# SPOT Tutorial Slides

2021 ECP Annual Meeting

Matthew LeGendre

April 12, 2021

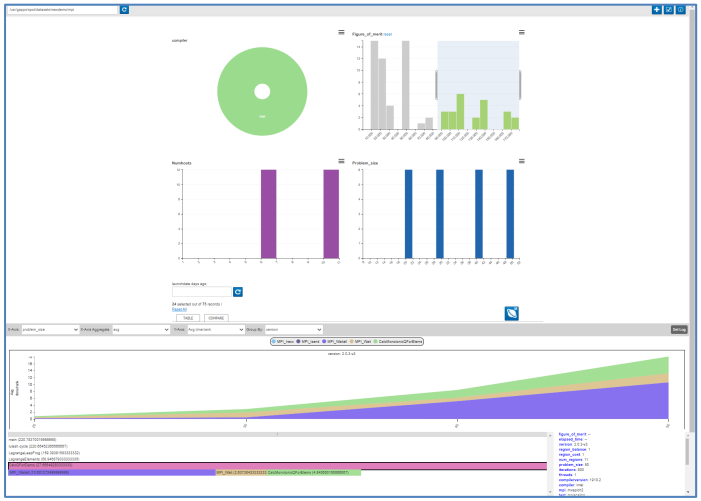
LLNL-PRES-XXXXXX

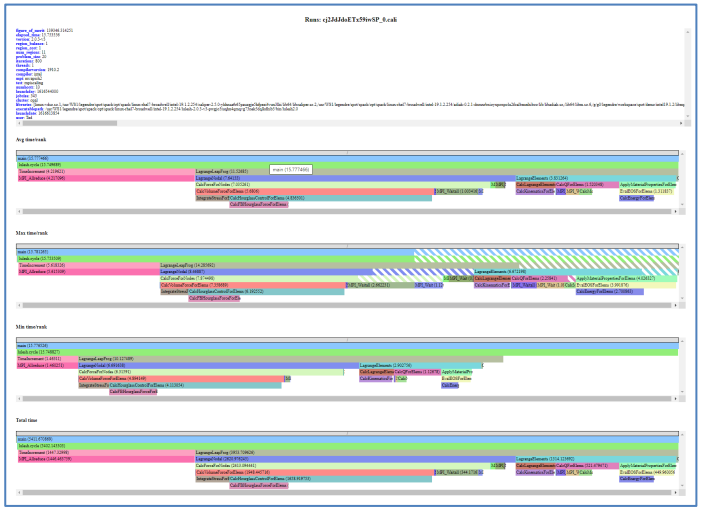
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

# SPOT: Software Performance and Analysis Tracking

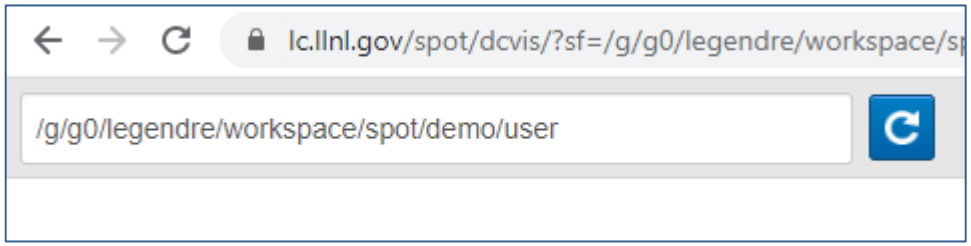
- SPOT is visualization for **collections of performance profiles**. Common uses cases are:
  - Performance comparison of nightly tests. Look for performance regressions.
  - Performance tracking of developer changes. Run an MPI scaling study.
  - Collect performance profiles from users. Understand how users run and hit performance issues.





