

Immigration Course on Formal Methods

Academic year 2022/2023

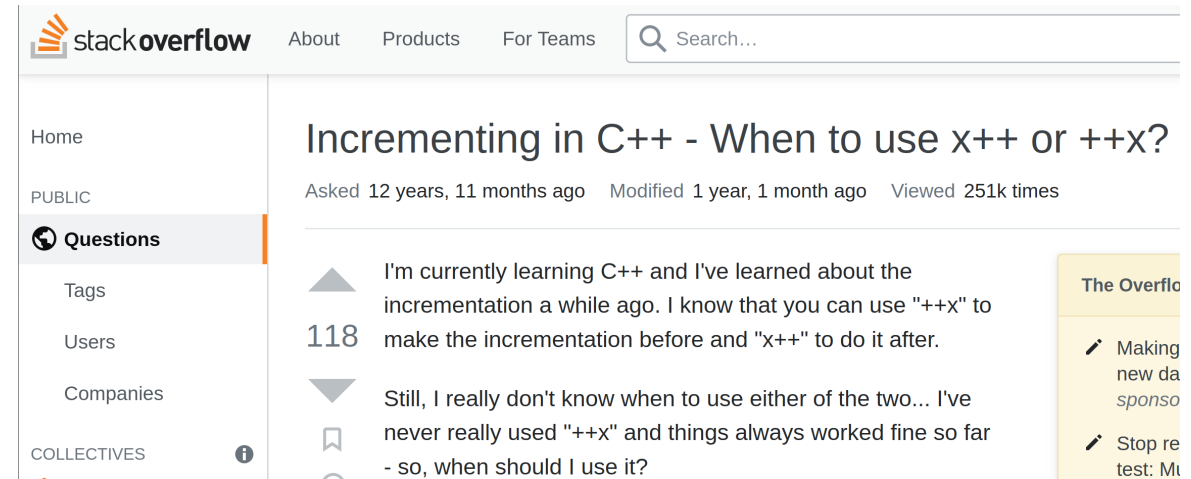
A couple of reasons to be rigorous

A converging **Inclusive Gateway** is used to merge a combination of alternative and parallel paths. A control flow *token* arriving at an **Inclusive Gateway** MAY be synchronized with some other *tokens* that arrive later at this **Gateway**. The precise synchronization behavior of the **Inclusive Gateway** can be found on page 292.

292

[<https://www.omg.org/spec/BPMN/2.0/>]

