サイエンスと技術のエンゲージメント

スーパー・コンピューターが拓く未来

サイエンスの進歩はスーパー・コンピューターの技術に出会うことにより加速しています。従来どおりの高性能計算に加え、最近ではビッグデータを用いた高性能分析の領域がスーパー・コンピューターの主要な用途になっており、IBMではこれらのワークロードをサポートするために「データ・セントリック・システム」というアーキテクチャーを提唱しています。本稿では、この思想に基づいて設計された「Minsky」システムとその特長、それを利用することでアプリケーションがどのように高速処理されるかについて解説します。

1. はじめに

スーパー・コンピューターはいわゆる高性能計算（HPC：High Performance Computing）をより大規模に、より高速に実行するために進化を続け、サイエンスの発展に貢献してきました。その計算能力は1秒間に実行することができる浮動小数点演算の数（FLOPS：Floating Operations Per Second）によって示されるのが一般的です。2008年にIBMのRoadrunnerが初めてペタFLOPS（1秒あたり1兆の千倍の演算数）の大台を超え、2017年現在では、エクサFLOPS（1秒あたり1兆の百万倍の演算数）を目指して、日米中を中心に各国で熾烈な開発競争が行われています。HPCが貢献するサイエンスの分野は多岐にわたり、量子力学、天候予測、石油探索、分子モデリング、物理シミュレーションなどが挙げられます。例えば高エネルギー加速器研究機構（KEK）では、量子色力学における自発的対称性の破れの現象を厳密な計算機シミュレーションにより世界で初めて実証し[1]、これには「IBM Blue Gene」が活用されました。

2000年代になりビッグデータ時代に突入すると、大容量のデータに巨大な計算能力を投入して処理し、隠れた知見を発見することが盛んになってきました。高性能分析（HPA：High Performance Analytics）とも呼ばれる分野の誕生で、計算能力とデータ量は相呼応し、さらなるデータはさらなる計算を、さらなる計算はさらなるデータを求める時代となっています。こうした中で特に深層学習（ディープ・ラーニング）は長足の進歩を遂げ、音声認識、画像処理、自然言語処理などに応用され画期的な成果を上げています。

HPCとHPAは別物ですが、それぞれに特化した専用の計算機を設計するのは経済的に得策ではありません。両者を包摂する形で計算機アーキテクチャーを設計するのが業界の方向性であり、IBMではそうしたシステムを「データ・セントリック・システム（DCS：Data Centric System）」（以下、DCSシステム）と呼んでいます。本稿

IBM Power System S822LC for HPC (“Minsky”)
– IBM POWER8 CPU x2 + NVIDIA P100GPU x4

図1. Minskyシステムのノード

では、DCSシステムの紹介をするとともに、その上でアプリケーションがどのように高速処理されるかについて解説します。

2. DCSシステム

DCSシステムは、ビッグデータのための新世代アーキテクチャーとして2011年のIBM Global Technology Outlook（IBM Researchが毎年作成している技術戦略策定文書）で初めて提案され、その後2014年にかけてHPCとHPAの双方を包摂する形で具体化されました。設計原理として、①データの移動の最小化、②メモリーおよびネットワークを含むシステム階層のすべてのレベルに「計算」を導入、③1ラック未満から数百ラックまでの構成に対応するモジュラーでアップグレード可能な設計、④具体的なアプリケーションに基づく設計、が掲げられています[2]。

DCSシステムは、OpenPOWERファウンデーション[3]から供給されるテクノロジーを最大限活用して設計開発が進められています。OpenPOWERファウンデーションは、POWERアーキテクチャーを用いたシステムのイノベーションを加速するために、IBMをはじめGoogle、NVIDIAなどを中心に2013年に設立されました。ソフトウェアで定着したオープン・ソースのモデルをハードウェアにも拡張し、オープンなエコ・システムによってフルスタックのイノベーションを推進することを目的としており、現在では300を越えるメンバーが参加しています。

