



January 10, 2014

FlexiBLAS

Switching BLAS made easy

Martin Köhler and Jens Saak

Chemnitz–Magdeburg–Kooperationsseminar

Outline



- 1 What is this BLAS-thing anyway?
- 2 Why do we need FlexiBLAS?
- 3 How does it work?
- 4 How can I use it then?

What is this BLAS-thing anyway?

What is this BLAS-thing anyway?



Basic Linear Algebra Subprograms (BLAS)

“The BLAS (Basic Linear Algebra Subprograms) are routines that provide standard building blocks for performing basic vector and matrix operations. . . . Because the BLAS are efficient, portable, and widely available, they are commonly used in the development of high quality linear algebra software, LAPACK for example.”^a

^aFrom: <http://www.netlib.org/blas/faq.html> – What and where are the BLAS?

What is this BLAS-thing anyway?



BLAS routine organization

Let α be scalar, x, y be vectors, A, B, C be matrices of appropriate dimensions.



What is this BLAS-thing anyway?

BLAS routine organization

Let α be scalar, x, y be vectors, A, B, C be matrices of appropriate dimensions.

level	included operations	data	flops
1	$\alpha x, \alpha x + y, x^* y, \ x\ _2, \ x\ _1, \ x\ _\infty$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
2	$\alpha Ax + \beta y, \alpha A^* x + \beta y, A + \alpha xy^*, A + \alpha xx^*, A + \alpha xy^* + \beta yx^*$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
3	$\alpha AB + \beta C, \alpha AB^* + \beta C, \alpha A^* B^* + \beta C, \alpha AA^* + \beta C, \alpha A^* A + \beta C$ rank k updates $\alpha A^* B + \beta C, \alpha B^* A + \beta C$ rank $2k$ updates	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$



What is this BLAS-thing anyway?

BLAS routine organization

Let α be scalar, x, y be vectors, A, B, C be matrices of appropriate dimensions.

level	included operations	data	flops
1	$\alpha x, \alpha x + y, x^*y, \ x\ _2, \ x\ _1, \ x\ _\infty$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
2	$\alpha Ax + \beta y, \alpha A^*x + \beta y, A + \alpha xy^*, A + \alpha xx^*, A + \alpha xy^* + \beta yx^*$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
3	$\alpha AB + \beta C, \alpha AB^* + \beta C, \alpha A^*B^* + \beta C, \alpha AA^* + \beta C, \alpha A^*A + \beta C$ rank k updates $\alpha A^*B + \beta C, \alpha B^*A + \beta C$ rank $2k$ updates	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$

Level 3 BLAS especially attractive for communication avoidance and parallelism.

What is this BLAS-thing anyway?



Some important BLAS implementations

Open Source

- NetLib BLAS: <http://www.netlib.org/blas>
(The Standard)

What is this BLAS-thing anyway?



Some important BLAS implementations

Open Source

- NetLib BLAS: <http://www.netlib.org/blas>
(The Standard)
- OpenBLAS: <http://www.openblas.net/>
(uses assembler level optimization and threading)



What is this BLAS-thing anyway?

Some important BLAS implementations

Open Source

- NetLib BLAS: <http://www.netlib.org/blas>
(The Standard)
- OpenBLAS: <http://www.openblas.net/>
(uses assembler level optimization and threading)
- Automatically Tuned Linear Algebra Software (ATLAS):
<http://math-atlas.sourceforge.net/>
(provides automatic tuning for specific processors and threading)

What is this BLAS-thing anyway?



Some important BLAS implementations

Open Source

- NetLib BLAS: <http://www.netlib.org/blas>
(The Standard)
- OpenBLAS: <http://www.openblas.net/>
(uses assembler level optimization and threading)
- Automatically Tuned Linear Algebra Software (ATLAS):
<http://math-atlas.sourceforge.net/>
(provides automatic tuning for specific processors and threading)

Hardware Vendor Implementations

What is this BLAS-thing anyway?



