

Prologマシン用

ガーベージコレクション処理について

幅田伸一，中崎良成

(日本電気 C & C システム研究所)

1. はじめに 知識情報処理で強力な武器になると考えられる述語論理型言語Prologを指向した専用マシン(Prologマシン)は、高速ユニフィケーション処理部を備えると共に、非常に大きなメモリ空間を効率的に使用する機構が必要である。本論文は図1の構成のPrologマシンに導入する予定の大規模メモリ空間を対象としたガーベージコレクション処理方式について述べる。

2. メモリ管理方式 DEC 10 Prologにおけるグローバルスタックのガーベージコレクション処理(以後GC処理と略す。)の必要性⁽¹⁾や、Prologの機能拡張によるヒープ領域のGC処理負荷が増大していることから、PrologマシンのGC処理機能を強化することは必要不可欠なことである。

図1のPrologマシンは2つのプロセッサで構成されており、ユニフィケーション処理(以後Unify処理と略す。)とGC処理の並列処理が可能という特徴を持っている。メモリ側のGC処理専用プロセッサ(ガーベージプロセッサ 以下GPと略す。)と高速Unify処理専用プロセッサ(ユニフィケーションプロセッサ 以下UPと略す。)が並列処理を行なうため、メモリ管理の基本単位をProlog処理系の論理上の基本要素(クローズ、コンパウンドターム等)に一致させる。このメモリ管理の基本単位をブロックと呼ぶ。ブロックはブロック識別番号で区別され、ブロック識別番号とブロック内オフセットを指定してアクセスする可変長連続領域で、図2のようにGP内のブロック管理ユニットがブロックの集合である論理空間と物理空間のマッピングを行なっている。

3. ガーベージコレクション処理 GC処理は3つのフェーズで構成されている。オ1のフェーズは、GC処理起動時のメモリ状態をメインメモリに固定するストアフェーズである。オ2のフェーズは、GPが未使用ブロックの解放を行なうマーキングフェーズである。オ3のフェーズは、GPがメインメモリ上の使用領域を1つにまとめるコンパクションフェーズである。

2プロセッサ構成のアーキテクチャによるUnify処理とGC処理の並列処理を実現するために、オ1のストアフェーズでは図3のようにcache memory上の更新されたデータを全部メインメモリにスワップアウトしている。スワップアウト

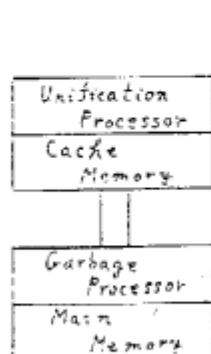


図1. Prologマシンの構成

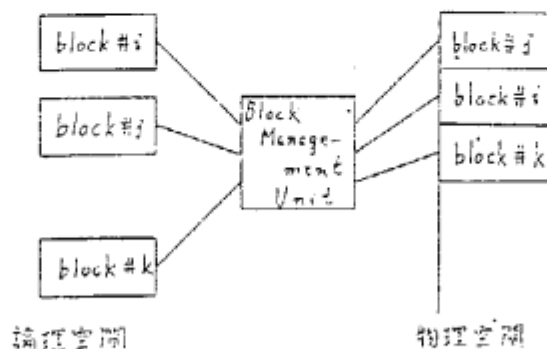


図2. Prologマシンのメモリ管理

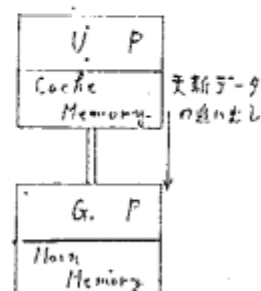


図3. UPとGPの分離

の結果、GC処理の対象となるメモリの状態がメインメモリ上に格納される。ストアフェーズが終了後、メインメモリ上のGC処理の対象となるメモリ状態を凍結する。(以後、GC処理が終了するまでUPがメインメモリに書き込みを行なうことを禁止する。)オ2のマーキングフェーズでは、GPが使用中のブロックをトレースして、使用中のブロック内のポインタが指しているブロックを全て使用中ブロックとしてマーキングして、未使用ブロックの解放を行なう(論理空間上のガーベージを回収する)。最後のコンパクションフェーズでは、GPが物理空間上の使用領域を1つにまとめる(物理空間上のガーベージを回収する)。メモリ管理の基本単位であるブロックが論理上の基本要素と一致しているため、論理空間上のGC処理と物理空間上のGC処理の分離ができた。この結果、今まで提案されていたGC処理方式(2)のようなポインタの書き換えが不用となり、コンパクション処理の負荷が軽減される。

4. ユニフィケーション処理との並列処理 GC処理のマーキングフェーズとコンパクションフェーズの間、UPがメインメモリを読み出し専用メモリとして使用し、Unify処理の途中結果をcache memoryに書き込むことで、GPによるGC処理とUPによるUnify処理の並列処理が行なわれる。(図4)

UPがUnify処理の過程で新しいブロックを生成すると、情報を格納する領域をcache memory上に確保し、ブロック識別番号だけGP内のブロック管理ユニットに登録する。

UPがメインメモリの内容をアクセスする時は、cache memoryを介して行なう。cache memory上に必要とするデータがない時は、page faultが発生して、必要データがスワップインされる。

5. まとめ メモリ管理の基本単位をProlog処理系の論理上の基本要素に一致させたことで、論理空間のGC処理と物理空間のGC処理に分離できた。その結果、GC処理の負荷を軽減することができた。

2プロセッサ構成のアーキテクチャを導入したことで、UPによるUnify処理とGPによるGC処理の並列処理が可能となり、UP側のメモリ管理によるオーバーヘッドを軽減することができた。

謝 辞

本研究の機会を与えて下さった当研究所箱崎部長、山本課長 ならびに、熱心な討論を頂いた梅村主任、小長谷研究員はじめ当グループ研究部員諸氏に感謝します。

参考文献

- (1). D.H. Warren, "Implementing PROLOG - compiling predicate logic programs", Vol. 1 D.A.I. Research Report No. 39, Dept. of A.I., Univ. of Edinburgh (1977)
- (2). H. Lieberman & C. Hewitt "A Real-Time Garbage Collector Based on the Lifetimes of Objects" Commun. ACM 26, 6 (June 1983)

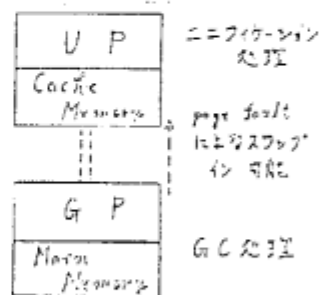


図4 GC処理中のUPとGPの並列動作