オープンソース・カンファレンス2011/Tokyo/Fall

話題のHadoop vs. MySQL

～性能比較だけでなくOSSの適用範囲を考えよう～

日本ヒューレット・パッカード株式会社
ESSNプリセールス統括本部
エンタープライズサーバー・ストレージ技術第1本部
Linuxソリューション部

古賀　政純

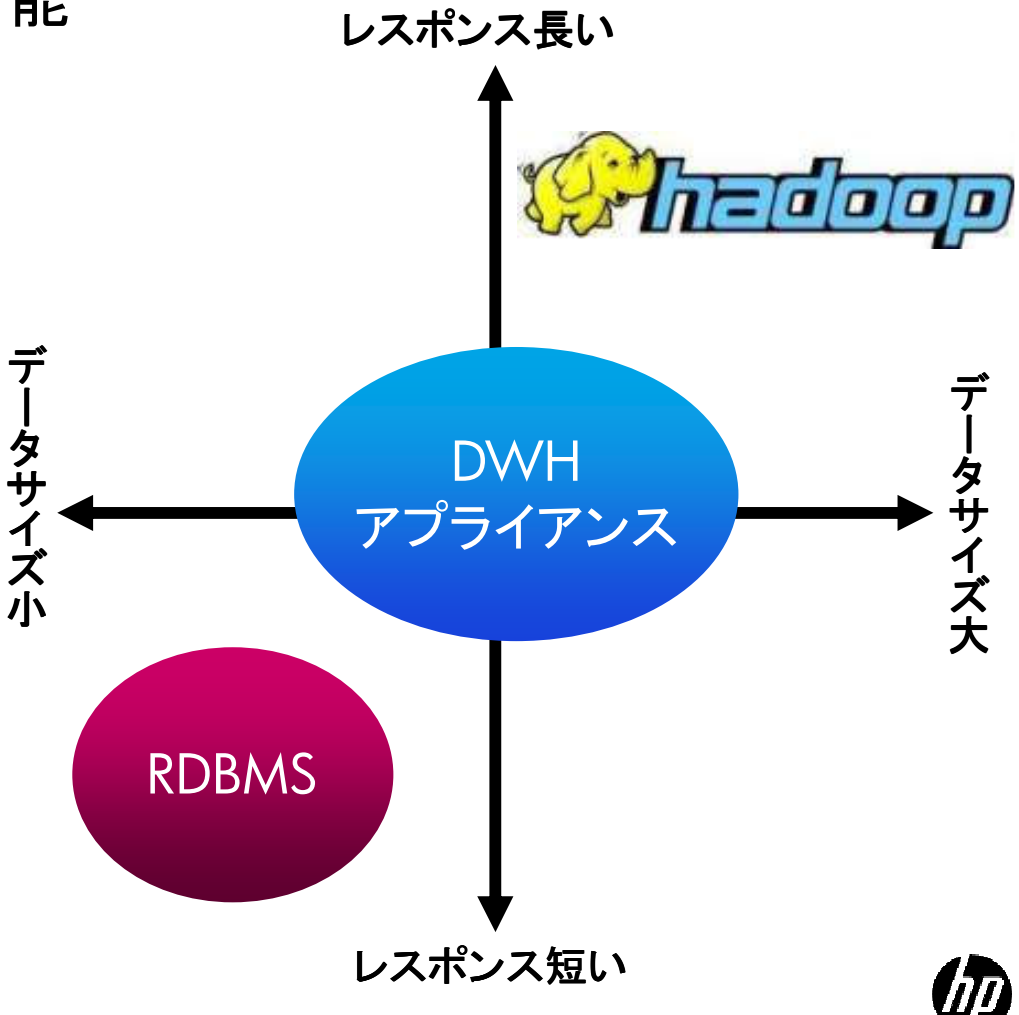
Twitter：　@masazumi_koga

2011年11月19日（土）

Hadoopが必要か？

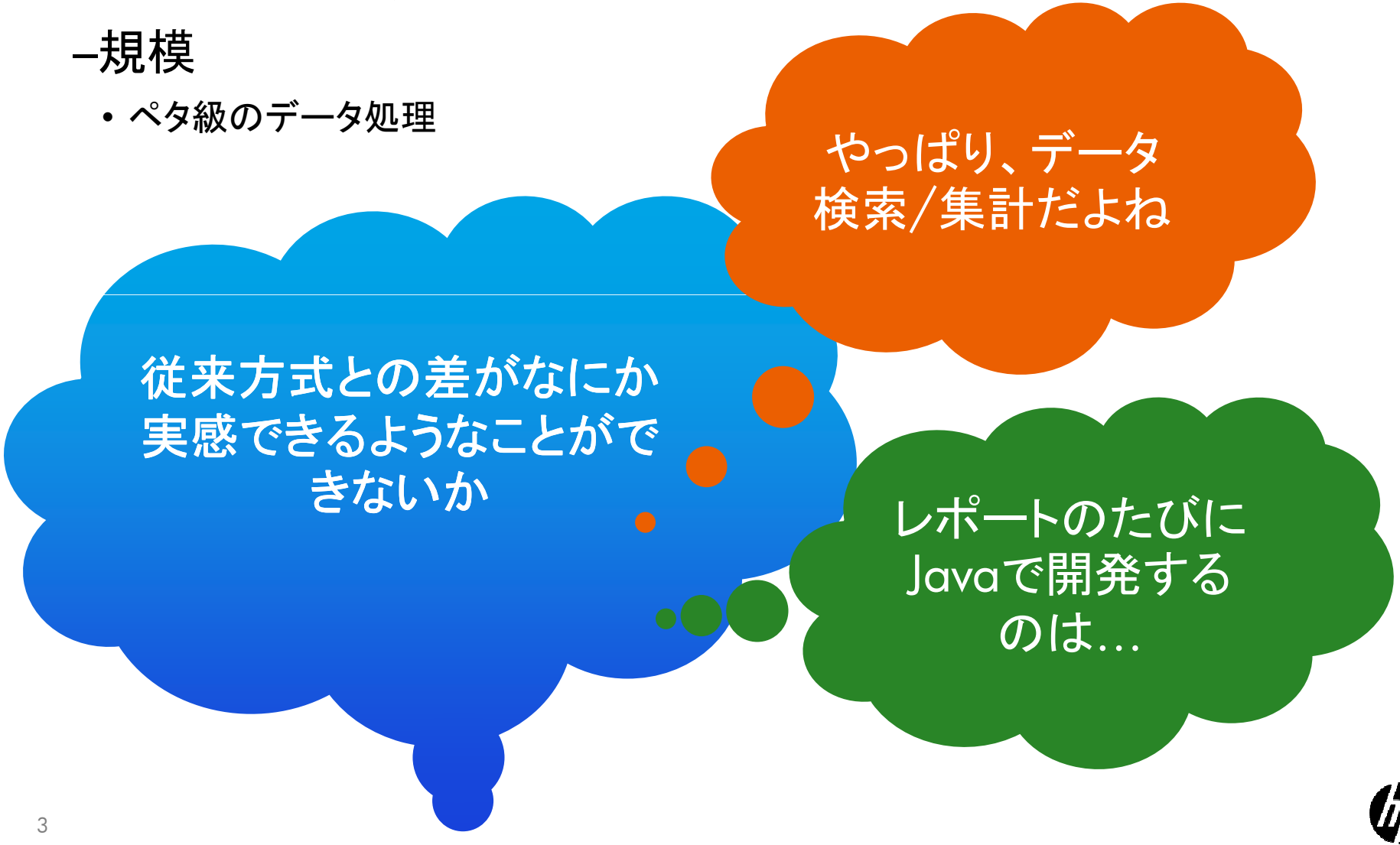
- ペタ級のデータ処理
  - –テラ級ならRDBMSで処理可能

項目	RDBMS	Hadoop
データサイズ	～テラ	ペタ
想定データ操作	小さなデータ参照/更新	大きなデータ挿入/参照
レスポンス	速い	遅い
スケールアップ・アウト	せいぜい1ケタ台のサーバ	数十台～数千台
データ構造	構造化データ	準構造化データ



なにをHadoopに期待するか

- 過去できない/できなかったことの実現
  - –規模
    - ペタ級のデータ処理



データの格納方式

- デーモンプロセスとしては存在しない
- データはHDFS上の特定ディレクトリ
  - –設定ファイル${HIVE_HOME}/conf/hive-site.xmlにて指定
    - 既定：/usr/hive/warehouse
- 上記ディレクトリ下にテーブル名と同一のディレクトリ
  - –複数ファイルに分割される

```
[hadoop@ProLiantSL6500]$ ./bin/hadoop dfs -ls /user/hive
Found 1 items
drwxrwxrwx   - hive supergroup          0 2011-07-28 20:30 /user/hive/warehouse
[hadoop@ProLiantSL6500]$ ./bin/hadoop dfs -ls /user/hive/warehouse
Found 3 items
drwxr-xr-x   - hive supergroup          0 2011-07-28 20:50 /user/hive/warehouse/namelist
drwxr-xr-x   - hive supergroup          0 2011-07-28 20:30 /user/hive/warehouse/output
drwxr-xr-x   - hive supergroup          0 2011-07-28 20:51 /user/hive/warehouse/result
```

DBで取り扱われるデータの例

売上明細

カラム	ラベル	属性
伝票番号	transno	string
日付	datet	string
担当者番号	empno	int
店番号	shopno	int
レジ番号	regno	int
顧客番号	memberno	int
品番	itemno	int
単価	price	int
個数	num	int

販売店

カラム	ラベル	属性
店番号	shopno	int
店名	name	string
県	stat	string
市町村	city	string

会員情報

カラム	ラベル	属性
会員番号	memberno	int
氏名	name	string
電話	stat	string
年齢層	city	int
性別	sex	string

