Using distributed technologies to analyze Big Data

Abhijit Sharma

Innovation Lab
BMC Software

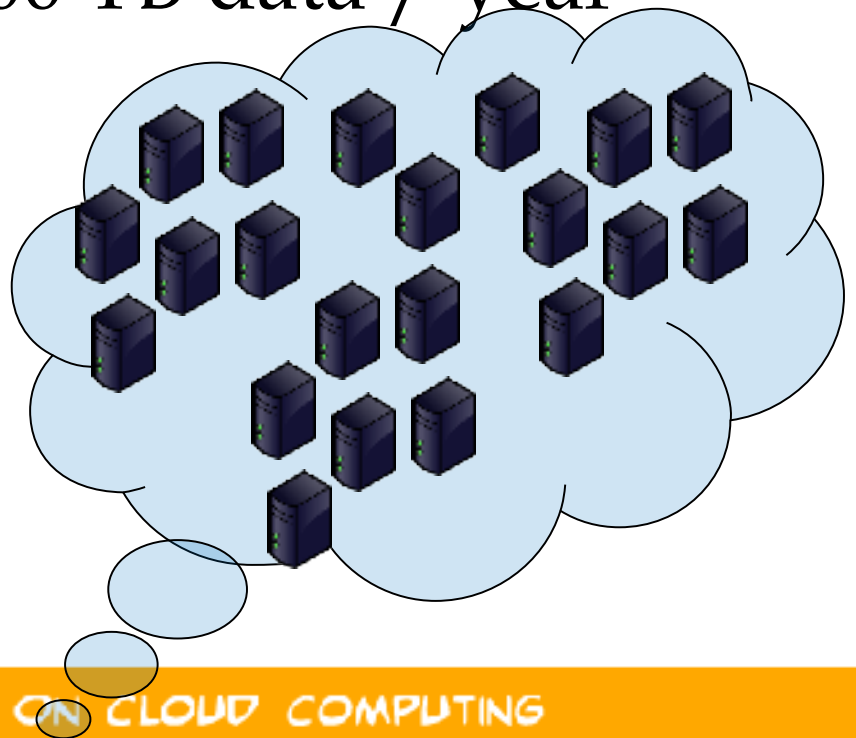
2nd IndicThreads.com Conference On Cloud Computing

3,4 JUNE 2011

PUNE, INDIA

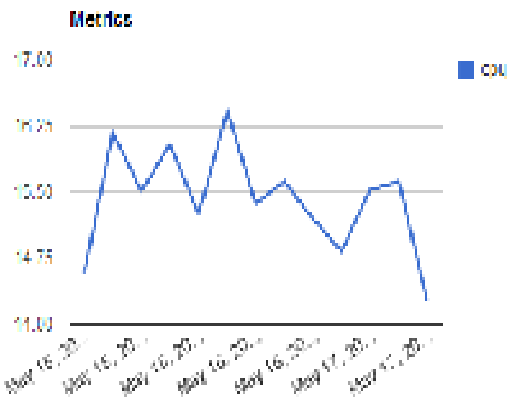
Data Explosion in Data Center

- Performance / Time Series Data
  - Incoming data rates ~Millions of data points/ min
  - Data generated/server/year ~ 2 GB
  - 50 K servers ~ 100 TB data / year



Online Warehouse - Time Series

- Extreme storage requirements – TS data for a data center e.g. last year
- Online TS data availability i.e. no separate ETL
- Support for common analytics operations
  - Roll-up data e.g. CPU/min to CPU/hour, CPU/day etc
  - Slice and Dice – CPU util. for UNIX servers in SFO data center last week
  - Statistical Operations : sum, count, avg., var, std. moving avg., frequency distributions, forecasting etc
- Ease of use – SQL interface, design schema for TS data
- Horizontal scaling - lower cost commodity hardware
- High R/W v

