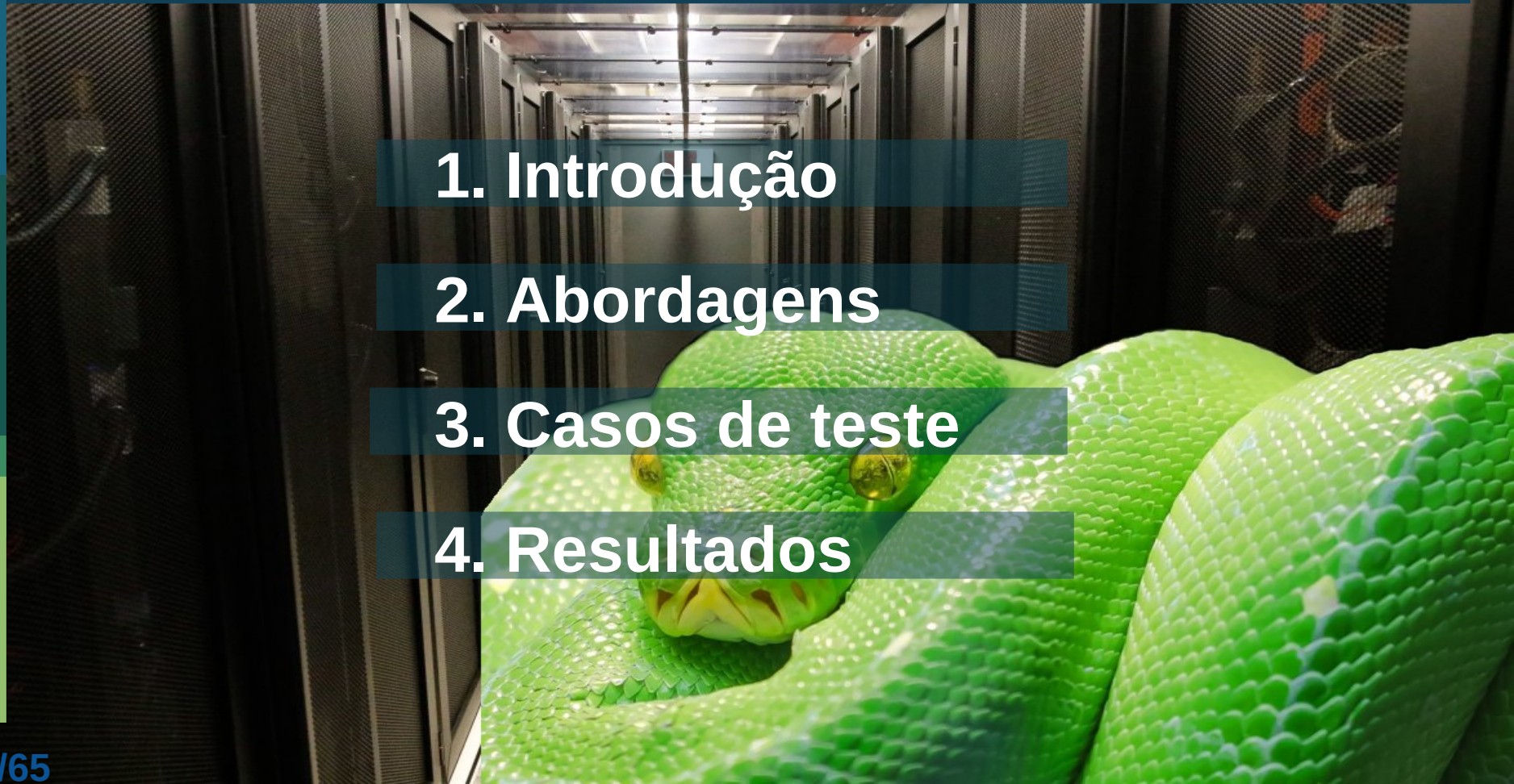
Soluções Comuns Baseadas em MPI para Processamento de Alto Desempenho em Python Avaliadas em Casos de Teste Selecionados

Eduardo F. Miranda

Mestrado CAP (orientador Stephan Stephany)

17 de fevereiro de 2022

Conteúdo



Por que Python?

- **Prototipagem rápida**
- **Interfaceamento fácil**
- **Fácil de ler e manter**
- **Ambiente moderno**
- **Comunidade código aberto**
- **Muitas bibliotecas disponíveis (incluindo PAD)**

Language Ranking: IEEE

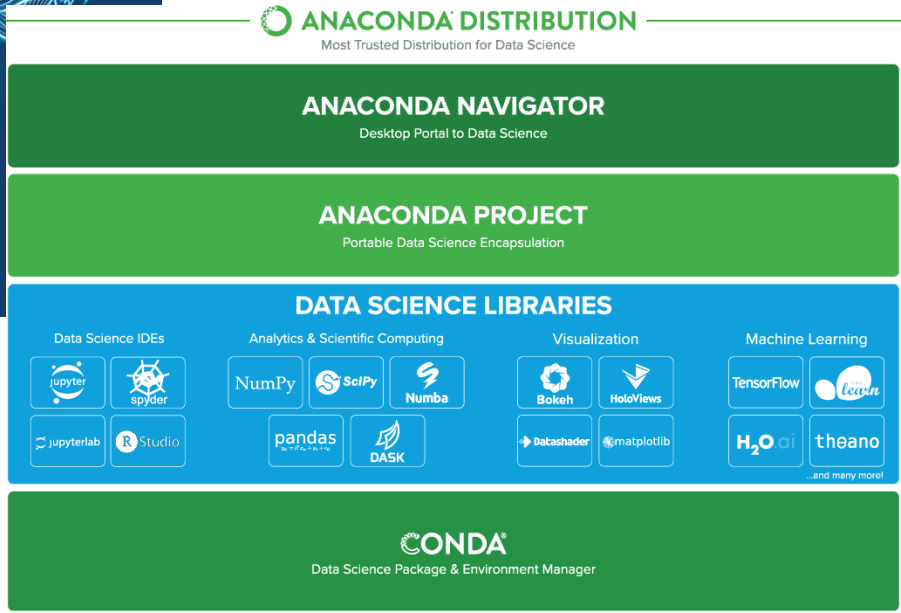
Rank	Language	Score
1	Python	100.0
2	Java	95.4
3	C	94.7
4	C++	92.4
5	JavaScript	88.1

http://spectrum.ieee.org

1

Introdução

Exemplos de pacotes de desenvolvimento Python



- **Ambos disponíveis no supercomputador Santos Dumont (LNCC)**

Objetivos (I)

Python -> programação e prototipagem rápida, mas permite processamento de alto desempenho (PAD)?

Compilação do código Python padrão para linguagem intermediária (bytecode)

Execução interpretada (lenta) por Python

Então Python não é adequado PAD?

Objetivos (II)

Abordagens de PAD/HPC para Python para 3 casos de teste selecionados

Foco em MPI, execução no Santos Dumont (LNCC)

Avaliar o desempenho das abordagens

Referência: versões F90 sequencial e MPI

Roteiro para o uso de recursos PAD em Python

2

Algumas abordagens e recursos Python para PAD/HPC

Algumas abordagens e recursos

IPython: shell + completo para computação interativa

IPython Parallel (IPP): pacote p/ disparar via srun/mpiexec/etc. processos mestre & escravos (engines) p/ execução paralela do programa Python

mpi4py: biblioteca para usar MPI em Python

NumPy: pacote CC, arrays multidimensionais, funções álgebra linear, I/F de conversão p/ F90 (F2PY)

Algumas abordagens e recursos

F2PY (NumPy): código F90 -> bibliotecas Python

PyCuda: biblioteca de acesso à API CUDA Nvidia

Scikit-learn: biblioteca de aprendizado de máquina (permite escolher backend de paralelização: OpenMP, LOKY, IPP, Dask)

Cython: compilador para Python e Cython

Numba: compilador JIT (just-in-time) para Python

Algumas abordagens e recursos

Python: implementação padrão interpretada (lenta!!!)

JupyterLab: aplicativo web interativo

Distribuições Python: Anaconda & Intel

F90 (referência): GNU & Intel

Outras possíveis, não usadas neste trabalho: computação distribuída, computação em nuvem ou grade, ambiente específicos (Dask), etc.

3

Três casos de teste:

Estêncil
FFT
Floresta Aleatória

Caso de teste Estêncil

Problema de difusão de calor 2D

Modelado matemático: equação de Poisson, método das diferenças finitas, estêncil de 5 pontos

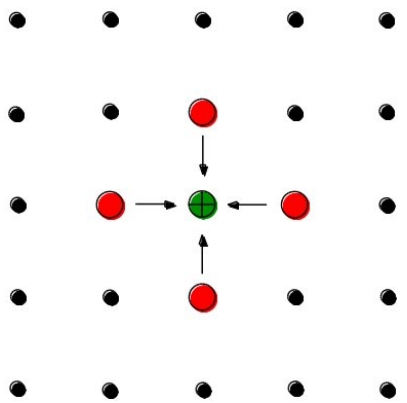
Simulação ao longo de timesteps com atualização sucessiva da grade 2D

Equação de Poisson 2D (difusão de calor)

Campo de temperatura U em malha discreta 2D (x, y)

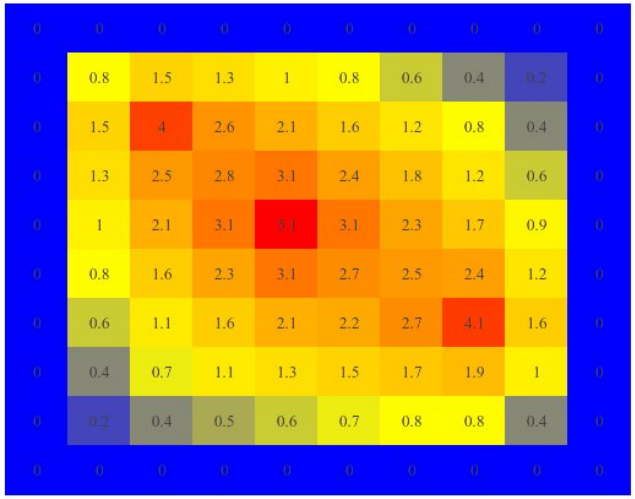
Resolução Δx = Δy = h

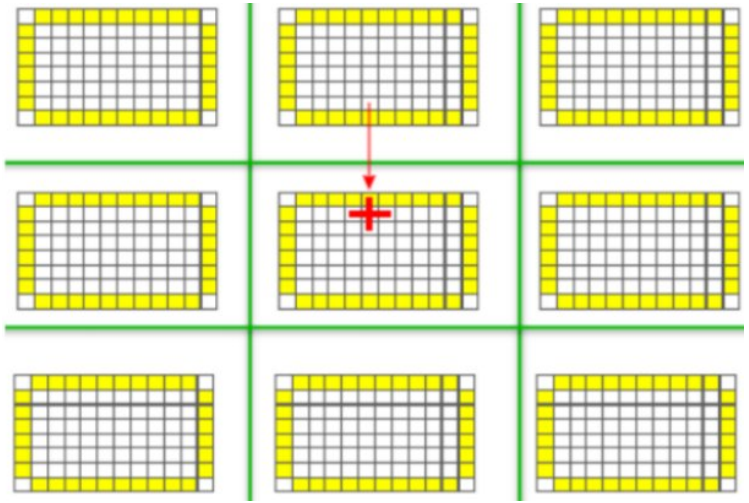
Estêncil de 5 pontos, expressão diferenças finitas:



$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} \approx \frac{U_{i+1,j} + U_{i,j+1} - 4U_{i,j} + U_{i-1,j} + U_{i,j-1}}{h^2}$$

Divisão do domínio 2D





Paralelização: divisão em subdomínios com replicação das bordas (temperaturas) nas fronteiras entre eles

A cada timestep, atualização da grade com estêncil 5 pontos exige comunicação MPI entre bordas subdomínios vizinhos

Parte intensiva do código em questão

F90

```
do j=2,by+1
  do i=2,bx+1
    anew(i,j)=1/2*(aold(i,j)+1/4*(aold(i-1,j)+aold(i+1,j)+aold(i,j-1)+aold(i,j+1)))
  enddo
enddo
```

Cython

```
cpdef kernel(double[:,::1] anew, double[:,::1] aold, Py_ssize_t  by, Py_ssize_t  bx):
  for i in range(1,bx+1):
    for j in range(1,by+1):
      anew[i,j]=1/2*(aold[i,j]+1/4*(aold[i-1,j]+aold[i+1,j]+aold[i,j-1]+aold[i,j+1]))
```

