

Python vs. Scala

Comparing speed, scalability and performance for a distributed movie recommender using ALS

Connor Capitolo, Ben Liu, Phil Bell



MovieLens Datasets

- MovieLens 100K (2.3 MB): 100,000 ratings from 1000 users on 1700 movies
- MovieLens 1M dataset (12 MB): 1 million ratings from 6000 users on 4000 movies
- **MovieLens 20M dataset (305.2 MB): 20 million ratings on 27,000 movies by 138,000 users**
- MovieLens 25M dataset (390.2 MB): 25 million ratings on 62,000 movies by 162,000 users.

Raw Data from ratings.csv

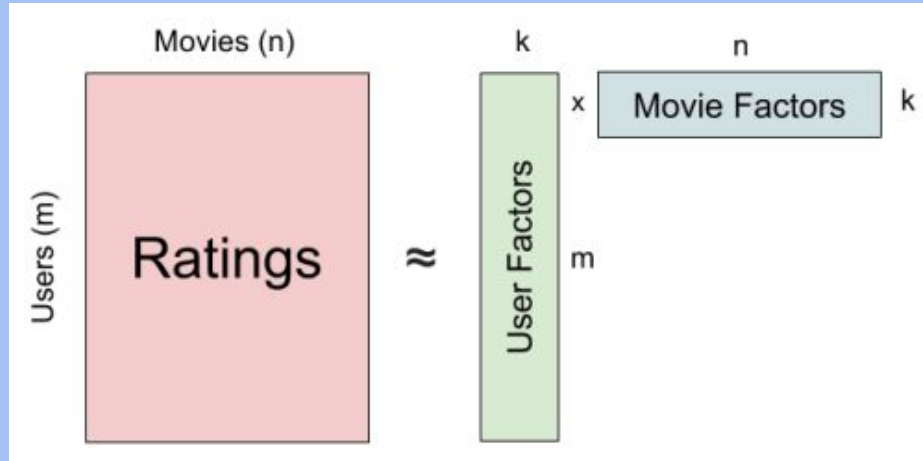
userid	movieid	rating	timestamp
1257	22349	4.0	964982703
2945	203948	1.0	964983605
6404	3038372	3.0	964984100

Utility Matrix

	movie1	movie2	movie3
User1	4.0		3.0
User2		2.0	
User3			5.0



Alternating Least Squares (ALS) algorithm



$$R (m \times n) = W (m \times k) \times H (k \times n)$$



'Block-to-block join' used to distribute user, item and ratings matrices efficiently.

'Hybrid partitioning' can reduce shuffling.

AWS Architecture

GPU + CPU

AWS emr-6.2.0 cluster: Spark 3.0.1 on Hadoop 3.2.1 YARN (1 master and 1 worker node)

