Advanced Munging

Parallel and distributed ordered join benchmark

H2O Open Tour New York

19 July 2016

Matt Dowle

H2O Frame

Columnar

Compressed

In-memory

One column can be larger than one node

Parallel and distributed data loader

API: Python, R, Java and REST

R's data.table join

1. Find order of the left join columns
2. Find order of the right join columns
3. Binary merge the two sorted indexes

No hash table at all

Fast **ordered joins**; e.g. rolling forwards, backwards, nearest and limited staleness

https://github.com/Rdatatable/data.table/wiki

But data.table is ...

Single threaded

Single node

Limited to 2 billion rows (2^31)

=> H2O

Every step has now been parallelized and distributed

Open benchmark

https://github.com/h2oai/db-benchmark/

by Jan Gorecki @ H2O

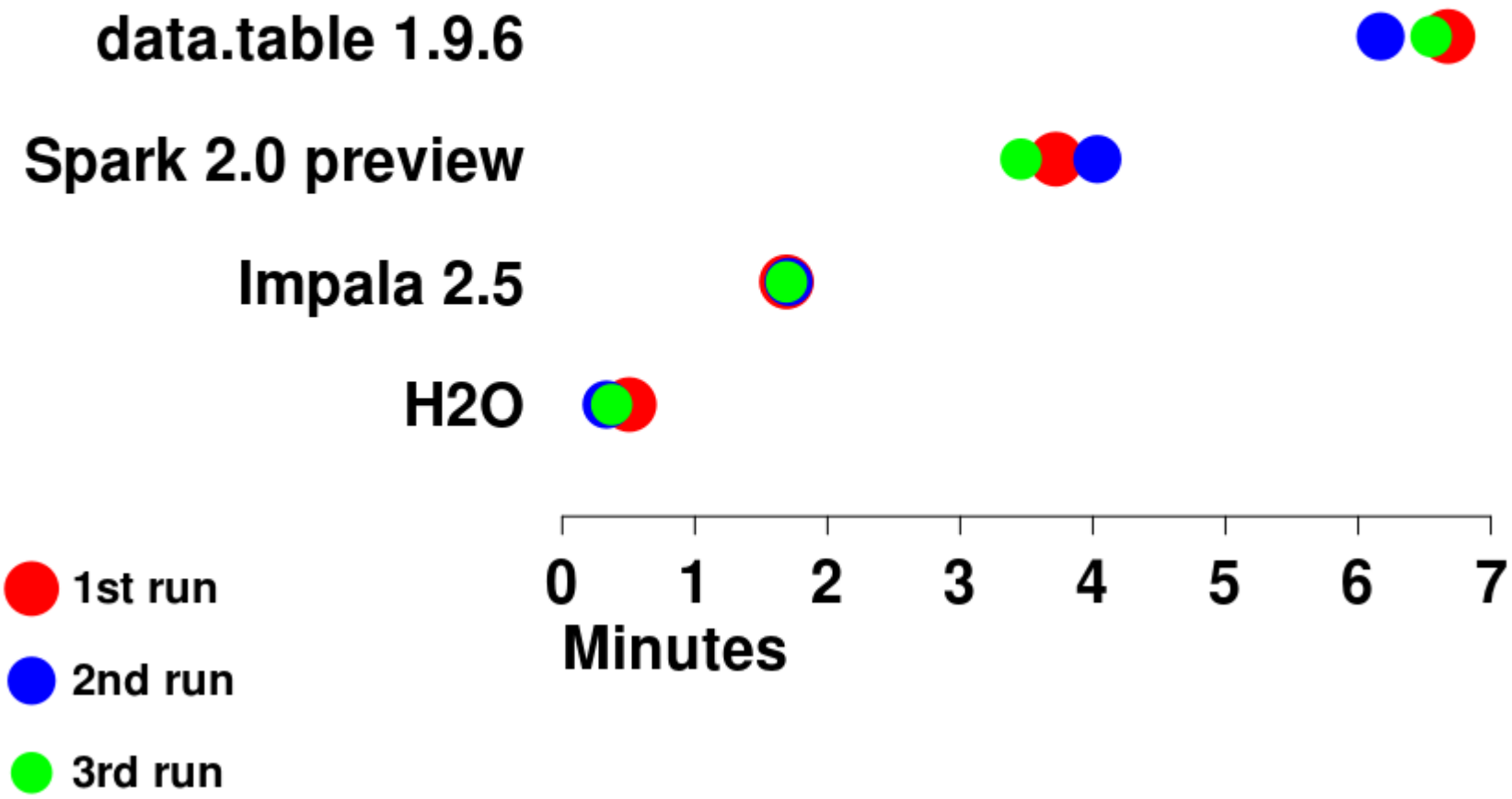
Feedback sought

Pull requests welcome

We started with one tough test :

high cardinality big join

1bn row results on 9 nodes



Cardinality

Not just the number of rows, but what's in the rows

A 10 billion row file could contain :

