We create **innovative software products** that appeal to **global audiences**

Big Data Training

Apache Pig

Jan 2015

BIG DATA & HIGH PERFORMANCE

Globant
we are ready

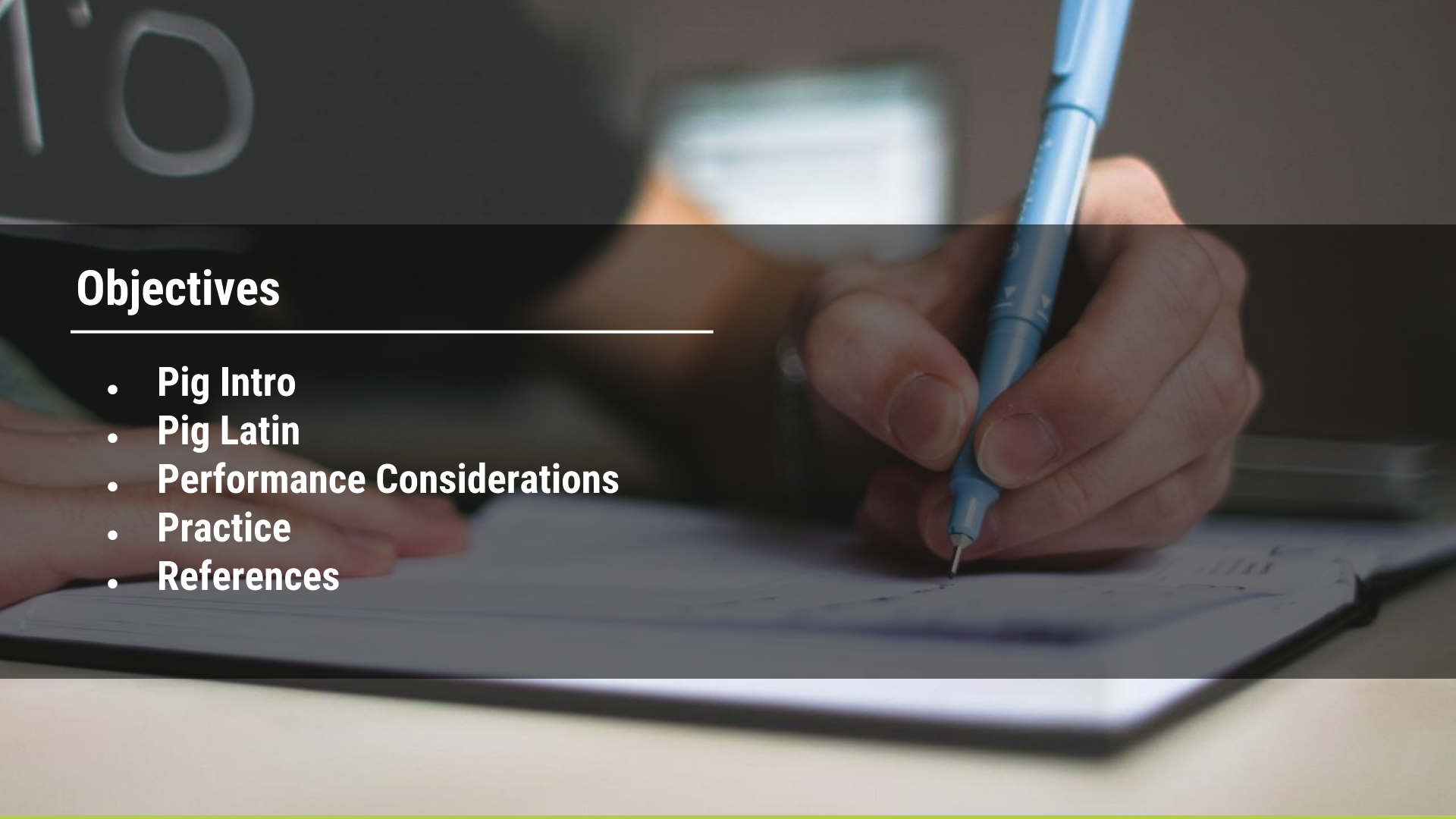
Speakers

Leandro Mora

Big Data Specialist

Gonzalo Zarza

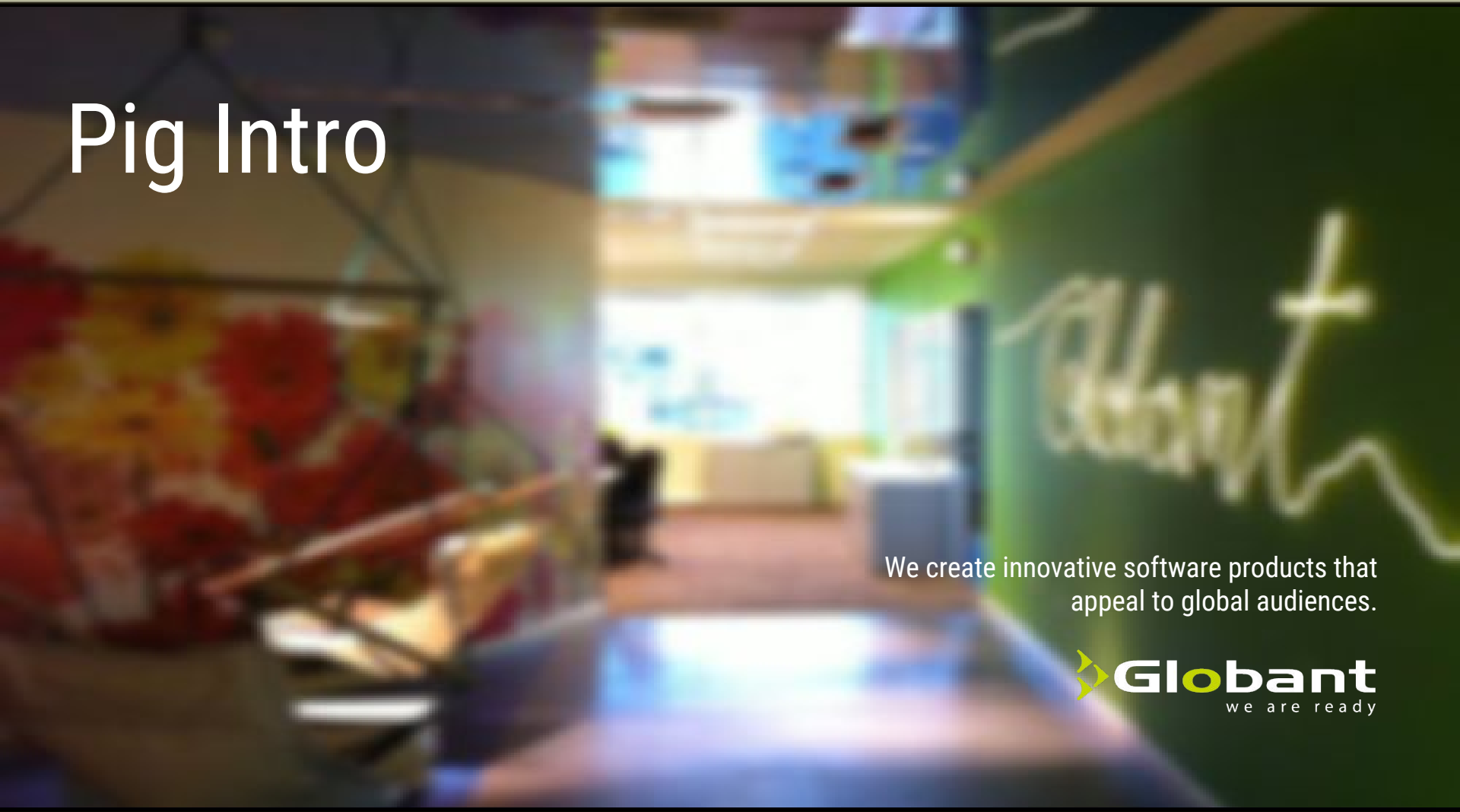
Big Data Specialist



Objectives

- Pig Intro
- Pig Latin
- Performance Considerations
- Practice
- References

Pig Intro


We create innovative software products that
appeal to global audiences.
Globant
we are ready

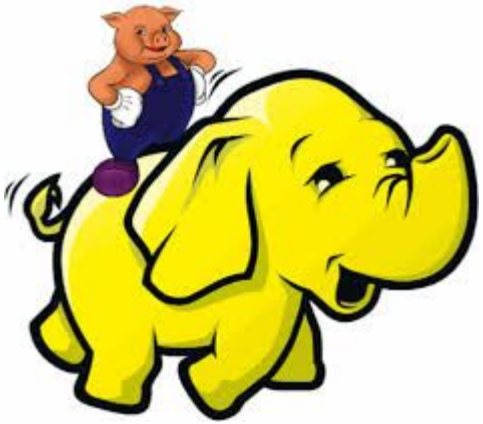
What is Pig?

Pig Latin

Pig provides an **engine for executing data flows in parallel on Hadoop**.

Runs on top of Hadoop

Pig runs on Hadoop. It makes use of both the Hadoop Distributed File System, HDFS, and Hadoop's processing system, MapReduce.



License

Pig is an Apache open source project.

