MDSplus Redis Based Distributed Dispatcher for ITER Neutral Beam Test Facility

Nuno Cruz[a,b], Gabriele Manduchi[a],Cesare Taliercio[a], Joshua Stillerman[c]

[a] Consorzio RFX, Corso Stati Uniti, 4, 35127 Padova Italy
[b] Instituto de Plasmas e Fusão Nuclear, Instituto Superior Técnico, Universidade de Lisboa, 1049-001, Lisboa, Portugal
[c] MIT Plasma Science and Fusion Center, Cambridge, MA, United States of America

Abstract - The ITER Neutral Beam Test Facility (NBTF) supports the development and validation of the neutral beam injection systems required for ITER. Its two major experiments, SPIDER and MITICA, operate complex infrastructures that demand reliable coordination across many distributed subsystems. Recent SPIDER campaigns, following the 2024 restart with caesium-assisted operation, demonstrated improvements in beam uniformity and current density with upgraded RF generators and enhanced pumping. In parallel, MITICA advanced toward integrated megavolt-level operation. These activities highlight the need for a robust dispatching framework capable of managing tightly coupled tasks under real-time constraints.

To address the limitations of the earlier centralized dispatcher, a new MDSplus Redis-based architecture has been developed. The system introduces a distributed model composed of a central Action Dispatcher and multiple Action Servers organized by function. Redis, used as both message broker and shared state repository, provides low-latency task queues, publish/subscribe communication, and persistent state recovery. This design improves scalability, modularity, and fault tolerance, while enabling parallel execution of independent actions and rapid system-wide synchronization.

A Python-native Web Monitor complements the dispatcher by offering real-time views of server activity, action progression, logging streams, and heartbeat diagnostics. Implemented with Flask and Gunicorn, it ensures alignment with the Redis-based backend and reduces integration complexity compared to earlier multi-language solutions. The new system has been validated in full discharge cycles in both SPIDER and MITICA and is now being adopted by external laboratories, demonstrating its suitability for large-scale, distributed fusion experiments.

Keywords— Distributed System, Dispatcher, MDSplus, Redis, Web Monitoring, ITER, Neutral Beam Test Facility.

I. Introduction

The ITER Neutral Beam Test Facility (NBTF) develops and validates neutral beam injection (NBI) systems for ITER. SPIDER, the prototype negative ion source, and MITICA, the full-scale 1 MeV injector under commissioning, demand a robust distributed control and data acquisition environment [1]. Following early operations and a major shutdown, SPIDER resumed in 2024 with caesium-assisted negative ion production. The 2025 campaign demonstrated improved beam uniformity and current density using solid-state RF generators and enhanced pumping, while MITICA progressed toward integrated high-voltage operation. These complex experiments require tight synchronization across many subsystems and reliable orchestration of actions [2] [3].

This document summarizes a new Redis-based MDSplus Action Dispatcher and its Python-native Web Monitor, designed to replace a centralized dispatcher with a scalable, fault-tolerant architecture suited for real-time operations in SPIDER and MITICA.

II. MDSplus Redis Dispatcher

Distributed control systems need coordination of multiple compute nodes and subsystems that must work synchronously and reliably (e.g., SPIDER/MITICA environments using MDSplus) [4] [5]. A dispatcher is the central brain and organizer that assigns all the work, enforces sequencing, and maintains system stability across servers [6].

The Development of a new MDSplus Redis [7] action dispatcher for SPIDER and MITICA, and applicable to other fusion facilities using MDSplus, focused on improving scalability and modularity by introducing a distributed, dispatch-less MDSplus Redis Action Server [8] [9] [10]. This new architecture addressed the limitations of the former dispatcher, particularly its reduced robustness and difficulty in managing multiple servers when faults occur. The initial architecture design evolved to a more centralized task coordination in order to tackle experimental drawbacks, improving reliability and performance.

