Karlsruher Institut für Technologie

Cilk

Sprache für Parallelprogrammierung

IPD Snelting, Lehrstuhl für Programmierparadigmen

David Soria Parra

Karlsruher Institut für Technologie

Geschichte

Geschichte

- Entwickelt 1994 am MIT Laboratory for Computer Science
- Cilk 1: Continuations
- Cilk 2 Einführung von cilk2c. Abstraktion von Continuation Passing.
- Cilk 3 Fokussiertes SHM System
- Cilk 4 Spekulative Berechnungen
- Cilk 5 Debugger, Portabilität
- 2006 Kommerzialisierung

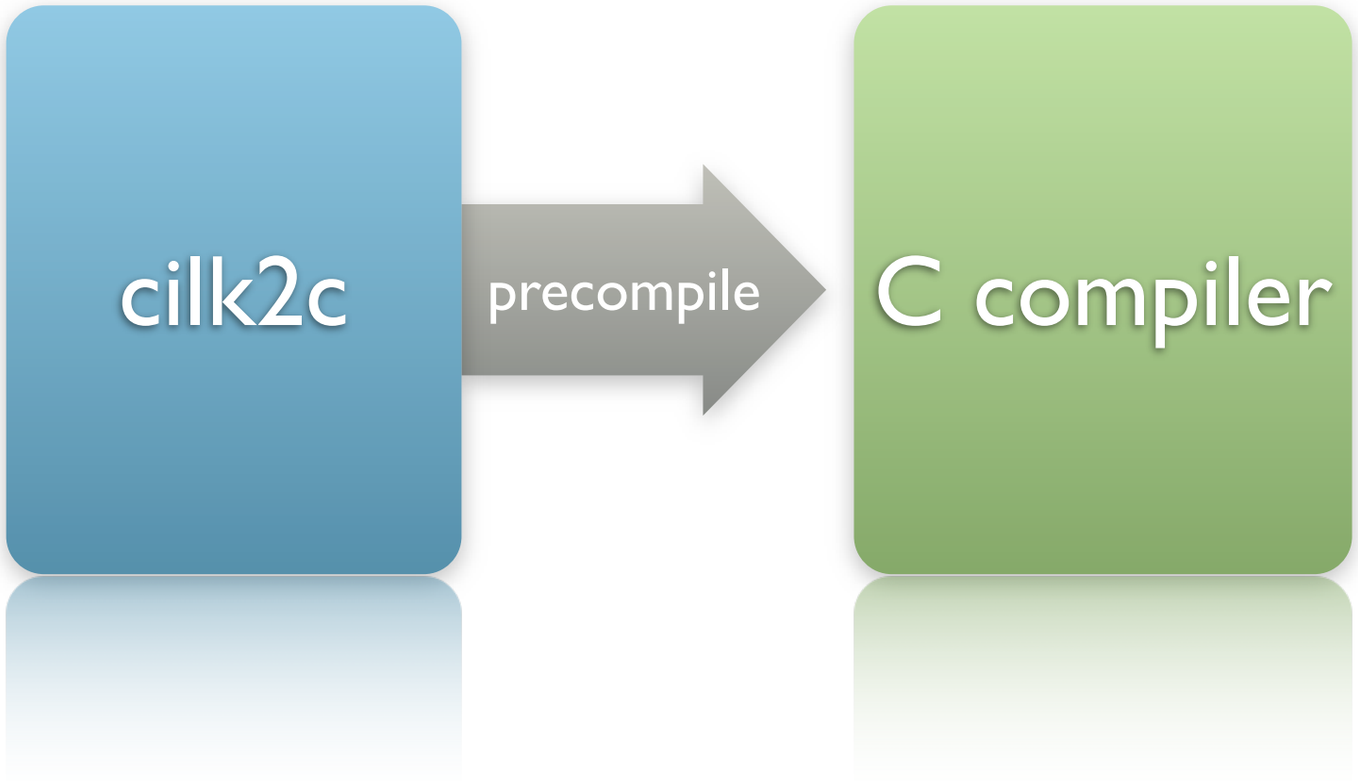
Cilk

+

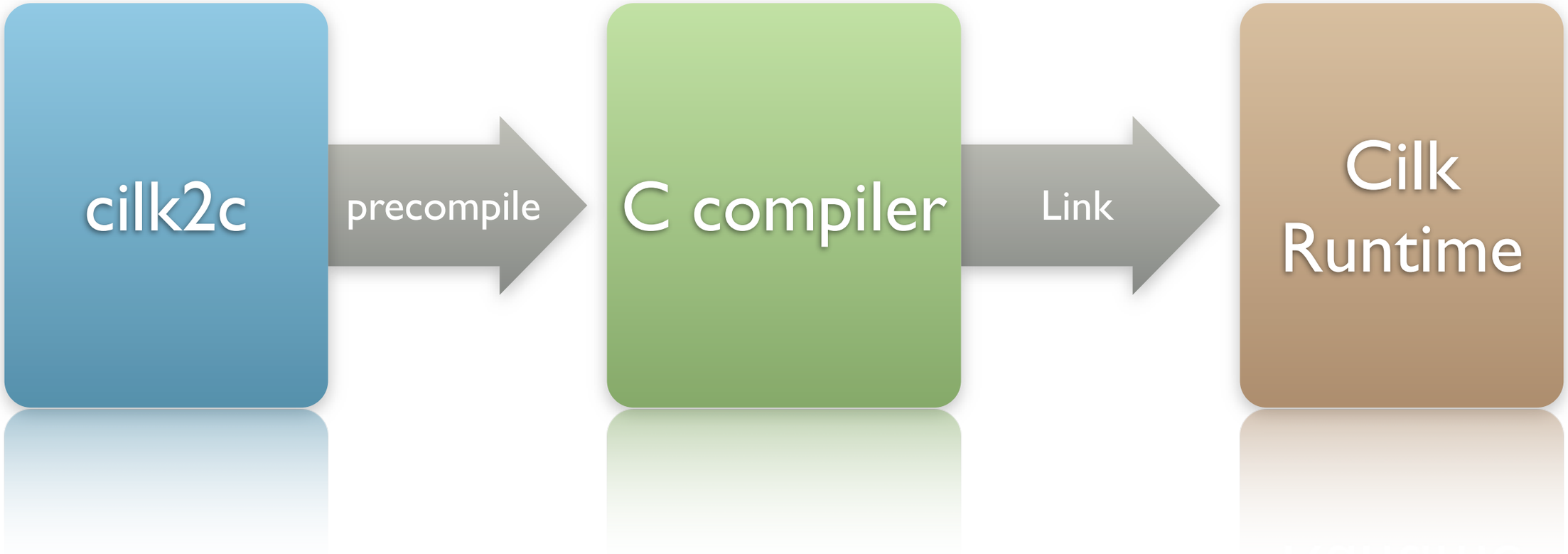
=

Überblick

Überblick



Überblick



Geschichte

- Cilk als reiner Preprozessor
- Erstellt mehrere Varianten
- Runtime System übernimmt Scheduling

Karlsruher Institut für Technologie

Sprache

Parallelisierungsprimitive

```
int fib(int n) {
    int x, y;
    if (n > 2) {
        return n;
    }

    x = fib(n-1);
    y = fib(n-2);
    return (x + y);
}
```

Parallelisierungsprimitive

```
cilk int fib(int n) {
    int x, y;
    if (n > 2) {
        return n;
    }

    x = fib(n-1);
    y = fib(n-2);
    return (x + y);
}
```

Cilk Prozeduren

- Definieren eine parallelisierbare Funktion
- Werden zu einer langsamen und einer schnellen Variante compiliert
- Schnelle Variante
- Langsame Variante
  - Falls der Thread gestohlen wurde.

Parallelisierungsprimitive

```
cilk int fib(int n) {
    int x, y;
    if (n > 2) {
        return n;
    }

    x = spawn fib(n-1);
    y = spawn fib(n-2);
    return (x + y);
}
```

