**Binary Input File Format for Combinatorial BLAS (CombBLAS)**

Each input file used in CombBLAS typically has to have a binary header that has the following fields and lengths. It is of size 52 bytes exact.

‘HKDT’: four 8-bit characters describing the beginning of header

Followed by six unsigned 64-bit integers:

* version number
* object size (including the row and column ids)
* format (0: binary, 1: ascii)
* number of rows
* number of columns
* number of nonzeros (nnz)

If format is ‘binary’, this is followed by nnz entries, each of which are of size “object size” and parsed by the HANDLER.binaryfill() function supplied by the user. The general signature of the function is:

void binaryfill(FILE \* rFile, IT & row, IT & col, NT & val)

IT is the index template parameter, and NT is the object template parameter. Below is an example:

template <class IT>

class TwitterReadSaveHandler

{

 void binaryfill(FILE \* rFile, IT & row, IT & col, TwitterEdge & val)

 {

 TwitterInteraction twi;

 fread (&twi,sizeof(TwitterInteraction),1,rFile);

 row = twi.from - 1;

 col = twi.to - 1;

 val = TwitterEdge(twi.retweets, twi.follow, twi.twtime);

 }

}

As seen, binaryfill reads indices as well. Please note that the file uses 1-based indices while C/C++ indices are zero based (hence the -1). In general, the number of bits used in the indices by the file should match the number of bits used by the program. If the program’s bits should be larger/smaller; then a cast after the original object creation can be employed. Here is an example to read a file with 64-bit integer indices into 32-bit local -per processor- indices (given that they fit):

typedef SpParMat < int64\_t, bool, SpDCCols<int64\_t,bool> > PSpMat;

typedef SpParMat < int64\_t, bool, SpDCCols<int32\_t,bool> > PSpMat\_s32;

PSpMat A;

A.ReadDistribute(string(argv[2]), 0);

PSpMat\_s32 Aeff = PSpMat\_s32(A);