Harnessing more compute power at NeSI with Open-Multiprocessing (OpenMP)

Alexander Pletzer (NeSI/NIWA), Wolfgang Hayek (NIWA/NeSI),
Chris Scott (NeSI/UoA)
NZ RSE Conference 9-10 September 2020
alexander.pletzer@nesi.org.nz

NeSI
New Zealand eScience Infrastructure

New Zealand eScience Infrastructure

Consultancy

- Analysis, debug and optimise user applications

Support

- Expert knowledge in multiple domains

Training

- Software Carpentry / Data Carpentry
- Intro & advanced HPC training

NeSI

New Zealand eScience Infrastructure

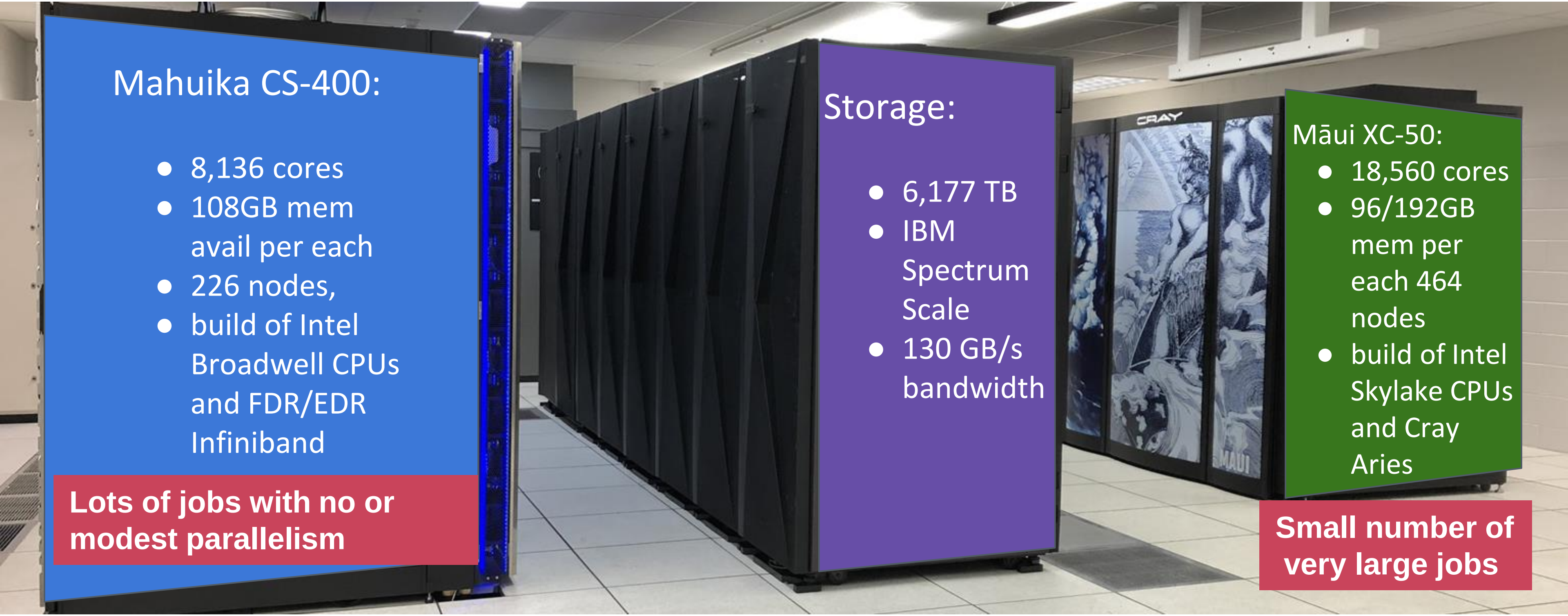
Data transfer

- high speed data input/output
  Partnership with Globus (global data management platform)

