スーパーコンピュータとのコミュニケーション
Communication with Supercomputer

富士通株式会社
次世代テクニカルコンピューティング開発本部
千葉　修一

スーパーコンピュータ「京」の誕生により、今日まで様々な研究で成果が上げられてきた。これらの研究は、「京」に搭載された様々な技術によって実現されている。その中でもコンパイラの技術は、研究における課題や実験をシミュレーションとしてスーパーコンピュータで実行するために必要不可欠な技術である。本稿では、コンパイラを通して人間とスーパーコンピュータのコミュニケーションについて紹介する。

1．はじめに

2012年９月、スーパーコンピュータ「京」［１］（以降、「京」）の共用利用が開始され、今日まで様々な成果が上げられてきた。「京」の誕生は、Top500［２］首位という話題性だけでなく日本の科学技術の発展にも大きく貢献している。これは、重点配分枠の優先課題７件、および一般配分枠の研究開発課題24件を中心に幅広い分野で成果が報告されていることからも分かる。たとえば、図１のような津波による浸水をリアルタイムに解析する研究［３］は非常に身近な社会問題に対する成果である。

これらの研究は、スーパーコンピュータと協力し課題や実験をシミュレーションすることで成果を得ている。そのためにはコンピュータとのコミュニケーションが必要不可

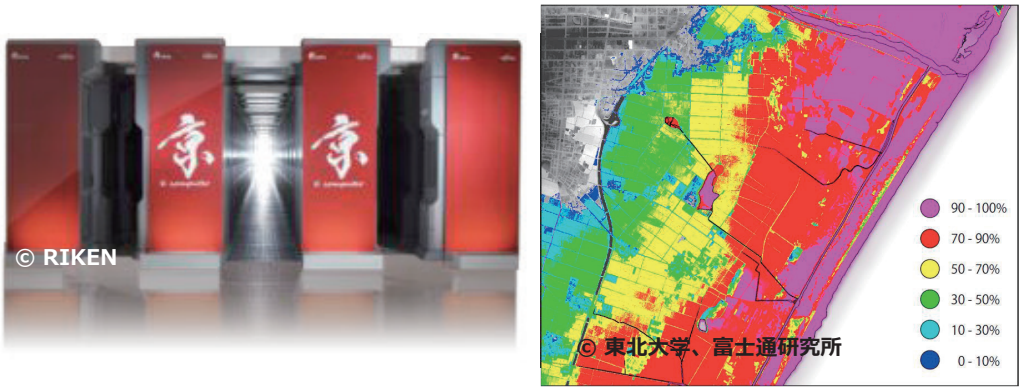

図１　スパコンで高解像度の津波モデルを用いてリアルタイムに浸水を解析
©理化学研究所、東北大学、富士通研究所

欠である。スーパーコンピュータとのコミュニケーションは、自然言語による会話ではなくコンパイラが提供するプログラミング言語によって実現している。コンパイラは、人間とスーパーコンピュータを繋げるために言葉を受け渡すソフトウェアである。本稿では、コンパイラを通して人間とスーパーコンピュータのコミュニケーションについて紹介する。

2．コンピュータとのコミュニケーション

コンピュータといって最初に想像するものは何があるだろう。事務作業を行っている人の場合は電卓かもしれないし、子供たちはゲーム機を想像するかもしれない。現在では人間が発する自然言語を理解するロボットも登場しているが、内部はプログラムの指示によって動作している。プログラムは、手紙のようなものだと考えると分かりやすい。図2のようにコンピュータに実行して欲しい要件をプログラムとして記述し依頼することで、コンピュータは依頼された要件を実行する。この時プログラムを記述するための手段が、プログラミング言語である。プログラミグ言語は、プログラムという手紙をコンピュータに理解してもらえるように記述するコンピュータの母国語となる。

プログラミング言語は、依頼する内容の種類に応じて数多く用意されており、利用者によって任意のものを選択することができる。これは、数か国語を話せる人間が、一番伝えやすい言語を選んで会話をすることに近い。

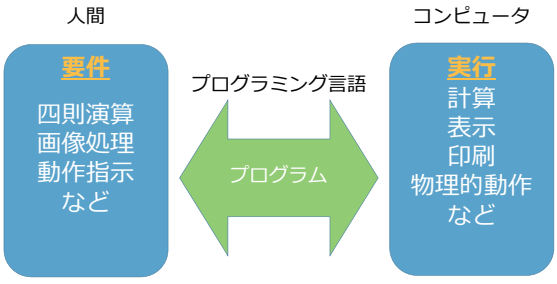

図2　プログラム

プログラムによって実行された内容は、ディスプレイやディスク装置に格納されるファイルなど入出力機を経由して人間に伝えられる。これらのやりとりがコミュニケーションとなる。

3．プログラミング言語

プログラミング言語は、コンピュータに指示を与えるプログラムを記述するための「言葉の集合」である。スーパーコンピュータだけでなくパーソナルコンピュータなどの一般的なコンピュータを含め、コンピュータが理解できる情報は“0”か“1”の二値を表現するビットであり、このビットの並びが様々な言葉となる。コンピュータによって違いはあるが、あらかじめ規定されたビットのパターンを言葉としてコンピュータに指示する。これらの言葉は機械語と呼ばれ、短点と長点の二音で言葉を表現するモールス符号［4］にも似ている（図3）。