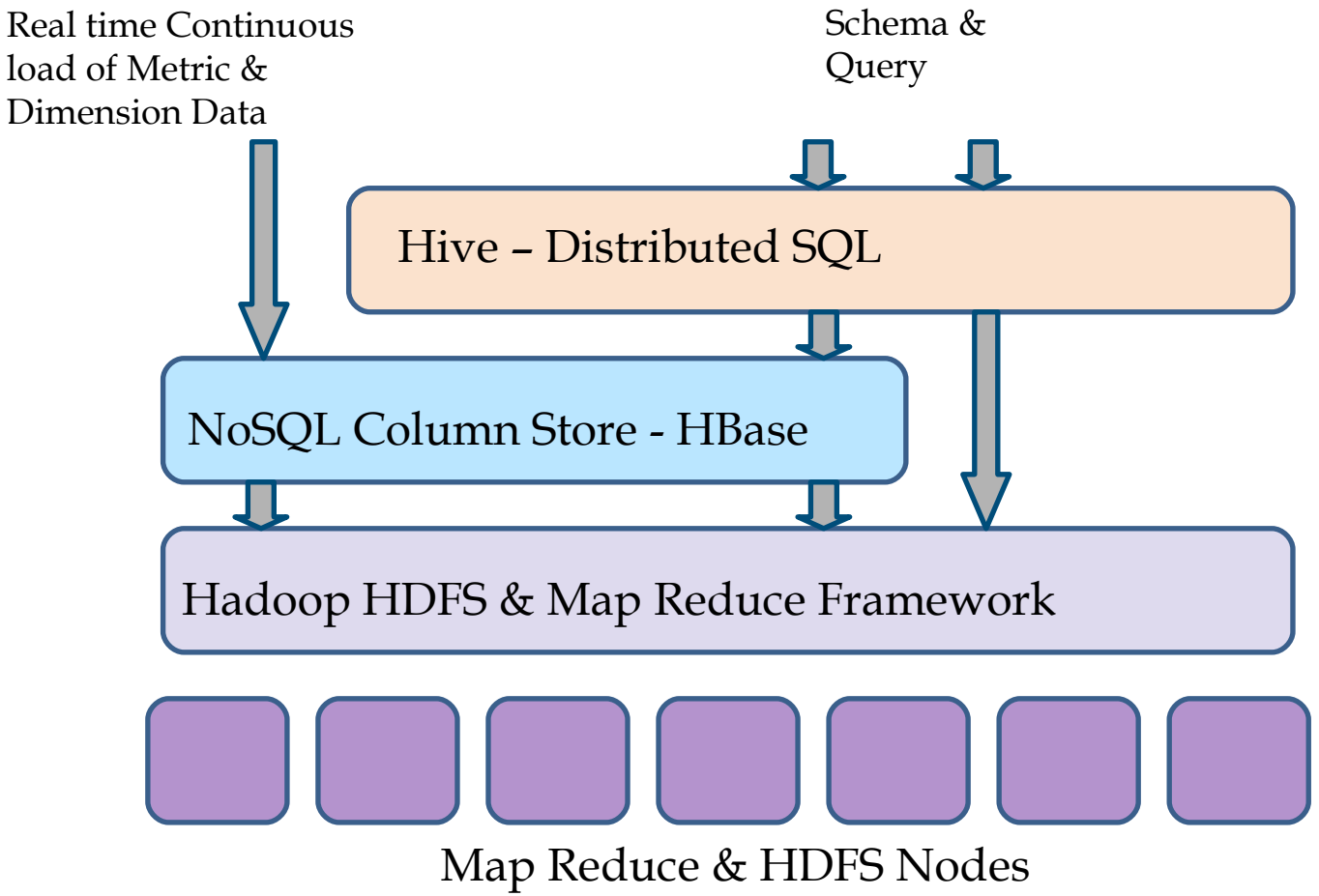


Why not use RDBMS based Data Warehousing?

