Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
ICSC Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

BrahMap

A scalable map-making framework for the future CMB experiments

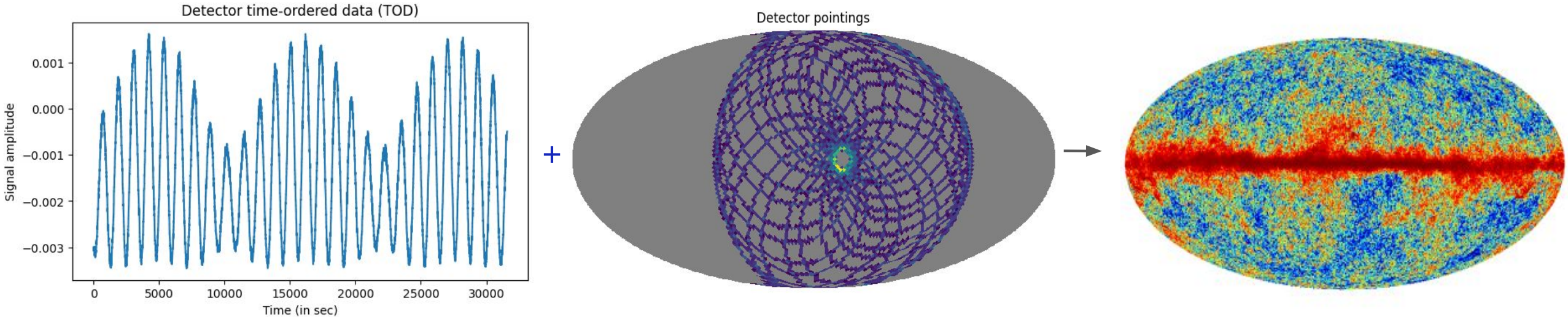
Avinash Anand[1], *Giuseppe Puglisi*[2]

[1]*University of Rome "Tor Vergata"*, [2]*University of Catania*

Spoke 3 General Meeting, Elba 5-9 / 05, 2024

Scientific Rationale

- Future CMB experiments: Targeting the B-mode polarization of CMB
- Detectors: $O(10^3)$ - $O(10^5)$ in number with a very high sampling rate
- Data acquisition: ~250 TB (from space) to ~10 PB (from ground)
- First step of analysis: Reduction of time-series data to sky maps *aka* Map-making
- Map-making goals:
  - Reduction of enormous amount of data in a reasonable timeframe
  - Mitigation of instrumental systematics
  - Removal of both un-correlated and correlated noise



- First step of analysis: Reduction of time-series data to sky maps *aka* Map-making
- Map-making goals:
  - Reduction of enormous amount of data in a reasonable timeframe
  - Mitigation of instrumental systematics
  - Removal of both un-correlated and correlated noise

Technical Objectives, Methodologies and Solutions

Data model of the sky signal

$$d_{p,t} = I_p + Q_p \cos 2\phi_t + U_p \sin 2\phi_t + n_t$$

- $d_{p,t}$ → Signal measured by the detector
- ϕ_t → Detector polarization angle
- I_p, Q_p, U_p → CMB stokes parameters
- n_t → Un-correlated and correlated noise contribution

Data model in matrix equation form

$$d = P \cdot s + n$$

- d → Signal vector
- n → noise vector
- s → Sky map vector
- P → Pointing matrix (sparse matrix with 3 non-zero elements per row)

