Efficient Implementation of High Performance Algorithms on Intel Xeon Phi

Generated by Doxygen 1.8.9.1

Tue May 19 2015 02:59:25

Contents

Chapter 1

Main Page

Bachelor thesis

Dominik Simek xsimek23@stud.fit.vutbr.cz

FIT VUT 2015

Abstract

This thesis is dedicated to the implementation of high performance algorithms on the Intel Xeon Phi coprocessor. The Xeon phi was introduced by Intel as a new MIC (Many Integrated Core) architecture in 2012. The theoretical part of the thesis is focused on the architecture of the coprocessor (with peak performance of 2 tFLOPS for a single precision data) and on the procedure of algorithms implementation and optimization. The theoretical knowledge is then applied to a practical examples with demonstration of the implementation and the optimization of algorithms and work with the coprocessor. In the practical part of the thesis, simple benchmarks such as a vector matrix multiplication and a matrix multiplication are explained and implemented. In the first benchmark 6.5% of theoretical coprocessor performance was achieved, in the second it was much more. In following chapter a more complex benchmark\,–\,simulation of a particles system (N-Body), that reached more than 35% of coprocessor performance (725 gFLOPS), is discussed. The following section is dedicated to some interesting problems such as optimization of a MATLAB module k-Wave (propagation of the ultrasound waves), extraction of I-vector (speech processing), cross-compilation of existing libraries, modules and programs. In the conclusion of the thesis the usage the potential of the Intel Xeon Phi is evaluated.

Repository structure

```
1 -.
2  |-cross-compilation/
3  |-Doxyfile
4  |-fft-mkl/
5  |-matmul/
6  |-matvec/
7  |-nbody/
8  |-python-mkl/
9  |-README.md
10  |-text/
```

Description of each directory

- cross-compilation:
  - the directory contains a instructions to the cross-compilation of the libraries HDF5, SZIP, ZLIB and GNU MAKE
  - see cross-compilation/README.md for more informations
- fft-mkl:

  - the directory contains the source code of a test of fft routines from the Intel MKL on the MIC
  - see fft-mkl/README.md for more informations

- matmul:
  - the directory contains the source code of the matrix multiplication (MKL)
  - see matmul/README.md for more informations

- matvec:
  - the directory contains the source code of the matrix vector multiplication
  - see matvec/README.md for more informations

- nbody:
  - the directory contains the source code of the N-Body simulation
  - see nbody/README.md for more informations

- python-mkl:
  - the directory contains the source code of the DGEMM (Python modules Numpy and Scipy lined with MKL)
  - see python-mkl/README.md for more informations

- text:
  - Latex source code of the thesis

Chapter 2

CROSS-COMPILATION OF HDF5, SZIP, ZLIB AND GNU MAKE

Dominik Simek xsimek23@fit.vutbr.cz

Bachelor thesis

FIT VUT 2015

Basic informations

- each .txt file contains instructions of a cross-compilation of some library

Chapter 3

FFT TEST (MKL) ON THE MIC

Dominik Simek xsimek23@fit.vutbr.cz

Bachelor thesis

FIT VUT 2015

Basic informations

- all directories have a same structure:

```
1 -.
2  |-fftw-3d-omp.c (source file)
3  |-Makefile (make, make run, make clean)
4  |-papi_cntr.h (simpler work with PAPI)
5  |-run_test.sh (run program for various matrix sizes)
```

- if you are working on the Anselm:
  - create interactive job on a MIC node
    qsub -I -q qmic -A PROJECT-ID
  - load necessary modules
    module load intel
    module load papi
- usage:
  - cd /ffw-mkl/cpu
  - make # create executable file
  - make run # run computation
  - make clean # clean up
  - or
    ./fftw-3d-omp MATRIX_DIMENSION (e.g. $ ./fftw-3d-omp 128)
  - or
    ./run_test.sh # run bash script

A description of each directories

- cpu:
  - implementation for the cpu
  - set a number of threads
    export OMP_SET_NUM_THREADS=8

  - run for matrixes of size 128
    make run
  - directory contains also bash script run_test.sh
  - run it for various matrix sizes and various number of threads
    ./run_test.sh

- mic:
  - compile program on a host
    [host]$ make
  - and run it on the MIC
    [host]$ ssh mic
    [mic0]$ # MIC shared libraries (it can be different for specific system)
    [mic0]$ export LD_LIBRARY_PATH=/apps/intel/composer_xe_2015.2.164/compiler/lib/mic:$LD_LIB↩
    RARY_PATH
    [mic0]$ export LD_LIBRARY_PATH=/apps/intel/composer_xe_2015.2.164/mkl/lib/mic/:$LD_LIBRAR↩
    Y_PATH
    [mic0]$ cd fftw-mkl/mic
    [mic0]$ export OMP_NUM_THREADS=120 # 60, 120, 180, 240
    [mic0]$ ./fftw-3d-omp 2048
  - directory contains also bash script run_test.sh
  - run it for various matrix sizes and various number of threads
    bash run_test.sh

Chapter 4

MATRIX MULTIPLICATION

Dominik Simek xsimek23@fit.vutbr.cz

Bachelor thesis

FIT VUT 2015

Basic informations

- all directories have a same structure:

```
1 - .
2   |-c_dgemm.c (source file)
3   |-Makefile (make, make run, make clean)
4   |-papi_cntr.h (simpler work with PAPI)
5   |-run_c.sh (run program for various matrix sizes)
```

- if you are working on the Anselm:
  - create interactive job on a MIC node
    qsub -I -q qmic -A PROJECT-ID
  - load necessary modules
    module load intel
- usage:
  - cd cpu
  - make # create executable file
  - make run # run computation
  - make clean # clean up
  - or
    ./c_dgemm MATRIX_DIMENSION (e.g. $ ./c_dgemm 2048)
  - or
    ./run_c.sh # run bash script

A description of each directories

- cpu:
  - implementation for the cpu
  - set a number of threads
    export OMP_SET_NUM_THREADS=8

  - run for matrixes of size 1024
    make run
  - directory contains also bash script run_c.sh
  - run it for various matrix sizes
    ./run_c.sh

- mic:
  - compile program on a host
    [host]$ make
  - and run it on the MIC
    [host]$ ssh mic
    [mic0]$ # MIC shared libraries (it can be different for specific system)
    [mic0]$ export LD_LIBRARY_PATH=/apps/intel/composer_xe_2015.2.164/compiler/lib/mic:$LD_LIB↩
    RARY_PATH
    [mic0]$ export LD_LIBRARY_PATH=/apps/intel/composer_xe_2015.2.164/mkl/lib/mic/:$LD_LIBRAR↩
    Y_PATH
    [mic0]$ cd matmul/mic
    [mic0]$ export OMP_NUM_THREADS=120 # 60, 120, 180, 240
    [mic0]$ ./c_dgemm 2048
  - directory contains also bash script run_c.sh
  - run it for various matrix sizes
    bash run_c.sh

Chapter 5

MATRIX VECTOR MULTIPLICATION

Dominik Simek xsimek23@fit.vutbr.cz

Bachelor thesis

FIT VUT 2015

Basic informations

- all directories have a same structure:

```
1 - .
2   |-main.cpp (main function)
3   |-Makefile (make, make run, make clean)
4   |-matvec.cpp (matvec kernel)
5   |-matvec.h (matvec header file)
6   |-papi_cntr.h (simpler work with PAPI)
```

- if you are working on the Anselm:
  - create interactive job on a MIC node
    qsub -I -q qmic -A PROJECT-ID
  - load necessary modules
    module load intel
    module load papi
- in the Makefile you can set:
  - ROWS, COLS (matrix size)
  - RUNS (number of repetitions)
  - PADDING (implemented only in the vec-padding version !)
  - PAPI_EVENTS (set PAPI counters to monitoring of HW events)
- usage:
  - cd naive # e.g.
  - make # create executable file
  - make run # run computation
  - make clean # clean up