Hardware and software for compute and analysis

- ~700 compute nodes
- hundreds of software packages

NeSI's work horses



Mahuika CS-400:

- 8,136 cores
- 108GB mem avail per each
- 226 nodes,
- build of Intel Broadwell CPUs and FDR/EDR Infiniband

Lots of jobs with no or modest parallelism

Storage:

- 6,177 TB
- IBM Spectrum Scale
- 130 GB/s bandwidth

Māui XC-50:

- 18,560 cores
- 96/192GB mem per each 464 nodes
- build of Intel Skylake CPUs and Cray Aries

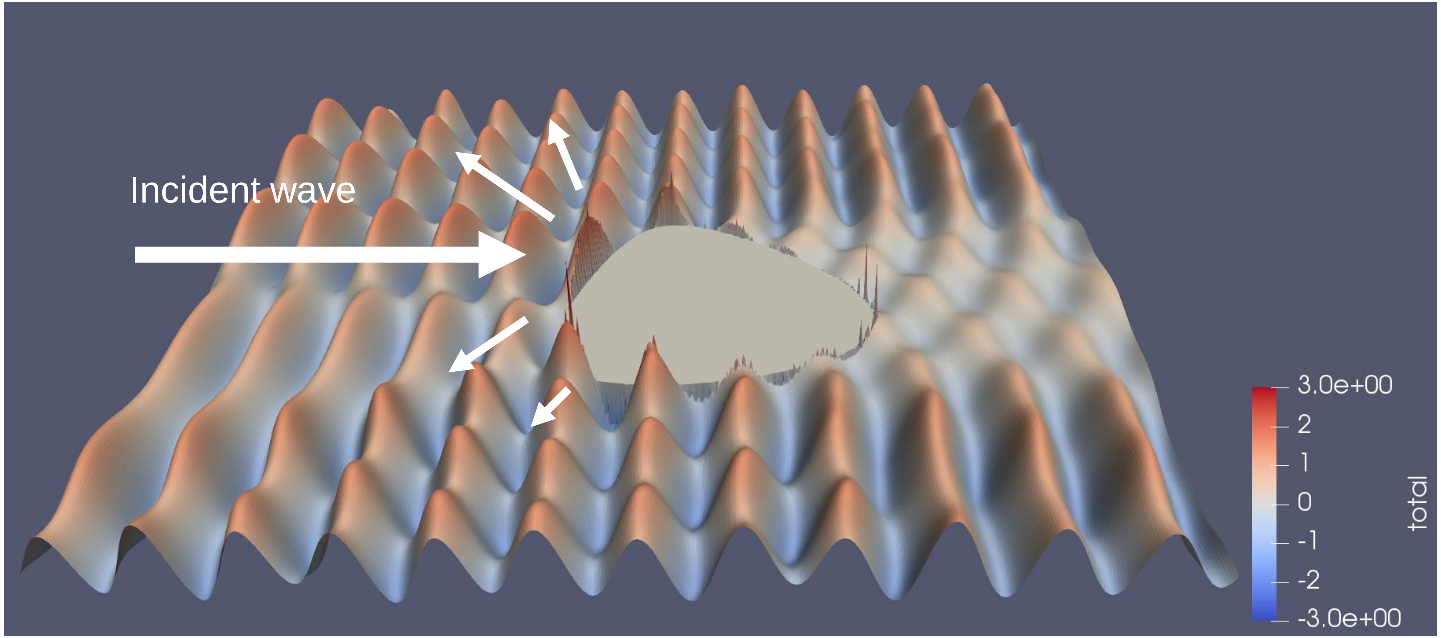
Small number of very large jobs

How we use OpenMP at NeSI

1. OpenMP parallelism through **third party libraries** (e.g. MKL, scipy)
2. OpenMP as a **springboard to parallel computing**
   - {R, Python, MATLAB} -> C -> OpenMP is an example
3. **Complements MPI** for improved scaling
   - Exploit OpenMP shared memory to reduce data traffic
   - Community code: Unified Model, specfem3D, …

