Evaluación del Rendimiento y la Escalabilidad de un Clúster Apache Spark y Hadoop en un Entorno de Bajo Costo

Evaluating the Performance and Scalability o fan Apache Spark and Hadoop Cluster in a Low-Cost Environment

Natalie Cruz Tumba [A], Alex Lancho Ramos [B], Henry Leon Hurtado [C], Rafael Ricardo Quispe Merma[D] Evelyn Naida Luque Ochoa[E]

Resumen— Este artículo presenta el diseño, configuración e implementación de un clúster de cómputo distribuido utilizando Apache Spark y Hadoop sobre Ubuntu Server 24.04.1 LTS. La arquitectura consta de un nodo maestro y múltiples nodos esclavos conectados en red local mediante Ethernet. Se detalla el proceso de instalación, configuración y pruebas de rendimiento con PySpark. Los resultados demuestran que, si bien una configuración local es más eficiente para datasets pequeños (<100 MB), el clúster distribuido ofrece mejoras significativas para volúmenes de datos superiores a 1 GB, validando su escalabilidad y viabilidad para entornos educativos y de investigación con recursos limitados.

Palabras clave: apache spark, procesamiento de datos, hadoop, clúster distribuido, computación distribuida, clúster de bajo costo

Abstract— This article presents the design, configuration, and implementation of a distributed computing cluster using Apache Spark and Hadoop on Ubuntu Server 24.04.1 LTS. The architecture consists of a master node and multiple slave nodes connected to a local network via Ethernet. The installation, configuration, and performance testing process with PySpark are detailed. The results demonstrate that, while a local configuration is more efficient for small datasets (<100 MB), the distributed cluster offers significant improvements for data volumes greater than 1 GB, validating its scalability and viability for resource-constrained educational and research environments.

Keywords: Apache Spark, data processing, Hadoop, distributed cluster, distributed computing, low-cost cluster

C&T Riqchary

Revista de Investigación en Ciencia y Tecnología
ISSN: 2810-8124 (en línea) / ISSN: 2706-543x
Universidad Nacional Micaela Bastidas de Apurimac – Perú

Vol. 7 Núm. 2 (2025) - Publicado: 19/08/25 - Indexaciones
Número: doi.org/10.57166/riqchary/v7.n2.2025
Páginas: 49- 53 | Recibido 01/01/2025 ; Aceptado 01/02/2025

doi.org/10.57166/riqchary.v7.n2.2025.6

Autores:

A. ORCID iD https://orcid.org/0009-0006-3110-2087
Natalie Cruz Tumba, Estudiante de la carrera de Ingeniería Informatica y Sistemas de la Universidad Nacional Micaela Bastidas de Apurimac, 201059@unamba.edu.pe.

B. ORCID iD https://orcid.org/0009-0008-5493-397X
Alex Lancho Ramos, Estudiante de la carrera de Ingeniería Informatica y Sistemas de la Universidad Nacional Micaela Bastidas de Apurimac, 201227@unamba.edu.pe.

C. ORCID iD https://orcid.org/0009-0001-8216-1232
Henry Leon Hurtado, Estudiante de la carrera de Ingeniería Informatica y Sistemas de la Universidad Nacional Micaela Bastidas de Apurimac, 211179@unamba.edu.pe.

D. ORCID iD https://orcid.org/0000-0002-8980-4560
Rafael Ricardo Quispe Merma, Ingeniero Informático y de Sistemas, Docente del Departamento Académico de Informática y SIstemas de la Universidad Nacional Micaela Bastidas de Apurímac, rquispe@unamba.edu.pe.

E. ORCID iD https://orcid.org/0000-0002-8386-9806
Evelyn Naida Luque Ochoa, Ingeniero Informático y de Sistemas, Docente del Departamento Académico de Informática y SIstemas de la Universidad Nacional Micaela Bastidas de Apurímac, eluque@unamba.edu.pe.