A description of each directories

- naive:
  - naive implementation of algorithm
  - no vectorization, no optimizations
- vec-padding:
  - vectorization, compiler optimizations
- dyn-aligned:
  - dynamic allocation
  - data aligned to 64B
- omp-parallel:
  - parallel version (OpenMP)
  - set number of threads
    export OMP_NUM_THREADS=N
  - you can also set binding of threads
    export KMP_AFFINITY=compact|scatter
  - make run
  - directory contains also bash script run_exp.sh
    - compile the program with your favourite matrix size at first
    - run the script
      ./run_exp.sh
    - this will run the program gradually on 1, 2, 4, 8 and 16 threads
- omp-parallel-mic:
  - parallel version (OpenMP) for the MIC
  - compile program on a host
    [host]$ make
  - and run it on the MIC
    [host]$ ssh mic
    [mic0]$ # MIC shared libraries (it can be different for specific system)
    [mic0]$ export LD_LIBRARY_PATH=/apps/intel/composer_xe_2015.2.164/compiler/lib/mic:$LD_LIB↩
    RARY_PATH
    [mic0]$ export LD_LIBRARY_PATH=/apps/intel/composer_xe_2015.2.164/mkl/lib/mic/:$LD_LIBRAR↩
    Y_PATH
    [mic0]$ cd naive
    [mic0]$ export OMP_NUM_THREADS=120 # 60, 120, 180, 240
    [mic0]$ ./matvec
  - directory contains also bash script run_exp.sh
    - compile the program with your favourite matrix size at first
    - run the script
      bash run_exp.sh
    - this will run the program gradually on 1, 2, 4, 8, 16, 30, 60, 120, 180 and 240 threads

Chapter 6

N-BODY SIMULATION

Dominik Simek xsimek23@fit.vutbr.cz

Bachelor thesis

FIT VUT 2015

Basic informations

- all directories have a same structure:

```
1 - .
2   |-gen.cpp (generate input file)
3   |-main.cpp (main function)
4   |-Makefile (make, make run, make clean)
5   |-nbody.cpp (nbody kernel)
6   |-nbody.h (nbody header file)
7   |-papi_cntr.h (simpler work with PAPI)
```

- if you are working on the Anselm:
  - create interactive job on a MIC node
    qsub -I -q qmic -A PROJECT-ID
  - load necessary modules
    module load intel
    module load papi
- in the Makefile you can set:
  - N (number of particles)
  - STEPS (number of simulation steps)
  - DT (time between steps)
  - PAPI_EVENTS (set PAPI counters to monitoring of HW events)
- usage:
  - cd naive # e.g.
  - make # create executable file
  - make run # run computation
  - make clean # clean up
- Makefile generate also executable file 'gen':
  - you can generate own input file (with "random" particle attributes)
    ./gen N input.dat # N is number of particles

A description of each directories

- naive:
  - naive implementation of algorithm
  - bad structure of loop
- no-jump-auto-opt:
  - enhanced loop structure
  - no branches
  - dynamic allocation
  - data aligned to 64B
  - compiler optimizations
- omp-parallel:
  - parallel version (OpenMP)
  - set number of threads
    export OMP_NUM_THREADS=N
  - you can also set binding of threads
    export KMP_AFFINITY=compact|scatter
  - make run
  - directory contains also bash script run_exp.sh
    - compile the program with your favourite matrix size at first
    - run the script
      ./run_exp.sh
    - this will run the program gradually on 1, 2, 4, 8 and 16 threads
- omp-parallel-mic:
  - parallel version (OpenMP) for the MIC
  - compile program on a host
    [host]$ make
  - and run it on the MIC
    [host]$ ssh mic
    [mic0]$ # MIC shared libraries (it can be different for specific system)
    [mic0]$ export LD_LIBRARY_PATH=/apps/intel/composer_xe_2015.2.164/compiler/lib/mic:$LD_LIB↩
    RARY_PATH
    [mic0]$ export LD_LIBRARY_PATH=/apps/intel/composer_xe_2015.2.164/mkl/lib/mic/:$LD_LIBRAR↩
    Y_PATH
    [mic0]$ cd naive
    [mic0]$ export OMP_NUM_THREADS=120 # 60, 120, 180, 240
    [mic0]$ ./nbody ../input/1m.dat ../output/omp-mic-out.dat
  - directory contains also bash script run_exp.sh
    - compile the program with your favourite matrix size at first
    - run the script
      bash run_exp.sh
    - this will run the program gradually on 1, 2, 4, 8, 16, 30, 60, 120, 180 and 240 threads
- cache-block-mic:
  - effort to better cache utilization
  - you gave to set a correct number of particles and a correct block size
    - the block size have to be set in the file nbody.cpp, constant BLOCK

        - e.g.
            - const int BLOCK = 2048; (nbody.cpp)
            - N=1208320 (in Makefile)
            - now there are lot of bodies - data are bigger than caches
            - 1208320/2048=590 (590 blocks of particles, it must be integer !)
    - to compile and run use process from omp-parallel-mic paragraph
    - make; ssh mic0; ...
- offload:
    - offload mode
    - you can set number of a MIC threads directly on the source code (main.cpp)
      const int mic_threads = 120;
    - or by enviroment variable
      export MIC_ENV_PREFIX=MIC
      export MIC_OMP_NUM_THREADS=120
    - compile and run on the host
      [host]$ make [host]$ make run
- input:
    - input files with particles attributes
- output:
    - output files are created here (after make run)

Chapter 7

NUMPY & SCIPY DGEMM TEST

Dominik Simek xsimek23@fit.vutbr.cz

Bachelor thesis

FIT VUT 2015

Basic informations

- test DGEMM from numpy/scipy modules (on the CPU)
- for best perfomance, numpy/scipy module have to be linked with Intel MKL
- usage:
  - set number of threads
    export OMP_NUM_THREADS=8
  - run python script
    python numpy_dgemm.py MATRIX_DIMENSION
    python numpy_dgemm.py MATRIX_DIMENSION
  - e.g.
    python numpy_dgemm.py 2048

Chapter 8

Namespace Index

8.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 9

Class Index

9.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 10

File Index

10.1 File List

Here is a list of all files with brief descriptions:

Chapter 11

Namespace Documentation

11.1 numpy_dgemm Namespace Reference

Variables

- tuple K = int(sys.argv[1])
- M = K
- N = K
- int ITER = 1
- tuple t_init = time.time()
- tuple a1 = np.array(np.random.random((M, K)), dtype=np.double, order='C', copy=False)
- tuple a2 = np.array(np.random.random((K, N)), dtype=np.double, order='C', copy=False)
- tuple m1 = np.matrix(a1, dtype=np.double, copy=False)
- tuple m2 = np.matrix(a2, dtype=np.double, copy=False)
- tuple t_start = time.time()
- tuple mf2 = np.dot(m1, m2)
- tuple t_end = time.time()

11.1.1 Variable Documentation

11.1.1.1 tuple numpy_dgemm.a1 = np.array(np.random.random((M, K)), dtype=np.double, order='C', copy=False)

Definition at line 25 of file numpy_dgemm.py.

11.1.1.2 tuple numpy_dgemm.a2 = np.array(np.random.random((K, N)), dtype=np.double, order='C', copy=False)

Definition at line 26 of file numpy_dgemm.py.

11.1.1.3 int numpy_dgemm.ITER = 1

Definition at line 18 of file numpy_dgemm.py.

11.1.1.4 tuple numpy_dgemm.K = int(sys.argv[1])

Definition at line 15 of file numpy_dgemm.py.

11.1.1.5 numpy_dgemm.M = K

Definition at line 16 of file numpy_dgemm.py.

11.1.1.6 tuple numpy_dgemm.m1 = np.matrix(a1, dtype=np.double, copy=False)

Definition at line 28 of file numpy_dgemm.py.

11.1.1.7 tuple numpy_dgemm.m2 = np.matrix(a2, dtype=np.double, copy=False)

Definition at line 29 of file numpy_dgemm.py.

11.1.1.8 tuple numpy_dgemm.mf2 = np.dot(m1, m2)

Definition at line 34 of file numpy_dgemm.py.

11.1.1.9 numpy_dgemm.N = K

Definition at line 17 of file numpy_dgemm.py.

11.1.1.10 tuple numpy_dgemm.t_end = time.time()

Definition at line 37 of file numpy_dgemm.py.

11.1.1.11 tuple numpy_dgemm.t_init = time.time()

Definition at line 23 of file numpy_dgemm.py.

11.1.1.12 tuple numpy_dgemm.t_start = time.time()

Definition at line 31 of file numpy_dgemm.py.

