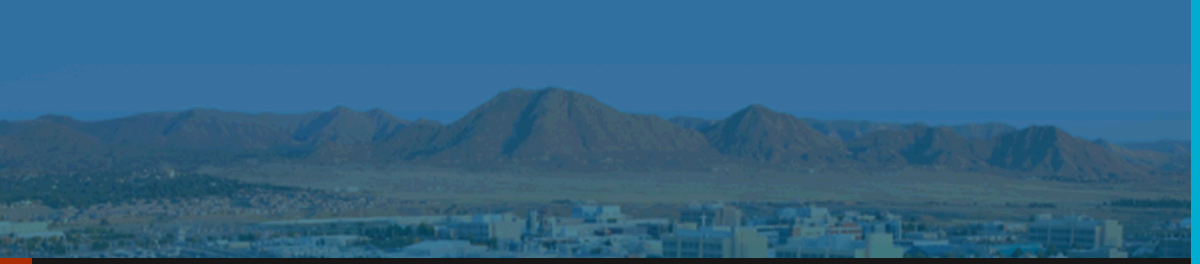
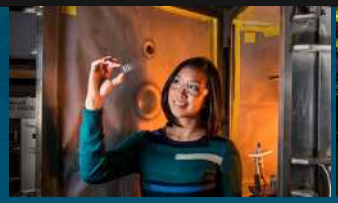


SAND2018-1801C

Plato Analyze: GPU-Based simulation tools in Plato



PRESENTED BY

Joshua Robbins

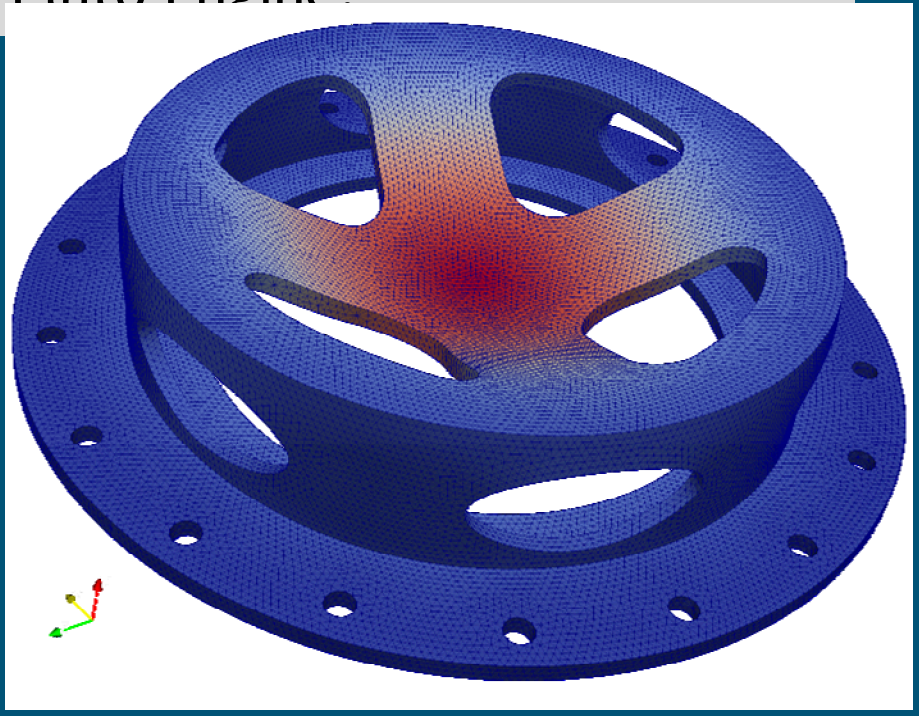
Introduction

Plato Engine provides the basic services necessary for coordinating multiple independent applications into a single solution.

Plato Analyze is a fast, extensible, and lightweight Partial Differential Equation (PDE) solver for production use and algorithm development within Plato Engine.

Outline:

- Objective
- Approach
- Results
- Summary



Objective

Given an optimization problem:

Objective:	$\min_{\boldsymbol{z},\boldsymbol{x}} f(\boldsymbol{u},\boldsymbol{z},\boldsymbol{x})$
PDE Constraint:	$\boldsymbol{g}(\boldsymbol{u},\boldsymbol{z},\boldsymbol{x}) = \boldsymbol{0}$
Inequality Constraint:	$h(\boldsymbol{u},\boldsymbol{z},\boldsymbol{x}) \leq 0$

It's necessary to solve the PDE constraint and compute total derivatives. To do so, we need:

$$\frac{\partial f}{\partial u_I}, \frac{\partial f}{\partial z_I}, \frac{\partial f}{\partial x_I}, \frac{\partial g_I}{\partial u_J}, \frac{\partial g_I}{\partial z_J}, \frac{\partial g_I}{\partial x_J}, \text{ etc.}$$

Approach

Plato Analysis is built on **Kokkos** for portable performance.

