言語トランスレータを用いる多種言語処理系性能の評価

野瀬貴史[†] 泊 久信[†] 平木 敬[†]

1. はじめに

High Performance Computing においては計算機プログラムの生産性と性能の両立が重要であり，これらのトレードオフを考慮した言語選択が重要となる．選択の際には，同一のベンチマークにより言語間の性能を比較することが一つの手段となる．多数の言語同士のパフォーマンスを比較するベンチマーク集に The Computer Language Benchmarks Game[1] があるが，これは複数の言語に対し手動でプログラムを作成していることによる問題点がある．一つは，ベンチマークプログラムを作る負担が大きいために，小規模なベンチマークプログラムの集合となっていることである．例えば 1) における Java で書かれたベンチマークプログラムの平均行数は 180 行であり，小さなベンチマークとして有名な Dhrystone[2] の Java 移植版が 300 行程度になることを考えるとそれほど大きくはない．もう一つは言語ごとに実装者が異なるために，チューニングにばらつきが出る可能性があることである．この二点を踏まえると，何らかの基準となる言語を決め，それから自動変換を施すことによりさまざまな言語で書かれた同じベンチマークプログラムを生成する方式に変えたなら，より大規模なベンチマークをばらつきなく比較することができる．

現在我々は，NAS Parallel Benchmarks (NPB) 3.0 の Java 版[3] を基準として各種言語の性能評価を行っている．本稿では，現在までに得られている評価結果を報告する．

2. ベンチマークの選定

基準となるベンチマークを選定するに当たって，自動変換のしやすさ，変換後のソースの可読性，ベンチマークとしての実績を考慮し，Java 実装が存在する NPB を本研究の対象とした．Java を基準とした理由は，1. プリプロセスがないために変換前後でソースの意味論の一対一対応が付けやすいこと，2. 静的型付けであること，3. クラスベースオブジェクト指向の標準的な機能が揃っていることの三つである．評価の客観性からは，アーキテクチャ評価に用いられる SPEC を使うことが望ましいが，SPEC の実装言語は C, C++, Fortran の混合であり，言語の評価には不適切である．一方，Java 版 NPB は実装言語が統一されており，かつ 4) で SPEC CFP2006 との高い相関があることから，NPB が本研究の目的に最も合致すると判断した．

† 東京大学
The University of Tokyo

3. 変　　換

Java から各言語への変換では，なるべく逐語訳となるような出力をするトランスレータを作成し使用した．変換の際には Java の構文木を受け取り，そこから直接ソースコードを出力している．For 文は言語間でのスタイルの差が大きく単純な置換が困難である．そのような場合は While 文で置換した．I/O 部，メモリ確保部は，言語間の仕様が異なるが，数値計算の計測時間には含まれず，量もわずかであるため，一部はトランスレータを用いずに手で変換した．

4. 性能評価

移植したプログラムの評価はすべて Dell PowerEdge R410 (Xeon E5530 2.4GHz，CentOS 5.5，メモリ 12GB) で行った．現在までに移植した言語はコンパイラ言語の C99, Fortran 2003 とインタプリタ言語の Ruby, Python, PHP である．インタプリタおよび FORTRAN77，C99，Fortran 2003 プログラムのコンパイルは GCC 4.5.1 を用い，最適化オプションは-O3 -msse4.2 を使用した．図の棒グラフに一部欠損があるのは，トランスレータあるいはコンパイラのバグによりベンチマークが正常動作しなかったことによるものである．

4.1 言語間比較

図 1 に，NPB のコンパイラ言語とインタプリタ言語の性能の比を示す．オリジナルでは，IS のみが C で，他はすべて FORTRAN77 で実装されている．オリジナル版と Ruby1.8.7 版を比較すると，ベンチマー