11.2 scipy_dgemm Namespace Reference

Variables

- tuple K = int(sys.argv[1])
- M = K
- N = K
- int ITER = 1
- tuple a1 = np.array(np.random.random((M, K)), dtype=np.double, order='C', copy=False)
- tuple a2 = np.array(np.random.random((K, N)), dtype=np.double, order='C', copy=False)
- tuple m1 = np.matrix(a1, dtype=np.double, copy=False)
- tuple m2 = np.matrix(a2, dtype=np.double, copy=False)
- tuple t_start = time.time()
- tuple mf2 = sp.dgemm(alpha=1.0, a=m1, b=m2)
- tuple t_end = time.time()

11.2.1 Variable Documentation

11.2.1.1 tuple scipy_dgemm.a1 = np.array(np.random.random((M, K)), dtype=np.double, order='C', copy=False)

Definition at line 24 of file scipy_dgemm.py.

11.2.1.2 tuple scipy_dgemm.a2 = np.array(np.random.random((K, N)), dtype=np.double, order='C', copy=False)

Definition at line 25 of file scipy_dgemm.py.

11.2.1.3 int scipy_dgemm.ITER = 1

Definition at line 19 of file scipy_dgemm.py.

11.2.1.4 tuple scipy_dgemm.K = int(sys.argv[1])

Definition at line 16 of file scipy_dgemm.py.

11.2.1.5 scipy_dgemm.M = K

Definition at line 17 of file scipy_dgemm.py.

11.2.1.6 tuple scipy_dgemm.m1 = np.matrix(a1, dtype=np.double, copy=False)

Definition at line 27 of file scipy_dgemm.py.

11.2.1.7 tuple scipy_dgemm.m2 = np.matrix(a2, dtype=np.double, copy=False)

Definition at line 28 of file scipy_dgemm.py.

11.2.1.8 tuple scipy_dgemm.mf2 = sp.dgemm(alpha=1.0, a=m1, b=m2)

Definition at line 35 of file scipy_dgemm.py.

11.2.1.9 scipy_dgemm.N = K

Definition at line 18 of file scipy_dgemm.py.

11.2.1.10 tuple scipy_dgemm.t_end = time.time()

Definition at line 37 of file scipy_dgemm.py.

11.2.1.11 tuple scipy_dgemm.t_start = time.time()

Definition at line 30 of file scipy_dgemm.py.

Chapter 12

Class Documentation

12.1 t_particle Struct Reference

`#include <nbody.h>`

Public Attributes

- float pos_x
- float pos_y
- float pos_z
- float vel_x
- float vel_y
- float vel_z
- float weight

12.1.1 Detailed Description

Definition at line 22 of file nbody.h.

12.1.2 Member Data Documentation

12.1.2.1 float t_particle::pos_x

Definition at line 24 of file nbody.h.

12.1.2.2 float t_particle::pos_y

Definition at line 25 of file nbody.h.

12.1.2.3 float t_particle::pos_z

Definition at line 26 of file nbody.h.

12.1.2.4 float t_particle::vel_x

Definition at line 27 of file nbody.h.

12.1.2.5 float t_particle::vel_y

Definition at line 28 of file nbody.h.

12.1.2.6 float t_particle::vel_z

Definition at line 29 of file nbody.h.

12.1.2.7 float t_particle::weight

Definition at line 30 of file nbody.h.

The documentation for this struct was generated from the following file:

- nbody/naive/nbody.h

12.2 t_particles_DA Struct Reference

`#include <nbody.h>`

Public Attributes

- float * pos_x
- float * pos_y
- float * pos_z
- float * vel_x
- float * vel_y
- float * vel_z
- float * weight
- float * fx
- float * fy
- float * fz
- float * ax
- float * ay
- float * az

12.2.1 Detailed Description

Definition at line 23 of file nbody.h.

12.2.2 Member Data Documentation

12.2.2.1 float * t_particles_DA::ax

Definition at line 35 of file nbody.h.

12.2.2.2 float * t_particles_DA::ay

Definition at line 36 of file nbody.h.

12.2.2.3 float * t_particles_DA::az

Definition at line 37 of file nbody.h.

12.2.2.4 float * t_particles_DA::fx

Definition at line 32 of file nbody.h.

12.2.2.5 float * t_particles_DA::fy

Definition at line 33 of file nbody.h.

12.2.2.6 float * t_particles_DA::fz

Definition at line 34 of file nbody.h.

12.2.2.7 float * t_particles_DA::pos_x

Definition at line 25 of file nbody.h.

12.2.2.8 float * t_particles_DA::pos_y

Definition at line 26 of file nbody.h.

12.2.2.9 float * t_particles_DA::pos_z

Definition at line 27 of file nbody.h.

12.2.2.10 float * t_particles_DA::vel_x

Definition at line 28 of file nbody.h.

12.2.2.11 float * t_particles_DA::vel_y

Definition at line 29 of file nbody.h.

12.2.2.12 float * t_particles_DA::vel_z

Definition at line 30 of file nbody.h.

12.2.2.13 float * t_particles_DA::weight

Definition at line 31 of file nbody.h.

The documentation for this struct was generated from the following file:

- nbody/cache-block-mic/nbody.h

Chapter 13

File Documentation

13.1 cross-compilation/README.md File Reference

13.2 fft-mkl/README.md File Reference

13.3 matmul/README.md File Reference

13.4 matvec/README.md File Reference

13.5 nbody/README.md File Reference

13.6 python-mkl/README.md File Reference

13.7 README.md File Reference

13.8 fft-mkl/cpu/fftw-3d-omp.c File Reference

```
#include <stdlib.h>
#include <fftw3.h>
#include <omp.h>
#include <mkl.h>
#include "papi_cntr.h"
```

Functions

- int main (int argc, char **argv)

13.8.1 Detailed Description

Simple FFTW benchmark Test FFTW performance on the Intel Xeon processor MKL version

Author

Dominik Simek xsimek23@stud.fit.vutbr.cz Bachelor thesis FIT VUT 2015

13.8.2 Function Documentation

13.8.2.1 int main (int *argc,* char ** *argv*)

Definition at line 19 of file fftw-3d-omp.c.

13.9 fft-mkl/mic/fftw-3d-omp.c File Reference

```
#include <stdlib.h>
#include <fftw3.h>
#include <omp.h>
#include <mkl.h>
```

Functions

- int main (int argc, char **argv)

13.9.1 Detailed Description

Simple FFTW benchmark Test FFTW performance on the Intel Xeon Phi MKL version Compare with Intel Xeon performance

Author

Dominik Simek xsimek23@stud.fit.vutbr.cz Bachelor thesis FIT VUT 2015

13.9.2 Function Documentation

13.9.2.1 int main (int *argc,* char ** *argv*)

Definition at line 19 of file fftw-3d-omp.c.

13.10 fft-mkl/mic/fftw-3d-omp-bpage.c File Reference

```
#include <stdlib.h>
#include <fftw3.h>
#include <omp.h>
#include <mkl.h>
#include <sys/mman.h>
```

Functions

- int main (int argc, char **argv)

13.10.1 Detailed Description

Simple fftwf benchmark Test fftwf performance at Intel Xeon Phi This use MKL Compare with Intel Xeon performance

13.10.2 Function Documentation

13.10.2.1 int main (int *argc,* char ** *argv*)

Definition at line 16 of file fftw-3d-omp-bpage.c.

13.11 matmul/cpu/c_dgemm.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include "mkl.h"
#include <omp.h>
#include "papi_cntr.h"
```

Functions

- int main (int argc, char *argv[ ])

Variables

- const double ALPHA = 1.0
- const double BETA = 0.0

13.11.1 Function Documentation

13.11.1.1 int main (int *argc,* char * *argv[ ]*)

Definition at line 22 of file c_dgemm.c.

13.11.2 Variable Documentation

13.11.2.1 const double ALPHA = 1.0

Definition at line 19 of file c_dgemm.c.

13.11.2.2 const double BETA = 0.0

Definition at line 20 of file c_dgemm.c.

13.12 matmul/mic/c_dgemm.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include "mkl.h"
#include <omp.h>
```

Functions

- int main (int argc, char ∗argv[ ])

Variables

- const double ALPHA = 1.0
- const double BETA = 0.0

13.12.1 Function Documentation

13.12.1.1 int main (int *argc,* char ∗ *argv[ ]*)

Definition at line 21 of file c_dgemm.c.

13.12.2 Variable Documentation

13.12.2.1 const double ALPHA = 1.0

Definition at line 18 of file c_dgemm.c.

13.12.2.2 const double BETA = 0.0

Definition at line 19 of file c_dgemm.c.