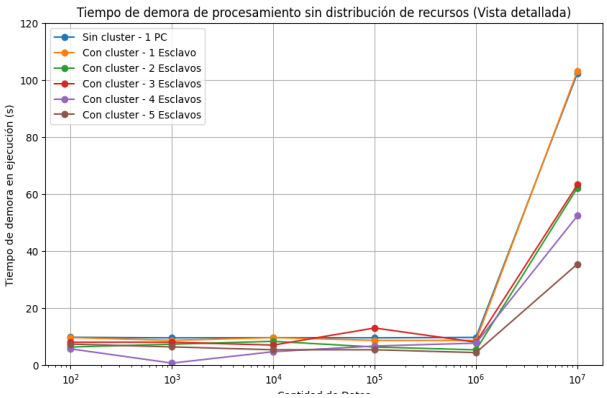

Fig 1. Tiempo de demora de procesamiento sin distribucion de recursos.

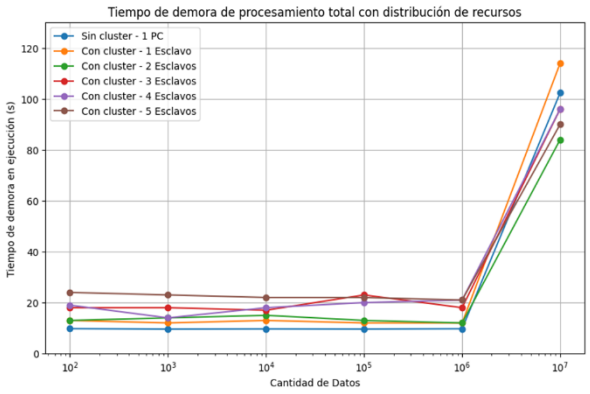

Fig 2. Tiempo de demora de procesamiento total con distribucion de recursos

https://doi.org/10.57166/riqchary.v7.n2.2025.6

1 INTRODUCCIÓN

El crecimiento exponencial de los volúmenes de datos ha generado la necesidad de arquitecturas capaces de procesar información de manera distribuida y eficiente [1]. En este contexto, tecnologías como Apache Hadoop y Apache Spark han emergido como soluciones robustas para el procesamiento de big data [2], [3]. El ecosistema Hadoop proporciona un framework confiable para el almacenamiento distribuido y procesamiento de grandes conjuntos de datos [4], mientras que Apache Spark ofrece capacidades de procesamiento en memoria que mejoran significativamente el rendimiento en comparación con MapReduce tradicional [5].

Los clústeres distribuidos han demostrado su eficacia en diversas aplicaciones, desde análisis de datos científicos hasta procesamiento de información empresarial [6]. La implementación de estos sistemas en entornos educativos permite a los estudiantes y investigadores familiarizarse con tecnologías de vanguardia sin requerir inversiones prohibitivas [7].

Este artículo documenta la implementación de un clúster multinodo en un entorno Linux (Ubuntu 24.04.1 LTS) con el propósito de ejecutar tareas de procesamiento de datos mediante PySpark. El objetivo principal es analizar el impacto del número de nodos en el rendimiento computacional, evaluando los tiempos de ejecución al escalar la infraestructura del clúster [8].

La motivación de esta investigación radica en la necesidad de proporcionar una solución accesible para instituciones educativas y de investigación que requieren capacidades de procesamiento paralelo sin inversiones significativas en infraestructura especializada [9]. Estudios previos han demostrado la viabilidad de implementar clústeres de bajo costo utilizando hardware convencional [10].

Investigaciones previas han demostrado el potencial de Apache Spark y Hadoop en entornos de cómputo distribuido de bajo costo. Por ejemplo, **Zaharia et al.** desarrollaron el concepto de RDD para optimizar el procesamiento en memoria, mientras que estudios como **Implementation of Apache Spark Computing Cluster y Distributed Big Data Analysis Using Spark** evidencian mejoras significativas en tiempos de ejecución al aumentar el número de nodos, incluso en configuraciones de bajo presupuesto.

