



WORKSHOP

13. - 16. June 2022

Prague, Czech Republic



Co-financed by the Connecting Europe
Facility of the European Union



TERRASIGNA™

High performance computing in python

Jun 13, 2021: 13:30 - 15:00

<https://bit.ly/3HfEim1>



Leandro Parente

leandro.parente@opengeohub.org



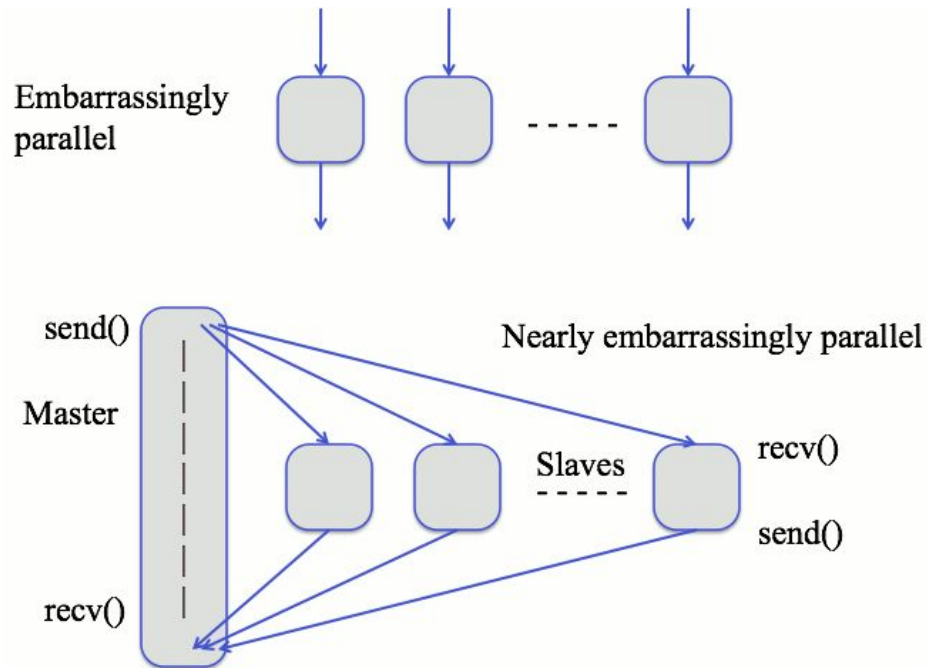
<https://opengeohub.org>

Introduction to ODSE datasets in Python - Outline

- Embarrassingly parallel problems
- Possibilities to optimize a raster processing workflow
- BLAS and LAPACK implementations
- Optimizing a temporal array reduction and a numeric operations

Embarrassingly parallel problems

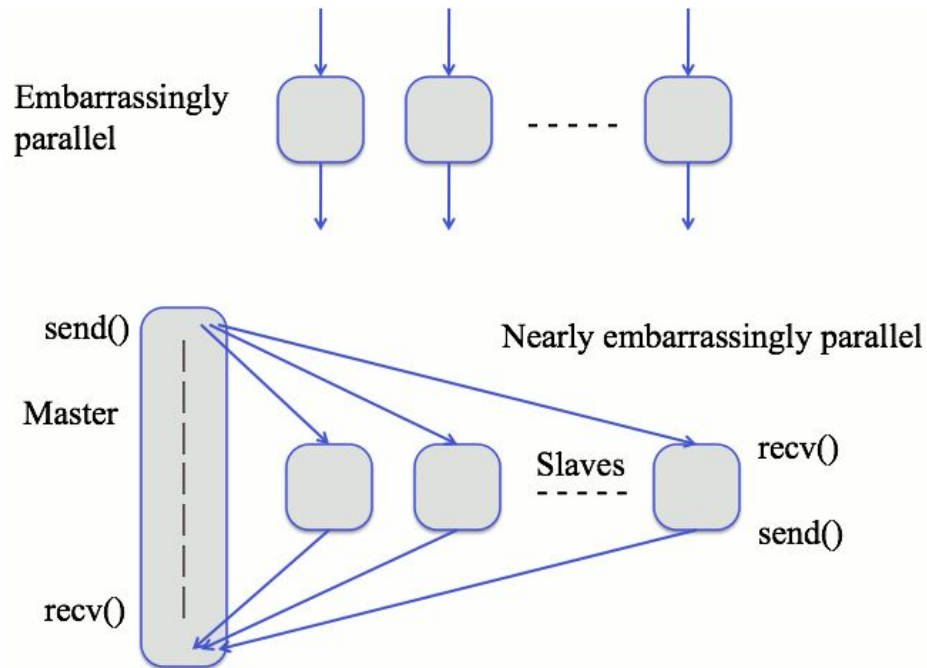
- Can be divided into completely **independent** parts,
- Requires none or very little communication,
- **Nearly embarrassingly parallel** is an embarrassingly parallel computation that requires initial data to be distributed and final results to be collected in some way



