SPOT Tutorial Slides

2021 ECP Annual Meeting

Matthew LeGendre

April 12, 2021

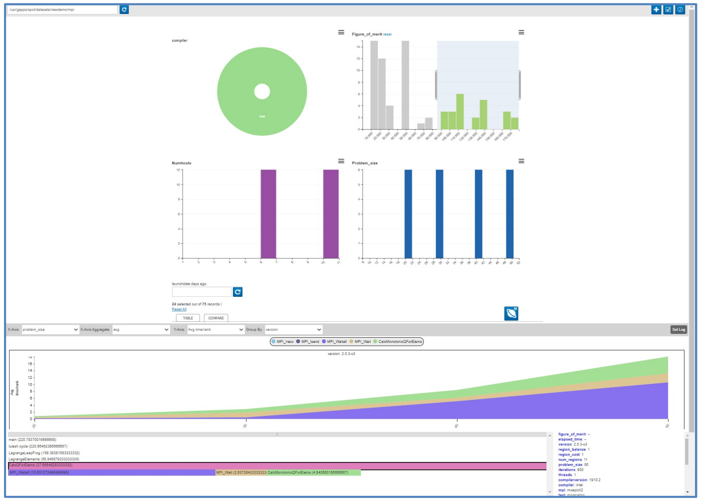
LLNL-PRES-XXXXXX

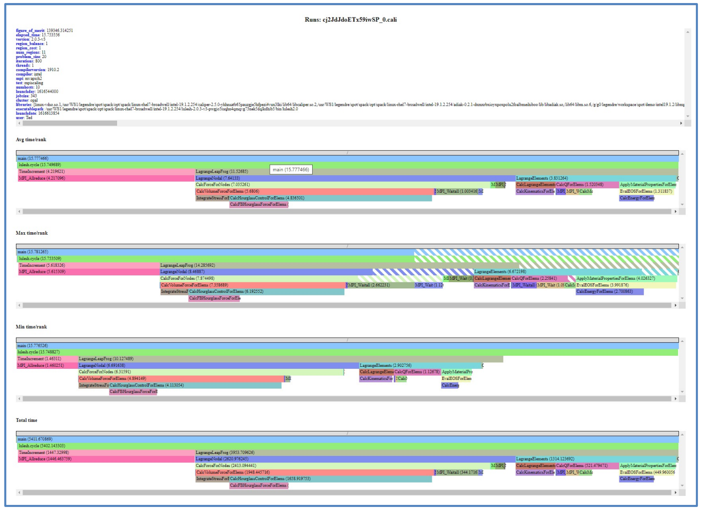
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

SPOT: Software Performance and Analysis Tracking

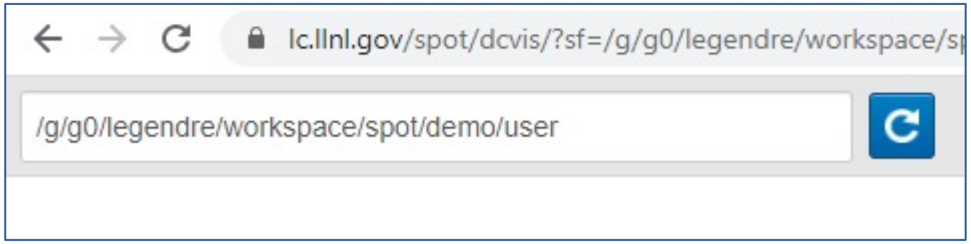
- SPOT is visualization for **collections of performance profiles**. Common uses cases are:
  - Performance comparison of nightly tests. Look for performance regressions.
  - Performance tracking of developer changes. Run an MPI scaling study.
  - Collect performance profiles from users. Understand how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





