発表概要

細粒度マルチスレッド言語における例外処理の効率良い実装

庄林宏和[†] 馬谷誠二[†] 八杉昌宏[†]
小宮常康[†,††] 湯淺太一[†]

本発表では，細粒度マルチスレッド言語における例外処理を効率良くポータブルに実装するための手法について提案し，性能評価を行う．我々は，動的スコープを利用し，同期処理・例外処理を階層的に構造化したオブジェクト指向並列言語 OPA を実装している．OPA コンパイラは OPA のソースコードを C コードに変換することで，高いポータビリティを実現している．また，値ベースのサスペンドチェックなどの実装手法を用いている．本提案では例外のチェックをサスペンドチェックと同時に行うことで，通常処理時の例外チェックのオーバヘッドをなくした．また `try-catch`，`try-finally` 構文を C 言語によって実装する手法についても提案する．さらに，細粒度マルチスレッド言語においては，多数のスレッドが生成され，fork-join のネストが深くなる．並列実行中の例外に対しては，仕事を分担した全スレッドの実行が中断または完了するのを待つのが望ましい．中断可能なスレッドをなるべく早く中断することができれば，無駄な計算が行われず，処理全体の効率が上がる．本発表では，そのための手法についても提案する．また，OPA 処理系には遅延タスク生成（LTC）が実装されているが，このような LTC が導入された処理系における実装上の工夫についても述べる．

An Efficient Implementation of Exception Handling for Fine-grained Multithreaded Languages

Hirokazu Shobayashi,[†] Seiji Umatani,[†] Masahiro Yasugi,[†]
Tsuneyasu Komiya[†,††] and Taiichi Yuasa[†]

In this presentation we propose an efficient and portable implementation scheme of exception handling for fine-grained multi-threaded programming languages, and evaluate its performance. We are implementing an object-oriented parallel language OPA which features hierarchically structured synchronization and exception handling using dynamic scope. OPA compiler translates source code in OPA into C code for portability. The OPA system employs a value-based suspension check method, and overhead for exception checks can be eliminated by unifying suspension checks and exception checks. We also propose C code generation techniques for implementing the `try-catch` construct and the `try-finally` construct. In fine-grained multi-threaded programs, a lot of threads are created, and the nest of fork-join becomes deeper. Before handling an exception thrown in the course of parallel execution, it is desired to wait for all threads sharing the goal of the parallel execution to abort or finish their execution. If we can abort execution of threads that can be aborted as soon as possible, useless computation is not performed and the total efficiency is improved. In this presentation we propose techniques for such case. Since OPA is implemented using Lazy Task Creation (LTC), we also describe the implementation issues on such language systems.

（平成 16 年 1 月 20 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
†† 豊橋技術科学大学情報工学系
Department of Information and Computer Sciences, Toyohashi University of Technology