Parallel Programming: Responsiveness vs. Performance

Joe Hummel, PhD

Microsoft MVP Visual C++

Technical Staff: Pluralsight, LLC

Professor: U. of Illinois, Chicago

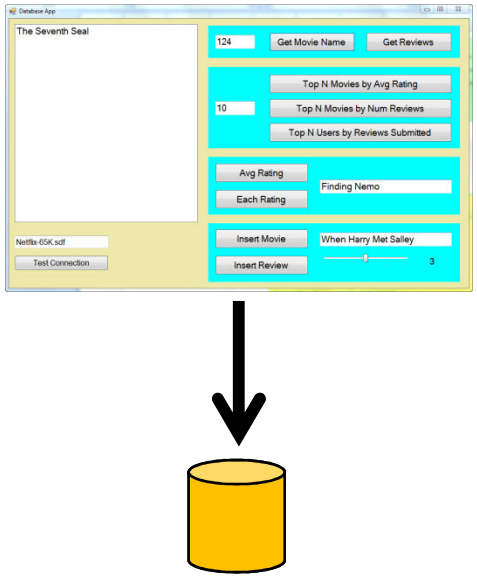
email: jhummel2@uic.edu

stuff: http://www.joehummel.net/downloads.html

- **Parallel programming…**
  - *Why is this important?*

1.

Operations may take unpredictable amounts of time — you cannot wait and do nothing…



2.

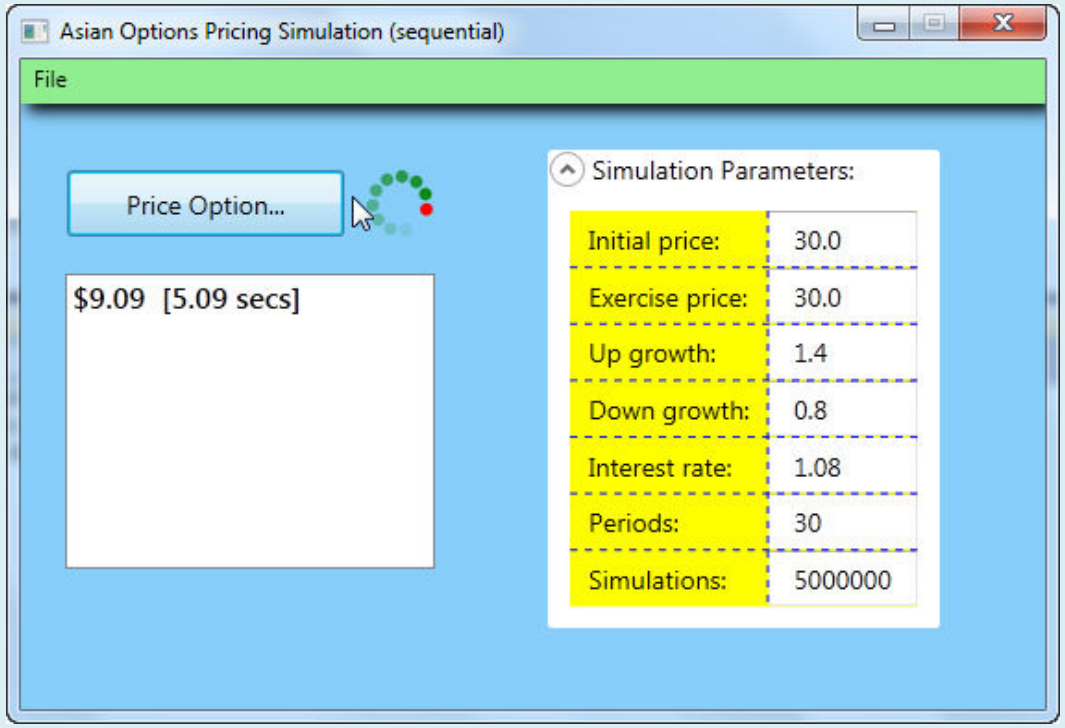
Some operations are data or time intensive — the only way to solve faster is to process in parallel…

Responsiveness vs. Performance

- **Better responsiveness?**
  - **Asynchronous programming**
    - Long-running operations
    - I/O operations
    - OS calls

