Juggle: Eine verteilte virtuelle Maschine für Java

Michael Schröder
Franz J. Hauck

IMMD IV, Lehrstuhl für Betriebssysteme
Universität Erlangen-Nürnberg

1 Inhalt



- Motivation
- Eigenschaften von Juggle
- Objektmigration & Objektreplikation
- Threadmigration
- Speedup-Messungen
- Zusammenfassung

2 Motivation

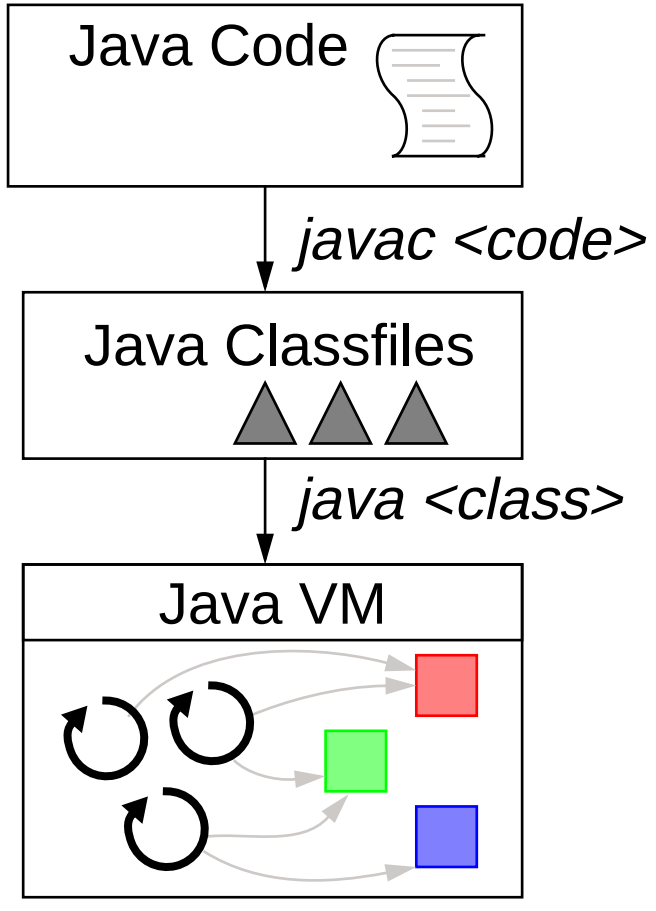


- Problem:
  - Anwender wollen (oder können) ihre High-Performance Applikationen nicht verteilt programmieren.
  - Sie interessieren sich nicht für den maximal erreichbaren Speedup.
  - Eine statische Konfiguration ist häufig nicht ausreichend.
- Lösung:
  - Juggle: Verteilte JVM, transparent für den Anwender.
  - Automatische Lastverteilung durch Objektmigration bzw. Replikation und Threadmigration.
  - *nicht*: automatische Parallelisierung.

