

Praktikum angewandte Systemsoftwaretechnik: Vergleich nichtblockierender Queues

Michael Banken Lorenz Haspel

Friedrich-Alexander Universität Erlangen-Nürnberg

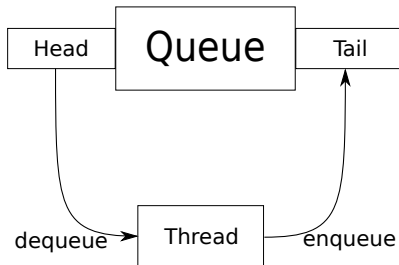
20.9.2013



- im Rahmen von LAOS/OctoPOS
- Queues als zentrale Datenstruktur für die Verwaltung von Threads
- paralleler Zugriff auf gemeinsame Daten
- unsere Zielsetzung: Performanzanalyse und die Entwicklung paralleler Tests mit verschiedenen nichtblockierenden Queues



Durchgeführte Tests: "Roundtest"



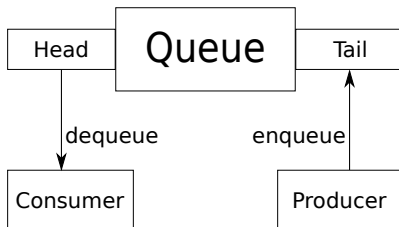
- eine Queue, zu Beginn gefüllt
- Threads entfernen je ein Element aus der Queue und fügen dieses wieder ein.
- Elemente bleiben konstant, werden immer wieder verwendet



- Versuchsreihen mit variierender Anzahl an Threads und Elementen
- feste Gesamtzahl an Operationen
- Zeit wird gemessen
- drei Testreihen:
 - halb so viele Elemente wie Threads
 - genau so viele Elemente wie Threads
 - doppelt so viele Elemente wie Threads



Durchgeführte Tests: "MPMC-Test"



- Multiple Producers Multiple Consumers
- Queue zu Beginn leer
- Produzententhreads fügen Elemente ein
- Konsumententhreads nehmen Elemente raus
- jedes Element wird nur einmal verwendet



- Versuchsreihe mit variierender Anzahl an Threads
- feste Anzahl an Elementen, die die Queue durchlaufen
- zwei Testreihen:
 - variable Zahl an Produzenten, ein Konsument
 - gleiche Zahl von Produzenten wie Konsumenten

