Arm is taking over

(your server and desktop)

Chris Mair - 2022-02-03 - NOI Techpark Developers' Thursdays

Arm, the company

- Arm is a British semiconductor design company based in Cambridge, England
  - 1983: ARM first appeared as Acorn RISC Machine, a CPU by Acorn Computers
  - 1990: the company incorporated as Advanced RISC Machines Ltd.
  - 1998: IPO as Arm Ltd. (Arm now is apparently not an acronym anymore)
- owned by SoftBank Group (who tried to sell it to Nvidia…)
- market dominance in processor designs for mobile phones and tablets
- 2.0E11 chips based on Arm's designs [1]

ARM, the architecture

(a.k.a. instruction set)

- Arm designs the architecture and licenses it to other companies who use it as a basis for their (more or less custom) chips - some well known examples:
- all iPhones, iPods and iPads (Apple)
- Switch, 3ds (Nintendo)
- the Exynos SoCs used in various mobile products by Samsung
- Qualcomm's Snapdragon SoC used in a lot of mobile phones from different brands
- networking devices by Marvell, Broadcom and a lot of others
- the ever popular Raspberry Pi family

Intermezzo: Nostalgia

(the Acorn Archimedes)

https://www.youtube.com/watch?v=MNXypBxNGMo

Instruction sets (assembly example)

x86-64 (Intel + AMD)

```
.L25:
    mov edi, ebx
    call    _Z7isPrimei.part.1
    cmp al, 1
    sbb ebp, -1
    add ebx, 2
    cmp ebx, 10000001
    jne .L25
    call    _ZNSt6chrono3_V212steady_clock3nowEv@PLT
    pxor    xmm0, xmm0
    mov edi, 10
    sub rax, r12
    cvtsi2sdq   xmm0, rax
    mulsd   xmm0, QWORD PTR .LC4[rip]
    movsd   QWORD PTR 8[rsp], xmm0
    call    putchar@PLT
    movsd   xmm0, QWORD PTR 8[rsp]
    mov edx, ebp
    mov esi, 10000000
    lea rdi, .LC5[rip]
    mov eax, 1
    call    printf@PLT
    mov edi, 10
    call    putchar@PLT
```

aarch64 (ARM)

```
.L24:
    mov w0, w19
    add w19, w19, 2
    bl  _Z7isPrimei.part.1
    tst w0, 255
    sub w1, w19, #9998336
    cinc    w20, w20, ne
    subs    w1, w1, #1665
    bne .L24
    bl  _ZNSt6chrono3_V212steady_clock3nowEv
    sub x21, x0, x21
    adrp    x0, .LC1
    scvtf   d8, x21
    ldr d0, [x0, #:lo12:.LC1]
    mov w0, 10
    fmul    d8, d8, d0
    bl  putchar
    fmov    d0, d8
    mov w2, w20
    mov w1, 38528
    movk    w1, 0x98, lsl 16
    adrp    x0, .LC0
    add x0, x0, :lo12:.LC0
    bl  printf
```

ARM - the (low-power) world is not enough...

server CPUs are coming
(and this time they compete)

(0/12022 ix magazine cover [2])

ARM - the (low-power) world is not enough...

The 2020 Mac Mini Unleashed: Putting Apple Silicon M1 To The Test

682 Comments

+ Add A Comment

by Andrei Frumusanu on November 17, 2020 9:00 AM EST

Posted in Apple | Mac | Mac mini | Arm | SoCs | Apple Silicon | Apple M1

APPLE SILICON M1: RECAP, POWER CONSUMPTION

Last week, Apple made industry news by announcing new Mac products based upon the company's new Apple Silicon M1 SoC chip, marking the first move of a planned 2-year roadmap to transition over from Intel-based x86 CPUs to the company's own in-house designed microprocessors running on the Arm instruction set.

desktop CPUs are coming
(and this time they impress)

(Anandtech article screenshot taken 3 Feb 2022 [3])