Star schema – dimensions & facts

- Offline data availability – ETL required – not online
- Expensive to scale vertically – High end Hardware & Software
- Limits to vertical scaling – big data may not fit
- Features like transactions etc are unnecessary and a overhead for certain applications
- Large scale distributed/partitioning is painful – sub optimal on high W/R ratios
- Flexible Schema support which can be changed on the fly is not possible

High Level Architecture



Map Reduce - Recap

Map Function

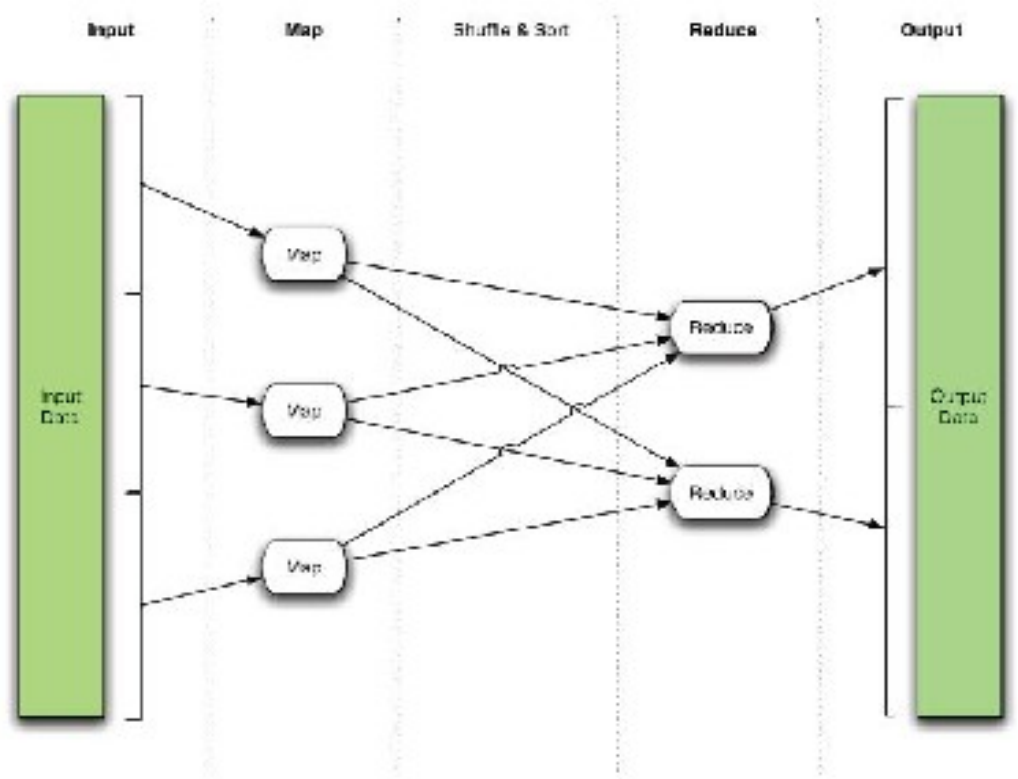
- Apply to input data, Emits reduction key and value
- Output of Map is sorted and partitioned for use by Reducers

Reduce Function

- Apply to data grouped by reduction key
- Often 'reduces' data (for example – sum(values))

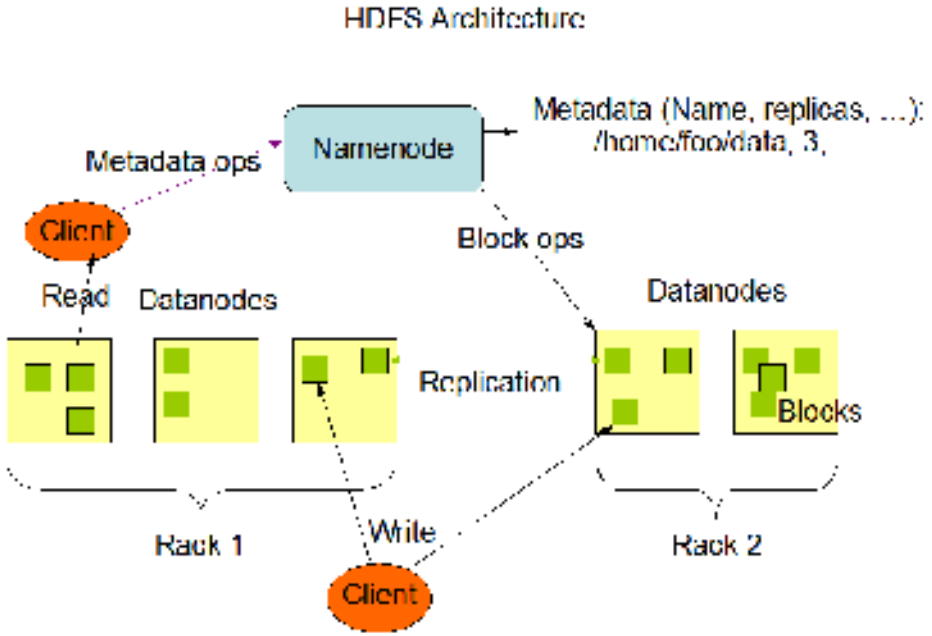
Mappers and Reducers can be chained together

