Informatik
Professur Betriebssysteme
Prof. Dr. Matthias Werner

Entwicklung einer echtzeitfähigen CLI-Laufzeitumgebung für den Einsatz in der Automatisierungstechnik

Echtzeit 2010

Alexej Schepeljanski Martin Däumler Matthias Werner

Echtzeit 2010 / 18. November 2010

TECHNISCHE UNIVERSITÄT
CHEMNITZ

Gliederung

1 Motivation

2 MonoVM

3 Prä-JIT

4 Trampoline

5 Ausblick

Gliederung

1 Motivation

2 MonoVM

3 Prä-JIT

4 Trampoline

5 Ausblick

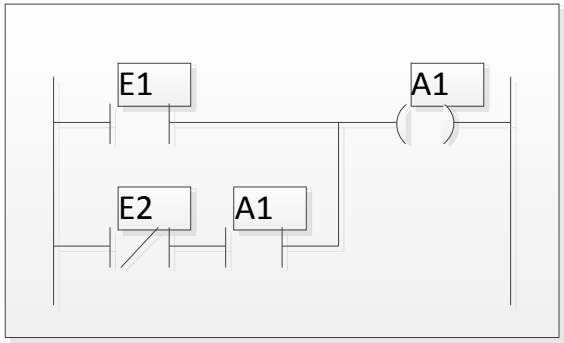
Aktuell verwendete Sprachen

EN 61131 Sprachen:

- AWL - Anweisungsliste
- KOP - Kontaktplan
- FBS - Funktionsbaustein-Sprache
- AS - Ablaufsprache
- ST - Strukturierter Text

Vorteile von EN 61131


```
LD   WERT1
ADD  WERT2
ST   OUTPUT
```




```
IF (MASCHINE_EINGESCHALTET = TRUE) THEN
    SOLLPOSITION := SOLLPOSITION + 100;
    AUSGANG1 := EINGANG1 AND EINGANG2;
ELSE
    AUSGANG1 := FALSE;
END_IF;
```

Vorteile:

- Verbreitet
- Einfach zu erlernen
- Schnell

Nachteile von EN 61131

Nachteile:

- Abstraktionsgrad zu gering
- Restriktives Typenmodell
- Keine Callback- und Interrupt-Funktionen
- Keine Objektorientierung
- Begrenzte Portierbarkeit

Rahmenbedingungen

Zusammenarbeit mit SYS TEC electronic GmbH:

- Die SPS soll mindestens in C# programmierbar sein
- Der Umstieg von den bestehenden SPS-Systemen soll einfach sein
- Es sollen strikte Echtzeitanforderungen realisiert werden können
- Als Hardware-Plattform ist mindestens eine auf 32-Bit x86-CPUs basierende Architektur vorgesehen
- Aus lizenzrechtlichen Gründen sollte Open-Source-Software verwendet werden

Vorteile einer Managed Umgebung

Vorteile von einer modernen Programmiersprache wie C#:

- Hoher Abstraktionsgrad
- Objektorientierung
- Vererbung, Interfaces, Exception-Handling und Events
- Gute Portierbarkeit

```
static class Program
 {
  [STAThread]
    static void Main()
    {
     Component tc2 = new Component("Komp 2");
     Component tc4 = new Component("Komp 4");
     tc2.EndEvent += tc4.StartEvent;
     ThreadPool.QueueUserWorkItem(new
WaitCallback(tc2.execute), tc2);
       }}
```

