Benefits of Concurrency in Java & Android: Program Structure

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA
Learning Objectives in this Part of the Module

• Recognize how concurrency can improve performance in Java & Android
• Recognize how concurrency can improve responsiveness in Java & Android
• Recognize how concurrency can improve program structure in Java & Android
Overview of Event-Driven Programs
Overview of Event-Driven Programs

- Many software programs have historically been structured via a purely event-driven programming model
• Many software programs have historically been structured via a purely event-driven programming model
• The program flow is guided by events

See developer.android.com/guide/components/activities/activity-lifecycle.html
Overview of Event-Driven Programs

- Many software programs have historically been structured via a purely event-driven programming model
- The program flow is guided by events
  - e.g., user interface actions, sensor I/O, messages from elsewhere, etc.

```java
Button button = (Button) findViewById(R.id.loadButton);

button.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        ...
    }
});
```

A GUI component sending an event to its registered listener

GUI Component (e.g., a button)
Overview of Event-Driven Programs

• Many software programs have historically been structured via a purely event-driven programming model
  • The program flow is guided by events
  • Event-driven programming is a common model in GUIs that perform actions in response to user input

See developer.android.com/training/multiple-threads/communicate-ui.html
Limitations with Purely Event-Driven Programs
Limitations with Purely Event-Driven Programs

- For certain types of apps purely event-driven software is hard to understand & is inefficient
Limitations with Purely Event-Driven Programs

- For certain types of apps purely event-driven software is hard to understand & is inefficient
  - e.g., an app that downloads/displays images from web servers

![Diagram of event loop and message queue with UI thread (main thread)]
Limitations with Purely Event-Driven Programs

• For certain types of apps purely event-driven software is hard to understand & is inefficient

• e.g., an app that downloads/displays images from web servers

To avoid starvation of other tasks, long-duration blocking operations are disallowed in the UI thread!

Moreover, there’s just a single thread, i.e., the UI thread!
Limitations with Purely Event-Driven Programs

• For certain types of apps purely event-driven software is hard to understand & is inefficient
• e.g., an app that downloads/displays images from web servers

Long-duration operations are run asynchronously by posting messages in the message queue of the UI thread’s event loop
Limitations with Purely Event-Driven Programs

- For certain types of apps purely event-driven software is hard to understand & is inefficient
- e.g., an app that downloads/displays images from web servers

\[\text{e.g., a message requesting an asynchronous read of a socket could be posted to the UI thread’s event loop}\]
Limitations with Purely Event-Driven Programs

- For certain types of apps purely event-driven software is hard to understand & is inefficient
  - e.g., an app that downloads/displays images from web servers

Asynchronous operation completion is handled later via messages in the UI thread’s event loop
Limitations with Purely Event-Driven Programs

- For certain types of apps purely event-driven software is hard to understand & is inefficient
- e.g., an app that downloads/displays images from web servers

*Message Queue*

- Event Loop
- Message
- Message
- Message
- Message
- Message

UI Thread (main thread)

*Message*

*e.g., the completion of the async read indicates how many bytes were read in from the socket*
Limitations with Purely Event-Driven Programs

- For certain types of apps purely event-driven software is hard to understand & is inefficient
- e.g., an app that downloads/dispays images from web servers

See en.wikipedia.org/wiki/Lather,_rinse,_repeat
Limitations with Purely Event-Driven Programs

- For certain types of apps purely event-driven software is hard to understand & is inefficient
  - e.g., an app that downloads/displays images from web servers

  These steps may be repeated multiple times until the image is completely downloaded

Note the disconnect in both time & space between operation invocation & operation completion
Limitations with Purely Event-Driven Programs

- Poorly structured event-driven software often leads to “spaghetti code” or a “big ball of mud”

Limitations with Purely Event-Driven Programs

• Poorly structured event-driven software often leads to “spaghetti code” or a “big ball of mud”

• This type of code is hard to understand since there’s little/no structure or linear flow of control to guide developers
Limitations with Purely Event-Driven Programs

• Poorly structured event-driven software often leads to “spaghetti code” or a “big ball of mud”

• This type of code is hard to understand since there’s little/no structure or linear flow of control to guide developers

• Single-threaded programs that are driven purely by events can obscure the flow of control in time & space
Limitations with Purely Event-Driven Programs

- Poorly structured event-driven software often leads to “spaghetti code” or a “big ball of mud”
- This type of code is hard to understand since there’s little/no structure or linear flow of control to guide developers
- Single-threaded programs that are driven purely by events can obscure the flow of control in time & space

It’s hard to understand software when its flow of control “bounces” all over the place
Limitations with Purely Event-Driven Programs

- Poorly structured event-driven software often leads to “spaghetti code” or a “big ball of mud”
- This type of code is hard to understand since there’s little/no structure or linear flow of control to guide developers
  - Single-threaded programs that are driven purely by events can obscure the flow of control in time & space
  - It’s also hard to sustain since small changes can break nearly anything due to non-intuitive dependencies
Simplifying Program Structure with Concurrency
Simplifying Program Structure with Concurrency

- Concurrency mechanisms & frameworks can help overcome drawbacks with purely event-driven programs.
Simplifying Program Structure with Concurrency

- Concurrency mechanisms & frameworks can help overcome drawbacks with purely event-driven programs.
- Multi-threading enables more intuitive (& often performant) ways to structure software.
Simplifying Program Structure with Concurrency

- Concurrency mechanisms & frameworks can help overcome drawbacks with purely event-driven programs
- Multi-threading enables more intuitive (& often performant) ways to structure software
  - e.g., operations that run for long-durations can block synchronously
Concurrent mechanisms & frameworks can help overcome drawbacks with purely event-driven programs.

