OOKAMI PROJECT APPLICATION

Date: September 1, 2021

Project Title: Linear Algebra At Scale

Usage: Testbed

Principal Investigator: Dennis C. Smolarski

- University/Company/Institute: Santa Clara University/Stony Brook University Visiting Scholar (Physics and Astronomy)
- Mailing address including country:
  Department of Mathematics and Computer Science
  Santa Clara University
  500 El Camino Real
  Santa Clara, CA 95053
- Phone number: (408) 309-4434
- Email: dennis.smolarski@stonybrook.edu / dsmolarski@scu.edu

Names & Email of initial project users:

Alan Calder, alan.calder@stonybrook.edu

Usage Description:

The ARM-based A64fx multi-core vector processors combined with ultrahigh-bandwidth memory architecture of Ookami promises to retain familiar and successful programming models while achieving very high performance, particularly for memory-intensive applications such as sparse-matrix solvers that are found in many engineering and physics codes. Our project is to port and tune our codes for Ookami to perform benmark calculations and explore scaling. Eventually the tuning and routines developed during this project will be incorporated in simulation codes for physics and astrophysics applications and thus will enable future science.

An example of a challenging modeling problem is slow, deeply subsonic convection, a situation found frequently in stellar astrophysics. Simulating several convective overturns, as is typically necessary to address the problem, an impossible job if the vastly larger sound speed sets the time steps of the evolution. Thus specialized methods for low-Mach-number flow are required.

Fully implicit methods offer a option to address the problem of subsonic convection as well as other problems involving disparate time scales, e.g. radiation hydrodynamics. Our typical approach to implicit methods is via Newton-Krylov sparse solver techniques for linear systems. As noted above, these numerical methods are a natural for the A64fx architecture and their development will enable a vast amount of science on this type of hardware.

We have previously implemented Newton-Krylov techniques on earlier generations of machines. Figure 1 shows strong scaling results for a multigroup radiation-hydrodynamics code [1] that uses implicit Newton-Krylov techniques for the radiation component. Figure 2 shows weak scaling for another radiation-hydrodynamics code, IBEAM. The solvers from these projects will serve as the basis for the methods to be developed, and the resulting software will be made freely available.

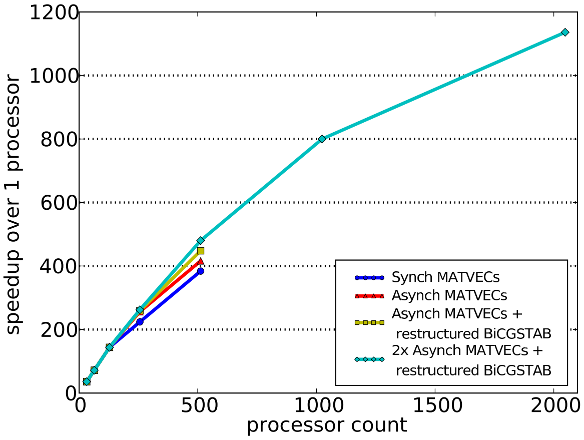

Figure 1: Strong scaling results for V2D, our 2D radiation-hydrodynamics code that employs Newton-Krylov iterative solvers for solving the radiation component. Results were obtained on `seaborg`, an IBM POWER3-based system at NERSC [2, 1].

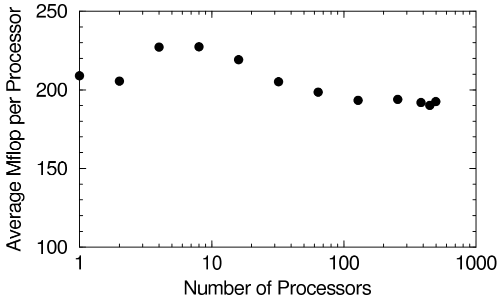

Figure 2: Weak scaling results from IBEAM for simulations of Boltzmann radiation transport (S_8) with relativistic hydrodynamics performed on `lomax`, an SGI Origin system at NASA Ames. The simulation domain was replicated so each processor had an identical domain (with all domains coupled). The largest simulation, 496 processors at 192 Mflop/proc., was 16% of the theoretical peak.

References

[1] F. D. Swesty and E. S. Myra. *ApJS*, 181:1–52, March 2009.

[2] F. D. Swesty and E. S. Myra. *J. Phys.: Conf. Ser.*, 16:380–389, 2005.

Computational Resources:

- Total node hours per year: 20–50 nodes* 1 hour/node * 300 runs = 6,000–15,000 node hours
- Size (nodes) and duration (hours) for a typical batch job: 2–5 nodes for 1 hour
- Disk space (home, project, scratch):
  - Home: 20GB for analysis, visualization, and batch scripts
  - Project: 5 TB for important results from the 300 runs
  - Scratch: 20 TB for the code and output data for 300 runs

Personnel Resources (assistance in porting/tuning, or training for your users):

None

Required software:

1. The V2D code, which we will provide.
2. HDF5
3. MPI
4. Python
5. The Fortran Compilers presently on Ookami

Production projects:

Production projects should provide an additional 1-2 pages of documentation about how

1. the code has been tuned to perform well on A64FX (ideally including benchmark data comparing performance with other architectures such as x86 or GPUs)
2. it can make effective use of the key A64FX architectural features (notably SVE, the high-bandwidth memory, and NUMA characteristics)
3. it can accomplish the scientific objectives within the available 32 Gbyte memory per node