

# Re: SMP LAPACK

---

*Source:* <http://sci.tech-archive.net/Archive/sci.math.num-analysis/2008-04/msg00369.html>

---

- *From:* "borchers@xxxxxxx" <borchers.brian@xxxxxxx>
  - *Date:* Sun, 20 Apr 2008 05:40:56 -0700 (PDT)
- 

On Apr 20, 4:42 am, Jon Harrop <j...@xxxxxxxxxxxxxxxxxxxx> wrote:

I believe the basic LAPACK library is single threaded. Is there a parallel LAPACK for SMP machines (i.e. not distributed like SCALAPACK)?

How well to real Fortran compilers (e.g. Intel's) automatically parallelize such code?

--

Dr Jon D Harrop, Flying Frog Consultancy <http://www.ffconsultancy.com/products/?u>

The LAPACK libraries make use of lower level routines from the BLAS libraries to perform more basic linear algebra operations (such as matrix-matrix multiplication.) In practice, nearly all of the time spent within a LAPACK subroutine call is actually spent in lower level BLAS subroutines. There are many threaded implementations of the BLAS that can be used with a single threaded LAPACK to greatly speed up your code.

In particular, look at the ATLAS implementation of the BLAS- it's open source software and runs on a wide variety of systems.

There are also a number of commercial packages that include threaded BLAS and LAPACK libraries. Intel's product is the "Math Kernel Library (MKL)", while AMD's is called "ACML" These libraries also contain other functions- for example the MKL includes fast fourier transform routines.

If your code spends most of its time in the LAPACK/BLAS routines, and if it is spending most of its time on the level 3 operations ( $O(n^3)$  operations such as matrix factorizations), then you'll typically see a very good speedup with the two to four processing cores common on today's desktop machines.

However, if your code spends a lot of time in level 1 and level 2 BLAS operations, then you probably will be disappointed with the speedup.

One reason for the poor performance is that these machines have

## Re: SMP LAPACK

relatively low memory bandwidth compared to their CPU speeds. In level 3 operations you can overcome this problem by bringing data into local cache memory and reusing it several times before flushing it out of the cache. For level 1 and level 2 BLAS operations there's no such advantage. As a result, level 1 and level 2 BLAS operations tend to be limited by the available memory bandwidth.

Note that in using these threaded libraries the compiler doesn't really have to do anything to parallelize the code. If you depend on the Fortran compiler to automatically parallelize the parts of the code that you've written, the compiler will have to do a lot more work to analyze your code. The performance of the resulting code is often disappointing.

An alternative that can be helpful is adding OpenMP directives to your code. With OpenMP, you add comments to your code that give the compiler explicit directions on how to parallelize the code. OpenMP is available in Intel's C and Fortran compilers as well as in the open source gcc compilers.

.