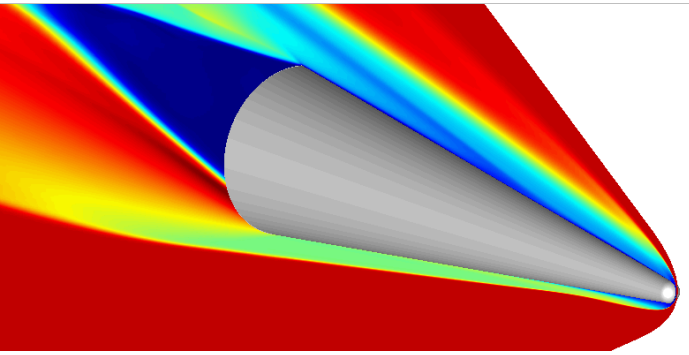
SAND2018-2074C



Performance Portability in SPARC

Using Kokkos in a Compressible CFD Code

Micah Howard, Andrew Bradley, Steven Bova, James Overfelt,
Ross Wagnild, Derek Dinzl, Mark Hoeman

U.S. DEPARTMENT OF ENERGY
NNSA National Nuclear Security Administration

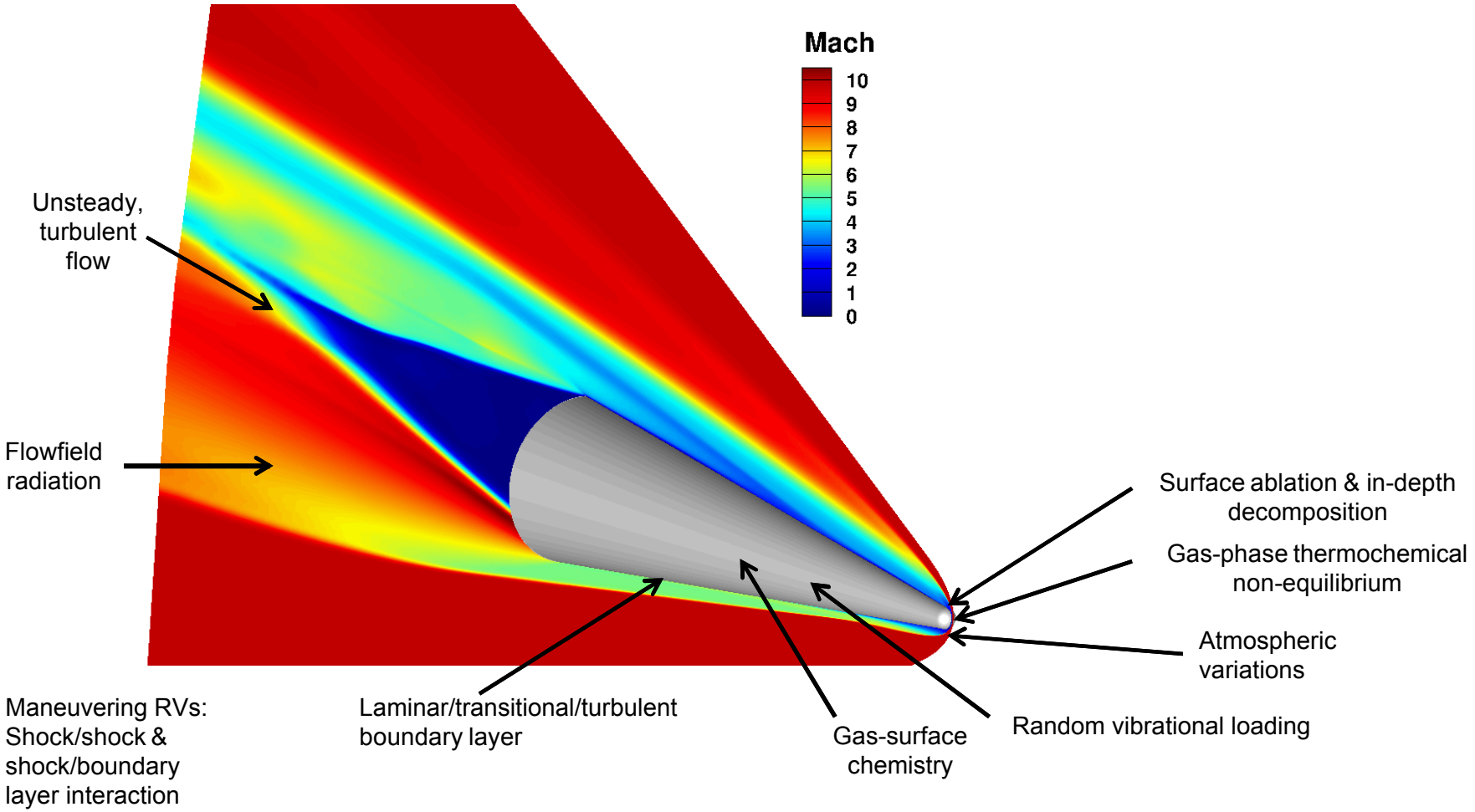
Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

