Comprehensive Plugin-Based Monitoring of Nextflow Workflow Executions

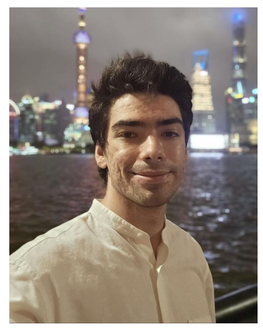
Sami Kharma, Tobias Wies, Florian Schintke

Abstract

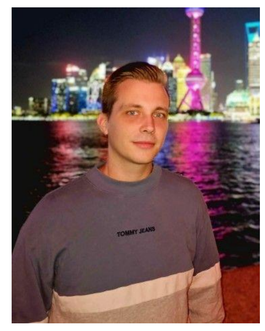
Nextflow [1] is a workflow management system commonly used in fields like bioinformatics [2] and earth observation [3]. It coordinates distributed data processing of various tools as an acyclic sequence of tasks while using, containerization (e.g., Docker), orchestration (e.g., Kubernetes), or batch processing (e.g., SLURM). Monitoring such workflow executions can be challenging but aids **performance analysis, debugging, and data provenance**.

Besides Nexflow's basic built-in monitoring, the wf-commons tool [4] for creating wf-instances is widely regarded as the standard in the Nextflow community. The monitoring plugin we develpoed provides a more **detailed and flexible alternative compatible with wf-instances** while removing the need for a custom Nextflow fork by using Nextflow's plug-in mechanism (version 21.10), optional direct .jar file changes of static artifacts **without recompilation** and allows online monitoring during execution.

Bio



Sami Kharma
Zuse Institute Berlin



Tobias Wies
TU Darmstadt

Florian Schintke
Zuse Institute Berlin

Research Problem

How can **real-time monitoring** of Nextflow workflows be achieved **without custom forks**, while maintaining both **compatibility** with wf-commons standards as well as **portability**?

Background

WfCommons [4]
WfCommons provides **tooling and data** for the scientific data-analysis workflow research community.

Our Goal: Improve on data and tooling.

Nextflow [1]
Nextflow is a **workflow management system** facilitating workflow executions with reproducibility and portability in mind.

Our Goal: Empower users with greater insights into their workflow executions.

Contribution

1. **Monitoring Plugin** collecting workflow execution data, available during execution
2. **Automated wf-instance generation** from workflow executions
3. **Removing the need for recompilation of a custom Nextflow fork** while still maintaining the ability to access the necessary Nextflow internals

Monitoring Overhead

Overhead on Nodes

Source Code from injection.

Cost Insignificant, $\mathcal{O}(1)$ with respect to task runtime, as **only one-time operations** to enumerate Node capabilities are performed.

Overhead on Nextflow

Source Plugin itself.

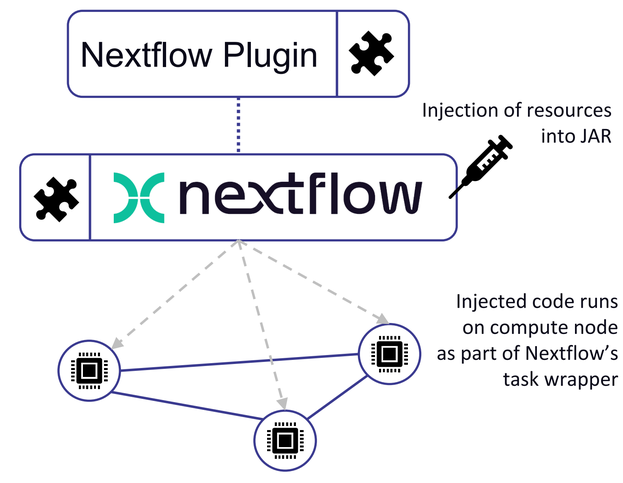
Cost Limited testing has shown a runtime overhead of **approximately 5%**. We believe this can be significantly reduced with further development efforts.

Monitoring Nextflow Executions

Developed a Nextflow Plugin enabling comprehensive full-workflow monitoring **without need for additional privileges** (/proc access allows more granularity).

Features:

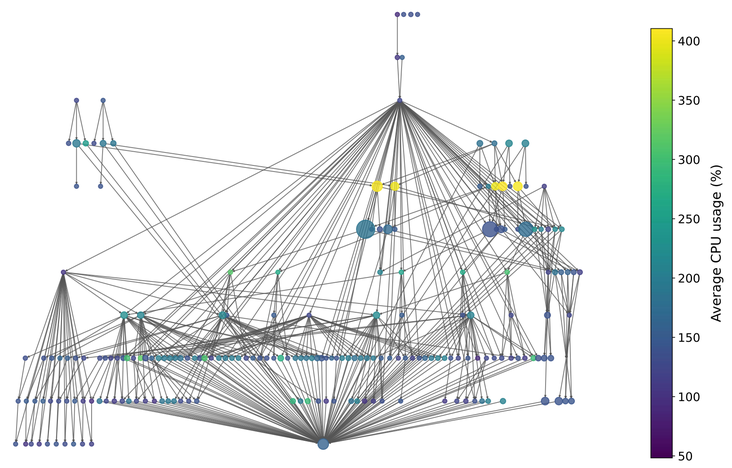
- **Hardware** capability enumeration in distributed infrastructure
- Output **wf-instances** with additional data
- Physical **execution-graph** with **physical nodes**
- **Online monitoring** supported



