

Semesterarbeit

Concurrent Garbage Collector

Eingereicht bei: Tomas Pospisek

Eingereicht von: Daren Thomas

Inhalt

- ☐ Aufgabenstellung
 - ☐ Vorbedingungen
 - ☐ Lösungsschritte
 - ☐ Resultate
 - ☐ Weiterführende Arbeiten
-

Aufgabenstellung

□ Hauptziel

- Implementation des Concurrent Garbage Collectors von Dijkstra

□ Sekundärziel

- Vergleich mit TwoSpace Algorithmus
-

Vorbedingungen

☐ Runtime

- Funktionsweise bekannt
- Anpassbar an spezielle Bedürfnisse

☐ Testproblem

- alloziert Speicher
 - möglichst konfigurierbar
-

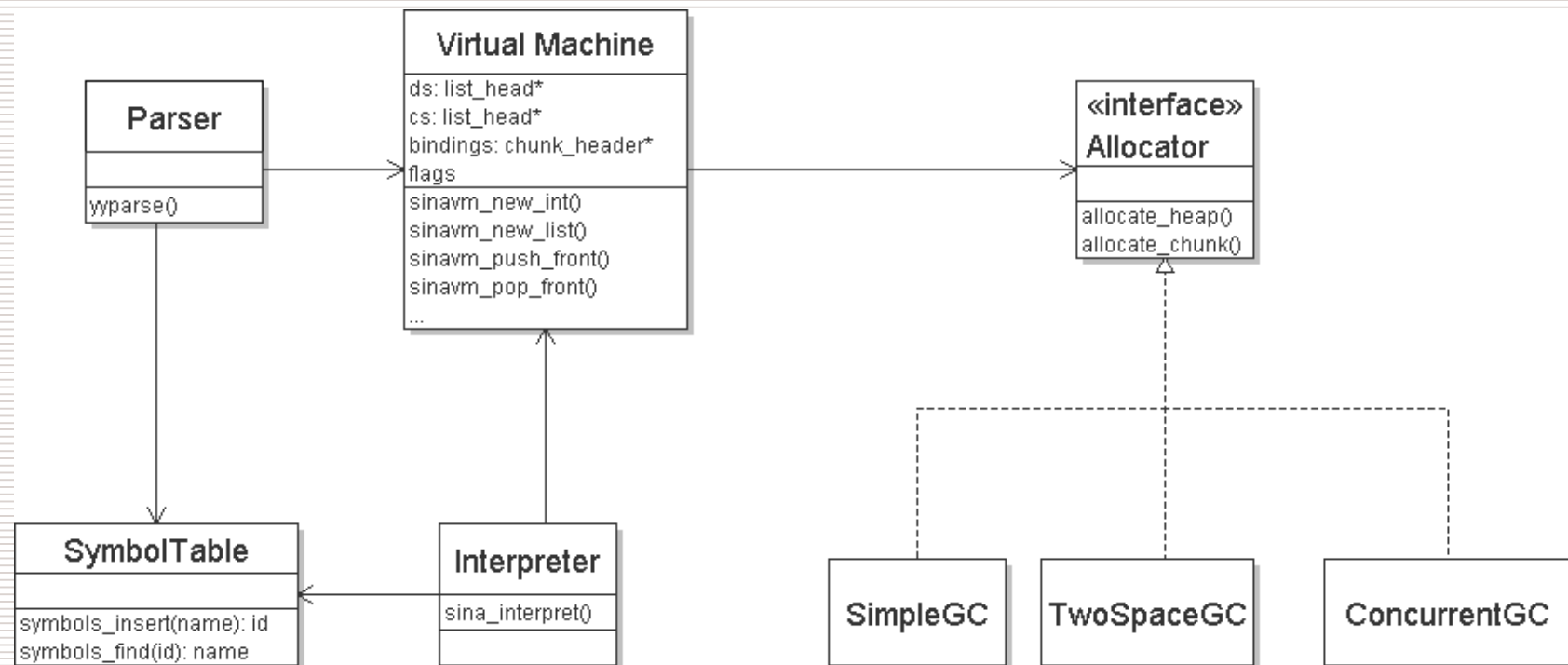
Lösungsschritte

- ☐ Definition Syntax / Semantik von Sina
 - ☐ Testalgorithmus entwerfen
 - ☐ Entwicklung des Parsers
 - ☐ Implementation des Interpreters
-