3 Motivation



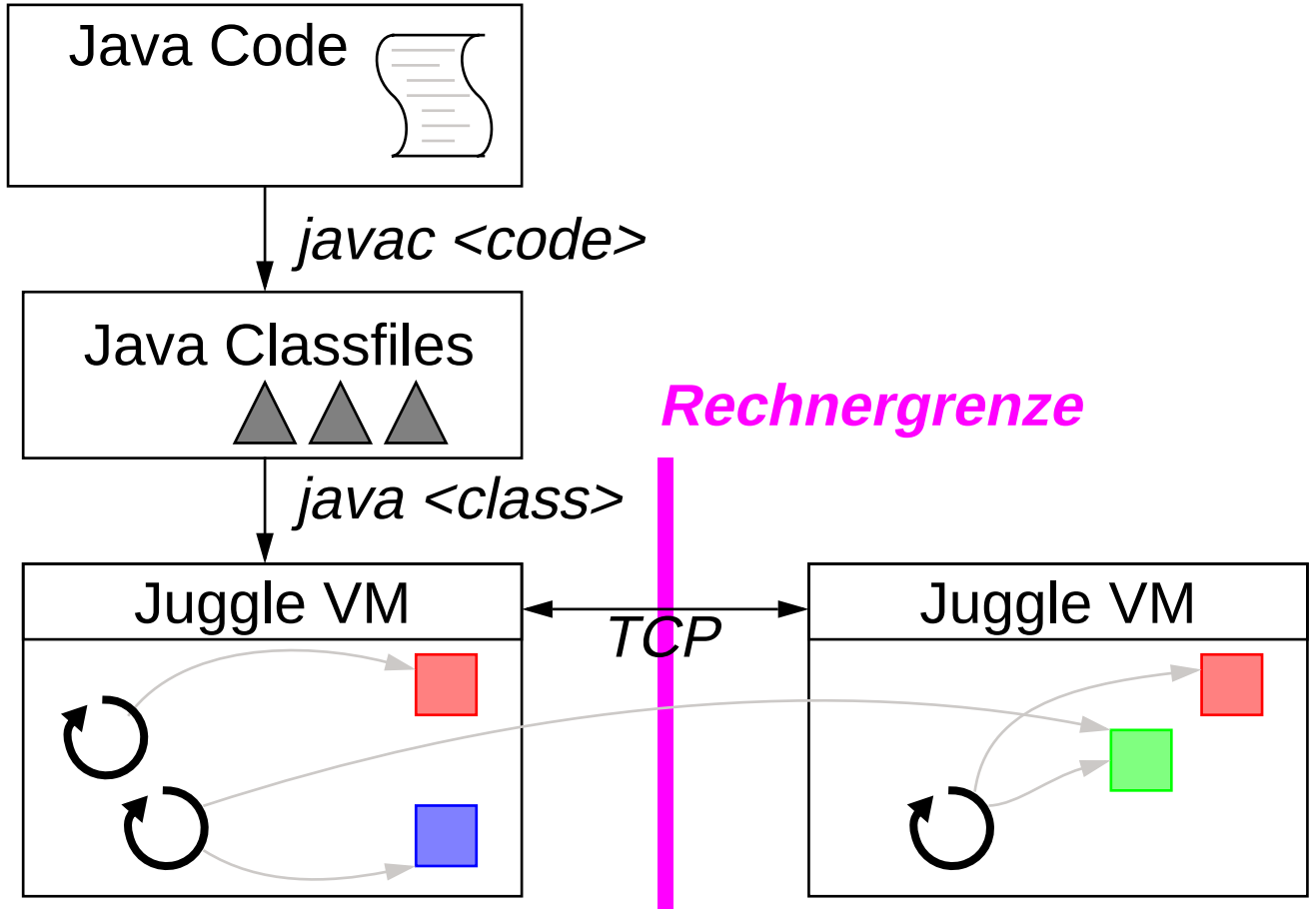
- Programmausführung mit Java:



4 Motivation



- Programmausführung mit Juggle



5 Eigenschaften von Juggle

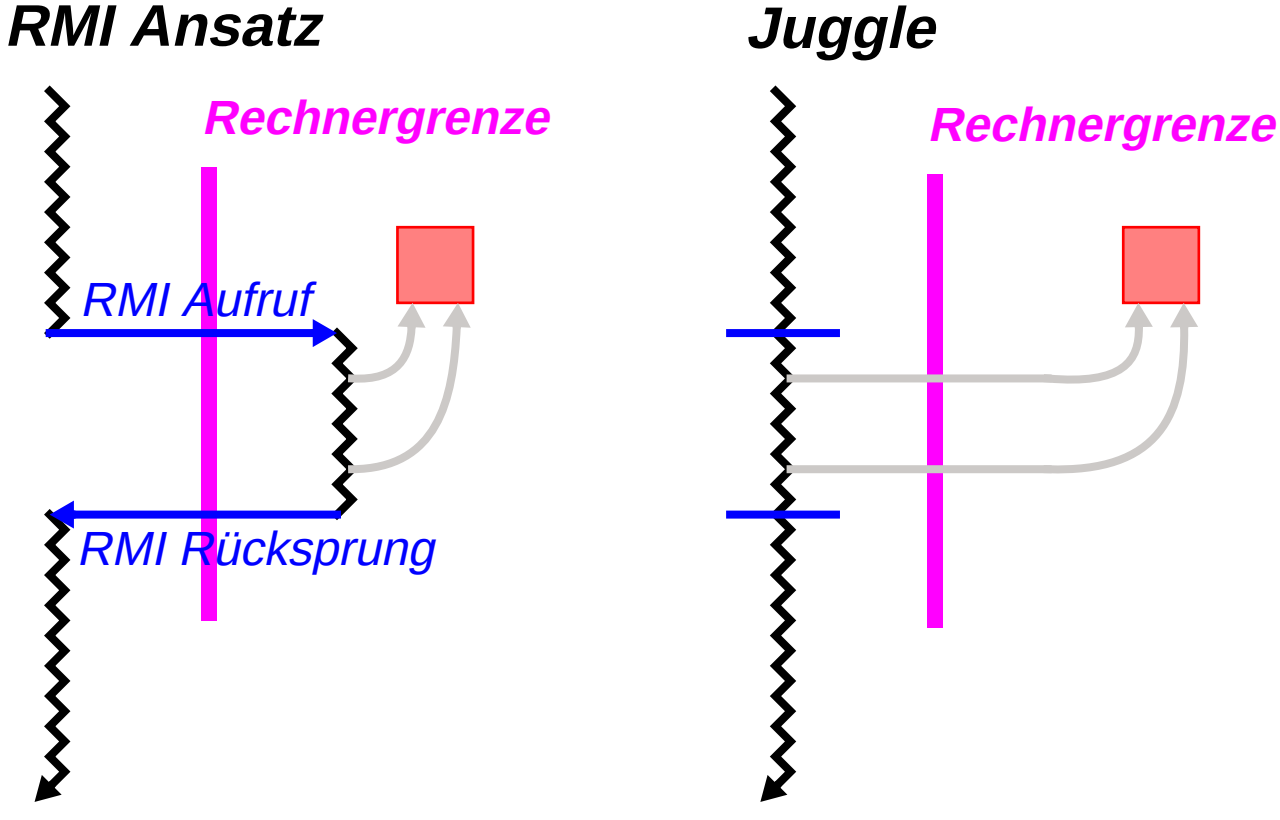


- Vorteile des JVM Ansatzes:
  - ▲ Jedes vorhandene Java Programm kann ohne Änderung über mehreren Rechnern verteilt ablaufen.
  - ▲ Es muß kein Quellcode zur Verfügung stehen.
- Nachteile:
  - ▲ Juggle erreicht nicht die Ergebnisse einer manuellen Verteilung.
  - ▲ JVM Entwicklungen müssen nachprogrammiert werden.

6 Besonderheiten von Juggle



- Juggle verwendet keinen “RMI” Ansatz, sondern nur der Zugriff auf Instanzvariablen eines entfernten Objektes führt zu einem Nachrichtenaustausch.



7 Besonderheiten von Juggle



- Folgende Bytecodes können Interaktionen mit anderen Knoten bewirken:
  - **getfield**, **getstatic**, **putfield**, **putstatic**
  - **?aload**, **?astore**
  - **monitorenter**, **monitorexit**,
    Aufruf von / Rücksprung aus **synchronized** Methoden
- Zusätzlich dazu gibt es Interaktionen durch E/A Operationen und einigen *native*-Methoden (z.B. **arraycopy**).

8 Aufruf



- Auf dem “Master”-Knoten:

  `jug -m #slaves classname <args>`

  startet Juggle im Master/Slave Modus.