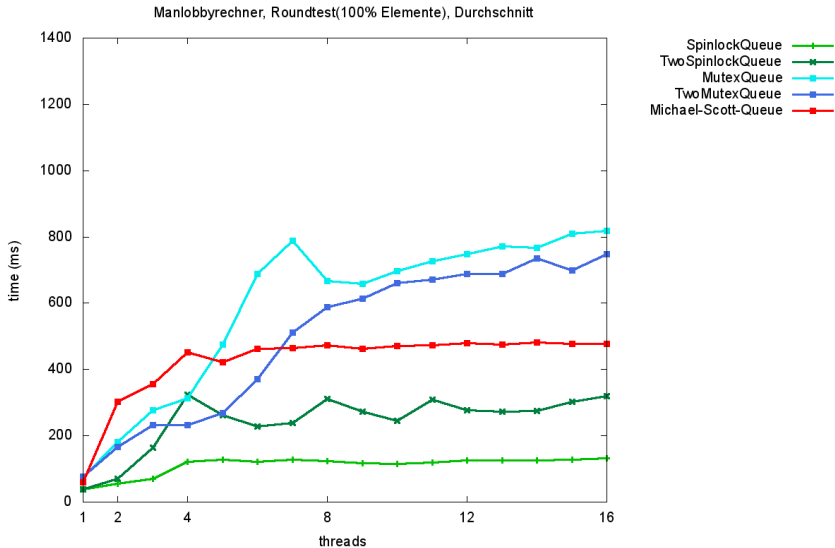


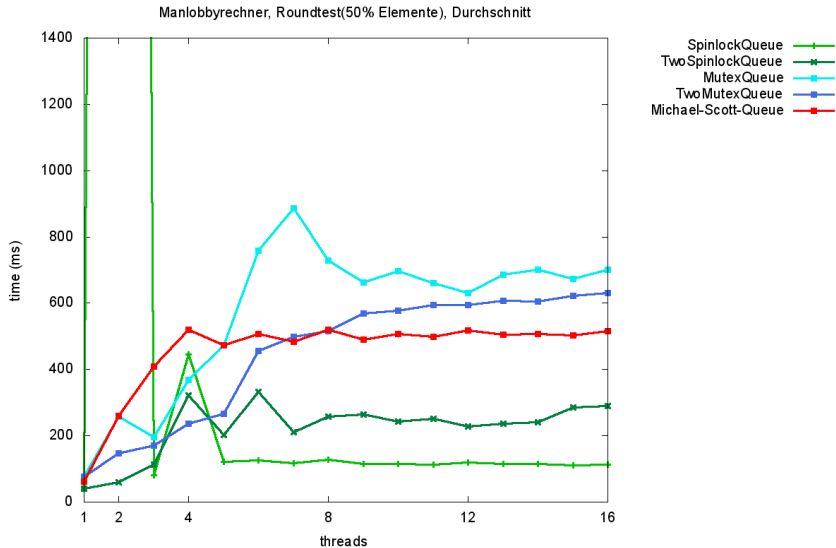
- Core i7 aus der Manlobby:
 - 4 Kerne, 8 durch Hyperthreading
 - Tests neben normaler Betriebslast (Programme andere User, X-server, Ubuntu-Hintergrundprogramme)
- Haswellrechner "Fastbox":
 - 8 Kerne, Intel-Haswell Prozessoren
 - Test von Queues mit naiver Implementierung von Haswell-Transaktionen
 - atomare Transaktionen auf Prozessorebene
- 48-Kern-Rechner "Bigbox":
 - Tests bei massiv parallelem Zugriff