言語	表現
日本語	加算
英語	add
モールス信号	・－、－・・、－・・
機械語	0100100010000011

図3　言語の比較

学生がコンピュータの基礎を学ぶ時、基盤がむき出しのCPUに配線で簡単な入出力機を結合し、機械語の辞書を片手にプログラムを入力して、音を鳴らしたり、LEDを点灯させたりするような勉強を行うことがある。このような勉強は、デジタルな世界を理解するために非常に有効である。しかし、このような指示の方法は非常に効率が悪い。そこで登場するのがプログラミング言語である。プログラミング言語は、図4のように大きく2つに分類される。

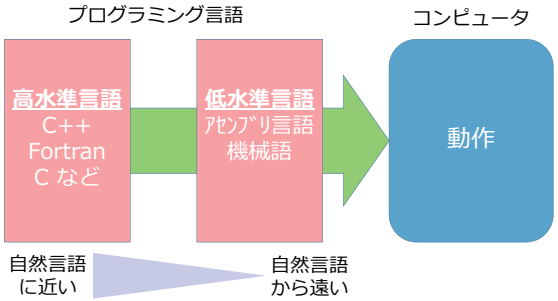

図 4　プログラミング言語の種類

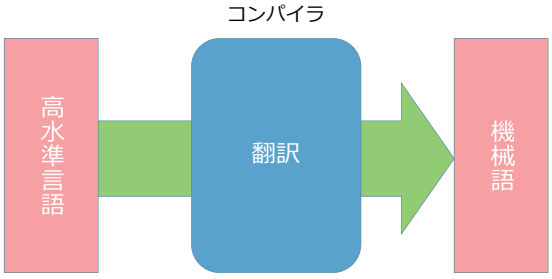

図 5　コンパイラによる翻訳

一つは、日本語や英語のような自然言語に近く、コンピュータの種類などに依存せず(抽象的)に記述することが可能な高水準言語であり、目的に応じて様々なものが存在する。たとえばスーパーコンピュータを活用する場合は、科学計算を表現しやすいプログラミング言語として、古くからFortranが多く利用される。また、オープンソースソフトウェアのような多人数かつ分散した開発を容易に行うために、C++のようなオブジェクト指向のプログラミング言語が利用される。もう一つの分類としては、コンピュータに特化した命令を直接記述する低水準言語がある。前述した機械語は、最も低水準なプログラミング言語である。この機械語を人間に分かりやすい形で表現したものがアセンブリ言語であり、この 2 つが低水準言語の代表例である。通常は、自然言語に近い高水準言語を利用してプログラムを作成するため、高水準言語をプログラミング言語として表現することが多い。しかしながら、コンピュータが理解できる言語は機械語であるため、高水準言語で記述されたプログラムを機械語で記述したプログラムへ変換する「翻訳」という作業が必要となる（図 5）。この作業を行うソフトウェアが「コンパイラ」である。

コンパイラは二つのプログラミング言語を通訳する「翻訳者」である。そのため、コンパイラがプログラムの表現を理解できなかったり、機械語への変換できなかったりした場合は、コンピュータへ正しく要件を依頼することができない。前述したようにプログラムは手紙のようなものである。コンパイラは、この手紙の誤字脱字などをチェックして、もしコンピュータが理解できないような内容であれば利用者に書き直しを要求することもある。翻訳者でありながら「添削者」でもある。

また近年、高水準言語は非常に多くの種類が存在している。さらにプログラムを依頼する先のコンピュータの種類も数多くあり、それごとに機械語が異なる。そのため、図 6 に示すように高水準言語と機械語の組み合わせは多岐に渡る。コンパイラは、これらの組み合わせを正しく理解し、翻訳することも必要となる。そのためにコンパイラの内部では、組み合わせごとに翻訳用の辞書を用意している。

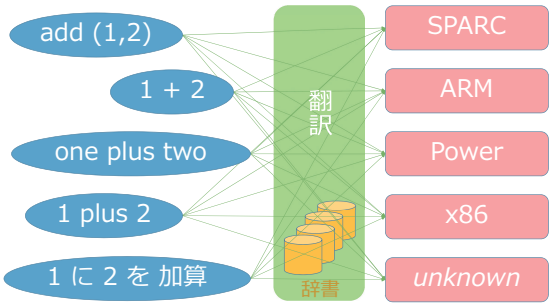

図 6　多岐に渡る翻訳

コンパイラは、コンピュータとコミュニケーションを図るためのコミュニケーションツールである。

4．翻訳

少しスーパーコンピュータの世界から離れて、翻訳という行為について考える。翻訳は、コミュニケーションにおいて異なる言語間を結び付ける場合に必要となる。元となる言語の意味を「理解」し、異なる言語に「変換」する。翻訳は、この二つの作業の組み合わせで実現される（図７）。

