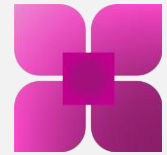
bwRSE4HPC

An Interface from M++ to Ginkgo for Accelerated Linear Algebra

Tim Schrader, Niklas Baumgarten, Marcel Koch

Ginkgo in M++: Why?



M++

- M++ (Meshes, Multigrid and more) is a parallel finite element-program for partial differential equations (PDEs)
- M++ is advanced but up to 80% computational time is linear solver

Ginkgo

- Ginkgo is a linear algebra backend that runs on CPUs and GPUs (OpenMP, CUDA , HIP, SYCL/DPC++)
- Ginkgo offers a large number of highly configurable solvers
- Ginkgo is maintained by an active community

Ginkgo in M++: Contributions

- Preliminary work by Suryansh Chaturvedi under Niklas Baumgarten
- Integration of Ginkgo into M++ build system and CI
- `Vector` and `Matrix` can convert to Ginkgo types
- `GinkgoSolver` implements `LinearSolver` base class
- Using Ginkgo's MPI-distributed data structures.
- Tests and benchmarks

Ginkgo in M++: Config

```
# ginkgoConf.conf

# Solver settings: must be ginkgo
LinearSolver = ginkgo

# Preconditioner must be "NoPreconditioner"
Preconditioner = NoPreconditioner

# reference (serial CPU), omp (parallel CPU),
# cuda (Nvidia GPU), hip (AMD GPU), or sycl (Intel GPU)
executor = reference

# Ginkgo solver configuration file
GinkgoConfigFile= ginkgoDefault.json
```

ginkgoDefault.json

```
{
  "type": "solver::Gmres",
  "criteria": [
    {
      "type": "Iteration",
      "max_iters": 10000
    },
    {
      "type": "ResidualNorm",
      "reduction_factor": 1e-12
    },
    {
      "type": "ResidualNorm",
      "baseline": "absolute",
      "reduction_factor": 1e-14
    }
  ],
  "preconditioner": {
    "type": "preconditioner::Schwarz",
    "local_solver": {
      "type": "preconditioner::Jacobi"
    }
  }
}
```

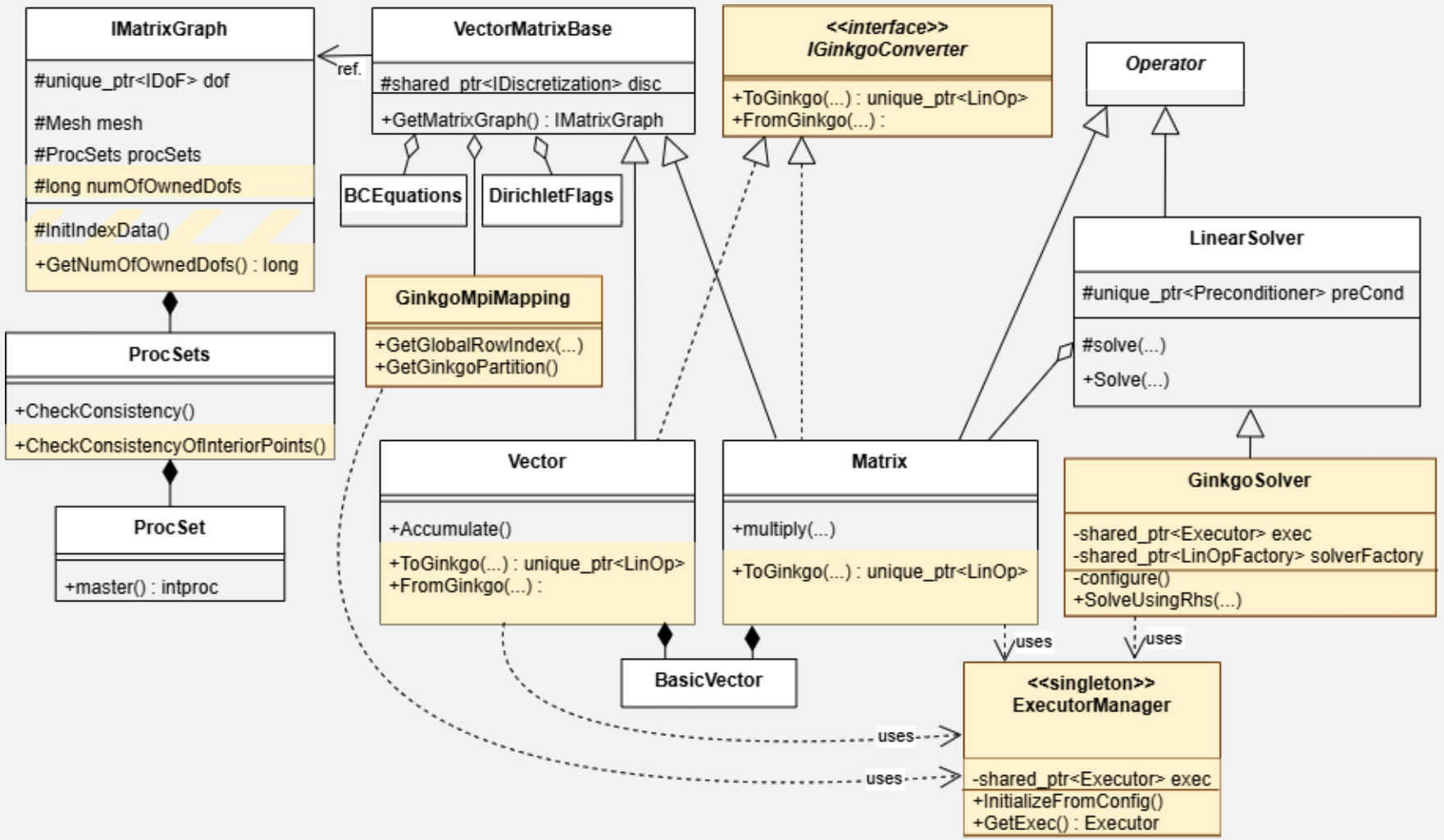
Ginkgo in M++: General Implementation Details