13.13 matvec/dynamic-aligned/main.cpp File Reference

```
#include <cstdio>
#include <cmath>
#include "matvec.h"
#include "papi_cntr.h"
```

Functions

- void mat_init (int row, int col, float off, float ∗a)
- void vec_init (int length, float off, float ∗a)
- double vec_sum (int length, float ∗vec)
- int main (int argc, char ∗∗argv)

13.13.1 Function Documentation

13.13.1.1 int main (int *argc,* char ∗∗ *argv*)

Main function

Parameters

argc	
argv	

Definition at line 71 of file main.cpp.

13.13.1.2 void mat_init (int *row*, int *col*, float *off*, float * *a*)

Initialize matrix with "random" values

Parameters

row	- matrix rows
col	- matrix col
off	- auxiliary offset
a	- matrix to initialize

Definition at line 26 of file main.cpp.

13.13.1.3 void vec_init (int *length*, float *off*, float * *a*)

Initialize vector with "random" values

Parameters

length	- vector length
off	- auxiliary offset
a	- vector to initialize

Definition at line 42 of file main.cpp.

13.13.1.4 double vec_sum (int *length*, float * *vec*)

Compute sum of final vector

Returns

sum - final vector sum

Definition at line 55 of file main.cpp.

13.14 matvec/naive/main.cpp File Reference

```
#include <cstdio>
#include <cmath>
#include "matvec.h"
#include "papi_cntr.h"
```

Functions

- void mat_init (int row, int col, float off, float a[ ][COLS])
- void vec_init (int length, float off, float a[ ])
- double vec_sum (int length, float vec[ ])
- int main (int argc, char **argv)

13.14.1 Function Documentation

13.14.1.1 int main (int *argc,* char ∗∗ *argv*)

Main function

Parameters

argc	
argv	

Definition at line 71 of file main.cpp.

13.14.1.2 void mat_init (int *row,* int *col,* float *off,* float *a[ ][COLS]*)

Initialize matrix with "random" values

Parameters

row	- matrix rows
col	- matrix col
off	- auxiliary offset
a	- matrix to initialize

Definition at line 26 of file main.cpp.

13.14.1.3 void vec_init (int *length,* float *off,* float *a[ ]*)

Initialize vector with "random" values

Parameters

length	- vector length
off	- auxiliary offset
a	- vector to initialize

Definition at line 42 of file main.cpp.

13.14.1.4 double vec_sum (int *length,* float *vec[ ]*)

Compute sum of final vector

Returns

sum - final vector sum

Definition at line 55 of file main.cpp.

13.15 matvec/omp-parallel/main.cpp File Reference

```
#include <cstdio>
#include <cmath>
#include <omp.h>
#include "matvec.h"
#include "papi_cntr.h"
```

Functions

- void mat_init (int row, int col, float off, float ∗a)
- void vec_init (int length, float off, float ∗a)
- double vec_sum (int length, float ∗vec)
- int main (int argc, char ∗∗argv)

13.15.1 Function Documentation

13.15.1.1 int main (int *argc,* char ∗∗ *argv*)

Main function

Parameters

argc	
argv	

Definition at line 76 of file main.cpp.

13.15.1.2 void mat_init (int *row,* int *col,* float *off,* float ∗ *a*)

Initialize matrix with "random" values

Parameters

row	- matrix rows
col	- matrix col
off	- auxiliary offset
a	- matrix to initialize

Definition at line 27 of file main.cpp.

13.15.1.3 void vec_init (int *length,* float *off,* float ∗ *a*)

Initialize vector with "random" values

Parameters

length	- vector length
off	- auxiliary offset
a	- vector to initialize

Definition at line 45 of file main.cpp.

13.15.1.4 double vec_sum (int *length,* float ∗ *vec*)

Compute sum of final vector

Returns

sum - final vector sum

Definition at line 60 of file main.cpp.

13.16 matvec/omp-parallel-mic/main.cpp File Reference

```
#include <cstdio>
#include <cmath>
#include <omp.h>
#include "matvec.h"
```

Functions

- void mat_init (int row, int col, float off, float *a)
- void vec_init (int length, float off, float *a)
- double vec_sum (int length, float *vec)
- int main (int argc, char **argv)

13.16.1 Function Documentation

13.16.1.1 int main (int *argc,* char ** *argv*)

Main function

Parameters

argc	
argv	

Definition at line 81 of file main.cpp.

13.16.1.2 void mat_init (int *row,* int *col,* float *off,* float * *a*)

Initialize matrix with "random" values

Parameters

row	- matrix rows
col	- matrix col
off	- auxiliary offset
a	- matrix to initialize

Definition at line 26 of file main.cpp.

13.16.1.3 void vec_init (int *length,* float *off,* float * *a*)

Initialize vector with "random" values

Parameters

length	- vector length
off	- auxiliary offset
a	- vector to initialize

Definition at line 48 of file main.cpp.

13.16.1.4 double vec_sum (int *length,* float * *vec*)

Compute sum of final vector

Returns

sum - final vector sum

Definition at line 65 of file main.cpp.

13.17 matvec/vec-padding/main.cpp File Reference

```
#include <cstdio>
#include <cmath>
#include "matvec.h"
#include "papi_cntr.h"
```

Functions

- void mat_init (int row, int col, float off, float a[ ][COL_PAD])
- void vec_init (int length, float off, float a[ ])
- double vec_sum (int length, float vec[ ])
- int main (int argc, char ∗∗argv)

Variables

- const unsigned COL_PAD = COLS + PADDING

13.17.1 Function Documentation

13.17.1.1 int main (int *argc,* char ∗∗ *argv*)

Main function

Parameters

argc	
argv	

Definition at line 105 of file main.cpp.

13.17.1.2 void mat_init (int *row,* int *col,* float *off,* float *a[ ][COL_PAD]*)

Initialize matrix with "random" values

Parameters

row	- matrix rows
col	- matrix col
off	- auxiliary offset
a	- matrix to initialize

Definition at line 28 of file main.cpp.

13.17.1.3 void vec_init (int *length,* float *off,* float *a[ ]*)

Initialize vector with "random" values

Parameters

length	- vector length
off	- auxiliary offset
a	- vector to initialize

Definition at line 62 of file main.cpp.

13.17.1.4 double vec_sum (int *length,* float *vec[ ]*)

Compute sum of final vector

Returns

sum - final vector sum

Definition at line 87 of file main.cpp.

13.17.2 Variable Documentation

13.17.2.1 const unsigned COL_PAD = COLS + PADDING

Definition at line 18 of file main.cpp.

13.18 nbody/cache-block-mic/main.cpp File Reference

```
#include <cstdio>
#include <cmath>
#include <omp.h>
#include "nbody.h"
```

Functions

- int main (int argc, char **argv)

13.18.1 Function Documentation

13.18.1.1 int main (int *argc,* char ** *argv*)

Main function

Parameters

argc	
argv	

Definition at line 24 of file main.cpp.

13.19 nbody/naive/main.cpp File Reference

```
#include <cstdio>
```

```
#include <cmath>
#include "nbody.h"
#include "papi_cntr.h"
```

Functions

- int main (int argc, char **argv)

13.19.1 Function Documentation

13.19.1.1 int main (int *argc,* char ** *argv*)

Main function

Parameters

argc	
argv	

Definition at line 24 of file main.cpp.

13.20 nbody/no-jump-auto-opt/main.cpp File Reference

```
#include <cstdio>
#include <cmath>
#include <omp.h>
#include "nbody.h"
#include "papi_cntr.h"
```

Functions

- int main (int argc, char **argv)

13.20.1 Function Documentation

13.20.1.1 int main (int *argc,* char ** *argv*)

Main function

Parameters

argc	
argv	

Definition at line 25 of file main.cpp.

13.21 nbody/offload/main.cpp File Reference

```
#include <cstdio>
```

```
#include <cmath>
#include <omp.h>
#include <offload.h>
#include "nbody.h"
```

Functions

- int main (int argc, char **argv)

13.21.1 Function Documentation

13.21.1.1 int main (int *argc*, char ** *argv*)

Main function

Parameters

argc	
argv	

Definition at line 27 of file main.cpp.

13.22 nbody/omp-parallel/main.cpp File Reference

```
#include <cstdio>
#include <cmath>
#include <omp.h>
#include "nbody.h"
#include "papi_cntr.h"
```

Functions

- int main (int argc, char **argv)

13.22.1 Function Documentation

13.22.1.1 int main (int *argc*, char ** *argv*)

Main function

Parameters

argc	
argv	

Definition at line 25 of file main.cpp.

