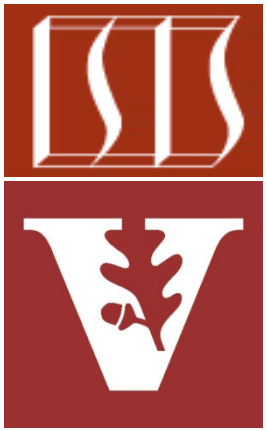
Android Concurrency: Motivations for Concurrency



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

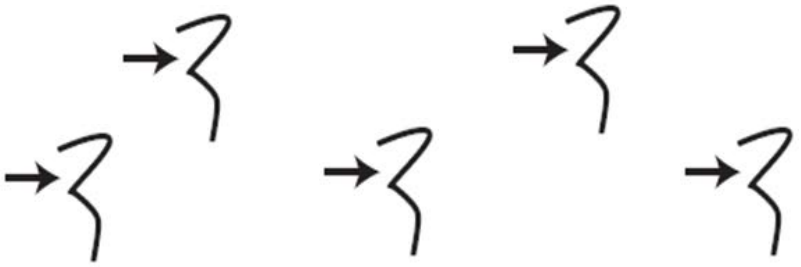
Institute for Software
Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA

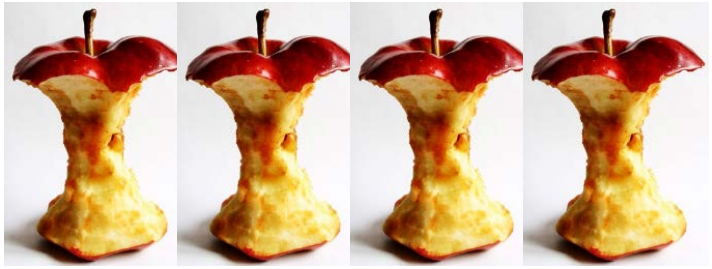
Learning Objectives in this Part of the Module

- Explore key motivations for developing concurrent mobile device software

Motivations for Concurrent Software

- Leverage hardware/software advances...
  - Multi-core processors

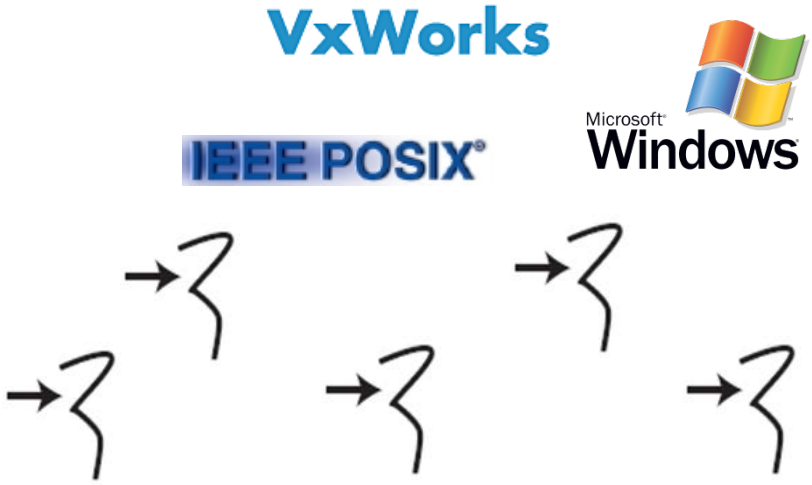


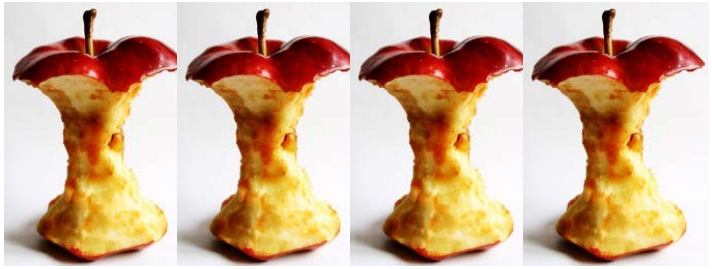


www.androidauthority.com/tag/quad-core-phones has more info

Motivations for Concurrent Software

- Leverage hardware/software advances...
  - Multi-core processors
  - Multi-threaded operating systems

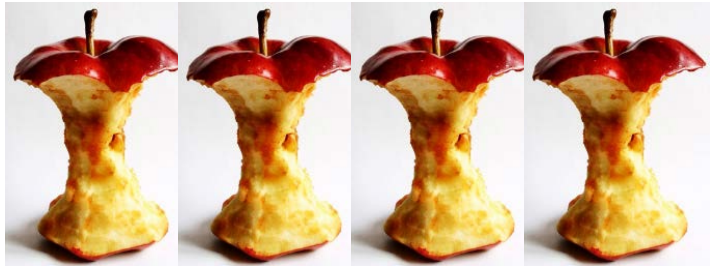




en.wikipedia.org/wiki/Thread_(computing) has more info on threads

Motivations for Concurrent Software

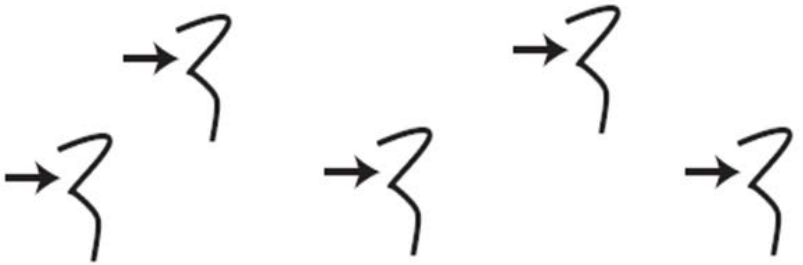
- Leverage hardware/software advances
  - Multi-core processors
  - Multi-threaded operating systems
  - Multi-threaded middleware

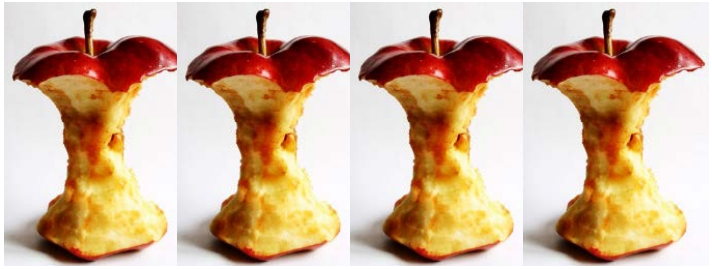


www.dre.vanderbilt.edu/~schmidt/concurrency-patterns.html has more info

Motivations for Concurrent Software

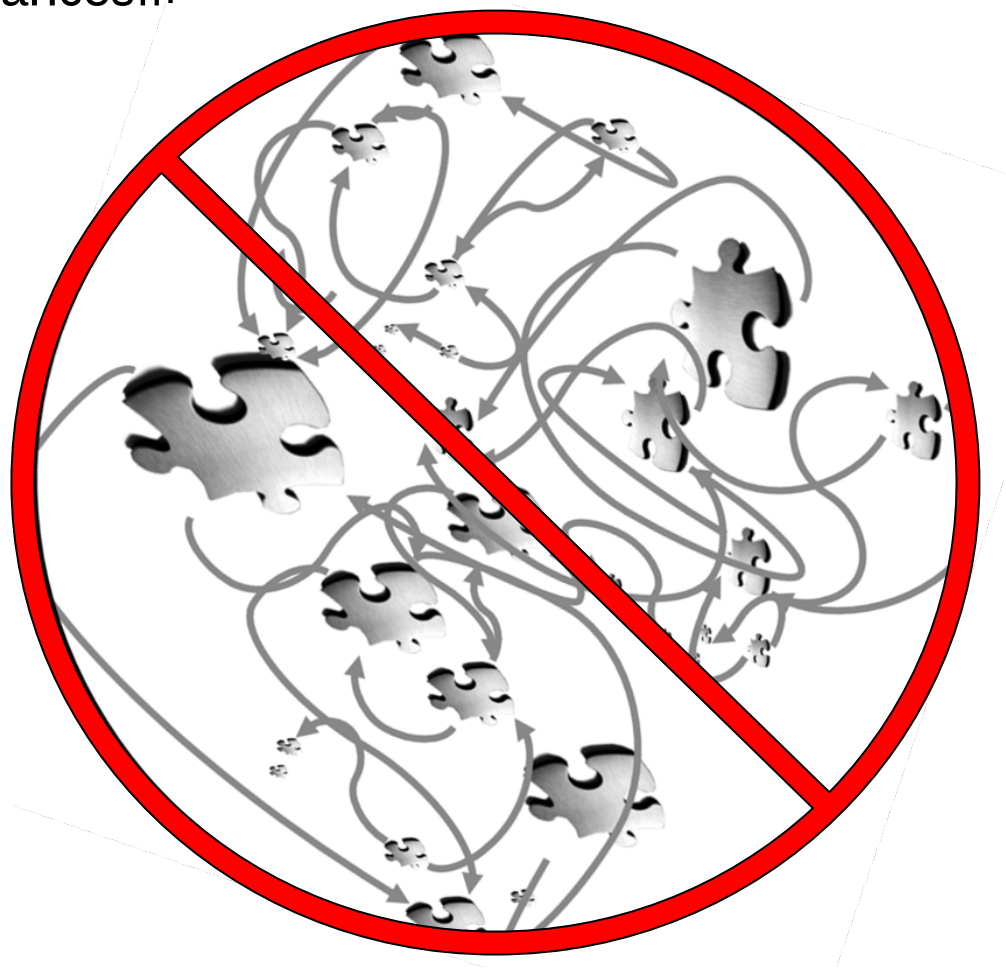
- Leverage hardware/software advances...
  - Multi-core processors
  - Multi-threaded operating systems
  - Multi-threaded middleware





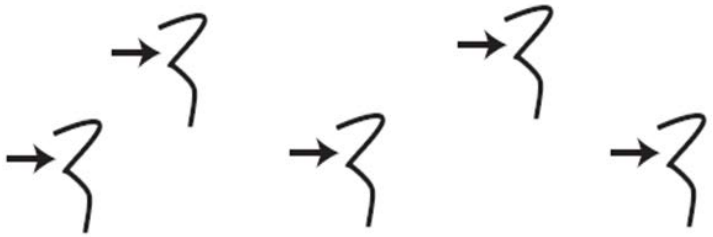
Motivations for Concurrent Software

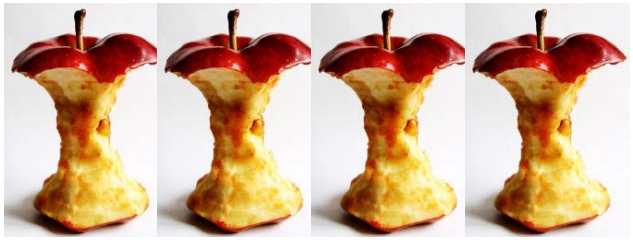
- Leverage hardware/software advances...
- ...to improve software quality attributes, e.g.
  - Simplify program structure



Motivations for Concurrent Software

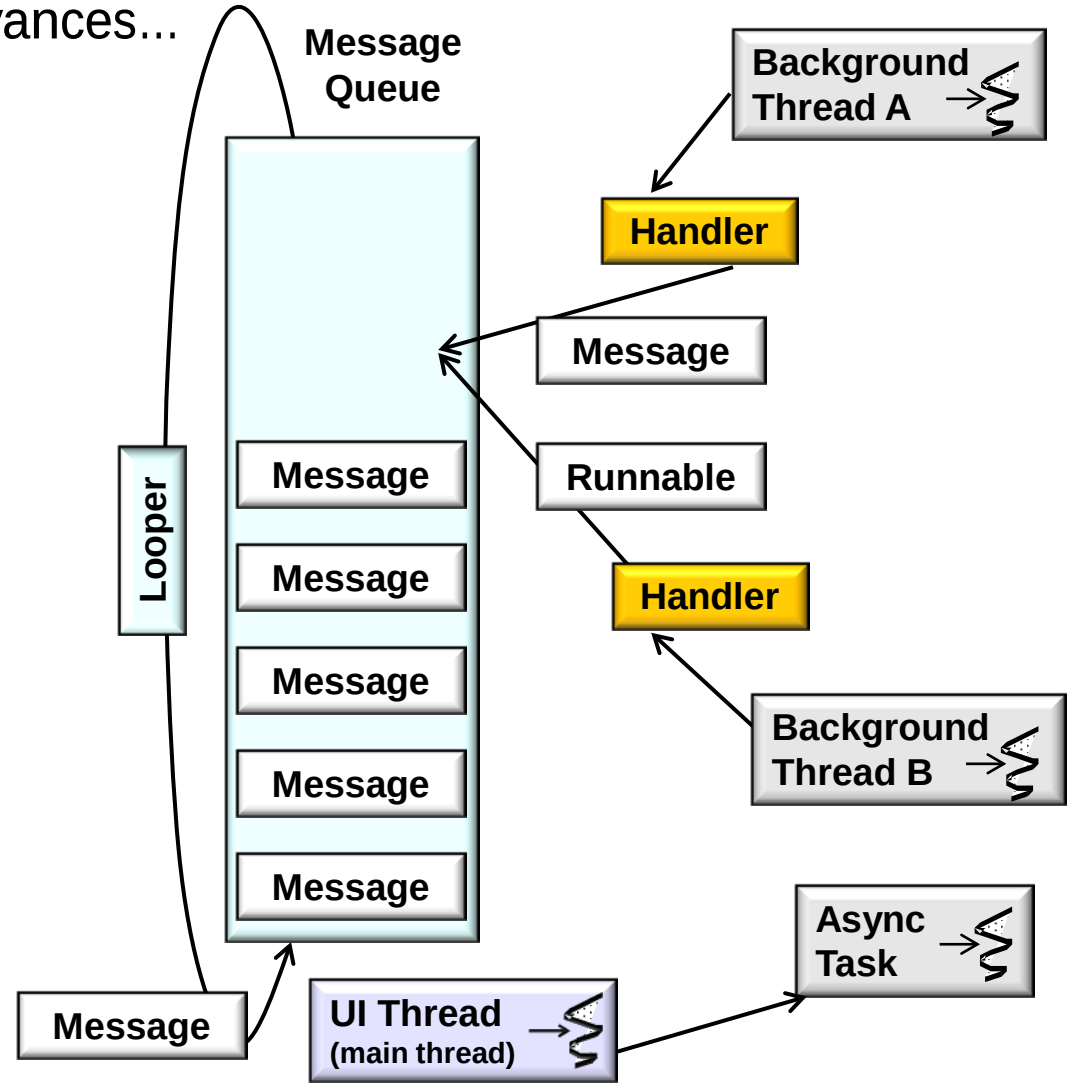
- Leverage hardware/software advances...
- ...to improve software quality attributes, e.g.
  - Simplify program structure
  - Increase performance





