Performance Analysis with Hatchet

12 April 2021

Olga Pearce, Stephanie Brink,
Abhinav Bhatele (Univ of Maryland), Todd Gamblin

LLNL-PRES-813307
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

Getting Hatchet Tutorial Materials

- The SPOT container includes a sample Jupyter notebook, Hatchet v1.3.0 install, and Lulesh datasets.
  - Alternatively, the sample Jupyter notebook and the Lulesh datasets are available directly at https://github.com/llnl/spotbe. This will require you to setup your own environment with a caliper and hatchet install (and setup the paths accordingly in the notebook)!

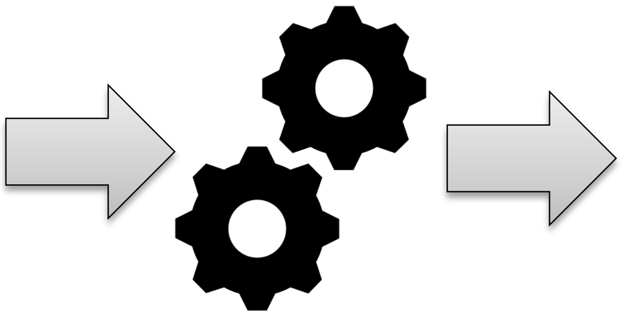
- Following this tutorial, you can substitute your own SPOT/Caliper data files into the example notebook.

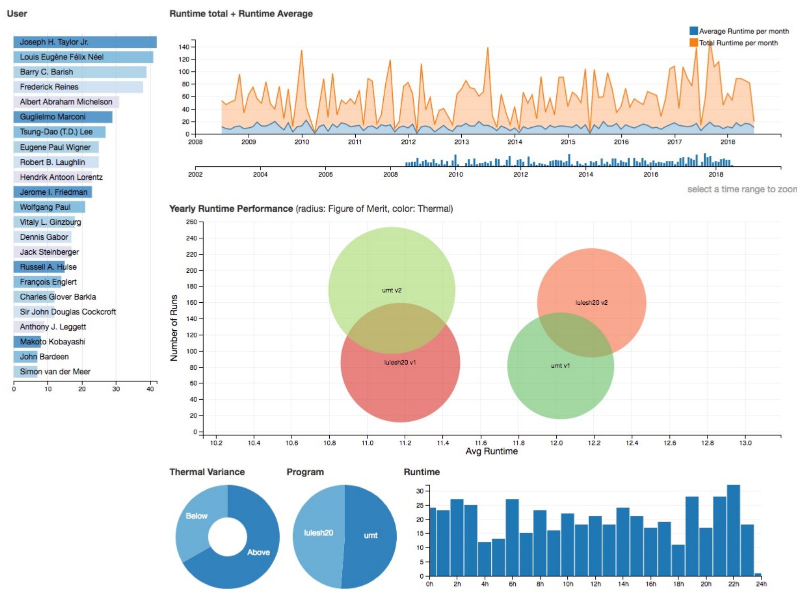
- We'll use this material in the hands-on portion of the tutorial.

Automated Application Performance Analysis: Caliper → SPOT → Hatchet

```
#include <caliper/cali.h>

static inline
void LagrangeElements(Domain&
domain, Index_t numElem)
{
    CALI_CXX_MARK_FUNCTION;
// ...
```





Caliper instrumentation in the application

At runtime: Performance and Metadata Collection

Web-based Visualization and Analysis Tools

Analyze caliper datasets in Python

Caliper

SPOT and Hatchet

c/o D Boehme

*Hatchet can analyze other datasets (HPCToolkit, gprof, TAU (WIP), Ascent (WIP))

SPOT Web Interface: Run Table and Jupyter Notebooks

All records selected. Please click on the graph to apply filters.

TABLE | COMPARE

Figure_of_merit	Problem_size	Threads	Jobsize	launchdate
1				
5491.526855	60	36	1	2019-Jul-16 14:03
12412.877512	30	4	8	2019-Jul-16 14:04
5525.153912	60	36	1	2019-Jul-16 14:05
11339.841229	30			
5100.923858	60			
11197.780437	30			
5071.148951	60			
10247.931707	30			
18468.799766	60			
18666.324557	60			

Buttons bring up Jupyter notebook or specialized analysis views

jupyterhub 190716-140428166192 Last Checkpoint: 11/30/2020 (unsaved changes)

Logout | Control Panel

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3

In [4]:
```
# Print the tree representation using the inclusive time metric
print(gf.tree(metric_column="time (inc)"))
```

```
v1.3.0

79.951 main
└─ 79.886 lulesh.cycle
   ├─ 79.840 LagrangeLeapFrog
   │  ├─ 0.871 CalcTimeConstraintsForElems
   │  ├─ 21.696 LagrangeElements
   │  │  ├─ 16.367 ApplyMaterialPropertiesForElems
   │  │  │  └─ 16.248 EvalEOSForElems
   │  │  │     └─ 12.419 CalcEnergyForElems
   │  │  ├─ 2.395 CalcLagrangeElements
   │  │  │  └─ 2.260 CalcKinematicsForElems
   │  │  └─ 2.870 CalcQForElems
   │  │     └─ 1.389 CalcMonotonicQForElems
   │  └─ 57.258 LagrangeNodal
   │     └─ 56.424 CalcForceForNodes
   │        └─ 56.273 CalcVolumeForceForElems
   │           ├─ 44.650 CalcHourglassControlForElems
   │           │  └─ 14.380 CalcFBHourglassForceForElems
   │           └─ 11.210 IntegrateStressForElems
   └─ 0.032 TimeIncrement

Legend (Metric: time (inc))
█ 71.96 - 79.95
█ 55.98 - 71.96
█ 39.99 - 55.98
█ 24.01 - 39.99
█ 8.02 - 24.01
█ 0.03 - 8.02

name User code    ◀ Only in left graph    ▶ Only in right graph
```

Jupyter notebook contains Hatchet functions

c/o D Boehme

Hatchet is a performance analysis tool for parallel profiles

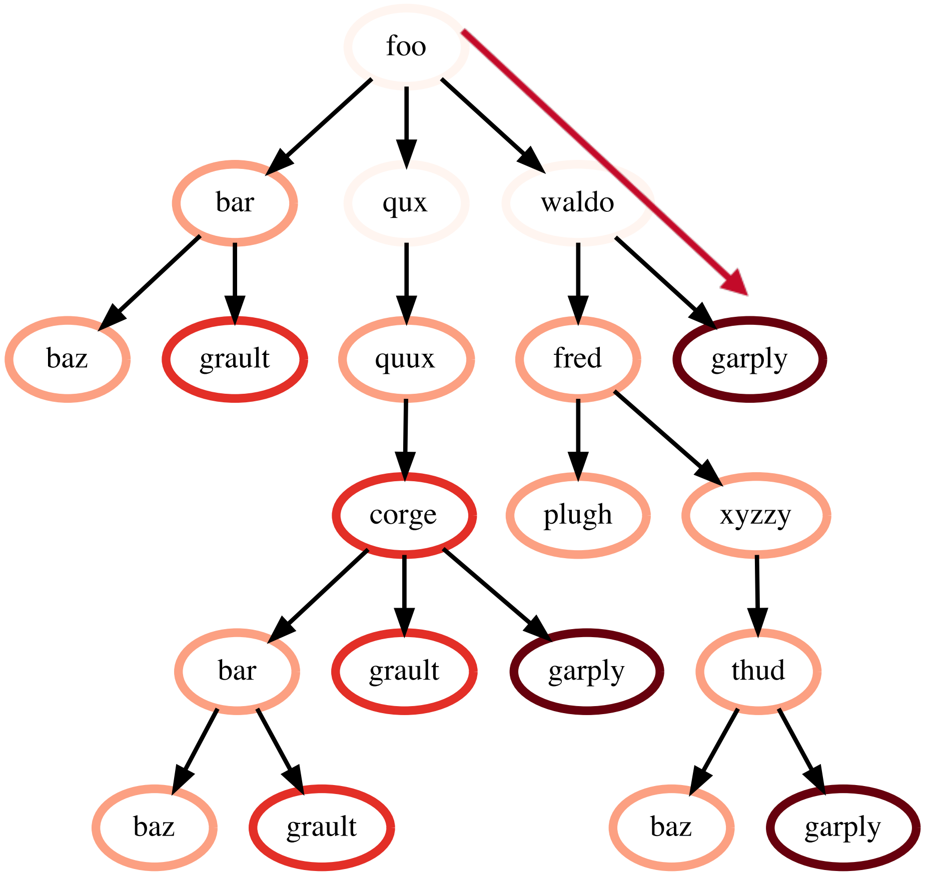
- Identify performance bottlenecks to enhance application development
  - Profiling and tracing tools (*e.g.*, Caliper, HPCToolkit, TAU, Score-P, Gprof, Callgrind) provide insights into parts of the code that consume the most time

- Hatchet is an open-source python-based tool for enabling programmatic analysis of structured (or hierarchical) data

- Hatchet can be used to sub-select and focus on a specific region of the data, compare multiple execution profiles, and automate analysis in python scripts

https://github.com/hatchet/hatchet/

What do profiling/tracing tools collect?



