GEMMA Electromagnetic Code and ADELUS - New Capabilities

Joseph D. Kotulski, Vinh Dang 1352

jdkotul@sandia.gov, vqdang@sandia.gov

Trilinos User Group Meeting 2022

October 25, 2022

Sandia National Laboratories

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

SAND2022-14841 C

GEMMA Description

- Frequency-domain method of moments solution
  - Steady state solution
  - With specialized algorithms (thin-slot, etc.)
- Boundary element formulation
  - Mesh surfaces of parts – interface between regions
- Exact radiation boundary condition
  - Due to Green's function
- Formulation results in dense (fully populated) matrix → **HPC**
  - Simulations can be limited by available memory
  - Entries are double precision complex
- Code has been ported and used for ND problems on CTS1, ATS1, and ATS2

The next generation version of EIGER

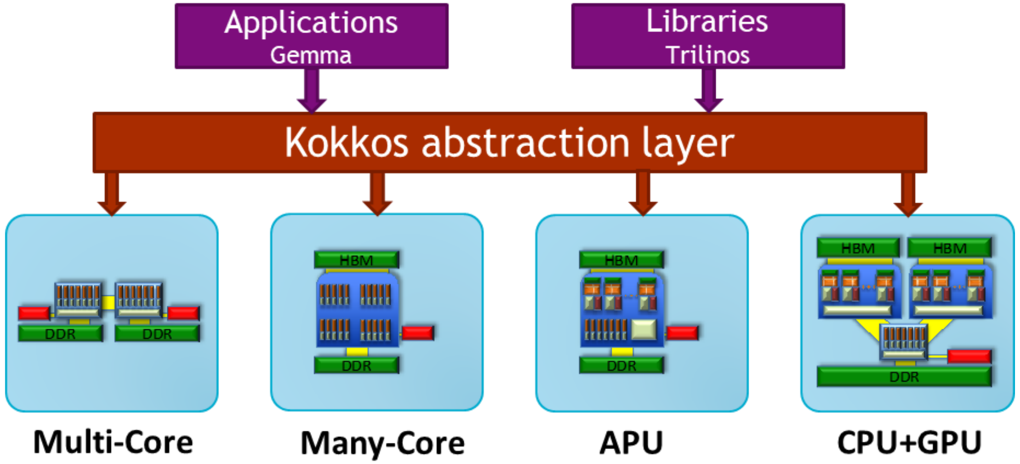
Capability on Next-Generation Hardware

eiger
FORTRAN
MPI Paradigm

- MPI inter- and intranode parallelism
- High processor clock speed
- High memory per processor