# SPOT's Workflow

1. Point SPOT webpage at directory containing *.cali files.

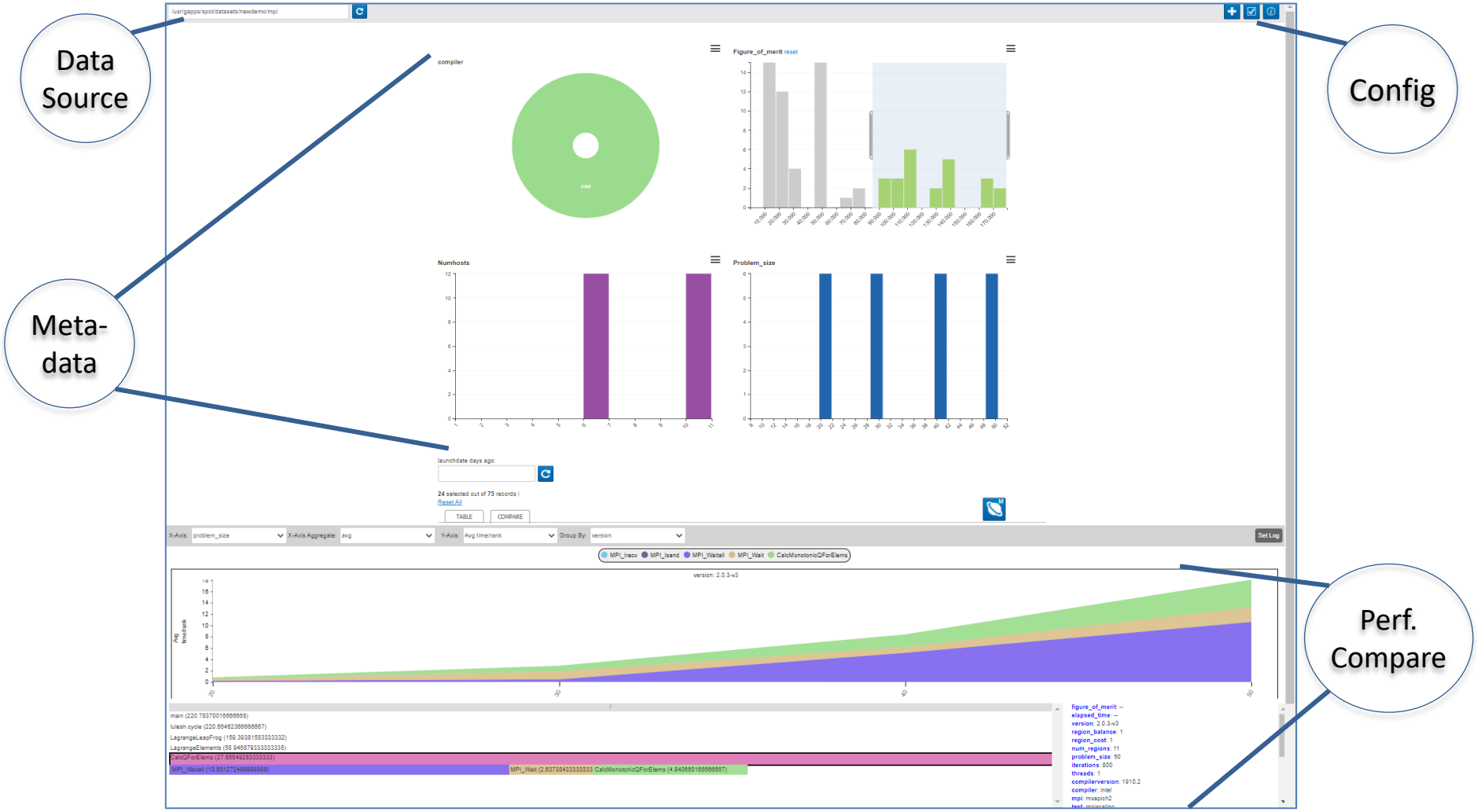



2. Select "interesting" sets of runs to analyze
   - Don't compare performance of 1d test run. vs 3d multi-physics run.

3. Run common analysis with SPOT.
   - i.e., Graph history of performance changes
   - i.e., View performance context trees

