Performance Paradigma

How to avoid multi-threading performance breaks, in general and particularly in Java

How to help the JVM understanding your goal

How to get pointers in the JVM, bypass security handlers and use low-level commands (how to invite the "there be dragons" from C)

Marc Schmit 09.11.2011
Synchronization / Locks

- Avoid critical sections as much as possible
- Prefer hardware locks over system locks
- Prefer system locks over software locks
- Design your code to support arbitrary asynchronous scheduling
- Prefer parametrized algorithms in an early development stage to (quickly / automatically) test scaling
Execution Flow Analysis

- Simulate how your program should run and how it will run in best/worst case
- Try to optimize the program to fit your expected design: parallelism, dependency
- Avoid wait&hold, prefer dedicate&sleep pattern (my kids will wake me when done)
- The hardware offers extremely fast units, use them (Branch prediction, Pre Caching, Local Allocation,...)
- If you can't access the devices manually, play into their hands³
Java & the JVM

- Avoid calling functions within loops
- Declare as much fields/methods private as you can
- Avoid static, it forces non-local storage access
- Inline your code as much as you can, linearise it if possible
- Prefer passing deep copies to sharing a variable
- Primitives are faster than wrappers or classes
- Make use of the CAS-ISA (Compare&Swap-Instructions)
Java & the JVM

- Use Java7, use its documentation, use Eclipse Juno, code in 64bit
- Fork&Join framework for functional recursion or work stealing pattern, since its threads are „lightweight“
- „java.util.concurrent.atomic“ - for shared variables
- ThreadLocals, TLRandom, make use of thread local allocation
- Stay away from „Services“ and high level constructs, they are fine for academic and general purposes but they are slow and don't work as you think they do

Marc Schmit 09.11.2011
Java & the JVM

- The JVM offers command line options that are, by default, balanced between safety and performance
  - -XX:PreBlockSpin=10
  - -XX:+UseSpinning
  - -XX:+RelaxAccessControlCheck
  - -XX:+UseTLAB
  - -XX:AllocatePrefetchStyle=2
  - -XX:+UseSplitVerifier
  - -XX:+UseThreadPriorities
  - -XX:+UseBiasedLocking
  - -XX:+UseFastAccessorMethods
  - -XX:+UseStringCache
  - -XX:+UseCompressedStrings
  - -XX:+OptimizeStringConcat
  - ...

The dragons ...

better switch to C/C++ if you really want to do this ;)

I am a link