13.23 nbody/omp-parallel-mic/main.cpp File Reference

```
#include <cstdio>
```

```
#include <cmath>
#include <omp.h>
#include "nbody.h"
```

Functions

- int main (int argc, char **argv)

13.23.1 Function Documentation

13.23.1.1 int main (int *argc,* char ** *argv*)

Main function

Parameters

argc	
argv	

Definition at line 24 of file main.cpp.

13.24 matvec/dynamic-aligned/matvec.cpp File Reference

```
#include "matvec.h"
```

Functions

- void mat_vec_mul (int rows, int cols, float *a, float *b, float *c)

13.24.1 Function Documentation

13.24.1.1 void mat_vec_mul (int *rows,* int *cols,* float * *a,* float * *b,* float * *c*)

Compute matrix and vector multiplication

Parameters

rows	- matrix rows
cols	- matrix cols
a	- matrix
b	- vector
c	- final vector

Definition at line 23 of file matvec.cpp.

13.25 matvec/naive/matvec.cpp File Reference

```
#include "matvec.h"
```

Functions

- void mat_vec_mul (int rows, int cols, float a[ ][cols], float b[cols], float c[rows])

13.25.1 Function Documentation

13.25.1.1 void mat_vec_mul (int *rows,* int *cols,* float *a[ ][cols],* float *b[cols],* float *c[rows]*)

Compute matrix and vector multiplication

Parameters

rows	- matrix rows
cols	- matrix cols
a	- matrix
b	- vector
c	- final vector

Definition at line 23 of file matvec.cpp.

13.26 matvec/omp-parallel/matvec.cpp File Reference

```
#include "matvec.h"
#include <omp.h>
```

Functions

- void mat_vec_mul (int rows, int cols, float *a, float *b, float *c)

13.26.1 Function Documentation

13.26.1.1 void mat_vec_mul (int *rows,* int *cols,* float * *a,* float * *b,* float * *c*)

Compute matrix and vector multiplication

Parameters

rows	- matrix rows
cols	- matrix cols
a	- matrix
b	- vector
c	- final vector

Definition at line 24 of file matvec.cpp.

13.27 matvec/omp-parallel-mic/matvec.cpp File Reference

```
#include "matvec.h"
#include <omp.h>
```

Functions

- void mat_vec_mul (int rows, int cols, float *a, float *b, float *c)

13.27.1 Function Documentation

13.27.1.1 void mat_vec_mul (int *rows*, int *cols*, float * *a*, float * *b*, float * *c*)

Compute matrix and vector multiplication

Parameters

rows	- matrix rows
cols	- matrix cols
a	- matrix
b	- vector
c	- final vector

Definition at line 24 of file matvec.cpp.

13.28 matvec/vec-padding/matvec.cpp File Reference

```
#include "matvec.h"
```

Functions

- void mat_vec_mul (int rows, int cols, float a[ ][cols], float b[cols], float c[rows])

13.28.1 Function Documentation

13.28.1.1 void mat_vec_mul (int *rows*, int *cols*, float *a[ ][cols]*, float *b[cols]*, float *c[rows]*)

Compute matrix and vector multiplication

Parameters

rows	- matrix rows
cols	- matrix cols
a	- matrix
b	- vector
c	- final vector

Definition at line 23 of file matvec.cpp.

13.29 matvec/dynamic-aligned/matvec.h File Reference

Functions

- void mat_vec_mul (int rows, int cols, float *a, float *b, float *c)

13.29.1 Function Documentation

13.29.1.1 void mat_vec_mul (int *rows*, int *cols*, float * *a*, float * *b*, float * *c*)

Compute matrix and vector multiplication

Parameters

rows	- matrix rows
cols	- matrix cols
a	- matrix
b	- vector
c	- final vector

Definition at line 23 of file matvec.cpp.

13.30 matvec/naive/matvec.h File Reference

Functions

- void mat_vec_mul (int rows, int cols, float a[ ][cols], float b[cols], float c[rows])

13.30.1 Function Documentation

13.30.1.1 void mat_vec_mul (int *rows*, int *cols*, float *a[ ][cols]*, float *b[cols]*, float *c[rows]*)

Compute matrix and vector multiplication

Parameters

rows	- matrix rows
cols	- matrix cols
a	- matrix
b	- vector
c	- final vector

Definition at line 23 of file matvec.cpp.

13.31 matvec/omp-parallel/matvec.h File Reference

Functions

- void mat_vec_mul (int rows, int cols, float ∗a, float ∗b, float ∗c)

13.31.1 Function Documentation

13.31.1.1 void mat_vec_mul (int *rows*, int *cols*, float ∗ *a*, float ∗ *b*, float ∗ *c*)

Compute matrix and vector multiplication

Parameters

rows	- matrix rows
cols	- matrix cols
a	- matrix
b	- vector

c	- final vector

Definition at line 23 of file matvec.cpp.

13.32 matvec/omp-parallel-mic/matvec.h File Reference

Functions

- void mat_vec_mul (int rows, int cols, float *a, float *b, float *c)

13.32.1 Function Documentation

13.32.1.1 void mat_vec_mul (int *rows*, int *cols*, float * *a*, float * *b*, float * *c*)

Compute matrix and vector multiplication

Parameters

rows	- matrix rows
cols	- matrix cols
a	- matrix
b	- vector
c	- final vector

Definition at line 23 of file matvec.cpp.

13.33 matvec/vec-padding/matvec.h File Reference

Functions

- void mat_vec_mul (int rows, int cols, float a[ ][cols], float b[cols], float c[rows])

13.33.1 Function Documentation

13.33.1.1 void mat_vec_mul (int *rows*, int *cols*, float *a[ ][cols]*, float *b[cols]*, float *c[rows]*)

Compute matrix and vector multiplication

Parameters

rows	- matrix rows
cols	- matrix cols
a	- matrix
b	- vector
c	- final vector

Definition at line 23 of file matvec.cpp.

13.34 nbody/cache-block-mic/gen.cpp File Reference

```
#include <cstdlib>
```

```
#include <cstdio>
#include <cfloat>
#include <ctime>
```

Functions

- float randf ()
- int main (int argc, char **argv)

13.34.1 Function Documentation

13.34.1.1 int main (int *argc,* char ** *argv*)

Main function

Definition at line 36 of file gen.cpp.

13.34.1.2 float randf ( )

Generate "random" float

Definition at line 23 of file gen.cpp.

13.35 nbody/naive/gen.cpp File Reference

```
#include <cstdlib>
#include <cstdio>
#include <cfloat>
#include <ctime>
```

Functions

- float randf ()
- int main (int argc, char **argv)

13.35.1 Function Documentation

13.35.1.1 int main (int *argc,* char ** *argv*)

Main function

Definition at line 36 of file gen.cpp.

13.35.1.2 float randf ( )

Generate "random" float

Definition at line 23 of file gen.cpp.

13.36 nbody/no-jump-auto-opt/gen.cpp File Reference

```
#include <cstdlib>
#include <cstdio>
#include <cfloat>
#include <ctime>
```

Functions

- float randf ()
- int main (int argc, char **argv)

13.36.1 Function Documentation

13.36.1.1 int main (int *argc,* char ** *argv*)

Main function

Definition at line 36 of file gen.cpp.

13.36.1.2 float randf ( )

Generate "random" float

Definition at line 23 of file gen.cpp.

13.37 nbody/offload/gen.cpp File Reference

```
#include <cstdlib>
#include <cstdio>
#include <cfloat>
#include <ctime>
```

Functions

- float randf ()
- int main (int argc, char **argv)

13.37.1 Function Documentation

13.37.1.1 int main (int *argc,* char ** *argv*)

Main function

Definition at line 36 of file gen.cpp.

13.37.1.2 float randf ( )

Generate "random" float

Definition at line 23 of file gen.cpp.

13.38 nbody/omp-parallel/gen.cpp File Reference

```
#include <cstdlib>
#include <cstdio>
#include <cfloat>
#include <ctime>
```

Functions

- float randf ()
- int main (int argc, char **argv)

13.38.1 Function Documentation

13.38.1.1 int main (int *argc,* char ** *argv*)

Main function

Definition at line 36 of file gen.cpp.

13.38.1.2 float randf ( )

Generate "random" float

Definition at line 23 of file gen.cpp.

