発表概要

局面単位キャッシュ機構を持つ分散計算環境とそのアプリケーション評価

二ッ森 大介[†] 鎌田 十三郎[†]
森本 昌治[†] 松葉 健敏[†]

PC クラスタなどの分散環境を対象とした並列プログラムを作成する際，ノード間に共有データがあると，多くの場合，プログラマ自身がコピーを配置し整合性の管理などを行う必要が生じる．このような状況を解決するために，我々は，局面単位キャッシュ機構を持った分散計算環境を提案している．システム提供の分散オブジェクトは，プログラマが計算局面ごとのキャッシュ方針を各データについて注釈付けするだけで，自動でキャッシュ管理を行う．本発表では，1) 局面単位キャッシュ機構の効率的な実装法と，2) データ整合性のため意図しないアクセスを検知する機構について述べる．システムの実装にあたっては，局面遷移時のキャッシュ送受信機構や，局面に基づいたメソッド切替えなどの実装が必要となる．本発表では，実行性能と実装の portability を両立させた実装法を紹介する．また，一部アプリケーションの効率化に必要であった，全 PE で同じ更新操作を行うことでデータ整合性を保つメソッド実行モードも導入した．一方，本提案システムでは，プログラマのキャッシュ方針と実際のデータアクセスが合致しないと，プログラムが意図せぬ挙動をとりかねない．そのため，キャッシュ中の値への書き込みなどを検知し，データ整合性のミスを検出する機構も提供する．本手法の有効性については，アプリケーション開発事例を通して，その記述面ならびに実行性能の評価を行う．

Distributed Objects with Phase-based Cache Mechanism and Its Evaluation through Applications

Daisuke Futatsumori,[†] Tomio Kamada,[†] Masaharu Morimoto[†] and Taketoshi Matsuba[†]

On distributed environments, parallel programs have to prepare data copy to efficiently share the data among processing elements, and many programming environments force programmers to manage data transfer and its coherency. To overcome the situation, we have proposed a programming environment with distributed objects that allow explicit cache management, where the programmer can control the range of objects/fields to be cached using code annotation for each calculation phase, and the system automatically prepares cache. This presentation describes 1) an efficient implementation for phase-based cache mechanism, and 2) a mechanism to detect unintended data access and avoid coherency errors. To accomplish our system, we have to provide cache send/receive mechanism for phase transition, and method invocation mechanism that coordinates with phase transition for data coherency. In our implementation, we use bytecode modification techniques for portability with keeping code efficiency. For efficiency, we also introduce a new execution mode for writer methods that execute the methods on all the processing elements simultaneously with keeping data coherency. The second feature is provided for programmability. As our system requires cache policy specification written by a programmer, misunderstanding of the programmer may cause data coherency errors, such as write operations against cached fields. Our read/write check mechanism detects such data accesses. To evaluate the availability of our system, we parallelize some sequential programs on our programming environment, and measure the performance of program execution

（平成 17 年 8 月 4 日発表）

† 神戸大学
Kobe University