DCSシステムの具体例として、2016年9月に発表されたIBM Power System S822LC for High Performance Computing（通称「Minsky」）が挙げられます（**図1**）[4]。このシステムは1ノードに2つのCPU（IBMのPOWER8プロセッサー）と4つのGPU（NVIDIAのTesla P100）を搭載しており、その構成は**図2**のようになっています。特筆すべきはPOWER8 CPUとP100 GPUがNVLinkと呼ばれる高速なリンクで接続されていることです。これにより、CPUとGPUは40GB/sの通信を双方向に行うことができ、CPU側に存在する大容量のデータをGPUに高速に供給して処理することが可能になっています。

Minskyの後継機として2017年中に、次世代のCPU（POWER9プロセッサー）とGPU（NVIDIAのTesla V100）を搭載したシステムが発表されます。POWER9 CPUには、マイクロ・アーキテクチャー、キャッシュ・メモリー階層、I/Oサブシステムなど、システム全般にわたって多くの革新的テクノロジーが採用されています。また、V100 GPUも、深層学習を加速するTensorCoreなどが追加され、さらなる高速化が行われています。これらを接続するNVLinkもバージョン2.0となり、さらなる高速化とコヒーレント・アクセス機能の追加などが行われます。この次世代POWERシステムは、米国エネルギー省が進めている「CORAL」プロジェクトにおいて、2つの国立研究所（ローレンス・リヴァ

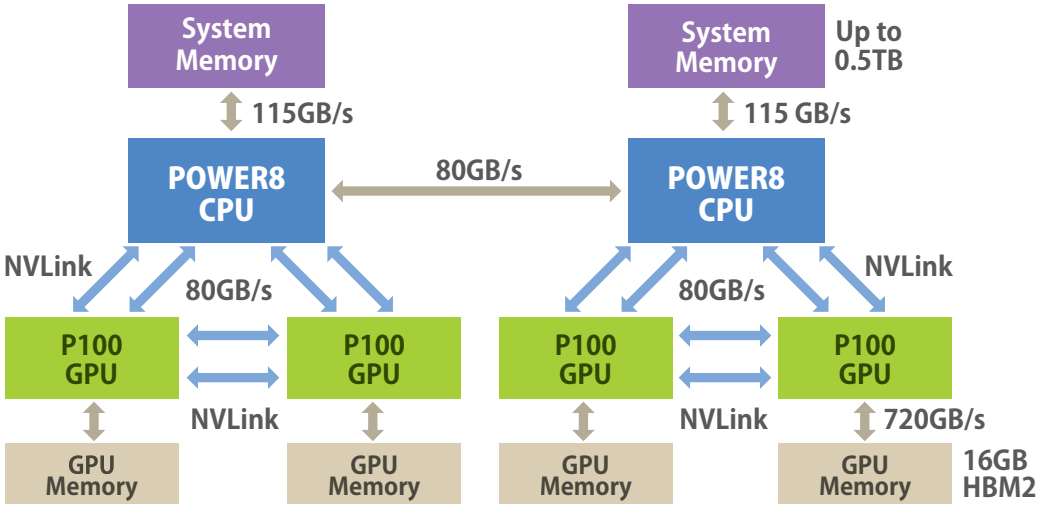

図2. Minskyの内部アーキテクチャー

モア国立研究所とオークリッジ国立研究所）に設置されるスーパー・コンピューターの計算ノードとしても用いられることになっています[5]。

本稿の後半では、このPOWERシステム技術のサイエンス向け活用事例について、HPCアプリケーションの高速化と深層学習フレームワークの最適化の2つに関して、IBM東京基礎研究所での取り組みを中心に紹介します。

3. HPCアプリケーションの高速化

大規模シミュレーションなどのHPCアプリケーションは従来、CPU主体の分散並列環境において実行されてきました。IBMの提供してきたスーパー・コンピューターであるBlue Gene、日本の地球シミュレーターや京コンピューターもその例といえます。次世代スーパー・コンピューターの活用には、これを今後の主流である加速プロセッサー（GPU）環境に対応させる作業が必要になります。もちろん、専門家がGPU向けプログラミング環境（CUDA）を用いて直接GPUを利用するコードに修正するということも可能ですが、既存のさまざまなHPCアプリケーションを書き換えるのは大きな手間がかかります。