Generalized least-square solution (GLS)

$$\hat{s} = \left(P^T N^{-1} P\right)^{-1} P^T N^{-1} d$$

$N \rightarrow n_t \, x \, n_t$ Noise covariance matrix

Technical Objectives, Methodologies and Solutions

$$d = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{n_t-2} \\ d_{n_t-1} \\ d_{n_t} \end{bmatrix}_{n_t \times 1} \qquad P = \begin{bmatrix} \dots & \dots & 1 & \cos 2\phi_{t_1} & \sin 2\phi_{t_1} & \dots \\ \dots & 1 & \cos 2\phi_{t_2} & \sin 2\phi_{t_2} & \dots & \dots \\ \dots & \dots & 1 & \cos 2\phi_{t_3} & \sin 2\phi_{t_3} & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & 1 & \cos 2\phi_{t_{n_t-2}} & \sin 2\phi_{t_{n_t-2}} & \dots \\ \dots & 1 & \cos 2\phi_{t_{n_t-1}} & \sin 2\phi_{t_{n_t-1}} & \dots & \dots \\ \dots & 1 & \cos 2\phi_{t_{n_t}} & \sin 2\phi_{t_{n_t}} & \dots & \dots \end{bmatrix}_{n_t \times 3\,\mathrm{npix}} \qquad \hat{s} = \begin{bmatrix} I_1 \\ Q_1 \\ U_1 \\ \vdots \\ I_{N_p} \\ Q_{N_p} \\ U_{N_p} \end{bmatrix}_{n_t \times 1}$$

CPU hours needed for $n_t \sim 10^9$:

~ 250,000 to 2,000,000

Generalized least-square solution (GLS)

$$\hat{s} = \left(P^T N^{-1} P\right)^{-1} P^T N^{-1} d$$

$N \rightarrow n_t \times n_t$ Noise covariance matrix

Technical Objectives, Methodologies and Solutions

- **BrahMap : A scalable map-making framework for future CMB experiments**

- A modular and object-oriented map-making framework based on COSMOMAP2[1,2]
- Python3 interface with C++ backend for compute-intensive parts

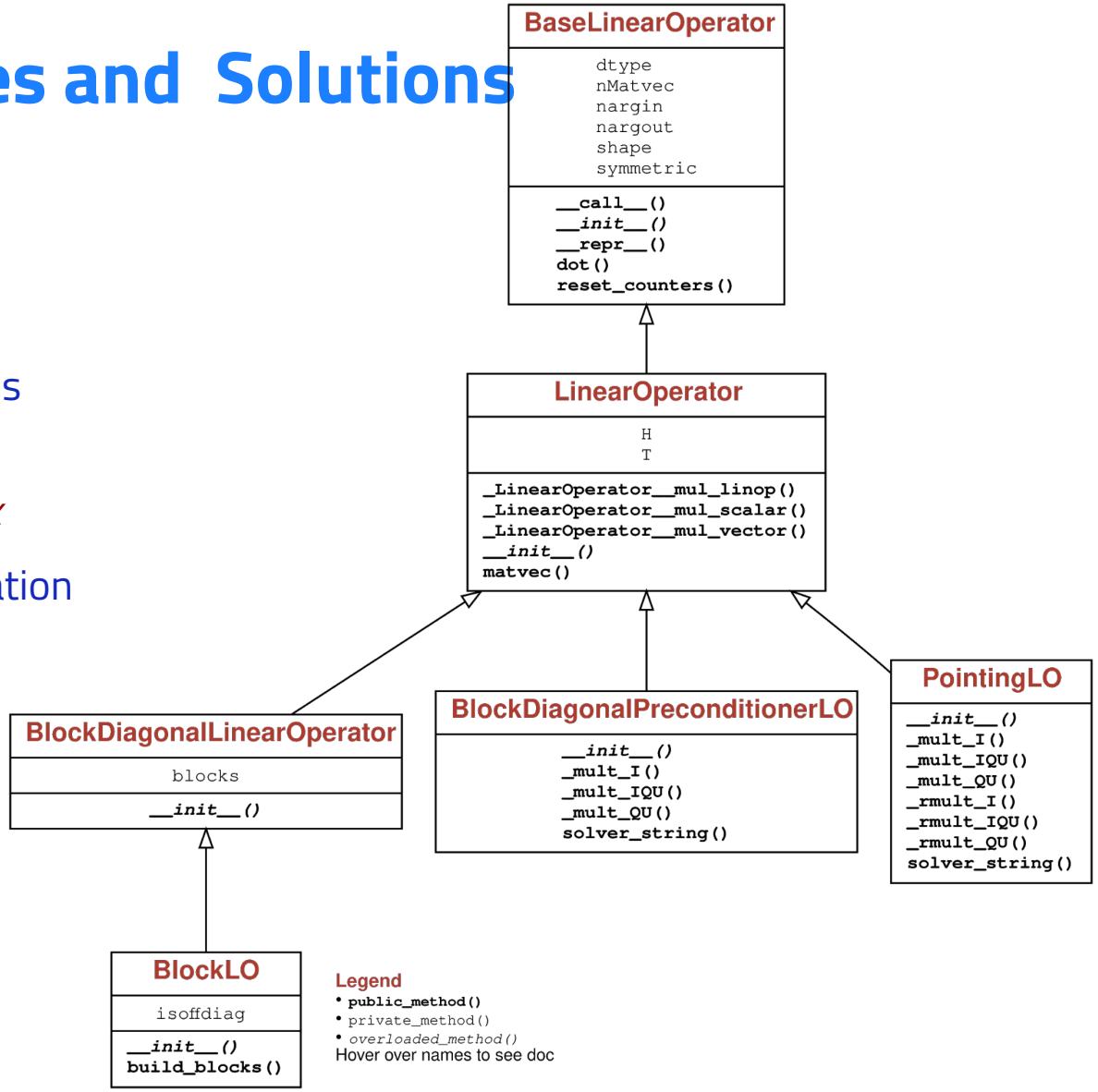
- Optimization to squeeze the most out of the supercomputing resources
- Scalability across multiple computing nodes
- Offloading the computations to multiple GPUs