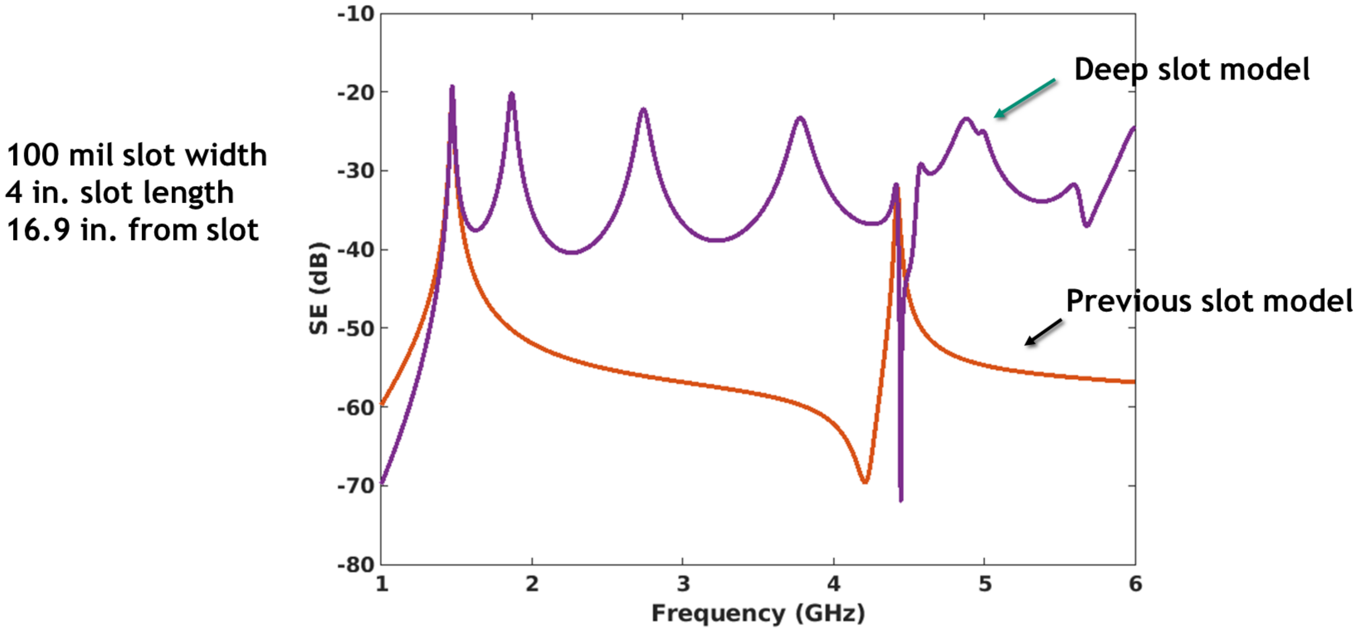
Gemma
Heterogenous
Paradigm

- MPI internode parallelism
- Threading intranode parallelism
- Low processor clock speed
- Low memory per processor



GEMMA – NEW FEATURES

- Improved slot algorithm
  - Takes into account the depth resonance of the slot

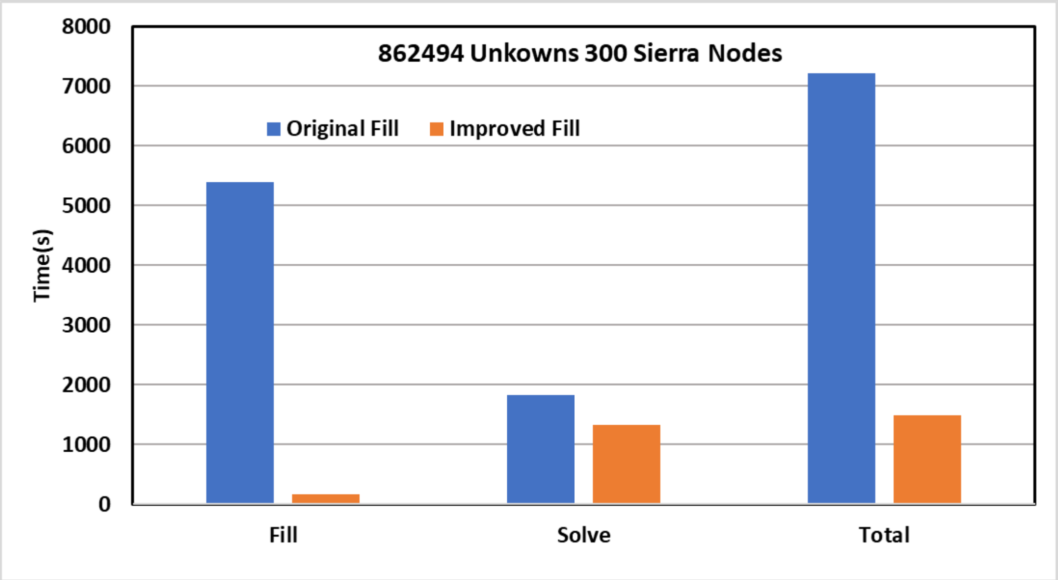


- Power Balance
  - Simplified power calculations to determine the high-frequency response.