SPOT's Workflow

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



lc.llnl.gov/spot/dcvis/?sf=/g/g0/legendre/workspace/s

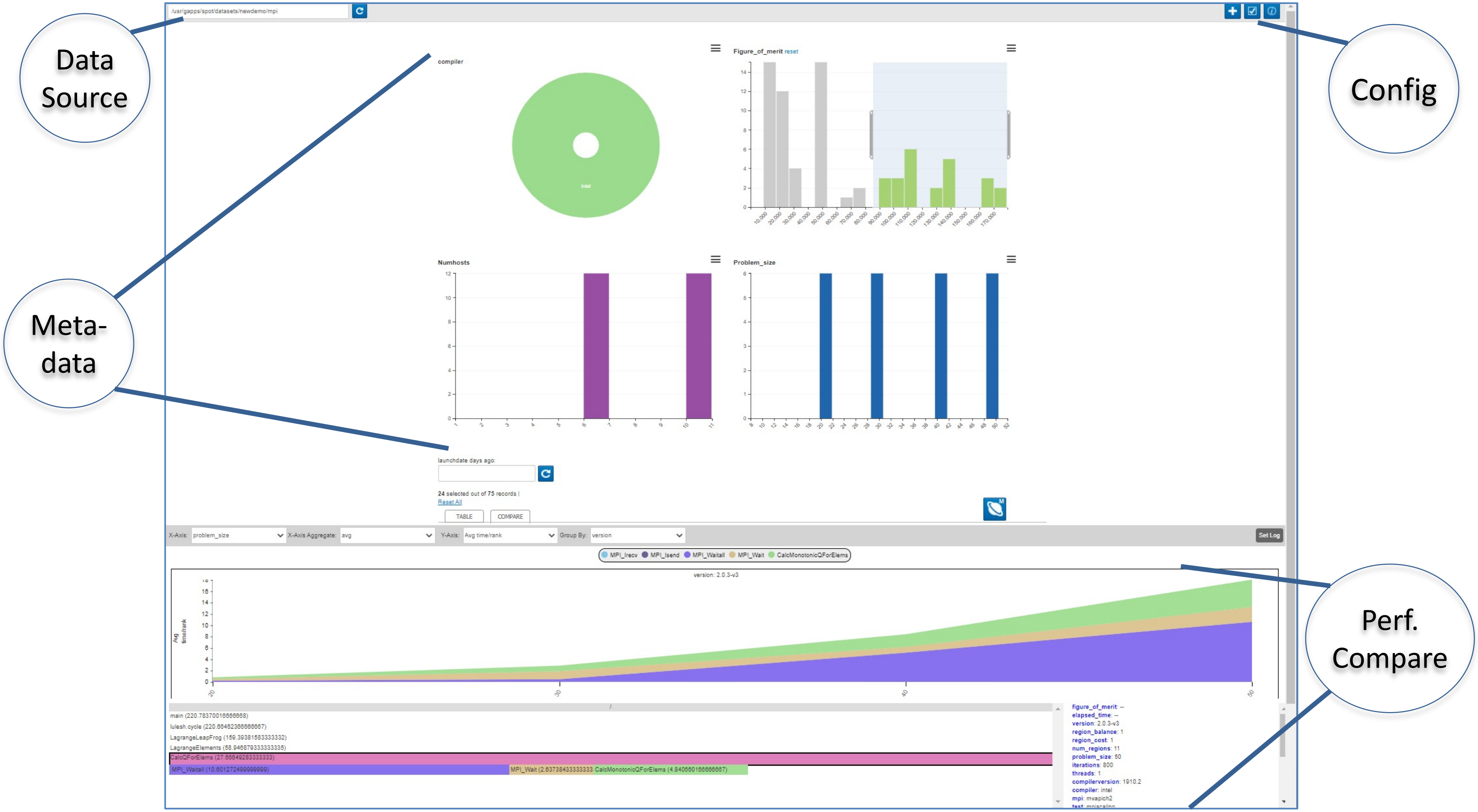
/g/g0/legendre/workspace/spot/demo/user

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



Data Source

Meta-data

Config

Perf. Compare

/usr/gapps/spot/datasets/newdemo/mpi

launchdate days ago:

