戦略的創造研究推進事業　SPPEXA(CREST)
研究領域「ポストペタスケール高性能計算に資する
システムソフトウェア技術の創出」
研究課題「ポストペタスケール時代のスーパーコン
ピューティング向けソフトウェア開発環境」

研究終了報告書

研究期間　平成28年1月～平成30年3月

研究代表者：千葉　滋
（東京大学情報理工学系研究科、教授）

§1 研究実施の概要

(1)実施概要

SPPEXA プロジェクトのひとつである ExaStencil プロジェクト内のひとつの研究グループとして研究を実施した。ExaStencil の目的はステンシル計算のためのドメイン専用言語 (DSL) を開発することである。この言語を用いることで、コンピュータ・プログラミングに詳しくないユーザでも、高度に最適化されたステンシル計算をおこなうアプリケーションプログラムをわかりやすく書けるようにする一方、そのプログラムの実行時には実行プラットフォームに合わせた高度な最適化をほどこして高速実行することを狙う。プロジェクトでは、そのような DSL として ExaSlang を開発しているが、これはいわゆる外部 DSL であり、コンパイラや開発環境として全てその専用のものを用いなければならない。

千葉の研究グループは、メタプログラミングの技術の研究を深化させ、この ExaSlang 相当のドメイン専用言語を埋め込み DSLとして実装することを狙った。埋め込みDSLであれば、アプリケーションプログラムの開発者は、ExaSlang が提供する高度な抽象化や最適化された実行時性能を、慣れた既存のプログラミング上で利用できるようになる。しかしながら任意の言語を埋め込み DSL として実装可能なわけではなく、現状の技術では ExaSlang のような DSL を実装することは困難である。本研究では、埋め込み DSL の実装技術を深化させるべく研究をおこなった。ExaSlang が実現する抽象化と同等の抽象化を埋め込み DSL で実現すること、ExaSlang が可能にする実行時性能に近い性能を埋め込み DSL でも可能にすること、が研究目的である。

ExaSlang はマルチグリッド法によるステンシル計算のための DSL であり、強力なステンシル演算子をアプリケーションプログラムの開発者が利用可能にする。アプリケーションプログラムの開発者は、グリッドの各要素に対してどのような計算をおこなうかを、ステンシル演算子などを用いて簡潔に記述する。ステンシル演算子は各要素の周辺の点に定数係数をかけるだけでなく、柔軟な計算をおこなうことができる。

千葉の研究グループでは、これまでの CREST 研究の成果である Bytespresso を用いて ExaSlang 相当の機能をもつ埋め込み DSL を Java 言語をホスト言語として開発した。この DSL によるアプリケーションプログラムの開発者は、Java 言語のクラスで表現されたステンシル演算子を用いてプログラムを書ける。この Java プログラムを Bytespresso の最適化機能を用いて効率の良い C 言語プログラムに変換、実行することで、元の ExaSlang に匹敵する記述力と実行時性能をもった埋め込み DSL を実現した。Bytespresso は、最適化機能の基盤として本研究で研究開発している技術 deep reification を用いる。この研究は CREST 研究の成果である Bytespresso の有用性の実証研究でもある。

またアプリケーションプログラムの開発者がより気軽に利用できるホスト言語として Ruby 言語を選び、これをホスト言語とする ExaSlang 相当の埋め込み DSL の開発もおこなった。Ruby 言語は動的に型付けされる言語であるので、型システムを用いる deep reification はそのままでは利用できない。Deep reification の研究を深め、Ruby 言語でも ExaSlang 相当の抽象化と実行時性能を利用可能にする。Ruby 言語は非コンピュータ技術者であっても気軽に使えるスクリプト言語であるというだけでなく、一般に広く使われている汎用言語の中では、非常に柔軟な構文を許す言語として知られている。このため Java 言語をホスト言語とする場合よりも、より元の ExaSlang に近い構文をアプリケーションプログラムの開発者が利用できるようになる。一方で Ruby 言語の実行時性能は Java 言語に比べると大きく劣るので、高い実行時性能を実現するのはより困難である。

(2)顕著な成果

＜優れた基礎研究としての成果＞

1. 埋め込み型ドメイン専用言語のための拡張可能構文機構

概要:
ホスト言語上のライブラリとして実現されるドメイン専用言語の構文上の自由度を増し、かつ実用的な時間での構文解析を可能にする手法を提案し、プログラミング言語分野の主要国際会議で発表することができた。この技術はライブラリによって簡易に HPC の特定ドメイン向けの DSL の開発できるようにし、ポストペタスケール時代の複雑化するプログラムの開発を大きく支援する可能性をもつ。
(CREST 研究からの引き継ぎ)

