Cassandra

송무찬

mcsong@gmail.com

Agenda

- 1.Definition
- 2.History & Reference
- 3.Features
- 4.Data Model
- 5.Project Package
- 6.Performance
- ~~7.적용가능 부분~~
- 8.Q&A

1.Definition(1/2)

- Cassandra is a highly scalable, eventually consistent, distributed, structured key-value store.

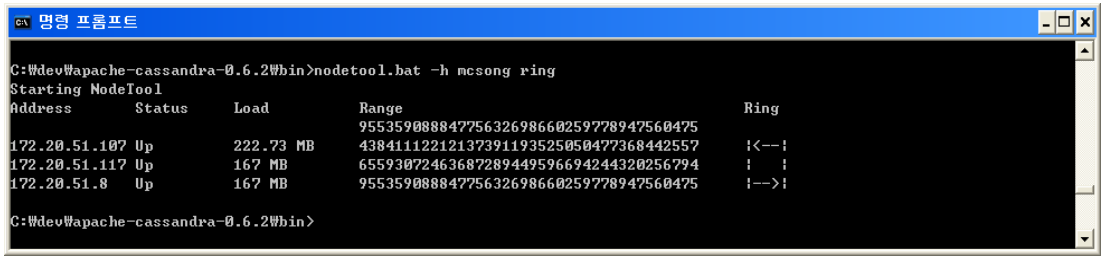
- Cassandra는 FaceBook(http://www.facebook.com), Twitter(http://www.twitter.com)등에서 사용하고 있는, 대용량 데이터 처리를 위한 분산 저장 시스템.

- Cassandra의 분산 시스템과 관련된 기술은 Amazon의 Dynamo에서, 데이터 모델은 Google's BigTable의 모델 (Apache Hadoop's hBase)을 이용해서 구현이 되어 있습니다.

1.Definition(2/2)

- 1.1 Amazon Dynamo
  - key, value 기반의 저장 모델 제공
  - distributed hash table의 형태로 key partitioning을 통한 데이터 분배 균일
  - 데이터 복제 제공(async)
  - eventual consistency 방식
  - ring topology
  - p2p 방식

```
<Seeds>
        <!--Seed>127.0.0.1</Seed-->
        <Seed>172.20.51.117</Seed>
        <Seed>172.20.51.107</Seed>
</Seeds>
```



- 1.2 Google's BigTable

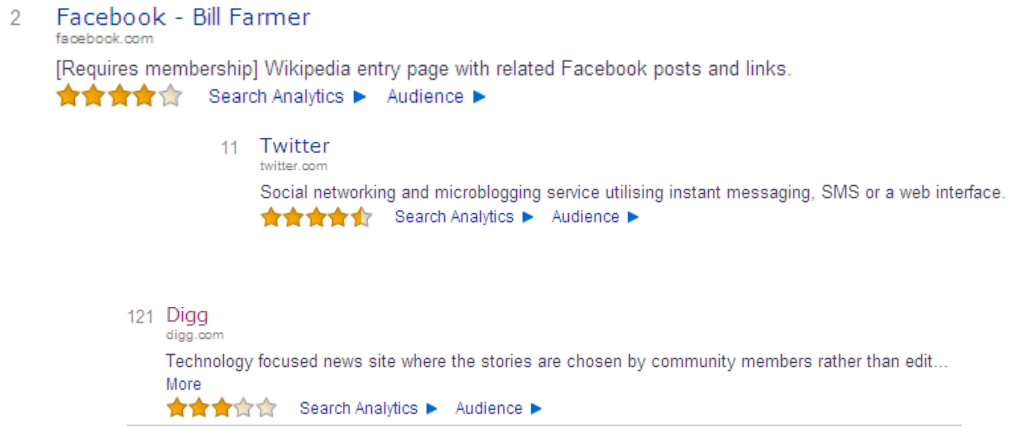
2.History & Reference(1/2)

- 2.1 History
  - 2008.07 Cassandra was released as open source project on Google code.
  - 2009.03 Cassandra started apache's incubation project.
  - 2010.02.17 Cassandra graduated to a top-level project.