Multi-threading enables more intuitive (and often performant) ways to structure software.

For example, operations that run for long durations can block synchronously.

Synchronous blocking within a thread can often yield a more natural & collaborative control flow.
Simplifying Program Structure with Concurrency

- Android's HaMeR concurrency framework can help simplify program structure

See next part of this lesson for an Android example for this code
Simplifying Program Structure with Concurrency

- Android’s HaMeR concurrency framework can help simplify program structure

Operations that run for long-durations can block in threads
Simplifying Program Structure with Concurrency

- Android’s HaMeR concurrency framework can help simplify program structure.
Simplifying Program Structure with Concurrency

- Android’s HaMeR concurrency framework can help simplify program structure

Android’s HaMeR concurrency framework supports a “hybrid” programming model, i.e., part concurrent (half-sync) & part event-driven (half-async)

See www.dre.vanderbilt.edu/~schmidt/PDF/HS-HA.pdf
Simplifying Program Structure with Concurrency

- Android’s HaMeR concurrency framework can help simplify program structure

No more spaghetti code or big ball of mud since the app is neatly organized in a more “linear” structure that doesn’t bounce around!
Simplifying Program Structure with Concurrency

- Android’s HaMeR concurrency framework can help simplify program structure
- This framework implements various POSA concurrency patterns
  - e.g., Command Processor, Active Object, Half-Sync/Half-Async

See www.dre.vanderbilt.edu/~schmidt/POSA & en.wikipedia.org/wiki/Concurrency_pattern
End of Benefits of Concurrency in Java & Android (Part 3)
Discussion Questions

1. Which of the following are problems with poorly structured event-driven programs?
   
a. It’s performance & responsiveness are poor since there is little/no structure or linear flow of control to guide developers
   
b. It’s hard to understand since there’s little/no structure or linear flow of control to guide developers
   
c. It’s hard to sustain since small changes can break nearly anything due to non-intuitive dependencies
   
d. Synchronous blocking in threads can often yield a less natural & collaborative control flow
2. How does concurrency help to simplify program structure?
   a. *It provides a disconnect in both time & space between operation invocation & operation completion*
   b. *Concurrent programs can obscure the flow of control in both time & space*
   c. *Synchronous blocking within a thread can often yield a more natural & collaborative control flow*
   d. *Synchronous blocking within a thread can often yield a less natural & collaborative control flow*
Benefits of Concurrency in Android: App Case Study

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA
Learning Objectives in this Part of the Module

- Recognize how concurrency can improve performance in Java & Android
- Recognize how concurrency can improve responsiveness in Java & Android
- Recognize how concurrency can improve program structure in Java & Android
- Know how concurrency benefits can be applied to avoid overly complex & tangled event-driven solutions in Java & Android

E.g., an app that concurrently downloads an image from a remote web server & displays it to the user
Analyzing the Image Download App
Analyzing the Image Download App

• This app downloads images from remote web servers & displays them.

Analyzing the Image Download App

- This app downloads images from remote web servers & displays them
- Uses a Java thread with a runnable implementation to download images

Allows image downloading to take advantage of multi-core processors
Analyzing the Image Download App

• This app downloads images from remote web servers & displays them
• Uses a Java thread with a runnable implementation to download images
• The following lambda expression specifies what work the thread should run

```java
new Thread(() -> {
    /* Code to run goes here */
}).start();
```

The Java 8 version using lambda expressions is very concise since it omits all unnecessary syntax!
Analyzing the Image Download App

- This app downloads images from remote web servers & displays them
- Uses a Java thread with a runnable implementation to download images
  - The following lambda expression specifies what work the thread should run

  ```java
  new Thread(() -> {
      /* Code to run goes here */
  }).start();
  ```

- The Java Thread class creates & controls the unit of execution that processes the lambda expression

See [docs.oracle.com/javase/tutorial/essential/concurrency/runThread.html](http://docs.oracle.com/javase/tutorial/essential/concurrency/runThread.html)
Analyzing the Image Download App

- This app downloads images from remote web servers & displays them
- Uses a Java thread with a runnable implementation to download images

See upcoming lesson on “Overview of Java Threads”
Analyzing the Image Download App

• This app downloads images from remote web servers & displays them
  • Uses a Java thread with a runnable implementation to download images
  • Uses Android “HaMeR” concurrency framework to pass results to UI thread