III. Using Redis In Memory Database

The action dispatcher is a key component of SPIDER and MITICA distributed control systems, as it coordinates resources and tasks across multiple subsystems [11]. It aims at managing resource assignment, scheduling and prioritizing operations, and balancing workloads to ensure efficient system performance. By monitoring system status in real time, and handling different workers the dispatcher shall respond to changing conditions and handle faults by reallocating tasks or isolating failures, thereby improving reliability and stability.

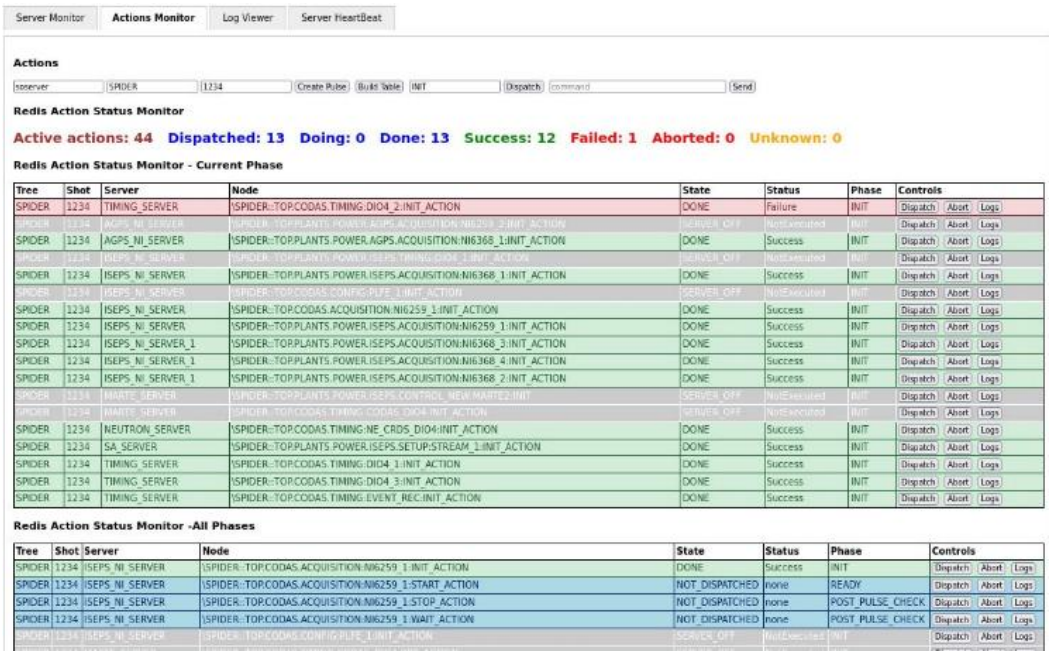

Figure 1 – The Actions Monitor tab

The new MDSplus dispatcher implementation is built around Redis, an open-source, in-memory data platform that can operate as a database, cache, and message broker. Redis supports a wide range of data structures, such as strings, hashes, lists, sets, and streams, which can be effectively used to manage task queues, priorities, and state information within the dispatcher.

Its in-memory architecture provides very low latency and high throughput, making it well suited for real-time dispatching where rapid response is required. Redis also includes a native publish/subscribe messaging mechanism that enables efficient, event-driven communication between distributed components of the control system. This allows actions and status updates to be propagated reliably and in near real time.

Scalability may be supported in the future through Redis clustering, which enables horizontal scaling, automatic data partitioning, and handling of many concurrent connections. In addition, Redis offers persistence mechanisms that allow the dispatcher to recover its state after restarts or failures, improving overall system robustness. Combined with its lightweight design and ease of integration through multiple client libraries.

The new MDSplus dispatcher is implemented as a distributed system composed of a single Action Dispatcher and multiple Action Servers, each one is responsible for executing a specific class of actions. Communication between components is completely managed using Redis, which is used both as the coordination agent and as the shared state repository.

IV. Web Monitoring and Control Tool

The MDSplus Redis Dispatcher Web Monitor is a web-based application designed to provide real-time monitoring and control of distributed action execution within the Redis-based dispatcher architecture. It is primarily intended for operators and developers who need to monitor the status of action servers, dispatched actions, and system logs during automated runs.

