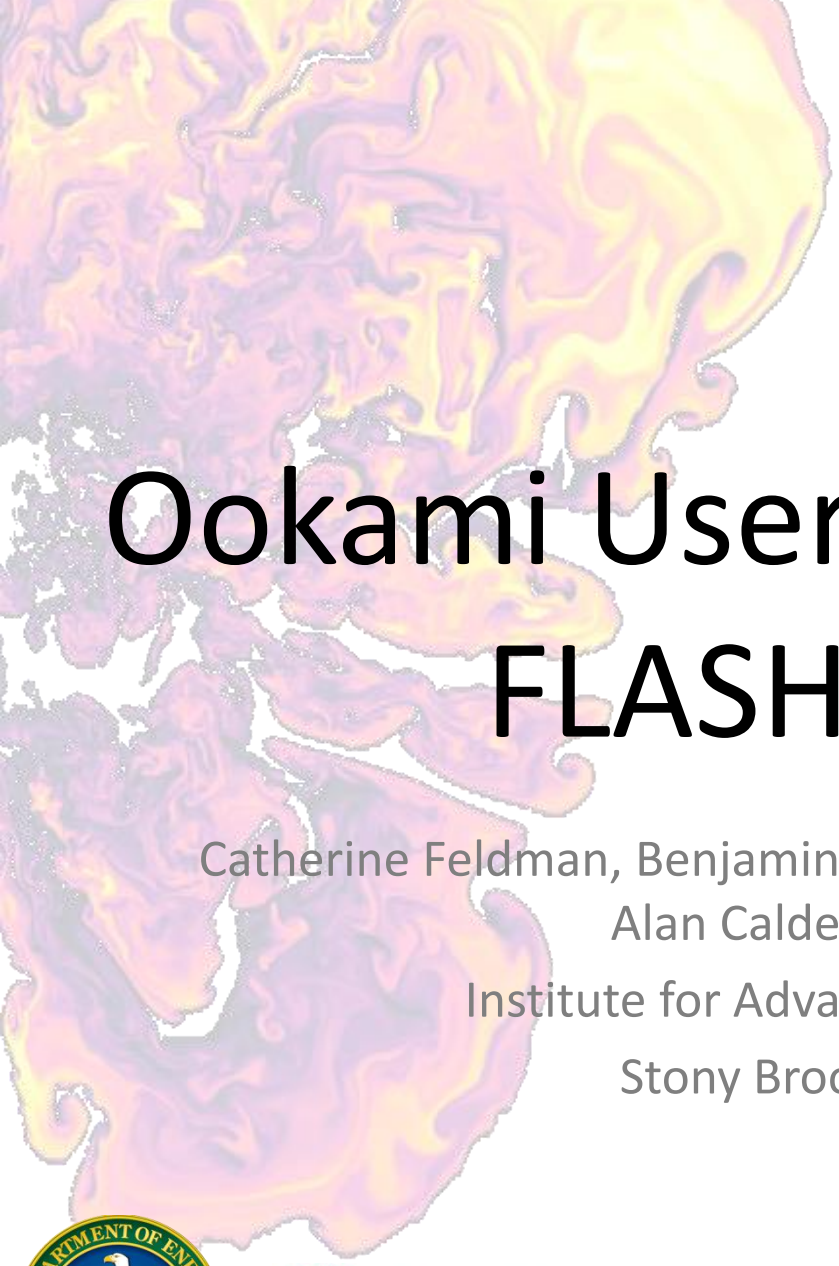


Ookami User Group Meeting: FLASH on A64FX

Catherine Feldman, Benjamin Michalowicz, Eva Siegmann, Tony Curtis, Alan Calder and Robert Harrison

Institute for Advanced Computational Science

Stony Brook University, NY, USA

FLASH Center

Founded in 1997 as part of the DOE's ASCI

THE ASTROPHYSICAL JOURNAL SUPPLEMENT SERIES, 131:273–334, 2000 November

FLASH: AN ADAPTIVE MESH HYDRODYNAMICS CODE FOR MODELING ASTROPHYSICAL THERMONUCLEAR FLASHES

B. FRYXELL,[1,2] K. OLSON,[1,2] P. RICKER,[2,3] F. X. TIMMES,[2,3] M. ZINGALE,[2,3] D. Q. LAMB,[1,2,3] P. MACNEICE,[4] R. ROSNER,[1,2,3] J. W. TRURAN,[1,2,3] AND H. TUFO[2,5]

Received 1999 November 9; accepted 2000 April 13

ABSTRACT

We report on the completion of the first version of a new-generation simulation code, FLASH. The FLASH code solves the fully compressible, reactive hydrodynamic equations and allows for the use of adaptive mesh refinement. It also contains state-of-the-art modules for the equations of state and thermonuclear reaction networks. The FLASH code was developed to study the problems of nuclear flashes on the surfaces of neutron stars and white dwarfs, as well as in the interior of white dwarfs. We expect, however, that the FLASH code will be useful for solving a wide variety of other problems. This first version of the code has been subjected to a large variety of test cases and is currently being used production simulations of X-ray bursts, Rayleigh-Taylor and Richtmyer-Meshkov instabilities, and th monuclear flame fronts. The FLASH code is portable and already runs on a wide variety of massiv parallel machines, including some of the largest machines now extant.