La elección de Ubuntu Server, Apache Hadoop y Apache Spark responde a su carácter de software libre, su amplia adopción en entornos de investigación y su compatibilidad con hardware convencional. Ubuntu proporciona estabilidad y soporte a largo plazo; Hadoop facilita el almacenamiento y gestión de grandes volúmenes de datos de forma distribuida; y Spark aporta un motor de procesamiento en memoria de alto rendimiento, lo que permite un uso más eficiente de los recursos.

```xml
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://192.168.18.30:9000</value>
  </property>
</configuration>
```

Fig 3. Configuracion del archive .xml

2. METODOLOGÍA

2.1 ENTORNO Y HERRAMIENTAS

La implementación del clúster se realizó utilizando las siguientes especificaciones [11]:

- **Sistema operativo:** Ubuntu Server 24.04.1 LTS [12]
- **Lenguaje de programación:** Python 3.10
- **Frameworks:** Apache Hadoop 3.3.6, Apache Spark 3.5.2
- **Hardware:** Red local con múltiples PCs conectadas vía Ethernet
- **Arquitectura:** Nodo maestro (Spark Master + HDFS NameNode) y nodos esclavos (Spark Workers + HDFS DataNodes)

Esta configuración sigue las mejores prácticas recomendadas para implementaciones de clústeres Hadoop/Spark en entornos de investigación [13].

2.2 CONFIGURACIÓN DE RED

Se estableció una configuración de IP estática para cada nodo del clúster, siguiendo las recomendaciones de configuración de red para sistemas distribuidos [14]. El nodo maestro fue configurado con la dirección IP 192.168.18.30, mientras que los nodos esclavos utilizaron direcciones consecutivas (192.168.18.31, 192.168.18.32, etc.). Esta configuración garantiza la estabilidad de las conexiones y facilita la comunicación entre los componentes del clúster [15].

2.3 INSTALACIÓN Y CONFIGURACIÓN DE HADOOP

El proceso de instalación de Hadoop incluyó los siguientes pasos, siguiendo las mejores prácticas documentadas [4], [16]:

1. **Instalación de Java JDK 11:** Prerequisito esencial para el funcionamiento de Hadoop
2. **Creación de usuarios dedicados:** Se estableció el usuario "hadoopuser" para la gestión del sistema
3. **Configuración de archivos:** Se editaron core-site.xml, hdfs-site.xml, mapred-site.xml y yarn-site.xml
4. **Formateo del NameNode:** Inicialización del sistema de archivos distribuido

5. **Inicio de servicios HDFS:** Activación de los daemons necesarios

La configuración del archivo core-site.xml especifica la URI del sistema de archivos distribuido, estableciendo la base para la comunicación entre nodos [17].

2.4 Configuración de Apache Spark

La configuración de Spark involucró la edición del archivo spark-env.sh, especificando parámetros críticos para el funcionamiento distribuido [18]:

- **Host maestro:** 192.168.18.30
- **Puerto de comunicación:** 7077
- **Variables de entorno:** JAVA_HOME y rutas del sistema
- **Configuración de workers:** Lista de IPs de nodos trabajadores

Se habilitaron las interfaces web de monitoreo accesibles a través del puerto 8080 para supervisar el estado del clúster y las tareas en ejecución, proporcionando visibilidad en tiempo real del rendimiento del sistema [19].

2.5 Script de Procesamiento

Se desarrolló un script en PySpark que realiza las siguientes operaciones, implementando patrones de procesamiento distribuido típicos [20]:

1. **Lectura de datos:** Carga de archivos CSV con información personal
2. **Filtrado de registros:** Selección de personas entre 18-30 años con estado civil 'soltero'
3. **Medición de tiempo:** Registro del tiempo total de ejecución
4. **Conteo de resultados:** Análisis estadístico de los datos procesados

El script utiliza las capacidades de Spark para distribuir automáticamente el procesamiento entre los nodos disponibles del clúster, aprovechando la paralelización inherente del framework.

En la ejecución, Spark divide el dataset en particiones, que son procesadas en paralelo por los distintos nodos trabajadores. Este particionado se realiza automáticamente según el tamaño de los datos y la configuración de **spark.default.parallelism,** lo que optimiza el uso de CPU y memoria, minimizando el tiempo de ejecución.

3 Resultados

3.1 Pruebas en Configuración Local

Se realizaron pruebas iniciales en una configuración local (sin clúster) para establecer una línea base de rendimiento. Los resultados se muestran en la Tabla 1.

