Performance Gain vs. MSRP of x86 and ARM MacBook Pros

CS 3339 Sec 251: Computer Architecture

Carson Holland, Chris Le, Robert Bonham

Git Link: https://github.com/Cnh1116/Computer-Architecture-Project.git

Abstract

In recent years, Apple has moved nearly all its desktop and laptop chips to in-house ARM architecture. The Apple Mac Pro Tower is one of the few devices that has not made the transition to ARM. With the move to an in-house ARM processor, Apple has touted incredible performance gains over previous generations of devices. However, this has not resulted in any significant price reductions for identical Intel or ARM Macs. Companies typically reduce the price of their products when parts of their component manufacturing and design are moved in-house, due to greater control of the development process. The goal of this project is to test the performance of three different generations of MacBook Pros using three different tests; scale.c and ranksort.c for raw data CPU performance and a Minecraft test for testing the CPU in a CPU-intensive application. These performance tests will provide data that will either support or question Apple's decision to produce its own Mac processors. This data will also show what if any improvement is made in processing power a year after the ARM transition is made.

Methodology

Two main testing schemes were used to test the performance of the three MacBooks. The first was a standard measure of the CPU's single and multi-threaded performance using two different benchmark programs. The first was *scale.c*, a program that accepts *N* amount of data items, *M* number of threads, and an input file that contains random values. *scale.c* utilizes the same instruction multiple data(SIMD) parallelism by scaling *N* amount of data items by the number 17, making it a great test for multi-thread performance. The second program was a parallel implementation of the rank sort algorithm: *ranksort.c*. Similar to *scale.c*, this program accepts *N* amount of data items and *M* number of threads. The sorting algorithm assigns values via a random number generator, so no input file is necessary. This program was implemented using Pragmas to take advantage of multiple threads, making this another great choice to test multi-threaded performance.