Business Process Model and Notation, v2.0

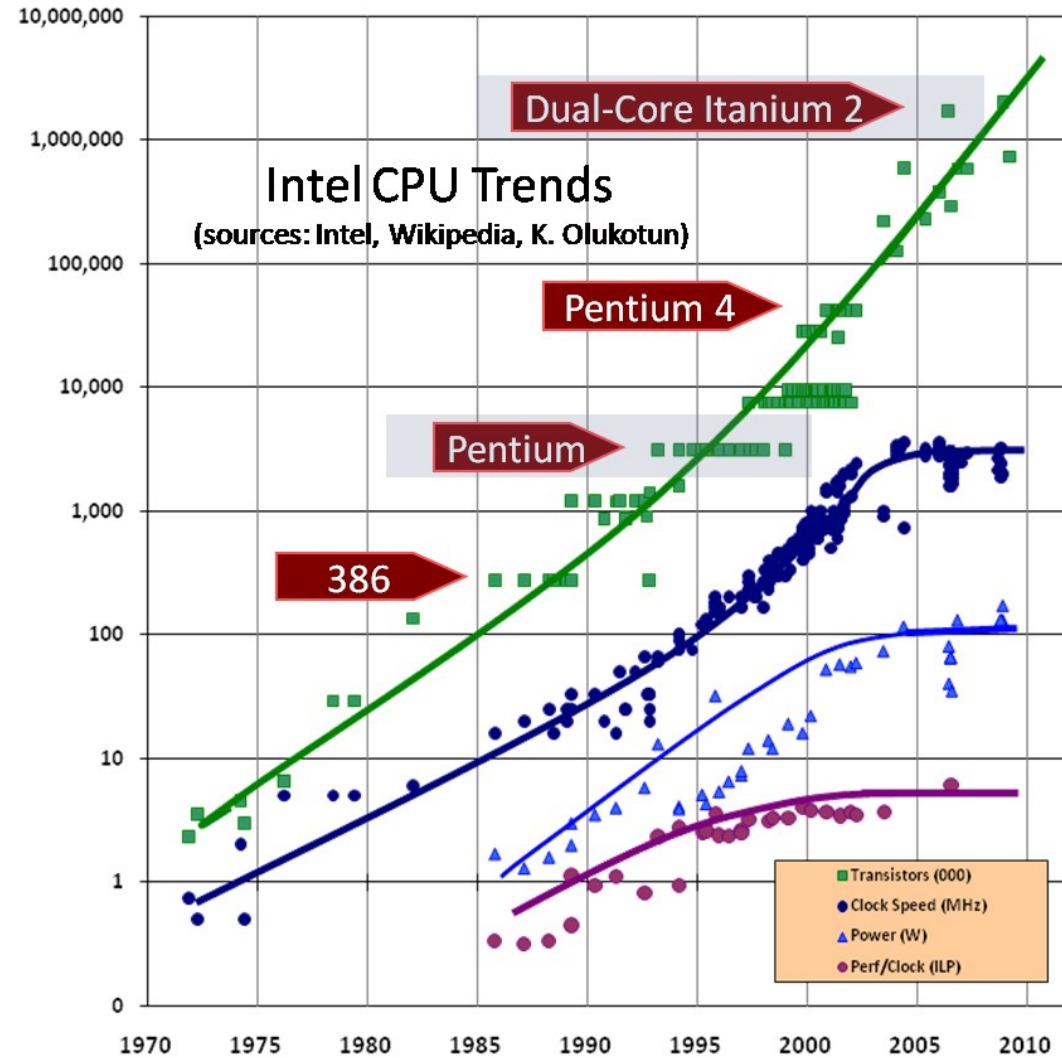


The screenshot shows the Stack Overflow website interface. At the top, the Stack Overflow logo is on the left, and navigation links for 'About', 'Products', and 'For Teams' are on the right, along with a search bar. Below the navigation bar, a left sidebar contains links for 'Home', 'PUBLIC', 'Questions' (which is highlighted), 'Tags', 'Users', 'Companies', and 'COLLECTIVES'. The main content area displays a question titled 'Incrementing in C++ - When to use x++ or ++x?'. Below the title, it indicates the question was 'Asked 12 years, 11 months ago', 'Modified 1 year, 1 month ago', and 'Viewed 251k times'. The question body shows two answers: the first, with an upvote arrow and '118' votes, states 'I'm currently learning C++ and I've learned about the incrementation a while ago. I know that you can use "++x" to make the incrementation before and "x++" to do it after.'; the second answer, with a downvote arrow, states 'Still, I really don't know when to use either of the two... I've never really used "++x" and things always worked fine so far - so, when should I use it?'. On the far right, a yellow sidebar contains a section titled 'The Overflow' with two items: 'Making new da sponso' and 'Stop re test: Mu'.

[<https://stackoverflow.com/questions/1812990/incrementing-in-c-when-to-use-x-or-x>]

A reson to go concurrent

[<http://www.extremetech.com/wp-content/uploads/2012/02/CPU-Scaling.jpg>]



Job interviews and prime numbers

"On the first day of your new job, your boss asks you to find all primes between 1 and 10^{10} (never mind why), using a parallel machine that supports ten concurrent threads. This machine is rented by the minute, so the longer your program takes, the more it costs. You want to make a good impression. What do you do?"

[Herlihy, Shavit: The Art of Multiprocessor Programming. Elsevier, 2012.]