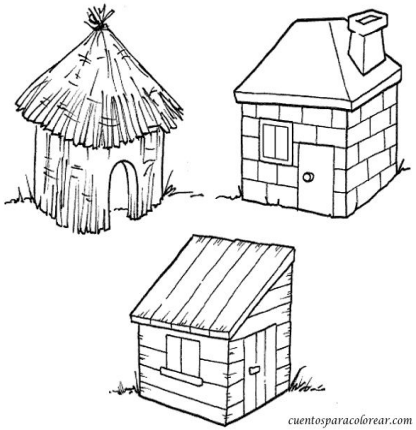
Pig Latin

It includes a language, Pig Latin, for expressing these data flows. Pig Latin includes operators for many of the traditional data operations (join, sort, filter, etc.), as well as the ability for users to develop their own functions for reading, processing, and writing data.

Pig Philosophy

EATS
anything

LOAD DATA FROM HBASE, HIVE, S3, HDFS, ETC...



LIVES
everywhere

USED WITH MAPREDUCE, TEZ, SPARK

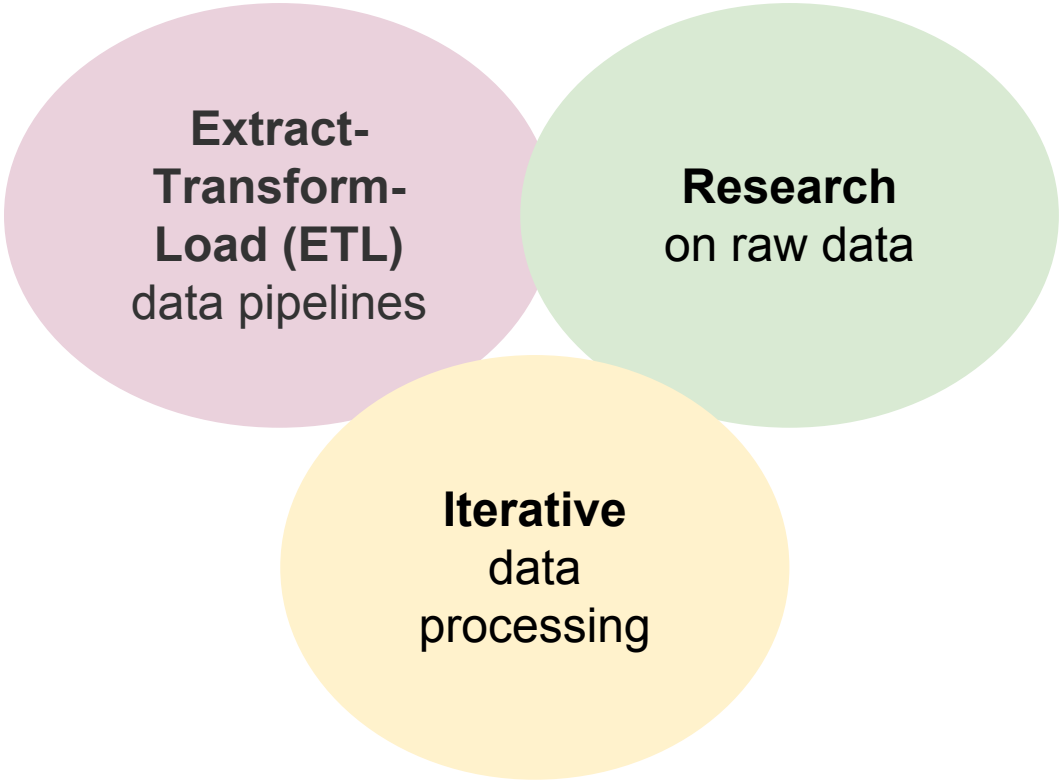
DOMESTIC
animals

SIMPLE, EASY TO CODE

FLY

PERFORM FAST

What Pig does?



Pig vs MapReduce

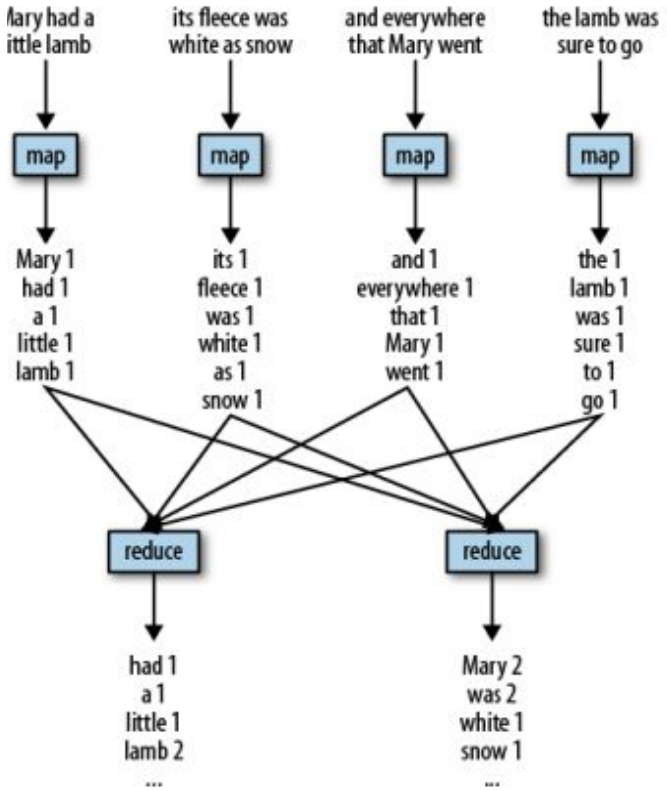
Example 1-1. Pig counts Mary and her lamb

```
-- Load input from the file named Mary, and call the single
-- field in the record 'line'.
input = load 'mary' as (line);

-- TOKENIZE splits the line into a field for each word.
-- flatten will take the collection of records returned by
-- TOKENIZE and produce a separate record for each one, calling the single
-- field in the record word.
words = foreach input generate flatten(TOKENIZE(line)) as word;

-- Now group them together by each word.
grpd  = group words by word;

-- Count them.
cntd  = foreach grpd generate group, COUNT(words);
-- Print out the results.
dump cntd;
```



