Hatchet: API

Stephanie Brink, LLNL

Sub-select the DataFrame using `filter()`

```
filtered_gf = gf.filter(lambda x: x["time"] > 10.0, squash=False)
```

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

node	name	nid	node	time	time (inc)
main	main	0	main	40.0	200.0
physics	physics	1	physics	40.0	60.0
psm2	psm2	3	psm2	15.0	15.0
hypre	hypre	5	hypre	65.0	65.0
psm2	psm2	7	psm2	25.0	25.0

Aggregate the DataFrame across ranks and/or threads

```
gf.drop_index_levels(function=np.max)
```

node	rank	time (inc)	time	nid	name
{'name': 'main', 'type': 'region'}	0	50720641.0	185341.0	0	main
	1	50721042.0	84509.0	0	main
	2	50721017.0	84492.0	0	main
	3	50721004.0	84433.0	0	main

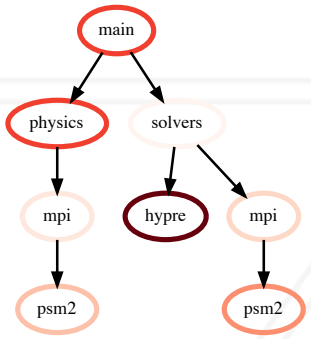
node	time (inc)	time	nid	name
{'name': 'main', 'type': 'region'}	50721169.0	225098.0	0	main
{'name': 'MPI_Finalize', 'type': 'region'}	140374.0	140374.0	17	MPI_Finalize
{'name': 'lulesh.cycle', 'type': 'region'}	50636511.0	2203.0	1	lulesh.cycle
{'name': 'LagrangeLeapFrog', 'type': 'region'}	49948805.0	1886.0	2	LagrangeLeapFrog
{'name': 'CalcTimeConstraintsForElems', 'type': 'region'}	193079.0	4887.0	11	CalcTimeConstraintsForElems

Graph operation: squash

```
filtered_gf = gf.filter(lambda x: x["time"] > 10.0, squash=False)
```

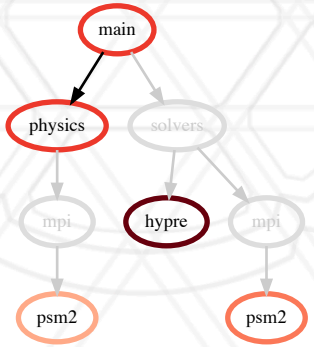
```
squashed_gf = filtered_gf.squash()
```

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



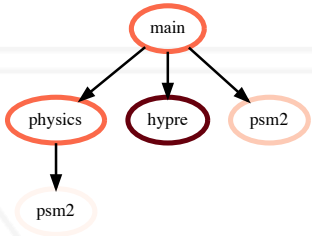
filter

node	name	nid	node	time	time (inc)
main	main	0	main	40.0	200.0
physics	physics	1	physics	40.0	60.0
psm2	psm2	3	psm2	15.0	15.0
hypre	hypre	5	hypre	65.0	65.0
psm2	psm2	7	psm2	25.0	25.0



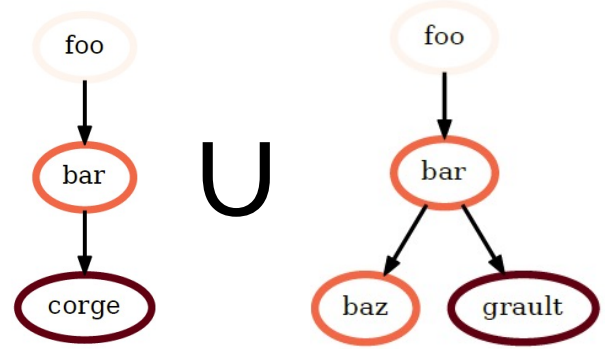
squash

node	name	nid	node	time	time (inc)
main	main	0	main	40.0	200.0
physics	physics	1	physics	40.0	60.0
psm2	psm2	3	psm2	15.0	15.0
hypre	hypre	5	hypre	65.0	65.0
psm2	psm2	7	psm2	25.0	25.0