Case 1: boundary element code in Python used as training material

Wave hits an obstacle and scatters (https://github.com/pletzer/scatter)



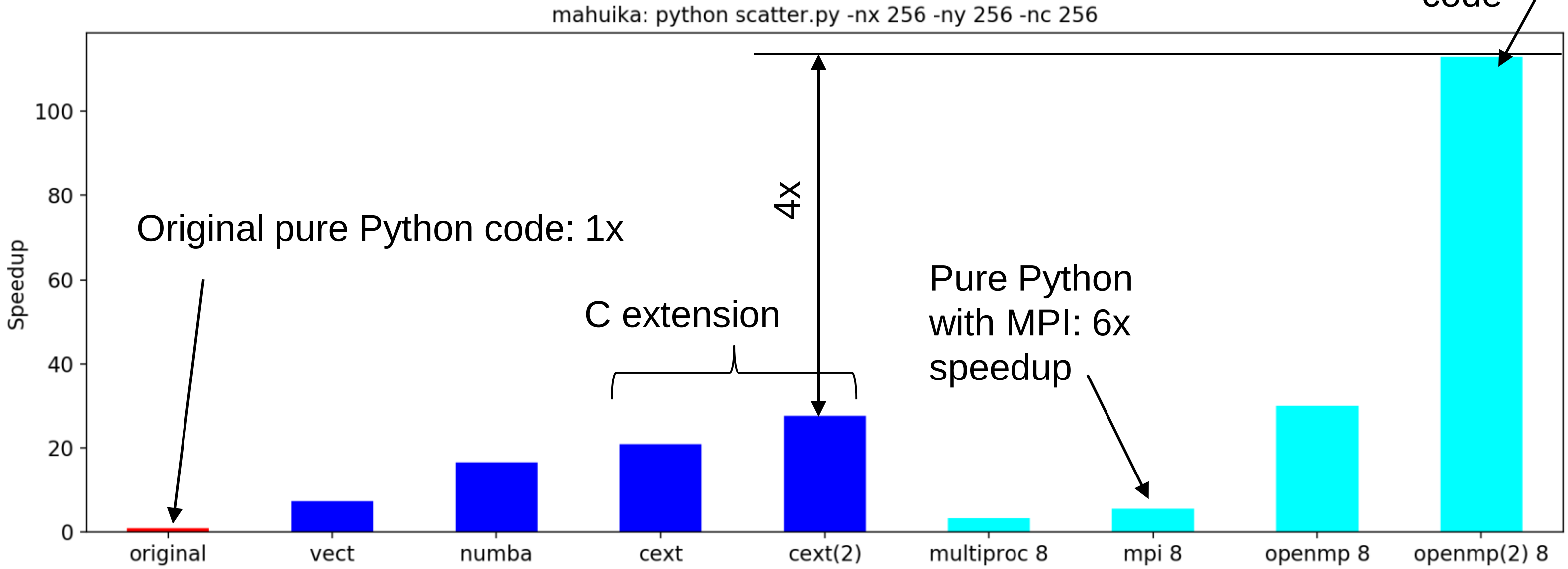
https://nesi.github.io/perf-training/

C extension + OpenMP wins

Comparing different implementations

C extension with **OpenMP**: **110x speedup** (8 threads) over pure Python single threaded code