OutLine

Sandia National Laboratories

- **Physics**
- **Discretization**
- **Programming Design Patterns**
- **Performance Analysis**

Hypersonic Reentry Simulation



Time and Spatial Discretization

Sandia National Laboratories

- **Time**
  - **4th order explicit Runge-Kutta**
  - **Implicit backward Euler**
    - **Newton**
    - **Block tri-diagonal fixed point solver**
- **Space**
  - **Block Structured Finite Volume**
  - **Unstructured Finite Volume**
  - **Unstructured Finite Element**
  - **……**

From CPU to GPU

Sandia National Laboratories

Application was chosen to demonstrate the ability to run on next generation hardware using performance portable code.

Easily an exascale problem.

Started with a working parallel cpu based code with very few dependencies.

It was realized that extensive rewriting was needed and this was considered acceptable.

Some Design Patterns

Sandia National Laboratories

- **Dynamic Dispatch**
- **Run time-to-Compile time**
- **Mesh Traversal**

Design Pattern: Dynamic Dispatch

Sandia National Laboratories

- **Compile time verses Run time polymorphism.**
  - **Compile time => Generate static combinations**
- **Basic idea:**

```
A* o = new C();
template <bool isd> struct Dispatcher;
template <> struct Dispatcher<true> {
   template <typename Type> static void foo (Type *o){
      o->foo();
}}
template <> struct Dispatcher<false> {
   template <typename Type> static void foo (Type *o){
      static_cast<Type*>(o)->Type::foo();
}}
Dispatcher<true> ::foo<A>(o);
Dispatcher<false>::foo<C>(o);
```

Design Pattern: Rt2Ct

- **Creating class instances with lots of template arguments**

```
template <class Ptr, class ...Parms, class ...Args>
Ptr create (const Boolean &b, Args... Args){
  if (b) return create<Ptr, Parms..., True > (args...);
  else   return create<Ptr, Parms..., False> (args...);
}
```

- **Lots of create functions that chain together**
- **Eventually a create that actually does something**
- **Can have special Tag() struct to terminate chain and call the correct constructor.**

Design Pattern: Mesh Traversal

Sandia National Laboratories

- **Structured**
  - **All structured classes derive from a flavor of MeshTraverser class and implement the operator:**

```
void operator() (int i, int j, int k) const;
```

  - **Allows the traversed class to be unaware of the traversal patter**
    - **Simple Red/Black pattern for coloring algorithm**
    - **Flatten array for indexing over everything**
    - **Build In buffers at the edges for simple indexing over large 3D arrays.**
- **Unstructured**
  - **All unstructured classes derive from a flavor of MeshTraverser and implement the operator:**

```
void operator() (int i) const;
```

