スーパーコンピュータを活用したディープラーニング分散学習の高効率開発基盤

Highly Efficient Infrastructure for R&D by Utilizing Distributed Deep Learning on a Supercomputer

竹中慎治 Shinji Takenaka
柴田俊介 Syunsuke Shibata
田口雄太 Yuta Taguchi
竹田博昭 Hiroaki Takeda
寺田徹 Toru Terada
合田陽一 Youichi Gouda

要　旨

画像や音声など，さまざまなセンサ情報を利用するAI技術の多くにディープラーニングが利用されており，その技術進歩は近年ますます加速している．ディープラーニングに関する研究・開発期間の短縮化は，技術進展に非常に有効であり，並列計算処理が可能なスーパーコンピュータの利用により実現可能である．本稿では，東京工業大学が保有するスーパーコンピュータTSUBAME 3.0を活用した分散学習とその効果について報告する．公開されている学習モデルとデータセットとを使用した実験において，GPU1基では約7日間かかる学習処理が，GPU128基を用いて1時間40分で完了することを確認した．

Abstract

Deep Learning is utilized for many AI technologies using information acquired from various sensors, such as those of images and sound, and its technological progress has been accelerating in recent years. Reducing the period required for research and development related to deep learning can contribute to progress in AI technology. This reduction can be realized by using a supercomputer capable of parallel processing. In this paper, we report on distributed deep learning and its performance using the supercomputer TSUBAME 3.0 at the Tokyo Institute of Technology. In experiments using published learning models and datasets, we demonstrated that a learning process that required seven days using a 1 GPU configuration was completed in one hour and forty minutes by using 128 GPUs..

1. はじめに

人間の脳の構造を模した多層化ニューラルネットワークの学習技術がGeoffrey Hintonらによって考案[1]されて以来，画像認識や音声認識などさまざまな分野でディープラーニングの研究･開発が盛んに行われている. 特に，画像認識分野では，2012年のILSVRC（ImageNet Large Scale Visual Recognition Challenge）において，Hintonらの提案方式が物体認識のエラー率で2位以下と10 %以上の大差をつけて圧勝してからは，精度の良いネットワークモデルが次々と考案され，2015年には152層からなるResidual Networks（ResNet）[2]が首位を獲得している．より深く，より複雑で精度の良いネットワークモデルが短期間で提案されるなか，技術者に対する研究・開発スピードの重要性も従来以上に高くなっている．

一般に, ディープラーニングは機械学習の1種であるため高い性能を実現するためには膨大な学習データ量・学習時間が必要である．クラス分類の場合，実用レベルの精度を確保するためには1カテゴリに対して5000データ以上を用いた学習処理が必要とされているため，ILSVRCで利用されるImageNetデータを1000クラスに分類する場合, 500万データ以上を用いた学習処理が必要となる．多くの場合，学習データ収集には公開データセットや撮影した画像データが用いられ，これを元に変形やノイズ付与によるデータの数増しが行われる．また，最近ではGAN（Generative Adversarial Networks）[3] のような技術を利用し画像データそのものを生成するようなモデルも考案されており，良質な学習データが大量に取得できる環境も整ってきている．

このように膨大な量のデータを学習するには, 1回の試行当たり数日間の学習時間を要する場合も多く，開発スピード改善のボトルネックとなっている．また，複雑なネットワークになるほど調整が必要なパラメータ量が増加するため，精度評価のための試行回数も増加する．従って，一度の学習に要する時間の短縮化は，ディープラーニングを利用した研究・開発を行ううえで非常に重要な要素となる．

そこで筆者らは，ディープラーニング開発に関わるモデル学習時間やモデルパラメータの試行・評価を効率的に行うことを目的として，複数の計算ノードを用いた並列計算が可能なスーパーコンピュータ（以降, スパコン）に，それを活用した分散計算が可能なディープラーニングの開発環境を構築した．

本稿では，東京工業大学のスパコンTSUBAME3.0（以降, TSUBAME）の計算ノード32台, NVIDIA Tesla(注1) P100

（注1）NVIDIAおよびTeslaは，米国およびその他の国のNVIDIA Corp.の商標および登録商標.