Numba

```
@njit
def kernel(anew, aold):
  anew[1:-1,1:-1]=(
  1/2*(aold[1:-1,1:-1]+1/4*(aold[2:,1:-1]+aold[:-2,1:-1]+aold[1:-1,2:]+aold[1:-1,:-2])))
```

Caso de teste FFT 3D

Transformação domínio espacial/temporal p/ domínio frequência (transformada discreta de Fourier):

FT -> DFT (FT discreta) -> algoritmo Fast DFT (FFT)

FFTW (Fast FFT in the West)

FFTW p/ array multidimensional 3D (dado sintético)

Elementos array gerados por um esquema particular

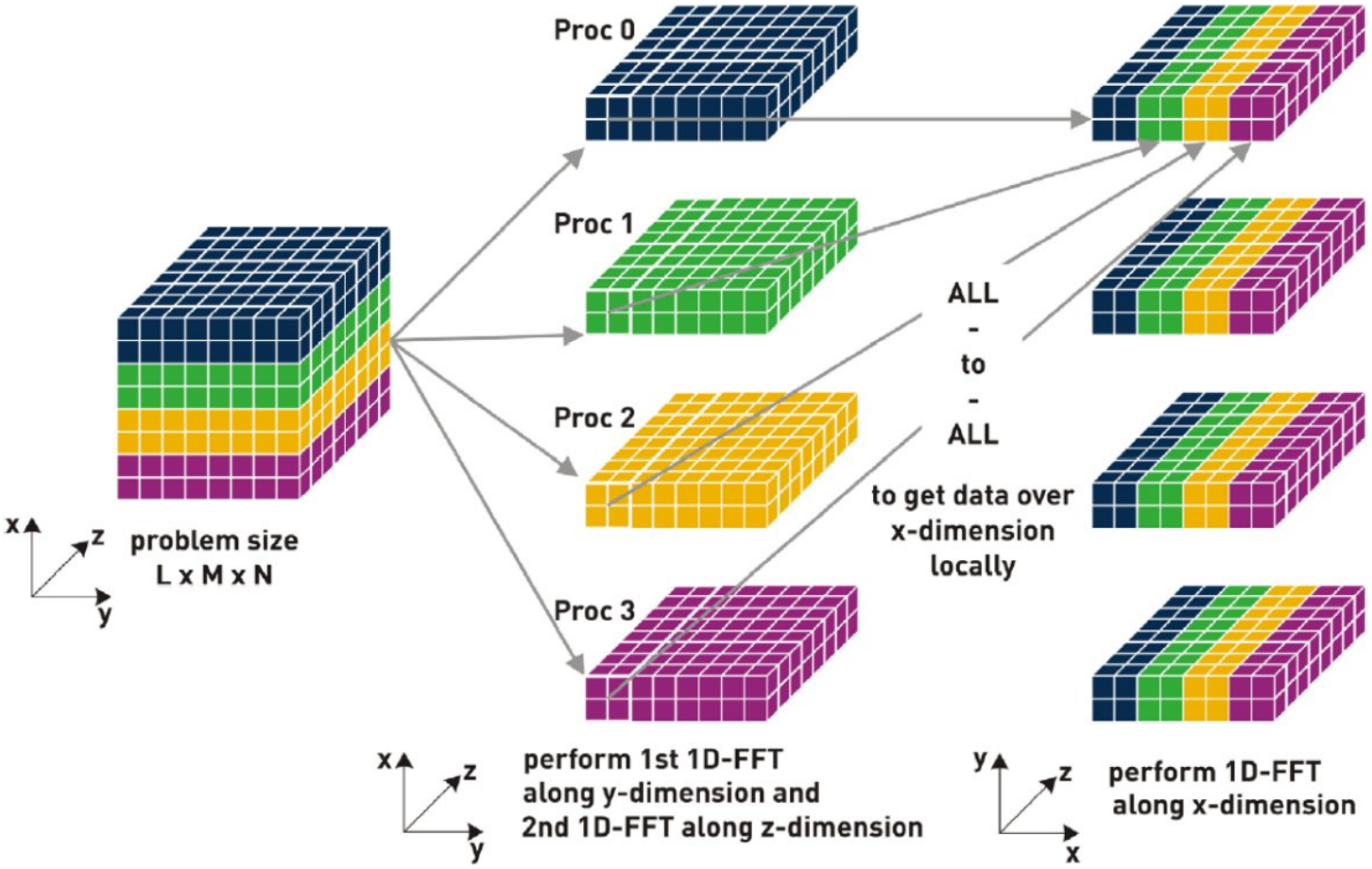
Caso de teste FFT 3D

Array multimensional 576 x 576 x 576 (múltiplo do número de processos MPI)

FFTs multidimensionais (composição de FFTs 1D)

$$H(k_1,k_2,k_3) = \sum_{n_3=0}^{N_3-1} \sum_{n_2=0}^{N_2-1} \sum_{n_1=0}^{N_1-1} h(n_1,n_2,n_3) e^{\frac{-2\pi i k_3 n_3}{N_3}} \frac{-2\pi i k_2 n_2}{N_2} \frac{-2\pi i k_1 n_1}{N_1}$$

Paralelização: FFT 3D (transposição)



Caso de teste Floresta Aleatória

N árvores de decisão treinadas independentemente

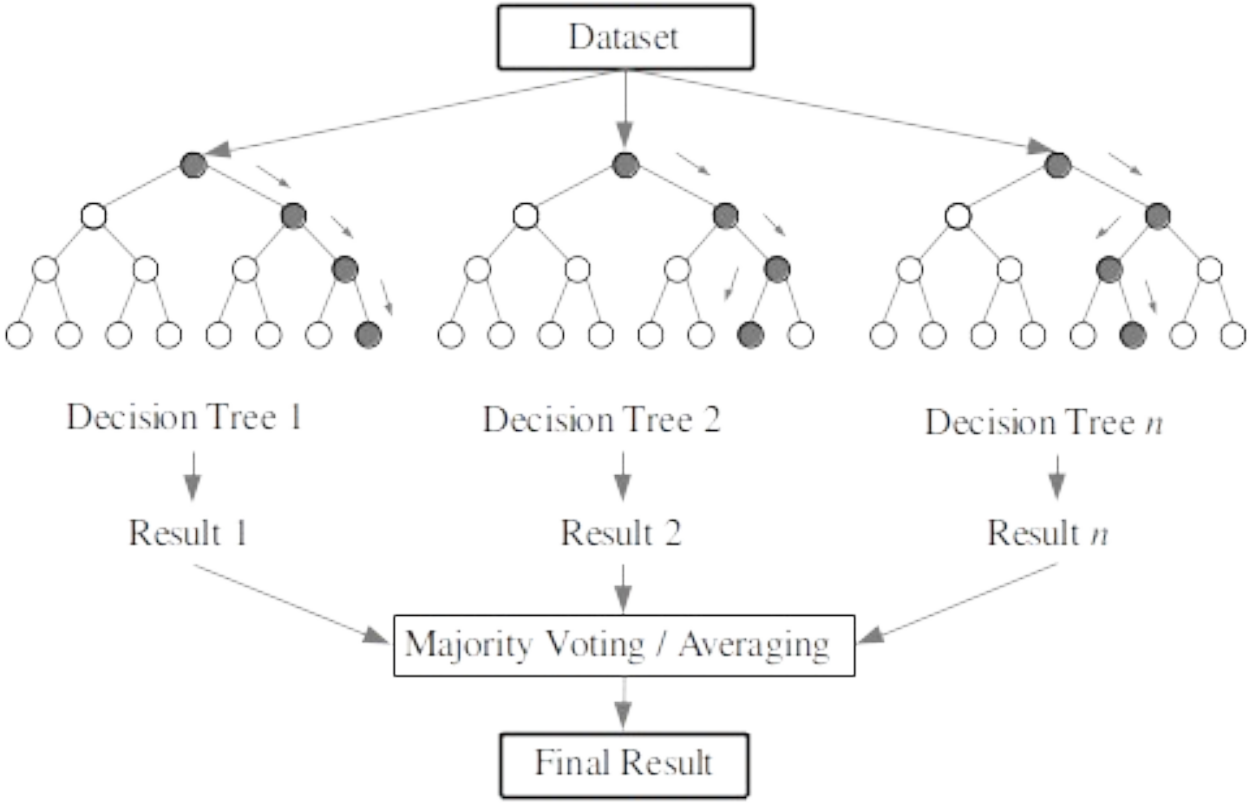
Teste/validação → N resultados (média/votação)

Classificação de órbitas de asteróides

Dataset com 100.000 asteróides JPL/NASA

Paralelização

Paralelização trivial:
N árvores mapeadas para p processos MPI/IPP/LOKY (N/p árvores cada processo)



4

Resultados: análise de desempenho sequencial e paralelo

Ambiente (Santos Dumont LNCC)

Nó B710: 2x Xeon E5-2695v2 12-core

Nó B715: 2x Xeon E5-2695v2 12-core + 2x Tesla K40

Nó Sequana X: 2x Xeon 6152 22-core + 4x Volta V100

GNU Fortran 7.4, GNU Fortran 8.3, OpenMPI 4.0.1, Intel Fortran 19.0.3, Intel MPI, Python 3.6.12, Cython 0.29.20, NumPy 1.18.1, Numba 0.41.0, e CUDA 10.1 e outros

4.1

Resultados:

Difusão de calor 2D: Estêncil de 5 pontos

Tempos de processamento [s] - nós B710

	Número de processos MPI							
	Seq.	**1**	**4**	**9**	**16**	**36**	**49**	**64**	**81**
F90	**19.3**	**21.9**	**7.3**	**6.2**	4.7	**2.1**	1.9	**1.2**	1.7
F2PY	**18.9**	23.6	7.5	6.2	**4.6**	2.1	**1.6**	1.3	**1.0**
Cython	24.0	24.0	7.5	6.3	4.7	2.2	1.7	1.3	2.1
Numba (CPU)	30.5	30.5	8.2	6.3	5.9	3.2	2.7	1.8	2.1
Python	212.4	227.2	64.7	44.8	33.5	15.2	10.4	7.8	6.7

