LEHIGH UNIVERSITY

Compiling and Benchmarking OpenFOAM on Ookami HPC

Computational Fluid Dynamics on ARM Architecture

Guanyang Xue
Mechanical Engineering & Mechanics, Lehigh University
March 23, 2023

Background

- OpenFOAM is an open-source (GPLv3) C++ toolbox for Computational Fluid Dynamics (CFD). Users can create customized numerical solvers based on its framework.
- OpenFOAM is now maintained by two different groups: OpenCFD Ltd. (https://www.openfoam.com) and OpenFOAM Foundation (https://www.openfoam.org). The **OpenCFD** version has better support for Clang and ARM CPUs.
- Fujitsu released a patch for OpenFOAM-v1812[1] including Fujitsu compiler support and performance tuning of sparse matrix solvers.
- This work, based on the Fujitsu patch for v1812, compiled the latest release OpenFOAM-v2212 on Ookami.

[1] https://github.com/fujitsu/oss-patches-for-a64fx/tree/master/OpenFOAM

Motivation

- I compiled OpenFOAM on a M1 Mac mini *natively* and ran benchmark on it.
- M1 achieved the fastest single core performance among all OpenFOAM hardware benchmark results on CFD online forum[2].
- From the successful experience of running OpenFOAM on ARM CPU, we applied for Ookami resources on NSF ACCESS for proof-of-concept research.
- The theoretical 1TB/s memory bandwidth of A64FX should resolve the bottleneck of large-scale sparse matrices.

[2] https://www.cfd-online.com/Forums/hardware/198378-openfoam-benchmarks-various-hardware.html

Dependencies

OpenFOAM uses several third party libraries to provide extra features.

Currently available in Ookami

```
fujitsu/compiler/4.8
fftw3/fujitsu/sve-1.0.0
petsc/fujitsu4.8/3.18.3
boost/1.71.0
```

Third party libraries need to be built[3]

```
Scotch-6
Metis
CGAL-4 #(without GMP/MPFR)
ADIOS2 #(work in progress)
```

Some libraries require specific versions to work with OpenFOAM.

[3] ADIOS2 is currently disabled due to a *libfabric* issue.

Environment Variables

Thanks to Fugaku users[4], most necessary patches have been merged into the master branch. The following variables need to be configured in etc/bashrc.

Select Fujitsu compiler

```
export WM_COMPILER=Fujitsu
export WM_MPLIB=FJMPI
```

Disable SIGFPE

```
export FOAM_SIGFPE=false
```

For most ARM CPUs, SIGFPE is not available.

Improve parallel performance (in SLURM script)

```
export XOS_MMM_L_PAGING_POLICY=demand:demand:demand
```

[4] https://develop.openfoam.com/Development/openfoam/-/issues/1671

Compiler Flags

Original v1812 patch from Fujitsu

```
CC              := mpiFCC
c++OPT          := -Nclang -std=gnu++11 --verbose -ffj-largepage -O3
↪  -stdlib=libstdc++ -march=armv8.3-a+sve # no idea why it's armv8.3
```

OpenFOAM default flags since v2006

```
CC              := FCC$(COMPILER_VERSION) -std=c++14
c++OPT          := -ffp-contract=fast -ffast-math -O3
↪  -funsafe-math-optimizations # too aggresive - code will crash
```

My proposed flags for v2212

```
CC              := FCC$(COMPILER_VERSION) -std=c++14
c++OPT          := -Nclang -O3 -march=armv8.2-a+sve -ffj-largepage
```

Benchmarking OpenFOAM

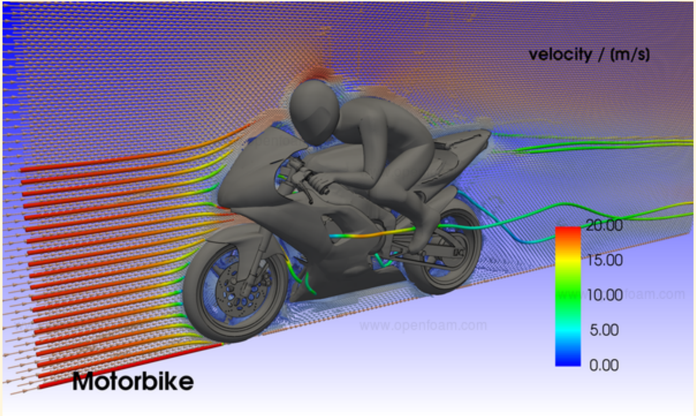
- This benchmark will compare the OpenFOAM simulation performance between Ookami (A64FX, 48 x 2.0GHz, 1TB/s), 2021 MacBook Pro (M1 Pro, 6 x 3.2GHz, 200GB/s) and Hawk (2 x Xeon Gold 6230R, 52 x 2.1 GHz, 280GB/s) from Lehigh University.
- The estimated power consumption when running CFD simulations: Ookami: 110W, MacBook: 35W, Lehigh Hawk: 450W.
- # Ideally, the Intel Ice Lake-SP (Intel 10 nm) or AMD EPYC Rome/Milan (TSMC N7) would be better x86 candidates for comparison purposes.