Larger is better



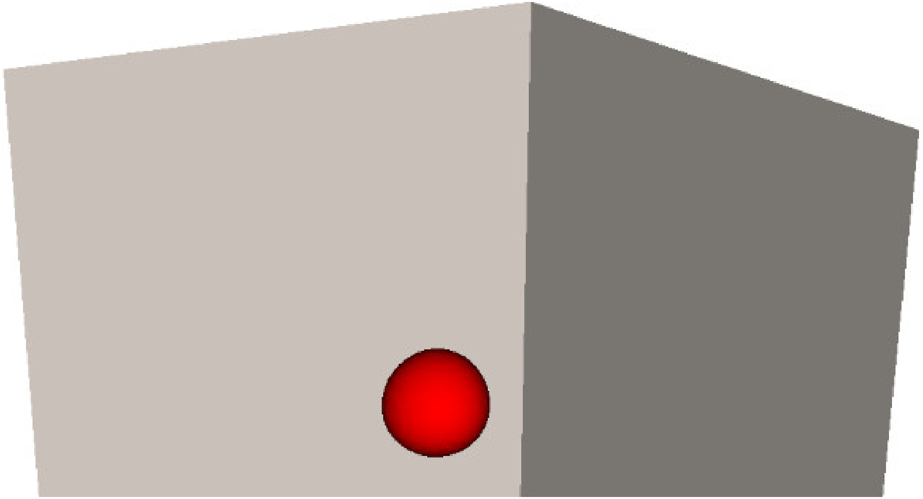
Original pure Python code: 1x

C extension

Pure Python with MPI: 6x speedup

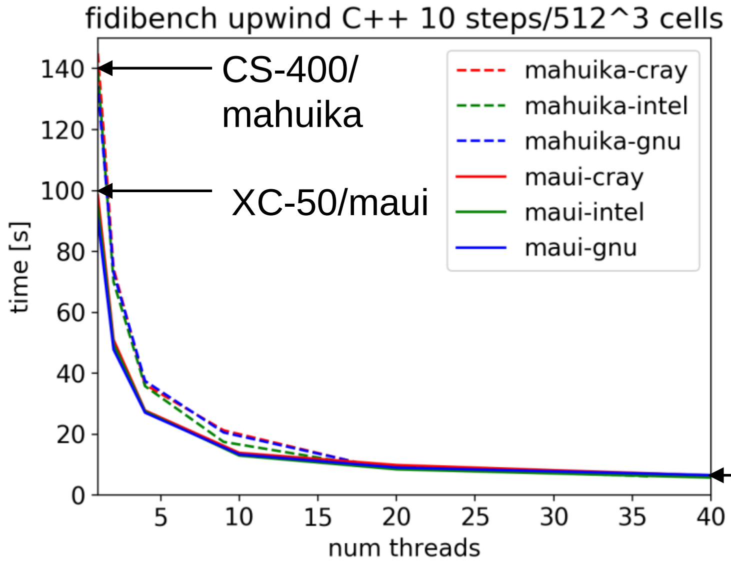
Case 2: different versions of finite difference code written in C++, Fortran, Python and Julia

- Advection of a bubble in space
- Compute pattern is similar to many finite difference/volume/element codes
- https://github.com/pletzer/fidibench



OpenMP performance on mahuika and maui are comparable for OMP_NUM_THREADS ~ 20-40

XC-50 is up to 40% faster than CS-400 at low OMP_NUM_THREADS counts (not much difference at higher thread counts)



