MATLAB Interface To The *afft* Library

David Bayer[1] and Jiri Jaros[1]
[1]Faculty of Information Technology, Brno University of Technology,
Centre of Excellence IT4Innovations, CZ

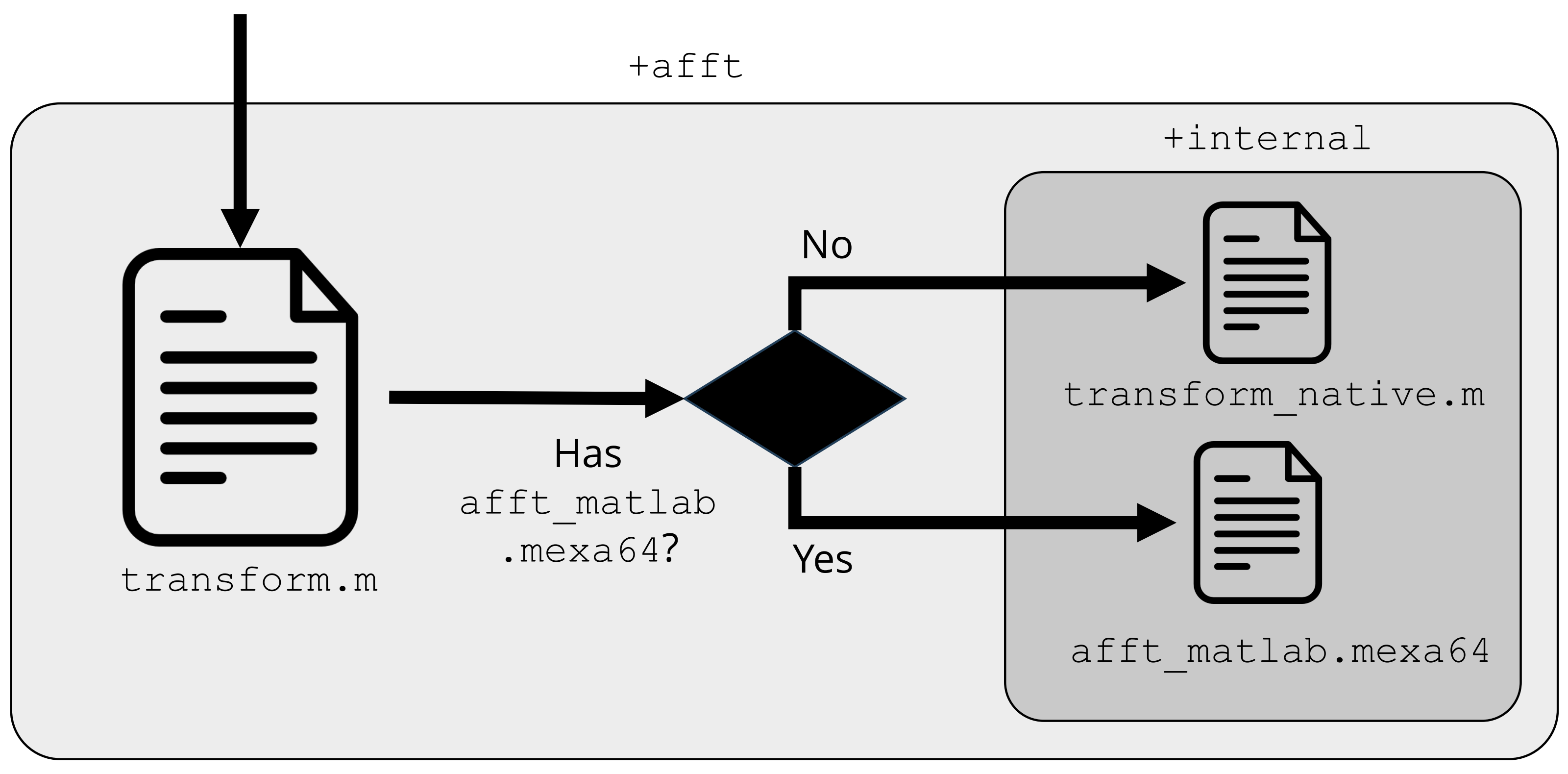
1 Introduction and Motivation

Fast Fourier Transformation (FFT) and other related transformations are very demanding and time-consuming computations often used by scientists in various areas of research. In many cases, the MATLAB software is used to run the computations, however, MATLAB's support for these transformations is quite limited. This poster presents the MATLAB interface to the *afft* library that wraps a lot of existing FFT implementations on CPUs and GPUs, extending the support of FFT-like computations in MATLAB, while improving their performance.