- Ginkgo is needed as a library (not submodule)
- `#ifdef USE_GINKGO` guards keep Ginkgo optional
- LinearSolver mismatch
  - M++ LinearSolver takes residual
  - Ginkgo expects the right hand side
  - -> conversion necessary

Ginkgo in M++: MPI Implementation Details

- `Matrix` and `Vector` implement `IGinkgoConverter` with `toGinkgo` and `fromGinkgo`
- M++ : Overlapping domain decomposition, local index
- Ginkgo: Non-overlapping Partitioning, global index
- `GinkgoMpiMapping` builds and holds data for conversion `GetGlobalRowIndex(), ginkgoPartition, …`

Ginkgo in M++: Contributions

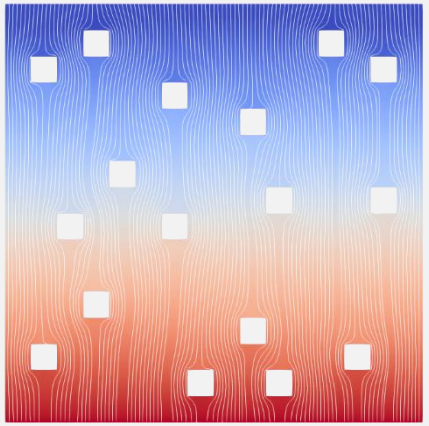


Ginkgo in M++: Vector Conversion Implementation Details

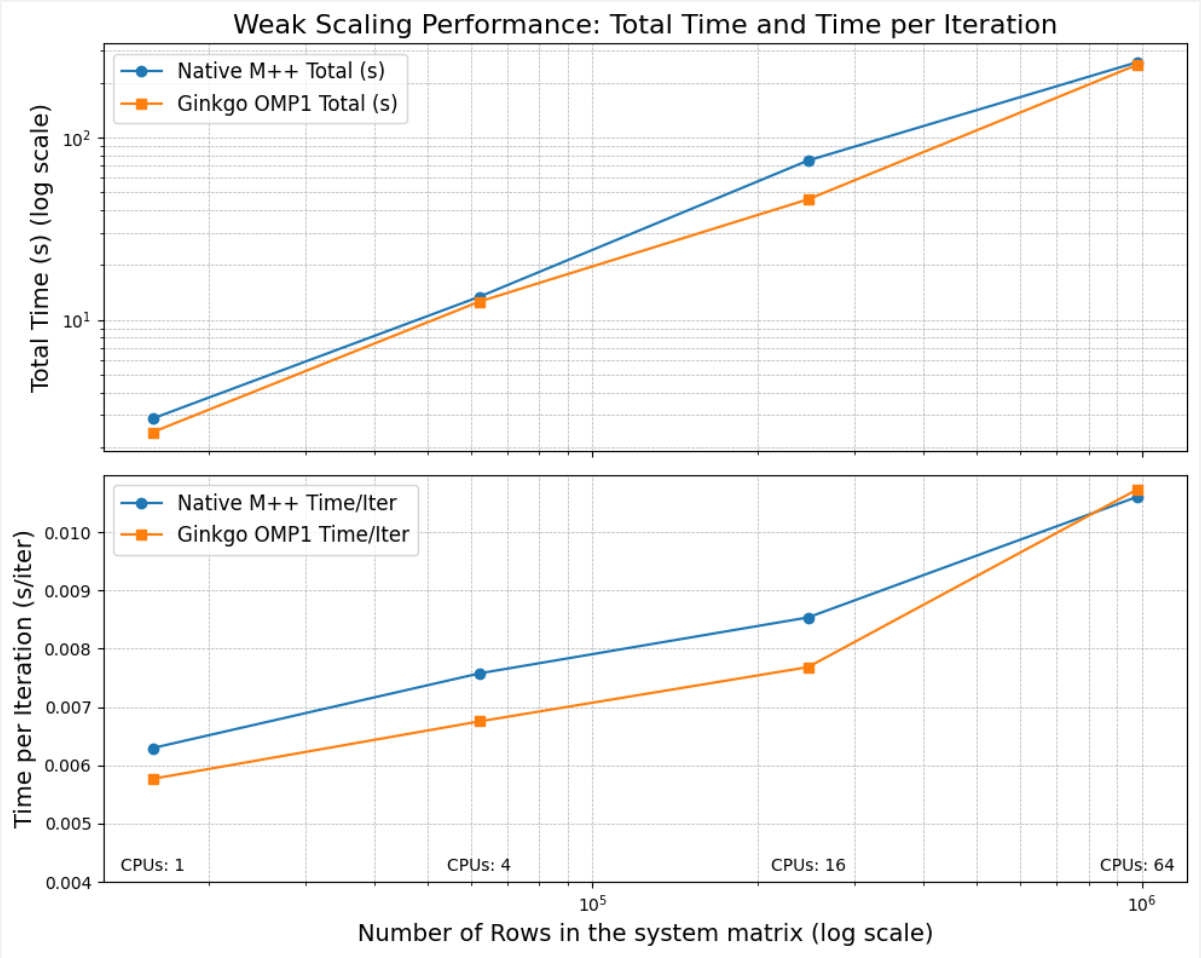
```
std::unique_ptr<Vector::GinkgoVector>
Vector::toGinkgoDistrVec(const std::shared_ptr<gko::Executor> &exec, GinkgoCommunicator *comm) {
  const size_t numOwnedDofs = this->graph.GetNumOfOwnedDofs();
  const size_t globalNumOfDofs = ginkgoMpiMapping->GetGlobalNumOfDofs();
  auto hostExec :shared_ptr<Executor> = exec->get_master();
  auto hostView :array<double> = gko::array<double>::view(hostExec, numOwnedDofs, this->data.data());

  // - If exec is GPU: Creates new memory on GPU and copies data from hostView.
  // - If exec is CPU: Uses the hostView directly (zero-copy).
  auto localVec :unique_ptr<Dense<>> = gko::matrix::Dense<double>::create(exec, size: gko::dim<2>(numOwnedDofs, 1),
                                                 std::move(hostView), stride: 1);
  // Wrap in Distributed Vector and return
  return GinkgoVector::create(exec, *comm, global_size: gko::dim<2>(globalNumOfDofs, 1), std::move(localVec));
}
```