この「GPU移行コスト」を最小化する方法の一つが、OpenMP[6]（バージョン4以降）やOpenACC[7]といった注釈ベースのGPUプログラミング環境を用いることです。これらは、既存アプリケーションのデータ構造とループ部分に注釈を追加し、GPUへのデータ移動やGPUでの実行を指定するというものです。**図3**に、HPCアプリケーションの一つである「姫野ベンチマーク」[8]にOpenACCに沿った注釈を入れてGPU上で実行されるようにした例を示します。左側がオリジナルのFORTRANで書かれたCPU用プログラム、右側がGPU実行のための注釈（紫の文字）を追加したものですが、ごく少量の注釈を加えるだけでGPU化できていることがお分かりいただけると思います。アルゴリズムを記述するプログラム本体部分には手を加えていないため、保守性が高いのもこの手法の特長です。

このような注釈ベースのGPU化では従来、CPUとGPUの間のデータ移動も指定する必要があり、正しさと性能を引き出すためにはデータの使われ方を十分理解した上での注釈付けが必要でした。しかし、「ユニファイド・メモリー」機構を活用することで、これを不要にすることが可能です。これは、アプリケーションが使用するデータ領域をCPUからもGPUからもアクセスできる形で確

```
!$OMP PARALLEL DO PRIVATE(i,j,k,s0,ss) REDUCTION(+:wgosa)
do k=2,kmax-1
  do j=2,jmax-1
    do i=2,imax-1
      s0=a(I,J,K,1)*p(I+1,J,K) +a(I,J,K,2)*p(I,J+1,K)+a(I,J,K,3)*p(I,J,K+1) &
        +b(I,J,K,1)*(p(I+1,J+1,K)-p(I+1,J-1,K)-p(I-1,J+1,K)+p(I-1,J-1,K)) &
        +b(I,J,K,2)*(p(I,J+1,K+1)-p(I,J-1,K+1)-p(I,J+1,K-1)+p(I,J-1,K-1)) &
        +b(I,J,K,3)*(p(I+1,J,K+1)-p(I-1,J,K+1)-p(I+1,J,K-1)+p(I-1,J,K-1)) &
        +c(I,J,K,1)*p(I-1,J,K)+c(I,J,K,2)*p(I,J-1,K) &
        +c(I,J,K,3)*p(I,J,K-1)+wrk1(I,J,K)
      ss=(s0*a(I,J,K,4)-p(I,J,K))*bnd(I,J,K)
      wgosa=wgosa+ss*ss
      wrk2(I,J,K)=p(I,J,K)+omega*ss
    enddo
  enddo
enddo
!$OMP END PARALLEL DO
!$OMP PARALLEL DO
 do k=2,kmax-1
   p(2:imax-1,2:jmax-1,k)= wrk2(2:imax-1,2:jmax-1,k)
 end do
!$OMP END PARALLEL DO
```

●KERNELS 指示文を入れる
●!$OMP PARALLEL DOを !$ACC LOOP INDEPENDENTに換える
●三重ループを COLLAPSE（3）で最大限に並列化

```
!$ACC KERNELS
!$ACC LOOP INDEPENDENT COLLAPSE(3) PRIVATE(i,j,k,s0,ss) REDUCTION(+:gosa)
 do k=2,kmax-1
   do j=2,jmax-1
     do i=2,imax-1
       s0=a(I,J,K,1)*p(I+1,J,K) +a(I,J,K,2)*p(I,J+1,K)+a(I,J,K,3)*p(I,J,K+1) &
          +b(I,J,K,1)*(p(I+1,J+1,K)-p(I+1,J-1,K)-p(I-1,J+1,K)+p(I-1,J-1,K)) &
          +b(I,J,K,2)*(p(I,J+1,K+1)-p(I,J-1,K+1)-p(I,J+1,K-1)+p(I,J-1,K-1)) &
          +b(I,J,K,3)*(p(I+1,J,K+1)-p(I-1,J,K+1)-p(I+1,J,K-1)+p(I-1,J,K-1)) &
          +c(I,J,K,1)*p(I-1,J,K)+c(I,J,K,2)*p(I,J-1,K) &
          +c(I,J,K,3)*p(I,J,K-1)+wrk1(I,J,K)
       ss=(s0*a(I,J,K,4)-p(I,J,K))*bnd(I,J,K)
       wgosa=wgosa+ss*ss
       wrk2(I,J,K)=p(I,J,K)+omega*ss
     enddo
   enddo
 enddo
!$ACC END LOOP
!$ACC LOOP INDEPENDENT COLLAPSE(3) PRIVATE(i,j,k)
 do k=2,kmax-1
    do j=2,jmax-1
      do i=2,imax-1
        p(i,j,k)= wrk2(i,j,k)
      enddo
    enddo
  end do
!$ACC END LOOP
!$ACC END KERNELS
```