Server example: Graviton 2

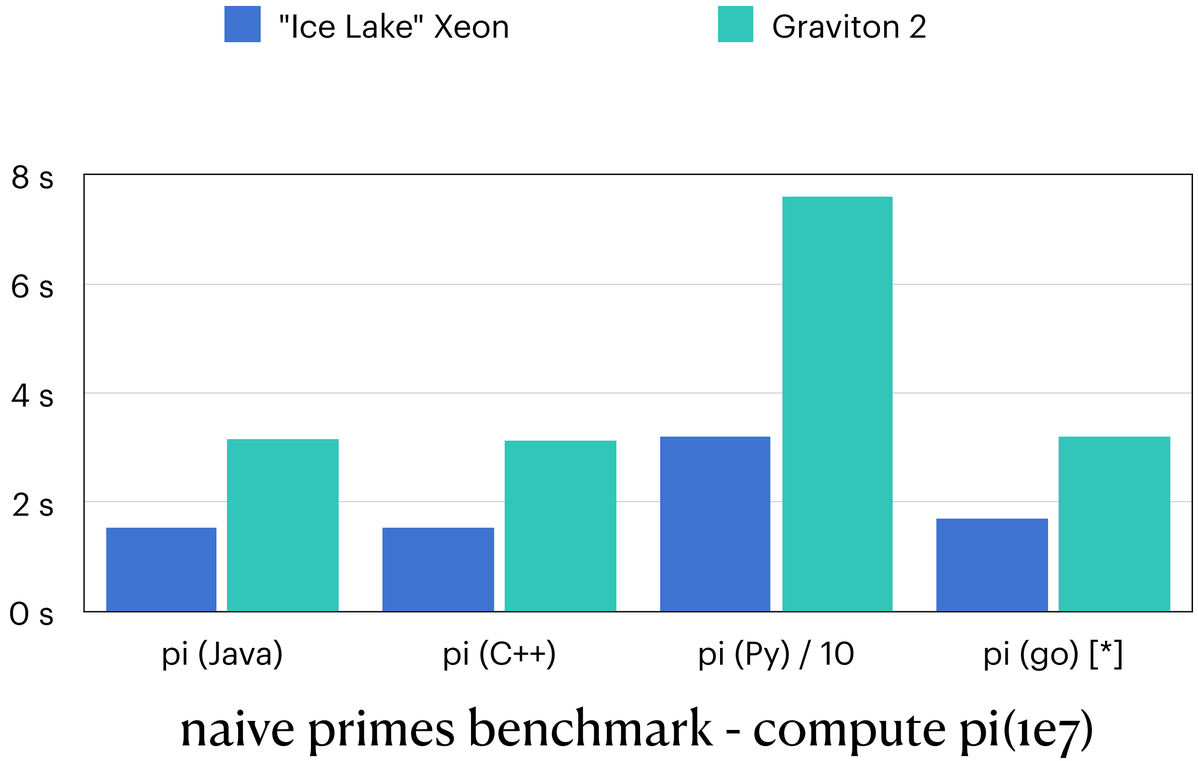
- Graviton 2 (by AWS)
- ARM v8.2/Neoverse-N1 - 64 cores at 2.5 GHz
- g.a. since May 2020
- Anandtech tested Graviton 2 in March 2020 against the Intel Xeon Platinum 82xx (2-nd gen. “Cascade Lake” and the AMD Epyc 7xx1 (1-st gen “Naples”) [4]
- Anandtech found the Graviton 2 can compete against these top offerings by Intel and AMD in performance and easily beats them in cost/performance
- things are moving quickly, however: Xeon 83xx (“Ice Lake”), Epyc 7xx2 (“Rome”), Epyc 7xx3 (“Milan”) are available and Graviton 3 is announced

Benchmarks: Graviton 2 vs. “Ice Lake” Xeon

- some tests on:
  - m6i.4xlarge (16/128 threads of a Xeon Platinum 3rd gen.“Ice Lake” 8375C @ 2.90GHz) vs.
  - m6g.4xlarge (16/64 threads of a Graviton 2 @ 2.50 GHz)
- fairness notes:
  - the “Ice Lake” Xeon is newer (m6g is g.a. May 2020 and m6i is g.a. August 2021)
  - for the Graviton 2 each thread is a core, whereas the Xeon has 2 threads per core
- costs:
  - m6i.4xlarge is $0.768 /h vs m6g.4xlarge is $0.616 / h

Benchmarks: Graviton 2 vs. “Ice Lake” Xeon

(run on Debian 11.2 for both, all compilers from default repo, Python 3.9.7 from condaforge)



naive primes benchmark - compute pi(1e7)

