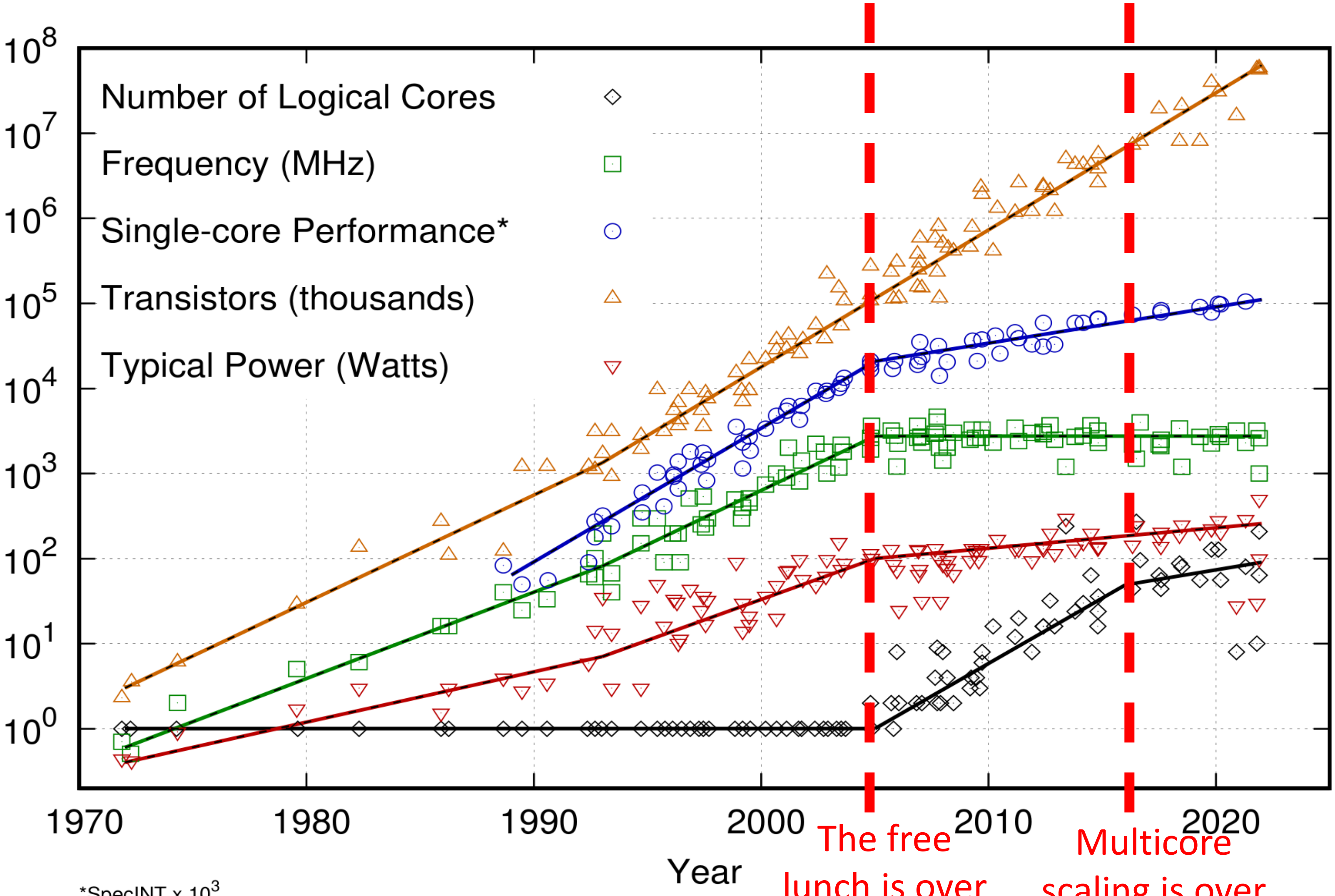
Programmazione concorrente

Laurea Magistrale in Ingegneria Informatica
Università Tor Vergata
Docente: Romolo Marotta

Introduzione

Trend in processor technology



The free lunch is over

Multicore scaling is over

*SpecINT x 10^3
Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2018 by K. Rupp

Trend in processor technology

- Multicore is a standard and established technology
- Applications should be AT LEAST scalable on homogenous cores
  - Necessarily when remote computing power is not available
  - Ideally able to exploit different “kinds” of computing units
- Concurrent and parallel programming is a requirement to exploit current and future hardware

Parallel programming

Ad-hoc concurrent programming languages

- Development tools
  - Compilers
  - MPI, OpenMP, libraries
  - Tools to debug parallel code (gdb, valgrind)
- Writing parallel code is an art
  - There are approaches, not prepackaged solutions
  - Every machine has its own singularities
  - Every problem to face has different requisites
  - The most efficient parallel algorithm might **not** be the most intuitive one

A classical example

INIT

```
Buffer b;
```

PRODUCER

```
while(1) {


    <Write on b>   ☹


}
```

CONSUMER

```
while(1) {


    <Read from b>   ☹


}
```

A classical example

INIT

```
Buffer b;
Semaphore p = 0;
```

PRODUCER

```
while(1) {
    wait(p);

    <Write on b>   ☹

    signal(p);
}
```

CONSUMER

```
while(1) {
    wait(p);

    <Read from b>   ☹

    signal(p);
}
```

A classical example

INIT

```
Buffer b;
Semaphore p = 0;
Semaphore c = 0;
```

PRODUCER

```
while(1) {
  wait(p);

  <Write on b>    ☹

  signal(c);
}
```

CONSUMER

```
while(1) {
  wait(c);

  <Read from b>    ☹

  signal(p);
}
```

A classical example

INIT

```
Buffer b;
Semaphore p = 1;
Semaphore c = 0;
```

PRODUCER

```
while(1) {
  wait(p);

  <Write on b>   ☹

  signal(c);
}
```

CONSUMER

```
while(1) {
  wait(c);

  <Read from b>   ☹

  signal(p);
}
```

Another example

- Challenge
  - Count primes between 1 and 10^8
- Given
  - N threads
  - 1 thread for each logical cpu
- Goals
  - Run N times faster