Übungen zur Vorlesung

"Paralleles Höchstleisungsrechnen "

Prof. Dr. P. Bastian, Ch. Engwer

Abgabe bis 17. 11. 2008 an christian.engwer@iwr.uni-heidelberg.de

---

In der Vorlesung haben Sie spezielle Hardware Befehle kennen gerlernt, die dabei helfen den Eintritt in einen *kritischen Abschnitt* zu koordinieren, indem sie stärkere Garantien bzgl. der Speicherkonsistenz machen, als dies bei gewöhnlichen read/write Operationen der Fall ist.

Operationen, die Sie in der Vorlesunge kennen gelernt haben sind:

- test-and-set
- atomic-swap
- fetch-and-increment
- compare-and-swap
- load-linked / store-conditional

ÜBUNG 1 QUEUE LOCK

In der Vorlesung wurde das TAS-Lock vorgestellt. Dabei viel auf, dass durch die Kokurenz um das Lock ständig Cache Misses erzeugt werden, die wiederum den Bus belasten und dadurch den Eintritt in den Kritischen Abschnitt unnötig verzögern.

Als Verbesserung wurde nun das TTAS-Lock konstruiert. Dadurch lässt sich die Anzahl an Cache Misses deutlich verringern, da erst nach Freigabe des Locks überhaupt versucht wird das Lock zu bekommen. Trotzdem kommt es in diesem Moment zu starken Verkehr auf dem Speicherbus, da alle Prozesse auf die gleiche Speicherstelle zugreifen möchten.

Eine Idee den Algorithmus weiter zu verbessern ist das sogenannte *Queue Lock*. Hierbei stellen sich alle Prozesse brav der Reihe nach an. Es gibt eine Liste von Speicherstelle (als Lock Variablen) und jeder Prozess testet auf eine andere Stelle, bis das Lock verschwindet, sprich der Wert der Speicherstelle *false* ist. Verläßt ein Prozess seinen kritischen Abschnitt, so setzt er das Lock des nächsten Prozesses auf *false*. Bei diesem Verfahren ist darauf zu achten, dass garantiert ist, dass nur ein Prozess auf eine Speicherstelle testet, d.h. dass ein Platz in der Queue an nur einen Prozess vergeben wird.

- Implementieren Sie das *Queue Lock* in der Peusdo Sprache, wie sie im Skript eingeführt wurde.
- Verwenden Sie einen der Hardware Befehle, um die Speicherkonsistenz zu garantieren.
- Begründen Sie die Wahl des Hardware Befehls.

ÜBUNG 2 HIERARCHISCHES LOCK

Moderne Mehrprozessor Systeme organisieren die Prozessoren in Clustern, hierbei ist die Kommunikation innerhalb eines Clusters deutlich Schneller, als zwischen den Clustern. Dies triff bereits auf Multi-Core-Multi-Prozessor Syteme zu, welche heute sehr verbreitet sind.

Um die Eigenschaften des *Queue Lock* für hierarchisch aufgebaute Systeme zu verbessern erinnern wir uns an das "Reduce". Die Prozesse werden in einem binären Baum angeordnet und innerhalb dieses Baumes findet dann die Kommunikation statt. Diese Idee läßt sich direkt auf unser Problem übertragen. Die Freigabe des nächsten Locks findet nun auch innerhalb einer hierarchischen Baumstruktur statt.

Wir nehmen an, dass 2^n Prozessoren im System sind und auf jedem Prozessor genau ein Prozess läuft. Diese konkurieren nun um den Kritischen Abschnitt.

- Entwickeln Sie ein *Hierarisches Lock* wo die Freigabe des Lock in eines Baumstruktur verwaltet wird.
- Implementieren Sie das Verfahren in der Peusdo Sprache der Vorlesung.
- Überlegen Sie, was bei Ihrem Lock die Caches machen.
- Ist Ihr Lock fair, d.h. kommt jeder wartende Prozess garantiert dran?