

Encapsulating the Java Fork-Join Framework's 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 how the common fork-join pool helps to maximize processor core utilization
- Recognize how the ManagedBlocker interface helps avoid starvation & improve performance
- Be able to apply the ManagedBlocker interface on blocking synchronizers & queues
- Know how to encapsulate ManagedBlocker & apply it on blocking I/O operations

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

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 callInManagedBlock(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 callInManagedBlock(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

```
public class BlockingTask {
```

```
...
```

```
public static<T> T callInManagedBlock(Supplier<T> supplier){
```

Create a helper object to encapsulate the 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 callInManagedBlock(Supplier<T> supplier){
```

```
        SupplierManagedBlocker<T> managedBlocker =  
            new SupplierManagedBlocker<T>(supplier);
```

```
        ...
```

```
        ForkJoinPool.managedBlock(managedBlocker);
```

```
        ...
```

```
        return managedBlocker.getResult();
```

```
    }
```

```
    ...
```

Submit managedBlocker to common ForkJoin thread pool

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

Encapsulating ManagedBlocker w/the BlockingTask Class

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

```
public class BlockingTask {  
    ...  
    public static<T> T callInManagedBlock(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;
```

*Keeps track of whether
blocking supplier is done*

```
        private SupplierManagedBlocker(final Supplier supplier)
```

```
        { mSupplier = supplier; }
```

```
        ...
```

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;
```

*Stores result obtained from
the supplier for later use*

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

```
        ...
```

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 BlockingTask

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  
        .callInManagedBlock  
            (() -> loadImage(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  
        .callInManagedBlock  
            (() -> loadImage(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  
        .callInManagedBlock  
          (() -> 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

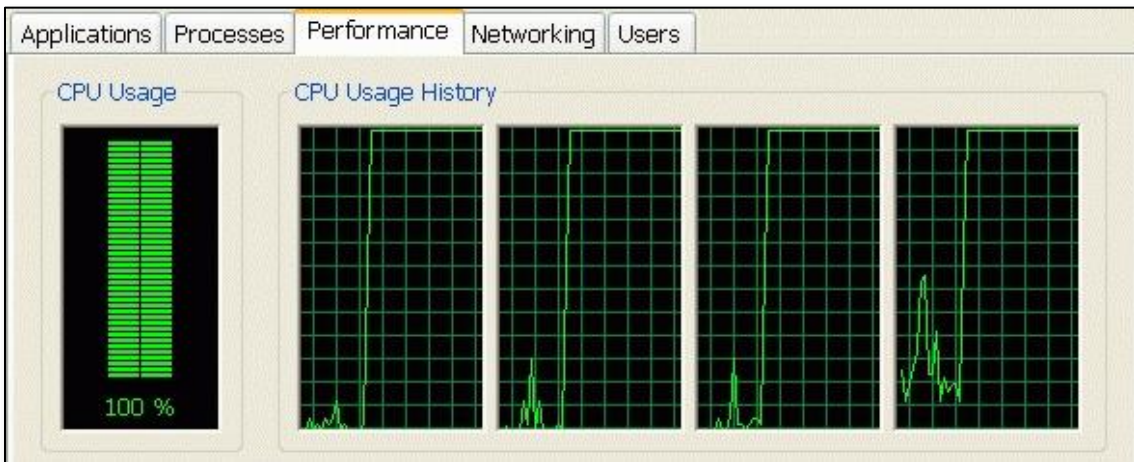
```
Image blockingDownload(URL url) {  
    return BlockingTask  
        .callInManagedBlock  
            (() -> loadImage(url));  
}
```



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  
        .callInManagedBlock  
            (() -> downloadImage(url));  
}
```



End of Encapsulating the Java Fork-Join Framework's Managed Blocker Interface