GPU Only

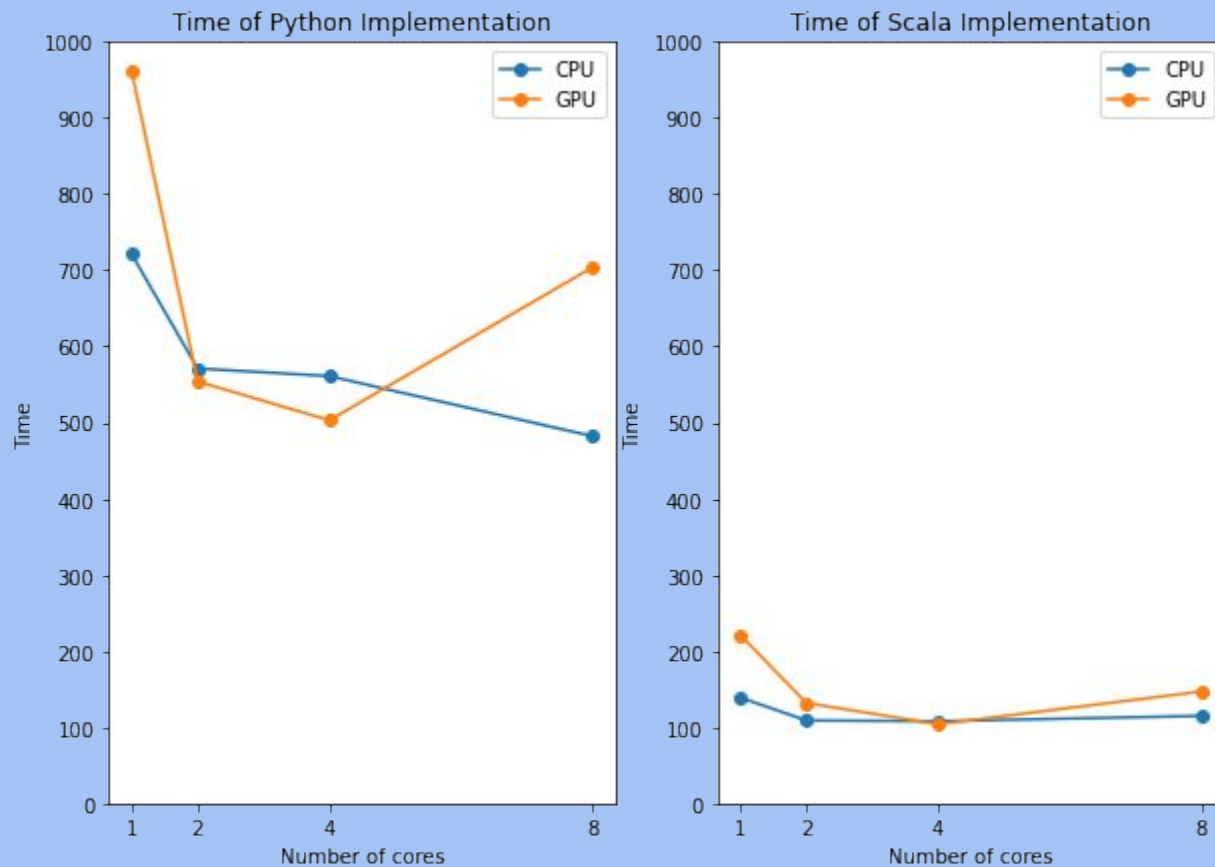
- g4dn.2xlarge, 1 GPU, 8 vCPUs, 32 GiB of memory, 225 NVMe SSD, up to 25 Gbps network performance
- **NVIDIA spark-rapids**

CPU Only

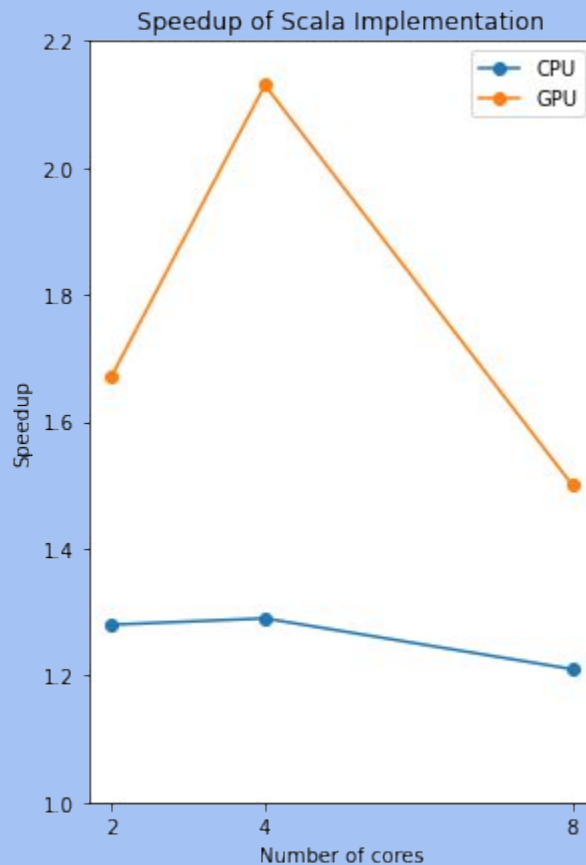
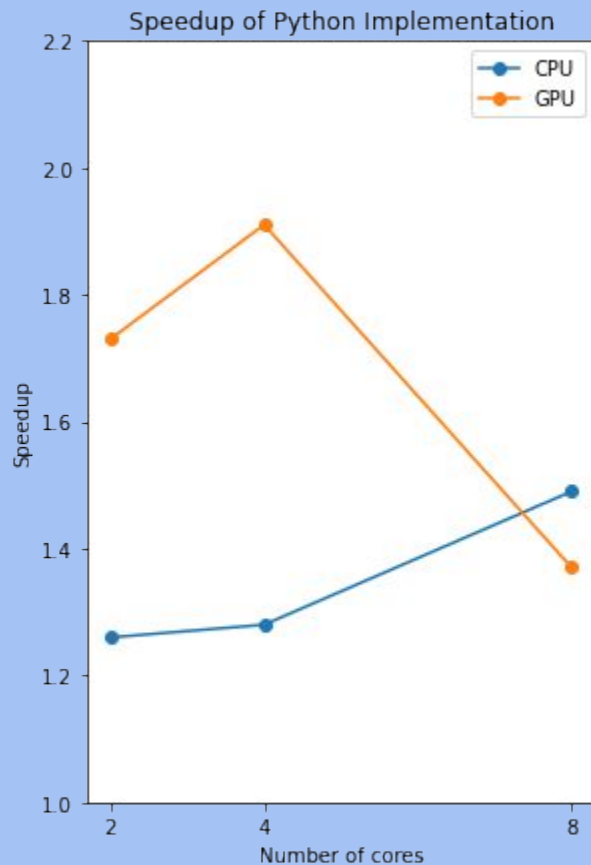
- m4.2xlarge: 32 GiB of memory, 8 vCPUs, EBS-only, 64-bit platform