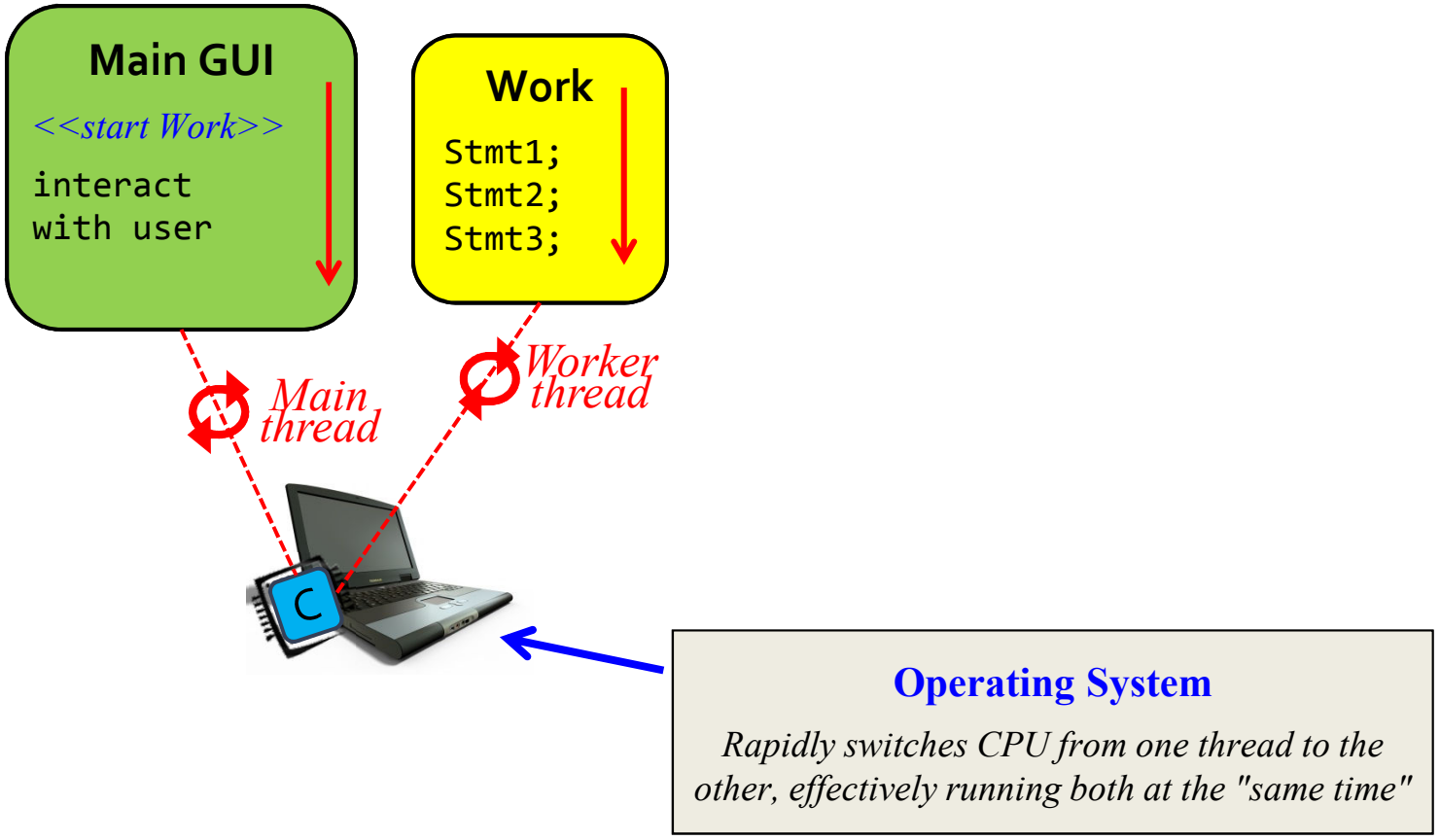
- **Better performance?**
  - **Parallel programming**
    - Numerical processing
    - Data analysis
    - Big data

DEMO


Asian Options Pricing Simulation (sequential)
File
Price Option...
$9.09 [5.09 secs]
Simulation Parameters:
Initial price: 30.0
Exercise price: 30.0
Up growth: 1.4
Down growth: 0.8
Interest rate: 1.08
Periods: 30
Simulations: 5000000

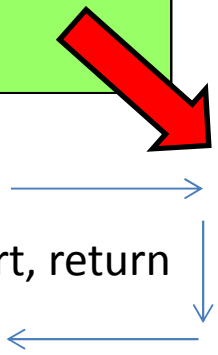
SW solution for better responsiveness?