図3. OpenACCとユニファイド・メモリーを活用したGPU化の例

保できるようにするもので、データ転送はCPUやGPUからアクセスが行われた際に必要に応じて自動的に行われます。まさに、データ・セントリック・コンピューティングに適した機能であり、これによりユーザーはデータ移動を意識せずにGPU化を行うことができるようになります。**図3**の例ではこの機能を利用しているため、データ移動に関する注釈は用いられていません。

このように、OpenACCなどの注釈プログラミングとユニファイド・メモリー機能を同時利用することで、非常に簡単に既存アプリケーションのGPU化が行えます。IBM東京基礎研究所の最新の研究[9]では、この手法でGPU化を行ったHPCアプリケーションは、手作業でデータ移動まで指定したGPU化よりもPOWERシステムでは高速な場合があるという驚くべき結果が得られています。**図4(a)**がその例ですが、これは上に述べた姫野ベンチマーク（問題サイズXL）を、①CPU上で実行、②OpenACCとユニファイド・メモリーを用いてGPU上で実行（ここで述べた手法）、③OpenACCの注釈でデータ移動も明示的に指定してGPU上で実行、した各場合のMinsky上での性能（FLOPS値）を比較したものです。GPU化により性能が10倍以上になっているだけでなく、ユニファイド・メモリーを用いた方が高速実行できていることが分かります。これは、ユニファイド・メモリーではデータ全体でなく必要な部分だけがページ単位で転送されることに加え、POWERシステムはCPU-GPU間のリンクが高速であることが大きく寄与している結果です。**図4(b)**は同じプログラム群をCPUとGPUがPCI Expressで接続された機器（ノード数やGPU数は同等）で実行した場合の性能です。GPUの種類と個数が同等であるにもかかわらず、このリンクが低速なためデータ移動がボトルネックとなり、十分な高速実行ができていないことが分かります。

IBMではこの手法などを活用し、お客様の既存HPCアプリケーションを、保守性を損なわずGPU向けに高速化するお手伝いも行っています。

4. 深層学習フレームワークの最適化

サイエンスを支える技術として最近脚光を浴びてきているのが、コグニティブ・コンピューティングの分野です。特に、ニューラル・ネットワークを用いた深層学習は、画像認識や自動車の自動運転などの実用化の基礎となる技術ですが、その学習処理は膨大な計算時間がかかり、それ