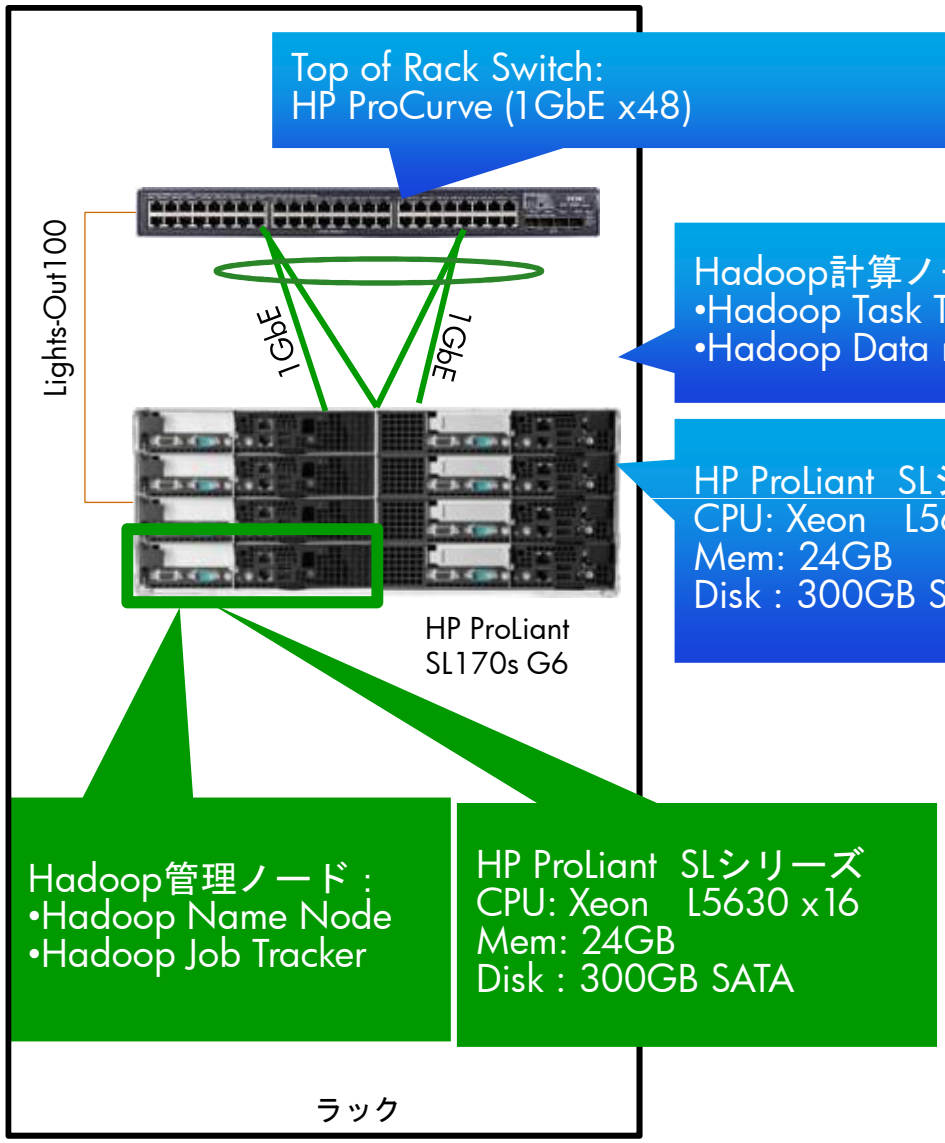
品目情報

カラム	ラベル	属性
品番	itemno	int
品名	name	string
大分類	m	string
小分類	s	string

テストデータの作成方法（ただし小規模）

- フリーソフト　プラデータ（テストデータ作成ソフト）
  - http://effect-work.com/modules/madesoft/index.php?content_id=3

Haoop/MySQLテスト構成



- Proof Of Concept構成
- HadoopのName Nodeは1台構成
- Name NodeはJob Trackerを兼用
- Name Nodeの可用性はなし

- HadoopのData Nodeは7台
- HDFSのレプリカは3
- 容量を確保するため、SATAのディスクを搭載
- Top of Rackスイッチはギガビットのポート数を多数もつものを用意

CPU: Intel(R) Xeon(R) L5630 2.13GHz
(4core x 2socket x HyperThreding = 16 logical core)

HBA: SmartArray P212(Firmware V3.00)
⇒ LUN 300GB(RAID1+0)

NIC: Intel 82576(GbE 2port)

OS: RHEL5.6/x86_64

Hadoop: Cloudera社 CDH3b4

テーブル構造の例

販売店テーブル
47レコード
ファイルサイズ 約2KB
都道府県毎に1支店のデータを作成。

```
$ head -5 shop.csv
1,札幌支店,北海道,札幌市
2,青森支店,青森県,青森市
3,盛岡支店,岩手県,盛岡市
4,仙台支店,宮城県,仙台市
5,秋田支店,秋田県,秋田市
$
```

品目情報テーブル
150レコード
ファイルサイズ 約3KB

```
$ head -5 item.csv
1,品目1,100,L,M,S
2,品目2,100,L,M,S
3,品目3,100,L,M,S
4,品目4,100,L,M,S
5,品目5,100,L,M,S
$
```

会員情報テーブル
1000レコード
ファイルサイズ 約36KB
以下のサイトでランダムな人名・電話番号のデータを生成後に加工。
http://kazina.com/dummy/

```
$ head -5 member.csv
1,富田 夏空,062-812-0278,21,F
2,首藤 照生,083-978-8524,26,M
3,谷村 一徳,089-661-6355,70,M
4,根岸 しほり,085-541-2899,67,F
5,野口 満,044-813-5554,76,M
$
```