Parallelisierungsprimitive

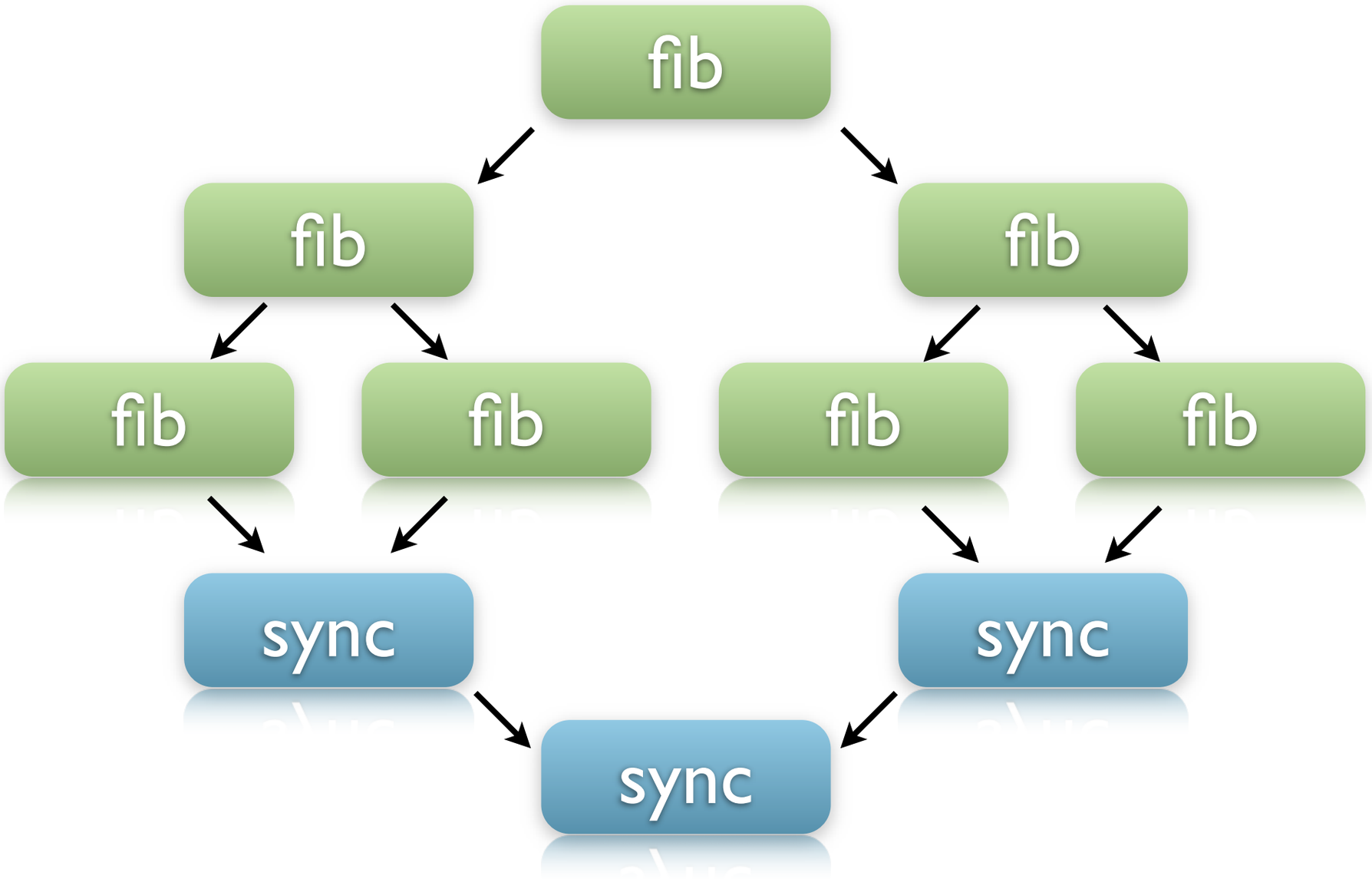
```
cilk int fib(int n) {
    int x, y;
    if (n > 2) {
        return n;
    }

    x = spawn fib(n-1);
    y = spawn fib(n-2);
    sync;
    return (x + y); // impliziter sync
}
```

Synchronisation

- Synchronisationspunkt für alle abhängigen Threads
- Rückgabewerte können verwendet werden
- Return als implizites sync.

Parallelisierungsprimitive



Inlets

```
cilk int fib (int n)
{
    int x = 0;
    inlet void summer (int result) {
        x += result; return;
    }
    if (n<2) {
        return n;
    } else {
        summer(spawn fib (n-1));
        summer(spawn fib (n-2));
        sync;
    }
    return (x);
}
```