- **Multithreading**



Multithreaded programming in the UI

```
void button1_Click(…)
{
  var result = DoLongLatencyOp();
  lstBox.Items.Add(result);
}
```



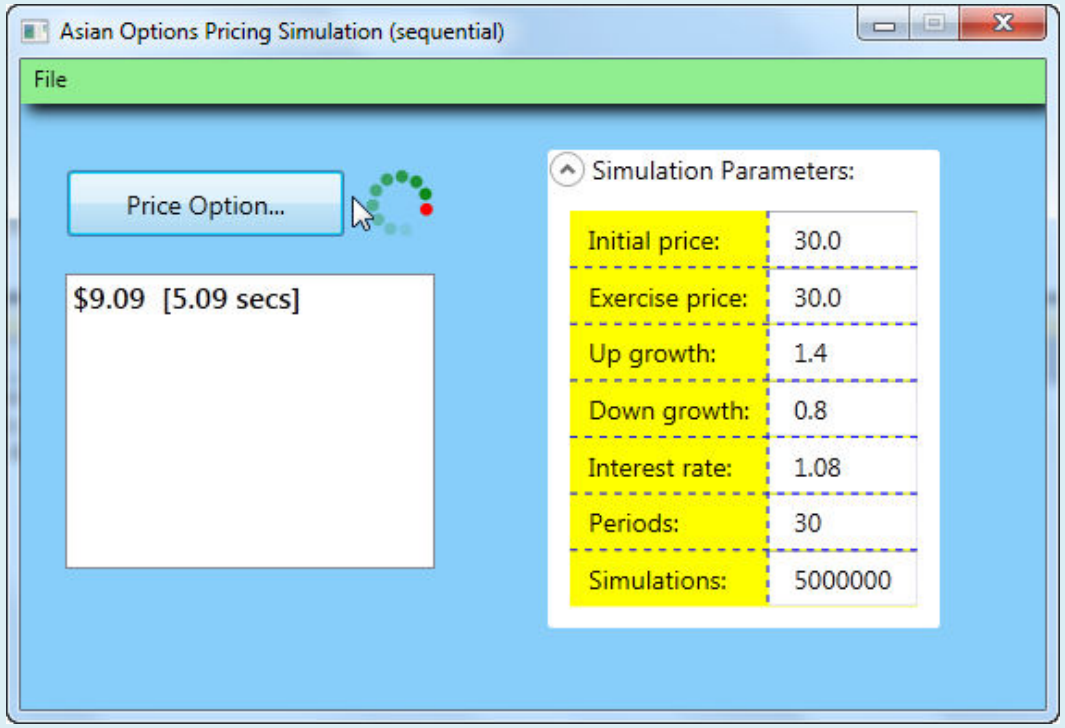
click, start, return

```
using System.Threading;

void button1_Click(…)
{
  t = new thread(code);
  t.Start();
}

void code()
{
  var result = DoLongLatencyOp();
  lstBox.Items.Add(result);
}
```

DEMO


Asian Options Pricing Simulation (sequential)
File
Price Option...
$9.09 [5.09 secs]
Simulation Parameters:
Initial price: 30.0
Exercise price: 30.0
Up growth: 1.4
Down growth: 0.8
Interest rate: 1.08
Periods: 30
Simulations: 5000000