Pig vs MapReduce

Pig	VS	MapReduce
Custom operations, easy to code		Custom code, lot of lines
Quick coding		Time consuming
Tested operations / Easy to maintain		Error Prone
Limited control on the number of maps and reduces generated		Full control on the number of maps and reduces generated by the platform
Not all kinds of operations are supported (example sub-queries over data)		Custom Code - Lots of lines

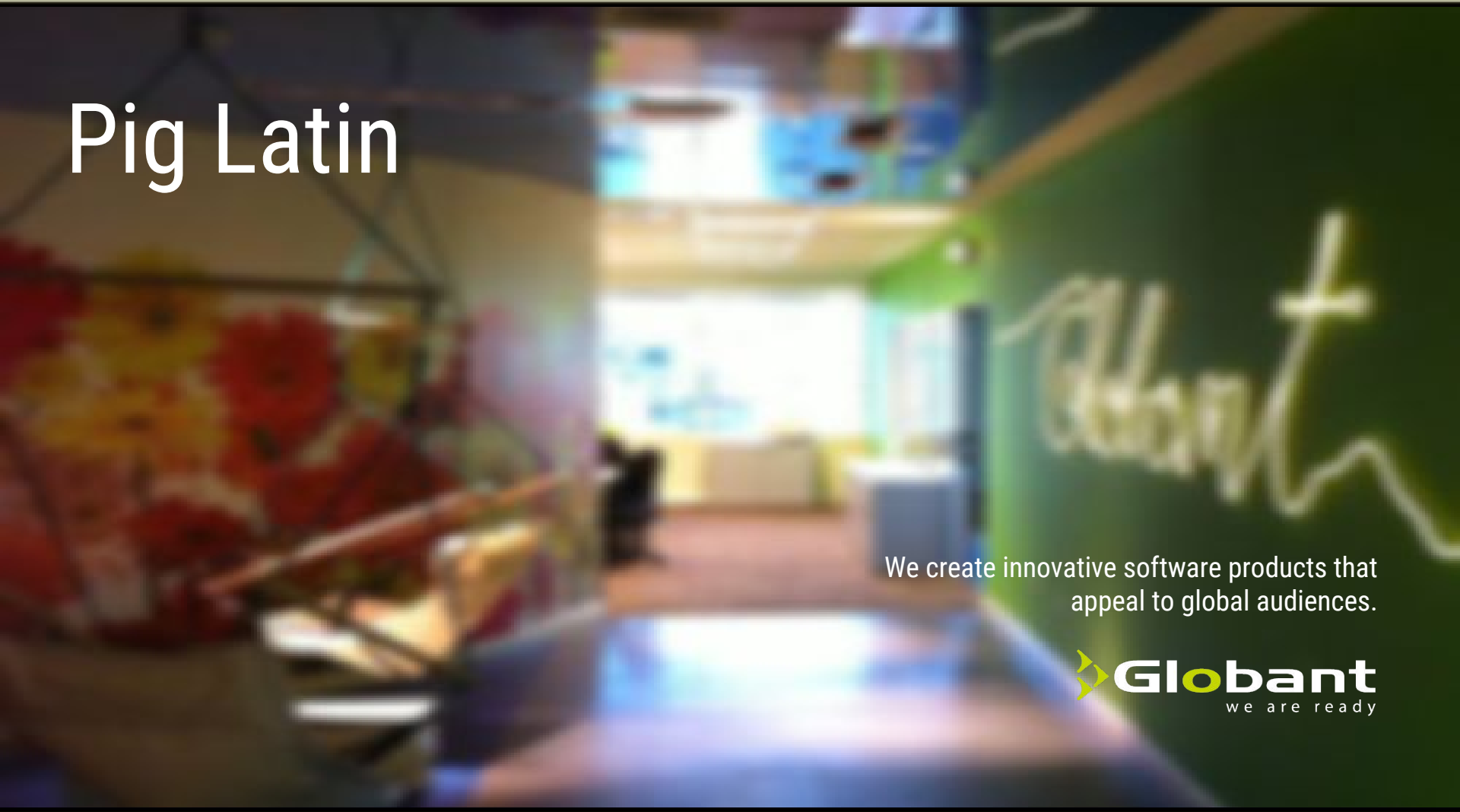
Data Model

TYPE	DESCRIPTION
int	4-byte signed integer. Represented in interfaces by *java.lang.Integer*.
long	8-byte signed integer. Longs are represented in interfaces by *java.lang.Long*.
float	floating-point number. Floats are represented in interfaces by *java.lang.Float*
double	A double-precision floating-point number. Doubles are represented in interfaces by *java.lang.Double*
chararray	String or character array. Chararrays are represented in interfaces by *java.lang.String*.
bytearray	A blob or array of bytes. Bytearrays are represented in interfaces by a *Java class DataByteArray*

Data Model

TYPE	DESCRIPTION
map	A map in Pig is a chararray to data element mapping, where that element can be any. Pig type, including a complex type. The chararray is called a key and is used as an index to find the element, referred to as the value. *['name'#'bob', 'age'#55]*
tuple	A tuple is a fixed-length, ordered collection of Pig data elements. Tuples are divided into fields, with each field containing one data element. These elements can be of any type—they do not all need to be the same type.
bag	A bag is an unordered collection of tuples. Because it has no order, it is not possible to reference tuples in a bag by position. Like tuples, a bag can, but is not required to, have a schema associated with it.