4. Run advanced analysis with Hatchet.

# SPOT's Landing Page



# SPOT's Landing Page


Table View

# SPOT is Based on CrossFilter

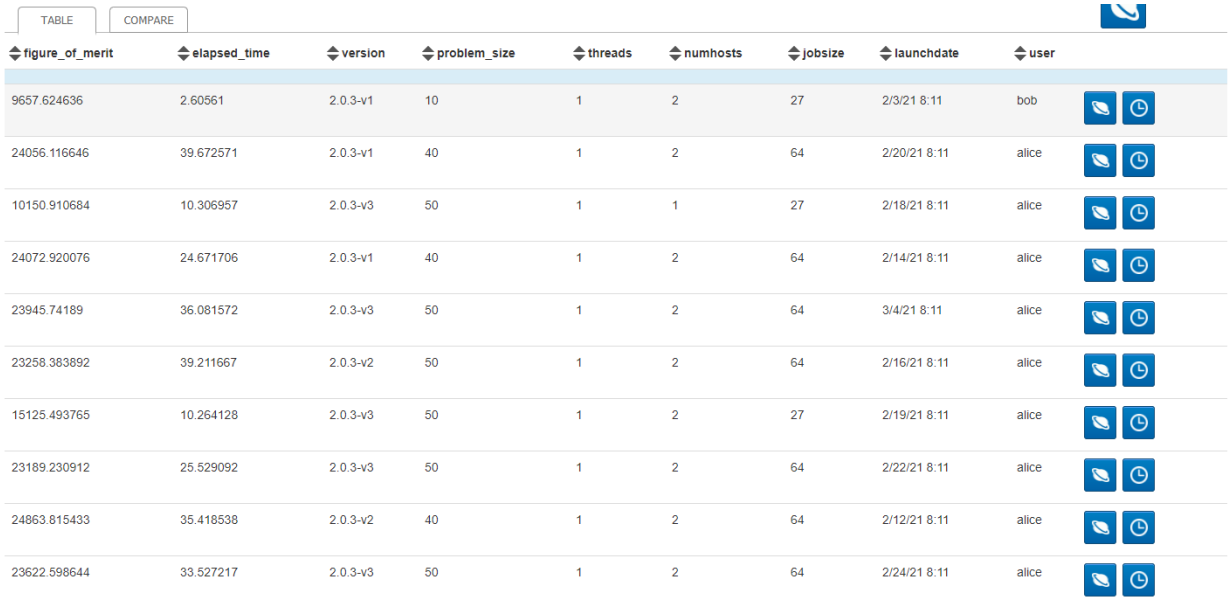


TABLE COMPARE

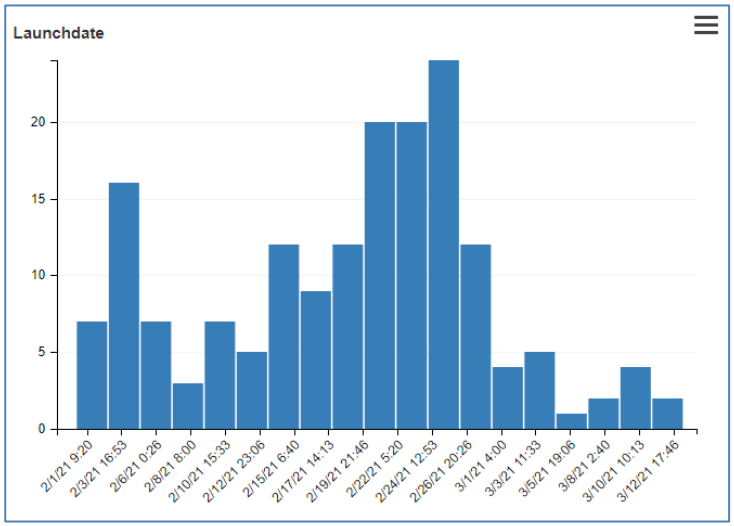
| figure_of_merit | elapsed_time | version | problem_size | threads | numhosts | jobsize | launchdate | user |
|---|---|---|---|---|---|---|---|---|
| 9657.624636 | 2.60561 | 2.0.3-v1 | 10 | 1 | 2 | 27 | 2/3/21 8:11 | bob |
| 24056.116646 | 39.672571 | 2.0.3-v1 | 40 | 1 | 2 | 64 | 2/20/21 8:11 | alice |
| 10150.910684 | 10.306957 | 2.0.3-v3 | 50 | 1 | 1 | 27 | 2/18/21 8:11 | alice |
| 24072.920076 | 24.671706 | 2.0.3-v1 | 40 | 1 | 2 | 64 | 2/14/21 8:11 | alice |
| 23945.74189 | 36.081572 | 2.0.3-v3 | 50 | 1 | 2 | 64 | 3/4/21 8:11 | alice |
| 23258.383892 | 39.211667 | 2.0.3-v2 | 50 | 1 | 2 | 64 | 2/16/21 8:11 | alice |