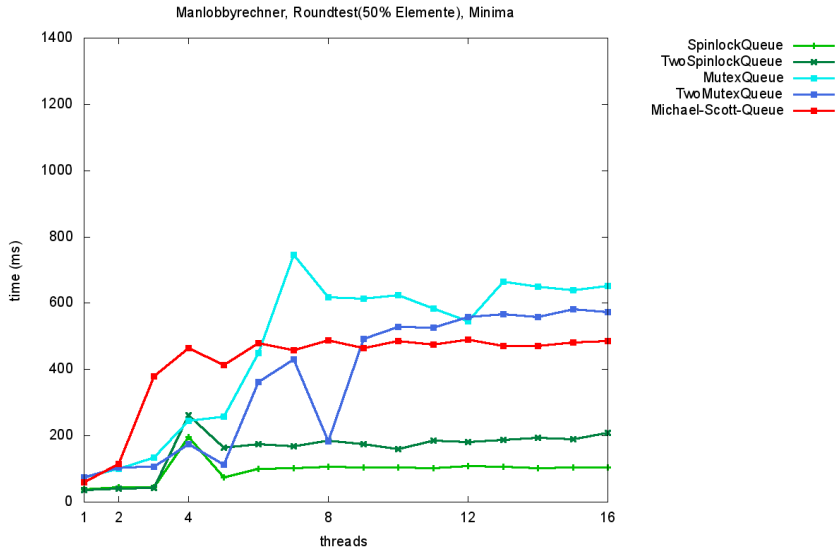


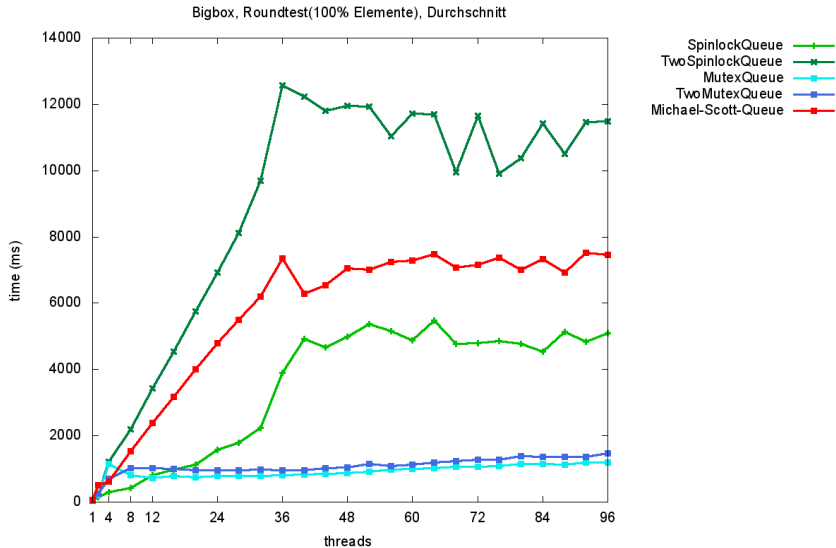
- SimpleQueue: Queue mit einem globalen Lock für en- und dequeue-Operationen
 - Varianten: Spinlock, Mutex, Haswell-Transaktion
- TwoLockQueue: Queue mit zwei Locks, je eines für en- und dequeue Operationen
 - Varianten: Spinlock, Mutex, Haswell-Transaktionen
- Michael-Scott-Queue: nichtblockierende Queue
 - verwendet CAS-Operationen
- MPSC-Queue: Multiple-Producers-Single-Consumer-Queue
 - spezialisierte nicht blockierende Queue