Subject headings: equation of state — hydrodynamics — methods: numerical — nuclear reactions, nucleosynthesis, abundances — stars: general

ACM Gordon Bell Prize
Recognizing Outstanding Achievement in High Performance Computing

Los Alamos · Sandia · Lawrence Livermore
ASCI
Accelerated Strategic Computing Initiative

APS physics
JOHN DAWSON AWARD
FOR EXCELLENCE IN PLASMA PHYSICS RESEARCH

FLASH CENTER

flash.rochester.edu

FLASH Code Features

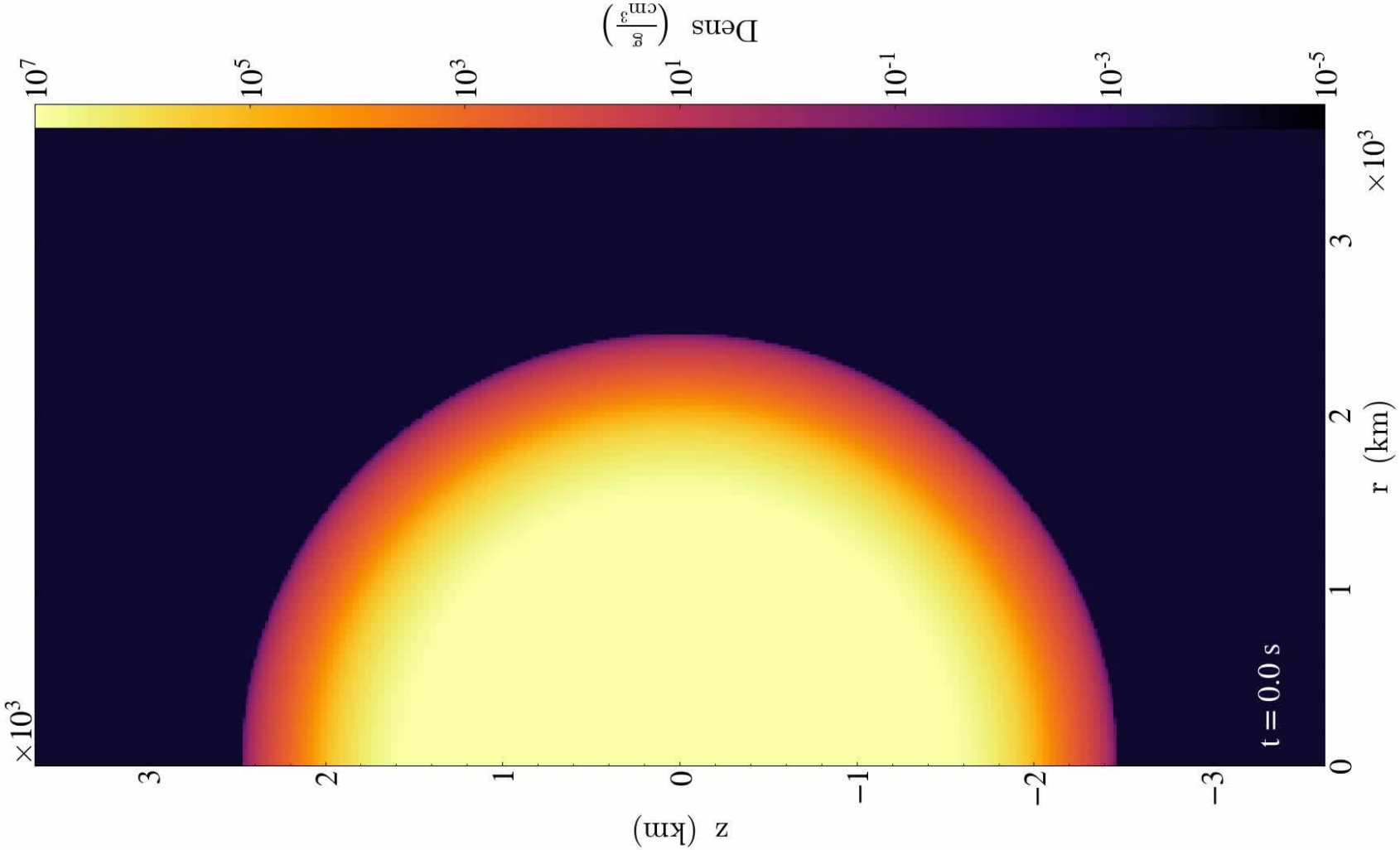
- Modular and extensible software package
- Multi-scale, multi-physics applications
- Solvers include hydrodynamics, magnetohydrodynamics, nuclear burning, high energy density physics, gravity, and more!
- Written in Fortran 90
- Parallelized with MPI
  - Not too successfully threaded (yet!)
- Uses PARAMESH adaptive mesh refinement to save on computation time
- 1D, 2D, and 3D problems