What is Kokkos?

- Provides abstractions for:
  - Parallel execution (e.g., parallel_for)
  - Data management (Kokkos::View)
- Designed to target complex architectures:
  - Currently supports OpenMP, Pthreads, CUDA
  - One implementation, instead of one per architecture.
- C++ library, not a language extension.

Kokkos is available at github.com/kokkos/kokkos

Plato Analyze uses Nvidia's algebraic multi-grid solver package, **AmgX**.

AmgX is available at github.com/NVIDIA/amgx

Approach

Plato Analyze is built on Sacado for automatic differentiation. AD is straightforward to use.

Templatize

```
template <typename ScalarT>
ScalarT func(const ScalarT& a, const ScalarT& b, const ScalarT& c) {
  return c*std::log(b+1.)/std::sin(a);
}
```

Create AD types, and call template function

```
int main(){
  typedef Sacado::Fad::Sfad<double,2> SFadType;
  SFadType a(2, 0, 1.0), b(2, 1, 2.0), c(3.0);
  auto r = func(a,b,c);
  cout << "Value: " << r.val() << endl;
  cout << "dr/da: " << r.dx(0) << endl;
  cout << "dr/db: " << r.dx(1) << endl;
}
```

Sacado is available at github.com/trilinos/trilinos

Progress

Templated abstractions: ScalarFunction and VectorFunction

VectorFunction: $\boldsymbol{g(u,z,x)=0}$

```
virtual void
evaluate( Alexa::ScalarMultiVectorT<typename EvaluationType::StateScalarType  > state,
          Alexa::ScalarMultiVectorT<typename EvaluationType::ControlScalarType> control,
          Alexa::ScalarArray3DT    <typename EvaluationType::ConfigScalarType > config,
          Alexa::ScalarMultiVectorT<typename EvaluationType::ResultScalarType > result);
```

Elastostatics: $g_{el}=\nabla(x)\cdot f(z)\sigma(\varepsilon(u,x))$

```
Kokkos::parallel_for(Kokkos::RangePolicy<int>(0,numCells), LAMBDA_EXPRESSION(int cellOrdinal)
    {
      ConfigScalarType cellVolume;
      computeGradient(cellOrdinal, gradient, config, cellVolume);
      cellVolume *= quadratureWeight;

      voigtStrain(cellOrdinal, strain, state, gradient);

      voigtStress(cellOrdinal, stress, strain);

      applyWeighting(cellOrdinal, stress, control);

      stressDivergence(cellOrdinal, result, stress, gradient, cellVolume);

    },"Elastostatics Residual");
```

Progress

Physics: Elastostatics, Poisson (thermal, electrostatic)
Objectives: Compliance, p-norm
Constraints: Volume/mass

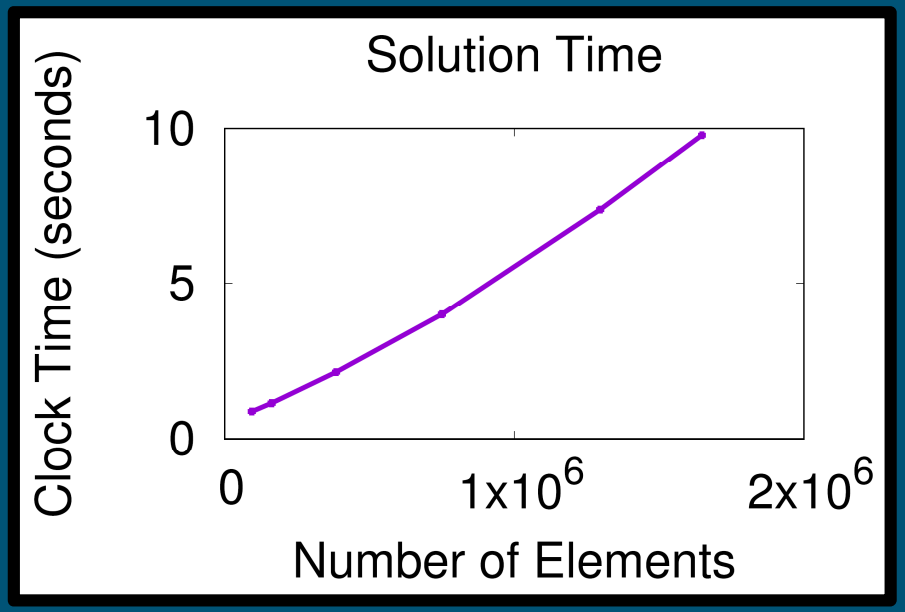
More physics, objectives, constraints coming soon.

