1 Methodology

The test consist of two part. A data generating part and an analyzing part. The purpose of this test is understand how these two port work together and find out how many cores should be assigned to each part in order to make the test as fast as possible.

The test for NVDIMM tries all the different combination of the cores in order to find the fastest combination.

Data generation
The test have 1,000,000 nodes and 16,000,000 edges. Every nodes have 16 outgoing edges. Every edge goes to another random node. The random node are generated by using the rand library as shown below.

```
toNode = rand() % (nodes-1);
```

Analyze The analyze in this test consist of taking the average of the array generated by the data generating part.

Data speed prediction

```
while( n<50000 ){
        for( i=0; i<nodes; i++){
                x[i] = 0;
                for( j=CRS_row_ptr[i]; j<CRS_row_ptr[i+1]; j++)
                        x[i]+=CRS_values[j]*xk_1[CRS_col_idx[j]];
                x[i] *= d;
                x[i] += Wk_1_product;
                if( x[i]-xk_1[i] > diff )
                        diff = x[i] - xk_1[i];
        }
}
The formula for estimating speed:
(3*C2*8+1.5*C3)*50000*8*0.000001/Stream_speed
```

The predicted speed are estimated by taking the total amount of data transferred to and from the memory and divide it with memory speed measured by Stream. The edges are read 1.5 times from memory. This is because CRS_values consist of double values and CRS_row_ptr consist of integers. The arrays x and xk_1 are read or written to 3 times and every time this happens the program only need one element from the cashe line.

How to use nvdimm

In order to make use of the NVDIMM on the computer the system administrator must have installed the ndctl package for Linux so that programs that want to make use of the NVDIMM can access NVDIMM directly. The computer must also have a pmempool that can be used by the NVDIMM program. This can be created by using the pmempool package.
The code must declare the memory pool it want to use in the program and assign the path for the pmempool on the NVDIMM and the layout name. If the assignment failed the program will close. For allocation of array one must use the POBJ_ALLOC that must include what pool to use, the location of the pointer, variable type and the size of the array. When the program closes the programmer must remember to free up the array or it will cause a permanent memory leak. Permanent memory leak can only be freed up by deleting the memory pool and create a new one in the command line.
[**testing**]

```
POBJ_LAYOUT_BEGIN(array);
POBJ_LAYOUT_TOID(array, double);
POBJ_LAYOUT_END(array);
#define LAYOUT_NAME "my_layout"
TOID(double) nvm_values;

static PMEMobjpool *pop;

int main(int argc, char **argv){
        const char path[] = "/mnt/pmem0-xfs/pool.obj";

        pop = pmemobj_open(path, LAYOUT_NAME);
    if (pop == NULL) {
                perror(path);
                return 1;
    }

    POBJ_ALLOC(pop,&nvm_values,double,sizeof(double)*nodes,NULL,NULL);
    POBJ_FREE(&nvm_values);
    pmemobj_close(pop);
    return 0;
}
```

Stream, nvdimm
Stream is a program used to test the speed of the memory and that I have used to test the DRAM and NVDIMM of the computer. Stream measure the speed by copying data from one array to another and divide the amount

of data transfered with time in seconds. For the test of the DRAM I have made no modification of the code. For the NVDIMM I have replaced the DRAM allocation with allocation to the NVDIMM. This has been done by using the libpmemobj library and giving the libpmemobj pointers the same name as the DRAM pointers that have been removed. This way the changes have been minimized and ensure that the measurements can be compared to DRAM Stream result.

Benchmarks