Introduction to Formal Methods

Academic year 2025/2026

Emilio Tuosto
emilio.tuosto@gssi.it
https://cs.gssi.it/emilio.tuosto

A couple of resasons to be rigorous

Exercise -1: Describe the behaviour of the program below

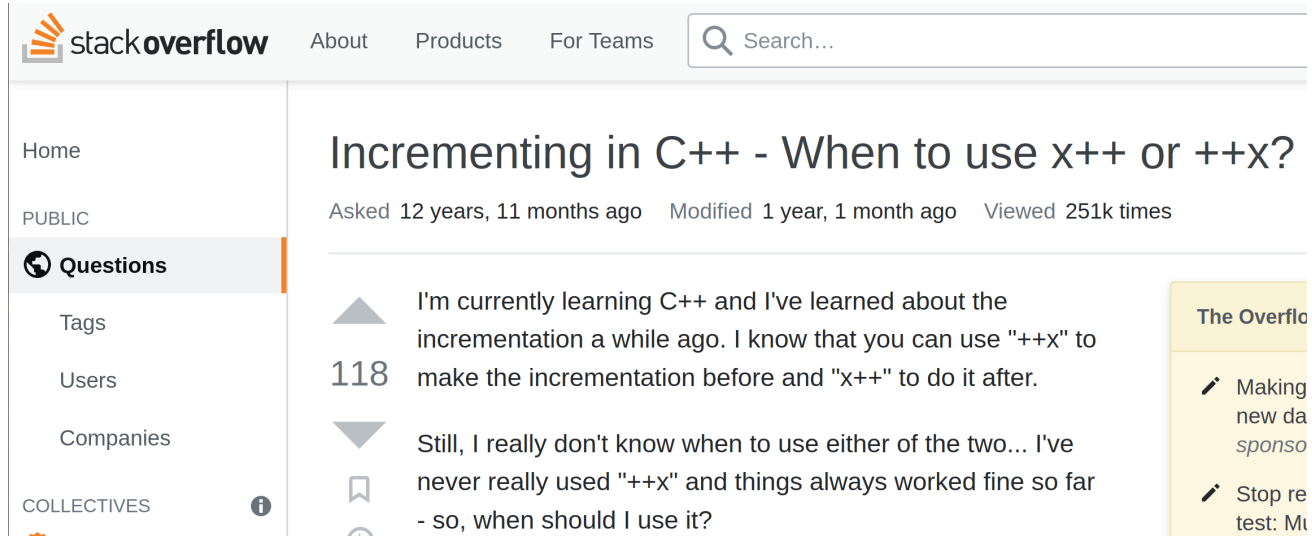
```
print("Start")
a = 0.0
while (a != 100.0):
    a += 0.1
print("moving on")
# ...
```