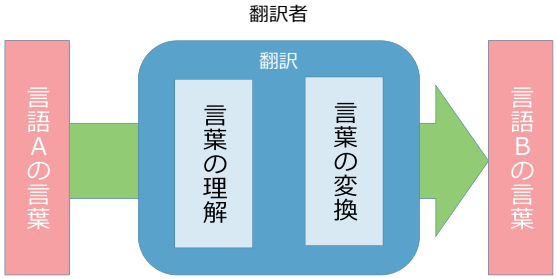

図７　翻訳

この時、片方の言語でしか表現できないような言葉や曖昧な言葉は、翻訳者が判断して任意の言葉へ変換することになるため、翻訳者によって異なる結果となる場合がある。自然言語において日本語から英語、英語から日本語に翻訳する場合も同じで、翻訳者は翻訳作業の「理解」において翻訳元となる言語の知識が必要となり、「変換」において翻訳先となる言語の知識を強く必要とする。

5．翻訳の難しさ

現代社会では、コミュニケーションツールとして携帯電話やスマートフォンが活用されている。これらのコミュニケーションツールでメッセージをやり取りする手段として、メールやLINE［５］などのアプリケーションが用意されている。人間同士がコミュニケーションをとる場合、会話で利用する日本語や英語などの自然言語でやり取りすることが最適であり、現在のアプリケーションもこれらの言語が利用できることは一般的となっている。

しかし、携帯電話やスマートフォンが無かった時代は簡単にコミュニケーションをとることができなかった。ここでは、1990年代に主流となっていたポケットベル［６］を使ったコミュニケーションを例に挙げ翻訳の難しさに触れたい。

まず、初期のポケットベルは、呼び出し者がポケットベル用の電話へ連絡をすると、ポケットベルに搭載したスピーカーから呼び出し音が流れるというシンプルなものであった。つまりポケットベル側は、緊急性を指示されるだけで、誰からなのか、その目的すら理解することができなかった。

そこで次に登場したポケットベルは、呼び出し者の電話番号を通知できるものであった。ポケットベル側は、呼び出し者が呼び出し時に入力した数字をディスプレイ上で認識することができ、それを確認することで呼び出し者が誰なのかを特定することが可能となった。この機能の本来の目的は電話番号の通知であったが、利用者の増加に合わせてこの数字に意味を持たせるコミュニケーション手段が確立された。これが「ポケベル語」の誕生である（図８）。

ポケベル語	意味
１９４	行くよ
４５１０	仕事
１１０１	会いたい
５１１０	ファイト
５９６３	ご苦労さん
８１８１	バイバイ

図８　ポケベル語の例

これは３．で述べた機械語のような数字の羅列とメッセージが対応づけられている。ポ

ケットベルを利用したコミュニケーションでは、ポケットベルを所有するメッセージの受信者は翻訳者となり数字の羅列を自然言語へ翻訳する。ポケベル語のようなメッセージは、最初から辞書が用意されるわけではなく、使われているうちに取捨選択されて浸透していく。つまり、曖昧な時期には一部の地域または個人のみが翻訳可能なメッセージも登場する。そのため、受信者によって誤認するケースも多くみられた。コミュニケーションにおいて言葉の意味の違いは致命的な問題を引き起こす。図９は、入院患者とその友人という関係において筆者が経験したコミュニケーションの失敗例である。

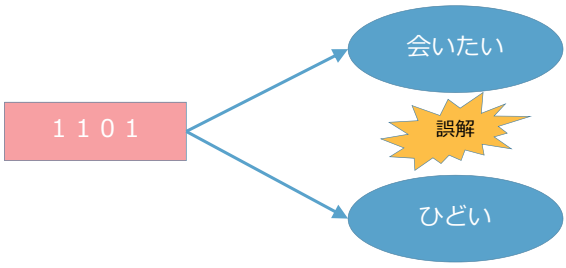

図９　理解の失敗

入院患者が通知した「会いたい」に対して、受信者は「ひどい」と誤った翻訳を行ってしまった。入院患者は「安心」を与えるメッセージを送ったつもりでいたが、受け取り側の友人は「不安」を受ける正反対の結果となってしまった。このように翻訳の失敗は、“YES”と“NO”の逆転現象さえ発生する。これと同じくスーパーコンピュータを利用したシミュレーションでは、翻訳の失敗が研究や実験の失敗に繋がる問題になることもある。

このような問題が発生する要因の一つに言葉の変化が考えられる。たとえば、言葉の利用頻度が高くなってきた時、それを表現するメッセージを短縮したいようなケースがある。

ポケベル語は電話機の番号を利用して作られる。たとえば、「愛してる」というキーワードは、「１」を“愛”、「４」を“し”、「10」を“て”、「６」を“る”として５文字を送信していた。しかし、“て”の表現に対して２文字割り当てることを無駄と考え、後期では「０」のみで表現することもあった（図10）。このようなケースにおいては、翻訳者がメッセージの変化を取り込む必要が出てくる。