Problem

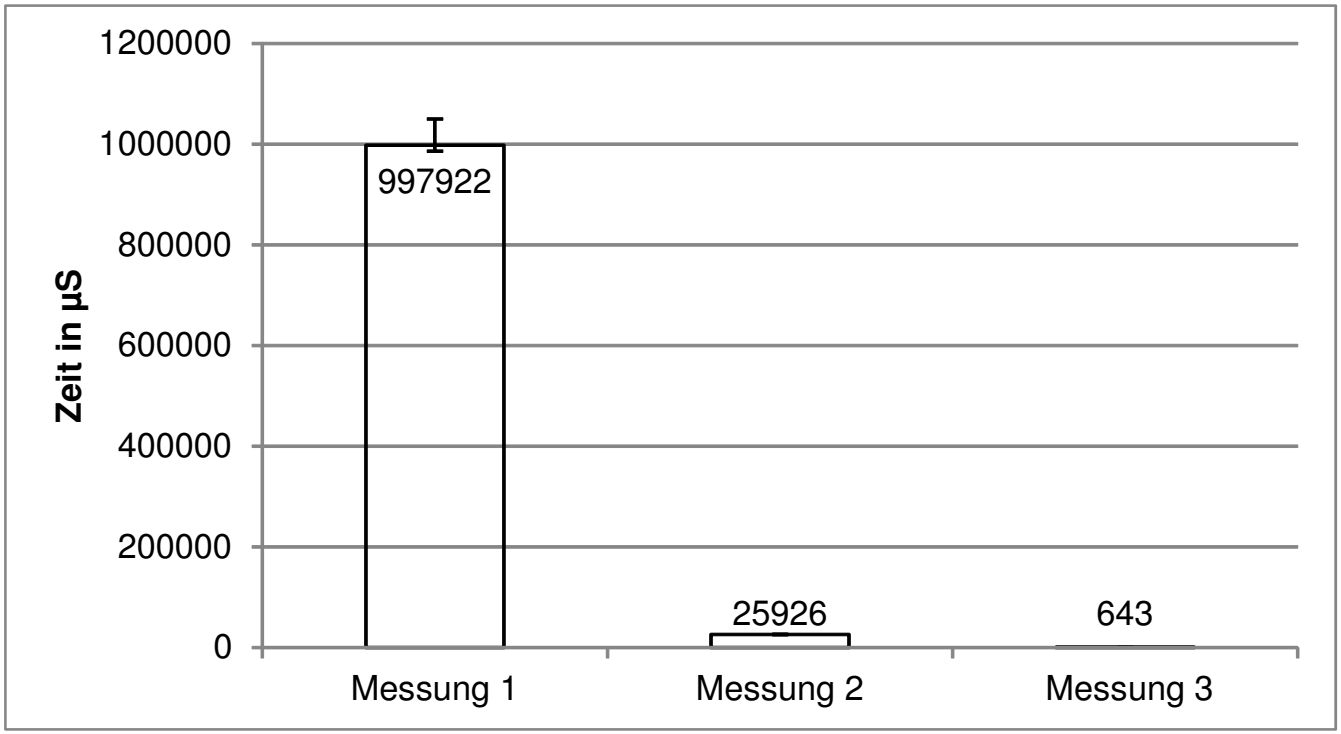


Abbildung: 5000 Methodenaufrufe

Gliederung

1 Motivation

2 MonoVM

3 Prä-JIT

4 Trampoline

5 Ausblick

Vorteile

- Am weitesten verbreitet unter den freien Implementierungen
- Größter Funktionsumfang unter den freien Implementierungen
- Debugger vorhanden
- Quellcode verfügbar
- Steht unter GPL

Probleme

- Nichtdeterminismen durch die VM
  - **Just In Time-Compilier**
  - Garbage Collector
  - Trampoline
  - *late-binding*-Prinzip
- Dynamische Sprachfeatures
- Nicht echtzeitfähige Bibliotheken ?

Just In Time-Compiler

Just In Time-Compiler:

- Code zur Laufzeit übersetzt
- Cache für bereits übersetzten Code
- Ausführung von Code dauert unterschiedlich lange
- Verwendung von „Trampolines“

