Optimalizace pomocí icc/gcc - vektorizace

ICC/ICPC

ICC/ICPC - překladače pro jazyky C/C++ od firmy Intel

- ke stažení po registraci na
  http://www.intel.com/cd/software/products/asmo-na/eng/compilers/clin/219856.htm
  google: icc intel download
- podpora vektorizace a OpenMP

GCC

GCC/G++ - GNU překladače pro jazyky C/C++

- www.gcc.gnu.org
- http://en.wikipedia.org/wiki/GNU_Compiler_Collection
- podpora vektorizace, podpora OpenMP ve vývoji

Některá nastavení pro ICC/ICPC

- `-O0--O3` - stupeň optimalizace
- `-mp` - zabraňuje optimalizacím na úkor přesnosti (úlohy s chaotickým chováním)
- `-IPF_fp_speculation[fast,safe,strict,off]` - předvídání výsledků operací s desitinnými čísly
- `-tpp[1,2,5,6,7]` - architektura - itanium1,2, pentium, pentiumpro,pentium4
- `-ipo` - interprocedural optimizations

Rozbalování smyček - loop unrolling

Místo cyklu

```
for( i = 0; i < 3; i ++ )
   a[ i ] = b[ i ] + c[ i ];
```

je efektivnější provést

```
a[ 0 ] = b[ 0 ] + c[ 0 ];
a[ 1 ] = b[ 1 ] + c[ 1 ];
a[ 2 ] = b[ 2 ] + c[ 2 ];
```

Rozbalování smyček - loop unrolling

Místo cyklu

```
for( i = 0; i < 3; i ++ )
   a[ i ] = b[ i ] + c[ i ];
```

je efektivnější provést

```
a[ 0 ] = b[ 0 ] + c[ 0 ];
a[ 1 ] = b[ 1 ] + c[ 1 ];
a[ 2 ] = b[ 2 ] + c[ 2 ];
```

- odstraňuje podmínku `i < 3` a umožní efektivněji využít pipline v CPU.
- automaticky toho lze dosáhnout pomocí tzv. rozbalovaní smyček.
- zapíná se pomocí `-unroll[n]` - určuje, kolik cyklů se má rozbalovat (přepínač ICC)

Načítání dat dopředu - prefetching

- zapíná se pomocí `-prefetch -O3`
- umožní načíst do cache celý blok dat
- provádí se pomocí direktivy `pragma`

Načítání dat dopředu - prefetching

- zapíná se pomocí `-prefetch -O3`
- umožní načíst do cache celý blok dat
- provádí se pomocí direktivy `pragma`

Vzor

```
#pragma variable:hint:distance
```

Příklad (načtení pole `x[80]`)

```
#pragma x:1:80
```

Vektorizace

Vektorizace = vyčíslení algebraického výrazu pro celé pole dat v jednom kroku (SIMD)

- zpíná se pomocí
  - `-ax[K,W,N,B,P]` - generuje obecný i optimalizovaný kód
  - `-x[K,W,N,B,P]` - generuje pouze optimalizovaný kód (nelze použít na slabších architekturách)
  - K - PentiumIII, W - Pentium4, B - PentiumM, P - SSE3
- `-vec_report[0-3]` - vypisuje informace o úspěšném/neuspěšném provedení vektorizace

Pravidla pro vektorizaci smyček

Příklad

```
for( i = 0; i < 1000; i ++ )
    a[ i ] = b[ i ] + c[ i ];
```

Aby mohl překladač úspěšně provést vektorizaci nesmí smyčka obsahovat

- volání funkce (z podstaty SIMD)
- datovou závislost

```
for( i = 1; i < 999; i ++ )
    a[ i ] = ( a[ i - 1 ] - 2.0 * a[ i ] +
                a[ i + 1 ] ) / ( h * h );
```

Pravidla pro vektorizaci smyček

- datově závislé opuštění smyčky

```
while( i < 100 ){
    if( a[ i ] == 0.0 ) break;
    a[ i ] = b[ i ] + c[ i ]; }
```

  Lze ale vektorizovat

```
while( i < 100 ){
    a[ i ] = b[ i ] + c[ i ];
    if( a[ i ] < 0.0 ) a[ i ] = 0.0;}
```

- počet smyček nesmí záviset na vnitřku cyklu

```
while( a[ i ] < 0.0 ) a[ i ] = 0.0;
```

Pravidla pro vektorizaci smyček

Příklad:

```
for( i = 0; i < 100; i ++ )
      a[ i ] = c * a[ i + k ];
```

Nelze vektorizovat pro k < 0, překladač vektorizaci neprovede.
Pokud programátor vi, že není k < 0, může použít direktivu

```
#pragma ivdep
```

Pravidla pro vektorizaci smyček

Podobný příklad

```
double *a = & b[ 2 ];
for( i = 0; i < 1000; i ++ )
    a[ i ] = b[ i ] + c[ i ];
```

Pokud překladač nemá jistotu, zda se dvě pole nepřekrývají, vektorizaci neprovede.

exflib - výpočty se zvýšenou přesností

`http://www-an.acs.i.kyoto-u.ac.jp/~fujiwara/exflib/`

Exflib is a simple software for scientific multiple-precision arithmetic for C++ and Fortran 90/95.

- Basic arithmetic and comparisons
- Input/Output in decimal
- Basic mathematical functions
- Parallel computation with MPI, OpenMP
- C++ class implementation
  - Overloaded operators, functions
  - Rounding Control, Interval Arithmetic (experimental, not available in Alpha version)
  - Useful numerical routines (Bessel, matrix, LU, QR, etc).
- Fortran90 Module implementation
  - Operators, functions with INTERFACE