Pig Latin


We create innovative software products that
appeal to global audiences.
Globant
we are ready

Pig Latin - Schemas

```
X = FOREACH C GENERATE FLATTEN(B) AS (f1:int, f2:int, f3:int),
group;

A = LOAD 'data' AS (f1:int, f2:int);

A = LOAD 'data' AS (T: tuple (f1:int, f2:int, f3:int));
```

Pig Latin - Basics

FILTER

```
A = LOAD 'data'
AS
(f1:int,f2:int,f3
:int);

X = FILTER A BY
f3 == 3;
```

(!) Used to select the data that you want; or, conversely, to filter out (remove) the data you don't want

FOREACH

```
A = LOAD 'data' AS
(f1:int,f2:int,f3:int);

X = FOREACH A GENERATE
f1, f2;
```

GROUP

```
A = load 'student' AS
(name:chararray,age:int,
gpa:float);

B = GROUP A BY age;
```

ORDER BY

```
A = LOAD 'data' AS
(a1:int,a2:int,a3:in
t);

X = ORDER A BY a3
DESC;
```

Pig Latin - Basics

RANK

```
A = load 'data'
AS (f1:chararray,
f2:int,
f3:chararray);

C = RANK A BY f1
DESC, f2 ASC;
```

CROSS

```
A = LOAD 'data1' AS
(a1:int,a2:int,
a3:int);

B = LOAD 'data2' AS
(b1:int,b2:int);

X = CROSS A, B;
```

SPLIT

```
A = LOAD 'data' AS
(f1:int,f2:int,
 f3:int);

SPLIT A INTO X IF
f1<7, Y IF f2==5, Z
IF (f3<6 OR f3>6);
```

UNION

```
A = LOAD 'data' AS
(a1:int,a2:int,
a3:int);

B = LOAD 'data' AS
(b1:int,b2:int);

X = UNION A, B;
```

(!)Does not ensure (as databases do) that all tuples adhere to the same schema
(!)Does not eliminate duplicate tuples

Pig Latin - Joins

```
daily = load 'NYSE_daily' as (exchange:chararray,
symbol:chararray,date:chararray, open:float,
high:float, low:float, close:float, volume:int,
adj_close:float);

divs = load 'NYSE_dividends' as
(exchange:chararray, symbol:chararray,
date:chararray, dividends:float);

jnd = join daily by (exchange, symbol), divs by
(exchange, symbol) using 'replicated';
```