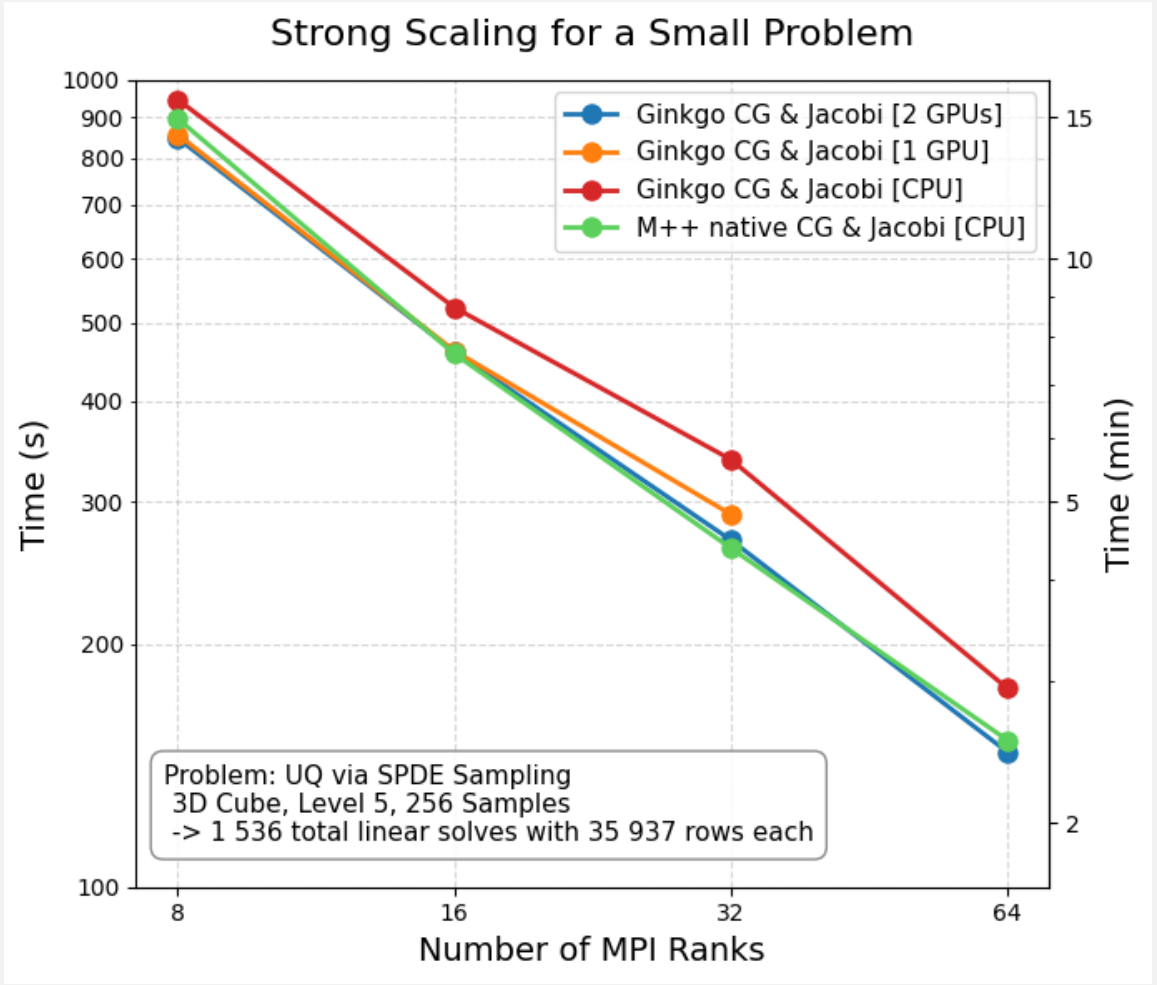
Ginkgo in M++: Benchmarks

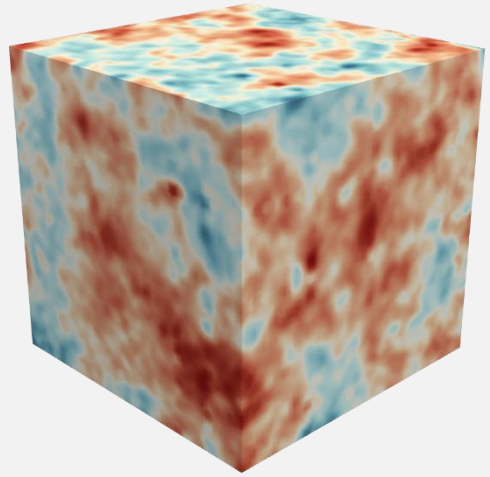


2D Laplace Problem
- Bad choice: GMRES with Jacobi
 ~24000 iterations
-> 10^-5 accuracy
=> Shows only linear solver performance



