Slide 1

Adaptive Enforcement of Interface Assertions

CCA Forum

Portland, Oregon

April 28, 2005

Tamara L. Dahlgren

dahlgren1@llnl.gov

SciDAC
Scientific Discovery through Advanced Computing

UCRL-PRES-xxxxxx

Slide 2

Research on *adaptive* interface assertion enforcement.

- **Target**
  - ▶ Scientific applications built of components
    - Well-defined interfaces
- **Focus**
  - ▶ *Demonstrating* correctness *during deployment*
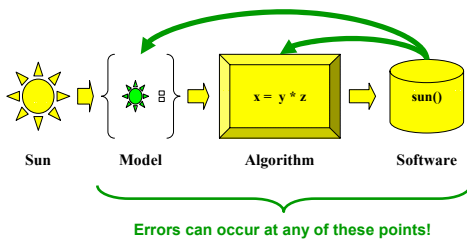   - Plug-and-play components
- **Goal**
  - ▶ Maximize failure detection within performance overhead constraints

Adapted from "Improving Scientific Software Component Quality Through Assertions," SE-HPCS '05.

Slide 3

Software correctness is a challenge in scientific computing.

Sun   Model   Algorithm   Software

$x = y * z$

sun()

Errors can occur at any of these points!

[Dubois03]

Slide 4

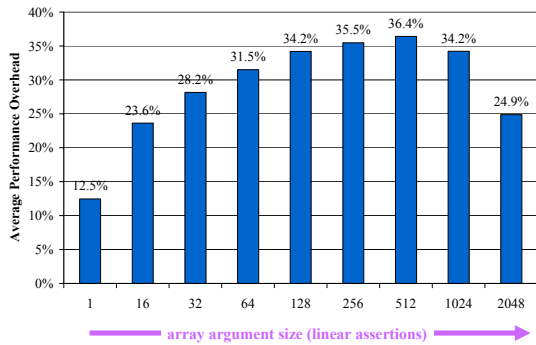**Reliance on components will exacerbate correctness concerns.**

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

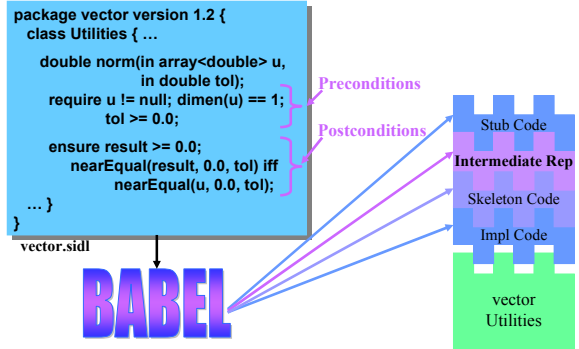**Executable assertions will become increasingly important.**

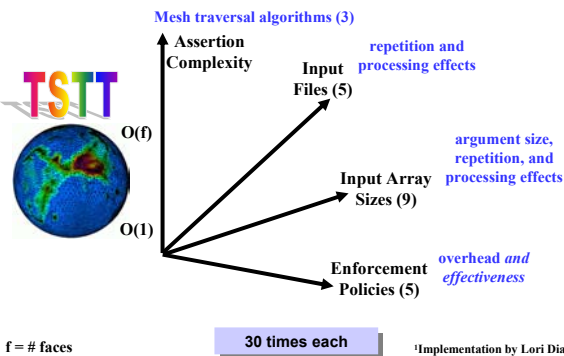## But performance overhead can be unacceptable during deployment.

Average Performance Overhead

- 12.5%
- 23.6%
- 28.2%
- 31.5%
- 34.2%
- 35.5%
- 36.4%
- 34.2%
- 24.9%

array argument size (linear assertions) →

1 16 32 64 128 256 512 1024 2048

5

Research relies on automatically generated enforcement.

```
package vector version 1.2 {
  class Utilities { …
    double norm(in array<double> u,
                in double tol);
      require u != null; dimen(u) == 1;
             tol >= 0.0;
      ensure result >= 0.0;
          nearEqual(result, 0.0, tol) iff
               nearEqual(u, 0.0, tol);
  … }
}
```
vector.sidl

Preconditions

Postconditions

BABEL

Stub Code
Intermediate Rep
Skeleton Code
Impl Code

vector Utilities

6

Gained insights through initial experiments using mesh component[1].

