Aula 02
Sistemas de HPC

Supercomputação

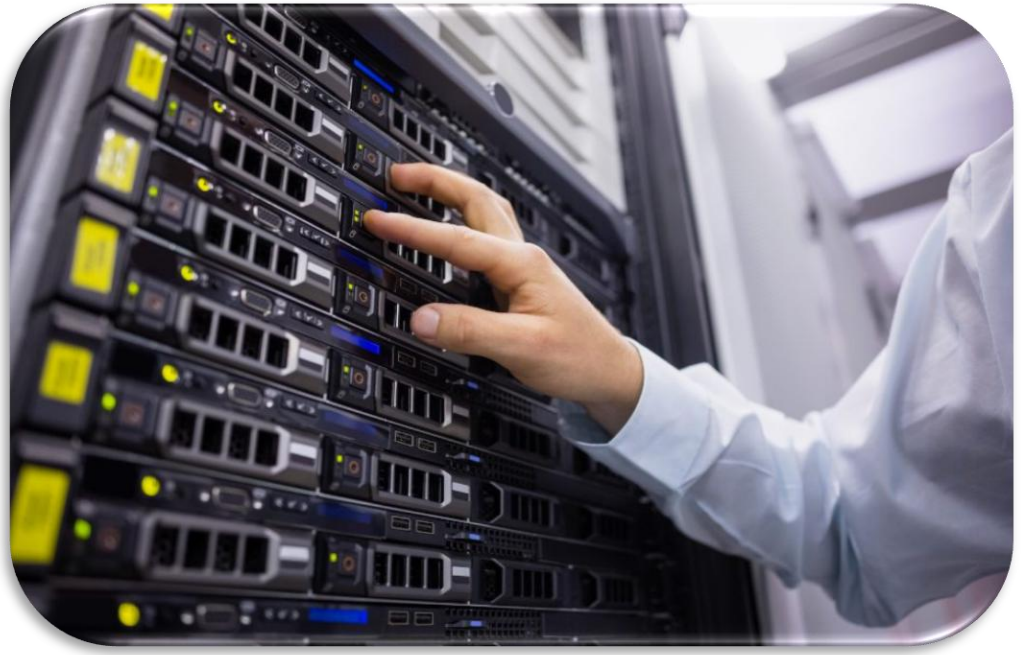
Prof. Lícia Sales Costa Lima

V 1.0

Recapitulando...

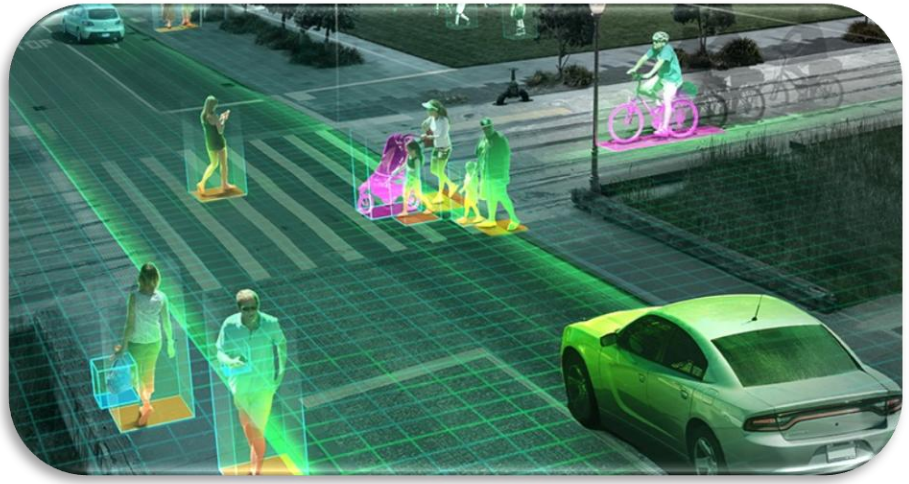
O que é HPC?