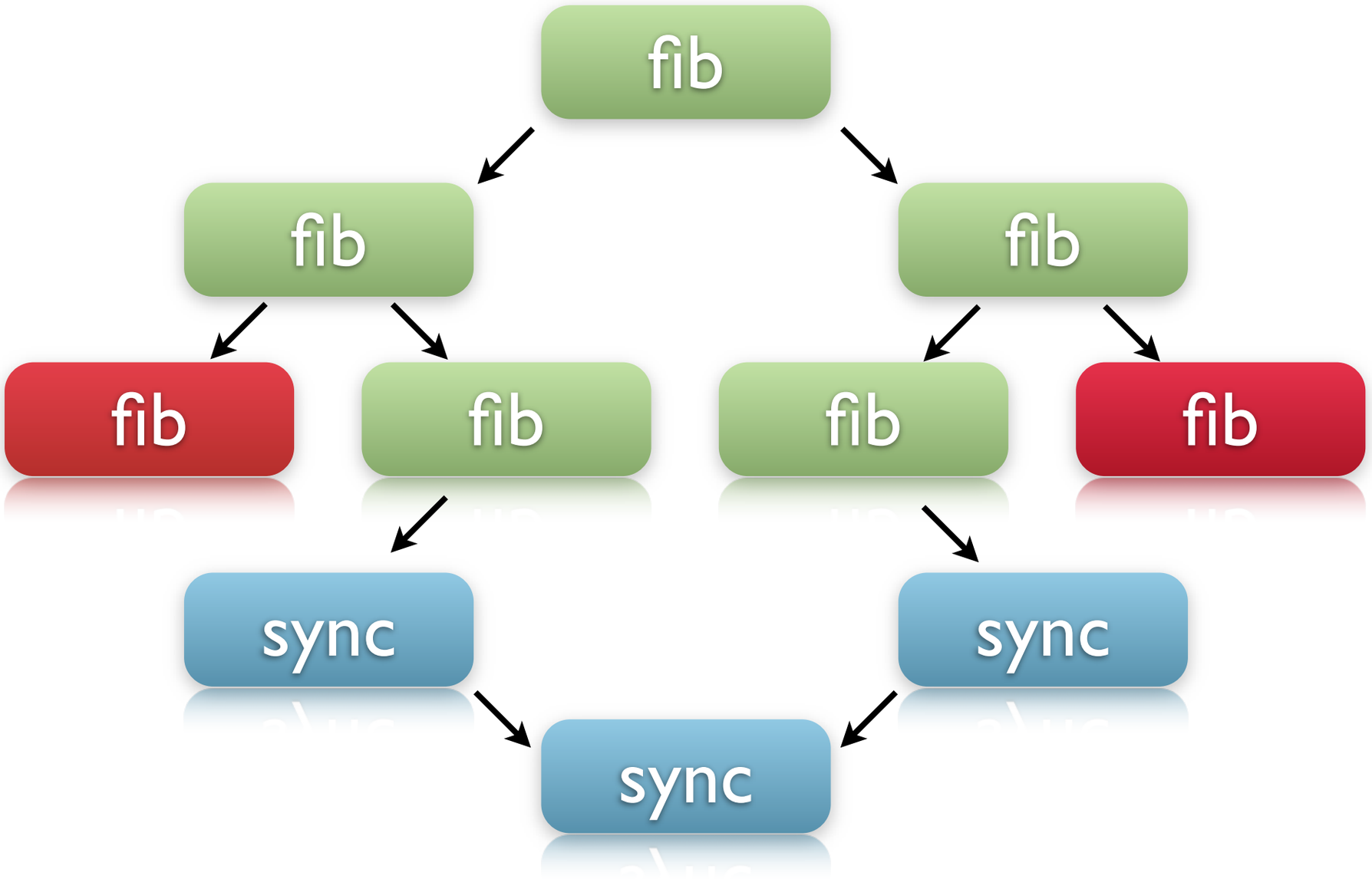
Cilk Prozeduren

- Innere Funktion
- Rückgabewerte von Spawns nutzen ohne sync.