24 selected out of 75 records |
Reset All

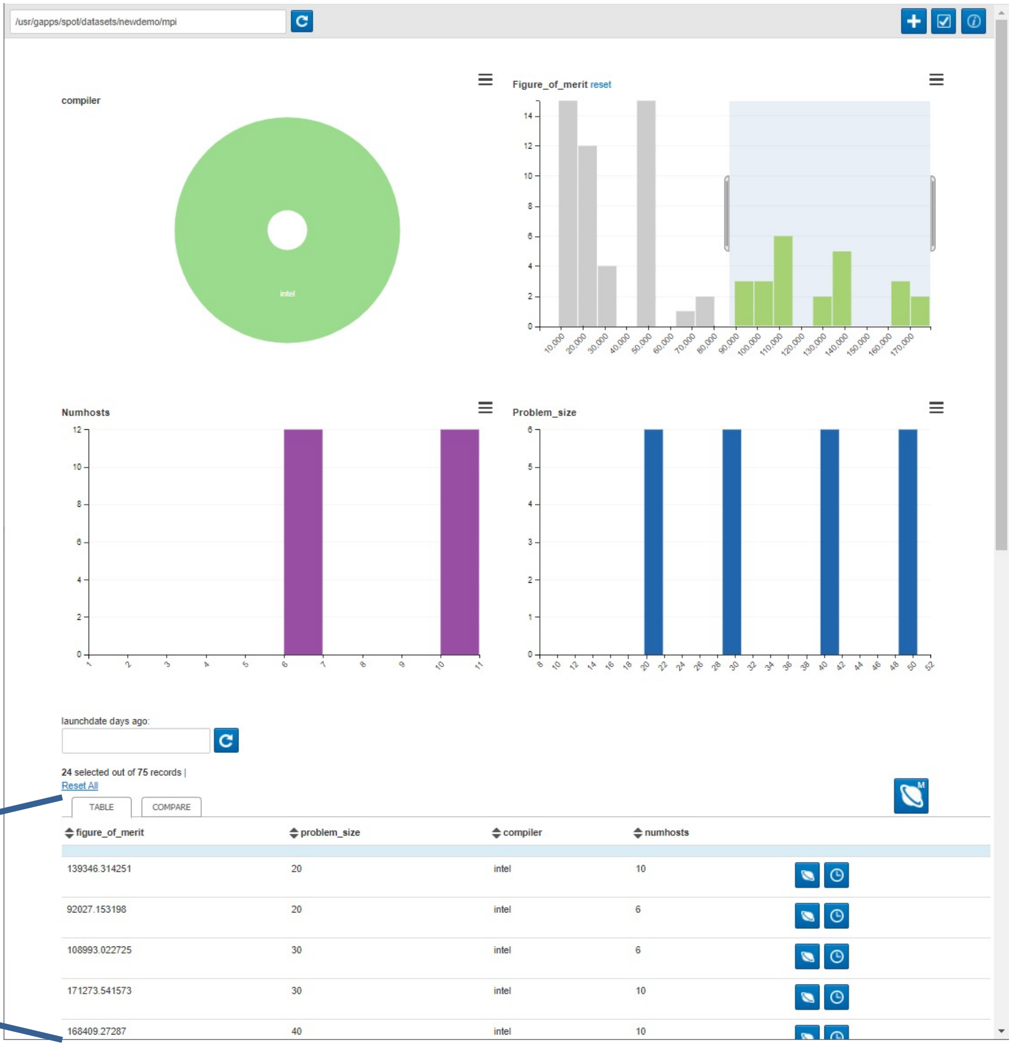
X-Axis: problem_size
X-Axis Aggregate: avg
Y-Axis: Avg time/rank
Group By: version

version: 2.0.3-v3

main (220.78370016666668)
lulesh.cycle (220.66462366666667)
LagrangeLeapFrog (159.39381583333332)
LagrangeElements (58.94687933333335)
CalcQForElems (27.66649283333333)
MPI_Waitall (10.601272499999999)
MPI_Wait (2.63738433333333
CalcMonotonicQForElems (4.84066016666667)

figure_of_merit: --
elapsed_time: --
version: 2.0.3-v3
region_balance: 1
region_cost: 1
num_regions: 11
problem_size: 50
iterations: 800
threads: 1
compilerversion: 1910.2
compiler: intel
mpi: mvapich2
test: mpiscaling