Example: Shared memory

Print all prime integer between 1 & 10^{10}

```
void primeSeq {  
    for (j = 1, j < 10^10; j++) {  
        if (isPrime(j))  
            print(j);  
    }  
}
```

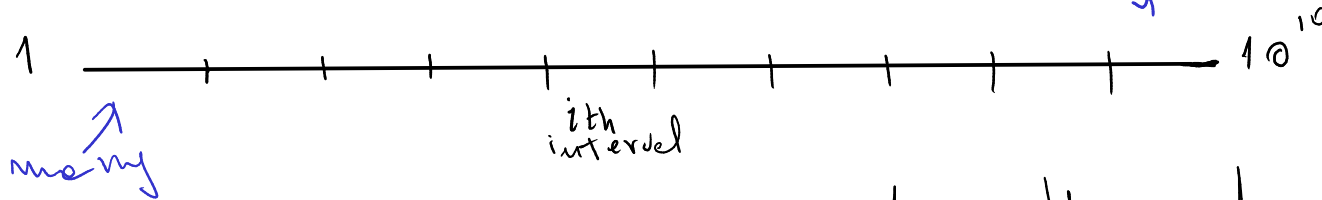
now let's try concurrently



Split the interval and launch a thread on each portion

primes are distributed unevenly

few

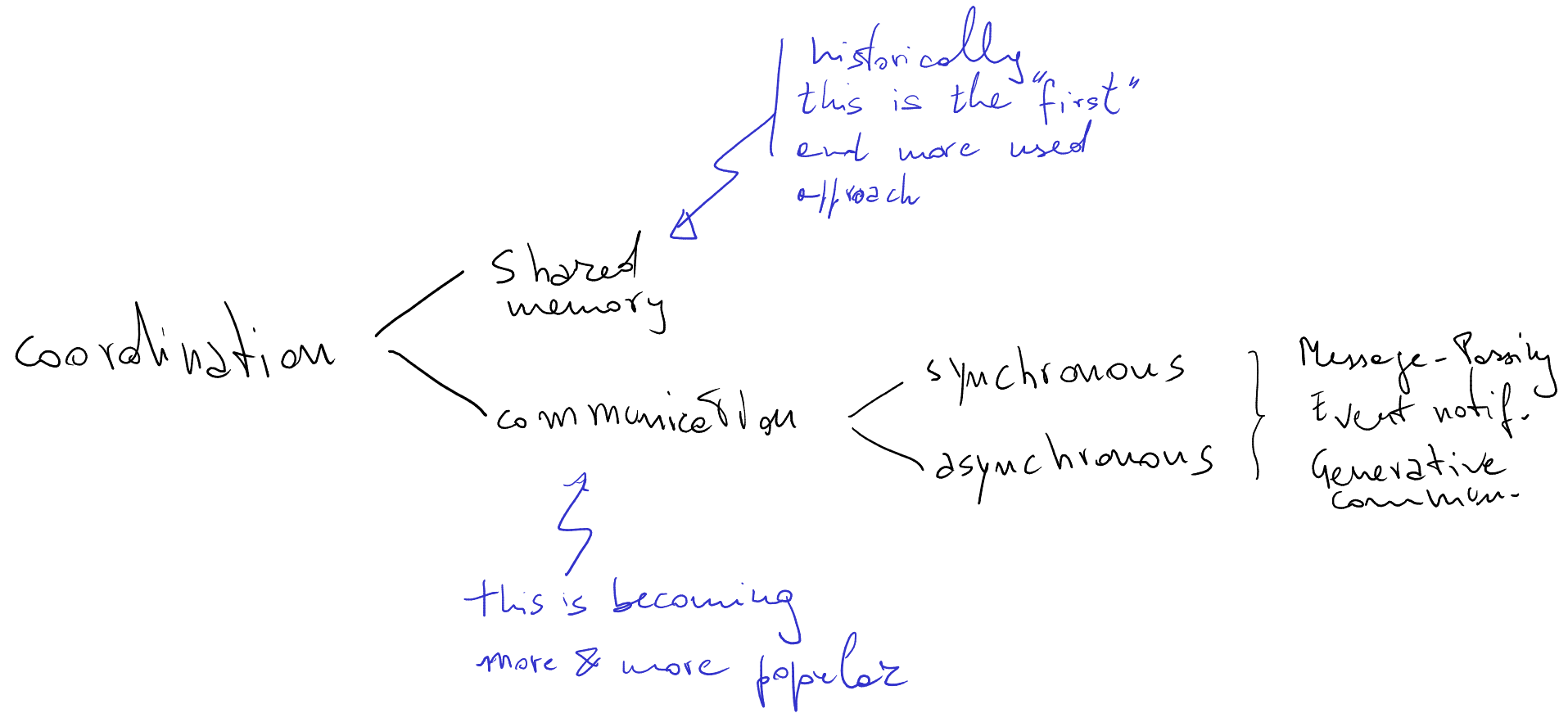


```
void primePrint(int i) { // i non-negative  
    for (j = i*10^9+1, j < (i+1)*10^9; j++) {  
        if (isPrime(j))  
            print(j);  
    }  
}
```

