**発表概要**

# マルチコア向け並列プログラミングモデルの設計と実装

境　　隆二[†1]　加藤宣弘[†1]
保科　　聡[†2]　島田智文[†1]

本発表では，マルチコア向け並列プログラムの実装方法として，多くの非永続スレッドをコンピュータの命令語と見なし，コンパイラの最適化やスーパスケーラにおける命令レベル並列処理のメカニズムをスレッドの並列実行に適用するというプログラミングモデルと実行モデルを提案する．アルゴリズムを直列に同期なしで実行する複数の直列基本モジュールと，これらのモジュールの並列実行を制御するための並列動作記述に分割して実装する．直列基本モジュールは，C 言語などによって実装して高い性能を確保する．並列動作記述は，直列基本モジュールを関数として定義し，関数の引数と返り値によってデータの入出力を表現する．実行順序は，この記述における直列基本モジュールの［定義—参照］による依存関係のみによって決定される．並列動作記述は，トランスレータによってランタイムが利用するグラフ生成情報に変換される．ランタイム環境は，利用するコアの数のネイティブスレッドを生成し，これらのネイティブスレッドが順番交代でグラフ構造を生成，アップデートしながら，次に実行するべき直列基本モジュールを決定して，これを呼び出す．このランタイム処理を「スレッドインタプリタ」と呼ぶ．スレッドインタプリタの最初の単純な実装は，ランタイム処理を排他的に実行するものである．直列基本モジュールのパラメータやアルゴリズムを，入力データの性質による計算量の増減やシステムリソース使用量の変動に，動的に適応させる「オートチューニング機能」をスレッドインタプリタに組み込み，高い計算効率と安定したリアルタイム性を実現した．

# Design and Implementation of a Programming Model for Multi Core

Ryuji Sakai,[†1] Nobuhiro Kato,[†1] Satoshi Hoshina[†2]
and Tomofumi Shimada[†1]

In this presentation, we propose a programming model for implementing a parallel program on multi-core. This model interprets a non-persistent thread called sequential basic module as a computer instruction and applies the instruction level parallel execution mechanism to the thread level parallel execution. In this model, an algorithm is divided into the sequential basic module which run sequentially without internal synchronization, and the parallel execution description which control parallel execution of these modules. The sequential basic module is implemented by C language to ensure high performance. In a parallel execution description, the sequential basics modules are defined as a function and data input and output of each module is expressed by arguments and return value of the function respectively. An execution order of basic module is determined only by data flow dependency between the sequential basic modules. The parallel execution description is converted into the graph generation data structure of C language by the translator. The runtime environment generates the native threads for the number of cores to use at first. Then these native threads generate, update and select the graph data structure one after another and start to run the selected module. This runtime environment is called 'Thread Interpreter', and the first simple implementation of 'Thread Interpreter' is that the runtime process is executed by multiple native threads exclusively. The 'auto-tuning function' which adapts parameters and algorithms of the sequential basic modules to a change of computational complexity by the property of input data and/or to a change of the system resource consumption. This is embedded in Thread Interpreter which results in the high calculation efficiency and stable realtimeness.

（平成 20 年 3 月 18 日発表）

†1 株式会社東芝デジタルメディアネットワーク社コアテクノロジーセンター
Core Technology Center, Digital Media Network Company, Toshiba Corporation
†2 株式会社東芝 PC&ネットワーク社 PC 開発センター
PC Development Center, PC & Network Company, Toshiba Corporation