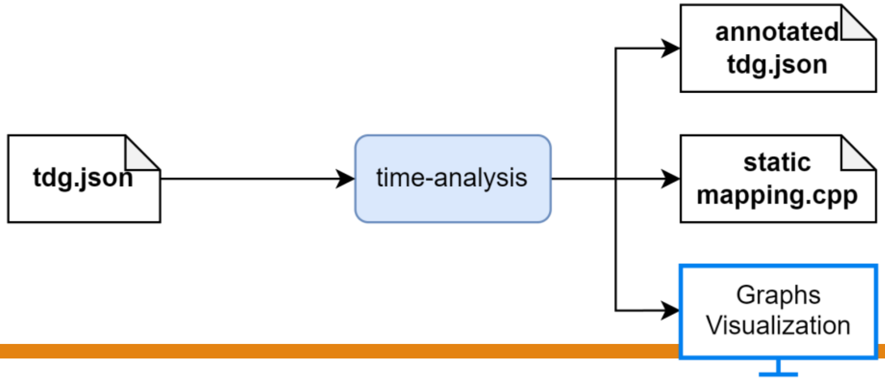
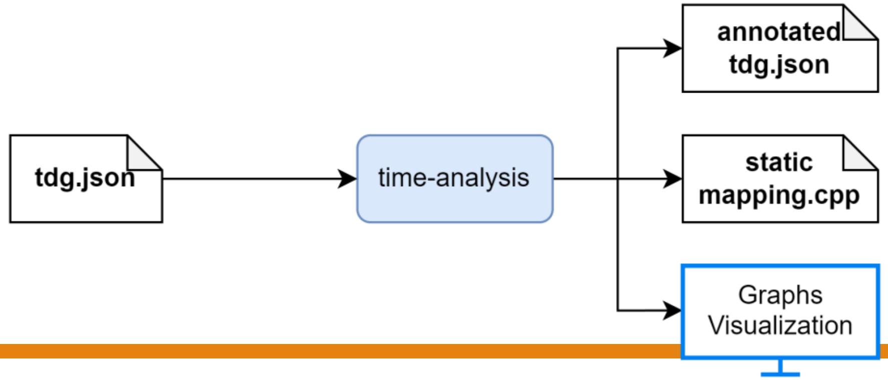
3. Time Analysis Phase

- Annotates TDG with a set of metrics based on
  - performance results per task
  - performance results of overall TDG
  - TDG structure
- Calculates static mapping using one of the existing heuristic-based mapping algorithm
- Explores multiple mapping algorithms to provide best static mapping
- Charts and GUI representation of results
- Takes as input a dot file or a json file (the output of previous tools)