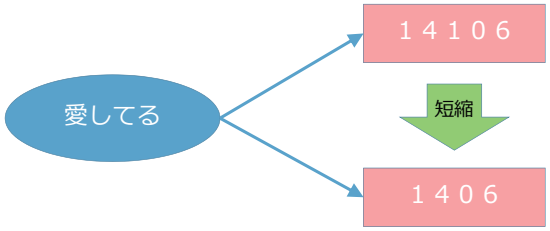

図10　言葉の変化

コミュニケーションツールで日本語や英語などのメッセージが利用できるようになった現在でも、ポケベル語と同じような現象が起こっている。たとえば、インターネットの掲示板などで利用されるネットスラング［７］や女子高校生などが流行らせる若者言葉もその一つである。毎年、ユーキャン新語・流行語大賞［８］が選出されることからも分かるようにコミュニケーションで活用される言葉は、日々変化している。これはプログラミング言語でも同じで、利用者に合わせて規格（言葉）が継続的に変化している。近年、プログラムは大規模化しており生産性を向上するために規格が更新されている。たとえば、重複する記述を削減したり、長文となる記述を簡略化したりする方法が用意されることで、プログラムを効率的に作成することが可能となる。

６．方言の標準化

前節で述べたようにコミュニケーションにおける言葉は常に変化していることが分か

る。また変化とは関係なく特定の範囲で利用者が存在する言葉として「方言」がある。たとえば、「ありがとう」という言葉には、都道府県でいろいろな言い方が存在する（図11）。

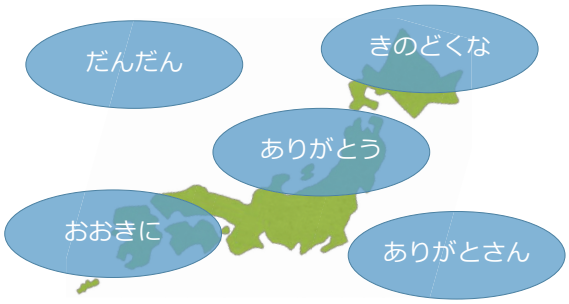

図　11　ありがとうの方言

これらの表現は、特定の地域のみで翻訳が可能である。「だんだん」などは中国地方の翻訳者のみが「ありがとう」として認識できる。ただし利用者や利用シーンが増えることで広く認知されるケースがある。「おおきに」は、関西地方でよく使われる表現であるが、全国的に認知されている。その結果、「おおきに」は日本語辞書にも登録され、日本語として一般化されている。プログラミング言語でも一部のコンパイラのみが翻訳できる方言が存在している。これはコンパイラによって提供される付加価値であり、差別化にも繋がっている。このような生産性を高めるための方言に加え、コンパイラに対する翻訳のヒントとなる方言もある。前者は、規格になる前段階の言葉を積極的に取り込んでいる状態である。後者は、コンパイラがプログラムをスムーズに翻訳できるように言葉を補足する時に使われる。

7．プログラミング言語の進化

コンピュータの世界における翻訳者であるコンパイラは、オープンソースソフトウェアや企業の製品として様々な種類が提供されているが、プログラミング言語を翻訳するという作業においては共通している。ただし、プログラマによるプログラミング作業の生産性を向上させる目的で、コンパイラが独自の記述方法を提供する（許可する）場合がある。これが前節で述べた方言のような位置づけの言葉となる。GNUコンパイラ［9］のようにオープンソースソフトウェアであるコンパイラは、複数の開発者によってグループ開発されており、利用者も様々である。そのためソフトウェア開発で利用頻度の高いプログラミング言語のCでは、利用者を支援するためのGNUコンパイラ独自の記述方式が次々に提供される。GNUコンパイラが独自に提供するプログラミング言語の拡張は、GNU拡張仕様として広く知られている。

コンパイラが独自で提供する拡張仕様であっても利用者が多い場合は、他のコンパイラがその拡張仕様を取り込むことがある。これがプログラミング言語における一般化である。仕様が一般化されることでプログラマがコンパイラを選択できる自由度も増える。また、一般化により標準的に利用する価値がある仕様については、プログラミング言語の言語規格に取り込まれることになる。自然言語において方言が標準語に取り込まれることは少ないが、プログラミング言語においては優秀な言葉を積極的に規格化することで言語を進化させている（図12）。この背景にはプログラムの開発スタイルの変化が影響している。開発スタイルは、プログラミングモデル［10］やプログラミング技法［11］としてソフトウェアの変化に合わせて新しく考えられる。こういったスタイルの変化に合わせる形で新しい仕様が誕生している。この結果、言語規格に取り込まれた仕様は、全てのコンパイラによって共通で利用が可能となる。プログラミング言語は自然言語よりも積極的な進化がみられる。

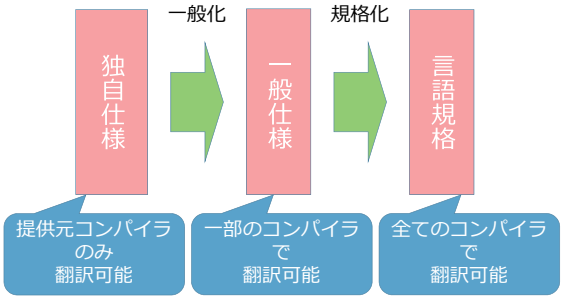

図12　言語の進化

8．シミュレーション

それでは、スーパーコンピュータがコミュニケーション相手の場合はどうだろう。スーパーコンピュータの一番の特徴は高速な計算処理であり、その性能に期待するような要件を依頼することが中心である。そのため、通常のコンピュータを利用する行為とは区別され、ハイパフォーマンスコンピューティング（以降、HPC: High Performance Computing）と呼ばれている。このHPC分野の要件は、天候や地震などの自然現象の解析、自動車の衝突解析などのシミュレーションである。シミュレーションもプログラミング言語で記述したプログラムを利用することでスーパーコンピュータへ指示を与えるが、通常のプログラムと区別してシミュレーションプログラムと呼ばれる（図13）。