13.39 nbody/omp-parallel-mic/gen.cpp File Reference

```
#include <cstdlib>
#include <cstdio>
#include <cfloat>
#include <ctime>
```

Functions

- float randf ()
- int main (int argc, char **argv)

13.39.1 Function Documentation

13.39.1.1 int main (int *argc,* char ** *argv*)

Main function

Definition at line 36 of file gen.cpp.

13.39.1.2 float randf ( )

Generate "random" float

Definition at line 23 of file gen.cpp.

13.40 nbody/cache-block-mic/nbody.cpp File Reference

```
#include <cfloat>
#include <cmath>
#include <omp.h>
#include <algorithm>
#include "nbody.h"
```

Functions

- t_particles_DA * particles_alloc (size_t size)
- void particles_free (t_particles_DA *p)
- void particles_init (t_particles_DA p)
- void particles_simulate (t_particles_DA p)
- void particles_read (FILE *fp, t_particles_DA p)
- void particles_write (FILE *fp, t_particles_DA p)

Variables

- const float SML_FLT = 1e-9f
- const int BLOCK = 512

13.40.1 Function Documentation

13.40.1.1 t_particles_DA* particles_alloc (size_t *size*)

Allocate memory for particles

Parameters

size	- number of particles to alloc

Returns

pointer to structure

Definition at line 29 of file nbody.cpp.

13.40.1.2 void particles_free (t_particles_DA * *p*)

Free memory

Parameters

p	- structure of particles

Definition at line 166 of file nbody.cpp.

13.40.1.3 void particles_init (t_particles_DA *p*)

Initialize structure of particles NUMA First Touch Policy

Parameters

p	- structure of particles

Definition at line 198 of file nbody.cpp.

13.40.1.4 void particles_read (FILE * *fp*, t_particles_DA *p*)

Read particles from file Fomat: pos_x pos_y pos_z vel_x vel_y vel_z weight

Parameters

fp	- input file descriptor
p	- structure of particles

Definition at line 334 of file nbody.cpp.

13.40.1.5 void particles_simulate (t_particles_DA *p*)

Simulate particle system

Parameters

p	- structure of particles

Definition at line 226 of file nbody.cpp.

13.40.1.6 void particles_write (FILE * *fp*, t_particles_DA *p*)

Write particles to file Fomat: pos_x pos_y pos_z vel_x vel_y vel_z weight

Parameters

fp	- output file descriptor
p	- structure of particles

Definition at line 355 of file nbody.cpp.

13.40.2 Variable Documentation

13.40.2.1 const int BLOCK = 512

Definition at line 20 of file nbody.cpp.

13.40.2.2 const float SML_FLT = 1e-9f

Definition at line 19 of file nbody.cpp.

13.41 nbody/naive/nbody.cpp File Reference

```
#include <cfloat>
#include <cmath>
#include "nbody.h"
```

Functions

- void particles_simulate (t_particles p)
- void particles_read (FILE *fp, t_particles p)
- void particles_write (FILE *fp, t_particles p)

13.41.1 Function Documentation

13.41.1.1 void particles_read (FILE * *fp*, t_particles *p*)

Read particles from file Fomat: pos_x pos_y pos_z vel_x vel_y vel_z weight

Parameters

fp	- input file descriptor
p	- structure of particles

Definition at line 92 of file nbody.cpp.

13.41.1.2 void particles_simulate (t_particles *p*)

Simulate particle system

Parameters

p	- structure of particles

Definition at line 21 of file nbody.cpp.

13.41.1.3 void particles_write (FILE * *fp*, t_particles *p*)

Write particles to file Fomat: pos_x pos_y pos_z vel_x vel_y vel_z weight

Parameters

fp	- output file descriptor
p	- structure of particles

Definition at line 113 of file nbody.cpp.

13.42 nbody/no-jump-auto-opt/nbody.cpp File Reference

```
#include <cfloat>
#include <cmath>
#include <omp.h>
#include <algorithm>
#include "nbody.h"
```

Macros

- #define SML_FLT 1e-9f

Functions

- t_particles_DA * particles_alloc (size_t size)

- void particles_free (t_particles_DA *p)
- void particles_init (t_particles_DA p)
- void particles_simulate (t_particles_DA p)
- void particles_read (FILE *fp, t_particles_DA p)
- void particles_write (FILE *fp, t_particles_DA p)

13.42.1 Macro Definition Documentation

13.42.1.1 #define SML_FLT 1e-9f

Definition at line 19 of file nbody.cpp.

13.42.2 Function Documentation

13.42.2.1 t_particles_DA* particles_alloc (size_t *size*)

Allocate memory for particles

Parameters

size	- number of particles to alloc

Returns

pointer to structure

Definition at line 28 of file nbody.cpp.

13.42.2.2 void particles_free (t_particles_DA * *p*)

Free memory

Parameters

p	- structure of particles

Definition at line 101 of file nbody.cpp.

13.42.2.3 void particles_init (t_particles_DA *p*)

Initialize structure of particles

Parameters

p	- structure of particles

Definition at line 126 of file nbody.cpp.

13.42.2.4 void particles_read (FILE * *fp,* t_particles_DA *p*)

Read particles from file Fomat: pos_x pos_y pos_z vel_x vel_y vel_z weight

Parameters

fp	- input file descriptor
p	- structure of particles

Definition at line 230 of file nbody.cpp.

13.42.2.5 void particles_simulate (t_particles_DA *p*)

Simulate particle system

Parameters

p	- structure of particles

Definition at line 147 of file nbody.cpp.

13.42.2.6 void particles_write (FILE * *fp*, t_particles_DA *p*)

Write particles to file Fomat: pos_x pos_y pos_z vel_x vel_y vel_z weight

Parameters

fp	- output file descriptor
p	- structure of particles

Definition at line 251 of file nbody.cpp.

13.43 nbody/offload/nbody.cpp File Reference

```
#include <cfloat>
#include <cmath>
#include <omp.h>
#include <algorithm>
#include "nbody.h"
```

Functions

- __attribute__ ((target(mic))) const float SML_FLT
- void particles_alloc (t_particles_DA *p, size_t size)
- void particles_free (t_particles_DA *p)
- void particles_read (FILE *fp, t_particles_DA p)
- void particles_write (FILE *fp, t_particles_DA p)

13.43.1 Function Documentation

13.43.1.1 __attribute__ ((target(mic))) const

Initialize structure of particles NUMA First Touch Policy

Parameters

p	- structure of particles

Simulate particle system

Parameters

p	- structure of particles

Definition at line 179 of file nbody.cpp.

13.43.1.2 void particles_alloc (t_particles_DA * p, size_t size)

Allocate memory for particles

Parameters

size	- number of particles to alloc

Returns

pointer to structure

Definition at line 28 of file nbody.cpp.

13.43.1.3 void particles_free (t_particles_DA * p)

Free memory

Parameters

p	- structure of particles

Definition at line 154 of file nbody.cpp.

13.43.1.4 void particles_read (FILE * fp, t_particles_DA p)

Read particles from file Fomat: pos_x pos_y pos_z vel_x vel_y vel_z weight

Parameters

fp	- input file descriptor
p	- structure of particles

Definition at line 308 of file nbody.cpp.

13.43.1.5 void particles_write (FILE * fp, t_particles_DA p)

Write particles to file Fomat: pos_x pos_y pos_z vel_x vel_y vel_z weight

Parameters

fp	- output file descriptor
p	- structure of particles

Definition at line 329 of file nbody.cpp.

13.44 nbody/omp-parallel/nbody.cpp File Reference

```
#include <cfloat>
#include <cmath>
#include <omp.h>
#include <algorithm>
#include "nbody.h"
```

Functions

- t_particles_DA ∗ particles_alloc (size_t size)
- void particles_free (t_particles_DA ∗p)
- void particles_init (t_particles_DA p)
- void particles_simulate (t_particles_DA p)
- void particles_read (FILE ∗fp, t_particles_DA p)
- void particles_write (FILE ∗fp, t_particles_DA p)

Variables

- const float SML_FLT = 1e-9f

13.44.1 Function Documentation

13.44.1.1 t_particles_DA∗ particles_alloc (size_t *size*)

Allocate memory for particles

Parameters

size	- number of particles to alloc

Returns

pointer to structure

Definition at line 28 of file nbody.cpp.

13.44.1.2 void particles_free (t_particles_DA ∗ *p*)

Free memory

Parameters

p	- structure of particles

Definition at line 165 of file nbody.cpp.