BenchMark Testing

Scale.c Data

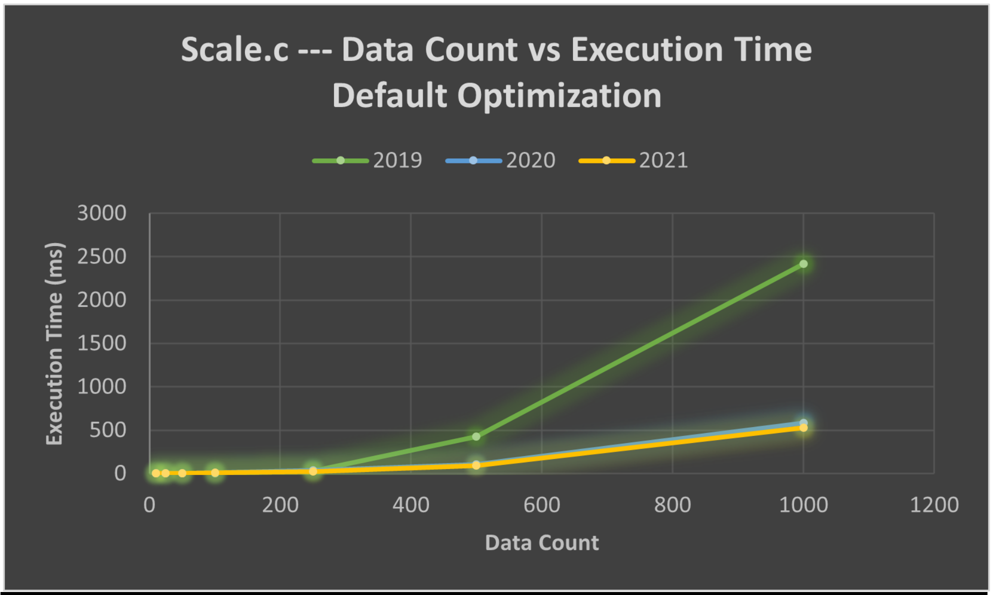

Figure 1: Single thread performance

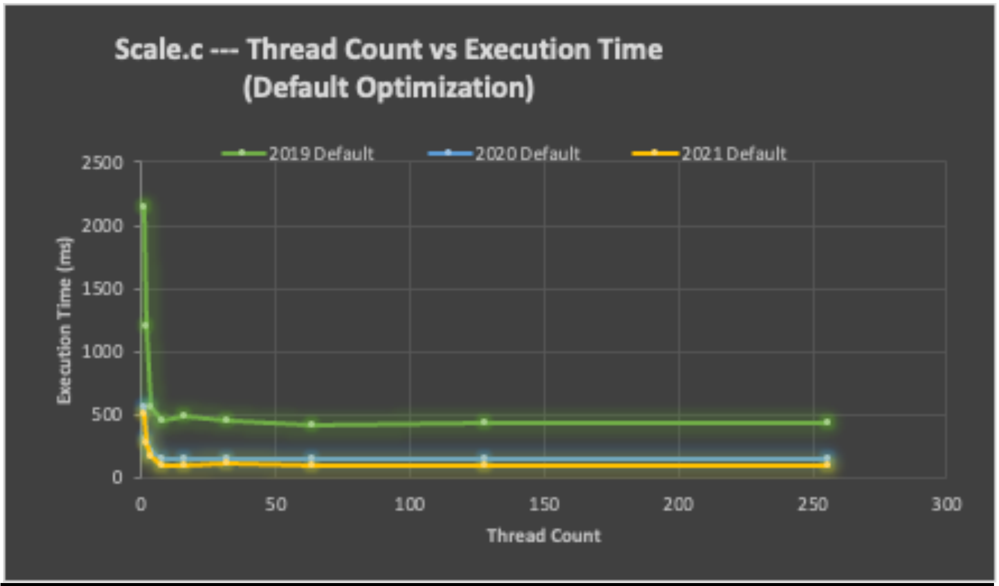

Figure 2: Multithreaded performance

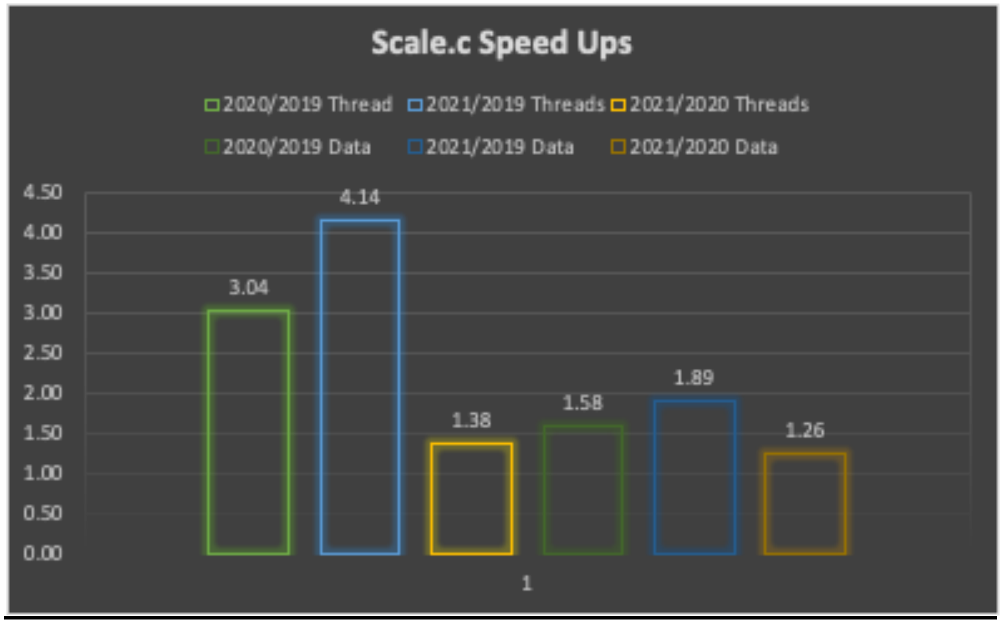

Figure 3: Speedup

Ranksort.c Data

Multithreading Efficiency		
Thread Count (1,000 Data)	2020	2021
1	164.89ms	103.41ms
2	163.24ms	95.65ms
4	167.51ms	102.22ms
8	168.68ms	104.55ms
16	160.47ms	91.52ms
32	164.72ms	101.14ms
64	166.30ms	95.52ms
128	162.57ms	106.89ms
256	168.56ms	98.80ms

Single Thread Efficiency		
Number of Data Items (Thread Count 1)	2020	2021
10	1.79ms	0.97ms
25	1.34ms	1.66ms
50	1.75ms	1.85ms
100	4.93ms	5.61ms
250	11.96ms	20.53ms
500	40.29ms	27.03ms
1000	164.20ms	108.35ms

*Note: Robert was able to create the object file for the Rank Sort file on the 2020 Mac using the default optimization, but the object file would not be created with the same terminal command on the 2019 or 2021 Mac. The 2020 Mac was also unable to create a new object file in subsequent attempts. Due to this unknown issue, we ran the first object file Robert created on all 3 Macs, only at the default optimization level. However, another peculiar issue arose where the object file could not be used on the 2019 Mac, likely due to the object file being created on a different architecture. Due to all this, there is data only for the default optimization on the 2020 and 2021 Macs for the Ranksort test.