[1]Puglisi, G., et al. "*Iterative map-making with two-level preconditioning for polarized cosmic microwave background data sets - A worked example for ground-based experiments.*" A&A, 618 (2018) A62, https://doi.org/10.1051/0004-6361/201832710
[2]https://github.com/giuspugl/COSMOMAP2

Technical Objectives, Methodologies and Solutions

- **BrahMap** : **Object-oriented framework**

- Matrix operators are defined using `LinearOperator` class
- `LinearOperator` class implements the linear mappings $x \longrightarrow A(x)$ for matrix-vector multiplication for a given vector x
  - `__mul__()` → matrix-scalar, matrix-vector multiplication
- Other derived operations:
  - `__mul__()` → matrix-matrix multiplication
  - `__add__()` → matrix addition
  - `__sub__()` → matrix subtraction
  - `__pow__()` → exponent operator
  - `T` → transpose operator
  - `to_array()` → explicit matrix form



Technical Objectives, Methodologies and Solutions

- **BrahMap** : Usage (GLS implementation)

```
inv_cov = InvNoiseCovLO_Uncorrelated(diag=np.ones(len(pointings)))    # Inverse noise covariance, N

processed_samples = ProcessTimeSamples(npix=npix, pointings=pointings,
                pointings_flag=pointings_flag, pol_angles=pol_angles,
                noise_weights=inv_cov.diag)                           # Pre-processing the time samples

pointing_op = PointingLO(processed_samples)                           # Pointing Matrix, P

preconditioner = BlockDiagonalPreconditionerLO(processed_samples)     # Jacobi preconditioner,
                                                                        M = P^T.diag(N)^-1.P

b = pointing_op.T * inv_cov * tod                                     # b = P^T N^-1 d
A = pointing_op.T * inv_cov * pointing_op                             # A = P^T N^-1 P

map_vector = scipy.sparse.linalg.cg(A, b, M=preconditioner)           # Solve for sky map vector,
                                                                        s in A*s = b preconditioned with M
```

Timescale, Milestones and KPIs

Milestone 7
Sept 23 - Feb 24

- Validation of the code with simulation data
- Writing the C++ extensions for compute intensive parts
- Code profiling and identification of the bottlenecks
- **KPI:**
  a. Code and documentation release
  b. **11x** performance improvement compared to original code