- **High-Performance Computing**
- **Computação de alto desempenho**
- **Supercomputação**



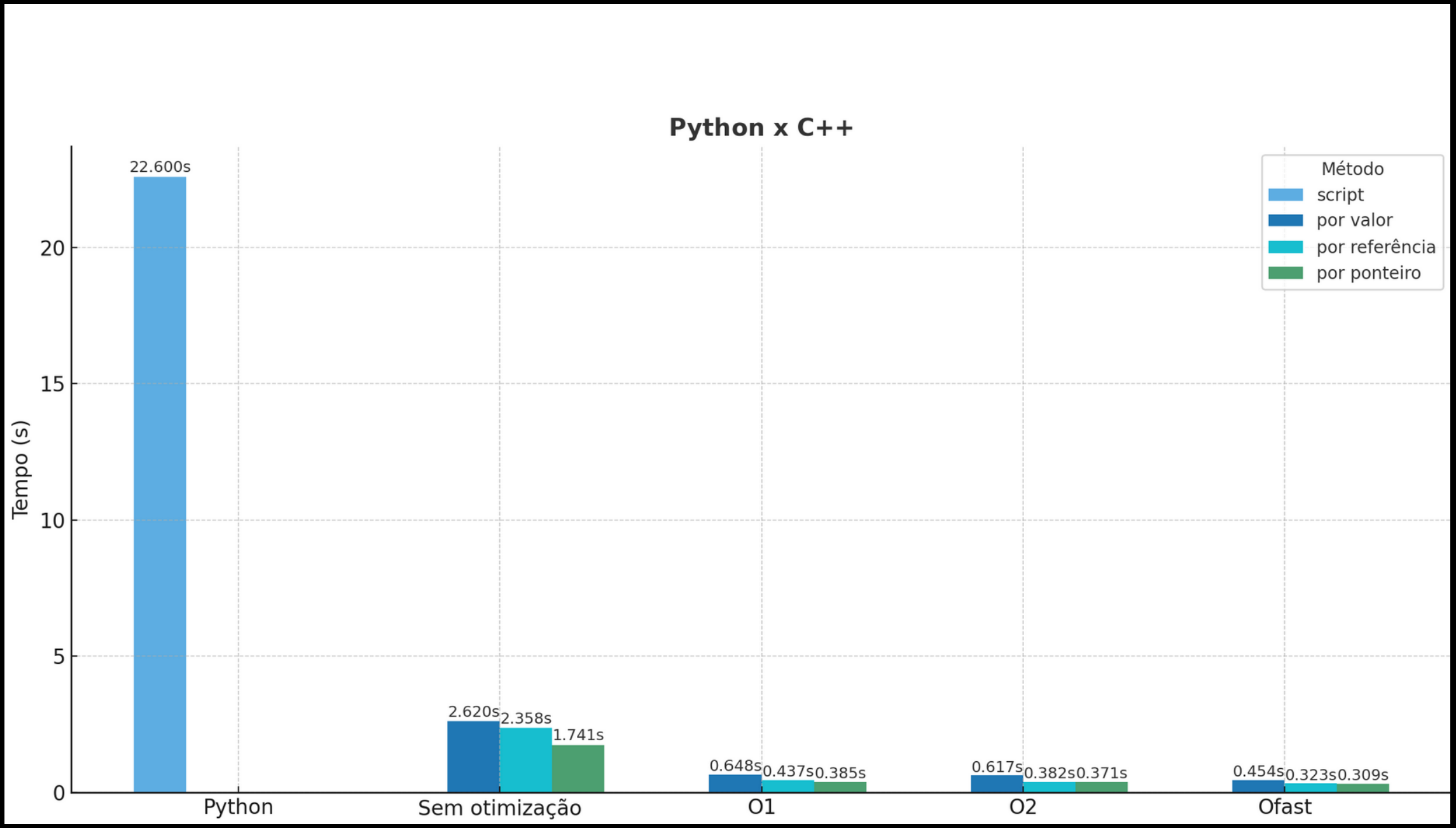
Quais são os Problemas de HPC?

- **Grandes:** uma quantidade de dados absurda, que não cabe em um computador de trabalho comum
- **Intensivos:** Realiza cálculos complexos e demorados, demandando horas ou dias de processamento intensivo
- **Combo:** As vezes o problema tem as duas características, tem uma grande quantidade de dados, demanda cálculos intensivos.



Insper

Por que usar C++?


Python x C++
Método
script
por valor
por referência
por ponteiro
Tempo (s)
0
5
10
15
20
22.600s
2.620s
2.358s
1.741s
0.648s
0.437s
0.385s
0.617s
0.382s
0.371s
0.454s
0.323s
0.309s
Python
Sem otimização
O1
O2
Ofast

Insper

Falando em Referências

```cpp
int first_v(std::vector<int> a){
    return a[0];
}

int first_r(std::vector<int> &a){
    return a[0];
}

int first_p(std::vector<int> *a){
    return (*a)[0];
}
```

Como fica o ASM dessas funções?