Mappers and Reducers can be chained together



HDFS Sweet spot

- Big Data Storage : Optimized for large files (ETL)
- Writes are create, append, and large
- Reads are mostly big and streaming
- Throughput is more important than latency
- Distributed, HA, Transparent Replication



When is raw HDFS unsuitable?

- Mutable data – Create, Update, Delete
- Small writes
- Random reads, % of small reads
- Structured data
- Online access to data – HDFS Loading is offline / batch process

NoSQL Data stores - Column

- Excellent W/R concurrent performance – fast writes and fast reads (random and sequential) – this is required for near real time update of data to TS Data
- Distributed architecture, horizontal scaling, transparent replication of data
- Highly Available (HA) and Fault Tolerant (FT) for no SPOF – shared nothing architecture
- Reasonably rich data model
- Flexible in terms of schema – amenable to ad-hoc changes even at runtime

HBase

- (Table, Row, Column Family:Column, Timestamp) tuple maps to a stored value
- Table is split into multiple equal sized regions each of which is a range of sorted keys (partitioned automatically by the key)
- Ordered Rows by key, Ordered columns in a Column Family
- Table schema defines Column Families
- Rows can have different number of columns
- Columns have value and versions (any number)
- Column range and key range queries

Row Key	Column Family (dimensions)		Column Family (metric)
112334-7782	server : host1	dc : PUNE	value:20
112334-7783	server:host2		value:10

Hive – Distributed SQL > MR

- MR is not easy to code for analytics tasks (e.g. group, aggregate etc.) chaining several Mappers & Reducers required
- Hive provides familiar SQL queries which automatically gets translated to a flow of appropriate Mappers and Reducers that execute the query leveraging MR.
- Leverages Hadoop ecosystem - MR, HDFS, HBase
- Hive defines a schema for the meta-tables it will use to build a schema its SQL queries can use and to store metadata
- Storage Handlers for HDFS, HBase
- Hive SQL supports common SQL select, filter, grouping, aggregation, insert etc clauses
- Hive stores the data partitioned by partitions (you can specify partitioning key while loading Hive tables) and buckets (useful for statistical operations like sampling)
- Hive queries can also include custom map/reduce tasks as scripts

Hive Queries - CREATE

TABLE

```
CREATE TABLE wordfreq (word
    STRING, freq INT)  ROW FORMAT
    DELIMITED FIELDS TERMINATED
    BY '\t' STORED AS TEXTFILE;

LOAD DATA LOCAL INPATH
    ‘freq.txt' OVERWRITE INTO TABLE
    wordfreq;
```

EXTERNAL TABLE

```
CREATE external TABLE iops(key
    string, os string, deploymentsize
    string, ts int, value int) STORED
    BY
    'org.apache.hadoop.hive.hbase.HB
    aseStorageHandler' WITH
    SERDEPROPERTIES
    ("hbase.columns.mapping" =
    ":key,data:os,data:deploymentSize,
    data:ts,data:value")
```

Hive Queries - SELECT

TABLE

```
select * from wordfreq where freq >
    100 sort by freq desc limit 3;

explain select * from wordfreq where
    freq > 100 sort by freq desc limit 3;

select freq, count(*) AS f2 from
    wordfreq group by freq sort by f2
    desc limit 3;
```

EXTERNAL TABLE

```
select ts, avg(value) as cpu from
    cpu_util_5min group by ts;

select architecture, avg(value) as cpu
    from cpu_util_5min group by
    architecture;
```