| 15125.493765 | 10.264128 | 2.0.3-v3 | 50 | 1 | 2 | 27 | 2/19/21 8:11 | alice |
| 23189.230912 | 25.529092 | 2.0.3-v3 | 50 | 1 | 2 | 64 | 2/22/21 8:11 | alice |
| 24863.815433 | 35.418538 | 2.0.3-v2 | 40 | 1 | 2 | 64 | 2/12/21 8:11 | alice |
| 23622.598644 | 33.527217 | 2.0.3-v3 | 50 | 1 | 2 | 64 | 2/24/21 8:11 | alice |

SPOT's Crossfilter: https://dc-js.github.io/dc.js/
D3-Based Crossfilter: https://square.github.io/crossfilter/

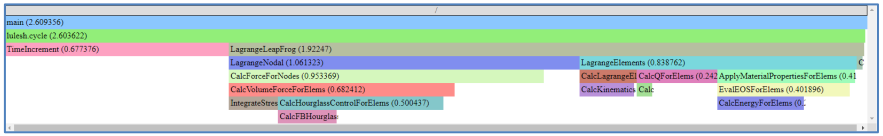
- CrossFilter is good at finding correlations in data.
- Crossfilter helps you manage/slice your data to make interesting comparisons.

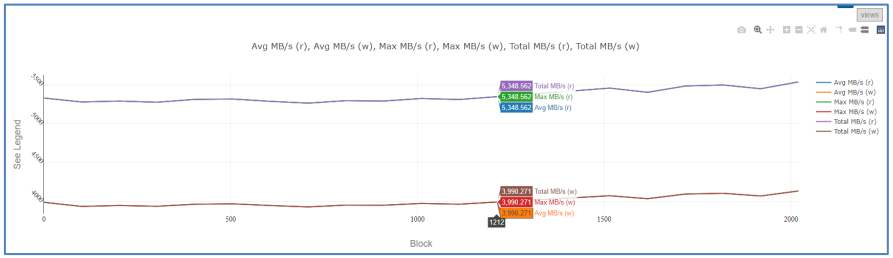
# Each App Run has Two Types of Data

- **Metadata**
  - Name/value data about job.
    - User, host, FOM, input parameters, …
  - Displayed as histograms