---

Milestone 8
Mar 24 - Ongoing

- Code refactoring - sustainability, user-friendliness
- Continuous integration - unit tests
- Code optimization - vectorization
- Parallelization with OpenMP + MPI (in progress)
- **KPI:**
  a. **22x** performance improvement compared to original code

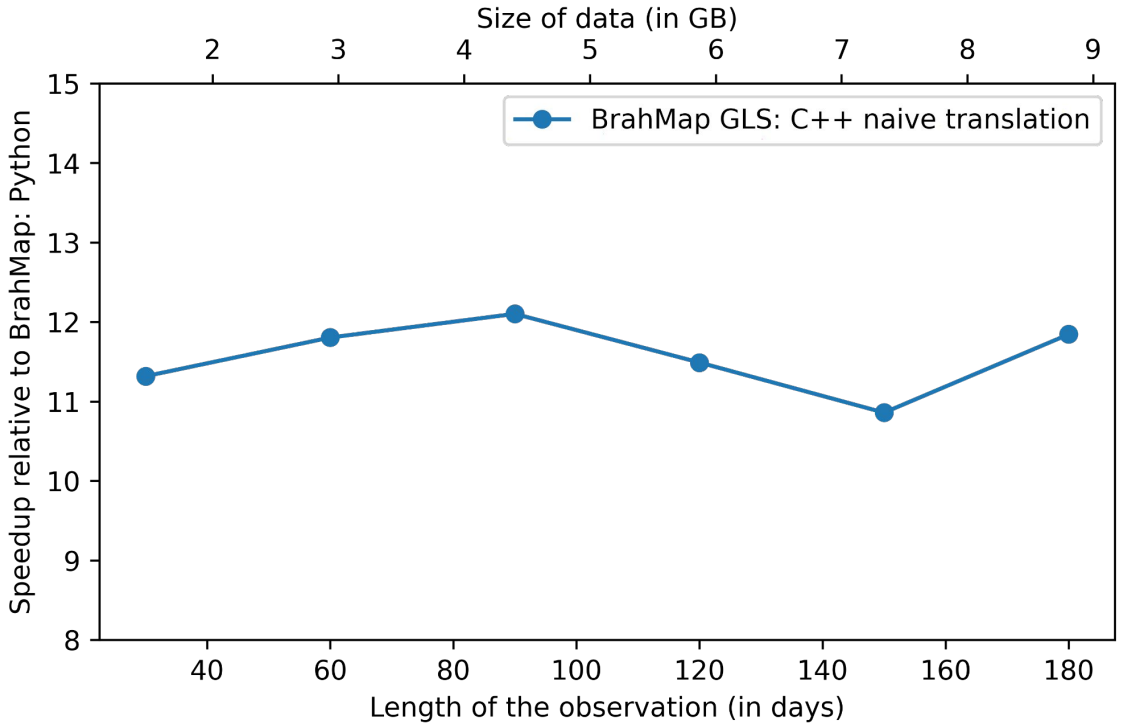
---

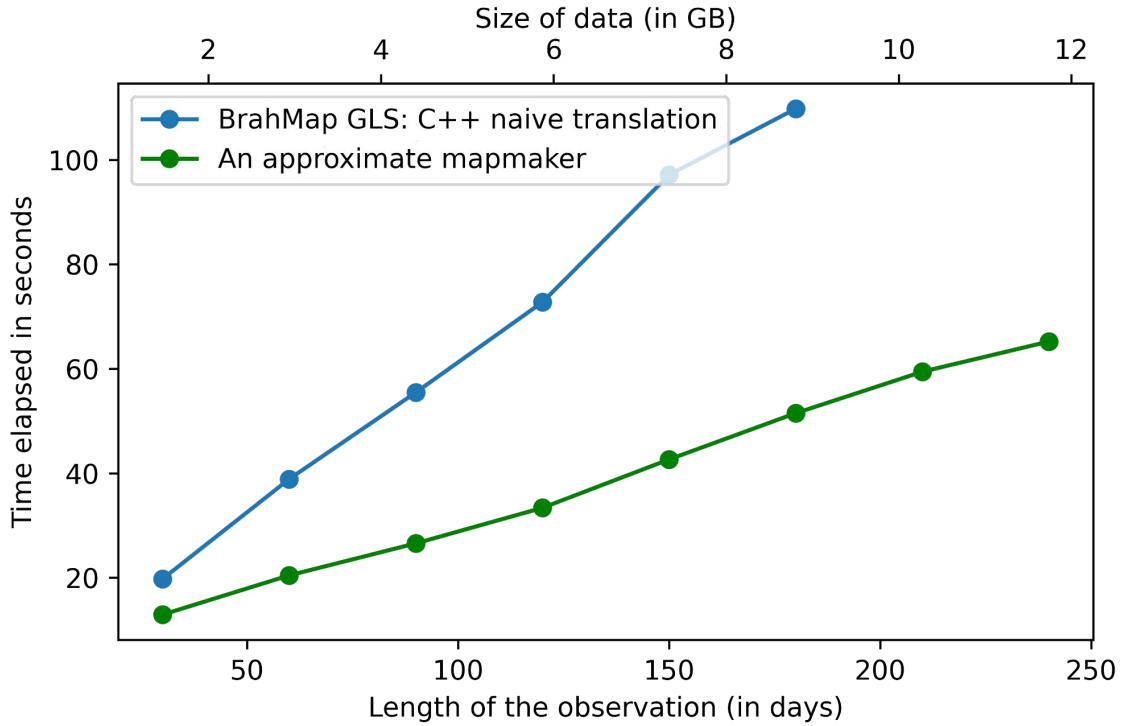
Milestone 9, 10

- GPU offloading and performance benchmark on large datasets
- Function instrumentation for profiling and benchmark

