TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

Program Execution Environments as Contextual Values

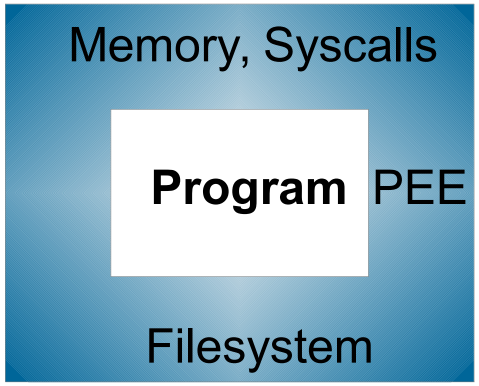
Markus Raab <markus.raab@complang.tuwien.ac.at>
Franz Puntigam <franz@complang.tuwien.ac.at>

COP'14, ECOOP Uppsala Sweden
July 29, 2014

Program Execution Environment (PEE)

- Interface of application to operating system
- Configuration management

- Consists of:
  - Configuration Files
  - Commandline Arguments
  - ...



Contextual Values (CV)

- CV are **variables**
- Values can change depending on context
- Side-effects are no longer potentially global

- **Usage in C++**
  - scoped by **with(out)** clause:

    ```
    cout << var; with<Layer>()([]{ cout << var;})
    ```
  - using **(de)activate**:

    ```
    cout << var; activate<Layer>(); cout << var++;
    ```

Outline

- Motivation
- Implementation
- **Evaluation** of Elektra
  - Benchmarks
  - Debugging
  - Persistence
- Conclusion

Elektra's Logo

Motivation

- Program Execution Environment (PEE)
  - PEE **error-prone** on multiple levels
    - Redundancy and wrong conversions within programs
    - Inconsistency in documentation and semantics (behavior)
    - Hardly any validation
  - Applications have **undefined behavior** on errors
  - Currently no standard way to change PEE by programs
  - No support for context

Motivating Example

- CV very useful for PEE
  - sessions, modes, host, internationalization, **profiles**, ..

```
int main (int argc, char**argv) {
    KeySet ks;
    parseConfigfiles(ks);
    parseCommandline(ks, argc, argv);
    Context c;
    Environment env(ks, c);
    c.activate<ProfileLayer>(env.profile);
    // the rest of the program works with profile
}
```

Connect PEE and CV

- PEE is defined using a specification

```
[/%language%/%country%/%dialect%/person/greeting]
  type=String
[/%country%/person/visits]
  type=Integer
  default=0
```

