

Lightning Quick Performance on Apple M1 with TVM

Phil Mazenett

Head of Field Engineering, OctoML



Agenda

- Recent results on the new M1 chips (posted in our blog)
- How to replicate
- Q&A

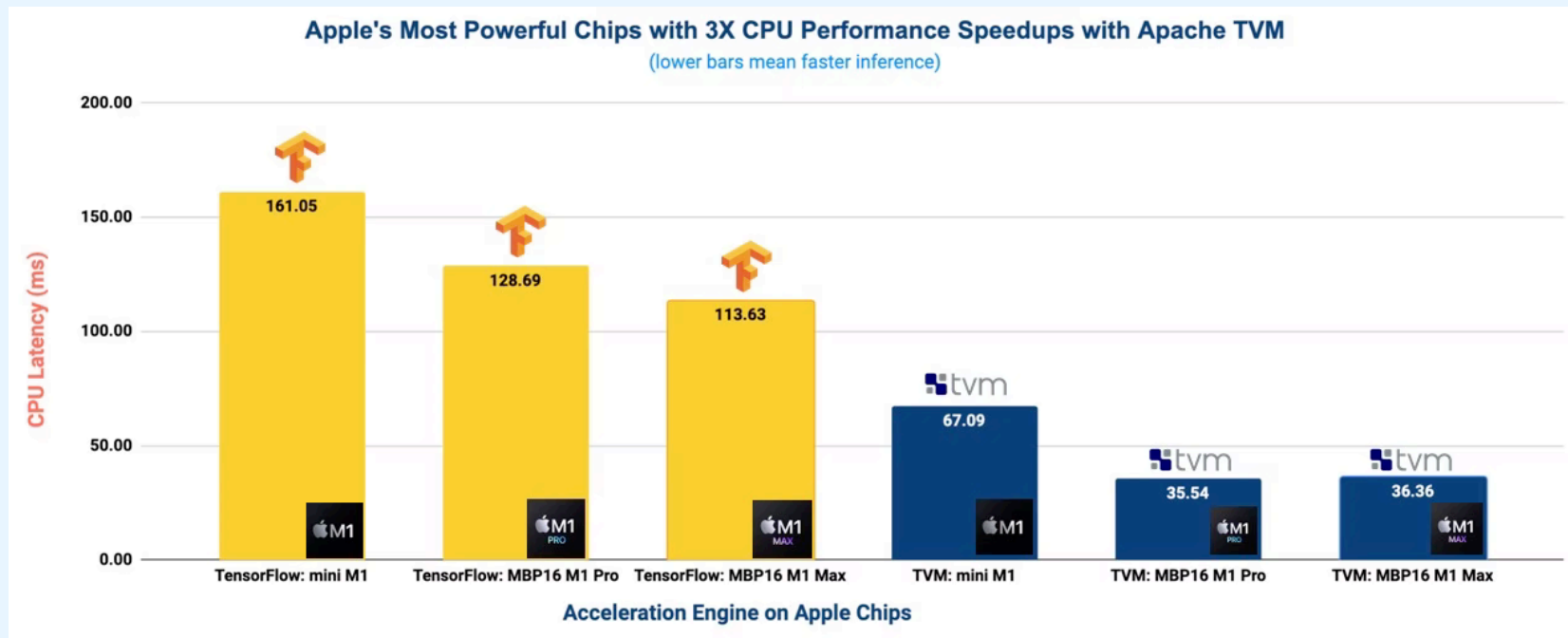
Recent results on the new M1 chips

Blog Post - octoml.ai/blog

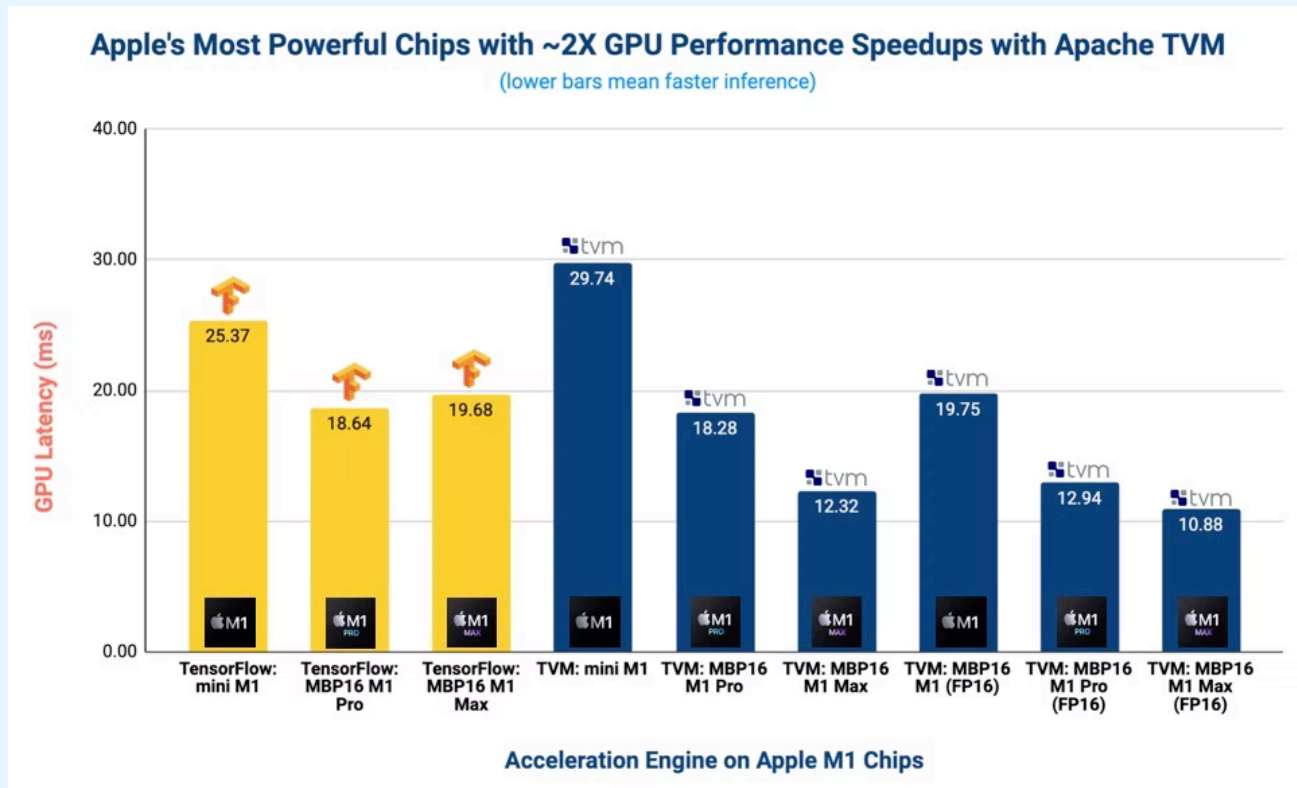
The screenshot shows a web browser displaying the OctoML blog post. The page features the OctoML logo, navigation menus for Product, Company, Blog, and Resources, and buttons for 'Start for free' and 'Login'. The main heading is 'OctoML's BERT Model Acceleration Proves Apple M1 Pro and Max Chips Make AI Accessible to Everyone'. The authors are Phil Mazenett and Jared Roesch, and the post is dated Dec 9, 2021. The text describes OctoML's work with Hugging Face's BERT implementation, highlighting a 3X speedup over Apple's TensorFlow plugin. A bar chart titled 'Apple's Most Powerful Chips with 3X CPU Performance Speedups with Apache TVM' compares CPU latency (ms) for M1 Pro, M1 Max, and M1 chips against TVM acceleration. The chart shows that TVM significantly reduces latency, with M1 Pro achieving a 3X speedup (indicated by an orange arrow).