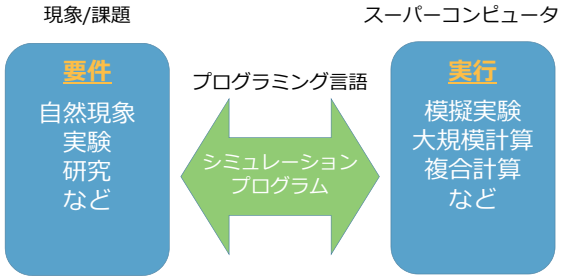

図13　シミュレーションプログラム

ただし、シミュレーションは非常に複雑な内容であり、シミュレーションプログラムを作成するためには多くの作業を経由することとなる。まず自然現象や実際の実験内容は、科学者や研究者によって何らかの数学的な式（数学モデル）で表現される。たとえば、微積分などを活用した数式がそれである。このような式は、プログラミング言語でプログラムとして記述することができない。そのため、プログラムとして記述できる四則演算のような簡単な数式に変換（離散化）する必要がある。この作業により、自然現象や実験などのアナログな世界をデジタルな世界で扱うことが可能となる。しかしながら、アナログなデータをデジタルで完全に表現することは難しく、ある程度の実験精度［12］が犠牲になる。そのため、実験精度を落とさないような数多くの変換方法が研究されている。変換が行われた後の式（数値計算モデル）は、プログラミング言語を利用して記述することが可能となり最終的にプログラムとなる（図14）。

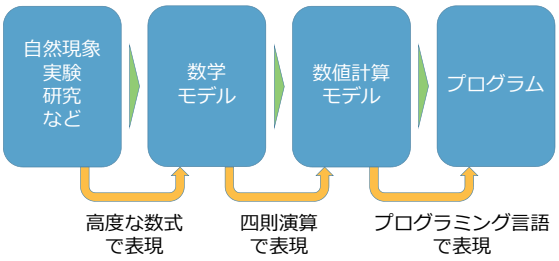

図14　シミュレーションの作成

HPC分野では、コンパイラの入力となるプログラムを作成するまでに、数多くの変換が行われることが特徴である。これらの変換作業もそれぞれに翻訳者が必要となる。

9．最適化

HPC分野でもプログラムが完成すれば、あとはコンパイラによって機械語に翻訳して実行することができる。しかしながら、HPC分野で要求される高速な計算処理は単純にプログラムを翻訳するだけでは実現できない。プログラムを通して指示された要件をスーパーコンピュータが高速に実行できるように効率

的な機械語を生成する必要がある。これは図７における「変換」の作業の一つである。たとえば、言葉としては意味が通じるメッセージであっても、必要のない言葉を省くことで情報を短縮し、メッセージの受け取り手が情報を素早く理解することが可能となる（図15）。

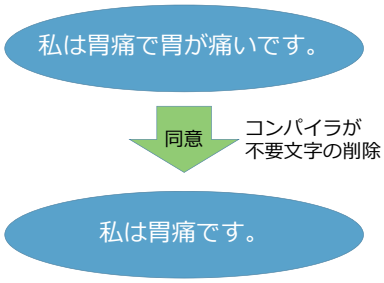

図15　メッセージの短縮

単純なメッセージを短縮することは簡単であるが、シミュレーションを実現するような膨大なプログラムの中でその意味を理解し、短縮することは非常に難しくコンパイラの力量が試される部分でもある。またスーパーコンピュータはパーソナルコンピュータと違い、高速に計算処理を行うための様々な機能が用意されている。これらの機能を利用する場合は、スーパーコンピュータに応じた機械語へ翻訳する必要がある。そのため、図16のようにHPC分野のコンパイラはスーパーコンピュータの特徴や特性を把握し、プログラムがその利点を最大限に活かせるような言葉の組み合わせ（命令の組み合わせ）を思考し、機械語へ変換しなければならない。このよう

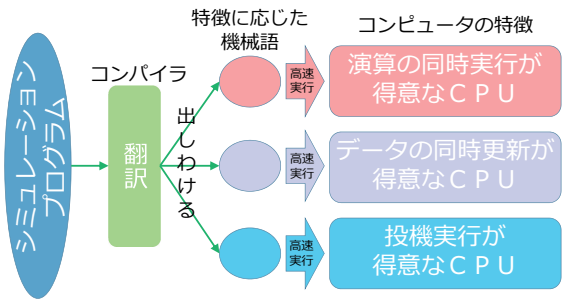

図16　コンピュータに応じた機械語へ変換

に高速に実行できるように変換すること「最適化」という。

最適化は、コンピュータの能力を引き出すための技術であり、最適化の優劣はスーパーコンピュータ上のシミュレーションプログラムの実行時間を上下させる。ここでは最適化の技術の一つであるスケジューリングについて説明する。

