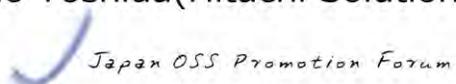

Survey of OSS in Big Data Platform

2017/11/17

Japan OSS Promotion Forum

Big Data Committee

Chair Yukio Yoshida(Hitachi Solutions, Ltd)



Japan OSS Promotion Forum

0-1. Self Introduction

【Career】

- ✓ At First, engaged in software development of financial terminal
- ✓ From 2000, responsible for the business development of the Linux / OSS.

【 Current business 】

- ✓ Support for business construction utilizing OSS
 - Excavation and evaluation validation of new technology
 - Start-up support of business solutions
 - Focus Area:
 - Cloud Infrastructure(OpenStack, Spark)、
 - Big Data Infrastructure (Hadoop, NoSQL)、
 - Enterprise(PostgreSQL, OpenCOBOL)



【 Activities outside of the company 】

- ✓ Japan OSS Promotion Forum Vice Chairman
 - ✓ Open License Laboratory Director
 - ✓ OSS consortium Vice Chairman
- etc

経歴

- 入社当時は、OSSに関係なく、金融端末のソフトウェア開発をしていました。
- 2000年頃からLinuxやOSSのビジネス企画を担当し、現在に至っています。

現在の業務

- 現在は、OSSを活用したビジネスの構築の支援をしています。
特に、新しい技術やOSSの発掘や技術検証をしています。
また、ビジネスソリューションの立ち上げの支援もしています。
現在、フォーカスしている領域は、
 - OpenStackに代表されるクラウドを構成する基盤のOSS
 - Hadoopに代表されるビッグデータを構成する基盤のOSS
 - 企業システムで活用されるデータベースやCOBOLの環境です。

社外での活動

社内だけではなく、社外でもさまざまな活動をしています。

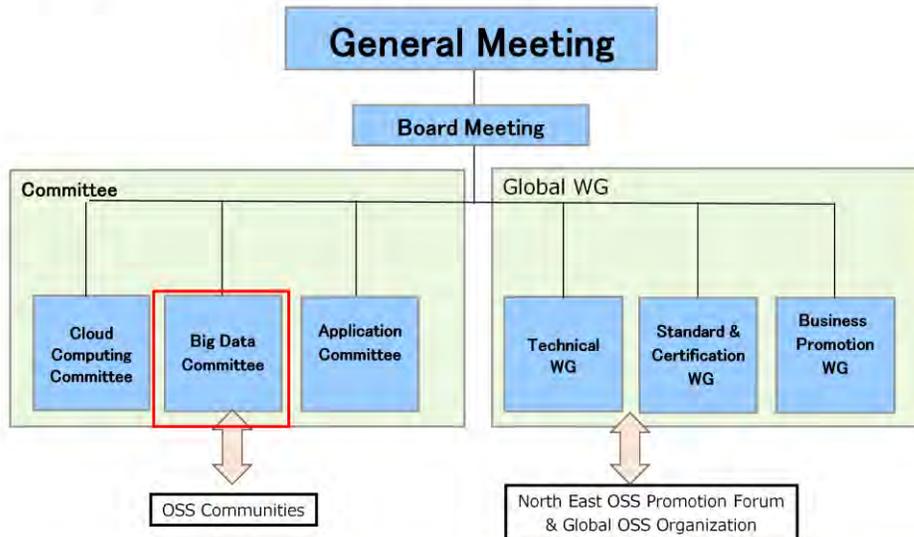
Contents

Introduction JOPF BigData Committee
Result of Data Analysis
Trend of Deep Learning

1-1. Japan OSS Promotion Forum(2016)

■ Our current activities

- ✓ 3 committees are organized to study SMAC(Social, Mobile, Analytics, Cloud) and IoT.
- ✓ 3 WG are organized to study global activities.



Copyright 2016 Japan OSS Promotion Forum

3

日本OSS推進フォーラムは、3つの部会とWGで構成されています。
部会は、クラウド、ビッグデータ、アプリケーションです。
特に注目しているのは、**Social**、モバイル、分析、クラウドとIoTです。
また、WGは、北東アジアOSS推進フォーラムと連携しています。

1-2. Activity of Big Data Committee

◆ Trend survey of the OSS

Lists the OSS used for big data system, what kind of situation, respectively, several aspects were investigated based on the analysis.

◆ Trend of Deep Learning

We conducted a survey on the framework for implementing the Deep Learning, and also to verify the performance by partially real machine.

ビッグデータ部会では、下記の活動を行っています。

ビッグデータ関連OSS動向調査

ビッグデータシステムに使われるOSSを洗い出し、それぞれがどのような状況であるか、いくつかの観点をもとに調査し、「今使えるビッグデータOSSは何か？」を分析しました。

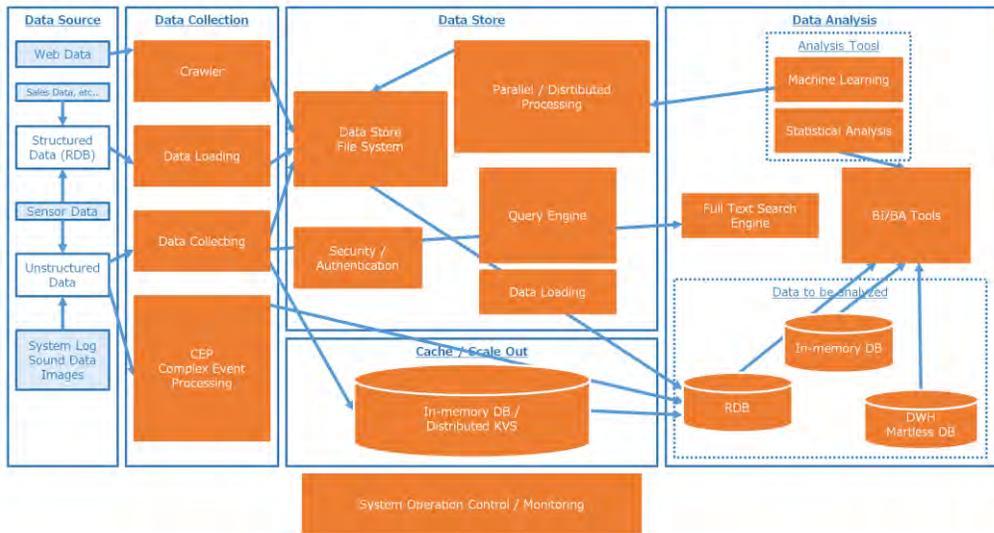
ディープラーニングの動向

今話題のディープラーニングを実現するためのフレームワークに関する調査を行い、また部分的に実機にて性能を検証しました。

Result of Data Analysis

Introduction: Big Data Platform

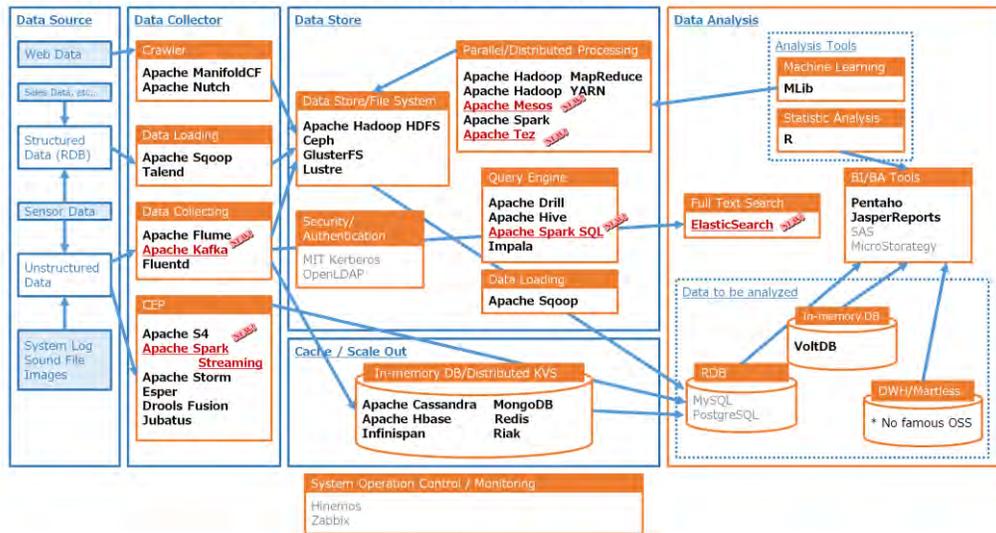
- Most of the Big Data Platforms are composed of 3 functions :
Data Collector, Data Store and Data Analysis



ビッグデータを処理する基盤は、主に、様々なデータを「収集・検知」する機能、収集・検知したデータを「蓄積・貯蔵」する機能、蓄積・貯蔵したデータを「分析」する機能から構成されています。

Introduction: OSS in Big Data Platform

- There are some Big Data Platform using only OSS
- Quality, Maturity and Functions of software are different, so **we need to decide** which software we should use



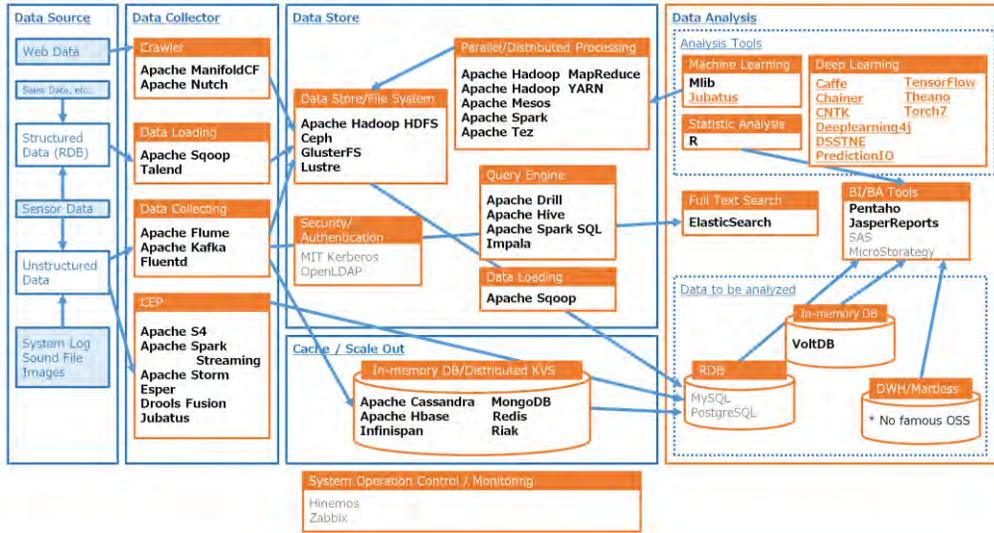
Copyright 2016 Japan OSS Promotion Forum

7

ビッグデータ基盤をオープンソースをベースとして構成する事は十分に可能です。
ただし、個々のソフトウェアにおいて品質や成熟度が異なる為、**見極め**が必要

Introduction: OSS in Big Data Platform

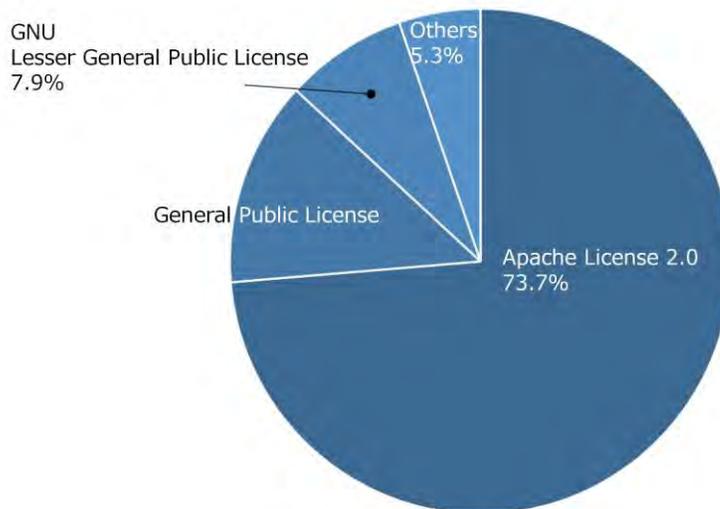
- There are some Big Data Platform using only OSS
- Quality, Maturity and Functions of software are different, so **we need to decide** which software we should use



これは、今年度情報を更新した図です。

Licenses

- Over 70% of software uses Apache License 2.0
- One reason is that there are a large number of software under the Apache Software Foundation

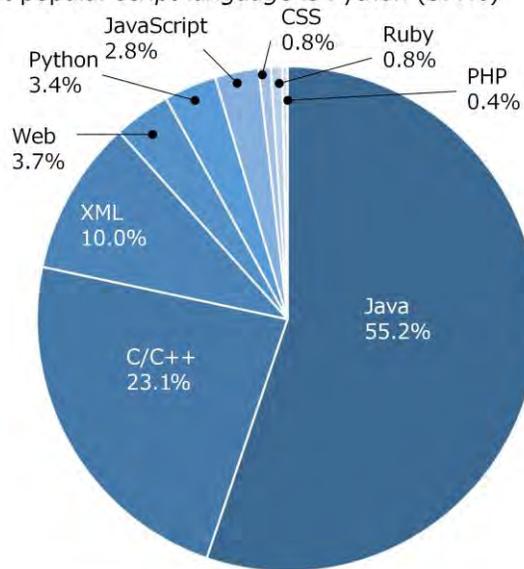


ライセンス

OSSの70%以上がApache2.0のライセンスを採用しています。
その理由は、多くのソフトウェアがApacheソフトウェア財団の下で、
開発しているからです。

Programming Language

- Java is the most popular language used in OSS (55.2%)
 - Second is C/C++ (23.1%)
 - The most popular *script language* is Python (3.4%)



Copyright 2016 Japan OSS Promotion Forum

10

次に開発言語です。

- 約半数のOSSがJavaで開発されています。
- 2番目は、C及びC++言語です。
- 最も使用されているスクリプト言語は、Pythonです。
最近では、OpenStackもPythonで記述されるなど、
全世界的に非常に使用されています。

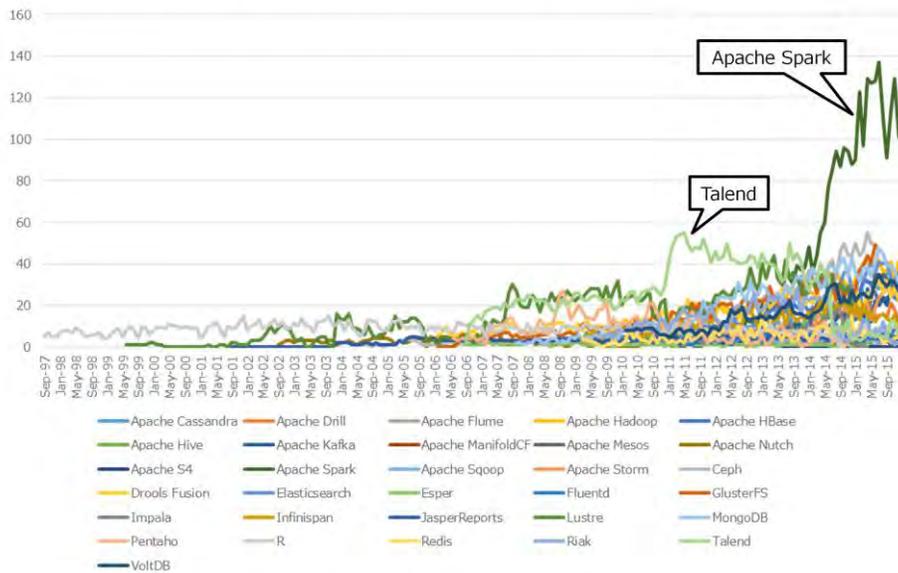
Developers' Activity

How active are developers?