Time Analysis Framework

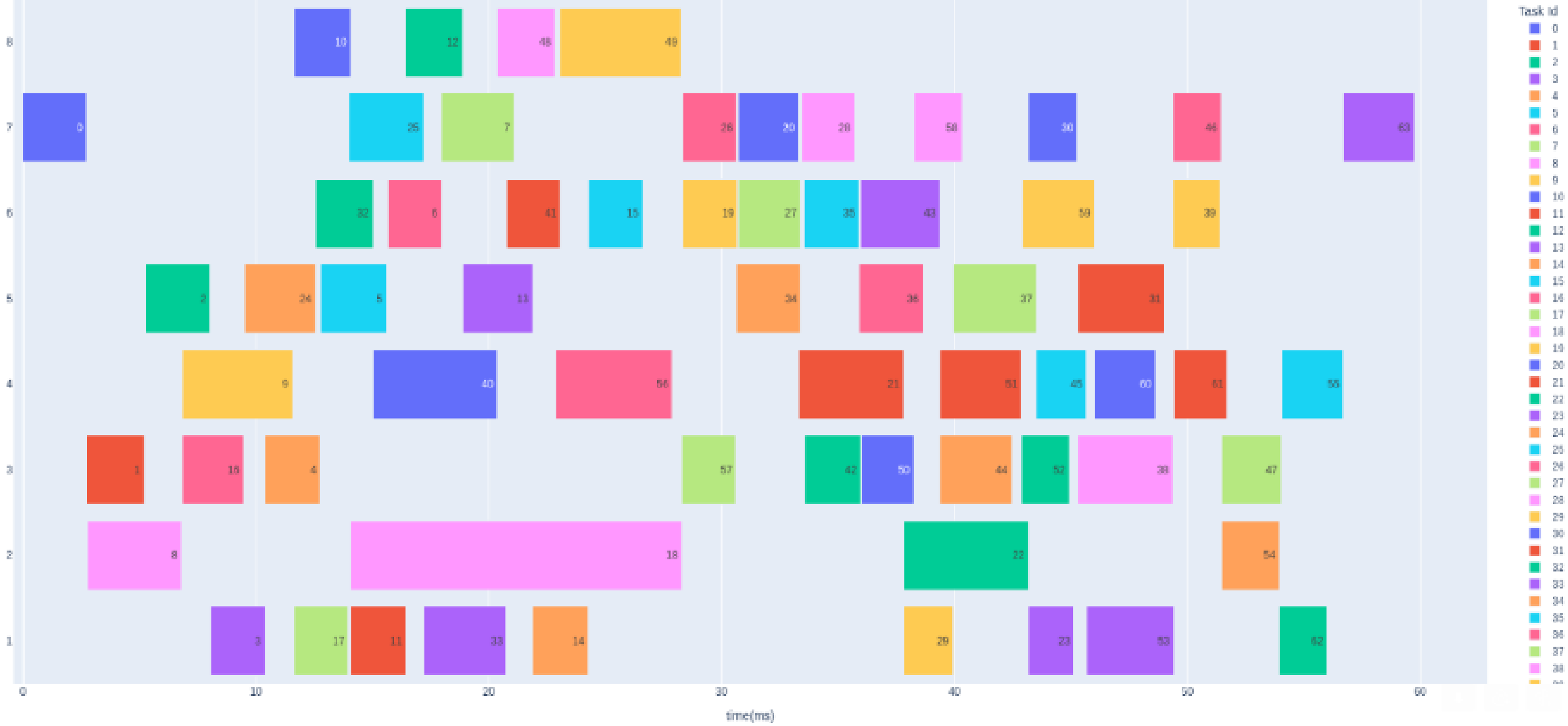
- Tools (run without arguments to see usage):
  - **dot2json**: converts dot file into json
  - **results-map-view**: gui representation of a “result”
  - **time-analysis**: calculates metrics from the “raw” results in the json file
  - **map-simulator**: task-to-thread mapping based on a mapping algorithm
  - **map-exploration**: explore several mapping algorithms to provide best mapping
  - **static-map-view**: gui representation of static task-to-thread mapping
  - **json2cpp**: converts json file into a tdg.cpp file



results-map-view

```
usage: <input.json> <range as: init[:fini[:step]]>
```

- Simple web page with a graphic representation of the task-to-thread mapping of the requested result(s).



time-analysis

- Calculates metrics regarding the OpenMP tasks and the overall TDG

```
time-analysis test_output.json test_analysed.json
```

- This will create a json file containing the metrics
  - Still retains the “results” key
  - Run with“-c” option to remove the “raw” results

```
{
  "test": [
    {
      "taskgraph_id": 2658744759,
      "nodes": {
        "0": {
          "ins": [], "outs": [],
per node  "metrics": {
            "wcet": 111680,
            "avg_time": 104356.5,
            "counters": {
              "42000000": {
                "avg": 451,
                "max": 475,
                "min": 427,
                ...}
            }}},
TDG   "metrics": {
        "volume": 200327,
        "critical_path_length": 111680,
        "max_parallelism": 2,
        ...
      }
    }...
```