Abort

- Spekulatives Berechnen
- Schachprogramme

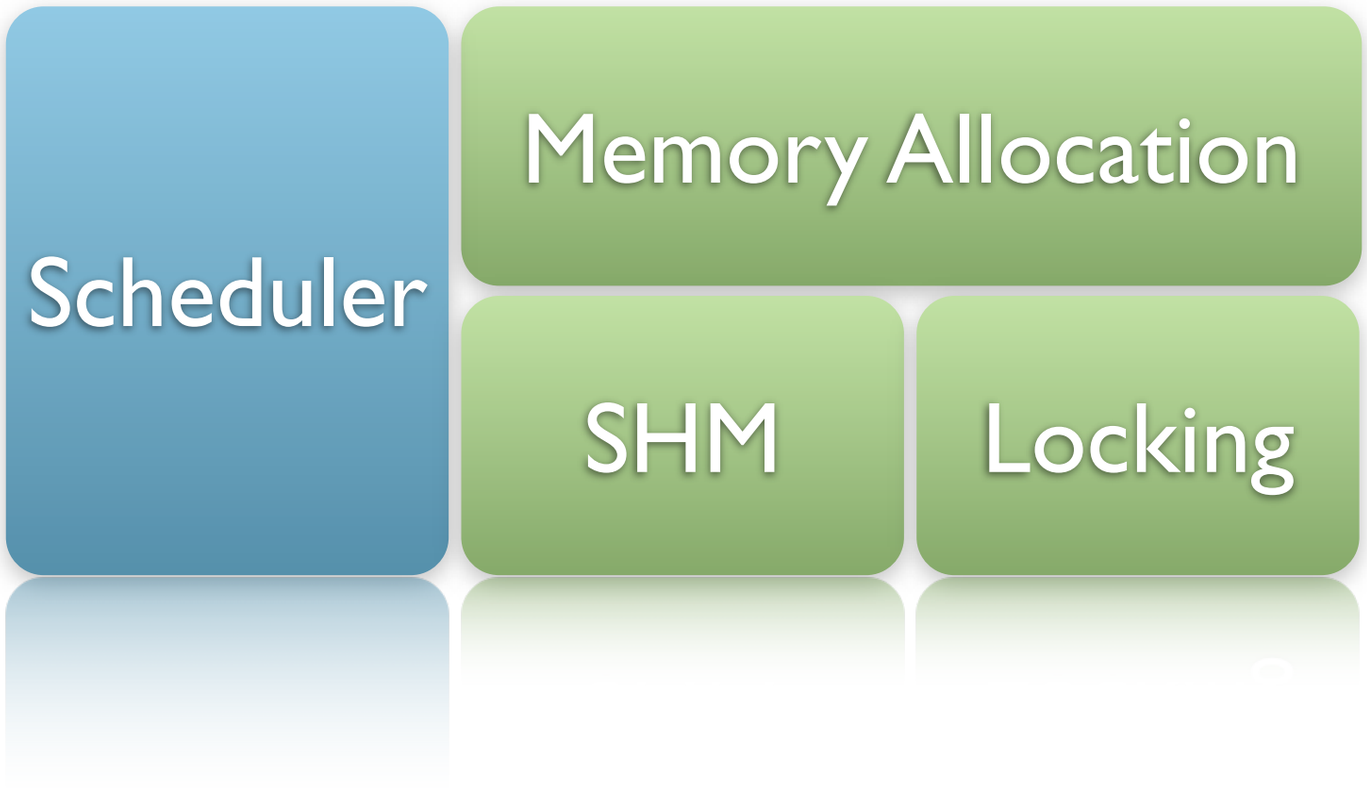
Parallelisierungsprimitive



Karlsruher Institut für Technologie

Runtime

Runtime

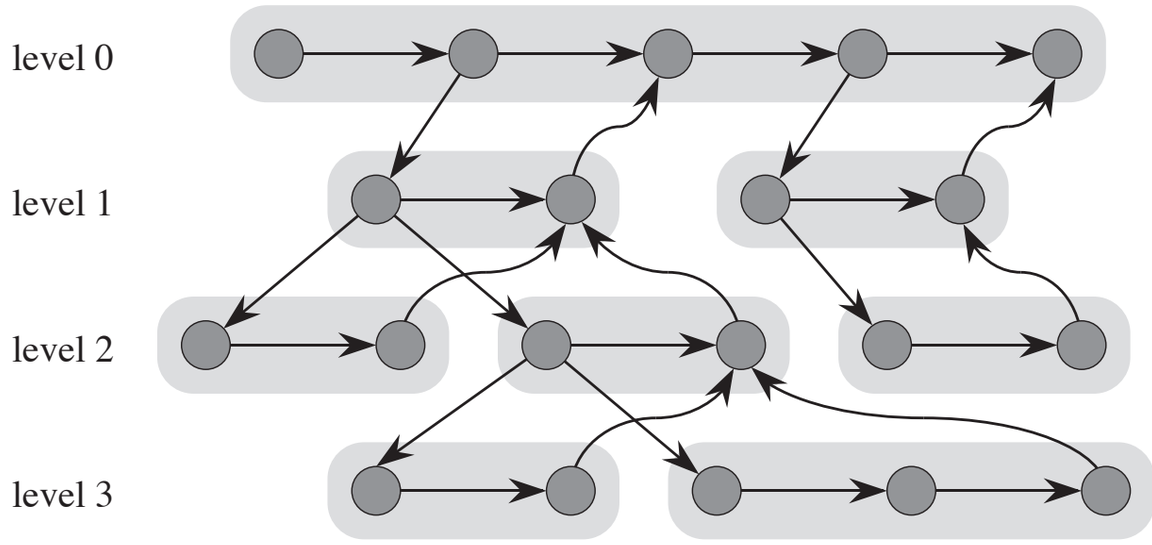


Memory Management

- Private variablen und Rückgabewerte werden synchronisiert
- Globale Variablen über `Cilk_fence` oder `Cilk_lock` parallelisieren.