開発の活性度について、説明します。

Number of committers (monthly)

- Apache Spark is rapidly increasing from 2014



Copyright 2016 Japan OSS Promotion Forum

12

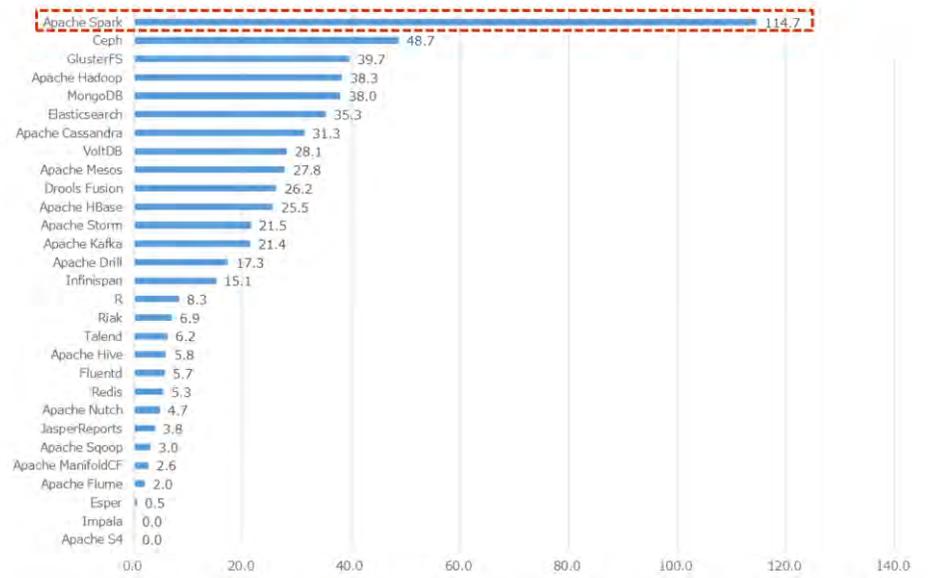
まず、コミッター数の推移です。

ソースコードを更新できるコミッターの増加を図ることで、開発の活性度を測ることができます。

ここでは、**Spark**が2014年ころから急速に成長していることがわかります。

Average number of committers in month (2015) Japan OSS Promotion Forum

- Over 100 committers make some commits to Apache Spark
- 2013 : 27.8 → 2014 : 72.1 → 2015 : 114.7



Copyright 2016 Japan OSS Promotion Forum

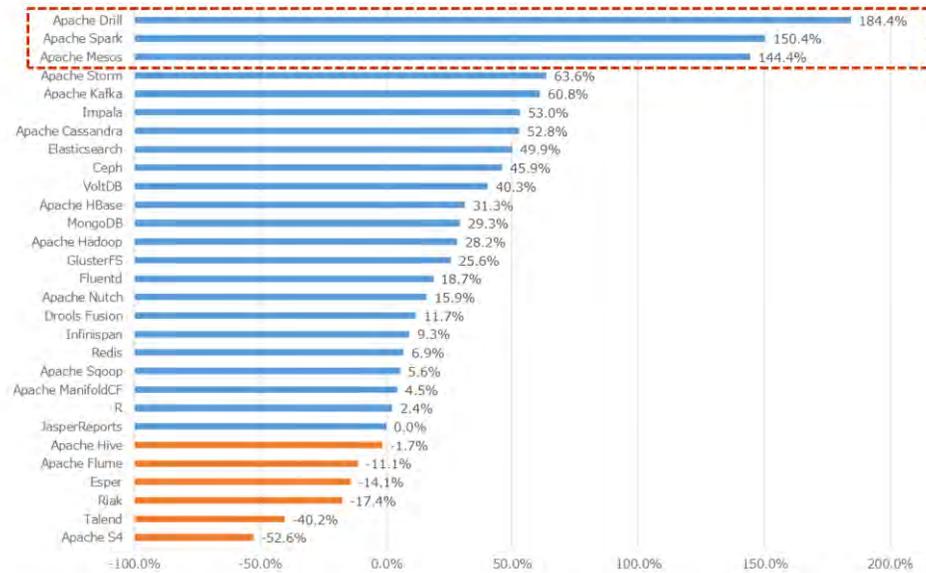
13

次に月当たりの平均のコミッタ数です。

Apache Sparkのコミッター数は、2013年から
飛躍的に増加しています。

CAGR of committers (2011-2015)

- Apache Drill is the fastest-growing
- Apache Spark and Apache Mesos are also growing



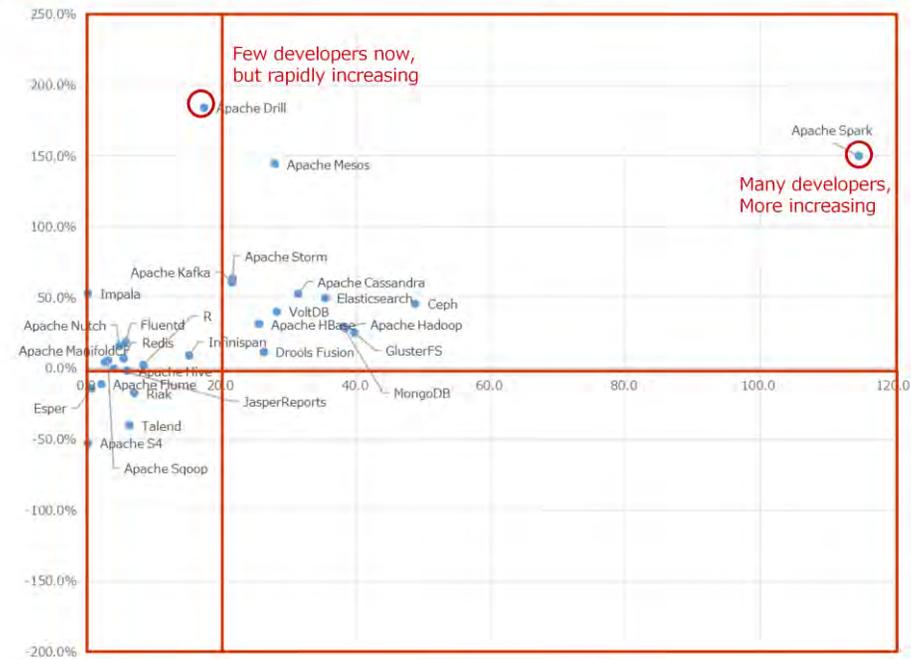
Copyright 2016 Japan OSS Promotion Forum

14

2011年から2015年までのコミッタ数の平均がこのグラフになります。
ここでは、**Spark**だけではなく、**Mesos**も増加していることが
わかります。

Average committers (x-axis) and CAGR (y-axis)

Japan OSS Promotion Forum



Copyright 2016 Japan OSS Promotion Forum

15

月平均コミッタ数を年平均成長率で相関関係をグラフにしました。

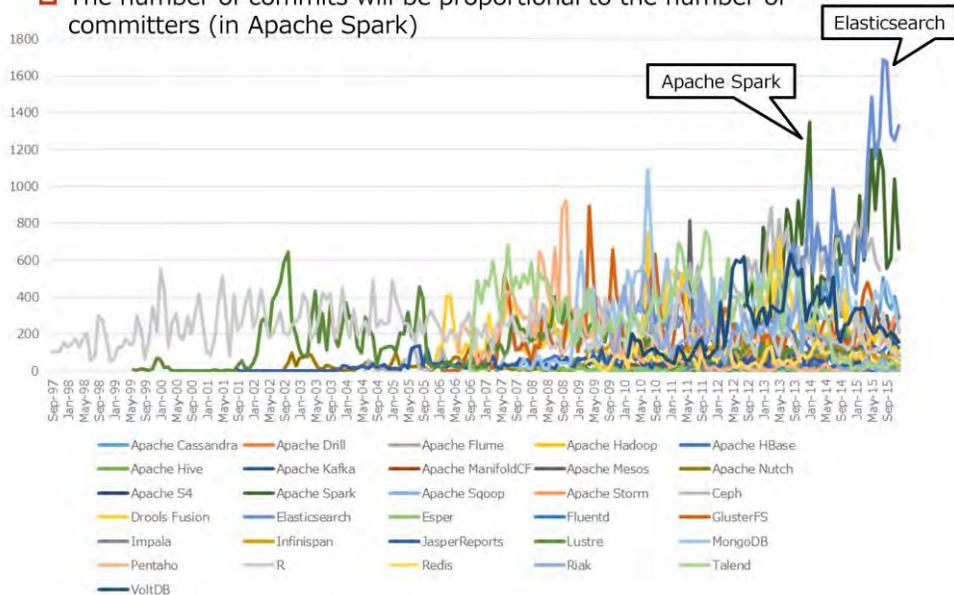
縦軸が成長率、横軸がコミッタ数になります。

左上のDrillは、開発者の数はまだ少ないですが、急増中ということがわかります。

また、左上のSparkは、開発者も多く、また増加中であることがわかります。

Number of Commits

- Recently, Elasticsearch gets the greatest number of commits
- The number of commits will be proportional to the number of committers (in Apache Spark)



Copyright 2016 Japan OSS Promotion Forum

16

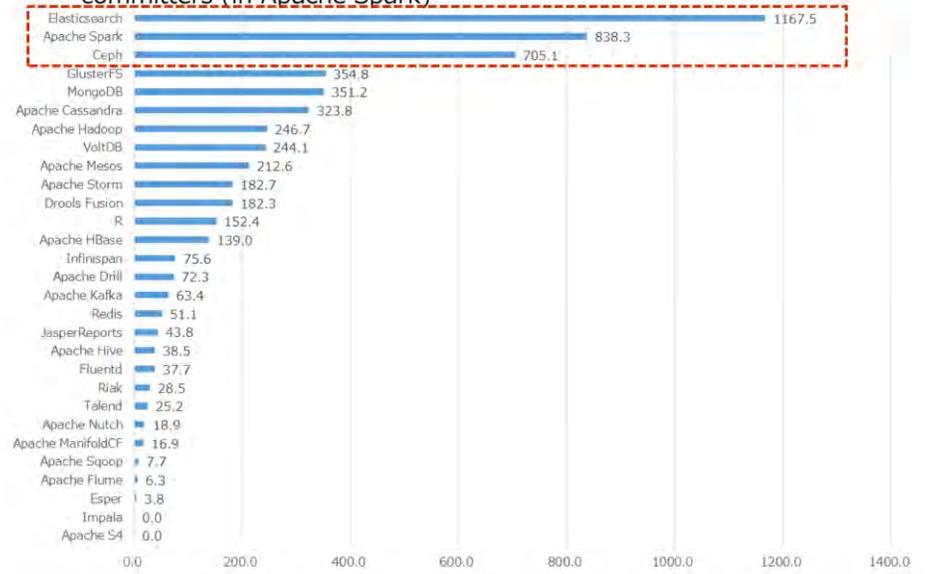
次にコミット数の推移です。

2013年ころは、**Spark**が活発でしたが、
現在、最も活発なのは、**ElasticSearch**という
検索エンジンです。

ほぼ、検索エンジンではデファクトの地位を占めていると
いってよいと思います。

Average number of commits in month (2015)

- Elasticsearch got the largest number of commits
 - The number of commits will be proportional to the number of committers (in Apache Spark)

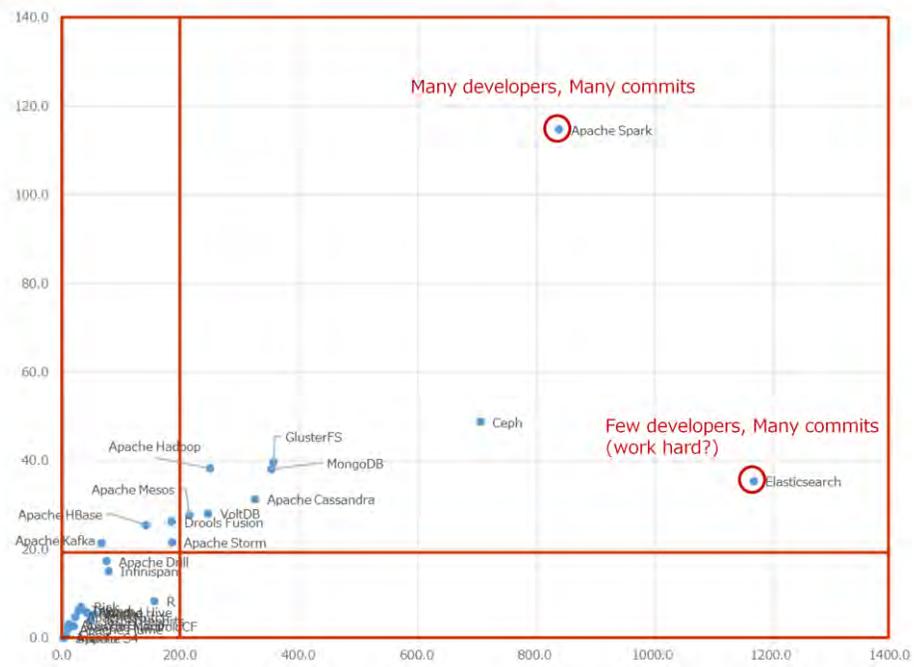


Copyright 2016 Japan OSS Promotion Forum

17

コミット数の月平均のグラフです。
ここでも、ElasticSearchが一番です。
もちろん、Sparkも多くなっています。

Commits (x-axis) and Committers (y-axis)



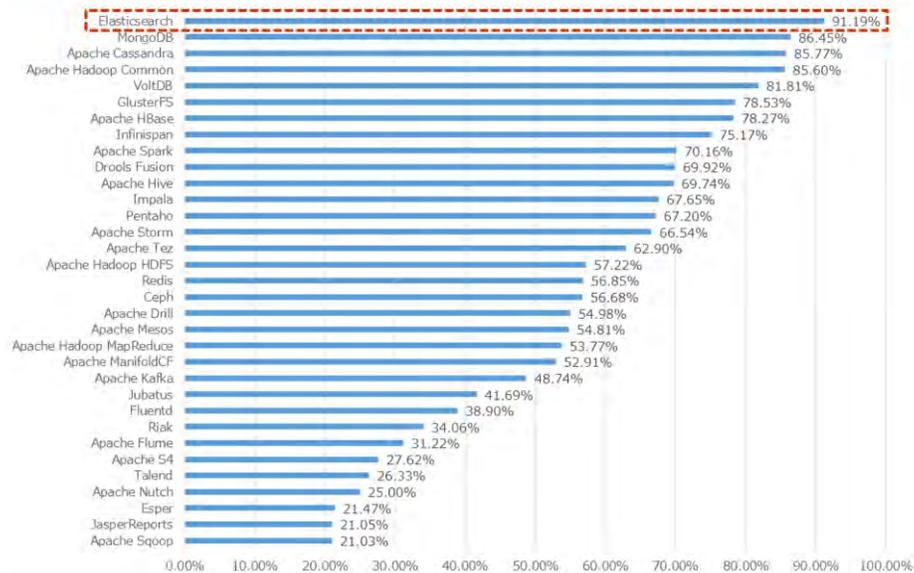
Copyright 2016 Japan OSS Promotion Forum

18

これは、コミッタ数をコミット数の相関関係を表したグラフになります。
左上のSparkは、開発者の数も多く、開発も活発であることが良く分かります。
一方、右下のElasticSearchの場合は、開発者の数はそれほど多くはありませんが、
開発が非常に活発であることがわかります。

Ratio of active days

- Active days : days with commits in Git
 - Elasticsearch has the most active days in these software