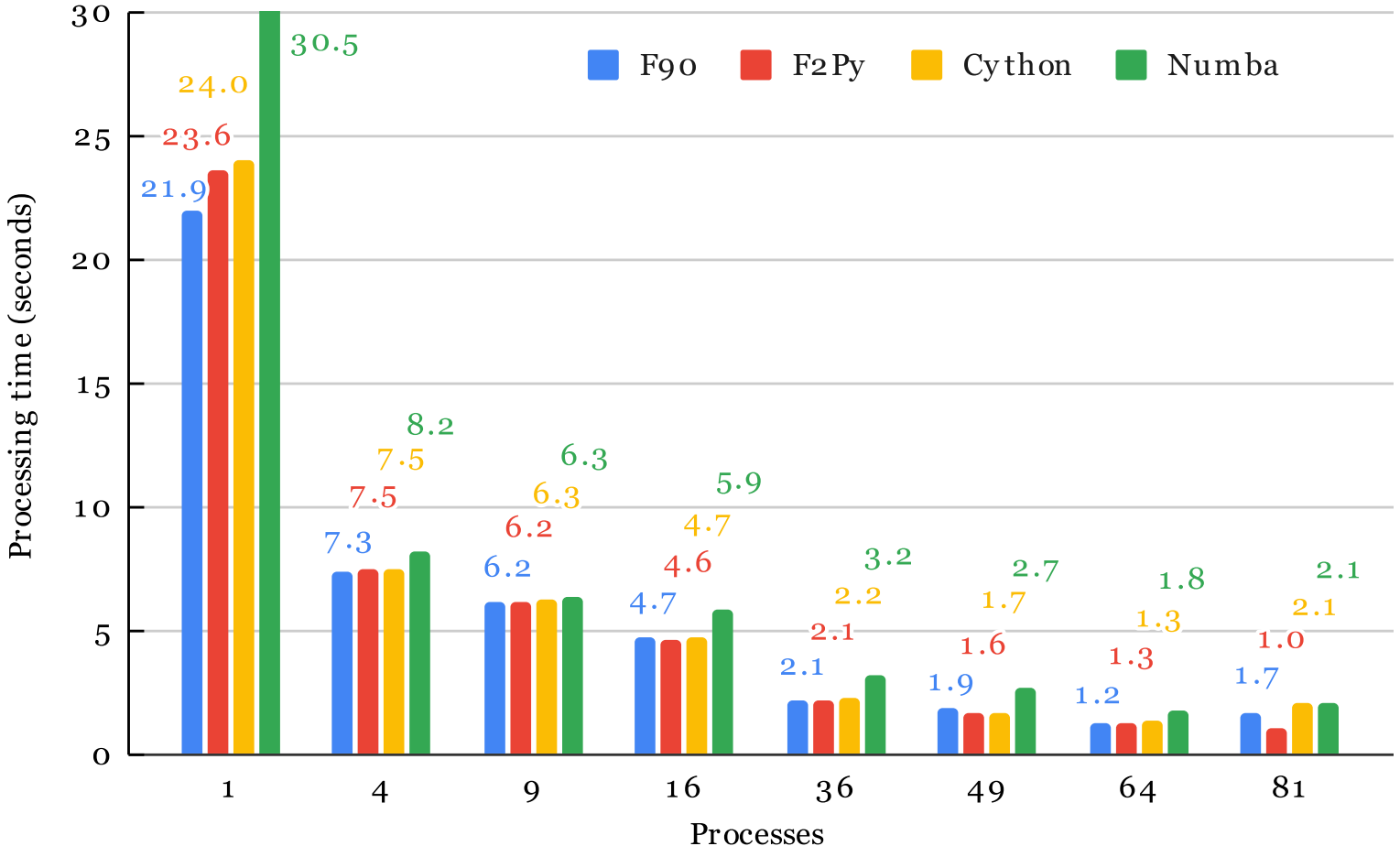
Speedup - nós B710

	Número de processos MPI							
	Seq.	1	4	9	16	36	49	64	81
F90	1.0	0.9	2.6	3.1	4.1	9.0	10.2	15.7	11.4
F2PY	1.0	0.8	2.6	3.1	4.2	9.0	11.8	15.1	19.0
Cython	0.8	0.8	2.6	3.1	4.1	8.6	11.6	14.7	9.4
Numba (CPU)	0.6	0.6	2.4	3.0	3.3	6.0	7.2	10.8	9.3
Python	0.1	0.1	0.3	0.4	0.6	1.3	1.8	2.5	2.9

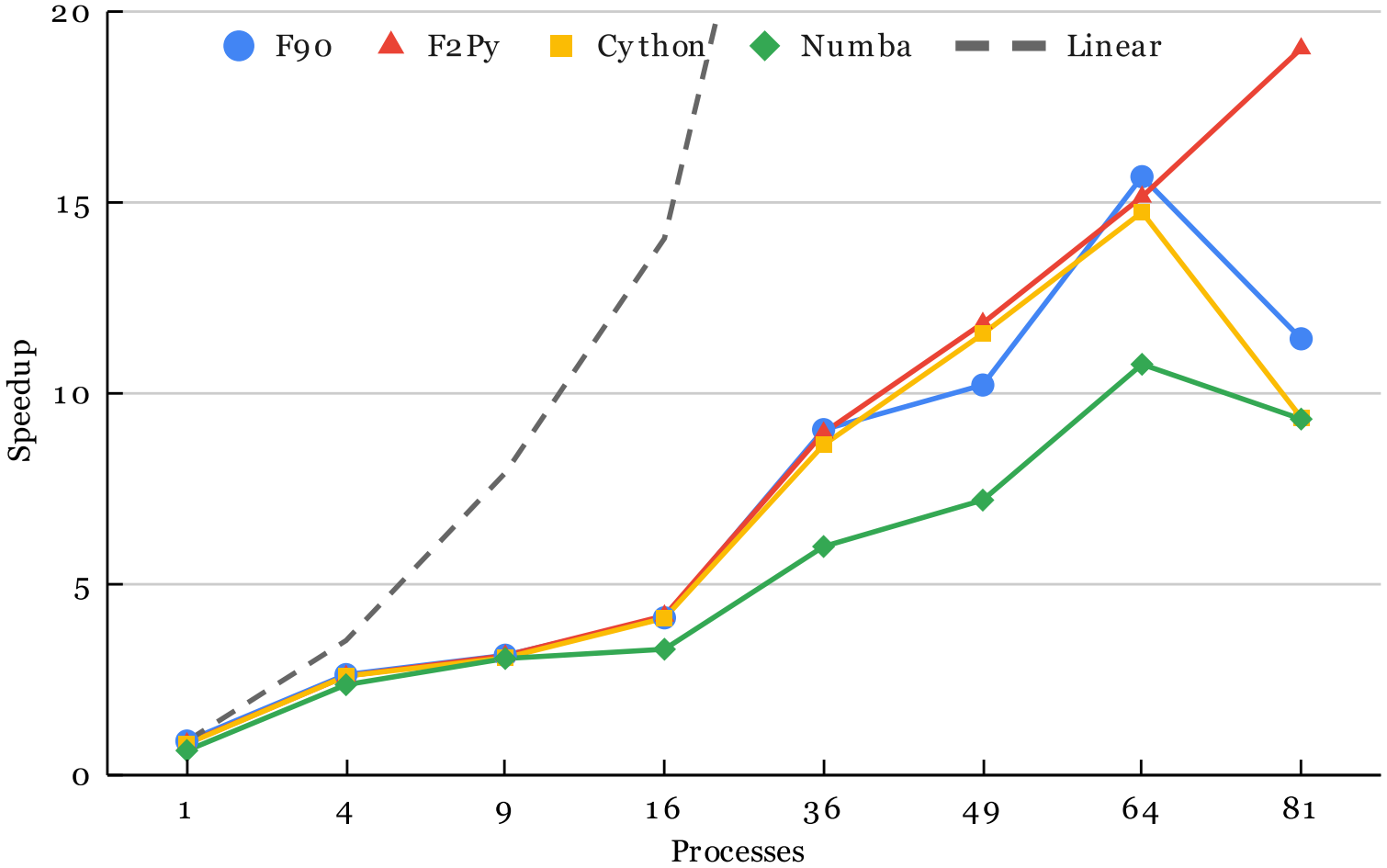
Eficiência paralela - nós B710

	Número de processos MPI							
	Seq.	1	4	9	16	36	49	64	81
F90	1.00	0.88	0.66	0.35	0.26	0.25	0.21	0.24	0.14
F2PY	1.02	0.82	0.65	0.35	0.26	0.25	0.24	0.24	0.23
Cython	0.80	0.80	0.65	0.34	0.26	0.24	0.24	0.23	0.12
Numba (CPU)	0.63	0.63	0.59	0.34	0.21	0.17	0.15	0.17	0.12
Python	0.09	0.08	0.07	0.05	0.04	0.04	0.04	0.04	0.04

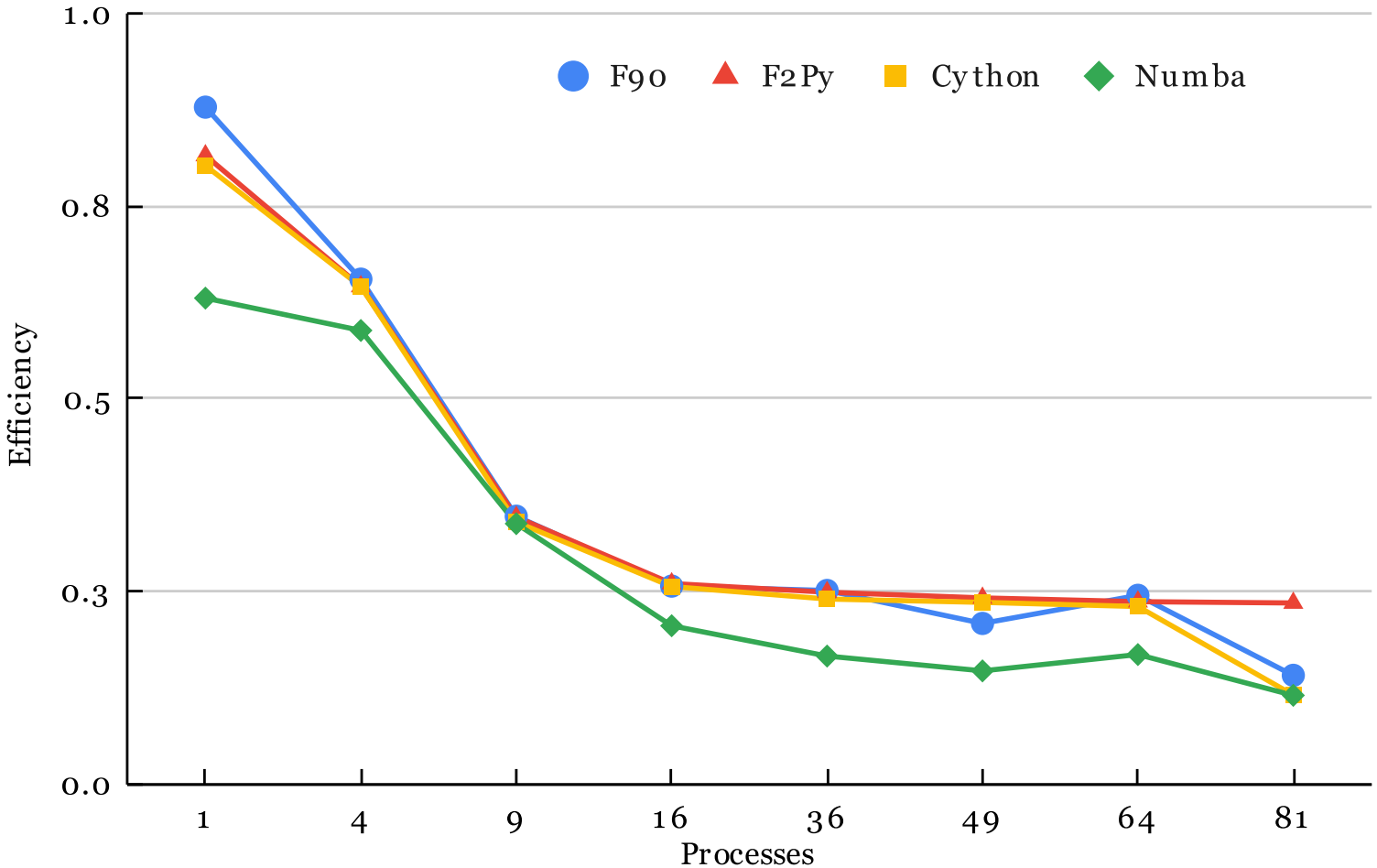
Tempos de processamento [s] - nós B710



Speedup - nós B710



Eficiência paralela - nós B710



Comparação tempos [s] Numba-GPU x F90

		Número de processos MPI				
Implementação	**Seq.**	**1**	**4**	**9**	**16**	**36**
F90/B715	**19.3**	21.9	7.3	6.2	4.7	**2.1**
F90/Seq-X	15.8	15.6	4.1	2.1	1.5	**1.2**
Numba-GPU/B715	**9.4**	104.8	27.2	11.7	7.9	4.3
Numba-GPU/Seq-X	**2.2**	49.5	15.2	6.9	6.7	9.1

4.2

Resultados:

FFT 3D

Tempos de processamento [s] - nós B710

	Número de processos MPI						
	Seq.	1	4	16	24	48	72	96
F90	19.3	23.4	6.4	2.6	2.4	2.2	2.2	2.3
F2PY	23.8	27.6	7.4	3.5	4.1	4.3	3.6	5.2
Python	161.7	174.8	44.9	11.1	7.5	12.3	10.5	8.7
Cython	109.0	124.2	29.1	7.9	7.7	10.2	10.6	8.6
Numba	48.6	50.0	13.3	5.2	4.3	13.0	11.5	9.0

