Karlsruhe Institute of Technology

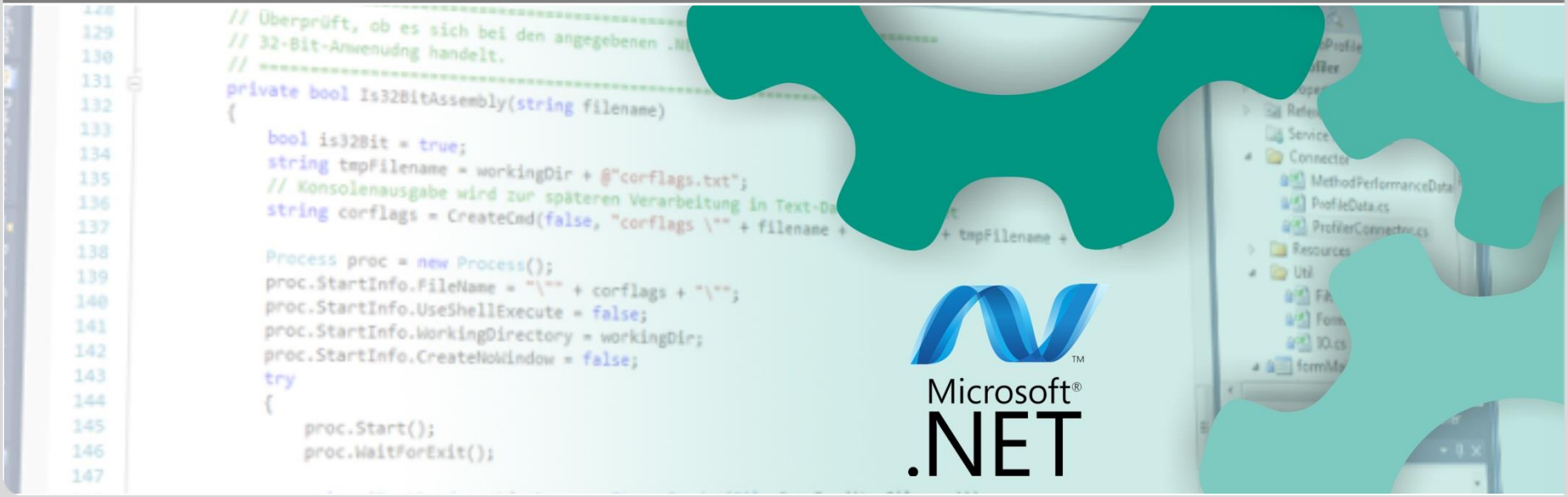
SIEMENS

Corporate Technology

Forschungskooperation KIT und Siemens: Musterbasierte Anwendungsparallelisierung

Luis Manuel Carril, Korbinian Molitorisz, Dr. Frank Padberg

IPD Prof. Walter F. Tichy – Lehrstuhl für Programmiersysteme



Forschungszentrum Karlsruhe
in der Helmholtz-Gemeinschaft

Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Wozu befassen wir uns mit Parallelisierung?

- „*The free lunch is over*“ - Multikernprozessoren benötigen parallele Software
- Aber: Entwicklung paralleler Software ist schwierig

Was macht die Entwicklung paralleler Software schwer?

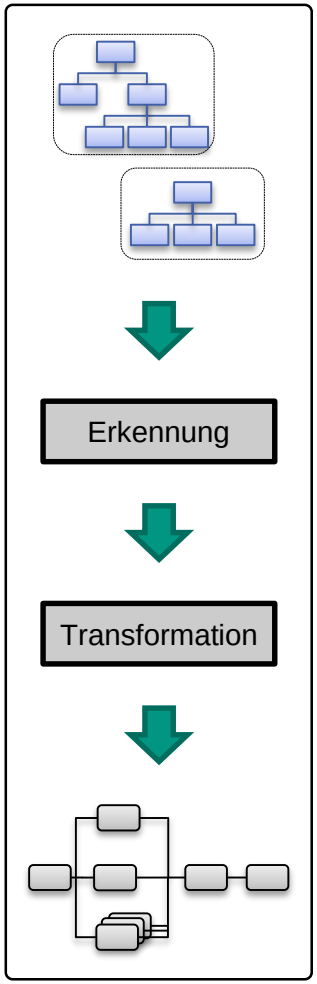
- Parallelisierung ist langwierig und fehleranfällig
  - Mehrere Wochen Arbeit zur Implementierung einer Pipeline (Parallelisierung einer Videosoftware [OS+10])
  - Erst die dritte Parallelisierungsstrategie erzielte Beschleunigung (Desktop-Suchmaschine [MT10])
  - „*Mehrkernprozessoren mittlerweile allgegenwärtig. Das Wissen, sie zu programmieren nicht*“ [VM11]