＜科学技術イノベーションに大きく寄与する成果＞
1. Java言語向け埋め込み型ドメイン専用言語開発基盤 Bytespresso
概要:
MPI、GPU を用いた高性能計算向けドメイン専用言語を、Java 言語をホスト言語としてそこに埋め込む、埋め込み型ドメイン専用言語として開発するためのソフトウェア基盤 Bytespresso を開発した。Bytespresso は、埋め込まれたドメイン専用言語のプログラムを抜き出し、効率のよい C あるいは CUDA プログラムに変換して実行するための処理系である。これにより lambda 式などを駆使する Java 風の構文をもち高速実行可能なドメイン専用言語を容易に開発できるようになった。
(CREST 研究からの引き継ぎ)

§2 研究実施体制

(1)研究チームの体制について

①「千葉」グループ
研究参加者

氏名	所属	役職	参加時期
千葉　滋	東京大学情報理工学系研究科	教授	H28.1～H.30.3
市川　和央	同上	学術支援職員	H28.1～H.30.2
山崎　徹郎	同上	博士課程学生	H29.4～H.30.3

研究項目
- スーパーコンピューティングのためのモジュール機構

(2)国内外の研究者や産業界等との連携によるネットワーク形成の状況について
ドイツ側共同研究者との間で日独の相互訪問を実施し、本 SPPEXA 研究開始前には存在しなかった国際研究ネットワークを形成した。また開発した Bytespresso を用いた民間企業との間の共同研究を実施し、実用化にむけた研究をおこなった。

§3 研究実施内容及び成果

3.1 スーパーコンピューティングのためのモジュール機構(東京大学　千葉グループ)
(1)研究実施内容及び成果

これまでの CREST 研究の成果である、HPC 向けドメイン専用言語やライブラリなどのソフトウェアの開発基盤である Bytespresso の開発を継続した。Bytespresso の基本性能を明らかにするために、NAS Parallel ベンチマークより CG と LU の MPI 版を選び Java 言語に移植し、Bytespresso 上で動作するようにした。LU については元の Fortran 版と言語の違いを除き、ほぼ同等のプログラムであるが、CG については移植にあたって行列オブジェクトを用いる形に大きく書き直した。この行列オブジェクトは異なるノードにまたがって分散配置され、アプリケーションプログラマは、MPI によるノード間通信を気にせず、行列やベクトルのかけ算、足し算という抽象レベルでプログラムを書ける。例えば CG の核となる部分は次のようになる。

```
q.setToMult(a, p);          // q = A.p
double d = inner(p, q);     // d = p･q
double alpha = rho / d;     // α = ρ / d
z.setToAdd(z, alpha, p);    // z = z + α*p
r.setToAdd(r, -alpha, q);   // r = r - α*q
double rho0 = rho;          // ρ0 = ρ
rho = r.norm();             // ρ = ρ･ρ
double beta = rho / rho0;   // β = ρ/ρ0
p.setToAdd(r, beta, p);     // p = r + β*p
```

これにより、CG のプログラムは、行列やベクトルのライブラリ部分を除くと、元の Fortran 版の半分以下の行数となった。実行時性能は、Fortran 版と同等ではないが、それに匹敵する性能を実現している。Bytespresso は github.com より一般に公開されており、利用のためのチュートリアルも英語・日本語で用意されている。公開された Bytespresso を利用した民間企業との共同研究も始まっている。

CG・LU の例では、Fortran 版に匹敵する実行時性能を実現する行列・ベクトルのライブラリが、Bytespresso 上に実装された埋め込み DSL ということになる。Bytespresso は埋め込み DSL の実装基盤であるが、このような高速ライブラリを実装するための基盤であるともいえる。CG のために実装した行列・ベクトルライブラリを他の MPI アプリケーションの作成に用いることもでき、その場合は、そのアプリケーションの性能も同様に高速化される。Bytespresso は高速化のため、Java プログラムに埋め込まれた DSL 部分を切り出し、C 言語プログラムに変換する機能を提供する。したがって Java 言語で書かれた CG プログラムの大半は、Bytespresso によって MPI を用いた C 言語プログラムに変換され、コンパイル、実行される。

