Parallel Scaling Performance of MOOSE on TACC

Alexy Skoutnev and Kevin T. Clarno

University of Texas, Department of Mechanical Engineering
Austin, TX 78712-2002, alexyskoutnev@utexas.edu

ABSTRACT

In this work, the scaling performance between MOOSE and MPI was investigated on the supercomputer, Frontera, located at the University of Texas at Austin. In this performance benchmarking problem, features of parallel computing such as strong and weak scaling metrics was used to evaluate the scale ability of MPI and MOOSE. A custom testing interface conducted performance experiments by varying the workload and processor count along the Frontera computing architecture. The MPI performance model was used as the base work comparison for performance benchmarking MOOSE. We discovered parallel inefficiency within MOOSE from our internode and intranode benchmark studies that caused performance efficiencies to drop below 40%. It was noticed that the inbuilt mesh generation system created serial bounded mesh that allocated more memory than each processor could handle. We concluded that MOOSE's inbuilt mesh generation system was hindering large scaling parallel computing capabilities for MOOSE.

INTRODUCTION

In this work, mathematical operators, for a multi-physics problem, were approximated with the help of parallel computing. Parallel computing resources such as MPI and MOOSE were used to investigate the scaling performance on Frontera located at the University of Texas at Austin. Message Passing Interface (MPI) [1] is a standard parallel programming library that is used to implement parallel communication within a computing architecture. MPI allows large computer architectures to efficiently communicate with each of its cores to maximize its computational power. Multiphysics Object-Oriented Simulation Environment (MOOSE) [2] is a parallel finite element framework used to solve a range of partial differential equations driven physics problems. In this paper, MOOSE was utilized to approximate the linear and non-linear heat conduction problem via a simple 3 dimensional cube. Performance test statistics such as strong efficiency, and weak efficiency were performed to evaluate the scale ability of MPI and MOOSE. With performance benchmarks, a upper computational bound to multiphysics problems is known and uncovers potential ways to optimized high-performance code for more accurate simulations. The assessment of the scaling performance of MPI and MOOSE was modified from https://github.com/mpitutorial and https://github.com/idaholab/moose on GitHub.

REQUIREMENTS

MOOSE's internal module, Heat Conduction, was used to find and solve the steady linear, and non-linear heat conduction equation. The MOOSE application code is mostly written in C++ while being compiled and executed with the use of makefiles and shell scripts. The MPI performance test was written in C while being compiled and executed with shell scripts.

- Evaluate the scale performance of a simple MPI program to benchmark the performance of MPI on Frontera.
- Numerically approximate the heat conduction problem using finite element principles.
- Model the weak and strong scale ability of MOOSE using MPI on Frontera.
- Perform scalability on linear and non-linear heat conduction via weak and strong scaling.
- Model the key performance differences between MPI scaling and MOOSE scaling.

BENCHMARKING MPI PERFORMANCE ON FRONTERA

To establish a control sample to compare performance tests between different parallel computing libraries, the MPI parallel programming library was subject to strong and weak scaling. With MPI implemented, a simple C program calculated the average value from a set of random values between 0 and 1 where each processor computed it own average value. The simple MPI program was used to benchmark the scaling performance on Frontera. The two benchmarks were focused on intranode testing and internode testing. Figure 1 displays the intranode scaling benchmarks varying its 56 cores.

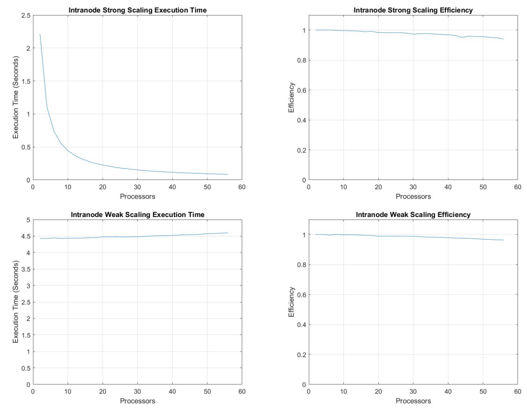

Fig. 1: Each intranode performance benchmark computed the average of a 2.327×10^8 sized array.