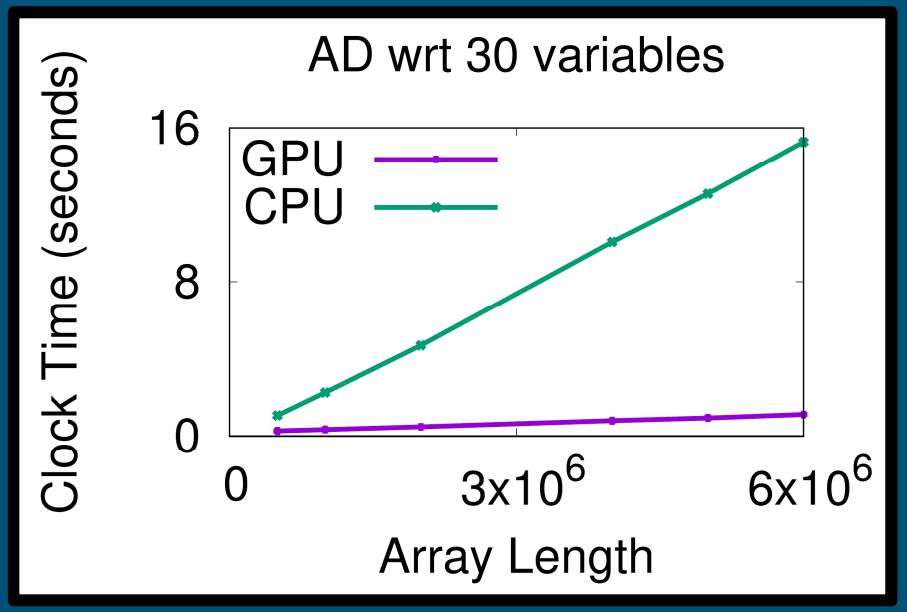
All of the above with gradients with respect to design, solution, and reference configuration.

Integration with Plato Engine is complete.

Progress

Initial performance data are encouraging





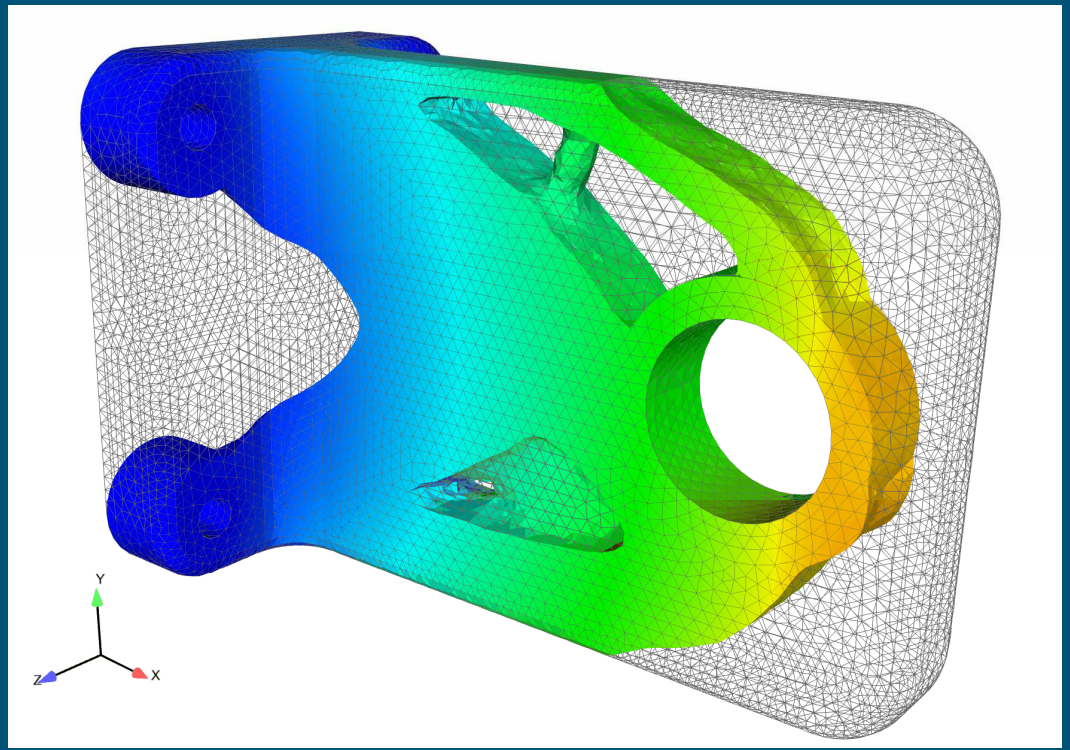
- Comparison of AD and direct stiffness show negligible effect on performance
- Roughly 15x speedup on GPU vs CPU

Results

Mechanical compliance minimization

Fixed at 4 bolt holes.
Vertical load on bottom half of opening.