Copyright 2016 Japan OSS Promotion Forum

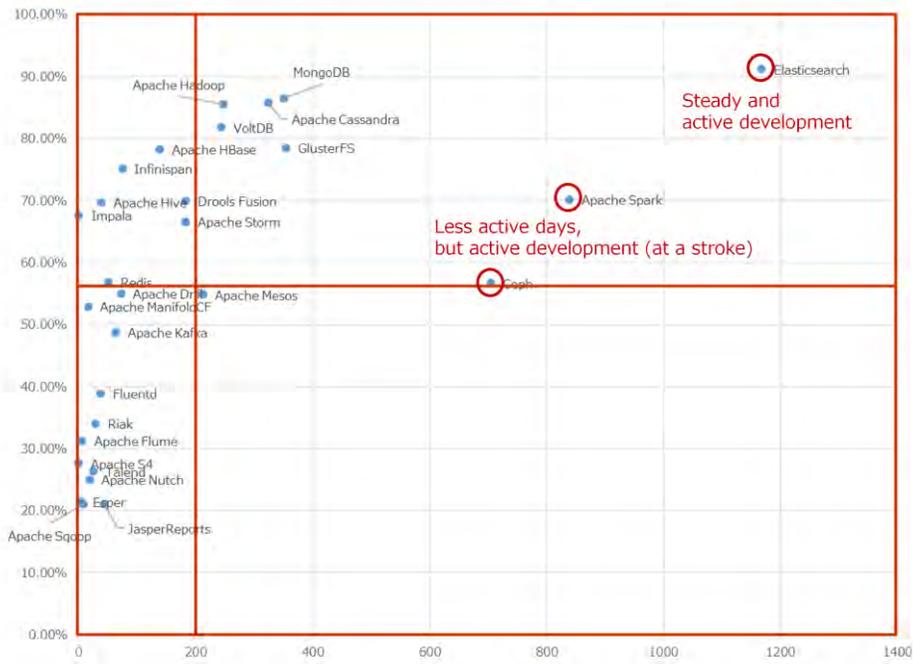
19

次は、稼働率のグラフになります。

一日のうち、どれくらいの時間を開発に使用しているかがわかるグラフです。

これを見ると、ElasticSearchが一番活発であることがわかります。

Commits (x-axis) ratio of active days (y-axis)



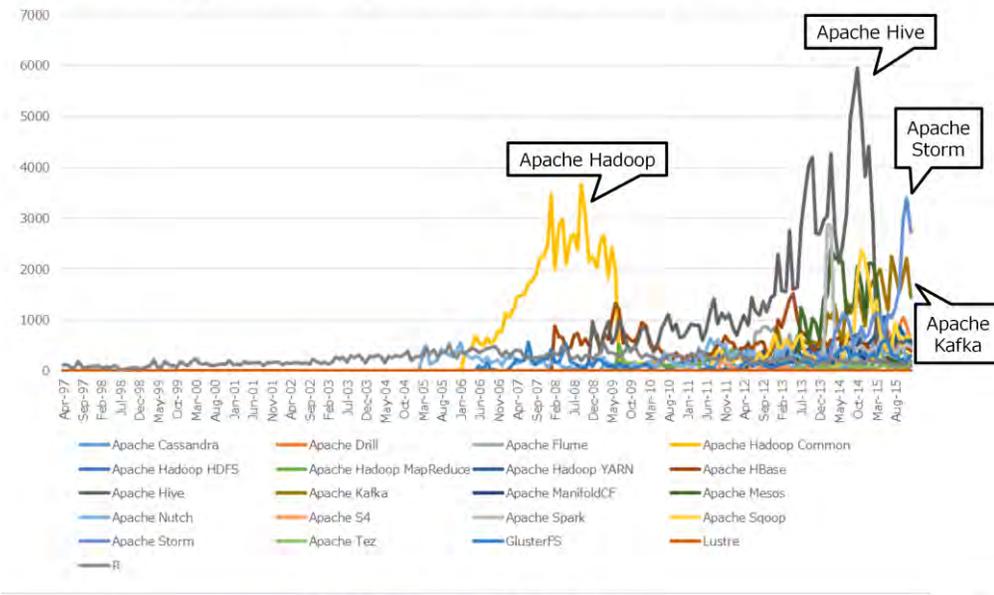
Copyright 2016 Japan OSS Promotion Forum

20

月平均のコミット数を横軸に、稼働率を縦軸にとった相関関係のグラフです。これを見ると、右上にあるElasticSearchは、稼働率も高く、開発も活発にするタイプで、いわゆるコツコツとやるタイプであることがわかります。また、真ん中上にあるSparkは、稼働率があまり高くありませんが、開発が活発なので、どちらかというと開発を一気にやるタイプであるといえると思います。

Number of mails in Mailing list for developers

- Discussion about Apache Hadoop was active in 2006 to 2009
- Recently Apache Hive, Apache Storm and Apache Kafka are also active



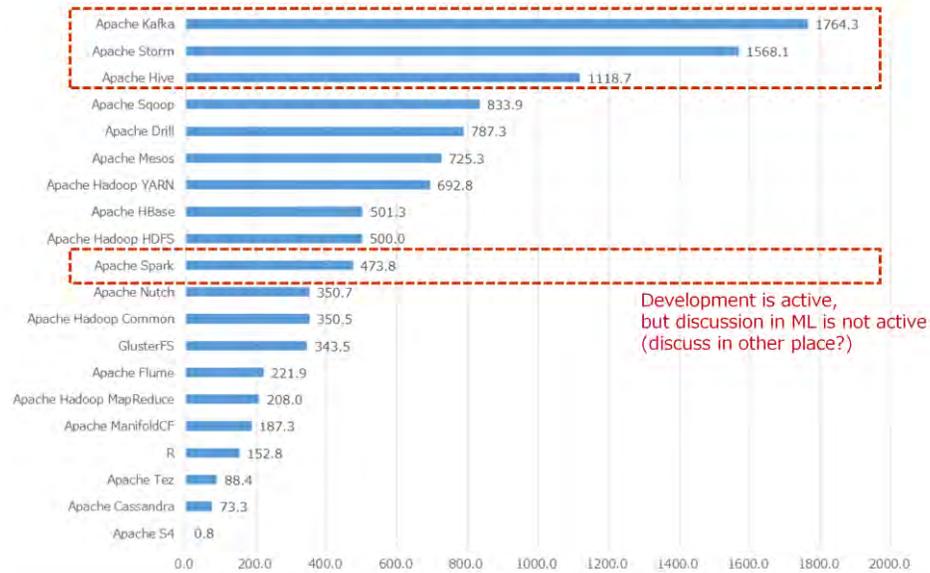
Copyright 2016 Japan OSS Promotion Forum

21

これは、開発者むけのメーリングリストの流量の推移を表したグラフです。
2006年から2009年にかけては、Hadoopの議論が活発でした。
最近ではApache HiveやApache Storm、Apache Kafkaの議論が活発

Average number of mails in ML for developers (2015)

- Developers actively discuss in mailing list of Apache Kafka, Apache Storm and Apache Hive



Copyright 2016 Japan OSS Promotion Forum

22

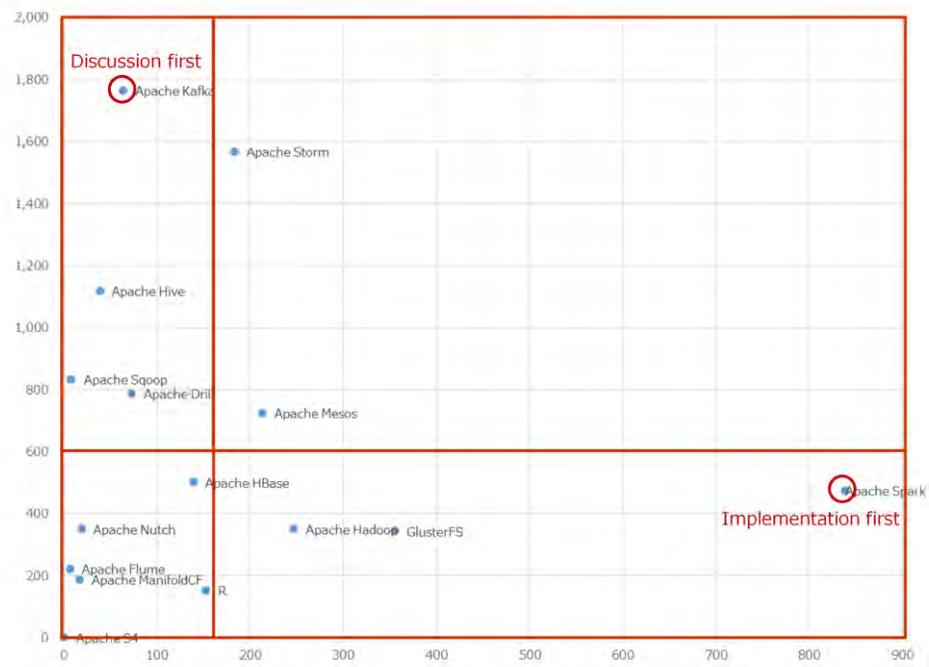
2015年の開発者向けメーリングリストの月平均流量です。

Apache KafkaやApache Stormの議論が活性化していることがわかります。

また、Apache Hiveは2014年をピークとして2015年は落ち着いている模様であることもわかります。

一方、Sparkは、開発は活発ですが、メーリングリストでの議論は、あまり多くありません。

Correlation between commits and mails



Copyright 2016 Japan OSS Promotion Forum

23

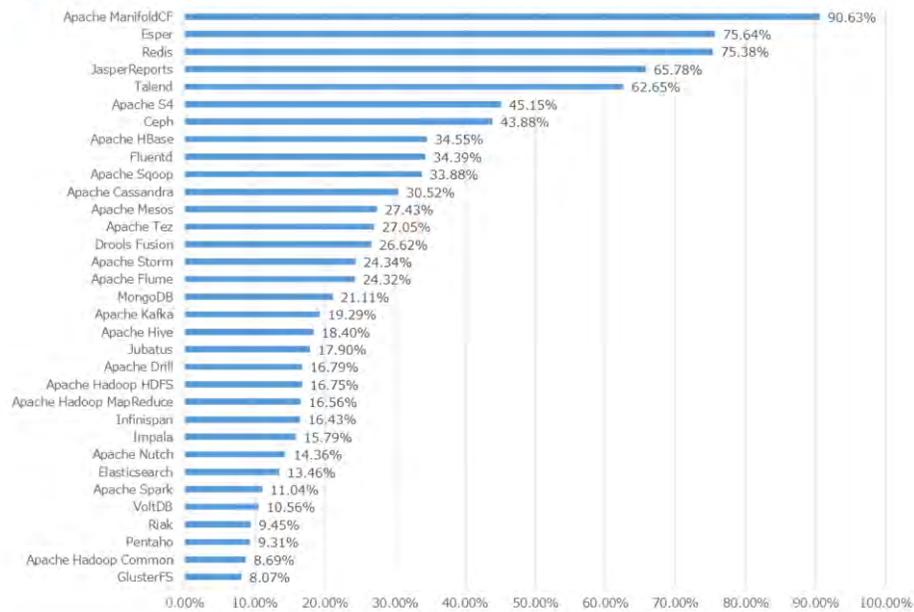
コミット数とメールの議論の相関関係です。

右下のSparkは、議論よりも開発に集中していることがわかります。

一方、左上のKafkaは、まだまだ議論が中心であることがわかります。

Ratio of commits by the top committer

- This ratio may show the power of top committer in the project (?)



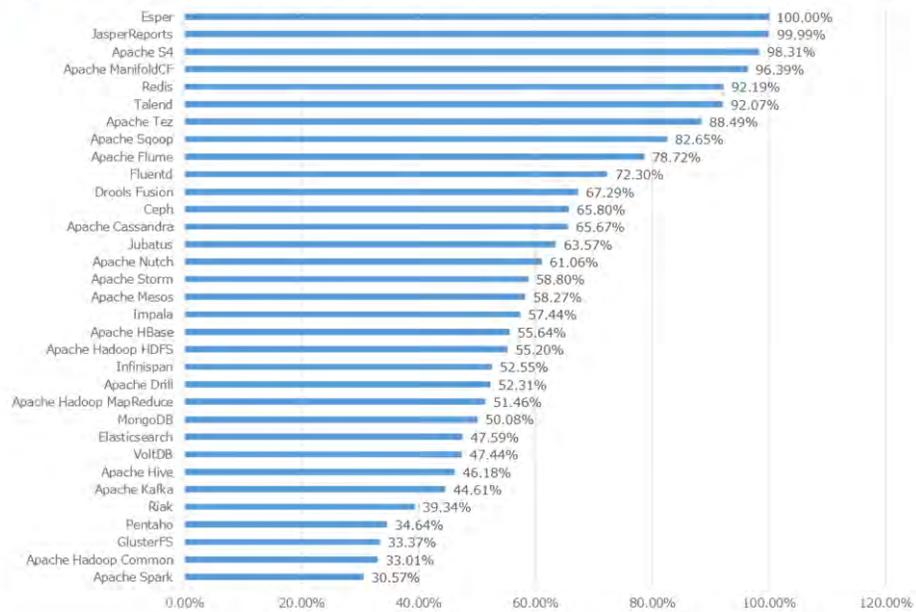
Copyright 2016 Japan OSS Promotion Forum

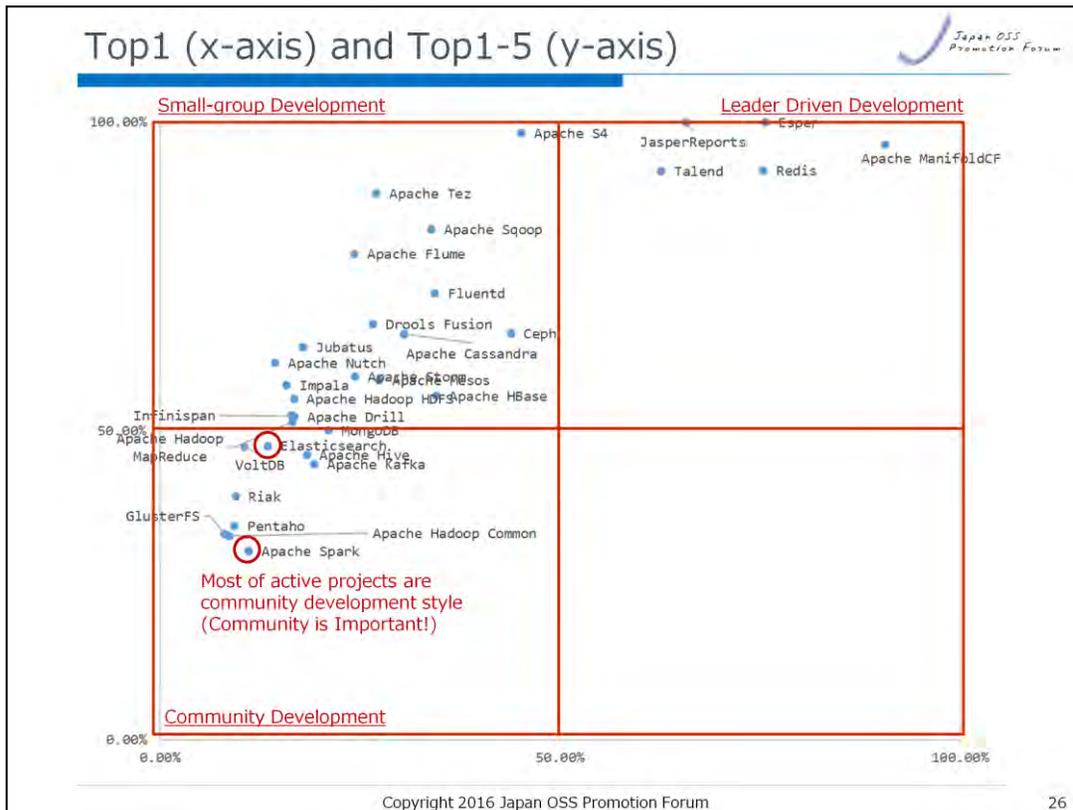
24

トップコミッターのコミット数の比率です。

Ratio of commits by the top 1-5 committers

- Esper and JasperReports may be committed by only 5 committers





コミュニティの開発形態がよくわかるグラフです。
 開発が活性化しているプロジェクトは、コミュニティ型開発が多いことがわかります。
 当初は、リーダーのみの開発から少人数への開発、やがてコミュニティ型へと
 開発のスタイルが変わっていることで、開発も進みますし、
 みなさんに使ってもらえるようになるということが良く分かります。

Users' Activity

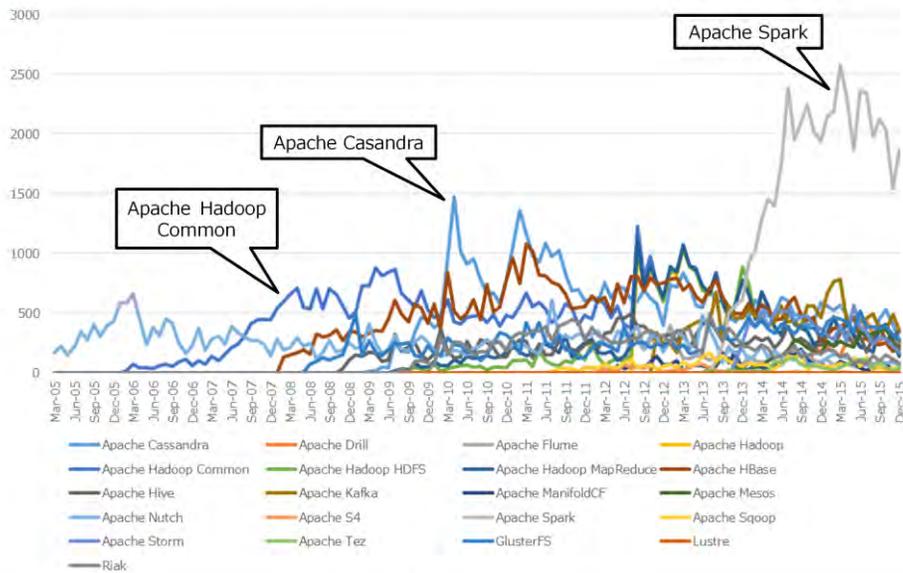
How popular is the software?