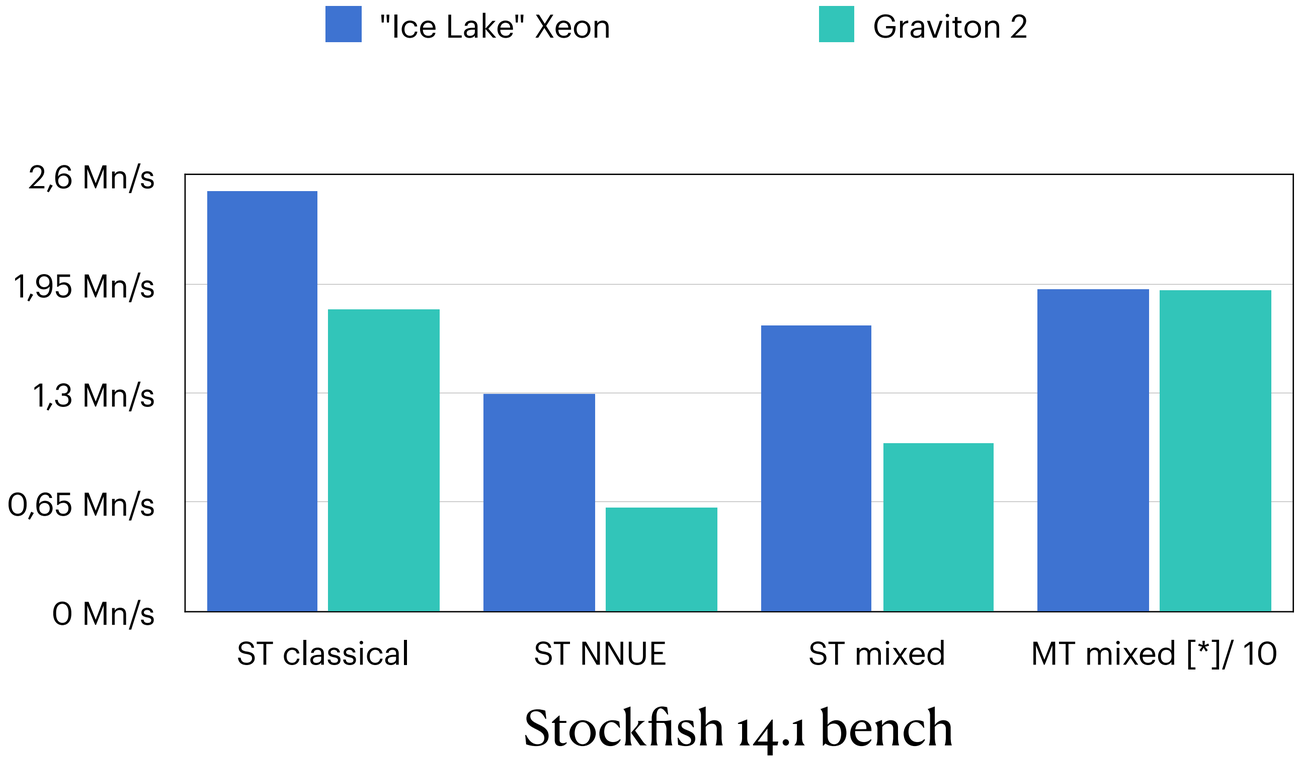
takeaways:

- in this strictly ST benchmark, the (**newer**) Xeon wins by a factor ~ 2
- Python’s interpreted nature really shows (note the scaling /10) :)

[*] An earlier version of this slide had a wrong number for the Xeon.

Benchmarks: Graviton 2 vs. “Ice Lake” Xeon

(run an Debian 11.2 for both, Stockfish 14.1 compiled with g++ 10.2.1, best options for each CPU)



Stockfish 14.1 bench

takeaways:

- Graviton 2 is weakest for NNUE (uses FP), but is closer to keeping up with classical (bit/byte fiddling)
- in the MT run Graviton 2 is equal, probably due to the 16 “real” cores vs. 16 hyper threads

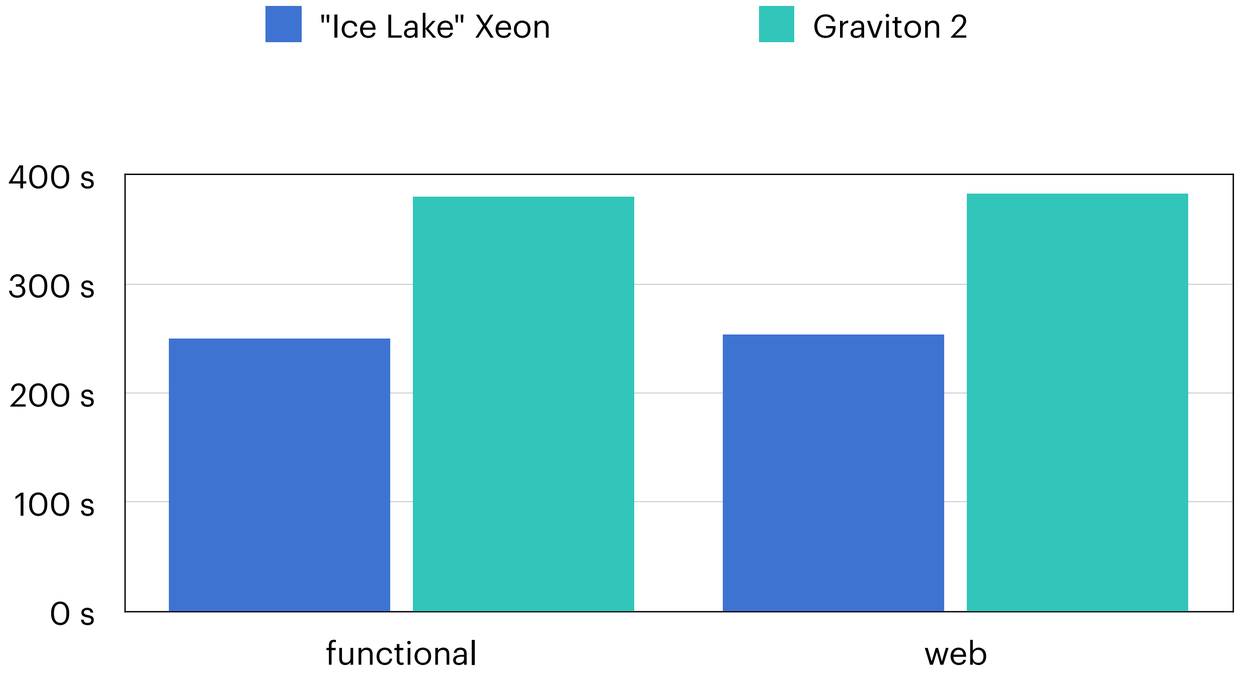
[*] Interestingly, I measured better-than-linear scaling for the mixed benchmark for the Graviton 2:

```
bench 16 1 13 default depth mixed      -> ~ 1.0 Mn/s
bench 16 16 13 default depth mixed     -> ~ 19 Mn/s
```