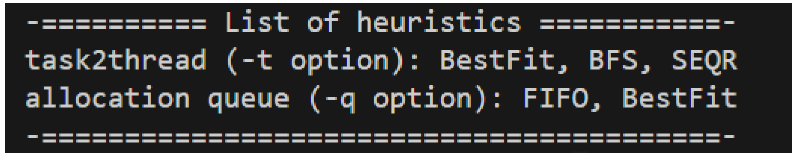
Time-Analysis Metrics

Key	Name	Description	Level	Active
wcet	Worst case execution time	Max execution time observed	Task	X
pmc	Performance counter metrics	Metrics related to each pmc read	Task	X
volume	Volume	Total volume of the TDG	TDG	X
cpl	Critical path length	Cost of the critical path	TDG	X
max_par	Maximum parallelism	Maximum possible level of parallelism	TDG	X
depth	Depth	Maximum depth of the TDG	TDG	X
makespan	Makespan	Execution time from the source task to the sink task. Requires all task to be annotated with ´static_thread´	TDG	
wcrt	Worst case response time	Calculates an upper bound of the excepted worst execution time of the TDG. It presents the results for tied and untied tasks.	TDG	X

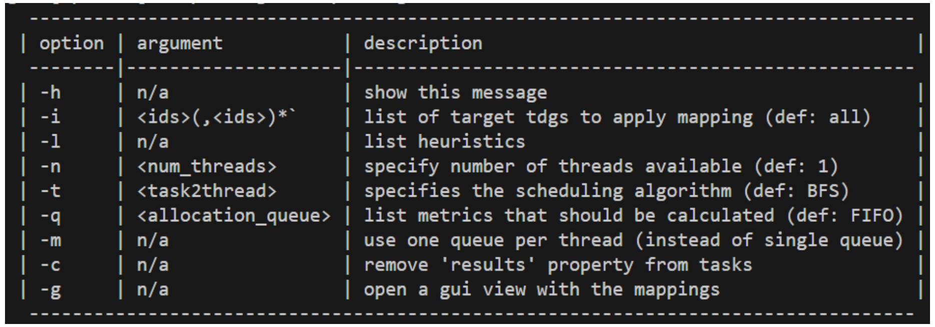
map-simulator

```
usage: [options] <input tdg> <output tdg>
```

- Use one of the existing heuristic-based task-to-thread mapping algorithm to provide a static thread mapping for each task
- Uses worst-case execution time per task!

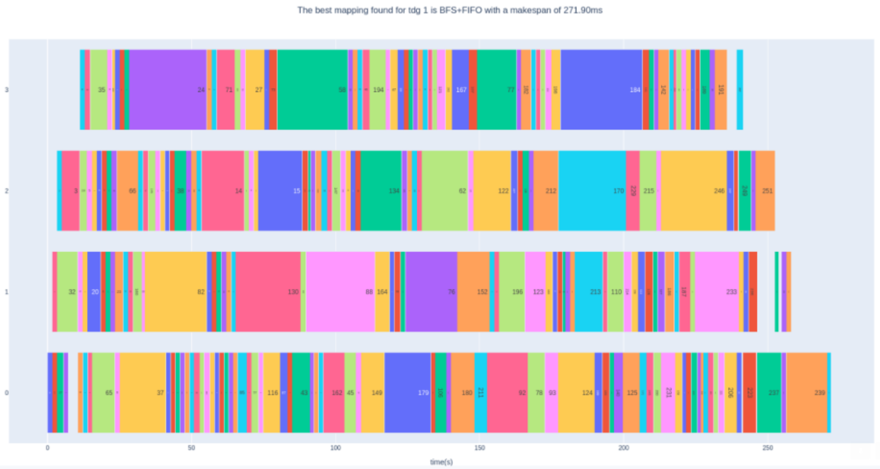


```
-========== List of heuristics ===========-
task2thread (-t option): BestFit, BFS, SEQR
allocation queue (-q option): FIFO, BestFit
-==========================================-
```



```
 ------------------------------------------------------------------------------------------
| option | argument            | description                                              |
 --------|---------------------|----------------------------------------------------------
| -h     | n/a                 | show this message                                        |
| -i     | <ids>(,<ids>)*`     | list of target tdgs to apply mapping (def: all)          |
| -l     | n/a                 | list heuristics                                          |
| -n     | <num_threads>       | specify number of threads available (def: 1)             |
| -t     | <task2thread>       | specifies the scheduling algorithm (def: BFS)            |
| -q     | <allocation_queue>  | list metrics that should be calculated (def: FIFO)       |
| -m     | n/a                 | use one queue per thread (instead of single queue)       |
| -c     | n/a                 | remove 'results' property from tasks                     |
| -g     | n/a                 | open a gui view with the mappings                        |
 ------------------------------------------------------------------------------------------
```