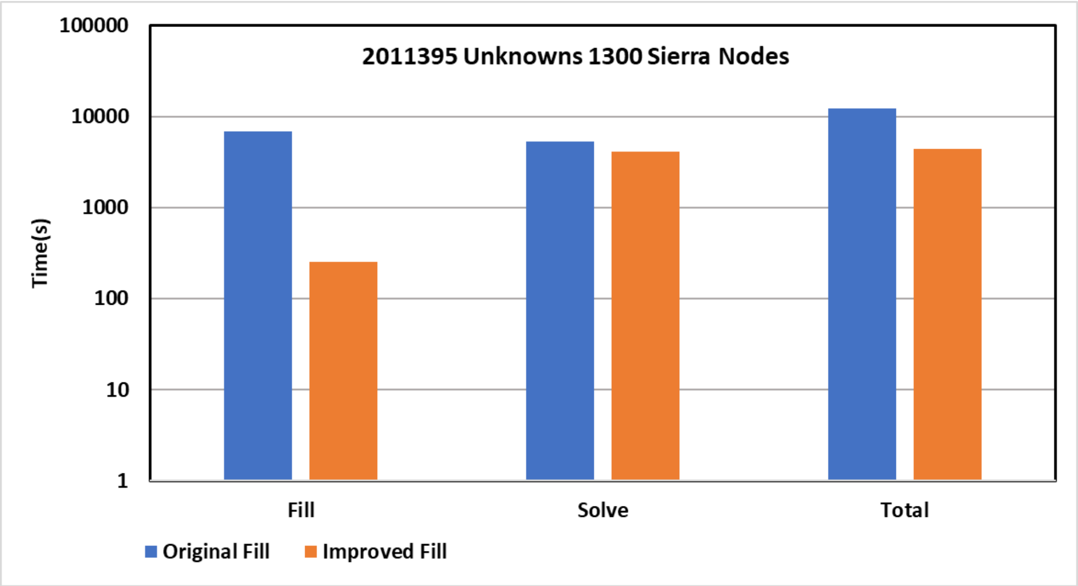
GEMMA – NEW FEATURES

- Rational Interpolation
  - Algorithm to locate peaks - important for calculation of electromagnetic coupling

- Matrix fill algorithm improved
  - Fill by unknowns (i and j) instead of by elements

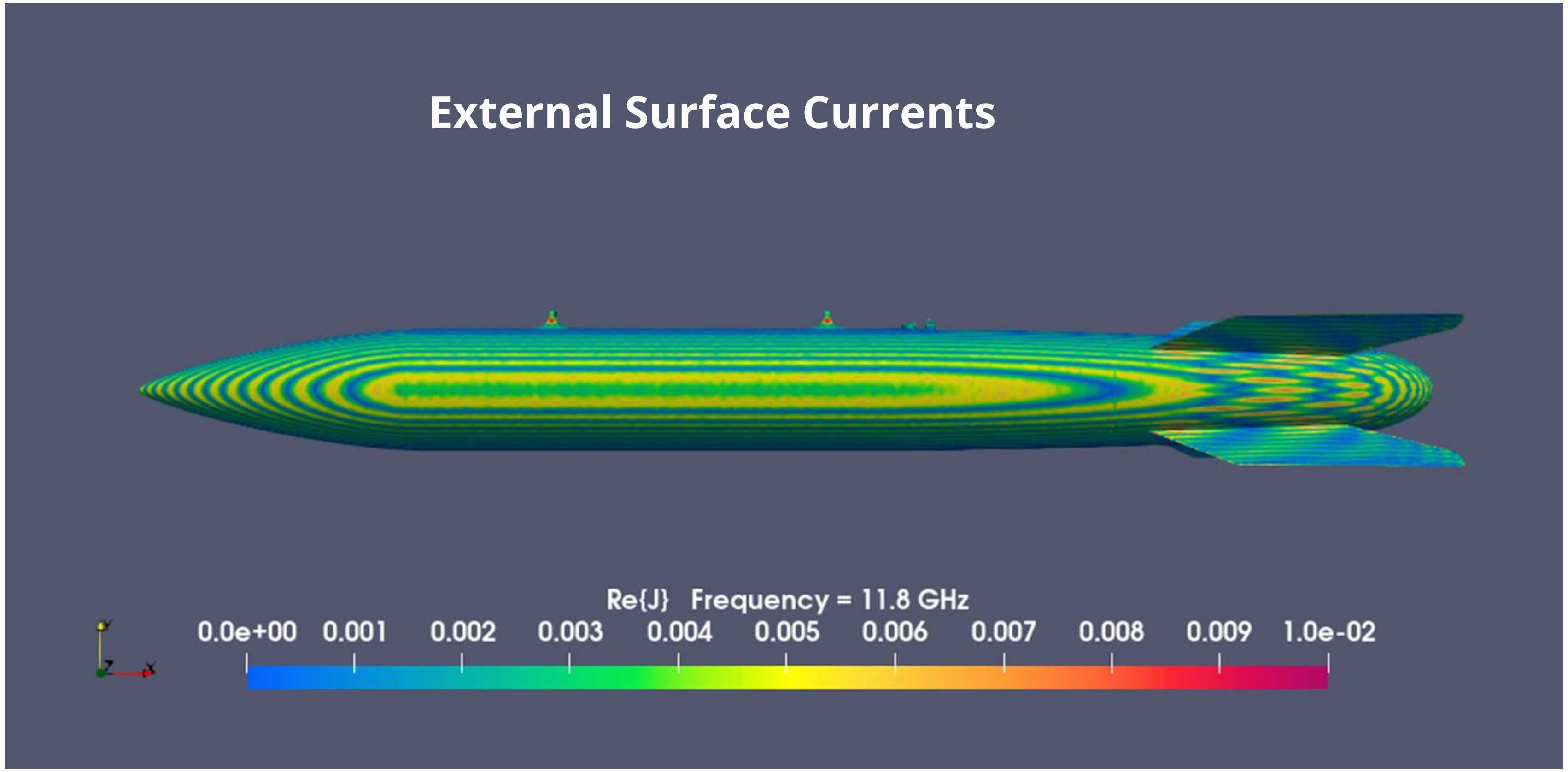


33 x Speedup



27 x Speedup

GEMMA – Example Problem (2 million unknowns)



GEMMA – Future Solver Development

- Preconditioner development
  - Matrices have behavior much different than what is experienced with FEM solvers.

