**2D Heat conduction problem:**

This tutorial is based on hands on tutorial from UPC. This tutorial guides through transformation of heat\_seq.c to HPX code and how we can get the performance boost from HPX.

1. Mathematical background

For heat conduction, the fundamental 2D Partial Differential Equation (1) in a stationary medium is:



where T is temperature, t is time, *α* is the thermal diffusivity.

We assumed that the source does not generate heat and the thermal properties are constant. In addition, we also assumed a solid surface with α = 1.0.

(1): *Modeling of Laser Machining Process, chapter 3.5*, Combined Research and Curriculum Development Nontraditional Manufacturing (NTM), Columbia University

(http://www.columbia.edu/cu/mechanical/mrl/ntm/pgIndex.html)

1. Finite Difference Method overview

Numerical approximation of the heat conduction over a solid plate is done using a finite difference method. Thus, a computational grid is mapped onto the surface.

N

N

Solid

The grid has a total dimension of (N+2)\*(N+2). The temperatures at the boundaries are fixed at initialization time.

So, for an interior point of the 2D grid, the temperature is calculated as follows:



Illustration of this stencil operation:



1. The sequential C program

The following sequential program (heat\_seq.c) is presented below:

The program heats one of the sides as the initial condition, and iterates to study how the heat transfers across the surface.

//heat\_seq.c

#include <stdio.h>

#include <math.h>

#include <sys/time.h>

#define N 30

double grid[N+2][N+2], new\_grid[N+2][N+2];

void initialize(void)

{

int j;

/\* Heat one side of the solid \*/

for( j=1; j<N+1; j++ )

{

grid[0][j] = new\_grid[0][j] = 1.0;

}

}

int main(void)

{

struct timeval ts\_st, ts\_end;

double dTmax, dT, epsilon, time;

int finished, i, j, k, l;

double T;

int nr\_iter;

initialize(); /\*set initial conditions\*/

*/\* Set the precision wanted \*/*

epsilon = 0.0001;

finished = 0;

nr\_iter = 0;

*/\* Start timing \*/*

gettimeofday( &ts\_st, NULL );

do

{

dTmax = 0.0;

for( i=1; i<N+1; i++ )

{

for( j=1; j<N+1; j++ )

{

T = 0.25 \*

(grid[i+1][j] + grid[i-1][j] +

grid[i][j-1] + grid[i][j+1]); */\* stencil \*/*

dT = T – grid[i][j]; */\* local variation \*/*

new\_grid[i][j] = T;

if( dTmax < fabs(dT) )

dTmax = fabs(dT); */\* max variation in this iteration \*/*

}

}

if( dTmax < epsilon ) */\* is the precision reached good enough ? \*/*

finished = 1;

else

{

for( k=0; k<N+2; k++ ) */\* Prepare for a new iteration\*/*

for( l=0; l<N+2; l++ )

grid[k][l] = new\_grid[k][l];

}

nr\_iter++;

} while( finished == 0 );

gettimeofday( &ts\_end, NULL ); */\* end the timed section \*/*

*/\* compute the execution time \*/*

time = ts\_end.tv\_sec + (ts\_end.tv\_usec / 1000000.0);

time -= ts\_st.tv\_sec + (ts\_st.tv\_usec / 1000000.0);

printf(“%d iterations in %.3lf sec\n”, nr\_iter,

time ); */\* and prints it \*/*

return 0;

}