- CPU?
  - Lots of user%
    - JVM is burning the cycles?
      - AR: Know your command-line options
      - AR: Upgrade to newer JVM?
      - GC
        - **TOOL: -verbose:gc, -XX:+PrintGCDetails, VisualGC**
        - AR: Tune Java heap, generations, and regions
        - AR: Thread stack size
        - AR: (Un)usual tuning
          - HotSpot GCs
            - **TOOL: -XX:+PrintGCDetails, -XX:+PrintGCTimeStamps, -XX:+PrintGCDateStamps, -XX:+PrintHeapAtGC, -XX:+PrintTenuringDistribution, -Xloggc=<file>**
            - -XX:+UseSerialGC
            - -XX:+UseParallelGC
              - The only GC supports NUMA
              - Collects young gen in parallel
              - Collects old gen in single thread
            - -XX:+UseParallelOldGC
              - Collects young gen in parallel
              - Collects old gen in parallel
            - -XX:+UseConcMarkSweepGC
              - Extensive tuning opportunities, see elsewhere
            - -XX:+UseG1GC
              - Replaces CMS
              - Set region size: -XX:G1HeapRegionSize=#
              - Set the maximum pause for GC: -XX:MaxGCPauseMillis=#
              - Set the usual time between GCs: -XX:GCPauseIntervalMillis=#
          - JRockit GCs
            - -XgcPrio:deterministic
            - -XpauseTarget=#
      - Classload
        - **TOOL: verbose:class, MXBeans**
        - AR: Turn off bytecode verification: --no-verify
        - AR: Turn on CDS: -Xshare:on
        - AR: Recompile your Java code with updated javac
        - AR: Increase the size of system dictionary
        - AR: Repackage classes into small amount of larger JARs
      - JIT
        - **TOOL: PrintCompilation, MXBeans**
        - AR: Choose the compiler
          - -server
          - -client
          - -XX:+TieredCompilation
        - AR: Low-level tuning
        - AR: Go to OpenJDK ML and ask
    - Algorithmic problems?
      - Algorithmic complexity
        - **TOOL: Profilers + Brain**
        - AR: Pick the algorithm with lower complexity
        - AR: Pick the algorithm with lower constants
      - Caching/Memoizing
        - **TOOL: Profilers + Brain**
        - AR: Memoize the results where appropriate
        - AR: Use new objects where appropriate
        - AR: For (distributed) caching the record size should be smaller
      - Busy-waiting
        - **TOOL: Profilers + Brain**
        - AR: Replace polling with timed waits
        - AR: Replace spinloops with spin-then-block
      - Batching and work scheduling
    - Memory problems?
      - TLB
        - **TOOL: Easier to fix and test; or, use HWCs to diagnose**
        - AR: -XX:+UseLargePages
        - AR: Large page sizes?
      - Caches
        - Capacity
          - **TOOLS: (HWC) oracle solaris studio performance analyzer, vtune**
          - AR: Enable/Disable prefetches
          - Temporal locality
            - AR: Blocking decompositions
          - Spatial locality
            - AR: Shrink data set
            - AR: -XX:+UseCompressedOops
            - AR: Denser data structures
        - Coherence
          - **TOOLS: Java-level profiling + HWC**
          - Primitives
            - Plain non-shared memory
            - Plain shared memory
            - Volatile: enforcing visibility
            - Atomics: enforcing atomicity
            - Spin-loops: enforcing mutual exclusion
          - Locks
            - Spin-locks
            - Wait-locks
              - synchronized
              - j.u.c.RL
          - Techniques
            - AR: Choose the correct primitive
              - Consistency
              - Expected contention
              - Expected contention overlap
            - AR: Optimistic checks
              - Try to optimistically check for easier condition
              - Fallback to pessimistic (costly) operation otherwise
            - AR: Striping
              - Locks
              - Queues
              - Counters
            - AR: Get rid of the communication whatsoever
              - Immutability
              - Distinct copies
              - Thread Locals
            - AR: False Sharing
              - Object Padding
              - Split the objects, and pad again
      - NUMA (NUCA)
        - **TOOL: numastat**
        - Fractal structure
          - AR: Communication cost is the major contender
          - ...not only between CPU packages, but between cores, hosts, clouds
        - AR: Locality of communication
        - AR: -XX:+UseNUMA
        - AR: Thread/Memory Affinity
      - Memory bandwidth
        - **TOOL: busstat, multevent**
        - AR: More faster memory
        - AR: Multiple channels to main memory
        - AR: Multiple IMCs to handle the load
    - CPU problems?
