MAX PLANCK INSTITUTE FOR DYNAMICS OF COMPLEX TECHNICAL SYSTEMS MAGDEBURG

COMPUTATIONAL METHODS IN SYSTEMS AND CONTROL THEORY

GPL Free Software — Free as in Freedom

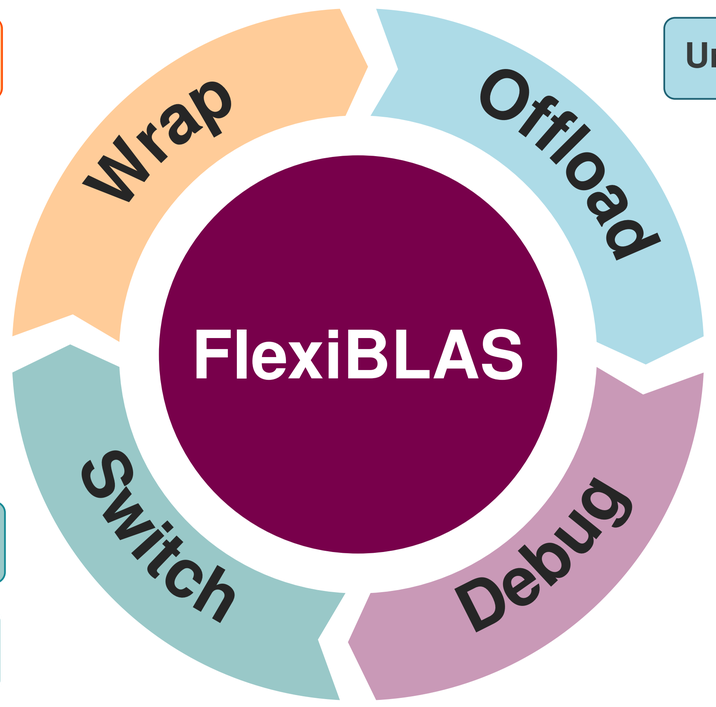
Profiling and Inspecting Linear Algebra Applications using FlexiBLAS

Christian Himpe, **Martin Köhler**, Jörn Papenbroock, Jens Saak

Max Planck Institute for Dynamics of Complex Technical Systems, Computational Methods in Systems and Control Theory, Magdeburg

150 BLAS Routines · **1960 LAPACK Routines** · **Lightweight Wrappers** · **Automatic Fallback** · **OpenBLAS/MKL Extensions**

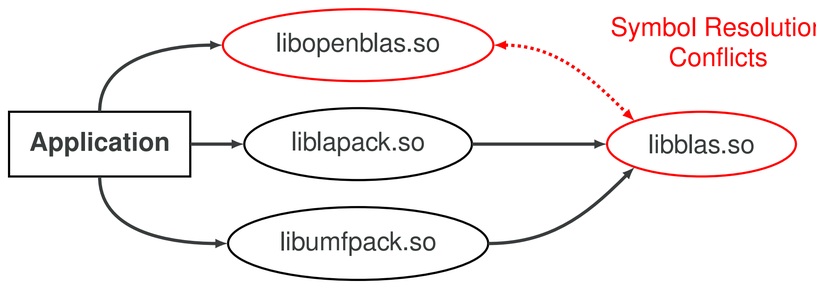
No Relinking · **No Environment Hacks** · **No Recompiling** · **System/Host/User Settings** · **No Superuser Privileges** · **Switch at Runtime**



Untouched Applications · **Drop-in replacements** · **MAGMA** · **PLASMA** · **OpenCL** · **CUDA** · **Alternative Implementations**

Replay and Compare Function Calls · **Profiling and Log BLAS/LAPACK calls** · **Performance Tuning** · **Trace Visualization**

Motivation

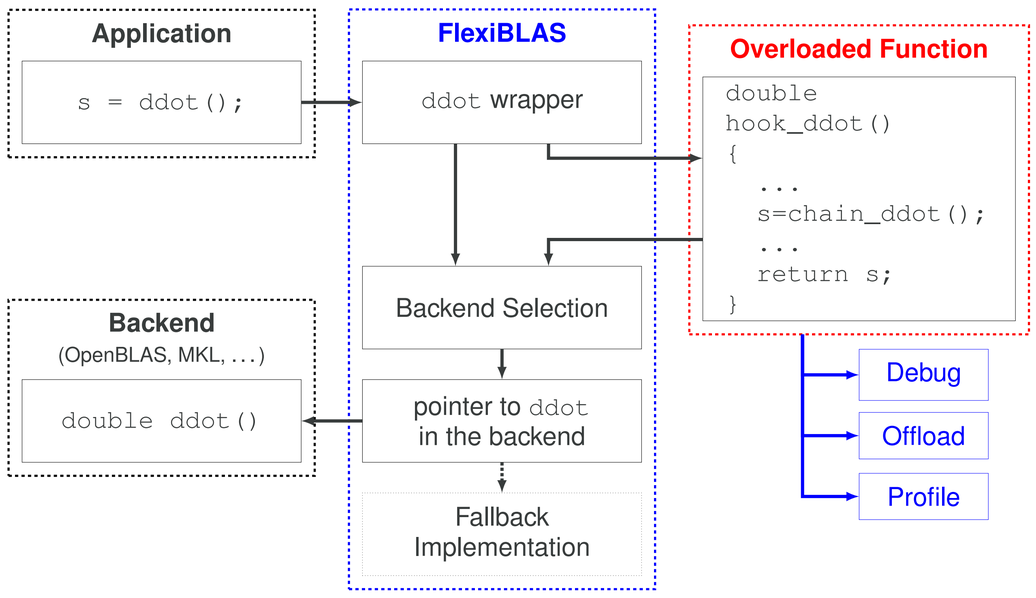


Internal dependencies and wrong linker calls may cause the integration of the same routine (symbols) from different libraries.