- Compression Techniques
  - Reduce the memory footprint
  - Iterative solution using BELOS

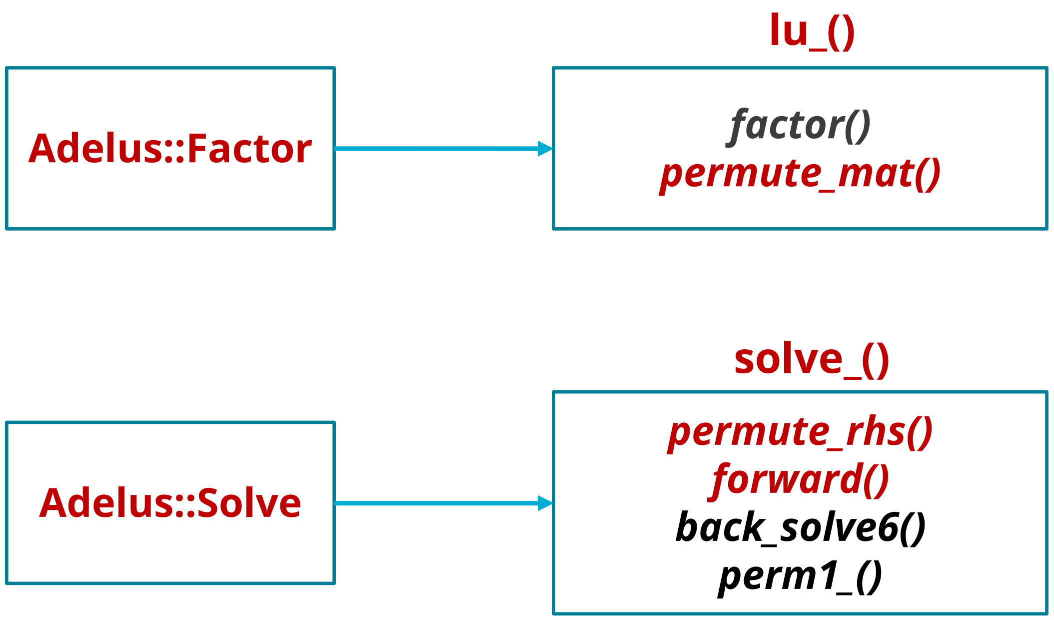
- Combining the above concepts

ADELUS – AMD HIP Backend Support

- Necessary changes made for code and CMake to support HIP backend
- Trilinos configuration:
    - hipcc compiler
    - Architecture flag for Kokkos (For MI100: Kokkos_ARCH_VEGA908=ON)
    - Kokkos_ENABLE_HIP=ON
    - KokkosKernels_ENABLE_TPL_ROCBLAS:BOOL=ON/OFF
- rocBLAS wrappers for GEMM, IAMAX, and SCAL kernels to Kokkos Kernels
- Future work: evaluate ADELUS performance on Crusher/Frontier (ORNL)

ADELUS – Factor and Solve Interfaces

- ADELUS previously only provided LU factorization and solve via a single interface Adelus::FactorSolve (matrix + RHS packed together)
- Create two separate interfaces which are useful for applications that (i) *do not have RHS at the time of factorization* OR (ii) *need to solve different RHSs with a pre-factorized matrix*
  - **Adelus::Factor**: LU factorization
  - **Adelus::Solve**: forward solve + backward solve
  - Support execution on GPUs and multiple RHS vectors



ADELUS – General Communicator and Global Variables Removal

- Enable ADELUS to run on an arbitrary communicator rather than MPI_COMM_WORLD
  - Create sub-communicators and launch Adelus to solve many linear equation systems
- A new class, AdelusHandle, contains:
  - a communicator, global variables, constructor and methods to retrieve these variables
- A handle needs to be created and passed through Adelus interfaces from application code