Calling Context Tree (CCT)

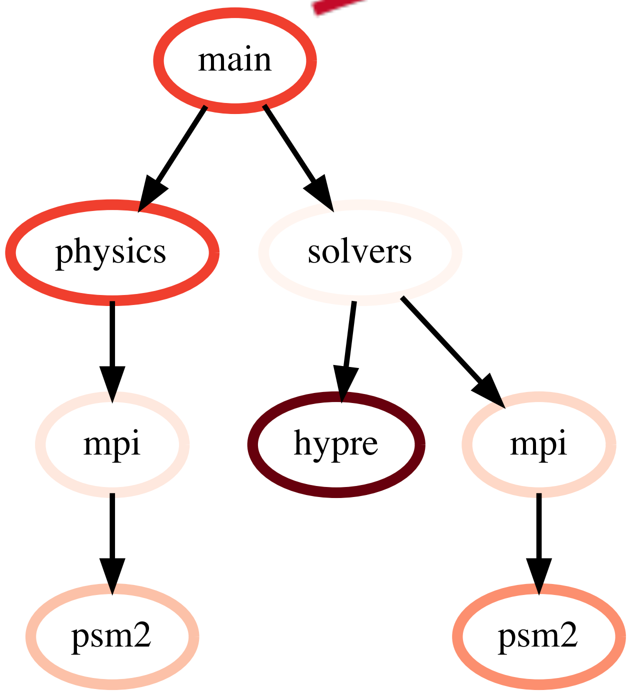
Each node may contain:

- Contextual Info
  - File
  - Line number
  - Function name
  - Callpath
  - Load module
  - Rank ID
  - Thread ID
- Performance Metrics
  - Time
  - Flops
  - Cache misses

Hatchet can read profiles from:

- Caliper
- HPCToolkit
- Gprof
- TAU (WIP)
- Ascent (WIP)

Hatchet's *GraphFrame*: a Graph and a Dataframe



node	name	nid	node	time	time (inc)
main	main	0	main	40.0	200.0
physics	physics	1	physics	40.0	60.0
mpi	mpi	2	mpi	5.0	20.0
psm2	psm2	3	psm2	15.0	15.0
solvers	solvers	4	solvers	0.0	100.0
hypre	hypre	5	hypre	65.0	65.0
mpi	mpi	6	mpi	10.0	35.0
psm2	psm2	7	psm2	25.0	25.0

Graph: Stores relationships between parents and children

Pandas Dataframe: 2D table storing numerical data associated with each node (may be unique per rank, per thread)

Visualizing Hatchet's GraphFrame components

```
>>> print(gf.tree())        # print graph
>>> print(gf.dataframe)    # print dataframe
```

```
0.000 foo
├─ 6.000 bar
│   └─ 5.000 baz
├─ 0.000 qux
│   └─ 5.000 quux
│       ├─ 10.000 corge
│       ├─ 15.000 garply
│       └─ 1.000 grault
└─ 15.000 waldo
    ├─ 3.000 fred
    │   └─ 5.000 plugh
    └─ 15.000 garply
```

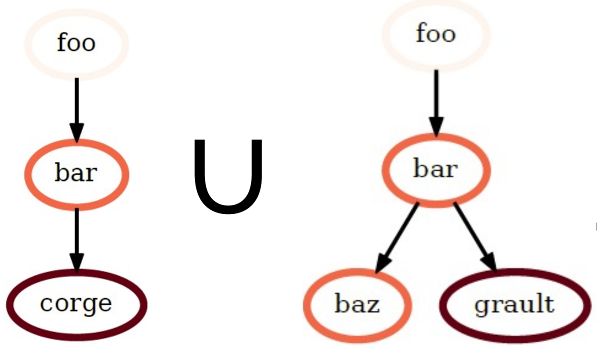
```
Legend (Metric: time)
■ 13.50 - 15.00
■ 10.50 - 13.50
■ 7.50 - 10.50
■ 4.50 - 7.50
■ 1.50 - 4.50
■ 0.00 - 1.50

name User code    ◀ Only in left graph    ▶ Only in right graph
```

	name	time	time (inc)
node			
{'name': 'foo'}	foo	0.0	130.0
{'name': 'bar'}	bar	5.0	20.0
{'name': 'baz'}	baz	5.0	5.0
{'name': 'grault'}	grault	10.0	10.0
{'name': 'qux'}	qux	0.0	60.0
{'name': 'quux'}	quux	5.0	60.0
{'name': 'corge'}	corge	10.0	55.0
{'name': 'bar'}	bar	5.0	20.0
{'name': 'baz'}	baz	5.0	5.0
{'name': 'grault'}	grault	10.0	10.0
{'name': 'garply'}	garply	15.0	15.0
{'name': 'grault'}	grault	10.0	10.0

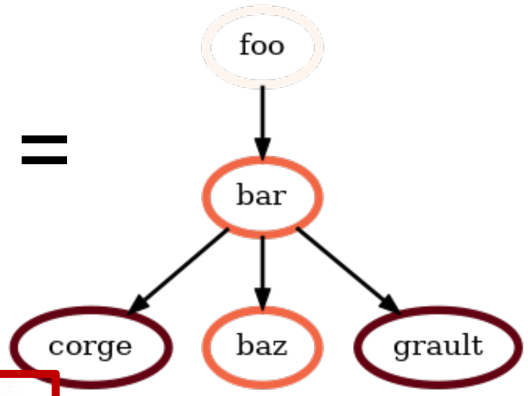
Compare GraphFrames using division (or add, subtract, multiply)

```
>>> gf3 = gf1 / gf2  # divide graphframes
```



*First, unify two trees since structure is different

=



gf3 = **gf1** / **gf2**

```
0.000 foo
├─ 2.000 bar
│   └─ 5.000 baz
└─ inf qux
    └─ 4.000 quux
        ├─ 2.000 corge
        ├─ nan garply ▶
        └─ nan grault ▶
```

```
0.000 foo
├─ 6.000 bar
│   └─ 5.000 baz
└─ 3.000 qux
    └─ 2.000 quux
        └─ 8.000 corge
```

```
0.000 foo
├─ 3.000 bar
│   └─ 1.000 baz
└─ 0.000 qux
    └─ 0.500 quux
        ├─ 4.000 corge
        ├─ 15.000 garply
        └─ 0.250 grault
```

```
>>> gf3 = gf1 + gf2  # add graphframes
>>> gf3 = gf1 - gf2  # subtract graphframes
>>> gf3 = gf1 * gf2  # multiply graphframes
```

Filter the GraphFrame *by node metrics in the dataframe*

```
>>> filter_func = lambda x: x["time"] > 1           # filter function
>>> filt_gf = gf.filter(filter_func, squash=True)  # apply filter and rewire graph
```

```
0.000 foo
├─ 6.000 bar
│   └─ 5.000 baz
├─ 0.000 qux
│   └─ 5.000 quux
│       ├─ 10.000 corge
│       ├─ 15.000 garply
│       └─ 1.000 grault
└─ 15.000 waldo
    ├─ 3.000 fred
    │   └─ 5.000 plugh
    └─ 15.000 garply
```