スケジューリングは、プログラムに列挙された要件をどのように実行すれば効率的に行えるかを考え、作業順序を並び替える。たとえば、ベランダに布団や洗濯物の干す場合はベランダのスペースを考えなければならない。もし、同時に干せない場合はどちらかを優先すべきである。また、買い物と料理は作業に関係性があり、買い物で食材を入手しないと料理を作ることができない(図17)。このように作業するために利用するものや作業の関係性からより効率的に実行ができるパターンを見つけ出す必要がある。

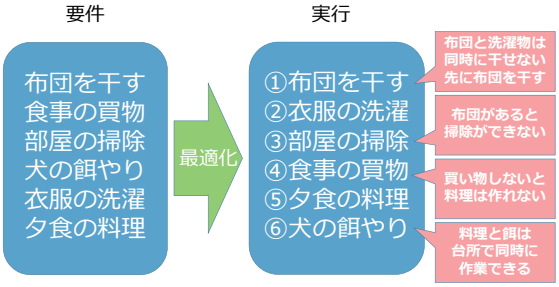

図17　スケジューリングのイメージ

10. 最適化の限界

スケジューリングを含め最適化の適用パターンは、一種類だけとは限らない。図17でも、洗濯の時間がかかるようであれば、布団を干す前に洗濯を行うべきである。実際の実行時の条件や環境によって最善のパターンは変化するため、完全な正解はない。コンパイラは、可能な限り最善のパターンを適用できるように数多くのパターンを思考した後、最善のパターンを選択する。しかし、最善のパ

ターンが選択できない要因もいくつか存在する。その一つにデータの分離化がある。計算に利用するデータはプログラムとは別に管理されることが多く、データの種類や量を翻訳時に判断することができない。そのため、データの種類や量に依存するハードウェア資源［13］を効率的に割り当てることができない場合がある。

図18では、料理を行うプログラムにおいて調理の準備（野菜を切る作業）と実際の調理（野菜を炒める作業）を分けて作業するように指示している。実行時、４人前の食材は家庭用のまな板で十分に処理することができるが、８人前となるとまな板に野菜が載り切らない。そのため、全部の野菜を切るためには、２回に分けて作業を行う必要がある。この時、２回目の切る作業中に１回目で切った野菜を並行して炒めることができれば効率的である。８人前に対しては、与えられたものよりも最適な手順が存在する結果となる。逆に、８人前用に効率的な半分ずつ調理するという手順を最適なプログラムとして４人前用に適用した場合は、作業回数が増えるため非効率である。これらをHPC分野に置き換えると、まな板はハードウェア資源といえる。４人前か８人前かは、コンパイラの翻訳時（静的な状態）では判断できず、スーパーコンピュータの実行時（動的な状態）でないと見えない事象となる。

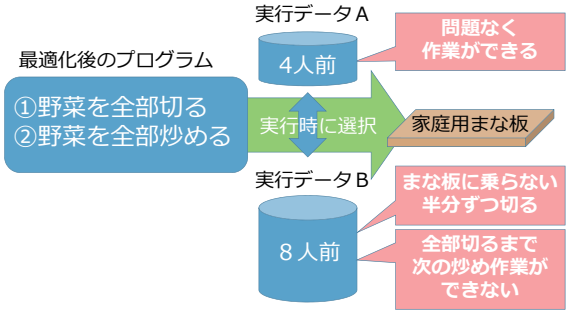

図18　データによる実行の変化

もう一つの要因は、パターンを思考するための翻訳に費やせる時間の限界である。たとえば、実行時間が10分のプログラムを高速化するために、10分以上もパターンの思考に時間がかかっていては意味がない。そこで、ある程度の時間や回数で制限をかけることでパターンの思考を打ち切り、制限内で思考したパターンの中から最善のパターンを選択する。一般的に考えてもプログラムで記述される要件は、無限の組み合わせがある。さらにHPC分野の場合はプログラムが複雑になるため、これらの制限により思考が打ち切られるケースが多い。HPC分野でも翻訳時間と実行時間のバランスを重視する利用者もいれば、実行時間の高速化を追求する利用者もいる。そのため、翻訳における思考時間の制限が問題視される時がある。

11. 最適化の進化

コンパイラの技術は、これらの問題への対策も検討されている。たとえば特定の要件に対する最善のパターンを事前に用意し、その要件がプログラムに含まれていた場合は、事前に用意した最善のパターンを適用する。これを「作業レベルの最適化［14］」という。

図19では、通常の最適化に加え、カレーをすばやく調理するための特殊な最適化（調理方法）、ボルシチをすばやく調理するための特殊な最適化（調理方法）が用意されている。そのため、この２つに関してはすばやく調理ができるが、シチューに関しては最善の実行速度が得られるとは限らない。これらの特定の要件を対象とした最適化は、コンパイラごとにポリシーをもって搭載されている。たとえば、料理におけるカレーのような一般的なものに対して搭載する場合もあるし、ボルシチのような日本では少数派となる層を狙って搭載するものもある。少数派を対象とすることは、システムとしての差別化であったり、特定の要人に向けた戦略的なアプローチで