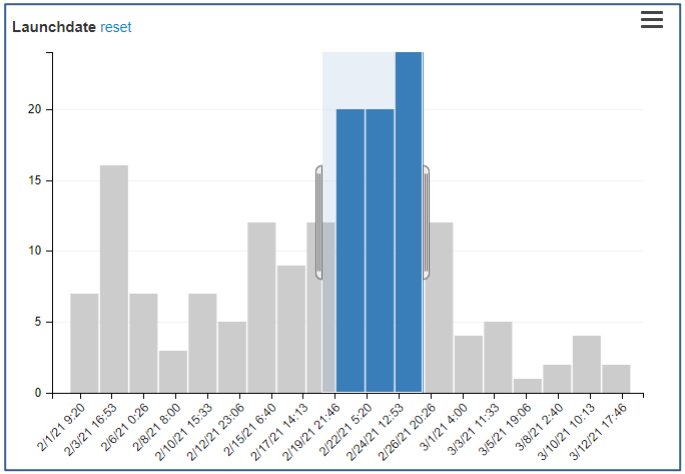

- **Performance Data**
  - Can be different metrics.
    - Avg Time/Rank, Max Time/Rank, bandwidth, …
  - Displayed as a flame graph or timeseries.

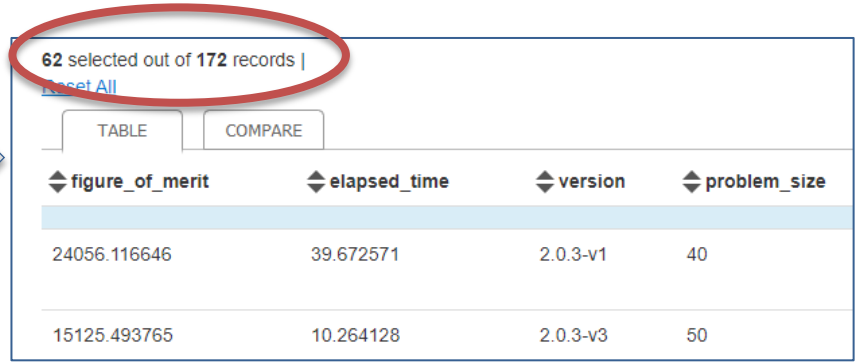


# Metadata Can be Filtered

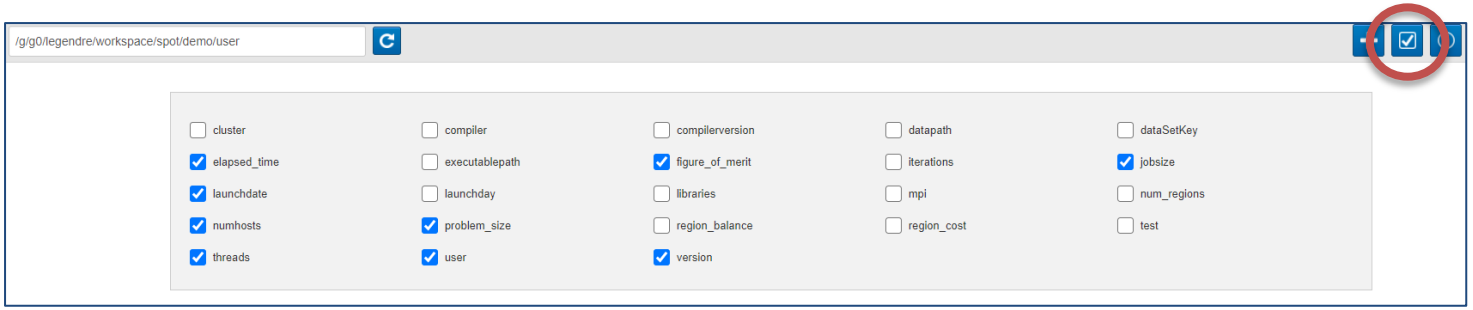
- Select regions of metadata histograms to filter.
  - Filtering in one histogram updates all histograms range.
    - E.g., Filtering Launchdate to ‘Feb 19-Feb26’ will update the ‘Users’ histogram to only show users who ran between those dates.
  - While filtered, comparisons and Jupyter notebooks will operate on the filtered data.