  - **Allows for sophisticated coloring algorithms**

Performance Portability

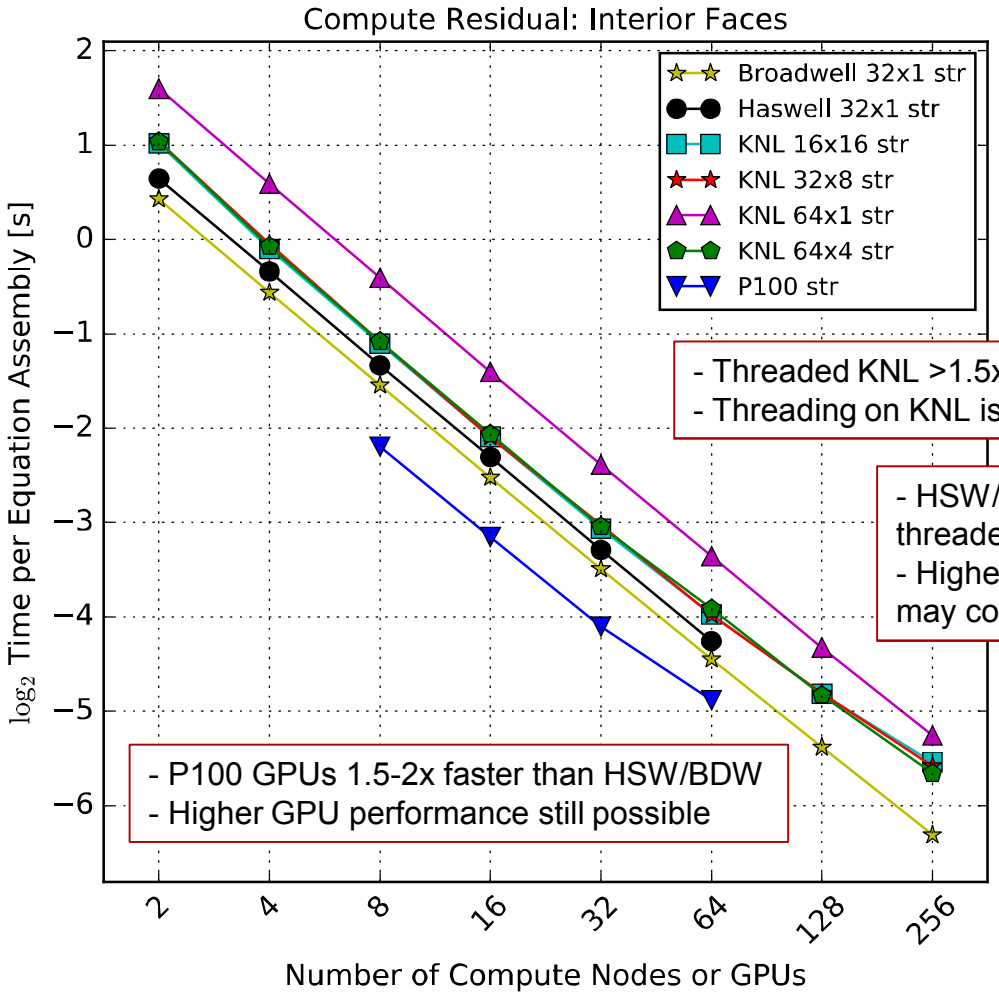
Sandia National Laboratories

- **Scaling results for:**
  - **Intel Xeon Broadwell (BDW) cluster, 32x1**
  - **Intel Xeon Haswell (HSW) cluster, 32x1**
  - **Intel Xeon Phi Knights Landing (KNL) testbed, 16x16, 32x8, 64x1, 64x4**
  - **IBM Power CPUs and four Nvidia Pascal 100 (P100) GPUs per host CPU**

