Parallel Computing with Apache Spark

MTAT.08.020

Tek Raj Chhetri
tek.raj.Chhetri@ut.ee
Computing Today

Bioinformatics
Image Source [2]

Computer Vision
Image Source [3]

Big Data
Image Source [4]

AI
Image Source [5]
Computing Today

• Is more intensive than yesterday and will be even more tomorrow.

• Time constraint – Need result faster, sometimes almost in real-time.

• Solution?
  • Parallel Computing

• The significance of parallel computing now is now than ever and will be even more tomorrow.
Parallel Computing?

- Parallel computing is a type of computation in which many calculations or the execution of processes are carried out simultaneously. – Wikipedia

- Large problems are divided into smaller ones and solved simultaneously – Not all problems can be parallelized.
Why Parallel Computing?

- Because problems are too costly with the classical approach [1].
- Faster Result.
- Cost efficiently – can make use of multiple cheap hardware for parallel processing – Hadoop.
Why Spark for Parallel Computing?

• Implicit data parallelism and fault tolerance.
• Provides fault tolerance unlike MPI, OpenMP.
• Speed.
• Ease of use - Java, Scala, Python, R, and SQL.
  • Spark offers over 80 high-level operators that make it easy to build parallel apps
• Ability to Scale.
• Runs Everywhere.
Why Spark for Parallel Computing?

- Supports Streaming, Complex Data Analysis, Machine Learning, Graph Processing.
- Scientific Computing [6].
- In memory computation.
Why Spark for Parallel Computing?

- Based on RDD and DAG (Direct Acyclic Graph)
- Spark applications run as independent sets of processes on a cluster, coordinated by the SparkContext object in your main program (called the driver program) [7].
- Workers nodes are the slave nodes on partitioned RDD and return back result to SparkContext.
- Increasing number of worker, jobs can be divided into more partition and execute them parallel. – Faster
- RDD – Resilient Distributed Dataset
Resilient Distributed Dataset

- RDD – a collection of elements partitioned across the nodes of the cluster that can be operated on in parallel.

- Immutable - cannot be modified but transformed.

- Operations in RDD
  - Transformations: Operations applied to create a new RDD.
  - Actions: Instruct Spark to apply computation and pass the result back to the driver.

- Lazy in nature – also a reason for faster performance.

- RDDs are created by starting with a file in the Hadoop file system (or any other Hadoop-supported file system), or an existing Scala collection in the driver program, and transforming it.
Let’s go for live spark execution

Notebook Access
Fig: Number of papers published 2019 (Google Scholar)
• Big Data Analytics in the Cloud: Spark on Hadoop vs MPI/OpenMP on Beowulf

• The main differences between these two frameworks are fault tolerance support and data replication.
• OpenMP/MPI provides a solution mostly oriented to high performance computing but susceptible to faults.
• KNN and SVM-Pegasos algorithms
• MPI drawback is that it is not suitable for small grain level of parallelism
• OpenMP it is not suitable for distributed memory systems
• Data I/O management, MPI/OpenMP and Spark are much closer in terms of disk access time although MPI/OpenMP is still faster
• MPI/OpenMP implementation is much more powerful than the Spark on Hadoop alternative in terms of speed.
References:

• [8] https://www.edureka.co/blog/spark-architecture/
Thank You!

Institute of Computer Science