Our Science Application

Dim supernova

Hybrid white dwarf

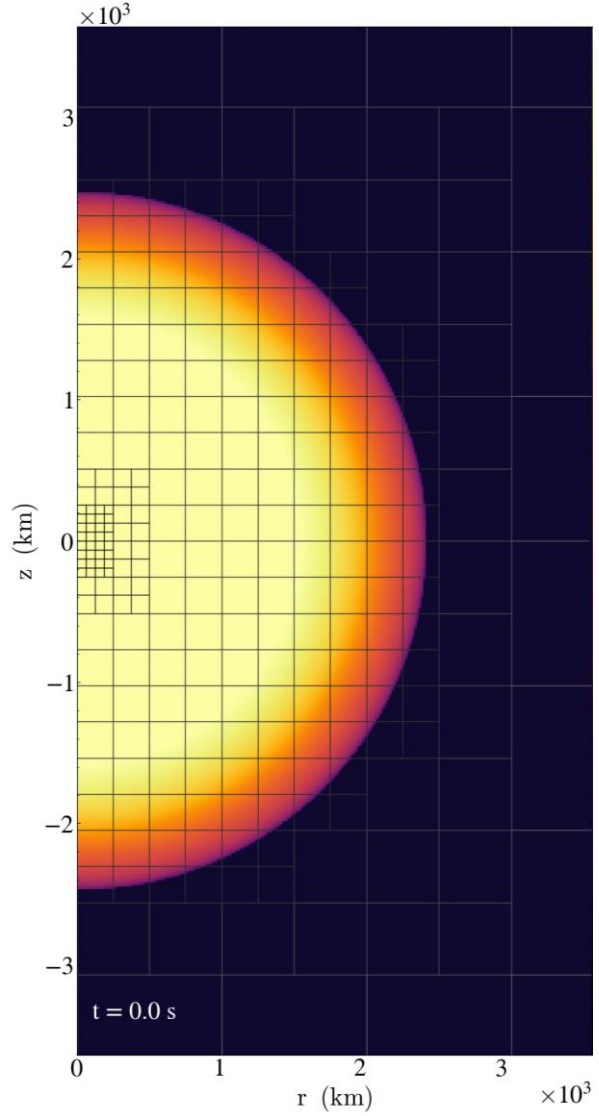
Pure deflagration – subsonic burning

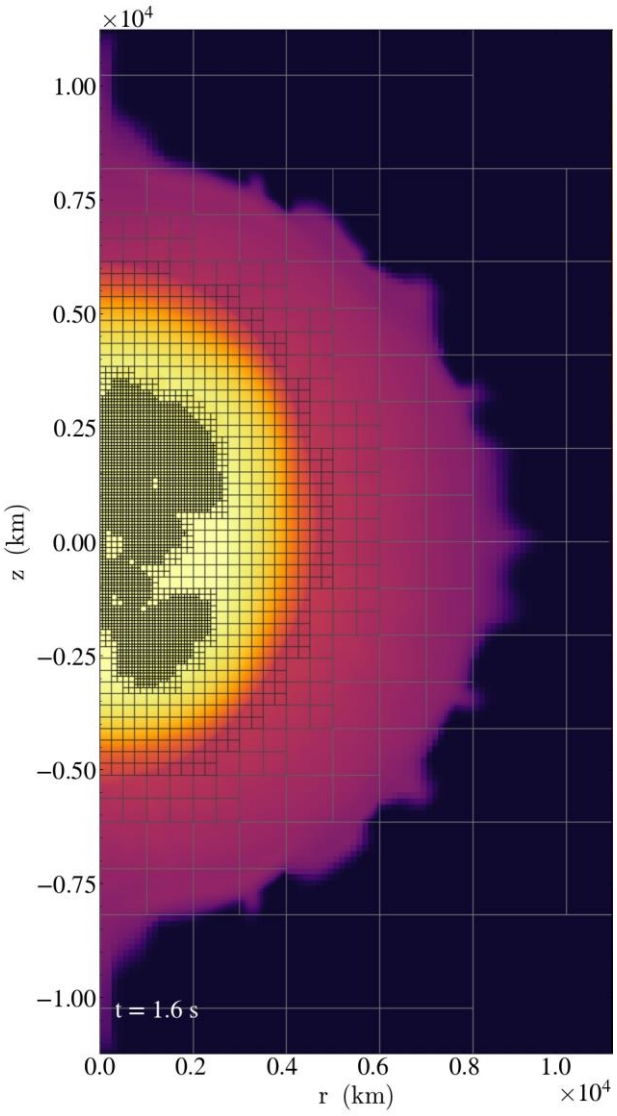


Adaptive Grid

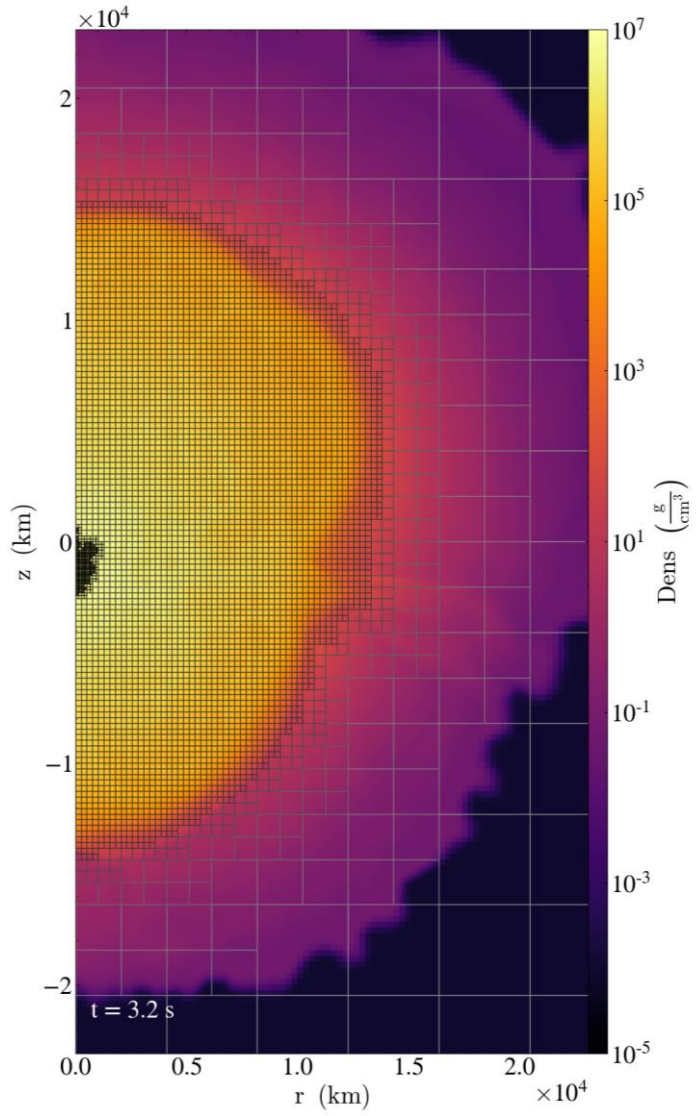
Complexity and size increase with time

10^2 blocks



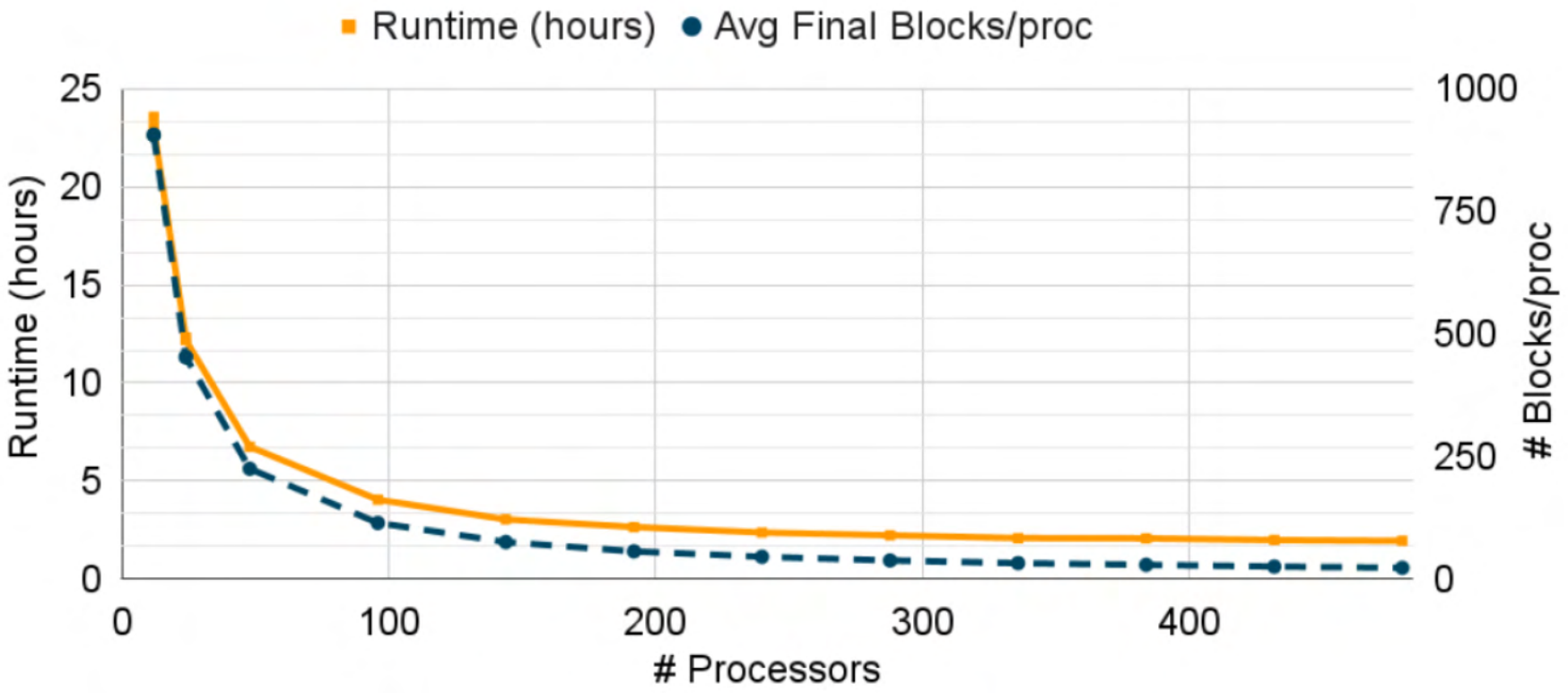


10^5 blocks



Strong Scaling

Run 2D supernova problem for 4s simulation time
GCC 10.3.0 + MVAPICH 2.3.5



Compiler Comparison

Run 2D supernova problem for 4s simulation time
on 240 cores (5 nodes, 48 cores/node)

Compilers:

- GCC
- Cray
- ARM

+ SVE

- NVIDIA

MPI Implementations:

- Open-MPI
- MVAPICH

(+ CUDA for GPUs)

Different combinations
are better for
different programming languages…
…so test away!