- 2.2 Current Status
  - 9명의 Committer(FaceBook 2, Rackspace 2, IBM Research 1, Digg 1, Riptano 1) 보유
  - 그 외로 179명의 Contributor 보유
  - Java 1.6 Version(Sun(http://java.sun.com), OpenJDK(http://www.openjdk.org/))에서 테스트 됨.
  - client library로 Ruby, Perl, Python, Scala, Java, PHP, C#등을 지원하고 있습니다.

2.History & Reference(2/2)

- 2.3 Reference
  - Alexa's(http://www.alexa.com) ranking

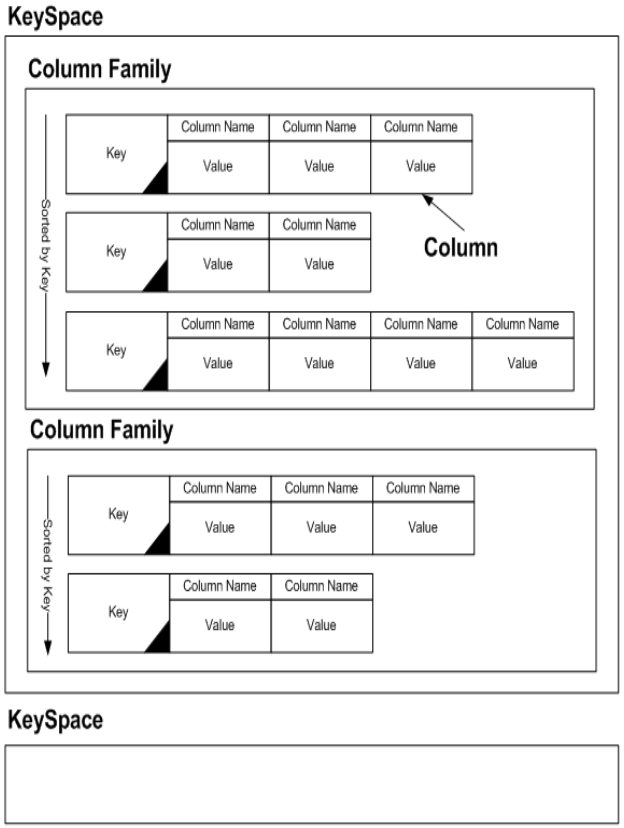


3.Features

- Proven
- Decentralized
- Fault Tolerant
- You're in Control
- Rich Data Model
- Elastic
- Durable
- Professionally Supported

4.DataModel(1/2)

- Cassandra의 Data Model은 Google BigTable의 key, column, column family의 구조를 가지고 있습니다.
  - Column & Super Column
  - Column Family & Super Column Family
  - KeySpace
    - 논리적으로 ColumnFamily를 묶어주는 개념