Gliederung

1 Motivation

2 MonoVM

3 Prä-JIT

4 Trampoline

5 Ausblick

Prä-JIT-Compilierung

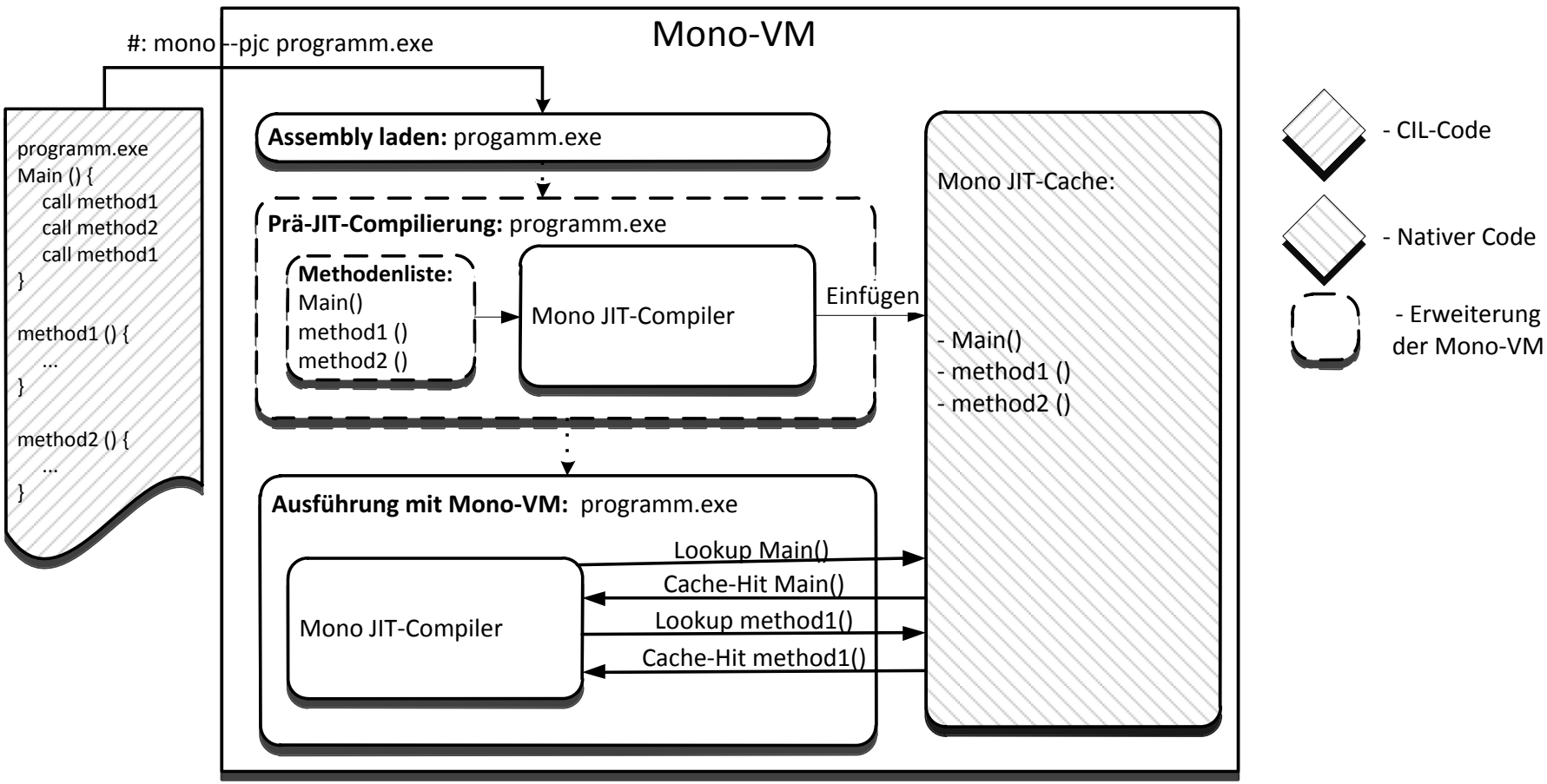
Idee:

- JIT auf alle Methoden schon beim Start anwenden
- Ergebnisse der JIT-Compilierung im JIT-Cache speichern

Vorteile:

- Generischer Ansatz
- Keine Veränderung am Compiler notwendig
- Wiederverwendung von Mono-Infrastruktur
- Alle Compileroptimierungen möglich