売上明細テーブル
5億レコード
ファイルサイズ 約21GB
スクリプトで、3000万、1000万、6000万、5億レコードの4ファイルを作成。

```
$ head -5 data.csv
1,2011-02-28,6,5,10,583,15,200,3
2,2011-03-23,9,14,1,441,109,1100,10
3,2011-06-24,8,35,7,742,21,300,5
4,2010-10-09,2,34,8,337,106,1100,4
5,2011-04-25,9,15,2,774,83,900,7
$
```

支店別売上合計実行結果 (Hiveの場合)

```
hive> set mapred.map.tasks=8;
hive> set mapred.reduce.tasks=35;
hive> select name, sum(num*price) total from gshop gp join gdata
gd on gp.shopno = gd.shopno group by name;
Total MapReduce jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Defaulting to jobconf value of: 35
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapred.reduce.tasks=<number>
Starting Job = job_201109061312_0083, Tracking URL =
http://hd01:50030/jobdetails.jsp?jobid=job_201109061312_0083
Kill Command = /usr/lib/hadoop/bin/hadoop job  -Dmapred.job.tracker=hd01:8021 -kill
job_201109061312_0083
2011-09-07 14:36:19,209 Stage-1 map = 0%,  reduce = 0%
2011-09-07 14:37:21,580 Stage-1 map = 78%,  reduce = 21%
…
…
2011-09-07 14:38:22,013 Stage-1 map = 100%,  reduce = 99%
2011-09-07 14:38:24,031 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_201109061312_0083
Launching Job 2 out of 2
Number of reduce tasks not specified. Defaulting to jobconf value of: 35
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapred.reduce.tasks=<number>
Starting Job = job_201109061312_0084, Tracking URL =
http://hd01:50030/jobdetails.jsp?jobid=job_201109061312_0084
Kill Command = /usr/lib/hadoop/bin/hadoop job  -Dmapred.job.tracker=hd01:8021 -kill
job_201109061312_0084
2011-09-07 14:38:31,364 Stage-2 map = 0%,  reduce = 0%
2…
2011-09-07 14:38:41,415 Stage-2 map = 100%,  reduce = 96%
2011-09-07 14:38:42,421 Stage-2 map = 100%,  reduce = 100%
Ended Job = job_201109061312_0084
OK
大津支店 47384415600
大阪支店 47386920700
岐阜支店 47374985800
長野支店 47394234500
横浜支店 47370704400
神戸支店 47372939600
金沢支店 47411010300
長崎支店 47389582800
大分支店 47413939300
仙台支店 47398701200
和歌山支店 47418373600
青森支店 47397433600
高松支店 47362537800
奈良支店 47443396200
札幌支店 47400462200
松山支店 47391967900
盛岡支店 47384157000
静岡支店 47410298400
千葉支店 47385263900
東京本店 47409422500
福井支店 47395218300
鹿児島支店 47400432700
山形支店 47445603000
津支店 47398738700
富山支店 47438104200
鳥取支店 47366455500
新潟支店 47389652900
甲府支店 47373573600
岡山支店 47390227100
福岡支店 47392878900
京都支店 47404916000
名古屋支店 47375402200
宮崎支店 47383469800
広島支店 47401081800
福島支店 47392961900
水戸支店 47397953200
宇都宮支店 47364306200
山口支店 47382205500
さいたま支店 47370562500
徳島支店 47387009600
松江支店 47391875100
秋田支店 47403853200
佐賀支店 47430434700
高知支店 47380327800
熊本支店 47372752000
前橋支店 47383843200
那覇支店 47375090900
Time taken: 146.796 seconds
hive>
```

支店別売上合計実行結果 (MySQLの場合)

```
mysql> select name, sum(num*price) total from
gshop,gdata where gshop.shopno = gdata.shopno group by
name;
+--------------------+-------------+
| name               | total       |
+--------------------+-------------+
| さいたま支店 | 47370562500 |
| 熊本支店       | 47372752000 |
| 甲府支店       | 47373573600 |
| 盛岡支店       | 47384157000 |
| 神戸支店       | 47372939600 |
| 福岡支店       | 47392878900 |
| 福島支店       | 47392961900 |
| 福井支店       | 47395218300 |
| 秋田支店       | 47403853200 |
| 那覇支店       | 47375090900 |
| 金沢支店       | 47411010300 |
| 長野支店       | 47394234500 |
| 長崎支店       | 47389582800 |
| 青森支店       | 47397433600 |
| 静岡支店       | 47410298400 |
| 高知支店       | 47380327800 |
| 高松支店       | 47362537800 |
| 鳥取支店       | 47366455500 |
| 鹿児島支店     | 47400432700 |
| 前橋支店       | 47383843200 |
| 千葉支店       | 47385263900 |
| 名古屋支店     | 47375402200 |
| 和歌山支店     | 47418373600 |
| 大阪支店       | 47386920700 |
| 大分支店       | 47413939300 |
| 大津支店       | 47384415600 |
| 奈良支店       | 47443396200 |
| 宇都宮支店     | 47364306200 |
| 宮崎支店       | 47383469800 |
| 富山支店       | 47438104200 |
| 山口支店       | 47382205500 |
| 山形支店       | 47445603000 |
| 岐阜支店       | 47374985800 |
| 岡山支店       | 47390227100 |
| 広島支店       | 47401081800 |
| 徳島支店       | 47387009600 |
| 新潟支店       | 47389652900 |
| 札幌支店       | 47400462200 |
| 東京本店       | 47409422500 |
| 松山支店       | 47391967900 |
| 松江支店       | 47391875100 |
| 横浜支店       | 47370704400 |
| 水戸支店       | 47397953200 |
| 津支店         | 47398738700 |
| 京都支店       | 47404916000 |
| 仙台支店       | 47398701200 |
| 佐賀支店       | 47430434700 |
+--------------------+-------------+
47 rows in set (36 min 31.05 sec)

mysql>
```

