Missions

The project aims in having an automatic task-based parallelization system for C++ code-based applications using the sequential task-flow (STF) model. In this model, a sequential code (i.e. a code that is originally not designed to run in parallel) is annotated to decide what sections to transform into tasks and how these ones access the data. From this description, a runtime system can build a graph of tasks and execute them in parallel, while respecting the data dependencies.

The transformation from an STF code to a graph of tasks is done by a runtime system. The current project will focus on the transformation from a sequential code to an STF code. To do so, a source-to-source transformation of sequential C++ source code will insert tasks and data accesses. The first step will be to consider one function/method as one task and will manage the data accesses at “object” level (one object == one data handle). Then, the research work will focus on fighting the successive pitfalls that will arise and increase the performance/degree of parallelism.

Principales activités

We provide here a non-exhaustive list of possible key challenges:

- How to manage the granularity to enable/disable tasking at runtime
- How/when to split a function/loop into tasks
- How to detect patterns for commutative data accesses to increase the degree of parallelism, such as in [2], or to enable speculative execution, as done manually in [3]
- How to remove/delay/regexp group synchronizations
- How to schedule an automatically parallelized application, including on heterogeneous architectures using our own LAHeteroprio scheduler [4]
- How to move to automatic distributed parallelizing using a common task declaration strategies, as the one used in [5]

The significant results from past and on-going researches on automatic parallelization will give few hints and will help to parallelize computational loops, see [47] for example. The validation of the proposed solutions will be done on large open-source C++ projects that are currently sequential.

Technology

The source-to-source transformation system will be based on clang-LVLM. It will be developed on a modern GitLab and proposed to the community with an open-source license. We will rely on our in-house runtime system SPETABARU to manage and execute the tasks.

References

Compétences
- C++ (17)
- Knowledge of compilation would be really appreciated
- Knowledge of parallel programming/computing is an asset
- Proactive, high interest in solving problems, interest in learning clang-LLVM

Avantages
- Subsidized meals
- Partial reimbursement of public transport costs
- Leave: 7 weeks of annual leave + 10 extra days off due to RTT (statutory reduction in working hours) + possibility of exceptional leave (sick children, moving home, etc.)
- Possibility of teleworking (after 6 months of employment) and flexible organization of working hours
- Professional equipment available (videoconferencing, loan of computer equipment, etc.)
- Access to vocational training
- Social, cultural and sports events and activities
- Social security coverage

Rémunération

Monthly salary after taxes: around 1596.05€ for 1st and 2nd year. 1678.99€ for 3rd year. (medical insurance included).