Some important BLAS implementations

Open Source

- NetLib BLAS: <http://www.netlib.org/blas>
(The Standard)
- OpenBLAS: <http://www.openblas.net/>
(uses assembler level optimization and threading)
- Automatically Tuned Linear Algebra Software (ATLAS):
<http://math-atlas.sourceforge.net/>
(provides automatic tuning for specific processors and threading)

Hardware Vendor Implementations

- Intel[®] Math kernel library (MKL):
<http://software.intel.com/en-us/intel-mkl/>
(the fastest implementation on ccNUMA machines; provides hardware optimization and threading)



What is this BLAS-thing anyway?

Some important BLAS implementations

Open Source

- NetLib BLAS: <http://www.netlib.org/blas>
(The Standard)
- OpenBLAS: <http://www.openblas.net/>
(uses assembler level optimization and threading)
- Automatically Tuned Linear Algebra Software (ATLAS):
<http://math-atlas.sourceforge.net/>
(provides automatic tuning for specific processors and threading)

Hardware Vendor Implementations

- Intel[®] Math kernel library (MKL):
<http://software.intel.com/en-us/intel-mkl/>
(the fastest implementation on ccNUMA machines; provides hardware optimization and threading)
- AMD Core Math Library (ACML): <http://developer.amd.com/tools/cpu-development/amd-core-math-library-acml/>
(An ATLAS version tuned by AMD?)



What is this BLAS-thing anyway?

Some important BLAS implementations

Open Source

- NetLib BLAS: <http://www.netlib.org/blas>
(The Standard)
- OpenBLAS: <http://www.openblas.net/>
(uses assembler level optimization and threading)
- Automatically Tuned Linear Algebra Software (ATLAS):
<http://math-atlas.sourceforge.net/>
(provides automatic tuning for specific processors and threading)

Hardware Vendor Implementations

- Intel[®] Math kernel library (MKL):
<http://software.intel.com/en-us/intel-mkl/>
(the fastest implementation on ccNUMA machines; provides hardware optimization and threading)
- AMD Core Math Library (ACML): <http://developer.amd.com/tools/cpu-development/amd-core-math-library-acml/>
(An ATLAS version tuned by AMD?)
- Apple Accelerate: (the same from Apple ?)

Why do we need FlexiBLAS?

Why do we need FlexiBLAS?

Linker Problems

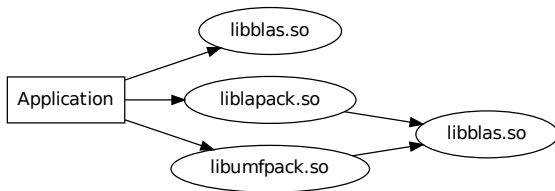


Figure: A sample application using BLAS

Why do we need FlexiBLAS?

Linker Problems

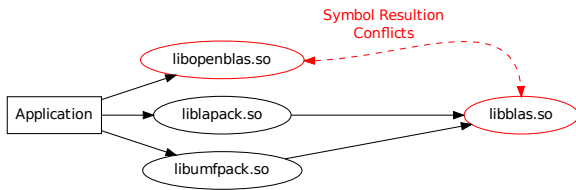


Figure: ...after linking with a different BLAS-implementation

Why do we need FlexiBLAS?



Linker Problems: Existing Solutions

- `LD_LIBRARY_PATH / LD_PRELOAD`
only applicable for single file implementations
(i.e. **NOT** Intel[®] MKL, or ATLAS)



Why do we need FlexiBLAS?

Linker Problems: Existing Solutions

- `LD_LIBRARY_PATH / LD_PRELOAD`
only applicable for single file implementations
(i.e. **NOT** Intel[®] MKL, or ATLAS)
- static libraries
drastically increased binary sizes, often complicated linking

Why do we need FlexiBLAS?



Linker Problems: Existing Solutions

- `LD_LIBRARY_PATH / LD_PRELOAD`
only applicable for single file implementations
(i.e. **NOT** Intel[®] MKL, or ATLAS)
- static libraries
drastically increased binary sizes, often complicated linking
- `update-alternatives`
requires super-user privileges and has similar restrictions as
`LD_LIBRARY_PATH / LD_PRELOAD`



Why do we need FlexiBLAS?