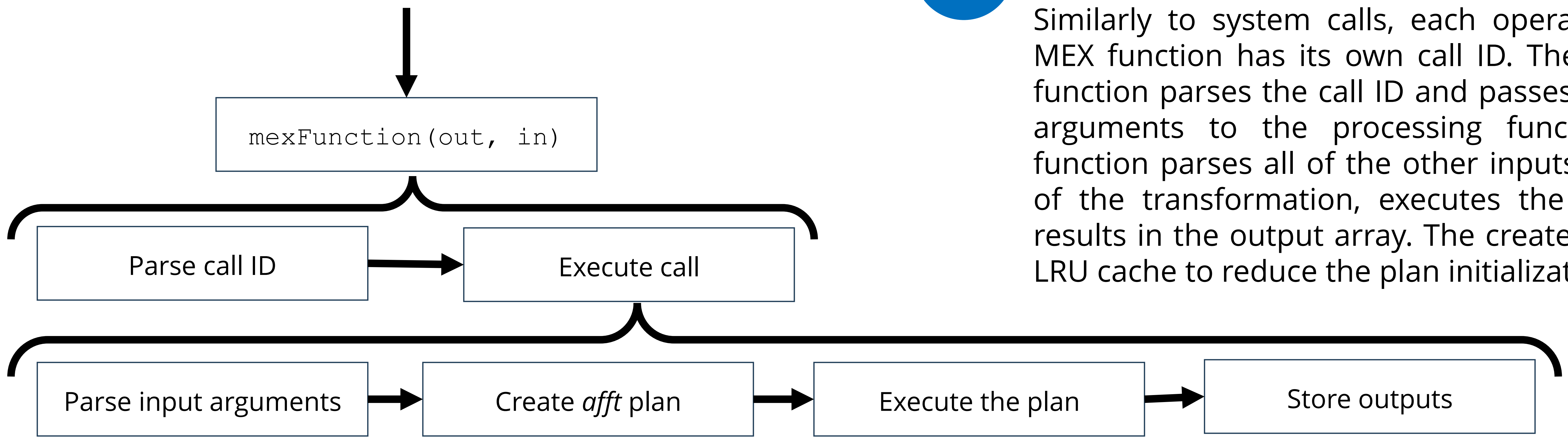
2 How does it work?

The *afft* toolbox for MATLAB implements transformations via functions analogous to the MATLAB's native functions with extended functionality. Initially, when the toolbox is installed to the system, the functions are implemented in pure MATLAB, allowing the use right out of the box. The main feature is hidden in a separately compiled MEX function that implements the API calls in C/C++ using the *afft* library. This way the computation may be executed on CPUs and NVIDIA GPUs via the cuFFT, FFTW3, Intel MKL, PocketFFT or VkFFT libraries.

```
>> Y = afft.transform(X);
```



```
>> out = afft_matlab(callId, varargin{:});
```



3 The MEX function in detail

Similarly to system calls, each operation executed by the MEX function has its own call ID. The gateway of the MEX function parses the call ID and passes the input and output arguments to the processing function. The processing function parses all of the other inputs, creates the *afft* plan of the transformation, executes the plan and stores the results in the output array. The created plans are put into a LRU cache to reduce the plan initialization time.

4 Supported functionality

All FFT-like transformations in MATLAB are implemented as `transform1`, `transform2` or `transformn` functions which compute the *transform* in 1, 2 or N dimensions respectively. The *afft* toolbox implements this API to allow easy transition for already existing projects and extends its capabilities of new parameters such as normalization, strategy of the backend library selection and others. Currently the supported transformations are:

- *Discrete Fourier Transformation* (DFT),
- *Discrete Cosine Transformation* (DCT),
- *Discrete Sine Transformation* (DST),
- *Discrete Trigonometric Transformation* (DTT) and
- *Discrete Hartley Transformation* (DHT).

	MATLAB			afft toolbox		
	1D	2D	ND	1D	2D	ND
DFT	x	x	x	x	x	x
DCT	x	x	x	x	x	x
DST	x			x	x	x
DTT				x	x	x
DHT				x	x	x

The library also provides a `Plan` class that can be configured to run any of the transformations with slightly lower overhead compared to the function style execution.

5 Conclusions

The *afft* toolbox is an alternative way to compute FFT-like transformations in MATLAB. In comparison to the native implementations, it offers improved user's experience, better performance and extra features. Due to the use of the *afft* library, it is a flexible and powerful tool that can support any target architecture and transformation libraries. It is designed to be easily extensible of new targets and transformations.

6 Current and Future Work

Next steps in the development are to

- finalize the implementation including unit and module testing,
- deploy the toolbox to real world applications, e. g. *k-Wave* project, and
- fine tune the transformation libraries.

This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic through the e-INFRA CZ (ID:90254). This project has received funding from the European Unions Horizon Europe research and innovation programme under grant agreement No 101071008.}