Source: [Embarrassingly Parallel Computations](#) & [Embarrassingly Parallel Algorithms](#)

Embarrassingly parallel problems

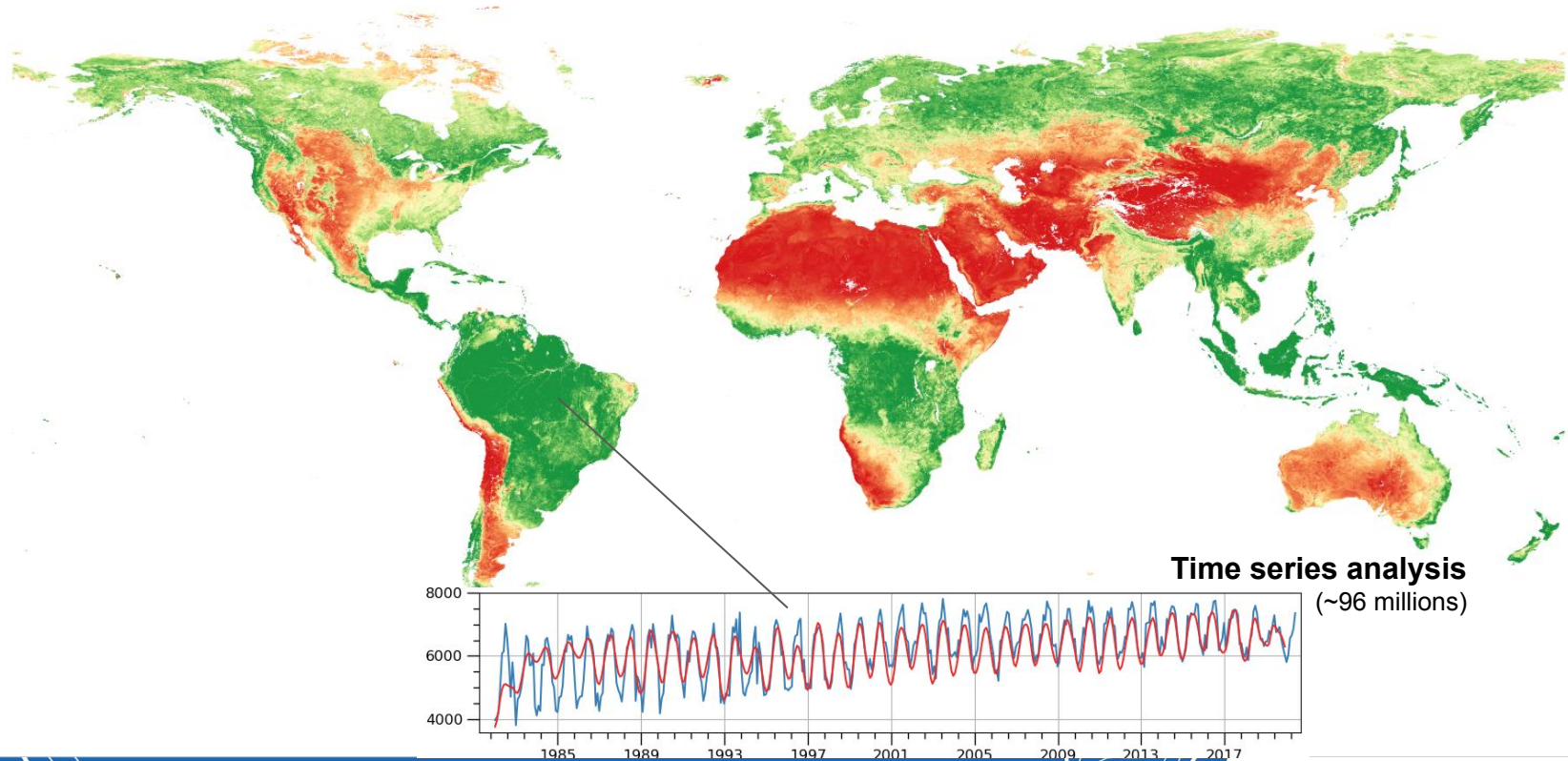
- Can be divided into completely **independent** parts,
- Requires none or very little communication,
- **Nearly embarrassingly parallel** is an embarrassingly parallel computation that requires initial data to be distributed and final results to be collected in some way