SPOT’s Landing Page

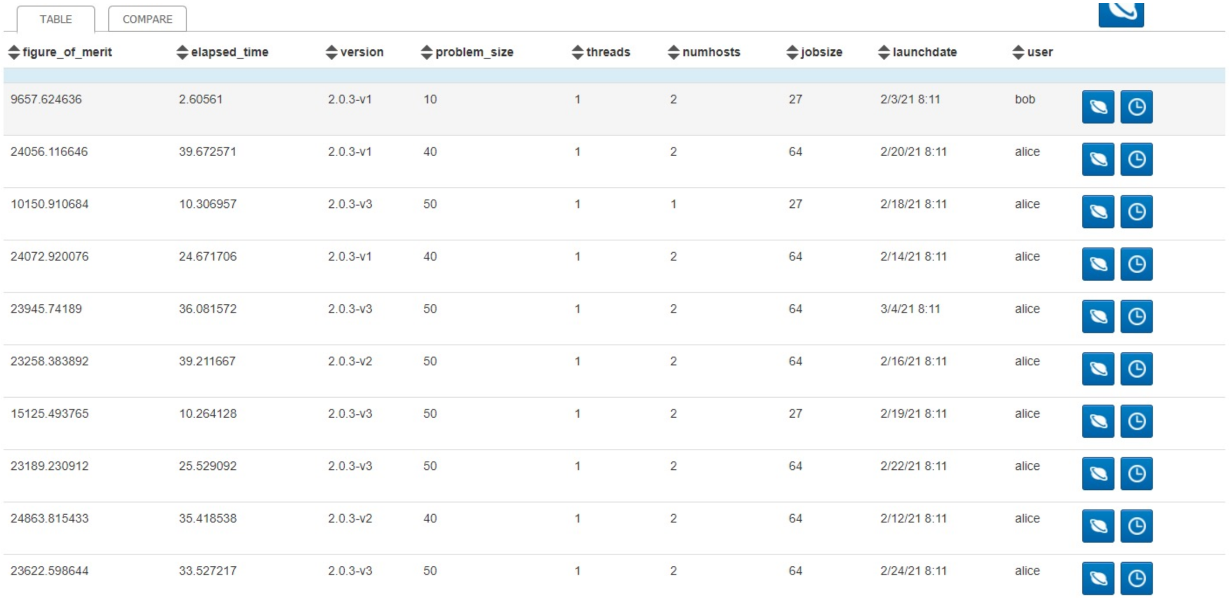


24 selected out of 75 records |
Reset All

figure_of_merit	problem_size	compiler	numhosts
139346.314251	20	intel	10
92027.153198	20	intel	6
108993.022725	30	intel	6
171273.541573	30	intel	10
168409.27287	40	intel	10

Table View

SPOT is Based on CrossFilter



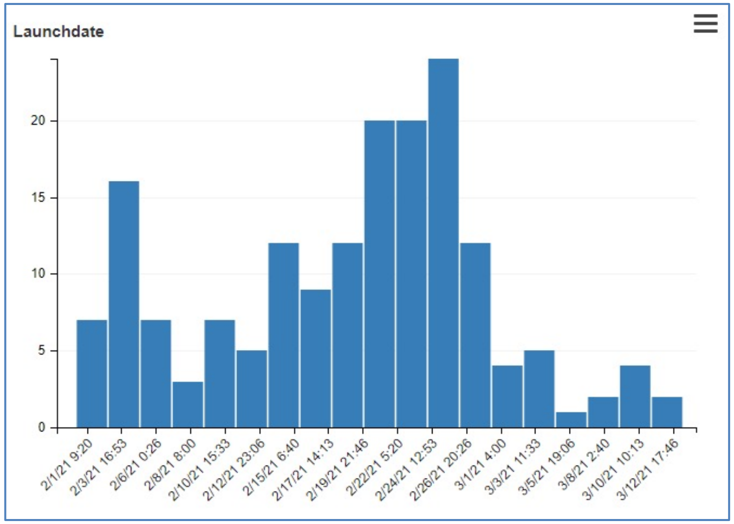
figure_of_merit	elapsed_time	version	problem_size	threads	numhosts	jobsize	launchdate	user
9657.624636	2.60561	2.0.3-v1	10	1	2	27	2/3/21 8:11	bob
24056.116646	39.672571	2.0.3-v1	40	1	2	64	2/20/21 8:11	alice
10150.910684	10.306957	2.0.3-v3	50	1	1	27	2/18/21 8:11	alice
24072.920076	24.671706	2.0.3-v1	40	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/

