ソフトウェアのグリーンITについて

氏　名†　山本　理浩
所　属†　ソラン株式会社

はじめに
省電力ソフトウェアを開発する技術の必要性

地球温暖化対策としてデータセンターのグリーン化が求められており、ハードウェア基盤は対策が進んでいるが、ソフトウェアについては、データベースソフトウェアにおいて利用していないエリアのディスク稼働を減らしたり、あるいは仮想化技術といったデータセンター運用にかかわるミドルウェア領域ではいくつかの対策が見られるものの、ソフトウェア全体では、生産物であるソフトウェアの環境特性についてまだあまり認識されていない状況といえる。
従来、ホストコンピュータの全盛時代には、ハードウェアの CPU や主記憶装置は大変高価でまた容量はごく限られたものであり、ソフトウェア開発者は、いかに効率的にハードウェア資源を使うかという最適化の技術を重要視していた。しかしながら Windows などの普及とともにハードウェアの資源は安くて大量に手に入る時代となり、ハードウェア資源を効率的に使うことより如何に早く多くの機能を実装するかの技術（開発効率性）が重視されるようになり現在も続いている。環境問題一般に求められているのは、資源量の限界の認識であり、ソフトウェアも同様に、多大な電力消費すなわち CO2 排出を行うハードウェア資源の浪費は許されなくなる趨勢になると考えられ、今後は資源利用を最適化した実行をおこなうソフトウェア技術が重視され、多大な電力消費（＝CO2 排出）をするソフトウエアは容認されなくなる趨勢になると考えられる。　本稿では、資源を最適化した実行をおこなうソフトウェア技術が重視され、電力を消費しないソフトウェアを開発する必要があると筆者は考えておりその技術の方向について述べる。

2．環境配慮型ソフトウェアの特性

ソフトウェアは電子的媒体であり、物質でないため廃棄物としての影響は少ないため環境に直接的な影響は少ないと長く見なされてきたが、LCA（Life Cycle Assessment）の観点からその生産過程から、流通、利用、廃棄の一連の側面からその環境特性を見る必要が唱えられている。（文献 1）温暖化現象という近年の最も大きな課題から考えるとソフトウェアの利用により電力消費の形で排出される CO2 の量は莫大なものといえる。社会全般をみると各種のハードウエアデバイスには組み込みソフトウェアの形でソフトウェアが稼働しており社会全般で利用されるソフトウェアから排出される CO2 は増大傾向にあると考えられる。組み込み系のソフトウェアにおいては、モバイル機器などの特性上、電池利用を前提として消費電力を抑え機器の稼働時間を長くするため、省電力技法についていくつか提案されてきている。（文献 2）しかしながらインターネットの普及により、WEB 系のアプリケーションの利用による CO2 排出は大きく増大している考えられるにも関わらず、WEB 系を含む多くのアプリケーションでは、OS を含め電力消費を抑える要求は現時点では強くないと考えられる。ホスト全盛時代に比べソフトウェアの環境は複雑化しており、Java などの仮想的な実行環境では、以前の最適化技術の延長上にグリーンなソフトウェアを作る技術：グリーンコーディングが今後必要になってくると筆者は考える。

3．Java グリーンコーディング事例

3－1　Java パフォーマンステクニック

ソフトウェアの実行時に排出される CO2 を減少させるひとつの大きな要素としては、CPU の資源消費を抑えることがあげられるが、これは組み込みソフトウェアに於いても重要視されている。（文献 2）一方 Web 系のアプリケーションにおいては Java 言語が普及しているが、これにはいくつかのパフォーマンステクニックの存在が発表されている。これは、処理速度の向上のためのコーディング上の記述上の技法ないし事例を集約したものであるが、パーフォーマンスの向上の実測方法として CPU の処理時間を基準としている。したがってこの技法によれば、CO2 の排出増のもととなる CPU の処理量を減らせる可能性

Software Green IT (for JIPS Demonstration Session 2009)
Michihiro Yamamoto　Sorun Cooporation