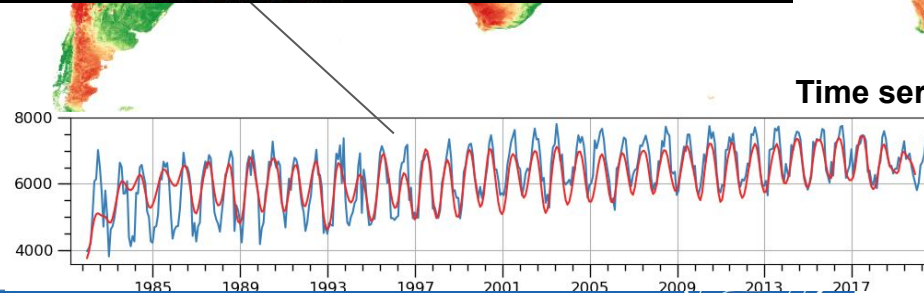
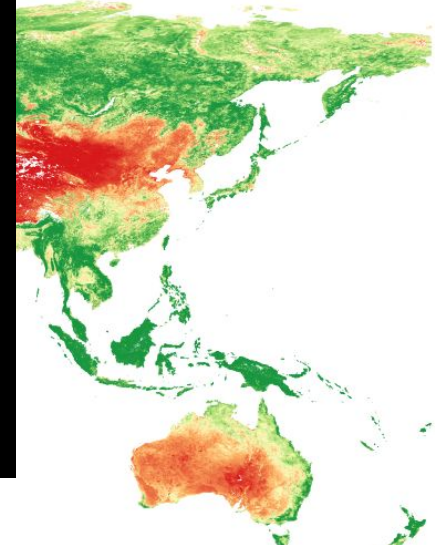
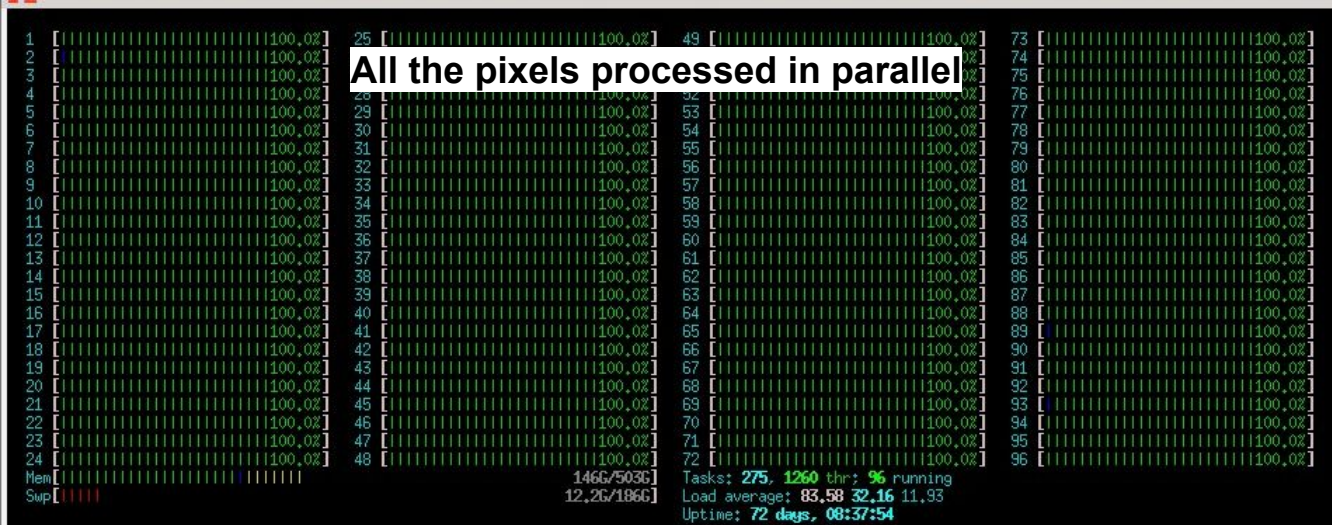
Raster processing