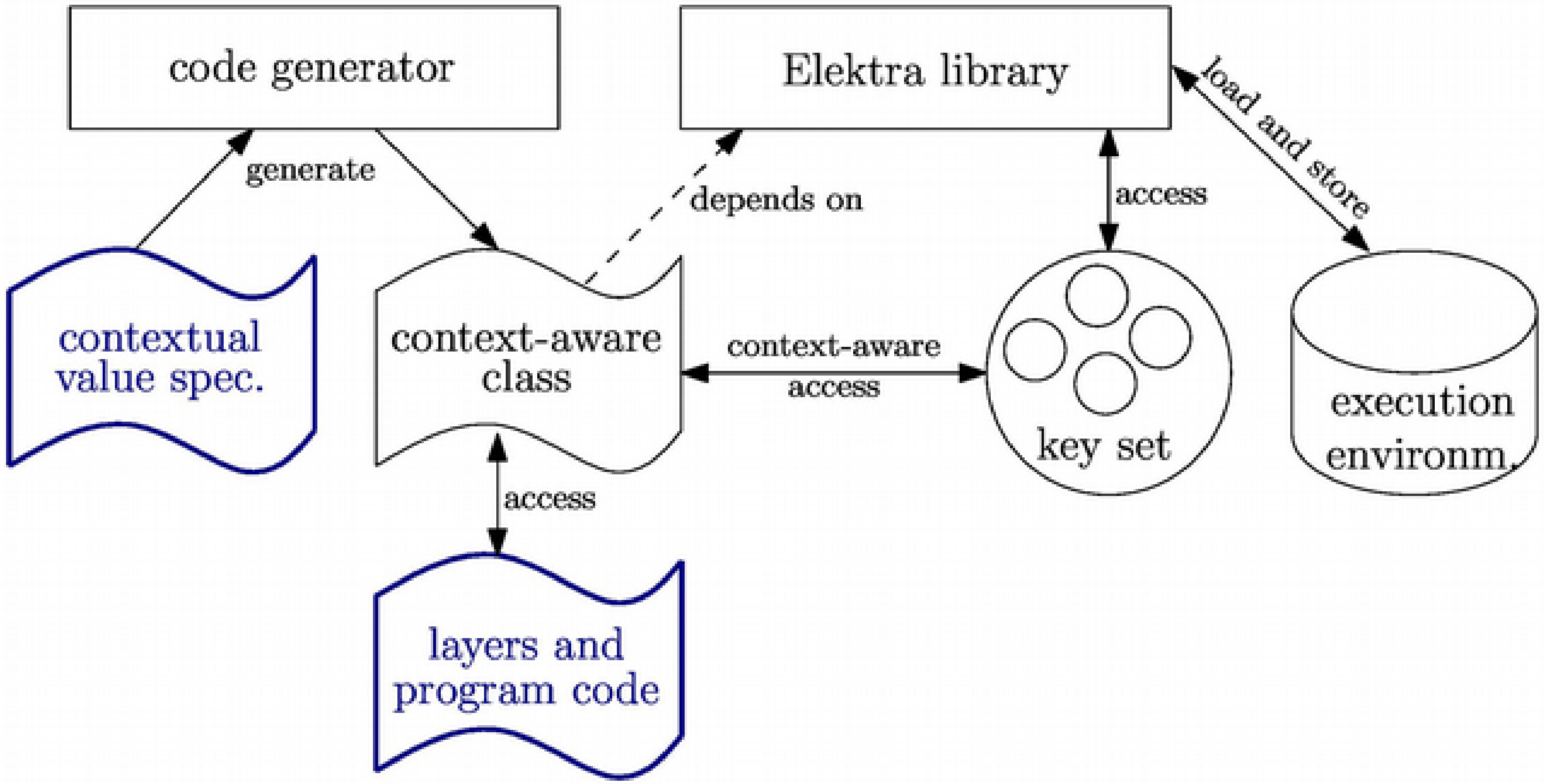
  - Generates hierarchy of CV (currently C++)
  - Placeholders for layers

Code Generation

- All CVs are initialized at runtime using PEE, that is:
  - Configuration files
  - Commandline arguments

Library

Usage



code generator
Elektra library
generate
depends on
access
load and store
contextual value spec.
context-aware class
context-aware access
key set
execution environm.
access
layers and program code

Problem

- Implementations of COP today:
  - **Performance penalties** 75%-99%
    (with active layers, w/o layer activation)
    - But PEE is accessed frequently
  - Hardly any **debugging facilities**

Methodology

- **Zero runtime-overhead on CV access?**
- Micro-benchmark
- Overhead relative to native, algorithmic code
  - int instead of Integer