GPU計128基を用いて，ImageNetの約120万データをResNet50モデルで学習した結果を記載する．

以降では，実装した複数のディープラーニングフレームワークの開発環境構築，および，分散計算による学習速度改善効果について報告する．

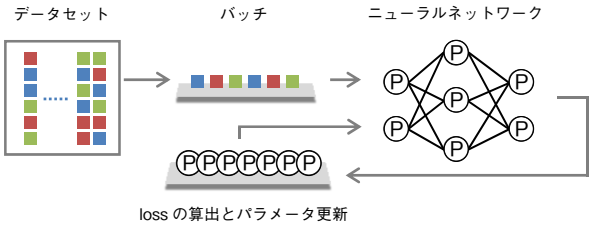
2. 分散学習環境

本章ではまず，分散学習の基本的な方法である，Data ParallelとModel Parallel[4]について述べる．次に，分散学習が実現可能なディープラーニングフレームワークについて述べ，最後に，分散学習環境を構築する土台であるTSUBAMEのスペック詳細，および，速度改善に関わる調整要素について述べる．

2.1 分散学習の方法

第1図にディープラーニングの一般的な教師あり学習の流れを示す．まず，ユーザーが形成した多層ニューラルネットワークに対して，画像と画像に映る物体名（正解ラベル）ペアを学習データとして入力する．ニューラルネットワークからの出力に対し，正解ラベルとの誤差（loss）をあらかじめ設計した誤差関数を用いて算出する．このlossが減少するように，誤差逆伝播（でんぱ）法を用いてパラメータを更新する．以上の流れを，準備した学習データをバッチと呼ばれる単位に分割し，lossの値が収束するまで，もしくは設定した上限回数に達するまで繰り返し学習する．（以降，1バッチに含まれる学習データ数をバッチサイズと呼ぶ．）上限回数は，準備したデータセットに含まれる学習用画像総数を1 epochとし，epoch単位で設定することが多い．

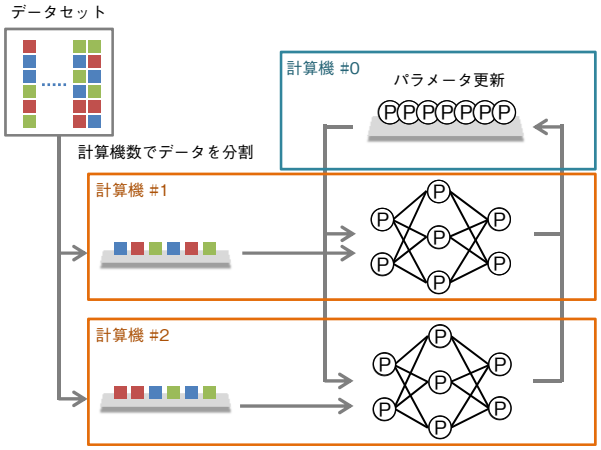
分散学習を行う際にも前述のバッチ単位で処理する．一般的に用いられる分散学習の方法は，大別すると，Data ParallelとModel Parallelの2つが存在する．



第1図　ディープラーニングの学習の流れ
Fig. 1　Training flow for deep learning

〔1〕Data Parallel

Data Parallelは，まず，学習データセットを計算機の台数で分割する．各計算機では，与えられたデータセットを使い，並列に学習を行う（**第2図**）．このとき，各計算機は，同じネットワークモデル，同じパラメータをもっている．並列で実行することから，計算機の数に比例して各計算機で処理するデータ量が減り，処理負荷も減少することになる．また，パラメータの更新方法として，同期型と非同期型が存在する．同期型では，全計算機のバッチ処理が終わったタイミングで，まとめ役となる計算機が個々の計算機のパラメータを収集する．そして，パラメータ更新を行い，更新したパラメータ情報を全計算機に送信する．対して非同期型では，計算機ごとのバッチ処理の結果が得られたタイミングで，順次パラメータを更新する．非同期型は，すべての計算機の処理終了を待つことなく次の計算を行うので高速に処理が進行する．しかし，学習が進むにつれ各計算機が保持するネットワークモデルのパラメータに差分が生じてしまうため，同期型と比較して学習性能が劣る[5]．

