

Procesamiento Masivo de datos: Procesos y Hilos

Alex Di Genova

29/08/2022

Outline

- Sistemas distribuidos (repaso)
- Procesos y Hilos

Sistemas Distribuidos

Sistemas distribuidos



Gran cantidad de computadores conectados por una red de alta velocidad.

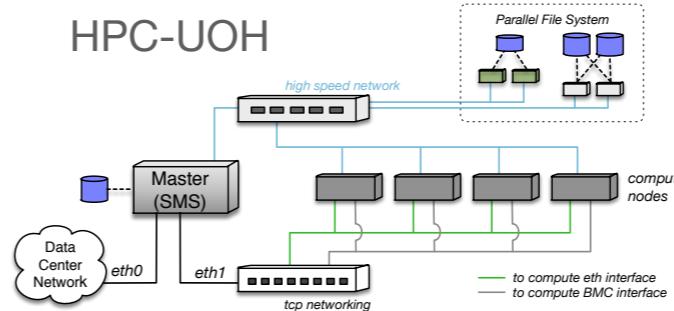
- Un sistema distribuido es una **colección de computadoras** independientes que aparece ante sus usuarios como un **solo sistema coherente**.

Sistemas distribuidos

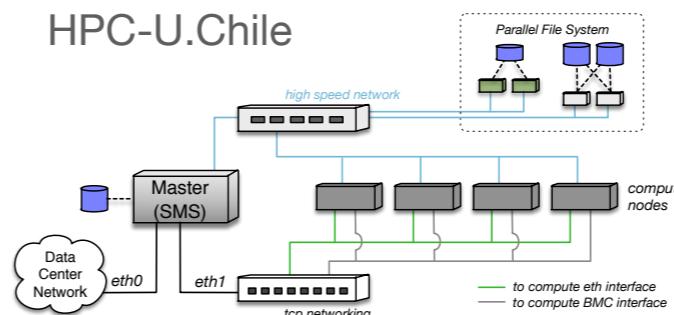
Tipos

- Sistemas de computo distribuido
 - Cluster de computo
 - Hardware similar
 - Red local de alta velocidad
 - Mismo sistema operativo
 - Grid de computo
 - Alto grado de heterogeneidad
 - Federación de sistemas de computo (cluster)

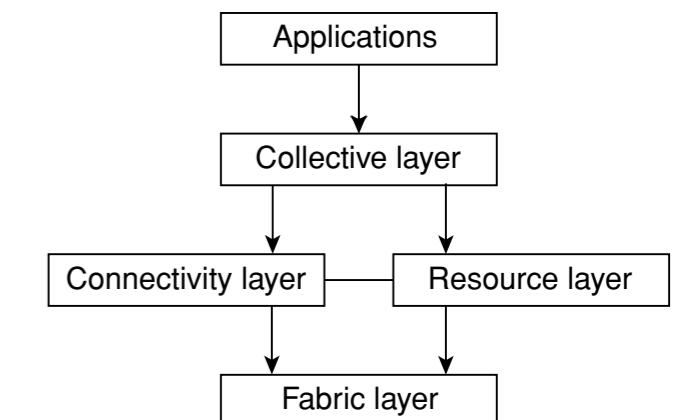
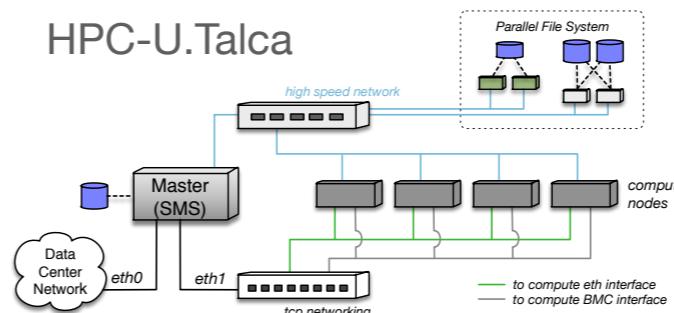
Universidades (organización virtual)



HPC-U.Chile



HPC-U.Talca



- Fabric layer:
 - proporciona interfaces a los recursos locales en un sitio específico
- Connectivity layer
 - protocolos de comunicación para soportar transacciones de red que abarcan el uso de múltiples recursos.
- Resource layer
 - responsable de administrar un solo recurso
- Collective layer:
 - maneja el acceso a múltiples recursos y generalmente consiste en servicios para el descubrimiento de recursos, asignación y programación de tareas