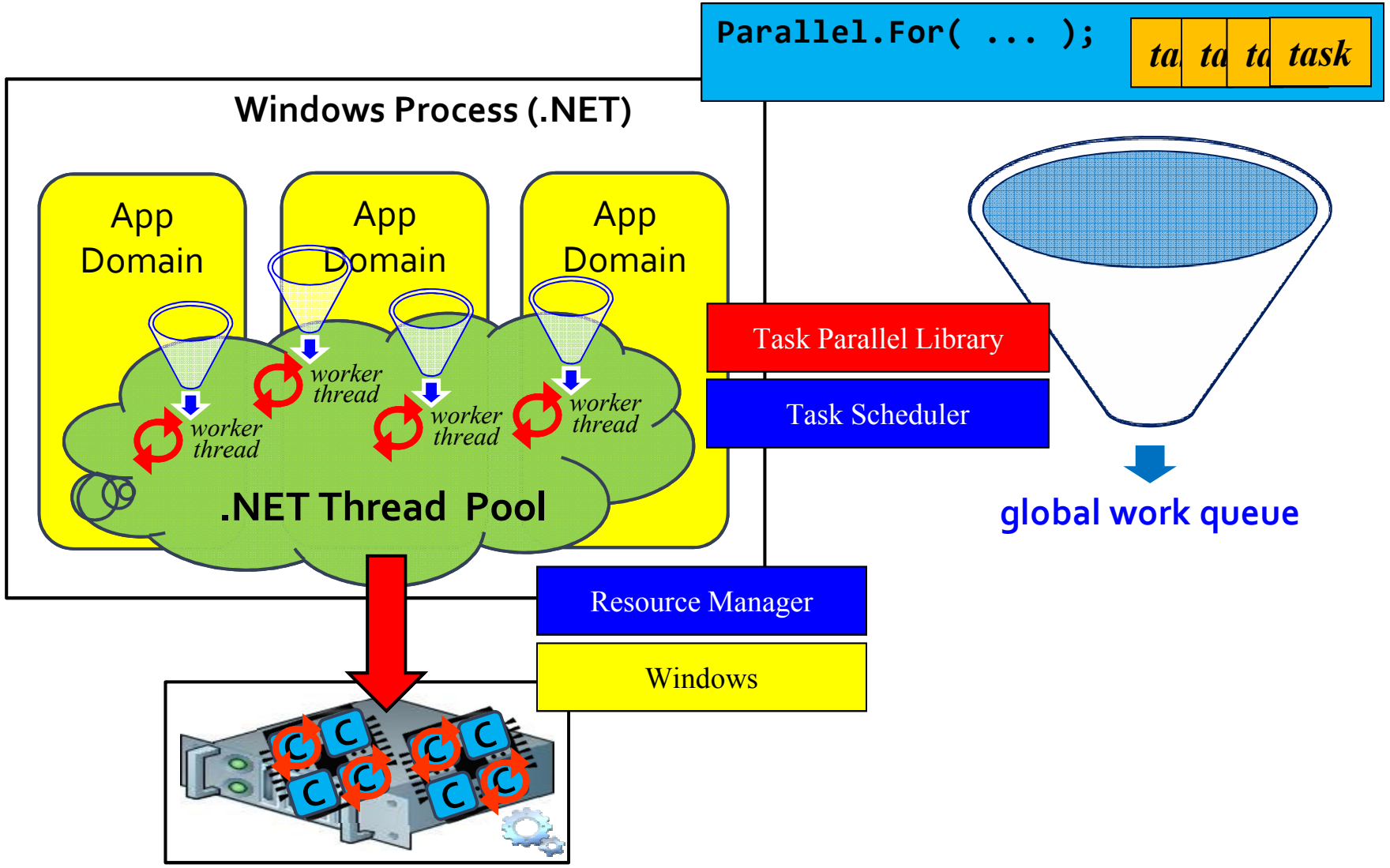
- **Threads are…**
  - ***expensive*** *to create (involves OS)*
  - *easy to* ***over-subscribe*** *(i.e. create too many)*
  - ***tedious*** *to program (lots of little details not shown in PPT)*
  - *an* ***obfuscation*** *(intent of code is harder to recognize)*