次に利用の活性度について説明します。

つまり、そのソフトウェアがどれだけ利用されているかということです。

Mails in ML for users

- Apache Spark has very active user mailing list



Copyright 2016 Japan OSS Promotion Forum

28

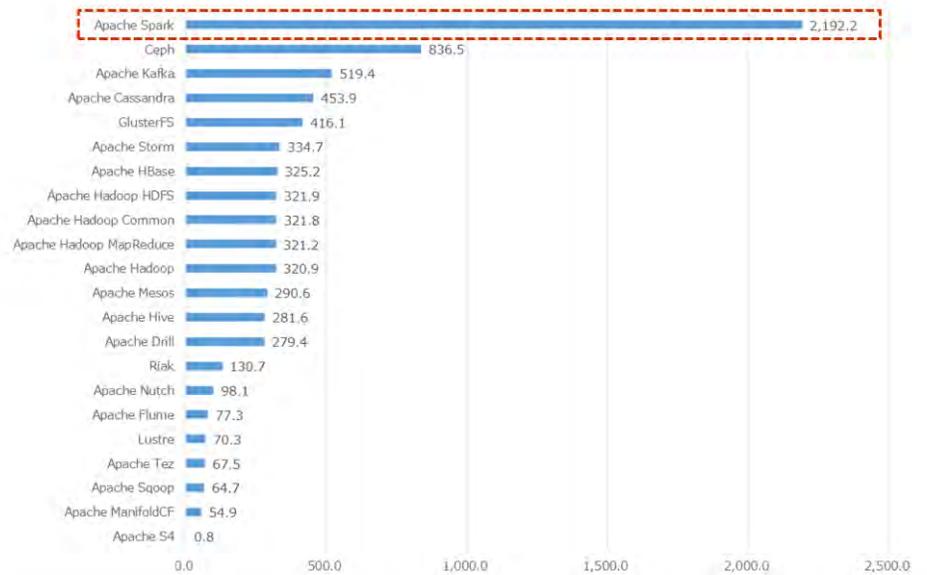
2008年頃は、Hadoopの登場したころなので、質問が集中しています。

その次は、Cassandraについての議論が増えています。

2014年以降は、こちらでもSparkについての議論が圧倒的に活発になっています。

Average number of mails in ML for users (2015) Japan OSS Promotion Forum

- Apache Spark has many active users and active developers

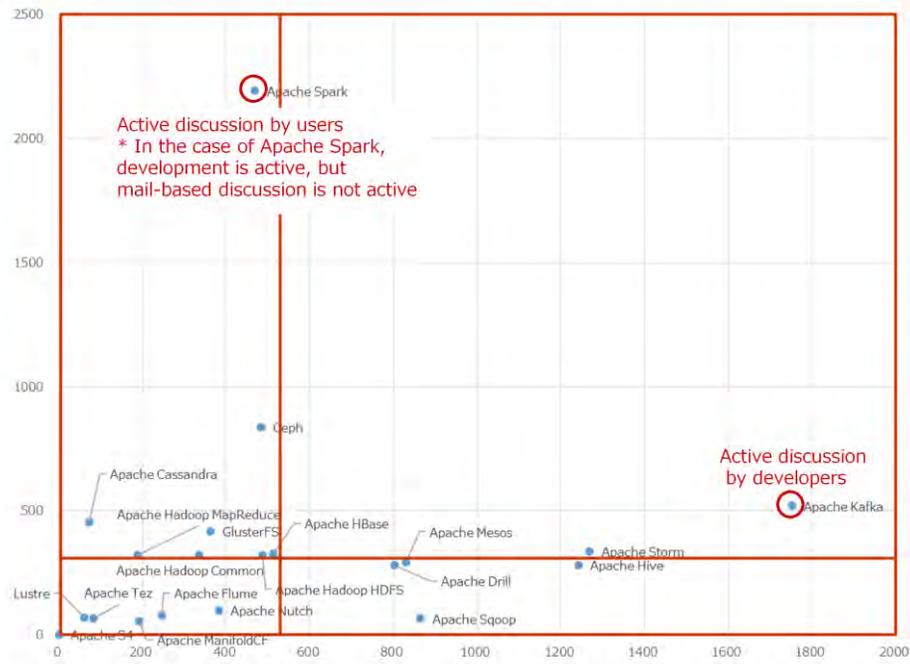


Copyright 2016 Japan OSS Promotion Forum

29

2015年だけを抜き出してグラフにしてみると圧倒的にSparkが活発です。
その次に活発なのは、SDS(Software Defined Storage)のCephです。

Mails by developers (x-axis) and users (y-axis)



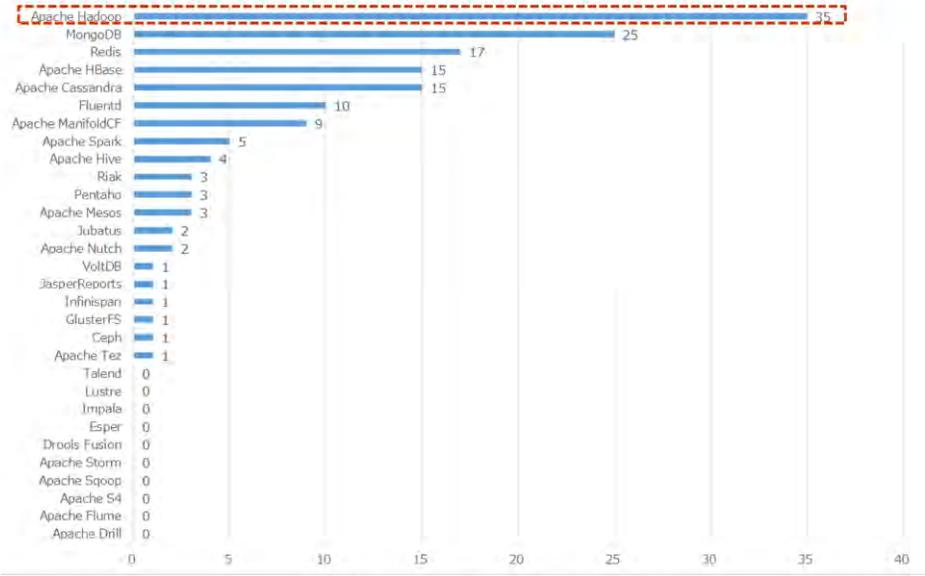
Copyright 2016 Japan OSS Promotion Forum

30

このグラフは、横軸に開発者、縦軸に利用者の相関関係を表したグラフです。左上の**Spark**は、利用者の議論が活発であることがわかります。**Spark**に関しては、先ほど説明したように開発は活発ですが、開発者のメールベースでの議論は、少なめです。かたや、右下の**Kafka**は開発者の議論が活発であることがわかります。しかしながら、まだそれほど利用されていないことが利用者のメールの少なさでわかります。

Books in Amazon (Japanese)

- There are many books about R (for statistic or programming, etc.)
(450 Books, not in graph)
- Then, major software like Apache Hadoop, MongoDB



Copyright 2016 Japan OSS Promotion Forum

31

次にアマゾンでの書籍数を比較しています。

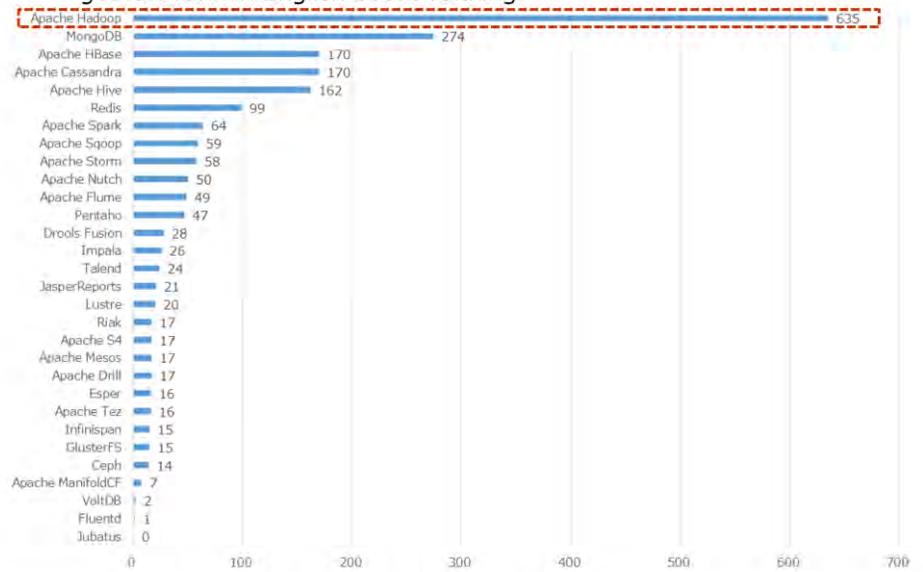
もちろん、日本で調べているので、日本語の書籍の数のみです。

しかしながら、R言語は言語や統計の分野で使われることが多いので、このグラフからは、削除しています。

このグラフでは、Hadoop、MongoDBなどの書籍が多いことがわかります。

Books in Amazon (English)

- Almost same tendency as Japanese (R is not in graph)
 - Software produced by Japanese developers (e.g. fluentd, jubatus) get low rank in English books ranking



Copyright 2016 Japan OSS Promotion Forum

32

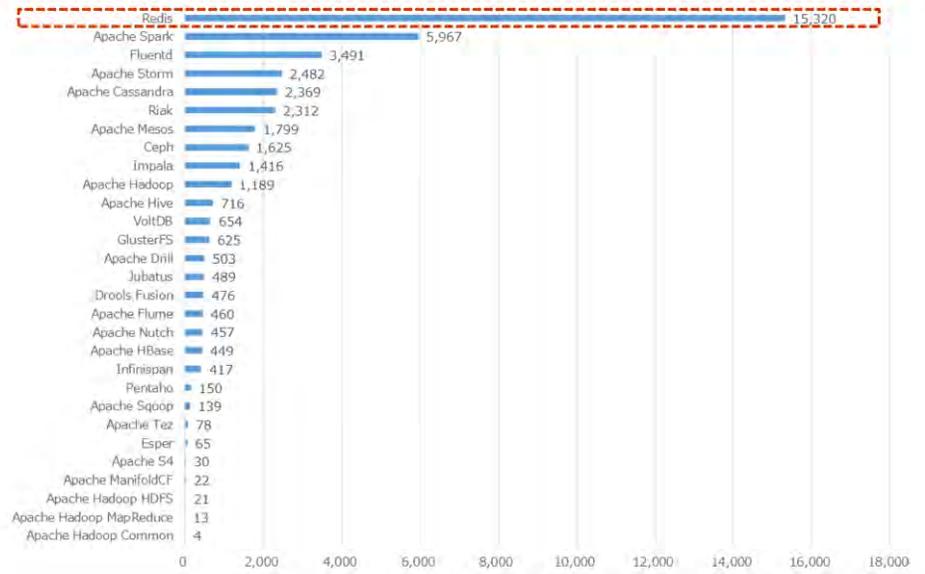
次は、英語の書籍のデータです。

ほとんど日本語の傾向と同じように見えます。

このなかで、**Fluentd**や**Jubatus**は日本人が中心になって開発しているソフトウェアなので、日本語の書籍に比べると少なくなっています。

Stars in GitHub

- Redis gets a large amount of stars
 - The number of stars can be related to the number of enterprise users(?)



Copyright 2016 Japan OSS Promotion Forum

33

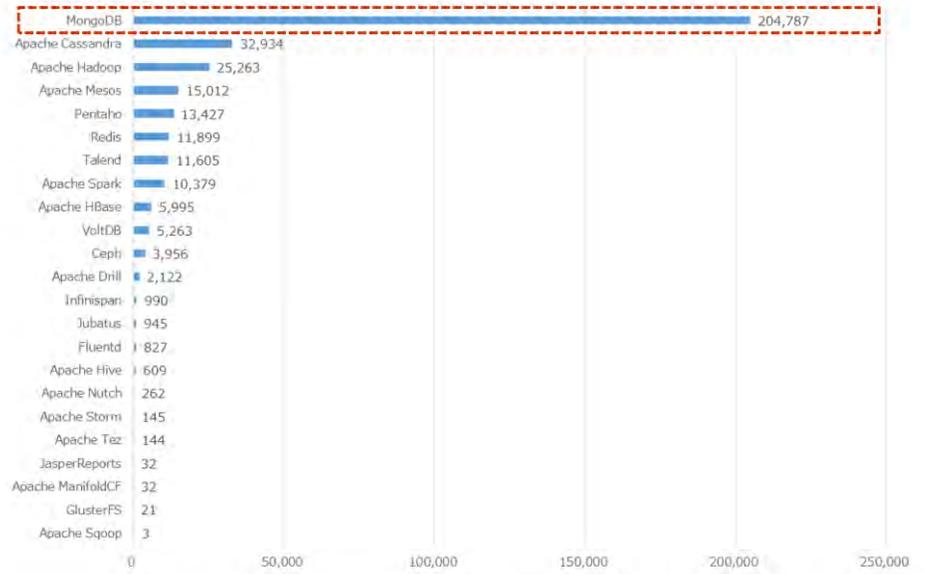
GitHubのスター数です。

これは、スター数が多いほど、人気の高いプロジェクトであることを表しています。
ここでは、Redisが圧倒的によい評価となっています。

商用で展開されているソフトウェアに関しては、組織票があるかも知れませんね。

Followers on Twitter

- MongoDB gets the greatest followers
 - It might depend on how early the account start