SPARC: Strong Scaling Analysis

For the heaviest kernel during equation assembly…

First…
lower = faster
&
this is a log2 scale



- Threaded KNL >1.5x faster than MPI-only KNL
- Threading on KNL is important

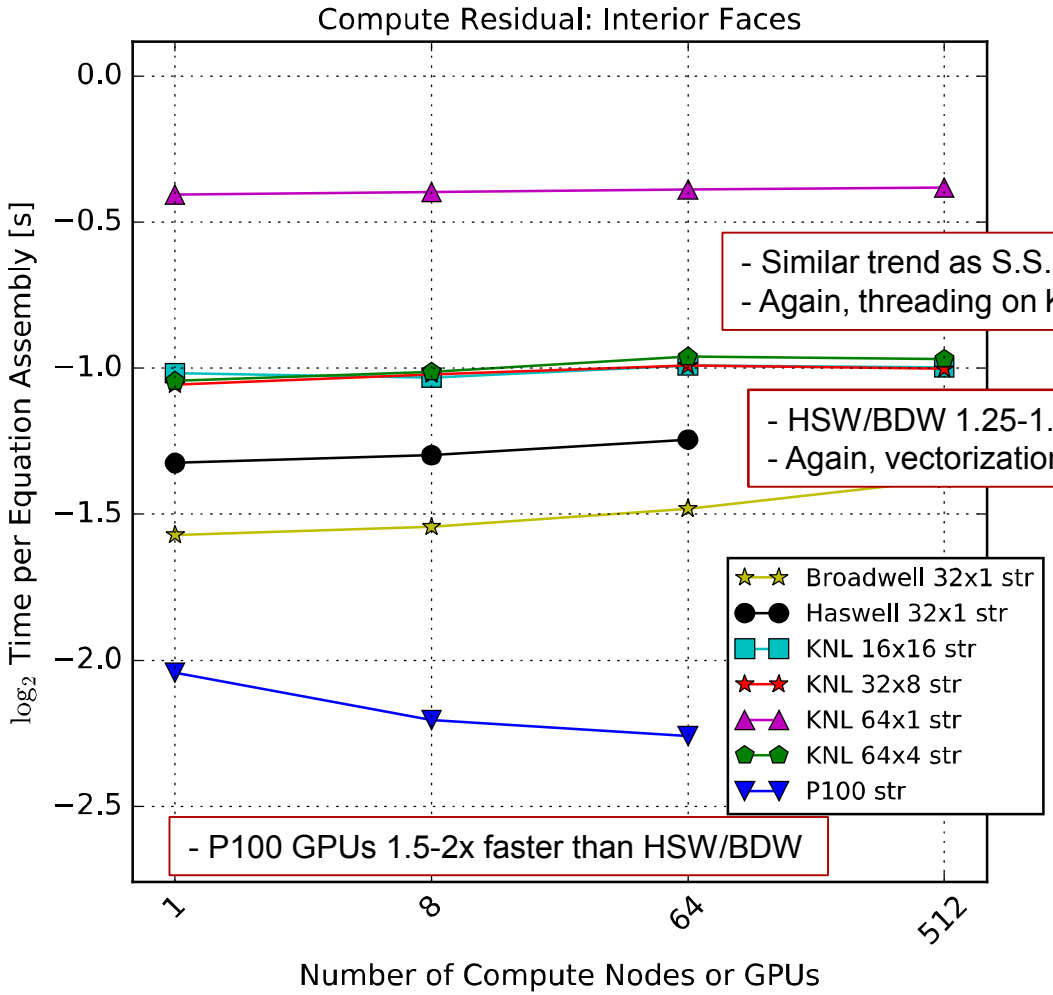
- HSW/BDW 1.25-1.5x faster than threaded KNL
- Higher KNL assembly performance may come from SIMD vectorization