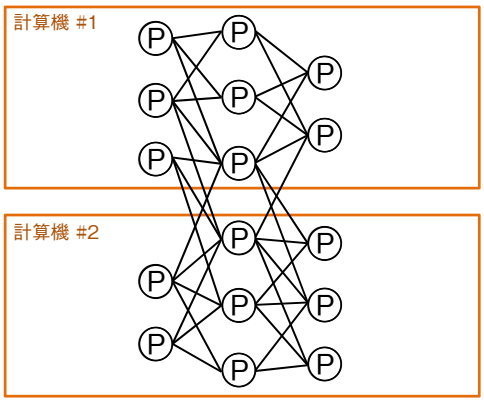


第2図　データ分散方式
Fig. 2　Data Parallel

〔2〕Model Parallel

Model Parallelは，複数の計算機で1つのネットワークモデルの処理を分割し，学習する方法である（**第3図**）．個々の計算機で担当する処理の完了を待って，次の計算機の処理に進むため，Data Parallelに比べると並列処理による処理速度の改善効果は低い．しかし，複数の計算機で1つのネットワークモデルを分割するため，メモリー不足により単体の計算機のみでは実装不可能な大きなネットワークモデルも学習することが可能となる．

本稿では，処理速度の改善効果や学習性能が高い，同期型のData Parallelによる分散学習を採用する．



第3図　モデル分散方式
Fig. 3　Model parallel

2.2 分散学習対応のディープラーニングフレームワーク

分散学習が行えるディープラーニングフレームワークとして，Googleが開発しているTensorFlow(注2)やFacebookが主として開発しているCaffe2，Preferred Networksが開発しているChainerMN(注3)[4]などさまざまなものがある．いずれのフレームワークも複数の計算機において，各計算機が保有する複数のGraphics Processing Unit（以降，GPU）を利用したData Parallel分散学習を可能とする．本稿では，多くの人が利用している上記3つのフレームワークを扱う．なお，各フレームワークにおける分散の方法には，次のような違いがある．

・TensorFlow

TensorFlowではパラメータ管理を担当するparameter server（以降，ps）と，学習を担当する複数のworkerで構成される．処理の流れは**第2図**とほぼ同様で，事前にデータセットをworkerの数で分割し，各workerは分割されたデータセットを利用し学習する．1回のバッチ処理が終了すると，各workerのパラメータ情報がpsに送信される．psは集まったパラメータを平均し更新を行う．その結果は全workerに送信され，次のバッチ処理が実施される[2]．なお，複数のpsで処理する場合は，全パラメータを各psで分担して管理する．

・Caffe2

Caffe2では，パラメータ管理を行うpsのような存在はなく，1回のバッチ処理が終了した時点でworker同士が更新済みパラメータを共有する．worker間（計算機間）の同期は，Halving and Doubling Algorithmと呼ばれるデータ転送アルゴリズムを用いることで，全パラメータの同期を効率的に実現している[6]．

・ChainerMN

ChainerMNもCaffe2と同様，psは存在せず各workerの処理が終了した時点で，worker同士でパラメータを共有する．通信管理にはMPI（Message Passing Interface）通信ライブラリを利用しており，集団通信命令の1つであるAll-Reduceによって同期を行っている[7]．

2.3 スパコン活用分散学習技術

第1表に，本稿で用いたスパコンTSUBAMEのシステム仕様を示す[8]．CPU2基とGPU4基を搭載した540台の計算ノードを高速ネットワークで接続した構成で，理論演算性能は12.15 PFLOPS（倍精度浮動小数点演算），47.2 PFLOPS（半精度浮動小数点演算）となる．スパコンの並列計算処理における計算ノード間の制御はMPI規格に準拠した命令の実行により行われる．計算ノード内のGPU間通信にはNVIDIA社のNVLink(注4)が利用可能であり，計算ノード間の通信にはOmni-Pathと呼ばれる100 Gbit/s（理論値）の光ファイバーが使用される．上記の構成により計算ノード間，計算ノード内ともに高速通信を実現している．

第1表　スパコンTSUBAME 3.0 構成
Table 1　Specifications of TSUBAME 3.0

CPU	Intel Xeon(注5) E5-2680 V4 Processor（Broadwell-EP, 14 コア, 2.4 GHz）× 2 Scoket
GPU	NVIDIA Tesla P100 for NVLink-Optimized Servers × 4
RAM	256GiB
ローカルストレージ	NVMe 2TB
ネットワーク	Intel Omni-Path 100 Gbit/s × 4
計算ノード	SGI ICE XA 540台
OS	SUSE Linux(注6) Enterprise Server 12 SP2
MPI	Intel MPI 17.3.196 SGI MPT 2.16 OpenMPI 2.1.1