I retested this on a fresh instance and it came out the same.

Benchmarks: Graviton 2 vs. “Ice Lake” Xeon

(run an Debian 11.2 for both, OpenJDK 11 from default repo)



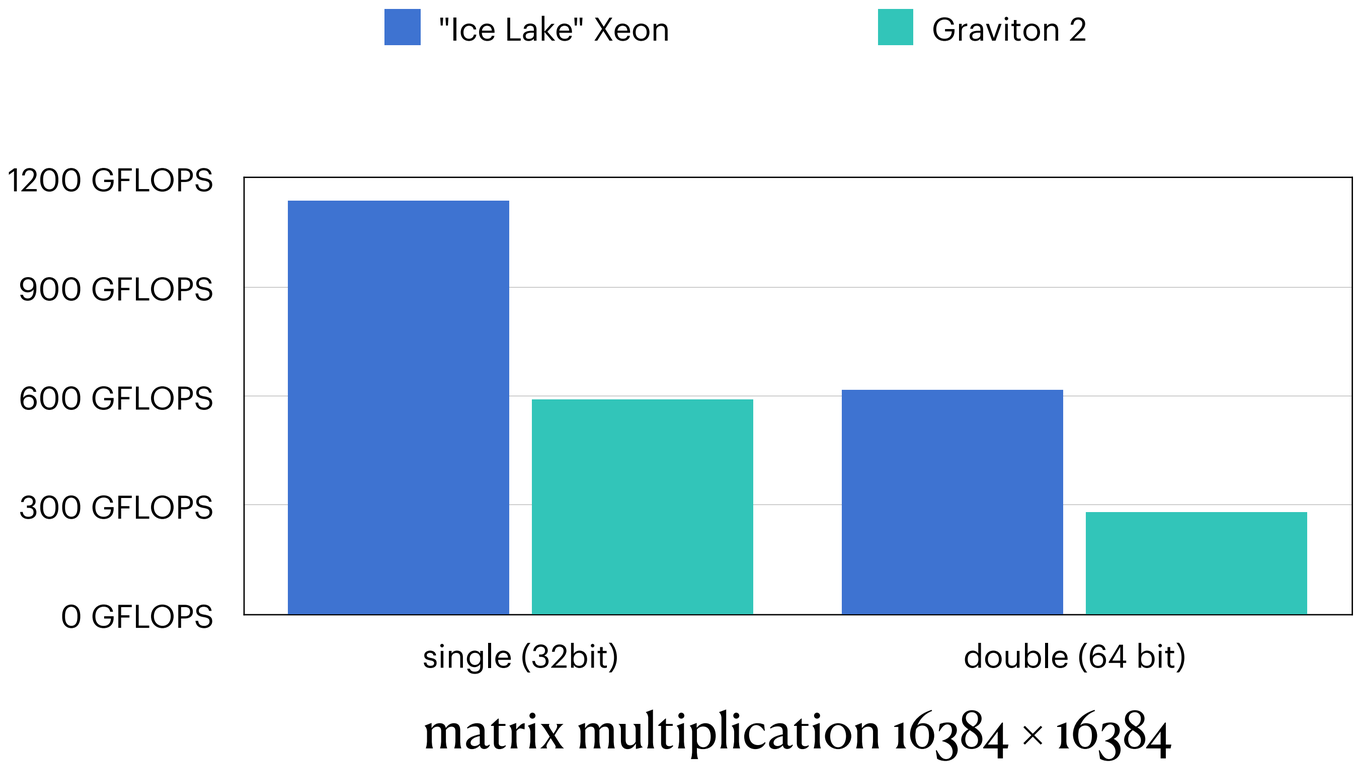
renaissance.dev Java benchmark 0.13

takeaways:

- Graviton 2 gets 2/3 the performance here (at 80% of the cost), this benchmark uses 2 to 4-ish threads, a higher thread count would likely close the gap...