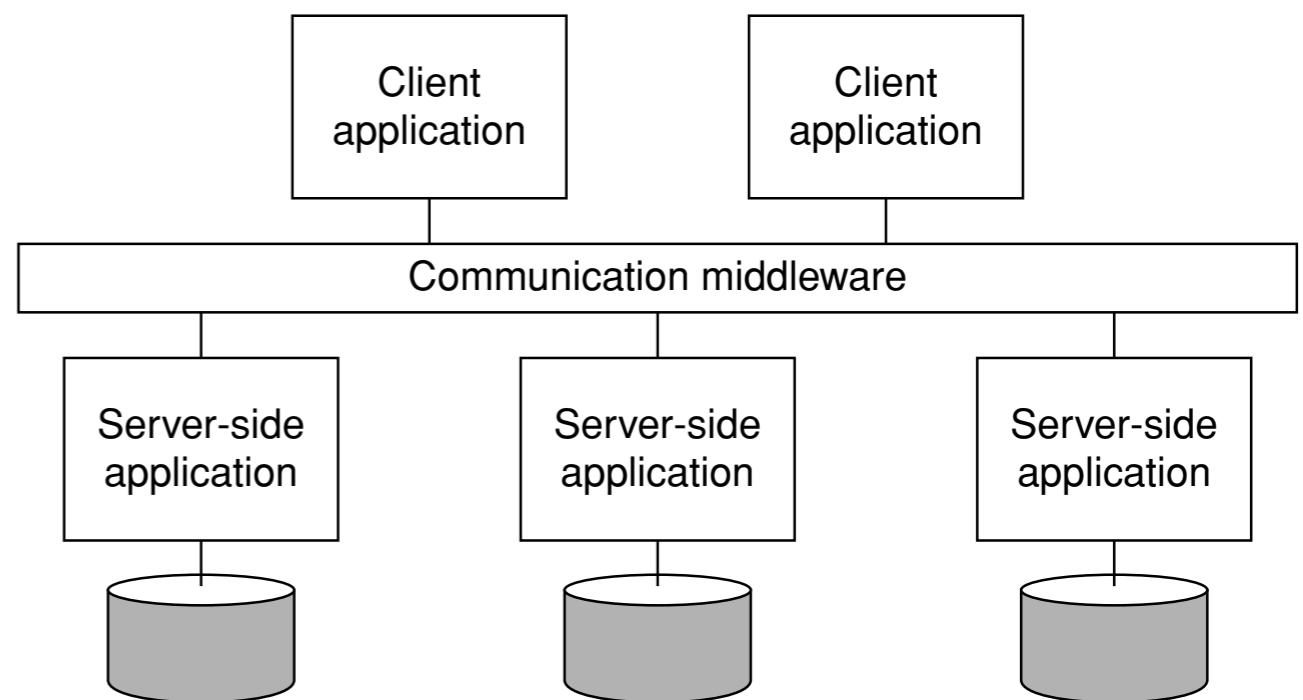


HTCondor
Software Suite

Sistemas distribuidos

Tipos

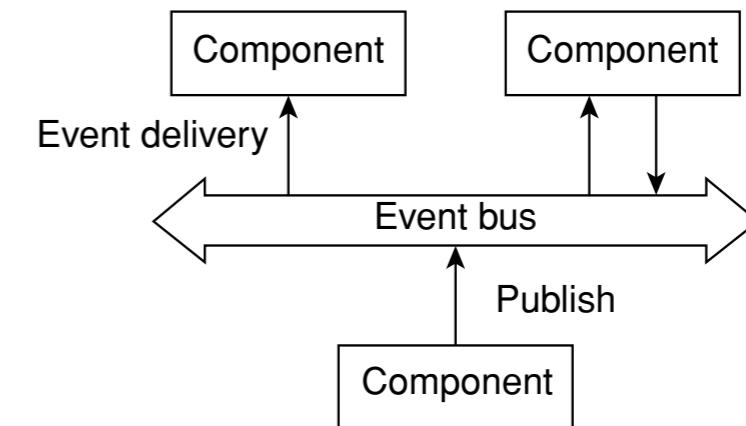
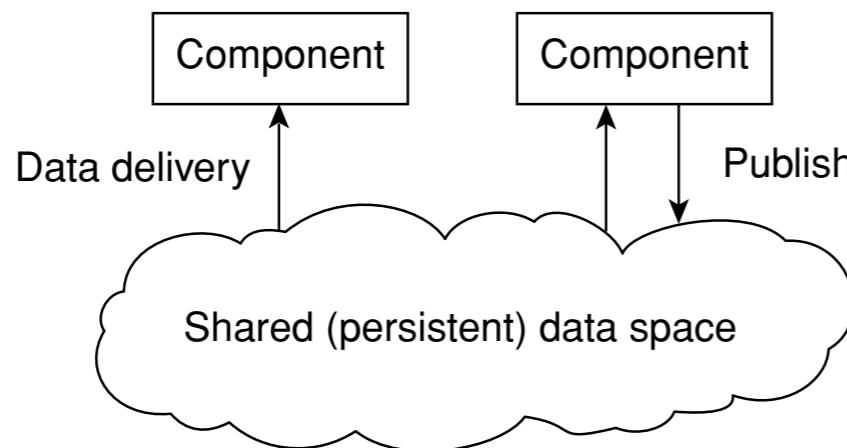
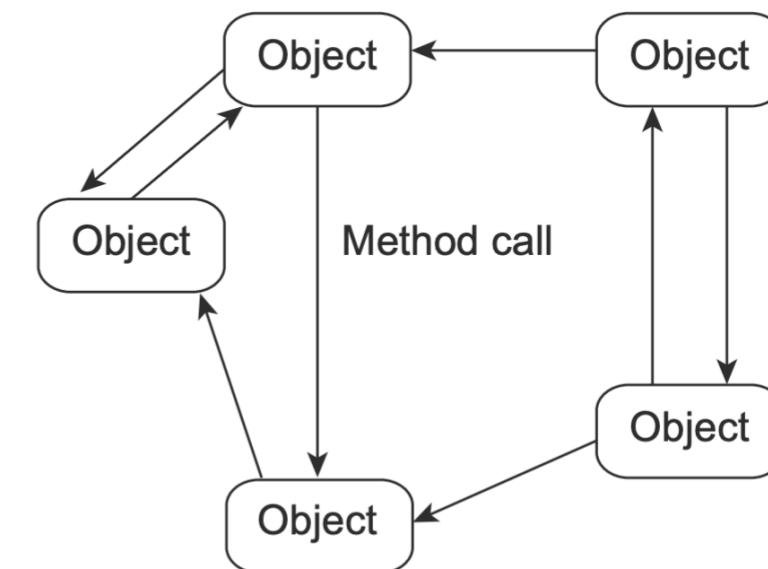
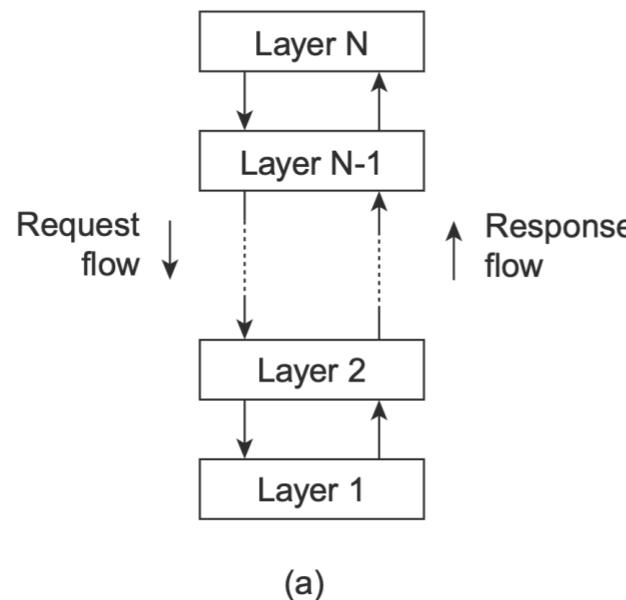
- Sistemas de información distribuido
 - Sistemas de procesamiento de transacciones
 - Sistemas de integración de aplicaciones.
 - Comunicación entre aplicaciones.
 - Idea: aplicaciones intercambien directamente información.
 - Invocaciones remotas de métodos (RMI)



Sistemas distribuidos

Arquitecturas

- Necesitamos definir cómo deben organizarse los diversos componentes de software y cómo deben interactuar.