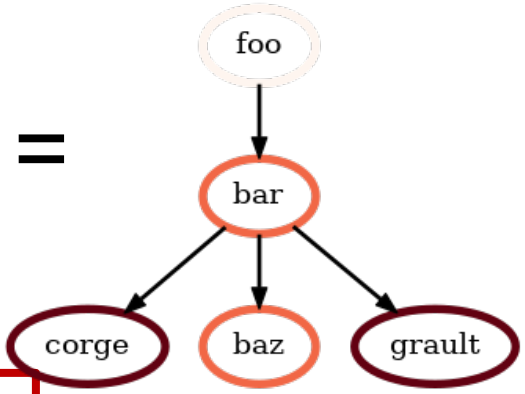


Compare GraphFrames using division (or add, subtract, multiply)

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



*First, unify two trees since structure is different



gf1

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

/

gf2

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

=

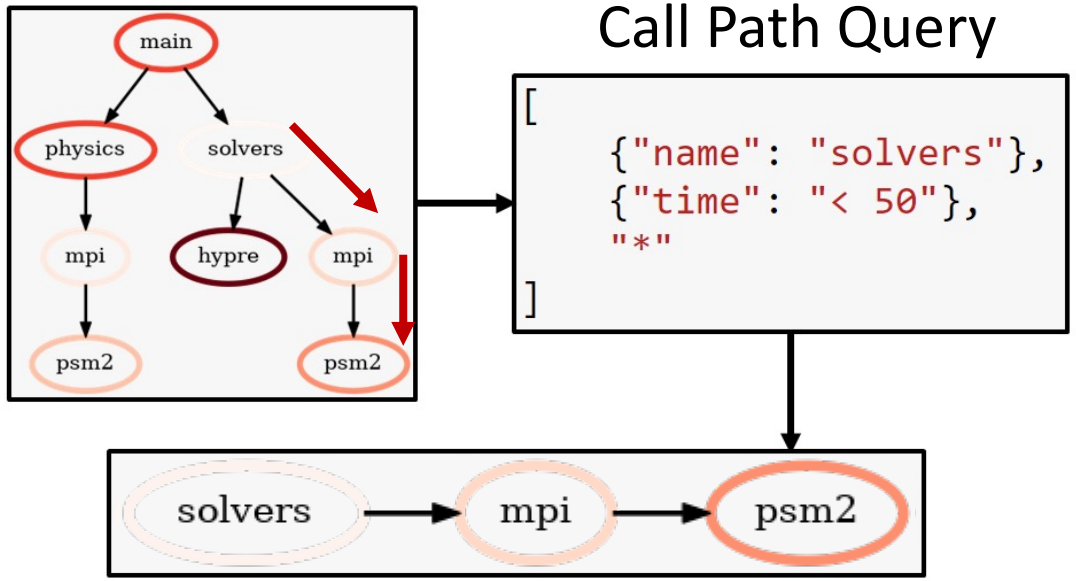
gf3

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

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

Hatchet's Call Path Query Language

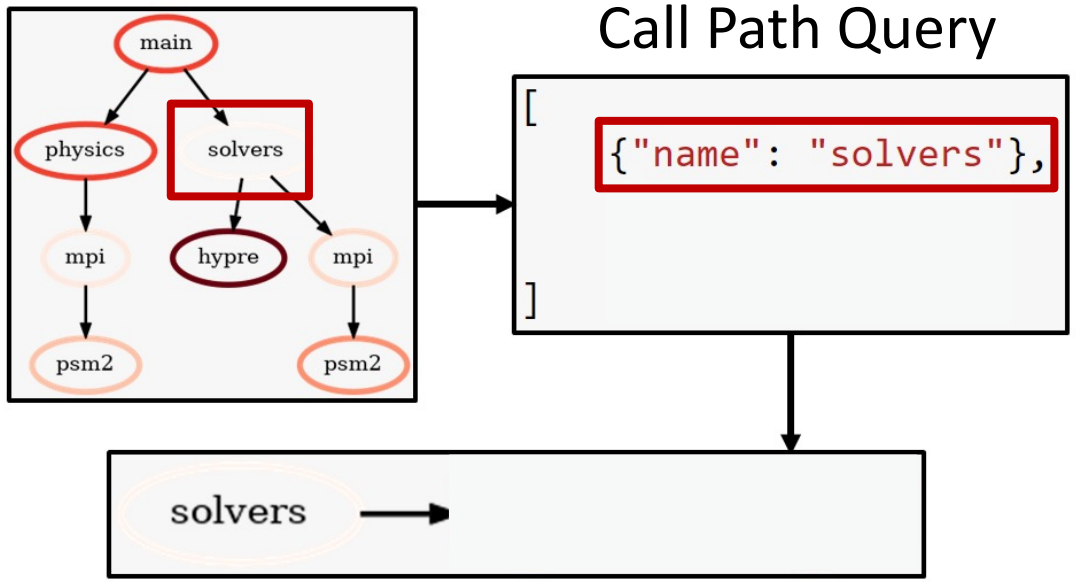
- Existing filter capability did not leverage relational caller-callee data
- Data reduction using *call path* pattern matching
- Query provided in the form of a list of abstract graph nodes to match call paths
  - Filter: { X: Y }
    - X: dataframe field
    - Y: user-specified filter
  - Wildcard: (Z, {X: Y})
    - Z: "." (1 node), "+" (1+ nodes), "*" (0+ nodes)