Linker Problems: Existing Solutions

- `LD_LIBRARY_PATH / LD_PRELOAD`
only applicable for single file implementations
(i.e. **NOT** Intel[®] MKL, or ATLAS)
- `static libraries`
drastically increased binary sizes, often complicated linking
- `update-alternatives`
requires super-user privileges and has similar restrictions as
`LD_LIBRARY_PATH / LD_PRELOAD`
- `eselect / pkg-config`
requires super-user privileges and switches at **build-time only**

Why do we need FlexiBLAS?

Compatibility Issues



`gfortran` vs `f2c/intel` interface style

- **different calling sequences:**
`f2c` and `intel` return complex numbers as additional function parameters.

Why do we need FlexiBLAS?

Compatibility Issues



`gfortran` vs `f2c/intel` interface style

- **different calling sequences:**
`f2c` and `intel` return complex numbers as additional function parameters.
- **affected routines:** `zdotc`, `zdotu`, `cdotc`, `cdotu` (level 1)

Why do we need FlexiBLAS?

Compatibility Issues



`gfortran` vs `f2c/intel` interface style

- **different calling sequences:**
`f2c` and `intel` return complex numbers as additional function parameters.
- **affected routines:** `zdotc`, `zdotu`, `cdotc`, `cdotu` (level 1)

auxiliary routine treatment

Routines `sc/dzabs1` are missing in ATLAS and derived implementations, such as Apple Accelerate / AMD ACML.

Why do we need FlexiBLAS?

Compatibility Issues



`gfortran` vs `f2c/intel` interface style

- **different calling sequences:**
`f2c` and `intel` return complex numbers as additional function parameters.
- **affected routines:** `zdotc`, `zdotu`, `cdotc`, `cdotu` (level 1)

auxiliary routine treatment

Routines `sc/dzabs1` are missing in ATLAS and derived implementations, such as Apple Accelerate / AMD ACML.

dependency detection problems

Correct/reliable detection of alternative BLAS implementations not guaranteed for many software packages.

Why do we need FlexiBLAS?

Profiling



- Profiling usually requires additional compiler settings

Why do we need FlexiBLAS?

Profiling



- Profiling usually requires additional compiler settings
- Profiler data requires additional (sometimes confusing) tools for evaluation

Why do we need FlexiBLAS?

Profiling



- Profiling usually requires additional compiler settings
- Profiler data requires additional (sometimes confusing) tools for evaluation
- Profilers often induce considerable overhead influencing the runtime behavior of the profiled application

Why do we need FlexiBLAS?

Profiling



- Profiling usually requires additional compiler settings
- Profiler data requires additional (sometimes confusing) tools for evaluation
- Profilers often induce considerable overhead influencing the runtime behavior of the profiled application
- Profiling needs to be active for entire applications

Why do we need FlexiBLAS?

Profiling



- Profiling usually requires additional compiler settings
- Profiler data requires additional (sometimes confusing) tools for evaluation
- Profilers often induce considerable overhead influencing the runtime behavior of the profiled application
- Profiling needs to be active for entire applications

Often only execution times and numbers of calls of single routines are of interest.

How does it work?

How does it work?

General Approach



We employ a plugin-like framework on top of the POSIX features for dynamic loading of shared libraries.

How does it work?

General Approach



We employ a plugin-like framework on top of the POSIX features for dynamic loading of shared libraries.

POSIX.1 2001 `dl*`-family

`dlopen` add a shared library and its dynamic dependencies to the current address space.

`dlsym` search for symbols in the current address space beginning in the handle retrieved by `dlopen`.

`dlclose` close a previously opened shared library if no other references to the library exist.

`dlerror` provide human readable error messages.

How does it work?

General Approach



dlopen based issues to solve

- 1 dlopen only integrates selected parts of the library:
Each required BLAS call needs to be initialized separately.



How does it work?

General Approach

`dlopen` based issues to solve

- 1 `dlopen` only integrates selected parts of the library:
Each required BLAS call needs to be initialized separately.
- 2 Dynamically (runtime) loaded symbols can not be resolved while linking a program.