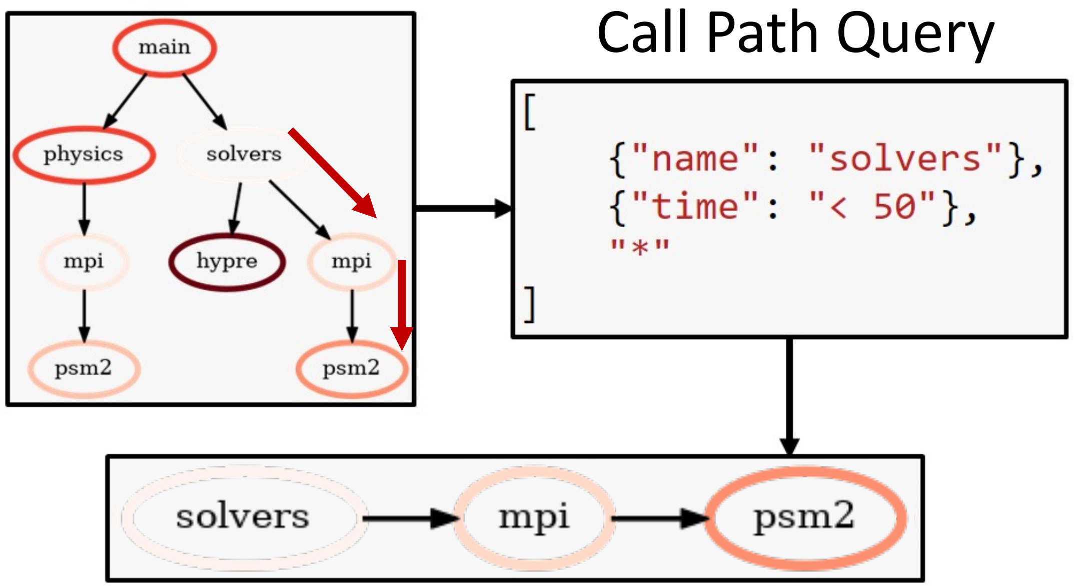
Keep only those nodes with a value greater than 1

```
6.000 bar
└─ 5.000 baz
5.000 quux
├─ 10.000 corge
└─ 15.000 garply
15.000 waldo
├─ 3.000 fred
│   └─ 5.000 plugh
└─ 15.000 garply
```

Filter the GraphFrame *using Hatchet's call path query language*

- Data reduction using *call path* pattern matching

```
# filter using call path query language
query = [
    { "name": "solvers" },
    { "time": "< 50" },
    "*"
]
filt_gf = gf.filter(query, squash=True)
```

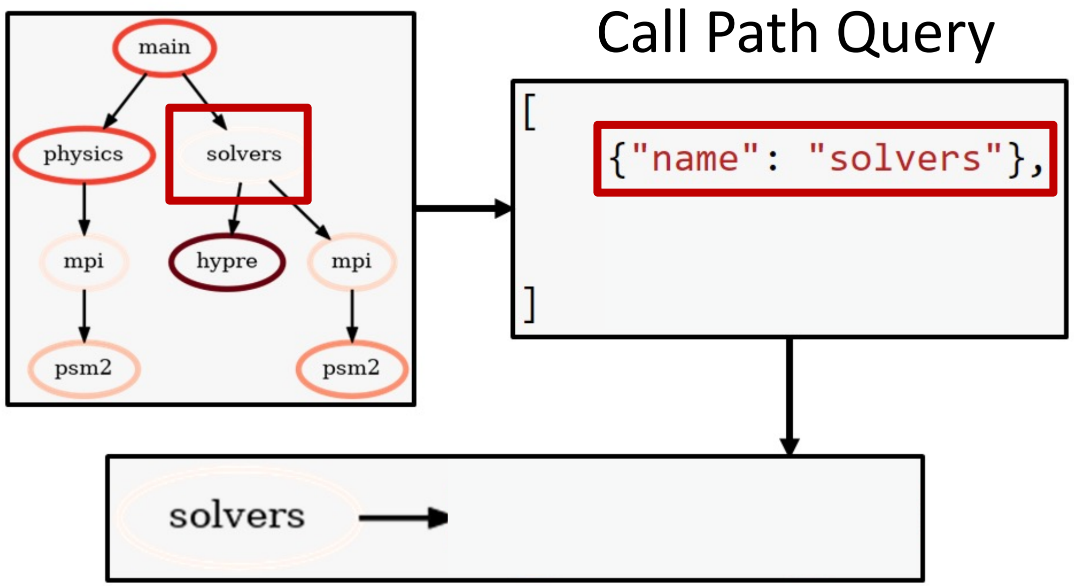


Matches a call path (1) rooted at a node with name "solvers", (2) followed by a node with a time metric value less than 50, and (3) followed by any number of children nodes.

Filter the GraphFrame *using Hatchet's call path query language*

- Data reduction using *call path* pattern matching

```
# filter using call path query language
query = [
    { "name": "solvers" },
    { "time": "< 50" },
    "*"
]
filt_gf = gf.filter(query, squash=True)
```

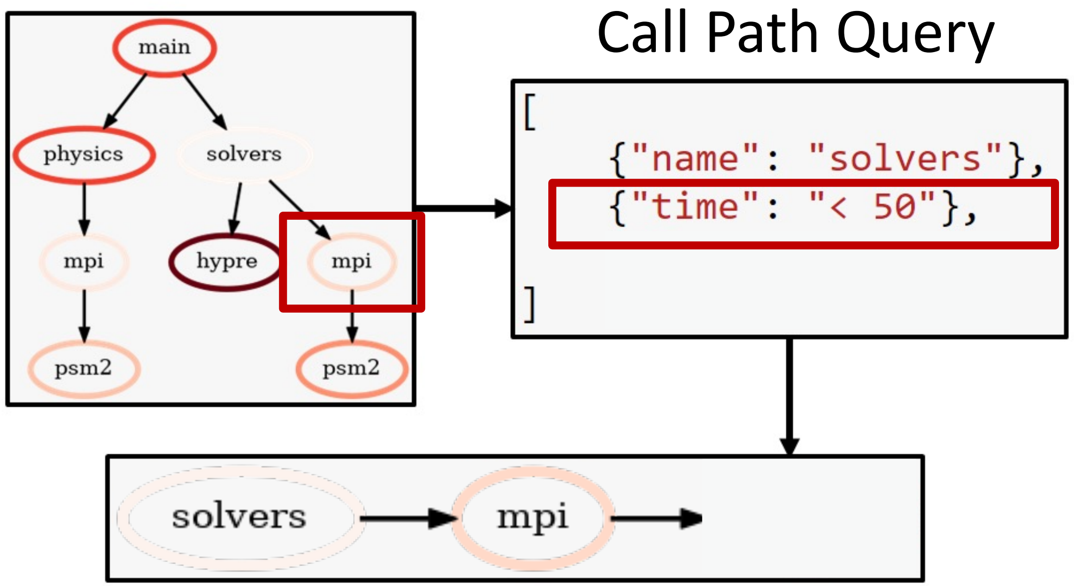


Matches a call path (1) rooted at a node with name "solvers", (2) followed by a node with a time metric value less than 50, and (3) followed by any number of children nodes.

