Reducing Component Contract Overhead by Offloading Enforcement

Kosta Damevski[1] Hui Chen[1] Tamara Dahlgren[2]

[1]Department of Mathematics and Computer Science
Virginia State University

[2]Lawrence Livermore National Laboratory

November 16, 2009

Outline

Benefits of Contracts

Executable component contracts have several advantages:

- Improve application quality, by promoting safer reuse
- Provide a better description of a component's functionality than plain IDL

To be fully effective, contracts should be enforced even after the application is deployed (in production scenarios)

- Contract excercised through a variety of real parameters

Contracts in Common Component Architecture (CCA)

Example

```
bool vuIsZero(in array<double> u, in double tol)
  require
    not_null : u != null;
    u_is_1d : dimen(u) == 1;
    non_negative_tolerance: tol >= 0.0;
  ensure
    no_side_effects : is pure;
```

- SIDL extensions for expressing contracts as method preconditions, postconditions and class invariants
- Support for both simple and complex contracts
  - Logical operations (and, or, implies, if and only if, etc.)
  - Contracts may contain function calls
    - Nested functions' contracts (if they exist) are turned off
  - Predefined functions also exist (e.g. max, sum, min, etc.)

- Contracts have to be enforced at runtime, which introduces overhead

- Contracts have to be enforced at runtime, which introduces overhead
- Overhead can be significant if contracts checks are frequent or if contracts are complex

- Contracts have to be enforced at runtime, which introduces overhead
- Overhead can be significant if contracts checks are frequent or if contracts are complex
- May lead to lack of contract adoption in HPC

- Contracts have to be enforced at runtime, which introduces overhead
- Overhead can be significant if contracts checks are frequent or if contracts are complex
- May lead to lack of contract adoption in HPC

Question

Is is worthwhile to postpone the enforcemenet of contracts to when the system is less busy in order to reduce their performance impact?

- Contracts have to be enforced at runtime, which introduces overhead
- Overhead can be significant if contracts checks are frequent or if contracts are complex
- May lead to lack of contract adoption in HPC

Question

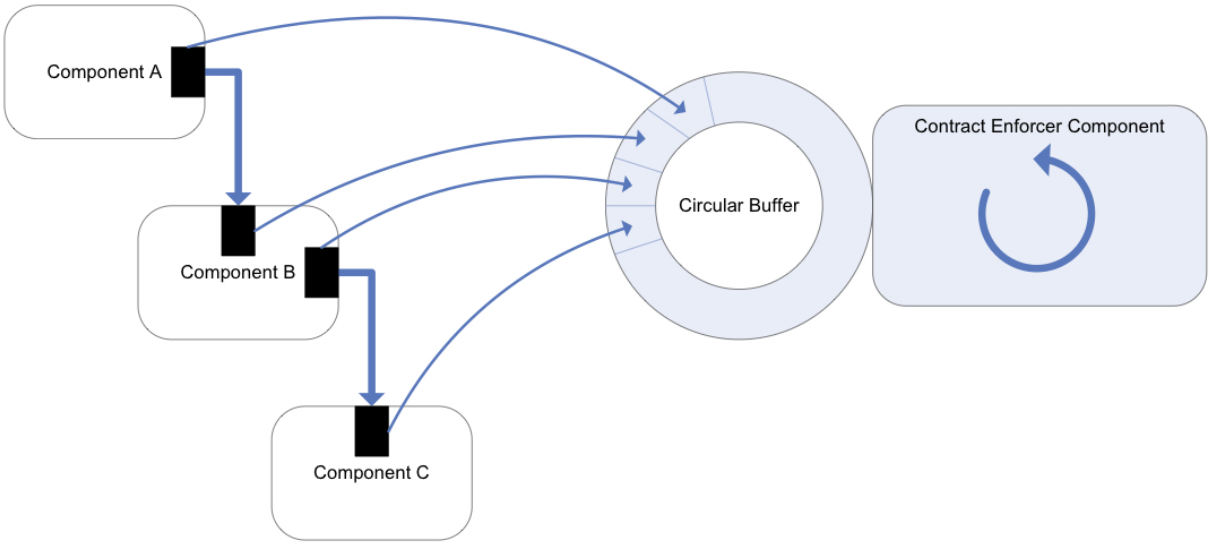
Is is worthwhile to postpone the enforcemenet of contracts to when the system is less busy in order to reduce their performance impact?