Bytespresso の機能上の特徴は、Java 言語プログラムに埋め込まれた DSL 部分を切り出す機能である。我々はこれを deep reification と呼ぶ。この機能は Lisp 系の言語に採用されているマクロがマクロ式の構文木を切り出してマクロ関数に渡す機能に近い。しかし従来のマクロ機能と異なり、マクロ式に相当する式が他のメソッドを呼んでいる場合、呼ばれたメソッドの本体の構文木も切り出す。動的呼び出しにより呼ばれたメソッドが一意に定まらない場合は、可能な全てのメソッドの本体の構文木を列挙する。これにより埋め込まれた DSL プログラム全体を切り出すことができる。また、切り出された構文木に対して C++の template に相当する最適化をほどこす機能も Bytespresso は備えており、アプリケーションプログラムの開発者は C++ template ライブラリに近い最適化を template のような特別な記述なしに利用できる。

本研究では、この Bytespresso を用いて ExaSlang 相当の埋め込み DSL を開発した。ExaSlang は、SPPEXA プロジェクトの一つで本研究が属す ExaStencil プロジェクトが開発する外部 DSL である。外部 DSL であるので、専用のエディタ（ないしは ExaSlang 向けの支援のない汎用エディタ）や専用のコンパイラなど、専用の開発環境を用いてプログラムを書き、実行する。本研究の目的は deep reification 等 Bytespresso の機能を駆使することで、そのような外部 DSL に性能上も匹敵する埋め込み DSL を実現できることを示すのが狙いである。それが可能であれば、アプリケーションプログラ

ムの開発者は慣れ親しんだ汎用言語の開発環境を用いながら、DSL の優れた抽象化、高速な実行を享受できる。

ExaSlang のステンシル演算子はオブジェクトで表現可能なので、ExaSlang の実行時性能に匹敵する性能がでるかが研究の焦点となる。ExaSlang では次のようなステンシル演算子を定義できる。

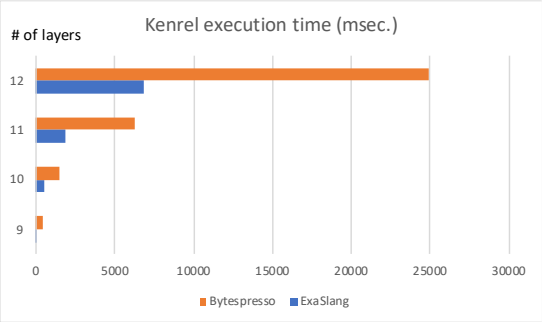
```
Stencil SmootherStencil_v@all {
  [ 1,  0] => -1.0
  [-1,  0] => -1.0
  [ 0,  1] => -1.0
  [ 0, -1] => -1.0
  [ 0,  0] => 4.0*alpha + GradientY@current * GradientY@current
}
```

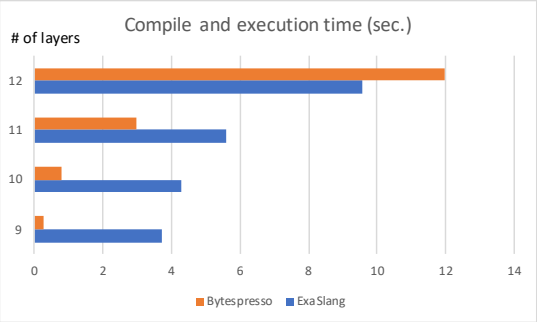
これは5点ステンシルであるが、中央の値の係数は定数ではなく、その時点での Y 方向の勾配 `GradientY@current` の関数である。Bytespresso 版の DSL では、これは例えば次のように定義することができる。

```
smoother = new Stencil().add( 1, 0, -1.0)
                        .add(-1, 0, -1.0)
                        .add( 0, 1, -1.0)
                        .add( 0,-1, -1.0)
                        .add( 0, 0, (i)-> 4.0*alpha+gradY.get(i)*gradY.get(i))
```

ここで `gradY` は勾配の値の配列である。係数をラムダ式で表しているため、高速化のためには C++ template のような適切なインライン展開が必要になる。そこで高速化のためにユーザがメソッドの形でステンシル演算子を定義する機能も提供している。

Bytespresso 版の ExaSlang とオリジナルの ExaSlang との性能比較を以下に示す。マルチグリッド法により2次元オプティカルフロー推定をおこなうプログラムをそれぞれで実装し、実行した。実行には Xeon E5-2637v3 を搭載した PC 上でおこなった。コンパイラは gcc 5.4.0 である。どちらも CPU のみを用いて計算をおこなう。