A couple of resasons to be rigorous

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

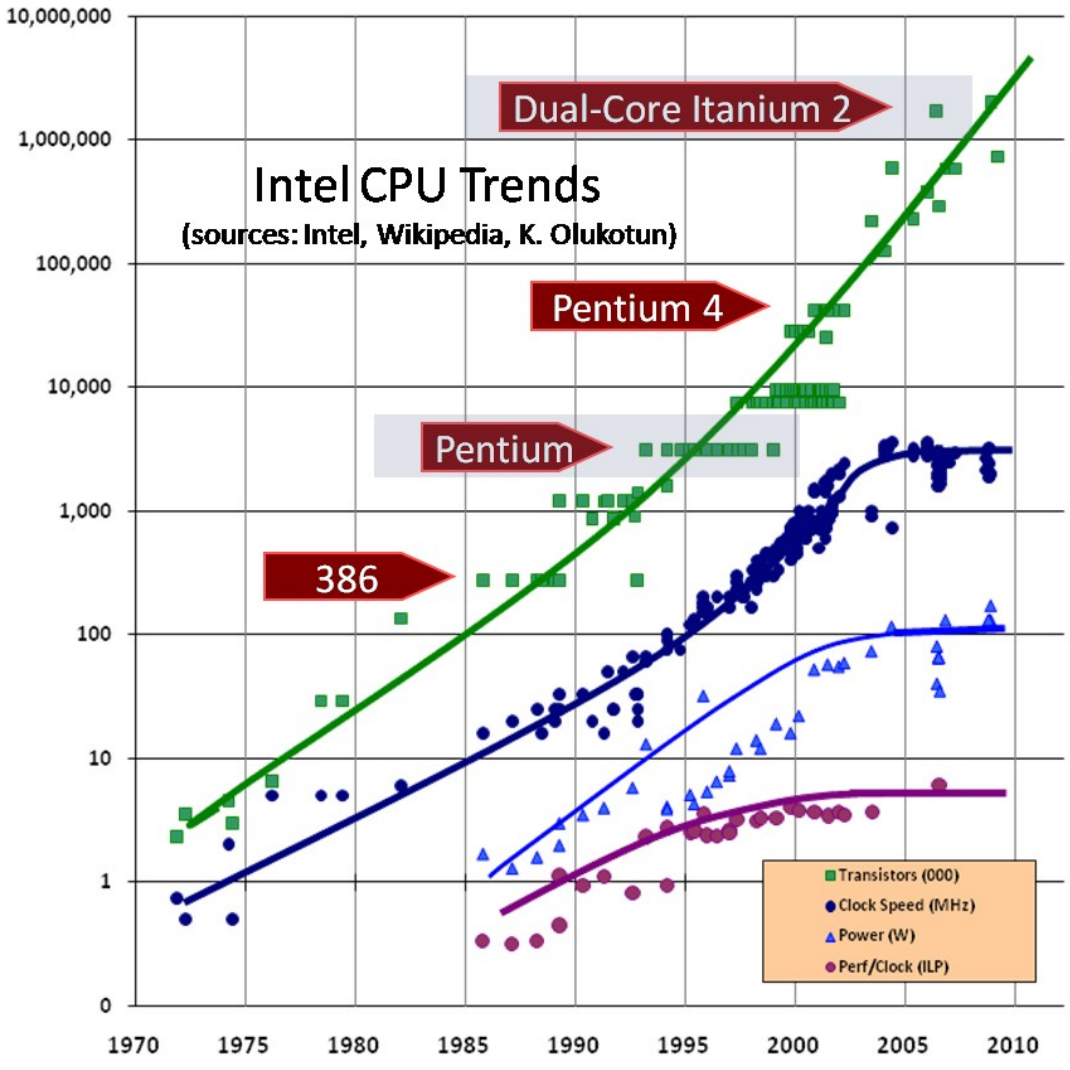
A converging **Inclusive Gateway** is used to merge a combination of alternative and parallel paths. A control flow *token* arriving at an **Inclusive Gateway** MAY be synchronized with some other *tokens* that arrive later at this **Gateway**. The precise synchronization behavior of the **Inclusive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