Filter the GraphFrame *using Hatchet's call path query language*

- Data reduction using *call path* pattern matching

```python
# filter using call path query language
query = [
    { "name": "solvers" },
    { "time": "< 50" },
    "*"
]
filt_gf = gf.filter(query, squash=True)
```

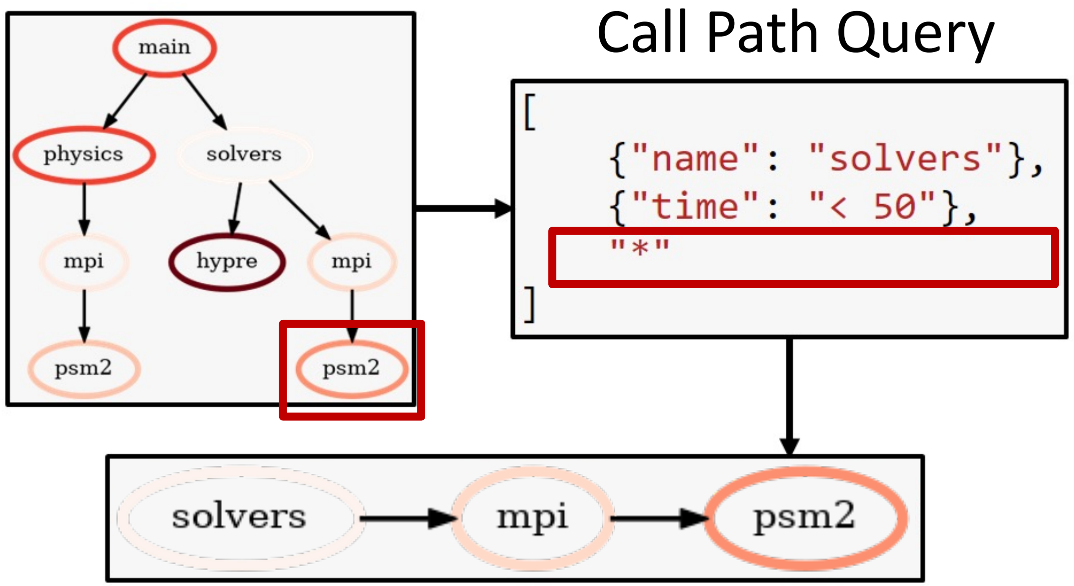


Matches a call path (1) rooted at a node with name "solvers", (2) followed by a node with a time metric value less than 50, and (3) followed by any number of children nodes.

Filter the GraphFrame *using Hatchet's call path query language*

- Data reduction using *call path* pattern matching

```
# filter using call path query language
query = [
    { "name": "solvers" },
    { "time": "< 50" },
    "*"
]
filt_gf = gf.filter(query, squash=True)
```



Matches a call path (1) rooted at a node with name "solvers", (2) followed by a node with a time metric value less than 50, and (3) followed by any number of children nodes.

How do I load SPOT/Caliper data into Hatchet?

1. Directory of SPOT/Caliper files

All records selected. Please click on the graph to apply filters.

TABLE COMPARE

Figure_of_merit	Problem_size	Threads	Jobsize	launchdate
1				
5491.526855	60	36	1	2019-Jul-16 14:03
12412.877512	30	4	8	2019-Jul-16 14:04
5525.153912	60	36	1	2019-Jul-16 14:05
11339.841229	30	4	8	2019-Jul-16 14:06
5100.923858	60	36	1	2019-Jul-16 14:07
11197.780437	30	4	8	2019-Jul-16 14:08
5071.148951	60	36	1	2019-Jul-16 14:09
10247.931707	30	4	8	2019-Jul-16 14:10
18468.799766	60	8	8	2019-Jul-16 14:06
18666.324557	60	8	8	2019-Jul-16 14:13

2. Buttons bring up filled-in Jupyter notebook loading 1 or many SPOT/Caliper files

jupyterhub 190716-140428166192 Last Checkpoint: 11/30/2020 (unsaved changes)

Logout Control Pane

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

Hatchet Notebook v0.1.0

In [ ]:
```
import sys
import subprocess
import json
import os
import platform

import pandas as pd
from IPython.display import display, HTML

machine = platform.uname().machine

# Add hatchet to PYTHONPATH
deploy_dir = "/usr/gapps/spot/live/"
sys.path.append(deploy_dir + 'hatchet/' + machine)
import hatchet as ht
```

In [ ]:
```
# Add cali-query to PATH
cali_query_path = "/usr/gapps/spot/live/caliper/" + machine + "/bin"
os.environ["PATH"] += os.pathsep + cali_query_path

cali_file = "/usr/gapps/spot/datasets/lulesh_new/190716-140428166192.cali"

grouping_attribute = "prop:nested"
default_metric = "avg#inclusive#sum#time.duration"
query = "select %s,sum(%s) group by %s format json-split" % (grouping_attribute, default_metric, grouping_attribute)
```

In [ ]:
```
gf = ht.GraphFrame.from_caliper(cali_file, query)
```

3. Caliper file(s) to explore

4. Setup cali-query to extract performance data

5. Hatchet's Caliper reader loads into Hatchet's GraphFrame object

Hands-On Time!

- The SPOT container includes a sample Jupyter notebook, Hatchet v1.3.0 install, and Lulesh datasets.
  - Alternatively, the sample Jupyter notebook and the Lulesh datasets are available directly at https://github.com/llnl/spotbe. This will require you to setup your own environment with a caliper and hatchet install (and setup the paths accordingly in the notebook)!

- Following this tutorial, you can substitute your own SPOT/Caliper data files into the example notebook.

- Hop over to Jupyter to run the notebook

- We'll be walking through `hatchet_ecp2021_tutorial_demo.ipynb`

Review: Topics covered in today's tutorial

- Single graph:
  - **Load** SPOT/Caliper data file
  - **Visualize** tree and dataframe
  - **Filter and squash** tree

```
# Read in a SPOT/Caliper file
gf = ht.GraphFrame.from_caliper(
        “my-file.cali”,
        query,
     )
```

```
# Print tree visualization
print(gf.tree(metric_column=“time (inc)”))
```

- Subtract two trees:
  - Load two SPOT/Caliper data files
  - Compute **percent change** of two nightly test runs (two different times)
  - **Update** existing **column** in dataframe
  - **Added** new **column** to dataframe
  - Visualize resulting tree

```
# Print dataframe
print(gf.dataframe)
```

- Speedup of two trees:
  - Load two SPOT/Caliper data files
  - **Divide** two graphs for speedup comparison
  - Visualize resulting tree
  - **Generate speedup plot** for interesting functions

```
# Divide two trees
gf3 = gf2 / gf1
```

```
# Diff two trees
gf3 = (gf2 – gf1) / gf1
```

Readily available features not covered in today's tutorial

- **Add** or **multiply** two graphframes

- **Insert new column** to dataframe of metrics
  - Scale and offset "time" column by some factor: https://hatchet.readthedocs.io/en/latest/advanced_examples.html#applying-scalar-operations-to-attributes
  - Compute imbalance across MPI ranks within a single application execution: https://hatchet.readthedocs.io/en/latest/advanced_examples.html#applying-scalar-operations-to-attributes

- **Groupby-and-aggregate** nodes by other columns (e.g., function name, file name)
  - `res = gf.groupby_aggregate([“file”], {“time”: np.sum})`

- For more details, please visit our User Guide: https://hatchet.readthedocs.io/en/latest/user_guide.html

Summary

- Hatchet is a performance analysis tool for parallel profiles
- It enables programmatic analysis of hierarchical data from one or multiple execution profiles
- Future Work:
  - Support other profile formats, add a format for outputting GraphFrames to disk
  - Implement a higher-level API for automating performance analysis
- Hatchet https://github.com/hatchet/hatchet/
- Caliper https://github.com/LLNL/Caliper
- SPOT https://github.com/LLNL/spot2_container

Please contact us at
hatchet-help@listserv.umd.edu
or submit GitHub issues for Hatchet questions, issues, or feature requests!