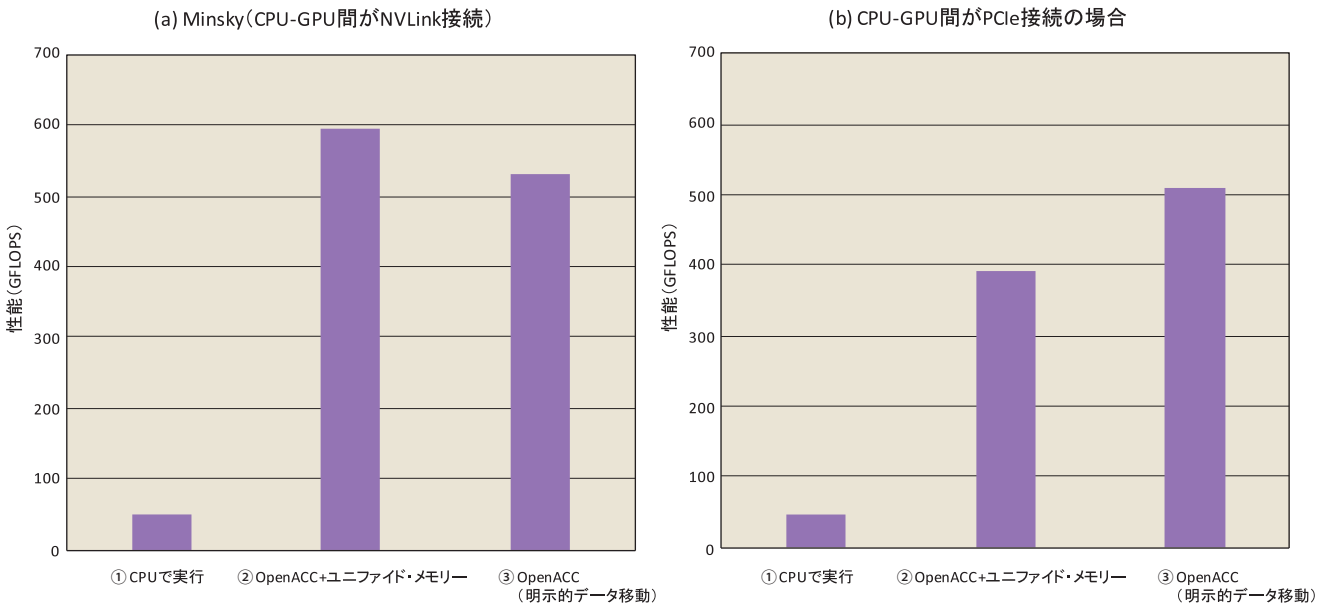

図4. 性能比較

を高速に行えるようにすることは非常に重要です。深層学習処理には、Caffe、Chainer、TensorFlow、Theano、Torchなどのオープン・ソースのフレームワークを利用するのが一般的で、これらはいずれもGPUを使った高速な学習処理が可能です。IBMでも、これらのフレームワークをMinskyのようなGPUを搭載したPOWERシステム向けに構成・最適化したものを「PowerAI」として提供しており[10]、無料ですぐに使えるようにしています。**図5**は、PowerAIに含まれるソフトウェア群を示したものです。

PowerAIに含まれる深層学習フレームワークは、POWERシステム向けに最適化されており、その一つが学習処理の一部にPOWER CPUを活用するというものです[11]。複数のGPUを利用して学習を行う場合、各GPUは同じニューラル・ネットワークを持ち、異なる入力データ（画像など）に対してそれぞれ学習を行うという「データ・パラレル」方式が一般的です。この方式では、各GPUにおいて学習した内容を定期的に集計する必要がありますが、PowerAIに含まれる「IBM Caffe」では、GPUによる学習処理と並行してPOWER CPUがこの集計処理を行うことで高速化を行っています（**図6**）[12]。この処理のためには、各GPUからCPUに学習内容（パラメーター）を送り、CPUから各GPUに集計後の新しいパラメーターを送る必要がありますが、CPU-GPU間がNVLinkにより高速接続されていることにより、この転送がボトルネックとなることを回避しています。

また、最新のPowerAIパッケージであるリリース4.0では、GPUメモリーに収まりきらない大きなニューラル・ネットワーク・モデルや、IBM独自の通信ライブラリー「DDL（Distributed Deep Learning）」を用いた大規模分散学習[13]をサポートしています。これらの機能でも、GPU上にある学習パラメーターをCPU側に退避したり、他ノードのGPUと交換したりする必要があるため、CPU-GPU間のリンクが高速であることが非常に重要となります。

さて、深層学習を用いる学習においては、フレームワーク自体が高速であることに加え、学習に用いるデータをどのように用意するかということも重要になります。この問題に対しては、GUIベースの画像解析ツールやApache Sparkを用いたデータ供給システムなどを提供していく予定です。