Falando em Referências

```
first_p(std::vector<int, std::allocator<int>>*):
        push    rbp
        mov     rbp, rsp
        sub     rsp, 16
        mov     QWORD PTR [rbp-8], rdi
        mov     rax, QWORD PTR [rbp-8]
        mov     esi, 0
        mov     rdi, rax
        call    std::vector<int, std::allocator<int>>::
        mov     eax, DWORD PTR [rax]
        leave
        ret
```

```
first_r(std::vector<int, std::allocator<int>>&):
        push    rbp
        mov     rbp, rsp
        sub     rsp, 16
        mov     QWORD PTR [rbp-8], rdi
        mov     rax, QWORD PTR [rbp-8]
        mov     esi, 0
        mov     rdi, rax
        call    std::vector<int, std::allocator<int>>:
        mov     eax, DWORD PTR [rax]
        leave
        ret
```

```
first_v(std::vector<int, std::allocator<int>>):
        push    rbp
        mov     rbp, rsp
        sub     rsp, 16
        mov     QWORD PTR [rbp-8], rdi
        mov     rax, QWORD PTR [rbp-8]
        mov     esi, 0
        mov     rdi, rax
        call    std::vector<int, std::allocator<int>>:
        mov     eax, DWORD PTR [rax]
        leave
        ret
```

E cadê as diferenças de passar por valor ou referência?

Insper

Falando em Referências

O overhead aparece quando a main chama as funções

```
lea     rax, [rbp-96]
mov     rdi, rax
call    first_p(std::vector<int, std::allocator<int>>*)
mov     DWORD PTR [rbp-28], eax
~
~
~
~
~
~
~
~
~
call_p.asm [+]                1,1           All
```

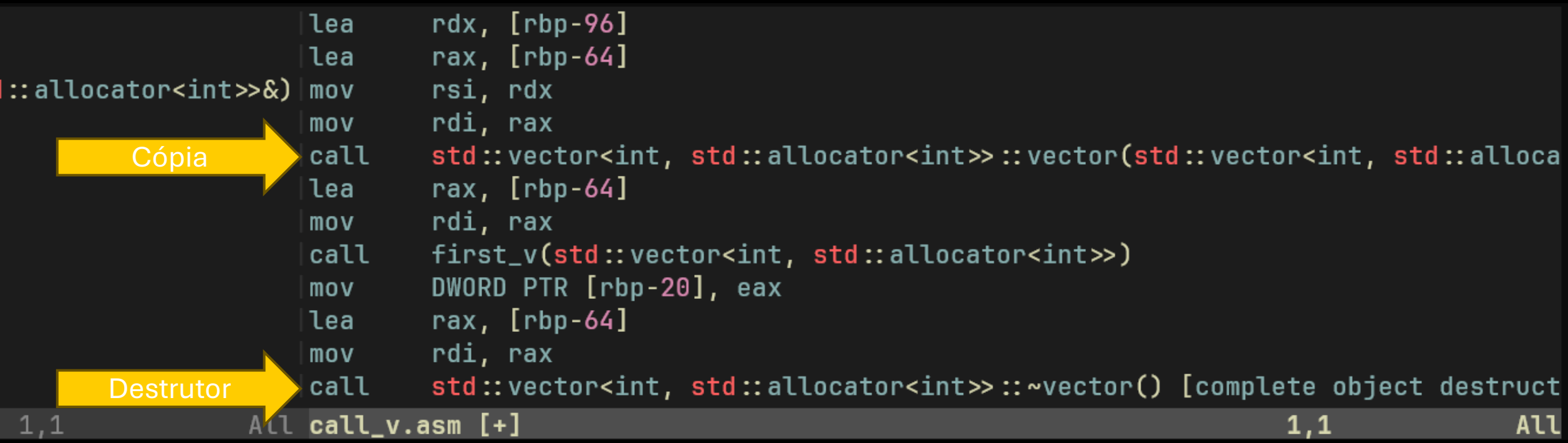
```
lea     rax, [rbp-96]
mov     rdi, rax
call    first_r(std::vector<int, std::allocator<int>>&)
mov     DWORD PTR [rbp-24], eax
~
~
~
~
~
~
~
~
~
call_r.asm [+]                1,1           All
```

```
lea     rdx, [rbp-96]
lea     rax, [rbp-64]
mov     rsi, rdx
mov     rdi, rax
call    std::vector<int, std::allocator<int>>::vector(std::vector<int, std::alloca
lea     rax, [rbp-64]
mov     rdi, rax
call    first_v(std::vector<int, std::allocator<int>>)
mov     DWORD PTR [rbp-20], eax
lea     rax, [rbp-64]
mov     rdi, rax
call    std::vector<int, std::allocator<int>>::~vector() [complete object destruct
call_v.asm [+]                                1,1           All
```

