Thesis Proposal

Title: Generating Executables from Excel spreadsheets

Scope: The Fixed Income Research and Strategy Team (FIRST), at BNP Paribas CIB, develops and supports systems for the quantitative analysis of market and trade data, to price fixed income assets and manage risk. Many of our systems rely on heavy computations and considerable amounts of data. Some of them are not implemented in the most reliable or efficient way. In particular, we have Excel *spreadsheets* that aggregate market data and trade parameters, to (but not exclusively) calculate relevant measures of risk. These *spreadsheets* run at specific times of the day and in a specific order because the results from one spreadsheet might serve as input for others. This setting is rather cumbersome since the volume of data processed along with the resource-intensiveness of certain computations make spreadsheet-runs fail, almost on a daily-basis. This inevitably delays our production pipeline.

We would like to be able to generate efficient and reliable executables from spreadsheets. This way, we could still visualize data in a graphical and organized manner, without sacrificing the performance and reliability of the jobs materialized by those spreadsheets.

Objectives: Develop a tool capable of:

1. Converting an Excel *spreadsheet* into a Dependency Graph. This Dependency Graph will represent the dependencies of data transformations done in the spreadsheet.
2. Generating executables given the generated Dependency Graph. The code generated should be ready to run in a multithreaded environment.

Description:

1. Survey and compare state-of-the-art Compilation Techniques and Parallel Computing paradigms. If possible, identify existing solutions that “compile” unconventional artefacts and comment on their pros/cons. Compare the implemented tools with existing solutions.
2. Gather Performance and Reliability metrics relevant to the scope of the project.
3. Decide on the System Architecture and Compilation Strategy to use (Static, Dynamic – JIT, etc), supported by the evidence and reasoning done in the related work section.
4. Implement the Dependency Graph and code generation tool.
5. Measure performance and reliability of executables against their spreadsheet counterparts. Comment on the results.
6. Highlight the merits/drawbacks of the work done – focus on the approaches and why certain choices where made over others.

Expected Results:

1. A tool that takes an Excel *spreadsheet* and generates a Dependency Graph.
2. A tool that takes a Dependency Graph and generates an executable.

Observations:

There are no requirements for the tools’ interfaces. They can be controlled via command line.
The generation of the Dependency Graph will be fully supported by the team. We have components that can be extended to output such a graph.
The Dependency Graph should have a representation sufficiently abstract (for instance based on Gantt charts), such that the same representation can be used to represent any artefact other than spreadsheets.
The code generation component should be versatile enough to be extendable. For instance, if in the future we decide the code generated should also be ran in a distributed setting instead of a single multithreaded machine. So, in general, special emphasis should be put onto the system’s design.
Finally, there are no restrictions regarding technology choices. However, the tool to deliver should be as close to production-ready, as possible. Ideally it should be developed using fully supported languages and frameworks.

goncalo.avelar@bnpparibas.com