- How good is this idea?

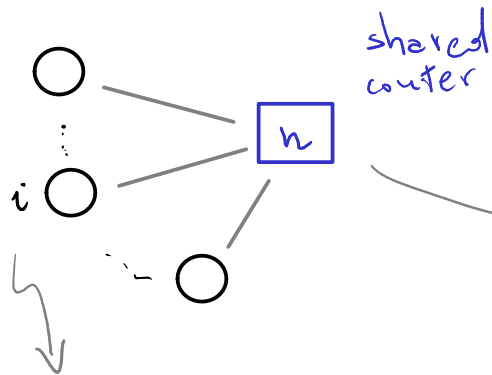
- uneven load

- Is there an "optimal" split?



Exercise 0

Find a better multi-threaded program for the primality test



```
void primePrint( Counter counter ) {  
    long j = 0;  
    while (j < 10^10) {  
        j = counter.getAndIncrement();  
        if (isPrime(j))  
            print(j);  
    }  
}
```

RACES

THIS IS NOT
GOOD!

```
public class Counter {  
    private long value;  
    synchronized  
    public long getAndIncrement() {  
        return value++;  
    }  
}
```

Say something
bad about
JAVA?

temp := value
value ++
return temp

even better

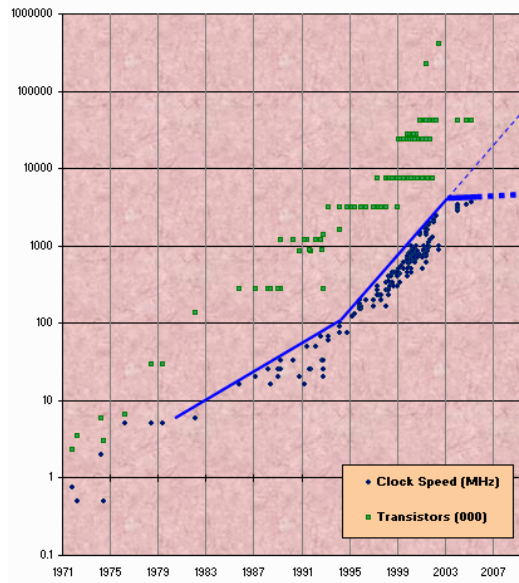
WHY?

```
public long getAndIncrement() {  
    synchronized {  
        temp = value;  
        value = temp + 1;  
    }  
    return temp;  
}
```

REFLECT about why this solution is better than splitting

the art of multi-processor programming

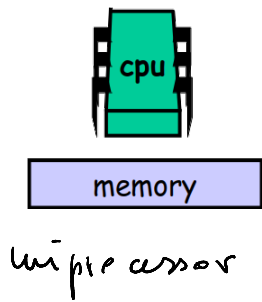
Hw $\xrightarrow{\text{Efficiency is no longer on hw thing}}$ Sw