Types of Joins
Inner Join
Outer Join

Join Implementations
Replicated
Skewed
Merge
Merge-Sparse

Apache DataFu Pig

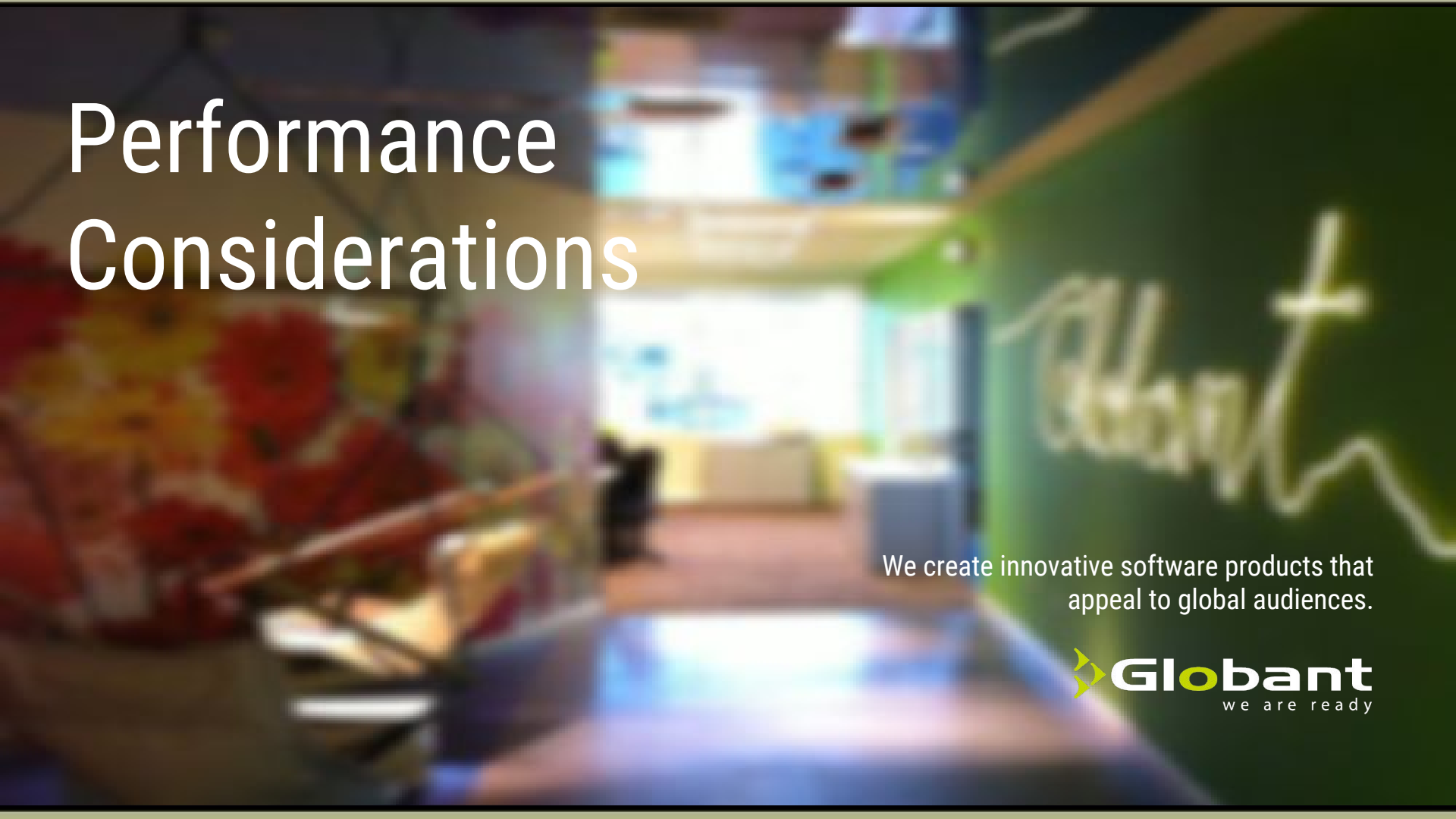
Apache DataFu Pig is a collection of **user-defined functions** for working with large scale data in Apache Pig. It has a number of useful functions available

Statistics Compute quantiles, median, variance, wilson binary confidence, etc.	**Estimation** Streaming implementations that can estimate quantiles, median, cardinality.	**Sampling** Simple random sampling with or without replacement, weighted sampling.
Set Operations Perform set intersection, union, or difference of bags.	**Link Analysis** Run PageRank on a graph represented by a bag of nodes and edges.	**Bags** Convenient functions for working with bags such as enumerate items, append, prepend, concat, etc.
Sessions Sessionize events from a stream of data.	**More** Other useful methods like Assert and Coalesce.	

```
DEFINE Median datafu.pig.stats.StreamingMedian();
...
data = FOREACH (GROUP data ALL) GENERATE Median(data);
```