A Better Solution: Tasks

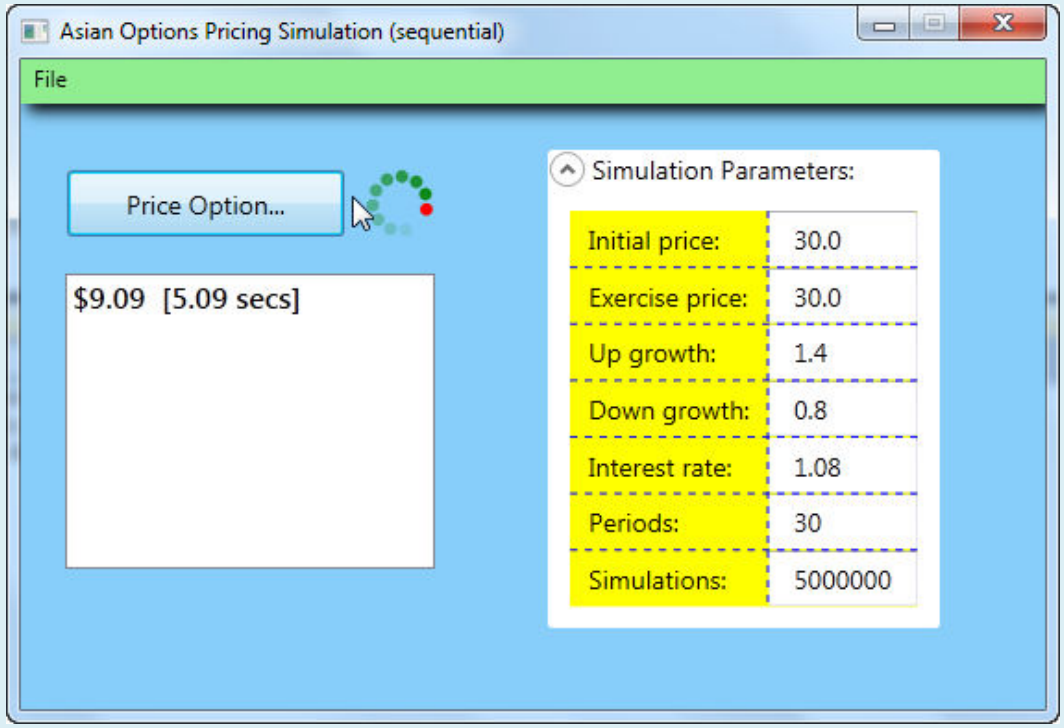
- Separate notion of work (“task”) from the workers (“threads”)

Task == *a unit of work; an object denoting an ongoing operation or computation.*

Task-based Execution Model

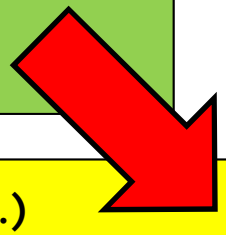


DEMO


Asian Options Pricing Simulation (sequential)
File
Price Option...
$9.09 [5.09 secs]
Simulation Parameters:
Initial price: 30.0
Exercise price: 30.0
Up growth: 1.4
Down growth: 0.8
Interest rate: 1.08
Periods: 30
Simulations: 5000000

Task-based programming in the UI

```
void button1_Click(…)
{
   var result = DoLongLatencyOp();
   lstBox.Items.Add(result);
}
```



using System.Threading.Tasks;

```
void button1_Click(…)
{
   var context =  // grab UI thread context to run UI task:
       TaskScheduler.FromCurrentSynchronizationContext();

   Task.Factory.StartNew(()=>
     {
       return DoLongLatencyOp();
     }
   ).ContinueWith((antecedent) =>
     {
       lstBox.Items.Add(antecedent.Result);
     },
     context  // execute this task on UI thread:
   );
}
```

An Even Better solution?

- **This is where language design can help…**
  - *Why not have the compiler generate pattern for us?*

