# spring-batch-db-cluster-partitioning: Database-driven clustering with heartbeats and failover for Spring Batch

**Janardhan Reddy Chejarla** [1]

**1** Independent Researcher

**DOI:** 10.xxxxxx/draft

**Software**

- Review
- Repository
- Archive

**Editor:**

**Submitted:** 23 August 2025
**Published:** unpublished



## Summary

`spring-batch-db-cluster-partitioning` is an open-source extension for Spring Batch (*Spring Batch Project*, 2025). It introduces **database-driven clustering** that provides **node heartbeats, partition lifecycle tracking, and automatic failover** using a shared relational database. The framework enables **horizontal scale-out and fault tolerance** for partitioned jobs while remaining fully compatible with the Spring Batch programming model.

## Statement of Need

Spring Batch supports local parallelism (*Spring Batch*, 2025) and remote partitioning through a messaging layer (*Remote Partitioning with Spring Batch and Spring Integration*, 2025). While effective, these modes require either execution within a single JVM or the addition of messaging infrastructure for coordination.
This framework instead uses the relational database itself as the coordination plane.
Nodes register with heartbeats, so the set of available workers is explicit and continuously visible, simplifying operations while improving **reproducibility, observability, and resilience**.

Typical use cases include **ETL (Extract–Transform–Load) workflows** where large datasets must be cleaned, transformed, or aggregated in parallel. By coordinating work through the database, this framework allows such pipelines to be executed across multiple machines in a cluster without requiring brokers or external schedulers.

## Functionality

The framework provides:

- **Cluster node registry**: active nodes register in a `BATCH_NODES` table with periodic heartbeats.
- **Two-phase liveness detection**: nodes are first marked `UNREACHABLE`, then removed after a cleanup threshold, reducing false positives.
- **Partition lifecycle management**: partitions tracked in `BATCH_PARTITIONS` with states `PENDING`, `CLAIMED`, `COMPLETED`, `FAILED`.
- **Automatic failover**: transferable partitions from failed nodes are reassigned to healthy ones.
- **Distribution strategies**: round-robin, fixed-node, and dynamic assignment.
- **Spring-native integration**: no changes required to job or step definitions.

## Architecture

Coordination state is persisted in three relational tables:

- **`BATCH_NODES`** records active cluster nodes and their heartbeats.
- **`BATCH_PARTITIONS`** tracks partition ownership and lifecycle transitions (pending, claimed, completed, failed).
- **`BATCH_JOB_COORDINATION`** manages per-step coordination metadata within a job.

These tables collectively enable liveness detection, partition assignment, and safe failover. Full DDL schemas and examples are provided in the project repository.

## Failure handling

- **Heartbeats** ensure each node regularly updates its liveness.
- **Two-phase detection** separates *marking unreachable* from *deletion and reassignment*.
- **Transactional partition claims** prevent double execution.
- **Automatic recovery** reassigns incomplete partitions from failed nodes to available ones.
- **Idempotent transitions** allow safe retries without duplicate processing.

## Configuration

Example Spring Boot properties:

```
spring.batch.cluster.enabled=true
spring.batch.cluster.node-id=${HOSTNAME:my-node-01}
spring.batch.cluster.heartbeat-interval=3000
spring.batch.cluster.task-polling-interval=1000
spring.batch.cluster.unreachable-node-threshold=15000
spring.batch.cluster.node-cleanup-threshold=60000
```

## State of the field

Scientific workflow systems such as Pegasus (Deelman et al., 2015), Kepler (Ludäscher et al., 2006), and Taverna (Wolstencroft et al., 2013) emphasize reproducibility and durable state. Spring Batch provides a lightweight framework for batch workloads within the Java ecosystem (*Spring Batch Project*, 2025).
This extension adds **cluster coordination and failover** directly within Spring Batch, using only a relational database.
It complements external schedulers (e.g., Spring Cloud Data Flow (*Spring Cloud Data Flow*, 2025)) and coordination services (e.g., ZooKeeper (Hunt et al., 2010), Chubby (Burrows, 2006), Raft (Ongaro & Ousterhout, 2014)) by embedding coordination at the framework level.

## Limitations

- Targets **small-to-medium clusters** where database throughput is sufficient.
- Relies on a highly available database; replication and backup are required in production.
- Not optimized for ultra-large clusters or sub-second coordination latency, where external services may be more appropriate.

The Journal of Open Source Software

# Diagram

```
flowchart TD
    A[Job Launcher Node] -->|Partitions| B[(Database)]
    B --> C[Worker Node 1]
    B --> D[Worker Node 2]
    B --> E[Worker Node N]
    C -->|Heartbeat + State Updates| B
    D -->|Heartbeat + State Updates| B
    E -->|Heartbeat + State Updates| B
    B -->|Reassign on Failure| C
    B -->|Reassign on Failure| D
```


# Acknowledgements

Built on Spring Batch / Spring Boot.
Source code: (Chejarla, 2025).


# References


Burrows, M. (2006). The chubby lock service for loosely-coupled distributed systems. *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. https://research.google/pubs/pub27897/

Chejarla, J. R. (2025). *Spring-batch-db-cluster-partitioning*. https://github.com/jchejarla/spring-batch-db-cluster-partitioning.

Deelman, E., Vahi, K., Juve, G., Rynge, M., Callaghan, S., Maechling, P. J., Mayani, R., Chen, W., Ferreira da Silva, R., Livny, M., & Wenger, K. (2015). Pegasus: A workflow management system for science automation. *Future Generation Computer Systems*, *46*, 17–35. https://doi.org/10.1016/j.future.2014.10.008

Hunt, P., Konar, M., Junqueira, F. P., & Reed, B. (2010). ZooKeeper: Wait-free coordination for internet-scale systems. *USENIX Annual Technical Conference (ATC)*. https://www.usenix.org/legacy/event/atc10/tech/full_papers/Hunt.pdf

Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E. A., Tao, J., & Zhao, Y. (2006). Scientific workflow management and the kepler system. *Concurrency and Computation: Practice and Experience*, *18*(10), 1039–1065. https://doi.org/10.1002/cpe.994

Ongaro, D., & Ousterhout, J. (2014). In search of an understandable consensus algorithm (raft). *USENIX Annual Technical Conference (ATC)*. https://www.usenix.org/system/files/conference/atc14/atc14-paper-ongaro.pdf

*Remote partitioning with spring batch and spring integration*. (2025). https://docs.spring.io/spring-batch/reference/partitioning.html.

*Spring batch project*. (2025). https://spring.io/projects/spring-batch.

*Spring batch: Scaling and parallel processing*. (2025). https://docs.spring.io/spring-batch/reference/scalability.html.

*Spring cloud data flow*. (2025). https://dataflow.spring.io/.

Wolstencroft, K., Haines, R., Fellows, D., Williams, A., Withers, D., Owen, S., Soiland-Reyes, S., Dunlop, I., Nenadic, A., Fisher, P., Bhagat, J., Belhajjame, K., Hardisty, A., Hidalga, A. N. de la, Vargas, M. R., & Goble, C. (2013). The taverna workflow suite: Designing and executing workflows of web services on the desktop, web or in the cloud. *Nucleic Acids Research*, *41*(W1), W557–W561. https://doi.org/10.1093/nar/gkt328