GPU vs CPU Execution Time



GPU vs CPU Speedup



Python

**Example Scala and Python Recommender
Runtimes**

Scala

Process RDD

18 Seconds

Process RDD

18 Seconds

Matrix Factorization

~69 seconds

Matrix Factorization

~52 Seconds

Aggregation/ Predictions

502 seconds

Aggregation/ Predictions

77 Seconds

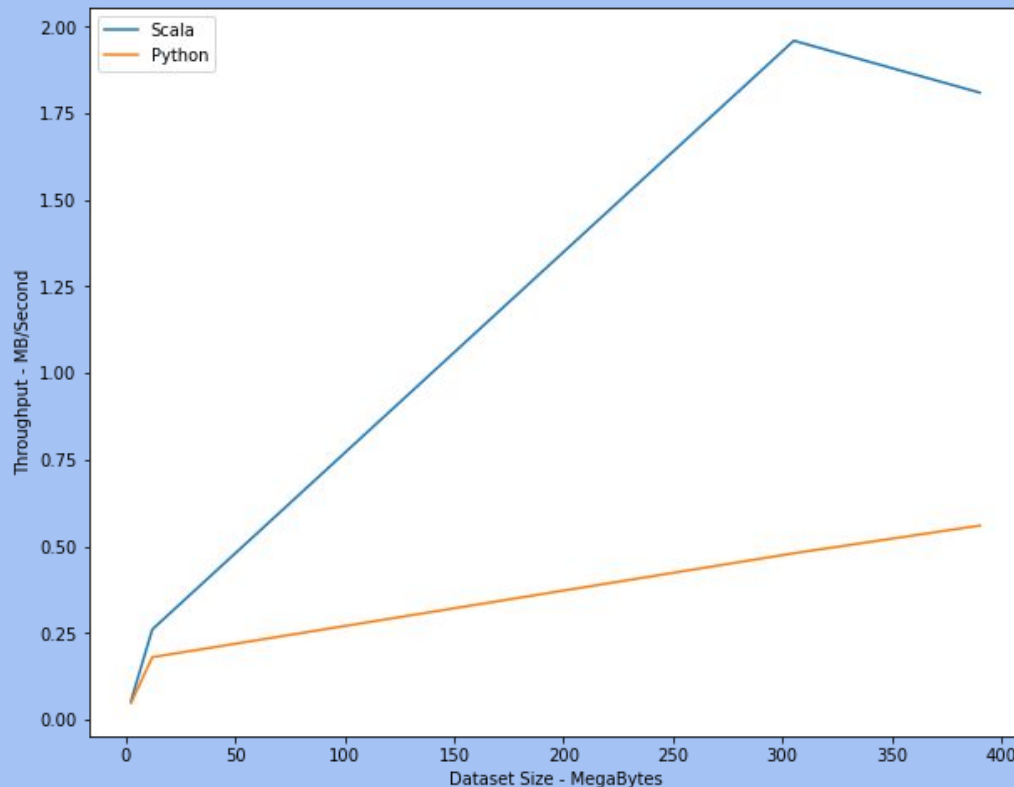
On 20ml dataset with
GPU cluster, 1 node, 8
threads.