Insper

Falando em Referências

O overhead aparece quando a main chama as funções



Falando em Referências

Referências e ponteiros são equivalentes*

```
lea     rax, [rbp-96]
mov     rdi, rax
call    first_p(std::vector<int, std::allocator<int>>*)
mov     DWORD PTR [rbp-28], eax
~
~
~
~
~
~
~
~
~
call_p.asm [+]                1,1           All
```

```
lea     rax, [rbp-96]
mov     rdi, rax
call    first_r(std::vector<int, std::allocator<int>>&)
mov     DWORD PTR [rbp-24], eax
~
~
~
~
~
~
~
~
~
call_r.asm [+]                1,1           All
```

* Existem casos específicos que se faz necessário o uso ponteiros

Insper

HPC no Mundo

Insper

HPC no mundo

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**El Capitan** - HPE Cray EX255a, AMD 4th Gen EPYC 24C 1.8GHz, AMD Instinct MI300A, Slingshot-11, TOSS, HPE DOE/NNSA/LLNL United States	11,039,616	1,742.00	2,746.38	29,581
2	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE Cray OS, HPE DOE/SC/Oak Ridge National Laboratory United States	9,066,176	1,353.00	2,055.72	24,607
3	**Aurora** - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory United States	9,264,128	1,012.00	1,980.01	38,698

Insper

HPC no Brasil

HPC no Brasil

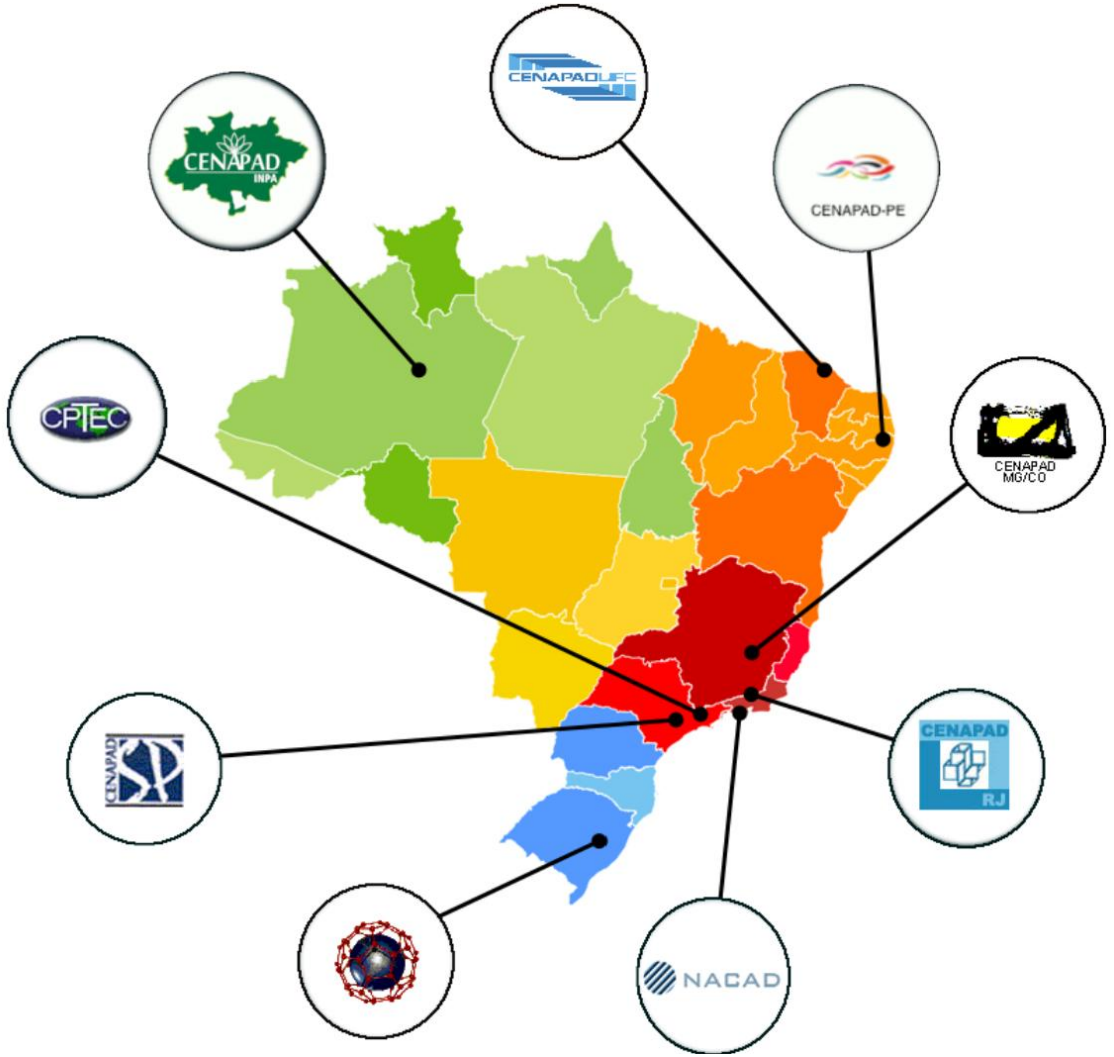
Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
86	**Pégaso** - Supermicro A+ Server 4124GO-NART+, AMD EPYC 7513 32C 2.6GHz, NVIDIA A100, Infiniband HDR, EVIDEN Petróleo Brasileiro S.A Brazil	233,856	19.07	42.00	1,033
107	**Santos Dumont** - BullSequana XH3000, Grace Hopper Superchip 72C 3GHz, NVIDIA GH200 Superchip, Quad-Rail NVIDIA InfiniBand NDR200, Red Hat Enterprise Linux, EVIDEN Laboratório Nacional de Computação Científica Brazil	68,064	14.29	20.26	312
160	**Dragão** - Supermicro SYS-4029GP-TVRT, Xeon Gold 6230R 26C 2.1GHz, NVIDIA Tesla V100, Infiniband EDR, EVIDEN Petróleo Brasileiro S.A Brazil	188,224	8.98	14.01	943
193	**Gaia** - PowerEdge XE8545, AMD EPYC 74F3 24C 3.2GHz, NVIDIA A100, Infiniband, DELL Petróleo Brasileiro S.A Brazil	84,480	6.97	13.73	574