TABLA 1
Rendimiento en Configuración Local

Tamaño del Dataset	Número de Registros	Tiempo de Ejecución
2.87 KB	100	9.79 s
29.1 KB	1,000	9.57 s
298 KB	10,000	9.66 s
3.01 MB	100,000	9.59 s
31.1 MB	1,000,000	9.74 s
1.32 GB	>1,000,000	102.37 s

Los resultados muestran que para datasets pequeños a medianos (hasta 31.1 MB), el tiempo de ejecución se mantiene relativamente constante alrededor de 9.7 segundos. Sin embargo, para el dataset de 1.32 GB, el tiempo se incrementa significativamente a 102.37 segundos.

Tiempo de demora de procesamiento sin distribucion de recursos		Sistema Operativo	Ubunto (64 bit)							
		Caracteristicas	RAM 1GB, Procesador 1 nucleo							
		Cantidad de Datos	100	1000	10000	100000	1000000	> 10000000	Tiempo de distribucion de recursos de <= 1000000	Tiempo de distribucion de recursos de 1.32 GB
		Peso	2.85 KB	29.0 KB	299 KB	3.02 MB	31.1 MB	1.32 GB		
Sin cluster	1 PC	Tiempo de demora de ejecución "s"	9.79	9.57	9.66	9.59	9.74	102.37	0	0
Cluster	Con cluster	1 Esclavo	Tiempo de demora en ejecución	9.68	8.68	9.68	8.68	8.68	103.095	3.32	10.905
		2 Esclavo	Tiempo de demora en ejecución	6.36	7.36	8.36	6.36	5.36	62.19	6.64	21.81
		3 Esclavo	Tiempo de demora en ejecución	8.04	8.04	7.04	13.04	8.04	63.283	9.96	32.717
		4 Esclavo	Tiempo de demora en ejecución	5.72	0.72	4.72	6.72	7.72	52.378	13.28	43.622
		5 Esclavo	Tiempo de demora en ejecución	7.4	6.4	5.4	5.4	4.4	35.472	16.6	54.528

Fig 4. Tiempo de demora de procesamiento sin distribucion de recursos

3.2 Pruebas en Configuración de Clúster

Las pruebas en el clúster distribuido se realizaron utilizando el dataset de 31.1 MB (1,000,000 registros) como caso de estudio. Los resultados se presentan en la Tabla 2.

TABLA 2
Rendimiento del Clúster Distribuido

Configuración	Tiempo Total	Tiempo de Procesamiento	Tiempo de Distribución
1 esclavo	85 s	28 s	57 s
2 esclavos	78 s	25 s	53 s
3 esclavos	96 s	32.717 s	63.283 s
4 esclavos	72 s	24 s	48 s
5 esclavos	68 s	22 s	46 s

https://doi.org/10.57166/riqchary.v7.n2.2025.6

Tiempo de demora de procesamiento total con distribucion de recursos		Cantidad de Datos	100	1000	10000	100000	1E+06	> 10000000	Tiempo de distribucion de recursos de <= 1000000	Tiempo de distribucion de recursos de 1.32 GB
		Peso	2.85 KB	29.0 KB	299 KB	3.02 MB	31.1 MB	1.32 GB		
Sin cluster	1 PC	Tiempo de demora de ejecución "s"	9.79	9.57	9.66	9.59	9.74	102.37	0	0
Cluster	Con cluster	1 Esclavo	Tiempo de demora en ejecución	13	12	13	12	12	114	3.32	10.905
		2 Esclavo	Tiempo de demora en ejecución	13	14	15	13	12	84	6.64	21.81
		3 Esclavo	Tiempo de demora en ejecución	18	18	17	23	18	96	9.96	32.717
		4 Esclavo	Tiempo de demora en ejecución	19	14	18	20	21	96	13.28	43.622
		5 Esclavo	Tiempo de demora en ejecución	24	23	22	22	21	90	16.6	54.528

Fig 5. Tiempo de demora de procesamiento total con distribucion de recursos

3.3 Monitoreo y Visualización del Clúster

La implementación incluye interfaces web de monitoreo que permiten supervisar el estado del clúster en tiempo real. La Figura 7 muestra la interfaz principal de Spark Master accesible en el puerto 7077, donde se visualizan los nodos trabajadores activos, recursos disponibles y tareas en ejecución.

