発表概要

静的解析と動的処理の組合せによるエージェント指向並列言語 Orgel のスケジューリング

山本繁弘† 大野和彦† 中島浩†

我々は並列言語 Orgel の開発を行っている．Orgel は，並行/並列の実行単位であるエージェントをストリームと呼ぶ通信路で結び，明示的なメッセージ送信を行う言語である．Orgel では，プログラマが問題をエージェントという並列実行単位に切り分けることに加え，ストリームによる通信路接続網の構造をすべて宣言的に記述するので，コンパイル時に並列モデルが明確になっており精度の高い静的解析が可能である．本発表では，この Orgel の特徴を生かして，動的なオーバヘッドを最小限にした最適化を行う手法を提案する．プログラム全体の構造および粒度，通信量が完全に把握できれば，すべて静的に最適化することも可能である．しかし，Orgel では再帰的な接続も記述できるため，実際に生成されるエージェント個数および構造は必ずしも静的には決まらない．また，通信対象は分かっても送信するメッセージの個数やエージェントの粒度は実行時にしか分からない．したがって，各プロセッサの処理量が偏らないよう静的に全体をスケジューリングすることは困難である．そこで，量的な性質が分かった時点でエージェントを，割当てやスケジューリングできるように，コンパイラは静的解析による結果をランタイムに渡す．ランタイムは，この情報をもとにノード数やプロセッサの現在の負荷などを考慮して，負荷が均等になり通信量が多いエージェントは同一プロセッサになるように割り当てる．また，同一プロセッサ内では依存解析などに基づいてスケジューリングを行う．

Scheduling of Agent-oriented Parallel Language Orgel Using Static Analysis and Dynamic Processing

Shigehiro Yamamoto,† Kazuhiko Ohno† and Hiroshi Nakashima†

We are developing a parallel language called Orgel. In the execution model of Orgel, a set of agents are connected with abstract communication channels called streams. The agents run in parallel sending asynchronous messages through the streams. In an Orgel program, each unit of parallel execution is specified as an agent by the programmer. The connections among agents and streams are declaratively specified. Thus, parallel execution model is clear and the highly accurate static analysis is possible. Utilizing these features, we propose an optimization scheme that minimizing the dynamic overhead. If the complete structure of the whole program is known at compile time, static optimization will be sufficiently effective. However, in Orgel, the number of agents and structures actually generated are not always static, because recursive connection is supported. Moreover, although a communicating pairs of agents are known at compile time, the number of messages and the granularity of agents are known only at runtime. Therefore, it is difficult to balance loads on the processor by whole static scheduling. Thus, in our scheme the compiler outputs an analysis result to instruct the runtime how to allocate and/or schedule an agent when its quantitative attributes are known. Considering the number of processors and the present load of each processor, the runtime uses this information for optimization; it allocates agents balancing loads and minimizing inter-node communication. It also schedules agents on each node considering dependencies.

（平成 13 年 7 月 27 日発表）

† 豊橋技術科学大学情報工学系
Department of Information and Computer Sciences, Toyohashi University of Technology