Benchmarks: Graviton 2 vs. “Ice Lake” Xeon

(run an Debian 11.2 for both, Python 3.9.7 and numpy 1.22.0 from condaforge)



matrix multiplication 16384 × 16384

takeaways:

- this is a MT benchmark using all cores - “Ice Lake”s FP-strength (with AVX512) shows and we get again a factor ~2

Benchmarks: Graviton 2 vs. “Ice Lake” Xeon

overall takeaways:

- Graviton 2 comes close to compete in multi-threaded non-floating-point loads
- don’t forget that the 3rd gen. “Ice Lake” Xeon is 0.5-1 generations ahead, Graviton 3 is already in closed beta since Nov 2021 [5]
- likely we will see the two families of chips compete generation by generation in the near future
- sorry for not testing AMD; I expect a similar story there, or even better for AMD: since Anandtech’s test, mentioned before, Epyc is already **two** generations further with Epyc 7xx3 “Milan”

Mobile / small form factor example: Apple M1

- M1: ARM SoC: CPU 4+4 cores, max 3.2 GHz, GPU 8-core, 8 GB RAM
- g.a. since November 2020
- high performance at low power consumption, partly due to TSMC's 5 nm process
- this CPU beat everything by AMD and Intel in the mobile space at that time
- things are moving quickly, here to: M1-Pro/Max (8+2 cores, g.a. October 2021) and Intel Core-i9 12-gen "Alder Lake" (8 + 6 cores, g.a. end 2021)
- latest benchmarks confirm laptop class "Alder Lake" Core i9 12xxxH series outperforms M1 Max at the cost of considerable more power usage (40W M1 Max vs 100W "Alder Lake") [6]

Benchmarks: M1 vs 8-th gen. Core i7

- low-end Mac mini (“Macmini9,1”), released Nov. 2020,

  currently sold at 819 EUR

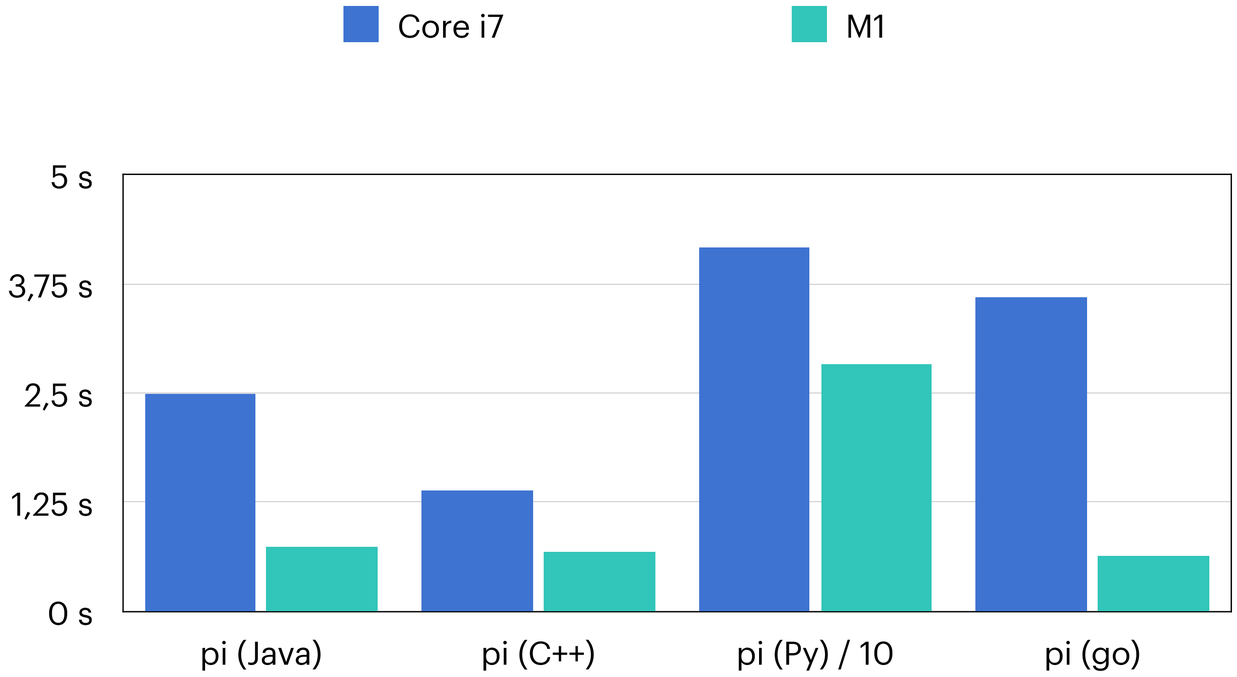
    (M1 CPU 4+4 cores max 3.2 GHz, GPU 8-core, 8 GB RAM, 256 GB disk)
- high-end Mac mini (“Macmini8,1”), released Oct. 2018,

  currently sold at 2289 EUR

    (i7-8700B 6 cores 3.2 GHz max 4.60 GHz, 32 GB RAM, 512 GB EBS vol.)
- i7 is older (8-th gen), but Intel made the huge jump only with Alder Lake (gen 12), and this i7-mini is still the highest configuration sold in the mini line