- Aus den “Slave”-Knoten

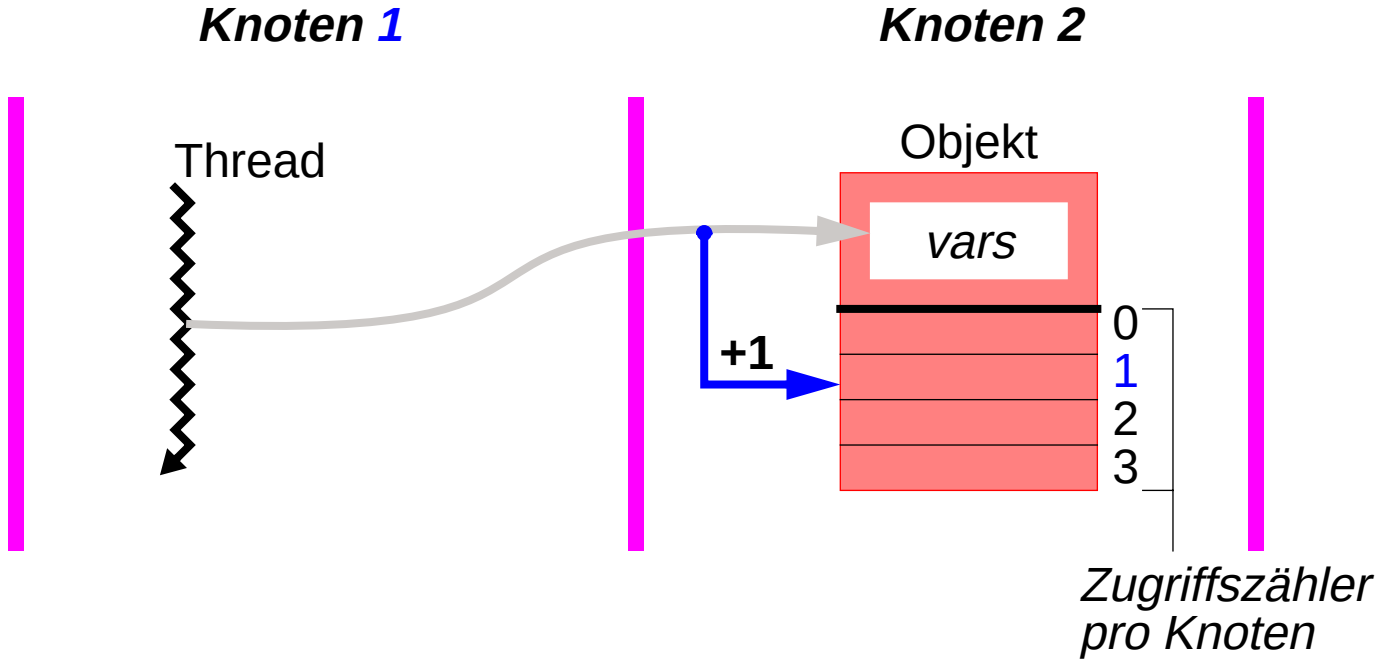
  `jug -c masterhost`

- Sämtliche E/A Operationen, also auch das Laden von Klassen, werden auf dem Master-Knoten durchgeführt.

9 Objektmigration



- Über Objektmigrationen versucht Juggle, die Zugriffe auf andere Knoten möglichst gering zu halten.
- Ermittlung der idealen Knotenposition nach einstellbarer Anzahl von Zugriffen.



10 Objektreplikation

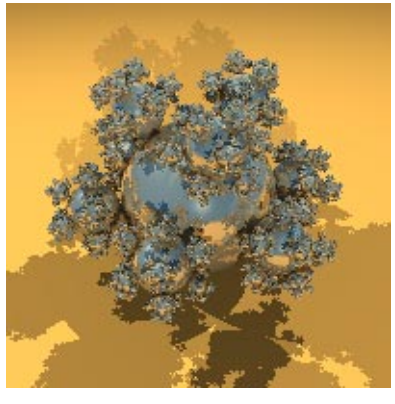


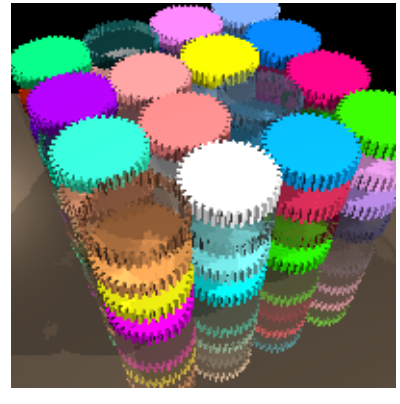
- Objektreplikation ist das Anlegen von Objektkopien.
  Einsatz in Juggle anstelle von Migration.
- Wurde auf ein Objekt überwiegend lesend zugegriffen, ist es für Replikationen freigegeben.
- Implementierung über einen weiteren Zähler, der periodisch erhöht und bei Schreibzugriffen zurückgesetzt wird.
- Bei einem Schreibzugriff auf ein repliziertes Objekt werden zuerst die Replikas vernichtet.