5. おわりに

本稿では、サイエンスと技術のエンゲージメントにおいて必須ともいえるスーパー・コンピューター活用について、既存のHPCアプリケーションの高速化手法と、深層学習フレームワークの最適化事例について紹介しました。IBMの提案するDCSアーキテクチャーとその具体例であるPOWERシステム技術が、サイエンスを進めるためのこれらの処理に適した構成になっていること、また研究所においてそれを最大限活用できる取り組みをしていることをご理解いただけたことと思います。

DCSシステムは今後も進化し、より大規模なHPCお

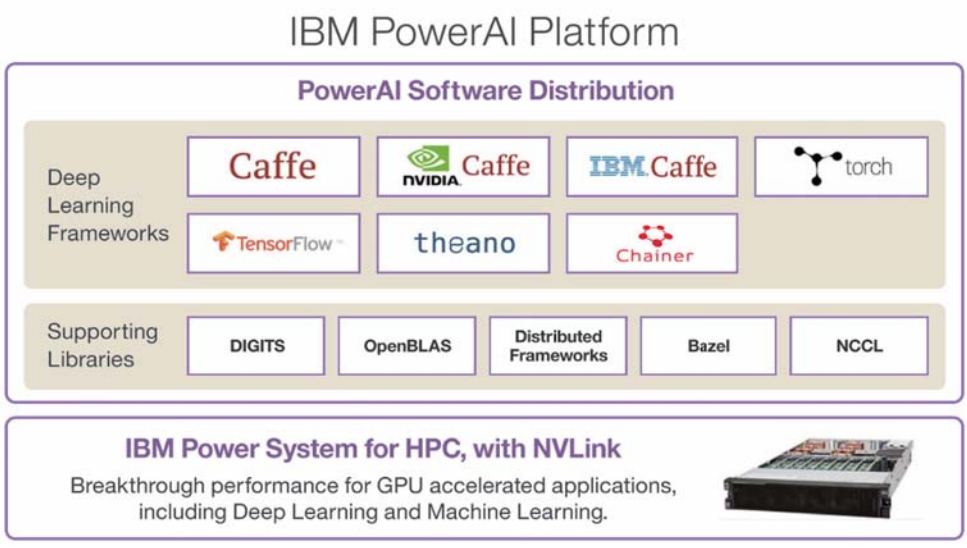

図5. PowerAIに含まれるソフトウェア群

よびHPAをより高速に実行し、サイエンスの発展に貢献していくでしょう。とりわけHPA、さらにはその中でも機械学習・深層学習に対する産学の取り組みは急速に拡大しており、DCSシステムの進化のベクトルを規定していくと考えられます。具体的には、これらに必要な浮動小数点演算を必要十分な精度で高速に実行する、いわゆるAIチップ、あるいは、人間の脳を模した超省電力のニューロモルフィック・チップなどの研究開発が進み、DCSシステムの主要コンポーネントとして組み込まれていくと思われます。すなわち、システムのヘテロ化が進んでいくのは止めようもなく、10年後のシステムは今とは随分違ったものになっていることでしょう。

[参考文献]

[1] 高エネルギー加速器研究機構，京都大学：量子色力学における自発的対称性の破れを厳密に実証（プレスリリース），https://www2.kek.jp/ja/news/press/2007/supercomputer2.html(2007).

[2] Tilak Agerwala：Data Centric Systems: The Next Paradigm in Computing，Keynote Lecture, 43rd International Conference on Parallel Processing (ICPP)(2014).

[3] OpenPOWER Foundation：http://openpowerfoundation.org

[4] IBM：IBM Power System S822LC for High Performance Computing，http://www.ibm.com/systems/jp-ja/power/hardware/s822lc-hpc(2016).

[5] IBM：米国エネルギー省、研究の推進とビッグデータの課題に対処するため、IBMのデータ・セントリック・システムを採用（ニュースリリース），http://www.ibm.com/press/jp/ja/pressrelease/48733.wss(2014).

