Karlsruhe Institute of Technology

Fortress

Seminar: Sprachen für die Parallelprogrammierung

Eduard Frank

IPD Snelting, Lehrstuhl Programmierparadigmen

Übersicht

Karlsruhe Institute of Technology

1. Motivation
2. Fortress Grundlagen
3. Big Deal
4. Zusammenfassung

Motivation

Karlsruhe Institute of Technology

Motivation

Karlsruhe Institute of Technology

For Schleife

```
for(i = 0; i < N; i++)
    sum = sum + x[i];
```

Motivation

Karlsruhe Institute of Technology

For Schleife

```
#pragma omp parallel for reduction(+:sum)
for(i = 0; i < N; i++)
    sum = sum + x[i];
```

Motivation

For Schleife

```
#pragma omp parallel for reduction(+:sum)
for(i = 0; i < N; i++)
    sum = sum + x[i];
```

Warum nicht ?

$$\sum x$$

Fortress Grundlagen

Aber wo ist die $\sum$ Taste ?

- Unicode
- “Twiki” Eingabe mittels ASCII

```
A UNION {1,2,3,4}
```

$$A \cup \{1,2,3,4\}$$

```
SUM[k <- 1:n] a[k] x^k
```

$$\sum_{k \leftarrow 1:n} a_k \, x^k$$

```
BIG MAX[(j,k)<-a.indices] |a[j,k]-b[j,k]|
```

$$\mathbf{MAX}_{(j,k) \leftarrow a.indices} |a_{j,k} - b_{j,k}|$$

Fortress Grundlagen

- Objektorientiert + funktionale Features
- Statisch typisiert
- Basistypen
  - $\mathbb{Z}32$, $\mathbb{Z}64$ - Ganzzahlenwert
  - $\mathbb{R}32$, $\mathbb{R}64$ - Fliesskommawert
- Variablen
  - Immutable: $one\colon \mathbb{Z}32 = 1$
  - Mutable: $one\colon \mathbb{Z}32 := 1$ oder `var` $one := 1$
- Bedingungen:
  - `if` $0 \leq n \leq 10$ `then`
- Comprehensions
  - $A = \{k \mid k \leftarrow 1:100, prime\, k\}$

Fortress Grundlagen

Ziele

- Fortress = **Secure Fortran** (“Do for Fortran what Java did for C”)
- Produktivität
- Skalierbarkeit
- „Growable“ [1]

Implizite Parallelisierung

Karlsruhe Institute of Technology

- “Potentially parallel”
- Also Blocks:

```
do f(x) also do g(y) end
```

- Tuples:

```
(a,b,c) = (f(x),g(y),h(z))
```

- Operatoren:

$x^y + z^x$

Implizite Parallelisierung

- For Schleifen

Parallele For Schleife

```
for i ← 1:10 do
    print i
end
```

4 2 1 3 5 8 9 10 7 6

Sequentielle For Schleife

```
for i ← seq(1:10) do
    print i
end
```

1 2 3 4 5 6 7 8 9 10

Implizite Parallelisierung

- Generator
  - Steuert die Parallelisierung eines Schleifen/Funktions-körpers
- Reduction
  - Verknüpft zwei Werte miteinander und bietet einen Standardwert
  - Operation (Verknüpfung) muss assoziativ sein und über ein neutrales Element verfügen. (Monoid)

Implizite Parallelisierung

- Generator
  - Steuert die Parallelisierung eines Schleifen/Funktions-körpers

Generator

```
trait Generator⟦E⟧
    generate⟦R⟧(r: Reduction⟦R⟧, body: E → R): R
end
```

- Reduction
  - Verknüpft zwei Werte miteinander und bietet einen Standardwert
  - Operation (Verknüpfung) muss assoziativ sein und über ein neutrales Element verfügen. (Monoid)

Reduction

```
trait Reduction⟦L⟧
    empty(): L
    join(a: L, b: L): L
end
```

Implizite Parallelisierung

- Beispiel: Range Parallelisierung

Summation - Generator, Reduction

$$\sum_{i \leftarrow 1:100} i$$

$$(1:100).generate\Big(\text{SumZZ32Reduction}, \texttt{fn}\ (i) \Rightarrow (i)\Big)$$

```
object SumZZ32Reduction extends Reduction [[Z32]]
    empty() : Z32 = 0
    join(a : Z32, b : Z32) : Z32 = a + b
end
```

Implizite Parallelisierung

Karlsruhe Institute of Technology

Range Generator

```
object Range(lo: ℤ64, hi: ℤ64) extends Generator⟦ℤ64⟧
    size := hi − lo + 1
    generate⟦R⟧(reduction: Reduction⟦R⟧, body: ℤ64 → R): R =
        if size < 10 then
            r : R = reduction.empty()
            i : ℤ64 := lo
            while i ≤ hi do
                v : R = body(i)
                r := reduction.join(r, v)
                i += 1
            end
            r
```

Implizite Parallelisierung

Range Generator

```
    else
      mid = ⌊(lo + hi)/2⌋
      reduction.join(Range(lo, mid).generate⟦R⟧(reduction, body),
                     Range(mid + 1, hi).generate⟦R⟧(reduction, body))
    end
  end
end
```

- Recursive Subdivision: Aufteilen des Eingaberaums in ungefähr gleich große Chunks.

