The Java Fork-Join Pool: Applying the ManagedBlocker Interface

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University

Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Understand the common fork-join pool
- Recognize how the ManagedBlocker interface helps avoid starvation & improve performance
- Know how to apply the ManagedBlocker interface in practice

```
public class BlockingTask {
  ...
  public static<T> T
    callInManagedBlocker
      (Supplier<T> supplier){
    ...
    ForkJoinPool.managedBlock
     (managedBlocker);
    ...
    return managedBlocker
      .getResult();
  }
  ...
```

Applying the Managed Blocker Interface

Applying the ManagedBlocker Interface

- This example applies a ManagedBlocker on a ReentrantLock (from Java docs)

```
class ManagedLocker implements ManagedBlocker {
  final ReentrantLock mLock;
  boolean mHasLock = false;

  ManagedLocker(ReentrantLock lock) { mLock = lock; }

  public boolean isReleasable()
  { return mHasLock || (mHasLock = mLock.tryLock()); }

  public boolean block() {
    if (!mHasLock) mLock.lock();
    return true;
  }
}
```

Handles a blocking synchronizer

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.ManagedBlocker.html

Applying the ManagedBlocker Interface

- This example applies a ManagedBlocker on a ReentrantLock (from Java docs)

```
class ManagedLocker implements ManagedBlocker {
  final ReentrantLock mLock;
  boolean mHasLock = false;

  ManagedLocker(ReentrantLock lock) { mLock = lock; }

  public boolean isReleasable()
  { return mHasLock || (mHasLock = mLock.tryLock()); }

  public boolean block() {
    if (!mHasLock) mLock.lock();
    return true;
  }
}
```

Constructor stores the lock

Applying the ManagedBlocker Interface

- This example applies a ManagedBlocker on a ReentrantLock (from Java docs)

```
class ManagedLocker implements ManagedBlocker {
  final ReentrantLock mLock;
  boolean mHasLock = false;

  ManagedLocker(ReentrantLock lock) { mLock = lock; }

  public boolean isReleasable()
  { return mHasLock || (mHasLock = mLock.tryLock()); }

  public boolean block() {
    if (!mHasLock) mLock.lock();
    return true;
  }
}
```

Tries to acquire the lock (non-blocking)

Applying the ManagedBlocker Interface

- This example applies a ManagedBlocker on a ReentrantLock (from Java docs)

```
class ManagedLocker implements ManagedBlocker {
  final ReentrantLock mLock;
  boolean mHasLock = false;

  ManagedLocker(ReentrantLock lock) { mLock = lock; }

  public boolean isReleasable()
  { return mHasLock || (mHasLock = mLock.tryLock()); }

  public boolean block() {
    if (!mHasLock) mLock.lock();
    return true;
  }
}
```

Performs a blocking lock operation

Applying the ManagedBlocker Interface

- This example applies a ManagedBlocker on a BlockingQueue (from Java docs)

```
class QueueTaker<E> implements ManagedBlocker {
  final BlockingQueue<E> mQueue;
  volatile E mItem = null;

  QueueTaker(BlockingQueue<E> q) { mQueue = q; }

  public boolean isReleasable()
  { return mItem != null || (mItem = mQueue.poll()) != null; }

  public boolean block() throws InterruptedException
  { if (mItem == null) mItem = mQueue.take(); return true; }

  public E getItem() { return mItem; }
}
```

Handles a blocking queue

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.ManagedBlocker.html

Applying the ManagedBlocker Interface

- This example applies a ManagedBlocker on a BlockingQueue (from Java docs)

```
class QueueTaker<E> implements ManagedBlocker {
  final BlockingQueue<E> mQueue;
  volatile E mItem = null;

  QueueTaker(BlockingQueue<E> q) { mQueue = q; }

  public boolean isReleasable()
  { return mItem != null || (mItem = mQueue.poll()) != null; }

  public boolean block() throws InterruptedException
  { if (mItem == null) mItem = mQueue.take(); return true; }

  public E getItem() { return mItem; }
}
```