- BestFit and SEQR require the "-m" option (means multiple queues)



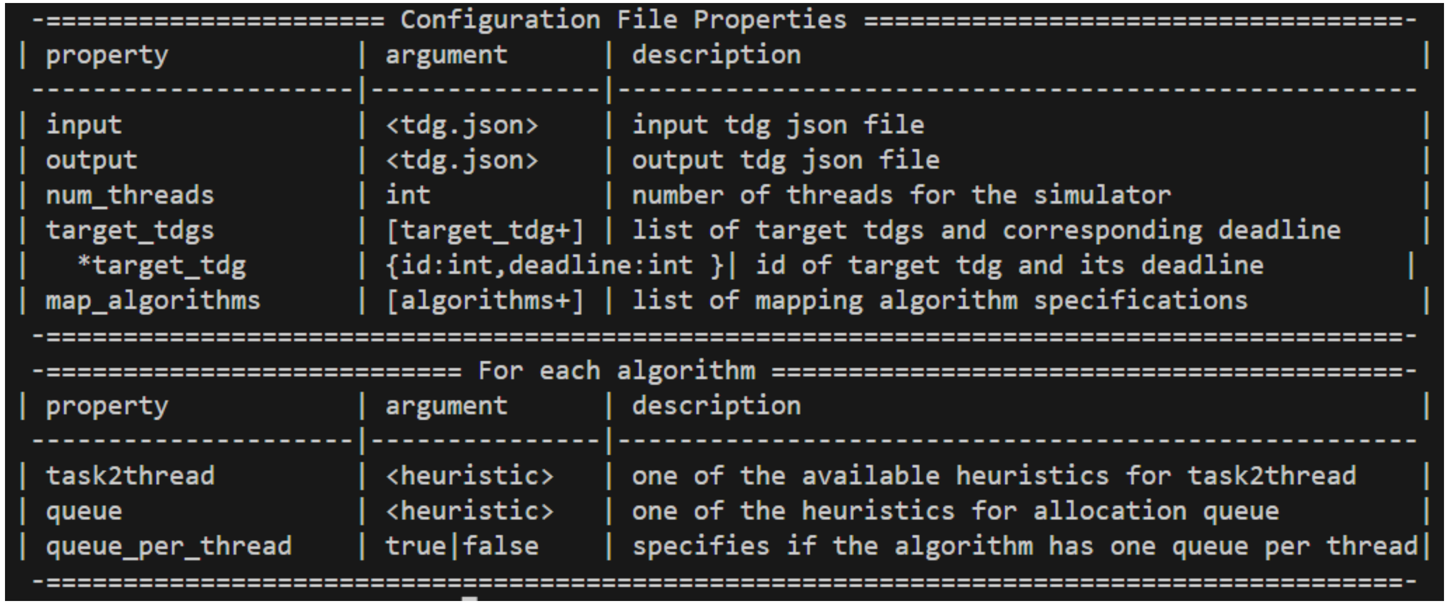
The best mapping found for tdg 1 is BFS+FIFO with a makespan of 271.90ms

