Fallstudie Wortliste

Michael Mandl

25.03.2024

Aufgabe

Entwicklung einer parallelen Suche in einer Liste mit Wörtern

Es ist eine Datenstruktur in Form einer Liste (Vektor) mit unterschiedlichen Wörtern gegeben. Ein Suchalgorithmus soll eine Liste mit Wörtern aus dieser Wortliste ermitteln, deren erste Zeichen mit einem Suchstring übereinstimmen.
Der Algorithmus soll moderne Multi-Core-Prozessoren ausnutzen, die Suche also auf möglichst viele Cores verteilen.

Anforderungen

- Datenstruktur für Wortliste
- Paralleler (schneller) Such-Algorithmus
- Test-Applikation
- Konzeptuelle Erweiterung: Inkrementelle Suche

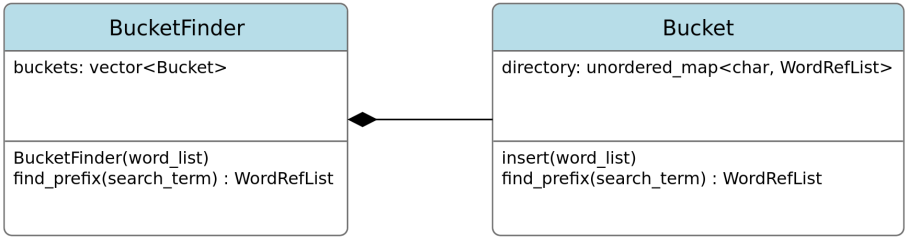
Umsetzung

- Algorithmus in zwei Phasen: Wortliste einlesen und Übereinstimmungen suchen
- Such-Phase zeitkritischer als Einlese-Phase
- Datenstruktur, die Locks minimiert

Aufbau

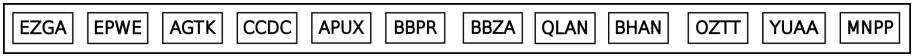
- Flacher Suchbaum
- Ebene 1: Ein Knoten je Thread
- Ebene 2: Ein Blatt je erster Wort-Buchstabe
- Blätter tragen Wort-Listen

Datenstrukturen - Implementierung



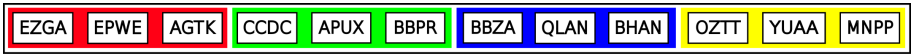
- buckets: Erste Baum-Ebene, nach Threads
- Bucket: Zweite Baum-Ebene, nach erstem Buchstaben
- WordRefList: Wort-Liste am Blatt

Algorithmus: Einfügen von Daten



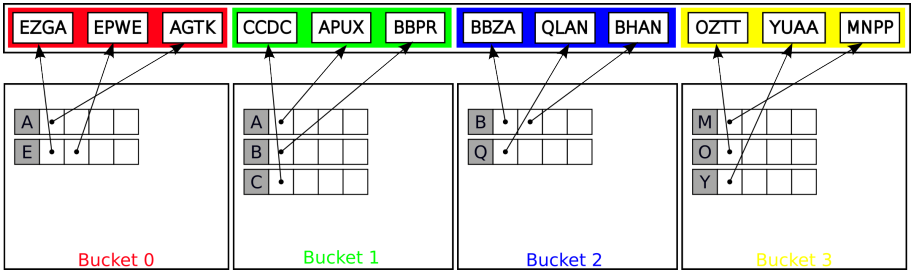
Eingabedaten

Algorithmus: Einfügen von Daten



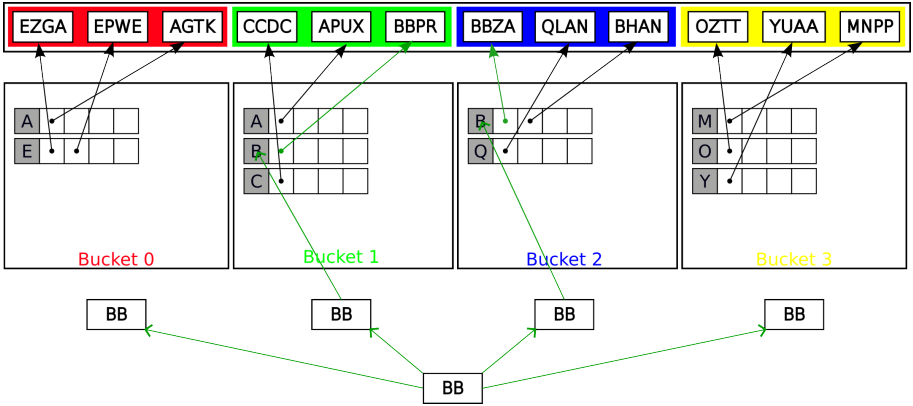
Aufteilen in Buckets

Algorithmus: Einfügen von Daten



Mapping nach erstem Buchstaben

Algorithmus: Suchen von Wörtern



Suche nach "BB"

Komplexitäts-Abschätzung

BucketFinder

- Suchraum einfügen: O(n) mit $n = |Suchraum|$
- Pattern suchen: O(n) mit $n = \frac{|Suchraum|}{|Threads| \times |Alphabet|}$
- Laufzeit: deutlich schnelleres Einfügen und schelleres Finden von größeren Ergebnismengen

Balancierter Baum

- Suchraum einfügen: O(n log n)
- Pattern suchen: O(log n)
- Laufzeit: deutlich schneller beim Finden kleinerer Ergebnismengen

Erweiterung inkrementelle Suche

- Such-Phase noch zeitkritischer
- Such-Ergebnis als gefiltertes Such-Objekt
- UI: dynamisch gefüllte Combo-Box
- Asynchrone und ggf. verzögerte Updates

Vielen Dank