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

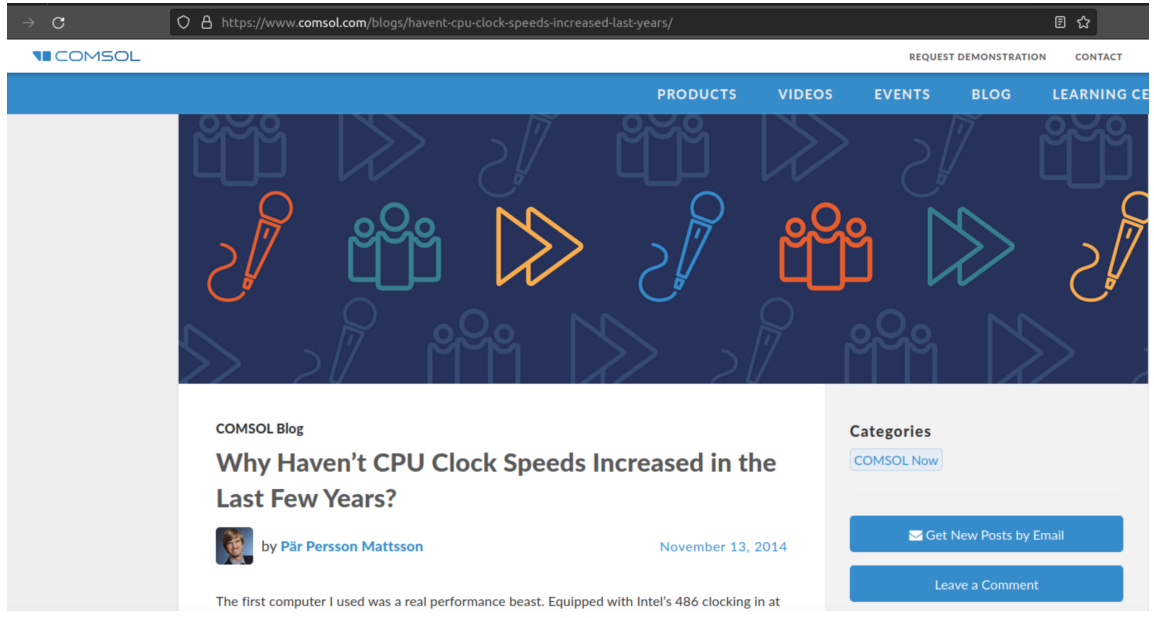


A reason to go concurrent

[https://i.extremetech.com/imagery/content-types/03zc6ghfKswe41smvPXi8Zh/images-6.jpg]

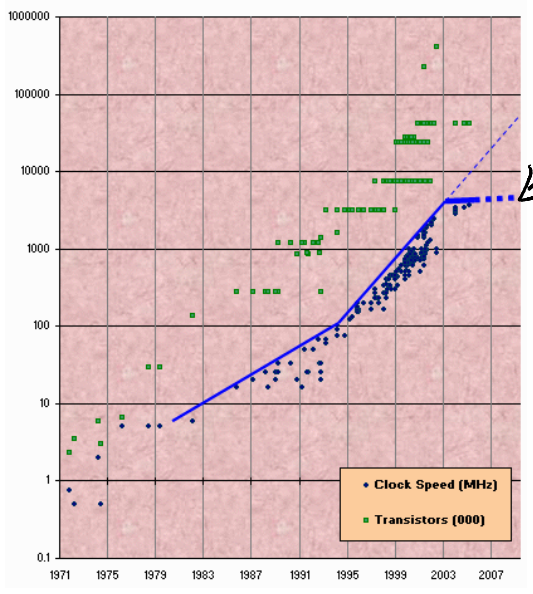


[https://www.comsol.com/blogs/havent-cpu-clock-speeds-increased-last-years]



The art - multi-processor programming

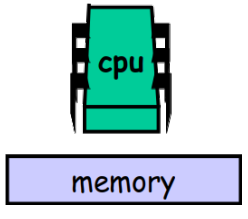
Hw → Efficiency is no longer an "hw thing" → Sw



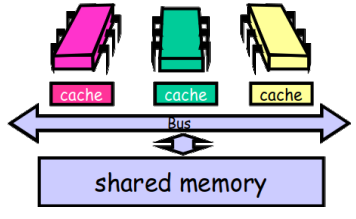
clock speed

transistors grow by a factor of 10 every 10 years!

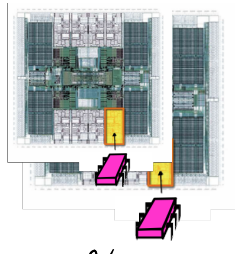
CPU speed is plateauing



uniprocessor



shared-memory processor



multicore

- Programming constructs everywhere
  - "new" languages
    - Golang
    - Scala
    - Erlang / Elixir
    - Ballerina
    - Concurrnas
- Libraries (AKKA)
- Modelling languages
  - BPEL
  - BPMN

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.]