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.

Hatchet's Call Path Query Language

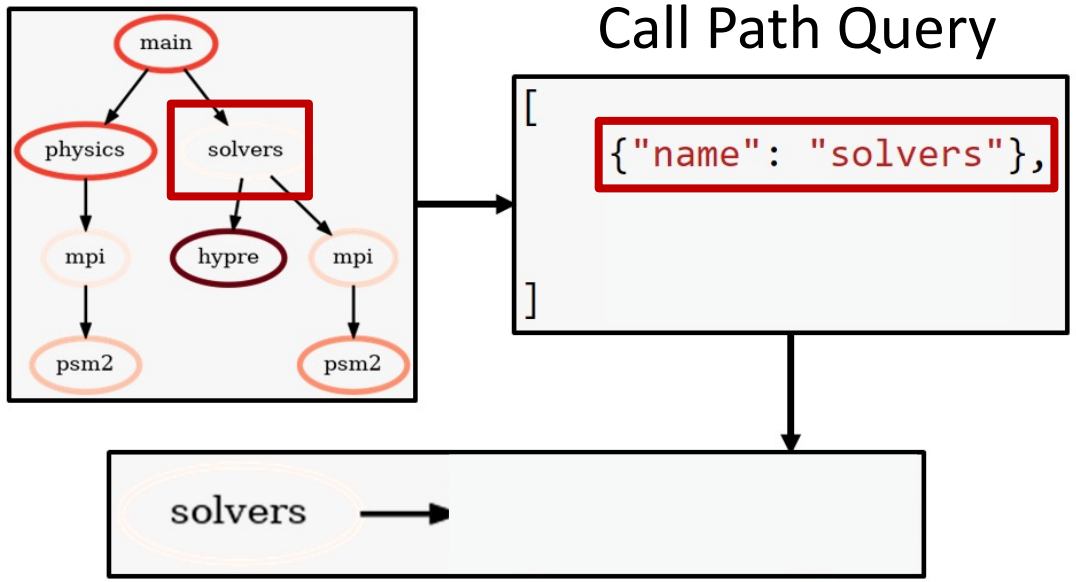
- Existing filter capability did not leverage relational caller-callee data
- Data reduction using *call path* pattern matching
- Query provided in the form of a list of abstract graph nodes to match call paths
  - Filter: { X: Y }
    - X: dataframe field
    - Y: user-specified filter
  - Wildcard: (Z, {X: Y})
    - Z: "." (1 node), "+" (1+ nodes), "*" (0+ nodes)



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.