Motivations for Concurrent Software

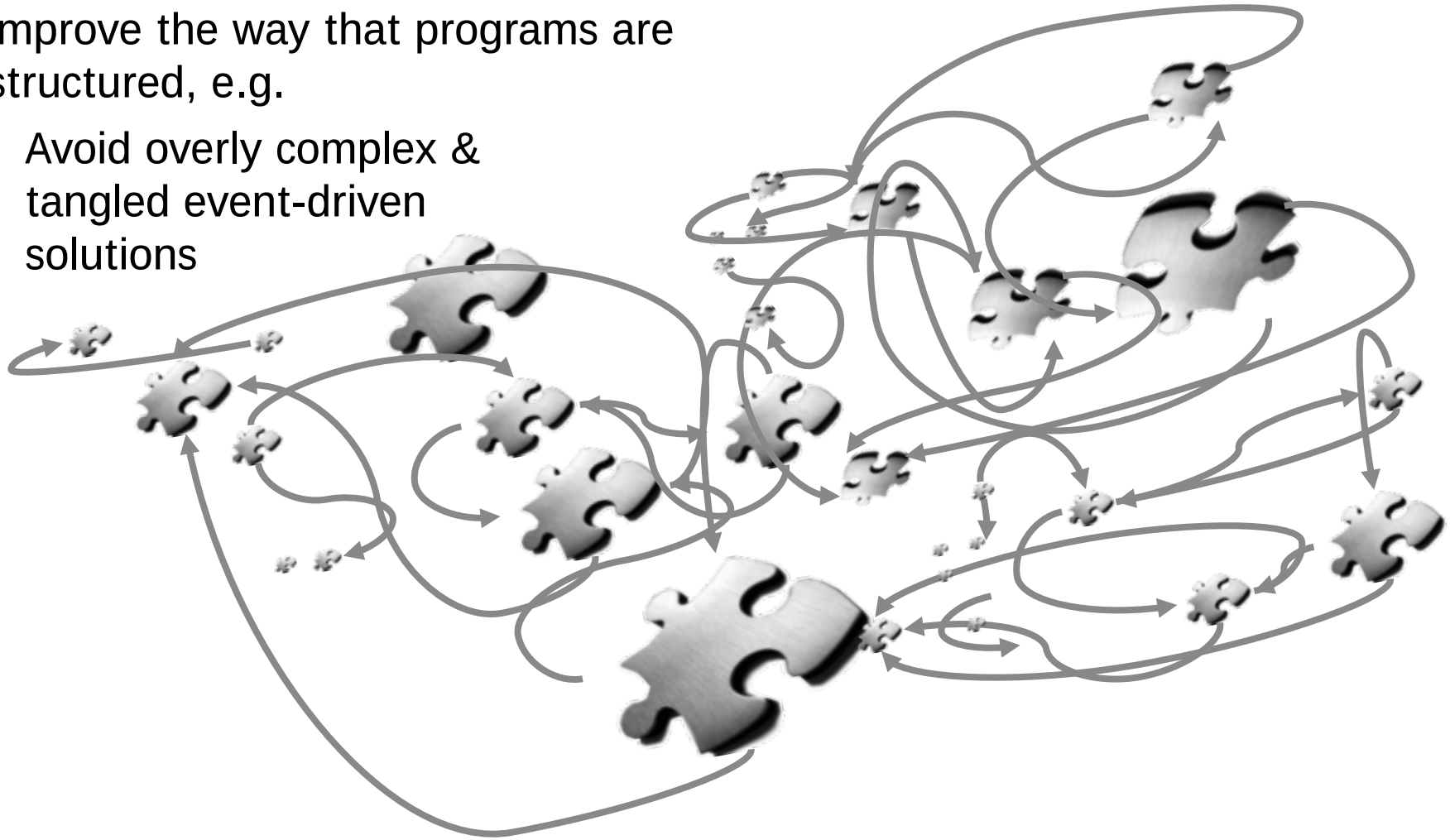
- Leverage hardware/software advances...
- ...to improve software quality attributes, e.g.
  - Simplify program structure
  - Increase performance
  - Improve responsiveness



Using Concurrency to Simplify Program Structure

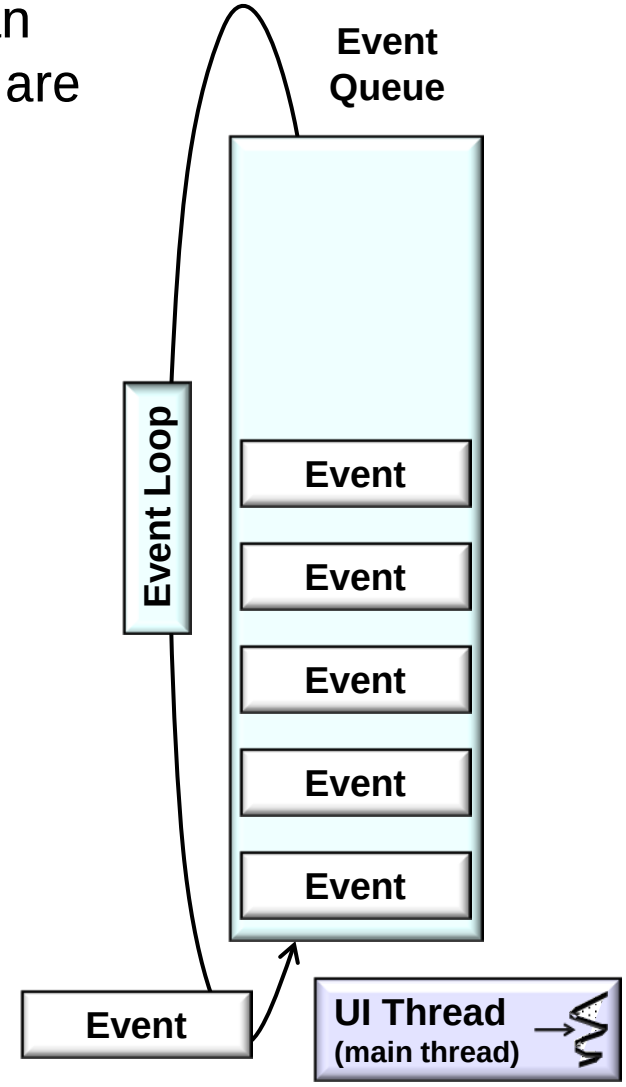
Simplify Program Structure

- Hardware/software advances can improve the way that programs are structured, e.g.
  - Avoid overly complex & tangled event-driven solutions



Simplify Program Structure

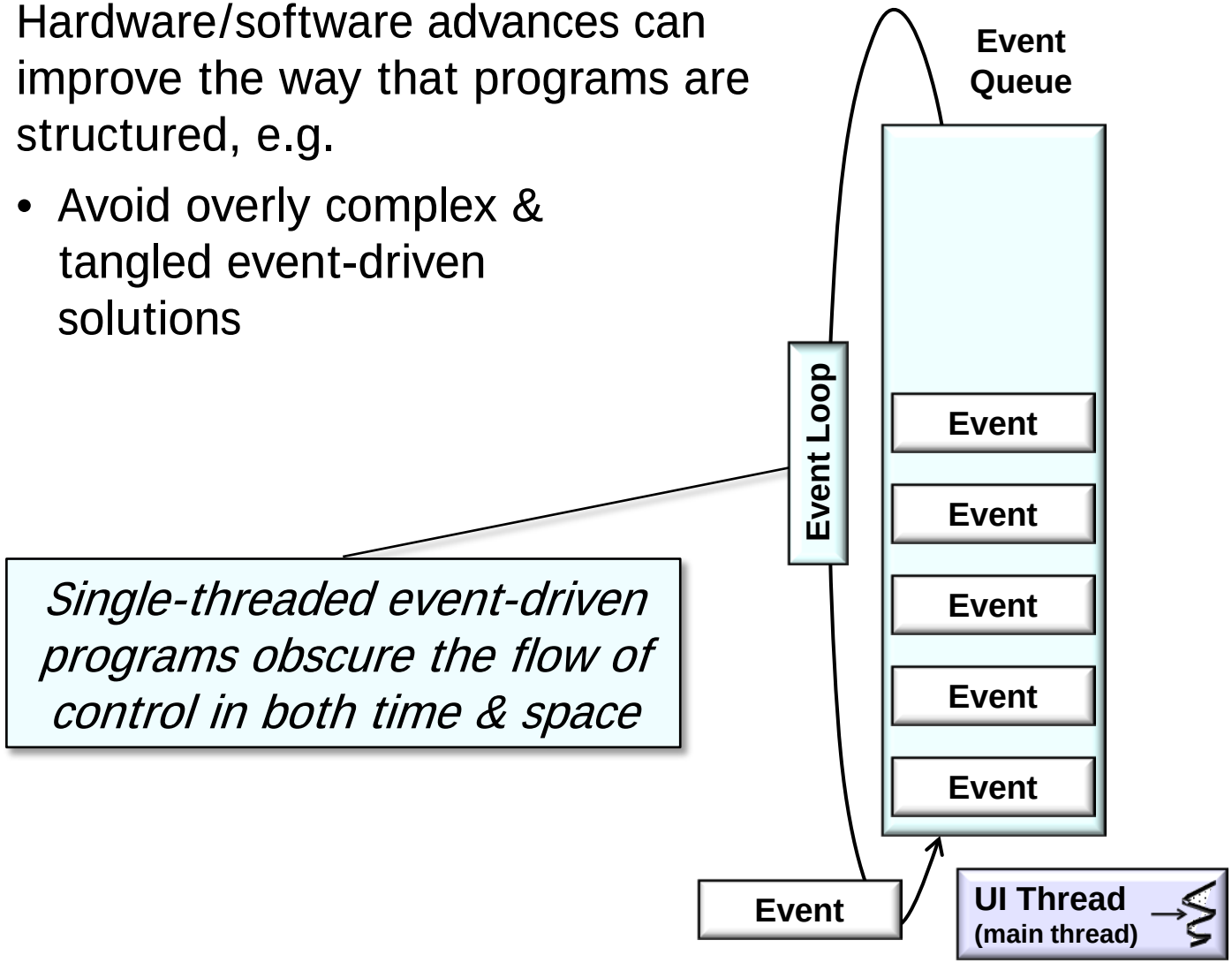
- Hardware/software advances can improve the way that programs are structured, e.g.
  - Avoid overly complex & tangled event-driven solutions



en.wikipedia.org/wiki/Event-driven_programming has more discussion

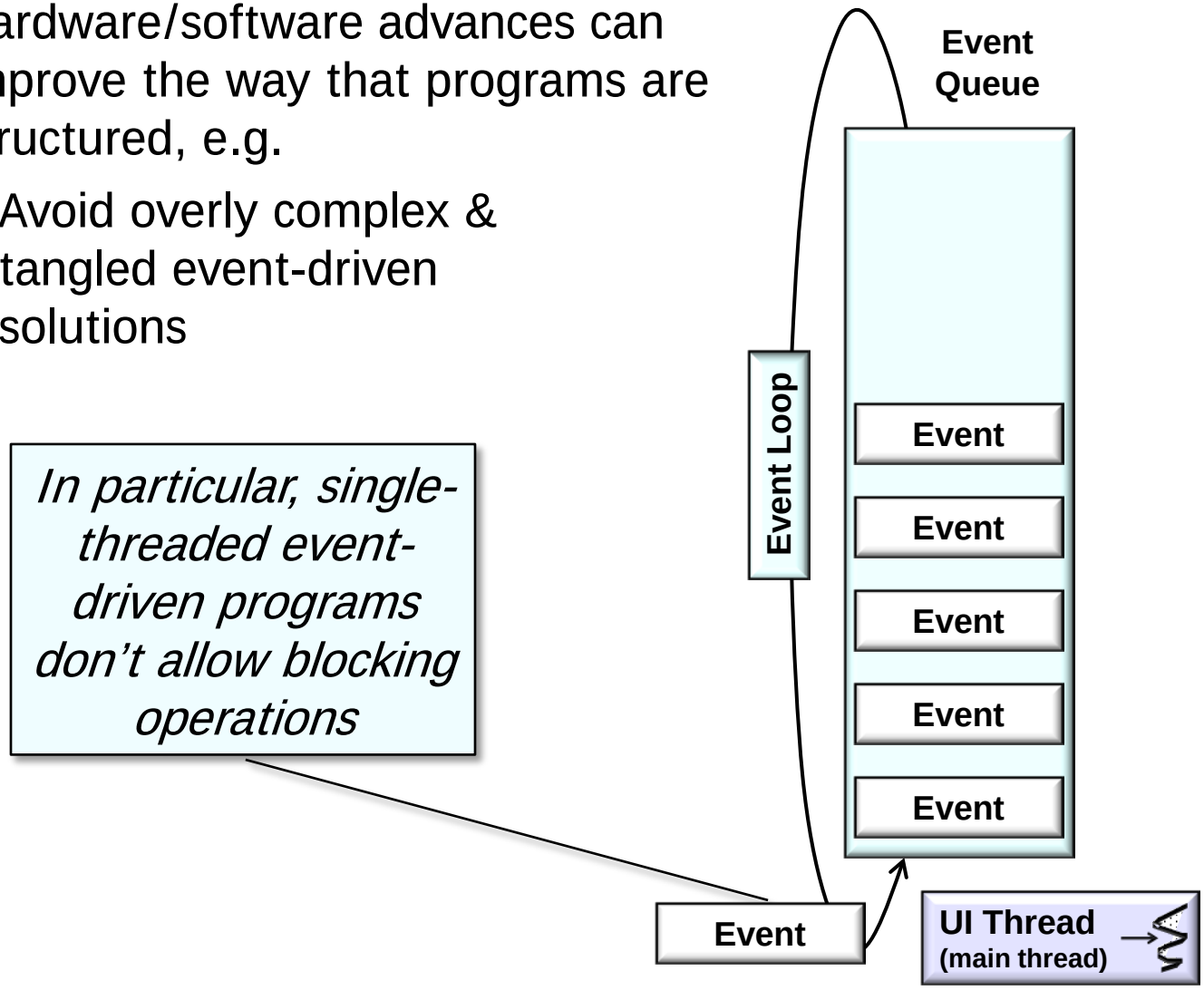
Simplify Program Structure

- Hardware/software advances can improve the way that programs are structured, e.g.
  - Avoid overly complex & tangled event-driven solutions