Bytespresso 版はオリジナルに比べてカーネル計算の実行速度は4分の1から3分の1であった。一方、Bytespresso 版はコンパイル時間は短いため、コンパイル時間と実行時間を合わせると、右のグラフのようになり、小規模な計算では Bytespresso 版の方がむしろ高速である。Bytespresso 版がオリジナルと同等の実行速度を達成できない理由は、オリジナルが実施する各レイヤごとに各ステンシル演算子等を特化する最適化を Bytespresso 版では行えていないためであると推測される。一方、Bytespresso 版の DSL の開発コストは、素朴な Java 版の埋め込み DSL の実装コストと同等であり、外部 DSL であるオリジナルに比べると非常に小さい。小さな開発コストで高コストな外部 DSL

版の4分の1から3分の1の実行速度を実現できているので、実用上は十分有益であると考えられる。

Bytespresso版の ExaSlang の開発のためドイツ側の共同研究者と連絡を取り合って研究を進めた。具体的には表1のように相互訪問をおこなった。

表1

2016 年 1 月	Munich	SPPEXA Symposium にて打合せ
2016 年 3 月	Erlangen	千葉が FAU のグループを訪問
2016 年 9 月	東京	Prof. Matthias Bolten が東大を訪問
2017 年 9 月	Santiago Compostela	Prof. Christian Lengaue と千葉が EuroPar'17 で打合せ
2017 年 2 月 2017 年 3 月	Erlangen Munich	千葉が FAU のグループを訪問 SPPEXA Symposium にて打合せ
2017 年 10 月	東京	Christian Schmit、Sebastian Kuckuk が FAU より東大を訪問
2017 年 12 月	東京	Prof. Matthias Bolten が東大を訪問
2018 年 3 月	東京	Sebastian Kuckuk が FAU より東大を訪問

Bytespresso 版の ExaSlang の開発と並行して、Bytespresso の技術を Ruby 言語上で展開する方法についても研究をおこなった。Java 言語は静的に型付けされる言語であり、deep reification も静的型を利用しているが、Ruby 言語は動的に型付けされるスクリプト言語であるので deep reification の技術も進化させなければならない。しかしながら、deep reification が Ruby 言語でも利用可能になり、ExaSlang の埋め込み DSL 版が Ruby 言語をホスト言語として実現できるようになると、より広範囲の利用者が見込める。Bytespresso は Java 言語を対象としているが、現在、HPC 界では低レベルだが高速な言語として Fortran, C (あるいは C++)が使われる一方、低速だが高い抽象レベルの言語としては Python, Ruby が注目されている。両者の中間に位置する Java 言語は、埋め込み DSL 技術の研究開発のプラットフォームとしては適しているが、実用システムのプラットフォームとしては、あまり適しているとはいいがたい。Ruby 版の開発により、CREST 研究で研究開発してきた技術を、将来的により広範囲へ普及させ、エクサスケール時代の HPC ソフトウェア開発に資する。

開発した Ruby 版の ExaSlang の埋込み DSL は、オリジナルの ExaSlang と非常によく似た構文や抽象データを Ruby 上で提供する。開発した Ruby 版の DSL で書かれた Ruby プログラムを実行すると、ExaSlang で書かれたソースコードが出力されるので、それを既存の ExaSlang コンパイラでコンパイルし実行できる。Ruby 版 DSL で書かれたプログラムはあくまで正規の Ruby プログラムであるので、前処理計算などは Ruby プログラムの実行中に済まし、高速実行したい部分だけを ExaSlang として実行させることができる。あた Ruby が備えるモジュール機構などを用いてプログラムを見通しよく書くこともできる。

以下は開発した DSL で書かれた Ruby プログラムでステンシル演算子を定義している。元の ExaSlang とほぼ同等の抽象化を提供している。

```
SmootherStencil_v = stencil(All,
  [ 1,  0] => -1.0,
  [-1,  0] => -1.0,
  [ 0,  1] => -1.0,
```

```
  [ 0, -1] => -1.0,
  [ 0,  0] => -> () { 4.0 * Alpha + GradientY.current * GradientY.current }
)
```

以下は関数定義の一部である。

```
Fragments.loop_over do
  Flow_u.current.loop_over do
    Flow_u[:next].current = Flow_u[:active].current +
        ((1.0 / diag(SmootherStencil_u.current)) *
         (RHS_u.current - SmootherStencil_u.current * Flow_u[:active].current -
             GradientX.current * GradientY.current * Flow_v[:active].current))
  end
end
```