See code.tutsplus.com/tutorials/concurrency-on-android-using-hamer-framework--cms-27129
Analyzing the Image Download App

- This app downloads images from remote web servers & displays them
- Uses a Java thread with a runnable implementation to download images
- Uses Android “HaMeR” concurrency framework to pass results to UI thread

Downloading of images is done in background threads.
Analyzing the Image Download App

- This app downloads images from remote web servers & displays them
- Uses a Java thread with a runnable implementation to download images
- Uses Android “HaMeR” concurrency framework to pass results to UI thread

Displaying of images is done in the UI thread
Analyzing the Image Download App

- This app downloads images from remote web servers & displays them
- Uses a Java thread with a runnable implementation to download images
- Uses Android “HaMeR” concurrency framework to pass results to UI thread
- This solution intentionally doesn’t handle interrupts or runtime configuration changes

See upcoming lesson on “Managing the Java Thread Lifecycle”
Analyzing the Image Download App

- As a reminder, the purely event-driven solution was *not* cohesive

Note disconnect between operation invocation & operation completion in both time & space

See the previous part of this lesson for details
public void downloadImage(View view) {
    Uri url = getUrl();
    showDialog("downloading via HaMeR");
    startDownload(url);
    ...
The structure of this concurrent app is more cohesive in time & space.

```java
public void downloadImage(View view) {
    Uri url = getUrl();
    showDialog("downloading via HaMeR");
    startDownload(url);
    ...
}
```

Get the URL entered by user.
public void downloadImage(View view) {
    Uri url = getUrl();
    showDialog("downloading via HaMeR");
    startDownload(url);
    ...
The structure of this concurrent app is more cohesive in time & space.

... public void downloadImage(View view) {
    ...  
    Uri url = getUrl();
    ...  
    showDialog("downloading via HaMeR");

    startDownload(url);
    ...

    **Initiate downloading of the image at the url**
Analyzing the Image Download App

• The structure of this concurrent app is more cohesive in time & space

... Performs the image download

```java
public void startDownload(Uri url) {
    new Thread(() -> {
        Uri pathname =
            downloadImage(this, url);

        runOnUIThread(() ->
            displayImage(pathname));
    }).start();
...```

http://www.dre.vanderbilt.edu/~schmidt/ka.png
Analyzing the Image Download App

• The structure of this concurrent app is more cohesive in time & space

... public void startDownload(Uri url) {
    new Thread(() -> {
        Uri pathname =
            downloadImage(this, url);

        runOnUIThread(() ->
            displayImage(pathname));
    }).start();
...

Define a lambda expression
Analyzing the Image Download App

• The structure of this concurrent app is more cohesive in time & space

```java
public void startDownload(Uri url) {
    new Thread(() -> {
        Uri pathname =
            downloadImage(this, url);

        runOnUiThread(() ->
            displayImage(pathname));
    }).start();

    // ... Start a new thread
```

See developer.android.com/guide/components/processes-and-threads.html#WorkerThreads
Analyzing the Image Download App

• The structure of this concurrent app is more cohesive in time & space

...
Create command to run in UI thread (non-blocking)

```java
public void startDownload(Uri url) {
    new Thread(() -> {
        Uri pathname =
        downloadImage(this, url);
        runOnUiThread(() ->
        displayImage(pathname));
    }).start();
    ...
    ...
```

See [www.dre.vanderbilt.edu/~schmidt/CommandProcessor.pdf](http://www.dre.vanderbilt.edu/~schmidt/CommandProcessor.pdf)
Analyzing the Image Download App

- The structure of this concurrent app is more cohesive in time & space

```java
public void startDownload(Uri url) {
    new Thread(() -> {
        Uri pathname =
            downloadImage(this, url);

        runOnUiThread(() ->
            displayImage(pathname));
    }).start();

    // Display image in UI thread (non-blocking)
}
```
Analyzing the Image Download App

• The structure of this concurrent app is more cohesive in time & space

... 

```java
public void startDownload(Uri url) {
    new Thread(() -> {
        Uri pathname =
            downloadImage(this, url);

        runOnUIThread(() ->
            displayImage(pathname));
    }).start();
...```

These methods run in different threads
Analyzing the Image Download App

- The structure of this concurrent app is more cohesive in time & space

... public void startDownload(Uri url) {
    new Thread(() -> {
        Uri pathname =
            downloadImage(this, url);

        runOnUIThread(() ->
            displayImage(pathname));
    }).start();
...

The Android HaMeR-based version of this code is easier to understand than the purely event-driven version since it doesn’t “bounce around” in time & space!
End of Benefits of Concurrency in Java & Android (Part 4)
Discussion Questions

1. Which of the following are reasons why the concurrent version of the app is better *structured* than the purely event-driven version?

   a. It can take advantage of Java 8 programming language features, such as lambda expressions

   b. It can take advantage of multi-core processors to run downloads in parallel

   c. It is more cohesive in time & space since the code that downloads an image can block synchronously

   d. It is more cohesive in time & space since the code that downloads an image must run asynchronously without blocking