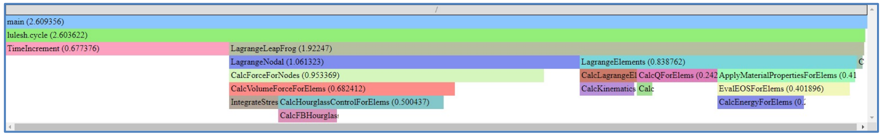
- CrossFilter is good at finding correlations in data.
- Crossfilter helps you manage/slice your data to make interesting comparisons.

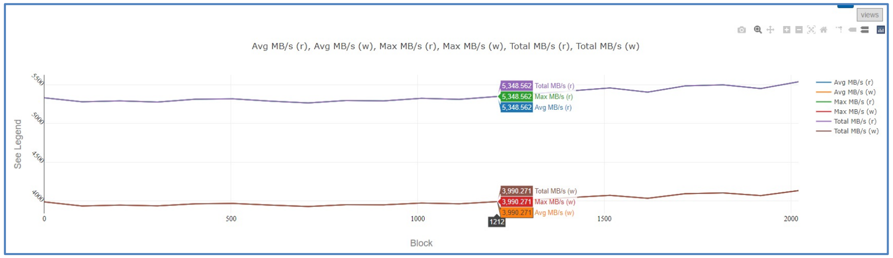
Each App Run has Two Types of Data

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



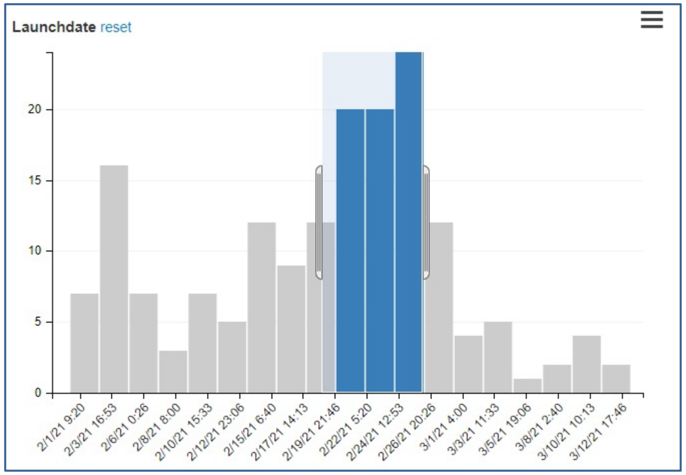
- **Performance Data**
  - Can be different metrics.
    - Avg Time/Rank, Max Time/Rank, bandwidth, …
  - Displayed as a flame graph or timeseries.