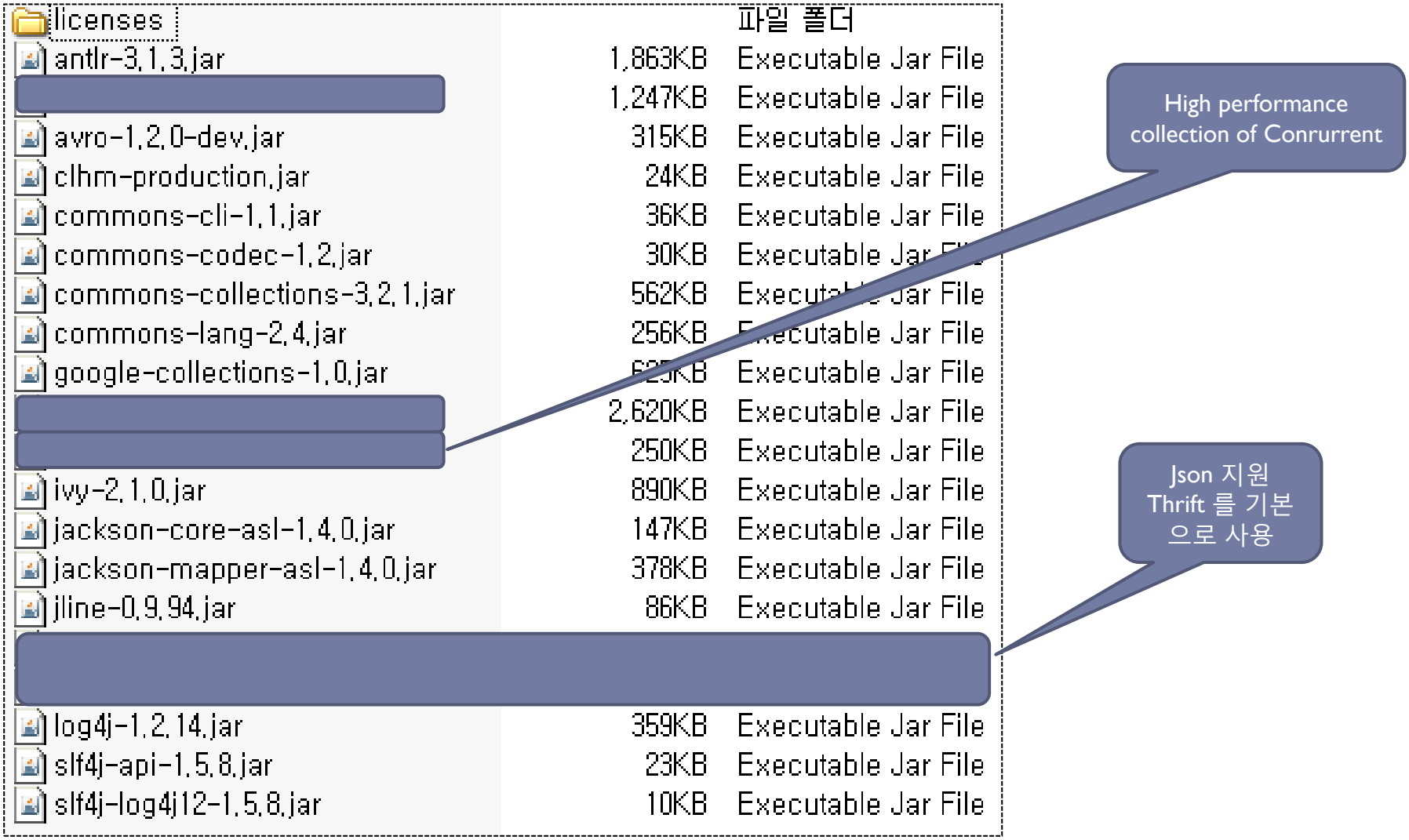


4.DataModel(2/2)

- Ex) select query

Data Model	Query Statement
Relational	SELECT `column` FROM `database`.`table` WHERE `id` = key;
BigTable	table.get(key, "column_family:column")
Cassandra: standard model	keyspace.get("column_family", key, "column")
Cassandra: super column model	keyspace.get("column_family", key, "super_column", "column")

5.Project Package



6.Performance(1/9)

- 6.1 Official Cassandra Website's Slide

Performance vs MySQL w/ 50GB

MySQL
300ms write
350ms read

Cassandra
0.12ms write
15ms read

6.Performance(2/9)

- 6.2 내부 테스트
  - 장비 및 시스템
    - Windows XP SP3, Dual Core 2.53G, 2G RAM의 Machine에서 Java는 1.6.0_20 Version, DB는 MSSQL 2008(SP1), Cassandra(0.6.2) 비교
    - DB는 공통적인 ODBC(Version 3), JDBC(Type 4, sqljdbc4.jar, jtds-1.2.5.jar)를 이용하였습니다.
    - Cassandra는 공식적으로 지원하는 thrift library를 사용하고 있습니다.
  - 포맷 및 데이터
    - 데이터 포맷 : GUID(long, PK), ID(char(20)), NAME(char(20)), Article(nvarchar(MAX))
    - 테스트 데이터 : 1~증가값, 아이디 , 이름 , 기록에 대한 내용입니다