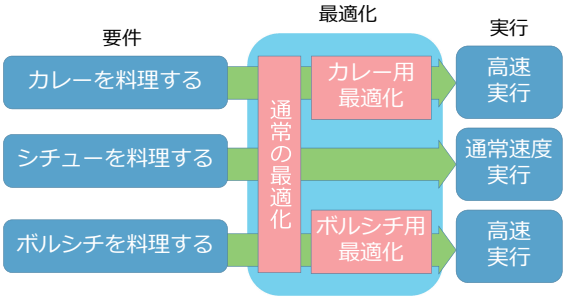

図19　作業レベルの最適化

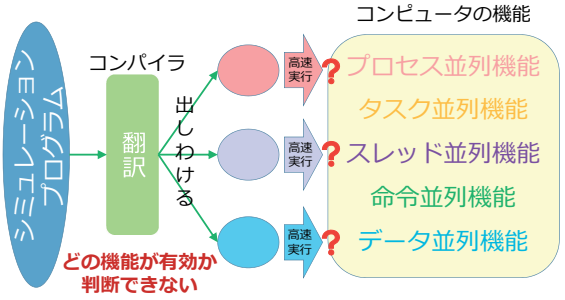

図20　高機能なスーパーコンピュータ

あったりするかもしれない。そのため、利用者は自身のプログラムの種類に応じて得意なコンパイラを選択する。

自然言語と同じようにプログラミング言語も言葉の組み合わせは無限大であり、全ての組み合わせに対して最善の最適化を適用することは不可能である。ただしHPC分野にフォーカスした時、プログラムの特徴を捉えることは可能である。プログラムで共通する特徴に対して作業レベルの最適化が拡充されることは、幅広いシミュレーションの高速化につながる。

12. コミュニケーションの強化

スーパーコンピュータは、現在もムーアの法則［15］に近似するように2年で約2倍の実行性能が向上している。この中でメニーコア・プロセッサ［16］やSIMD演算器［17］といった、プログラムの並列性を向上する新しいハードウェア技術が次々に誕生している。前述したようにハードウェア技術はコンパイラの最適化によって効率的に利用される。この作業はコンパイラによって暗黙的に行われるため、利用者はハードウェア技術の進化を大きく意識する必要がない。しかしながら、非常に多くのハードウェア技術が提供される昨今では、コンパイラはどの技術を使うことが高速化につながるかを判断することが難しくなってきている（図20）。

そのため、高機能なスーパーコンピュータを使いこなすためには、利用者とコンパイラもコミュニケーションを深める必要がある。たとえば、利用者がハードウェアを意識し、スーパーコンピュータ上でシミュレーションプログラムをどのように動かしたいか、コンパイラにどのような最適化を期待するかを指示することで、より効果的な最適化を適用することができる（図21）。

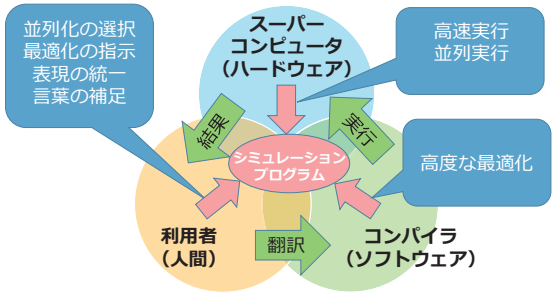

図21　コミュニケーションの強化

また、コンパイラにとって最適化しやすいシミュレーションプログラムを作成することも手段の一つである。たとえば、シミュレーションプログラムの内容をコンパイラが解析しやすいように言葉の順序を統一したり、言葉の意味や状態を補足したりすることである。

言葉の順序が統一されることでコンパイラはプログラムの意味を正確に理解しやすくなる。また、「すばやく」のように形容詞や副詞を付け加えて動作をより明確にすることで、翻訳者となるコンパイラがプログラム全体の動作がイメージできるようにする。たとえ

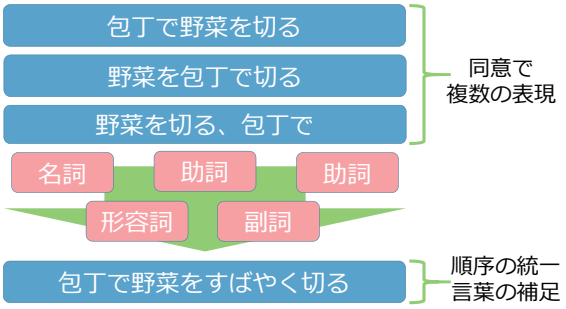

図22　順序の統一と言葉の補足

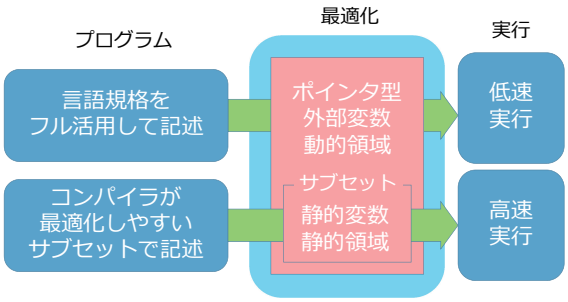

図23　規格の限定利用による高速化

ば、プログラミング言語のCは自由度の高い高水準言語である。そのため、図22のような同意となるプログラムの表現も多数存在する。この表現をコンパイラが翻訳しやすい形で記述することは有効である。また、それ以上に重要となるのが動作を明確にする修飾子である。たとえば、ポインタ型変数のアクセス先の領域が他の変数がアクセスする領域と重ならないことを指示するrestrictキーワードを付与し、明示的に変数間の依存関係を取り除くことで変数に対する最適化を促進することができる。