Insper

SINAPAD

Sistema Nacional de Processamento de Alto Desempenho

SINAPAD Centros Portais Projetos Recursos

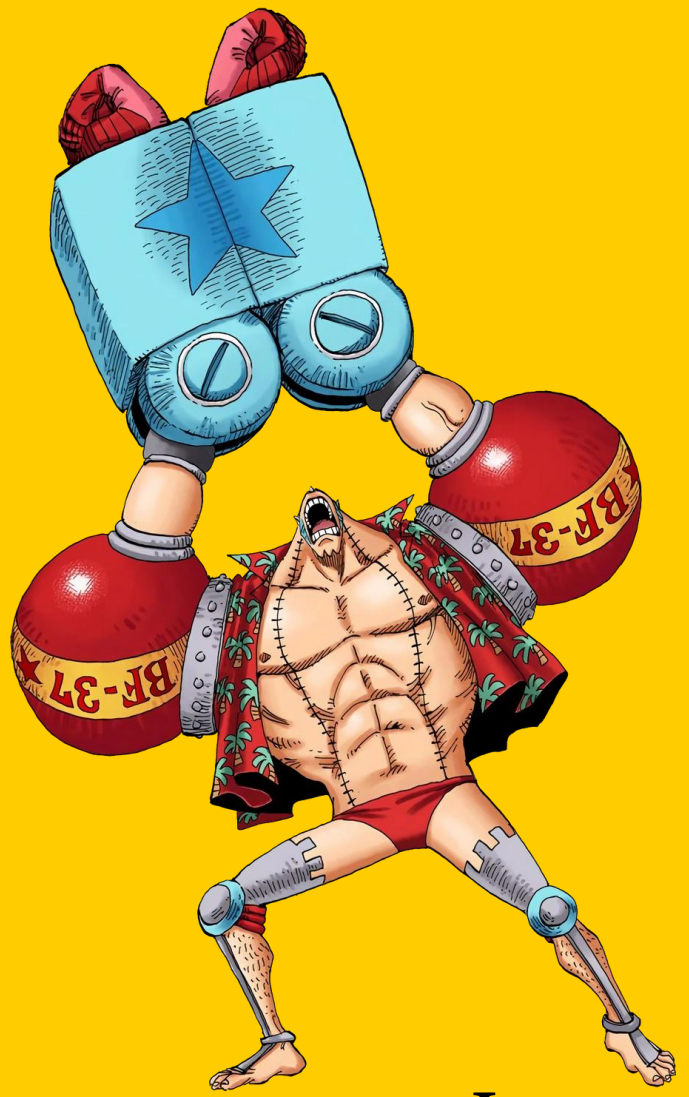


Insper

Está no LNCC/MCTI também

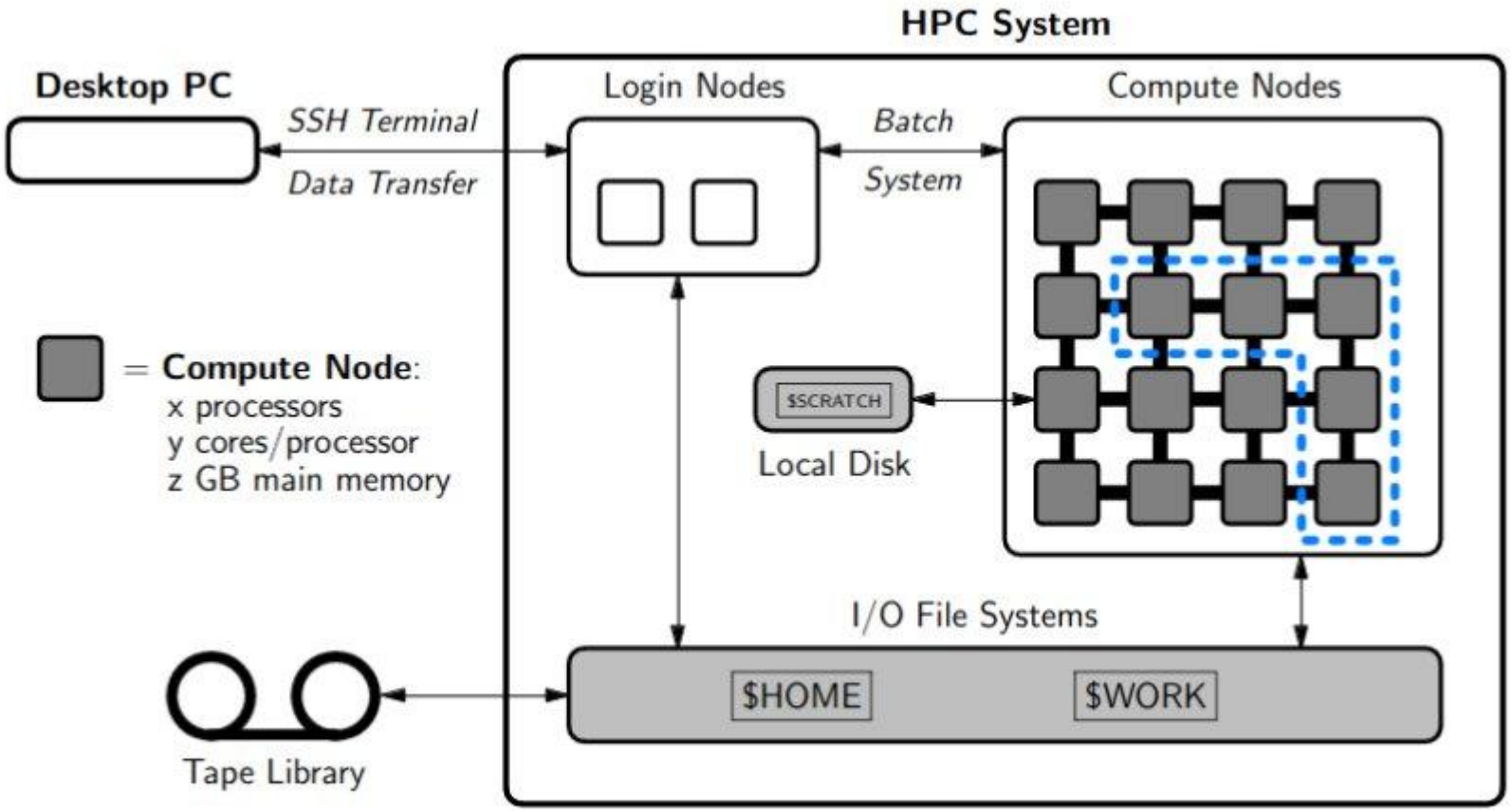
E no Insper?