The blocking queue

Applying the ManagedBlocker Interface

- This example applies a ManagedBlocker on a BlockingQueue (from Java docs)

```
class QueueTaker<E> implements ManagedBlocker {
  final BlockingQueue<E> mQueue;
  volatile E mItem = null;

  QueueTaker(BlockingQueue<E> q) { mQueue = q; }

  public boolean isReleasable()
  { return mItem != null || (mItem = mQueue.poll()) != null; }

  public boolean block() throws InterruptedException
  { if (mItem == null) mItem = mQueue.take(); return true; }

  public E getItem() { return mItem; }
}
```

Try to get an item (non-blocking)

Applying the ManagedBlocker Interface

- This example applies a ManagedBlocker on a BlockingQueue (from Java docs)

```
class QueueTaker<E> implements ManagedBlocker {
  final BlockingQueue<E> mQueue;
  volatile E mItem = null;

  QueueTaker(BlockingQueue<E> q) { mQueue = q; }

  public boolean isReleasable()
  { return mItem != null || (mItem = mQueue.poll()) != null; }

  public boolean block() throws InterruptedException
  { if (mItem == null) mItem = mQueue.take(); return true; }

  public E getItem() { return mItem; }
}
```

Block until an item is available

Applying the ManagedBlocker Interface

- This example applies a ManagedBlocker on a BlockingQueue (from Java docs)

```
class QueueTaker<E> implements ManagedBlocker {
  final BlockingQueue<E> mQueue;
  volatile E mItem = null;

  QueueTaker(BlockingQueue<E> q) { mQueue = q; }

  public boolean isReleasable()
  { return mItem != null || (mItem = mQueue.poll()) != null; }

  public boolean block() throws InterruptedException
  { if (mItem == null) mItem = mQueue.take(); return true; }

  public E getItem() { return mItem; }
}
```

Call after pool.managedBlock() completes

Encapsulating ManagedBlocker w/the BlockingTask Class

Encapsulating ManagedBlocker w/the BlockingTask Class

- BlockingTask integrates blocking suppliers with the common fork/join pool

```
public class BlockingTask {
  ...
  public static<T> T callInManagedBlocker(Supplier<T> supplier){




    SupplierManagedBlocker<T> managedBlocker =
      new SupplierManagedBlocker<T>(supplier);
    ...
    ForkJoinPool.managedBlock(managedBlocker);
    ...
    return managedBlocker.getResult();
  }
  ...
```

See github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex20

Encapsulating ManagedBlocker w/the BlockingTask Class

- BlockingTask integrates blocking suppliers with the common fork/join pool

```
public class BlockingTask {
  ...
  public static<T> T callInManagedBlocker(Supplier<T> supplier){
```

Enables the use of blocking suppliers with the common Java fork/join thread pool

```
    SupplierManagedBlocker<T> managedBlocker =
      new SupplierManagedBlocker<T>(supplier);
    ...
    ForkJoinPool.managedBlock(managedBlocker);
    ...
    return managedBlocker.getResult();
  }
  ...
```

See stackoverflow.com/q/37512662 for pros & cons of this approach

Encapsulating ManagedBlocker w/the BlockingTask Class

- BlockingTask integrates blocking suppliers with the common fork/join pool

Create a helper object to encapsulate the supplier

```
public class BlockingTask {
  ...
  public static<T> T callInManagedBlocker(Supplier<T> supplier){

    SupplierManagedBlocker<T> managedBlocker =
      new SupplierManagedBlocker<T>(supplier);
    ...
    ForkJoinPool.managedBlock(managedBlocker);
    ...
    return managedBlocker.getResult();
  }
  ...
```

Encapsulating ManagedBlocker w/the BlockingTask Class

- BlockingTask integrates blocking suppliers with the common fork/join pool

```
public class BlockingTask {
  ...
  public static<T> T callInManagedBlocker(Supplier<T> supplier){




    SupplierManagedBlocker<T> managedBlocker =
      new SupplierManagedBlocker<T>(supplier);
    ...
    ForkJoinPool.managedBlock(managedBlocker);
    ...
    return managedBlocker.getResult();
  }
  ...
```