use the –g option to see the mapping

map-exploration

```
usage: [options] <config.json>
```

- Same as before but multiple algorithms are used for the exploration
- Will select the static mapping with best “makespan”
- Requires a configuration file using the following structure



```
 -====================== Configuration File Properties ===================================-
| property             | argument      | description                                        |
 ---------------------|---------------|----------------------------------------------------
| input                | <tdg.json>    | input tdg json file                                |
| output               | <tdg.json>    | output tdg json file                               |
| num_threads          | int           | number of threads for the simulator                |
| target_tdgs          | [target_tdg+] | list of target tdgs and corresponding deadline     |
|   *target_tdg        | {id:int,deadline:int }| id of target tdg and its deadline         |
| map_algorithms       | [algorithms+] | list of mapping algorithm specifications           |
 -=======================================================================================-
 -=========================== For each algorithm =========================================-
| property             | argument      | description                                        |
 ---------------------|---------------|----------------------------------------------------
| task2thread          | <heuristic>   | one of the available heuristics for task2thread    |
| queue                | <heuristic>   | one of the heuristics for allocation queue         |
| queue_per_thread     | true|false    | specifies if the algorithm has one queue per thread|
 -=======================================================================================-
```

map-exploration configuration

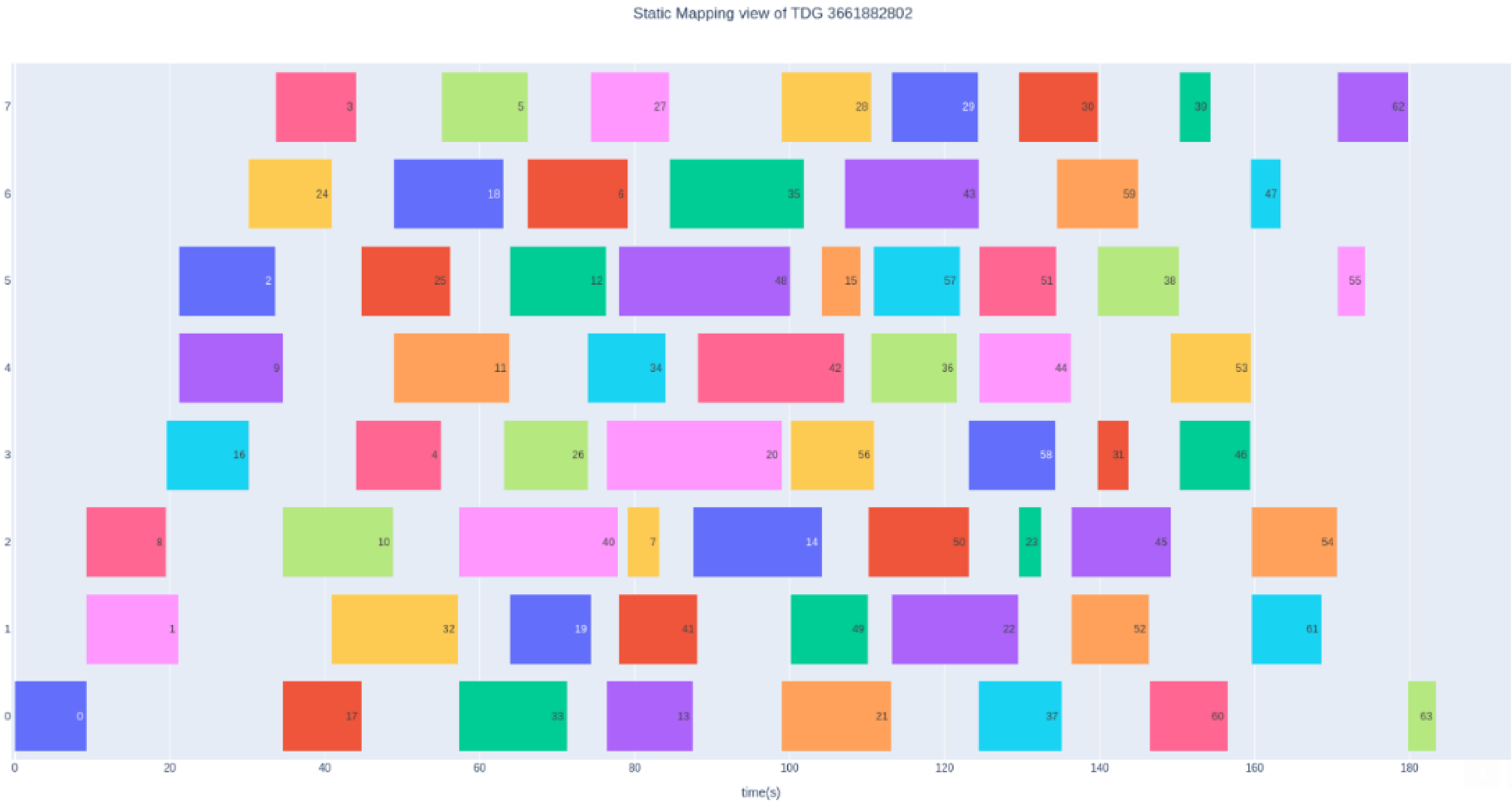
- This configuration explores the currently available algorithms

```
{
    "input": “path/to/tdg.json",
    "output": “path/of/output.json",
    "num_threads": 4,
    "map_algorithms":[
        {
            "task2thread": "BestFit",
            "queue": "BestFit",
            "queue_per_thread": true
        },
        {
            "task2thread": "BestFit",
            "queue": "FIFO",
            "queue_per_thread": true
        },
        {
            "task2thread": "BFS",
            "queue": "FIFO",
            "queue_per_thread": false
        },
        {
            "task2thread": "SEQR",
            "queue": "FIFO",
            "queue_per_thread": true
        }
    ]
}
```