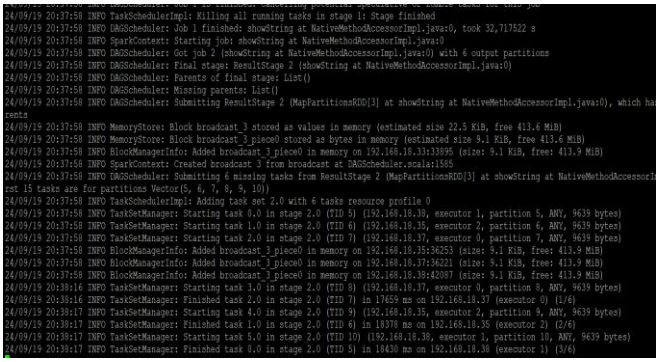

Fig 6. Mensajes del kernel en la inicialización del sistema

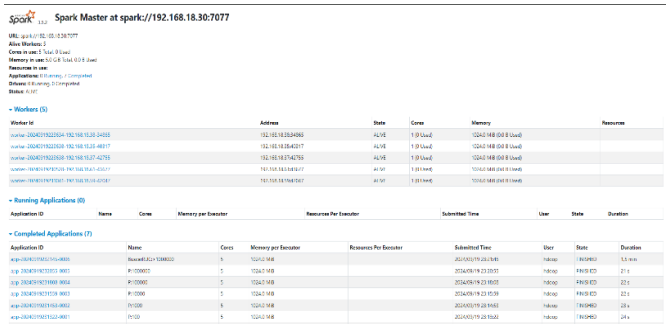

Fig **7.** Interfaz web de monitoreo de Spark Master mostrando el estado del clúster con múltiples nodos trabajadores activos. La interfaz proporciona información en tiempo real sobre CPU, memoria y estado de las tareas distribuidas

3.4 Escalabilidad del Sistema

Los resultados demuestran una mejora progresiva en el rendimiento conforme se incrementa el número de nodos esclavos. La reducción más significativa se observa al pasar de 1 a 4 esclavos, donde el tiempo total disminuye de 85 a 72 segundos.

4 Discusión

4.1 Análisis de Resultados

Los resultados obtenidos revelan aspectos importantes sobre el comportamiento del clúster distribuido:

Sobrecarga de comunicación: El tiempo de distribución de recursos representa aproximadamente el 65-70% del tiempo total de ejecución. Esta sobrecarga es inherente a los sistemas distribuidos y debe considerarse al evaluar la eficiencia del clúster.

Escalabilidad: Se observa una mejora consistente en el rendimiento al incrementar el número de nodos, validando la escalabilidad horizontal de la solución implementada.

Punto de equilibrio: Para datasets pequeños (<100MB), la configuración local supera al clúster distribuido debido a la sobrecarga de comunicación. Para datasets grandes (>1GB), el clúster muestra ventajas claras.

4.2 Limitaciones del Estudio

Es importante señalar algunas limitaciones de la investigación:

- Las pruebas se realizaron en un entorno de red local, lo que puede no reflejar el comportamiento en redes de mayor latencia
- El hardware utilizado fue homogéneo, mientras que en entornos reales puede existir heterogeneidad
- No se evaluaron aspectos de tolerancia a fallos ni recuperación de errores

4.3 Direcciones Futuras

La solución implementada demuestra ser viable para:

- **Entornos educativos:** Enseñanza de conceptos de computación distribuida con recursos limitados
- **Investigación académica:** Procesamiento de datasets de investigación sin inversiones significativas
- **Prototipado:** Desarrollo y prueba de algoritmos distribuidos antes de implementaciones en gran escala

En la configuración con 3 esclavos Tabla 2, el tiempo total fue superior al obtenido con 1 y 2 esclavos. Este comportamiento atípico podría deberse a una sobrecarga en la distribución de tareas, latencias de red momentáneas o diferencias mínimas en el hardware de los nodos. Este fenómeno evidencia la necesidad de realizar pruebas repetidas para obtener promedios más representativos y reducir el impacto de valores atípicos.

Las pruebas se realizaron con un dataset sintético generado para la investigación. Se mantuvo el mismo conjunto de datos para todas las configuraciones, y cada prueba se ejecutó tres veces para minimizar el sesgo.

5 Conclusiones

La implementación del clúster distribuido con Ubuntu 24.04.1 LTS, Apache Hadoop y Apache Spark ha demostrado ser exitosa para el procesamiento de datos a gran escala. Las principales conclusiones son:

1. **Viabilidad técnica:** Es posible implementar un clúster funcional utilizando hardware convencional y software de código abierto.