For internode performance testing as seen in Figure 2, each node utilized all of its fifty six processors to push its hardware to its full potential. The scaling performance test started with one node and incremented up by one node until the upper memory bound is reached on a single processor.

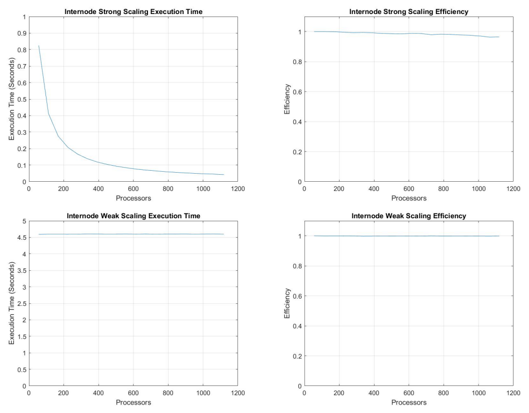

Fig. 2: Each internode performance benchmark computed the average of a 2.327×10^8 sized array.

DEFINING A DIFFUSION TEST PROBLEM

The heat conduction equation was used as the basis for the diffusion performance experiment. The heat conduction equation is derived as,

$$\nabla \cdot k\nabla T = 0 \tag{1}$$

Where k is the thermal conductivity, and ∇T is the temperature gradient in terms of spatial variables x, y, z. Linear and non-linear partial differential equations were tested to compare scaling performance between them. For non-linearity, the thermal conductivity was coupled as a simple function depending on the temperature in the medium. The thermal conductivity in the non-linear problem was defined as,

$$k(T) = 5 + 10T \tag{2}$$

Where in the linear case, the thermal conductivity was set to a constant value of 18. The performance test made use of MOOSE's optimized automatic differentiation kernels for better performance. The problem was a finite element problem that made use of the Newton-Raphson method to approximate the solution for the partial differential equation. For optimal parallelism performance, the DistributedRectilinearMeshGenerator object was used as the sample mesh since MOOSE deleted parts of the mesh not assigned to the working processor.

BENCHMARKING MOOSE PERFORMANCE

To compare against the MPI baselines, MOOSE was subject to strong and weak scaling performance by varying the mesh size and processor amount under the heat conduction kernel. The strong scaling benchmark utilized a 200^3 partitioned cube mesh and the weak scaling benchmark size started with a 90^3 partitioned cube mesh in the weak scaling sequence. Figure 3 displays the linear heat conduction intranode study between strong and weak scaling.

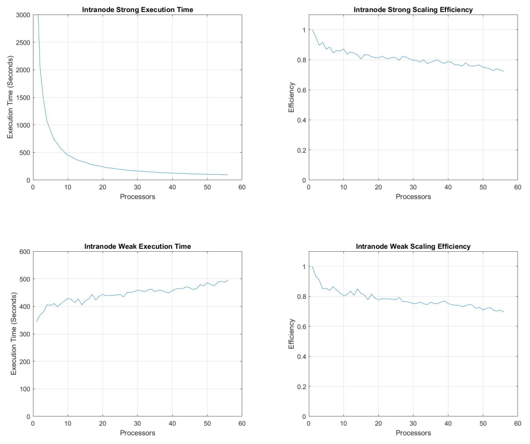

Fig. 3: Linear Heat Conduction Intranode Scaling

To test the full scale parallel computing capabilities of MOOSE, a internode benchmark study was performed. Equivalent to the intranode case, the problem sizes were the same for strong and weak scaling. MOOSE's parallel computing capabilities across multiple nodes is seen in Figure 4.