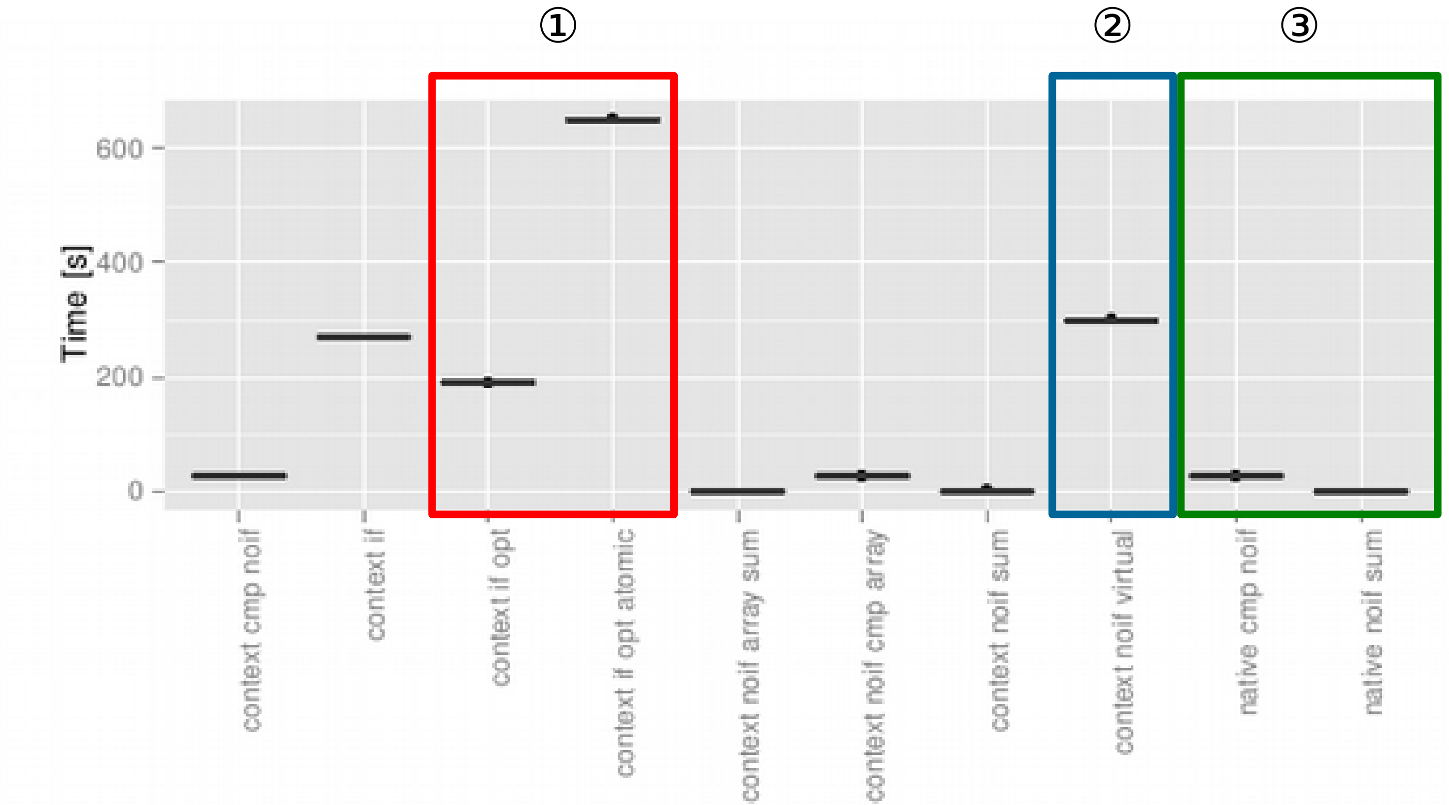
```
Integer::type add_contextual (Integer const & c1 ,
                              Integer const & c2)
{ return c1 + c2; }


with<Layer1>().with<Layer2>() ([&c]{
  int x=0;
  for (long long i =0; i < iterations ; ++ i)
  {  x ^= add_contextual (c,c); }
  dump << x << endl;
});
```

Benchmark

① Use `if` to check for context switch (**190s, 652s**)

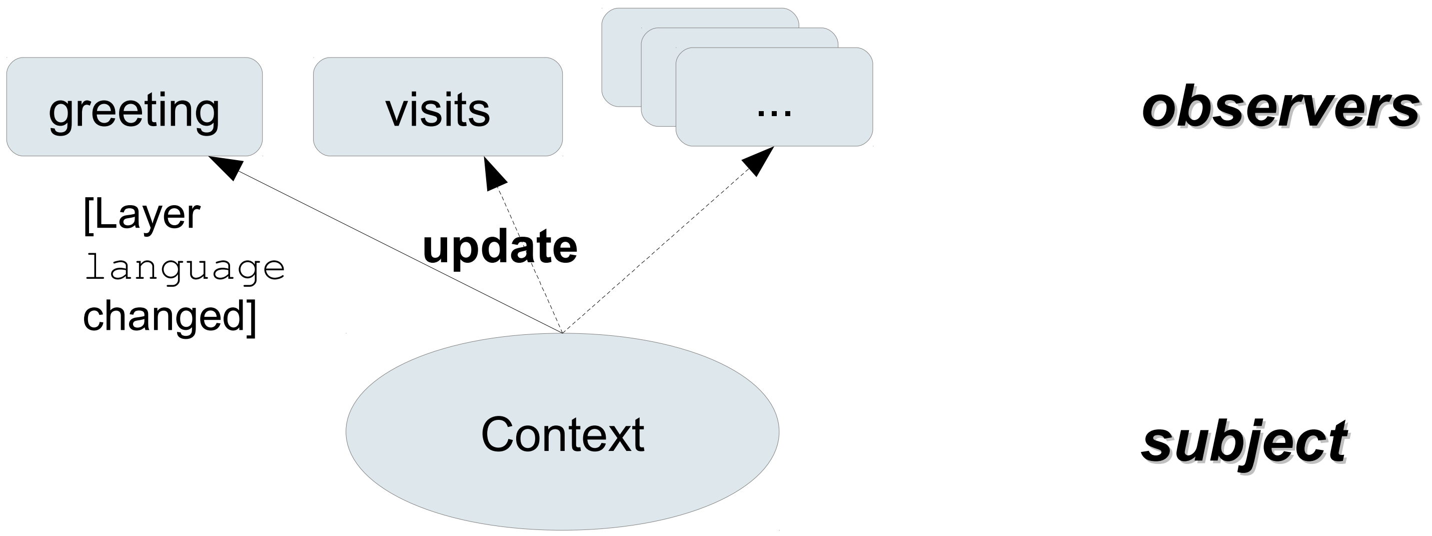
② Virtual Dispatch (**299s**)

