Parallel Computing

[Floating Pointer] Aritmetic: issues

João Luís Ferreira Sobral
www.di.uminho.pt/~jls
jls@di.uminho.pt

Web: Elearning

Parallel Computing

When 3 * X / 3 is != 1

- **Computers use a finite number of digits to store numbers**
  - Lets assume that we have a 5-digit decimal arithmetic and X=1
    - 3 * (X / 3) = 3 * 0,33333 = 0,99999   != 1,0000
  - Thus, (3 * X) / 3 can be different from 3 * (X / 3)
  - Arithmetic is not commutative when using fixed number of digits
    - Corollary: the result of X/n can be different from X * (1/n)

- **IEEE doubles have 53 significant digits**
  - Corresponds approximately to 16 decimal digits
  - The floating point position is stored in a different set of bits (exponent)

```c
double a = 1.0/3.0;
printf("a is %.20f\n",0.1*a);
printf("a is %.20f\n",a);
printf("a is %.20f\n",10.0*a);
```

```
a is 0. 0333 3333 3333 3333 3287
a is 0. 3333 3333 3333 3333 1483
a is 3. 3333 3333 3333 3330 3727
```

Parallel Computing

Impact of the “finite” number of digits

- **Rounding error accumulate as the order of operations changes**

```
double a = 1.0/3.0;
printf("a is %.20f\n",0.1*(100.0*a));
printf("a is %.20f\n",100.0*(0.1*a));
```

```
a is 3.333 3333 3333 3333 0372 7
a is 3.333 3333 3333 3333 4813 6
```

- **What is an acceptable error?**
  - It depends on the number of floating point operations
  - In our project we require 12 decimal digits
    - Bigger errors will be accepted but should be justified in the report.