static-map-view

```
usage: <input tdg>
```

- Previous tools output a json with static mapping (per node)
- This can be viewed either in those tools or with *static-map-view*

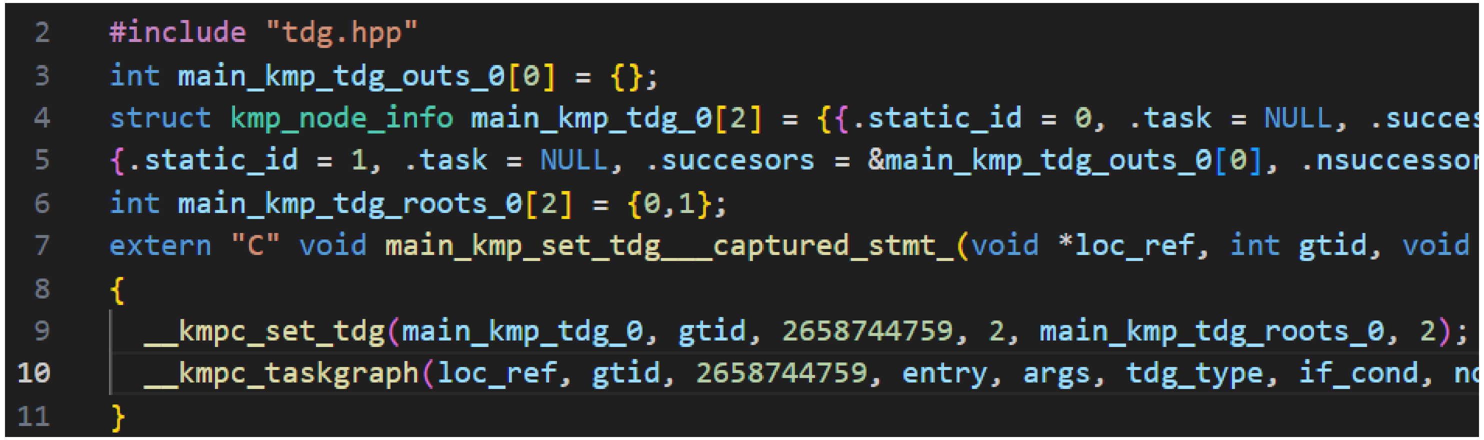


json2cpp

```
usage: [options] <input.json> [<output.cpp>]
```

- Convert the json file into a .cpp file
- Important to apply the static mapping in the application
- “-n <name>” option is important to specify the name of the file in which the “parallel region” resides
  - Example if in main.c file

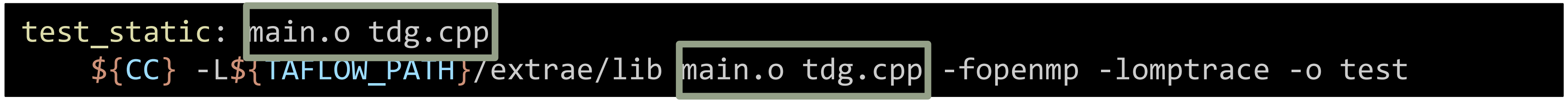
```
json2cpp -n main tdg_with_mapping.json
```



```
#include "tdg.hpp"
int main_kmp_tdg_outs_0[0] = {};
struct kmp_node_info main_kmp_tdg_0[2] = {{.static_id = 0, .task = NULL, .succes
{.static_id = 1, .task = NULL, .succesors = &main_kmp_tdg_outs_0[0], .nsuccessor
int main_kmp_tdg_roots_0[2] = {0,1};
extern "C" void main_kmp_set_tdg___captured_stmt_(void *loc_ref, int gtid, void
{
  __kmpc_set_tdg(main_kmp_tdg_0, gtid, 2658744759, 2, main_kmp_tdg_roots_0, 2);
  __kmpc_taskgraph(loc_ref, gtid, 2658744759, entry, args, tdg_type, if_cond, nc
}
```

Applying static mapping

- Uses the output of json2cpp and
- Requires the .o files from previous compilation
- Let’s add one new line in Makefile



```
test_static: main.o tdg.cpp
    ${CC} -L${TAFLOW_PATH}/extrae/lib main.o tdg.cpp -fopenmp -lomptrace -o test
```

- Now the program can be executed in both ways (static or dynamic)
- For static mapping prepend the program: OMP_TASK_SCHEDULE=static
- E.g. with static mapping:
- E.g. with dynamic mapping:

OMP_TASK_SCHEDULE=static ./test
./test

Analysis and Optimization Flow Resume

1. Compile with **extrae** lib and **taskgraph** directive (dynamic mapping)
2. **dot2json** to get json file (or **parsePrvAndTdg.py** script)
3. Run **application** several times
   - In each iteration run **parsePrvAndTdg** script to append results to json file
4. **time-analysis** to obtain metrics
5. **map-exploration** to obtain best static mapping
6. **json2cpp** to obtain tdg.cpp file
7. **Compile** with .o files and tdg.cpp
8. Run **application** with static mapping
   - Create **new json** file to store new results
9. Compare with previous version

Some Takeaways

- Performance depends on many factors
  - Code Design
  - Compilation flags/optimization
  - Target Platform
  - System configuration/workload
  - …
- Static task-to-thread mapping does not necessarily mean best performance
  - It might be slightly worse
  - If within requirements/restrictions, it is acceptable
  - More importantly, execution become more predictable
    - Relevant for real-time systems

Thank You!

Luis Miguel Pinho & **Tiago Carvalho**

CERCIRAS Training School 2023: Advanced Topics in Resource-Aware Computing

isep Instituto Superior de **Engenharia** do Porto