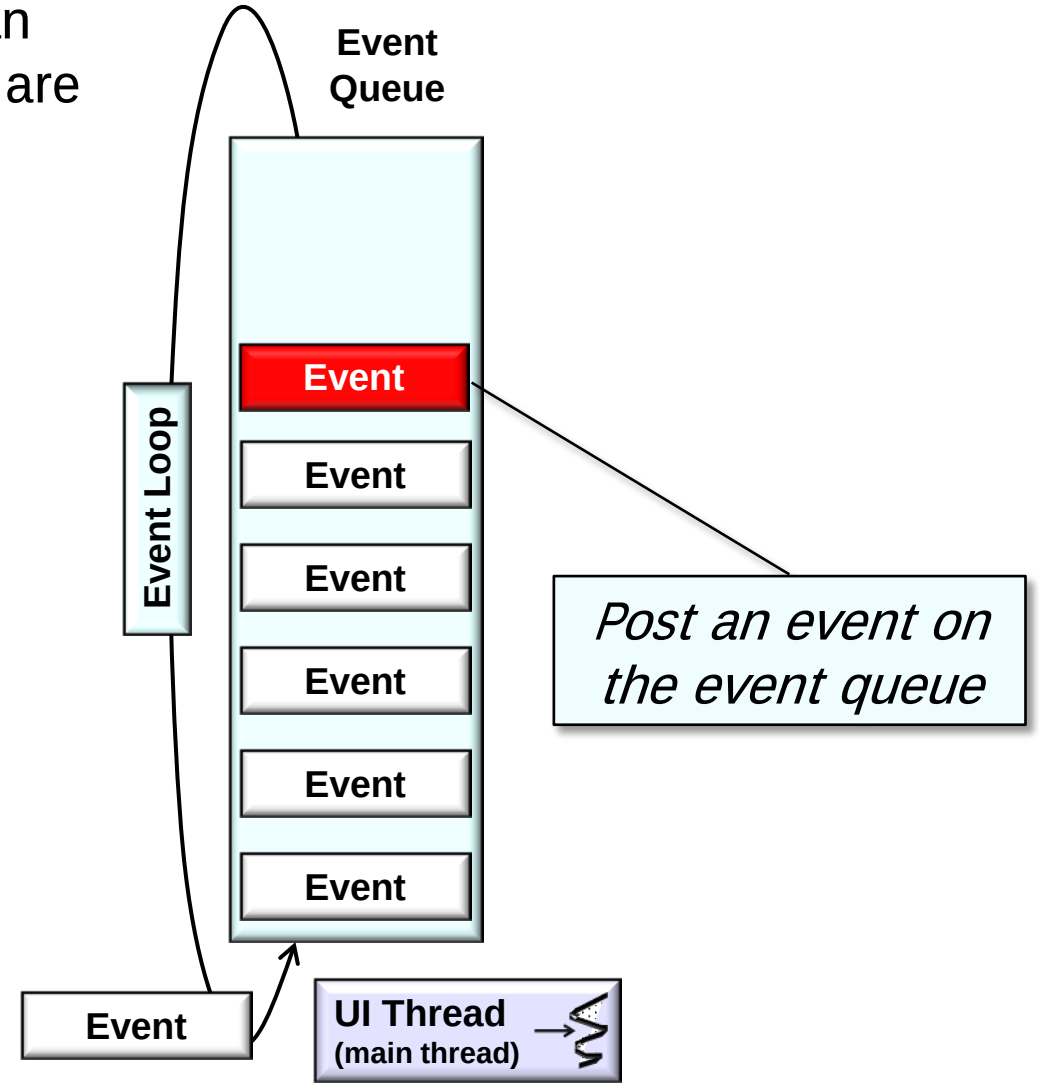
Simplify Program Structure

- Hardware/software advances can improve the way that programs are structured, e.g.
  - Avoid overly complex & tangled event-driven solutions



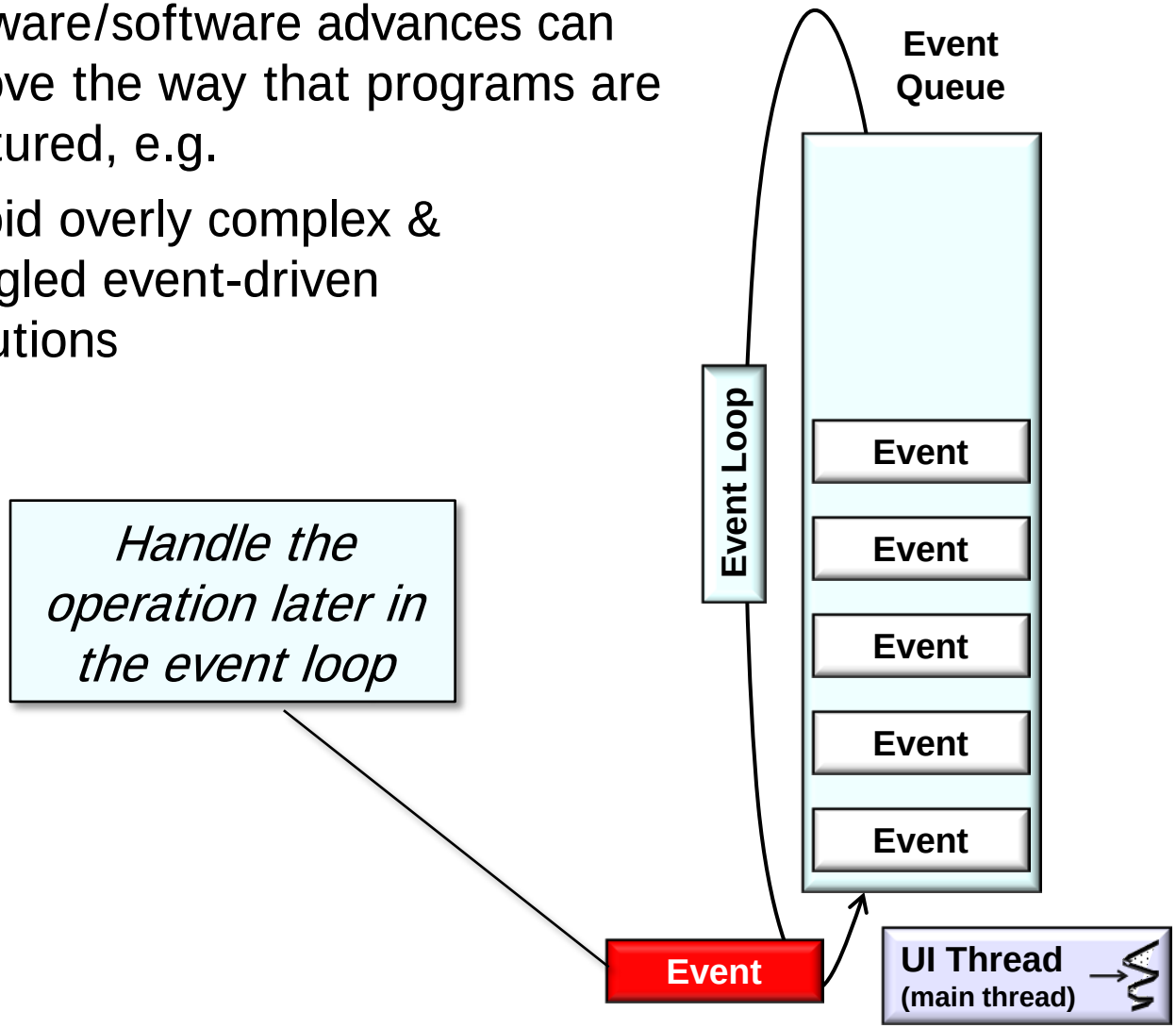
Simplify Program Structure

- Hardware/software advances can improve the way that programs are structured, e.g.
  - Avoid overly complex & tangled event-driven solutions



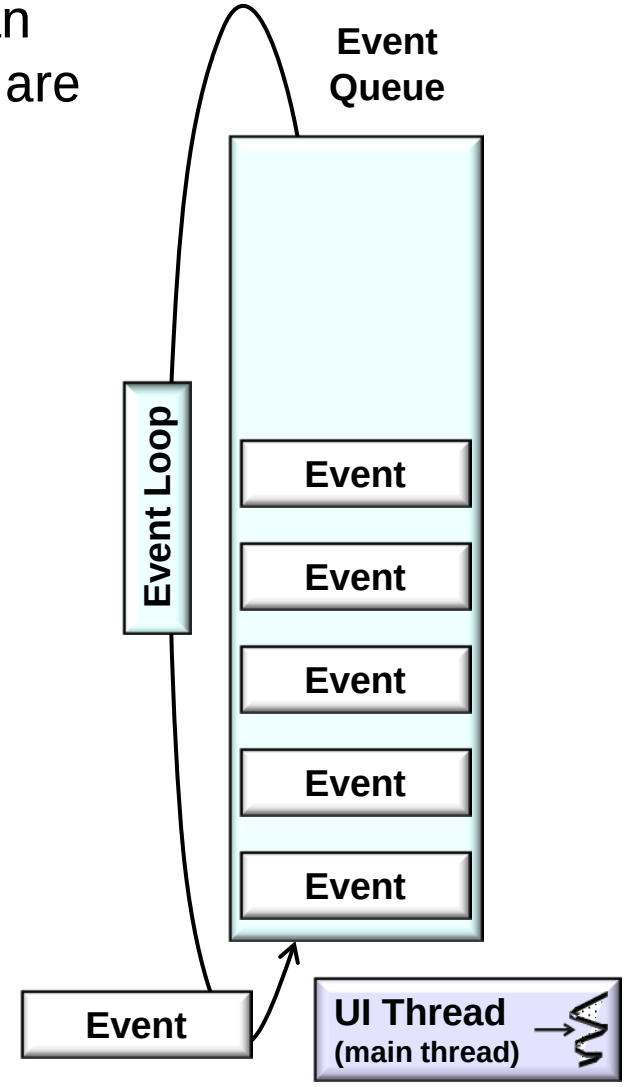
Simplify Program Structure

- Hardware/software advances can improve the way that programs are structured, e.g.
  - Avoid overly complex & tangled event-driven solutions



Simplify Program Structure

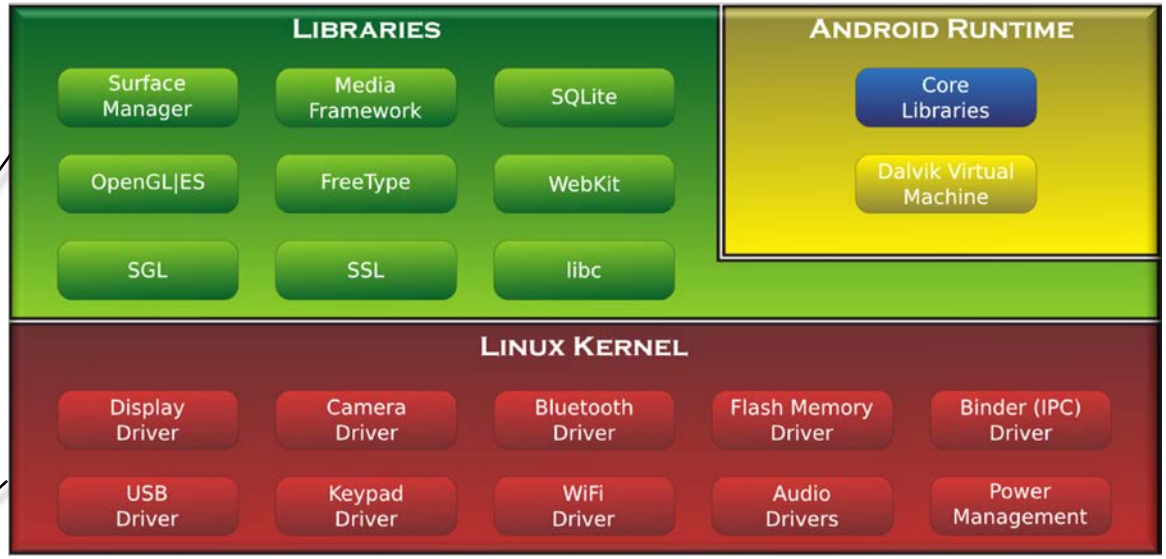
- Hardware/software advances can improve the way that programs are structured, e.g.
  - Avoid overly complex & tangled event-driven solutions



See www.dre.vanderbilt.edu/~schmidt/PDF/Reactor.pdf for more on *Reactor*

Simplify Program Structure

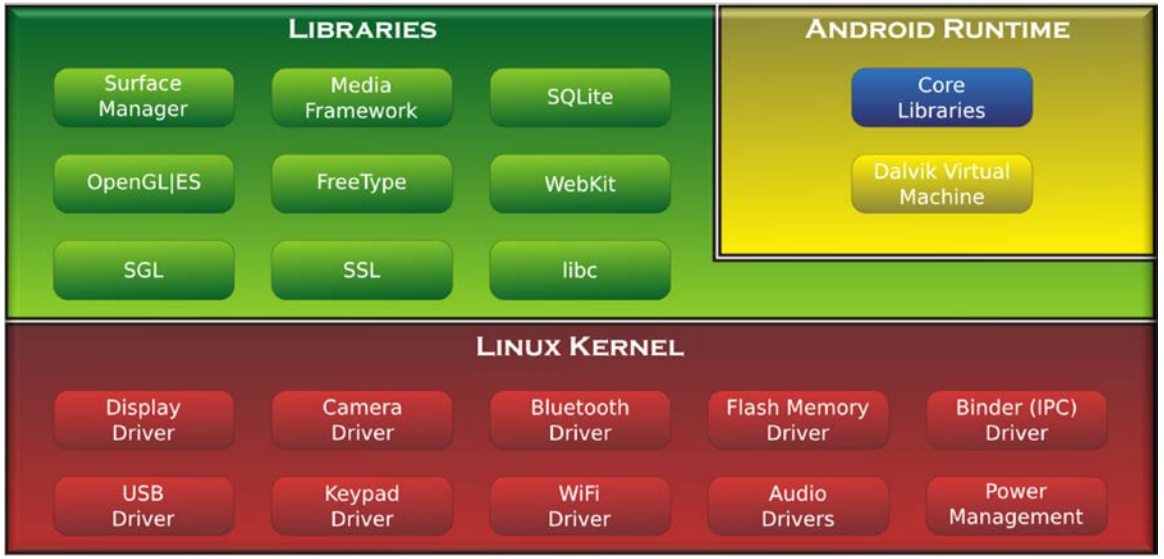
- Hardware/software advances can improve the way that programs are structured, e.g.
  - Avoid overly complex & tangled event-driven solutions