Benchmarks: M1 vs 8-th gen. Core i7

(run an macOS 12.1, Clang 13, OpenJDK 17, Python 3.9.7, go 1.17)

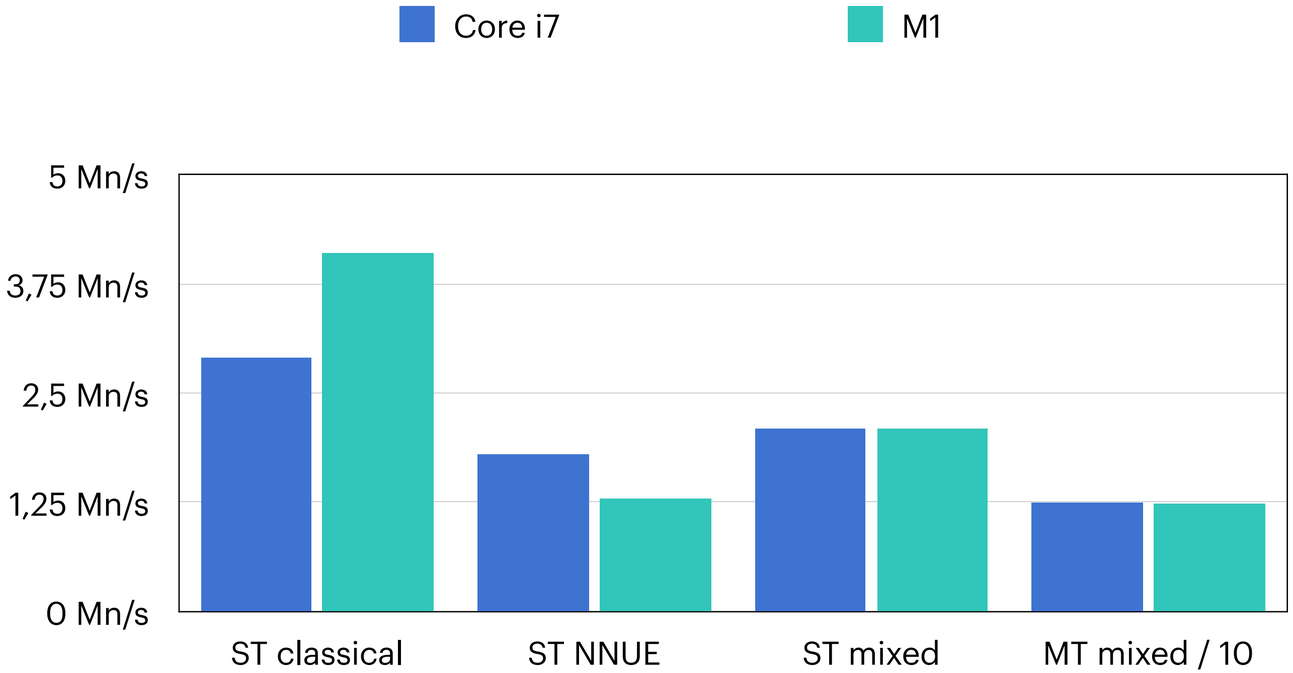


naive primes benchmark - compute pi(1e7)

takeaway:

Benchmarks: M1 vs 8-th gen. Core i7

(run an macOS 12.1, Stockfish 14.1 compiled with Clang 13, best options for each CPU)