An example of shared memory concurrency

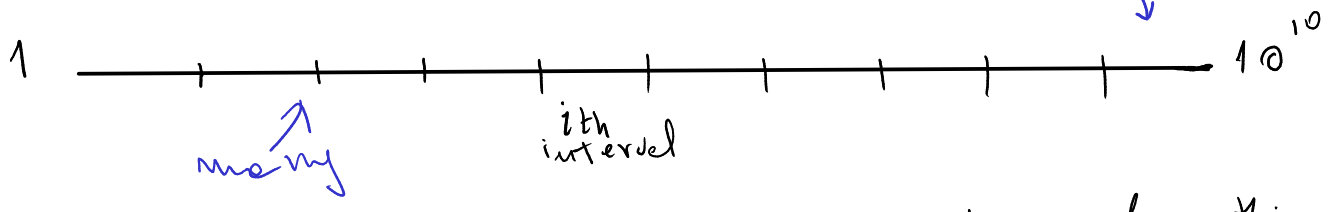
Printing all the prime numbers below 10^{10} ... sequentially

```
1 void primeSeq {
2   for (j = 1, j<10^10; j++) {
3     if (isPrime(j))
4       print(j);
5   }
6 }
```

Now let's try concurrently

Split the interval & launch a thread on each position

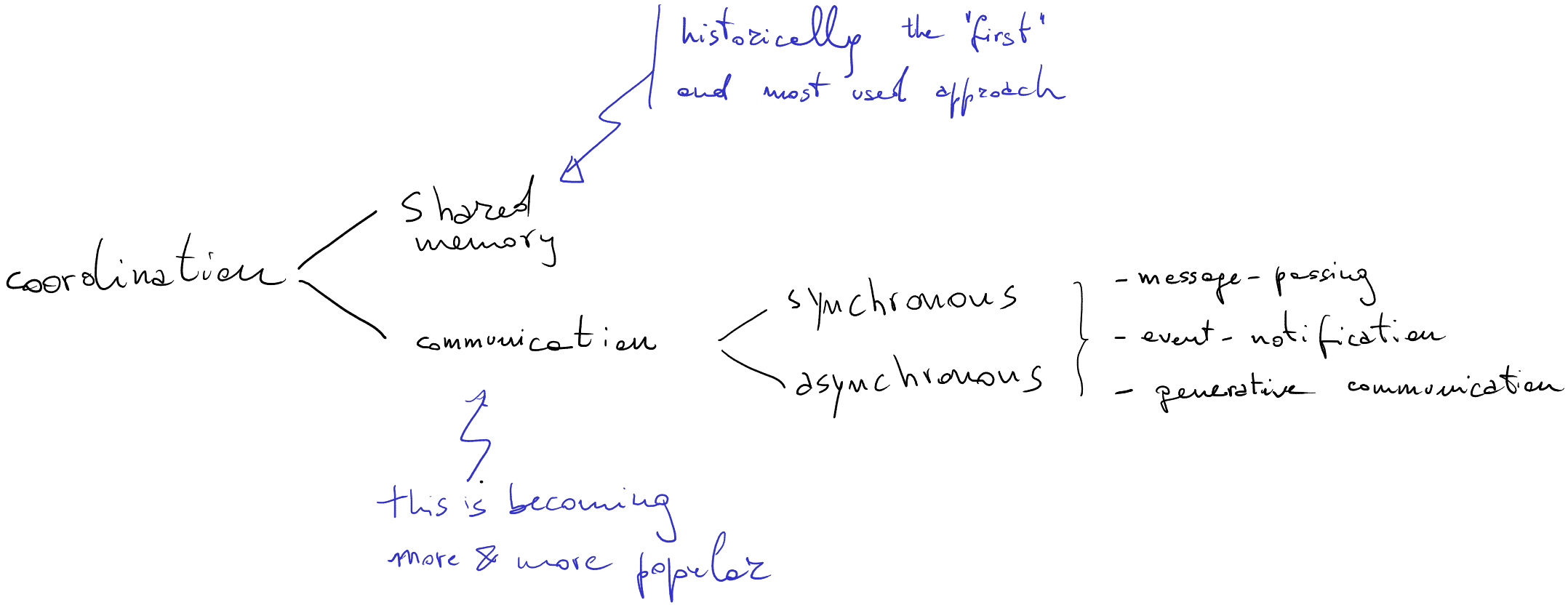
primes are distributed unevenly



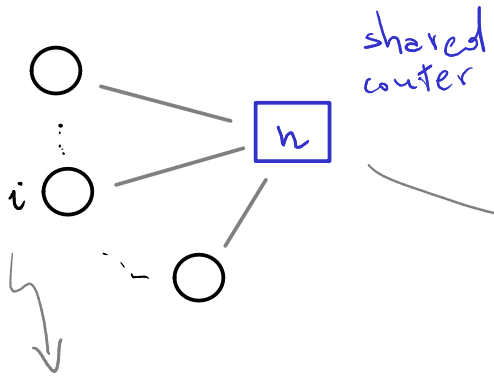

```
void primePrint(int i){ // i non-negative
  for (j = i*10^9+1, j<(i+1)*10^9; j++) {
    if (isPrime(j))
      print(j);
  }
}
```

How good is this idea?

- uneven load
- should we look for an optimal split?


historically the "first"
and most used approach
coordination
shared memory
communication
synchronous
asynchronous
- message-passing
- event-notification
- generative communication
this is becoming
more & more popular

Exercise 0 Find a better multi-threaded program for printing the first 10^{10} primes




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

RACES

synchronised!

THIS IS NOT GOOD!

```
public class Counter {
  private long value;

  public long getAndIncrement() {
    return value++;
  }
}
```