身近なシステムにおける適用は？

- かなりの規模でないと導入効果がない
  - –RDBMSで処理したほうがよいケース
  - –既存資産に手を入れるのが厳しいケース
- ビジネス・インテリジェンスであれば
  - –SQL Serverアプライアンス
  - –Vertica

Hadoop/OSS Integration
2011年下半期　海外におけるHadoop/OSS最新情報

Hadoop Worker nodes

•SL160s/SL165 **Best Balance**

- Up to 6 LFF HD / 8 SFF HD
- 18/24 DIMM slots with 384GB Max Memory
- 1U

•DL180 **Storage Capacity per Node**

- Up to 14 LFF HD / 25 SFF HD
- 12 DIMM sockets with 192GB Max Memory
- 2U

•SL335 **Storage Capacity per U**

- Up to 4 LFF HD / 8 SFF HD
- 12 DIMM slots with 192GB Max Memory
- Lisbon CPU Dual Socket Hexa Core
- 2 servers per 1U

海外OSS最新情報

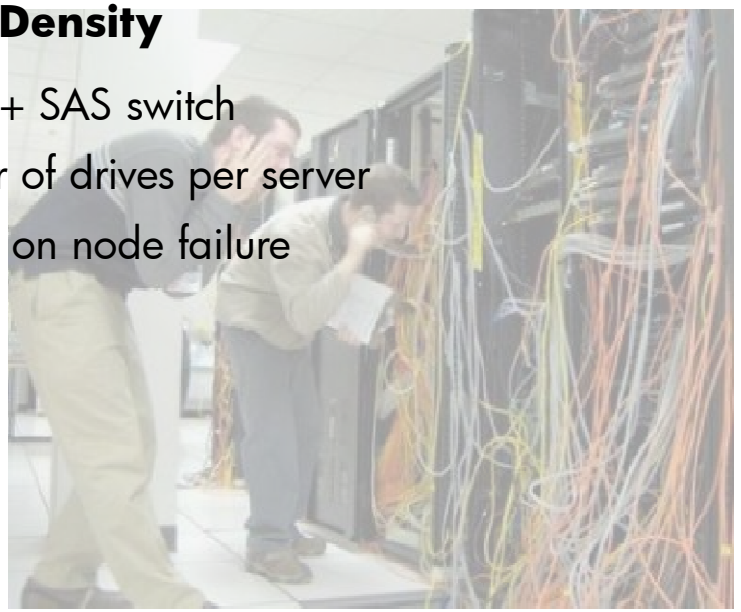
Alternative Hadoop Platforms

Hot-Swap w/ Density

- DL2000 / DL170e
  - 2 or 4 nodes per 2U chassis
  - Split up to 12 LFF HD / 24 SFF HD between nodes

Flexibility, High Density

- BL460 + MDS600 + SAS switch
  - Flexible number of drives per server
  - Reassign drives on node failure



Hadoop at Macy's

RDBMS & Hadoop Comparison*

	Traditional RDBMS (Oracle, DB2)	Hadoop
Maximum Data Capacity	Up to 100's of TBs	Up to 10's of PBs (hundreds times more)
Processing Capacity	Up to 10's of TBs	Up to 10's of PBs (thousands times more)
Costs	High software, license and hardware/storage costs	Cost effective: commodity hardware + open source software
Transactional	Yes	No (batch process)
Update Patterns	Supported	Not Supported Yet
Schema Complexity	Structured (tables only)	Structured or Unstructured
Processing Freedom	SQL	MapReduce, SQL (Hive), Streaming, Pig, HBase, etc..
Scalability	Non-linear scaling	Fully distributed and linearly scalable
Reliability	Fault-tolerant at high cost, but without self-healing by design	Fault-tolerant and self-healing by desing
Real Time Response	Yes	No (HBase required)