ディープラーニングの分散学習ではGPUごとにバッチ単位で学習データを割り当てる．すなわち，一度の並列計算における総バッチサイズは，並列ノード数×1ノード当たりのGPU数（4基）×バッチサイズという計算から求まる．バッチサイズを大きく設定すれば少ない計算回数で多くの学習データが学習でき，学習に必要な計算時間が短縮される．しかし，そのような設定では学習時のネ

（注2）Google Inc.の登録商標または商標．
（注3）（株）Preferred Networksの日本およびその他の国における商標または登録商標．
（注4）NVIDIA Corp.の登録商標または商標．
（注5）IntelおよびXeonは，Intel Corp.の登録商標または商標．
（注6）Linus Torvalds氏の米国およびその他の国における登録商標または商標．

ットワーク更新の際に計算する勾配の分散が小さくなることにより，局所解に陥って汎化性能の悪いモデルができてしまう[9].

以上の懸念から，分散学習による計算時間の短縮度合いとともに，テストデータによるモデルの精度評価も実施することで分散学習の有用性を示す.

3. ディープラーニングの分散学習評価

筆者らは，分散学習が可能な代表的な3つのディープラーニングのフレームワーク，TensorFlow，Caffe2，ChainerMNにおける分散学習環境をTSUBAME上で構築した.

分散学習に利用するGPU数は1基から128基までの範囲内で変化させ，GPU数の増加に対する処理速度の改善具合を，90 epoch分の学習が完了するまでに要する時間で評価する．また，テストデータにおける正解率も評価することで精度を維持した学習ができていることを確認する．なお，学習モデルはResNet50を採用し，学習データはILSVRCで利用されているImageNetを用いた. 本実験で設定したパラメータは，全フレームワークで共通にしており，GPU数に関係なくバッチサイズは32で固定し，勾配最適化手法にMomentum-SGDを利用，学習率は初期値0.1から30 epochごとに0.1倍する設定とした.

以降，実験結果と考察を述べる.

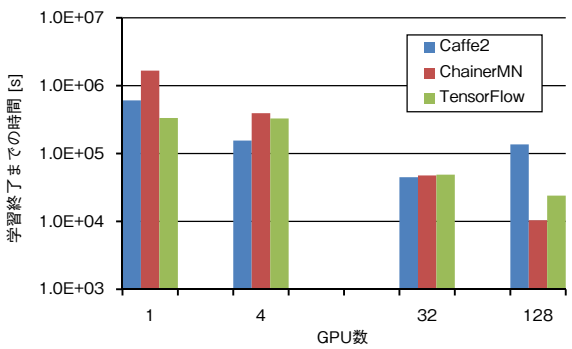
3.1 各フレームワークの分散学習評価

利用するGPU数を1, 4, 32, 128と変化させた際の学習完了までに要した時間を示す（**第4図**）.

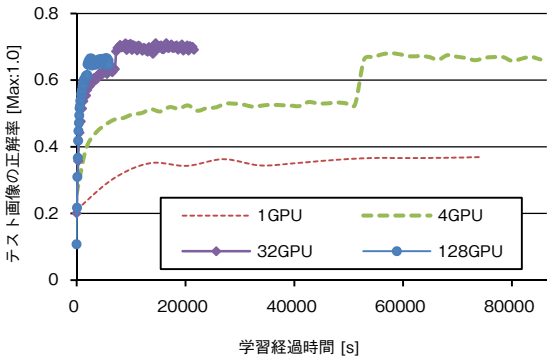
まず，TensorFlowでは，分散処理を実施していない場合でも分散時と同様の仕組みで学習処理が実行されることが知られており，他のフレームワークと比較すると，GPU数増加に対する速度改善率が低い．これは，psやworker間の通信に使用しているgRPCフレームワークのオーバヘッドが一因となっている[10][11]．ChainerMNはGPU数が少ないときの学習終了時間が他と比べて遅いが，GPU数増加に比例した速度改善率は一番良い結果となった. GPU数が少ない場合の速度については，ChainerMNがPythonで実装されていることから，最適化効率の面で不利なためと考えられる[9]．また，GPU数が多い場合に速度改善率が高いのは，学習データの読み込み方法が寄与していると考えられる．ChainerMNでは，ファイルサーバ上で小容量の学習データを大量に確保することで，複数ノードからの読み込みに対して競合が起こりにくい仕組みとなっている．一方，Caffe2では学習データを1つのデータベースとして保持しているため，複数ノードから一度に読み込み処理が行われると，128 GPUの結果のように速度劣化が発生してしまう.