Compiler Flags

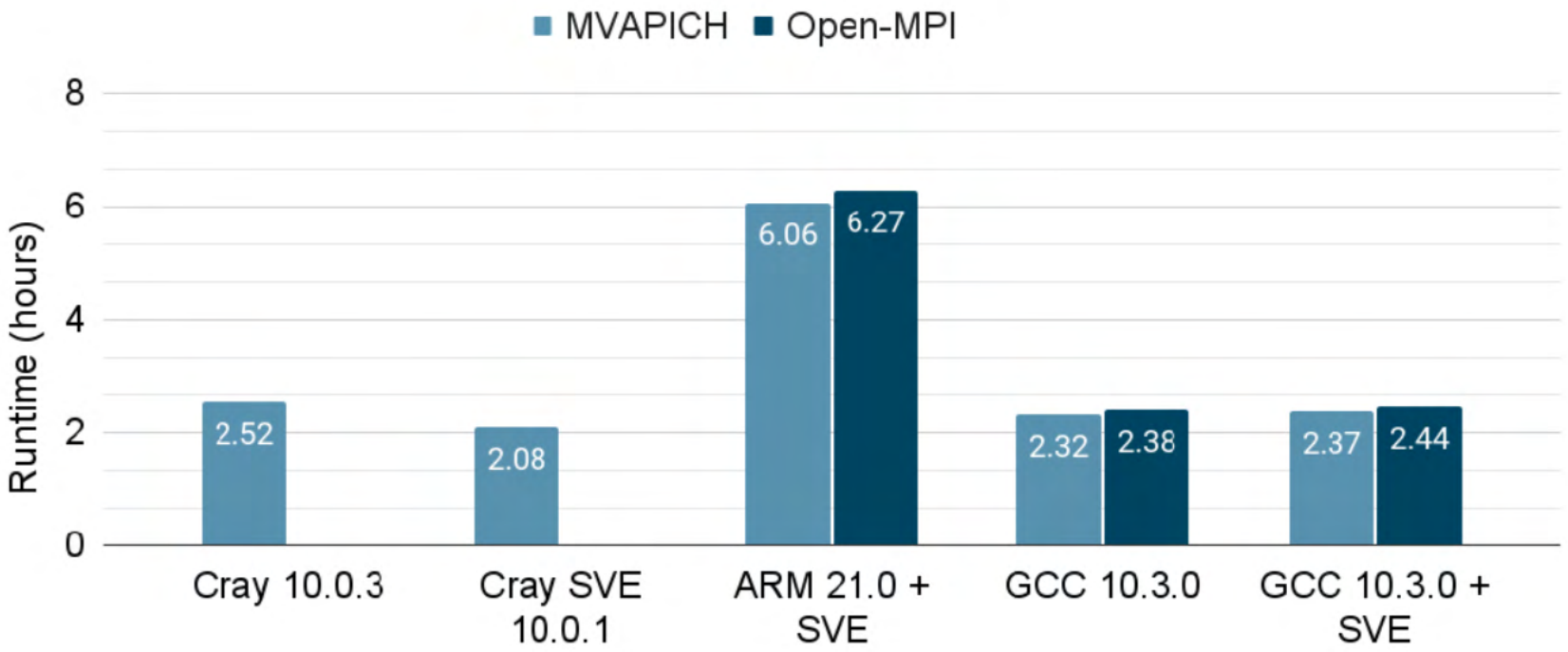
Tested 3 compilers and 2 MPI implementations

Compiler	Compiler Flags	MPI Implementation	Additional SVE Flags
GCC 10.3.0	`-fdefault-real-8 -Wuninitialized -fdefault-double-8 -fallow-argument-mismatch`	MVAPICH 2.3.5 Open-MPI 4.0.5	`-O3 -mcpu=a64fx`
Cray 10.0.3	`-O3 -h vector3 -s real64 -s integer32`	MVAPICH 2.3.5	Load the Cray 10.0.1 SVE module
ARM 21.0	`-r8 -armpl`	MVAPICH 2.3.5 Open-MPI 4.0.5	`-O3 -mcpu=a64fx`

Test if your executable is using SVE instructions, see if it's using the 'z' registers:

`objdump -d` *`executable`* `| grep 'z[0-9]'`

Compiler Comparison - Results



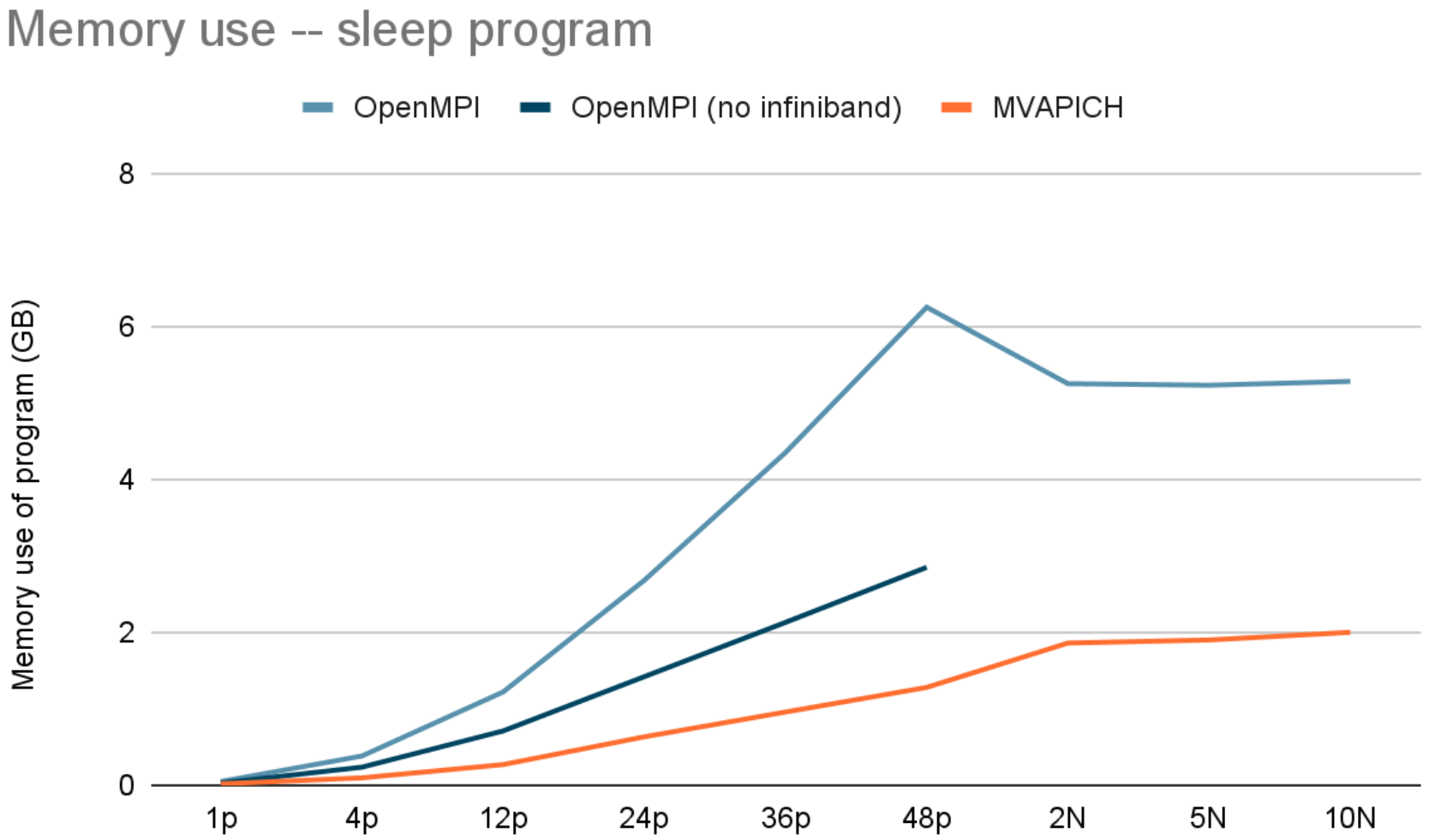
MVAPICH slightly faster than OpenMPI

ARM compiler extremely slow

Enabling SVE doesn't automatically produce a speedup – more tuning needed

The GCC 10 compiler is unable to vectorize certain math functions including `exp` and `pow` on its own, **but** can be linked to the ARM performance libraries:
(`-L<ARMPL_install_dir>/lib -larmpl_lp64 -lamath -lm`)
(Use `-larmpl_lp64_mp` for threaded apps)

Compiler Comparison - Results



Adding `UCX_TLS=cma,self,knem,xpmem` to OpenMPI job submission turns off infiniband and other communication and only activates intra-node communication. This brings OpenMPI memory allocation down. But you can only do this when running on 1 node (we need infiniband to talk between nodes!)

MPI Binding and Placement

Mostly point-to-point communication
Can we take advantage of this in CMGs?

MPI Implementation	Nodes	Cores	MPI Placement	Runtime (hours)
MVAPICH 2.3.5	4	96	Block	2.79
	4	96	Cyclic	2.80
Open-MPI 4.1.0	4	96	Block	2.86
	4	96	Cyclic	2.73
MVAPICH 2.3.5	2	96	Block	4.05
	4	192	Block	2.65

You can set the placement with the variable `MV2_BINDING_POLICY` in MVAPICH, and by using the `--map-by` option in OpenMPI