13.44.1.3 void particles_init (t_particles_DA *p*)

Initialize structure of particles NUMA First Touch Policy

Parameters

p	- structure of particles

Definition at line 197 of file nbody.cpp.

13.44.1.4 void particles_read (FILE ∗ *fp*, t_particles_DA *p*)

Read particles from file Fomat: pos_x pos_y pos_z vel_x vel_y vel_z weight

Parameters

fp	- input file descriptor
p	- structure of particles

Definition at line 329 of file nbody.cpp.

13.44.1.5 void particles_simulate (t_particles_DA *p*)

Simulate particle system

Parameters

p	- structure of particles

Definition at line 225 of file nbody.cpp.

13.44.1.6 void particles_write (FILE * *fp*, t_particles_DA *p*)

Write particles to file Fomat: pos_x pos_y pos_z vel_x vel_y vel_z weight

Parameters

fp	- output file descriptor
p	- structure of particles

Definition at line 350 of file nbody.cpp.

13.44.2 Variable Documentation

13.44.2.1 const float SML_FLT = 1e-9f

Definition at line 19 of file nbody.cpp.

13.45 nbody/omp-parallel-mic/nbody.cpp File Reference

```
#include <cfloat>
#include <cmath>
#include <omp.h>
#include <algorithm>
#include "nbody.h"
```

Functions

- t_particles_DA * particles_alloc (size_t size)
- void particles_free (t_particles_DA *p)
- void particles_init (t_particles_DA p)
- void particles_simulate (t_particles_DA p)
- void particles_read (FILE *fp, t_particles_DA p)
- void particles_write (FILE *fp, t_particles_DA p)

Variables

- const float SML_FLT = 1e-9f

13.45.1 Function Documentation

13.45.1.1 t_particles_DA* particles_alloc (size_t *size*)

Allocate memory for particles

Parameters

size	- number of particles to alloc

Returns

pointer to structure

Definition at line 28 of file nbody.cpp.

13.45.1.2 void particles_free (t_particles_DA * *p*)

Free memory

Parameters

p	- structure of particles

Definition at line 165 of file nbody.cpp.

13.45.1.3 void particles_init (t_particles_DA *p*)

Initialize structure of particles NUMA First Touch Policy

Parameters

p	- structure of particles

Definition at line 197 of file nbody.cpp.

13.45.1.4 void particles_read (FILE * *fp,* t_particles_DA *p*)

Read particles from file Fomat: pos_x pos_y pos_z vel_x vel_y vel_z weight

Parameters

fp	- input file descriptor
p	- structure of particles

Definition at line 328 of file nbody.cpp.

13.45.1.5 void particles_simulate (t_particles_DA *p*)

Simulate particle system

Parameters

p	- structure of particles

Definition at line 225 of file nbody.cpp.

13.45.1.6 void particles_write (FILE * *fp,* t_particles_DA *p*)

Write particles to file Fomat: pos_x pos_y pos_z vel_x vel_y vel_z weight

Parameters

fp	- output file descriptor
p	- structure of particles

Definition at line 349 of file nbody.cpp.

13.45.2 Variable Documentation

13.45.2.1 const float SML_FLT = 1e-9f

Definition at line 19 of file nbody.cpp.

13.46 nbody/cache-block-mic/nbody.h File Reference

```
#include <cstdlib>
#include <cstdio>
```

Classes

- struct t_particles_DA

Macros

- #define G 6.67384e-11f

Functions

- t_particles_DA * particles_alloc (size_t size)
- void particles_free (t_particles_DA *p)
- void particles_init (t_particles_DA p)
- void particles_simulate (t_particles_DA p)
- void particles_read (FILE *fp, t_particles_DA p)
- void particles_write (FILE *fp, t_particles_DA p)

13.46.1 Macro Definition Documentation

13.46.1.1 #define G 6.67384e-11f

Definition at line 20 of file nbody.h.

13.46.2 Function Documentation

13.46.2.1 t_particles_DA* particles_alloc (size_t *size*)

Allocate memory for particles

Parameters

size	- number of particles to alloc

Returns

pointer to structure

Definition at line 29 of file nbody.cpp.

13.46.2.2 void particles_free (t_particles_DA * p)

Free memory

Parameters

p	- structure of particles

Definition at line 166 of file nbody.cpp.

13.46.2.3 void particles_init (t_particles_DA p)

Initialize structure of particles

Parameters

p	- structure of particles

Initialize structure of particles NUMA First Touch Policy

Parameters

p	- structure of particles

Definition at line 198 of file nbody.cpp.

13.46.2.4 void particles_read (FILE * fp, t_particles_DA p)

Read particles from file Fomat: pos_x pos_y pos_z vel_x vel_y vel_z weight

Parameters

fp	- input file descriptor
p	- structure of particles

Definition at line 334 of file nbody.cpp.

13.46.2.5 void particles_simulate (t_particles_DA p)

Simulate particle system

Parameters

p	- structure of particles

Definition at line 226 of file nbody.cpp.

13.46.2.6 void particles_write (FILE * fp, t_particles_DA p)

Write particles to file Fomat: pos_x pos_y pos_z vel_x vel_y vel_z weight

Parameters

fp	- output file descriptor
p	- structure of particles

Definition at line 355 of file nbody.cpp.

13.47 nbody/naive/nbody.h File Reference

```
#include <cstdlib>
#include <cstdio>
```

Classes

- struct t_particle

Macros

- #define G 6.67384e-11f

Typedefs

- typedef t_particle t_particles[N]

Functions

- void particles_simulate (t_particles p)
- void particles_read (FILE *fp, t_particles p)
- void particles_write (FILE *fp, t_particles p)

13.47.1 Macro Definition Documentation

13.47.1.1 #define G 6.67384e-11f

Definition at line 19 of file nbody.h.

13.47.2 Typedef Documentation

13.47.2.1 typedef t_particle t_particles[N]

Definition at line 34 of file nbody.h.

13.47.3 Function Documentation

13.47.3.1 void particles_read (FILE * *fp,* t_particles *p*)

Read particles from file Fomat: pos_x pos_y pos_z vel_x vel_y vel_z weight

Parameters

fp	- input file descriptor
p	- structure of particles

Definition at line 92 of file nbody.cpp.

13.47.3.2 void particles_simulate (t_particles *p*)

Simulate particle system

Parameters

p	- structure of particles

Definition at line 21 of file nbody.cpp.

13.47.3.3 void particles_write (FILE * *fp*, t_particles *p*)

Write particles to file Fomat: pos_x pos_y pos_z vel_x vel_y vel_z weight

Parameters

fp	- output file descriptor
p	- structure of particles

Definition at line 113 of file nbody.cpp.

13.48 nbody/no-jump-auto-opt/nbody.h File Reference

```
#include <cstdlib>
#include <cstdio>
```

Classes

- struct t_particles_DA

Macros

- #define G 6.67384e-11f

Functions

- t_particles_DA * particles_alloc (size_t size)
- void particles_free (t_particles_DA *p)
- void particles_init (t_particles_DA p)
- void particles_simulate (t_particles_DA p)
- void particles_read (FILE *fp, t_particles_DA p)
- void particles_write (FILE *fp, t_particles_DA p)

13.48.1 Macro Definition Documentation

13.48.1.1 #define G 6.67384e-11f

Definition at line 20 of file nbody.h.

13.48.2 Function Documentation

13.48.2.1 t_particles_DA* particles_alloc (size_t *size*)

Allocate memory for particles

Parameters

size	- number of particles to alloc

Returns

pointer to structure

Definition at line 29 of file nbody.cpp.

13.48.2.2 void particles_free (t_particles_DA * *p*)

Free memory

Parameters

p	- structure of particles

Definition at line 166 of file nbody.cpp.

13.48.2.3 void particles_init (t_particles_DA *p*)

Initialize structure of particles

Parameters

p	- structure of particles

Initialize structure of particles NUMA First Touch Policy

Parameters

p	- structure of particles

Definition at line 198 of file nbody.cpp.

13.48.2.4 void particles_read (FILE * *fp,* t_particles_DA *p*)

Read particles from file Fomat: pos_x pos_y pos_z vel_x vel_y vel_z weight

Parameters

fp	- input file descriptor
p	- structure of particles

Definition at line 334 of file nbody.cpp.

13.48.2.5 void particles_simulate (t_particles_DA *p*)

Simulate particle system

Parameters

p	- structure of particles

Definition at line 226 of file nbody.cpp.