Accomplished Work, Results

- Identification of bottleneck





Accomplished Work, Results

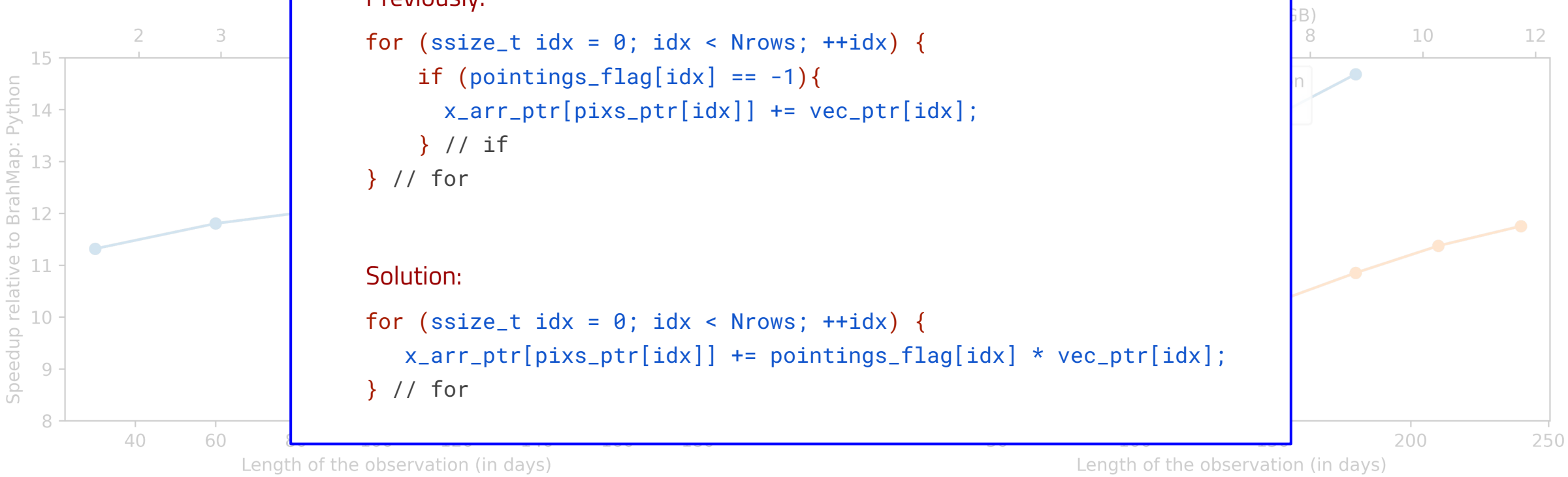
- Identification of bottleneck

Previously:

```
for (ssize_t idx = 0; idx < Nrows; ++idx) {
    if (pointings_flag[idx] == -1){