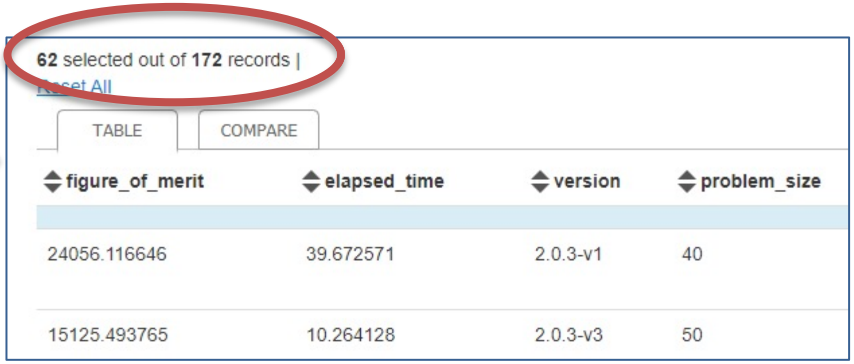

/
main (2.609356)
lulesh.cycle (2.603622)
TimeIncrement (0.677376)
LagrangeLeapFrog (1.92247)
LagrangeNodal (1.061323)
LagrangeElements (0.838762)
C
CalcForceForNodes (0.953369)
CalcLagrangeEl
CalcQForElems (0.242
ApplyMaterialPropertiesForElems (0.41
CalcVolumeForceForElems (0.682412)
CalcKinematics
Calc
EvalEOSForElems (0.401896)
IntegrateStres
CalcHourglassControlForElems (0.500437)
CalcEnergyForElems (0.
CalcFBHourglas



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.



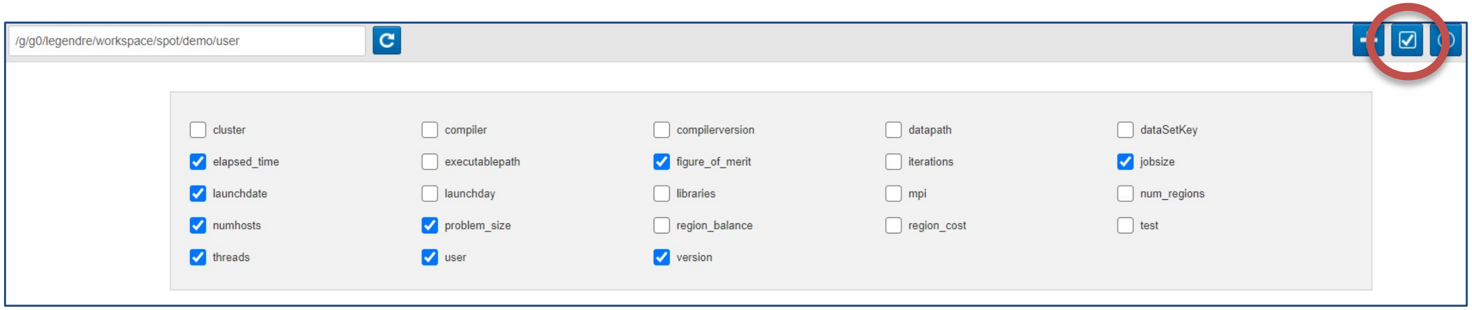


62 selected out of **172** records | Reset All

figure_of_merit	elapsed_time	version	problem_size
24056.116646	39.672571	2.0.3-v1	40
15125.493765	10.264128	2.0.3-v3	50

Metadata is Configurable

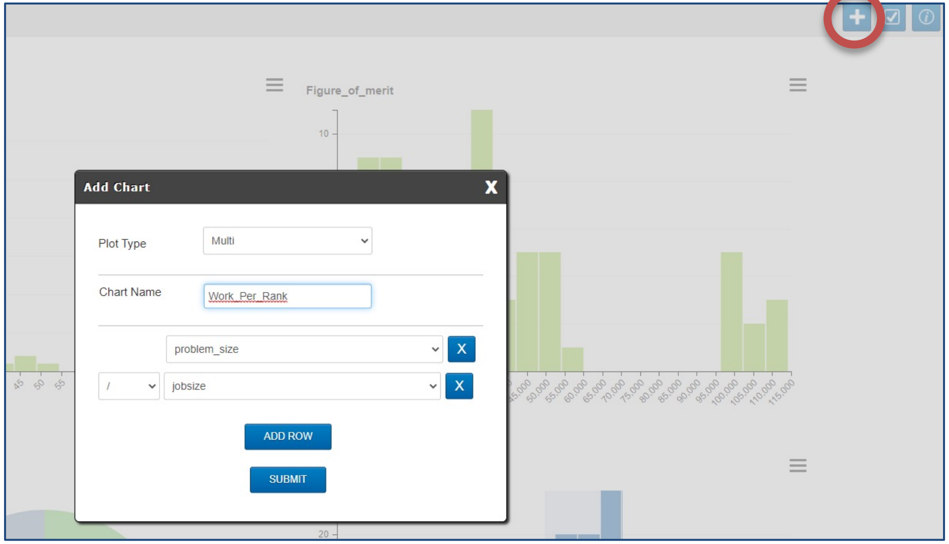
- Can show/hide metadata:



/g/g0/legendre/workspace/spot/demo/user

□ cluster
□ compiler
□ compilerversion
□ datapath
□ dataSetKey
☑ elapsed_time
□ executablepath
☑ figure_of_merit
□ iterations
☑ jobsize
☑ launchdate
□ launchday
□ libraries
□ mpi
□ num_regions
☑ numhosts
☑ problem_size
□ region_balance
□ region_cost
□ test
☑ threads
☑ user
☑ version

- Can create composite metadata:



Add Chart X

Plot Type: Multi

Chart Name: Work_Per_Rank

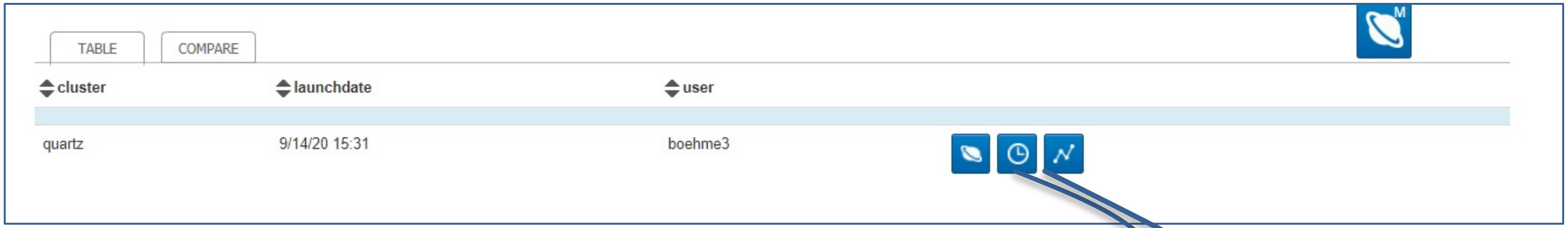
problem_size X

/ jobsize X

ADD ROW

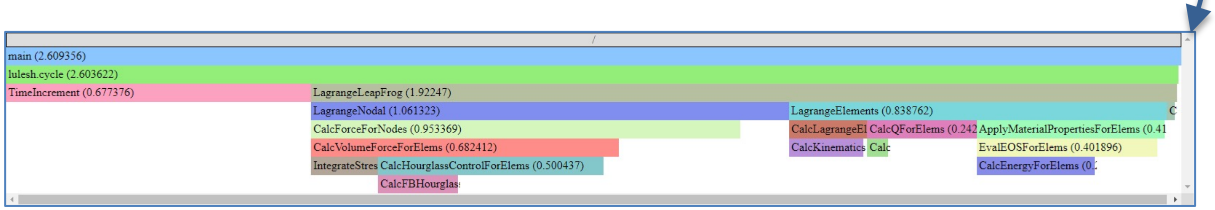
SUBMIT

Performance Data can be Visualized Per-Run

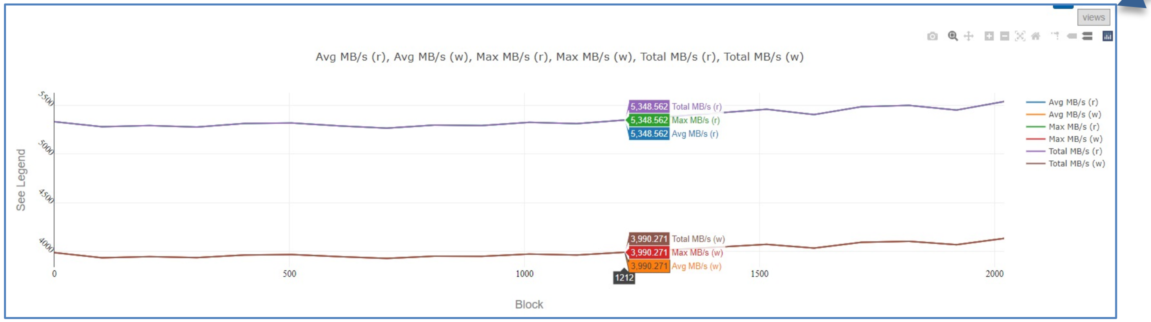


cluster	launchdate	user