⇨ Mustergestütztes Parallelisierungsverfahren zur Anwendungsparallelisierung nötig [SM+13]

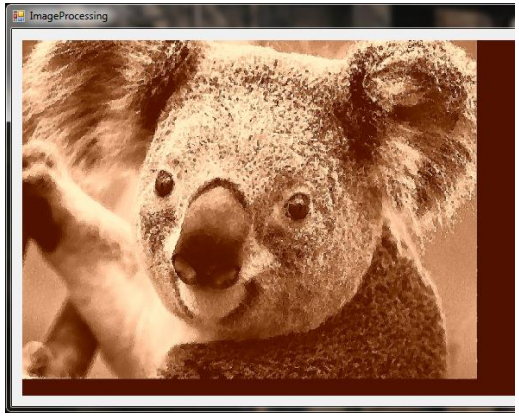
[OS+10] – Frank Otto, Christoph Schafer et al. *A language-based tuning mechanism for task and pipeline parallelism*, Euro-Par 2010
[MT10] – David Meder, Walter F. Tichy. Parallelizing an Index Generator for Desktop Search, ISCA 2010
[SM+13] – Jochen Schimmel, Korbinian Molitorisz, Ali Jannesari, Walter F. Tichy. *Automatic Generation of Parallel Unit Tests*, AST 2013
[VM11] – Hans Vandierendonck, Tom Mens. *Averting the Next Software Crisis*, 2011

Bestandteile musterbasierter Parallelisierung

- Musterkatalog
  - Quellmuster
  - Zielmuster
- Prozedurale Methode
  - Erkennung
  - Transformation
- Spezifikation der erkannten Muster
- Validierung der Korrektheit
- Laufzeitoptimierung



Erkennung von Quellmustern




```
01  AviStream Process(AviStream aviIn)
02  {
03   AviStream aviOut = new AviStream();
04   foreach(Image i in aviIn.Images)
05   {
06    Image crop  = cropFilter.Apply(i);
07    Image histo = histogramFilter.Apply(i);
08    Image oil   = oilFilter.Apply(i);
09    Image res   = ConvTo32bpp.Apply(crop, histo, oil);
10    aviOut.Images.Add(res);
11   }
12   return aviOut;
13  }
```

- Coderegion mit hohem Laufzeitanteil
- Schachtelung von Anweisungen
  - Ausdrucksanweisung – Anweisungsblock – `return`-Anweisung
- Schrittweise Anwendung verschiedener Bildfilter
- Programmstrukturen
  - Programmschleife: Iterativer Anweisungsstrom
  - Anweisungsfolge: Sequenz aufeinanderfolgender Anweisungen

Erkennung von Quellmustern

```
01  AviStream Process(AviStream aviIn)
02  {
03   AviStream aviOut = new AviStream();
04   foreach(Image i in aviIn.Images)                    1.440x
05   {
06    Image crop  = cropFilter.Apply(i);                    8%
07    Image histo = histogramFilter.Apply(i);               9%
08    Image oil   = oilFilter.Apply(i);                    68%
09    Image res   = ConvTo32bpp.Apply(crop, histo, oil);   12%
10    aviOut.Images.Add(res);                               3%
11   }
12   return aviOut;
13  }
```

- Programmstruktur: Programmschleife
  - Ein Faden pro Iteration bzw. Iterationsfolge
  - Ein Faden pro Anweisung
- Programmstruktur: Anweisungsfolge
  - Ein Faden pro Anweisung
  - Ein Faden pro Anweisungsfolge

Abbilden auf Zielmuster

```
01  AviStream Process(AviStream aviIn)
02  {
03   AviStream aviOut = new AviStream();
04   foreach(Image i in aviIn.Images)                    1.440x
05   {
06    Image crop  = cropFilter.Apply(i);                    8%
07    Image histo = histogramFilter.Apply(i);               9%
08    Image oil   = oilFilter.Apply(i);                    68%
09    Image res   = ConvTo32bpp.Apply(crop, histo, oil);   12%
10    aviOut.Images.Add(res);                               3%
11   }
12   return aviOut;
13  }
```