Adelus

```
class AdelusHandle {
 private:
  //Comm. variables and used-to-be global variables
  int my_rows;        // num of rows I own
  int my_cols;        // num of cols I own
  int my_rhs;         // number of RHSs that I own
  …
  MPI_Comm row_comm;  // row sub-communicator
  MPI_Comm col_comm;  // column sub-communicator
  MPI_Comm comm;      // communicator that I belong to

 public:
  AdelusHandle (MPI_Comm comm_, const int matrix_size_,
const int num_procsr_, const int num_rhs_, …) {
  //Calculate global vars and create row and col sub-comms
     …
  }
  KOKKOS_INLINE_FUNCTION
  MPI_Comm get_comm() const { return comm; }
  …
  KOKKOS_INLINE_FUNCTION
  int get_my_rows() const { return my_rows; }.
};
```

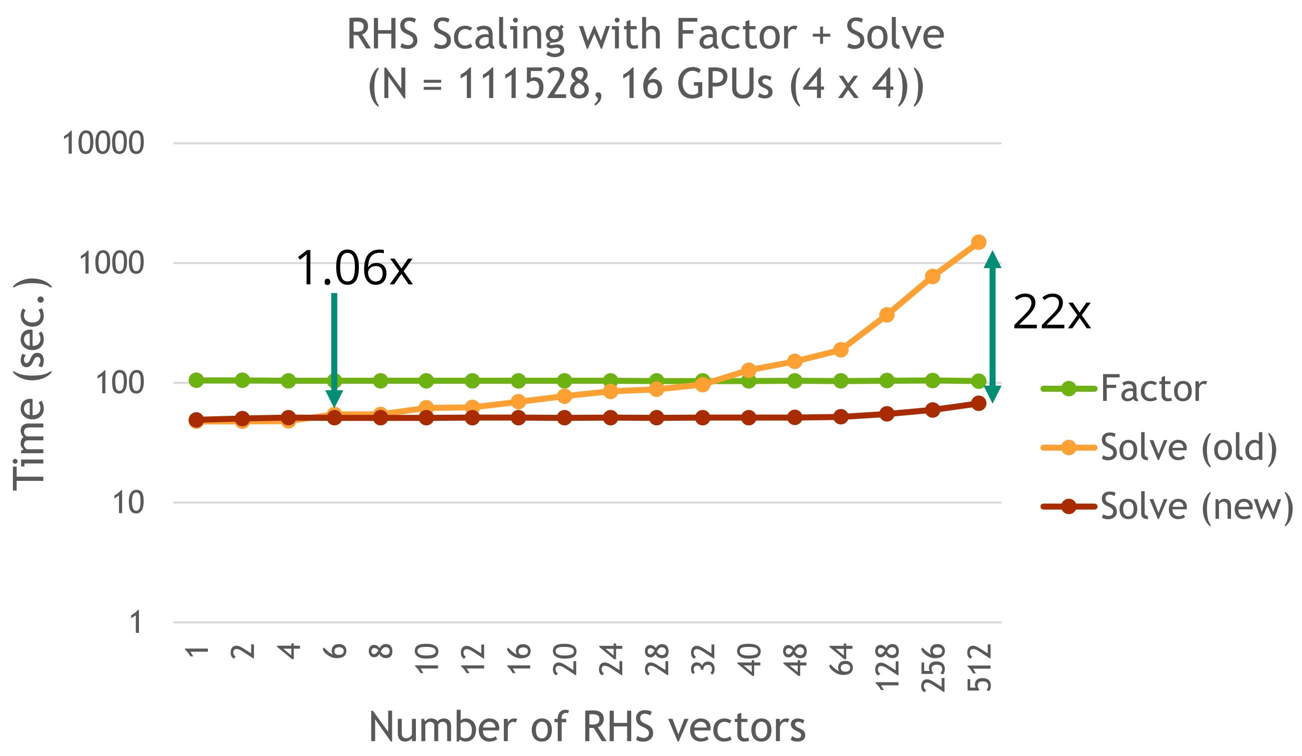
Application

```
//Create handle
Adelus::AdelusHandle ahandle(my_color, sub_comm,
matrix_size, nprocs_row, nrhs );

//Pass through Adelus interfaces
Adelus::Factor (ahandle, my_A, h_permute, &secs);
```

ADELUS – Backsolve Performance Improvement

- Issue: backsolve previously did not scale well with large numbers of RHS vectors
  - Using pipelined communication for the whole RHS mutivectors across MPI processes at each column iteration
- Improvement:
  - **Broadcasting only one** current active column within row communicators at each iteration → communication overhead is significantly reduced



ADELUS – Future Performance Improvement

- Allow using tile size greater than 1
- Allow using mixed-precision