      x_arr_ptr[pixs_ptr[idx]] += vec_ptr[idx];
    } // if
} // for
```

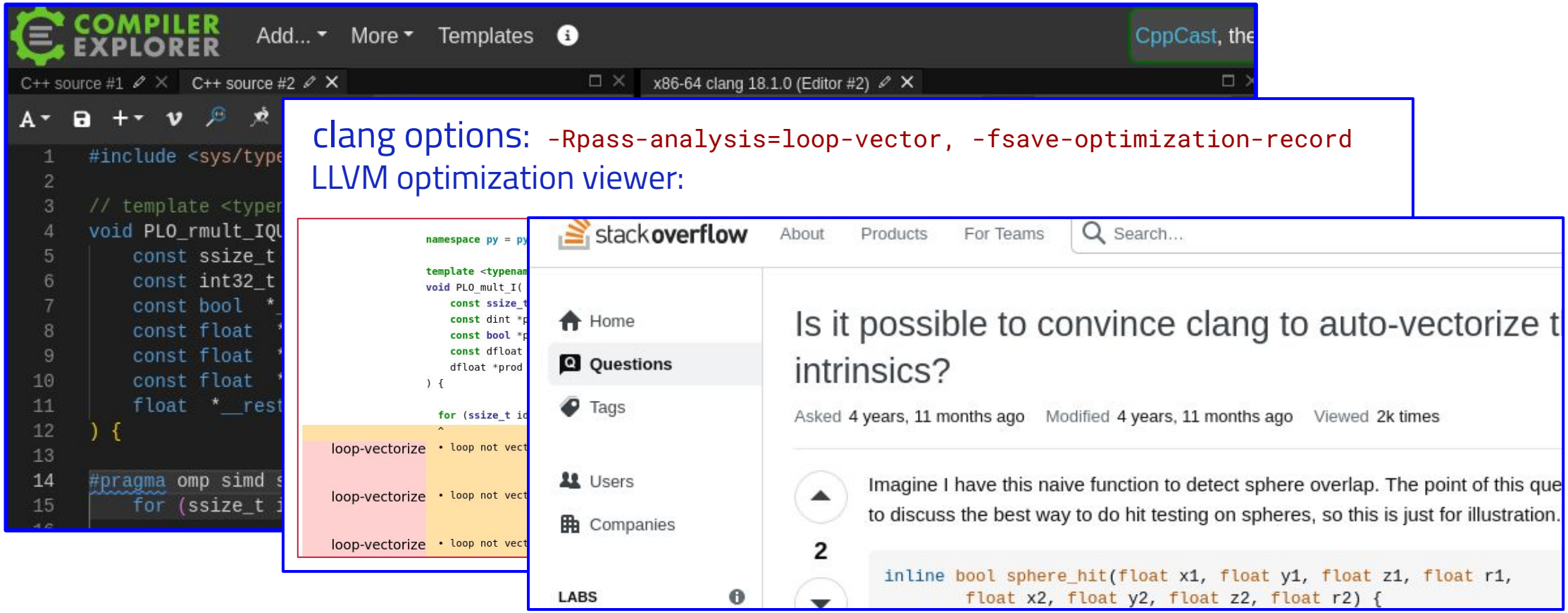
Solution:

```
for (ssize_t idx = 0; idx < Nrows; ++idx) {
   x_arr_ptr[pixs_ptr[idx]] += pointings_flag[idx] * vec_ptr[idx];
} // for
```