Copyright 2016 Japan OSS Promotion Forum

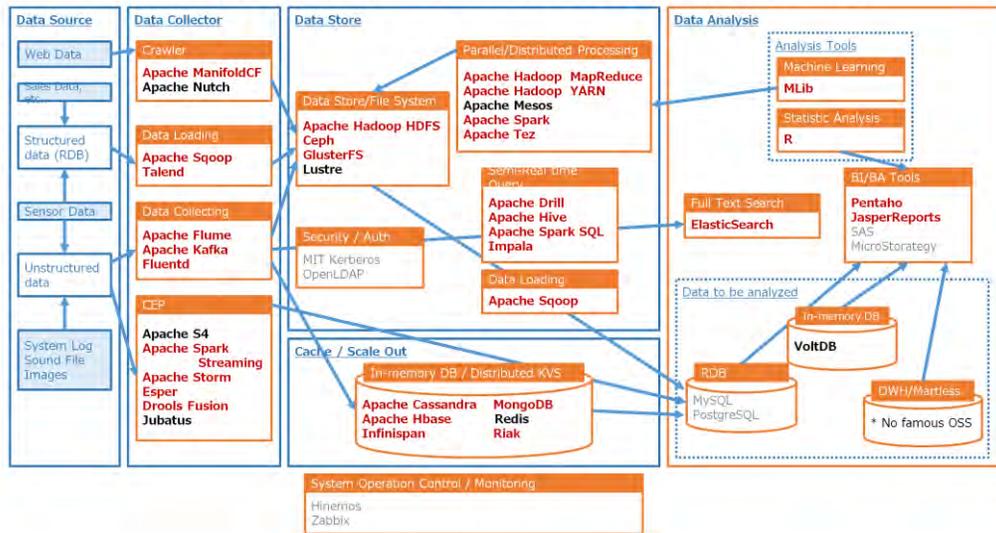
34

次は、Twitterのフォロワー数です。

アカウントの開設時期にもよりますが、MongoDBが圧倒的に多数になっています。

Enterprise support

- We can receive Enterprise Support for software in red letters in Japan
 - We can build the Big Data Platform only with Enterprise supported OSS
 - Some software are provided as a service in Cloud (e.g. Jubatus)



この図は、有償サポートが提供されているものは、赤字で表示してあります。このようにビッグデータ基盤を構成するOSSは、有償サポートされているものが多いので、これらOSSだけで基盤を構築することは十分可能です。また、Jubatusなどはクラウドでサービス提供されているOSSもあります。

Enterprise track record

■ From The Linux Foundation SI Forum 2015 in Japan

Many Introduction Track records	Apache Hadoop, GlusterFS, MongoDB, JasperReports
Some Introduction Track records	Talend, Fluentd, Jubatus, Apache Spark, Ceph, R, Lustre, Apache Cassandra, Apache Hbase, Redis, Elasticsearch, Pentaho
Some Verification Track records	VoltDB
No track records	The Others (that is a little disappointing result)

これは、LinuxFoundationの日本でのWGである「SI Forum」が毎年実施している2015年の活用動向調査の結果です。全体的に拡大傾向ではありますが、本格的な導入はこれからという印象です。

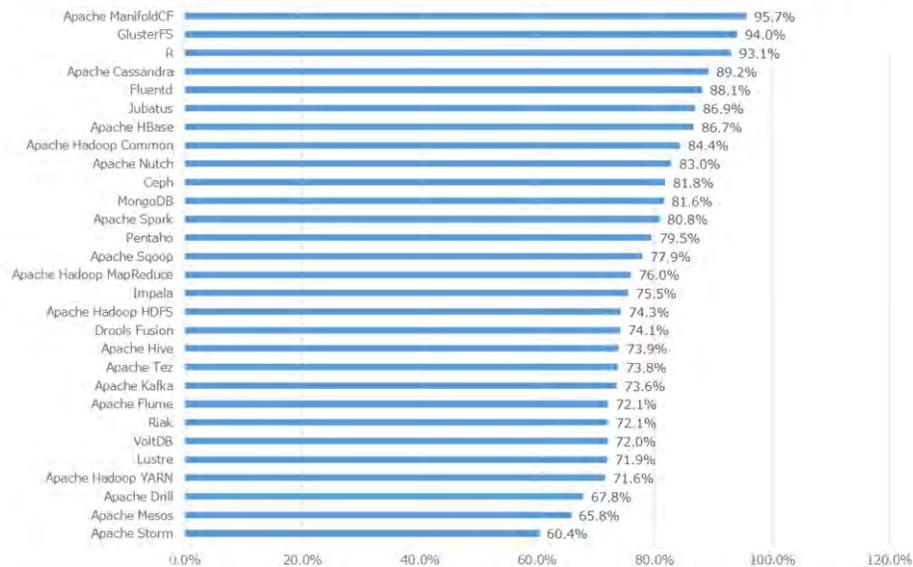
Quality of Software

Can we use the software without defects?

次にソフトウェアの品質について、調査しましたので、ご紹介します。

Bug resolution rate

- Apache ManifoldCF, GlusterFS and R perform high bug fix rate
- Apache Storm performs the lowest bug fix rate, but the rate is 60%



Copyright 2016 Japan OSS Promotion Forum

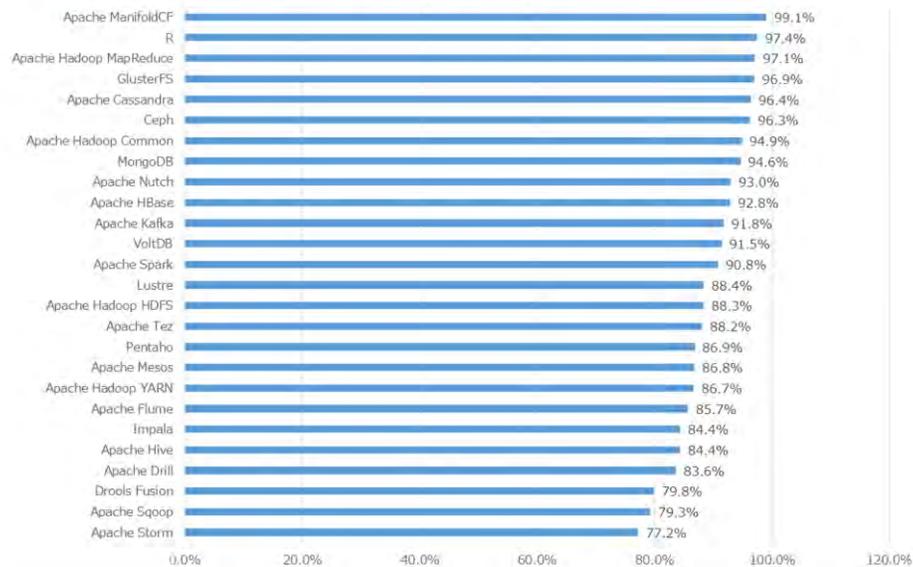
38

このグラフは、バグの解決率のグラフです。

Apache ManifoldCF, GlusterFS や R など解決率が高いという結果がでました。
また、最低のStormでも、60%ということで、全体的に解決率が高いといえることができます。

Blocker/Critical bug resolution rate

- About critical bugs, over 80% of them are resolved
- Apache ManifoldCF also performs high fix rate



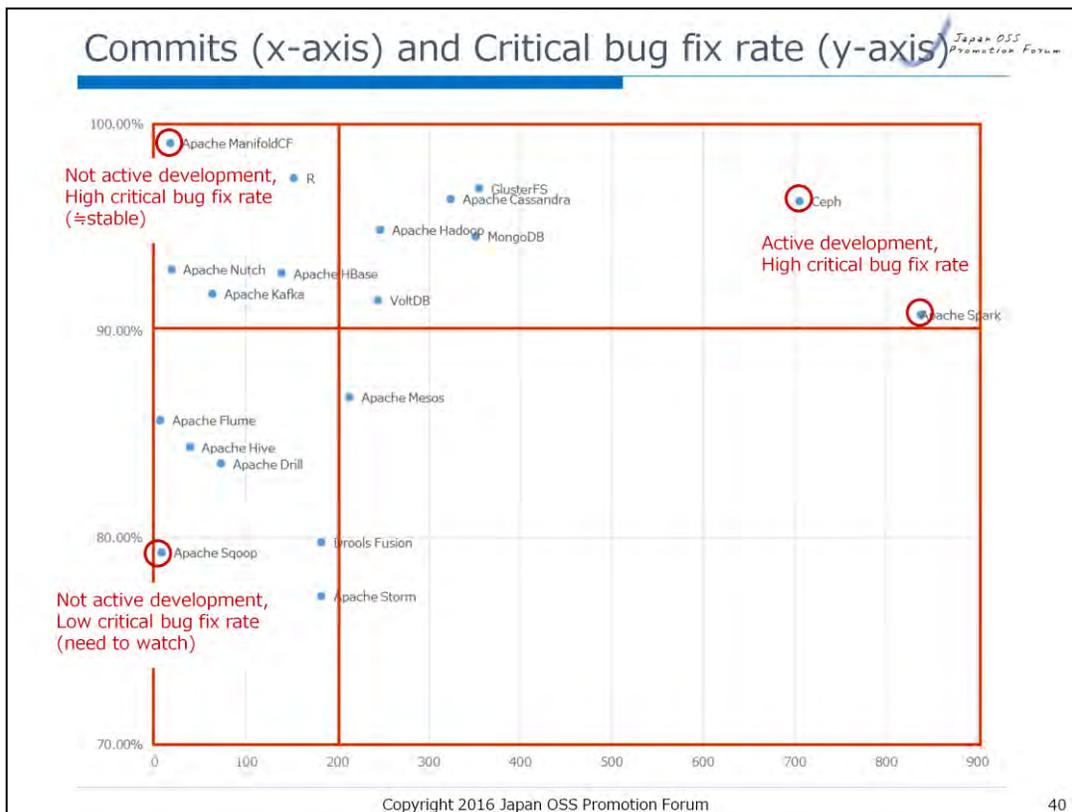
Copyright 2016 Japan OSS Promotion Forum

39

次は、重要なバグの解決率についてです。

重要なバグの解決率に限定すると解決率は80%を超えています。

特に、Apache ManifoldCF の解決率が非常に高くなっています。



コミット数(横軸)と重要バグ解決率(縦軸)の相関関係です。

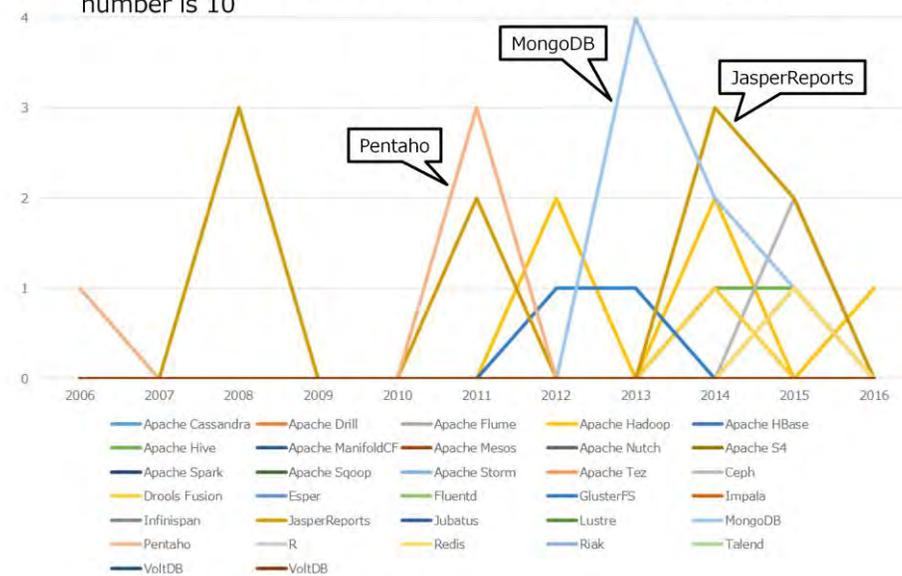
左上の**Apache ManifoldCF** は、開発は活発ではありませんが、重要バグの解決率は高めで、安定しているように見えます。

左下の**Sqoop**は、重要バグの解決率も低く、開発もあまり活発ではないようなので、利用する場合は、注意が必要かも知れません。

また、右上の**Ceph**や**Spark**は、開発も活発で、重要バグの解決率も高いので、利用する場合はそれほど心配しなくても良いかも知れません。

Vulnerabilities (2006 – 2016)

- Totally there are small number of vulnerabilities
 - JasperReports had the largest number of vulnerabilities, but the number is 10



Copyright 2016 Japan OSS Promotion Forum

41

これは、脆弱性の発生状況を表しています。

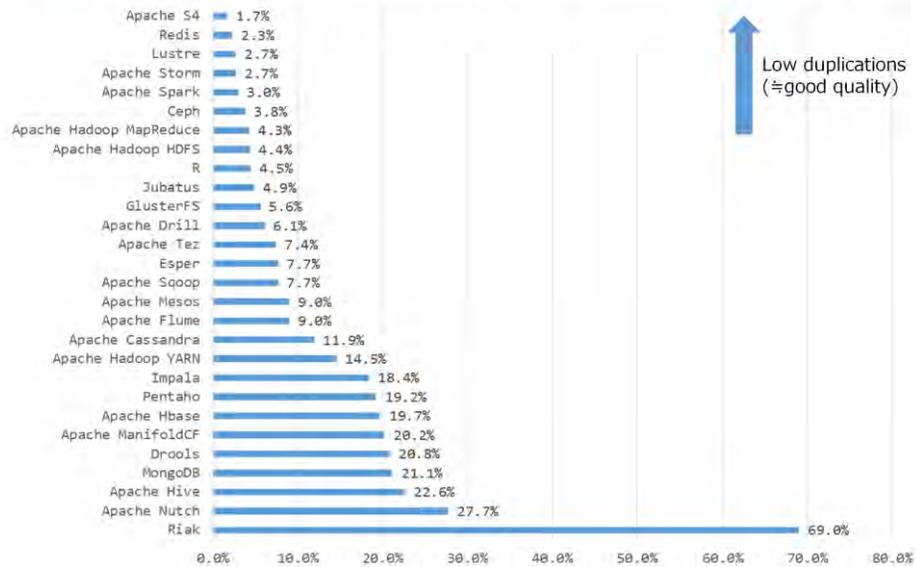
全体的に脆弱性の検出は少なく、セキュリティでの問題はあまり心配しなくても良いようです。

一番多いJasperReportでも全体で10件くらいです。

Duplications in source code

■ Result of static source code analysis by SonarQube

- Apache S4 and Redis have less duplicated source code (maybe good)



↑ Low duplications
(≒good quality)

Copyright 2016 Japan OSS Promotion Forum

42

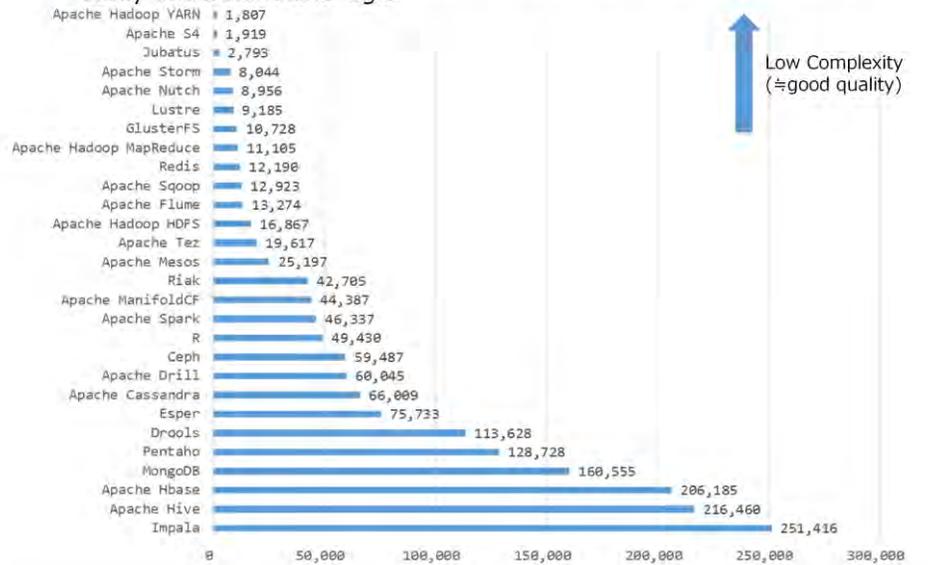
これは、**SonarQube**というソフトウェアのソースコード解析機能を使って抽出したソースコードの重複率です。

ソースコードの重複率が少ないということは、品質が良いということなので、このグラフでは、上にあるソフトウェアほど、品質が良いということになります。

Complexities of source code

■ Result of static source code analysis by SonarQube

- Apache Hadoop YARN, Apache S4 and Jubatus may have relatively easily understandable logic