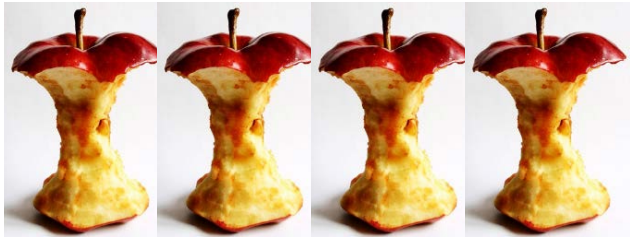


Modern middleware, operating systems, & hardware has better concurrency support

Simplify Program Structure

- Hardware/software advances can improve the way that programs are structured, e.g.
  - Avoid overly complex & tangled event-driven solutions
    - Enables blocking operations that "detangle" event-driven programs

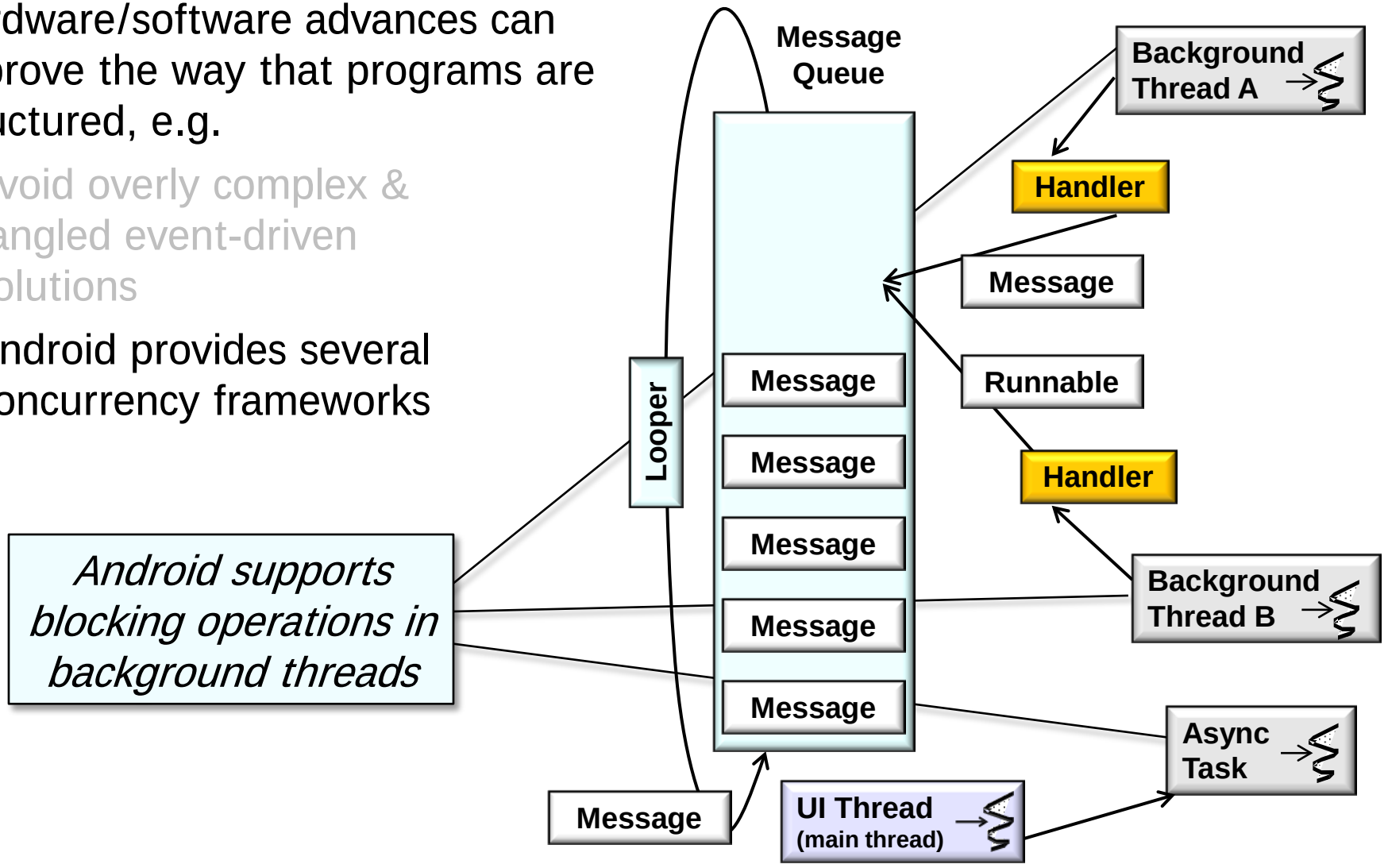




en.wikipedia.org/wiki/Multithreading_(computer_architecture) has more info

Simplify Program Structure

- Hardware/software advances can improve the way that programs are structured, e.g.
  - Avoid overly complex & tangled event-driven solutions
  - Android provides several concurrency frameworks



en.wikipedia.org/wiki/Concurrency_pattern has more on concurrency patterns

Simplify Program Structure

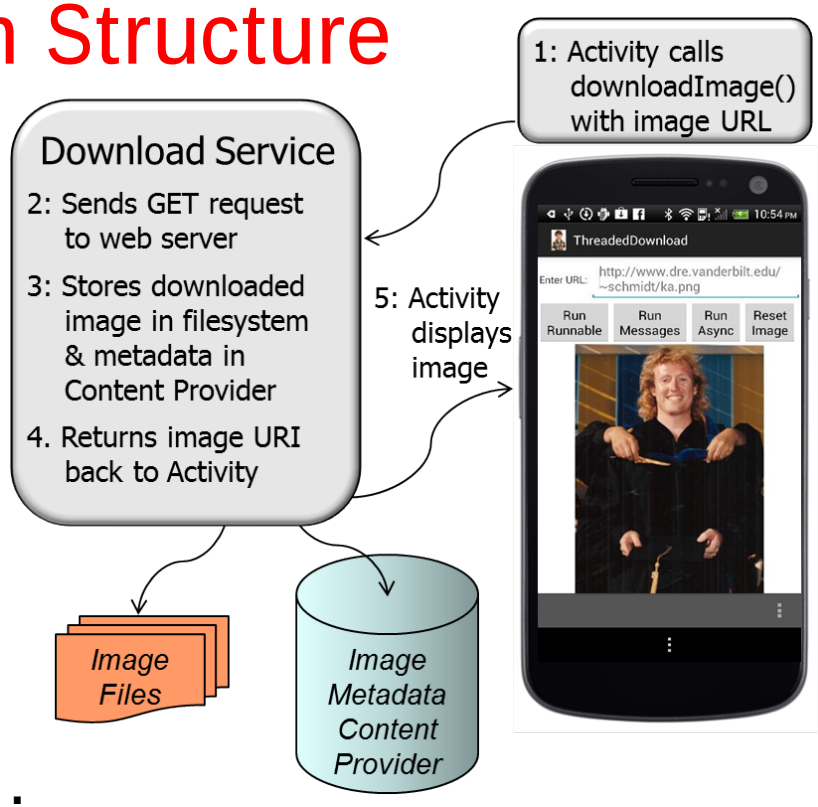
- Hardware/software advances can improve the way that programs are structured, e.g.

```
private Bitmap bitmap;
final ImageView iview = ...
final Button button = ...
button.setOnClickListener
   (new OnClickListener() {
  public void onClick(View v) {
    new Thread(new Runnable() {
      public void run() {
        bitmap = downloadImage(URI);
        iview.post(new Runnable() {
        public void run()
        { iview.setImageBitmap(bitmap); }
      });
    }
  }).start();
```

Simplify Program Structure

- Hardware/software advances can improve the way that programs are structured, e.g.

```
private Bitmap bitmap;
final ImageView iview = ...
final Button button = ...
button.setOnClickListener
   (new OnClickListener() {
  public void onClick(View v) {
    new Thread(new Runnable() {
      public void run() {
        bitmap = downloadImage(URI);
        iview.post(new Runnable() {
        public void run()
        { iview.setImageBitmap(bitmap); }
      });
    }
  }).start();
```



Multi-threaded Android app that downloads an image

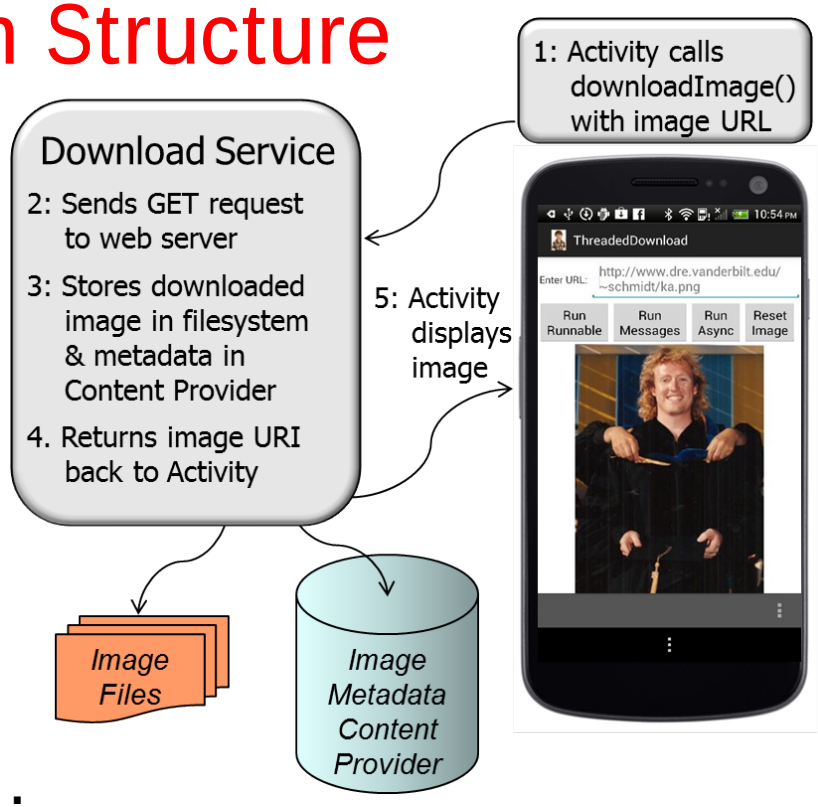
See www.dre.vanderbilt.edu/~schmidt/PDF/HS-HA.pdf for *Half-Sync/Half-Async*

Simplify Program Structure

- Hardware/software advances can improve the way that programs are structured, e.g.

```
private Bitmap bitmap;
final ImageView iview = ...
final Button button = ...
button.setOnClickListener
   (new OnClickListener() {
  public void onClick(View v) {
    new Thread(new Runnable() {
      public void run() {
        bitmap = downloadImage(URI);
        iview.post(new Runnable() {
        public void run()
        { iview.setImageBitmap(bitmap); }
      });
    }
  }).start();
```

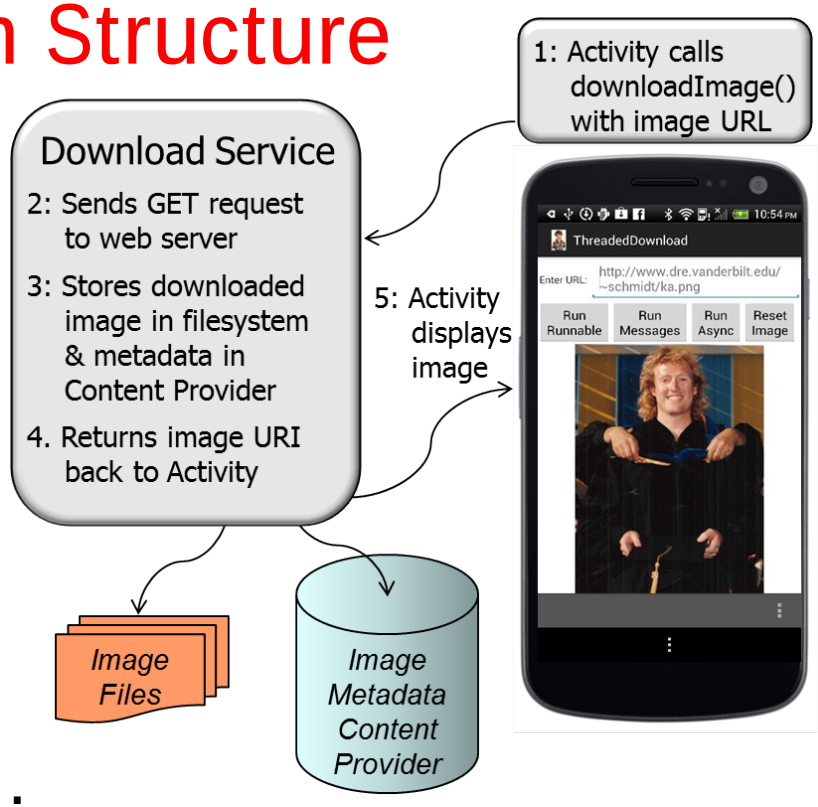
Handles button clicks



Simplify Program Structure

- Hardware/software advances can improve the way that programs are structured, e.g.

```
private Bitmap bitmap;
final ImageView iview = ...
final Button button = ...
button.setOnClickListener
   (new OnClickListener() {
  public void onClick(View v) {
    new Thread(new Runnable() {
      public void run() {
        bitmap = downloadImage(URI);
        iview.post(new Runnable() {
        public void run()
        { iview.setImageBitmap(bitmap); }
      });
    }
  }).start();
```