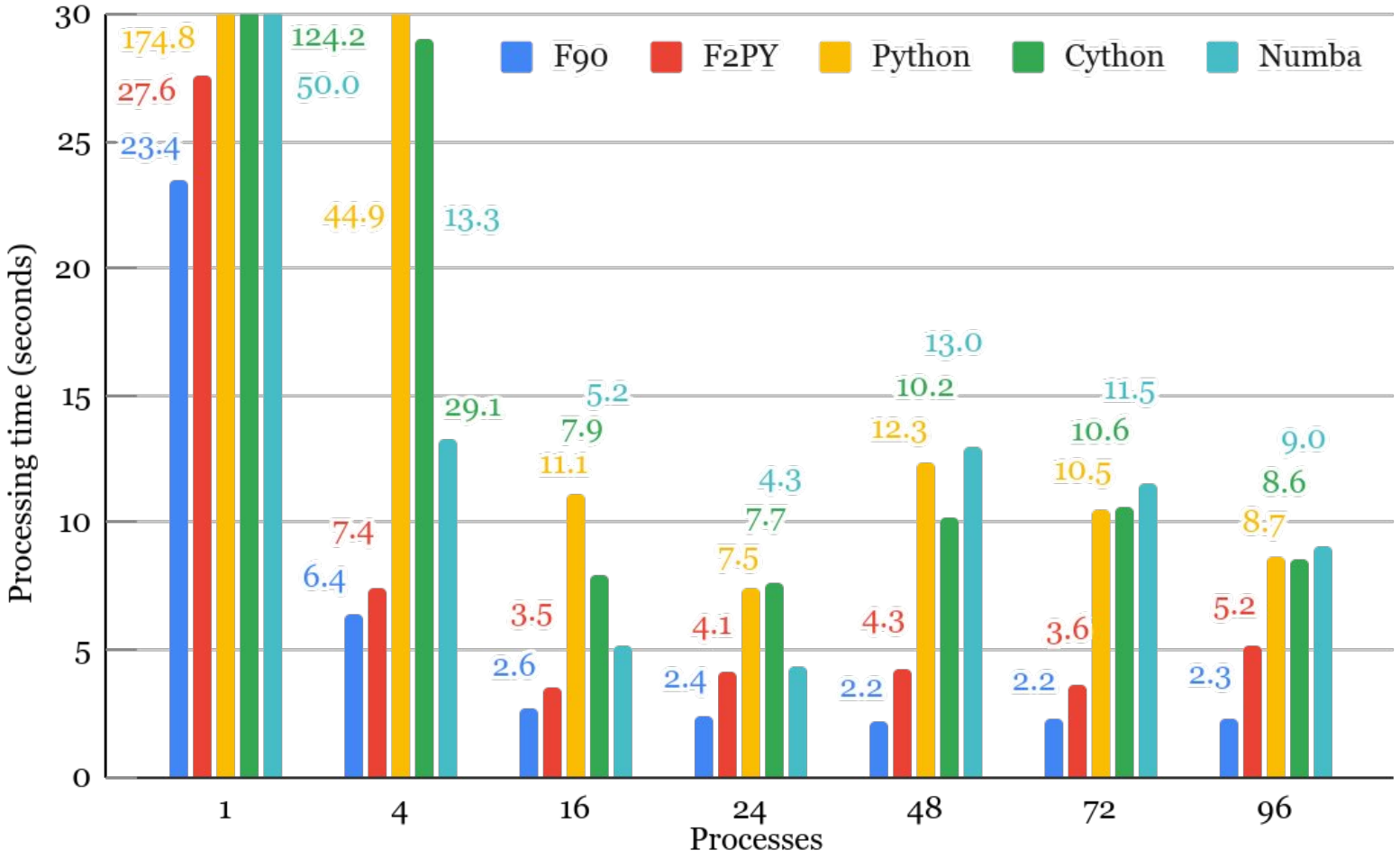
Speedup - nós B710

	Seq.	Número de processos MPI 1	4	16	24	48	72	96
F90	1.0	0.8	3.0	7.3	8.2	8.7	8.6	8.3
F2PY	0.8	0.7	2.6	5.5	4.7	4.5	5.4	3.7
Python	0.1	0.1	0.4	1.7	2.6	1.6	1.8	2.2
Cython	0.2	0.2	0.7	2.4	2.5	1.9	1.8	2.3
Numba	0.4	0.4	1.5	3.7	4.5	1.5	1.7	2.1

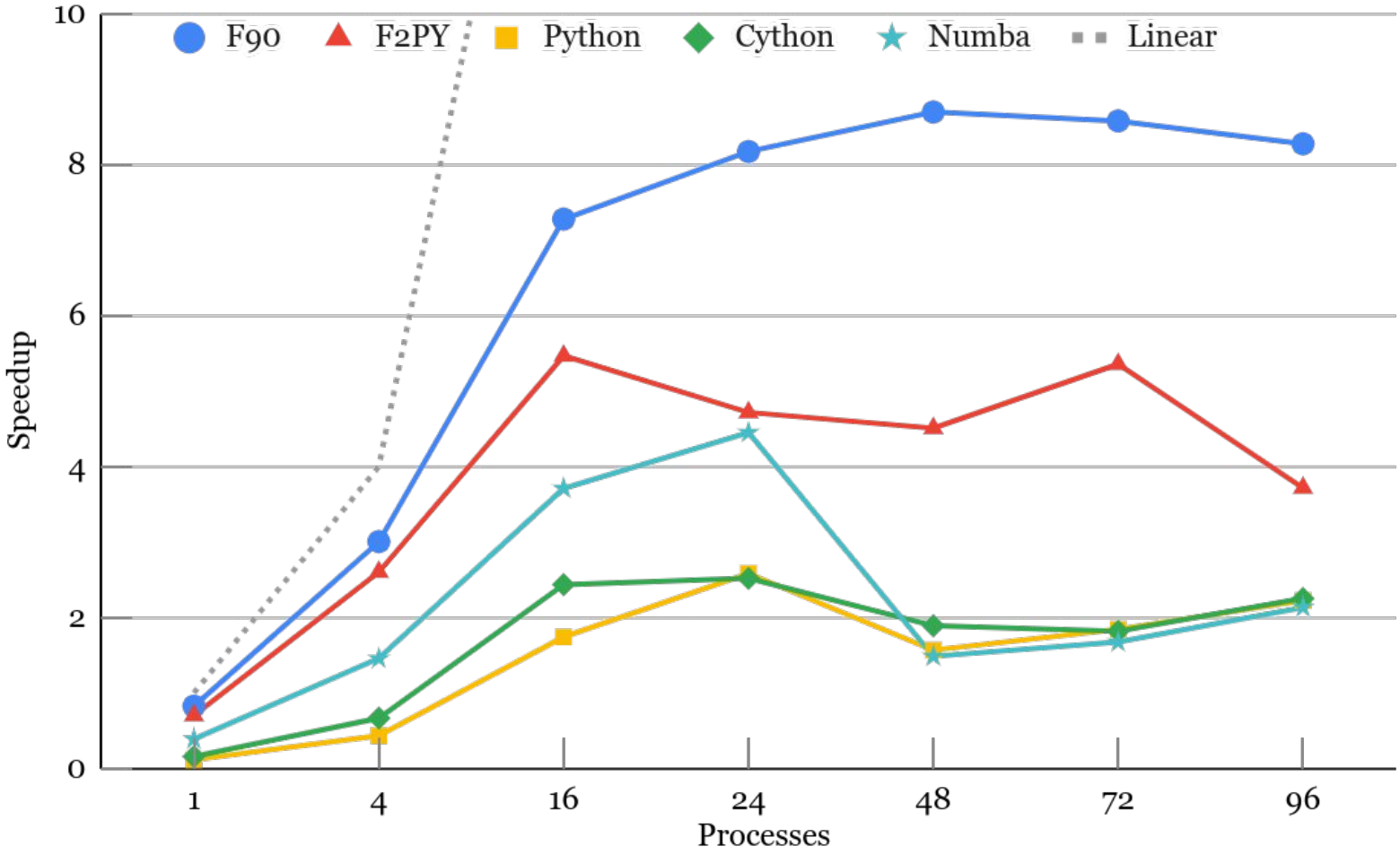
Eficiência paralela - nós B710

	Seq.	Número de processos MPI						
		1	4	16	24	48	72	96
F90	1.00	0.82	0.75	0.46	0.34	0.18	0.12	0.09
F2PY	0.81	0.70	0.65	0.34	0.20	0.09	0.07	0.04
Python	0.12	0.11	0.11	0.11	0.11	0.03	0.03	0.02
Cython	0.18	0.16	0.17	0.15	0.10	0.04	0.03	0.02
Numba	0.40	0.39	0.36	0.23	0.19	0.03	0.02	0.02

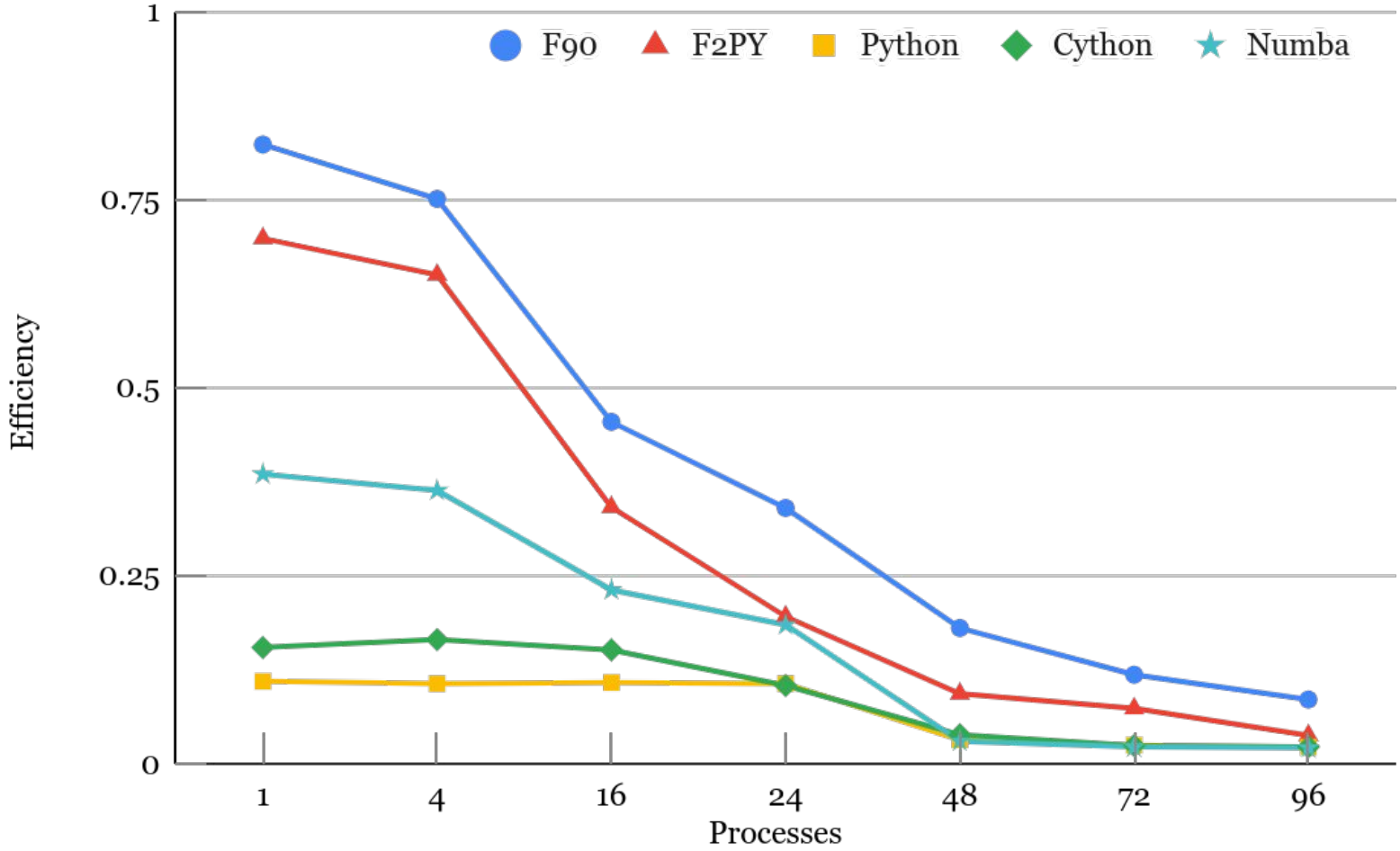
Tempos de processamento [s] - nós B710



Speedup - nós B710



Eficiência paralela - nós B710



Comparação tempos [s] CuPy x F90

			Número de processos MPI			
Implementação	**GPU**	**Seq.**	**1**	**4**	**16**	**24**
F90/B715		**23.77**	23.41	6.48	2.73	2.48
F90/Seq-X		12.74	14.85	4.06	1.37	**0.96**
CuPy(GPU)/B715	38.16					
CuPy(GPU)/Seq-X	19.62					