elements: 187k
dofs: 110k
iterations: 26
Run time: 47s

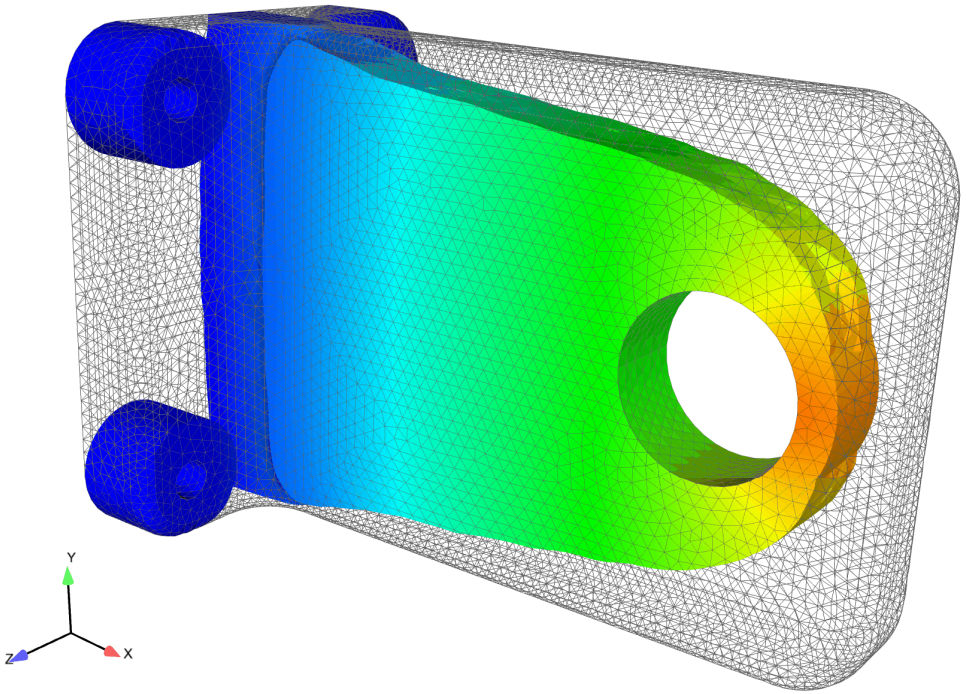


Results

Thermal compliance minimization

Fixed at 4 bolt holes and back plane.
Flux applied to opening.

elements: 187k
dofs: 110k
iterations: 26
Run time: 12s

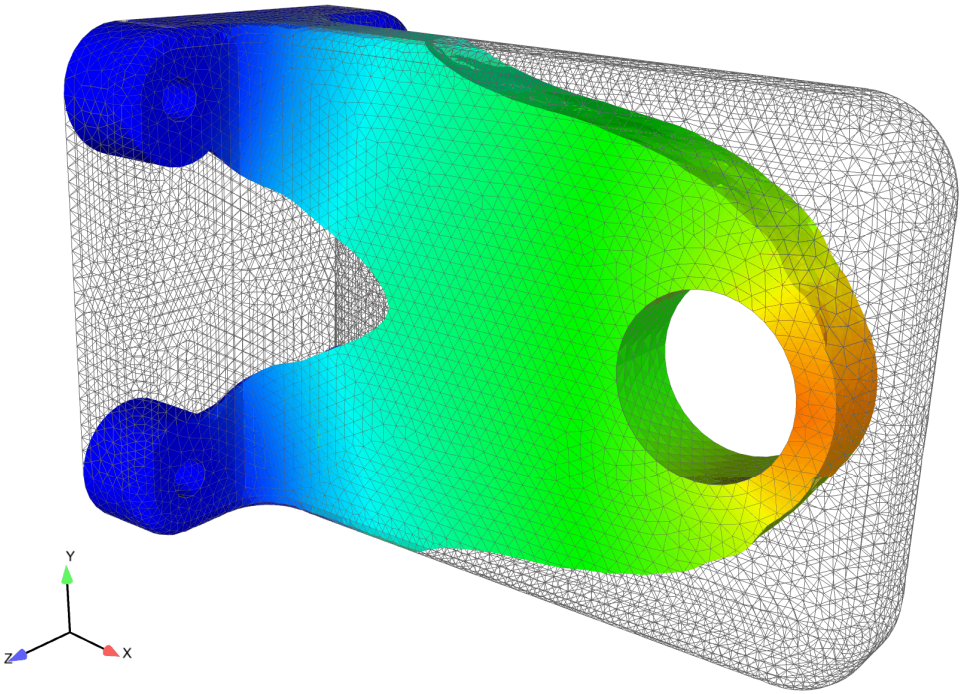


Results

Thermal and mechanical compliance minimization

Mechanical and thermal load cases. 2 GPUs.