Source: [Embarrassingly Parallel Computations](#) & [Embarrassingly Parallel Algorithms](#)

Embarrassingly parallel problems



Embarrassingly parallel problems



Possibilities to optimize a raster processing workflow

- Increase the number of CPU cores
- Improve data transfer speed
- Improve the processing code (new algorithms/functions):



Possibilities to optimize a raster processing workflow

- Increase the number of CPU cores
- Improve data transfer speed
- Improve the processing code (new algorithms/functions):
 - Drop-in replacement
 - New code implementation



BLAS and LAPACK implementations

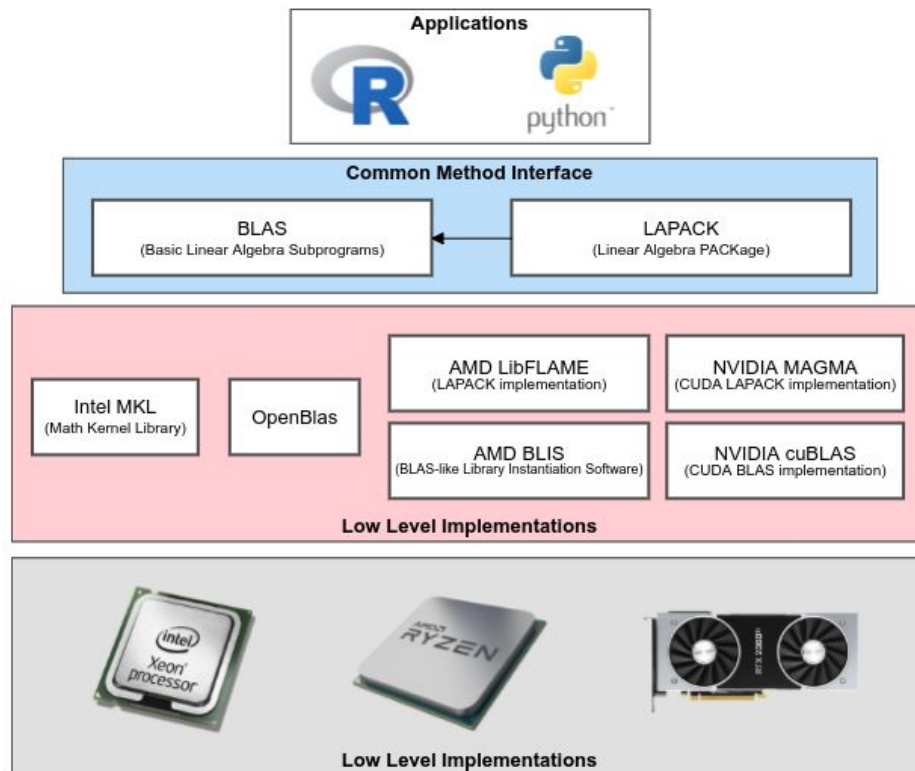
BLAS (Basic Linear Algebra Subprograms) is a C library to provide a set of routines for basic vector and matrix operations