      - Not enough CPU frequency?
        - AR: Overclocking
        - AR: CPU frequency governors
      - Not enough Execution Units?
        - **TOOL: (HWC), vtune, solstudio**
        - Specialized code
          - AR: Going for native platfrom-specific code
          - AR: JIT intrinsics
        - Specialized Hardware
          - AR: cryptoaccelerators
          - AR: GPU
        - AR: Moar CPUs!
      - ILP depleted?
        - **TOOL: (HWC) solstudio, vtune**
        - AR: less branches?
        - AR: more ILP
  - Lots of sys%
    - Network problems?
      - **TOOL: netstat, sar, iptraf, bwm-ng**
      - AR: Reduce the traffic, packet count
      - AR: Compression
      - AR: Bufferization, BDP
      - AR: MTU
      - AR: Faster hardware/links
      - AR: Virtual interfaces
    - Scheduling overheads?
      - **TOOL: vmstat, mpstat, sar**
      - AR: Reduce the amount of worker threads
      - AR: Less context switches
      - AR: Scheduling groups, quanta adjustments, priority
    - Swapping?
      - **TOOL: top, sar**
      - AR: Constrain the usage of physical memory
      - AR: Decrease memory per process
      - AR: Swappiness
      - AR: Lock pages in memory
      - AR: Compress swap
    - Other kernel?
      - **TOOL: strace, perf, oprofile**
      - AR: Time spent in other kernel?
      - AR: Time spent in kernel-space with locking
      - AR: Kernel bugs?
  - Lots of irq%, soft%
    - Interacting with devices?
      - **TOOL: mpstat, sar**
      - AR: Interrupt offload
      - AR: IRQ balancing
  - Lots of iowait%
    - Extensive disk activity?
      - **TOOL: iostat, sar**
      - AR: Reduce the disk activity
      - AR: HW caching/bufferization
      - AR: SW caching/bufferization
      - AR: More disks always help (but not your budget)
    - Not enough disk/block caches?
      - **TOOL: top, sar**
      - AR: Increase cache memory (reduce other usages)
      - AR: Get easy on flush()-es and cache invalidations
      - AR: More disks always help (but not your budget)
  - Lots of idle%
    - Not enough SW threads
      - **TOOL: vmstat, mpstat**
      - AR: Get more threads! Parallelize application
      - AR: Make the scheduler to use physical cores first (affinity)
      - AR: Turn off CMT / use critical strands
    - Not enough RUNNABLE SW threads
      - **TOOL: lock profilers, jstack**
      - Wait locks?
        - AR: Get rid of the locks
        - AR: Use lock-free algos
      - Network latencies
    - GC pauses?
      - **TOOL: -verbose:gc, etc**
      - AR: More threads for GC
      - AR: Pause-targeted GC-specific tuning
  - Credits
    - Aleksey Shipilev
      - Original mindmap
      - aleksey.shipilev@oracle.com
    - Sergey Kuksenko
      - Original mindmap
      - sergey.kuksenko@oracle.com
    - Vladimir Ivanov
      - GC parts
      - vladimir.x.ivanov@oracle.con
    - Igor Maznitsa
      - Partial translation to English
      - igor.maznitsa@igormaznitsa.com