The main interface is organized into several tabs addressing specific aspect of system supervision. the interface is organized into four primary functional areas: (i) Server Monitoring; (ii) Actions Monitoring; (iii) Log Viewer and (iv) Server HeartBeat.

The Server Monitor gives a real-time view of all registered action servers, showing their active or inactive state and allowing basic controls such as start, restart, or stop. Server entries list their identifiers and ON/OFF status, with the display refreshing automatically and active servers shown first.

The Actions Monitor handles action dispatching, presenting execution status by tree and shot. It provides separate tables for active actions in the current phase and for all remaining defined actions. For each action, the interface displays the tree, shot, server class, node, execution state, status, and phase.

The Log Viewer supports debugging by showing real-time system logs and enabling the injection of test log messages. Additional server-side logs can be redirected to this interface for broader verification.

The Server Heartbeat tab reports the most recent heartbeat timestamps for each server, helping operators identify stalls or failures. This complements the activity information provided by the Server Monitor.

Figure 1 depicts the Actions Monitor tab that provides a detailed view of action execution status.

The WebMonitor is implemented as a Flask application served by Gunicorn, configured with asynchronous

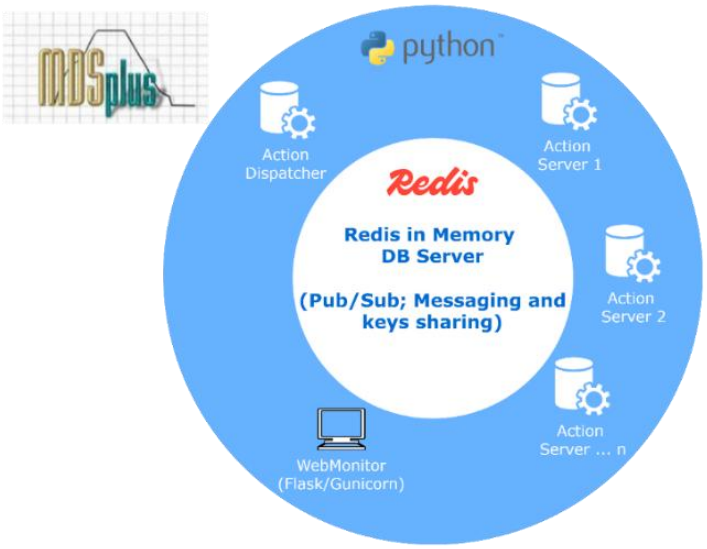

Figure 2 – Simplified Conceptual View of the New MDSplus Redis Dispatcher Architecture

Gevent workers to maintain long-lived connections and provide real-time updates during continuous operation. Deploying the interface within a Python-based environment ensures full compatibility with the dispatcher and action servers, all implemented in the same language. This avoids cross-language integration layers, facilitates direct access to Redis and MDSplus APIs, and enables shared code between the dispatcher and monitoring tools. Gunicorn contributes a stable WSGI runtime with reliable process management and flexible worker models, reducing deployment overhead and simplifying maintenance for I/O-intensive monitoring tasks.

The earlier Node.js/React implementation introduced additional complexity due to the coexistence of two programming ecosystems. Maintaining parallel toolchains, managing rapidly changing JavaScript dependencies, and integrating Node.js services with Python-based Redis and MDSplus components increased development effort and caused repeated compatibility issues. The need for a separate runtime environment also reduced operational uniformity, which the current fully Python-based solution resolves.

V. Conclusions

The new MDSplus dispatcher has been tested during complete discharge cycles in both SPIDER and MITICA. Following a successful demonstration during a SPIDER discharge, the software package is now also being adopted by the Plasma Science and Fusion Center at the Massachusetts Institute of Technology.

