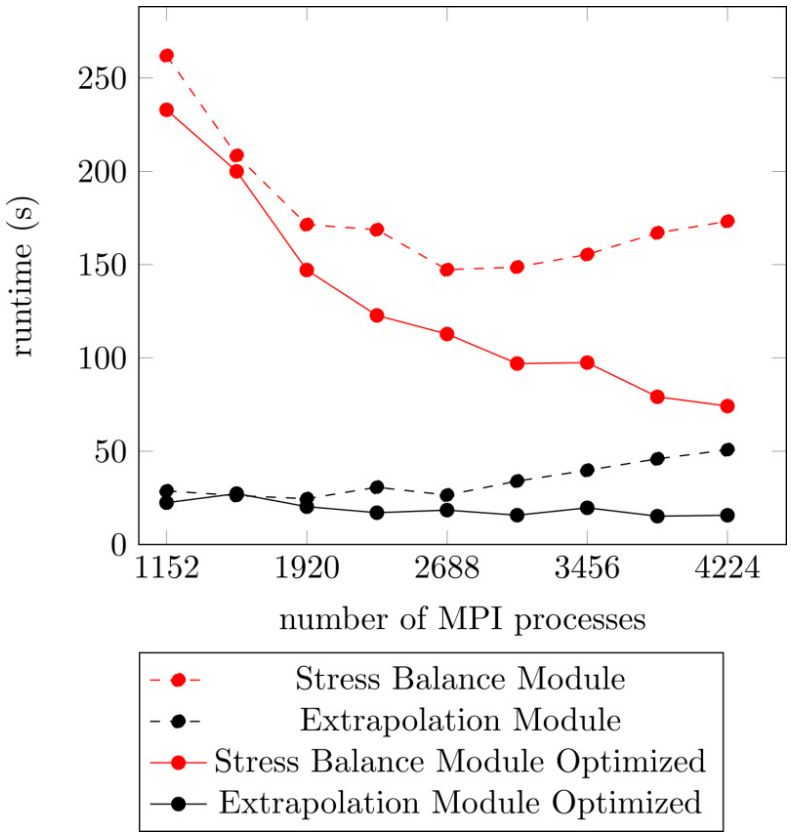
Optimized Scaling of ISSM



Project Manager
Yannic Fischler

Researchers
Jonathan Otto and Raban Emunds

Principal Investigator
Prof. Dr. Christian Bischof

Project Term
2021 - 2022

Clusters
Lichtenberg Cluster Darmstadt

Additional Software
OpenMPI, ISSM, GCC, PETSc, Score-P, Open-MP, kokkos

Institute
Institute of Scientific Computing

University
Technische Universität Darmstadt

Introduction

In previous project we investigated on the scaling and the load balancing of the Ice-sheet and Sea-level System Model (ISSM), which is a ice-sheet modeling framework using process parallelism through the Portable, Extensible Toolkit for Scientific Computation (PETSc). We showed, that the code scales up to approximately 3000 MPI processes on realistic setups as the Greenland ice-sheet and observed performance potential by improved load balancing. The scaling limit of ISSM is also visible in Figure 1. Here we plot the runtime of the Stress Balance Module (red) and the Moving Front Module (black). The dashed lines show the vanilla version and the solid lines show our optimized version. The Stress Balance Module is the most expensive module of ISSM and the Extrapolation Module scales worst and thereby becomes significant on large calculations. Therefore both are worth to investigate on.
In this project we tried to overcome the scaling limits of ISSM by optimized usage of PETSc and hybrid parallelism (MPI + threads). We used OpenMP and kokkos for efficient and portable shared memory parallelism. We run realistic Greenland setups of different resolution from 250 m up to 4.000 km. Both were used in previous studies.

JUSTUS-LIEBIG-UNIVERSITÄT GIESSEN
TECHNISCHE UNIVERSITÄT DARMSTADT
UNIKASSEL VERSITÄT
Philipps Universität Marburg

Methods

We execute all timings on the Lichtenberg HPC system using modern compiler and highest optimization level. We execute ISSM using up to 4224 MPI processes on 88 nodes and a maximum of 24 threads per MPI process, because each Non-Uniform Memory Access (NUMA) domain of the Lichtenberg system contains 24 cores. Our runtime measurements are done using the Score-P performance instrumentation infrastructure and the timings are evaluated using the tool Cube. To detect inefficiency in the usage of PETSc we wrote our own tool and applied it during the runtime of the code. Finally, we manually optimized the code.
We identified code regions for thread parallelism by manual analysis and implemented multiple parallel schemes. Race conditions were identified and fixed manually.