11 Meßwerte



- Einfacher Raytracer
  - ▲ 23 Klassen (Geom, Shader, Light, Camera, Frame, ...)
  - ▲ ca. 1200 Lines of Code
  - ▲ Lichtbrechung, Spiegeleffekte, Schatten
  - ▲ Testbilder aus Eric Haines SPD Paket:



12 Meßwerte



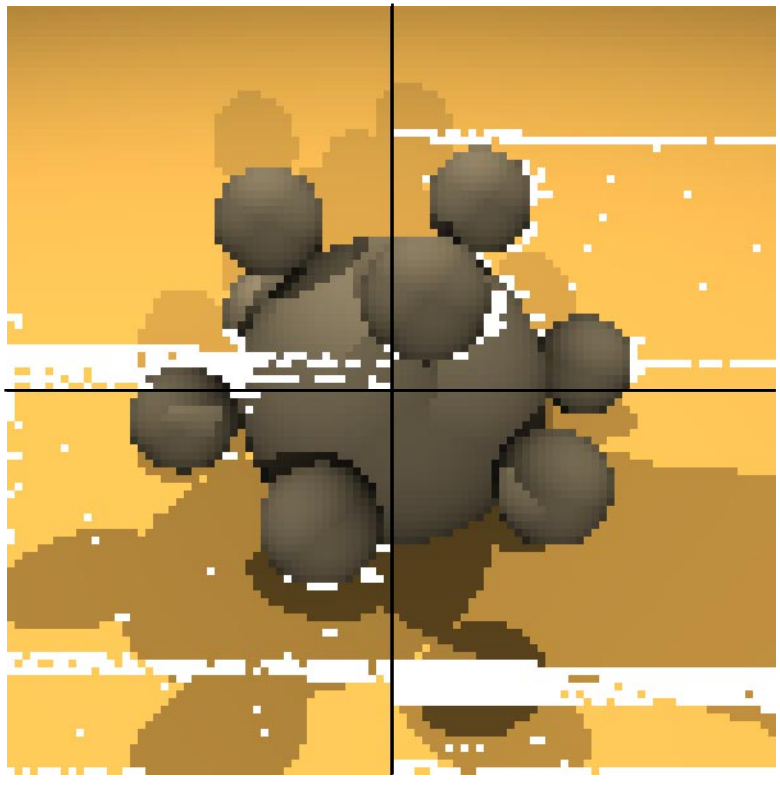
- Ergebnisse für die Balls Szene mit 9 Kugeln und 4 Rechenknoten:



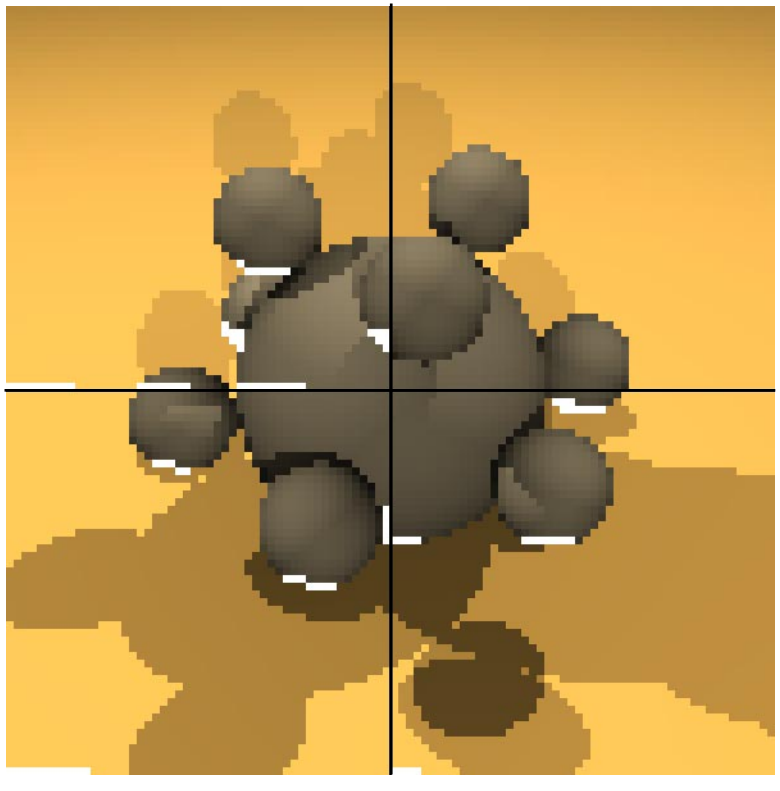
	feste Positionen	mit Migrationen	Migrationen und Replikationen
Objekte	101331		
verteilte Objekte	188		
Migrationen		3302	54
Replikationen			251
lokale Zugriffe	$6,9 \cdot 10^6$	$12 \cdot 10^6$	$12,4 \cdot 10^6$
Fernzugriffe	$5,5 \cdot 10^6$	$0,41 \cdot 10^6$	$0,02 \cdot 10^6$