elements: 187k
dofs: 110k
iterations: 26
Run time: 28s

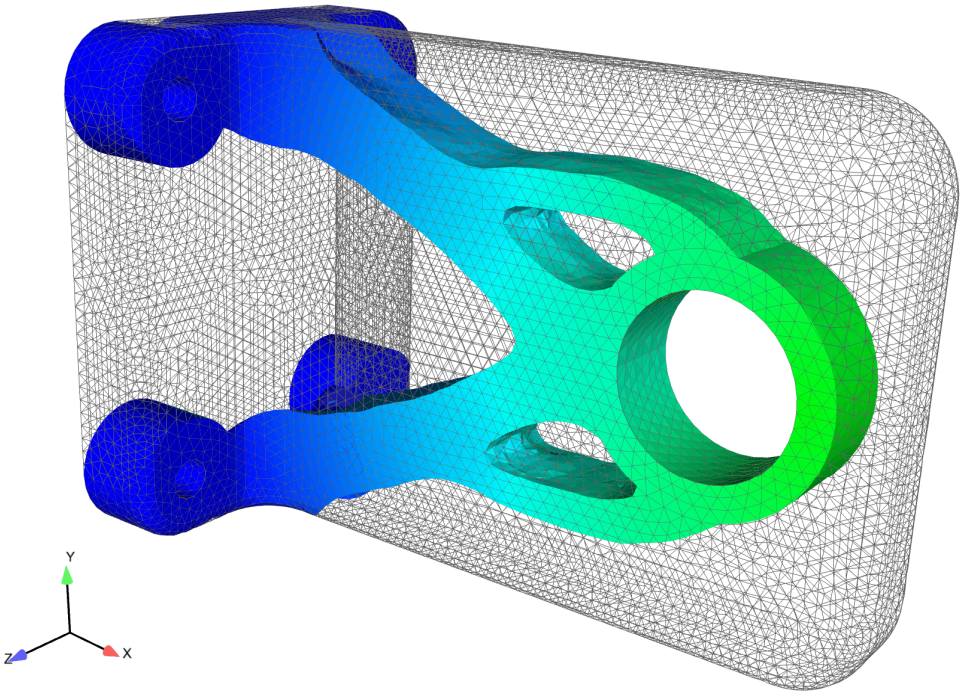


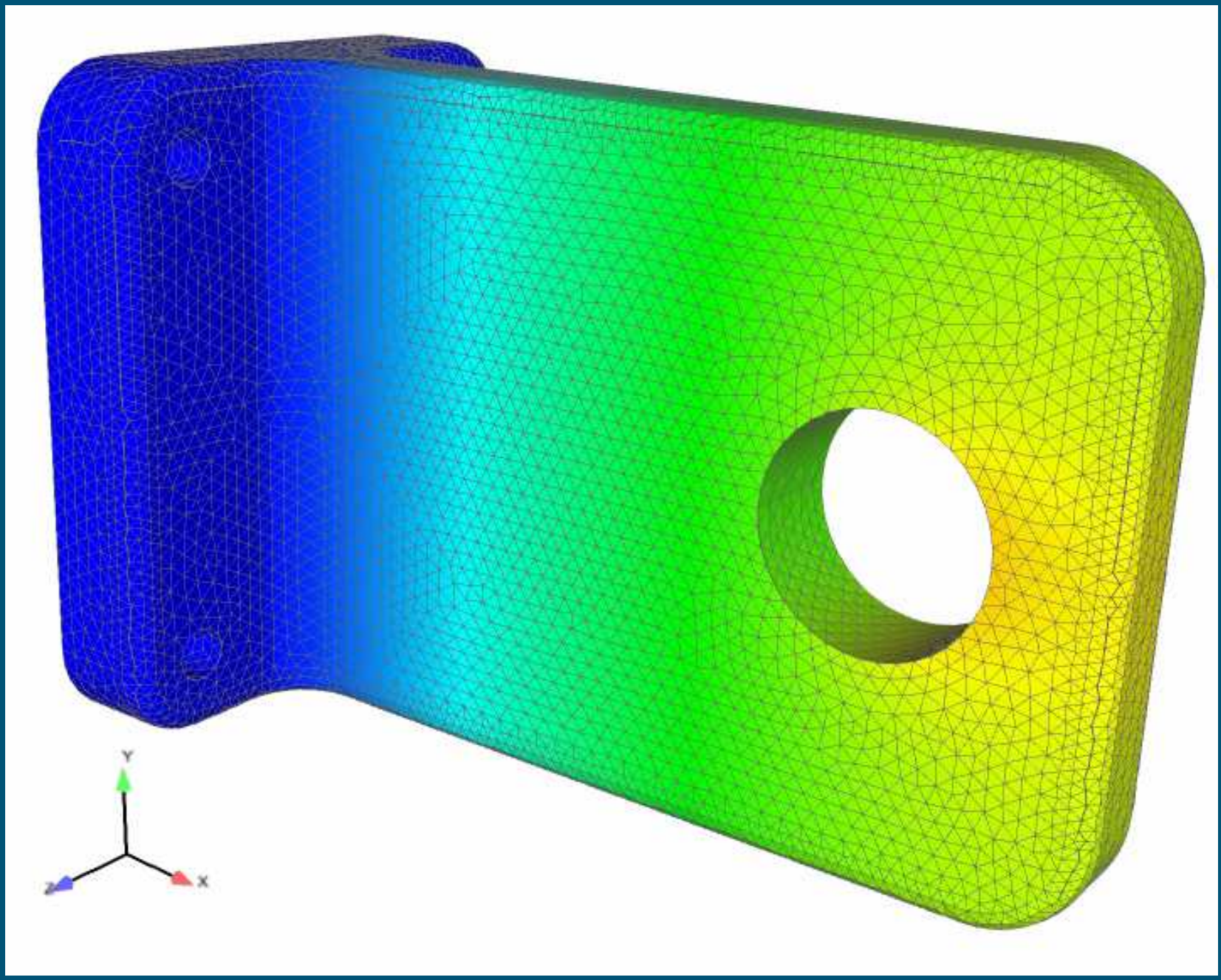
Results

