発表概要

共有メモリプログラミングのための拡張C言語

高田 潤[†] 八杉昌宏[†]
小宮常康[†] 湯淺太一[†]

従来の並列プログラミングでは，共有メモリに対する不可分操作やメモリアクセス完了順序について，専用ライブラリの呼び出しや asm 文などを使用してきた．しかしそれらを用いると，コードの移植性やライブラリ呼び出しのオーバヘッド，コンパイラによる最適化や型チェックを妨げるなどの問題があった．本研究では，C 言語を拡張し，共有メモリ向け機能を追加した．実装にはコンパイラとして GCC を用いた．オーバヘッドを減らし，最適化を最大限に生かすためには GCC の持つ RTL を拡張するのが望ましいが，それには GCC のすべての最適化器に手を入れる必要があり非常に繁雑である．したがって本研究では既存の RTL をうまく利用することで GCC が持つ最適化器自体には触れないように実装することにした．これにより最適化は GCC の既存部分に任せることができる．一方で，メモリアクセス完了順序を保証する場合など，最適化してはならない部分も存在する．それらを考慮に入れたうえで，最適化を最大限に生かすようにした．また本発表では，実装した拡張 C 言語を用いて，共有メモリに関するいくつかのプログラミング例について紹介する．

An Extended C Language for Shared-memory Programming

Jun Takada,[†] Masahiro Yasugi,[†] Tsuneyasu Komiya[†]
and Taiichi Yuasa[†]

In parallel programming in C, we often use library calls or asm statements for atomic memory operations and constraints on memory access order. However, asm statements reduce portability and the overhead of library calls reduce performance. Furthermore, they often invalidate optimizations and the type checking of the compiler. So, we extended the C language with shared-memory functionalities. The C compiler we implemented is based on GCC. In order to minimize the overhead and maximize the effect of optimizations, it is ideal to extend the RTL in GCC. However, it is very costly because we must extend all optimizers in GCC. In this research, we effectively used the combination of the original RTL without extension. The optimization process is left to the original optimizers of GCC because we did not extend the optimizers. However, some code should not be optimized by the compiler, in particularly, the code for constrains on memory access order. Our implementation can maximize the effect of optimization even if the program contains such code. We also present some examples of shared-memory programming in our extended C language.

（平成 15 年 1 月 24 日発表）

† 京都大学大学院情報学研究科通信情報システム専攻
Department of Communications and Computer Engineering, Graduate School of Informatics, Kyoto University