temp := value
value ++
return temp

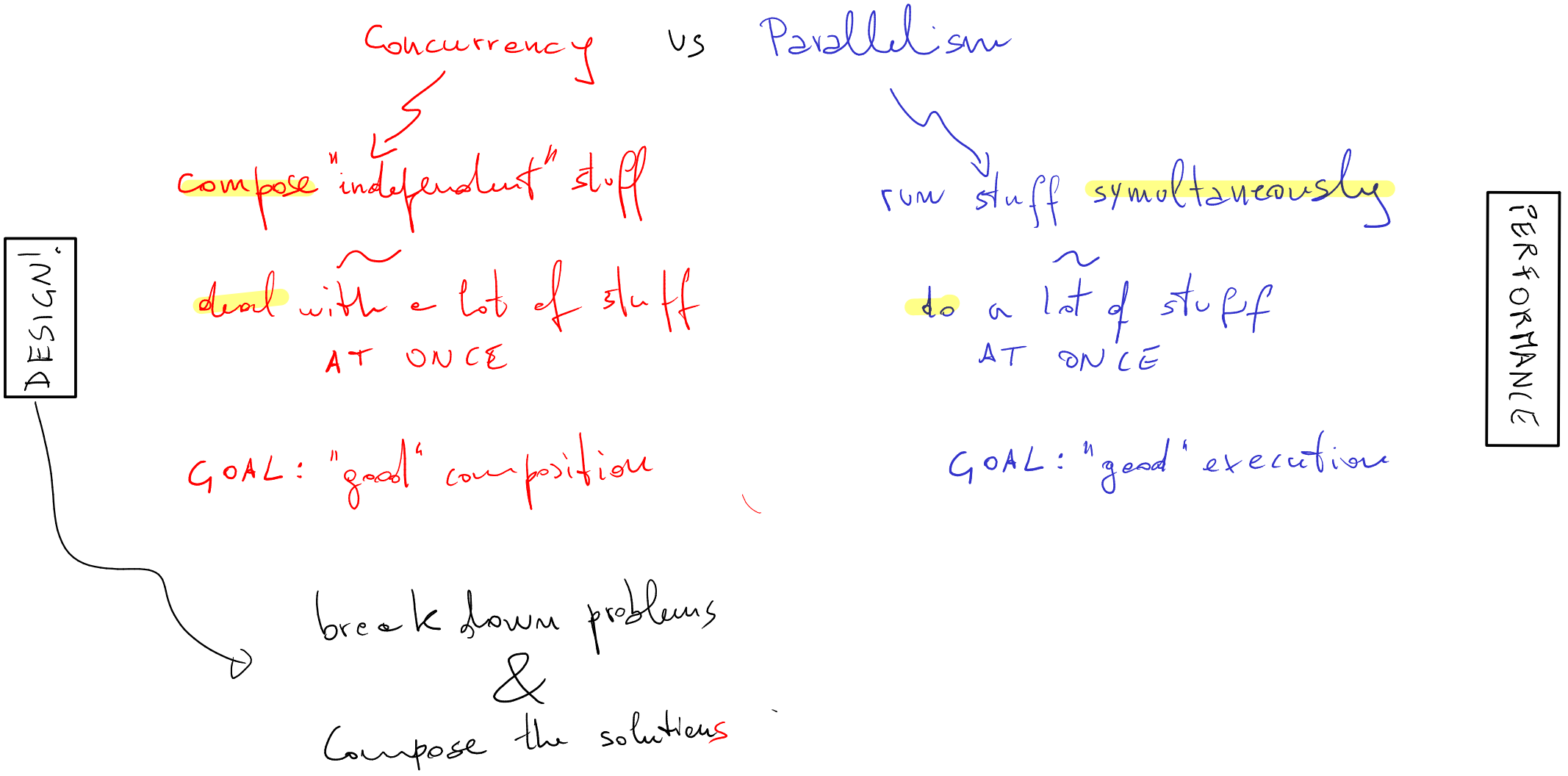
even better

WHY?

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

REFLECT about why this solution is better than splitting

Some terminology



A refresher on minimal maths

$A \subseteq B \quad \forall x \in A.\ x \in B$

$A \subset B \quad A \subseteq B \wedge B \nsubseteq A$

$A = B \quad A \subseteq B \wedge B \supseteq A$

$A \cup B = \{x : x \in A \vee x \in B\}$ $\quad \bigcup_{i \in I} A_i$

$A \cap B = \{x : x \in A \text{ and } x \in B\}$ $\quad \bigcap_{i \in I} A_i$

$A \setminus B = \{x : x \in A \wedge \text{and } x \notin B\}$

$2^A = \{X \mid X \subseteq A\}$

$\prod_{i \in I} A_i$

Relations

$A \times B = \{(a,b) : a \in A \text{ and } b \in B\}$

$a R b \equiv R(a,b) \equiv (a,b) \in R$

$Id_A = \{(a,a) \mid a \in A\}$

Let $R \subseteq A \times B$ (i.e., R is a relation with domain A and codomain B)

- <u>inverse</u> $\quad R^{-1} = \{(b,a) \in B \times A \mid (a,b) \in R\}$
- <u>composition</u> $\quad S \subseteq B \times C \Rightarrow R \circ S = \{(a,c) \in A \times C \mid \exists b \in B : (a,b) \in R \wedge (b,c) \in S\}$

If $A = B$ $\quad R^0 = Id_A \quad R^{n+1} = R \circ R^n$

A refresher on induction