LAPACK (Linear Algebra PAcage) Fortran 90 library to solve linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, singular value problems and the associated matrix factorization

Level 1 BLAS

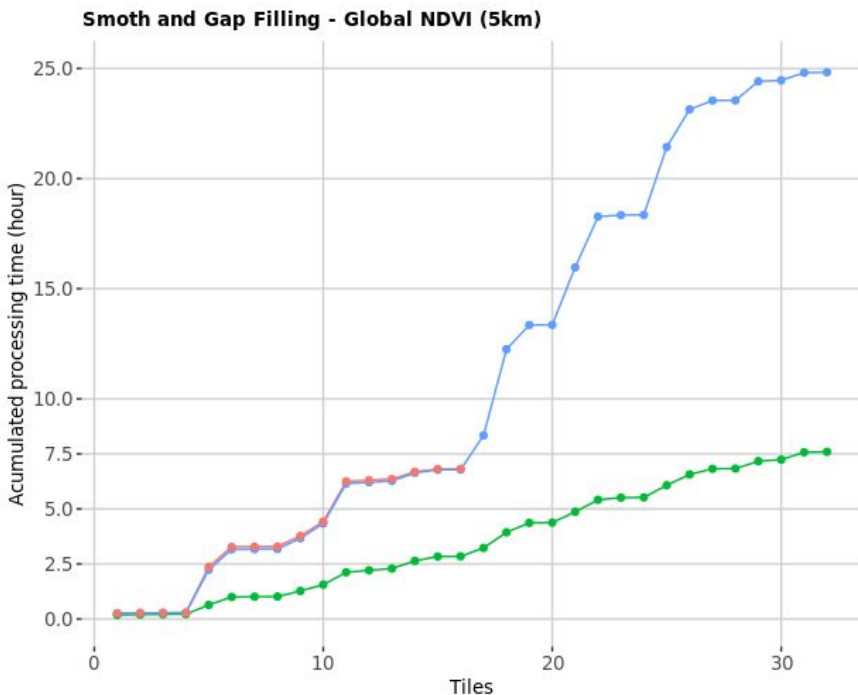
	dim	scalar	vector	vector	scalars	5-element array		prefixes
SUBROUTINE xROTG (A, B, C, S)		Generate plane rotation	S, D
SUBROUTINE xROTMG (D1, D2, A, B, C, S)	PARAM)	Generate modified plane rotation	S, D
SUBROUTINE xROT (N,			X, INCX, Y, INCY,		C, S)	PARAM)	Apply plane rotation	S, D
SUBROUTINE xRTM (N,			X, INCX, Y, INCY,				Apply modified plane rotation	S, D
SUBROUTINE xSWAP (N,			X, INCX, Y, INCY)				$x \leftrightarrow y$	S, D, C, Z
SUBROUTINE xSCAL (N,	ALPHA,		X, INCX)				$x \leftarrow \alpha x$	S, D, C, Z, CS, ZD
SUBROUTINE xCOPY (N,			X, INCX, Y, INCY)				$y \leftarrow x$	S, D, C, Z
SUBROUTINE xAXPY (N,	ALPHA,		X, INCX, Y, INCY)				$y \leftarrow \alpha x + y$	S, D, C, Z
FUNCTION xDOT (N,			X, INCX, Y, INCY)				$dot \leftarrow x^T y$	S, D, DS
FUNCTION xDOTU (N,			X, INCX, Y, INCY)				$dot \leftarrow x^T y$	C, Z
FUNCTION xDOTC (N,			X, INCX, Y, INCY)				$dot \leftarrow x^H y$	C, Z
FUNCTION xxDOT (N,			X, INCX, Y, INCY)				$dot \leftarrow \alpha + x^T y$	SDS
FUNCTION xNRM2 (N,			X, INCX)				$nrm2 \leftarrow \ x\ _2$	S, D, SC, DZ
FUNCTION xASUM (N,			X, INCX)				$asum \leftarrow \ re(x)\ _1 + \ im(x)\ _1$	S, D, SC, DZ
FUNCTION IxAMAX (N,			X, INCX)				$amax \leftarrow 1^{st} k \ni re(x_k) + im(x_k) $ $= \max(re(x_i) + im(x_i))$	S, D, C, Z