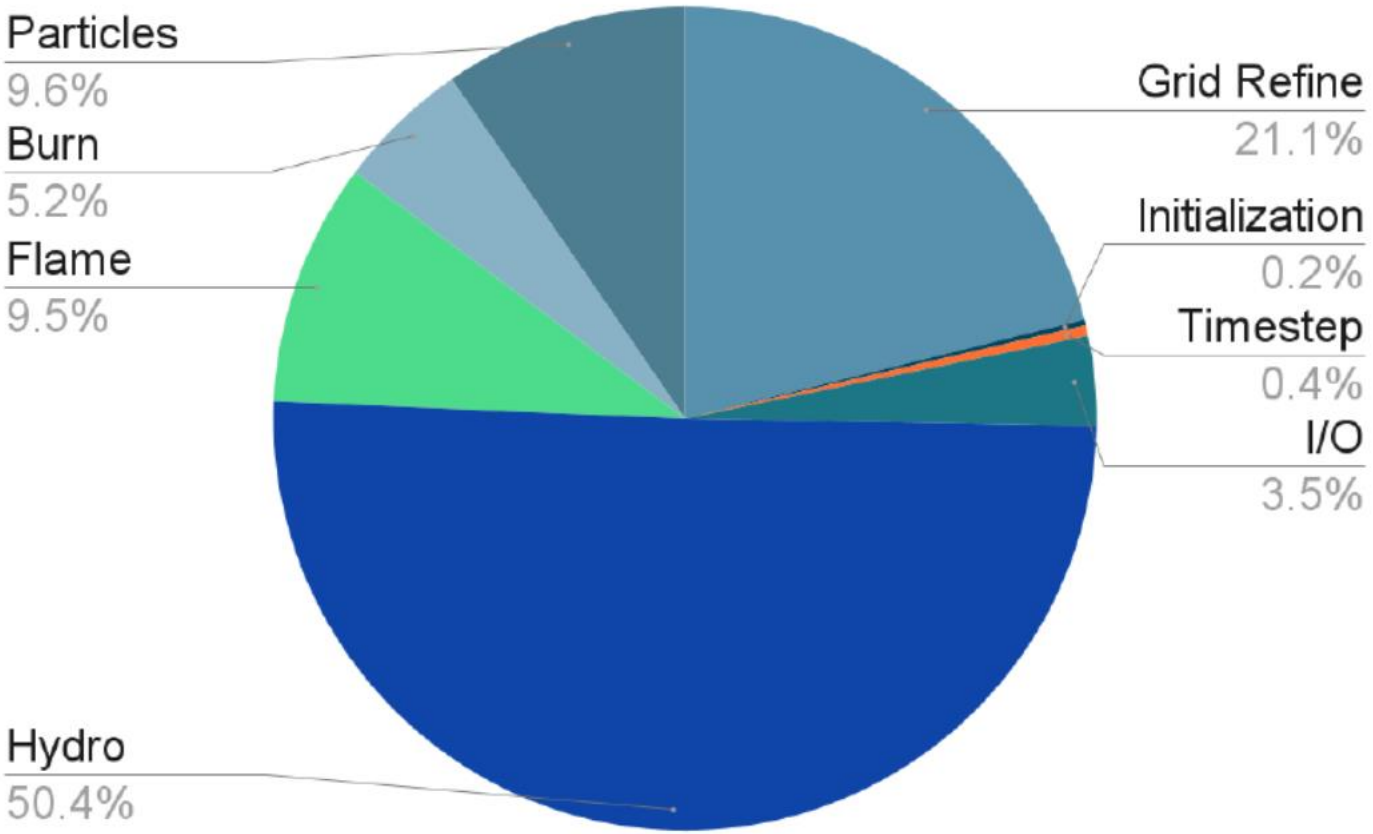
Architecture Comparison

Cluster	SeaWulf	vs	Ookami
Processor	328 Intel Xeon E5-2683v3 processors • Several queues • 2 CPUs per node, with 24/28/40 cores per node		174 A64FX processors • 48 cores per node • NUMA node; processors are in groups of 12
Processor Speed	2.0 GHz		1.8 GHz
Memory	128 GB DDR4 --16GB reserved for system		32 GB HBM2
Run time	GCC 10.2.0 compiler + MVAPICH 2.3.5 240 cores – 10 nodes, 24 cores/node **0.77 hours**		Cray 10.0.1 SVE compiler + MVAPICH 2.3.5 240 cores – 5 nodes, 48 cores/node **2.08 hours** (fastest run time)

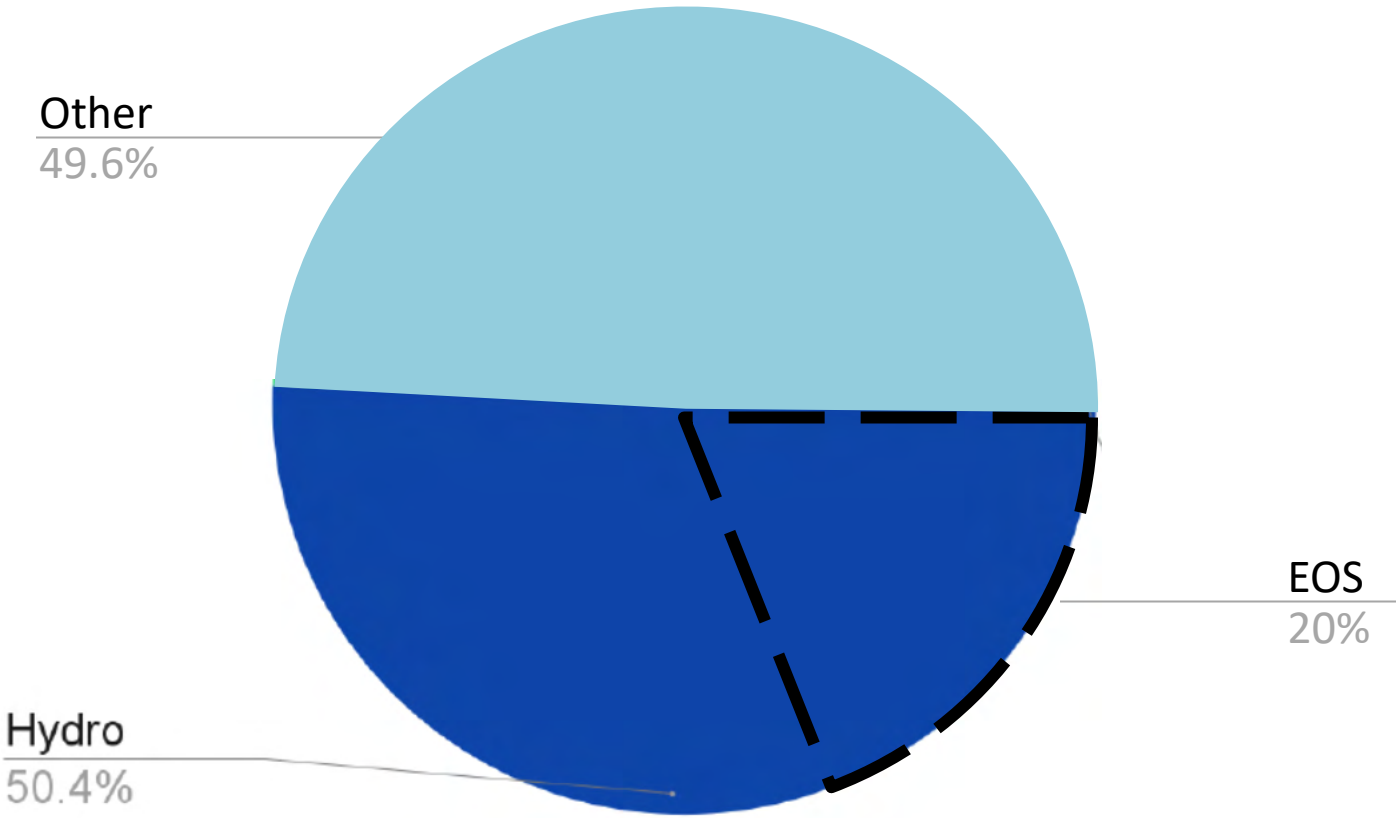
This gap is larger than expected from the difference in processor speed alone

To get the most out of A64FX, need to tune code manually for SVE instructions and use HBM2

Profiling


Particles
9.6%
Burn
5.2%
Flame
9.5%
Hydro
50.4%
Grid Refine
21.1%
Initialization
0.2%
Timestep
0.4%
I/O
3.5%