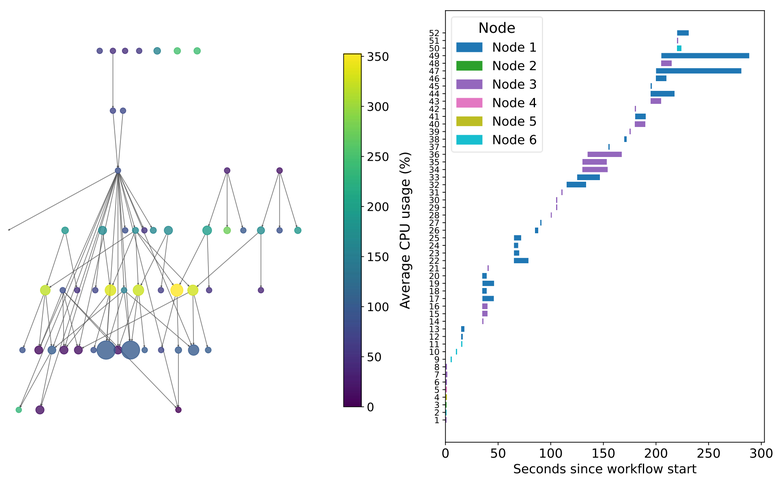
How it works:

1. Plugin registers **callbacks** for certain Nextflow pipeline execution events (i.e., process submit)
2. Nextflow provides **context** to callback
3. Plugin uses Java **reflection** to access Nextflow internals through context
4. Relevant information is gathered; **shared filesystem** is accessed to collect further information (including results of code injection written by each task)

Potential for extension:
Injected code can be easily modified to customize node-level monitoring. I.e., collecting time-series data with tools such as collectl (https://github.com/sharkcz/collectl).



Example task DAG and based on the result wf-instance from a complete execution of the nf-core [6] rnaseq (3.22.1) bioinformatics workflow [5]. Node size is runtime.



Example task DAG and Gantt chart of a partial (failed) execution of the nf-core [6] rnaseq (3.22.1) bioinformatics workflow [5] on distributed infrastructure. Nodes represent tasks, arrows represent dependencies. Size is runtime. The scheduler used by Nextflow was SLURM. Visualized with a Python script using Graphviz and matplotlib, exclusively based on the result wf-instance file.

For each task, available metadata includes:

- Name, id, parents, children, exact command(s)
- input files, output files, file sizes
- runtime, timestamps, CPU load, bytes read/written, memory usage (vmem, rss), context switches
- Node name, OS release, architecture, memory, CPU specifications, boot ID (enables matching nodes despite containerization)
- I/O syscall counts, Nextflow status, work directory, container data, environment variables, etc.

Repository: https://github.com/cookiephone/nf-bigbrother

Future Work

Large-scale Data Collection:

We plan to utilize the plugin in conjunction with other monitoring tooling to collect large amounts of rich workflow execution data.

Further Development:

- Expand **features** as needed
- Improve **ease-of-use**
- Proper specification of **extended wf-instances** for better usability
- **Integration** with other monitoring solutions

Communication/Outreach:

Continue **Open-source** releases of improved versions of the monitoring plugin for Nextflow.

Reach out to **wf-commons** to replace existing wf-instance generation for Nextflow executions.

References

- [1] P. Di Tommaso et al. 2017. Nextflow enables reproducible computational workflows. en. Nat Biotechnol, 35, 4, (Apr. 2017), 316–319.
- [2] B. Van de Sande et al. 2020. A scalable SCENIC workflow for single-cell gene regulatory network analysis. en. Nat Protoc, 15, 7, (June 2020), 2247–2276.
- [3] F. Lehmann et al. 2021. FORCE on Nextflow: scalable analysis of earth observation data on commodity clusters. In vol. 3052. CEUR-WS.org. https://ceur-ws.org/Vol-3052/short12.pdf.
- [4] T. Coleman, H. Casanova, L. Pottier, M. Kaushik, E. Deelman, and R. Ferreira da Silva, "WfCommons: A Framework for Enabling Scientific Workflow Research and Development," Future Generation Computer Systems, vol. 128, pp. 16-27, 2022.
- [5] [SW] H. Patel et al., nf-core/rnaseq: nf-core/rnaseq v3.19.0 - Tungsten Turtleversion 3.19.0, June 2025. doi:10.5281/zenodo.15631172.
- [6] P. A. Ewels et al. 2020. The nf-core framework for community-curated bioinformatics pipelines. Nature Biotech., 38, 3, 276–278. doi:10.1038/s41587-020-0439-x.

Acknowledgements

This work received funding from the German Research Foundation (DFG), CRC 1404:

FONDA: Foundations of Workflows for Large-Scale Scientific Data Analysis