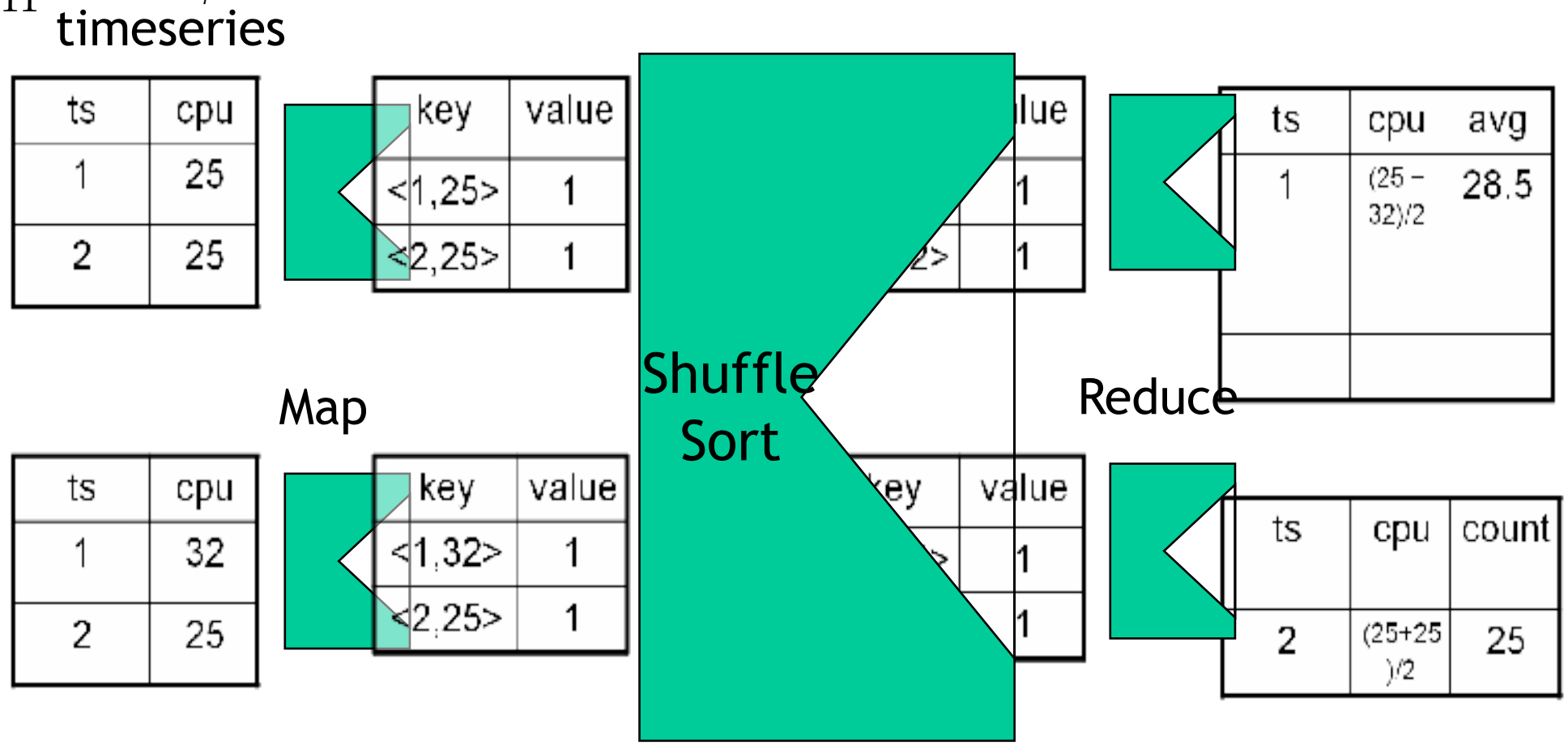
Hive – SQL -> Map Reduce

CPU utilization / 5 min with dimensions server, server-type, cluster, data-center, group by server-type and filter by value *Unix*

SELECT timestamp, AVG(value)

FROM timeseries WHERE server-type = 'Unix'

GROUP BY timestamp



Thanks