127,901,816	13,650,372,649	2,962,276,427	1,071,320	2,964,889,115
OPT	12,519,110,759	4,206,904,003	1,981,700,839	525,261	158,694,402
OPT_MAX	9,289,911,841	2,510,448,383	1,323,015,752	599,484	151,724,854

a. The speed up from knapsack_baseline to knapsack_opt is over 5.19x.

   The speed up from knapsack_opt to knapsack_opt_max is over 1.54x.

   The speed up is as I would expect, from the previous assignment I saw that the time difference between optimization levels O1 and O3 are minimal.

b. Total instructions from knapsack_baseline to knapsack_opt are roughly 38Billion to 12Billion. When calculated from the chart I get a 304.55% reduction in instructions at two decimal places.

   From knapsack_opt to knapsack_opt_max, the instructions are roughly 12Billion down to 9Billion. That percentage translates to 134.76% reduction in instructions at two decimal places.

Hector Guzman – ole10

The reduction in instructions is roughly proportional to the speed ups, however the initial speedup has a magnitude of 5x meanwhile the magnitude of the instruction reduction has a magnitude of 3x which shows a difference of 2x. Meanwhile the second optimization level in the speedups and instruction magnitudes are similar. The discrepancy in the initial optimization would be because some instructions take longer time than others, therefore the first optimization level has a bigger impact on time because there are instructions that take a long time that are being reduced. However, that does not translate to the overall instruction count going down, hence the difference.