Modernize common concurrency patterns with actors

Barry Feigenbaum

Transcript

0:01

Hello, my name is Barry Feigenbaum.

0:03

Welcome to the "Concurrent programming made easy" demonstration video.

0:11

Let's begin with the first demo, which shows many actors of the same type,

0:16

each of which sends a message to other actors selected at random.

0:19

The actors send a large number of messages.

0:22

As we watch the demo, we see that more messages are sent

0:27

than can be processed and many get buffered.

0:30

Over time, the message send rate slows down and the buffered messages get processed.

0:40

Fewer and fewer messages are buffered.

0:44

Eventually all message sends stop and the system comes to a halt.

0:47

The important thing to note is that the threads are well utilized as long

0:51

as the message arrival rate is fairly high.

0:53

As the arrival rate drops, the thread utilization also drops.

1:04

While this demo runs, let's discuss the user interface.

1:07

At the top left is a control panel, which selects the demonstrations to run

1:12

and has some controls and meters.

1:14

The thread bar shows busy threads in green and idle threads in yellow.

1:19

In the center is the simulation display, which also shows the threads

1:24

as squares and the actors as circles.

1:28

The actors are arranged in a circle with messages between them sent shown as lines.

1:33

On the right is a trace log showing the time, messages processed,

1:38

the thread it is on, and the processing event.

1:41

Actors come in two shades: light when idle and drank when processing a message.

1:48

Actors can be transparent or opaque.

2:15

These simulations run until there are no new messages arriving.

2:18

The simulation then pauses to allow viewing in the log.

2:38

The second demo is a variant of the classic producer/consumer problem.

2:42

Each blue producer creates one to three green consumers

2:45

and then sends the requests to the consumers.

2:54

Note the message lines have small circles on them that indicate message targets

3:02

and the number of buffered messages.

3:04

Note the construction of items in the log.

3:16

Note the history view at the bottom of the display.

3:24

Note the numbers next to the threads showing the actor currently running on the thread.

3:54

Note the use of opaque and transparent actors.

4:23

While this simulation runs, let's look at the meters.

4:27

The top meter, messages per second, shows the message arrival rate.

4:37

The middle meter, dispatches per second, shows the message completion rate.

4:45

The bottom meter, net message rate, should be near zero over time for a balanced system.

4:59

Here, the last of the buffered messages are processed

5:09

and the simulation halts.

5:27

The third demo is an implementation of Map/Reduce.

5:35

We calculate "the sum of the squares"

5:37

of the first 1000 random number values in 100 sets of 10 values each.

5:47

And then the sets are summed in a reduce process.

6:11

Finally, the last reduction uses only one thread.

6:22

Note the result of the trace log.

6:45

The fourth demo shows a disk scan looking for suspect (many line) text files.

6:55

Each actor processes a message to either scan a directory or scan a file.

7:12

Processing starts at a root directory and then quickly spreads out,

7:29

creating actors as directories are processed.

7:56

Eventually all the directories and files are scanned after which the execution completes.

8:18

The last demo is all the above demos running concurrently.

8:22

Many actions are present and increasing over time.

8:29

Notice there are many messages and threads are very busy.

8:33

Under this load, more threads are needed.

8:38

Some more are added.

8:52

Even with the additional threads, the threads remain mostly busy for much of this demo,

8:58

but the message completion rate is increased and thus the run time

9:07

on this demo is significantly reduced.

9:23

This demo shows the access system under heavy message load ...

9:27

... and with a large number of actors of different types.

9:52

Notice the log shows all the various types of messages from the earlier simulations.

11:07

While not shown, there are controls to enable sounds to indicate thread usage.

11:20

These sounds indicate transitions from fully busy or fully idle states.

11:45

These examples show both the flexibility of the actor system and its capability

11:49

for processing high loads, including support for a variable number of threads and actors.

12:05

It also makes clear the system's ability to distribute the message load

12:08

across threads in an efficient and fair way.

12:28

Thank you for watching this demonstration.