Submit managedBlocker to common ForkJoin thread pool

Encapsulating ManagedBlocker w/the BlockingTask Class

- BlockingTask integrates blocking suppliers with the common fork/join pool

```
public class BlockingTask {
  ...
  public static<T> T callInManagedBlocker(Supplier<T> supplier){




    SupplierManagedBlocker<T> managedBlocker =
      new SupplierManagedBlocker<T>(supplier);
    ...
    ForkJoinPool.managedBlock(managedBlocker);
    ...
    return managedBlocker.getResult();
  }
  ...
```

Return the result of the blocking call

Encapsulating ManagedBlocker w/the BlockingTask Class

- BlockingTask integrates blocking suppliers with the common fork/join pool

```
public class BlockingTask {
  ...
  private static class SupplierManagedBlocker<T>
                 implements ForkJoinPool.ManagedBlocker {
    private final Supplier<T> mSupplier;

    private boolean mDone = false;

    private T mResult;


    private SupplierManagedBlocker(final Supplier supplier)
    { mSupplier = supplier; }
    ...
```

Blocking Supplier work w/common fork/join pool

Encapsulating ManagedBlocker w/the BlockingTask Class

- BlockingTask integrates blocking suppliers with the common fork/join pool

```
public class BlockingTask {
  ...
  private static class SupplierManagedBlocker<T>
                implements ForkJoinPool.ManagedBlocker {
    private final Supplier<T> mSupplier;

    private boolean mDone = false;

    private T mResult;


    private SupplierManagedBlocker(final Supplier supplier)
    { mSupplier = supplier; }
    ...
```

Store supplier param for subsequent use

Encapsulating ManagedBlocker w/the BlockingTask Class

- BlockingTask integrates blocking suppliers with the common fork/join pool

```
public class BlockingTask {
  ...
  private static class SupplierManagedBlocker<T>
                  implements ForkJoinPool.ManagedBlocker {
    private final Supplier<T> mSupplier;

    private boolean mDone = false;

    private T mResult;


    private SupplierManagedBlocker(final Supplier supplier)
    { mSupplier = supplier; }
    ...
```

Keeps track of whether blocking supplier is done

Encapsulating ManagedBlocker w/the BlockingTask Class

- BlockingTask integrates blocking suppliers with the common fork/join pool

```
public class BlockingTask {
  ...
  private static class SupplierManagedBlocker<T>
                 implements ForkJoinPool.ManagedBlocker {
    private final Supplier<T> mSupplier;

    private boolean mDone = false;

    private T mResult;


    private SupplierManagedBlocker(final Supplier supplier)
    { mSupplier = supplier; }
    ...
```

Stores result obtained from the supplier for later use

Encapsulating ManagedBlocker w/the BlockingTask Class

- BlockingTask integrates blocking suppliers with the common fork/join pool

```
public class BlockingTask {
  ...
  private static class SupplierManagedBlocker<T>
                 implements ForkJoinPool.ManagedBlocker {
    ...
    public boolean block()
    { mResult = mSupplier.get(); mDone = true; return true; }

    public boolean isReleasable()
    { return mDone; }

    public T getResult()
    { return mResult; }
}
```

Sets result via the blocking supplier's get() method

Encapsulating ManagedBlocker w/the BlockingTask Class

- BlockingTask integrates blocking suppliers with the common fork/join pool

```
public class BlockingTask {
  ...
  private static class SupplierManagedBlocker<T>
                implements ForkJoinPool.ManagedBlocker {
    ...
    public boolean block()
    { mResult = mSupplier.get(); mDone = true; return true; }

    public boolean isReleasable()
    { return mDone; }

    public T getResult()
    { return mResult; }
}
```

Indicate the result's been obtained

Encapsulating ManagedBlocker w/the BlockingTask Class

- BlockingTask integrates blocking suppliers with the common fork/join pool

```
public class BlockingTask {
  ...
  private static class SupplierManagedBlocker<T>
                  implements ForkJoinPool.ManagedBlocker {
    ...
    public boolean block()
    { mResult = mSupplier.get(); mDone = true; return true; }

    public boolean isReleasable()
    { return mDone; }

    public T getResult()
    { return mResult; }
}
```