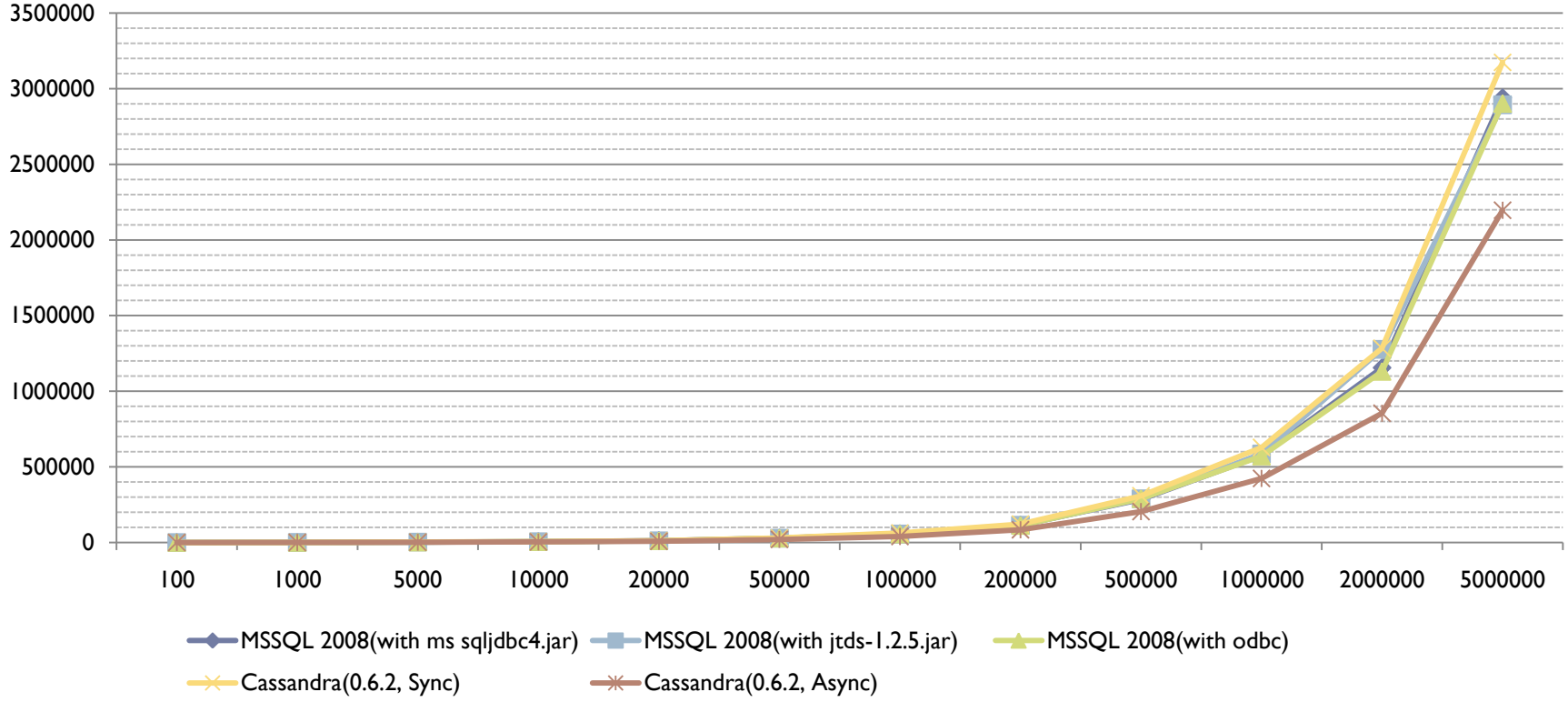
6.Performance(3/9)