Results

Our newly developed tool reported important regions, which we were able to optimize. As shown in Figure 1 the scaling of the Stress Balance and the Extrapolation Module of ISSM were improved significantly by optimizing the usage of PETSc. Furthermore the runtime improved significantly even on small partitions (1152 MPI processes). In the previous version the Stress Balance Module scales approximately linear up to 1920 MPI processes and reaches its throughput maximum on 2688 MPI processes. On the other hand the optimized version scales approximately linear up to 4224 MPI processes which is a significant improvement of at least doubling the throughput potential of this code region. The runtime of the Extrapolation Module raises on large computation partitions, when we execute the original ISSM. Our improved implementation at least constant runtime, which is a significant improved, because now the runtime does not get worse and the entire program can benefit from good scaling of other regions, e.g. Stress Balance Module.
In the second part of this project we put effort into hybrid parallelization. As PETSc is not a thread safe library we found the concept of multiple threads generating data stored in a queue and one thread reading data from the queue and calling PETSc, the most effective parallelization. This concept is visualized in Figure 2. This scheme allows the usage of additional thread parallelism in the compute intensive regions of ISSM and thereby improves the scaling and the maximum throughput. The major challenge is to find all race conditions in ISSM, because it is not designed as a thread safe framework. A second opportunity in using hybrid parallelism is adapted load balancing, which we want to exploit in the future.

Discussion

In this project we have shown two approaches to overcome the scaling limits of ISSM. Both hybrid parallelization and the reuse of large parallel data structures show in practical examples that scaling beyond the previous limits is possible. This will make it possible to achieve higher throughput in the future.

Our newly developed tool is helpful and very efficient in finding recurring data structures. In future projects, the approach can be expanded to include additional functionality.

Figures

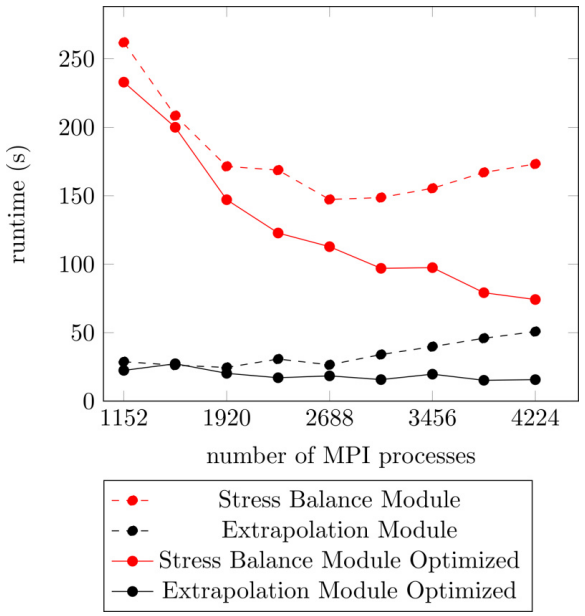


Figure 1: Scaling of ISSM Stress Balance & Moving Front Module (Setup: G250)

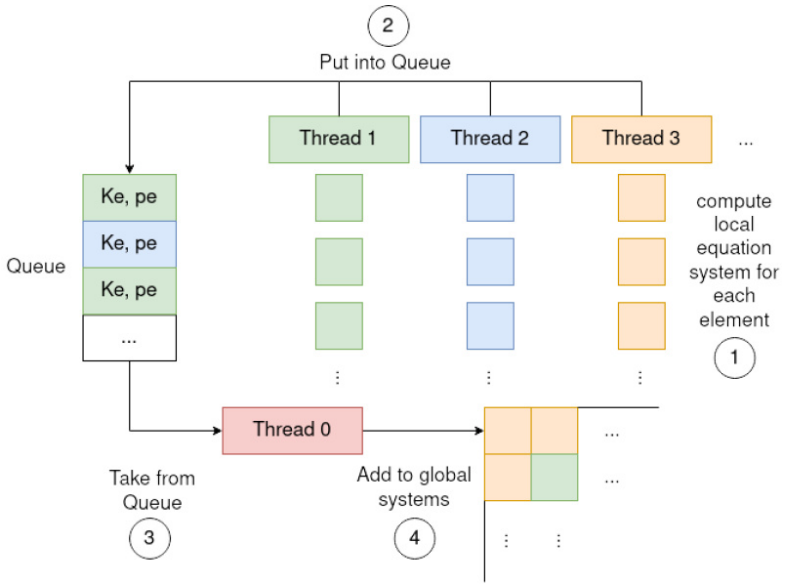


Figure 2: A concept for efficient thread parallelism in ISSM

Last Update: 2024-02-27 17:44