- P100 GPUs 1.5-2x faster than HSW/BDW
- Higher GPU performance still possible

SPARC: Weak Scaling Analysis

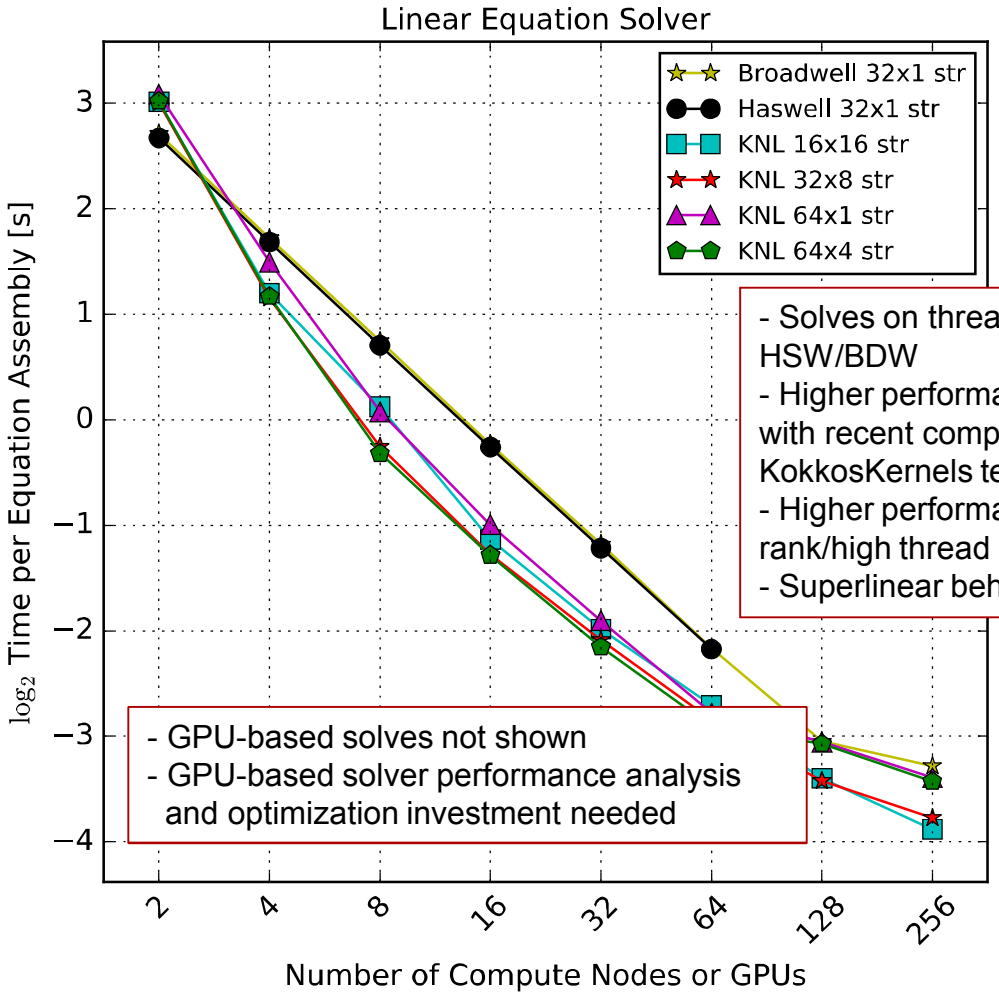
For the heaviest kernel during equation assembly…