③ Native Values : **27s** and **0s**



Implementation

```
[/%language%/%country%/%dialect%/person/greeting]
  type=String
[/%country%/person/visits]
  type=Integer
  default=0
```



Implementation Choice

- Member Array
  - No performance impact
  - But memory impact

```
operator uint32_t()
{ return g_ar[m_ind]; }
```

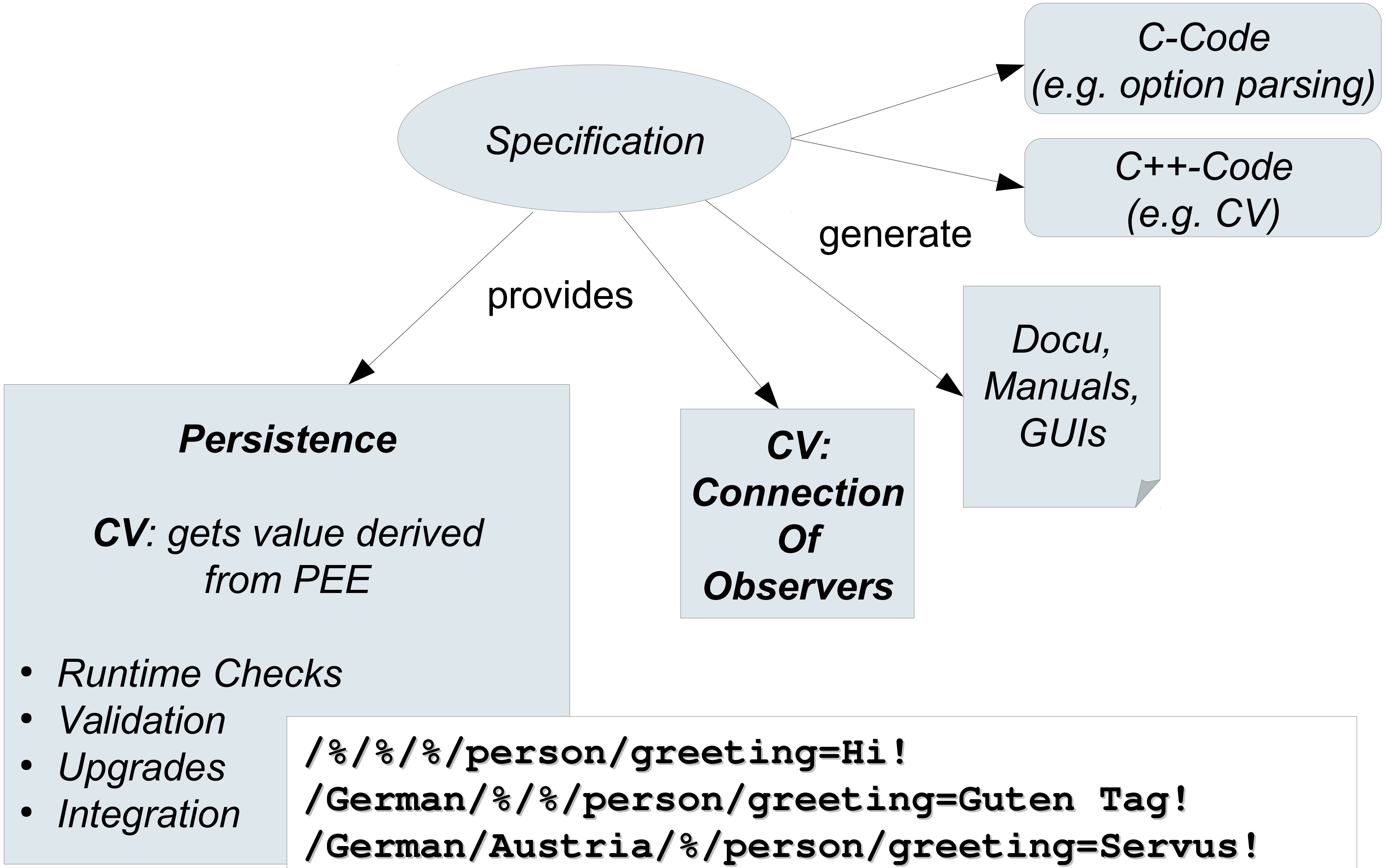
- Member Variable
  - No performance impact
  - Nearly no memory impact

```
operator uint32_t()
{ return m_cache; }
```