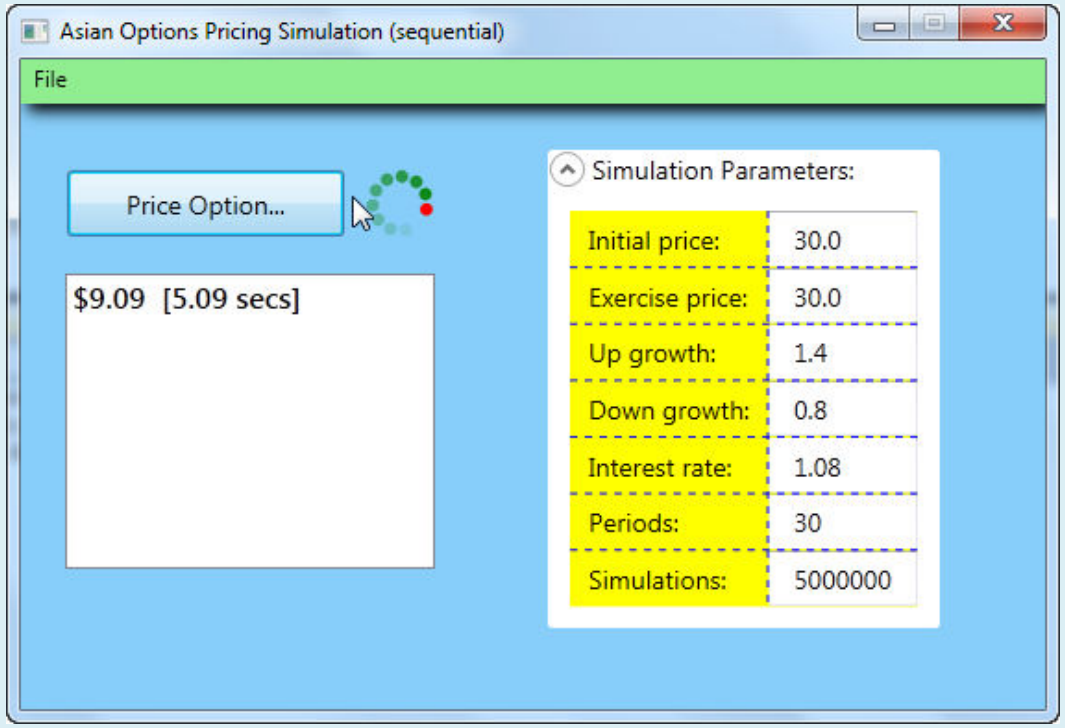
- **Solution: async / await keywords in C#**

```
void button1_Click(…)
{
   var result = DoLongLatencyOp();
   lstBox.Items.Add(result);
}
```

```
using System.Threading.Tasks;
```

```
async void button1_Click(…)
{
  var result = await Task.Run(() => DoLongRunningOp());
  lstBox.Items.Add(result);
}
```

DEMO

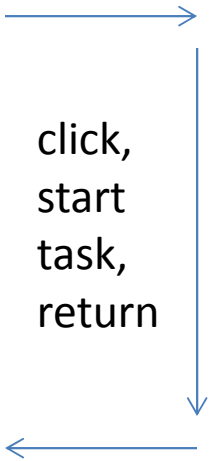

Asian Options Pricing Simulation (sequential)
File
Price Option...
$9.09 [5.09 secs]
Simulation Parameters:
Initial price: 30.0
Exercise price: 30.0
Up growth: 1.4
Down growth: 0.8
Interest rate: 1.08
Periods: 30
Simulations: 5000000

async / await

*Method *may* perform async, long-latency op*

```
async void button1_Click(…)
{
   var result = await Task.Run(() => DoLongRunningOp());
   lstBox.Items.Add(result);
}
```

Tells compiler to run this task on a separate thread AND setup a callback to execute remaining code when thread finishes…




```
void button1_Click(…)
{
   Task.Run( () =>
   {
      return DoLongLatencyOp();
   }
   .ContinueWith( (prev) =>
   {
      var result = prev.Result;
      lstBox.Items.Add(result);
   }
}
```