- Zielmuster
  - Programmschleife → Fließband
    - Anweisungen: Stufen
    - Videostrom: Daten
  - Anweisungsfolge → Master-/Worker
    - Kontrollfaden: Master
    - Anweisungen: Worker
- Speedup (auf 8-Kernrechner)
  - Manuell: 6,47. Aufwand: 8 Stunden
  - Automatisch: 7,18. Aufwand: 10 Minuten

Demo

Evaluierung: Benchmark

- Bewertung der musterbasierten Parallelisierung für *pipelines* und *master/worker* in 26.500 LOC

Projekt	Zeilen	Methoden	Programmstrukturen	
			Anweisungsfolge	Programm-schleife
MergeSort	94	6	14	5
RayTracer	522	37	58	5
DesktopSearch	315	22	40	7
CompGeo	1.058	73	149	20
ImageProcessing	110	12	14	2
PowerCollections	24.481	1.252	3.354	287

Evaluierung: Suchraumreduktion und Trefferquote

Projekt	Programm-strukturen	Zielmuster	Surchraum-reduktion
MergeSort	19	1	95 %
RayTracer	63	6	90 %
DesktopSearch	47	2	96 %
CompGeo	169	1	98 %
ImageProcessing	16	1	94 %
PowerCollections	3.641	172	97 %
			Ø = 95%

Evaluierung: Präzision und Speedup

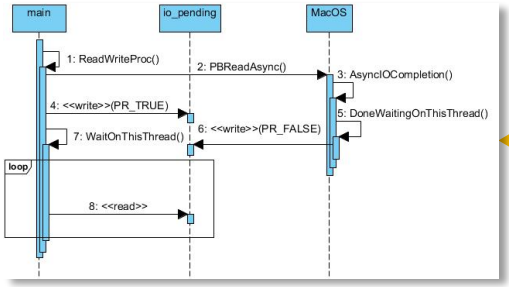
- Testsystem: AMD FX 8120h, 8 Prozessorkerne, je 3,1 GHz

Projekt	Ziel-muster	Korrektheit		Präzision	Speedup		Speedup
		K	¬K		S	¬S	
MergeSort	1	1	0	100 %	1	0	4,60
RayTracer	6	6	0	17 %	1	5	3,13
DesktopSearch	2	2	0	50 %	1	1	1,67
CompGeo	1	1	0	100 %	1	0	1,20
ImageProcessing	1	1	0	100 %	1	0	7,18
PowerCollections	15*	13	2	27 %	6	9	--
	∑ = 26	∑ = 24	∑ = 2	Ø = 66%	∑ = 11	∑ = 15	

Musterbasiert Wettlauferkennen in Entwurf

- Frühzeitige Erkennung von parallelen Defekten (Aktuell: Im Entwurf)
- Formale Modellierung von Design-Diagrammen
- Fehlermustersuche durch *model checking*
- Markierung verdächtiger Teile im Diagramms
- Automatisierter Ansatz

Werkzeugkette



Potentielle Fehler!

mc_MAIN_MACOS.pbreadasync -> wr_MACOS_IOPENDING -> wr_MAIN_IOPENDING -> rd_MAIN_IOPENDING -> STOP

Modell-erzeugung

Modell-prüfer

MOZILLA97866 = (MAIN || IOPENDING || MACOS)
MAIN = (readwriteproc_MAIN -> mc_MAIN_MACOS.pbreadasync -> wr_MAIN_IOPENDING -> waitonthisthread_MAIN -> MAIN#L)
MAIN#L = (rd_MAIN_IOPENDING -> MAIN#L Π rd_MAIN_IOPENDING -> SKIP)
IOPENDING = (IOPENDING#S1 || IOPENDING#S2)
IOPENDING#S1 = (wr_MACOS_IOPENDING -> IOPENDING#S1) …