PräJIT funktionsweise



Testergebnisse

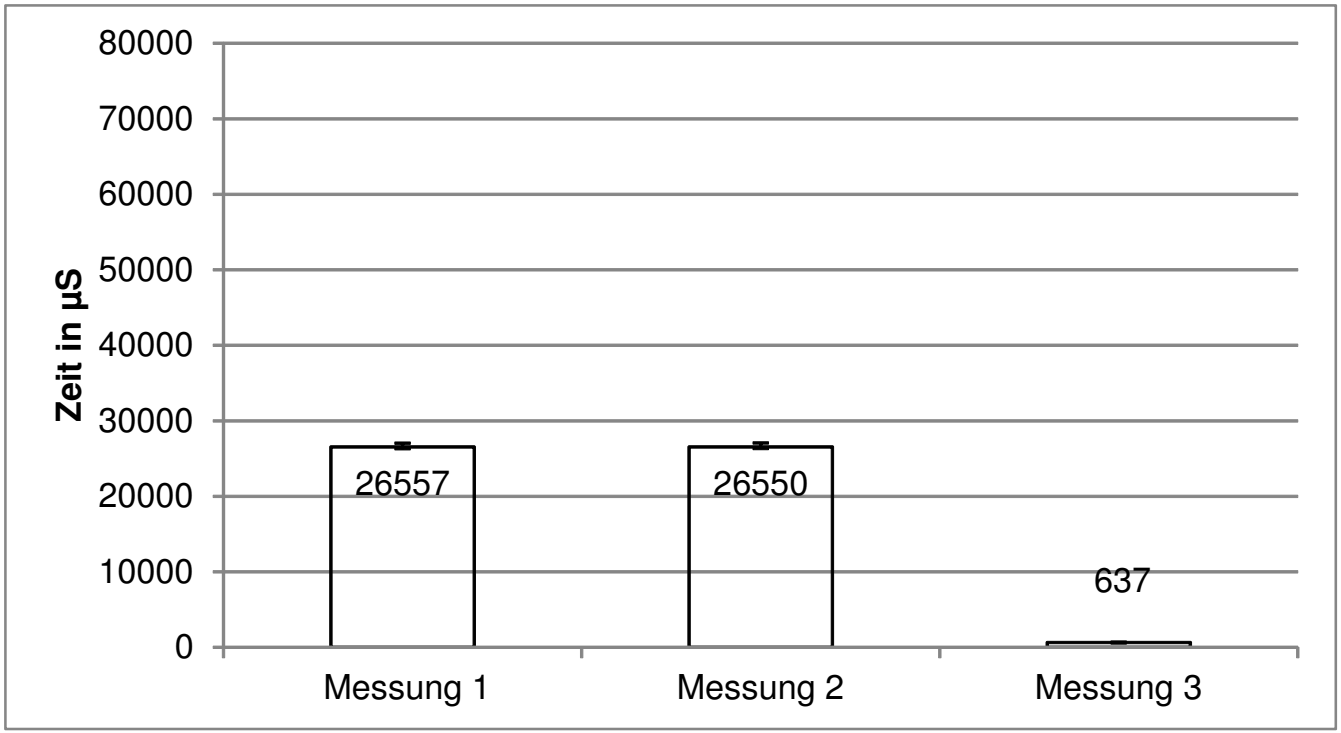


Abbildung: 5000 Methodenaufrufe mit PräJIT

Gliederung

1 Motivation

2 MonoVM

3 Prä-JIT

4 Trampoline

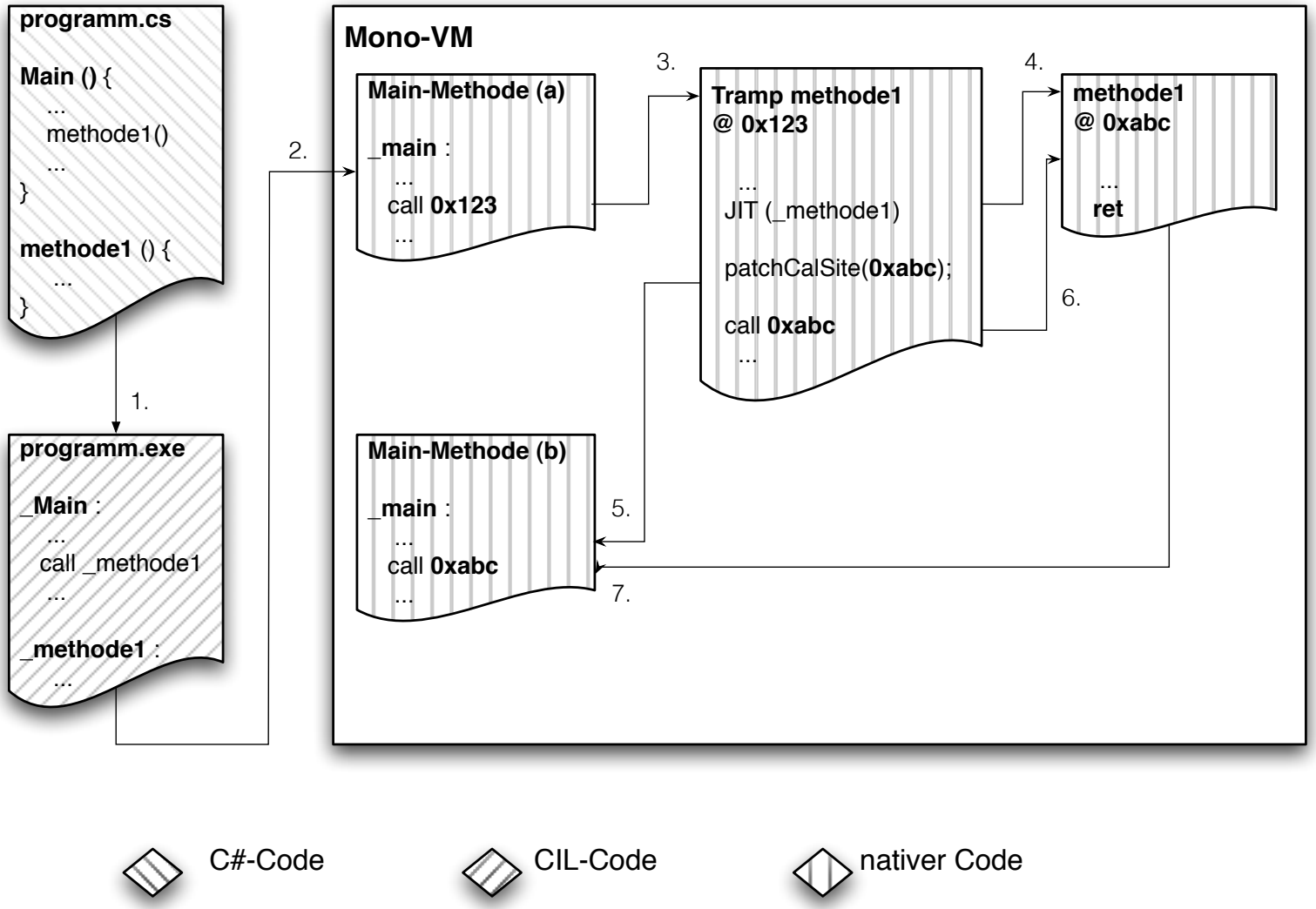
5 Ausblick

Trampoline

Trampoline:

- Kleine Mono-VM Funktionen
- Werden zwischen Aufruf und Call-Ziel eingeschoben
- Bereiten den Aufruf vor
- Können die ursprüngliche Call-Anweisung patchen
- Funktionsaufrufe benötigen dadurch unterschiedlich lange

Trampoline


programm.cs
Main () {
...
methode1()
...
}
methode1 () {
...
}
1.
programm.exe
_Main :
...
call _methode1
...
_methode1 :
...
2.
Mono-VM
Main-Methode (a)
_main :
...
call 0x123
...
3.
Tramp methode1
@ 0x123
...
JIT (_methode1)
patchCalSite(0xabc);
call 0xabc
...
4.
methode1
@ 0xabc
...
ret
6.
Main-Methode (b)
_main :
...
call 0xabc
...
5.
7.
C#-Code
CIL-Code
nativer Code