★macys

* Cloudera comparison chart

出展：http://www.basas.com/newsletters/20110705/Presentation/Kerem%20Tomak.ppt

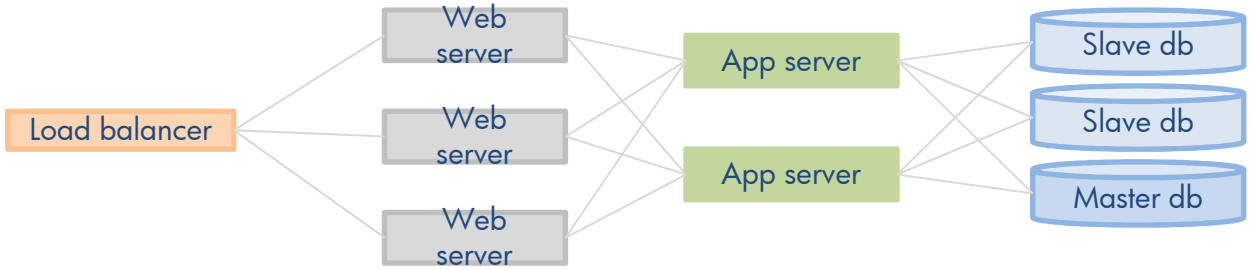
OSS DBの高速化手法 memcached

世間で知られるMemcachedの利用事例

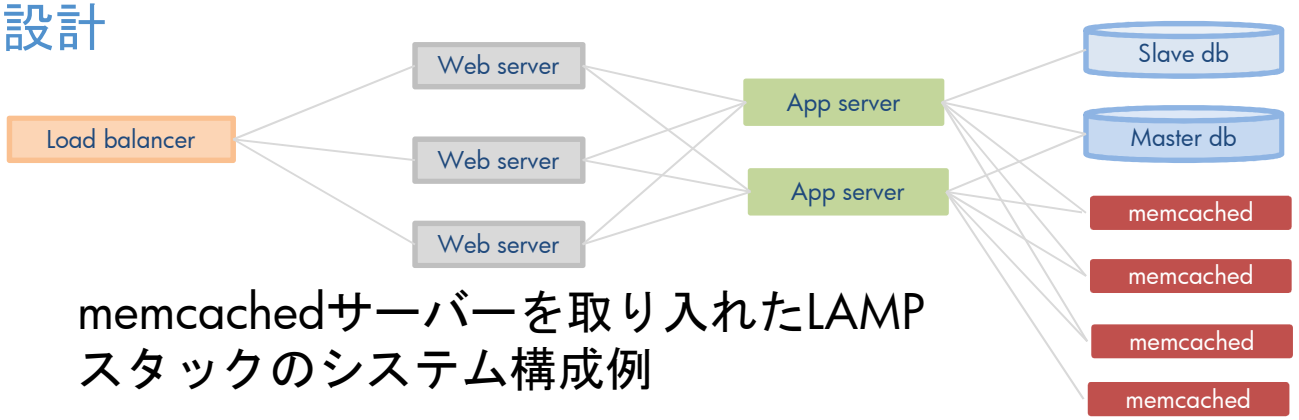
- Webサーバーの応答性能を向上を狙うのが目的
- MySQLやPostgreSQLデータベースのボトルネックの解消に利用される
- Wikipedia、facebook、Myspace、Livejournal、Digg、SlashdotなどのWebサイトで採用

Memcached：データベースの結果をキャッシュ

- 課題: Webサイトのトラフィックをどのように低減するか?
  - Webサーバーのトラフィックは Webサーバーを増やすことで対応可能
  - しかし、SQLデータベースはボトルネックになりやすい



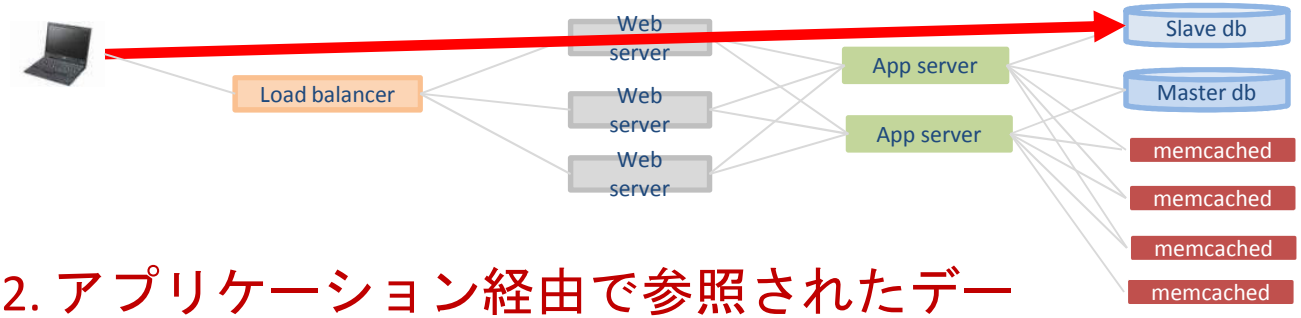
- 解決策: **Brad Fitzpatrick氏**（LiveJournal.com所属）がmemcachedを開発
  - Read系のDBシステムでのSQLサーバーのボトルネックを解消
  - Memcachedサーバーを並べることで、ボトルネックを大幅に低減する設計



