Adapted from “Improving Scientific Software Component Quality Through Assertions,” SE-HPCS 05.

Software correctness is a challenge in scientific computing.

- Sun
- Model
- Algorithm
- Software

Errors can occur at any of these points!

Reliance on components will exacerbate correctness concerns.

- Risks include...
  - Misuse
    - Third-party
    - Binary
  - Complexity
    - Different implementation languages
  - Untested features
    - Unanticipated input data
    - Poorly tested paths

Executable assertions will become increasingly important.
But performance overhead can be unacceptable during deployment.

Research relies on automatically generated enforcement.

Gained insights through initial experiments using mesh component.

The five enforcement policies included simple sampling strategies.

---

The five enforcement policies included simple sampling strategies.

- Never
- Always
- Periodic
- Random
- Adaptive Timing

---

Research relies on automatically generated enforcement.

Gained insights through initial experiments using mesh component.

The five enforcement policies included simple sampling strategies.
Findings confirmed expectations and led to new ideas.

Adaptive Timing was clearly more effective in these experiments.

So now pursuing policies aimed at automatically tuning enforcement.

In order to be effective, need heuristics to guide enforcement.

- “Typical” interface assertions?
- Which types are important/critical?
  - Corrupt data
  - Invalidate results
- Which types lead to component assembly failures?

Suggestions, questions, or components? Please contact me at dahlgren1@llnl.gov or (925) 423-2685.
Adaptive Enforcement of Interface Assertions

The End