Michael Newton + Omer Syed

Professor Daniels

CSC 411

12/06/2023

Profiling Laboratory Notes

Benchmark	Time* (sec)	Instructions	Rel to Start	Rel to Prev	Improvement
Sandmark	10.3 s	$2.11 * 10^9$	1.000	1.000	None (Starting point)
Midmark	0.43 s	$8.51 * 10^7$	1.000	1.000	
Sandmark	9.62 s	$2.11 * 10^9$	0.934	0.934	Changed code so that ra, rb, rc, rl, and vl were not all instantiated at the start of each loop, but were instead instantiated when the match statement needed them. (ra, rb, and rc were not needed for Load Value, and rl and vl were not needed for add.)
Midmark	0.41 s	$8.51 * 10^7$	0.953	0.953	
Sandmark	10.13 s	$2.11 * 10^9$	0.983	1.053	Attempted to in-line a UM built-in function “get”, but it slowed down overall performance on Sandmark. Reverting.
Midmark	0.41 s	$8.51 * 10^7$	0.953	1.000	
Sandmark	9.094 s	$2.11 * 10^9$	0.883	0.898	Made the “get” function use unsafe rust so that index checking was not performed.
Midmark	0.367 s	$8.51 * 10^7$	0.853	0.895	
Sandmark	8.883 s	$2.11 * 10^9$	0.862	0.977	Made the “set” function unsafe rust like “get” in the previous step.
Midmark	0.358 s	$8.51 * 10^7$	0.832	0.975	
Sandmark	8.771 s	$2.11 * 10^9$	0.851	0.987	Implemented Wrapping Add and Wrapping Mul to the program.
Midmark	0.354 s	$8.51 * 10^7$	0.823	0.989	
Sandmark	9.551 s	$2.11 * 10^9$	0.927	1.089	Instead of using Vec::new() on unmap, we tried using

Midmark	0.383 s	$8.51 * 10^7$	0.890	1.082	".clear();". This slowed performance down quite significantly. Reverting.
Sandmark	8.689 s	$2.11 * 10^9$	0.861	0.910	Removed instruction counter code from the benchmarking.
Midmark	0.351 s	$8.51 * 10^7$	0.816	0.916	
Sandmark	8.376 s	$2.11 * 10^9$	0.813	0.964	Compiled with LTO
Midmark	0.341 s	$8.51 * 10^7$	0.793	0.972	

*Time based on local machine run-time, not gradescope compiler time