Chip	Latency (ms)	Acceleration
M1 Pro	163.48	3x faster
M1 Max	128.89	3x faster
M1	113.83	3x faster
TVM (M1 Pro)	57.99	3x faster
TVM (M1 Max)	35.54	3x faster
TVM (M1)	28.38	3x faster

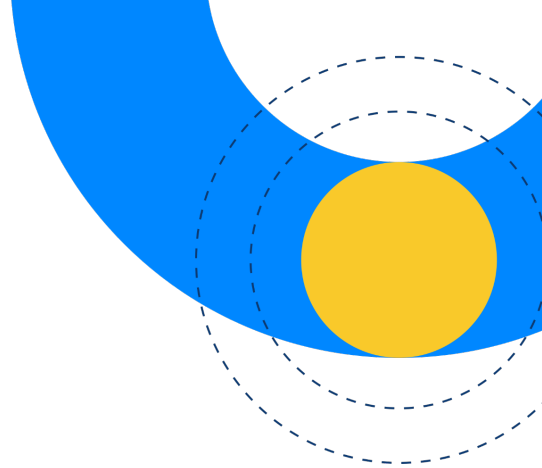
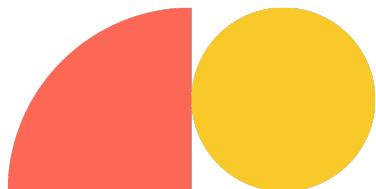
CPU



GPU



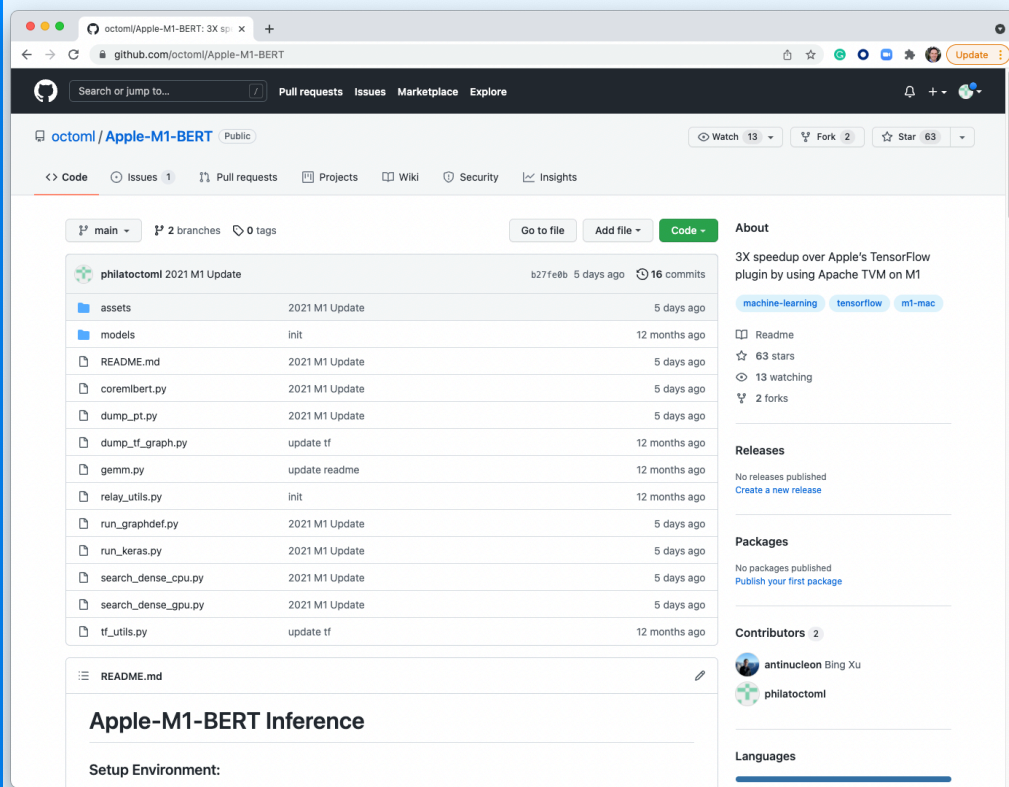
How to replicate



Clone the Repo

Things to Note:

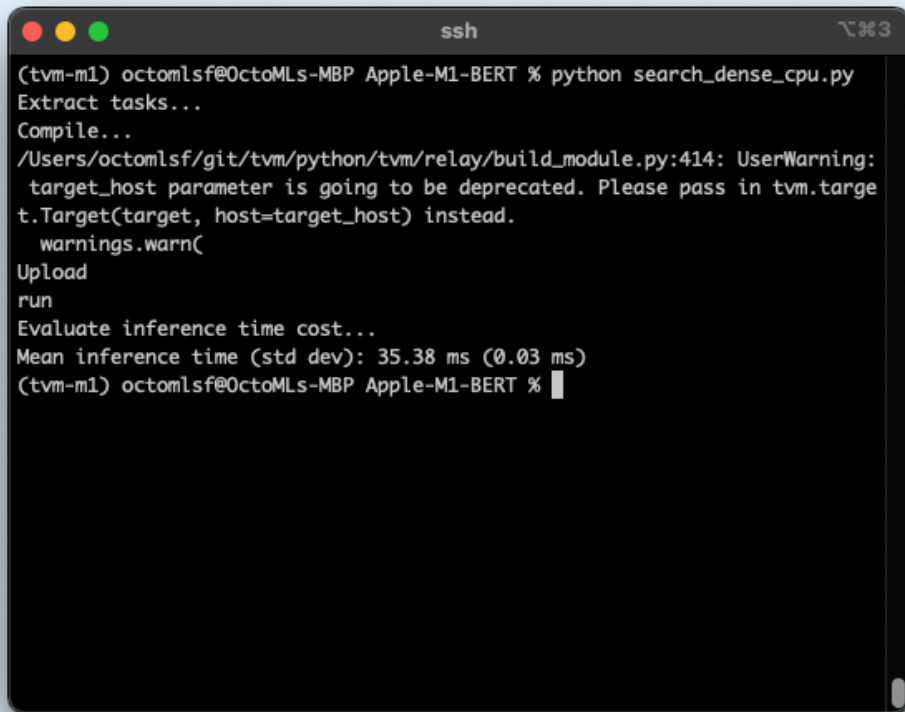
- Miniforge continues to be the best way to run all the necessary packages on M1
- While Python 3.9 is the default version installed at this time, all tests have been done with 3.8
- When using TVM, ensure you are building with METAL and LLVM set to ON and OPENMP set to gnu
- There is a CoreML script you can use to run some benchmarks using the CoreML libraries. The coremltools library is used for these tests



Running Benchmarks

Things to Note:

- In order to achieve the best results (results highlighted in the blog post) you will need to set the TVM_NUM_THREADS environment variable to the number of your Performance cores; 8 in the Pro and Max
- You should not be running any other CPU intensive app at the same time or apps that can limit performance. The results shown were achieved on a vanilla, clean install of MacOS Monterrey with a local account (No background iCloud processes running)



```
(tvm-m1) octomlsf@OctoMLs-MBP Apple-M1-BERT % python search_dense_cpu.py
Extract tasks...
Compile...
/Users/octomlsf/git/tvm/python/tvm/relay/build_module.py:414: UserWarning:
  target_host parameter is going to be deprecated. Please pass in tvm.target.Target(target, host=target_host) instead.
  warnings.warn(
Upload
run
Evaluate inference time cost...
Mean inference time (std dev): 35.38 ms (0.03 ms)
(tvm-m1) octomlsf@OctoMLs-MBP Apple-M1-BERT %
```

Thank you!

Q&A following session

Also feel free to email me at phil@octoml.ai if you have any other questions