[6] OpenMP：OpenMP, Enabling HPC since 1997，http://openmp.org

[7] OpenACC：OpenACC, More Science, Less Programming，http://openacc.org

[8] 理化学研究所 情報基盤センター：姫野ベンチマーク，http://accc.riken.jp/supercom/himenobmt

[9] 土井淳：NVLinkにおけるUnified MemoryとOpenACCによるプログラミングの性能評価，情報処理学会研究報告，Vol. 2017-HPC-159，No. 5（2017).

[10] IBM：IBM PowerAI，http://ibm.biz/powerai

[11] Tung Le Duc：IBMCaffe: The Harmony of CPU and GPU in Training Deep Neural Networks，Linux on Power Blog，https://developer.ibm.com/linuxonpower/2017/04/11/ibmcaffe-harmony-cpu-gpu-training-deep-neural-networks(2017).

[12] Tung Le Duc, et al.：Accelerating Multi-GPU Deep Learning by Collecting and Accumulating Gradients on CPUs，IPSJ SIG Technical Report，Vol. 2017-HPC-159，No. 8(2017).

[13] Hillery Hunter：IBM Research achieves record deep learning performance with new software technology，IBM Research Blog，https://www.ibm.com/blogs/research/2017/08/distributed-deep-learning/(2017).

日本アイ・ビー・エム株式会社
東京基礎研究所 副所長
ディスティングイッシュト・エンジニア（技術理事）

小野寺 民也
Tamiya Onodera

1988年日本IBM入社。以来、同社東京基礎研究所にて、プログラミング言語、ミドルウェアおよび分散システム等の研究開発に従事。最近ではとくにAIおよびコグニティブ・システムの基盤ソフトウェアに興味をもつ。米国計算機学会（ACM）ディスティングイッシュト・サイエンティスト、情報処理学会シニア会員、日本ソフトウェア科学会会員。理学博士。
http://ibm.biz/onodera

日本アイ・ビー・エム株式会社
東京基礎研究所
サービス型コンピューティング 部長
シニア・テクニカル・スタッフ・メンバー

河内谷 清久仁
Kiyokuni Kawachiya

1987年日本IBM入社。以来、同社東京基礎研究所にて、オペレーティング・システム、Javaをはじめとするプログラミング言語、システムの性能改善などの研究に従事。最近ではコグニティブ・フレームワークの性能改善をリード。米国計算機学会（ACM）ディスティングイッシュト・エンジニア、情報処理学会理事、日本ソフトウェア科学会会員。博士（政策・メディア）。
http://ibm.biz/kawatiya

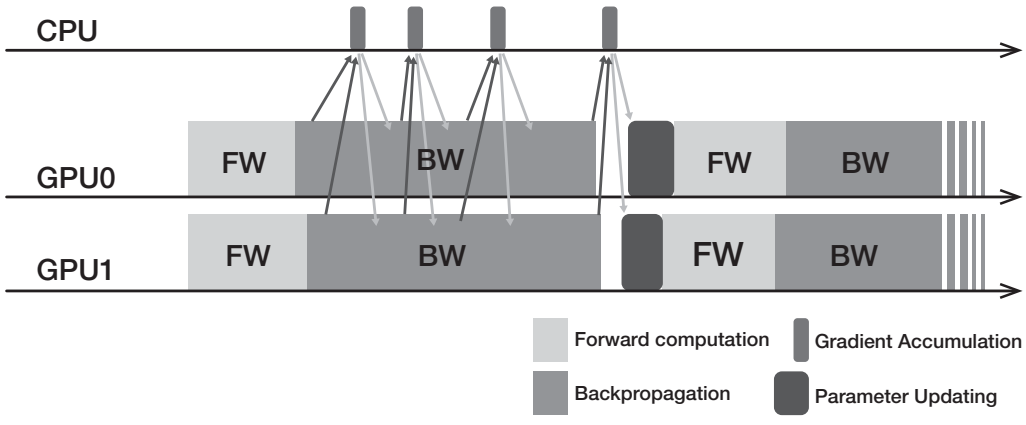

図6. パラメーター集計をCPUで並列に行い高速化