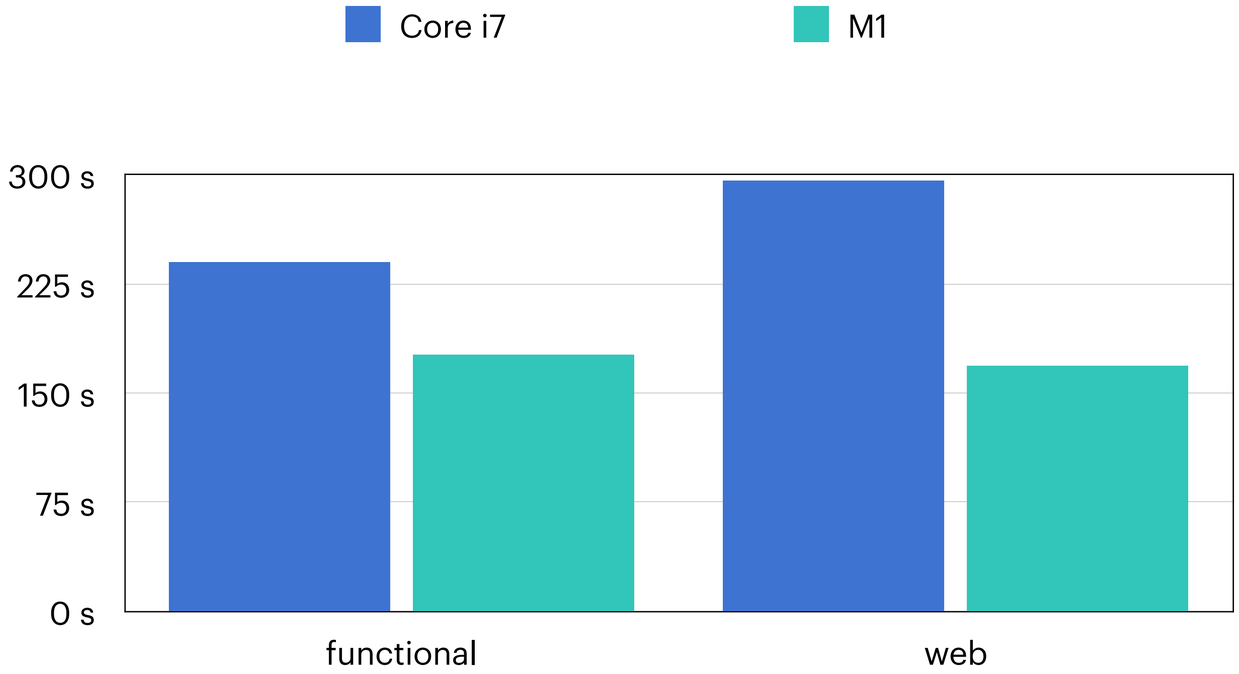
Stockfish 14.1 bench

takeaways:

- the Core i7 wins for ST NNUE (uses FP), and the M1 wins for ST classical (bit/byte fiddling)
- for mixed loads and in the MT run both chips reach the same performance, however... the max. package during the MT run was:
  75 W for the Core i7 vs.
  17 W for the M1

Benchmarks: M1 vs 8-th gen. Core i7

(run an macOS 12.1, OpenJDK 17 from Homebrew)



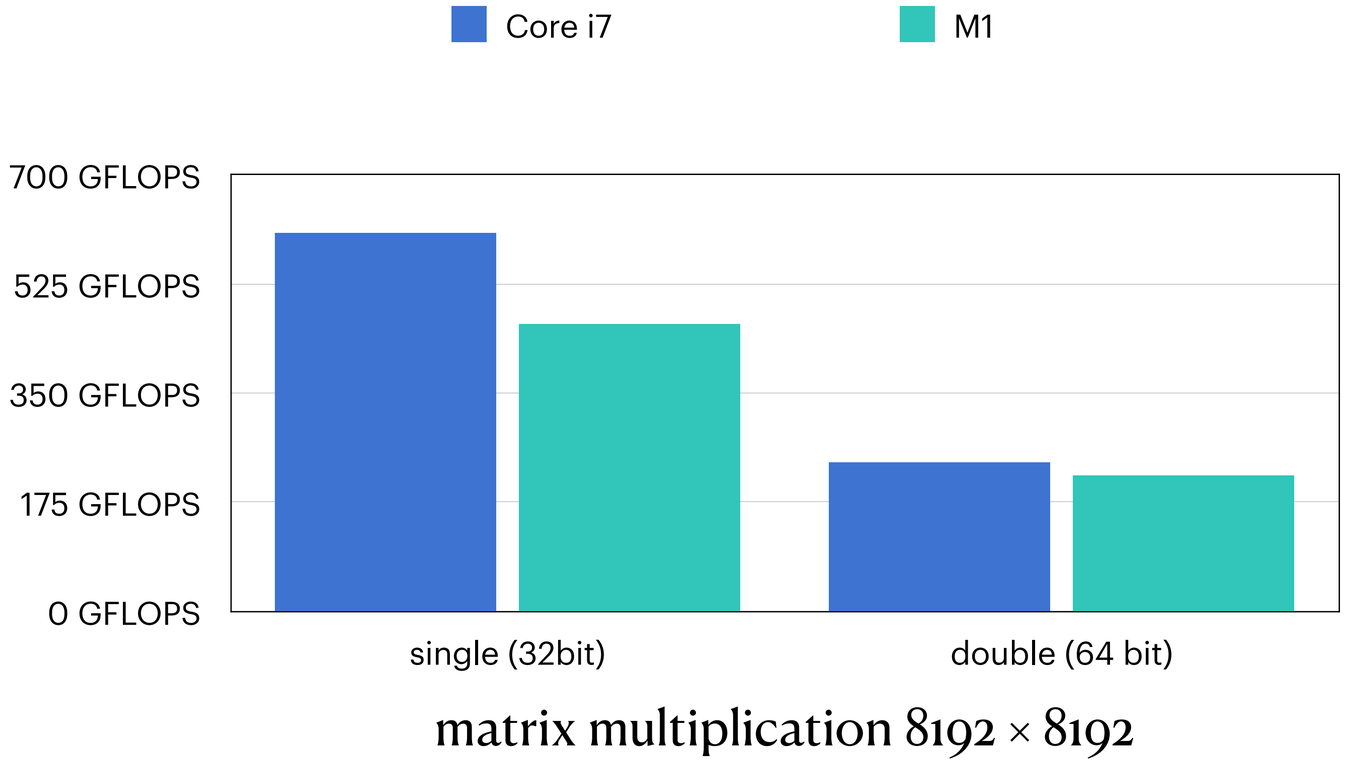
renaissance.dev Java benchmark 0.13

takeaways:

- the M1 likes Java, apparently

Benchmarks: M1 vs 8-th gen. Core i7

(run an macOS 12.1, Python 3.9.7, numpy 1.19.5 from condaforge)



matrix multiplication 8192 × 8192

takeaways:

- this is a MT benchmark using all cores - again Intel shows its FP-strength, in this case with “only” AVX2

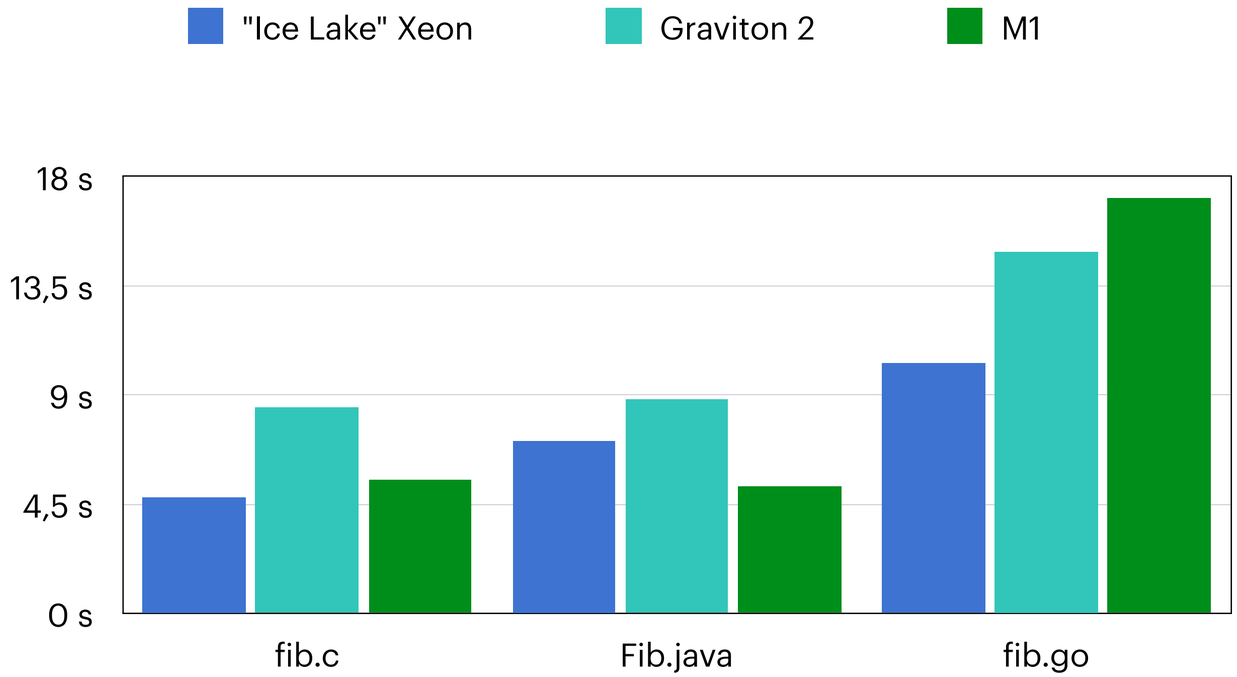
Benchmarks: M1 vs 8-th gen. Core i7

overall takeaways:

- M1 is really strong, especially in non-floating-point loads
- interestingly, on the M1 the naive primes benchmark completes in the same time for JAVA, C++ or Go, is this a hint at easier compile time optimisation for this architecture?
- the M1 is really much more power efficient when running under full load like in Stockfish MT benchmark and even more so when under light load (due to the 4 efficient cores)
- again sorry for missing out on AMD...

Let’s try out stuff live!

The audience suggested: https://github.com/drujensen/fib.
We didn’t get Docker to work, but compiled three versions of fib on three machines.
Here are the results.



Dru Jensen’s fib

takeaways:

- the comparison is a bit tricky, because we’re comparing high-core-count server CPUs with a mobile CPU on a different OS (with newer compilers)

Sources

- [1] https://www.arm.com/blogs/blueprint/200bn-arm-chips
- [2] https://www.heise.de/select/ix/2022/1
- [3] https://www.anandtech.com/show/16252/mac-mini-apple-m1-tested
- [4] https://www.anandtech.com/show/15578/cloud-clash-amazon-graviton2-arm-against-intel-and-amd
- [5] https://aws.amazon.com/blogs/aws/join-the-preview-amazon-ec2-c7g-instances-powered-by-new-aws-graviton3-processors/
- [6] https://www.tomshardware.com/news/intel-core-i9-12900hk-outpaces-apple-M1-max-but-theres-a-catch