を示唆している。例えば、Dav Bulka（文献 3）による技法をグリーンコーディングの技法と捉えた場合、（机上の計算ではあるが、）表 1 に示すような、CPU 使用率の減少が期待できる。

表 1　Dav　Bulka による最適化事例

Dov Bulkaの分類	事例	効果測定(CPU処理時間)
文字列処理	16文字以上のStringとStringBuffer(32)	37%
	StringObj生成	93%
	String比較	260%
	バイト変換	680%
	String Tokennizer	340%
純粋なOH	Trace機能	40%
	冗長な初期化	78%
VectorとHashtable	Vectorの追加	720%
	Vectorの容量	720%
	equals()	390%
IOストリーム	Output Buffering	73%
	Input Buffering	850%
再利用	Vecor再利用	245%

こうした事例は、Java に限らず C や COBOL などでも同様な最適化事例があり、また生産組織においては古くからコーディング規約の一部としてまとめられていたものである。しかしながら、品質特性として環境面の配慮のためという項目はなく使われていたものである。こうしたコーディングテクニックや規約を見直し生産現場に定着することが、環境側面で有効であることを示唆している。

3－2　リファクタリング技術

Java 開発に於いては、アジャイル開発などの発展にともなって、コード保守を目的としてリファクタリング技術の必要性が唱えられている。リファクタリングの直接の目的は、機能追加でもなくバグ減少でもなく、コードの可視化による保守性向上であるが、技法の実態には for ループによる無駄な資源消費をおさえる技法など最適化技法に通じるものがある。リファクタリングの技法は 100 近い技法が集積されているが、大きくカテゴライズすると以下表 2 のようになる。

表 2　リファクタリング要素

適切な名前を付ける
一時変数をなくす
条件分岐をシンプルにする
条件判定か例外か
オブジェクトをつくる
クラスやインターフェースを抜き出す
修正できないクラスにメソドを追加する
カプセル化
状態を変えるかどうか
フィールドの位置を変える
メソドの位置を整える
メソッドを作る
メソッドの引数を整える
タイプコードを置き換える
古いシステムにオブジェクト指向を導入する
値オブジェクトか参照オブジェクトか
モデルとビュー
継承と委譲を切り替える
クラスの関連

3－3　グリーンコーディングの期待効果

以上、グリーンコーディングの事例について簡単に述べたが、その効果として以下があげられる。

1）直接的効果

ソフトウェア使用時における消費電力の減少（＝CO2 排出減）

2）間接効果

ソフトウェアの LCA を考えた場合、ソフトウェアの保守が行い易いことは、開発保守時間の短縮になりマシンの使用時間が減少することにより CO2 排出を抑制する効果もあると考えられる。間接的な効果として最適化に必要なコーディング規約やまたリファクタリングの充実により開発時の効率、修正時の効率により開発マシン利用時間減少による CO2 排出減が期待できる。

今後の課題

本稿では、机上での計算のみで示したが、OS やミドルウェアを含めた環境での測定が必要であると考えている。コーディング上では効果が見られても、ミドルウェア、オペーレーションシステムは独自の動作を行うため、ソフトウェア全体として効果を測定する必要がある。

まとめ

温暖化対策はソフトウェア開発者、開発企業すべてが今後 10 年で全力で取り組まなければならない重要な課題と考える。またそのことは従来からの SW 品質技術の延長にあるとともに、温暖化対策を目指したグリーンコーディングの発展はソフトウェア品質向上に大きく貢献するものと考えられる。

参考文献

[1]浅田哲臣、木村文彦、加藤悟　環境配慮型ソフトウェアの分類と評価基準の作成（2002）ECO Design 2002 Japan Symposium

[2]石原 亨ソフトウェアの消費エネルギー解析と最適化技術 Techniques for Estimating and Reducing the Energy Consumption of Embedded Software Tohru ISHIHARA (2008)

[3]DovBulka Java Performance and Scalability Vol.1 (2000)

[4]結城浩 Java 言語で学ぶリファクタリング入門(2007)