Other Problems:

- **Compatibility:** Intel/GNU interface style
- **Vendor extensions:** `AXBPY`, `OMATCOPY`, ...
- **Multi-library implementations:** MKL, ATLAS
- **Incomplete implementations:** IBM ESSL
- **Time consuming recompilation**
- **Superuser-only solutions:** `update-alternatives`, `eselect`, ...
- **Error-prone solutions:** `LD_PRELOAD`, `LD_LIBRARY_PATH`

Implementation



Our wrapper uses the POSIX `dlopen` mechanism to load the selected BLAS/LAPACK backend. Each symbol is resolved using `dlsym` when the application starts. These symbols are used to call the real BLAS function from the wrapper functions in our library. Multiple overloaded functions are arranged in chains to execute several of them for each function call.

Usage

Compile Time:

Link everything against FlexiBLAS instead of BLAS and LAPACK:

```
gcc ...  -lflexiblas
```

or use an operating system mechanism, like `update-alternatives` or `eselect`, to set it system-wide.

Runtime:

Everything is managed via the `flexiblas` tool:

- List all BLAS/LAPACK backends:
  `flexiblas list`
- Set the default backend:
  `flexiblas set BACKEND`
- Add a new BLAS/LAPACK library:
  `flexiblas add NAME blas.so`

Debug Data Acquisition and Analysis

Technical Details:

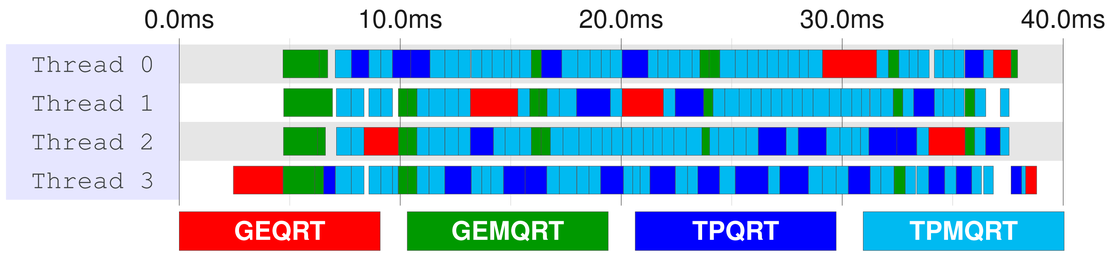
- Data acquisition through overloaded functions,
- Combination with offloading and other hooks
- SQLite-based data store
- No binary patching required
- Negligible overhead

Features:

- Python tools to analyze data
- Traces as TikZ/Latex/PDF graphic
- Only one program run for trace generation, profiling, and inspecting
- No commercial tools required

Trace Visualization, Profiling, and Inspecting

Example 1 – TileQR Implementation on top of OpenMP4



Example 2 – CG in GNU Octave

```
function [x] = conjgrad(A, b, x)
  r = b - A * x;
  p = r;
  rsold = r' * r;
  for i = 1:length(b)
    Ap = A * p;
    alpha = rsold / (p' * Ap);
    x = x + alpha * p;
    r = r - alpha * Ap;
    rsnew = r' * r;
    if sqrt(rsnew) < 1e-10
       break;
    end;
    p = r + (rsnew / rsold) * p;
    rsold = rsnew;
  end
end

A = full(sprandsym(1000,1.0));
b = A*ones(1000,1);
x = conjgrad(A,b,zeros(1000,1));
norm(A * x-b)/norm(b)
```

Profiling:

Subroutine	# Calls	acc. Time
dlamch	5	2.69e-05s
ddot	**1000**	**1.11e-03s**
dsyrk	**1001**	**5.61e-03s**
dgemv	**1003**	**4.04e-01s**

Inspecting:

- `DAXPY` and `DNRM2` not used at all
- `DSYRK` computes

  $C := \alpha A \cdot A^T + \beta C$ if `trans='N'`

  or

  $C := \alpha A^T A + \beta C$, if `trans='T'`

  with $A \in \mathbb{R}^{n \times k}$ or $A \in \mathbb{R}^{k \times n}$ and $C \in \mathbb{R}^{n \times n}$.

All 1001 `DSYRK` calls use `trans = 'T'`, $n = 1$, $k = 1000$, $\alpha = 1.0$, and $\beta = 0$.

Misuse of `DSYRK` in Octave!
$\Longrightarrow$ `ddot` would be correct.

Resources

- **Web:** `https://www.mpi-magdeburg.mpg.de/projects/flexiblas/`
- **Contact:** `koehlerm@mpi-magdeburg.mpg.de`
- Martin Köhler and Jens Saak, FlexiBLAS 2.0 – A runtime BLAS switch, 2017, `https://doi.org/10.5281/zenodo.569102`
- Martin Köhler and Jens Saak, FlexiBLAS - A flexible BLAS library with runtime exchangeable backends, LAPACK Working Note #284, 2014, `http://www.netlib.org/lapack/lawnspdf/lawn284.pdf`