Answer

In some cases, yes.

- More detail follows. . .

Design of Contract Offloading System



- Save a copy of the data needed to enforce the contract and some small meta data
  - A circular buffer caps the memory expenditure
- A low priority thread in a ContractEnforcer component enforces the contracts
  - The ContractEnforcer may exist on a separate computational resource

Offloading's Impact on the Expected Semantics of Contracts

- If the contract is eventually enforced...
  - ...and if there was a violation then the application may have produced incorrect results or spent a little bit extra time computing

Offloading's Impact on the Expected Semantics of Contracts

- If the contract is eventually enforced...
  - ...and if there was a violation then the application may have produced incorrect results or spent a little bit extra time computing
    - User is informed of contract violation, even if application has finished

Offloading's Impact on the Expected Semantics of Contracts

- If the contract is eventually enforced...
  - ...and if there was a violation then the application may have produced incorrect results or spent a little bit extra time computing
    - User is informed of contract violation, even if application has finished
  - ...and if there was no violation then no harm no foul

Offloading's Impact on the Expected Semantics of Contracts

- If the contract is never enforced, due to lack of space in the circular buffer
  - This may violate user expectations

Offloading's Impact on the Expected Semantics of Contracts

- If the contract is never enforced, due to lack of space in the circular buffer
  - This may violate user expectations
  - Unlikely to occur in many executions or to the same contract

Offloading's Impact on the Expected Semantics of Contracts

- If the contract is never enforced, due to lack of space in the circular buffer
  - This may violate user expectations
  - Unlikely to occur in many executions or to the same contract
  - Perhaps a reasonable price to pay for runtime contracts in HPC

When to Offload?

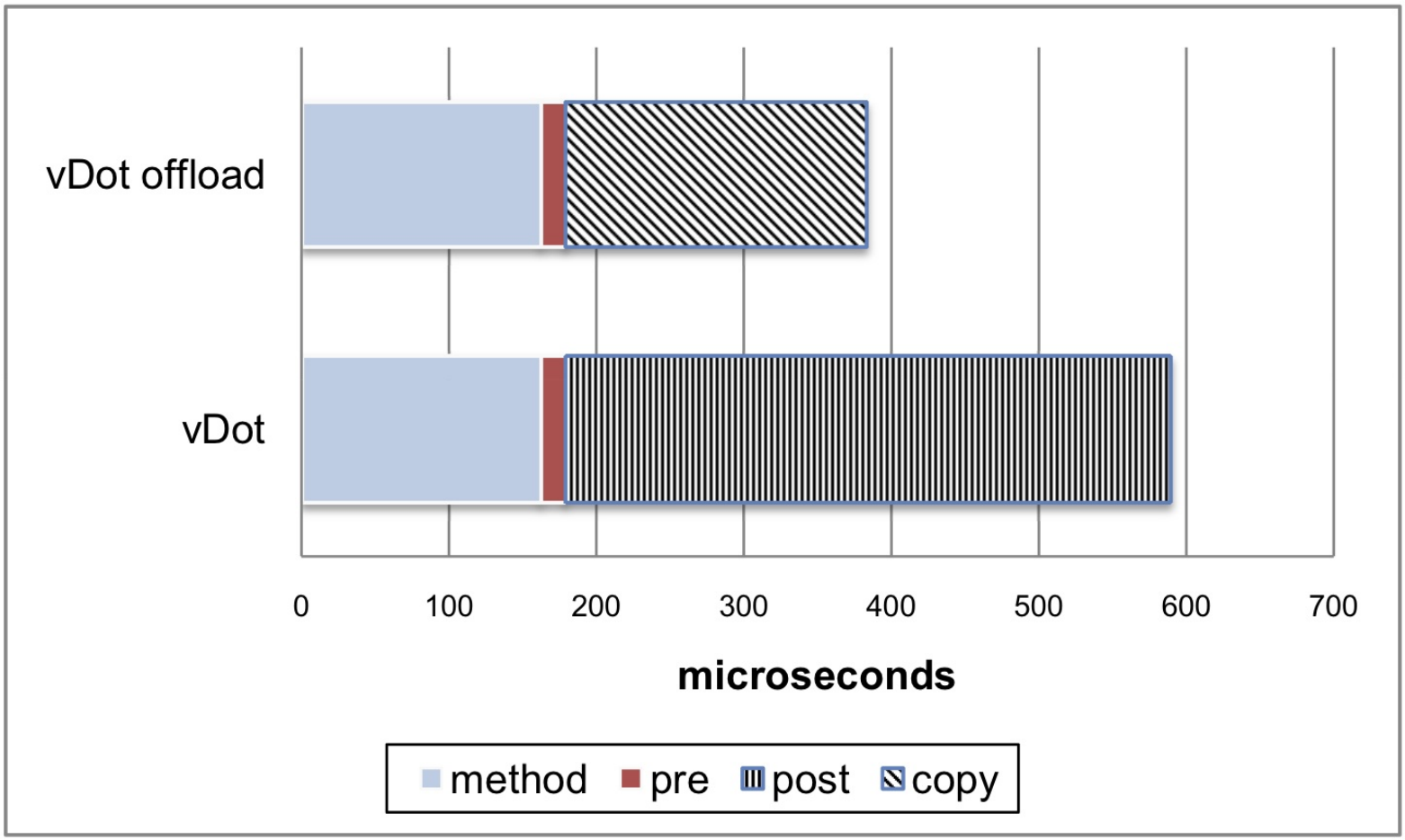
- The cost of offloading the contracts is a copy of the contract's data
- Some contracts are more expensive to copy than to enforce in place
  - Do not offload computationally cheap (per data unit) contracts
- Determine the contracts that have significant computational complexity per data unit
  - Prefer offloading complex contracts for later enforcement