Randomized Work Stealing

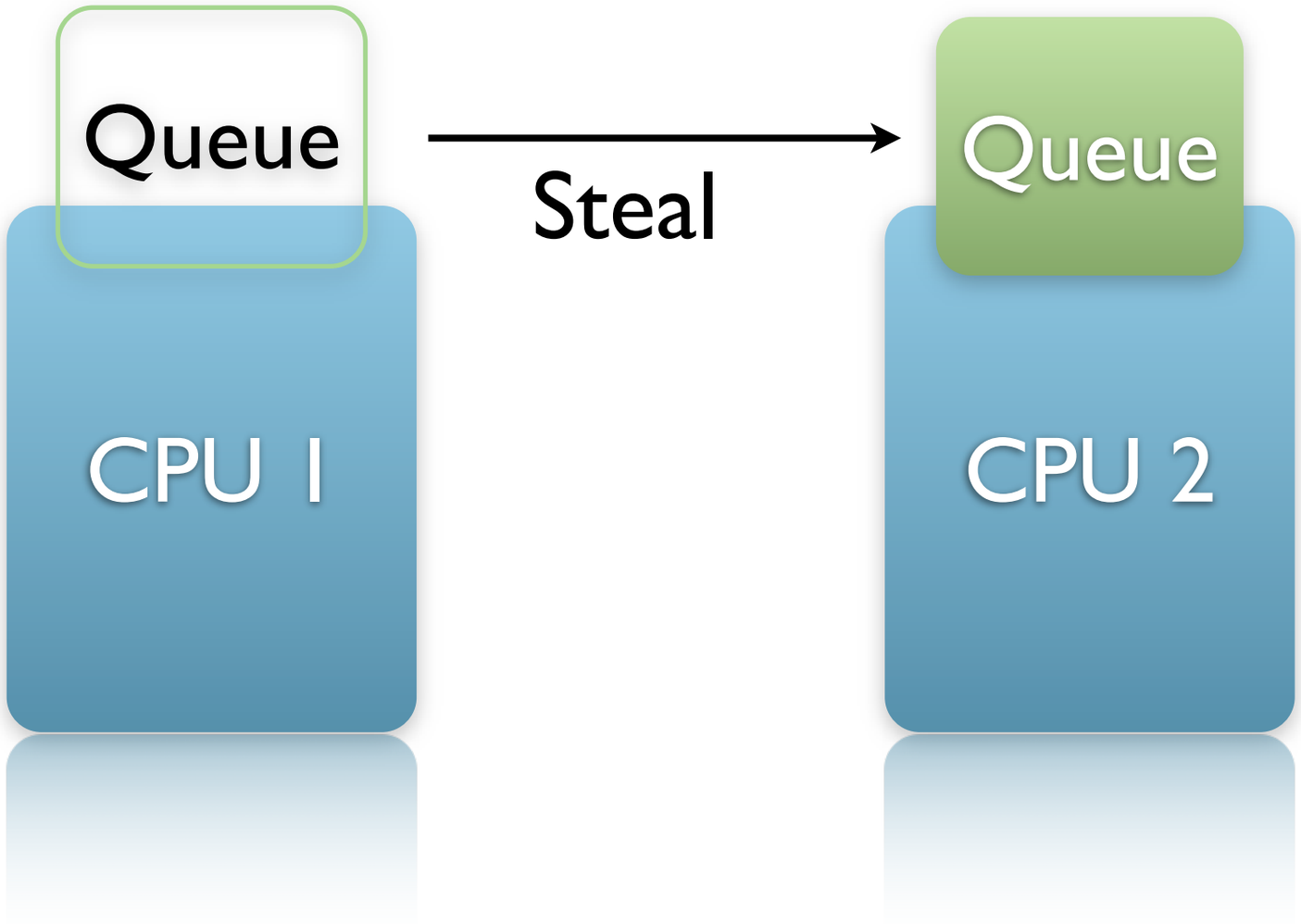
- Geordneter DAG
- 2x Varianten einer Funktion
- Nanoscheduler
- Microscheduler