Lösungsschritte

- ☐ Implementation von TwoSpace
 - ☐ Implementation des Concurrent Garbage Collectors
 - ☐ Vergleich TwoSpace vs. Concurrent
-

Architektur



Vergleich der Algorithmen

□ Testmaschinen

- Sony Vaio mit Windows XP und Intel Pentium Mobile Prozessor (1.19 GHz) und 512 MB RAM
 - MacBook mit OS X und Intel Core 2 Duo (je 1.8 GHz) und 1 GB RAM
-

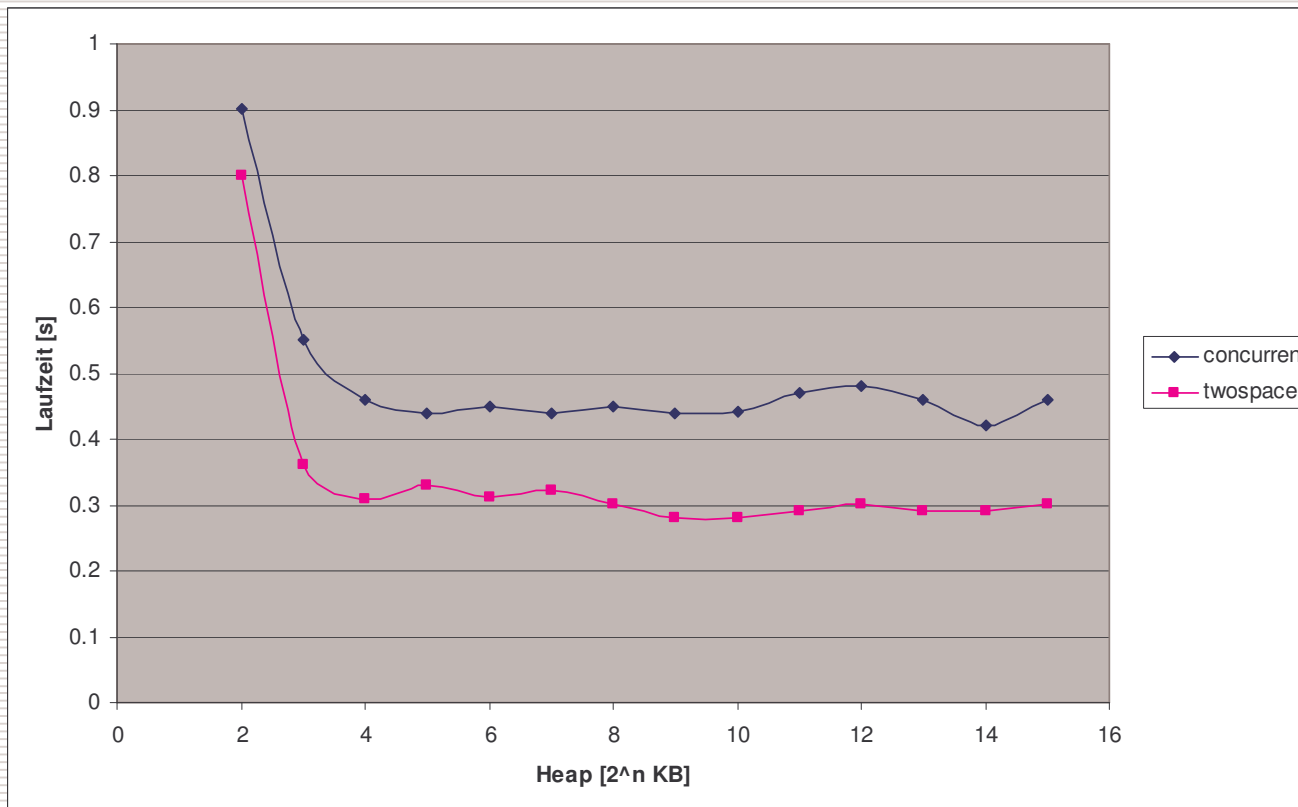
Vergleich der Algorithmen

□ Testreihe

- 500 / 1000 Zeilen
 - 10 / 100 / 500 Zeichen
 - Heapgrösse = 2^n KB für $1 \leq n \leq 15$
-

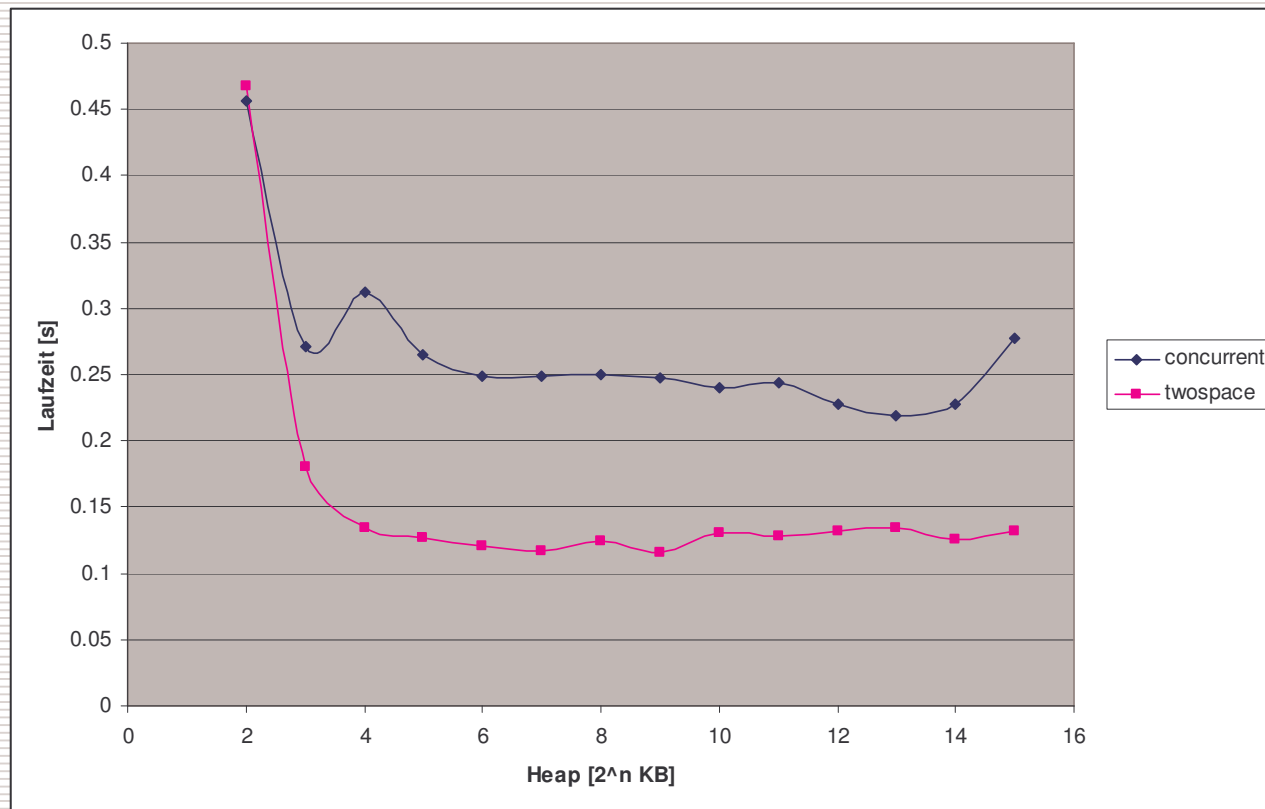
Vergleich der Algorithmen

□ 1000 Zeilen à 10 Zeichen (1 CPU)