Tempos [s] - otimização para NUMA

Implementação	16 processos MPI			
	B710		Seq-X	
	None	Bind	None	Bind
F90	**2.65**	**2.47**	**1.65**	**1.33**
F2PY	3.53	3.20	2.09	1.54
Python	11.09	10.10	6.60	5.66
Cython	7.92	6.52	5.31	4.51
Numba	5.20	4.70	3.40	2.30

None (sem opção): **P1 (12 cores) e P2 (4 cores)**

Opção Binding: **P1(8 cores) e P2 (8 cores)**

4.3

Resultados:

Floresta Aleatória para classificação de órbitas de asteróides

Tempos de processamento [s] - nós B710

		Número de processos						
	Seq.	1	4	16	24	48	72	96
Python	25.7	28.6	14.9	11.1	11.0	12.0	13.9	14.5
Numba	25.0	33.6	16.6	13.0	13.8	15.7	15.5	16.7
Cython	29.5	65.6	25.7	15.3	15.4	17.7	15.6	16.9
F90	137.7	138.9	45.3	20.8	19.1	14.6	14.6	16.0
F2PY	135.1	141.7	48.3	23.0	19.2	17.1	17.5	18.0

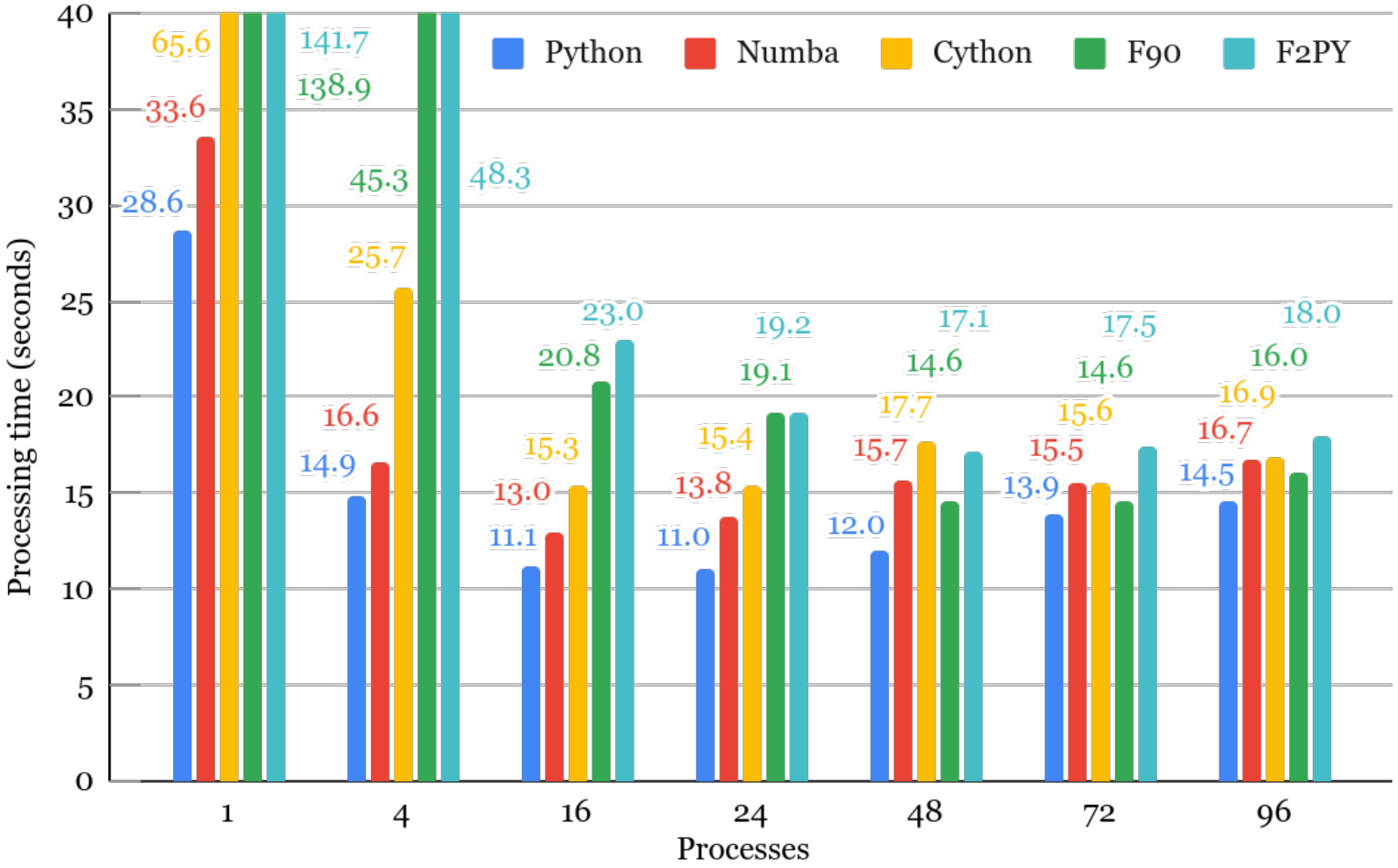
Speedup - nós B710

	Seq.	Número de processos						
		1	4	16	24	48	72	96
Python	1.0	0.9	1.7	2.3	2.3	2.1	1.8	1.8
Numba	1.0	0.8	1.5	2.0	1.9	1.6	1.7	1.5
Cython	0.9	0.4	1.0	1.7	1.7	1.4	1.6	1.5
F90	0.2	0.2	0.6	1.2	1.3	1.8	1.8	1.6
F2PY	0.2	0.2	0.5	1.1	1.3	1.5	1.5	1.4

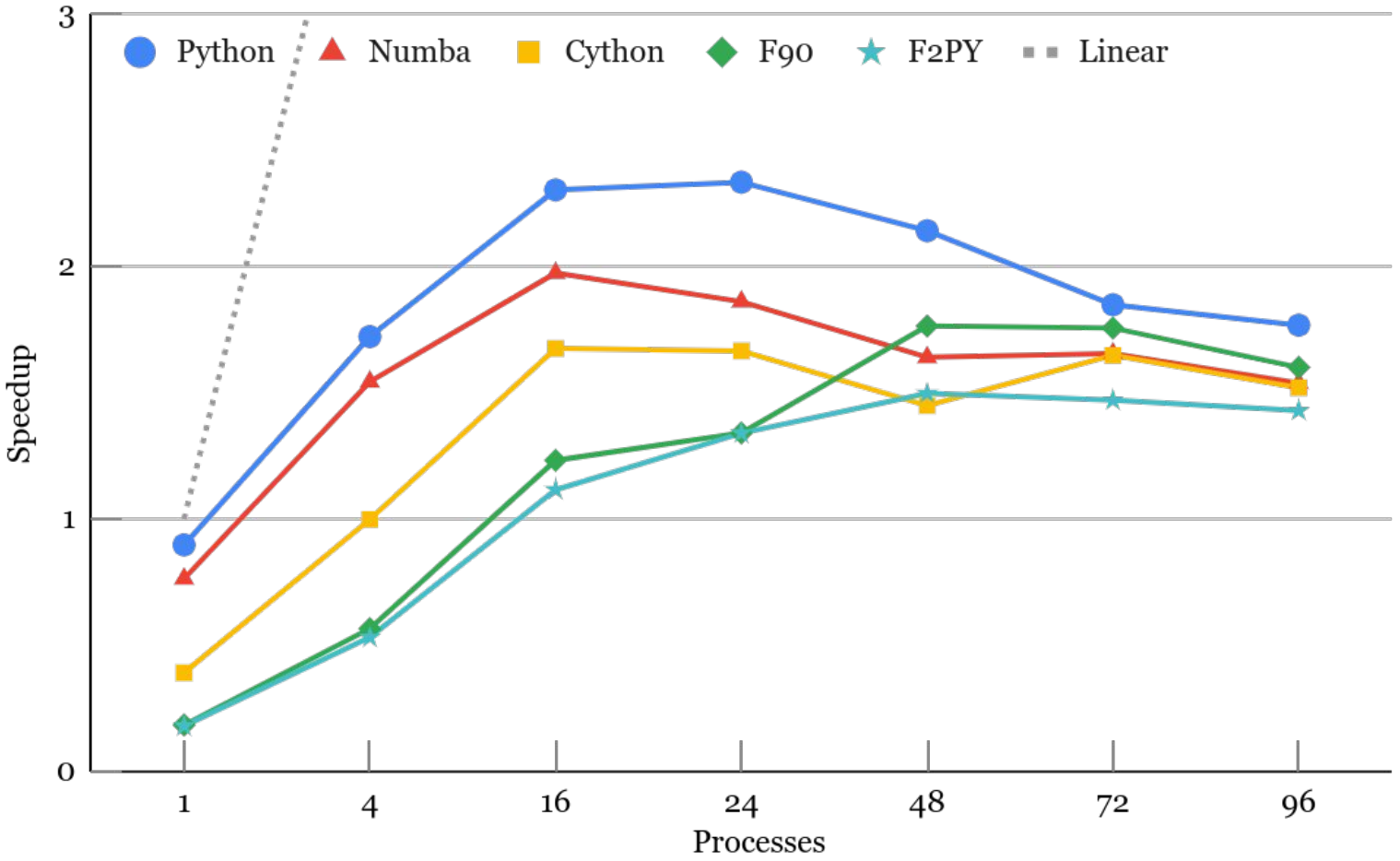
Eficiência paralela - nós B710

	Seq.	Número de processos 1	4	16	24	48	72	96
Python	1.00	0.90	0.43	0.14	0.10	0.04	0.03	0.02
Numba	1.03	0.76	0.39	0.12	0.08	0.03	0.02	0.02
Cython	0.87	0.39	0.25	0.10	0.07	0.03	0.02	0.02
F90	0.19	0.18	0.14	0.08	0.06	0.04	0.02	0.02
F2PY	0.19	0.18	0.13	0.07	0.06	0.03	0.02	0.01

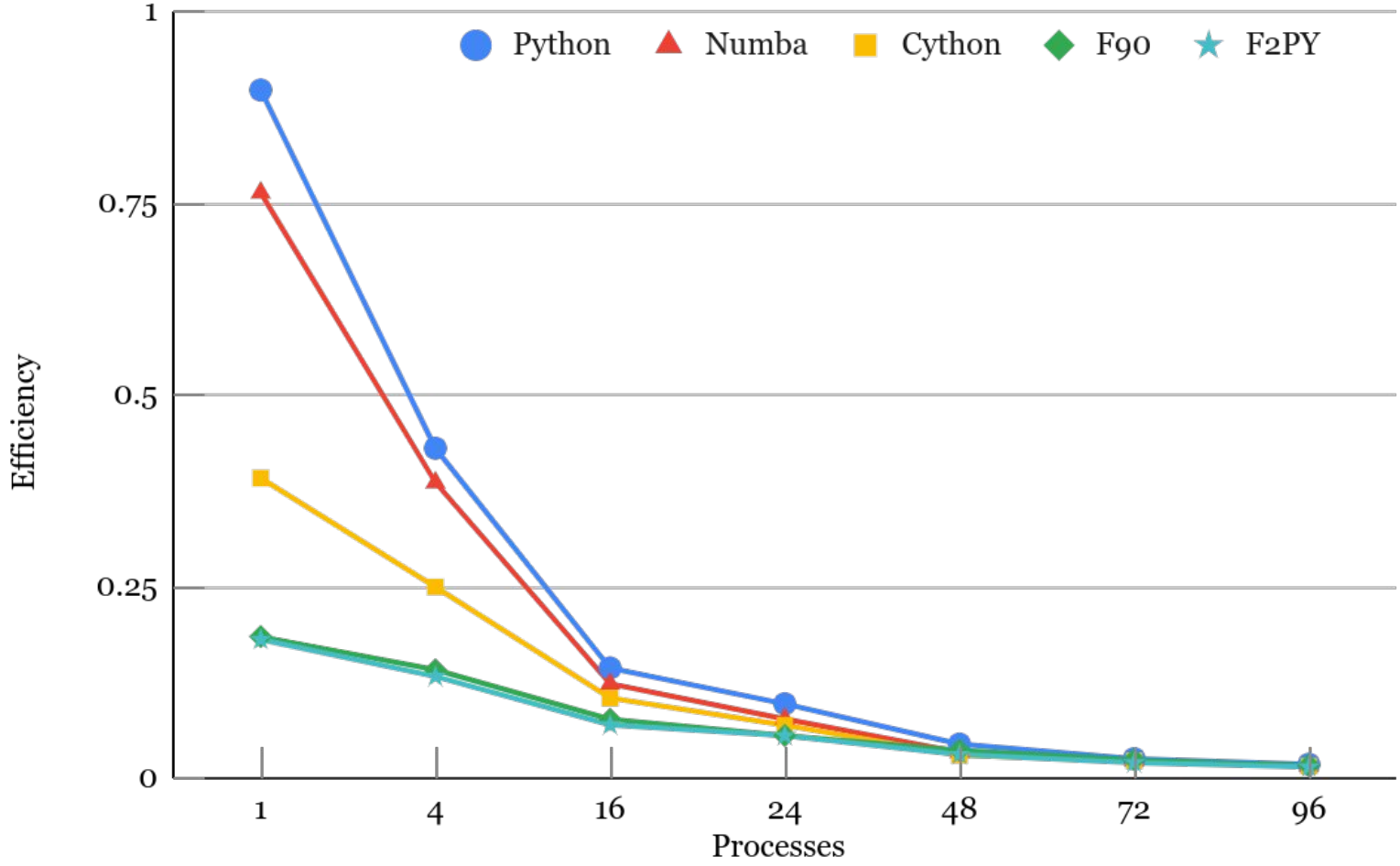
Tempos de processamento [s] - nós B710



Speedup - nós B710



Eficiência paralela - nós B710



4.4

Floresta aleatória (Implementação alternativa)