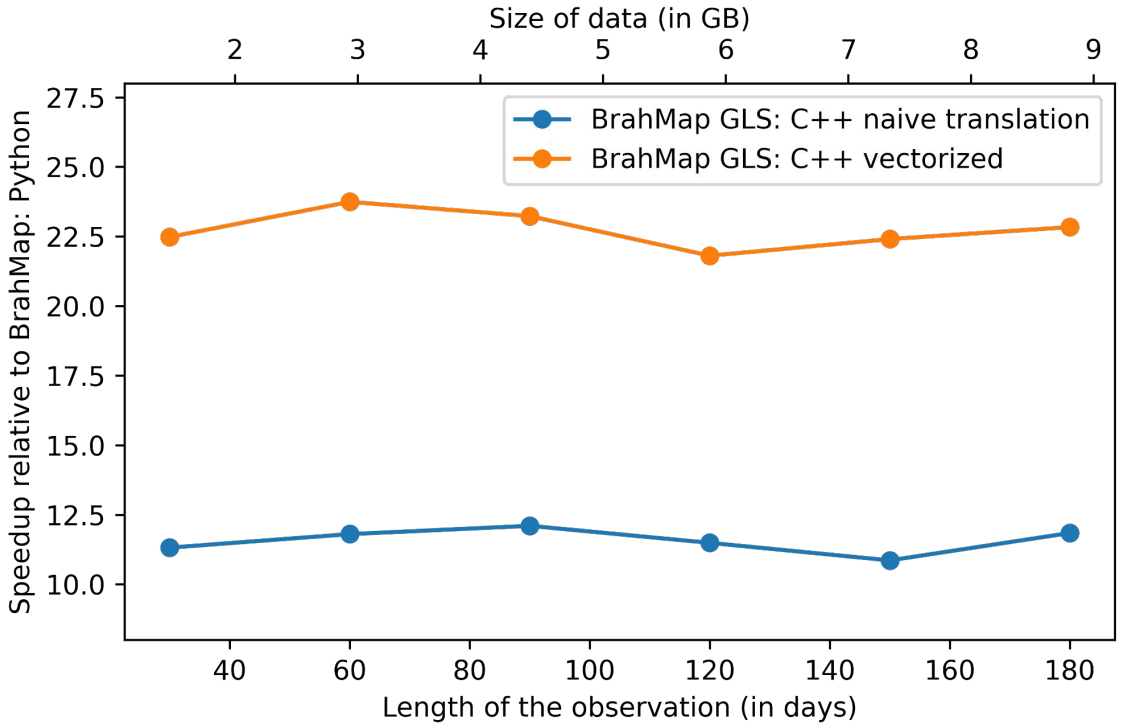
Accomplished Work, Results

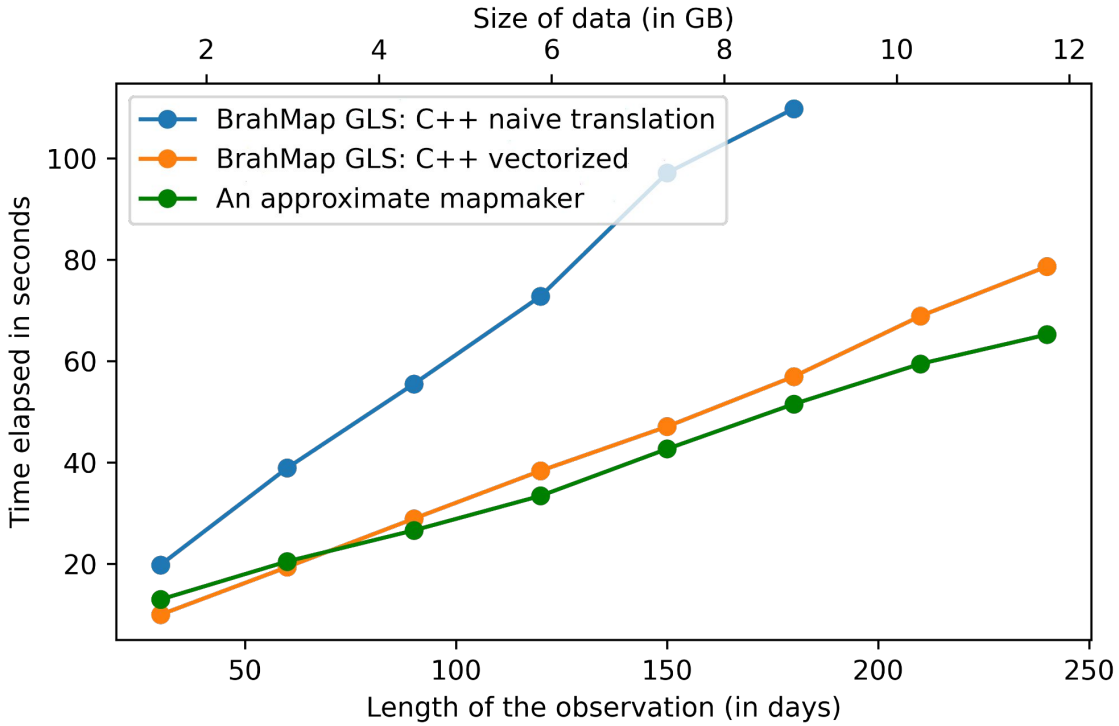
- Performance improvement



Accomplished Work, Results

- Performance improvement: Now **~22x faster** than the original code





Accomplished Work, Results

- Extensive unit testing parameterized with `int32_t | int64_t` and `float | double` data types

```
(brahmap_beta) dexm@inspiron:[BrahMap]$ pytest
========================================= test session starts ==========================================
platform linux -- Python 3.11.7, pytest-7.4.4, pluggy-1.4.0
rootdir: /mnt/Data/Projects/uniroma2/coding/dev_cosmomap2/BrahMap
configfile: pyproject.toml
plugins: anyio-4.2.0
collected 136 items

tests/test_BlkDiagPrecondLO.py ......................                                          [ 17%]
tests/test_BlkDiagPrecondLO_tools_cpp.py ...........                                           [ 26%]
tests/test_InvNoiseCov_tools_cpp.py ....                                                       [ 29%]
tests/test_PointingLO.py ......................                                                [ 47%]
tests/test_PointingLO_tools_cpp.py ...........                                                 [ 55%]
tests/test_ProcessTimeSamples.py ......................                                        [ 73%]
tests/test_compute_weights_cpp.py ..................                                           [ 88%]
tests/test_repixelization_cpp.py ................                                              [100%]

========================================= warnings summary =========================================
================================== 136 passed, 2 warnings in 24.53s ==================================
```

Next Steps and Expected Results

- Documentation update

1-2 Months

- **KPI**: Hybrid parallelization with OpenMP + MPI
  - Passing MPI communicator from Python to C++
  - Handling almost every MPI communication on C++ side

---

1 Month

- Profiling and benchmark
  - Function instrumentation on C++ side
  - Time profiler on Python side

2-4 Months

- **KPI**: Offloading to GPUs
  - cupy arrays on Python side
  - Exposing cupy arrays to C++ with Array API

Supplementary Slides

BrahMap: Derivation from COSMOMAP2

- Conversion of codebase from Python 2 to Python 3
  - Automated conversion using Python `2to3` tool followed by manual debugging and validation
- Writing the compute extensive parts to C++
  - To control hardware specific low level optimization
  - To use generic data types to using templates (to use generic Python data types)
- Writing Python binding for C++ codes using `pybind11`
  - `pybind11` is a header-only lightweight library, with no dependencies
  - `pybind11` can be shipped with Python package
  - Supports C++11 and STL out of the box
  - Compatible with all major compiler