How does it work?

General Approach



dlopen based issues to solve

- 1 dlopen only integrates selected parts of the library:
Each required BLAS call needs to be initialized separately.
- 2 Dynamically (runtime) loaded symbols can not be resolved while linking a program.
- 3 dlopen only loads a single file:
Multi-file implementations require additional treatment.

How does it work?

Initialization



```
__attribute__((constructor))
```

- automatically executed before the `main()` function
- replaces deprecated `_init()`
- Here used to read configuration and explicitly resolve all BLAS-routines to make sure they get loaded by `dlopen` as an initialization stage.

How does it work?

Initialization



`__attribute__((constructor))`

- automatically executed before the `main()` function
- replaces deprecated `_init()`
- Here used to read configuration and explicitly resolve all BLAS-routines to make sure they get loaded by `dlopen` as an initialization stage.

`__attribute__((destructor))`

- automatically executed after the `main()` function exits.
- replaces deprecated `_fini()`.
- Here used to cleanly close the loaded shared library and potentially print profiling data.

How does it work?

Wrapper Functions



Goal

Provide a 100% Netlib-BLAS compatible API and ABI for use in user applications.



How does it work?

Wrapper Functions

Goal

Provide a 100% Netlib-BLAS compatible API and ABI for use in user applications.

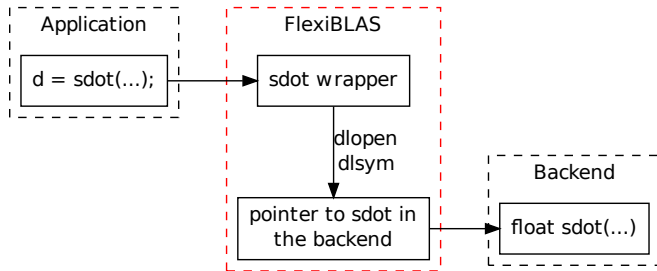


Figure: Calling `sdot` from an application via FlexiBLAS.

How does it work?

Profiling

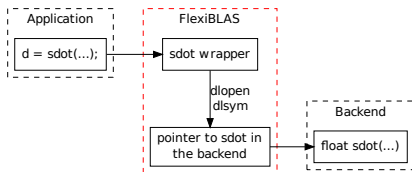


Figure: Calling `sdot` from an application via FlexiBLAS.



How does it work?

Profiling

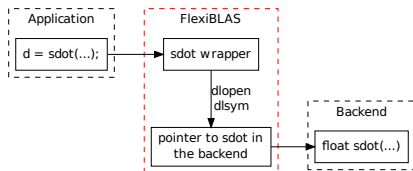


Figure: Calling `sdot` from an application via FlexiBLAS.

Basic Profiling

- Use `__attribute__((constructor))` to initialize global counters and timer variables for each BLAS-routine.

How does it work?

Profiling

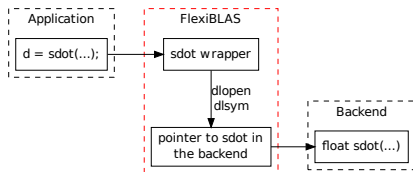


Figure: Calling `sdot` from an application via FlexiBLAS.

Basic Profiling

- Use `__attribute__((constructor))` to initialize global counters and timer variables for each BLAS-routine.
- Increase counters and timers inside the wrapper functions.



How does it work?

Profiling

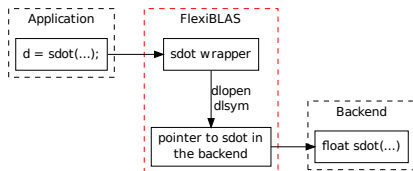


Figure: Calling `sdot` from an application via FlexiBLAS.

Basic Profiling

- Use `__attribute__((constructor))` to initialize global counters and timer variables for each BLAS-routine.
- Increase counters and timers inside the wrapper functions.
- Use `__attribute__((destructor))` for evaluation of the global variables and printing of statistics.

How can I use it then?



How can I use it then?