- INSERT 결과 테이블(MSSQL 1.2 G, Cassandra 2.8G)

	MSSQL 2008(sqljdbc4.jar)	MSSQL 2008(with jtds-1.2.5.jar)	MSSQL 2008(with odbc)	Cassandra(0.6.2, Sync)	Cassandra(0.6.2, Async)
100	0.094 초	0.078 초	0.046 초	0.125 초	0.094 초
1,000	0.610 초	0.641 초	0.563 초	0.703 초	0.469 초
5,000	2.672 초	3.000 초	2.797 초	3.078 초	2.000 초
10,000	5.734 초	6.000 초	5.719 초	6.078 초	4.000 초
20,000	12.250 초	11.813 초	11.344 초	12.281 초	7.844 초
50,000	29.094 초	28.687 초	28.454 초	30.047 초	20.218 초
100,000	57.938 초	56.797 초	56.316 초	62.750 초	40.281 초
200,000	1분 55.422 초	1분 54.344 초	1분 52.331 초	2분 2.219 초	1분 24.313 초
500,000	4분 44.610 초	4분 49.156 초	4분 50.125 초	5분 8.468 초	3분 24.656 초
1,000,000	9분 35.469 초	9분 45.937 초	9분 30.875 초	10분 28.719 초	7분 3.813 초
2,000,000	19분 25.390 초	21분 2.875 초	18분 8.906 초	21분 4.094 초	14분 14.531 초
5,000,000	49분 0.188 초	48분 2.531 초	48분 3.336 초	52분 8.891 초	36분 6.250 초

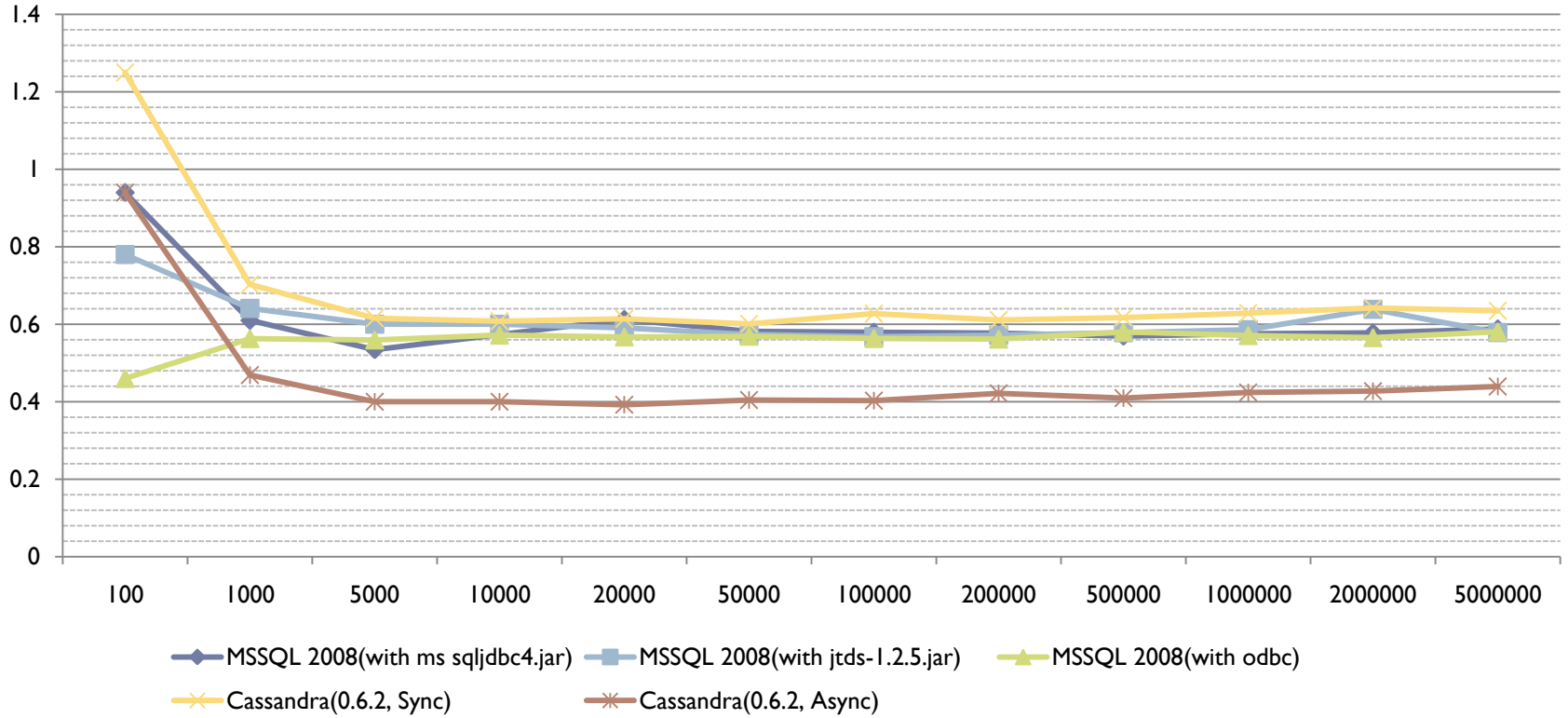
6.Performance(4/9)