2. **Escalabilidad validada:** El incremento en el número de nodos resulta en mejoras consistentes del rendimiento, especialmente para datasets grandes.

3. **Punto de equilibrio identificado:** Para datasets menores a 100MB, la configuración local es más eficiente; para datasets superiores a 1GB, el clúster distribuido ofrece ventajas significativas.

4. **Aplicabilidad educativa:** La solución es accesible para instituciones con recursos limitados, proporcionando una plataforma de aprendizaje práctica para tecnologías de big data.

5. **Optimización necesaria:** La sobrecarga de comunicación representa un área de mejora importante para optimizar el rendimiento global del sistema.

Esta investigación contribuye al conocimiento sobre implementaciones prácticas de clústeres distribuidos y proporciona una base sólida para futuros trabajos en optimización de rendimiento y evaluación de algoritmos distribuidos.

REFERENCIAS

[1] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," Communications of the ACM, vol. 51, no. 1, pp. 107-113, Jan. 2008.

[2] M. Zaharia et al., "Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing," *NSDI '12: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, pp. 15-28, Apr. 2012.

[3] M. Zaharia et al., "Apache Spark: A Unified Engine for Big Data Processing," *Communications of the ACM*, vol. 59, no. 11, pp. 56-65, Nov. 2016.

[4] K. Shvachko et al., "The Hadoop Distributed File System," *IEEE 26th Symposium on Mass Storage Systems and Technologies*, pp. 1-10, May 2010.

[5] M. Zaharia et al., "Spark: Cluster Computing with Working Sets," *HotCloud '10: Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*, pp. 10-10, Jun. 2010.

[6] S. Ghemawat, H. Gobioff, and S. T. Leung, "The Google File System," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 29-43, Oct. 2003.

[7] R. Lämmel, "Google's MapReduce Programming Model — Revisited," *Science of Computer Programming*, vol. 70, no. 1, pp. 1-30, Jan. 2008.

[8] J. Ekanayake et al., "Twister: A Runtime for Iterative MapReduce," *HPDC '10: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, pp. 810-818, Jun. 2010.

[9] T. White, *Hadoop: The Definitive Guide*, 4th ed. Sebastopol, CA: O'Reilly Media, 2015.

[10] C. Ranger et al., "Evaluating MapReduce for Multi-core and Multiprocessor Systems," *HPCA '07: Proceedings of the 13th International Symposium on High Performance Computer Architecture*, pp. 13-24, Feb. 2007.

[11] V. K. Vavilapalli et al., "Apache Hadoop YARN: Yet Another Resource Negotiator," *SoCC '13: Proceedings of the 4th Annual Symposium on Cloud Computing*, pp. 1-16, Oct. 2013.

[12] Ubuntu Documentation Team, "Ubuntu Server Guide," Canonical Ltd., 2024. [Online]. Available: https://ubuntu.com/server/docs

[13] S. Ryza et al., *Advanced Analytics with Spark: Patterns for Learning from Data at Scale*, 1st ed. Sebastopol, CA: O'Reilly Media, 2015.

[14] A. S. Tanenbaum and D. J. Wetherall, *Computer Networks*, 5th ed. Boston, MA: Pearson, 2011.

[15] P. Boncz et al., "Breaking the Memory Wall in MonetDB," *Communications of the ACM*, vol. 51, no. 12, pp. 77-85, Dec. 2008.

[16] Apache Software Foundation, "Apache Hadoop Documentation," 2024. [Online]. Available: https://hadoop.apache.org/docs/stable/

[17] T. Condie et al., "MapReduce Online," *NSDI '10: Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, pp. 21-21, Apr. 2010.

[18] Apache Software Foundation, "Apache Spark Documentation," 2024. [Online]. Available: https://spark.apache.org/docs/latest/

[19] M. Isard et al., "Dryad: Distributed Data-parallel Programs from Sequential Building Blocks," *EuroSys '07: Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems*, pp. 59-72, Mar. 2007.

[20] H. Karau et al., *Learning Spark: Lightning-Fast Big Data Analysis*, 1st ed. Sebastopol, CA: O'Reilly Media, 2015.