Evaluation

- Source Code released as free software within Elektra
  - Code generator for CV
  - Get/Set PEE ↔ CV
  - Mounting many configuration file standards
- http://www.libelektra.org

Specification



```
/%/%/%/person/greeting=Hi!
/German/%/%/person/greeting=Guten Tag!
/German/Austria/%/person/greeting=Servus!
/German/Austria/t/person/greeting=Griaß enk!
```

Debugging

- Assertions

```
assert(i.context()["language"] == "german");
assert(i.getEvaluatedName() == "/german/%/%/test");
```

- Backtraces

```
#3  0x0000000000407a56 in operator() at first.cpp:1521
    i = @0x7fffe36b69a0: { ...
      m_evaluated_name = "/german/germany/%/test" }
```

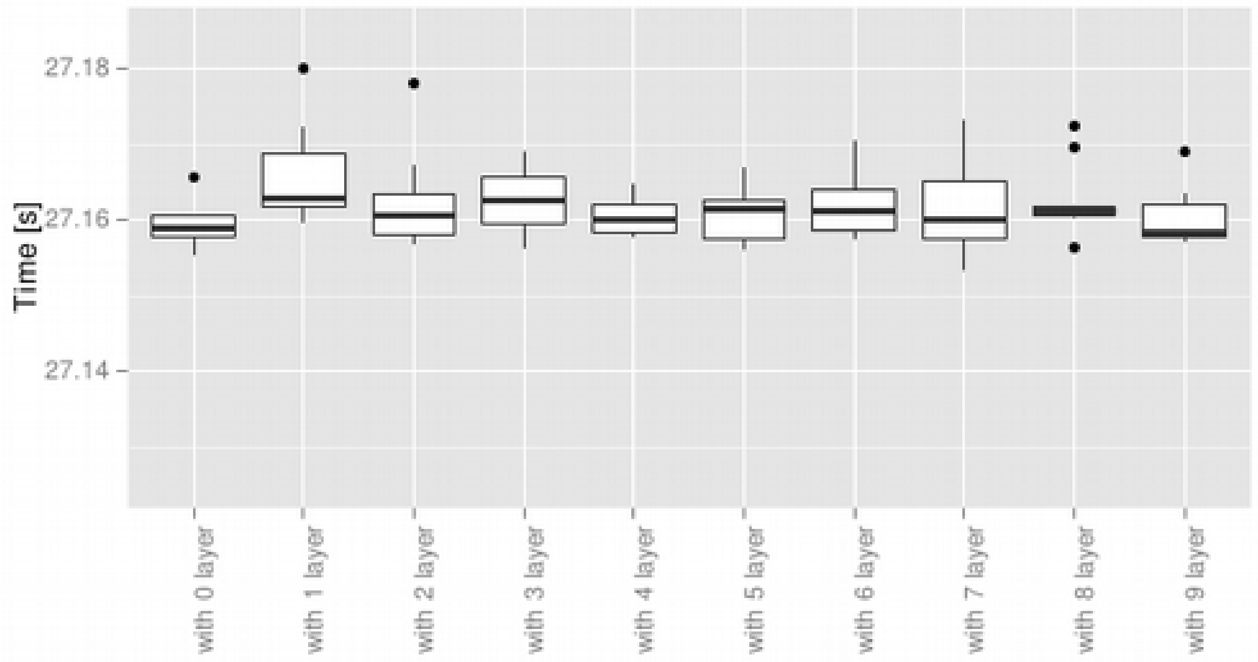
- Breakpoints

```
break 1520 if i.getEvaluatedName()
              .compare("/german/germany/%/test") == 0
```

Benchmark

- Number of active layers
- ±0.07%

  → No overhead with layers



Further Work

- Layer switching
  - Caching techniques
  - Real World Benchmark
  - More wildcards (still w/o ambiguity)
- Specification
  - More guarantees
  - Validate PEE
  - Generate Layers
- Exploit flexibility and extensibility
  - Support other programming languages
  - Support more types
  - ...