Overview of Hatchet Tutorial Examples

jupyter hatchet_ecp2021_tutorial_demo Last Checkpoint: 22 minutes ago (autosaved)

Logou

File Edit View Insert Cell Kernel Help

Trusted | Python 3

Run | Code

In [ ]:
```
import sys
import os
import platform

import pandas as pd
import numpy as np
from IPython.display import display, HTML

machine = platform.uname().machine

# Add hatchet to PYTHONPATH
deploy_dir = "/usr/gapps/spot/"
sys.path.append(deploy_dir + 'hatchet/' + machine)
import hatchet as ht
```

Setup cali-query to extract performance data

In [ ]:
```
# Add cali-query to PATH
cali_query_path = "/usr/gapps/spot/caliper-install/bin"
os.environ["PATH"] += os.pathsep + cali_query_path

# Setup cali-query to read Spot/Caliper files into Hatchet
grouping_attribute = "prop:nested"
default_metric = "avg#inclusive#sum#time.duration"
query = "select %s,sum(%s) group by %s format json-split" % (grouping_attribute, default_metric, grouping_attribute)
```

Caliper file to explore

Load in a single data file and visualize the tree and dataframe

In [ ]:
```
# Path to a Spot/Caliper file
cali_file = "./hatchet-data/cDPu64825TuLB5ujG_0.cali" # problemsize=10, iter=215, jobsize=1, 2/1/21 8:04

# Read Spot/Caliper file into a Hatchet GraphFrame
gf = ht.GraphFrame.from_caliper(cali_file, query)
```

Hatchet's Caliper reader loads into Hatchet's data object called a GraphFrame

In [ ]:
```
# Print the tree representation using the inclusive time metric
print(gf.tree(metric_column="time (inc)"))
```

Visualizing the call graph

jupyter hatchet_ecp2021_tutorial_demo Last Checkpoint: 25 minutes ago (autosaved)

File Edit View Insert Cell Kernel Help

Run Code

```
In [4]: # Print the tree representation using the inclusive time metric
        print(gf.tree(metric_column="time (inc)"))
```

```
v1.3.0

0.299 main
└─ 0.299 lulesh.cycle
   ├─ 0.297 LagrangeLeapFrog
   │  ├─ 0.011 CalcTimeConstraintsForElems
   │  ├─ 0.203 LagrangeElements
   │  │  ├─ 0.152 ApplyMaterialPropertiesForElems
   │  │  │  └─ 0.149 EvalEOSForElems
   │  │  │     └─ 0.109 CalcEnergyForElems
   │  │  ├─ 0.027 CalcLagrangeElements
   │  │  │  └─ 0.025 CalcKinematicsForElems
   │  │  └─ 0.023 CalcQForElems
   │  │     └─ 0.008 CalcMonotonicQForElems
   │  └─ 0.083 LagrangeNodal
   │     └─ 0.079 CalcForceForNodes
   │        └─ 0.077 CalcVolumeForceForElems
   │           ├─ 0.053 CalcHourglassControlForElems
   │           │  └─ 0.032 CalcFBHourglassForceForElems
   │           └─ 0.022 IntegrateStressForElems
   └─ 0.001 TimeIncrement

Legend (Metric: time (inc))
 0.27 - 0.30
 0.21 - 0.27
 0.15 - 0.21
 0.09 - 0.15
 0.03 - 0.09
 0.00 - 0.03

name User code   ◀ Only in left graph   ▶ Only in right graph
```

- Each node in the tree visualization maps to a function call in the application
- Nodes that are red have high execution time
- Nodes highlighted in grey indicate user functions (code from external libraries are not highlighted)

```
In [5]: # Get the man page of different parameters for Hatchet's tree function
        help(gf.tree)

Help on method tree in module hatchet.graphframe:

tree(metric_column='time', precision=3, name_column='name', expand_name=False, context_column='file', rank=0, thread=
0, depth=10000, highlight_name=False, invert_colormap=False) method of hatchet.graphframe.GraphFrame instance
    Format this graphframe as a tree and return the resulting string.
```

Overview of single tree dataset (Lulesh Data)

```
0.299 main
└─ 0.299 lulesh.cycle
   ├─ 0.297 LagrangeLeapFrog
   │  ├─ 0.011 CalcTimeConstraintsForElems
   │  ├─ 0.203 LagrangeElements
   │  │  ├─ 0.152 ApplyMaterialPropertiesForElems
   │  │  │  └─ 0.149 EvalEOSForElems
   │  │  │     └─ 0.109 CalcEnergyForElems
   │  │  ├─ 0.027 CalcLagrangeElements
   │  │  │  └─ 0.025 CalcKinematicsForElems
   │  │  └─ 0.023 CalcQForElems
   │  │     └─ 0.008 CalcMonotonicQForElems
   │  └─ 0.083 LagrangeNodal
   │     └─ 0.079 CalcForceForNodes
   │        └─ 0.077 CalcVolumeForceForElems
   │           ├─ 0.053 CalcHourglassControlForElems
   │           │  └─ 0.032 CalcFBHourglassForceForElems
   │           └─ 0.022 IntegrateStressForElems
   └─ 0.001 TimeIncrement

Legend (Metric: time (inc))
  0.27 - 0.30
  0.21 - 0.27
  0.15 - 0.21
  0.09 - 0.15
  0.03 - 0.09
  0.00 - 0.03

name User code    ◀ Only in left graph    ▶ Only in right graph
```

node	time (inc)	nid	name
{'name': 'main', 'type': 'region'}	0.299168	0	main
{'name': 'lulesh.cycle', 'type': 'region'}	0.298705	1	lulesh.cycle
{'name': 'LagrangeLeapFrog', 'type': 'region'}	0.297491	3	LagrangeLeapFrog
{'name': 'CalcTimeConstraintsForElems', 'type': 'region'}	0.010763	18	CalcTimeConstraintsForElems
{'name': 'LagrangeElements', 'type': 'region'}	0.203010	10	LagrangeElements
{'name': 'ApplyMaterialPropertiesForElems', 'type': 'region'}	0.151823	15	ApplyMaterialPropertiesForElems
{'name': 'EvalEOSForElems', 'type': 'region'}	0.149239	16	EvalEOSForElems
{'name': 'CalcEnergyForElems', 'type': 'region'}	0.108948	17	CalcEnergyForElems
{'name': 'CalcLagrangeElements', 'type': 'region'}	0.026737	11	CalcLagrangeElements
{'name': 'CalcKinematicsForElems', 'type': 'region'}	0.025106	12	CalcKinematicsForElems
{'name': 'CalcQForElems', 'type': 'region'}	0.023439	13	CalcQForElems
{'name': 'CalcMonotonicQForElems', 'type': 'region'}	0.008134	14	CalcMonotonicQForElems
{'name': 'LagrangeNodal', 'type': 'region'}	0.083145	4	LagrangeNodal
{'name': 'CalcForceForNodes', 'type': 'region'}	0.078704	5	CalcForceForNodes
{'name': 'CalcVolumeForceForElems', 'type': 'region'}	0.076960	6	CalcVolumeForceForElems
{'name': 'CalcHourglassControlForElems', 'type': 'region'}	0.053464	8	CalcHourglassControlForElems
{'name': 'CalcFBHourglassForceForElems', 'type': 'region'}	0.032474	9	CalcFBHourglassForceForElems
{'name': 'IntegrateStressForElems', 'type': 'region'}	0.021547	7	IntegrateStressForElems
{'name': 'TimeIncrement', 'type': 'region'}	0.000633	2	TimeIncrement

Filtering a tree

```python
# Add new column to the dataframe transforming the inclusive time column to a percentage of the max inclusive time
max_time = gf.dataframe["time (inc)"].max()
gf.dataframe["pct-of-max"] = gf.dataframe["time (inc)"] / max_time
# Filter the tree to contain only nodes consuming at least 60% of max time
filter_func = lambda x: x["pct-of-max"] > 0.6
filtered_squashed_gf = gf.filter(filter_func,
                                 squash=True)
```

Filter graph to keep nodes whose time is greater than 60% of the max time, then rewire graph

