ReCoTOS: a Platform for Resource-Sparing Computing Task Optimization

Jānis Kampars, Guntis Mosāns, Jānis Zuters, Aldis Gulbis and Rasa Gulbe

Institute of Information Technology, Riga Technical University, 10 Zunda krastmala, Riga, Latvia
Faculty of Computing, University of Latvia, 19 Raiņa bulvāris, Riga, Latvia
Dati Group, 80 A Balasta dambis, Riga, Latvia

Background

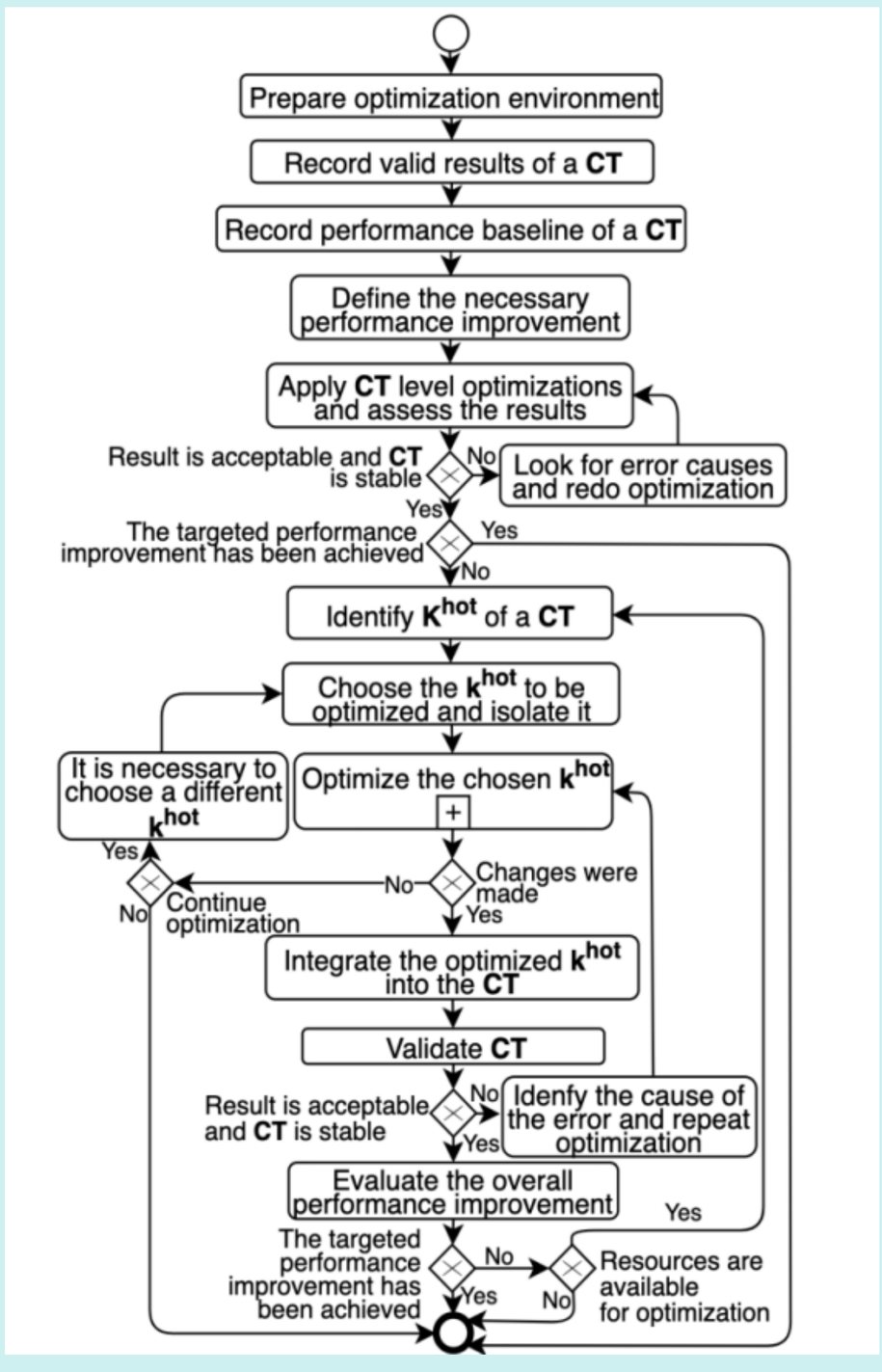
- The data volume and complexity of computing tasks are constantly increasing, requiring more power, resources, and energy.
- One of the possible solutions is the horizontal and vertical scaling of the computing infrastructure, however, the approach is associated with several shortcomings.
- Another approach to solving the problem of computing capacity deficiency is code refactoring or optimization activities allowing developers to get better performance with less resources.
- In order to address the challenges associated with computing task optimization, we have previously proposed a methodology and demonstrated its usage.
- The goal of this paper is to define a cloud-based platform that facilitates the optimization of computing tasks according to the aforementioned methodology.

Optimization approaches

- Assembly Level Programming - optimization are implemented in the assembly language in the form of modules.
- Intrinsic Programming Model - relies on assembly-type instructions, which are added in the source code of a high-level programming language.
- Pragma Syntax - allows one to insert instructions for the compiler in the source code of a high-level programming language.
- Compiler Automatic Vectorization - provides indirect code vectorization, without the need to make any changes in the code.