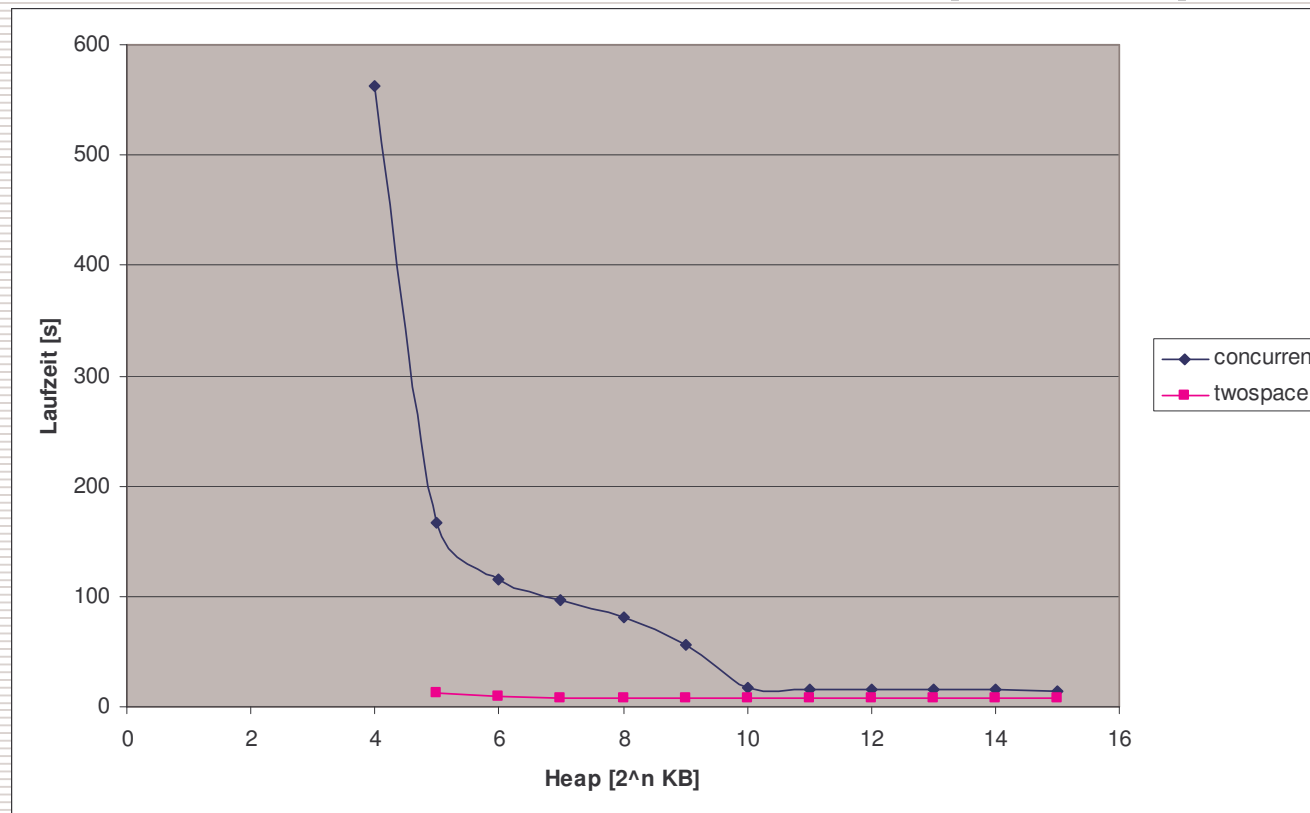
Vergleich der Algorithmen

□ 1000 Zeilen à 10 Zeichen (2 CPU)