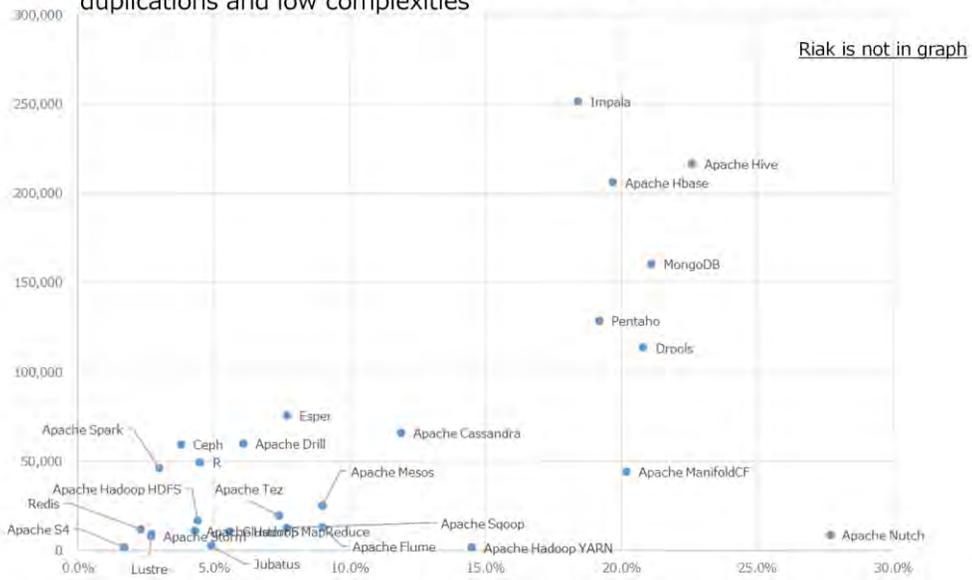
Copyright 2016 Japan OSS Promotion Forum

43

同じくこれは、ソースコード全体の複雑度を表したグラフです。
ソースコードの複雑度が低いほど、高品質であるといえます。

Duplications (x-axis) and Complexity (y-axis)

- There are weak correlation between duplications and complexities
 - However, there are software like Apache Nutch, which has many duplications and low complexities



Copyright 2016 Japan OSS Promotion Forum

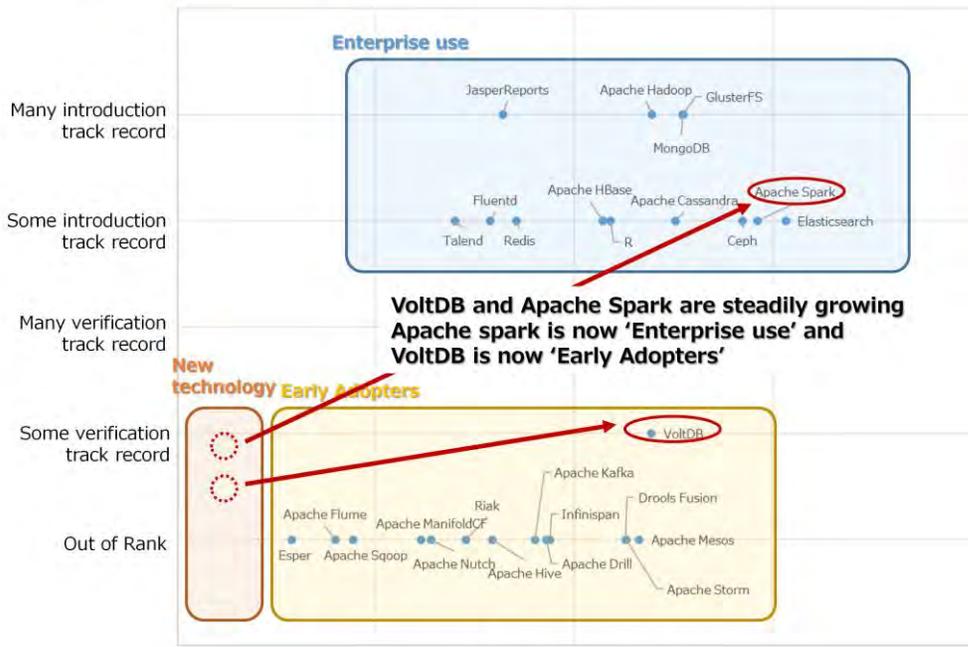
44

このグラフは、重複率(横軸)と複雑度(縦軸)の相関関係を表したものです。何らかの関係があるように見えますが、それほど強い関係があるとはいえない感じです。

Summary

全体とまとめます。

Summary



昨年から今年への変化した部分で見るとVoltDBやSparkが一気に伸びてきています。
特にSparkは、エンタープライズでもすでに事例が出ています。

Conclusion

- We can build the Big Data Platform only with OSS
 - Enterprise supports are getting better
 - However, it is necessary to check the functions and quality of softwares
- Apache Spark and the ecosystem are hot
- Developers for Elasticsearch may be hardworkers
- MongoDB and Ceph are going to be stable

- We should continue to watch the situation of OSS

ビッグデータ基盤をオープンソースだけで構築することが十分に可能となってきました。

また、有償サービスも揃いつつあります。ただし、機能や品質の見極めは当然必要です。

- ・Apache Sparkとそのエコシステムは開発、利用ともに活発度が高い
- ・ElasticsearchはGitコミット日の割合が91%程度でとても開発が活発
- ・MongoDBやCephはこの領域では比較的安定期に入りつつある

→ただし、状況は変化しており今後も継続的なウォッチが必要です。

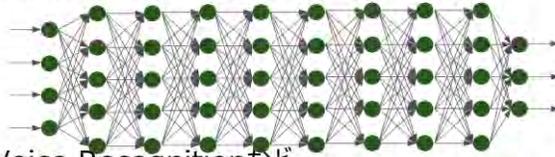
Trend of DeepLearning

次にDeepLearningに動向についてご紹介します。

What is DeepLearning?

■ **Deep Learning** is a type of Machine Learning technique.

- Deep layer, width is to use a broad neural network



■ **What Can do?**

- Image Recognition , Voice Recognitionなど
- There are various types of neural network, the area of application is different
- Higher accuracy than the system using the conventional Machine Learning

Name	Application Areas
Convolutional Neural Network	CNN <u>Image Recognition</u>
Recurrent Neural Network	RNN <u>Voice Recognition</u> , Video Recognition, NLP
Long Short-Term Memory	LSTM <u>Voice Recognition</u> , Video Recognition, NLP
Deep Belief Network	DBN Attribute Classification
Deep Boltzmann Machine	DBM Attribute Classification
Deep Auto Encoder	DAE Feature Extraction
Restricted Boltzmann Machine	RBM Feature Extraction, Attribute Classification, NLP

(*)NLP : Natural Language Processing

Copyright 2016 Japan OSS Promotion Forum

49

まず、DeepLearningとは何かということを説明します。

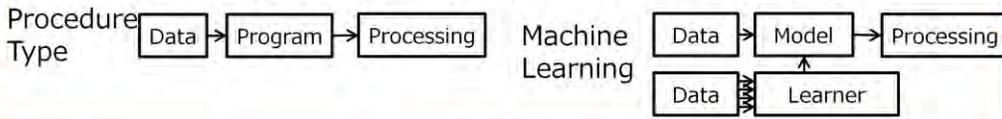
DeepLearningは、機械学習の手法の一種で、
多層に重ねたニューラルネットワークを利用しています。

何ができるかという、現在一番活用されている部分は、
画像認識や音声認識になります。

とはいえ、さまざまな種類のニューラルネットが存在するので、適用領域が異なります。
また、従来の機械学習より、精度が高くなっています。

What is Machine Learning?

Acquisition by learning the behavior of the computer (model) from the (large amounts of) data



Problem	Regression Stock price prediction, etc.	Classification E-mail filters, etc.	Clustering E-mail filters, etc.
Category	Supervised Learning		Unsupervised Learning
Algorithm	<ul style="list-style-type: none"> •GLM •Discriminant analysis •Decision Tree •SVM •Neural Network 	<ul style="list-style-type: none"> •Logistic Regression •K-nearest neighbor Algorithm •Naive Bayes classifier •Random forest 	<ul style="list-style-type: none"> •K-means clustering •Hierarchical clustering •Apriori •One-Class SVM

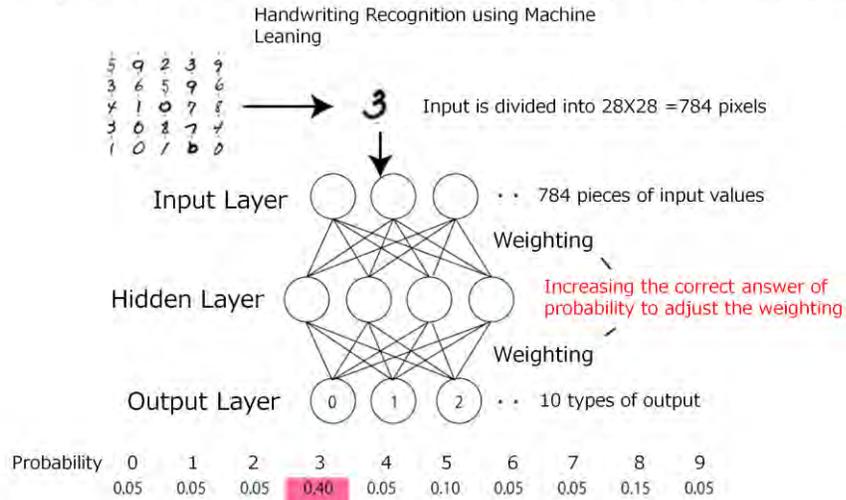
次に機械学習について、説明します。

機械学習とは、コンピュータの振舞い方を大量のデータからコンピュータ自身に学習させることです。

解決できる問題は、株価予測や迷惑フィルタなどになります。

GLM: 線形モデル

- Mimics the mechanism of neural circuitry of the human brain model
- Examples of handwriting recognition using a neural network



次にニューラルネットワークです。

これは、人間の脳の神経回路を模したモデルです。

手書き文字の認識の例を示します。

このように**784**ピクセルに分割してデータを入力し、それぞれのデータに重み付けをし、その重み付けを調整することで、正解にたどり着くようにします。この例の場合では、「3」の確率が一番高くなっています。

Why now deep learning ?

■ Third time AI boom spark

- SuperVision : Dramatically Improve the accuracy of Image Recognition
 - Landslide victory at Large Scale Visual Recognition Challenge2012.
- Google : Recognize the cat without a teacher(2012)
 - 10 billion of the neural network over a period of 3 days at 1,000 units of machine analysis

It reaffirmed the validity of the deep learning,
recognized the feasibility

■ "Recognized the feasibility" is :

- Need a great deal of computer resources for learning
⇒ Computing power improvement of the computer has to be realistic.
- A large amount of learning required (for learning) data
⇒ Internet development, facilitated the availability of training data

なぜ、ディープラーニングかというお話をします。

現在、第三次人工知能ブームといわれています。

最初は、1956年のダートマス会議で初めて「人工知能」という用語が使われたことです。このころは、数学の定理を証明するのが、関の山でした。

次は、1980年代で、そのころは専門家の思考過程をシミュレーションするような「エキスパートシステム」が流行りました。

今回は、以下の二つの事例が大きく影響しています。

一つ目は、2012年に画像認識コンテストで、従来の方法に比べて2倍の精度で認識できたことです。

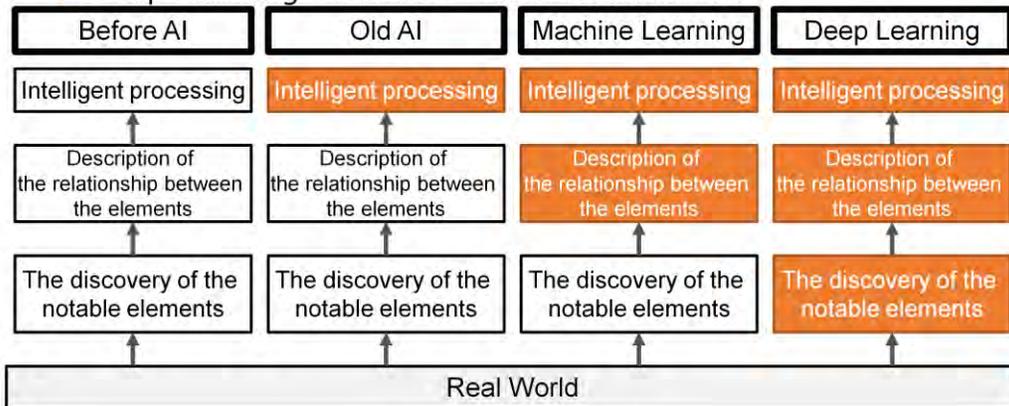
二つ目は、Googleが1000台のコンピュータを三日間かけて学習した結果、猫の画像を識別できるようになったことです。

このような事象が大きく影響し、ディープラーニングが有効であることが再認識されました。

しかしながら、コンピュータの計算能力が大幅に向上したこととインターネットの発展によって、大量の学習データが入手できるようになったことも大きな要因となっています。

The difference of "Machine Learning" and "Deep Learning" Japan OSS Promotion Forum

- Deep Learning broke the "Moravec's Paradox (1988)"
 - "Moravec's Paradox" = 「Impossible to give them the skills of a one-year-old when it comes to perception and mobility」
- Why this paradox exists
 - "Feature Extraction" is the most difficult process
 - Deep Learning automate "Feature Extraction"



Copyright 2016 Japan OSS Promotion Forum

53

機械学習とディープラーニングの違いについて、説明します。

ディープラーニングは、子供のできることの方が難しいというモラベックのパラドックスを崩したと言われています。

人工知能には、「大人の人工知能」と「子供の人工知能」があると
言われています。要素間の関係の記述については、データを分析
することで認知できますが、注目すべき要素の発見は、データだけ
では、獲得することができません。言葉をどのように獲得するかという
ような言葉で説明できない要素の発見を自らできるよう
なっています。また、その実現スピードも速くなってきています。

Comparison of Popular Deep Learning OSS

Developer Software	Lang	Area	License, First Release, Feature, etc
BVLC Caffe	C++	Image	BSD License, 2013/10, CPU/GPU Support. Appearance time is at an early OSS, specializing in the Image. Fast, but low flexibility. Suitable for CNN. High description degree of difficulty of the RNN
Preferred Networks, Inc Chainer	Python	Image Voice Text	MIT License, 2015/6, CPU/GPU Support, Japanese Developer, Japanese Documents Exist. High flexibility. Corresponding to CNN, RNN
Google社 TensorFlow	C++ Python	Image Voice Text	Apache License, 2015/11, CPU/GPU Support. Just released, but proud of the overwhelming popularity Corresponding to CNN, RNN

次にディープラーニングを実現する代表的なオープンソースのフレームワークを説明します。

一つ目は、「**Caffe**」です。これは、米国のバークレー大学の研究所で開発されたもので、**2013年の10月**という早い時期に登場したものです。画像に特化した高速ですが、柔軟性がひくいという特徴があります。

二つ目は、「**Chainer**」です。これは、日本のプリファードネットワークという会社が開発したものです。日本で開発されたこともあって、日本語のドキュメントも多く日本人としては、とっつきやすいものになっています。

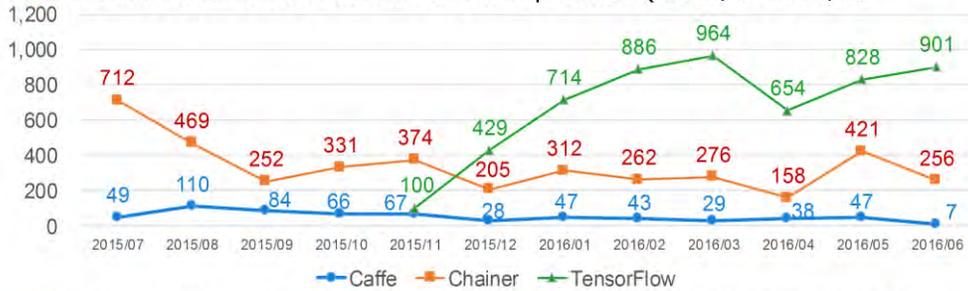
三つ目は、「**TensorFlow**」です。これは、**Goggle**が開発したもので、**2015/11**にOSS化されたということで、一番新しいのですが、圧倒的な人気を誇っています。