Original Graph

```
0.299 main
└─ 0.299 lulesh.cycle
   ├─ 0.297 LagrangeLeapFrog
   │  ├─ 0.011 CalcTimeConstraintsForElems
   │  ├─ 0.203 LagrangeElements
   │  │  ├─ 0.152 ApplyMaterialPropertiesForElems
   │  │  │  └─ 0.149 EvalEOSForElems
   │  │  │     └─ 0.109 CalcEnergyForElems
   │  │  ├─ 0.027 CalcLagrangeElements
   │  │  │  └─ 0.025 CalcKinematicsForElems
   │  │  └─ 0.023 CalcQForElems
   │  │     └─ 0.008 CalcMonotonicQForElems
   │  └─ 0.083 LagrangeNodal
   │     └─ 0.079 CalcForceForNodes
   │        └─ 0.077 CalcVolumeForceForElems
   │           ├─ 0.053 CalcHourglassControlForElems
   │           │  └─ 0.032 CalcFBHourglassForceForElems
   │           └─ 0.022 IntegrateStressForElems
   └─ 0.001 TimeIncrement

Legend (Metric: time (inc))
0.27 - 0.30
0.21 - 0.27
0.15 - 0.21
0.09 - 0.15
0.03 - 0.09
0.00 - 0.03

name User code    ◀ Only in left graph    ▶ Only in right graph
```

Filtered Graph

```
1.000 main
└─ 0.998 lulesh.cycle
   └─ 0.994 LagrangeLeapFrog
      └─ 0.679 LagrangeElements

Legend (Metric: pct-of-max)
0.97 - 1.00
0.90 - 0.97
0.84 - 0.90
0.78 - 0.84
0.71 - 0.78
0.68 - 0.71

name User code    ◀ Only in left graph    ▶ Only in right graph
```

Computing percent change between two trees

gf1

```
0.002 MPI_Allreduce
0.003 MPI_Bcast
0.123 MPI_Comm_dup
0.000 MPI_Comm_free
0.064 MPI_Comm_split
0.000 MPI_Gather
0.000 MPI_Initialized
267.883 main
├─ 0.008 MPI_Barrier
├─ 0.001 MPI_Irecv
├─ 0.015 MPI_Isend
├─ 0.001 MPI_Reduce
├─ 0.003 MPI_Wait
├─ 0.000 MPI_Waitall
└─ 267.768 lulesh.cycle
   ├─ 170.292 LagrangeLeapFrog
   │  ├─ 0.715 CalcTimeConstraintsForElems
   │  ├─ 62.024 LagrangeElements
   │  │  ├─ 17.252 ApplyMaterialPropertiesForElems
   │  │  │  └─ 16.847 EvalEOSForElems
   │  │  │     └─ 10.240 CalcEnergyForElems
   │  │  ├─ 13.460 CalcLagrangeElements
   │  │  │  └─ 12.795 CalcKinematicsForElems
   │  │  └─ 31.131 CalcQForElems
   │  │     ├─ 4.659 CalcMonotonicQForElems
   │  │     ├─ 0.020 MPI_Irecv
   │  │     ├─ 0.039 MPI_Isend
   │  │     ├─ 5.025 MPI_Wait
   │  │     └─ 11.666 MPI_Waitall
   │  └─ 107.549 LagrangeNodal
   │     ├─ 90.621 CalcForceForNodes
   │     │  ├─ 79.408 CalcVolumeForceForElems
   │     │  │  ├─ 65.453 CalcHourglassControlForElems
   │     │  │  │  └─ 15.538 CalcFBHourglassForceForElems
   │     │  │  └─ 11.806 IntegrateStressForElems
   │     │  ├─ 0.022 MPI_Irecv
   │     │  ├─ 0.062 MPI_Isend
   │     │  ├─ 2.367 MPI_Wait
   │     │  └─ 6.978 MPI_Waitall
   │     ├─ 0.017 MPI_Irecv
   │     ├─ 0.072 MPI_Isend
   │     ├─ 2.684 MPI_Wait
   │     └─ 8.848 MPI_Waitall
   └─ 97.472 TimeIncrement
      └─ 97.466 MPI_Allreduce
```

gf2

```
0.001 MPI_Allreduce
0.003 MPI_Bcast
0.106 MPI_Comm_dup
0.000 MPI_Comm_free
0.017 MPI_Comm_split
0.000 MPI_Gather
0.000 MPI_Initialized
257.411 main
├─ 0.004 MPI_Barrier
├─ 0.001 MPI_Irecv
├─ 0.017 MPI_Isend
├─ 0.002 MPI_Reduce
├─ 0.004 MPI_Wait
├─ 0.000 MPI_Waitall
└─ 257.300 lulesh.cycle
   ├─ 171.049 LagrangeLeapFrog
   │  ├─ 0.716 CalcTimeConstraintsForElems
   │  ├─ 63.021 LagrangeElements
   │  │  ├─ 17.173 ApplyMaterialPropertiesForElems
   │  │  │  └─ 16.777 EvalEOSForElems
   │  │  │     └─ 10.249 CalcEnergyForElems
   │  │  ├─ 13.482 CalcLagrangeElements
   │  │  │  └─ 12.847 CalcKinematicsForElems
   │  │  └─ 32.186 CalcQForElems
   │  │     ├─ 4.585 CalcMonotonicQForElems
   │  │     ├─ 0.020 MPI_Irecv
   │  │     ├─ 0.039 MPI_Isend
   │  │     ├─ 4.413 MPI_Wait
   │  │     └─ 13.395 MPI_Waitall
   │  └─ 107.308 LagrangeNodal
   │     ├─ 91.624 CalcForceForNodes
   │     │  ├─ 79.865 CalcVolumeForceForElems
   │     │  │  ├─ 65.855 CalcHourglassControlForElems
   │     │  │  │  └─ 15.643 CalcFBHourglassForceForElems
   │     │  │  └─ 11.857 IntegrateStressForElems
   │     │  ├─ 0.022 MPI_Irecv
   │     │  ├─ 0.062 MPI_Isend
   │     │  ├─ 2.300 MPI_Wait
   │     │  └─ 7.601 MPI_Waitall
   │     ├─ 0.018 MPI_Irecv
   │     ├─ 0.070 MPI_Isend
   │     ├─ 2.993 MPI_Wait
   │     └─ 7.520 MPI_Waitall
   └─ 86.247 TimeIncrement
      └─ 86.241 MPI_Allreduce
```

abs((gf2-gf1)/gf1)

```
15.459 MPI_Allreduce
10.480 MPI_Bcast
14.103 MPI_Comm_dup
7.692 MPI_Comm_free
72.650 MPI_Comm_split
3.896 MPI_Gather
0.000 MPI_Initialized
3.909 main
├─ 47.213 MPI_Barrier
├─ 2.260 MPI_Irecv
├─ 15.934 MPI_Isend
├─ 26.740 MPI_Reduce
├─ 20.120 MPI_Wait
├─ 8.333 MPI_Waitall
└─ 3.909 lulesh.cycle
   ├─ 0.445 LagrangeLeapFrog
   │  ├─ 0.067 CalcTimeConstraintsForElems
   │  ├─ 1.608 LagrangeElements
   │  │  ├─ 0.457 ApplyMaterialPropertiesForElems
   │  │  │  └─ 0.415 EvalEOSForElems
   │  │  │     └─ 0.083 CalcEnergyForElems
   │  │  ├─ 0.162 CalcLagrangeElements
   │  │  │  └─ 0.400 CalcKinematicsForElems
   │  │  └─ 3.388 CalcQForElems
   │  │     ├─ 1.591 CalcMonotonicQForElems
   │  │     ├─ 1.300 MPI_Irecv
   │  │     ├─ 0.038 MPI_Isend
   │  │     ├─ 12.175 MPI_Wait
   │  │     └─ 14.818 MPI_Waitall
   │  └─ 0.224 LagrangeNodal
   │     ├─ 1.107 CalcForceForNodes
   │     │  ├─ 0.576 CalcVolumeForceForElems
   │     │  │  ├─ 0.614 CalcHourglassControlForElems
   │     │  │  │  └─ 0.670 CalcFBHourglassForceForElems
   │     │  │  └─ 0.428 IntegrateStressForElems
   │     │  ├─ 0.182 MPI_Irecv
   │     │  ├─ 0.136 MPI_Isend
   │     │  ├─ 2.840 MPI_Wait
   │     │  └─ 8.935 MPI_Waitall
   │     ├─ 2.642 MPI_Irecv
   │     ├─ 3.152 MPI_Isend
   │     ├─ 11.509 MPI_Wait
   │     └─ 15.008 MPI_Waitall
   └─ 11.516 TimeIncrement
      └─ 11.517 MPI_Allreduce
```

