ORACLE

@Contended (a.k.a. JEP 142)

Aleksey Shipilev

aleksey.shipilev@oracle.com, @shipilev

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

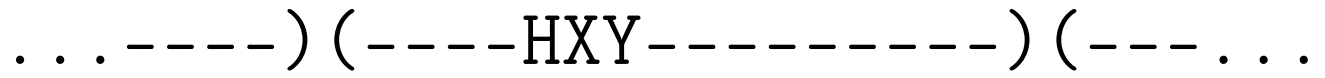
Basics

Basics: False Sharing

Java view:

```
public class Point {
    int x;
    int y;
}
```

Memory view:



Basics: False Sharing

Java view:

```
public class Point {
    int x;
    int y;
}
```

Memory view:

Approaches

Approaches: Padding

Java view:

```
public class Point {
    int x;
    int p01, p02, ..., p16;
    int y;
    int p17, p18, ..., p32;
}
```

Memory view (expected):

```
...----)(----HXpppppppppppp)(ppYpp...
...--HX)(ppppppppppppppppp)(Yppp...
```

Approaches: Padding

Java view:

```
public class Point {
    int x;
    int p01, p02, ..., p16;
    int y;
    int p17, p18, ..., p32;
}
```

Memory view (legal):

```
...-----)(----HXYpppppppppppp)(ppppp...
```


Approaches: Arrayifying

Java view:

```
public class Point {
    int x;
    int[] storage = new int[2*64+1];

    int getX() { return x; }
    int getY() { return storage[64]; }
}
```

Memory view (hopefully):

```
...----)(----HX**********)(****Y****...
```


Approaches: Subclass trick

Relies on «superclass fields laid out first» mojo.

Java view:

```
class L1                    { int x; }
class L2 extends L1         { int p01, ..., p16; }
class L3 extends L2         { int y; }
class L4 extends L3         { int p17, ..., p32; }
class Point extends L4 { // x, y visible }
```

Memory view:

```
...-----)(----HXpppppppppp)(ppppYpp...
```


Approaches: @Contended

Java view:

```
public class Point {
    int x;

    @Contended
    int y;
}
```

Memory view:

@Contended

@Contended: Semantics

@Contended declares the **intent**.

- «This field/object should not experience false sharing»
- Only advertises the memory contention effects, makes no promises about the actual implementation
- sun.misc.Contended :(

@Contended: Actual implementation

- HS already packs the fields to minimize the footprint
- Field-level @Contended «bails out» the specific fields from that packing, laying them out at sparse offsets
- Class-level @Contended «shifts» the entire instance field block away from the header; and also beefs up instance size to provide the pad in the back
- Done during the initial field layout (class load)

ORACLE

@Contended: Caveats

- Object header is not protected
  - ...only concerns itself with the fields (blocks)
- Pads for twice the cache line size
  - Without the GC support, the object is not guaranteed to be aligned to cache line
  - Pessimistically handling the case of enabled adjacent cache line prefetchers
- Static fields are not supported
  - Requires significant rework of static field mechanics
- Restricted on user classpath
  - Unlock: -XX:-RestrictContended

@Contended: Usages

```
$ ack-grep -Q \@sun.misc.Contended
src/share/classes/java/util/concurrent/ConcurrentHashMap.java
2458:     @sun.misc.Contended static final class CounterCell {

src/share/classes/java/util/concurrent/Exchanger.java
310:      * bookkeeping. Padded via @sun.misc.Contended to reduce m
313:     @sun.misc.Contended static final class Node {

src/share/classes/java/util/concurrent/ForkJoinPool.java
161:@sun.misc.Contended
643:     @sun.misc.Contended

src/share/classes/java/util/concurrent/atomic/Striped64.java
55:      * (via @sun.misc.Contended) to reduce cache contention. Pa
119:     @sun.misc.Contended static final class Cell {

src/share/classes/java/lang/Thread.java
2004:     @sun.misc.Contended("tlr")
2008:     @sun.misc.Contended("tlr")
2012:     @sun.misc.Contended("tlr")
```

ORACLE

@Contended:
Try it. Use it. Break it.

Thanks!