# GEOGRAPHER

MACSy

HU Berlin · Institut für Informatik · Modellierung und Analyse komplexer Systeme

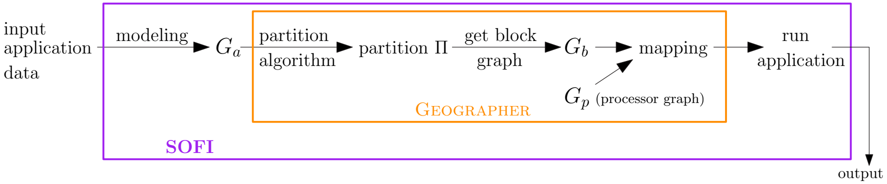
http://bit.ly/hub-macsy

## What is GEOGRAPHER?

**A tool for partitioning graphs, meshes and points sets.**

- Algorithms to partition point sets: *k*-means, Space Filling Curves (SFC), MultiSection.
- Improve a partitioned mesh by applying a local refinement approach combined with a multilevel coarsening scheme.
- Improve the load balance of an partitioned mesh by doing repartition.
- Given information about the physical network, map blocks to processors.

# Part of the $\mathcal{WAVE}$ toolbox



## Fraunhofer Scai - Lama

A framework for developing hardware-independent, high performance code.

GEOGRAPHER's and SOFI's implementations use Lama mainly for operations on distributed data (like DenseVector and CSRSparseMatrix) that may involve communications between processors and linear algebra operations etc.

## High level view

1. Input $G_a$, a mesh, i.e., an embedded graph in 2 or 3 dimensions.
   - Read $G_a$ from two provided files: one for the graph and one for the coordinates, see at FileIO class.
   - Another option is to create uniform meshes, see at MeshGenerator class.
2. At first, vertices are distributed among PEs using a BlockDistribution. To increase locality, vertices are redistributed based on their Hilbert index, see HilbertCurve class.
3. Partition the points (using only the geometric information) using KMeans or MultiSection.
4. Further improve the cut of the partition by using LocalRefinemnt and MultiLevel.
5. Different parameters can be controlled via the Settings struct. To measure the quality of the partition use Metrics struct.

## Using GEOGRAPHER as an executable

See the README file and the documentation about how to install GEOGRAPHER and the required libraries.

**Minimal example**

```
mpirun -np k installation/path/Geographer --dimensions 2 --graphFile
rotation-00000.graph --coordFile rotation-00000.graph.xyz
```

Partition the 2D mesh `rotation` into k blocks with a 3% imbalance.

More options:

- `numBlocks`: the number of desired blocks, default is k
  warning: local refinement works only when `numBlocks`=k.
- `epsilon`: desired imbalance, default is 3%
- `fileFormat`: to read different file formats, default is METIS

## More option to control the partitioner

- `outFile`: the name of the file to store metrics and the partition.
- `initialPartition`: choose the initial, geometric partition method between geoSFC for the Hilbert curve, geoKmeans for the balanced *k*-means, geoHierKM for a hierarchical version of *k*-means and geoMS for the MultiSection. See also ITI::Tool; default is geoKmeans
- `multiLevelRounds`: number of rounds for multilevel scheme.
- `minBoorderNodes`: number of nodes considered in the local refinement, default 1.
- `minSamplingNodes`: starting sample size for *k*-means, default 100.
- `metricsDetail`: detail level of the metrics possible values: no, easy, all; default no.
- `noRefinement`: if true then no local refinement is done, default false.

More options can be found in class Setting of using the `--help` flag when calling the executable.

## Using GEOGRAPHER as a library

Main entry point is function:

```
static scai::lama::DenseVector<IndexType>
                        ITI::ParcoRepart::partitionGraph(
    scai::lama::CSRSparseMatrix<ValueType>& graph,
    std::vector<DenseVector<ValueType>>& coordinates,
    std::vector<DenseVector<ValueType>>& nodeWeights,
    struct Settings settings,
    struct Metrics& metrics);
```

The input consists of the graph, the coordinates and the node weights and it is controlled by various parameters in the Settings struct. It returns a distributed dense vector with the block that every point in assigned to.

warning: input can be redistributed during the partitioning.

# Minimal example program

```
//read graph from file
scai::lama::CSRSparseMatrix<ValueType> graph =
    ITI::FileIO<IndexType, ValueType>::readGraph(file);
//read coordinates
std::vector<scai::lama::DenseVector<ValueType>> coords =
    ITI::FileIO<IndexType, ValueType>::readCoords(
            coordsFile, N, dims);

struct Settings settings;
//set different parameter values: settings.X= ... ;
//init metris
struct Metrics metrics(settings);

// get partition; use wrapper without node weights
scai::lama::DenseVector<IndexType> partition =
    ITI::ParcoRepart<IndexType, ValueType>::partitionGraph(
        graph, coords, settings, metrics);
```

## Compilation, installation

- Pay attention that all used libraries (lama, parmetis, e.t.c.) must be compiled with the same compiler otherwise linking could fail.
- The templated classes of GEOGRAPHER are instantiated only for ValueType as *float* or *double*. IndexType is taken from the lama installation.
- An example cmake can be found in Notes.
- For how to call each function, examples can be found in the src/*Test.cpp files.

## cmake

Example of how to use GEOGRAPHER in your project with cmake using `find_package()`.
`Geographer_DIR` is the installation path to geographer that is passed to cmake like
`cmake ... -DGeographer_DIR=/path/to/geographer/install ...`

```
set( GeoPathHints ${Geographer_DIR} $ENV{Geograhper_HOME} )
find_package( Geographer CONFIG PATHS ${GeoPathHints} )
include_directories( ${Geographer_INCLUDE_DIRS} )
add_library( my_lib ${SOURCE_FILES} )
target_link_libraries( my_lib ${Geographer_LIBRARIES} )
#if external libraries are installed with geographer,
#    e.g. parmetis, parhip, zoltan
add_definitions( "-DUSE_GEOGRAPHER_WRAPPERS" )
```