なお，現時点ではTSUBAMEの制約により，連続稼動時間が24時間を超過することができず，GPU数が1基と4基の結果は24時間経過時点の測定値から予測して求めたものとなっている.

第5図は学習経過時間とテスト用データにおける正解率との関係を表しており，正解率は1.0に近いほど精度が良いことを示す．GPU数が多いほど早いタイミングで良い値が観測されている．一般に，一度の学習に利用するバッチサイズが大きくなり過ぎると算出される勾配が鈍化するため，学習時の精度改善に悪影響を与える．第5図の結果も同様に，バッチサイズを固定値で設定していることから，GPU数に比例して総バッチサイズが大きくなり，精度向上率の鈍化タイミングも早くなっていることがわかる．なお，本実験では，32GPUと比べて128GPU学習終了時の正解率が0.05程度劣っているが，双方の総バッチサイズを同値に調整することで，正解率が0.7を超え，32GPUと同程度まで改善することを確認している.



第4図　ImageNetデータの学習時間
Fig. 4　Learning time for ImageNet classification



第5図　テスト画像における正解率（Caffe2）
Fig. 5　Validation accuracy

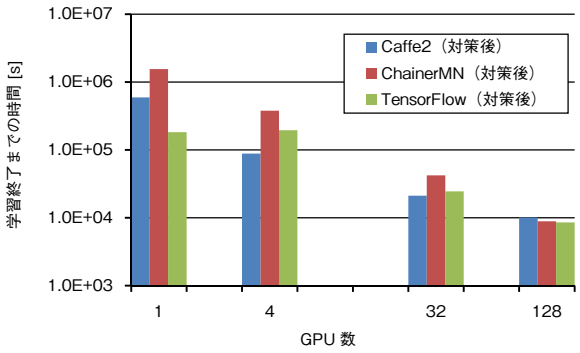
3.2 ノード間通信に関する高速化評価

本節では，更なる学習速度の向上のための，GPU間，ノード間の通信速度改善対策について紹介する．

〔1〕学習データ転送オーバヘッドの削減

TSUBAME上で複数ノードを用いた分散学習を行う場合，学習データはデータサーバに配置し，バッチごとの学習を行う際に各計算ノードがデータ取得して学習を行っている．このデータ転送にかかる時間を低減するため，学習データをあらかじめデータサーバから各ノードに配置するよう実装した．

第6図にデータ転送時間を削減した際の学習完了までに要した時間を示す．Caffe2の128GPUの結果では，**第4図**で示した改善前の結果と比べ，大幅に学習時間が短縮されている．前述したように1つのデータベースからの読み込み処理に関するオーバヘッドが解消された効果と考えられる．ChainerMNでは，学習データを小容量で大量に分割保持しているため，元々データ読み込みに関するオーバヘッドが小さく，対策前後で大きな変化が見られなかったと考える．



第6図　ImageNetデータの学習時間（対策後）
Fig. 6　Learning time for ImageNet classification

〔2〕ノード間通信の高速化

TSUBAMEでは，計算ノード内のGPUはNVLinkで接続されており，PCIバスやCPUのメモリーを介さない高速なGPU間通信が可能である．しかし，NVLinkに対応したマルチGPU向け集合通信ライブラリNVIDIA NCCL2が，現状ではTSUBAMEのノード間通信に利用されているOmni-Pathに対応していない．そこで，各ノード内のGPU間通信にのみNCCL2を利用し，ノード間通信はOmni-Pathの通信APIであるverbsを利用する実装としている．

Caffe2において上述のとおりverbs APIによるノード間通信と，NCCL2を利用したノード内GPU間通信が動作するように実装したうえで，さらに，学習データの転送オーバヘッド対策を施した環境で学習時間を評価した．比較対象として，**第4図**（(a) Default），**第6図**（(b) データ転送対応）で示した学習時間を記す（**第2表**）．なお，(a) と (b) におけるノード間通信には，スパコンのサーバ間通信で広く利用されているInfiniBand規格のうえでTCP/IP通信を実現するIPoIB（IP over InfiniBand）プロトコルを使用している．**第2表**に示すとおり，(a) の1GPUでは604 987秒間も要していた学習時間を (c) の128GPUでは5 989秒まで短縮でき，約101倍に処理速度が改善されたことを確認した．