- INSERT 결과 그래프(total time, milisecond)



6.Performance(5/9)

- INSERT 결과 그래프(average time, milisecond)



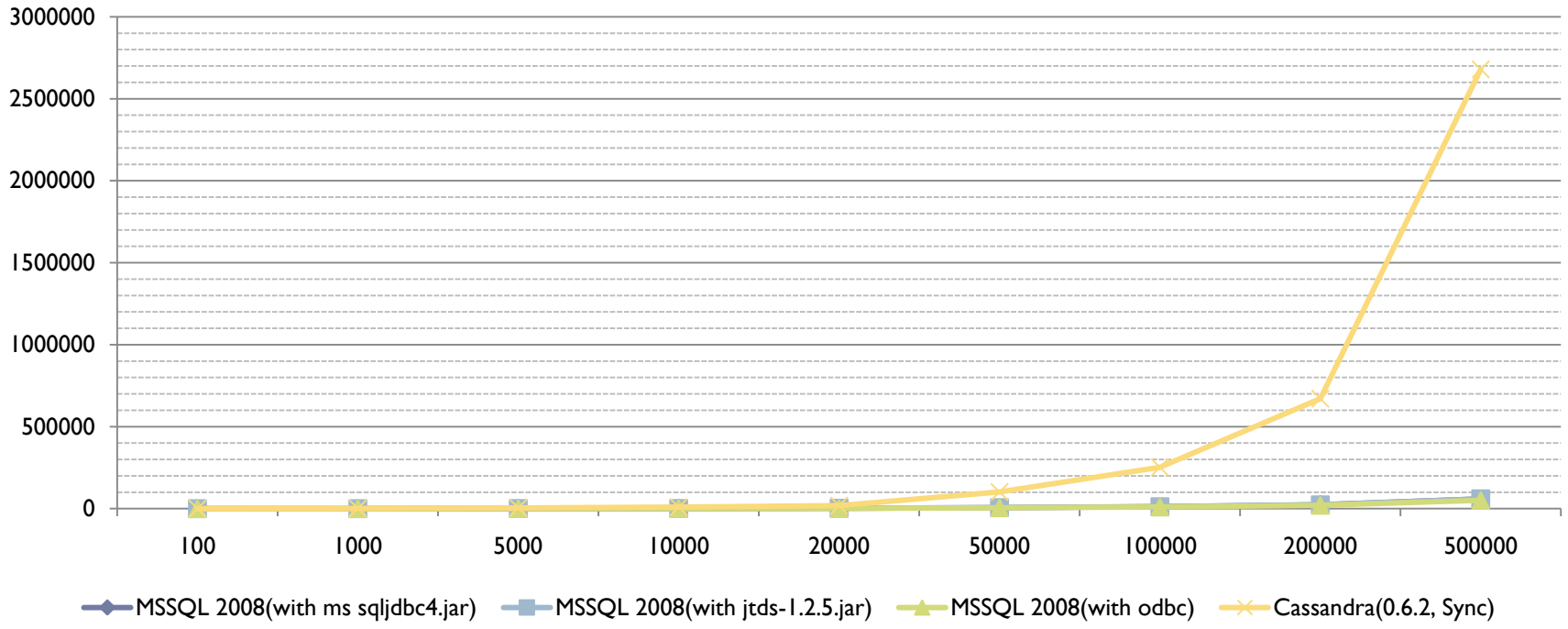
6.Performance(6/9)