SUUUUUUPER!!!



Insper

Arquitetura HPC tradicional



Fonte: https://www.rz.uni-kiel.de/de/angebote/hiperf/hpc-course-12feb2020

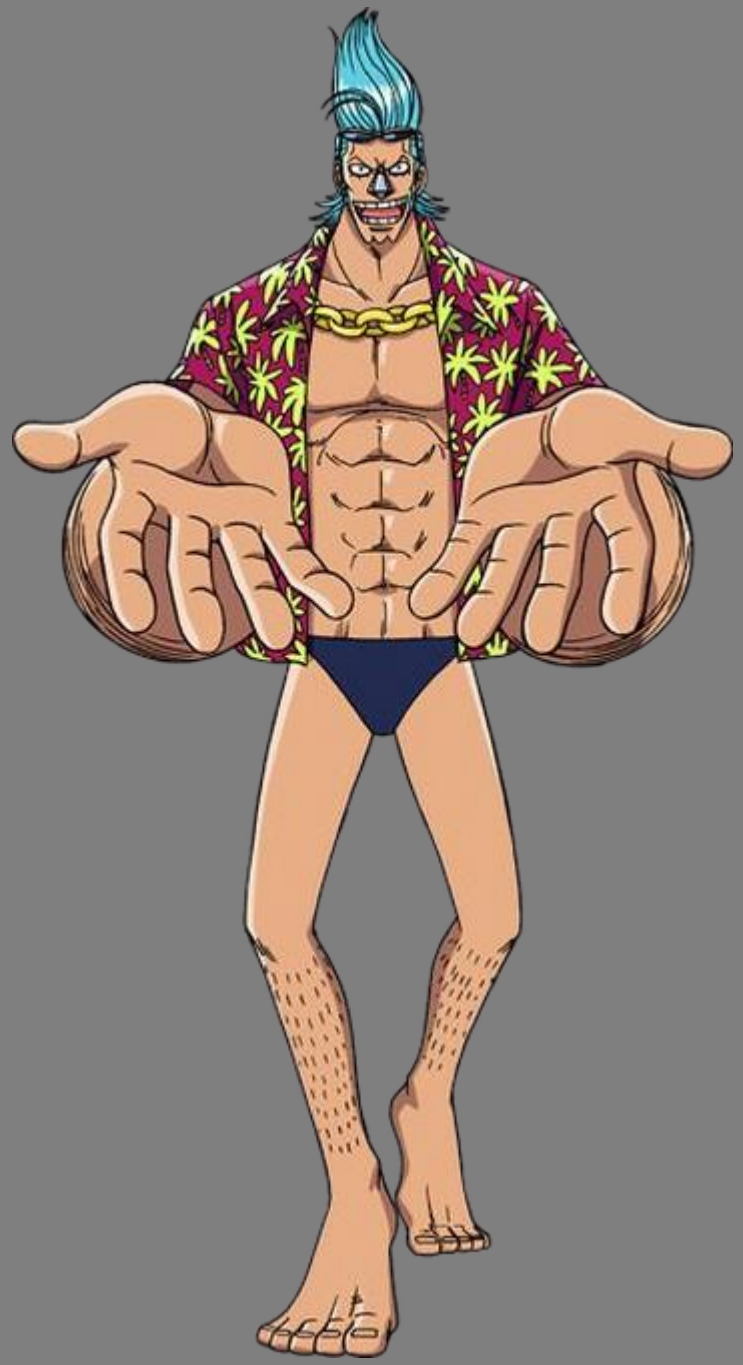
Insper

Arquitetura Cluster Franky

Como compartilhar esse recurso (ambiente)?

SLURM

- É um acrônimo para "Simple Linux Utility for Resource Management"
- É um sistema de gerenciamento de filas e recursos para clusters Linux
- Sua primeira versão foi em 2003
- Amplamente adotado em ambientes acadêmicos e industriais devido a sua eficiência e simplicidade
- Desenvolvimento é contínuo com contribuições da comunidade open-source
- Atualmente é mantido e atualizado pelo SchedMD
- Aproximadamente 60% dos supercomputadores da lista TOP500 usam SLURM



Vamos colocar a mão na massa!