注目すべきは、これら三つとも**BSD系**のライセンスを採用していることです。すなわち、自社製品の中に取り込めるという特徴があります。これらフレームワークの普及のカギは、そのあたりにあるかもしれません。

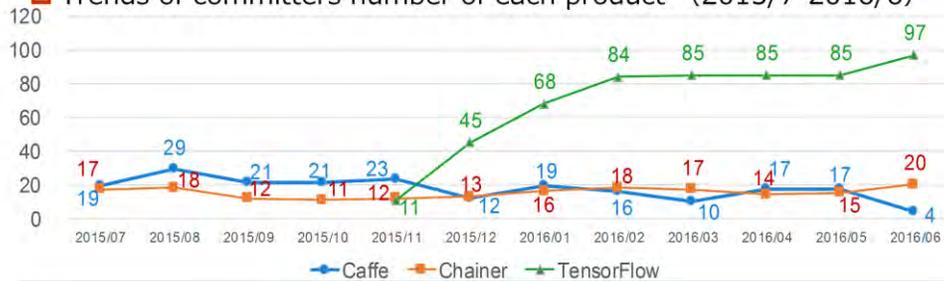
Comparison of the development situation

■ The most active of the development of TensorFlow

□ Trends of commits number of each product (2015/7-2016/6)



□ Trends of committers number of each product (2015/7-2016/6)



Copyright 2016 Japan OSS Promotion Forum

55

開発状況の比較をしてみましょう。

各製品のコミット数の推移を見てみると、TensorFlowのコミット数が非常に伸びていることがわかります。

また、コミット数の推移をみてもTensorFlowが圧倒的に伸びています。

このような状況を見るとTensorFlow開発が最も活性化していると言てよいを思います。

Verification environment

- OS
 - Ubuntu 14.04
- Virtual Machine
 - CPU 1core 2.4GHz
 - Memory 2GB
- Software
 - Python 2.7.6
 - Caffe (master)
 - Chainer 1.9.1
 - TensorFlow 0.9.0

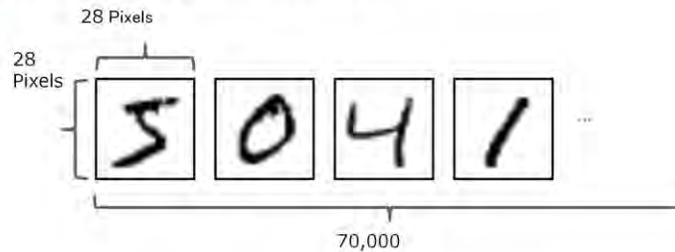
次に検証環境をご紹介します。

皆さんがお持ちのPCのような環境です。

■ MNIST

- Dataset correct label is applied to the handwritten numbers 0-9
 - Each handwritten numbers, provided as an image of 28 pixels × 28 pixels
 - For each image, provided together also label data to be correct
 - Image, provide a total of 70,000 (60,000 for learning, 10,000 for testing)

□ The most common model



検証モデルについて、説明します。

これは、MNISTから提供されているデータを採用します。

手書きの数字0から9に正解ラベルが与えられているデータで、
28×28ピクセルの画像で提供されています。

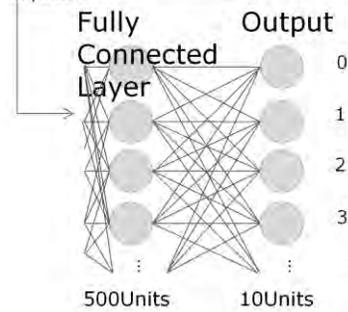
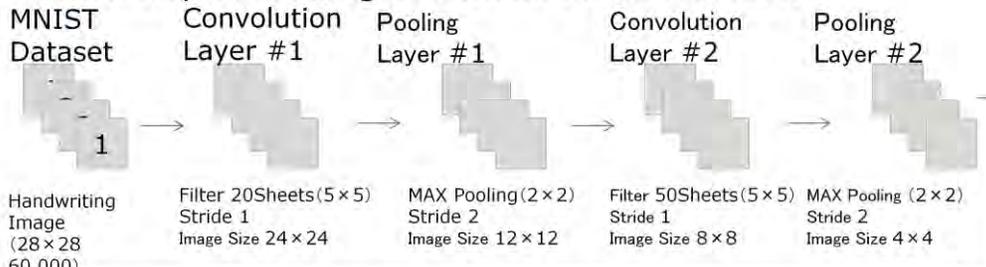
各画像には、正解となるデータも提供されています。

画像は、学習用60,000枚、テスト用10,000枚の70,000枚を提供されています。
。

このモデルは、機械学習やディープラーニングの世界で
最も一般的なモデルで、いわゆる「Hello World」のようなものです。

Neural network architecture

■ Modify according to official MNIST of Caffe



Convolutional Neural Network(CNN) Model
Middle 2 Layers(Conventional・Pooling × 2)
Optimization :SGD
Dropout Rate:0.5(Fully Connected Layer)
Activation Function:ReLU,Sigmoid,TanH
(Conventional 2Layers・ Fully Connected Layer)

- Caffe Sample
<https://github.com/BVLC/caffe/tree/master/examples/mnist>
- Chainer Sample
<https://github.com/pfnet/chainer/archive/v1.9.1.tar.gz>
- TensorFlow Sample
<https://www.tensorflow.org/versions/master/tutorials/mnist/pros/index.html>

CaffeのMNISTを使用したサンプルにほかのフレームワークも合わせた形で、検証のパターンを作成しました。

CNNと呼ばれる畳みこみニューラルネットワークモデルを採用しています。

中間は2層で、畳み込みとプーリングをそれぞれ2層)

最適化手法は、SGD関数

ドロップアウト率は、0.5(全結合層)

活性化関数は、ReLU,Sigmoid,TanH(畳み込み2層・全結合層

を使用しています、

Verify condition(1)

- The “Learning Time” and the “Percentage of correct answers” due to the changes in the “Learning count” measure

Learning count	Learning rate	Mini Batch	Activation Function	Optimization
2,000	0.01	100	ReLU	SGD
4,000	0.01	100	ReLU	SGD
6,000	0.01	100	ReLU	SGD
8,000	0.01	100	ReLU	SGD
10,000	0.01	100	ReLU	SGD

検証条件の一つ目は、
学習回数の変化に伴う学習時間及び正答率の変化です。

Verify condition(2)

- The “Learning time” and the “Percentage of correct answers” due to the changes in the “Learning rate” measure

Learning count	Learning rate	Mini Batch	Activation Function	Optimization
10,000	<u>0.0001</u>	100	ReLU	SGD
10,000	<u>0.0005</u>	100	ReLU	SGD
10,000	<u>0.001</u>	100	ReLU	SGD
10,000	<u>0.005</u>	100	ReLU	SGD
10,000	<u>0.01</u>	100	ReLU	SGD
10,000	<u>0.05</u>	100	ReLU	SGD
10,000	<u>0.1</u>	100	ReLU	SGD
10,000	<u>0.5</u>	100	ReLU	SGD

二つ目は、
学習率の変化に伴う、学習時間及び正答率の変化です。

Verify condition(3)

- The "Learning time" and the "Percentage of correct answers" due to the changes in the "Learning rate" measure

Learning count	Learning rate	Mini Batch	Activation Function	Optimization
2,000	0.01	100	<i>Sigmoid</i>	SGD
4,000	0.01	100	<i>Sigmoid</i>	SGD
6,000	0.01	100	<i>Sigmoid</i>	SGD
8,000	0.01	100	<i>Sigmoid</i>	SGD
10,000	0.01	100	<i>Sigmoid</i>	SGD
2,000	0.01	100	<i>TanH</i>	SGD
4,000	0.01	100	<i>TanH</i>	SGD
6,000	0.01	100	<i>TanH</i>	SGD
8,000	0.01	100	<i>TanH</i>	SGD
10,000	0.01	100	<i>TanH</i>	SGD

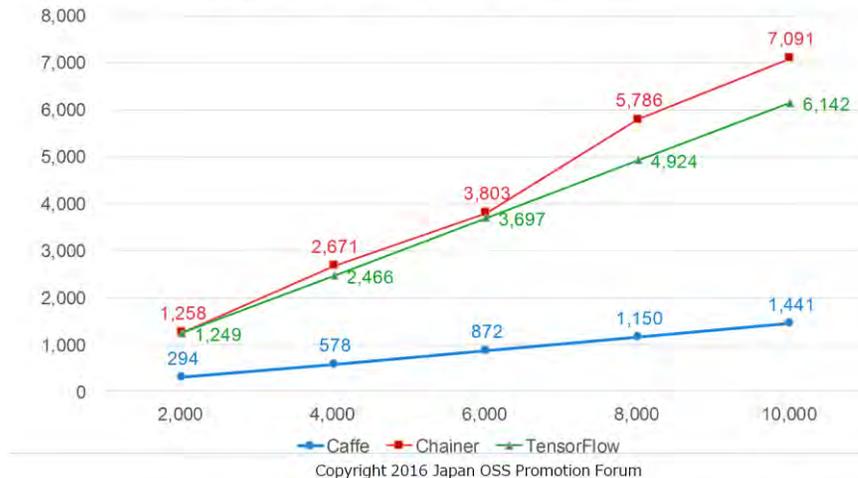
三つ目は、
活性化関数の種類に伴う、学習時間及び正答率の変化を測定します。

Changes the "Learning time" due to the "Learning count"

Japan OSS
Promotion Forum

■ The "Learning time" and the "Percentage of correct answers" due to the changes in the "Learning rate" measure

- Caffe : 1,441sec/1,000, Chainer : 7,091sec/1,000 (Until "Learning Count" is 6,000), TensorFlow : 6,142sec/1,000



Copyright 2016 Japan OSS Promotion Forum

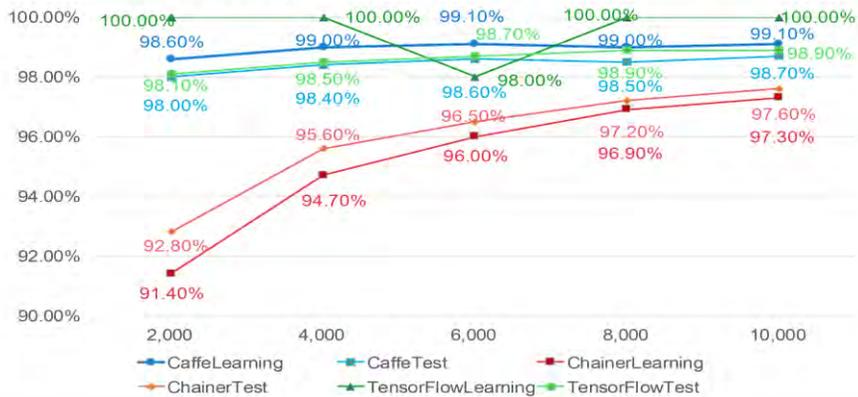
62

<事実のみ>

- Caffeは、各学習回数において、他フレームワークよりも約4.0～4.7倍高速に学習
- Caffe、TensorFlowともに学習回数に単純比例、Chainerもほぼ学習回数に単純比例
- Chainer、TensorFlowの学習時間は、学習回数6,000までであればほぼ同じ
- Chainerは、学習回数が6,000より大きくなると、何らかの原因で学習時間が増加
- ※一般的に学習回数が少なすぎると正答率が低く、多くなると学習時間が大きくなる
- ※一般的に中間層の数や中間層の要素数が多くなると正答率は大きくなるが、時間がかかる。いずれ頭打ちになる。

Changes in the "Percentage of correct answers"
by the "Learning count"

- As the "Learning count" increases the "Percentage of correct answers" of test.
 - TensorFlow is, peaking around the test correct answer rate has exceeded the number of times of learning 8,000
 - Chainer is possible the "Percentage of correct answers" of test improved by increasing the "Learning count"



Copyright 2016 Japan OSS Promotion Forum

63

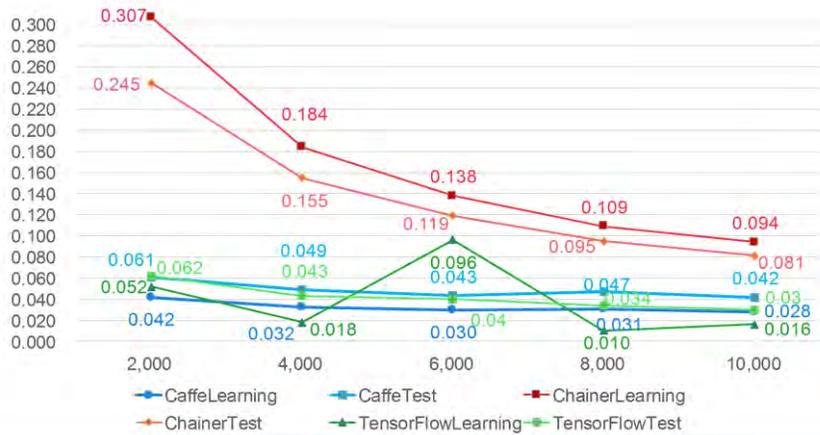
<事実のみ>

- Caffeは、学習回数が大きくなっても正答率がほぼ横ばい
 - Caffeは、各学習回数において、学習時とテスト時の正答率の差が常に小さい
 - Chainerは、学習回数が大きくなるにつれ、正答率が順調に増加
 - Chainerは、各学習回数において、他フレームワークよりも正答率が低い
 - TensorFlowは、学習回数が大きくなっても正答率がほぼ横ばい
 - TensorFlowは、学習回数2,000回で、学習時の正答率100%
 - TensorFlowは、全ての学習回数において、他フレームワークより学習時とテスト時の正答率の差が大きい
- ※一般的に学習回数が少なすぎると正答率が低く、多くなると学習時間が大きくなる

Changes in the loss at the time of learning and testing by the "Learning count"

■ As the "Learning count" increases, test loss steadily decline

- TensorFlow is, the percentage of correct answers of the test, but had leveled off around that exceeds the number of times of learning 8,000, test loss steadily decline



Copyright 2016 Japan OSS Promotion Forum

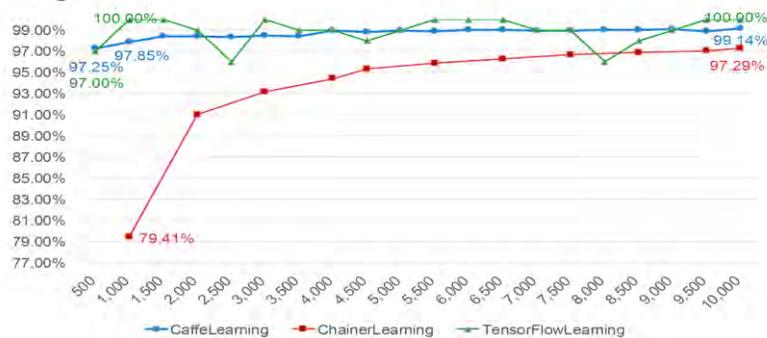
64

<事実のみ>

- Caffeは、学習回数が大きくなるにつれ、損失がゆらぎながら低下
- Chainerは、学習回数が大きくなるにつれ、損失が順調に低下
- Chainerは、学習時・テスト時ともに、損失が他フレームワークに比べ常に大きい
- TensorFlowは、学習時の損失がゆらぎながら低下
- TensorFlowは、テスト時の損失は順調に低下
- TensorFlowは、学習回数4,000、8,000、10,000の場合、学習時の損失が他フレームワークに比べ最も低い

Changes in the percentage of correct answers at the time of learning by the learning count

- Caffe, Chainer is, as the learning progresses, the percentage of correct answers improves
 - Chainer is, there is the possibility of the percentage of correct answers improved by increasing the number of times of learning not learning converges
 - TensorFlow is, faster convergence of learning, there is a high percentage of correct answers in a small number of times of learning