Implizite Parallelisierung

Karlsruhe Institute of Technology

Generator, Reduction Beispiele

$\texttt{for}\ i \leftarrow 1:100\ \texttt{do}\ a_i := 2^i\ \texttt{end}$

$A = \{\, 2^i \mid i \leftarrow 1:100 \,\}$

$$n = \sum_{i \leftarrow 1:100} 2^i$$

$z = (1:100).generate([\![\mathbb{Z}32]\!]\big(\text{ForReduction}, \texttt{fn}\ (k) \Rightarrow (a_k := 2^k)\big)$

$z = (1:100).generate([\![\mathbb{Z}32]\!]\big(\text{SetReduction}, \texttt{fn}\ (k) \Rightarrow (2^k)\big)$

$z = (1:100).generate([\![\mathbb{Z}32]\!]\big(\text{SumZZ32Reduction}, \texttt{fn}\ (k) \Rightarrow (2^k)\big)$

Zusammenfassung

- Mathematische Notation bietet sehr viele Vorteile
- Einfache Parallelisierung
- Interpreter (Java 1.6) - sehr langsam
- Compiler in Entwicklung (JVM)
- “Growable” ?
- Viele weitere Features:
  - Software Transactional Memory
  - Versionskontrolle für Komponenten
  - Operatorüberladung
  - “Progamming in the large”
  - . . .

Literatur I

- Guy Steele *“Growing a Language” keynote talk, OOPSLA 1998*
- Maurice Herlihy, Victor Luchangco, Mark Moir *A Flexible Framework for Implementing Software Transactional Memory*
- David Chase, Yossi Lev *Dynamic CircularWork-Stealing Deque*
- Eric Allen, David Chase, Joe Hallett, Victor Luchangco, Jan-Willem Maessen, Sukyoung Ryu, Guy L. Steele, and Sam Tobin-Hochstadt *TheFortress Language Specification*
- Project Fortress *http://projectfortress.sun.com/*
- Wikipedia *http://en.wikipedia.org/wiki/Fortress_%28programming_language%29*
- Fortress Programming Language Tutorial `http://research.sun.com/projects/plrg/PLDITutorialSlides9Jun2006.pdf`

Literatur II

- Fortress: A New Programming Language for Scientific Computing *http://labs.oracle.com/projects/plrg/Publications/1.02_steele.pdf*
- Fortress 0.62 *http://labs.oracle.com/projects/plrg/Publications/PPoPPPanel.pdf*
- Fortress for Productive Computing *http://labs.oracle.com/projects/plrg/Publications/Fortress-PMUA.pdf*
- Parallelism in Fortress *http://labs.oracle.com/projects/plrg/Publications/PGAS.pdf*
- Parallel Programming and Parallel Abstractions in Fortress *http://labs.oracle.com/projects/plrg/Publications/PACTSept2005.pdf*
- The Fortress Programming Language *http://labs.oracle.com/projects/plrg/Publications/JapanLecture2006public.pdf*

Literatur III

- Parallel Programming and Parallel Abstractions in Fortress *http://labs.oracle.com/projects/plrg/Publications/Aarhus-Fortress-Parallelism-2006public.pdf*
- A Growable Language *http://labs.oracle.com/projects/plrg/Publications/OOPSLA-GrowableLanguage-2006public.pdf*
- Project Fortress *http://labs.oracle.com/projects/plrg/Publications/allen-fortressintro.pdf*
- Object-Oriented Programming in Fortress *http://labs.oracle.com/projects/plrg/Publications/allen-oo-fortress.pdf*
- Fortress: A New Programming Language for Scientific Computing *http://labs.oracle.com/projects/plrg/Publications/SNU.pdf*
- What's Cool about Fortress *http://labs.oracle.com/projects/plrg/Publications/2007-0410.pdf*

Literatur IV


Growing the Fortress Programming Language by Example *http://labs.oracle.com/projects/plrg/Publications/2008-0157.OH08-Ryu.pdf*


A Growable Language *http://labs.oracle.com/projects/plrg/Publications/OOPSLA-GrowableLanguage-2006public.pdf*


Project Fortress: A New Programming Language from Sun Labs *http://labs.oracle.com/projects/plrg/Publications/2008-0218.JavaOne.pdf*


Fortress Boot Camp Material *http://labs.oracle.com/projects/plrg/Publications/BootCamp2008.html*


Fortress: Parallel Programming Through Extensible Bulk Operations *http://labs.oracle.com/projects/plrg/Publications/Rochester-Nov2008.pdf*

Literatur V


The Extraordinary Algebra of List Comprehensions *http://labs.oracle.com/projects/plrg/Publications/NEPLS-Mar2009-comprehensions.pdf*


Growing the Fortress Programming Language by Example *http://labs.oracle.com/projects/plrg/Publications/2008-0157.OH08-Ryu.pdf*


The Future Is Parallel: What's a Programmer to Do? Breaking Sequential Habits of Thought *http://labs.oracle.com/projects/plrg/Publications/NEPLSMarch2009Steele.pdf*


A Short Hands-On Introduction to Fortress *http://labs.oracle.com/projects/plrg/Publications/MITtutorial2009.pdf*


Organizing Functional Code for Parallel Execution; or, foldl and foldr Considered Slightly Harmful *http://labs.oracle.com/projects/plrg/Publications/ICFPAugust2009Steele.pdf*

Literatur VI


Project Fortress: A Multicore Language for Multicore Processors *http://labs.oracle.com/projects/plrg/Publications/linuxMagazine.pdf*


Fortress Presentation *http://www.slideshare.net/alexmiller/project-fortress*


The Soul of Fortress *http://labs.oracle.com/minds/2005-0302/*


Parallel by Default *http://blogs.sun.com/simons/entry/fortress_parallel_by_default*

Ende

```
println("Vielen Dank fuer ihre Aufmerksamkeit")
            for s <- students do
                s.askQuestion
            end
```