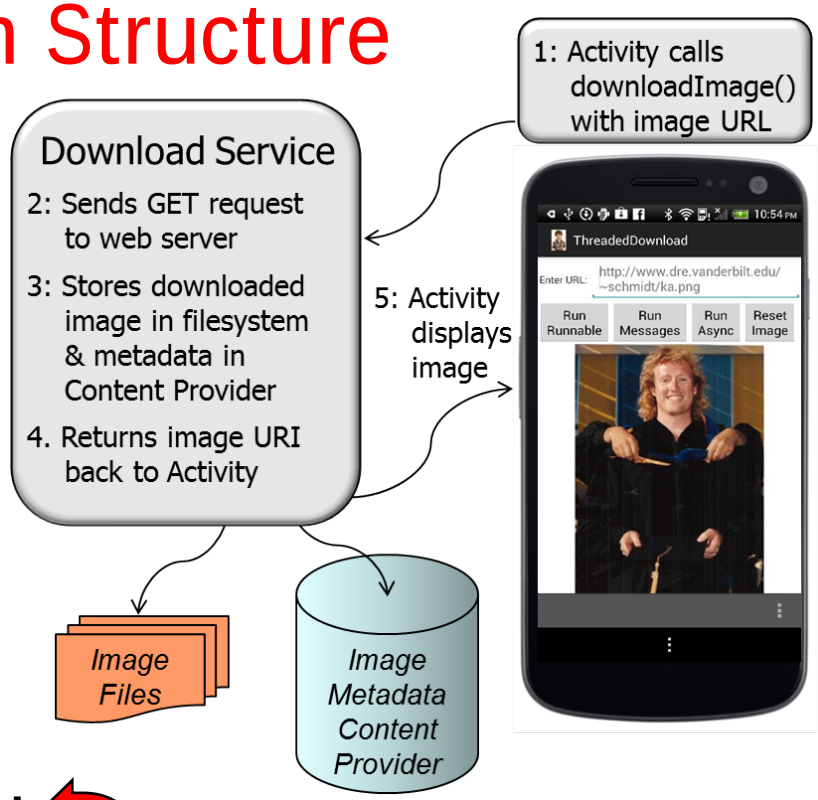


Start a new thread

Simplify Program Structure

- Hardware/software advances can improve the way that programs are structured, e.g.

```
private Bitmap bitmap;
final ImageView iview = ...
final Button button = ...
button.setOnClickListener
   (new OnClickListener() {
  public void onClick(View v) {
    new Thread(new Runnable() {
      public void run() {
        bitmap = downloadImage(URI);
        iview.post(new Runnable() {
        public void run()
        { iview.setImageBitmap(bitmap); }
      });
    }
  }).start();
```

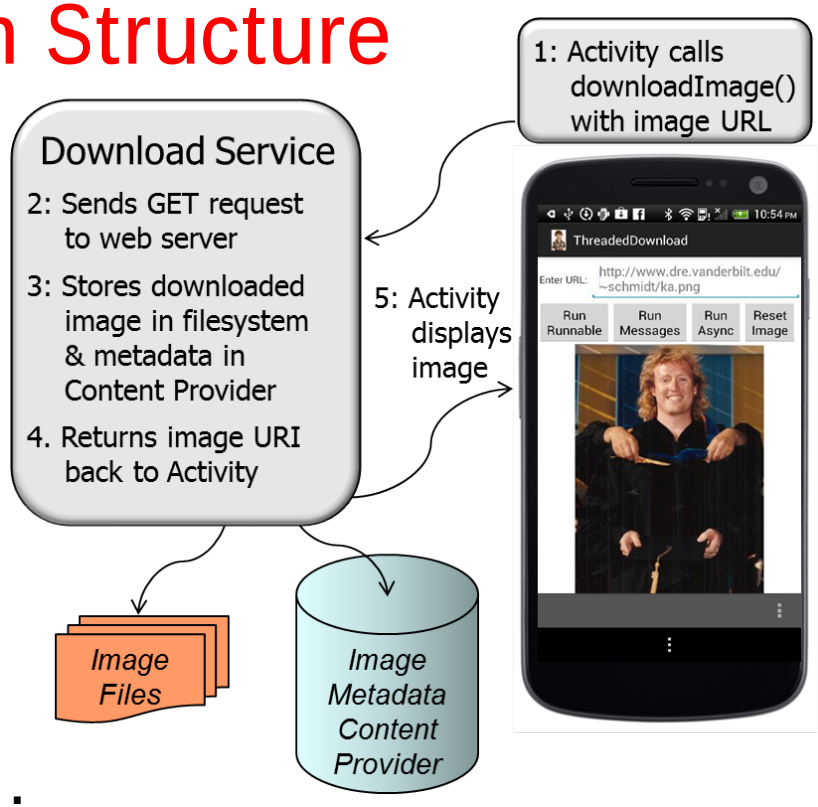


Download an image (blocking context)

Simplify Program Structure

- Hardware/software advances can improve the way that programs are structured, e.g.

```
private Bitmap bitmap;
final ImageView iview = ...
final Button button = ...
button.setOnClickListener
   (new OnClickListener() {
  public void onClick(View v) {
    new Thread(new Runnable() {
      public void run() {
        bitmap = downloadImage(URI);
        iview.post(new Runnable() {
        public void run()
        { iview.setImageBitmap(bitmap); }
      });
    }
  }).start();
```

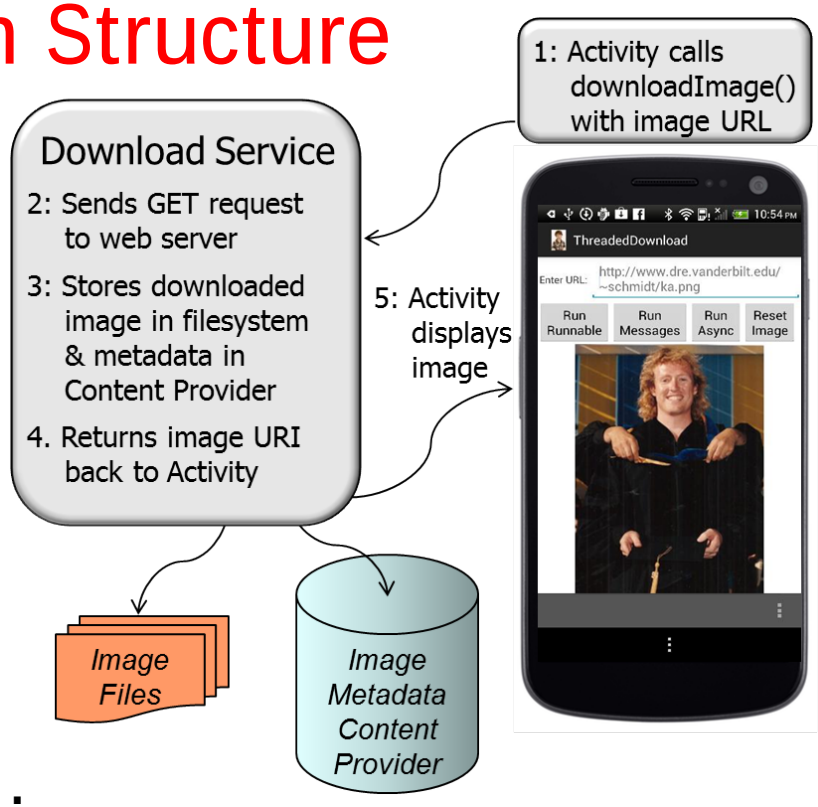


Post a Runnable that displays bitmap in the UI thread (non-blocking context)

Simplify Program Structure

- Hardware/software advances can improve the way that programs are structured, e.g.

```
private Bitmap bitmap;
final ImageView iview = ...
final Button button = ...
button.setOnClickListener
   (new OnClickListener() {
  public void onClick(View v) {
    new Thread(new Runnable() {
      public void run() {
        bitmap = downloadImage(URI);
        iview.post(new Runnable() {
        public void run()
        { iview.setImageBitmap(bitmap); }
      });
    }
  }).start();
```



This solution is more cohesive in time & space than a purely event-driven solution

Using Concurrency to Increase Performance

Increase Performance

- Increasing performance is a key motivation for using concurrency

en.wikipedia.org/wiki/Up_to_eleven has more on maximizing performance

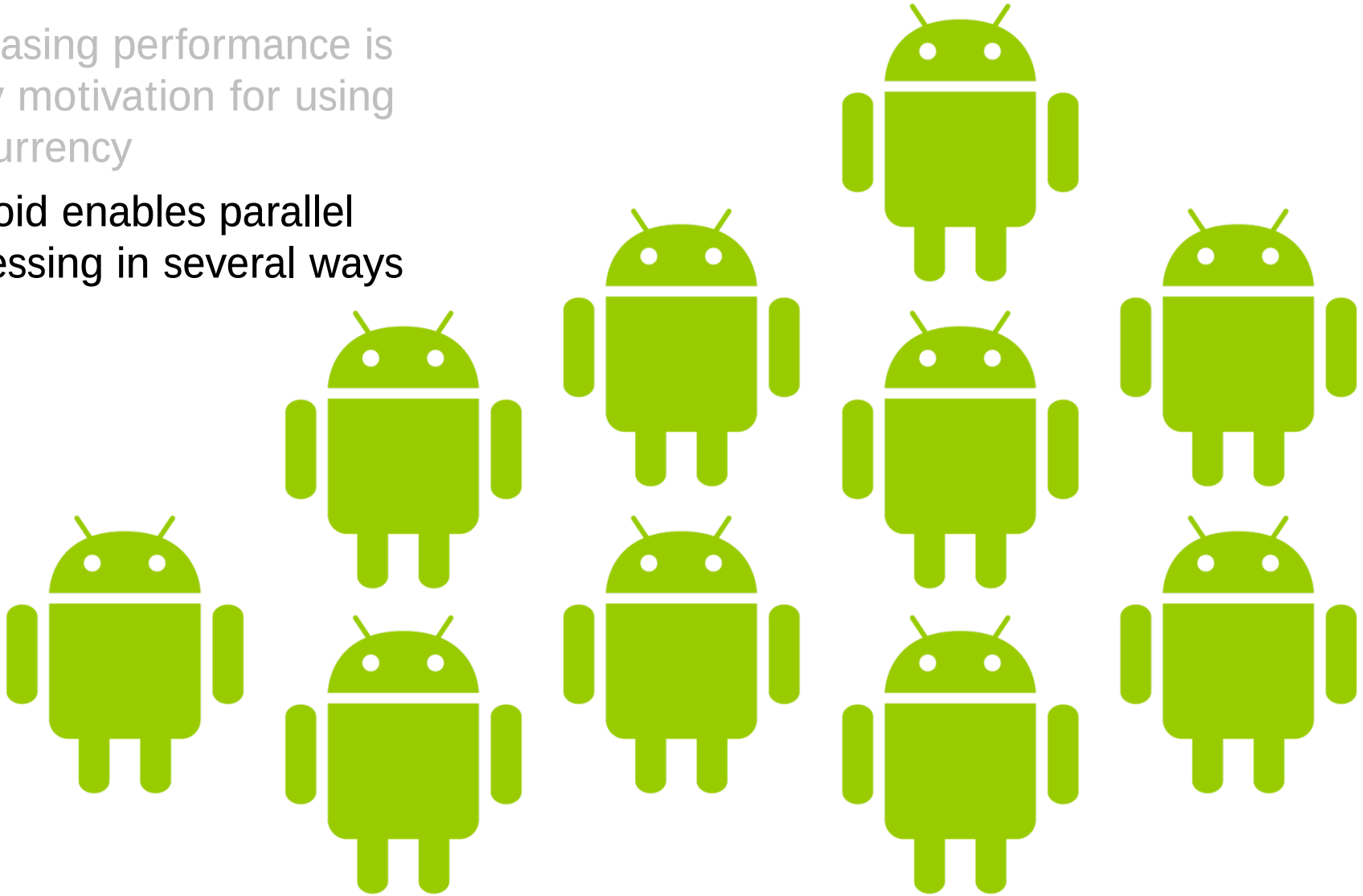
Increase Performance

- Increasing performance is a key motivation for using concurrency
  - Performance can be accelerated via parallel processing