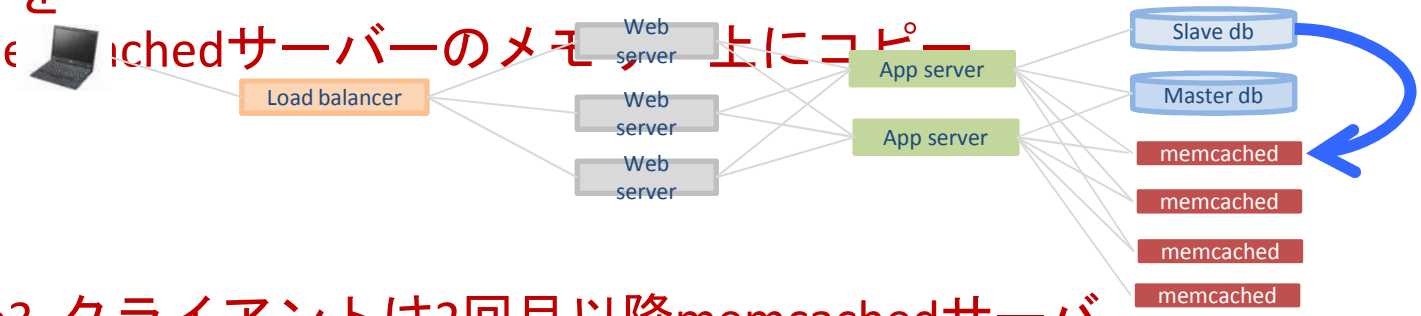
memcachedサーバーを取り入れたLAMPスタックのシステム構成例

Memcachedの通信

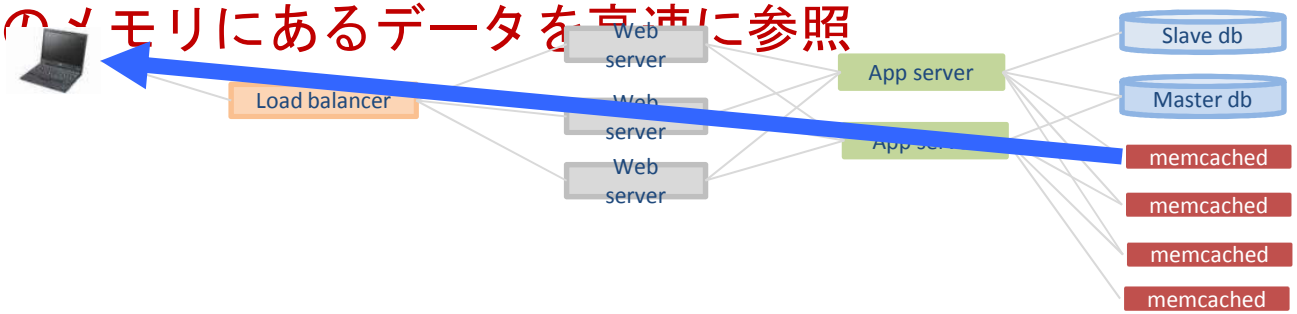
Step1. クライアントがDBを参照（1回目）



Step2. アプリケーション経由で参照されたデータを
memcachedサーバーのメモリ上にコピー

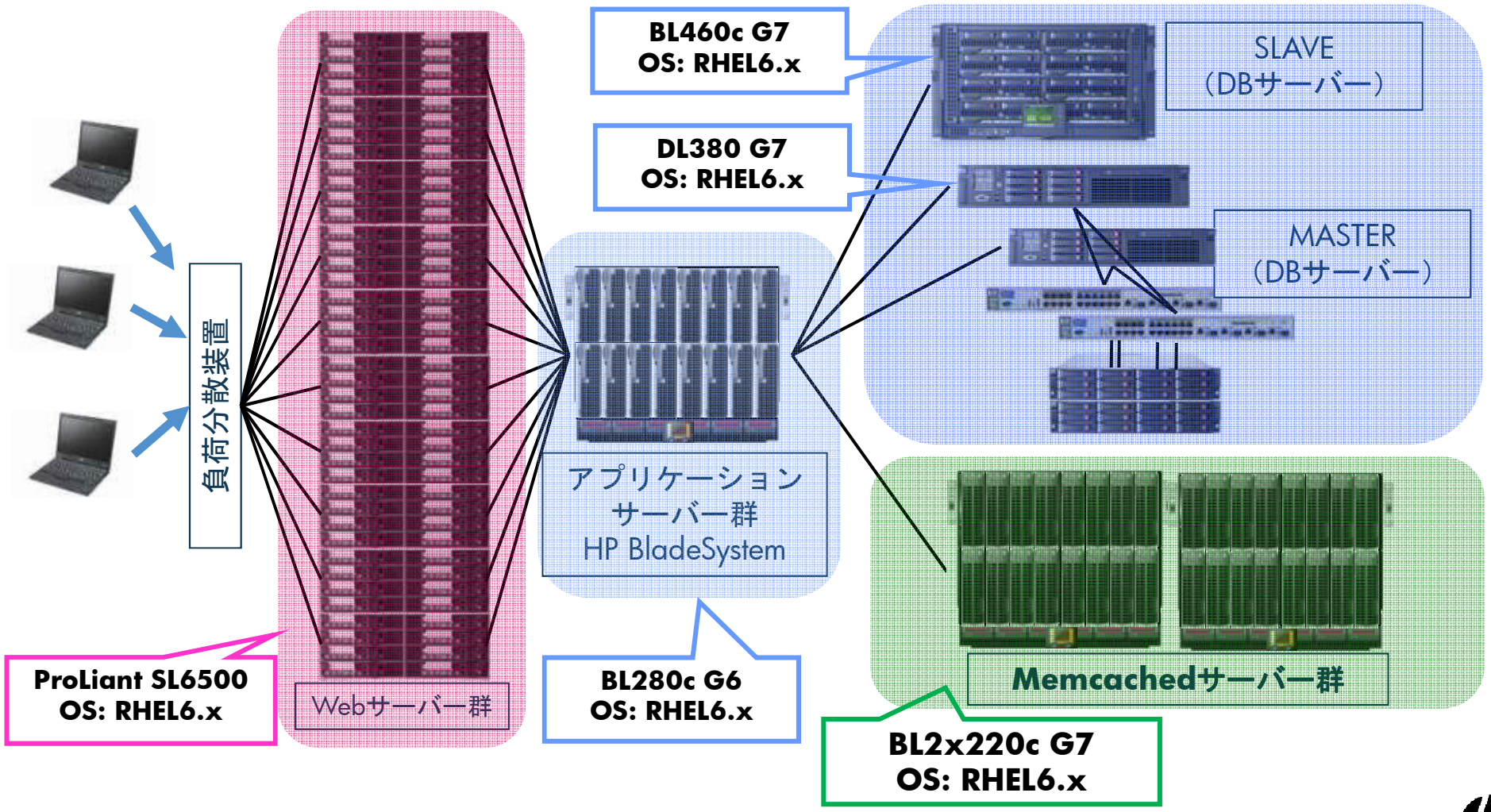


Step3. クライアントは2回目以降memcachedサーバー
上のメモリにあるデータを高速に参照



Memcachedサーバーはスケールアウト型

- リード系DBシステムで威力を発揮
- Memcachedサーバーの可用性はない
- DBが更新されたら、Memcachedサーバーも更新する作りこみ等



まとめ

- 今話題のHadoop
  - 大手SNS、検索サイトで採用
  - スケールアウト型
  - ログ解析を超高速処理
- Web応答高速化技術
  - リード系DBシステムに適
  - スケールアウト型の Memcachedサーバーの導入
  - アプリケーションサーバーでの対応
- すべてをOSSで置き換えるわけではない
- 適材適所をよく考えよう

Thank you

Appendix.

HPにおける Hadoop/OSSの取り組み 技術文書のダウンロード

Intel社/Cloudera社共同Hadoopクラスターのホワイトペーパーは HP ProLiantを採用

WHITE PAPER
Cloud Computing

Optimizing Hadoop* Deployments

Designing the solution stack to maximize productivity while limiting energy consumption and total cost of ownership

Tuning Hadoop* clusters is vital to improve cluster performance, optimize resource utilization, and minimize operating costs. Tests conducted in Intel labs have established a number of best practices to help meet those goals.

EXECUTIVE SUMMARY

This paper provides guidance, based on extensive lab testing conducted with Hadoop* at Intel, to organizations as they make key choices in the planning stages of Hadoop deployments. It begins with best practices for establishing server hardware specifications, helping architects choose optimal combinations of components. Next, it discusses the server software environment, including choosing the OS and version of Hadoop. Finally, it introduces some configuration and tuning advice that can help improve results in Hadoop environments.

1 Overview

Having moved beyond its origins in search and Web indexing, Hadoop is becoming increasingly attractive as a framework for large-scale, data-intensive applications. Because Hadoop deployments can have very large infrastructure requirements, hardware and software choices made at design time can have a significant impact on performance and TCO.