Copyright 2016 Japan OSS Promotion Forum

65

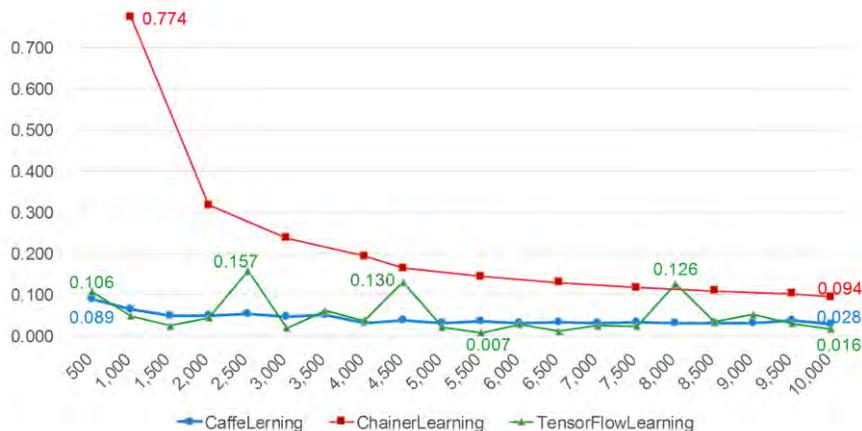
<事実のみ>

- Caffeは、他フレームワークに比べ学習回数の増加に伴う正答率が安定
- Chainerは、正答率の伸び率は最も大きいですが、正答率は他フレームワークより低い
- TensorFlowは、正答率に揺らぎがあり、安定していないが、他フレームワークに比べ高い正答率(学習回数1,000の時点で既に正答率100%)

Changes in the loss at the time of learning by the number of times of learning

■ As the learning progresses, learning loss is steadily reduced

- Chainer is slow convergence, a large loss
- TensorFlow fast convergence, less loss



Copyright 2016 Japan OSS Promotion Forum

66

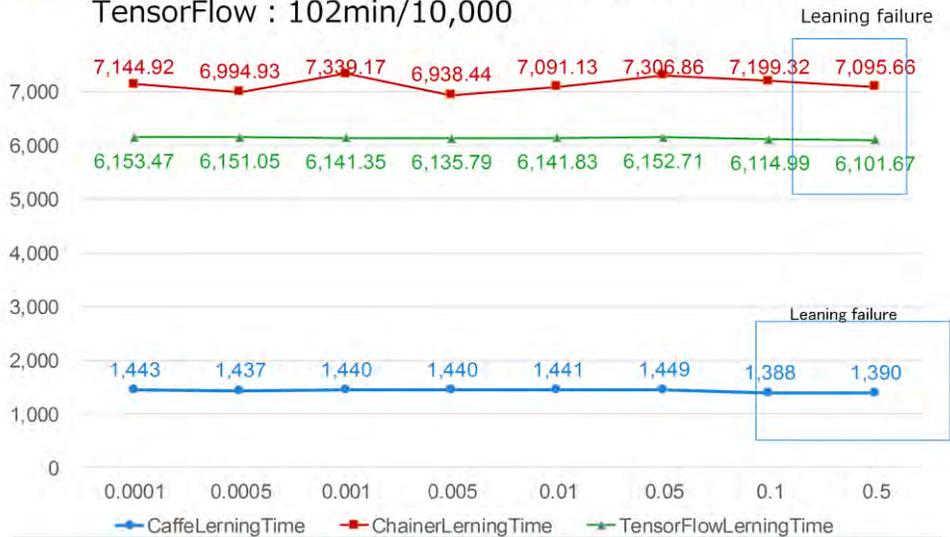
<事実のみ>

- Caffeは、多少の揺らぎはあるが、順調に低下
- Chainerは、下げ幅は大きいですが、他フレームワークに比べ損失が大きい
- TensorFlowは、揺らぎがあり、安定しないが、他フレームワークに比べ損失が小さい

Changes in the learning time by learning rate

■ Learning time is almost constant regardless of the learning rate

□ Caffe : 24min/10,000、Chainer : 119min/10,000、TensorFlow : 102min/10,000



Copyright 2016 Japan OSS Promotion Forum

67

<事実のみ> ※Caffeは、学習率0.1、0.5で学習失敗(発散)したため、学習時間を反映していない。

•Caffe、TensorFlowは、学習率に依らず学習時間は一定

(Caffeは、各学習率において、他フレームワークにより4.2~4.8倍高速)

•Chainerは、学習時間に揺らぎがあり安定しない

※一般的に、学習率が大きくなると過学習、小さくなると収束に時間がかかるようになる。

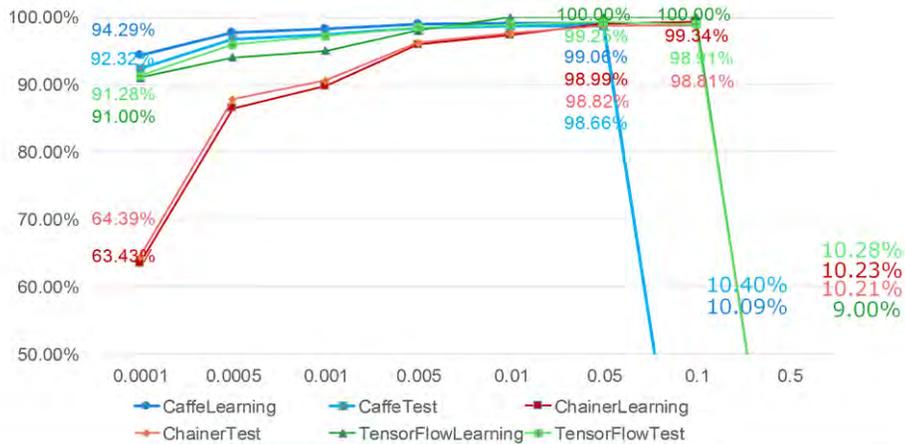
※一般的に学習率が小さすぎると学習が遅いが、Caffe, TensorFlowの場合は、常に一定の学習時間となっている。Chainerは、学習時間が波打っているが...

※確率的勾配降下法(SGD)だと学習率が大きい場合に、全てがNanになってしまう(wが行方不明になる)つまり、Caffeの場合、学習率を大きくしすぎて学習失敗(発散)してしまった。<http://www.slideshare.net/ToruUenoyama/tensorflow-gdg>

※Caffe は、学習率0.1、0.5で学習に失敗

Changes in the percentage of correct answers a by the learning rate

- As the learning rate increases, steadily increase the percentage of correct
- Caffe is, fail to learn to be larger than the learning rate 0.05
- Chainer, TensorFlow is, fail to learn to be larger than the learning rate 0.1



Copyright 2016 Japan OSS Promotion Forum

68

※一般的に、学習率が大きくなると過学習、小さくなると収束に時間がかかるようになる。

※一般的に学習率が小さすぎると学習が遅いが、Caffe, TensorFlowの場合は、常に一定の学習時間となっている。Chainerは、学習時間が波打っているが...

※確率的勾配降下法 (SGD)だと学習率が大きい場合に、全てがNanになってしまう (wが行方不明になる)つまり、Caffeの場合、学習率を大きくしすぎて学習失敗 (発散)してしまった。http://www.slideshare.net/ToruUenoyama/tensorflow-gdg

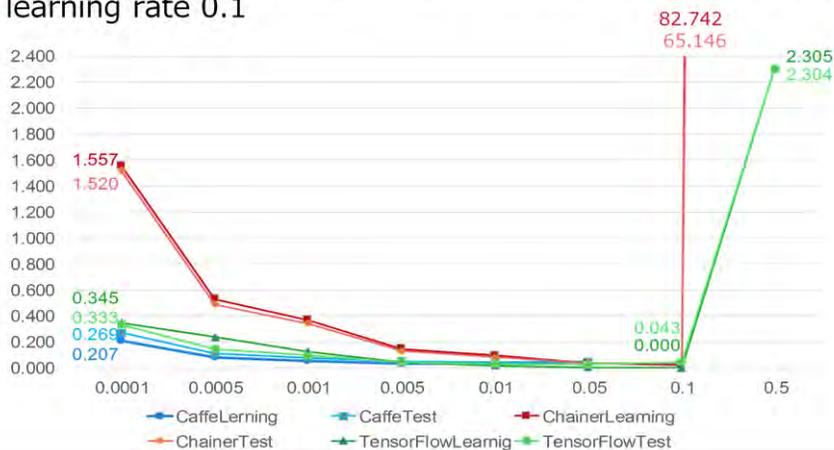
※Caffe は、学習率0.1、0.5で学習に失敗

<事実のみ>

- Caffeは、学習率を0.05より大きくすると学習に失敗 (発散)
- Chainerのみ、学習率0.5において高い正答率 (学習に失敗しない)
- TensorFlowは、学習率を0.1より大きくすると学習に失敗 (発散)

Changes in the loss by the learning rate

- As the learning rate is increased, the loss at the time of learning and testing to steadily decline
 - Caffe is, fail to learn to be larger than the learning rate 0.05 (Nan Output)
 - Chainer, TensorFlow is, fail to learn to be larger than the learning rate 0.1



Copyright 2016 Japan OSS Promotion Forum

69

<事実のみ>

- Caffeは、学習率0.0.5より大きくなると学習に失敗 (Nanで怒られる)
- Chainerのみ学習率0.5まで順調に低下
- TensorFlowは、学習率0.1より大きくなると学習に失敗 (発散)

Changes in the learning time due to the activation function

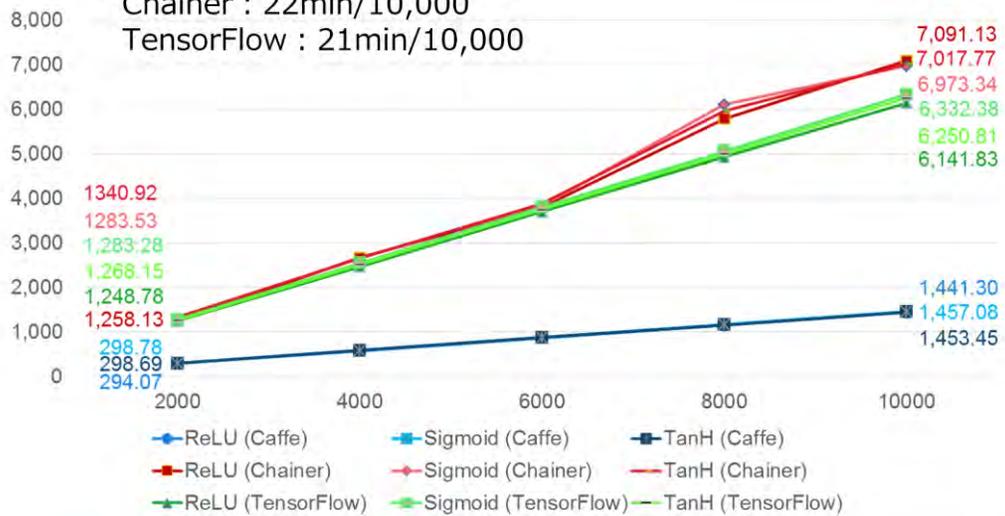
Japan OSS Promotion Forum

■ Learning time is simply proportional to the Learning count

□ Caffe : 5min/10,000

Chainer : 22min/10,000

TensorFlow : 21min/10,000



Copyright 2016 Japan OSS Promotion Forum

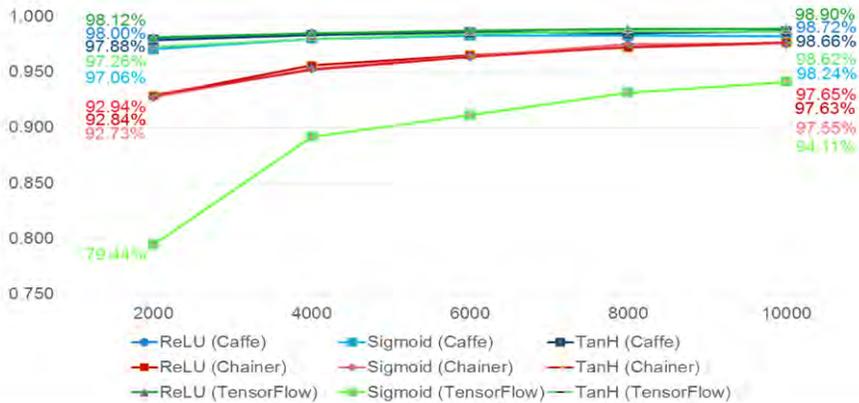
70

<事実のみ>

- **Caffe**は、活性化(非線形)関数を変更しても学習時間はほぼ同じ(学習回数に単純比例)
- **Chainer**は、学習回数が6,000より大きくなると、何らかの原因で学習時間が増加
- **TensorFlow**は、活性化関数を変更しても学習時間はほぼ同じ(学習回数に単純比例)

Changes in the rate of correct answers during the test by function

- A high percentage of correct answers in the order of ReLU function > TanH function > Sigmoid function
 - Caffe, TensorFlow has a higher correct answer rate than the other functions when using ReLU function
 - Chainer is, almost the same percentage of correct answers in all functions



Copyright 2016 Japan OSS Promotion Forum

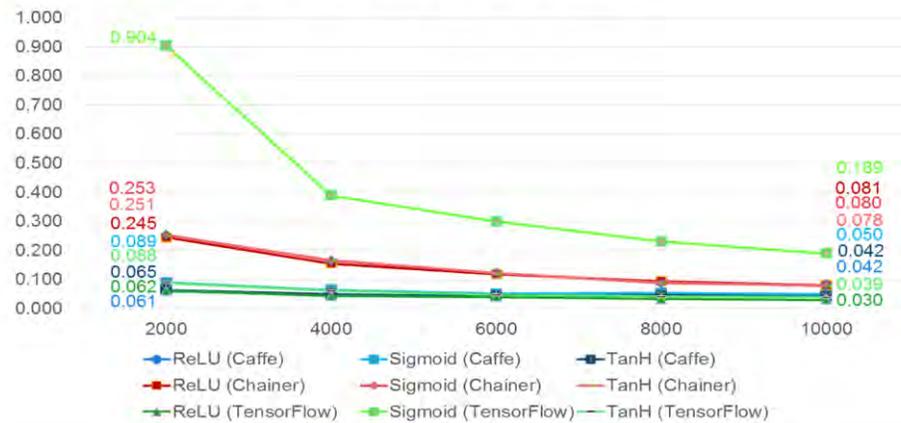
71

<事実のみ>

- Sigmoid (Chainer) は、正答率が常に最も低い
- Sigmoid (TensorFlow) は、正答率が他フレームワークよりも低い
- ※ Sigmoid (Chainer) を除く
- Sigmoid 関数は、全てのフレームワークで最も正答率が低い
- ReLU 関数は、全てのフレームワークで最も正答率が高い
- ※ 一般的に、昔から使われている sigmoid、tanh は勾配消滅問題あり、ReLU は最近流行りで学習が速く勾配消失問題の心配なし

Change of losses due to function

- A small overall loss in the order of ReLU function > TanH function > Sigmoid function
 - Caffe, TensorFlow has a smaller loss than any other function in the case of using the ReLU function
 - Chainer is, almost the same loss in all functions



Copyright 2016 Japan OSS Promotion Forum

72

<事実のみ>

- Sigmoid (Chainer) は、損失が常に最も大きい
- Sigmoid (TensorFlow) は、損失が他フレームワークよりも大きい
- ※ Sigmoid (Chainer) を除く
- Sigmoid 関数は、全てのフレームワークで最も損失が大きい
- ReLU 関数は、全てのフレームワークで最も損失が小さい

- The survey result is the investigation the middle of the material. Since it conducted a further investigation towards the end of this year (2017/3), please stay tuned.

本日、ご紹介した資料は、調査途中の資料です。
2017/3の年度末に向けてさらなる調査を実施しますので、
ご期待ください。

Thank you
감사합니다
谢谢！

ありがとうございました。