User's Guide
libipsfc v0.9

April 19, 2010

Introduction

The In-Place Storage Format Conversion library, `libipsfc`, leverages the effectiveness of the recently developed algorithm for in-place matrix transposition [1] to convert between standard data layouts such as column- and row-major and blocked data layouts. The conversion requires only a small amount of workspace, typically not larger than a single block of the matrix that is being converted. Parallelism is exploited on many levels and is expressed using the OpenMP framework. The library is written in Fortran 95 and OpenMP.

In certain cases, the performance delivered by `libipsfc` rivals the performance of out-of-place conversion. For this reason, it might be advantageous to use this library even when out-of-place conversion is possible. The primary intended application of the library is to support conversion in face of scarce memory resources.

API

`convert_storage_format`

This subroutine handles conversion between any pair of storage formats. The parameters are listed below.

- `A`
  The input and output matrix.
- `A11`, `A12`, `A21`, `A22`
  Pointers to the blocks A_{11}, A_{12}, A_{21}, and A_{22}. Offsets, in terms of matrix elements, from the base location of A.
- `m`, `n`
  The matrix A has `m` rows and `n` columns.
- `fromformat`
  The storage format of each block of A on input.
- `mb1`, `nb1`
  The primary block size, i.e., the block size of A_{11}, is `mb1` rows by `nb1` columns on input.
- `toformat`
  The storage format of each block of A on output.
- `mb2`, `nb2`
  The primary block size is `mb2` rows by `nb2` columns on output.

`transpose_explicitly`

Driver for explicit conversion in any storage format. The parameters are listed below.

- `A`
  The input and output matrix.
- `A11`, `A12`, `A21`, `A22`
  Pointers to the blocks A_{11}, A_{12}, A_{21}, and A_{22}. Offsets, in terms of matrix elements, from the base location of A. On output, the pointers `A12` and `A21` have been swapped.
- `informat`
  The storage format of each block of A on input and output.
- `m`, `n`
  The matrix A is `m` rows by `n` columns on input and `n` rows by `m` columns on output.
- `mb`, `nb`
  The primary block size, i.e., the block size of A_{11}, is `mb` rows by `nb` columns on input and `nb` rows by `mb` columns on output. These parameters are optional in case of the CM and RM formats.

`inplace_transposition_driver`

An implementation of the in-place matrix transposition algorithm developed in [1]. The parameters are listed below.

- `p`
  Number of OpenMP threads to use.
- `m`, `n`
  The size of the matrix is `m` rows by `n` columns.
- `A`
  The input and output matrix.
- `nprob`
  Number of independent problems.
- `me`, `ne`
  The size of each problem is `me` rows by `ne` columns.
- `L`
  Each block is `L` elements.

Testing/Timing Executables

`test_conversion`

Tests the storage format conversion aspect of the library. Currently, there are no command line options to this program, so parameter changes must be applied directly in the source. Each test case produces one line of output, such as

```
4  448  512  1   64   64  3   64   64  0.005054 0.220E-07  1  0
```

The quantities are, in order:

- OpenMP threads,

- rows in A,
- columns in A,
- input format (1:CM, 2:CCRB, 3:CRRB, 4:RCRB, 5:RRRB, 6:RM),
- rows in an input block,
- columns in an input block,
- output format,
- rows in an output block,
- columns in an output block,
- parallel execution time,
- normalized parallel execution time (seconds per matrix element),
- 0: wrong answer, 1: correct answer,
- 0: no packing performed, 1: packing performed.

`test_transposition`

Tests the inplace matrix transposition kernel. The command line options are, in order:

- rows in A,
- columns in A,
- rows in transposition problem,
- columns in transposition problem,
- size of one block,
- 'yes': verify answer, 'no': do not verify answer,
- number of repetitions of each test case.

The output of the command

```
test_transposition 450 125 45 25 50 yes 1
```

might look something like

```
nth         d4    d2(me)    d3(ne)     d1(L)         Tp     Tp/el ... verified?
  4          1        45        25        50  0.000725 0.129E-07 ...          1
```

A number of less important columns have been removed to make room. The first column is the number of OpenMP threads. The four columns after that are the problem parameters. The column `Tp` is the parallel execution time, and the column after that is the normalized execution time. A one in the last column means that the answer was verified to be correct.

`test_memory`

The `test_memory` executable can be used to obtain two benchmarks of the system: namely the time to copy an element and the time to scale an element. These values can be used as practical peak when evaluating the performance of the library.

The command line options are, in order:

- 'copy': copy one vector to another, 'scale': scale one vector in-place,
- exponent e that determines the length $n = 2^e$ of each vector,
- number of repetitions of each test case.

The output of the command

```
test_memory copy 22 1
```

will be something similar to

```
nth         n        Tp     Tp/el     copy?    scale?
  4   4194304  0.060855 0.145E-07         1         0
```

The last two columns determine the executed benchmark, in this case the 'copy' benchmark.

References

[1] Fred Gustavson, Lars Karlsson, and Bo Kågström, "Parallel and Cache-Efficient In-Place Matrix Storage Format Conversion,", *ACM Transactions on Mathematical Software*, (submitted) 2010. Also available as Technical Report UMINF 10.05, Department of Computing Science, Umeå University, SE-901 87, Sweden.