en.wikipedia.org/wiki/Parallel_computing has more info

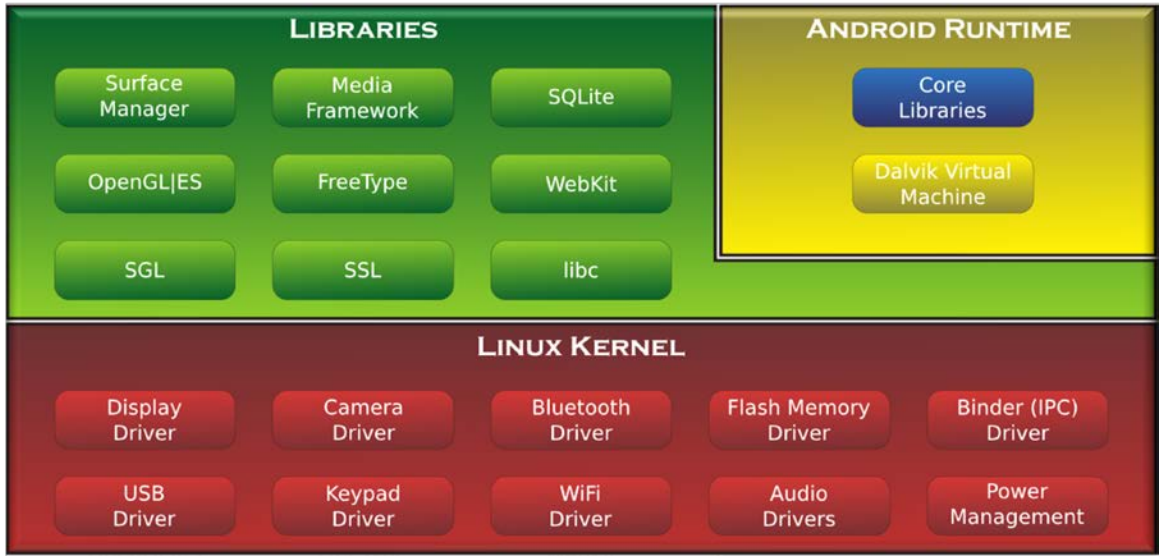
Increase Performance

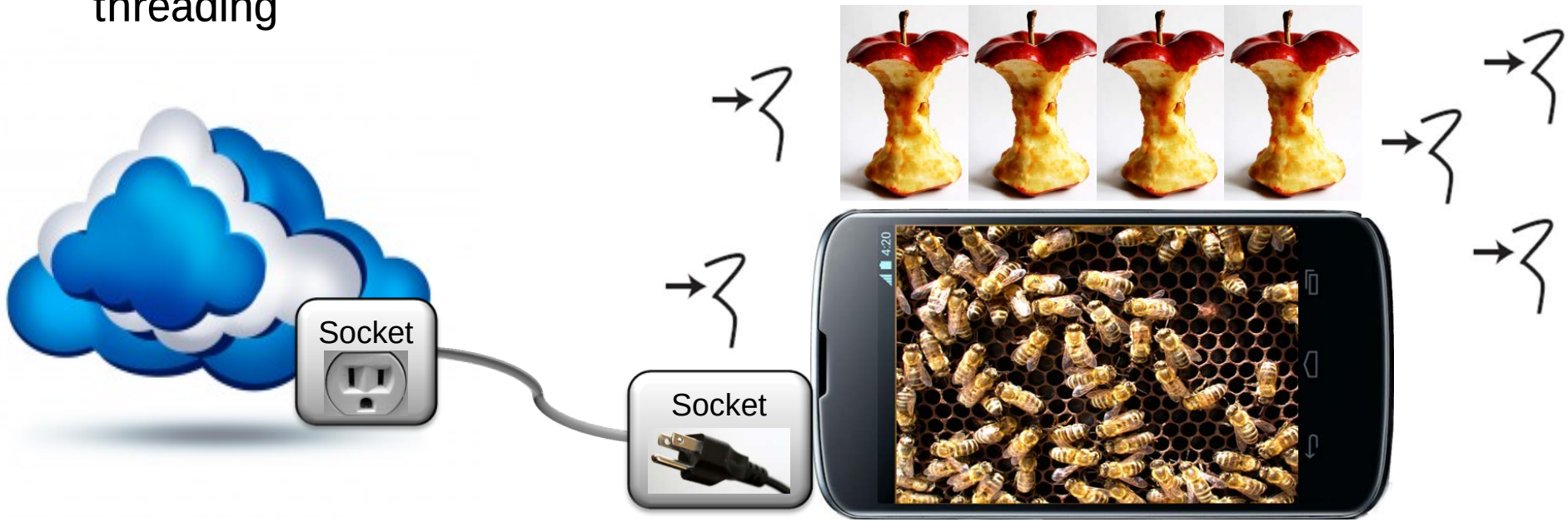
- Increasing performance is a key motivation for using concurrency
- Android enables parallel processing in several ways



Increase Performance

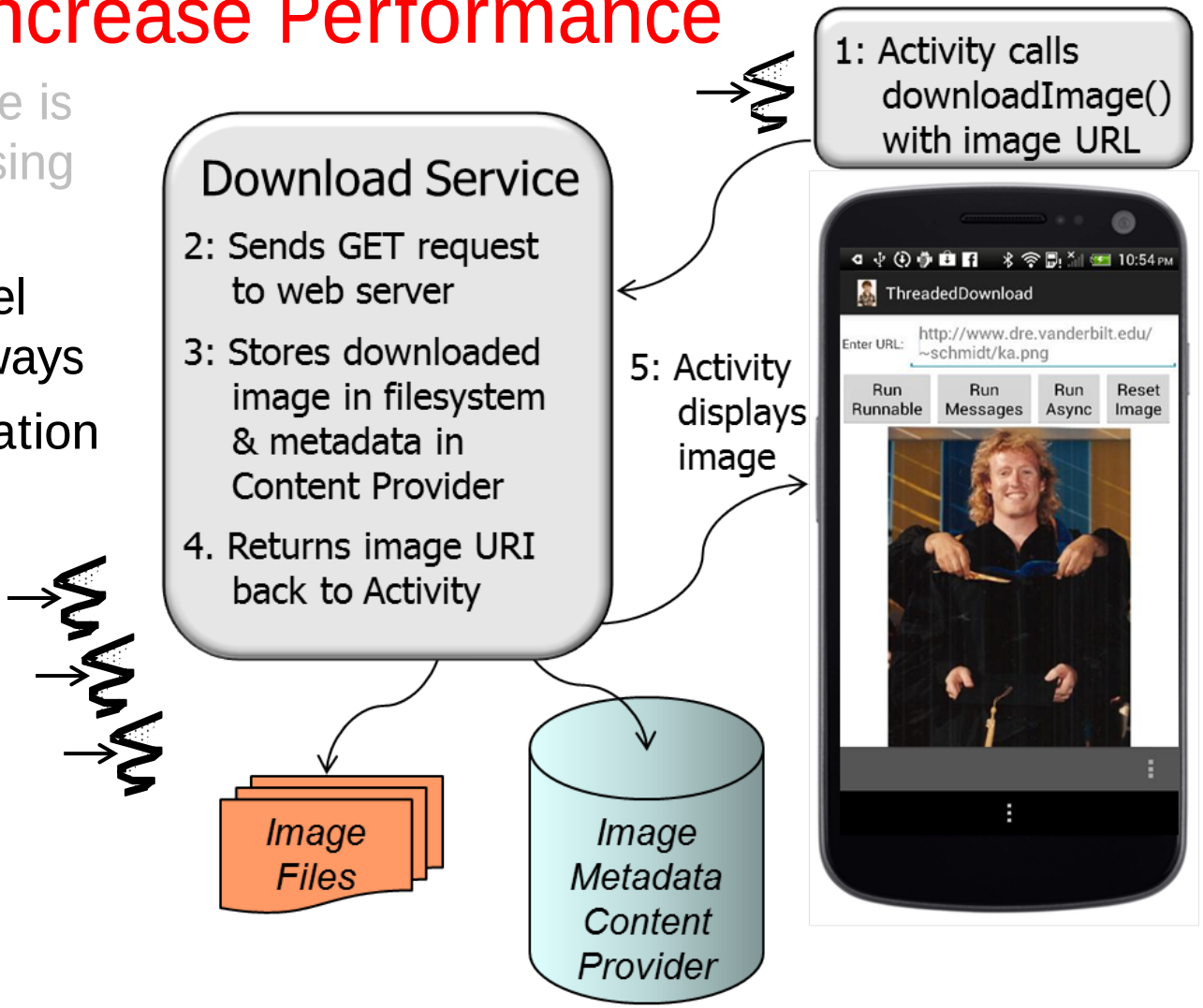
- Increasing performance is a key motivation for using concurrency
- Android enables parallel processing in several ways
  - Overlapping computation & communication operations via multi-threading





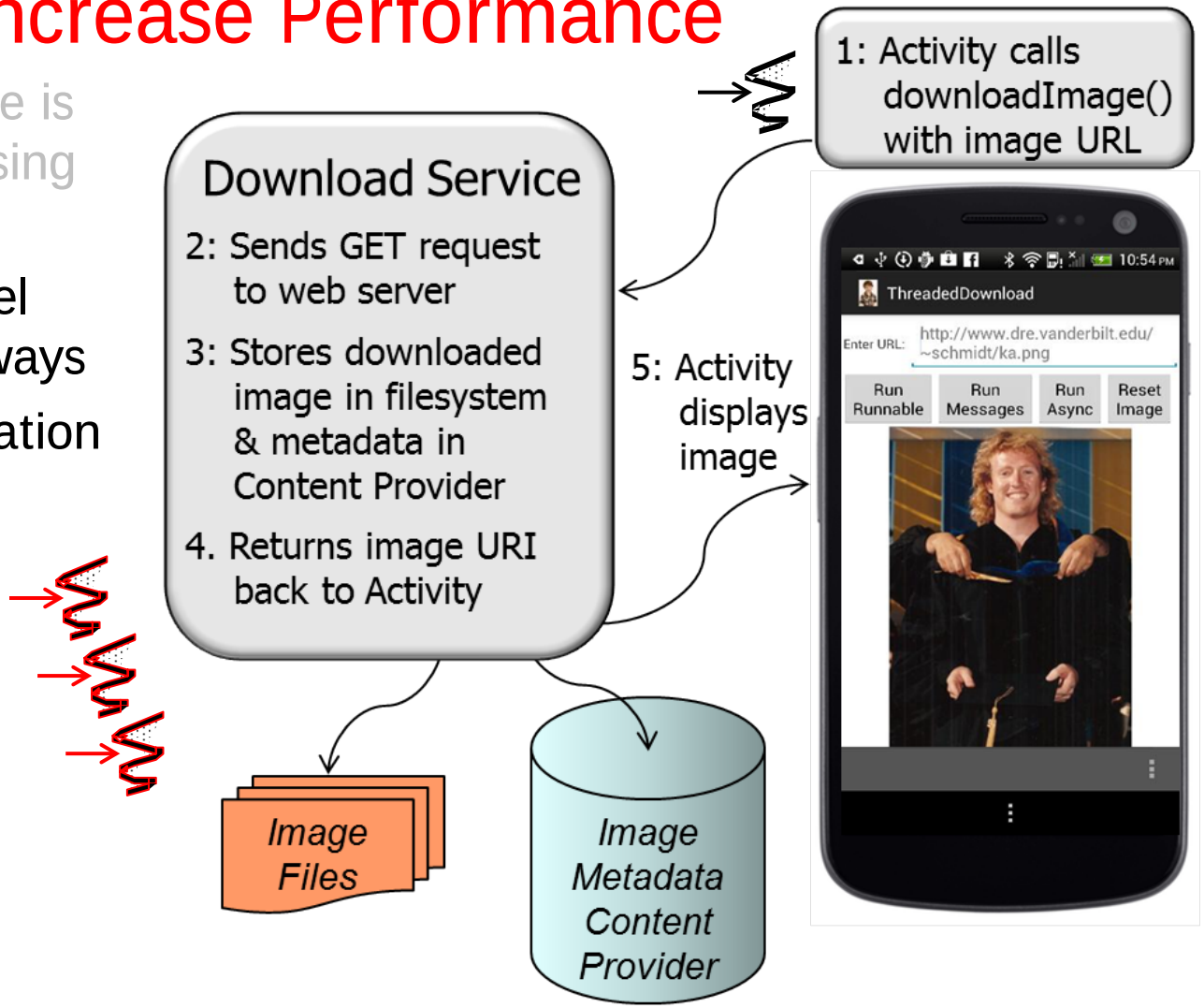
Increase Performance

- Increasing performance is a key motivation for using concurrency
- Android enables parallel processing in several ways
  - Overlapping computation & communication operations via multi-threading



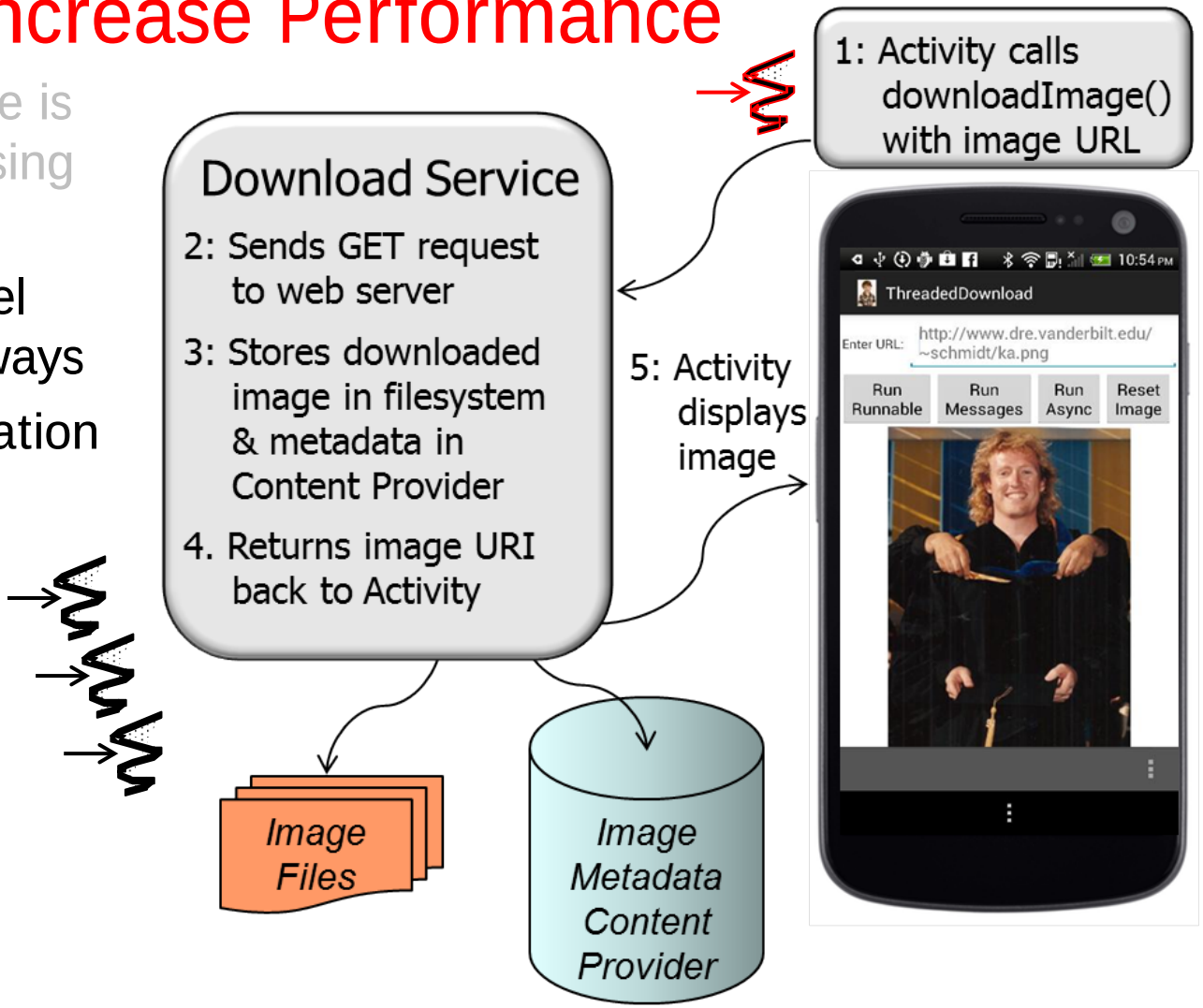
Increase Performance

- Increasing performance is a key motivation for using concurrency
- Android enables parallel processing in several ways
  - Overlapping computation & communication operations via multi-threading



