Abstract:
In the last years, there has been much effort in commercial compilers (icc, gcc) to exploit efficiently the SIMD capabilities and the memory hierarchy that the current processors offer. However, the small numbers of compilers that can automatically exploit these characteristics achieve in most cases unsatisfactory results. Therefore, the programmers often need to apply by hand the optimizations to the source code, write manually the code in assembly or use compiler built-in functions (such intrinsics) to achieve high performance. In this work, we present source-to-source transformations that help commercial compilers exploiting the memory hierarchy and generating efficient SIMD code which can be applied in an automated way. Results obtained on our experiments show that our solutions achieve as excellent performance as hand-optimized vendor-supplied numerical libraries (written in assembly). Our source-to-source transformations are based on the tiling, strip-mining, scalar replacement and unroll and jam transformations. In particular, we apply these transformations to loop nests and show their effectiveness; the tiling transformation permits us to exploit the reuse at the register bank, the strip-mining transformation helps us applying outer loop vectorization, the unroll and jam transformation permits unrolling vectorized outer loops and finally, the scalar replacement concept is applied to vectorized loops to obtain what we call vector replacement (scalar replacement applied to vector registers). We have compared the performance obtained with these transformations against what MKL and ATLAS get and concluded that it is possible to achieve high performance in numerical applications applying only source-to-source transformations and letting the compiler to do the low-level optimization work.
**Introduction**

Comercial compilers do not fully exploit the new characteristics of the current processors. To achieve high performance compilers should:
1. Generate efficient SIMD code
2. Exploit efficiently the memory hierarchy

Source to source transformations help the compiler to generate efficient SIMD code.

**Objective**

Perform the following transformations to matrix product:
- Tiling at the register level
- Outer Loop Vectorization

**icc behaviour**

we observe that icc:
- only performs inner loop vectorization
- does not unroll strip-mined vectorized loops
- does not apply vector replacement* (VR) in unrolled loop body
- identification of adjacent array references with pointer variables to expose vector register reuse

**Matrix product**

**icc does not apply tiling at register level**

Apply explicidy register tiling

**icc does not vectorize outer loops**

Expose the vector statement*

Unroll & jam the strip-mined loop

**icc does not apply vector replacement**

Identify individual array references with pointers*

To achieve high performance compilers should:
1. Generate efficient SIMD code
2. Exploit efficiently the memory hierarchy

Source to source transformations help the compiler to generate efficient SIMD code.

**References**


This research has been supported by an ICREA-UPC Research Grant to the Spanish Ministry of Education (contract nos. TIN2004-02045 and TIN2004-02046), and the European Union (under the HPRN-CT-2000-00136).