Experiment

- Tested contract offloading using Babel on an interface performing a set of vector operations
- Measured contract offloading to a ContractEnforcer component in the same address space
- Confirmed suspicion that offloading is appropriate for certain complex contracts, where computational complexity is linear or more

The *vDot* Method

```
double vDot(in array<double> u, in array<double> v,
            in double tol)
  require
    not_null_u: u != null;
    u_is_1d : dimen(u) == 1;
    not_null_v: v != null;
    v_is_1d : dimen(v) == 1;
    same_size:  size(u) == size(v);
    non_negative_tolerance: tol >= 0.0;
  ensure
    same_uv: nearEqual(u,v,tol) implies
                (result >= 0.0);
    zero_case: (irange(u, 0.0, tol) and
                irange(v, 0.0, tol)) implies
                nearEqual(result, 0.0, tol);
```



500 elements per array, averaged over 100 runs

Related Work

- Ability to specify limits to overhead

  Dahlgren, T. L. "Performance-driven interface contract enforcement for scientific components". *In Proceedings of the 10th International Symposium on Component-Based Software Engineering (CBSE-07)*

- Selectively enforce assertions in software with a large user base

  Liblit, B., Aiken, A., Zheng, A. X., and Jordan, M. I. "Bug isolation via remote program sampling". *In Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation (PLDI '03)*

- Post processing log data to determine if an assertion was violated

  Sokolsky, O., Sammapun, U., Regehr, J., and Lee, I. "Runtime verification for wireless sensor network applications". *In Runtime Verification (2008), B. Finkbeiner, K. Havelund, G. Rosu, and O. Sokolsky, Eds., no. 07011 in Dagstuhl Seminar Proceedings*

Summary

- Contracts can present a significant performance overhead
- Offloading (postponing) the contract enforcement to improve performance is a reasonable approach
- Contracts whose computational complexity is significant are prime candidates
- Future Work: Try this approach on applications with significant contract overhead

Acknowledgements

- CCA Forum (http://www.cca-forum.org)
- Babel Team (http://computation.llnl.gov/casc/components/babel.html)

QUESTIONS?