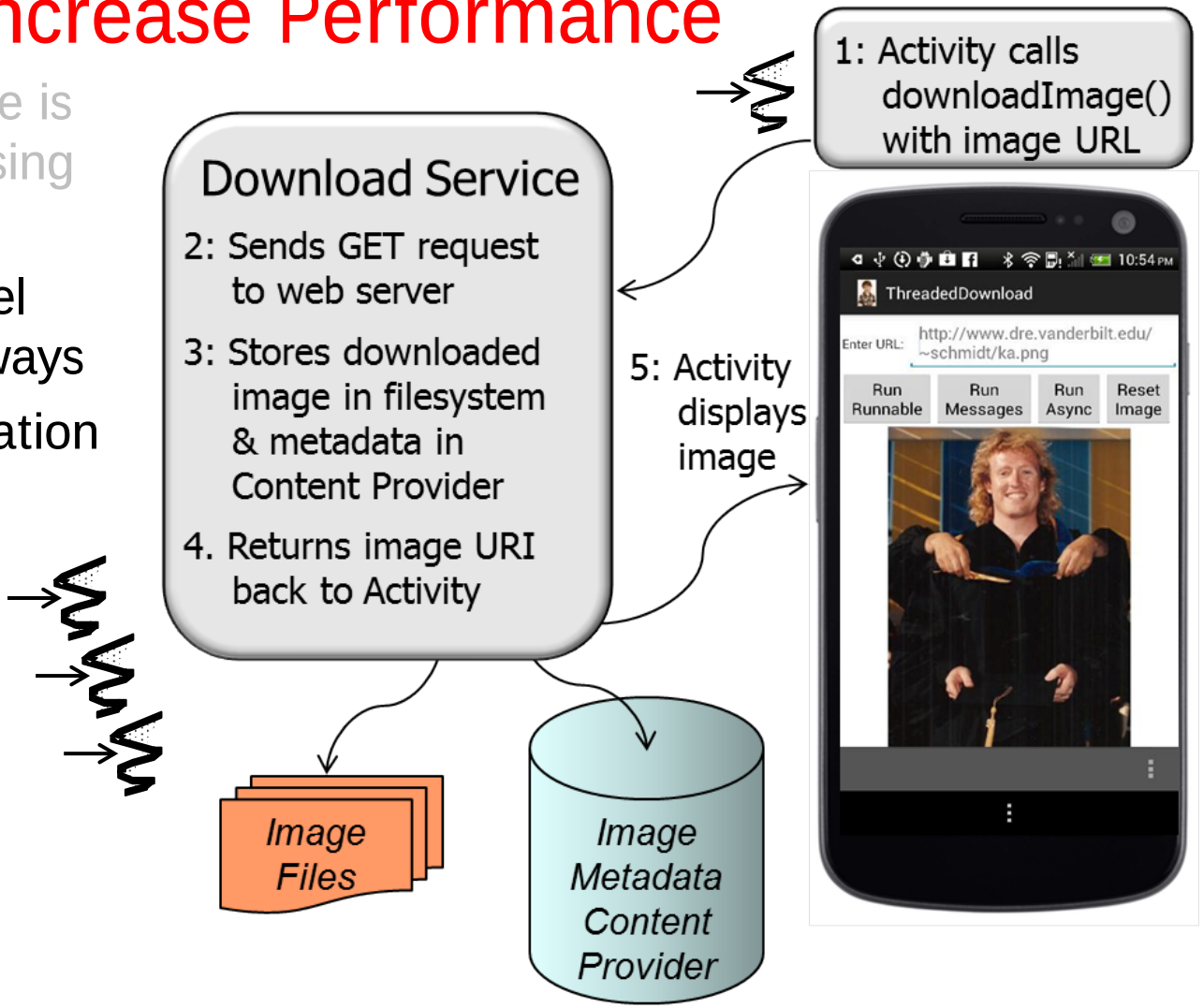
Increase Performance

- Increasing performance is a key motivation for using concurrency
- Android enables parallel processing in several ways
  - Overlapping computation & communication operations via multi-threading



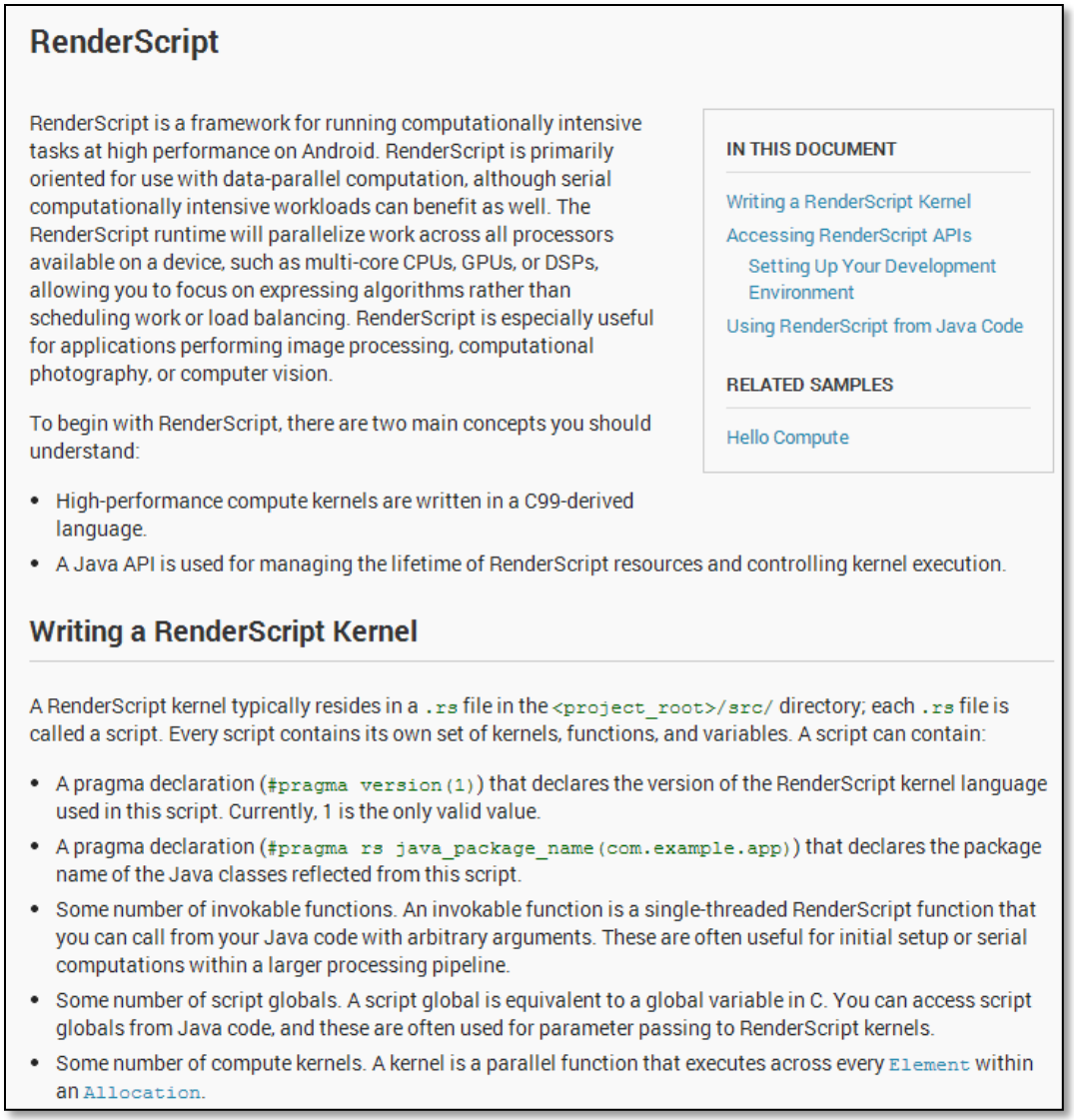
Increase Performance

- Increasing performance is a key motivation for using concurrency
- Android enables parallel processing in several ways
  - Overlapping computation & communication operations via multi-threading



Increase Performance

- Increasing performance is a key motivation for using concurrency
- Android enables parallel processing in several ways
  - Overlapping computation & communication operations via multi-threading
  - Parallelize work across all processors available on a device



RenderScript

RenderScript is a framework for running computationally intensive tasks at high performance on Android. RenderScript is primarily oriented for use with data-parallel computation, although serial computationally intensive workloads can benefit as well. The RenderScript runtime will parallelize work across all processors available on a device, such as multi-core CPUs, GPUs, or DSPs, allowing you to focus on expressing algorithms rather than scheduling work or load balancing. RenderScript is especially useful for applications performing image processing, computational photography, or computer vision.

To begin with RenderScript, there are two main concepts you should understand:

- High-performance compute kernels are written in a C99-derived language.
- A Java API is used for managing the lifetime of RenderScript resources and controlling kernel execution.

IN THIS DOCUMENT

Writing a RenderScript Kernel
Accessing RenderScript APIs
Setting Up Your Development Environment
Using RenderScript from Java Code

RELATED SAMPLES

Hello Compute

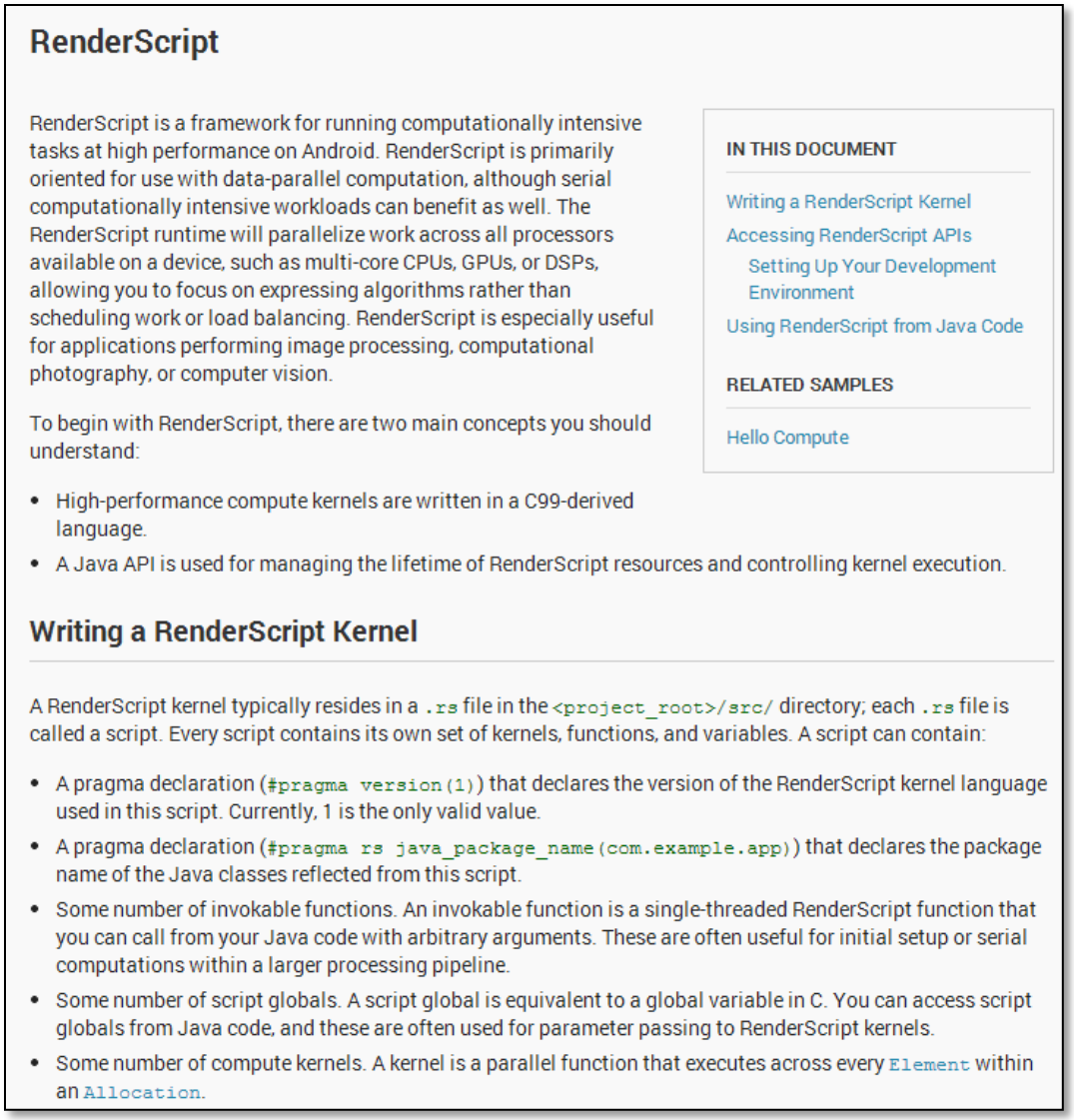
Writing a RenderScript Kernel

A RenderScript kernel typically resides in a `.rs` file in the `<project_root>/src/` directory; each `.rs` file is called a script. Every script contains its own set of kernels, functions, and variables. A script can contain:

- A pragma declaration (`#pragma version(1)`) that declares the version of the RenderScript kernel language used in this script. Currently, 1 is the only valid value.
- A pragma declaration (`#pragma rs java_package_name(com.example.app)`) that declares the package name of the Java classes reflected from this script.
- Some number of invokable functions. An invokable function is a single-threaded RenderScript function that you can call from your Java code with arbitrary arguments. These are often useful for initial setup or serial computations within a larger processing pipeline.
- Some number of script globals. A script global is equivalent to a global variable in C. You can access script globals from Java code, and these are often used for parameter passing to RenderScript kernels.
- Some number of compute kernels. A kernel is a parallel function that executes across every `Element` within an `Allocation`.

Increase Performance

- Increasing performance is a key motivation for using concurrency
- Android enables parallel processing in several ways
  - Overlapping computation & communication operations via multi-threading
  - Parallelize work across all processors available on a device
    - e.g., multi-core CPUs, GPUs, or DSPs



RenderScript

RenderScript is a framework for running computationally intensive tasks at high performance on Android. RenderScript is primarily oriented for use with data-parallel computation, although serial computationally intensive workloads can benefit as well. The RenderScript runtime will parallelize work across all processors available on a device, such as multi-core CPUs, GPUs, or DSPs, allowing you to focus on expressing algorithms rather than scheduling work or load balancing. RenderScript is especially useful for applications performing image processing, computational photography, or computer vision.