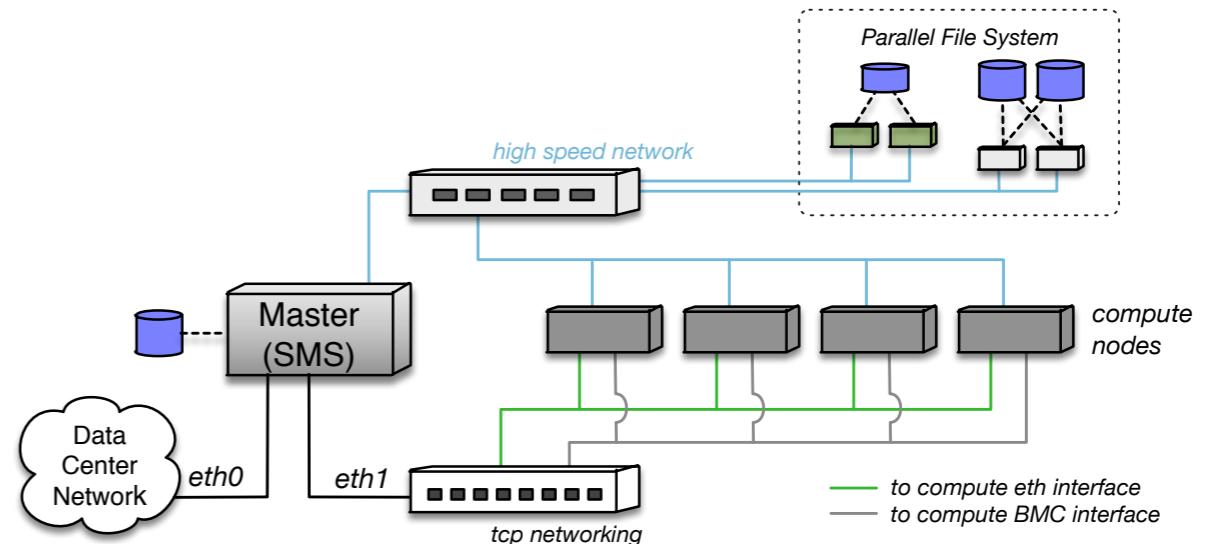
Procesos y Hilos

Sistemas distribuidos

Vista Global



1 Maquina
X cores



1 Cluster
X maquinas
Y Cores
?

- **PTHREAD (hilos)**
- **OpenMP**
- **MPI**
- **Hadoop/Nextflow**

NVIDIA QUADRO P6000
• GPUs 3840
• RAM 24Gb



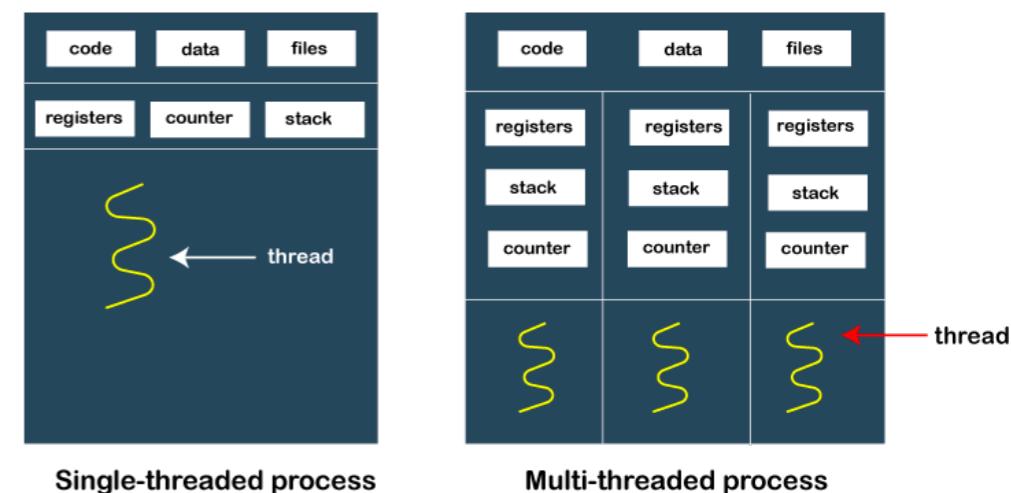
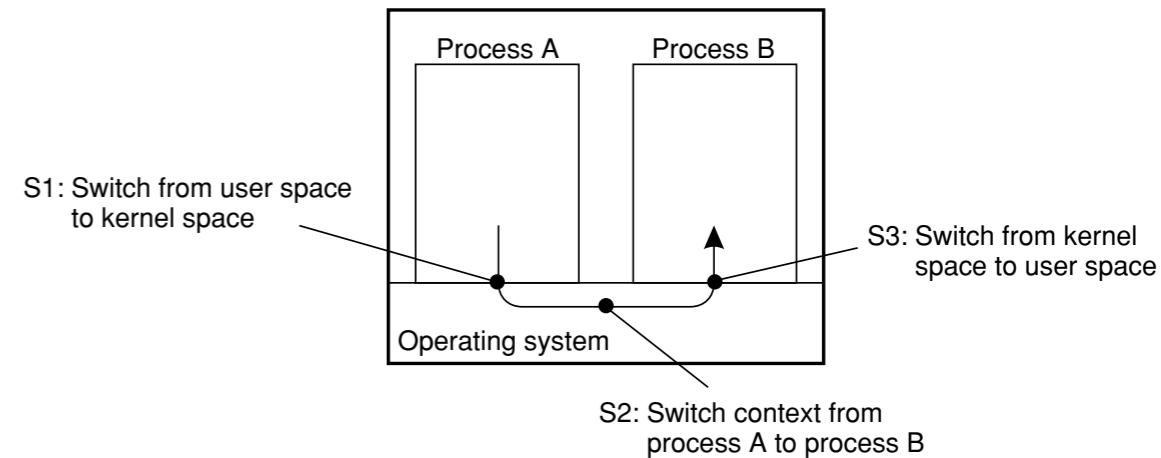
1 Tarjeta
Y Cores
?

