C++言語の性能に関する評価

3F−5

松岡正晴、谷口秀夫
NTTデータ通信（株）　開発本部

1．はじめに

近年、マイクロプロセッサの高性能化に伴い、汎用ワークステーション上において、オブジェクト指向のプログラミングパラダイムを持つC++言語が普及してきている。また、プログラムの生産性向上に関する要求が高まり、再利用性、拡張性、デバッグの容易性により、C++言語を用いたプログラム開発も行われている。C++言語はC言語にオブジェクト指向の機能を追加した言語であり、プログラムの生産性の向上が見込まれる。一方、追加機能に付随した処理が不必要な場合、性能低下が考えられる。

本稿では、C++言語で記述したプログラムの性能に関して述べている。具体的には、C++言語の中で性能が低下する機能を洗い出し、その性能低下の程度を明確にしている。評価したC++言語のコンパイラは、C言語のトランスレータを用いている。

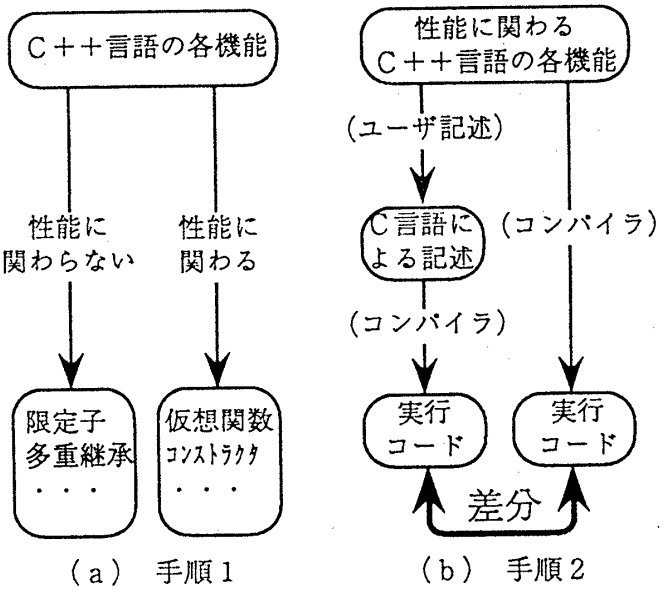

図1　評価手順

Evaluation of Performance for Program described with C++ language.
Masaharu MATSUOKA, Hideo TANIGUCHI
NTT DATA COMMUNICATIONS SYSTEMS CORPORATION

2．評価手順

以下の手順により評価を行う。

[手順1] C++言語の各機能をC言語に変換したときの結果を解析することにより、性能に関して以下の基準で言語の機能を分類する。（図1（a））

（1）コンパイル時に解決され、性能に関わらない言語機能

（2）実行時に解決され、性能に関わる言語機能

[手順2] 性能に関わるC++言語の各機能をC言語でも記述し、双方の実行ステップ数の違いにより、効率を評価する。C言語での記述が複数考えられる言語機能は、冗長性を無くすように記述する。（図1（b））

[手順1]により、性能に関わるC++言語の機能を明らかにする。[手順2]により、C++言語の各機能について、C言語と比較した効率を明らかにする。

3．性能評価

各評価手順に基づく結果と考察を述べる。

[手順1] 性能に関わる言語機能の分類

言語の機能がどちらに分類されるかは、コンパイル時の解析（静的な解析）と実行時の解析（動的な解析）をどこまで行うかによる。つまり、言語の仕様とコンパイラに依存する。評価したコンパイラは、アクセスの保護、関数への飛び先、および継承時のデータの配置を静的に解析している。一方、継承時の初期化／終期化処理、仮想関数の生成処理や呼出し処理、および仮想基底クラスの生成処理やアクセス処理を動的に解析している。結果を表1に示す。表1において、C++言語とC言語で差がない機能については除いている。

[手順2] 性能に関わるC＋＋言語の機能の効率

性能に関わるC＋＋言語の各機能をC＋＋言語とC言語の両方で記述した場合について、各実行ステップ数とその実行ステップ数比を求めた。その結果を表2に示す。この時、C言語記述における冗長性の削除は、以下のように行った。

（1）コンストラクタ／デストラクタ処理は、メイン関数上に一括して初期化／終期化の記述をする。
（2）仮想関数は、メイン関数上に直接的な飛び先の指定を記述する。
（3）仮想基底クラスは、メイン関数上にデータ格納位置への直接的なアドレス指定を記述する。
（4）メンバ関数やフレンド関数は、データメンバを大域変数として記述する。
（5）参照型のアクセスは、ポインタ型として記述する。

表2より、以下のことが明確になる。

（1）コンストラクタ／デストラクタ処理は、継承元のクラス単位で領域の確認と初期化を行っていくため、階層の段数が深くなると冗長性も大きくなる。C言語と比べて、実行ステップ数は2階層のとき最大3．6倍であった。
（2）仮想関数は、生成時のポインタ設定処理や呼出し時の間接的なジャンプにより冗長性が増える。C言語と比べて、実行ステップ数は最大10．7倍であった。
（3）仮想基底クラスは、生成時のポインタ設定処理や間接的なアクセスにより冗長性が増える。C言語と比べて、実行ステップ数は最大14．0倍であった。
（4）メンバ関数やフレンド関数は、呼出し時データメンバのポインタを引数として渡すため冗長性が増える。C言語と比べて、実行ステップ数は最大1．5倍であった。
（5）参照型のアクセスは、差がない。

4．まとめ

C＋＋言語から生成される実行コードの効率を評価した。具体的には、C＋＋言語とC言語の両方で記述した場合について実行ステップ数比を求め、以下の結果が得られた。

表1　C＋＋言語の機能と性能の関係

	性能に関わる言語機能	性能に関わらない言語機能
C＋＋言語の機能	コンストラクタ デストラクタ 仮想関数 仮想基底クラス メンバ関数 フレンド関数 参照型	非公開部(private) 公開部(public) 保護部(protected) 限定子(const) スコープ演算子 データメンバ 多重継承 関数の多重定義 インライン関数 演算子形式関数

表2　C＋＋言語とC言語の性能比

		①C＋＋言語 (実行ｽﾃｯﾌﾟ数)	②C言語 (実行ｽﾃｯﾌﾟ数)	実行ｽﾃｯﾌﾟ数比 (①/②)
コンストラクタ／デストラクタ	0階層	1	1	1．0
	1階層	8	3	2．7
	2階層	18	5	3．6
仮想関数	生成	23	0	10．7
	呼出し	9	3	
仮想基底クラス	生成	25	0	14．0
	アクセス	3	2	
メンバ関数	呼出し	3	2	1．5
フレンド関数	呼出し	3	2	1．5
参照型	アクセス	1	1	1．0

（1）多段の階層クラスのコンストラクタ／デストラクタ処理、仮想関数の生成と呼出し処理、および仮想基底クラスの生成とアクセス処理の実行ステップ数が大きく、最大で14倍になる。
（2）上記（1）の機能を使わないプログラムでは最大で1．5倍で実行できる。

今後は、以下の評価を進める予定である。
（1）言語の機能を実現するために使用するメモリ量の冗長性評価
（2）各種応用プログラムに対する性能の定量的評価