runs on a separate worker thread

runs on main GUI thread when above finishes

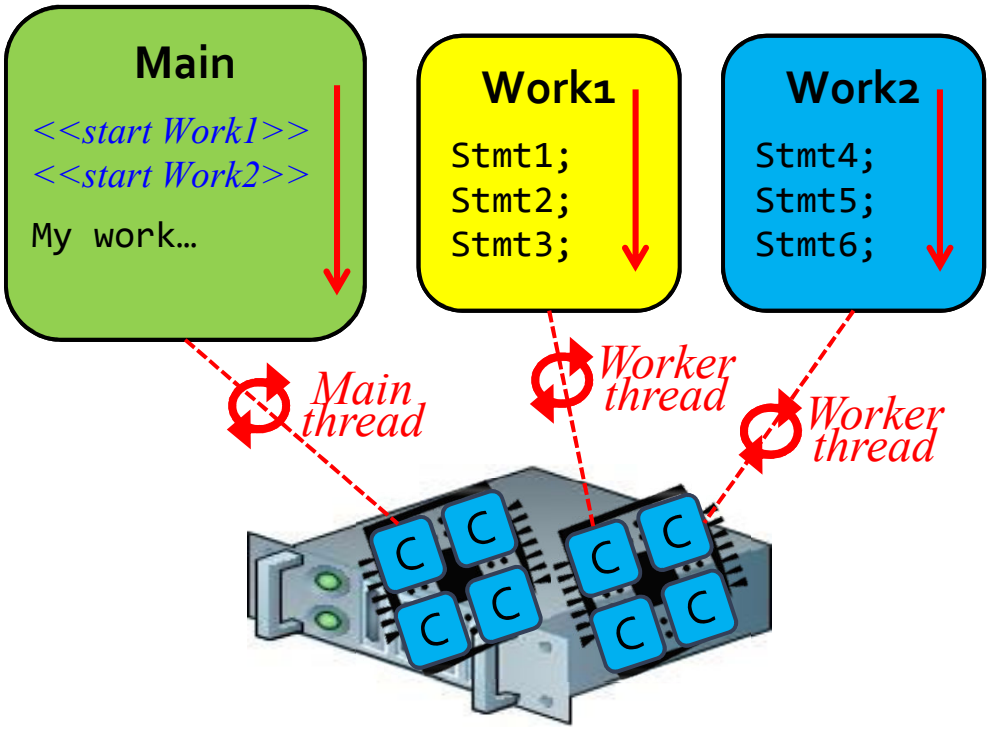
Responsiveness vs. Performance

- Better responsiveness?
  - Asynchronous programming
    - Long-running operations
    - I/O operations
    - OS calls

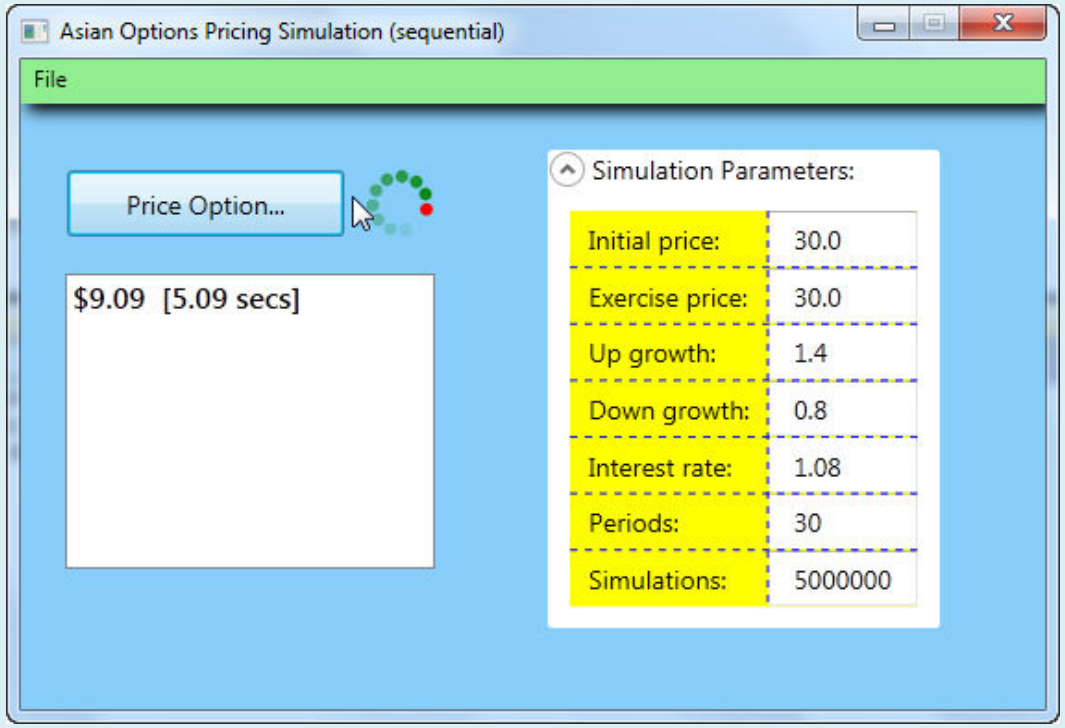
- **Better performance?**
  - **Parallel programming**
    - Numerical processing
    - Data analysis
    - Big data

SW solution for better performance?

- **Threading across different cores…**



DEMO


Asian Options Pricing Simulation (sequential)
File
Price Option...
$9.09 [5.09 secs]
Simulation Parameters:
Initial price: 30.0
Exercise price: 30.0
Up growth: 1.4
Down growth: 0.8
Interest rate: 1.08
Periods: 30
Simulations: 5000000