- SELECT 결과 테이블

	MSSQL 2008(with ms sqljdbc4.jar)	MSSQL 2008(with jtds-1.2.5.jar)	MSSQL 2008(with odbc)	Cassandra(0.6.2, Sync)
100	0.042 초	0.047 초	0.016 초	0.172 초
1,000	0.187 초	0.188 초	0.141 초	1.000 초
5,000	0.750 초	0.719 초	0.547 초	4.750 초
10,000	1.297 초	1.281 초	1.062 초	9.469 초
20,000	2.438 초	2.469 초	2.109 초	18.875 초
50,000	5.953 초	5.969 초	5.157 초	1분 41.657 초
100,000	12.203 초	11.891 초	10.297 초	4분 12.343 초
200,000	24.156 초	23.609 초	20.563 초	11분 10.687 초
500,000	59.250 초	59.782 초	51.141 초	44분 6.578 초
1,000,000	1분 58.828 초	1분 58.922 초	1분 43.750 초	
2,000,000	3분 58.938 초	3분 56.469 초	3분 31.546 초	
5,000,000	10분 21.312초	10분 32.547 초	8분 46.985 초	

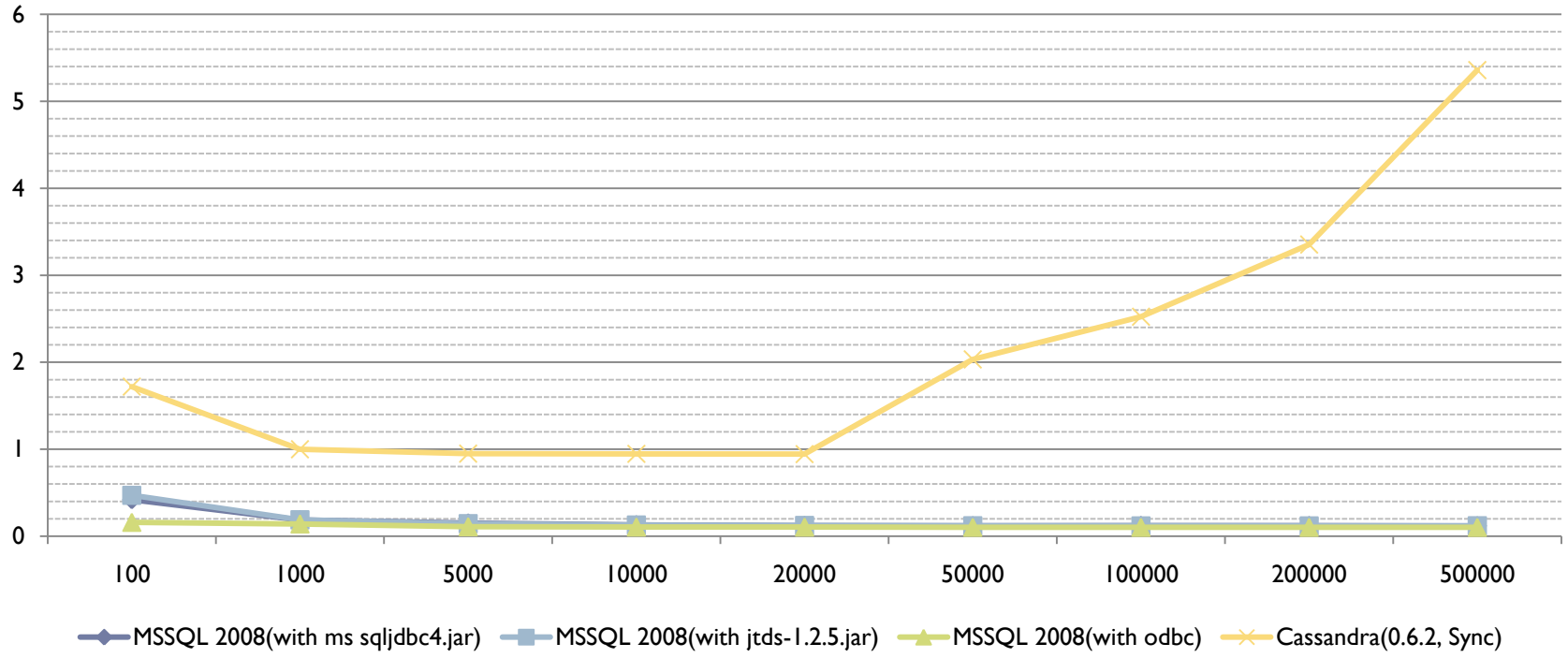
6.Performance(7/9)