Intel is a major contributor to open source initiatives, such as Linux*, Apache*, and Xen*, and has also devoted resources to Hadoop analysis, testing, and performance characterizations, both internally and with fellow travelers such as HP and Cloudera. Through these technical efforts, Intel has observed many practical trade-offs in hardware, software, and system settings that have real-world impacts.

This paper discusses some of those optimizations, which fall into three general categories:

- **Server hardware.** This set of recommendations focuses on choosing the appropriate hardware components for an optimal balance between performance and both initial and recurring costs.
- **System software.** In addition to the choice of OS and Java* Virtual Machine (JVM), the specific version of Hadoop and other software components have implications for performance, stability, and other factors.
- **Configuration and tuning.** The settings made to the Hadoop environment itself are an important factor in getting the full benefit of the rest of the hardware and software solution stacks.

It is important to note that Hadoop deployments will vary considerably from customer to customer and from project to project. The suggestions for optimization in this paper are meant to be widely relevant to Hadoop, but results may be quite different depending on actual workloads.

October 2010
Version 2.0

- HP ProLiant SL6000シリーズを採用
- HP ProLiant SL170z G6を使ってテスト
- Cloudera版Hadoop 0.20.2

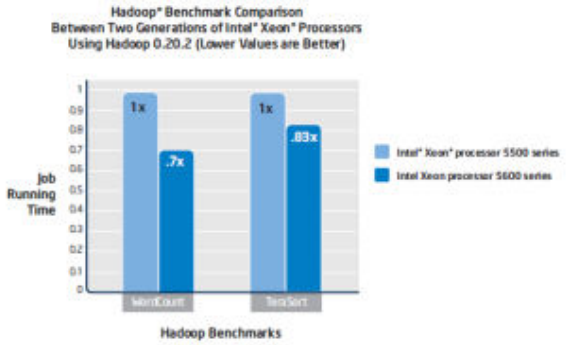


Figure 1. Successive generations of the Intel® Xeon® processor improve performance across a range of Hadoop* benchmarks.[2]

HP ProLiant SL170z G6

http://communities.intel.com/docs/DOC-5645

日本HP Linux技術情報サイト

Hadoop/Nagios/Ganglia/memcached/Ubuntu/FreeBSD
技術文書公開

http://h50146.www5.hp.com/products/software/oe/linux/mainstream/support/lcc/

Ganglia

Nagios®