CS-400/ mahuika

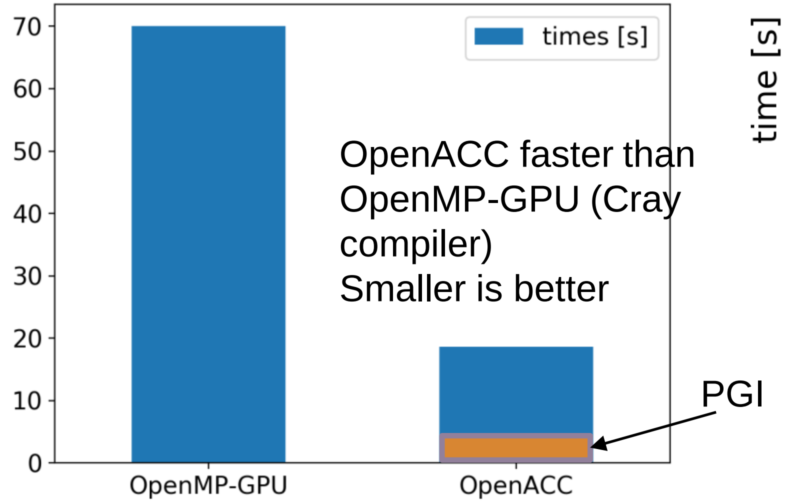
XC-50/maui

20x speedup

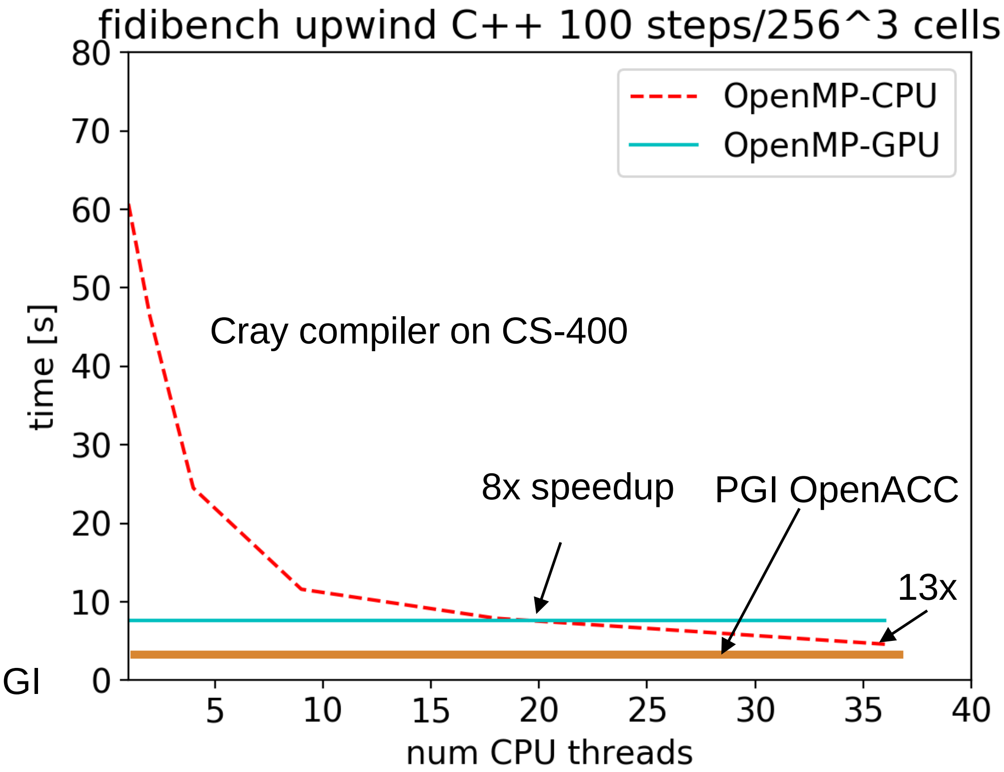
No difference between compilers

Testing OpenMP offloading to accelerator capability

Best performance obtained by offloading to P-100 GPU and using OpenACC



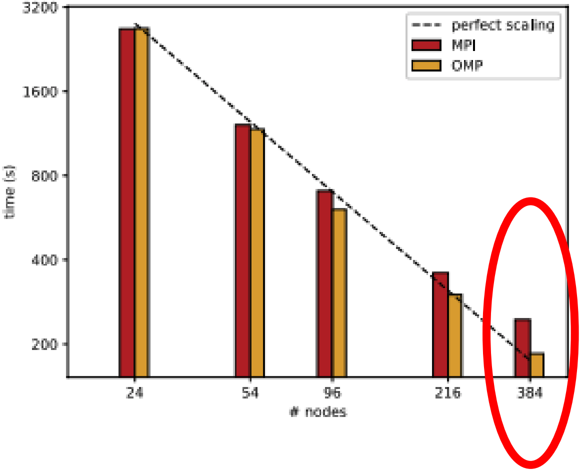
OpenACC faster than OpenMP-GPU (Cray compiler)
Smaller is better