quartz	9/14/20 15:31	boehme3

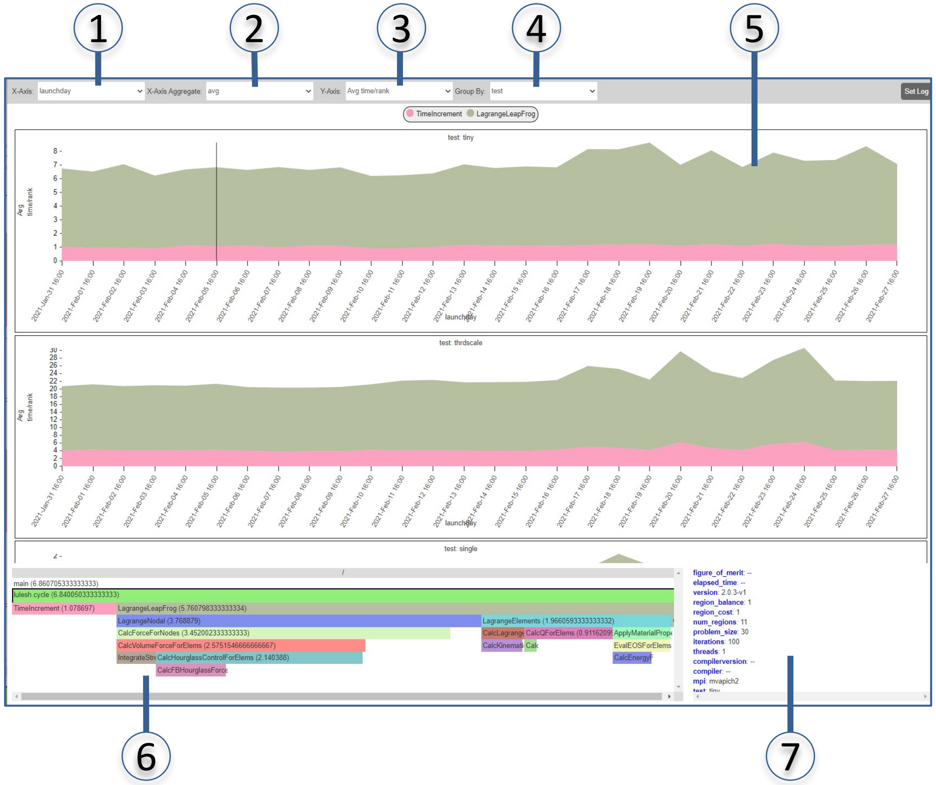
- FlameGraph: performance against code



- Timeseries: performance against time



Performance Data can be Compared Across Runs



X-Axis: launchday
X-Axis Aggregate: avg
Y-Axis: Avg time/rank
Group By: test

figure_of_merit: --
elapsed_time: --
version: 2.0.3-v1
region_balance: 1
region_cost: 1
num_regions: 11
problem_size: 30
iterations: 100
threads: 1
compilerversion: --
compiler: --
mpi: mvapich2
test: tiny

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

Lawrence Livermore National Laboratory

Disclaimer
This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.