To begin with RenderScript, there are two main concepts you should understand:

- High-performance compute kernels are written in a C99-derived language.
- A Java API is used for managing the lifetime of RenderScript resources and controlling kernel execution.

IN THIS DOCUMENT

Writing a RenderScript Kernel
Accessing RenderScript APIs
Setting Up Your Development Environment
Using RenderScript from Java Code

RELATED SAMPLES

Hello Compute

Writing a RenderScript Kernel

A RenderScript kernel typically resides in a `.rs` file in the `<project_root>/src/` directory; each `.rs` file is called a script. Every script contains its own set of kernels, functions, and variables. A script can contain:

- A pragma declaration (`#pragma version(1)`) that declares the version of the RenderScript kernel language used in this script. Currently, 1 is the only valid value.
- A pragma declaration (`#pragma rs java_package_name(com.example.app)`) that declares the package name of the Java classes reflected from this script.
- Some number of invokable functions. An invokable function is a single-threaded RenderScript function that you can call from your Java code with arbitrary arguments. These are often useful for initial setup or serial computations within a larger processing pipeline.
- Some number of script globals. A script global is equivalent to a global variable in C. You can access script globals from Java code, and these are often used for parameter passing to RenderScript kernels.
- Some number of compute kernels. A kernel is a parallel function that executes across every `Element` within an `Allocation`.

Increase Performance

- Increasing performance is a key motivation for using concurrency
- Android enables parallel processing in several ways
  - Overlapping computation & communication operations via multi-threading
  - Parallelize work across all processors available on a device
    - e.g., multi-core CPUs, GPUs, or DSPs

See developer.android.com/guide/topics/renderscript/compute.html

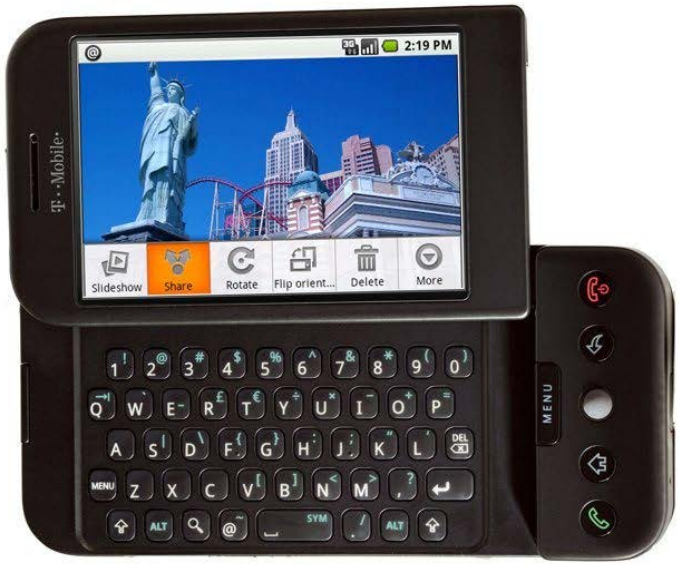
Using Concurrency to Improve Responsiveness

Improve Responsiveness

- Not every computing platform supports multiple cores

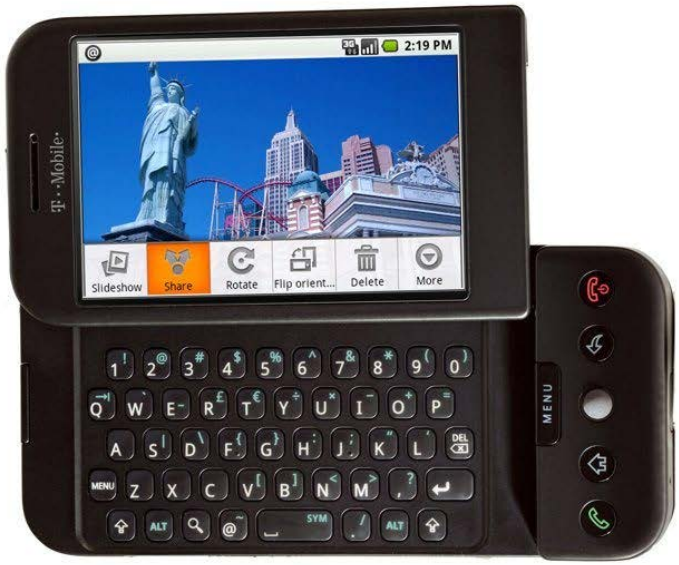
Improve Responsiveness

- Not every computing platform supports multiple cores



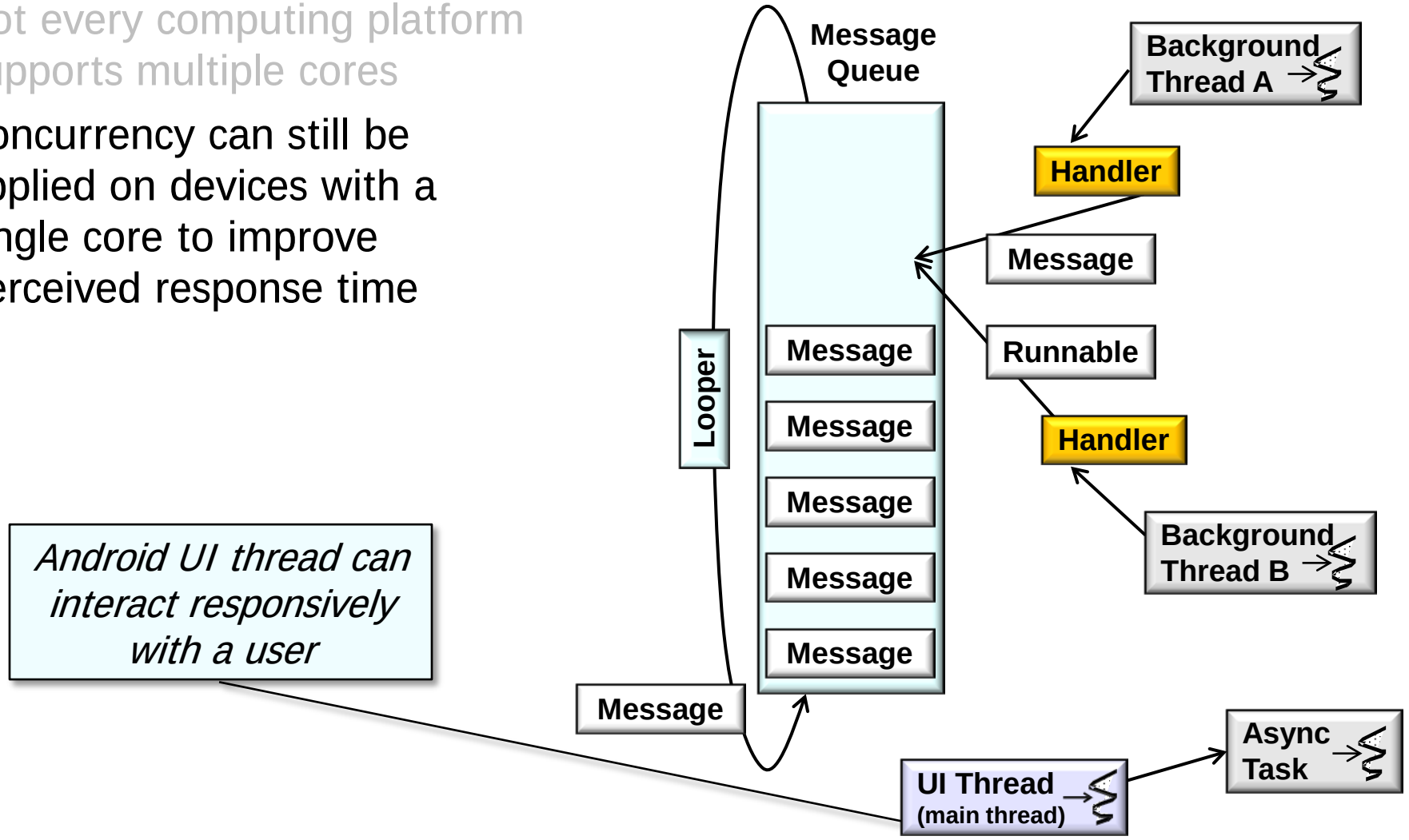
Improve Responsiveness

- Not every computing platform supports multiple cores
- Concurrency can still be applied on devices with a single core to improve perceived response time
  - e.g., don't ignore user input



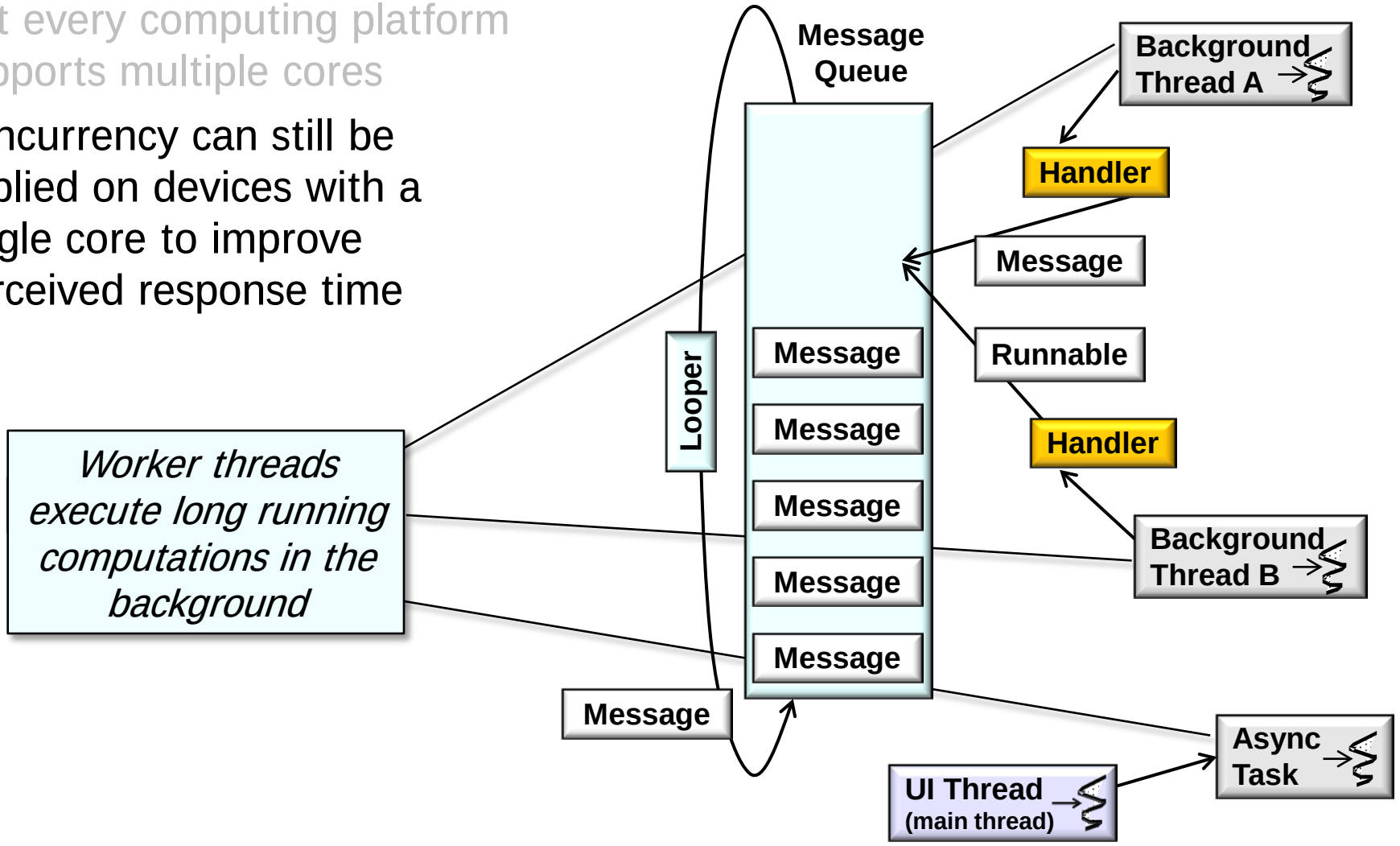
Improve Responsiveness

- Not every computing platform supports multiple cores
- Concurrency can still be applied on devices with a single core to improve perceived response time



Improve Responsiveness

- Not every computing platform supports multiple cores
- Concurrency can still be applied on devices with a single core to improve perceived response time



en.wikipedia.org/wiki/Responsiveness has more info

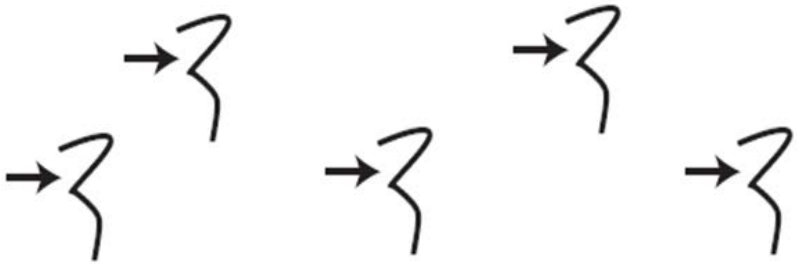
Summary

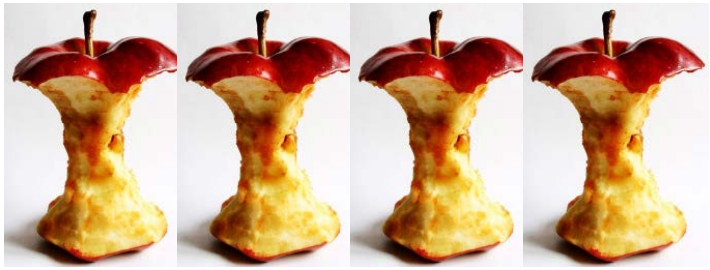
Summary

- Motivations for concurrent software

Summary

- Motivations for concurrent software
  - Leverage advances in hardware & software infrastructure technologies





Summary

- Motivations for concurrent software
  - Leverage advances in hardware & software infrastructure technologies
  - Meet the quality & performance needs of apps & services

Summary

- Motivations for concurrent software
  - Leverage advances in hardware & software infrastructure technologies
  - Meet the quality & performance needs of apps & services