13 Meßwerte

- Wirkung der Migrationen und Replikationen:



nur Migrationen

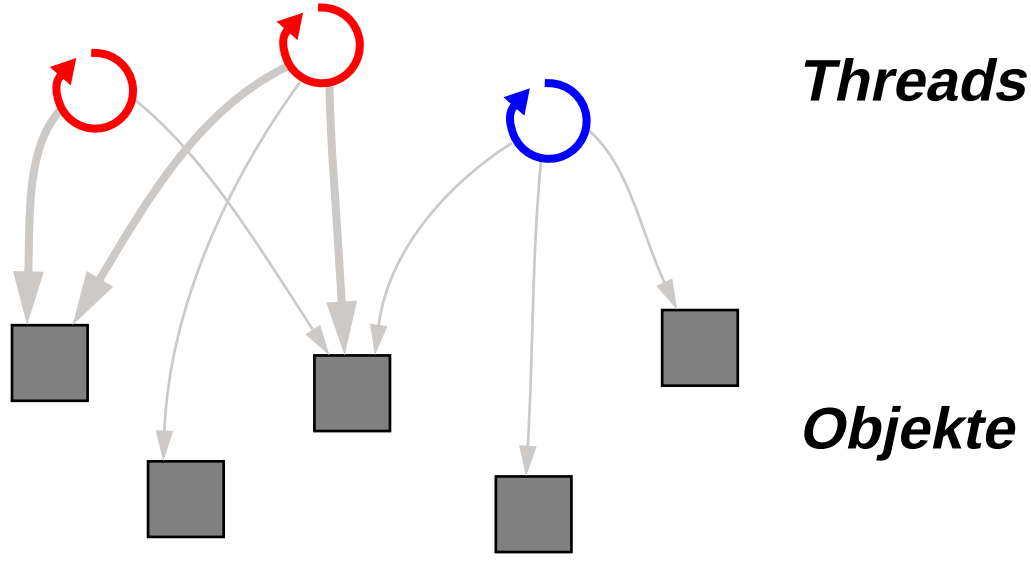


Migrationen & Replikationen

14 Threadmigration



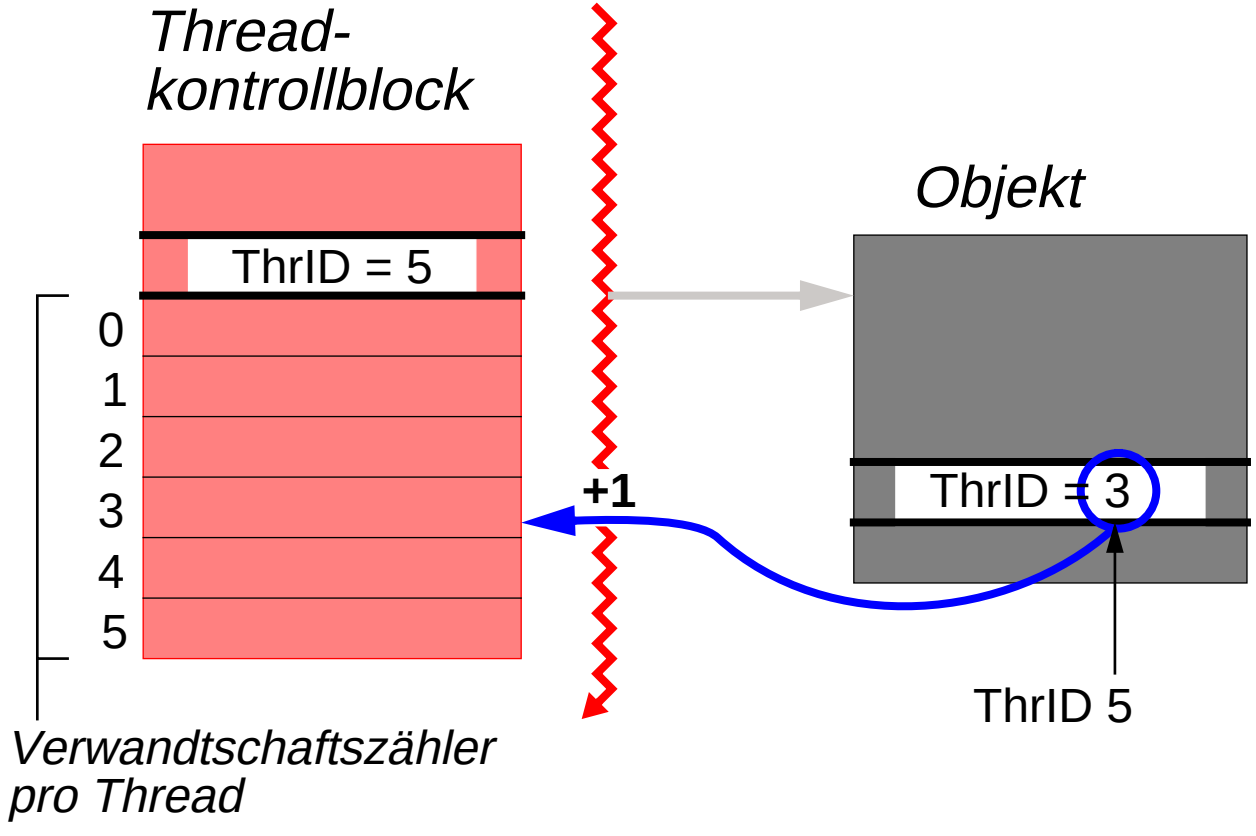
- Bewirkt eine gleichmäßige Auslastung der Rechenknoten
  - ▲ Umsetzung über periodische Ermittlung der Knotenlast und Umverteilung der arbeitenden Threads.
  - ▲ Einbeziehung der Threadverwandtschaft:



15 Threadmigration

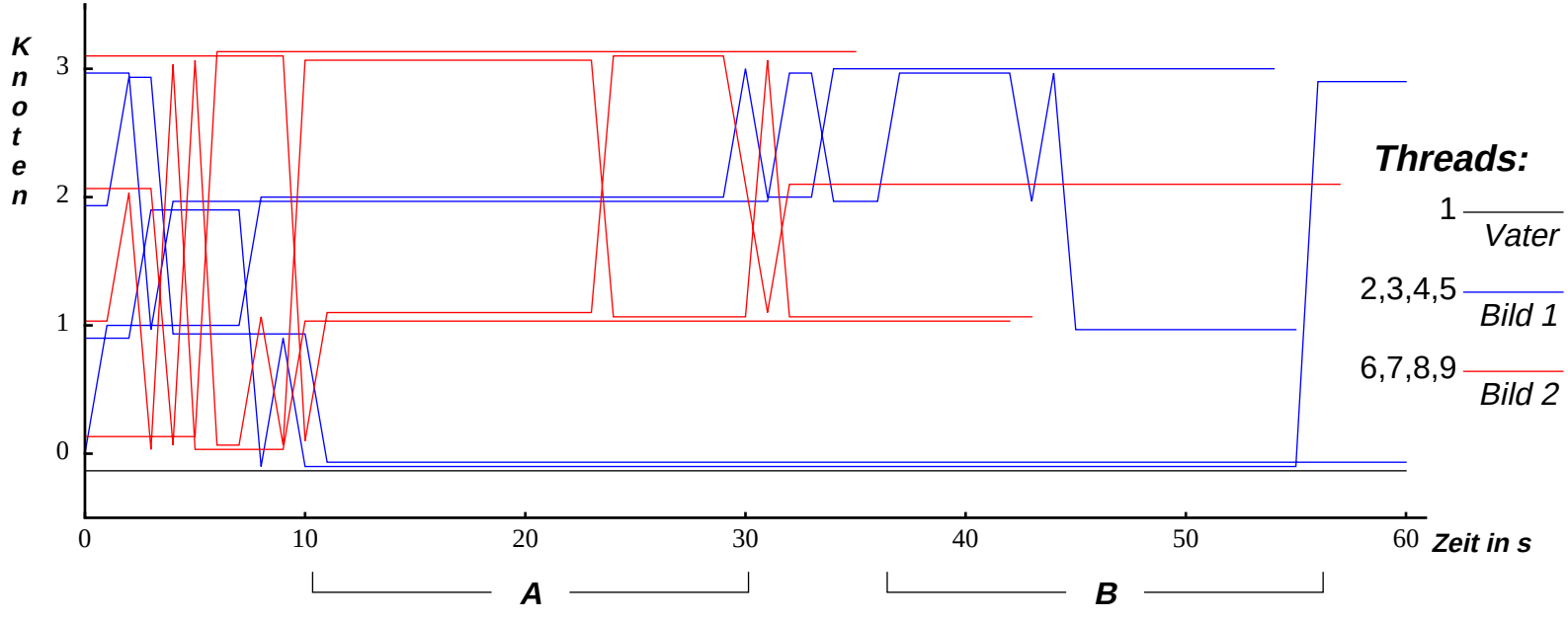


- Einfacher Samplingalgorithmus zur Feststellung von Verwandtschaftsbeziehungen zwischen den Threads:



16 Meßwerte

- Gleichzeitige Berechnung zweier Bilder mit jeweils 4 Threads, Darstellung der Threadposition über der Zeit:



- Automatische Klassifizierung der Threads je nach berechnetem Bild bei gleichmäßiger Auslastung der Knoten.

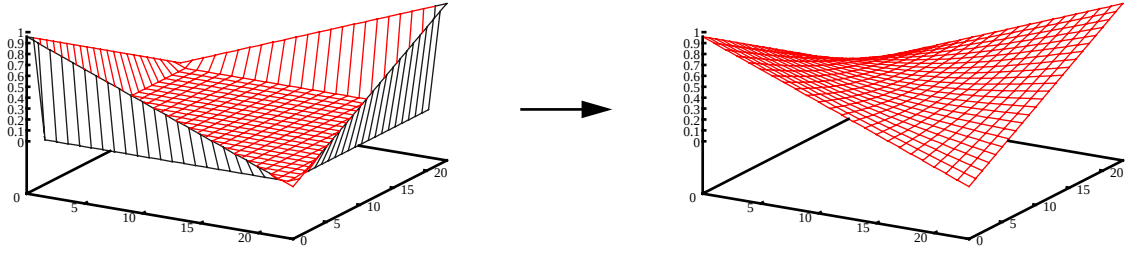
17 Speedup-Messungen



- Komplexe Balls-Szene (91 reflektierende Kugeln), 4 Threads:

Anzahl der Knoten	1	2	4
Dauer der Berechnung	120s	75s	56s
Speedup	–	1,6	2,14

- Einfacher Glättungsalgorithmus auf einem Gitter, 8 Threads:



Anzahl der Knoten	1	2	4	8
Dauer der Berechnung	520s	341s	208s	143s
Speedup	–	1,53	2,5	3,64

18 Zusammenfassung



- Juggle bietet die Abstraktion einer verteilten virtuellen Maschine.
- Programme können ohne Codeänderung über mehrere Rechner hinweg verteilt ablaufen.
- Juggle sorgt durch Objektmigrationen/-replikationen und Threadmigrationen für einen effizienten Ablauf des Programms.
- Erweiterung von “*write once, run everywhere*” um die Dimension der Rechnercluster.