Masterpraktikum Scientific Computing – High Performance Computing

Gaurav Kukreja, Evangelos Drossos

Exercise Sheet 1

Exercise 1

1.a

Following loops can be vectorized using the Intel Compiler

- Number of iterations known beforehand : When the number of iterations (or, upper iteration limit) of the loop is known beforehand. In the case of nested loops, the number of iterations should be a function of the index of the outer loop.
- Straight Code : The loop should not contain branch instructions like switch, if, function calls or return statements. Math library functions and inline functions are an exception.
- Should not have Data Dependence :The loops should not have data dependency, especially True Dependence (Read after Write)
- Should not use pointers : Use of pointers makes analysis of data dependency difficult for the compiler.
- Stride length should preferably be 1.

1.b

- 32 bit, 16 bit and 8 bit data types are supported. Limited support for 64 bit data types.
- 32 bit Single precision floating point and 64 bit double precision floating points, but only for certain operations, viz. +, -, *, /
- It is preferable to use Structure of Arrays instead of Array of structures for efficient memory access, but this is not mandatory.
- The statements inside the innermost loop must only be arithmetic operations and bit operations. Some functionality provided by Math libraries can be used. Inline functions can be used but there are exceptions.

1.c

Intel Compiler performs following dependency analysis

- Flow Dependency : When operations in different iterations of the loop may access the same memory location. Checks for true, anti and output dependencies.
- When pointers are used checking if the aliases are accessing the same memory location.
- Also in case of arrays, it tries to analyze the relations between subscripts to check for memory access overlap.

1.d

Programming style can influence auto-vectorization

- Use of simple for loop, with invariant limits and no special termination conditions.
- Unrolling of loops by programmers will lead to a stride length more than 1. The compiler may not be able to vectorize such loops. Therefore, unrolling of loops should not be done, because the compiler will automatically perform this optimization.
- Array access should use the iteration index directly for subscripts.
- Array notation is preferable to use of pointers, because it helps compiler in dependence analysis.
- Structure of arrays is preferable to Array of structure, because it leads to efficient memory access.
- 32 bit or 16 bit data types should be preferred instead of 64 bit when possible.
- Minimize indirect addressing.

1.e

- We can force the compiler to vectorize code which contains indirect addressing, when we are sure that there is no data dependency.

1.f

- Loop Blocking : Compiler divides the loop into smaller chunks to efficiently utilize the Cache.
- Loop Interchanging : eg. In case of Matrix Multiplication, the compiler can interchange the inner and outer loop for improving memory access patterns.