さらに、言葉を限定することでプログラムを高速化することも可能である。たとえば、プログラミング言語の中でもポインタ型変数や外部変数は他の変数との依存関係が不明確である。そのため、命令の並び替えなど依存関係を崩すような最適化が適用できない。そこで、できる限り依存関係が明示的になるような記述を行うことで最適化の阻害要因を削減し高速化に繋げることができる。

利用者がプログラムを作成する時、生産性を確保するためにプログラミング言語の規格をフル活用することは重要である。ただし、高い実行性能を要求する場合は、図23のようにプログラミングの生産性を犠牲にすることも有効である。生産性と実行性能のトレードオフはプログラムを作成する際に事前に決定しておく必要がある。

このように、HPC分野では利用者がコンパイラの動作を理解した上でプログラムを作成することは重要である。利用者とコンパイラの相互理解によって、スーパーコンピュータの性能を引き出すことができる。

13. まとめ

人間がスーパーコンピュータとコミュニケーションを図るためにはコンパイラが必要である。コンパイラは、人間がプログラミング言語を利用して作成したシミュレーションプログラムをスーパーコンピュータが理解できる言葉に翻訳する。スーパーコンピュータは、翻訳後のプログラムで理解したシミュレーションを実行し、その結果を通知する。これらの一連の流れがスーパーコンピュータとのコミュニケーションとなる。

これらのコミュニケーションは、人間同士のコミュニケーションと同じく相手を理解することが重要である。利用者はコンパイラを理解して効果的なシミュレーションプログラムを作成し、コンパイラは利用者の要求を理解し翻訳時に最善となる最適化を適用する。そして、スーパーコンピュータはシミュレーションを高速に処理する。質の高いコミュニケーションは、シミュレーションの実行時間の短縮や実験精度の向上という効果が得られる。今後も継続的に優れた研究成果を上げるためには、スーパーコンピュータとのコミュニケーションをより深めていく必要がある。

図24　コミュニケーションの強化

参考文献

［1］　スーパーコンピュータ「京」：独立行政法人　理化学研究所と富士通株式会社が共同で開発したスーパーコンピュータ、 理化学研究所計算科学研究機構（所在地：兵庫県神戸市）に設置

［2］　Top500：スーパーコンピュータにおける計算速度の世界ランキング

［3］　国立大学法人東北国立大学、株式会社富士通研究所：スパコンで高解像度な津波モデルを用いた浸水解析のリアルタイム化に成功、
http://www.tohoku.ac.jp/japanese/newimg/pressimg/tohokuuniv-press_20150227_01web.pdf
http://pr.fujitsu.com/jp/news/2015/02/27-1.html

［4］　モールス信号：通信分野で利用される可変長な符号化文字コード

［5］　LINE：スマートフォンやタブレットで利用できるアプリケーション、インスタントメッセージ、音声通話、グループチャットなどが利用可能なコミュニケーションツール

［6］　ポケットベル：日本電信電話公社によって1968年にサービスが開始された無線呼び出しサービス

［7］　ネットスラング：ネットワークを介したコミュニケーション、主にインターネットで使用されるスラングで、文字や記号を使った特殊な表現が用いられる。

［8］　ユーキャン新語・流行語大賞：自由国民社が１年を通して話題となった言葉を選び、 関係者を顕彰する賞

［9］　GNUコンパイラ：GCC(GNU Compiler Collection）で提供されるコンパイラ

［10］　プログラミングモデル：目的に応じてプログラムをモデル化したもので、通常はハードウェアの特徴に合わせて抽象化したモデルを指す

［11］　プログラミング技法：プログラムの設計や開発の効率を上げる目的で利用される技術

［12］　実験精度：実験の正確性を指す。スーパーコンピュータを利用したシミュレーションは物理的な実験に比べて制度は劣るが、実験コストや速さの面で非常に優れているといえる。

［13］　ハードウェア資源：ＣＰＵ、メモリ、

入出力機などの装置

［14］ 作業レベルの最適化：意味のある単位の命令列、または特定のアルゴリズムに対して適用する最適化で、機械語の出力時に特定のコードを出力するタイプ、プログラムのソースコード自体を内部的に加工するタイプ、特定の関数やライブラリ呼び出しに置き換えるタイプなど様々なタイプの最適化がある

［15］ ムーアの法則：1965年、Gordon E. Mooreが提唱した「半導体の集積密度は18－24カ月で2倍になる」という法則で、コンピュータの性能もこれに準じるとされるもの

［16］ メニーコア・プロセッサ：多くのコアを所有するプロセッサ

［17］ SIMD演算器：SIMD（Single Instruction Multiple Data）技術によって1命令によって複数のデータに対して同時に演算を行うことが可能な演算器