Figure 2 depicts a simplified conceptual view of the global architecture of the new dispatcher, where a single Action Dispatcher controls the execution of actions performed by multiple Action Servers (workers) organized into different server classes. The configuration is retrieved from MDSplus, while all dispatcher management and inter-component communication are handled through the Redis in-memory database by the central Action Dispatcher. Actions can be triggered and monitored through the WebMonitor interface.

The entire software stack is developed in a single programming language, Python, which significantly simplifies development, integration, and long-term maintainability of the overall system.

It is relevant to say that this new implementation replaces a highly centralized dispatcher with a more modular, fault-tolerant, and scalable Redis-based architecture suited for complex distributed experiments.

Moreover, the WebMonitor is a very relevant component of the new Redis based dispatcher, providing real-time visibility into the system state, enabling direct operator interaction with the distributed actions and servers. By combining server and actions live monitoring, dispatching control capabilities, logging visualization within a single interface, the dispatcher also improves the synchronization, diagnosis, efficient actuation, and reliable operation of the dispatcher in SPIDER and MITICA complex distributed environments.

Acknowledgment

This work has been carried out within the framework of the EUROfusion Consortium, funded by the European Union via the Euratom Research and Training Programme (Grant Agreement No 101052200 — EUROfusion). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

The work leading to this presentation has been funded partially by ITER under activity work programs AWP2020-2021. This publication reflects the views only of the authors and ITER Organization cannot be held responsible for any use, which may be made of the

information contained therein. The views and opinions expressed herein do not necessarily reflect those of the ITER Organization.

IPFN activities were supported by FCT – Fundação para a Ciência e Tecnologia, I.P. through project UID/50010/2025.

VI. References

[1] V. Toigo, "The ITER Neutral Beam Test Facility," *Nuclear Fusion,* vol. 55(8), p. 083025, 2015.

[2] D. Marcuzzi, V. Toigo, M. Boldrin e et al., "Lessons learned after three years of SPIDER operation and the first MITICA integrated tests," *Fusion Engineering and Design,* vol. 191, p. 113590, 2023.

[3] G. Serianni, E. Sartori e et al., "SPIDER, the Negative Ion Source Prototype for ITER: Overview of Operations and Cesium Injection," *IEEE Transactions on Plasma Science,* vol. 51, pp. 927-935, 2023.

[4] N. Cruz, C. Taliercio, A. Luchetta, G. Manduchi e A. Rigoni, "The SPIDER Pulse Plant Configuration Environment," *IEEE Transactions on Nuclear Science,* vol. 70, pp. 1149-1156, 2023.

[5] A. Luchetta, C. Taliercio, N. Cruz e et al., "As built design of the control systems of the ITER full-size beam source SPIDER in the neutral beam test facility - A critical review," *Fusion Engineering and Design,* vol. 191, p. 113624, 2023.

[6] G. Coulouris, J. Dollimore, T. Kindberg e G. Blair, Distributed Systems: Concepts and Design (5th edition), Boston, MA: Addison-Wesley, 2012.

[7] J. Carlson, Redis in Action, Shelter Island, NY: Manning Publications, 2013.

[8] A. Luchetta, G. Manduchi, A. Barbalace, A. Soppelsa e C. Taliercio, "EPICS – MDSplus integration in the ITER Neutral Beam Test Facility," *Fusion Engineering and Design,* vol. 86, pp. 1252-1255, 2011.

[9] G. Manduchi, A. Rigonii, T. W. Fredian, J. A. Stillerman, A. Neto e F. Sartori, "MARTe2 and MDSplus integration for a comprehensive fast control and data acquisition system," *Fusion Engineering and Design,* vol. 161, p. 111892, 2020.

[10] G. Manduchi, C. Taliercio, A. Luchetta, A. Rigoni, N. Cruz, G. Martini e L. Trevisan, "CODAS for long lasting experiments. The SPIDER experience," *Fusion Engineering and Design,* vol. 190, p. 113497, 2023.

[11] S. D. Bello e et al., "Safety systems in the ITER neutral beam test facility," *Fusion Engineering and Design,* vol. 146, pp. 246-249, 2019.