BLAS and LAPACK implementations



BLAS and LAPACK implementations

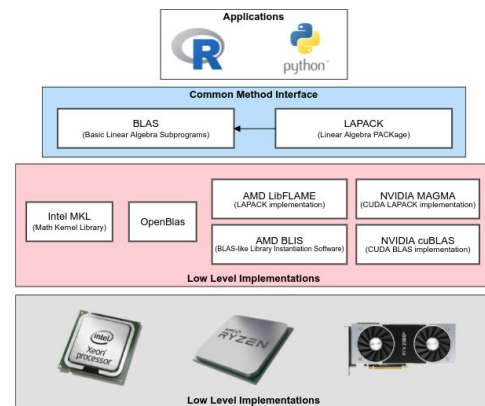
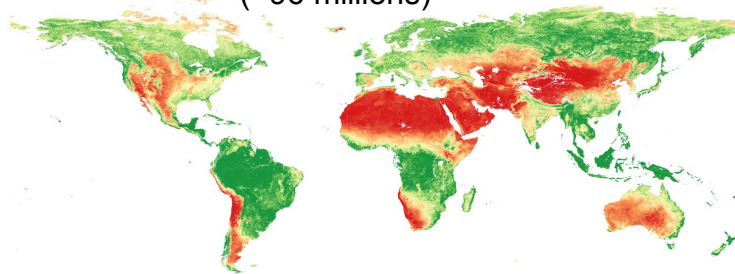
MKL is 3x faster than OpenBlas



cat

- Blis+LibFLAME ((AMD EPYC 7702P - 108 Threads)
- MKL (Intel Xeon Gold 6248 - 80 Threads)
- OpenBlas (AMD EPYC 7702P - 108 Threads)


Time series analysis
(~96 millions)



Hands-on

<https://colab.research.google.com/drive/1ekDvZB3Zz9Y7JE42YjN6U97S7oMjG39m>

https://gitlab.com/geoharmonizer_inea/odse-workshop-2022/-/blob/main/python_training/03_high_performance_computing.ipynb

 03_high_performance_computing.ipynb ☆

File Edit View Insert Runtime Tools Help Last saved at 12:12 PM

+ Code + Text

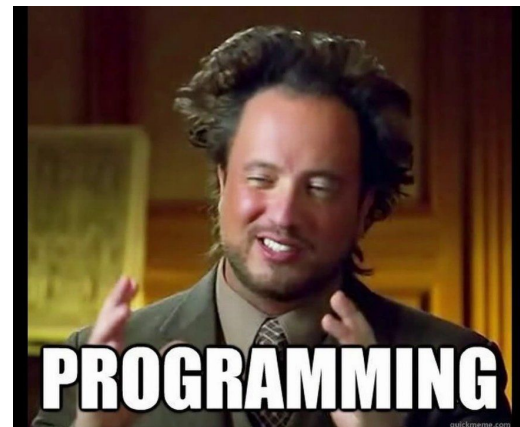
High performance computing in Python

In this tutorial, you will learn how to use different Python libraries, compatible with Numpy, to reduce the computational time for multidimensional array operations. The input data for the operations will be prepared considering several raster layers accessible through cloud ([STAC](#) + [COG](#)) and a specific bounding box provided by the user, using [ipyleaflet](#), for any part of Europe.

Additionally to Numpy, the follow libraries/modules will be used during the tutorial:

- [Bootleneck](#): Collection of Numpy fast functions written in C, which are able to manage NaN values,
- [NumExpr](#): Numeric expression evaluator memory optimized,
- [Numba](#): Just-in-time Python compiler able to translate user defined numerical algorithms in C or FORTRAN,

Python environment & libraries





WORKSHOP

13. - 16. June 2022

Prague, Czech Republic



Co-financed by the Connecting Europe
Facility of the European Union