Floresta aleatória (implementação alternativa)

Classificação de órbitas de asteróides (Python), biblioteca Scikit-learn, nós Sequana-X (2x 24-core)

IPP (IPython parallel): até 4 nós, 192 processos

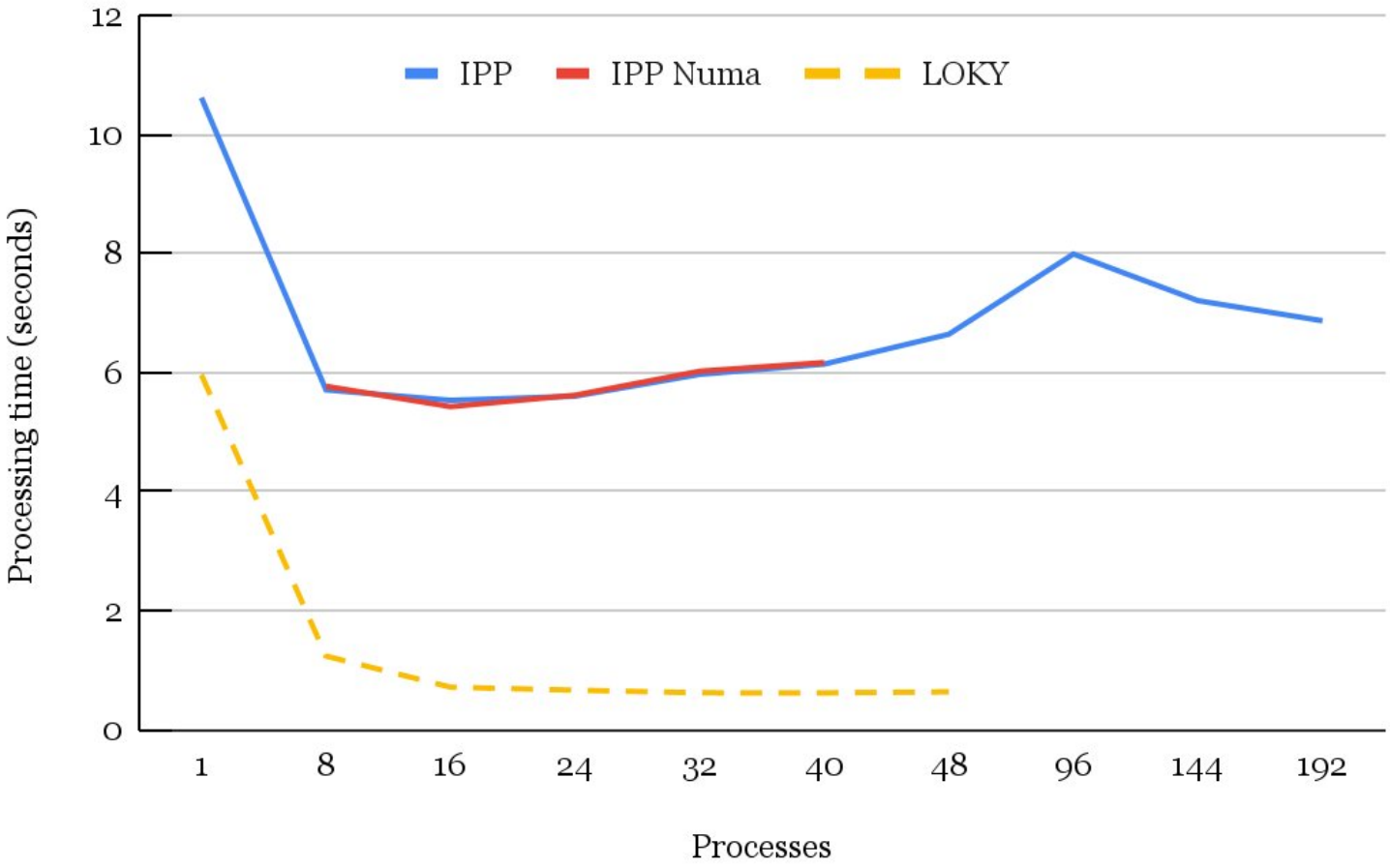
LOKY: único nó, até 48 processos

Implementação extra feita após entrega da dissertação

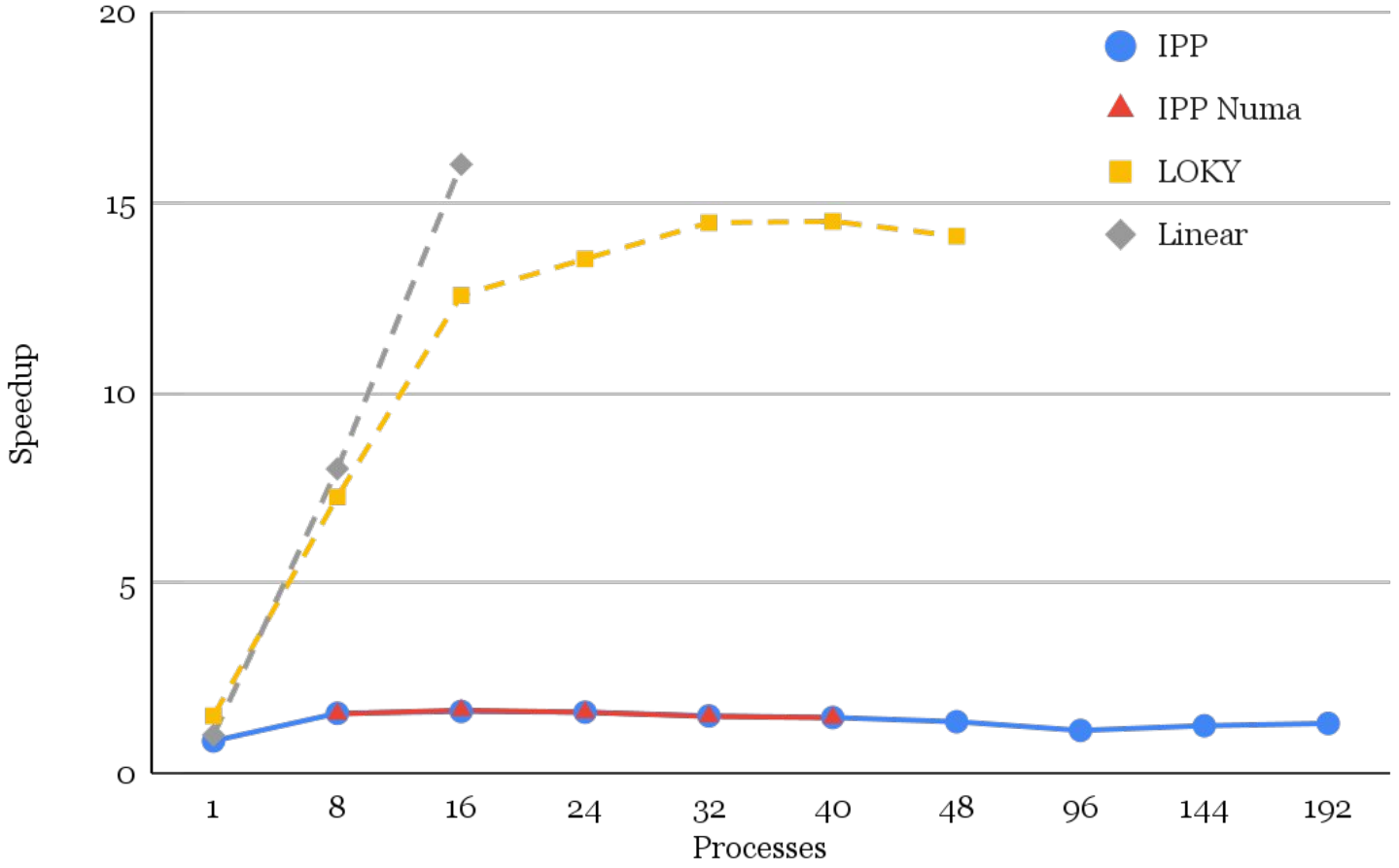
Floresta aleatória (implementação alternativa)

Implementação	Seq.	Número de processos: 1	8	16	24	32	40	48	96	144	192
Tempos de processamento [s]											
IPP	**8.99**	10.62	5.71	5.53	5.60	5.97	6.14	6.64	**7.99**	**7.21**	**6.87**
IPP Numa			5.77	5.43	5.62	6.02	6.17				
LOKY		**5.96**	**1.24**	**0.72**	**0.67**	**0.62**	**0.62**	**0.64**			
Speedup											
IPP	**1.00**	0.85	1.57	1.62	1.60	1.51	1.46	1.35	**1.12**	**1.25**	**1.31**
IPP Numa			1.56	1.66	1.60	1.49	1.46				
LOKY		**1.51**	**7.26**	**12.56**	**13.51**	**14.47**	**14.50**	**14.12**			
Eficiência paralela											
IPP	**1.00**	0.85	0.20	0.10	0.07	0.05	0.04	0.03	**0.01**	**0.01**	**0.01**
IPP Numa			0.19	0.10	0.07	0.05	0.04				
LOKY		**1.51**	**0.91**	**0.79**	**0.56**	**0.45**	**0.36**	**0.29**			

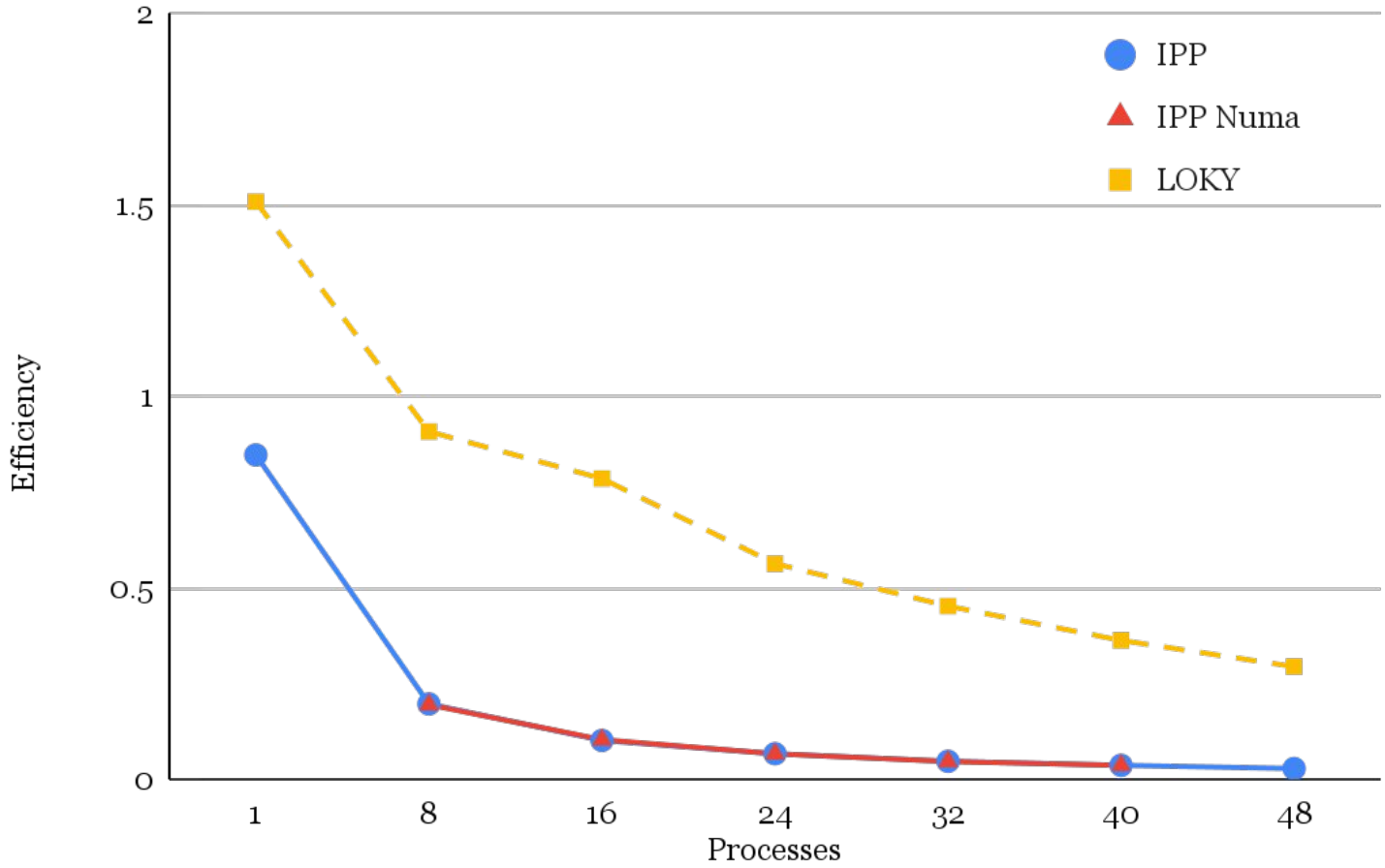
Tempo processamento [s] x no. processos - nós Seq-X