第2表　学習時間の改善結果（Caffe2）
Table 2　Improvement of learning speed

	1GPU	4GPU	32GPU	128GPU
(a) Default [s]	604 987	154 931	44 862	135 905
(b) データ転送対応 [s]	593 992	88 185	21 111	10 089
(c) (b) +ノード間通信対応 [s]	608 629	152 852	21 774	5 989

4. まとめ

筆者らは，ディープラーニングの主要なフレームワークTensorFlow，Caffe2，ChainerMNを利用した分散学習環境をスパコンTSUBAME上に構築し，並列分散処理に適した構成の検討とパラメータ調整を実施した．本環境を利用することで，1GPUの構成では約7日間かかっていた学習処理を128GPUの利用で1時間40分まで短縮できることを確認した．これにより，ディープラーニングのモデル検討・評価期間が短縮でき，今後の研究開発を進めるうえでの一助となることを期待する．

本稿で評価した開発環境については，更なる分散効率の改善検討を継続するとともに，大規模計算リソースを活用した高効率学習の実事例を増やすことで，将来に向けたAI技術の進展に貢献していきたい．

参 考 文 献

[1] Hinton, G. E. et al., “A fast learning algorithm for deep belief nets,” Neural Computation, vol. 18, no. 7, pp.1527-1554, 2006.
[2] Kaiming He et al., “Deep Residual Learning for Image Recognition,” IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.770-778, 2016.
[3] Ian J. Goodfellow, et al., “Generative Adversarial Networks”, arXiv:1406.2661, Jun. 2014.
[4] T. Akiba et al., “ChainerMN: scalable distributed deep learning framework,” arXiv:1710.11351, Oct. 2017.
[5] J. Chen et al., “Revisiting Distributed Synchronous SGD”, ICLR Workshop Track, Puerto Rico, May 2016.
[6] Priya Goyal et al., “Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour”, arXiv:1706.02677, June 2017.
[7] Takuya Akiba, “ChainerMN Documentation Release 1.0.0b2,”

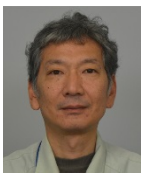
Read the Docs, https://media.readthedocs.org/pdf/chainermn/v1.0.0b2_a/chainermn.pdf, 参照 Apr. 20, 2018.
[8] 東京工業大学, “システム構成,” TSUBAME計算サービス TSUBAME3.0, http://www.t3.gsic.titech.ac.jp/architecture, 参照 Apr. 20, 2018.
[9] 秋葉拓哉, “ChainerMNによる分散深層学習の性能について,” Preferred Research, https://research.preferred.jp/2017/02/chainermn-benchmark-results/, 参照 Apr. 20, 2018.
[10] Shaohuai Shi et al., “Benchmarking State-of-the-Art Deep Learning Software Tools,” arXiv:1608.07249, Feb. 2017.
[11] Rengan Xu et al., “Deep Learning Performance with P100 GPUs,” Dell Inc., http://en.community.dell.com/techcenter/high-performance-computing/b/general_hpc/archive/2016/11/11/deep-learning-performance-with-p100-gpus, 参照 Apr. 20, 2018.
[12] K. Simonyan et al., “Very Deep Convolutional Networks for Large-Scale Image Recognition,” arXiv:1409.1556, Apr. 2014.

執筆者紹介


竹中　慎治　Shinji Takenaka
（株）パナソニック システムネットワークス開発研究所
Panasonic System Networks R&D Lab. Co., Ltd.


柴田　俊介　Syunsuke Shibata
（株）パナソニック システムネットワークス開発研究所
Panasonic System Networks R&D Lab. Co., Ltd.


田口　雄太　Yuta Taguchi
（株）パナソニック システムネットワークス開発研究所
Panasonic System Networks R&D Lab. Co., Ltd.


竹田　博昭　Hiroaki Takeda
（株）パナソニック システムネットワークス開発研究所
Panasonic System Networks R&D Lab. Co., Ltd.


寺田　徹　Toru Terada
（株）パナソニック システムネットワークス開発研究所
Panasonic System Networks R&D Lab. Co., Ltd.


合田　陽一　Youichi Gouda
（株）パナソニック システムネットワークス開発研究所
Panasonic System Networks R&D Lab. Co., Ltd.