Thermal and mechanical compliance minimization

Mechanical and thermal load cases. 2 GPUs.

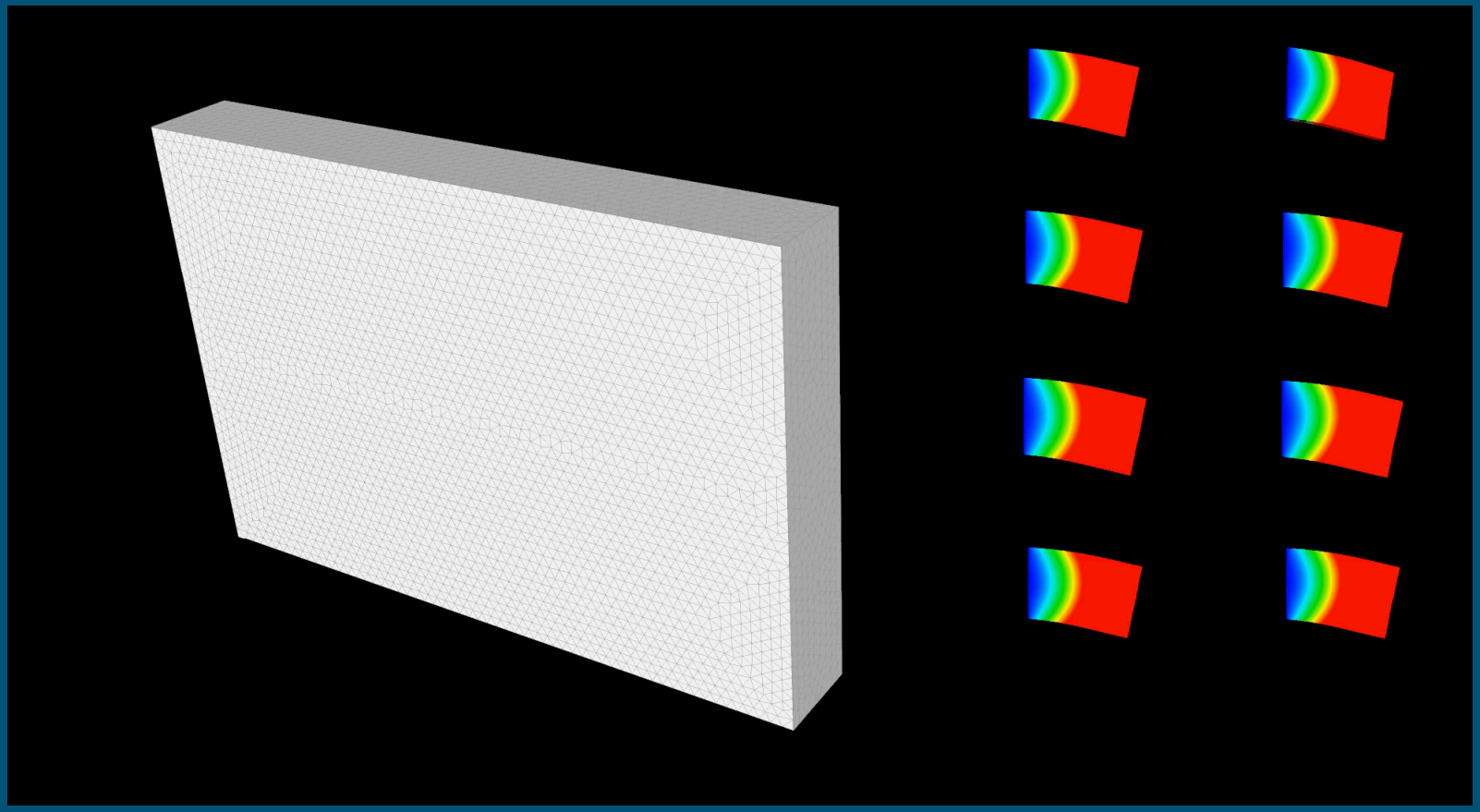
elements: 187k
dofs: 110k
iterations: 26
Run time: 28s