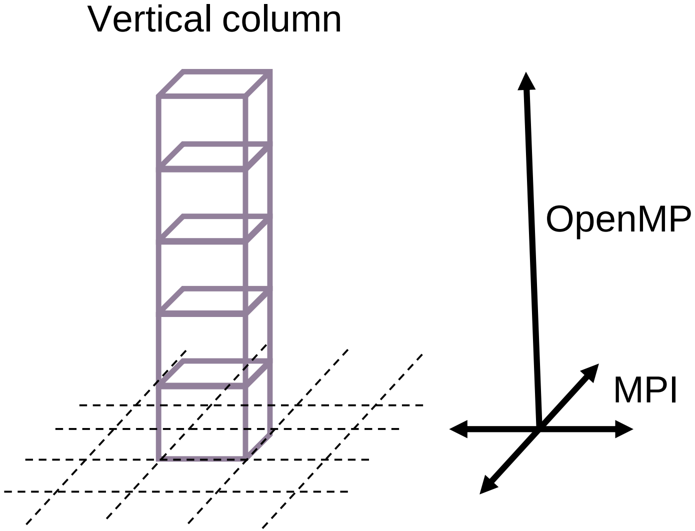


Cray compiler on CS-400

Case 3: LFRic is a next generation weather and climate code

Improved scalability at the high end





Lessons learned

- **Beware of heap allocation** in OMP parallel regions
- Acquiring experience with OpenMP offloading
  - More complex to use (need to map data between CPU and GPU
  - **Limited by memory** on GPU compared to memory on a node (200GB)
  - Limited (but improving) support among compilers
  - OpenMP-GPU currently slower than OpenACC (hopefully this will change in the future)

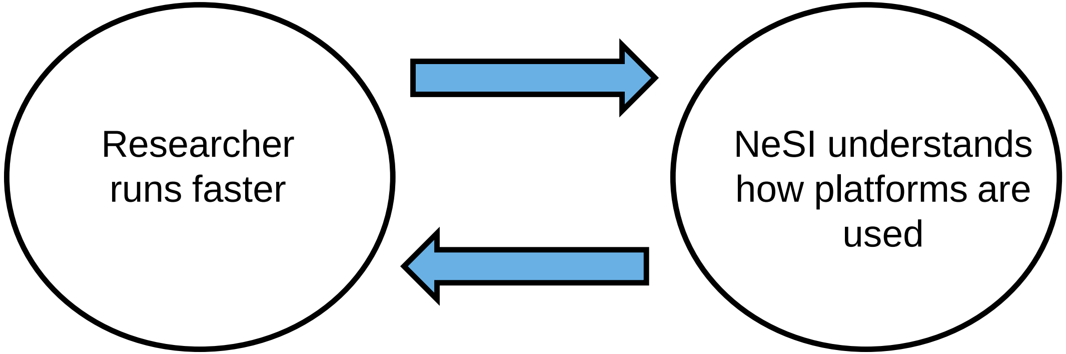
Summary

- We’re actively promoting OpenMP at NeSI
  - Great entry point for users new to parallel computing
- Glad to see compilers adopting OpenMP 5, eager to see Intel catching up with offloading to NVIDIA GPU (but won’t hold my breath)
- Complexity of hardware is likely to become an increasing concern
  - More complex #pragma omp directives
  - More complex run time environment (OMP_PLACES, OMP_PROC_BIND, …._)

NeSI's consulting services can help you

Talk to us if you need to run faster

- NeSI provides 1.5 engineer to a researcher for up to 3 months (~20-100 hours)
- Outcome is a case study: https://www.nesi.org.nz/case-studies



Thank you. Time for questions

NeSI
New Zealand eScience Infrastructure