13.48.2.6 void particles_write (FILE * *fp*, t_particles_DA *p*)

Write particles to file Fomat: pos_x pos_y pos_z vel_x vel_y vel_z weight

Parameters

fp	- output file descriptor
p	- structure of particles

Definition at line 355 of file nbody.cpp.

13.49 nbody/offload/nbody.h File Reference

```
#include <cstdlib>
#include <cstdio>
```

Classes

- struct t_particles_DA

Macros

- #define G 6.67384e-11f

Functions

- void particles_alloc (t_particles_DA *p, size_t size)
- void particles_free (t_particles_DA *p)
- __attribute__ ((target(mic))) void particles_init(t_particles_DA p)
- void particles_read (FILE *fp, t_particles_DA p)
- void particles_write (FILE *fp, t_particles_DA p)

13.49.1 Macro Definition Documentation

13.49.1.1 #define G 6.67384e-11f

Definition at line 20 of file nbody.h.

13.49.2 Function Documentation

13.49.2.1 __attribute__ ((target(mic)))

Initialize structure of particles

Parameters

p	- structure of particles

Simulate particle system

Parameters

p	- structure of particles

Initialize structure of particles NUMA First Touch Policy

Parameters

p	- structure of particles

Simulate particle system

Parameters

p	- structure of particles

Definition at line 179 of file nbody.cpp.

13.49.2.2 void particles_alloc (t_particles_DA * *p*, size_t *size*)

Allocate memory for particles

Parameters

size	- number of particles to alloc

Returns

pointer to structure

Definition at line 28 of file nbody.cpp.

13.49.2.3 void particles_free (t_particles_DA * *p*)

Free memory

Parameters

p	- structure of particles

Definition at line 166 of file nbody.cpp.

13.49.2.4 void particles_read (FILE * *fp*, t_particles_DA *p*)

Read particles from file Fomat: pos_x pos_y pos_z vel_x vel_y vel_z weight

Parameters

fp	- input file descriptor
p	- structure of particles

Definition at line 334 of file nbody.cpp.

13.49.2.5 void particles_write (FILE * *fp*, t_particles_DA *p*)

Write particles to file Fomat: pos_x pos_y pos_z vel_x vel_y vel_z weight

Parameters

fp	- output file descriptor
p	- structure of particles

Definition at line 355 of file nbody.cpp.

13.50 nbody/omp-parallel/nbody.h File Reference

```
#include <cstdlib>
#include <cstdio>
```

Classes

- struct t_particles_DA

Macros

- #define G 6.67384e-11f

Functions

- t_particles_DA * particles_alloc (size_t size)
- void particles_free (t_particles_DA *p)
- void particles_init (t_particles_DA p)
- void particles_simulate (t_particles_DA p)
- void particles_read (FILE *fp, t_particles_DA p)
- void particles_write (FILE *fp, t_particles_DA p)

13.50.1 Macro Definition Documentation

13.50.1.1 #define G 6.67384e-11f

Definition at line 20 of file nbody.h.

13.50.2 Function Documentation

13.50.2.1 t_particles_DA* particles_alloc (size_t *size*)

Allocate memory for particles

Parameters

size	- number of particles to alloc

Returns

pointer to structure

Definition at line 29 of file nbody.cpp.

13.50.2.2 void particles_free (t_particles_DA * p)

Free memory

Parameters

p	- structure of particles

Definition at line 166 of file nbody.cpp.

13.50.2.3 void particles_init (t_particles_DA *p*)

Initialize structure of particles

Parameters

p	- structure of particles

Initialize structure of particles NUMA First Touch Policy

Parameters

p	- structure of particles

Definition at line 198 of file nbody.cpp.

13.50.2.4 void particles_read (FILE * *fp,* t_particles_DA *p*)

Read particles from file Fomat: pos_x pos_y pos_z vel_x vel_y vel_z weight

Parameters

fp	- input file descriptor
p	- structure of particles

Definition at line 334 of file nbody.cpp.

13.50.2.5 void particles_simulate (t_particles_DA *p*)

Simulate particle system

Parameters

p	- structure of particles

Definition at line 226 of file nbody.cpp.

13.50.2.6 void particles_write (FILE * *fp,* t_particles_DA *p*)

Write particles to file Fomat: pos_x pos_y pos_z vel_x vel_y vel_z weight

Parameters

fp	- output file descriptor
p	- structure of particles

Definition at line 355 of file nbody.cpp.

13.51 nbody/omp-parallel-mic/nbody.h File Reference

```
#include <cstdlib>
#include <cstdio>
```

Classes

- struct t_particles_DA

Macros

- #define G 6.67384e-11f

Functions

- t_particles_DA ∗ particles_alloc (size_t size)
- void particles_free (t_particles_DA ∗p)
- void particles_init (t_particles_DA p)
- void particles_simulate (t_particles_DA p)
- void particles_read (FILE ∗fp, t_particles_DA p)
- void particles_write (FILE ∗fp, t_particles_DA p)

13.51.1 Macro Definition Documentation

13.51.1.1 #define G 6.67384e-11f

Definition at line 20 of file nbody.h.

13.51.2 Function Documentation

13.51.2.1 t_particles_DA∗ particles_alloc (size_t *size*)

Allocate memory for particles

Parameters

size	- number of particles to alloc

Returns

pointer to structure

Definition at line 29 of file nbody.cpp.

13.51.2.2 void particles_free (t_particles_DA ∗ *p*)

Free memory

Parameters

p	- structure of particles

Definition at line 166 of file nbody.cpp.

13.51.2.3 void particles_init (t_particles_DA *p*)

Initialize structure of particles

Parameters

p	- structure of particles

Initialize structure of particles NUMA First Touch Policy

Parameters

p	- structure of particles

Definition at line 198 of file nbody.cpp.

13.51.2.4 void particles_read (FILE * *fp*, t_particles_DA *p*)

Read particles from file Fomat: pos_x pos_y pos_z vel_x vel_y vel_z weight

Parameters

fp	- input file descriptor
p	- structure of particles

Definition at line 334 of file nbody.cpp.

13.51.2.5 void particles_simulate (t_particles_DA *p*)

Simulate particle system

Parameters

p	- structure of particles

Definition at line 226 of file nbody.cpp.

13.51.2.6 void particles_write (FILE * *fp*, t_particles_DA *p*)

Write particles to file Fomat: pos_x pos_y pos_z vel_x vel_y vel_z weight

Parameters

fp	- output file descriptor
p	- structure of particles

Definition at line 355 of file nbody.cpp.

13.52 python-mkl/numpy_dgemm.py File Reference

Namespaces

- numpy_dgemm

Variables

- tuple numpy_dgemm.K = int(sys.argv[1])
- numpy_dgemm.M = K
- numpy_dgemm.N = K
- int numpy_dgemm.ITER = 1
- tuple numpy_dgemm.t_init = time.time()
- tuple numpy_dgemm.a1 = np.array(np.random.random((M, K)), dtype=np.double, order='C', copy=False)
- tuple numpy_dgemm.a2 = np.array(np.random.random((K, N)), dtype=np.double, order='C', copy=False)

- tuple numpy_dgemm.m1 = np.matrix(a1, dtype=np.double, copy=False)
- tuple numpy_dgemm.m2 = np.matrix(a2, dtype=np.double, copy=False)
- tuple numpy_dgemm.t_start = time.time()
- tuple numpy_dgemm.mf2 = np.dot(m1, m2)
- tuple numpy_dgemm.t_end = time.time()

13.53 python-mkl/scipy_dgemm.py File Reference

Namespaces

- scipy_dgemm

Variables

- tuple scipy_dgemm.K = int(sys.argv[1])
- scipy_dgemm.M = K
- scipy_dgemm.N = K
- int scipy_dgemm.ITER = 1
- tuple scipy_dgemm.a1 = np.array(np.random.random((M, K)), dtype=np.double, order='C', copy=False)
- tuple scipy_dgemm.a2 = np.array(np.random.random((K, N)), dtype=np.double, order='C', copy=False)
- tuple scipy_dgemm.m1 = np.matrix(a1, dtype=np.double, copy=False)
- tuple scipy_dgemm.m2 = np.matrix(a2, dtype=np.double, copy=False)
- tuple scipy_dgemm.t_start = time.time()
- tuple scipy_dgemm.mf2 = sp.dgemm(alpha=1.0, a=m1, b=m2)
- tuple scipy_dgemm.t_end = time.time()

Index