Profiling



Instrumenting FLASH for PAPI

```
program hellotest

  use region_mod, only: region
  use perf_mod, only: pperf, finalizer

  implicit none

  type(pperf) :: perfmon1

  region = 'Greeting'
  perfmon1 = pperf()

  write(*,*)'Hello, Ookami pals!'

  call finalizer(perfmon1)
  stop

end program hellotest
```

import modules

Create profiler object and set up region

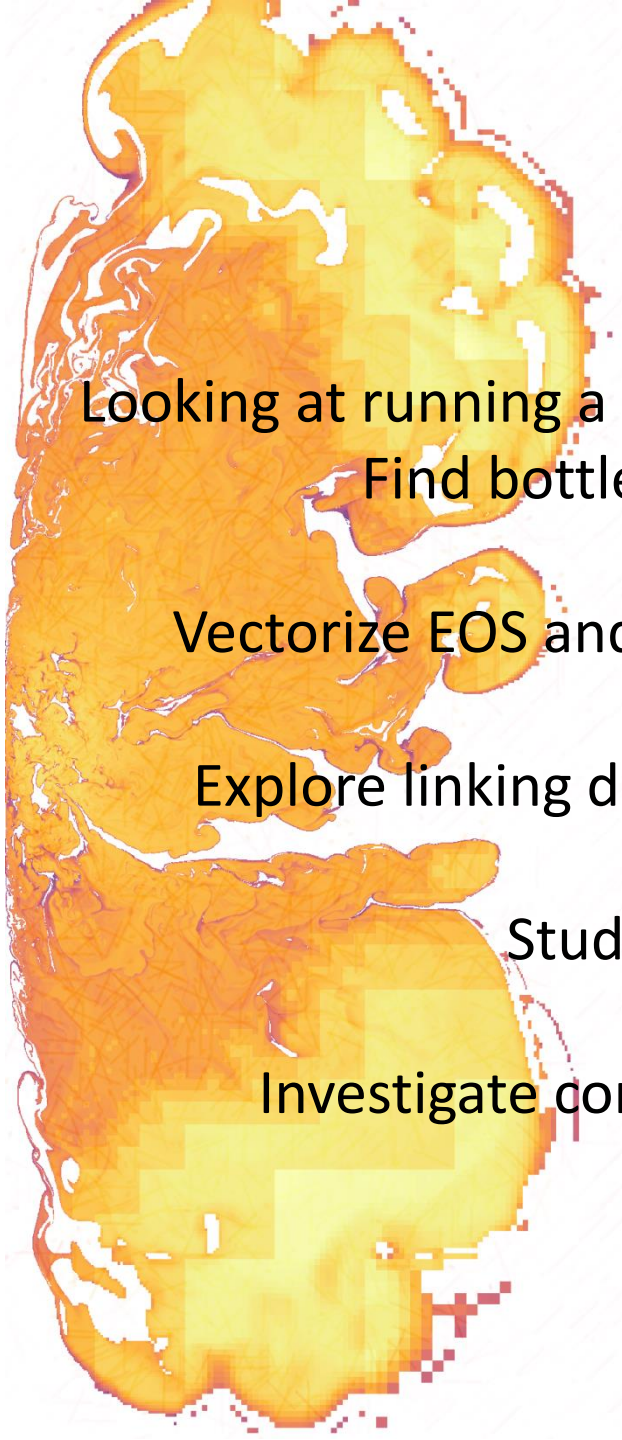
Code you want to profile

Deallocate profiler object

Instrumenting FLASH for PAPI

```
$ python /lustre/projects/calder-group/papiread.py

{'cpu in mhz': '4',
'threads': [{'id': '3004619',
'regions': [{'Greeting': {'region_count': '1',
'cycles': '4161',
'PERF_COUNT_HW_CPU_CYCLES': '23451',
'PERF_COUNT_HW_CACHE_MISSES': '177',
'DTLB-LOAD-MISSES': '86',
'SVE_INST_RETIRED': '0',
'PERF_COUNT_HW_STALLED_CYCLES_BACKEND': '1444',
'PERF_COUNT_HW_STALLED_CYCLES_FRONTEND': '17614'}}]}]}
 thread : 0 3004619
   Greeting:
     HW cycles: 23451
     Seconds: 1.30e-05
     SVE instructions per cycle: 0.00
     Main memory bandwidth (Gbyte/s): 3.48
     DTLB misses/s: 6600997.83
```



Future Work

Looking at running a pure hydrodynamics problem, Sedov explosion
Find bottlenecks using PAPI in both 2D and 3D

Vectorize EOS and other routines for our supernova problem

Explore linking different math libraries to the GCC compiler

Study MPI binding and placements

Investigate communication patterns and memory use

Acknowledgements

- Ookami is a computer technology testbed supported by the National Science Foundation under grant OAC 1927880. The authors are grateful to the Ookami team for their efforts procuring and deploying the machine, for the arduous process of setting up the software used, and for hours of valuable debugging and help.

- The authors would like to thank Research Computing and Cyberinfrastructure, and the Institute for Advanced Computational Science, at Stony Brook University for access to the high-performance SeaWulf cluster, which was made possible by a $1.4M National Science Foundation grant (#1531492).

- The FLASH code was developed in part by the DOE NNSA ASCI and DOE Office of Science ASCR-supported Flash Center for Computational Science at the University of Chicago.

- Work involving supernovae research was supported in part by the US Department of Energy under grant DE-FG02-87ER40317.

Questions, Comments, or Further Discussion?

Get in touch at: catherine.feldman@stonybrook.edu

Thank you for your attention!