

# SuperComputação

## Aula 3 – Profiling + Projeto

2020 – Engenharia

Luciano Soares <[lpsoares@insper.edu.br](mailto:lpsoares@insper.edu.br)>

Igor Montagner <[igorsm1@insper.edu.br](mailto:igorsm1@insper.edu.br)>

# HPC Linux Engineer

at Jump Trading ([View all jobs](#))

Chicago, New York, Singapore, London

This rare hands-on individual will be highly skilled in the details and nuances of managing Linux environments with a strong software development background necessary to support uniquely customized systems.

What you'll do:

- Assist in designing, implementing, and maintaining high performance file storage systems.
- Implement and support performance monitoring and fault monitoring systems
- Monitors systems and storage performance, up to and including network components
- Compiles, packages, installs and upgrades software and operating system components
- Creates scripts and uses tools to automates frequently performed tasks
- Develops, improves, documents and promotes standard operating procedures
- Develop and monitor the tools used to maintain a production computing environment
- Other duties as assigned or needed

What you'll need:

- High proficiency with at least one of the following languages: Python, Ruby, Go, C
- Extensive experience profiling and debugging application stacks
- Extensive experience ~~designing~~ building, and maintaining complicated, interdependent, and distributed systems
- Experience with system configuration management tools (Cfengine, Salt, Puppet, Ansible, etc.)
- Experience in an academic research computing (HPC) background
- A compulsion to perform root cause analysis
- Reliable and predictable availability

# Hoje

- Medição de desempenho
- Início do projeto

# Medição de tempo

# Profiling

Análise de um programa durante sua execução para determinar seu consumo de memória e/ou tempo.

Podemos responder duas importantes perguntas:

- onde o programa consome mais recursos?
- onde devo concentrar meus esforços de otimização?

# KCachegrind

**callgrind.out.7030 [./euclid] — KCachegrind**

File View Go Settings Help

Open... Back Forward Up Relative Cycle Detection Relative to Parent Shorten Templates Instruction Fetch

### Flat Profile

Self	Called	Function	Location
.00	0.00	(0)	ld-2.27.so
.96	0.00	_start	euclid
.96	0.00	(below main)	libc-2.27.so: libc-start.c
.74	0.43	main	euclid: euclid.cpp, std_vector.h,
.34	1.44	std::ostream& std::ostream::...	libstdc++.so.6.0.25
.11	0.11	std::num_put<char, std::ostr...	libstdc++.so.6.0.25
.98	3.53	std::ostreambuf_iterator<char...	libstdc++.so.6.0.25
.73	1.47	0x0000000000d9c40	libstdc++.so.6.0.25
.09	1.39	vsnprintf	libc-2.27.so: vsnprintf.c
.92	7.39	vfprintf	libc-2.27.so: vfprintf.c, printf-pa
.31	0.17	__printf_fp	libc-2.27.so: printf_fp.c
.15	26.45	__printf_fp_l	libc-2.27.so: printf_fp.c, localeir
.69	6.55	hack_digit	libc-2.27.so: printf_fp.c
.08	0.33	__gnu_cxx::stdio_sync_filebuf<...	libstdc++.so.6.0.25
.74	4.13	fwrite	libc-2.27.so: iofwrite.c, libioP.h
.53	7.53	__mpn_divrem	libc-2.27.so: divrem.c
.33	1.48	std::basic_ostream<char, std::...	libstdc++.so.6.0.25

### main

#	Ir	Co	Source
219	0.00		{ return _M_extract(__f); }
240	2.94		call(s) to 'std::istream& std::istream::_M_extract<double>(double&) (libstdc++...
941	0.00		if (this->_M_impl_M_finish != this->_M_impl_M_end_of_storage)
41	0.00		for (int i = 0; i < n; i++) {
943	0.00		_Alloc_traits::construct(this->_M_impl, this->_M_impl_M_finish,
945	0.00		++this->_M_impl_M_finish;
136	0.00		{ ::new((void *)__p)_Up(std::forward<Args>(_args)...); }
32	0.00		for (int i = 0; i < n; i++) {
948	0.00		_M_realloc_insert(end(), __x);
16	0.01		call(s) to 'void std::vector<double, std::allocator<double>>::_M_realloc_insert<d...
25	0.00		int main()