URL : http://datafu.incubator.apache.org/docs/datafu/getting-started.html

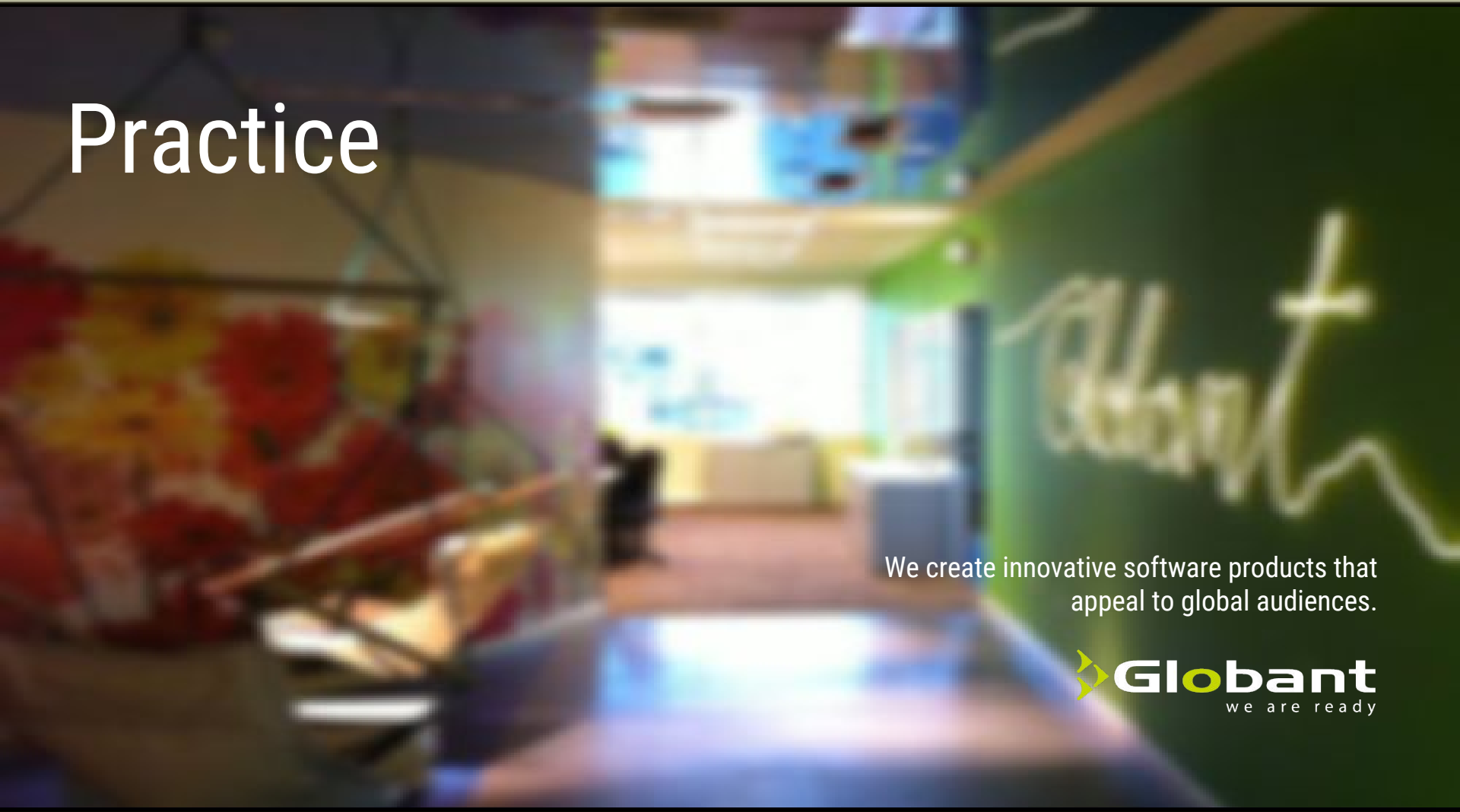
Performance Considerations



Performance Considerations and Tips

- ✓ Pre--process data
- ✓ Extract only the fields that you will need. “Project Early And Often”
- ✓ Filter first. “Filter Early and Often”
- ✓ Drop Nulls Before a Join
- ✓ Avoid Joins
- ✓ Choose the appropriate join command “Take advantage of Join Operations”
- ✓ Set Level of parallelism
- ✓ Check Mapper-Reducer Statistics from Console
- ✓ Understand what kind of mapper-reducers are being runned behind. Use EXPLAIN for this purpose
- ✓ Prefer DISTINCT over GROUP BY/GENERATE
- ✓ Compress the results of intermediate Jobs
- ✓ Combine Small Input Files

Practice


We create innovative software products that
appeal to global audiences.
Globant
we are ready

Example

```
Users = load 'users' as (name, age);
Fltrd = filter Users by age >= 18 and age <= 25;
Pages = load 'pages' as (user, url);
Jnd   = join Fltrd by name, Pages by user;
Grpd  = group Jnd by url;
Smmd  = foreach Grpd generate group, COUNT(Jnd) as clicks;
Srtd  = order Smmd by clicks desc;
Top5  = limit Srtd 5;
store Top5 into 'top5sites';
```

Practice

A. Investigating Crimes

1. Given a real dataset that contains data about Crimes in Chicago from 2001 till now (already stored in MySQL), import it into HDFS using Sqoop.
2. Count all the crimes which description is “Simple”; order those crimes by year; print the results to console, and also store them to a text file.

- Dataset: https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2
- MySQL Table: bdtraining.chicago_crimes

Hint 1: There are many ways for resolving this exercise. Check the “performance considerations” slide and find the most suitable approach.