Speedup x no. processos - nós Seq-X



Eficiência paralela x no. processos - nós Seq-X



4.5

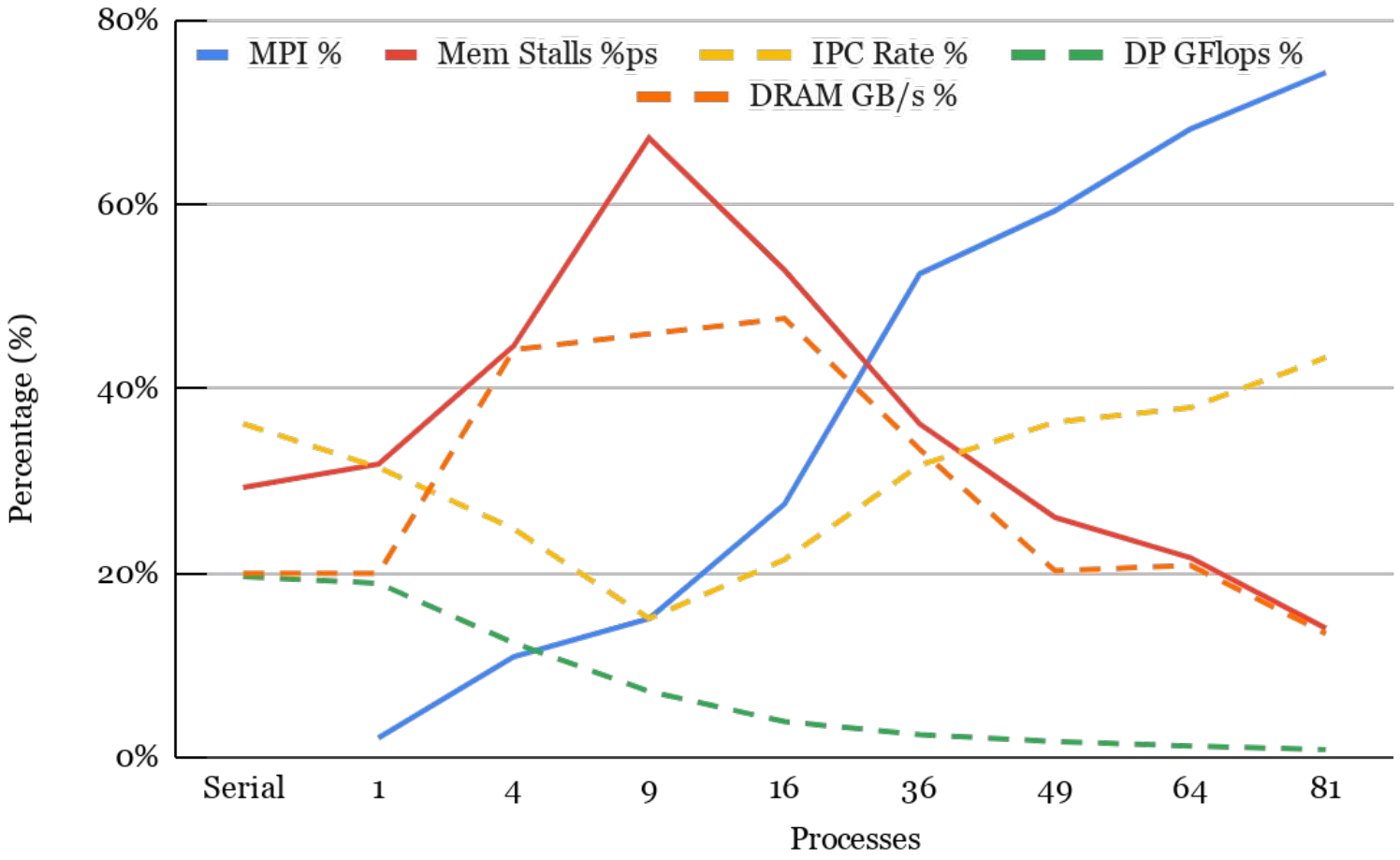
PROFILING:

Intel APS (Application Performance Snapshot)
e outras

Perfil (Intel APS) - Estêncil F90 - nós B710

		Número de processos MPI							
Parâmetro	**Seq.**	**1**	**4**	**9**	**16**	**36**	**49**	**64**	**81**
Elapsed Time [s]	**22.6**	23.3	9.1	7.9	7.5	5.0	4.8	5.4	5.7
ETime - MPI [s]	**-**	**22.8**	**8.1**	**6.7**	**5.5**	**2.4**	**2.0**	**1.8**	**1.5**
Speedup	**-**	**1.0**	**2.5**	**2.9**	**3.0**	**4.5**	**4.7**	**4.2**	**4.0**
Efficiency	**-**	**1.0**	**0.6**	**0.3**	**0.2**	**0.1**	**0.1**	**0.1**	**0.0**
MPI Time [s]	-	0.5	1.0	1.2	2.0	2.6	2.8	3.6	4.2
MPI Time [%]	-	2.1	**10.9**	**15.1**	**27.5**	**52.5**	**59.4**	**68.2**	**74.4**
MPI_Init [s]	-	1.0	0.8	1.0	1.1	1.5	1.4	1.9	**2.0**
MPI_Wait [s]	-	-	0.2	0.1	0.9	0.9	1.0	0.8	1.2
MPI_Bcast [s]	-	-	0.0	0.0	0.0	0.2	0.3	0.8	0.8
MPI Imbalance [s]	-	-	0.1	0.0	0.7	0.9	1.0	1.2	1.6
DP [GFlops]	3.8	3.6	9.6	12.4	12.0	17.2	16.4	15.4	13.6
IPC Rate	1.5	1.3	**1.0**	**0.6**	**0.9**	1.3	1.5	1.5	1.7
Bound	**mem**	**mem**	**mem**	**mem**	**mem**	**MPI**	**MPI**	**MPI**	**MPI**
Cache Stalls [%c]	13.0	14.4	**23.8**	**32.2**	**26.9**	16.2	16.3	12.6	10.2
DRAM Stalls [%c]	13.3	14.3	**29.2**	**33.2**	**27.7**	**24.4**	14.0	10.6	7.1
DRAM [GB/s]	-	11.9	26.4	-	28.5	20.0	12.1	12.5	8.0
Mem Stalls [%ps]	**29.3**	**31.9**	**44.7**	**67.3**	**52.9**	**36.2**	**26.1**	**21.7**	14.0
Vectorization [%]	100.0	100.0	100.0	100.0	100.0	99.9	98.5	99.8	98.0

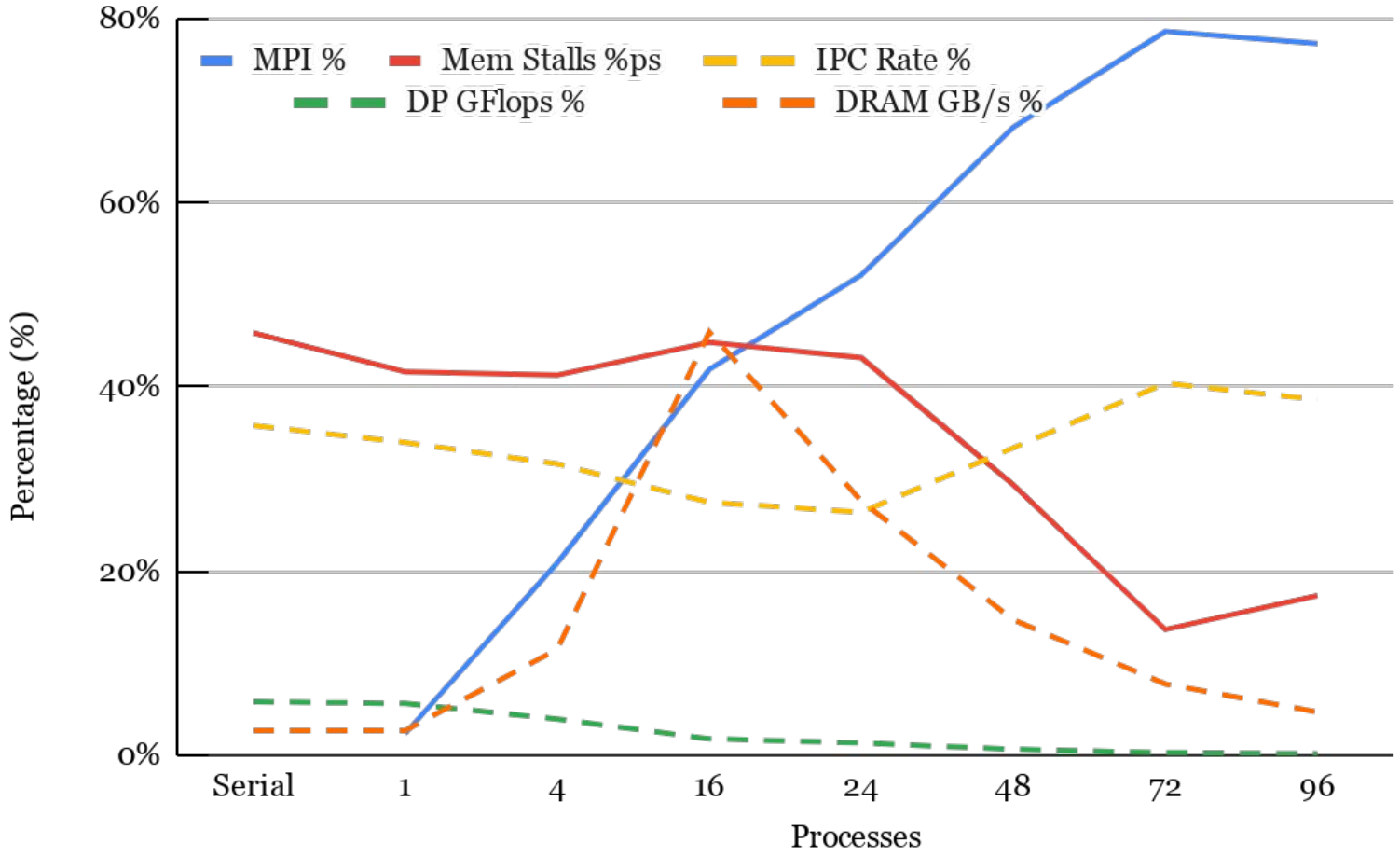
Estêncil F90: % métricas x processos MPI - nós B710