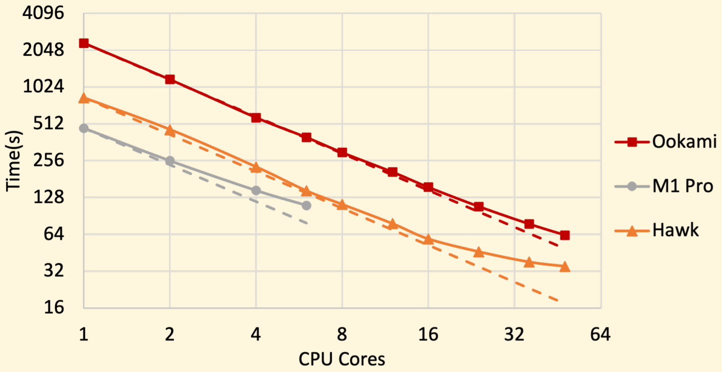
Motorbike

Description

"Motorbike" is the de-facto standard benchmark case. It has 1.9M cells and is designed to test the CPU performance.

The benchmark case solves the steady-state flow field around the motorbike using SIMPLE algorithm with $k-\omega$ turbulence model.





An M1 core is 3x faster than a Ookami core.

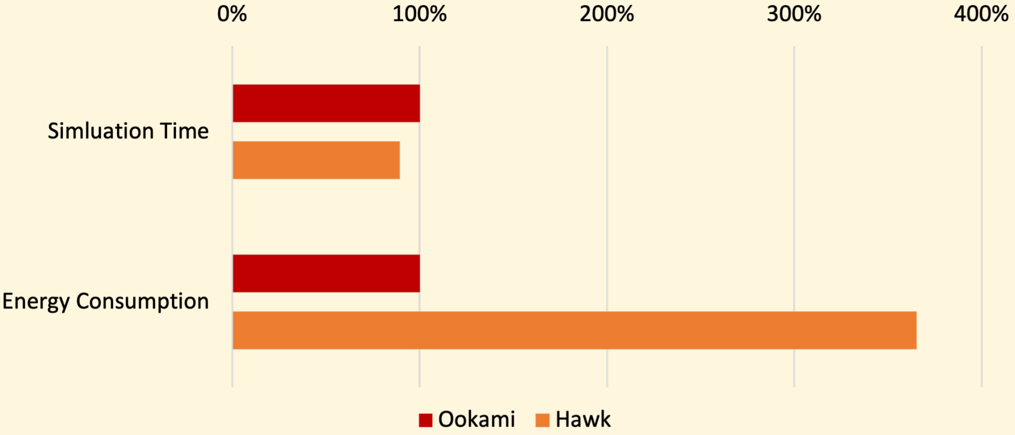
A Hawk core is 2x faster than a Ookami core.

Ookami has better parallel efficiency (77%) than hawk (50%) when using 48 cores.

Never perform pre-processing on Ookami as single-threaded utilities will take forever to run.

HPC Motorbike
Small (8.5M Cells)

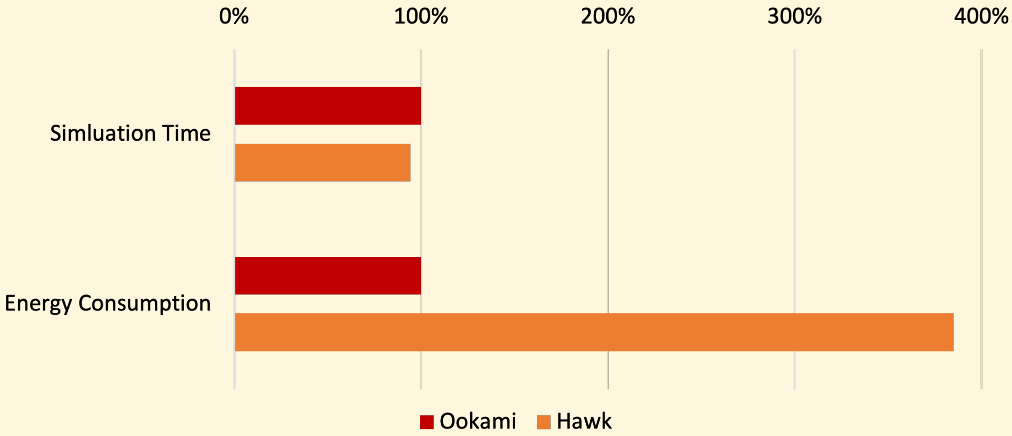
The HPC motorbike case has finer mesh (8.5M for small and 17M for medium).



HPC Motorbike

Medium (17M Cells)

This 17M mesh used 29GB RAM on A64FX and almost ran out of memory.