The induction principle is very useful, as you all probably know. Let's refresh it.

.

- Proof method

To show that a property, say P, holds of every natural number n (i.e., to prove P(n) for all n) it suffices to show that

- P(0) is true &
- for all k, P(k) implies P(n+1)

Example: for all n, sum(n) = n(n+1)/2 where sum(k) = 1 + ... + k

- sum(0) = 0 = 0(0+1)/2
- for all k, if sum(k) = k(k+1)/2 then

sum(k+1) = sum(k) + (k+1)		by definition
	= k(k+1)/2 + (k+1)	by inductive hypothesis
	= (k(k+1) + 2(k + 1)) / 2	by arithmetic laws
	= (k + 1)(k+2)/2	by distributivity of multiplication over sum on natural numbers

- Definitional mechanism

To define a set S inductively using a finite number of constructors f1,....,fn each with a finite arity on a set of 'basic elements'

- fix a set I of basic elements (you can think of the elements in I as constructors of arity 0) basis
- if e1,...,ek are in S and f is a constructor of arity k then f(e1,...,ek) is an element of S induction
- nothing else can be an element of S closure

- Example: I ={0} and s(_) is a constructor of arity 1, then the inductively defined set S = {0, s(0), s(s(0)), ...} is isomorphic to natural numbers

(Indeed basis / induction / and closure boil down to the axioms of Peano).