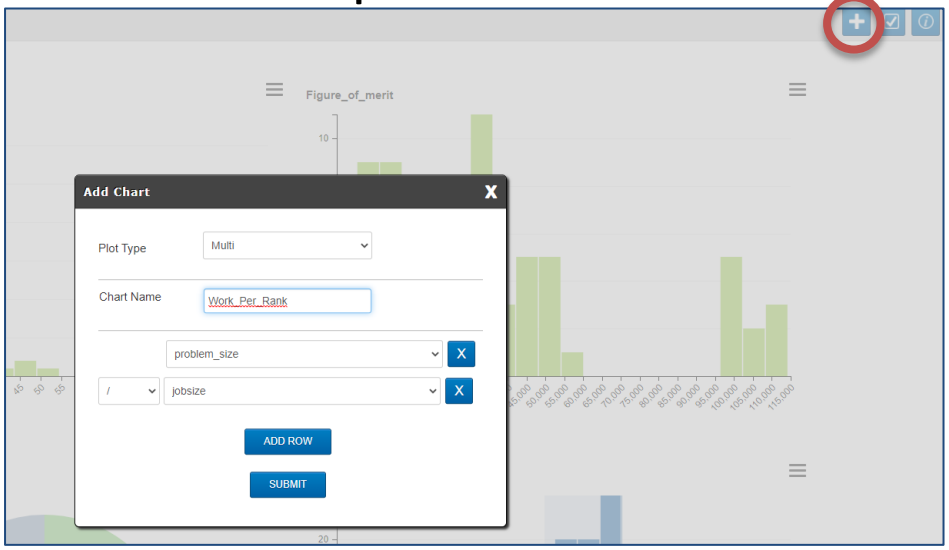



# Metadata is Configurable

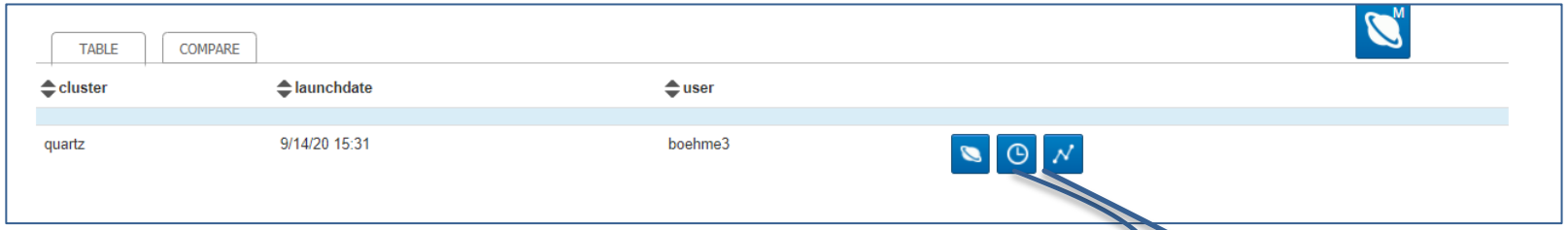
- Can show/hide metadata:



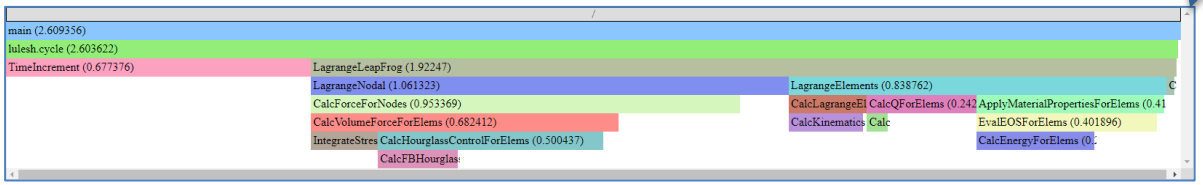

- Can create composite metadata:



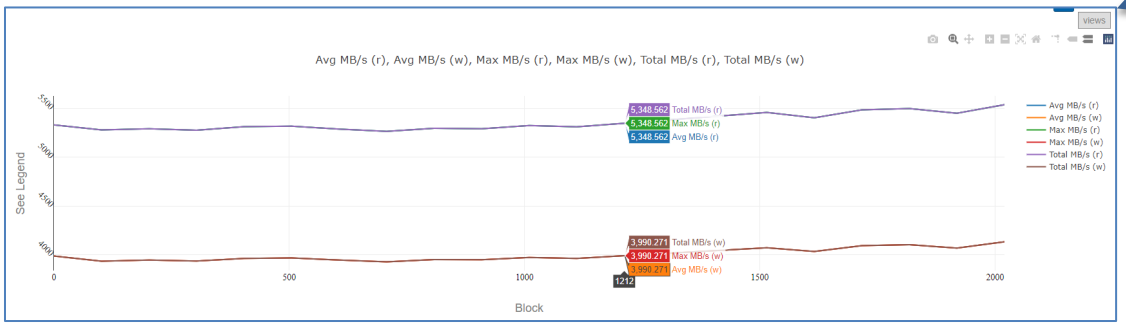
# Performance Data can be Visualized Per-Run



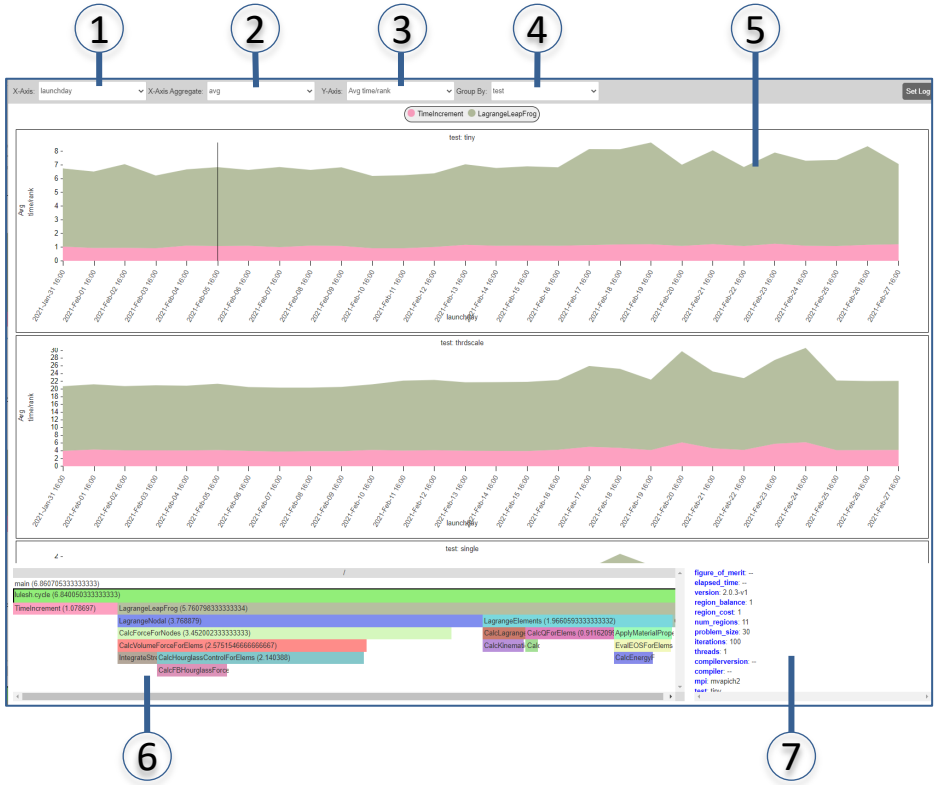

- FlameGraph: performance against code




- Timeseries: performance against time



# Performance Data can be Compared Across Runs




1. Metadata value on x-axis
2. Aggregation (min/max/avg…) if multiple runs at same point.
3. Metric to graph on y-axis.
4. Grouping metadata. Each unique value is a graph.
5. Stacked performance graph. Click to drill-down.
6. Flame graph for runs currently under cursor.
7. Metadata for runs currently under cursor.

# Live Demo