Legend (Metric: abs-pct-change)

```
65.38 - 72.65
50.85 - 65.38
36.32 - 50.85
21.79 - 36.32
7.26 - 21.79
0.00 - 7.26
```

Computing speedup of two trees

1 Rank

```
0.299 main
└─ 0.299 lulesh.cycle
   ├─ 0.297 LagrangeLeapFrog
   │  ├─ 0.011 CalcTimeConstraintsForElems
   │  ├─ 0.203 LagrangeElements
   │  │  ├─ 0.152 ApplyMaterialPropertiesForElems
   │  │  │  └─ 0.149 EvalEOSForElems
   │  │  │     └─ 0.109 CalcEnergyForElems
   │  │  ├─ 0.027 CalcLagrangeElements
   │  │  │  └─ 0.025 CalcKinematicsForElems
   │  │  └─ 0.023 CalcQForElems
   │  │     └─ 0.008 CalcMonotonicQForElems
   │  └─ 0.083 LagrangeNodal
   │     └─ 0.079 CalcForceForNodes
   │        └─ 0.077 CalcVolumeForceForElems
   │           ├─ 0.053 CalcHourglassControlForElems
   │           │  └─ 0.032 CalcFBHourglassForceForElems
   │           └─ 0.022 IntegrateStressForElems
   └─ 0.001 TimeIncrement
```

/

64 Ranks

```
0.000 MPI_Allreduce
0.001 MPI_Bcast
0.002 MPI_Comm_dup
0.000 MPI_Comm_free
0.001 MPI_Comm_split
0.000 MPI_Gather
0.000 MPI_Initialized
3.031 main
├─ 0.000 MPI_Barrier
├─ 0.001 MPI_Irecv
├─ 0.000 MPI_Isend
├─ 0.000 MPI_Reduce
├─ 0.000 MPI_Wait
├─ 0.000 MPI_Waitall
└─ 3.029 lulesh.cycle
   ├─ 2.203 LagrangeLeapFrog
   │  ├─ 0.017 CalcTimeConstraintsForElems
   │  ├─ 0.899 LagrangeElements
   │  │  ├─ 0.420 ApplyMaterialPropertiesForElems
   │  │  │  └─ 0.405 EvalEOSForElems
   │  │  │     └─ 0.265 CalcEnergyForElems
   │  │  ├─ 0.167 CalcLagrangeElements
   │  │  │  └─ 0.158 CalcKinematicsForElems
   │  │  └─ 0.306 CalcQForElems
   │  │     ├─ 0.045 CalcMonotonicQForElems
   │  │     ├─ 0.005 MPI_Irecv
   │  │     ├─ 0.010 MPI_Isend
   │  │     ├─ 0.103 MPI_Wait
   │  │     └─ 0.022 MPI_Waitall
   │  └─ 1.283 LagrangeNodal
   │     ├─ 1.100 CalcForceForNodes
   │     │  ├─ 0.756 CalcVolumeForceForElems
   │     │  │  ├─ 0.570 CalcHourglassControlForElems
   │     │  │  │  └─ 0.185 CalcFBHourglassForceForElems
   │     │  │  └─ 0.144 IntegrateStressForElems
   │     │  ├─ 0.015 MPI_Irecv
   │     │  ├─ 0.027 MPI_Isend
   │     │  ├─ 0.122 MPI_Wait
   │     │  └─ 0.104 MPI_Waitall
   │     ├─ 0.008 MPI_Irecv
   │     ├─ 0.015 MPI_Isend
   │     ├─ 0.072 MPI_Wait
   │     └─ 0.024 MPI_Waitall
   └─ 0.822 TimeIncrement
      └─ 0.820 MPI_Allreduce
```

=

```
nan MPI_Allreduce
nan MPI_Bcast
nan MPI_Comm_dup
nan MPI_Comm_free
nan MPI_Comm_split
nan MPI_Gather
nan MPI_Initialized
0.099 main
├─ nan MPI_Barrier
├─ nan MPI_Irecv
├─ nan MPI_Isend
├─ nan MPI_Reduce
├─ nan MPI_Wait
├─ nan MPI_Waitall
└─ 0.099 lulesh.cycle
   ├─ 0.135 LagrangeLeapFrog
   │  ├─ 0.621 CalcTimeConstraintsForElems
   │  ├─ 0.226 LagrangeElements
   │  │  ├─ 0.361 ApplyMaterialPropertiesForElems
   │  │  │  └─ 0.368 EvalEOSForElems
   │  │  │     └─ 0.411 CalcEnergyForElems
   │  │  ├─ 0.160 CalcLagrangeElements
   │  │  │  └─ 0.159 CalcKinematicsForElems
   │  │  └─ 0.077 CalcQForElems
   │  │     ├─ 0.181 CalcMonotonicQForElems
   │  │     ├─ nan MPI_Irecv
   │  │     ├─ nan MPI_Isend
   │  │     ├─ nan MPI_Wait
   │  │     └─ nan MPI_Waitall
   │  └─ 0.065 LagrangeNodal
   │     ├─ 0.072 CalcForceForNodes
   │     │  ├─ 0.102 CalcVolumeForceForElems
   │     │  │  ├─ 0.094 CalcHourglassControlForElems
   │     │  │  │  └─ 0.176 CalcFBHourglassForceForElems
   │     │  │  └─ 0.149 IntegrateStressForElems
   │     │  ├─ nan MPI_Irecv
   │     │  ├─ nan MPI_Isend
   │     │  ├─ nan MPI_Wait
   │     │  └─ nan MPI_Waitall
   │     ├─ nan MPI_Irecv
   │     ├─ nan MPI_Isend
   │     ├─ nan MPI_Wait
   │     └─ nan MPI_Waitall
   └─ 0.001 TimeIncrement
      └─ nan MPI_Allreduce
```

```
Legend (Metric: time (inc))
0.56 - 0.62
0.43 - 0.56
0.31 - 0.43
0.19 - 0.31
0.06 - 0.19
0.00 - 0.06
```

Computing speedup of two trees (invert color scheme of result)

1 Rank

```
0.299 main
└─ 0.299 lulesh.cycle
   ├─ 0.297 LagrangeLeapFrog
   │  ├─ 0.011 CalcTimeConstraintsForElems
   │  ├─ 0.203 LagrangeElements
   │  │  ├─ 0.152 ApplyMaterialPropertiesForElems
   │  │  │  └─ 0.149 EvalEOSForElems
   │  │  │     └─ 0.109 CalcEnergyForElems
   │  │  ├─ 0.027 CalcLagrangeElements
   │  │  │  └─ 0.025 CalcKinematicsForElems
   │  │  └─ 0.023 CalcQForElems
   │  │     └─ 0.008 CalcMonotonicQForElems
   │  └─ 0.083 LagrangeNodal
   │     └─ 0.079 CalcForceForNodes
   │        └─ 0.077 CalcVolumeForceForElems
   │           ├─ 0.053 CalcHourglassControlForElems
   │           │  └─ 0.032 CalcFBHourglassForceForElems
   │           └─ 0.022 IntegrateStressForElems
   └─ 0.001 TimeIncrement
```