Profile Part	Incl.	Self	Called	Comment
--------------	-------	------	--------	---------

Parts Calleees Call Graph All Calleees Caller Map Machine Code

callgrind.out.7030 [1] - Total Instruction Fetch Cost: 51 844 483

# Demonstração

# Atividade prática

## Profiling na prática

1. Usar o KCachegrind para analisar nossa tentativa de otimização
2. Fazer novas otimizações e medir seu desempenho

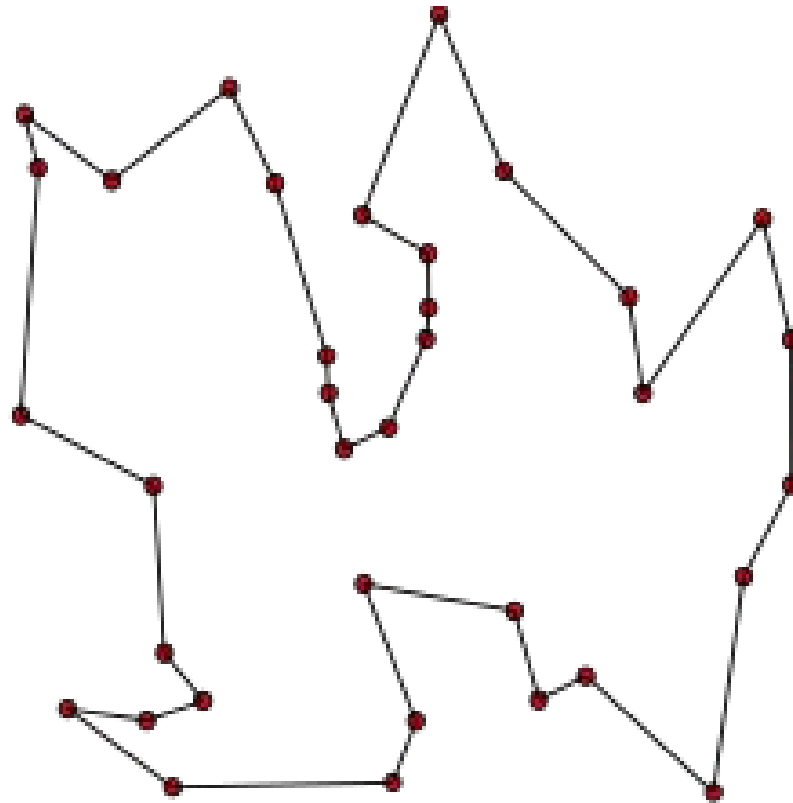


# Fechamento

Depois de escolher um bom algoritmo e ter uma implementação ingênua,

- Entrada/Saída custa caro.
- Implementações diferentes do mesmo algoritmo podem ter desempenho diferente
- Medimos esses ganhos com ferramentas de profiling.

# Projeto - Travelling Salesperson (TSP)



# Projeto - Travelling Salesperson (TSP)

Dadas  $N$  cidades, escolher um caminho fechado (tour) tal que

- cada cidade é visitada somente uma vez
- o caminho é o mais curto possível

**Problema de otimização difícil**

# Projeto - Travelling Salesperson (TSP)

Teremos três grandes partes no projeto

- algoritmos
- análise de desempenho
- paralelismo

Toda parte será conectada com alguma aula, em que faremos discussões e fixaremos um prazo para os exercícios

# Projeto - Travelling Salesperson (TSP)

<https://insper.github.io/supercomp/projetos/>

# Insper

[www.insper.edu.br](http://www.insper.edu.br)