- 500 stock tickers (low cardinality)
- Millions of people (medium cardinality)
- Billions of devices (high cardinality)

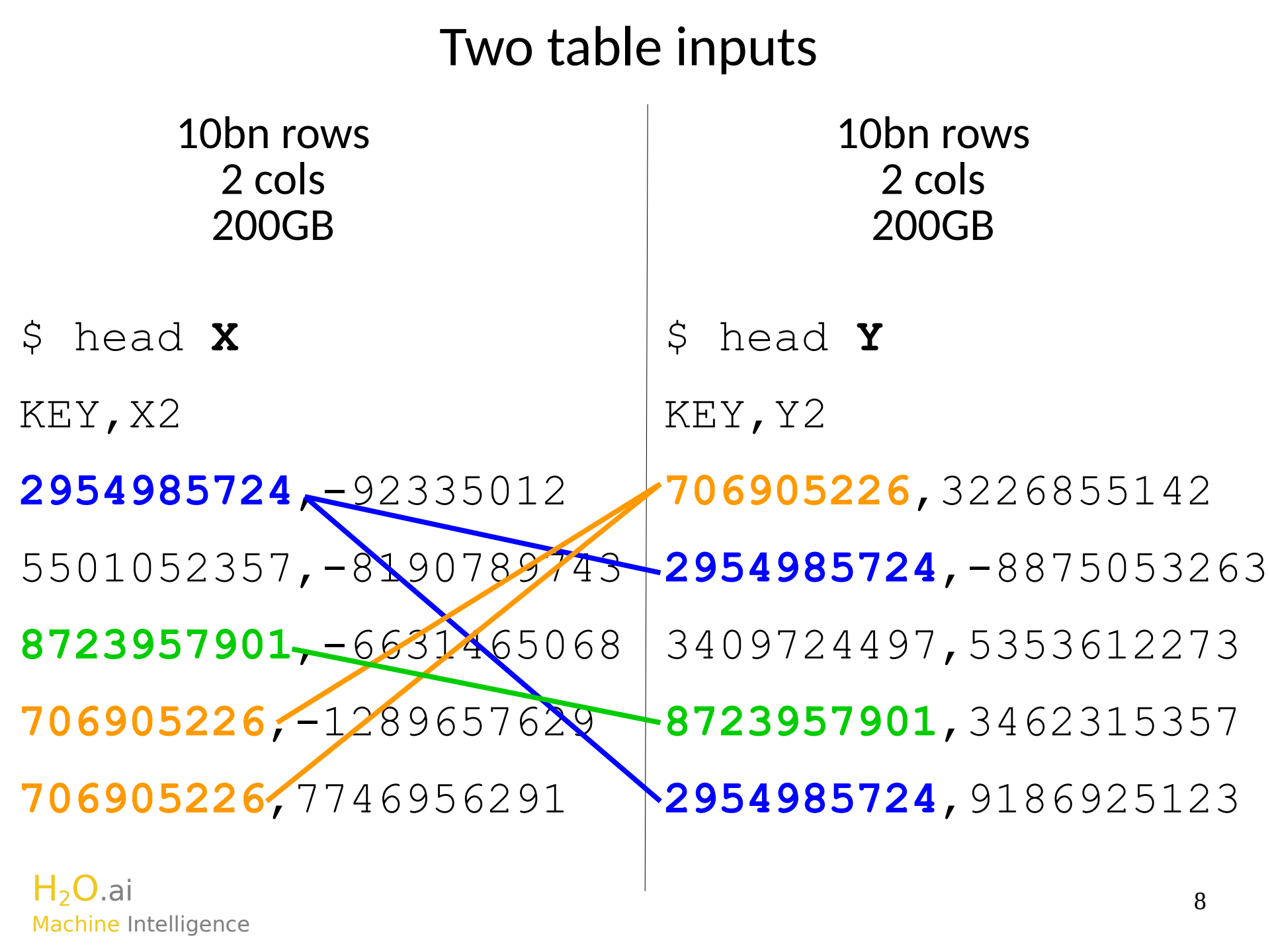
Two table inputs

10bn rows
2 cols
200GB

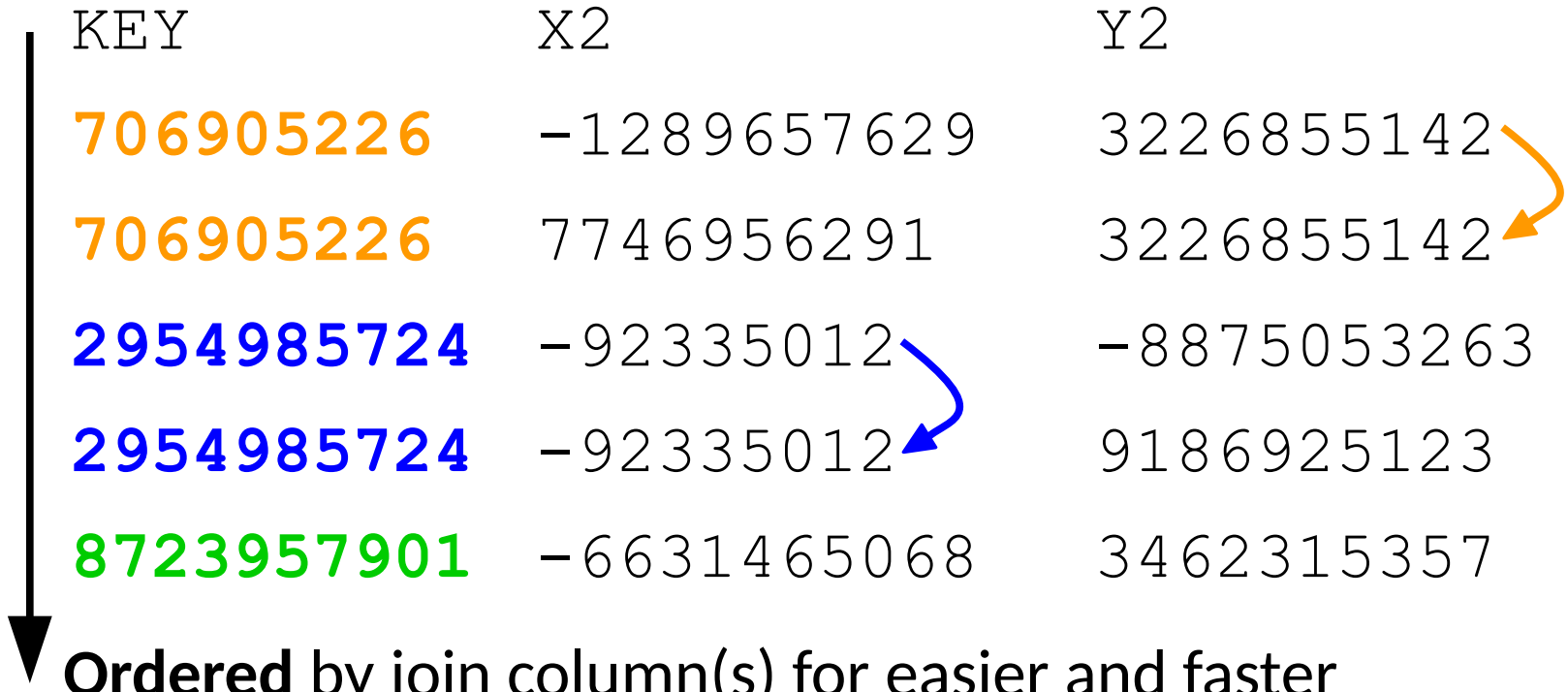
```
$ head X
KEY,X2
2954985724,-92335012
5501052357,-819078974 3
8723957901,-6631465068
706905226,-1289657629
706905226,7746956291
```

10bn rows
2 cols
200GB

```
$ head Y
KEY,Y2
706905226,3226855142
2954985724,-8875053263
3409724497,5353612273
8723957901,3462315357
2954985724,9186925123
```



Result
~10bn rows; 3 cols; 300GB



KEY	X2	Y2
706905226	-1289657629	3226855142
706905226	7746956291	3226855142
2954985724	-92335012	-875053263
2954985724	-92335012	9186925123
8723957901	-6631465068	3462315357

Ordered by join column(s) for easier and faster subsequent operations

NB: Outer join is also implemented. Inner join is illustrated.

H2O commands are easy

```
library(h2o)

h2o.init(ip="mr-0xd6", port=55666)

X = h2o.importFile("hdfs://mr-
0xd6/datasets/mattd/X1e10_2c.csv")

Y = h2o.importFile("hdfs://mr-
0xd6/datasets/mattd/Y1e10_2c.csv")

ans = h2o.merge(X, Y, method="radix")

system.time(print(head(ans)))
```

Live demo ...

Video recording here:
https://github.com/Rdatatable/data.table/wiki/Presentations