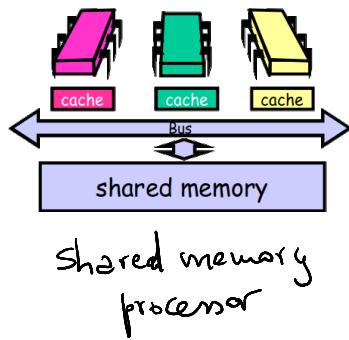
clock speed

transistors grows by a factor of 10 every 10 years

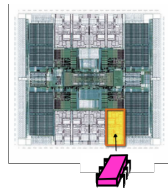
CPU speed is plateauing



uniprocessor



shared memory processor



multicores

- programming constructs in ALL languages
 - "new" languages
 - Go
 - Scala
 - Elixir / Erlang
 - Ballerina
 - Concurmas
 - supporting library, AKKA
 - Modelling languages
 - BPEL
 - BPMN

Same terminology

Concurrency vs Parallelism

compose "independent" stuff

deal with a lot of stuff
AT ONCE

GOAL: "good" composition

run stuff simultaneously

do a lot of stuff
AT ONCE

GOAL: "good" execution

DESIGN

PERFORMANCE

break down problems
&
compose the pieces

Immigration course on formal methods

Emilio Tuosto @ GSSI

Academic year 2022/2023

A glimpse of Erlang

```
ping(N, Pong_PID) ->
  Pong_PID ! {ping, self()},
  receive
    pong ->
      io:format("Ping received pong~n", [])
  end,
  ping(N - 1, Pong_PID).
```

```
ping(0, Pong_PID) ->
  Pong_PID ! finished,
  io:format("ping finished~n", []);
```

```
pong() ->
  receive
    finished ->
      io:format("Pong finished~n", []);
    {ping, Ping_PID} ->
      io:format("Pong received ping~n", []),
      Ping_PID ! pong,
      pong()
  end.
```

```
start() ->
  Pong_PID = spawn(example, pong, []),
  spawn(example, ping, [3, Pong_PID]).
```

Semantics

- Message passing
- FIFO buffers **[[mailboxes in Erlang's jargon]]**
- Spawn of threads

A glimpse of Erlang

```
ping(N, Pong_PID) ->
  Pong_PID ! {ping, self()},
  receive
    pong ->
      io:format("Ping received pong~n", [])
  end,
  ping(N - 1, Pong_PID).
```

```
ping(0, Pong_PID) ->
  Pong_PID ! finished,
  io:format("ping finished~n", []);
```

```
pong() ->
  receive
    finished ->
      io:format("Pong finished~n", []);
    {ping, Ping_PID} ->
      io:format("Pong received ping~n", []),
      Ping_PID ! pong,
      pong()
  end.
```

```
start() ->
  Pong_PID = spawn(example, pong, []),
  spawn(example, ping, [3, Pong_PID]).
```

Semantics

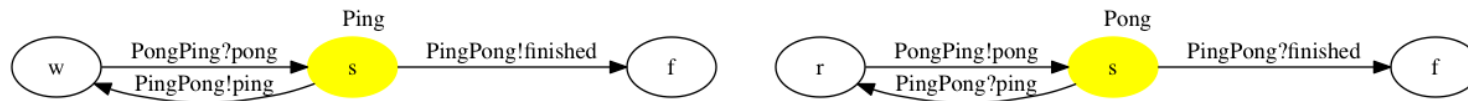
- Message passing
- FIFO buffers **[[mailboxes in Erlang's jargon]]**
- Spawn of threads

Asynchrony by design

Erlang is an embodiment of the well-known **actor model** of Hewitt and Agha...dates back to '73!