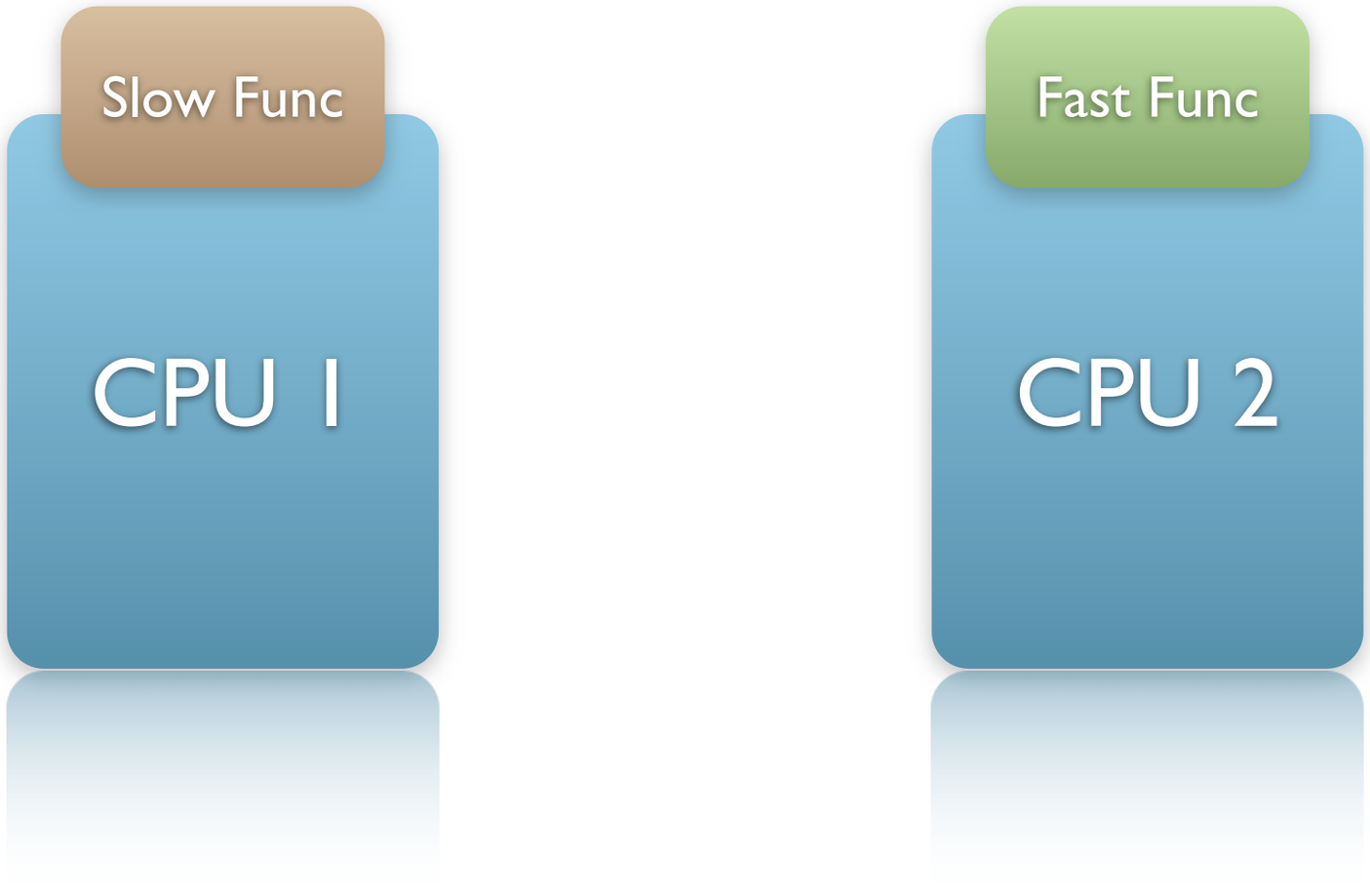
DAG

- Well-structured DAG
  - Effizientes Work Stealing möglich
- Nanoscheduler
  - Scheduled Prozeduren auf einem Prozessor
  - Eincompiliert in das Programm
- Microscheduler
  - Scheduling über eine fixe Anzahl Prozessoren
  - Work Steal Algorithmuis

Stealing



Stealing



Stealing

- Nach dem Steal wird die langsame Variante ausgeführt
- Stellt Datenzugriffe von abhängigen Threads sicher

Erwartete Laufezeit

$$T_P \approx T_1/P + T_\infty$$

Overhead

- Procedure Frame Allokation
- Sicherung des State vor jedem Spawn
- Frame Check nach jedem Spawn
- Procedure Frame Freigabe
- Aber gering in der Praxis