元の ExaSlang がもつダブルバッファリングの機能やレベルの概念を同様の記述で可能にしている。

開発したソフトウェアのうち公開するものは以下のとおり。

公開済:
Bytespresso https://github.com/csg-tokyo/bytespresso
HPC 向け埋込みドメイン専用言語やライブラリを開発するための Java 言語用基盤。Java to C/MPI への変換機能を内蔵し、Java 言語のプログラムとして書かれた DSL コードを高速に実行できる。Java 8 以上が動作する計算機上で動作する。潜在ユーザは抽象的な記述を提供する HPC 向けライブラリや埋込みドメイン専用言語の開発者。

公開予定:
Yadriggy https://github.com/csg-tokyo/
HPC 向け埋込みドメイン専用言語やライブラリを開発するための Ruby 言語用基盤。Ruby の高度な構文を駆使して書かれた DSL プログラムを C 言語に実行時変換して高速実行できる。Ruby 言語を実行できる任意の計算機上で利用可能。潜在ユーザは抽象的かつ簡潔な記述で HPC プログラムを書きたいプログラマ。

現在でもペタスケール以上の性能を実現するため、スーパーコンピュータのハードゥエアおよび基盤ソフトウェアは、一般的なデスクトップ PC のハードウェアやソフトウェアから大きく離れたものとなっており、アプリケーションの開発者の負担は大きい。広い意味での仮想化により、アプリケーション開発者はそれをあまり意識しないでプログラムを書くことも可能だが、性能を追求する場合は、そのような仮想化は効果的ではない。例えば各ノード計算機には物理的なディスクがないことが多いので、それの存在を仮定してプログラムを書くと、ディスクアクセスが遠隔の中央ファイルシステムへのアクセスとなり、性能の低下をまねく。

これを避けるためには、過度に汎用的な仮想化ではなく、アプリケーションの分野ごとに最適化されたライブラリや埋め込み DSL を用いて開発する方がよい。これはアプリケーションと基盤的なソフトウェアの間の co-design であり、本研究の成果はステンシル計算を例として、そのような co-design を促進する基礎技術を提供する。Bytespresso のようなソフトウェアを用いることで、高性能な埋め込み DSL の開発コストが低減するので、様々なアプリケーション分野ごとに個別にライブラリや DSL を開発することが実行しやすくなる。

§4 成果発表等

(1)原著論文発表　(国内(和文)誌　0　件、国際（欧文）誌　3　件)

1. Shigeru Chiba, YungYu Zhuang, Maximilian Scherr, Deeply Reifying Running Code for Constructing a Domain-Specific Language, Proc. of the 13th International Conference on Principles and Practices of Programming on the Java Platform: Virtual Machines, Languages, and Tools (PPPJ'16), Article No. 1, August 2016.
2. Thanh-Chung Dao and Shigeru Chiba, HPC-Reuse: efficient process creation for running MPI and Hadoop MapReduce on supercomputers, 16th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, pp.342-345, May 2016. (short paper)
3. Thanh-Chung Dao and Shigeru Chiba, SEMem: Deployment of MPI-Based In-Memory Storage for Hadoop on Supercomputers, Proc. Of Euro-Par 2017, pp.442-454, 2017.

(2)その他の著作物(総説、書籍など)

なし

(3)国際学会発表及び主要な国内学会発表

① 招待講演　　(国内会議　0　件、国際会議　0　件)

なし

② 口頭発表　　(国内会議　1　件、国際会議　2　件)

1. Thanh-Chung Dao (東大), Shigeru Chiba (東大), In-memory Hadoop on super-computers using external memory of additional nodes, 日本ソフトウェア科学会第 33 回大会, 東北大学, 2016 年 9 月 6 日～9 日.
2. Shigeru Chiba, Toward embedded domain-specific languages for supercomputing, JST/CREST International Symposium on Post Petascale System Software, 2018.
3. Shigeru Chbia, Implementation Techniques of Domain-Specific Languages for Parallel Solvers, 18th SIAM Conference on Parallel Processing for Scientific Computing, 2018.

③ ポスター発表　　(国内会議　0　件、国際会議　0　件)

なし

(4)知財出願

①国内出願 (0件)

なし

②海外出願 (0件)

なし

③その他の知的財産権

(5)受賞・報道等

①受賞

Thanh-Chung Dao, 日本ソフトウェア科学会第 33 回大会学生奨励賞受賞(In-memory Hadoop on super- computers using external memory of additional nodes)

②マスコミ(新聞・TV等)報道
なし

③その他
なし

(6)成果展開事例

①実用化に向けての展開

＜公開可能なもの＞

・本研究で開発した Bytespresso を用いた実用的な DSL 開発について、民間企業と共同研究中

②社会還元的な展開活動

なし

§５ 研究期間中の活動

５．１　主なワークショップ、シンポジウム、アウトリーチ等の活動

特になし