Minecraft Test

Why Minecraft? Looking at the main difference between 2019, 2020, and 2021 MacBook; the biggest difference between the three is the switch from an x86 processor to an ARM processor, a decision made by Apple themselves. So, we wanted to find a way to push the CPU on each laptop. That is where Minecraft comes into play. Minecraft prioritizes an intense dependency on CPU usage more than GPU, which typical games tend to do. This is because Minecraft is a procedure-generating game, with each new world being distinctly unique. The further you go out, the more you are creating new **chucks**. Chucks are sections of the world that are broken into parts that make it easier for the game to create and load the world. This is where the CPU dependency comes into play; the world is created dynamically with each newly generated chuck. So, how do we create a testing environment that will give us reliable results? Well, Minecraft gives us a way around that, called a **seed.** Think of it as a blueprint to create the world again and again for multiple tests, which will allow us to mitigate the number of uncontrolled variables when performing tests.

The first big test we did was something any beginner Minecraft player loves to do, seeing how many TNT explosions we can set off until the game completely "breaks". A break is used loosely as the most that will happen is the game will lock up and you are unable to move at all until the CPU is able to catch up. That is the main idea for the Minecraft test. How much TNT would we need until we saw a drastic increase in CPU percentages that will cause the game to freeze. First, we started off small. We did a 10x10x10 cube of TNT, then moved on to a 15x15x15 cube, until finally a 20x20x20 cube. However, to make sure we produced enough destruction coming from our cube, we went down to the same depth level as the cube. So, if our cube was n blocks high, we went down n blocks deep.

The ARM MacBook Pros are created with 2 efficiency cores + 6/8 performance cores(6 cores for 2020 and 8 cores for 2021). The difference in CPU utilization between the 2020 and 2021 ARM MacBooks ranges between .12% - 59.7%. In all three tests, there were no scenarios where more than 4 cores were fully utilized, regardless of the MacBook model. An interesting note about the Minecraft tests is that it seems the 2 efficiency cores on both MacBooks were used first, followed by an additional 2 performance cores when additional CPU performance was needed. With the exception of test #3, the 2020 MacBook initially started utilizing the CPU more than the 2021 MacBook. Another interesting occurrence was the expected CPU utilization for larger TNT blocks. CPU utilization increased from test 1 to test 2. However, test 3 saw lower CPU utilization for both ARM Macs compared to the previous two tests.

Minecraft Test Data

Test #1 10x10x10 TNT	CPU Utilization		
MP 2020	Initial: 83.60%	Mid: 161.90%	End: 443.90%
MP 2021	Initial: 68.80%	Mid: 159.90%	End: 471.70%
Test #2 15x15x15 TNT			
MP 2020	Initial: 71.54%	Mid: 222.63%	End: 473.46%
MP 2021	Initial: 64.60%	Mid: 215.54%	End: 468.70%
Test #3 20x20x20 TNT			
MP 2020	Initial: 55.70%	Mid: 259.70%	End: 450.00%
MP 2021	Initial: 71.80%	Mid: 250.00%	End: 450.12%

*Note: Every 100% for the CPU utilization is equivalent to 1 CPU core. ie. 200%= 2cores.
*450% and 450.12% are essentially the same since CPU utilization is not refreshed every second.

Cost Analysis

The two ARM Macs perform similarly to each other, with the 2021 Mac performing slightly better in multi-threaded tests as depicted in Fig. 1 & 2. This is likely due to the higher core count. However, the ARM architecture has a clear advantage in both single-thread and multi-thread performance over the x86 architecture. Fig. 3 in the benchmark testing section shows the speedup comparison of the 3 different MacBooks used in this project. Both the 2020 and 2021 MacBooks show significant speedup over the 2019 MacBook in multithreading performance. There is some improvement in speedup when comparing 2020 to the 2021 MacBook, but nowhere near the improvement over the 2019 MacBook. The speedup for the thread efficiency tests also improves when comparing x86 to ARM, but again, the speedup from the 2020 to 2021 MacBooks is relatively minor.

In addition to the benchmark testing, the Minecraft tests performed indicate the performance gap between the two ARM MacBooks is relatively close, in terms of CPU utilization. This indicates that there is a real benefit to choosing an ARM Mac over an x86 Mac, especially since the 2020 Mac costs the same as the 2019 Mac. However, the 2021 Mac costs significantly more than the 2019 and 2020 with minimal performance enhancements over the 2020 Mac. However, there are a few key aspects to keep in mind. This project does not cover every type of performance metric and does not take into account I/O or aesthetics, both of which can play a major role in a purchasing decision. In conclusion, Apple made the right choice to switch from an x86 Intel processor to an in-house ARM processor. The decision for which ARM Mac to choose rests solely on the shoulders of the consumer, as each consumer knows which aspects of the MacBook are most important for their needs.