Y
X
Z

Results

Mechanical compliance minimization w/ uncertain load



8 Mechanical load cases.
8 CPUs, 8 GPUs.
elements: 175k

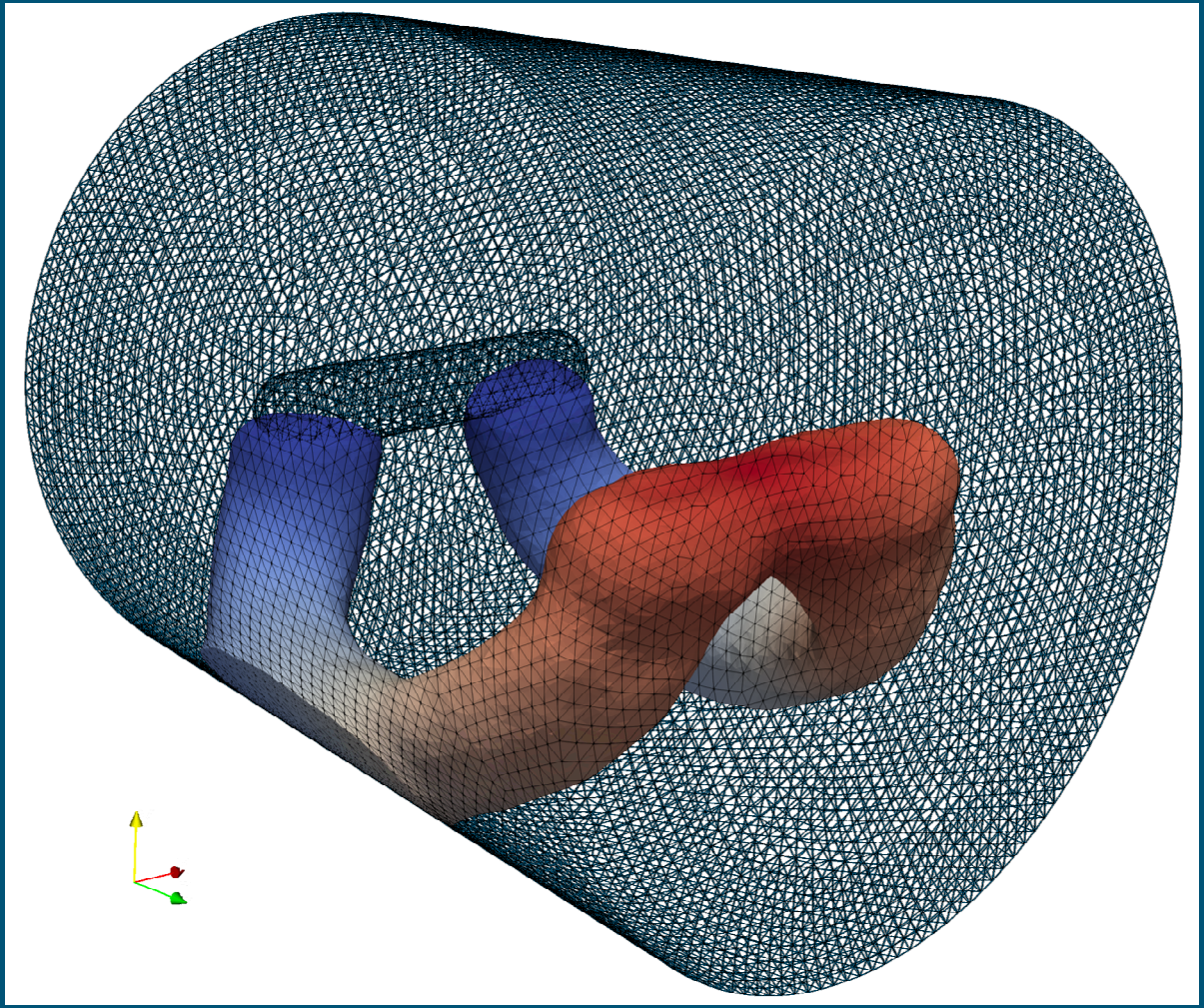
dofs: 100k
iterations: 26
Run time: 75s

Results

Objective: 6-norm of current density

Zero potential on back post, charge density on near post.

