発表概要

指定されたメソッドの呼び出し中でオブジェクトのメモリレイアウトを整列化できるグラフ解析向け Javaコンパイラ

汐田 徹也[1,a)]　佐藤 芳樹[2]　千葉 滋[1]

2014年3月18日発表

指定されたメソッドの呼び出し中でオブジェクトのメモリレイアウトを整列化できるグラフ解析向けJavaコンパイラを提案する．一般に，オブジェクト指向で表現されたグラフ構造は，各頂点や各辺が持つ多様な属性がオブジェクトとして抽象化されるため，グラフ全体を探索しながら属性を使って計算するような処理で高いキャッシュ効率が期待できない．キャッシュ効率を高めるために，連続してアクセスするデータをメモリ上に整列させるテクニックはよく知られているが，Java言語ではユーザがオブジェクトの整列順序による性能への影響を意識しながら開発することは難しい．そこで本研究では，プログラマの付加したアノテーションに基づくプログラム変換で，指定されたフィールドの配列化やグラフ探索順序に沿ったメモリレイアウト整列化を実現するJavaコンパイラを開発した．コンパイラは，フィールド配列化のためにクラス定義およびオブジェクトアクセスを静的に変更し，フィールドをアクセス順序に従った頂点や辺オブジェクトを再生成するコードを埋め込む．また，グラフオブジェクトを，そのIDを保持するポインタオブジェクトと，それ以外のフィールドを格納するオブジェクトに分割することで，高速なメモリレイアウト整列化およびその解放を達成しつつ，ポインタオブジェクト導入による間接参照のオーバヘッドもできる限り軽減した．典型的なグラフ解析としてダイクストラ法を用いた実験から，最大で14倍程度の性能向上が見込めることを確認した．

A Java Compiler Which Optimizes Memory Layout of Objects for Graph Analysis Programs

Tetsuya Shiota[1,a)]　Yoshiki Sato[2]　Shigeru Chiba[1]

Presented: March 18, 2014

We propose a Java compiler which optimizes memory layout of objects for graph analysis programs. In general, programs which have calculations from attributes corresponding to graph components with traversing a graph are not expected to run with high cache hit ratio if they use graph data represented by object-orientation. This is due to abstraction which express attributes of graph components as object. Although there are some data alignment techniques for cache efficient, it is difficult for user to develop program in consideration of performance of data layout in Java. In this study, we developed a Java compiler which constructs optimized object alignment based on the order traversing a graph and which transforms the fields specified by user to arrays. The compiler transforms class-declarations and object-accesses for field-arraying and embeds a code which reallocations data to construct object alignment which is based on the order traversing a graph in the original program. And our compiler separates the id as “Pointer Object” from the original objects to optimize memory layout efficiency and to free the data of the graph. We confirmed remarkable effect that our optimization improves the execution speed fourteen times in performance through an experimentation using Dijkstra method.

[1] 東京大学大学院情報理工学系研究科
Graduate School of Information Science and Technology, The Univiersity of Tokyo, Bunkyo, Tokyo 113–8656, Japan
[2] 東京大学情報基盤センター
Information Technology Center, The University of Tokyo, Bunkyo, Tokyo 113–8658, Japan
[a)] shioshiota@csg.ci.i.u-tokyo.ac.jp