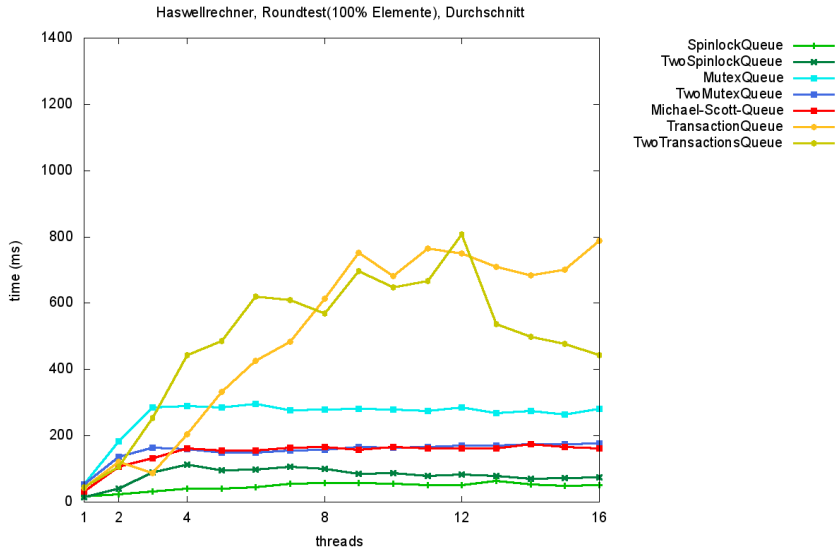


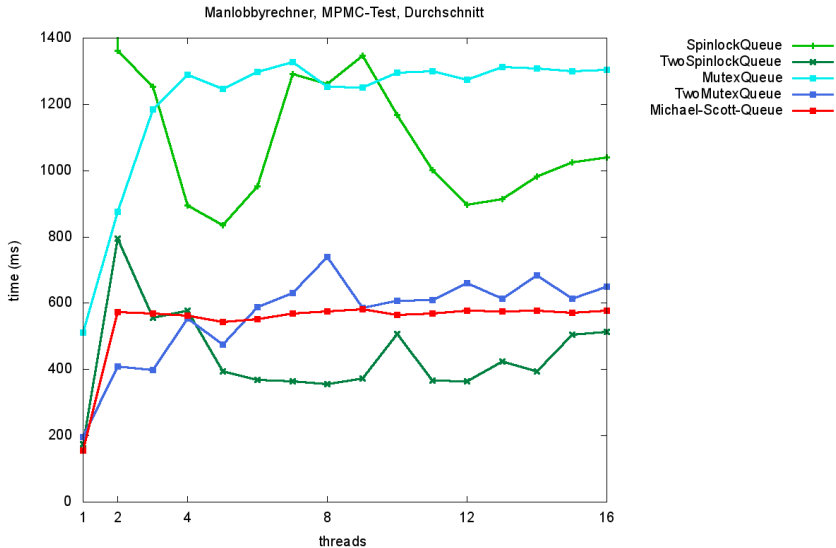


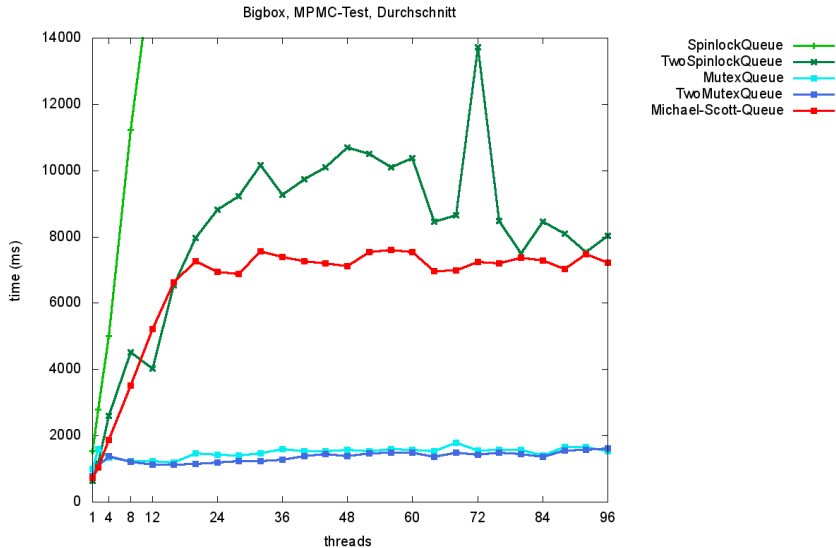


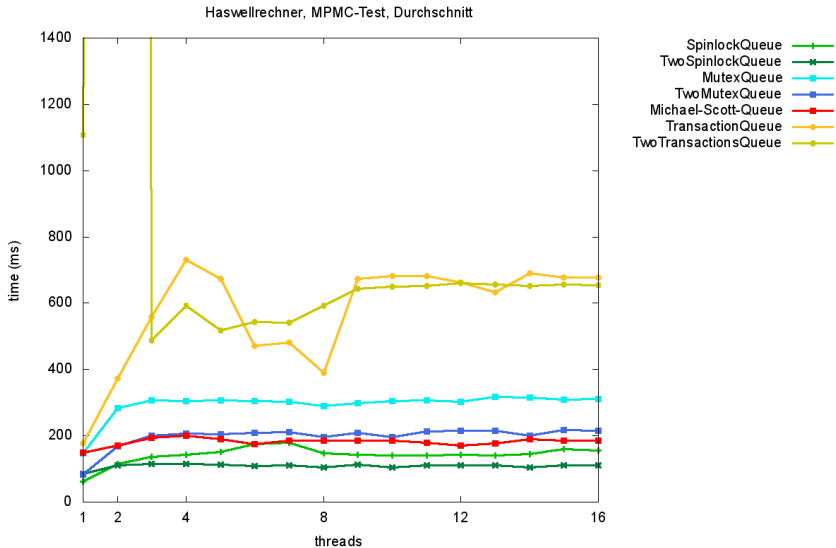


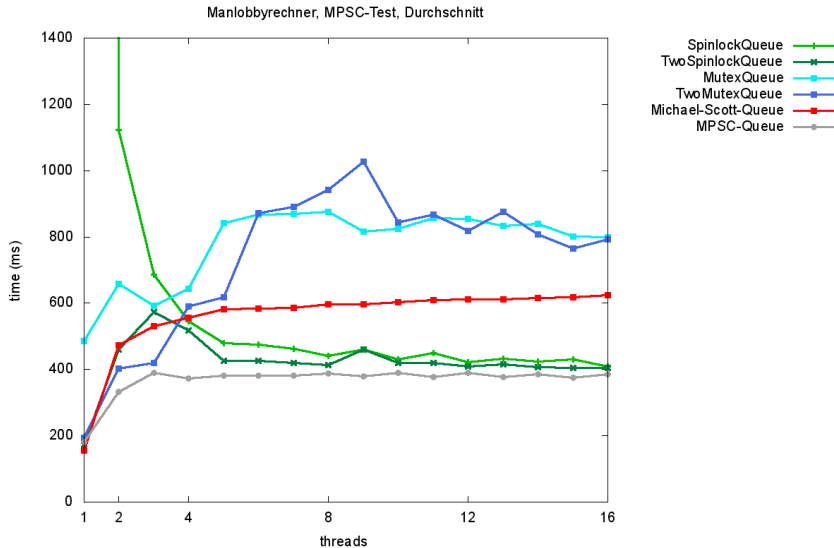


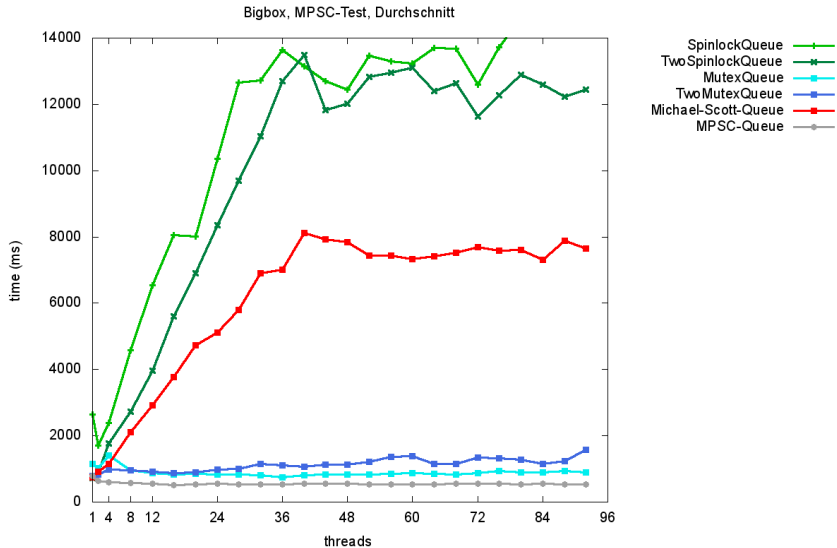


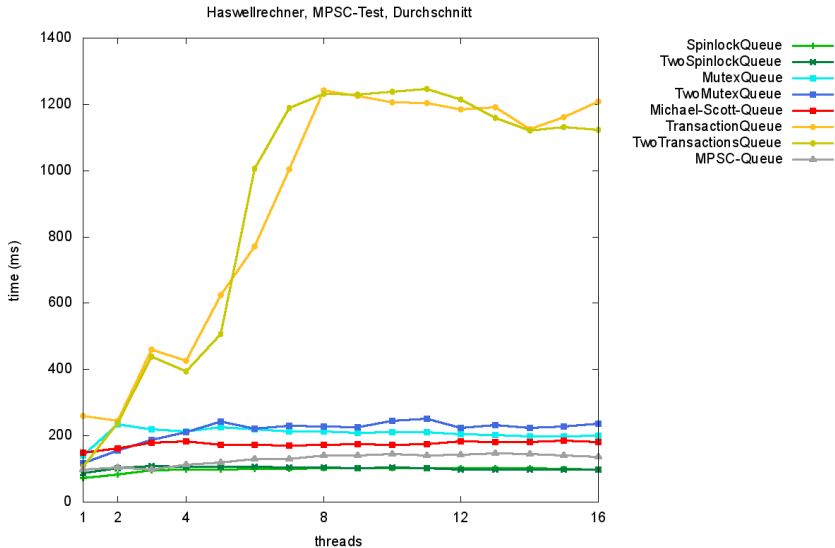












- Michael-Scott Queue ist durchgehend suboptimal. Allerdings gibt es auch immer schlechtere Varianten. Insgesamt ist sie damit in jeder Hinsicht mittelmäßig.
- MPSC-Queue ist auf den meisten Systemen besser als blockierende Queues, aber auf Haswell Architektur wird sie von der Spinlock Queue überholt.
- Insgesamt ist es nicht offensichtlich, welche Queue unter welchen Bedingungen optimal arbeitet.
- Messdaten, Testumgebung und verwendete Queues auf:
[git://github.com/Lorenz-badgers/qtpie](https://github.com/Lorenz-badgers/qtpie)
<https://github.com/Lorenz-badgers/qtpie.git>



- nachträgliche Implementierung gemeinsamer Schnittstelle zwischen verschiedenen Queues und Testsystem komplexer als erwartet
- Messungen für präsentierte Graphen später als geplant durchführbar
- Umgang mit Messdaten: mehr Daten speichern, um z.B. Streuungen ermitteln zu können



Vielen Dank für Ihre Aufmerksamkeit!

