Universität Augsburg
University

Vertiefte Multicore-Programmierung

Universität Augsburg
Institut für Informatik
Prof. Dr. Theo Ungerer
Dr. Sebastian Weis

11. Mai 2016

Aufgabenblatt 4

1. Aufgabe (5 + 1 + 4 Punkte)

Implementieren Sie eine thread-sichere Rot-Schwarz-Baum-Klasse[1]. Der Rot-Schwarz-Baum ist eine Variante des binären Suchbaums[2] mit den folgenden zusätzlichen Eigenschaften:

- Alle Blattknoten (`NIL`) sind schwarz.
- Ist ein Knoten rot, so sind beide Kinder schwarz.
- Jeder Pfad von einem gegebenen Knoten zu seinen Blattknoten enthält die gleiche Anzahl schwarzer Knoten.

Die Klasse soll die folgenden Methoden unterstützen:

- `bool insert(int key)`: Fügt einen neuen Knoten mit dem Schlüssel `key` in den Baum ein und balanciert ihn anschließend aus. Wenn der Schlüssel im Baum schon existiert, soll er nicht eingefügt werden und `false` zurückgeliefert werden, ansonsten `true`.
- `bool search(int key, Node& n)`: Gibt in `n` eine Kopie des Knotens mit dem Schlüssel `key` zurück. Wenn der Knoten nicht existiert, soll `false` zurückgeliefert werden, ansonsten `true`.
- `bool deleteValue(int key)`: Entfernt den Knoten mit dem Schlüssel `key` und kümmert sich bei Bedarf um die Neubalancierung des Baums. Wenn der Schlüssel nicht existiert, soll `false` zurückgegeben werden, ansonsten `true`.

Die Methoden sollen später parallel von verschiedenen C++11-Threads aufgerufen werden können.

---

[1] `http://de.wikipedia.org/wiki/Rot-Schwarz-Baum`
[2] `http://de.wikipedia.org/wiki/Bin%C3%A4rer_Suchbaum`

a) Implementieren Sie zunächst eine nicht thread-sichere Variante des Rot-Schwarz-Baumes. Der Baum soll dabei nur Werte in seinen inneren Knoten speichern können. Die Blätter des Baumes sollen durch explizite `NIL` Blattknoten dargestellt werden, die selbst keine Werte enthalten. Zum Debuggen soll der Baum auf der Kommandozeile ausgegeben werden können.

b) Erweitern Sie ihre Implementierung um einen globalen Mutex, damit paralleles Einfügen, Suchen und Löschen möglich wird. Untersuchen Sie die Skalierbarkeit ihrer Lösung, wenn mehrere Threads parallel auf den Baum zugreifen.

c) Ersetzen Sie den globalen Mutex aus Teilaufgabe b) durch eine `LockElision`-Klasse, die kritische Abschnitte spekulativ mit Restricted Transactional Memory (RTM) ausführt. Die `LockElision`-Klasse soll bei fortwährenden Transaktionsabbrüchen den Programmfortschritt durch Locks sicherstellen können. Zusätzlich soll eine `RTMLock`-Klasse implementiert werden, welche die Transaktionen der `LockElision`-Klasse nach dem RAII-Idiom im Konstruktor startet und im Destruktor beendet. Welche Auswirkung hat die Verwendung von RTM auf die Skalierbarkeit des Rot-Schwarz-Baums?

Jede Teilaufgabe soll mit einer Funktion, die zunächst 10^7 zufällige Werte einfügt und danach einzeln wieder löscht, getestet werden. Mit `make profile` und `make record` können Sie das Abbruchverhalten ihrer Transaktionen untersuchen.

Hinweis

Die Abnahme der Aufgaben soll bis 18. Mai erfolgen.

Linksammlung

Transactional Synchronization Extensions

- `https://software.intel.com/en-us/articles/tsx-anti-patterns-in-lock-elision-code`
- `https://gcc.gnu.org/onlinedocs/gcc/x86-transactional-memory-intrinsics.html`
- `http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-optimization-manual.pdf`

Perf Tool

- `https://perf.wiki.kernel.org/index.php/Main_Page`