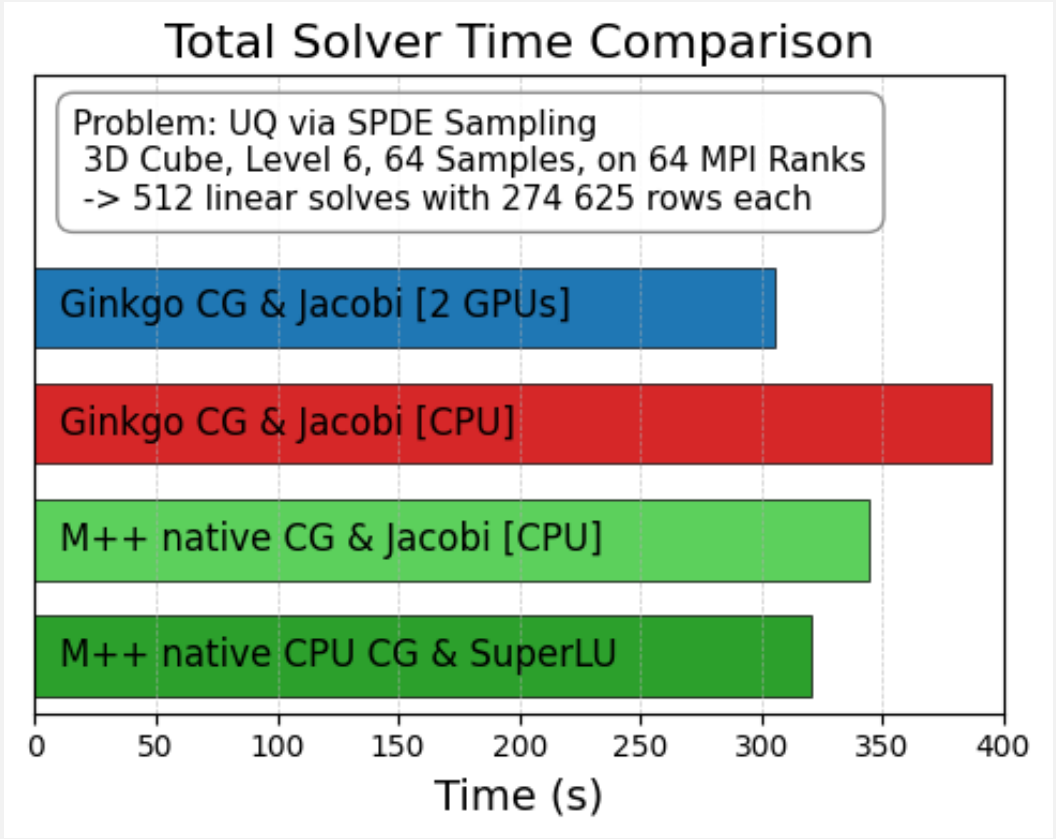
Ginkgo in M++: Benchmarks

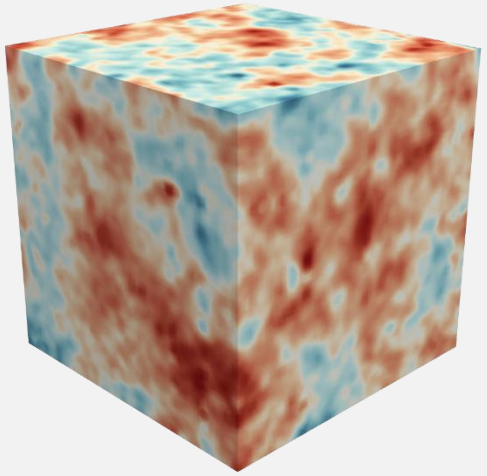




- Small conversion overhead exists
- Oversubscribing GPUs works
- Small problem does not fully use GPUs

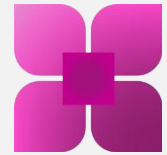
Ginkgo in M++: Benchmarks





- At larger scale GPUs are worth the overhead & different preconditioner

An Interface from M++ to Ginkgo for Accelerated Linear Algebra



M++ gitlab.kit.edu/kit/mpp/mpp/

Ginkgo ginkgo-project.github.io/

bw RSE 4 HPC www.bwrse4hpc.de/

support@bwrse4hpc.de

Any questions or comments ?