IACS
INSTITUTE FOR APPLIED
COMPUTATIONAL SCIENCE
AT HARVARD UNIVERSITY

Weak Scaling

- Scala programme scales to large datasets more effectively than Python implementation.
- The throughput declines from using the 20ml dataset to the 25 ml dataset.
- This may reveal a limit of Scala's scalability to the ALS recommender application.



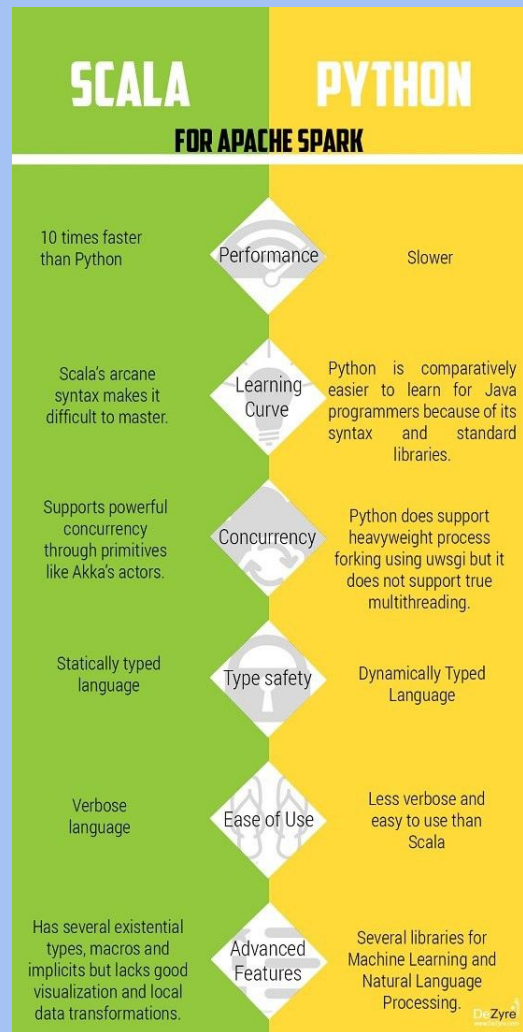
Let a hundred big data languages bloom?

- The trend towards heterogeneous programming languages may not be as strong as the trend towards heterogeneous hardware.



Comparison with other research

- JVM is a critical reason that Scala scales better than Python.
- Difficult to disentangle this from other Scala features.
- ALS is thought to scale badly to large datasets.



Conclusion

- Scala is a little more challenging to use but provides better performance for this task.
- Integrating GPUs with Spark may not be the best option for some applications.
- Distributing ALS matrix factorisation is not enough to fully parallelise an ALS recommender.

```
def echoWhatYouGaveMe(x: Any): String = x match {  
  // constant patterns  
  case 0 => "zero"  
  case true => "true"  
  case "hello" => "you said 'hello'"  
  case Nil => "an empty list"  
  
  // sequence patterns  
  case List(0, _, _) => "a three-element list with 0 as the first element"  
  case List(1, _) => "a list beginning with 1, having any number of elements"  
  case Vector(1, _) => "a vector starting with 1, having any number of elements"  
  
  // tuples  
  case (a, b) => s"got $a and $b"  
  case (a, b, c) => s"got $a, $b, and $c"  
  
  // constructor patterns  
  case Person(first, "Alexander") => s"found an Alexander, first name = $first"  
  case Dog("Suka") => "found a dog named Suka"  
  
  // typed patterns  
  case s: String => s"you gave me this string: $s"  
  case i: Int => s"thanks for the int: $i"  
  case f: Float => s"thanks for the float: $f"  
  case a: Array[Int] => s"an array of int: $a.mkString(",")"  
  case as: Array[String] => s"an array of strings: $as.mkString(",")"  
  case d: Dog => s"dog: $d.name" // use: $d.name  
  case list: List[_] => s"thanks" // use: $list  
  case m: Map[_] => m.toString  
  
  // the default wildcard pattern  
  case _ => "unknown"  
}
```



Scala

Programming

Next Steps

- Compare with Java to compare two languages that use the JVM.
- Use 1B dataset to test Scala scalability past 300mb.
- Compare RDD implementation and dataframe implementation.
- Speed up the aggregation and MSE.