True if blocking supplier has finished, else false

There is no "non-blocking" behavior for this abstraction

Encapsulating ManagedBlocker w/the BlockingTask Class

- BlockingTask integrates blocking suppliers with the common fork/join pool

```
public class BlockingTask {
  ...
  private static class SupplierManagedBlocker<T>
                 implements ForkJoinPool.ManagedBlocker {
    ...
    public boolean block()
    { mResult = mSupplier.get(); mDone = true; return true; }

    public boolean isReleasable()
    { return mDone; }

    public T getResult()
    { return mResult; }
}
```

Returns supplier's result (called after pool.managedBlock() completes)

Encapsulating ManagedBlocker w/the BlockingTask Class

- This example uses BlockingTask to ensure there are enough threads in the common thread pool

```
Image blockingDownload(URL url) {
  return BlockingTask
    .callInManagedBlocker
      (() -> downloadImage(url));
}
```

See github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex20

Encapsulating ManagedBlocker w/the BlockingTask Class

- This example uses BlockingTask to ensure there are enough threads in the common thread pool

```
Image blockingDownload(URL url) {
  return BlockingTask
    .callInManagedBlocker
      (() -> downloadImage(url));
}
```

Transform a URL to an Image by downloading each image via its URL

Encapsulating ManagedBlocker w/the BlockingTask Class

- This example uses BlockingTask to ensure there are enough threads in the common thread pool

```
Image blockingDownload(URL url) {
  return BlockingTask
    .callInManagedBlocker
      (() -> downloadImage(url));
}
```

This method call ensures the common fork/join thread pool is expanded to handle the blocking image download

Encapsulating ManagedBlocker w/the BlockingTask Class

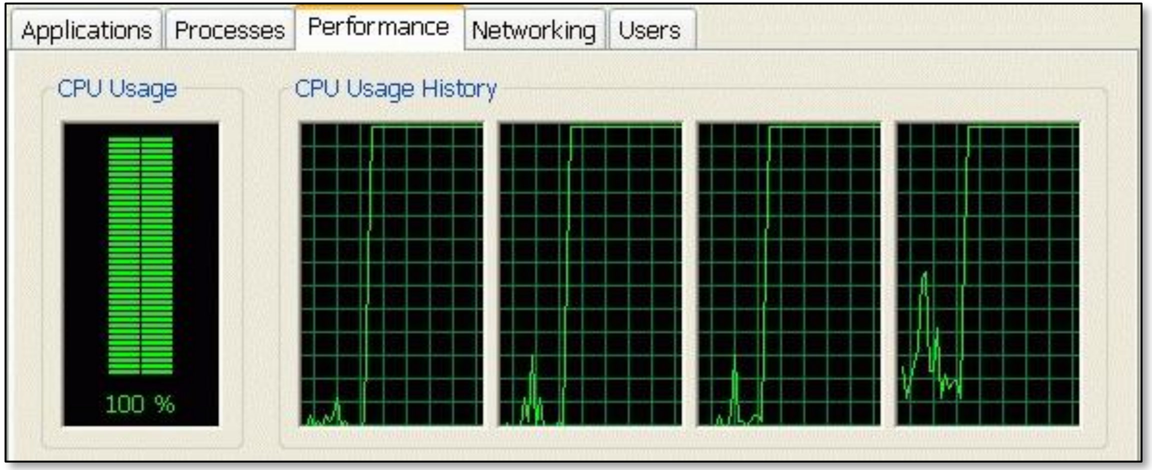
- This example uses BlockingTask to ensure there are enough threads in the common thread pool
  - Extra threads in the common fork-join pool are automatically terminated later

```
Image blockingDownload(URL url) {
  return BlockingTask
    .callInManagedBlocker
      (() -> downloadImage(url));
}
```

Encapsulating ManagedBlocker w/the BlockingTask Class

- This example uses BlockingTask to ensure there are enough threads in the common thread pool
  - Extra threads in the common fork-join pool are automatically terminated later
  - However, it's possible to saturate the CPU cores during bursty workloads

```
Image blockingDownload(URL url) {
  return BlockingTask
    .callInManagedBlocker
      (() -> downloadImage(url));
}
```



End of the Java Fork-Join Pool: Applying the Managed Blocker Interface