Perfil (Intel APS) - FFT 3D F90 - nós B710

		Número de processos MPI						
Parâmetros	**Serial**	**1**	**4**	**16**	**24**	**48**	**72**	**96**
Elapsed Time [s]	**20.3**	21.1	7.4	4.0	4.0	3.8	5.1	5.5
ETime - MPI [s]	**-**	**20.6**	**5.8**	**2.3**	**1.9**	**1.2**	**1.3**	**1.3**
Speedup	**-**	**1.0**	**2.7**	**5.1**	**5.1**	**5.4**	**4.0**	**3.7**
Efficiency	**-**	**1.0**	**0.7**	**0.3**	**0.2**	**0.1**	**0.1**	**0.0**
MPI Time [s]	-	0.5	1.5	1.7	2.1	2.5	3.7	4.1
MPI Time [%]	-	2.4	**20.9**	**41.9**	**52.2**	**68.3**	**78.7**	**77.3**
MPI_Init [s]	-	0.5	0.7	1.0	1.5	1.8	2.2	**2.6**
MPI_Sendrecv [s]	-	-	0.8	0.6	0.4	0.3	0.5	0.3
MPI Imbalance [s]	-	-	0.1	0.3	0.2	0.4	1.2	1.1
DP [GFlops]	1.1	1.1	3.0	5.6	6.3	6.2	4.5	3.6
IPC Rate	1.4	1.4	1.3	1.1	1.1	1.3	1.6	1.5
Bound	**mem**	**mem**	**mem**	**MPI**	**MPI**	**MPI**	**MPI**	**MPI**
Cache Stalls [%c]	9.1	8.8	19.2	17.9	18.0	13.7	9.3	10.6
DRAM Stalls [%c]	**32.2**	**29.2**	16.0	**21.0**	**21.5**	12.3	5.7	4.9
DRAM [GB/s]	-	1.6	6.9	15.5	16.5	8.8	4.6	2.8
Mem Stalls [%ps]	**45.9**	**41.7**	**41.3**	**44.9**	**43.2**	**29.4**	13.7	17.4
Vectorization [%]	**5.3**	**5.5**	**5.3**	**5.4**	**7.8**	**5.0**	**31.0**	**18.4**

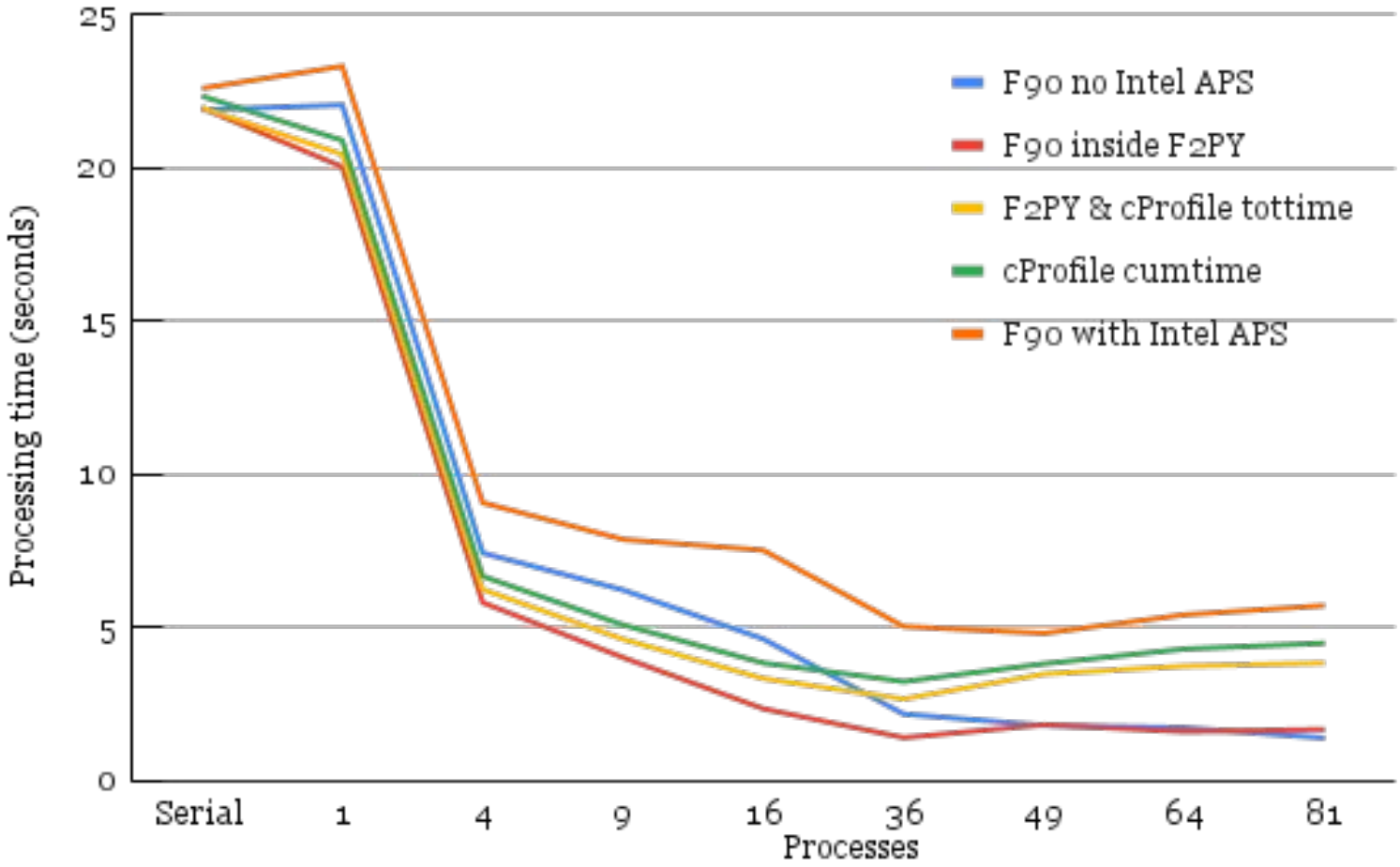
FFT 3D F90: % métricas x processos MPI - nós B710



Estêncil F90(sem APS) & F2PY(com cProfiler) [s] - nós B710

Implementação	Seq	Número de processos MPI 1	4	9	16	36	49	64	81
F90									
comando *time* do SO	21.9	22.6	8.0	7.0	5.6	3.8	3.5	4.1	3.8
chamada F90 wall time	**21.9**	22.0	7.4	6.2	4.6	2.2	**1.8**	1.7	**1.4**
F2PY									
comando *time* do SO	22.5	21.0	6.8	5.3	4.0	3.4	4.2	4.5	4.7
chamada Python wall time	21.9	20.4	6.2	4.6	3.3	2.7	3.5	3.7	3.8
chamada F90 wall time	21.9	**20.0**	**5.8**	**4.0**	**2.3**	**1.4**	1.8	**1.6**	1.7
cProfile tottime	21.9	20.4	6.2	4.6	3.3	2.7	3.5	3.7	3.8
cProfile cumtime	22.3	20.9	6.7	5.1	3.8	3.2	3.8	4.3	4.5

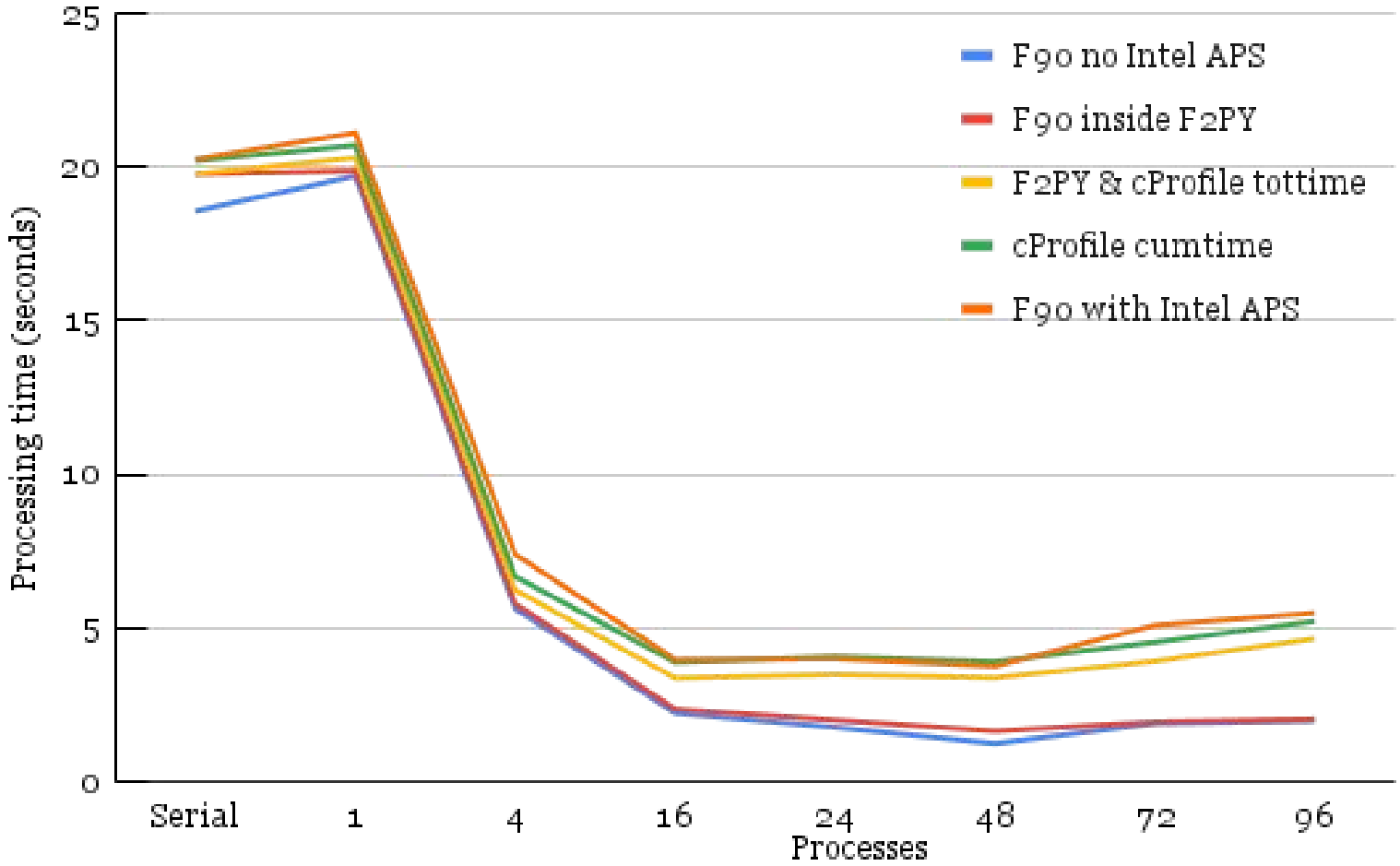
Estêncil F90 (sem APS) & F2PY (com cProfiler) - nós B710



FFT 3D F90(sem APS) & F2PY(com cProfiler) [s] - nós B710

Implementação	Seq.	Número de processos MPI 1	4	16	24	48	72	96
F90								
comando *time* do OS	18.8	20.1	6.4	3.2	3.6	3.2	4.2	4.3
chamada F90 wall time	**18.6**	**19.7**	**5.6**	**2.2**	**1.8**	**1.2**	**1.9**	**2.0**
F2PY								
comando *time* do OS	20.4	20.9	6.8	4.1	4.3	4.1	4.8	5.4
chamada Python wall time	19.8	20.3	6.2	3.4	3.5	3.4	3.9	4.6
chamada F90 wall time	19.8	19.9	5.8	2.4	2.0	1.6	1.9	2.0
cProfile tottime	19.8	20.3	6.2	3.4	3.5	3.4	3.9	4.6
cProfile cumtime	20.2	20.7	6.7	3.9	4.1	3.9	4.5	5.2

FFT 3D F90 (sem APS) & F2PY (com cProfiler) - nós B710



5

Considerações Finais

Considerações finais

Abordagens HPC/PAD comuns p/Python

3 casos de teste executados no Santos Dumont

Referências: versões F90 sequencial & MPI

Versões F2PY, Cython, Numba/GPU, Python

Comparação versões p/ métricas desempenho

Profiling (Intel APS e outros)

Trabalhos futuros

Obrigado!

Código fonte: https://github.com/efurlanm/msc22
Contato: Eduardo Furlan Miranda. Programa de Pós Graduação em Computação Aplicada (CAP) / INPE
E-mail: efurlanm@gmail.com
Orientador: Dr. Stephan Stephany. Coordenação de Pesquisa Aplicada e Desenvolvimento Tecnológico (COPDT) (CAP) / INPE
E-mail: stephan.stephany@inpe.br