elements: 750k
dofs: 400k
linear solves: 122
Run time: 130s

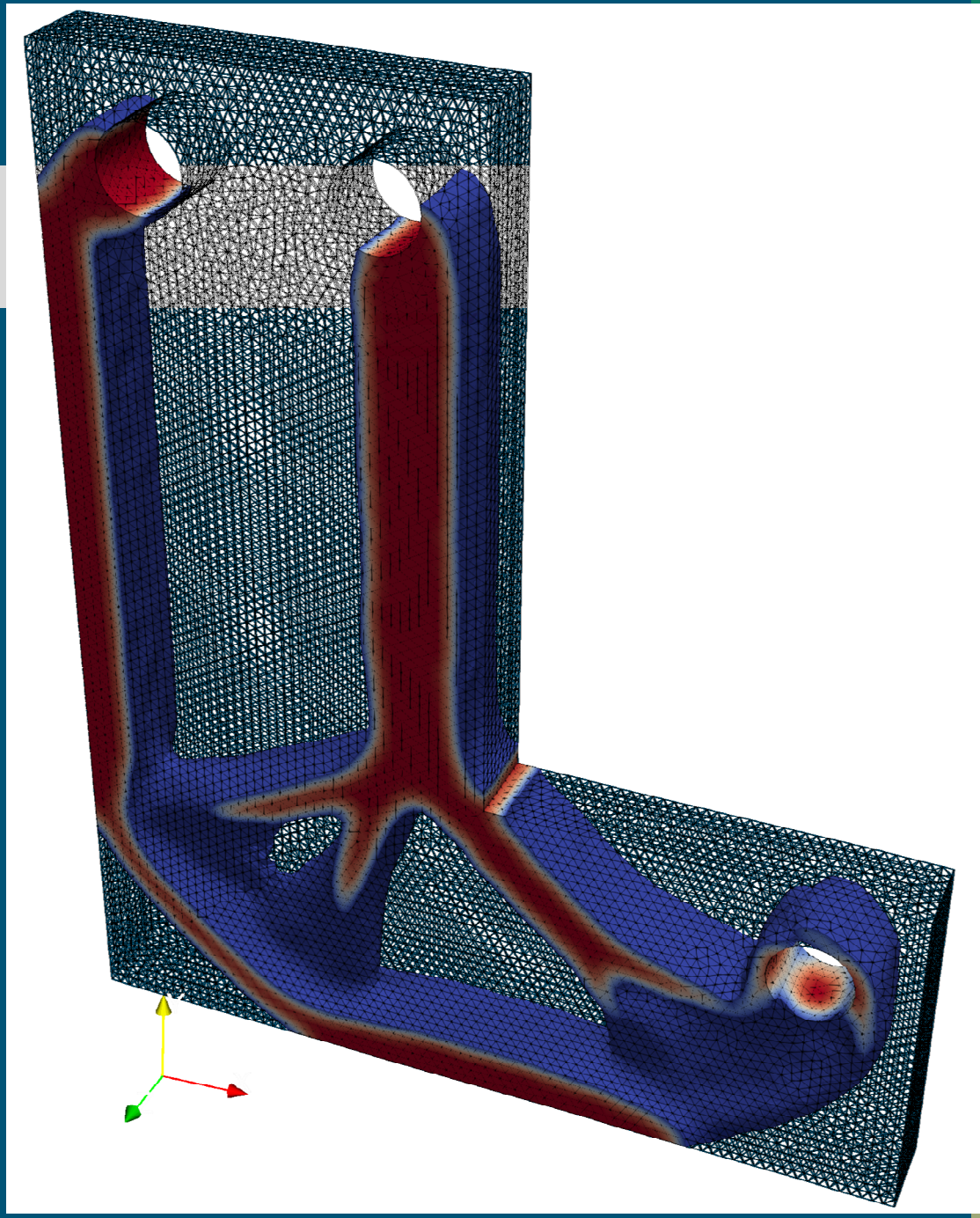


Results

Objective: 6-norm of effective stress

Fixed at top holes, vertical load at bottom hole.

elements: 368k
dofs: 200k
linear solves: 232
Run time: TBD



Summary

- Plato Analyze uses AmgX (Nvidia) for fast linear solves on the GPU.
- Plato Analyze uses AD (Sacado) with Kokkos to compute essential gradients.
- Objectives and constraints are templatized on AD types so a single implementation is used for computing values and all gradients.
- Plato Analyze implements a Plato::Application for Multiple Program, Multiple Data (MPMD) parallel calculations. This permits simultaneous evaluation of multiple load scenarios and/or samples for uncertainty quantification and propagation.
- Initial comparisons between global stiffness computed directly and with AD show negligible performance cost.