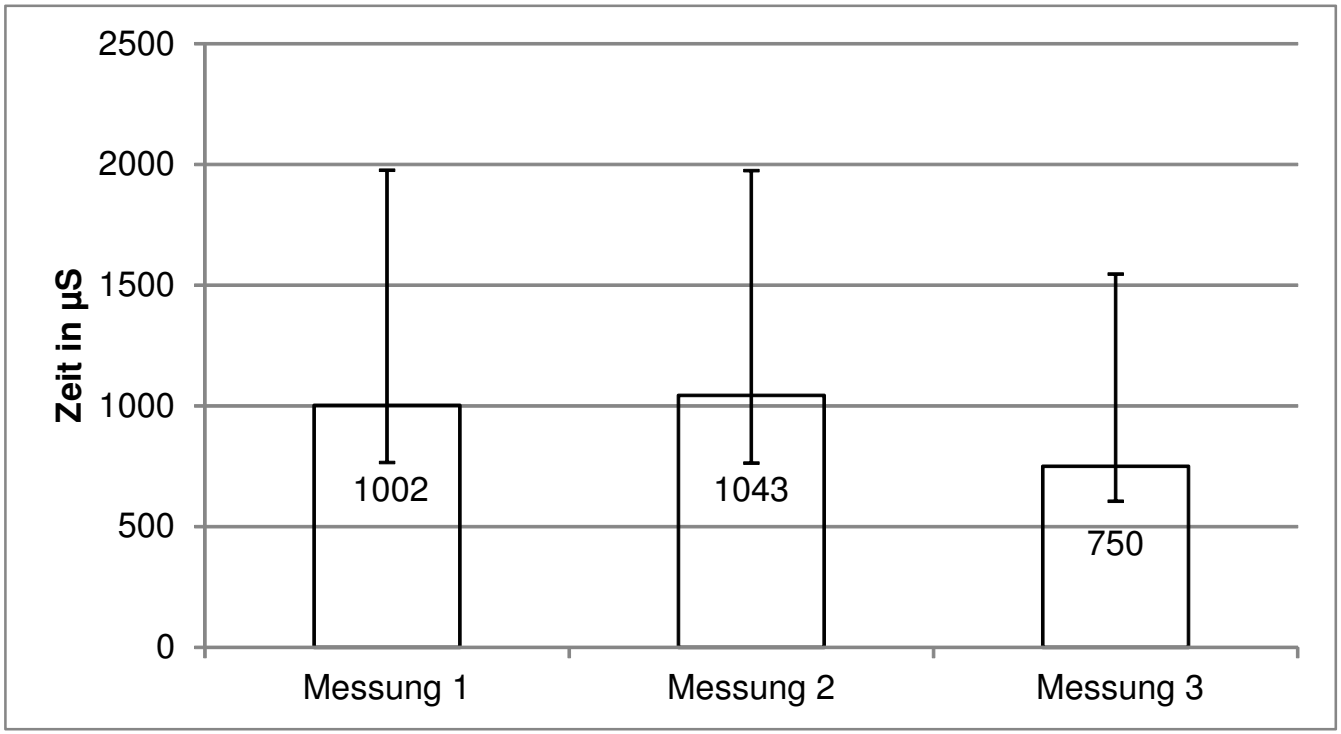


Abbildung: 5000 Methodenaufrufe mit PräPatch und PräJIT

Gliederung

1 Motivation

2 MonoVM

3 Prä-JIT

4 Trampoline

5 Ausblick

Zusammenfassung

- Moderne Programmiersprache bietet wesentliche Vorteile
- JIT einer der größten Nichtdeterminismus-Quellen
- Durch Prä-JIT lässt sich diese Quelle vermeiden
- Aber: Höhere Startzeit

Ausblick

- Laufzeit-Patchen
- Komponentenbasiertes Programmiermodell (Ähnlich EN61499)
- Garbage Collector
- *Late binding* -Prinzip

FRAGEN?