Recall…
lower = faster
&
this is a log2 scale



SPARC: Strong Scaling Analysis

For the linear equation solve…

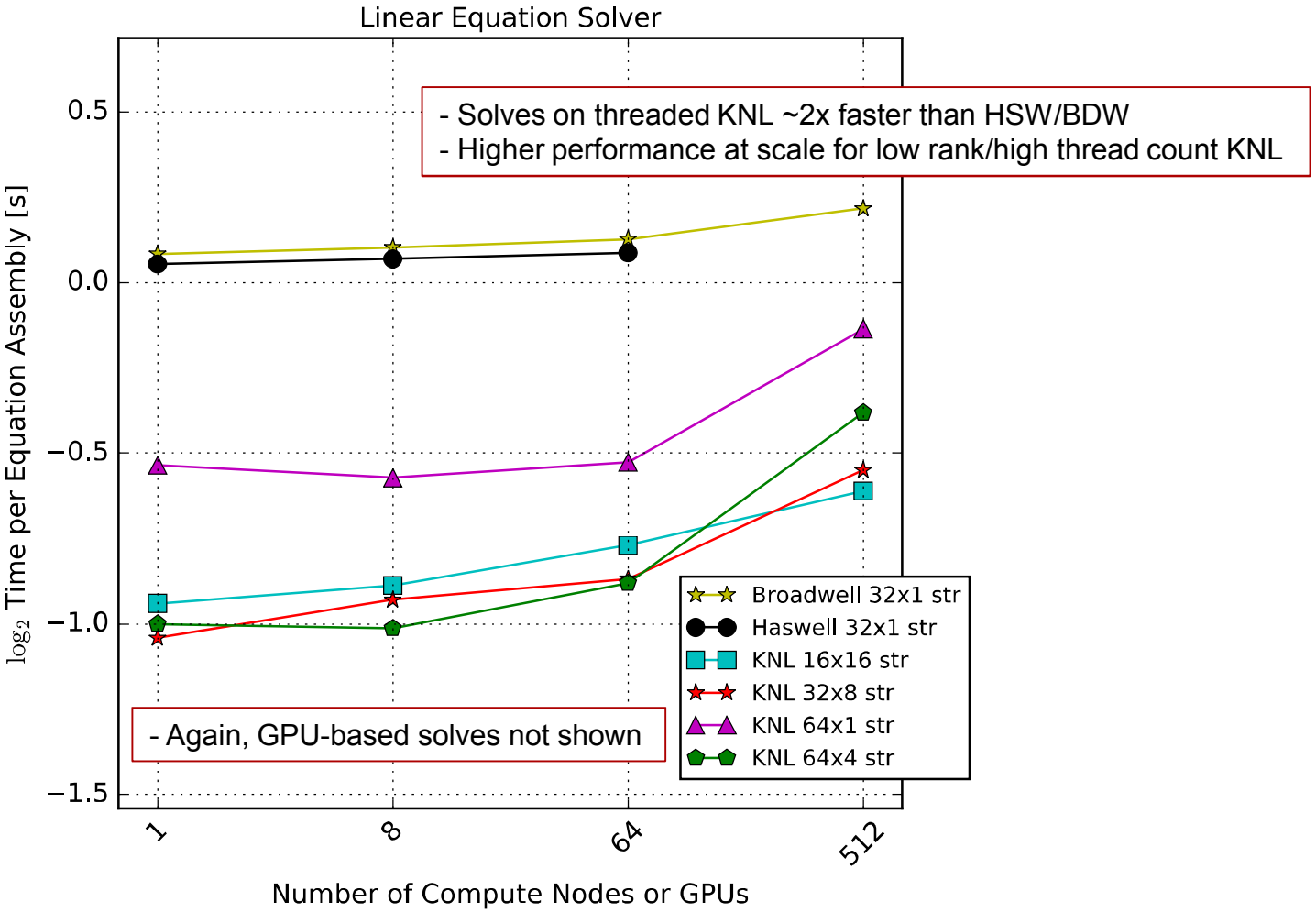


- Solves on threaded KNL ~2x faster than HSW/BDW
- Higher performance on KNL still possible with recent compact BLAS work by the KokkosKernels team
- Higher performance at scale for low rank/high thread count KNL
- Superlinear behavior a DDR/HBM effect

- GPU-based solves not shown
- GPU-based solver performance analysis and optimization investment needed

SPARC: Weak Scaling Analysis

For the linear equation solve…



Remaining Challenges

Sandia National Laboratories

- **Pushing performance portability through the entire solver software stack, Trilinos.**
- **Continuing major development and even rewrites while trying to maintain functionality across many platforms.**
  - **Very slow compilation times on our GPU platforms discourages programmers from continuously compiling before pushing code.**
  - **GPU tests are easily broken and hard to fix.**