Mesh traversal algorithms (3)

Assertion Complexity

Input Files (5)

repetition and processing effects

O(f)

argument size, repetition, and processing effects

Input Array Sizes (9)

O(1)

Enforcement Policies (5)

overhead *and* *effectiveness*

f = # faces

30 times each

[1]Implementation by Lori Diachin   7

The five enforcement policies included simple sampling strategies.

Never

Always

Periodic

Random
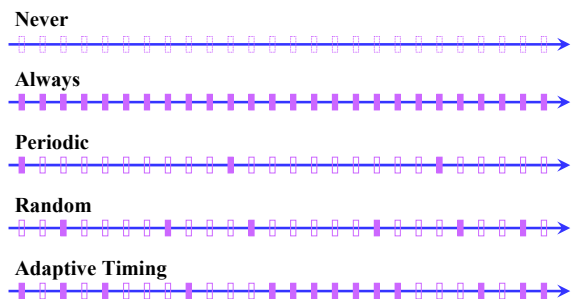
Adaptive Timing

8

Findings confirmed expectations and led to new ideas.

Invoked methods whose assertions were checked

Average performance overhead

9
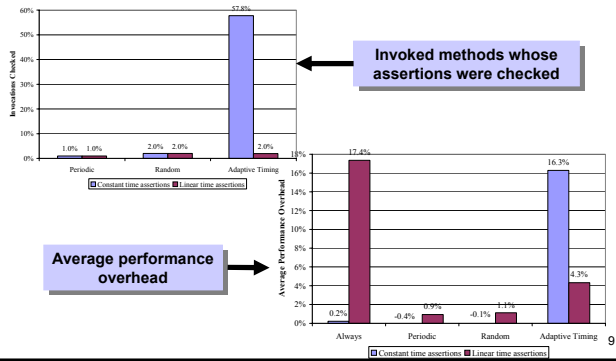
Adaptive Timing was clearly more effective in these experiments.

Subject to sampling bias

	Periodic	Random	Adaptive Timing
	20.0%	2.0%	61.3%

150 failures   **30**   **3**   **92**

10
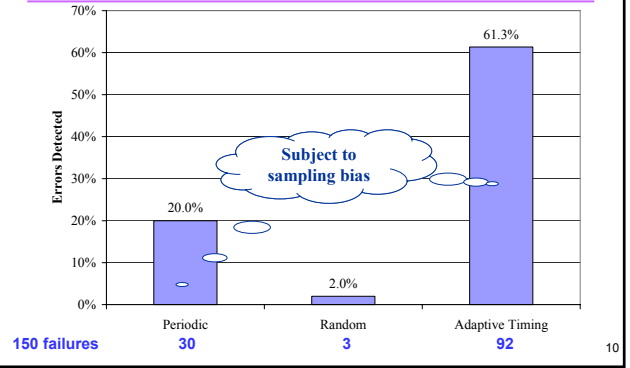
So now pursuing policies aimed at automatically tuning enforcement.

**GOAL:
Maximize failure detection capability within overhead limit!**

11
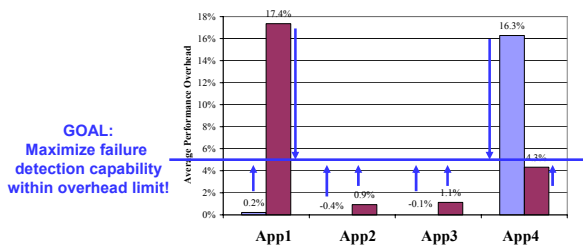
In order to be effective, need heuristics to guide enforcement.

- **"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.**

12

Adaptive Enforcement of
Interface Assertions

The End