/

64 Ranks

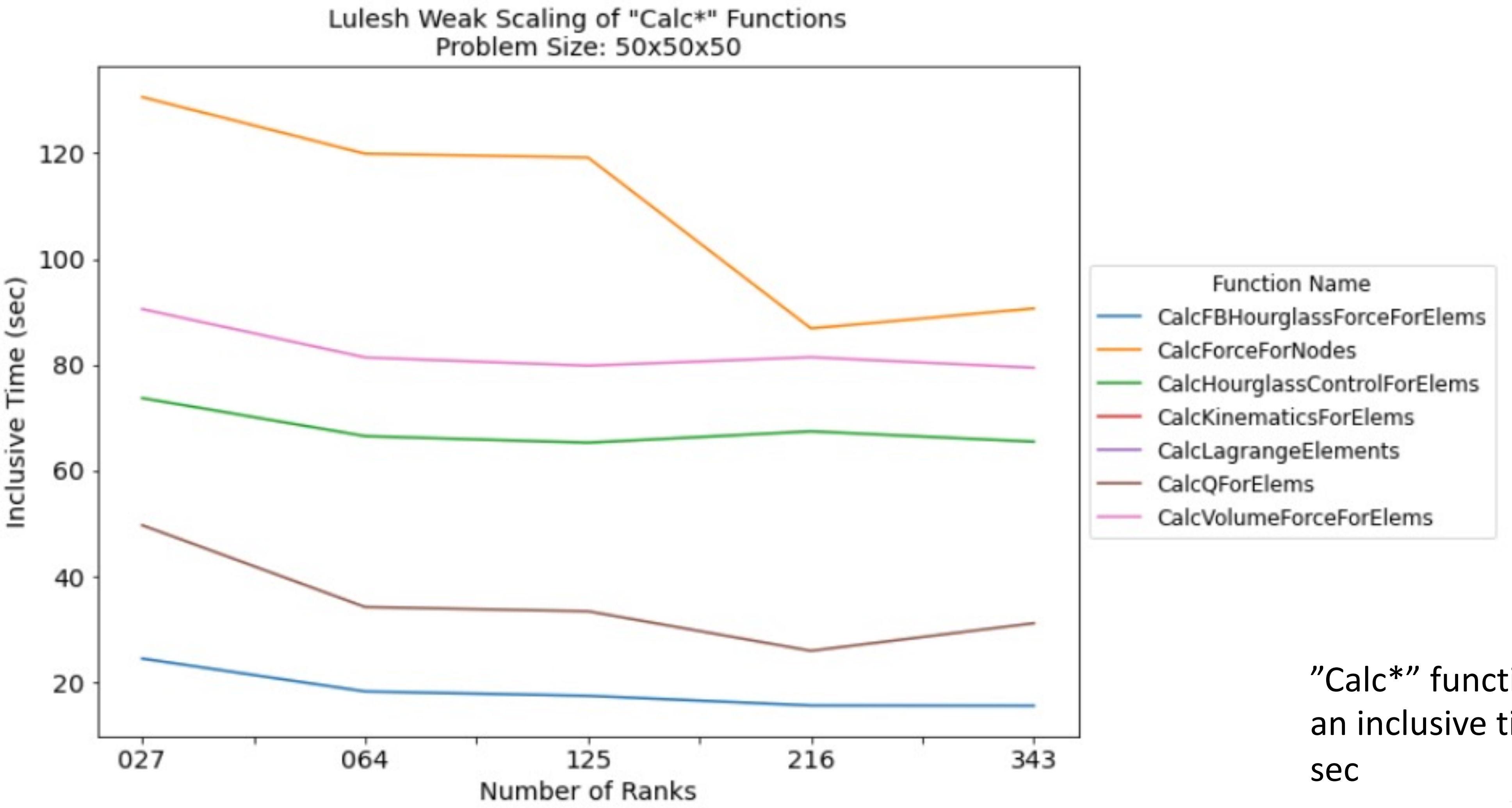
```
0.000 MPI_Allreduce
0.001 MPI_Bcast
0.002 MPI_Comm_dup
0.000 MPI_Comm_free
0.001 MPI_Comm_split
0.000 MPI_Gather
0.000 MPI_Initialized
3.031 main
├─ 0.000 MPI_Barrier
├─ 0.001 MPI_Irecv
├─ 0.000 MPI_Isend
├─ 0.000 MPI_Reduce
├─ 0.000 MPI_Wait
├─ 0.000 MPI_Waitall
└─ 3.029 lulesh.cycle
   ├─ 2.203 LagrangeLeapFrog
   │  ├─ 0.017 CalcTimeConstraintsForElems
   │  ├─ 0.899 LagrangeElements
   │  │  ├─ 0.420 ApplyMaterialPropertiesForElems
   │  │  │  └─ 0.405 EvalEOSForElems
   │  │  │     └─ 0.265 CalcEnergyForElems
   │  │  ├─ 0.167 CalcLagrangeElements
   │  │  │  └─ 0.158 CalcKinematicsForElems
   │  │  └─ 0.306 CalcQForElems
   │  │     ├─ 0.045 CalcMonotonicQForElems
   │  │     ├─ 0.005 MPI_Irecv
   │  │     ├─ 0.010 MPI_Isend
   │  │     ├─ 0.103 MPI_Wait
   │  │     └─ 0.022 MPI_Waitall
   │  └─ 1.283 LagrangeNodal
   │     ├─ 1.100 CalcForceForNodes
   │     │  ├─ 0.756 CalcVolumeForceForElems
   │     │  │  ├─ 0.570 CalcHourglassControlForElems
   │     │  │  │  └─ 0.185 CalcFBHourglassForceForElems
   │     │  │  └─ 0.144 IntegrateStressForElems
   │     │  ├─ 0.015 MPI_Irecv
   │     │  ├─ 0.027 MPI_Isend
   │     │  ├─ 0.122 MPI_Wait
   │     │  └─ 0.104 MPI_Waitall
   │     ├─ 0.008 MPI_Irecv
   │     ├─ 0.015 MPI_Isend
   │     ├─ 0.072 MPI_Wait
   │     └─ 0.024 MPI_Waitall
   └─ 0.822 TimeIncrement
      └─ 0.820 MPI_Allreduce
```

=

```
nan MPI_Allreduce
nan MPI_Bcast
nan MPI_Comm_dup
nan MPI_Comm_free
nan MPI_Comm_split
nan MPI_Gather
nan MPI_Initialized
0.099 main
├─ nan MPI_Barrier
├─ nan MPI_Irecv
├─ nan MPI_Isend
├─ nan MPI_Reduce
├─ nan MPI_Wait
├─ nan MPI_Waitall
└─ 0.099 lulesh.cycle
   ├─ 0.135 LagrangeLeapFrog
   │  ├─ 0.621 CalcTimeConstraintsForElems
   │  ├─ 0.226 LagrangeElements
   │  │  ├─ 0.361 ApplyMaterialPropertiesForElems
   │  │  │  └─ 0.368 EvalEOSForElems
   │  │  │     └─ 0.411 CalcEnergyForElems
   │  │  ├─ 0.160 CalcLagrangeElements
   │  │  │  └─ 0.159 CalcKinematicsForElems
   │  │  └─ 0.077 CalcQForElems
   │  │     ├─ 0.181 CalcMonotonicQForElems
   │  │     ├─ nan MPI_Irecv
   │  │     ├─ nan MPI_Isend
   │  │     ├─ nan MPI_Wait
   │  │     └─ nan MPI_Waitall
   │  └─ 0.065 LagrangeNodal
   │     ├─ 0.072 CalcForceForNodes
   │     │  ├─ 0.102 CalcVolumeForceForElems
   │     │  │  ├─ 0.094 CalcHourglassControlForElems
   │     │  │  │  └─ 0.176 CalcFBHourglassForceForElems
   │     │  │  └─ 0.149 IntegrateStressForElems
   │     │  ├─ nan MPI_Irecv
   │     │  ├─ nan MPI_Isend
   │     │  ├─ nan MPI_Wait
   │     │  └─ nan MPI_Waitall
   │     ├─ nan MPI_Irecv
   │     ├─ nan MPI_Isend
   │     ├─ nan MPI_Wait
   │     └─ nan MPI_Waitall
   └─ 0.001 TimeIncrement
      └─ nan MPI_Allreduce
```

```
Legend (Metric: time (inc))
0.56 - 0.62
0.43 - 0.56
0.31 - 0.43
0.19 - 0.31
0.06 - 0.19
0.00 - 0.06
```

Generate Lulesh weak scaling plot



”Calc*” functions with an inclusive time > 15 sec