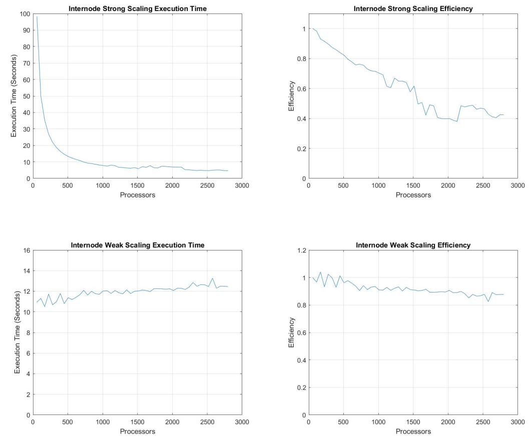

Fig. 4: Linear Heat Conduction Internode Scaling

With the increase of cores, more communication bias and parallel inefficiency in MOOSE started to become noticeable along the performance benchmarks. Through a desire for additional exploration, a similar study was done to the nonlinear diffusion problem resulting in a more intensive computational heavy workload, as seen in Figure 5.

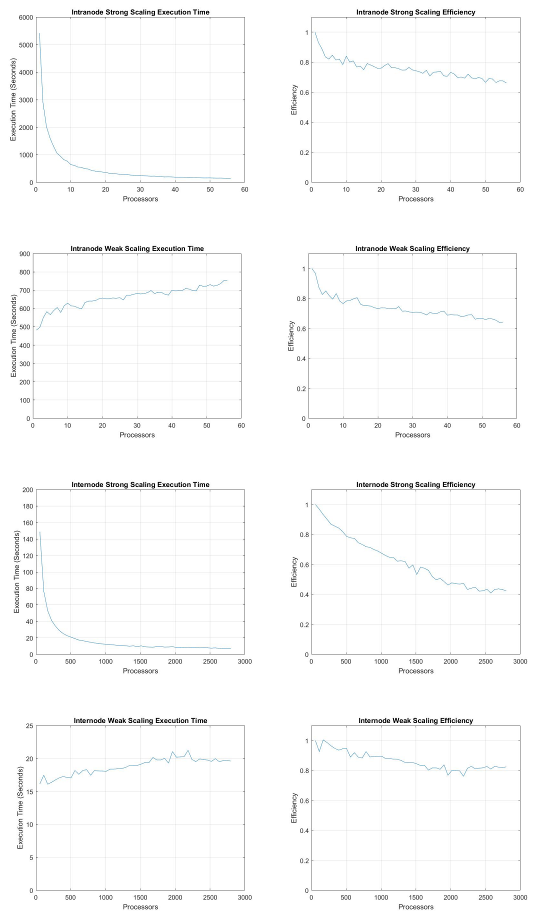

Fig. 5: Nonlinear Heat Conduction Problem Scaling

CONCLUSIONS

The MPI benchmarks was used as the base comparison in this study because the scaling efficiency was near perfect for all the scaling benchmarks. In relation to the MPI benchmarks, MOOSE did not perform to MPI's ideal scaling performance. Two major drawbacks in scaling performance within MOOSE were the inability to create a proper parallel mesh and MOOSE's communication class not scaling when collecting all the data vectors together. Across all the performance tests, MOOSE's internal parent classes scaled better than the MOOSE's customized classes such as the heat conduction kernel. There was no difference in efficiency between the linear and nonlinear diffusion problem except that the solver execution time doubled in some cases. Each diffusion problem (linear or nonlinear) was able to be solved in reasonable amount of time in MOOSE however non-linearity did not increase the efficiency as hoped. ParMETIS combined with DistributedMeshGenerator mesh object was the optimal configuration for loading and splitting the mesh. In most cases, a single node utilizing its 56 cores were sufficient enough to solve the 200^3 element diffusion problem with the help of ParMETIS and a distributed mesh. The most substantial problem with the parallel scaling performance of MOOSE is its serial structured internal mesh objects that causes major slowdowns in the scaling benchmarks as indicted in the performance figures in the study.

REFERENCES

1. MESSAGE PASSING INTERFACE FORUM, *MPI: A Message-Passing Interface Standard Version 4.0* (Jun. 2021).
2. C. J. PERMANN, D. R. GASTON, D. ANDRŠ, R. W. CARLSEN, F. KONG, A. D. LINDSAY, J. M. MILLER, J. W. PETERSON, A. E. SLAUGHTER, R. H. STOGNER, and R. C. MARTINEAU, "MOOSE: Enabling massively parallel multiphysics simulation," *SoftwareX*, **11**, 100430 (2020).