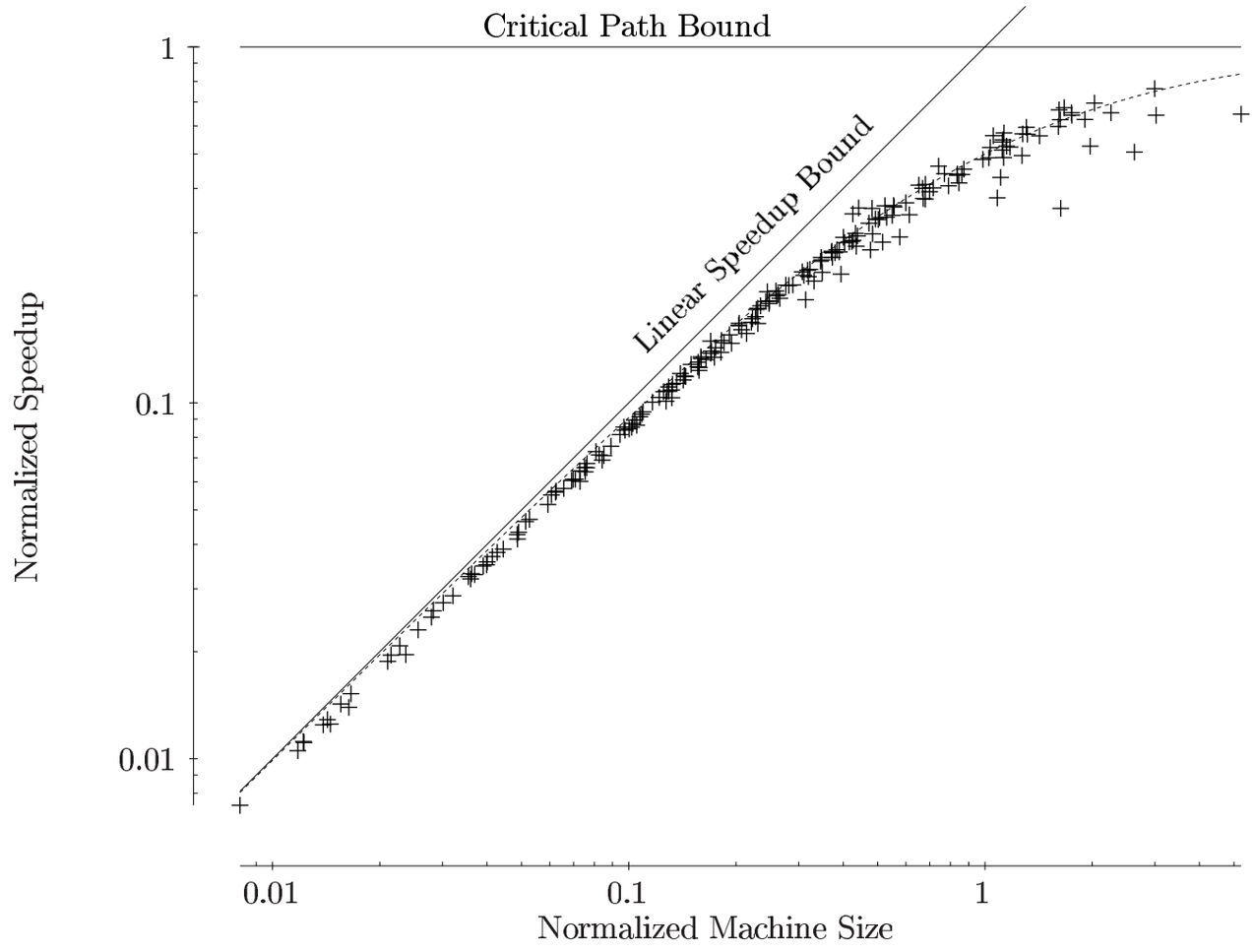
Karlsruher Institut für Technologie

Beispiele

Beispiele

- Fast lineare Skalierung
  - Schachprogramm Sokrates
- Abhängig von der Parallelisierung

$$\overline{P} = T_1/T_\infty$$



Karlsruher Institut für Technologie

Fazit/Einsatz

Fazit

- Sprache
  - Basierend auf C
    - Wenige Spracherweiterungen
    - Work Steal Algorithmus
- Einsatz
  - Akademisch

Literatur

- D. Daily, C. E. Leiserson, *Using Cilk to Write Multiprocessor Chess Programs*, MIT Laboratory of Computer Science, 2001

- C. E. Leiserson, *Cilk 5.4.6 Reference Manual*, MIT Laboratory of Computer Science, 1998

- M. Frigo, K. H. Randall, C. E. Leiserson, *The Implementation of the Cilk-5 Multithreaded Language*, MIT Laboratory of Computer Science, 1998

- C. F. Joerg, *The Cilk System for Parallel Multithreaded Computing*, MIT Department of Electrical Engineering and Computer Science, 1996