- SELECT 결과 그래프(total time, milisecond)



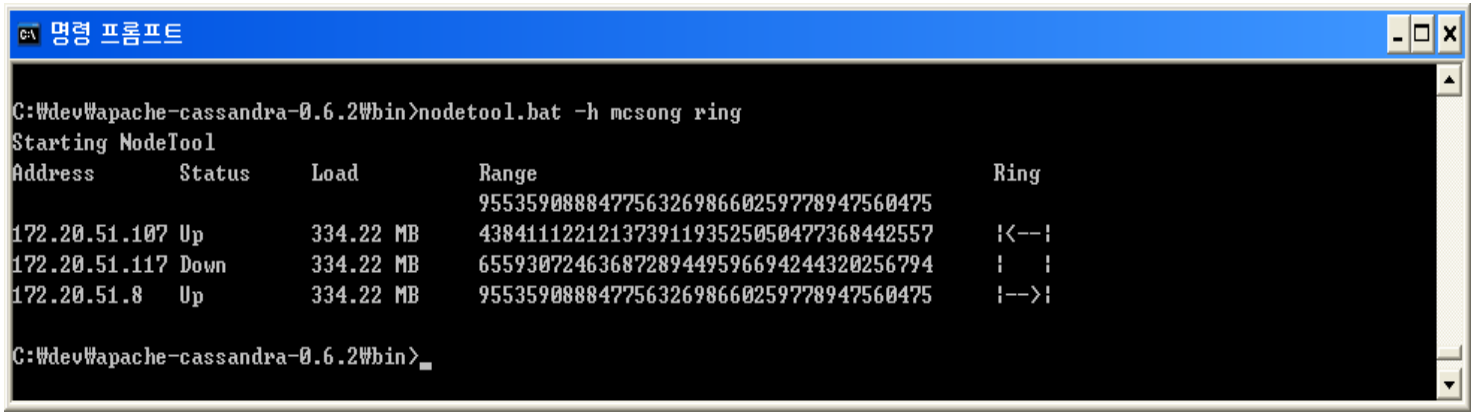
6.Performance(8/9)

- SELECT 결과 그래프(average time, milisecond)



6.Performance(9/9)

- Fail-Over 테스트
  - INFO 15:33:03,351 error writing to sohleeop/172.20.51.117
  - INFO 15:33:10,368 InetAddress /172.20.51.117 is now dead.
  - INFO 15:34:07,090 Node /172.20.51.117 has restarted, now UP again ← Detected
  - INFO 15:34:07,090 Node /172.20.51.117 state jump to normal



8.Q&A

Q&A