Conclusion

- Motivation for PEE as CV
- A **specification** describes PEE
- A **library** to read/modify persistent PEE
- **Evaluation** of Elektra
  - **No run-time overhead** (w/o layer activation)
  - **Unique names** support debugging and persistence

TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

Thank you for your attention!

Markus Raab <markus.raab@complang.tuwien.ac.at>
Franz Puntigam <franz@complang.tuwien.ac.at>

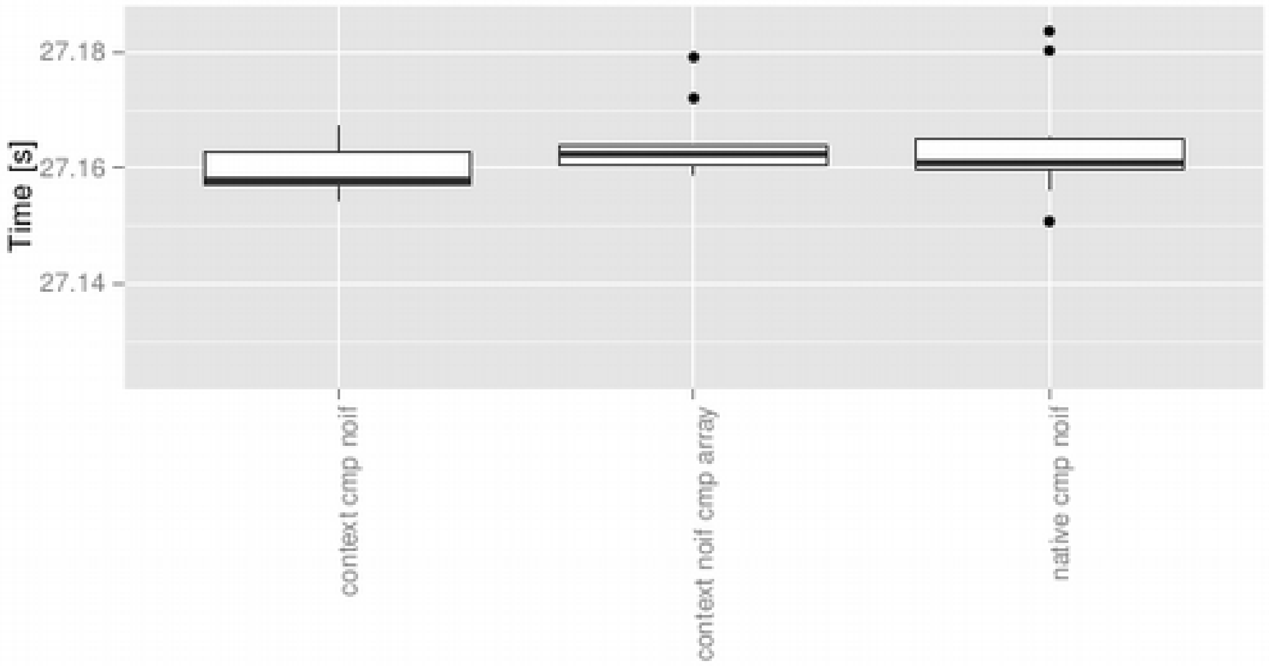
COP'14, ECOOP Uppsala Sweden
July 29, 2014

References

- Malte Appeltauer, Robert Hirschfeld,et al. *A comparison of context-oriented programming languages*.
- Pascal Costanza, Robert Hirschfeld, and Wolfgang De Meuter. *Efficient layer activation for switching context-dependent behavior.*
- Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns: elements of reusable object-oriented software*
- John Plaice and Blanca Mancilla. *The cartesian approach to context.*
- Markus Raab. *A modular approach to configuration storage.*
- Éric Tanter. *Contextual values.*
- Martin von Löwis, Marcus Denker, and Oscar Nierstrasz. Context-oriented programming: *Beyond layers.*

Benchmark

- CV compared to native variable
- Without active layer



Benchmark Setup

- Laptop: hp ® EliteBook 8570w ™
  - CPU Intel ® Core i7-3740QM @ 2.70GHz
  - 7939 MB Ram
- GNU/Linux Debian Wheezy 7.5
- gcc compiler Debian 4.7.2-5
  - with the options -std=c++11, -O2 and -Dopt=unlikely
- measured the time using **`gettimeofday`**
- Median of eleven executions
- 100 billion iterations