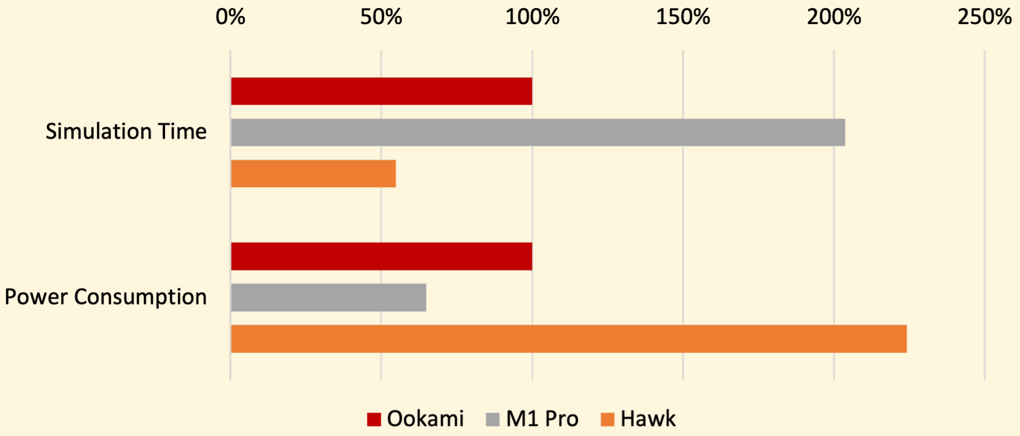
- On x86, the simulation slows down as the memory bandwidth becomes the bottleneck.
- On A64FX, the memory bandwidth is sufficiently large to make the job always being CPU-bounded. With the largest possible mesh, an Ookami node can achieve 95% performance of a similar x86 node.
- A64FX doesn't slow down with more cells. It is suggested to keep at least 10M cells per node to utilize the advantage of HBM memory.
- # Intel has launched Sapphire Rapids CPU equipped with 64GB HBM memory[5].

[5] https://ark.intel.com/content/www/us/en/ark/products/codename/230303/products-formerly-sapphire-rapids-hbm.html

Viscoelastic Flow

4.6M Cells

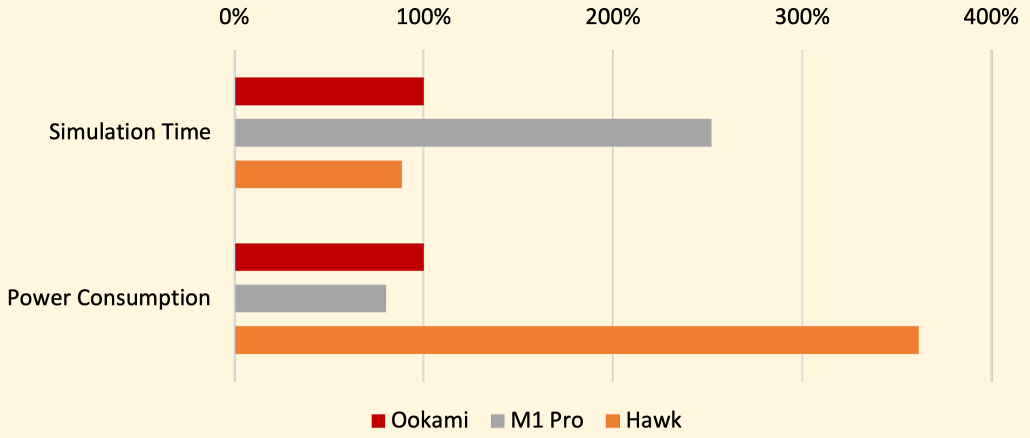
The viscoelastic flow solver adds an extra constitutive equation of stress tensor to the incompressible fluid solver. The benchmark case is a contraction-expansion microfluidic device.



Membrane Distillation

9M Cells

The Vacuum Membrane Distillation (VMD) simulation solves coupled flow, temperature and mass concentration fields in a long wavy channel.



Best Practice

- Always perform pre-processing (e.g. meshing, decomposition) on local workstations then upload the ready-to-run case file to Ookami.
- $<$5M cells per node ($<$100K cells per core) may cause heavily CPU-bounded jobs.
- $>$17M cells per node ($>$350K cells per core) may cause out of memory error.
- The sweet spot is 10M-15M cells per node (200-300K cells per core).
- For CFD simulations, the current ACCESS credit exchange rate (1000:62) makes Ookami SUs approximately 2.5x valuable as x86 SUs.
- For CFD simulations, Ookami saves more than 70% of electricity compared to Intel 14nm CPUs.

Acknowledgement

- I would like to thank my colleague Justin Caspar for providing benchmark cases on membrane distillation.
- Many thanks to Eva Siegmann and George Liang for creating the OpenFOAM module with me. Also thanks Tony Curtis for debugging during office hours.