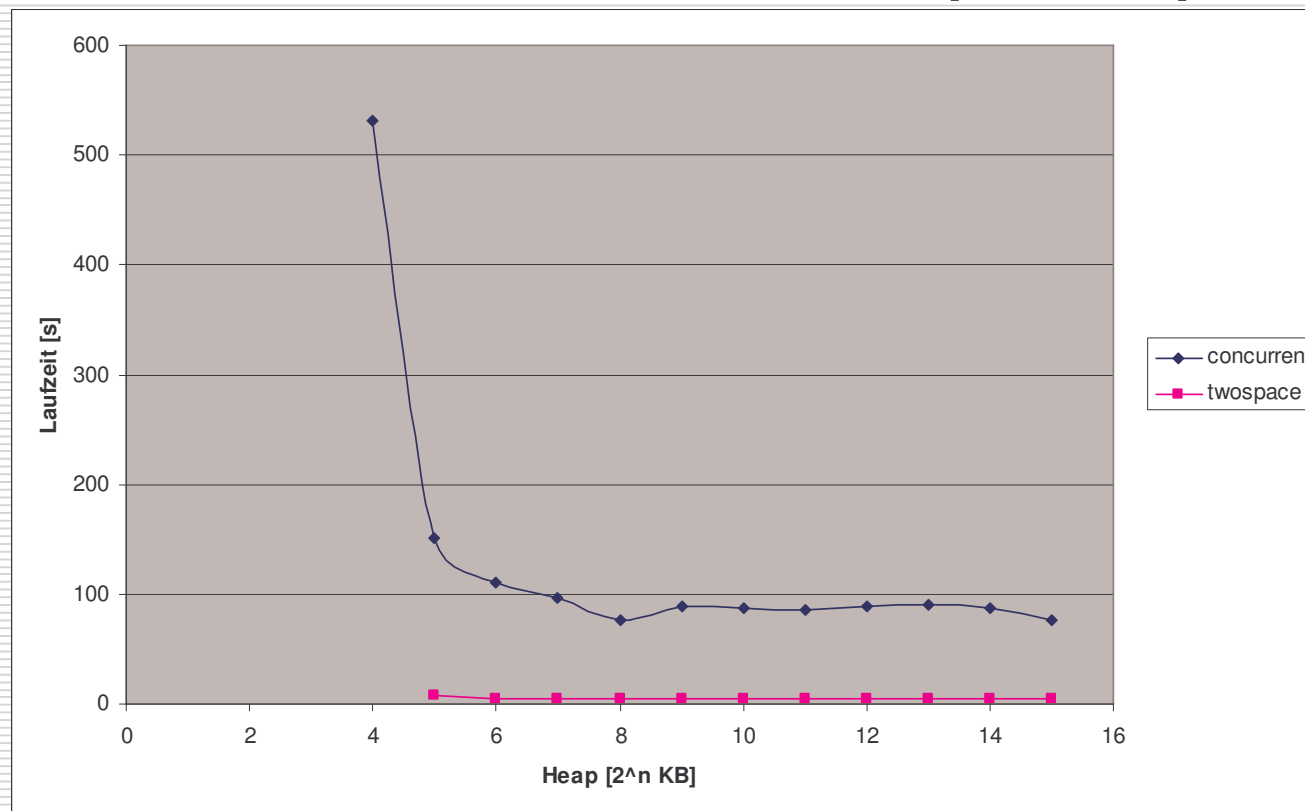
Vergleich der Algorithmen

□ 1000 Zeilen à 500 Zeichen (1 CPU)



Vergleich der Algorithmen

□ 1000 Zeilen à 500 Zeichen (2 CPU)



Resultate

- ❑ TwoSpace Algorithmus ist mindestens doppelt so schnell
 - ❑ und einfacher
 - bessere Wartbarkeit
 - ❑ Nebenläufigkeit und Prozessorarchitektur
 - Datenzugriffe möglichst trennen
-

Weiterführende Arbeiten

- ☐ Portierung auf andere Architekturen
 - ☐ Implementation von Multithreading in Sina
 - ☐ Optimierung der Garbage Collectors
-