Friendlier representations

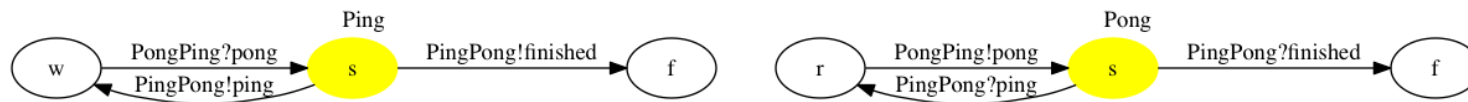
Local behaviour: communicating machines



CFSMs (Brand & Zafiropulo 1983!): FIFO buffers as well

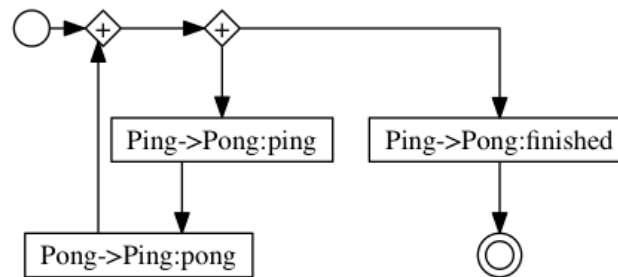
Friendlier representations

Local behaviour: communicating machines



CFSMs (Brand & Zafiropulo 1983!): FIFO buffers as well

Choreography: global graph



...“synchronous” distributed workflow (Deniélou and Yoshida 2012)

A glimpse of Erlang

```
ping(N, Pong_PID) ->
  Pong_PID ! {ping, self()},
  receive
    pong ->
      io:format("Ping received pong~n", [])
  end,
  ping(N - 1, Pong_PID).
```

```
ping(0, Pong_PID) ->
  Pong_PID ! finished,
  io:format("ping finished~n", []);
```

```
pong() ->
  receive
    finished ->
      io:format("Pong finished~n", []);
    {ping, Ping_PID} ->
      io:format("Pong received ping~n", []),
      Ping_PID ! pong,
      pong()
  end.
```

```
start() ->
  Pong_PID = spawn(example, pong, []),
  spawn(example, ping, [3, Pong_PID]),
  spawn(example, ping, [2, Pong_PID]).
```

A glimpse of Erlang

```
ping(N, Pong_PID) ->
  Pong_PID ! {ping, self()},
  receive
    pong ->
      io:format("Ping received pong~n", [])
  end,
  ping(N - 1, Pong_PID).
```

```
ping(0, Pong_PID) ->
  Pong_PID ! finished,
  io:format("ping finished~n", []);
```

```
pong() ->
  receive
    finished ->
      io:format("Pong finished~n", []);
    {ping, Ping_PID} ->
      io:format("Pong received ping~n", []),
      Ping_PID ! pong,
      pong()
  end.
```

```
start() ->
  Pong_PID = spawn(example, pong, []),
  spawn(example, ping, [3, Pong_PID]),
  spawn(example, ping, [2, Pong_PID]).
```

Q:

Is this program correct?

A glimpse of Erlang

```
ping(N, Pong_PID) ->
  Pong_PID ! {ping, self()},
  receive
    pong ->
      io:format("Ping received pong~n", [])
  end,
  ping(N - 1, Pong_PID).
```

```
ping(0, Pong_PID) ->
  Pong_PID ! finished,
  io:format("ping finished~n", []);
```

```
pong() ->
  receive
    finished ->
      io:format("Pong finished~n", []);
    {ping, Ping_PID} ->
      io:format("Pong received ping~n", []),
      Ping_PID ! pong,
      pong()
  end.
```

```
start() ->
  Pong_PID = spawn(example, pong, []),
  spawn(example, ping, [3, Pong_PID]),
  spawn(example, ping, [2, Pong_PID]).
```

Q:

Is this program correct?

A:

No!

Exercise:

find the bug

Send ping-pong to shell !!! ... I mean, use ChoSyn

