Concurrency is Hard

Assume there are eight threads with indices 0, . . . , 7. The threads share an array a of size 800,000. Each thread executes the following code snippet.

```
for (int i = 0; i < 100000; i++) {
    for (int j = 0; j < 100000; j++) {
        a[index + j * 8]++
    }
}
```

Concurrency is Hard

Assume there are eight threads with indices 0, . . . , 7. The threads share an array a of size 800,000. Each thread executes the following code snippet.

```
for (int i = 0; i < 100000; i++) {
    for (int j = 0; j < 100000; j++) {
        a[index * 100000 + j]++
    }
}
```

Question

Do both snippets give rise to the same number of loads and stores?

Answer

Yes.

Concurrency is Hard

Question

Do the snippets take roughly the same amount of time?

Answer

No.

Later in the course, we will study memory models.

Concurrent Programming Languages

Question

Can you name some concurrent programming languages?

Concurrent Programming Languages

Ada, BPEL, C, C++, Caml, Concurrent ML, CUDA, Erlang, Java, JavaScript, Linda, Pict, POOL, Occam, Scala

Concurrent Programming Languages

Most concurrent programming languages consist of a sequential programming language plus support for

- thread creation,
- communication, and
- synchronization.

Thread Creation

We distinguish between

- static thread creation
  only allowing a predefined number of threads
- dynamic thread creation
  allowing new threads to be created “on-the-fly”

Communication

We distinguish between communication using

- shared variables
- messages
  - synchronous (blocking send, blocking receive)
  - asynchronous (non-blocking send, blocking receive)

Shared variable communication

Question

What is a real life analogue for shared variable communication?

Answer

Message board.

Synchronous message passing communication

Question

What is a real life analogue for synchronous message passing communication?

Answer

Phone.

Asynchronous message passing communication

Question

What is a real life analogue for asynchronous message passing communication?

Answer

Email.

Synchronization

- semaphores
- locks
- monitors
- barriers
- compare-and-swap
- ...

Java

In this course we will focus on Java.

Java has

- dynamic thread creation,
- shared variable communication,
- semaphores,
- locks,
- monitors,
- barriers,
- compare-and-swap,
- ...

But before diving into the details of Java, let's study these concepts first.

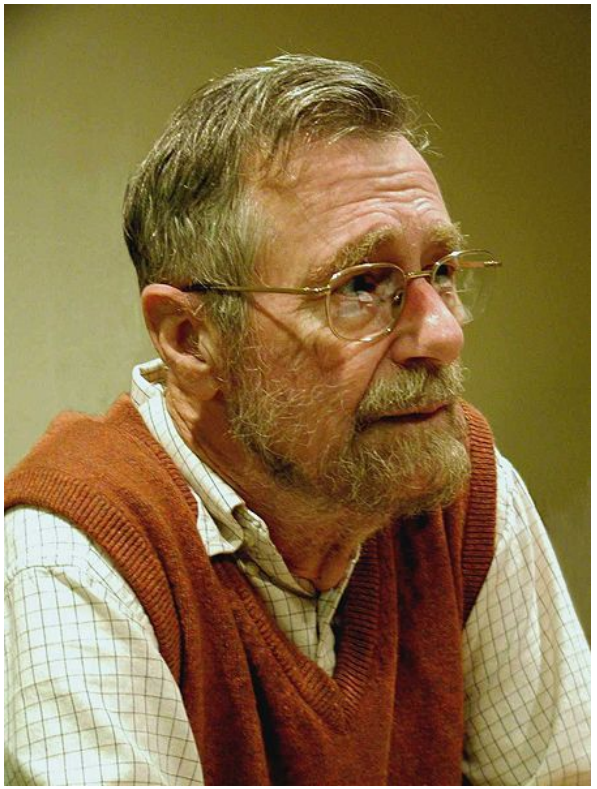
Semaphore

A semaphore is a nonnegative integer, say s, with two atomic operations:

- V(s): increment s by 1.
- P(s): decrement s by 1 as soon as the result is nonnegative.

Edsger Wybe Dijkstra

- Member of the Royal Netherlands Academy of Arts and Sciences (1971)
- Distinguished Fellow of the British Computer Society (1971)
- Recipient of the Turing Award (1972)
- Foreign Honorary Member of the American Academy of Arts and Sciences (1975)



Edsger Wybe Dijkstra

(1930–2002)

source: www.computer.org

Semaphore

Question

Let three threads share a semaphore s with initial value 0. One thread executes

```
P(s); print(1)
```

Another thread executes

```
P(s); print(2)
```

The other thread executes

```
sleep(1 hour); print(3); V(s)
```

What will be printed?

Answer

31 or 32.

The Critical Section Problem

Consider two threads both defined by

```
while (true)
{
    critical section
    non-critical section
}
```

Mutual exclusion: Make sure that at any moment at most one of the threads is in its critical section.

Problem

Introduce one or more semaphores and add P- and V-operations to be above code snippet so that mutual exclusion is ensured.

The Producer-Consumer Problem

The producer-consumer problem (also known as the bounded-buffer problem) is a classical concurrency problem.

The problem is to synchronize two threads, the producer and the consumer, who share a common, fixed-size buffer. The producer repeatedly generates a data item and puts it into the buffer. At the same time, the consumer removes data items from the buffer, one item at a time.

The problem is to make sure that the producer will not try to add data items to a full buffer and that the consumer will not try to remove data items from an empty buffer.

The Producer-Consumer Problem

We assume that the items are integers. We represent the buffer by means of an array of integers. The array has a fixed size.

```
int N = 10; // capacity of buffer
```

The producer and consumer share the following variables.

```
int[] buffer; // array representing buffer
int next = 0; // index of cell for next item
int size = 0; // number of items stored in buffer
```

The Producer-Consumer Problem

Producer:

```
while (true)
  int value = produce an item;
  buffer[next] = value;
  size++;
  next = (next + 1) mod N;
```

Consumer:

```
while (true)
  int value = buffer[(next - size) mod N];
  size--;
```

The Producer-Consumer Problem

Question

How can we make sure that the producer will not try to add data items to a full buffer?

Question

How can we make sure that the consumer will not try to remove data items from an empty buffer?

Find a Concurrent Algorithm

Before the lecture on Tuesday September 22,

- Find a concurrent algorithm in the literature (paper in journal or conference proceedings).
- Ask me if the paper and algorithm are appropriate. Email the doi of the paper describing the algorithm to me.

Assignment 1

Before the lecture on Thursday October 1, email me your Assignment 1.