- **CUDA**

Sistemas distribuidos

Procesos e Hilos

- Un proceso es un programa que se está ejecutando actualmente en uno de los procesadores virtuales del sistema operativo.
 - Asignación de recursos (memoria, datos temporales [stack], etc)
 - Cambiar CPU entre dos processos.
 - Más procesos que CPUs -> mover procesos de la RAM al disco y viceversa.
 - ¡Solo se puede ejecutar un proceso en la CPU en un momento dado!
- Hilos: procesos livianos.
 - Un proceso puede contener multiples hilos.
 - Ejecutan su propio código.
 - Comparten todos los recursos asignados a un proceso (memoria).
 - Programación de hilos requiere más esfuerzo (programación concurrente y paralela).



Google Colab

The screenshot shows a Google Colab notebook titled "Rust-C-basics.ipynb". The notebook contains two sections: "Hola Mundo" and "Arreglos y funciones".

Hola Mundo:

```
[ ] 1 !apt install rustc cargo

[ ] 1 @@writefile helloworld.rs
2 // This is a comment, and is ignored by the compiler
3
4 // This is the main function
5 fn main() {
6     // Statements here are executed when the compiled binary is called
7     // Print text to the console
8     println!("Hola Mundo!");
9 }
10

Writing helloworld.rs

[ ] 1 !rustc /content/helloworld.rs
2 ./helloworld

[ ] Hola Mundo!
```

Arreglos y funciones:

```
[ ] 1 @@writefile arreglos.rs
2
3 use std::mem;
4
5 // This function borrows a slice
6 fn analyze_slice(slice: &[i32]) {
7     println!("primer elemento del arreglo: {}", slice[0]);
8     println!("el ultimo elemento es: {}", slice[slice.len()-1]);
9     println!("el arreglo tiene {} elementos", slice.len());
10 }

11
12 fn main() {
13     // arreglo de largo fijo
14     let xs: [i32; 5] = [1, 2, 3, 4, 5];
15     // arreglo de largo 500 inicializado en 0
16     let ys: [i32; 500] = [0; 500];
17
18     // indices comienzan en 0
19     println!("primer elemento arreglo: {}", xs[0]);
20     println!("tercer elemento del arreglo: {}", xs[2]);
21
22     // `len` retorna el largo del arreglo
23     println!("numero de elementos arreglo xs: {}", xs.len());
24     println!("numero de elementos arreglo ys: {}", ys.len());
25
26     // Arrays are stack allocated
27     println!("tamaño en memoria xs {} bytes", mem::size_of_val(&xs));
28     println!("tamaño en memoria ys {} bytes", mem::size_of_val(&ys));
29     // Arrays can be automatically borrowed as slices
30     println!("pasamos el arreglo por puntero a función");
31     analyze_slice(&xs);
32     analyze_slice(&ys);
33
34     println!("pasamos una slice del array");
35     analyze_slice(&ys[1 .. 4]);
36
37     // Se puede acceder a las matrices de forma segura mediante `.get`,
38     // que devuelve un
39     // `Option`. Esto se puede combinar como se muestra a continuación, o se puede usar con
40     // `expect()` si desea que el programa finalice con un buen
41     // mensaje en lugar de continuar.
42     for i in 0..xs.len() + 1 { // iteramos un elemento mas del largo
43         match xs.get(i) {
44             Some(xval) => println!("{}: {}", i, xval),
45             None => println!("Fuera de indice! {} no existe!", i),
46         }
47     }
48 }
```

<https://github.com/adigenova/uohpmd>

Consultas?

Consultas o comentarios?

Muchas gracias