Linking with Threading Libraries

Last updated on May 10, 2017

Intel MKL threading layer defines how Intel MKL functions utilize multiple computing cores of the system that the application runs on. You must link your application with one appropriate Intel MKL library in this layer, as explained below. Depending on whether this is a threading or a sequential library, Intel MKL runs in a parallel or sequential mode, respectively.

In the *parallel mode*, Intel MKL utilizes multiple processor cores available on your system, uses the OpenMP* or Intel TBB threading technology, and requires a proper threading run-time library (RTL) to be linked with your application. Independently of use of Intel MKL, the application may also require a threading RTL. You should link not more than one threading RTL to your application. Threading RTLs are provided by your compiler. Intel MKL provides several threading libraries, each dependent on the threading RTL of a certain compiler, and your choice of the Intel MKL threading library must be consistent with the threading RTL that you use in your application.

The OpenMP RTL of the Intel® compiler is the `libiomp5.dylib` library, located under `<parent directory>/compiler/lib`. This RTL is compatible with the GNU* compilers (gcc and gfortran). You can find additional information about the Intel OpenMP RTL at https://www.openmprtl.org.

The Intel TBB RTL of the Intel® compiler is the `libtbb.dylib` library, located under `<parent directory>/tbb/lib`. You can find additional information about the Intel TBB RTL at https://www.threadingbuildingblocks.org.

In the *sequential mode*, Intel MKL runs unthreaded code, does not require an threading RTL, and does not respond to environment variables and functions controlling the number of threads. Avoid using the library in the sequential mode unless you have a particular reason for that, such as the following:

- Your application needs a threading RTL that none of Intel MKL threading libraries is compatible with
- Your application is already threaded at a top level, and using parallel Intel MKL only degrades the application performance by interfering with that threading
- Your application is intended to be run on a single thread, like a message-passing Interface (MPI) application

It is critical to link the application with the proper RTL. The table below explains what library in the Intel MKL threading layer and what threading RTL you should choose under different scenarios:

Application		Intel MKL		RTL Required
Uses OpenMP	Compiled with	Execution Mode	Threading Layer	

no	any compiler	parallel	Static linking:
			`libmkl_intel_thread.a`
			Dynamic linking:
			`libmkl_intel_thread.dylib`
				`libiomp5.dylib`

Application		Intel MKL		RTL Required
Uses OpenMP	Compiled with	Execution Mode	Threading Layer	

no	any compiler	parallel	Static linking:
			`libmkl_tbb_thread.a`
			Dynamic linking:
			`libmkl_tbb_thread.dylib`
				`libtbb.dylib`

no	any compiler	sequential	Static linking:
			`libmkl_sequential.a`
			Dynamic linking:
			`libmkl_sequential.dylib`
				none†

yes	Intel compiler	parallel	Static linking:
			`libmkl_intel_thread.a`
			Dynamic linking:
			`libmkl_intel_thread.dylib`
				`libiomp5.dylib`

yes	GNU compiler	parallel	Static linking:
			`libmkl_intel_thread.a`
			Dynamic linking:
			`libmkl_intel_thread.dylib`
				`libiomp5.dylib`

yes	PGI* compiler	parallel	Static linking:
			`libmkl_pgi_thread.a`
			Dynamic linking:
			`libmkl_pgi_thread.dylib`
				PGI OpenMP RTL

yes	any other compiler	parallel	Not supported. Use Intel MKL in the sequential mode.	

† For the sequential mode, add the POSIX threads library (`libpthread`) to your link line because the `libmkl_sequential.a` and `libmkl_sequential.dylib` libraries depend on `libpthread`.

Optimization Notice

Parent topic: Linking in Detail (/node/9ae72e69-2c18-482f-8e78-3e78e6e1a856)

See Also

Layered Model Concept (/node/c4849acc-d263-425e-b209-4a4e64f568ac#C4849ACC-D263-425E-B209-4A4E64F568AC)

Notational Conventions (/node/4eac9eb9-829d-4461-8851-fa48fa541022#4EAC9EB9-829D-4461-8851-FA48FA541022)

Last Updated: Wednesday, May 10, 2017