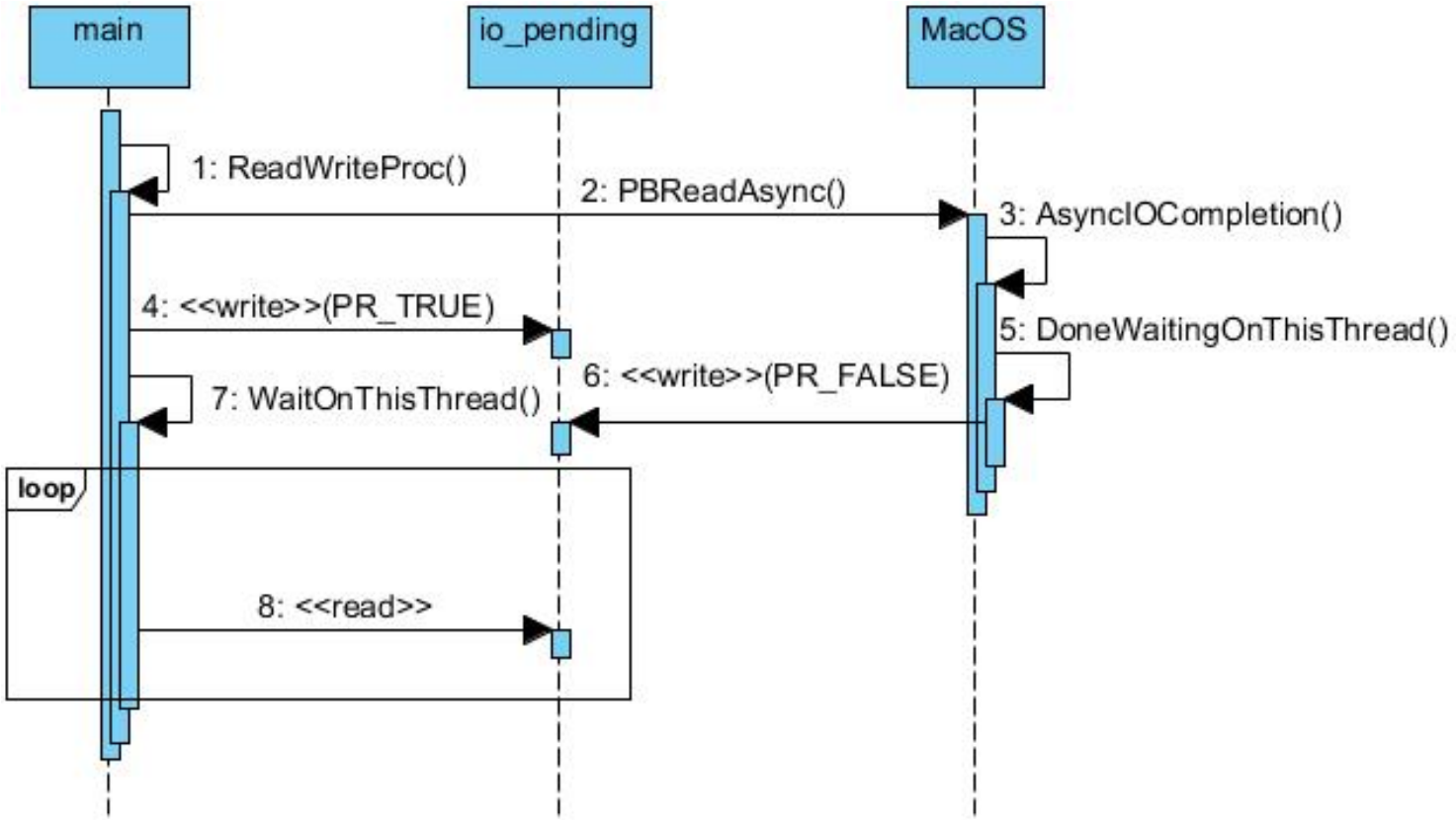
Muster-erkennung

Muster-katalog

Katalog der Fehlermuster

Name	Muster
Atomicity violation with one variable	rd_P1_SD → wr_P2_SD → wr_P1_SD → STOP rd_P1_SD → wr_P2_SD → rd_P1_SD → STOP
Access after deletion	od_SD → mc_P1_SD.method → STOP
Access before creation	mc_P1_SD.method → oc_SD → STOP

Demo: Mozilla Bug 97866



Ausblick

- Erweiterung des Verfahrens auf Laufzeitspuren
- Performanzvergleich gegen gängige Wettlauferkenner
- Fokus auf *atomicity violations* und korrelierte Variablen