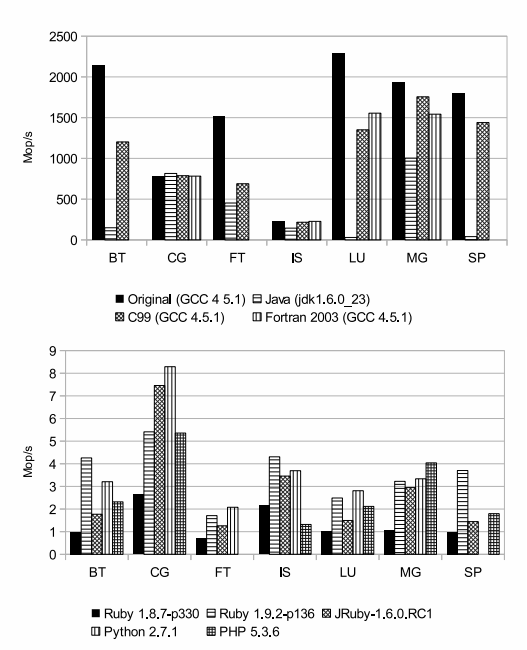

図 1 コンパイラ言語とインタプリタ言語の性能比較

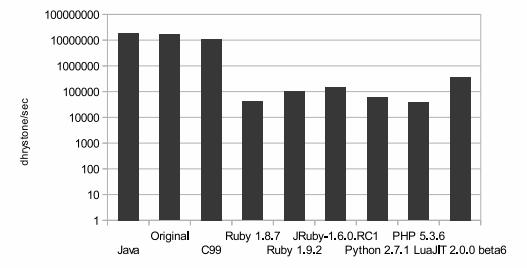

図 2 Dhrystone を用いた性能比較

クによって 105 倍から 2240 倍の性能差があった．

図 2 では Dhrystone の比較も示した．Dhrystone では Lua の移植も完了している．イテレーション回数は 50000000 回である．

4.2 スケーリング

Ruby と Python では Java 版に実装されているマルチスレッドモードも含めて移植した．Ruby1.9 以降や Python ではネイティブスレッドをサポートしているものの，Global Interpreter Lock (GIL) が存在する処理系では，同時に実行されるスレッドは一つに制限されるため，実行性能はスレッド数を増やすとかえって低下する．しかし JRuby や Jython といった JVM 上の実装は GIL が存在しないため，スレッド数に応じて性能がスケールする．現在 JRuby について実験が完了しているので，**図 3** に JRuby での結果を示す．

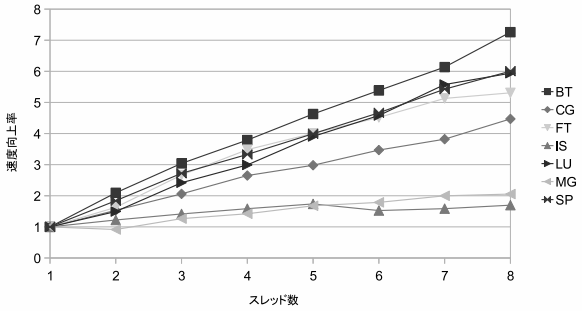

図 3 JRuby 1.6.0 RC1 でのスケーリング

5. まとめと今後の課題

本稿では Java から各種言語へのトランスレータを用い，NPB を意味論の一対一対応を保持しながら変換し，言語処理系の性能評価を行った．手動で行うよりも公平性を確保でき，かつ多数の言語の比較を少ない労力で達成できた．

今後は，ベンチマークを取る言語・処理系のさらなる拡充を行う．特に，関数型言語や，高速な JIT を備える処理系がある JavaScript を評価の対象に加える予定である．また，評価する言語基盤を JVM に加えて LLVM と.NET Framework も対象にする予定である．将来的には，言語・言語処理系の数値計算における性能を測るための統一的指標を導きたい．

参 考 文 献

1) Gouy, I.: The Computer Language Benchmarks Game, http://shootout.alioth.debian.org/.
2) Weicker, R.: Dhrystone: a synthetic systems programming benchmark, *Communications of the ACM*, Vol.27, No.10, pp.1013–1030 (1984).
3) Frumkin, M., Schultz, M., Jin, H. and Yan, J.: Implementation of NAS parallel benchmarks in Java, *a Poster session at ACM 2000 Java Grande Conference* (2000).
4) Tomari, H. and Hiraki, K.: Retrospective Study of Performance and Power Consumption of Computer System, *SACSIS2011* (2011).