`flexiblas list / set`

FlexiBLAS provides the command line tool `flexiblas` for checking for the supported BLAS implementations and choosing the one to be used.

The tool closely follows `gentoo's eselect` syntax.



How can I use it then?

`flexiblas list / set`

FlexiBLAS provides the command line tool `flexiblas` for checking for the supported BLAS implementations and choosing the one to be used.

The tool closely follows gentoo's `eselect` syntax.

- To check for backends, do

```
flexiblas list
```



How can I use it then?

`flexiblas list / set`

FlexiBLAS provides the command line tool `flexiblas` for checking for the supported BLAS implementations and choosing the one to be used.

The tool closely follows gentoo's `eselect` syntax.

- To check for backends, do

```
flexiblas list
```

- To select the active backend, use

```
flexiblas set BLAS_BACKEND_NAME
```


How can I use it then?

`flexiblas list / set`



FlexiBLAS provides the command line tool `flexiblas` for checking for the supported BLAS implementations and choosing the one to be used.

The tool closely follows gentoo's `eselect` syntax.

- To check for backends, do

```
flexiblas list
```

- To select the active backend, use

```
flexiblas set BLAS_BACKEND_NAME
```

Both rely on configuration files generated automatically in `/etc/flexiblasrc` and `~/.flexiblasrc`

How can I use it then?

Environment Variables



Two main environment variables influence the behavior of FlexiBLAS.

How can I use it then?

Environment Variables



Two main environment variables influence the behavior of FlexiBLAS.

`FLEXIBLAS` can be used to override the default backend selected in the configuration files. Accepts either an alias from the configuration, a file name or a global file name.

How can I use it then?

Environment Variables



Two main environment variables influence the behavior of FlexiBLAS.

`FLEXIBLAS` can be used to override the default backend selected in the configuration files. Accepts either an alias from the configuration, a file name or a global file name.

That means we use something like:

```
export FLEXIBLAS=/usr/lib/libopenblas.so
```

or

```
export FLEXIBLAS=libblas_atlas.so
```

`libblas_atlas.so` must then reside somewhere in the default shared library search path.

How can I use it then?

Environment Variables



Two main environment variables influence the behavior of FlexiBLAS.

`FLEXIBLAS` can be used to override the default backend selected in the configuration files. Accepts either an alias from the configuration, a file name or a global file name.

`FLEXIBLAS_VERBOSE` If set to 1 additional information on the selected backend will be displayed.

How can I use it then?



Profiling

Two more environment variables influence the profiling behavior when FlexiBLAS is built with profiling support.

How can I use it then?

Profiling



Two more environment variables influence the profiling behavior when FlexiBLAS is built with profiling support.

`FLEXIBLAS_NOPROFILE` If set to 1 additional profiling information will **not** be displayed.

How can I use it then?



Profiling





Two more environment variables influence the profiling behavior when FlexiBLAS is built with profiling support.

`FLEXIBLAS_NOPROFILE` If set to 1 additional profiling information will **not** be displayed.

`FLEXIBLAS_PROFILE_FILE` can be used to select the file the profiling result is written to. The default file descriptor is the standard error output `stderr`.

Literature



-  J. DONGARRA, J. DU CROZ, S. HAMMARLING, AND R. HANSON, *An extended set of FORTRAN Basic Linear Algebra Subprograms*, ACM Trans. Math. Softw., 14 (1988), pp. 1–17.
-  J. DONGARRA, J. DU CROZ, I. DUFF, AND S. HAMMARLING, *A set of Level 3 Basic Linear Algebra Subprograms*, ACM Trans. Math. Softw., 16 (1990), pp. 1–17.
-  M. KÖHLER AND J. SAAK, *FlexiBLAS - A flexible BLAS library with runtime exchangeable backends*, Tech. Rep. 284, LAPACK Working Note, Jan. 2014.
-  C. L. LAWSON, R. J. HANSON, D. R. KINCAID, AND F. T. KROGH, *Basic linear algebra subprograms for fortran usage*, ACM Trans. Math. Softw., 5 (1979), pp. 308–323.

Vielen Dank für die Aufmerksamkeit!