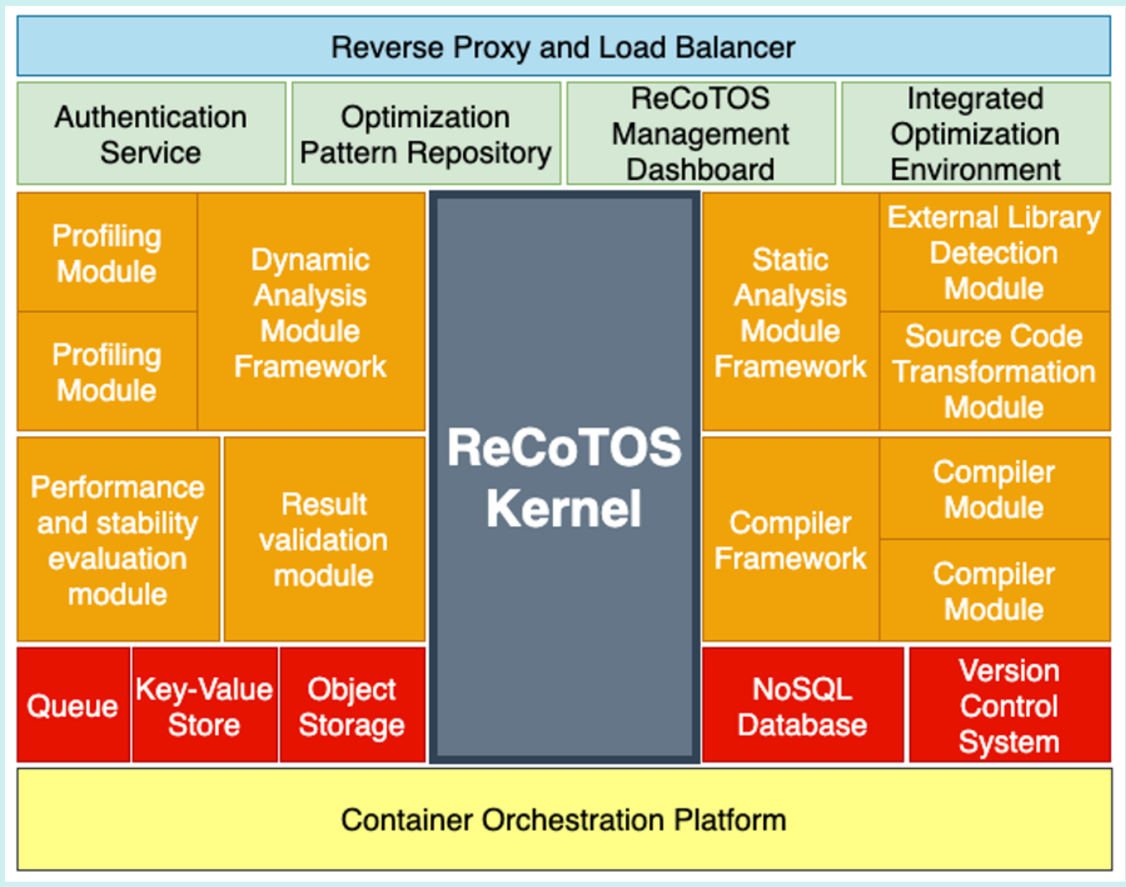
Optimization methodology

- Typical development environments and code management systems do not provide support to evaluate and to select the most suitable approach and lack methodological support. To address this a platform is developed on the basis of a methodology for vectorization-based resource-saving computing task optimization.



Design of ReCoTOS platform

- ReCoTOS platform components can be divided into three parts. The infrastructure service layer (red), the backend components used to optimize the code (orange) and the user interface components (green). The kernel orchestrates all components on the Kubernetes, but access to the components is organized through a proxy.



Use case

- The functionality of the platform is demonstrated by optimizing C based software correlator for earth observation data processing (KANA), which was developed by Ventspils International Radio Astronomy Centre of the Ventspils University of Applied Sciences Institute of Engineering (IE VIRAC) in 2012 as part of the project “Earth’s near-field radio-astronomical research”.
- Experiments are performed in the CloudStack (Train release) cloud computing platform. The cloud environment is equipped with Intel(R) Xeon(R) Gold 5218R CPU processors.
- The tuning of compiler version and its flags have allowed to reduce the execution time by approximately 58 seconds or by 24.8%.
- According to the optimization methodology the first optimization step was switching from FFTW to the Intel MKL library. Using new library the execution time was reduced by approximately 82 seconds or 35.0% from the baseline originally measured.
- The optimization ends with gaining execution time reduction by approximately 140 seconds or 59.9%.

Conclusion and future work

- The ReCoTOS platform has been designed, implemented, and its functionality has been demonstrated using optimization of a C-based software correlator.
- Initially, it targets C and C++ programming languages, and the applied optimization methodology is mostly concerned with improving the vectorization of the computing tasks.
- Our experience while developing the methodology and the platform shows that software optimization is a complicated process which would benefit from availability of a supporting methodology, cloud-based platform and best practices expressed as patterns.
- We have also formalized 7 patterns in the pattern repository with the measured execution time reduction as follows:
  - 2D array processing pattern (structural changes in the code) - 47%.
  - External library usage pattern (external library replacement) - 43%.
  - Array sort pattern variant #1 (structural changes in the code) - 66%.
  - Array sort pattern variant #2 (structural changes in the code) - 78%.
  - Matrix multiplication pattern #1 (structural changes in the code) - 78%.
  - Matrix multiplication pattern #2 (structural changes in code) - 91%.
  - Square matrix transposition pattern (structural changes in code) - 28%.

The research leading to these results has received funding from the project "Competence Centre of Information and Communication Technologies" of EU Structural funds, contract No. 1.2.1.1/18/A/003 signed between IT Competence Centre and Central Finance and Contracting Agency, Research No. 1.13" Resource-saving Computing Task Optimization Solutions"