Practice

B. The 100 most popular words

1. We would like to know the 100 most popular words which appeared in **both** of the following books, in descending order by number of appearances:

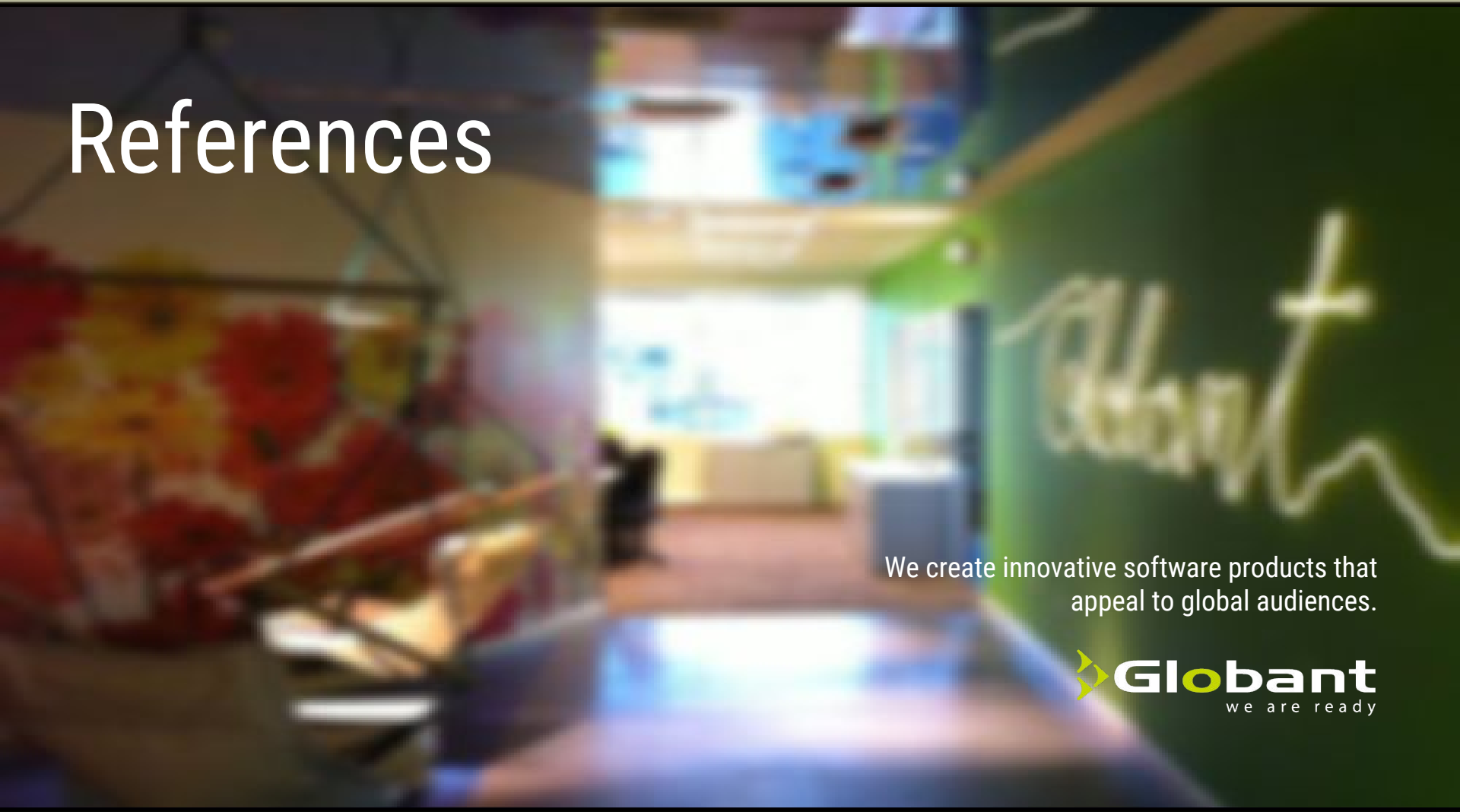
    - the_adventures_of_sherlock_holmes.txt
    - the_adventures_of_tom_sawyer.txt

    (*) Filter articles and pronouns

- URL: /home/hadoop/mapreduce/data/books

Hint 1: Remember that the JOIN operation is one of the most expensives in terms of performance. Consider to pre-process the data before joins.

References


We create innovative software products that
appeal to global audiences.
Globant
we are ready

References

- **Apache Pig Documentation.**- http://pig.apache.org/docs/r0.12.1/
- **Hadoop: The Definitive Guide, 2nd Edition** (Chapter 11)**.** O'Reilly Media / Yahoo Press - Online @ Globant's Big Data Training Site
- **Pig Programming**. O'Reilly Media - Online @ Globant's Big Data Training Site
- **Pig Design Patterns**. Packt Publishing - Online @ Globant's Big Data Training Site

Thanks