Hatchet's Call Path Query Language

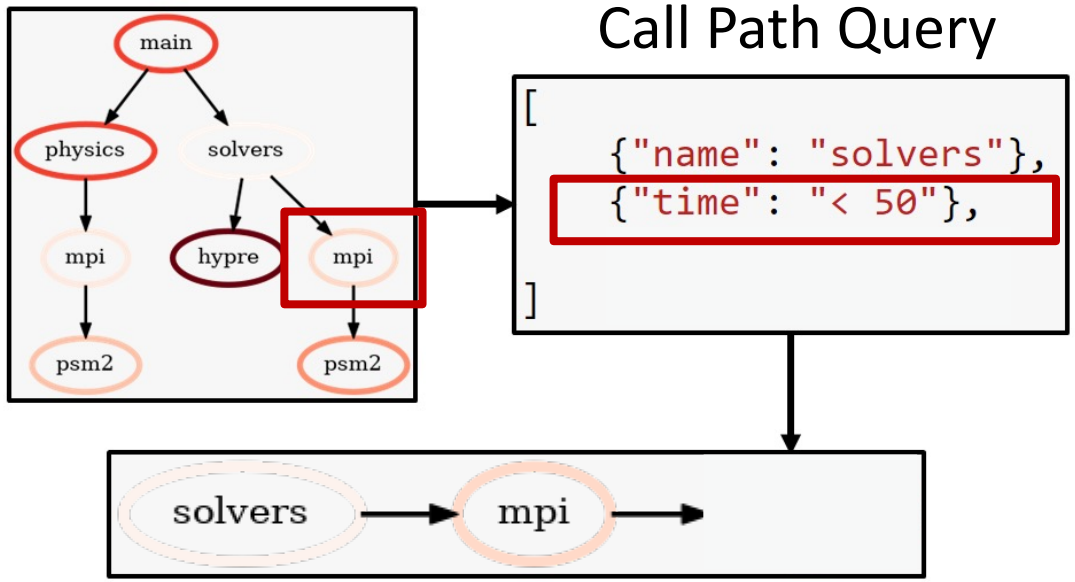
- Existing filter capability did not leverage relational caller-callee data
- Data reduction using *call path* pattern matching
- Query provided in the form of a list of abstract graph nodes to match call paths
  - Filter: { X: Y }
    - X: dataframe field
    - Y: user-specified filter
  - Wildcard: (Z, {X: Y})
    - Z: "." (1 node), "+" (1+ nodes), "*" (0+ nodes)



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.

Hatchet's Call Path Query Language

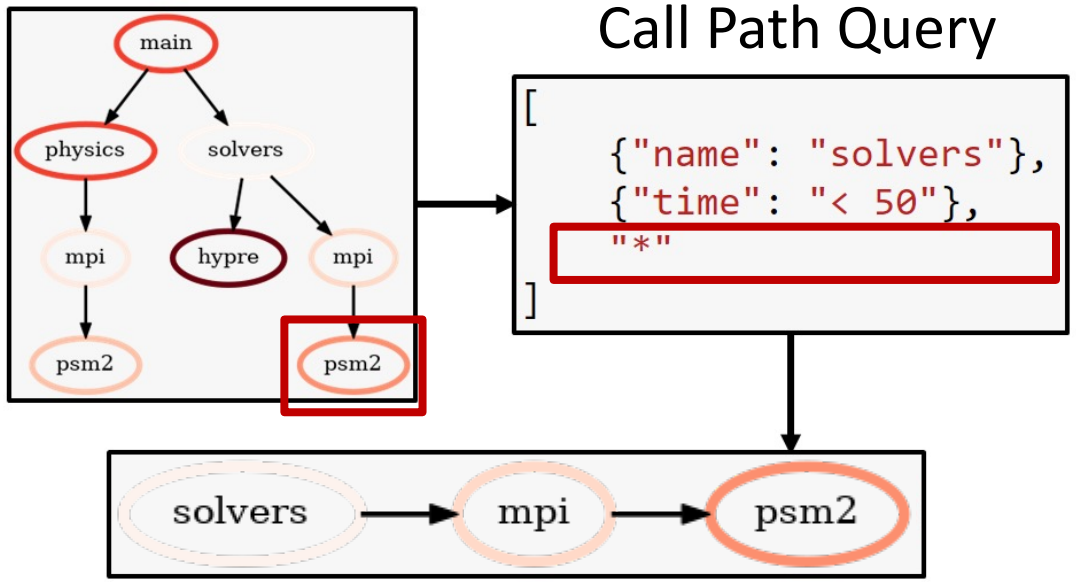
- Existing filter capability did not leverage relational caller-callee data
- Data reduction using *call path* pattern matching
- Query provided in the form of a list of abstract graph nodes to match call paths
  - Filter: { X: Y }
    - X: dataframe field
    - Y: user-specified filter
  - Wildcard: (Z, {X: Y})
    - Z: "." (1 node), "+" (1+ nodes), "*" (0+ nodes)



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.

Hatchet's Call Path Query Language

- Existing filter capability did not leverage relational caller-callee data
- Data reduction using *call path* pattern matching
- Query provided in the form of a list of abstract graph nodes to match call paths
  - Filter: { X: Y }
    - X: dataframe field
    - Y: user-specified filter
  - Wildcard: (Z, {X: Y})
    - Z: "." (1 node), "+" (1+ nodes), "*" (0+ nodes)



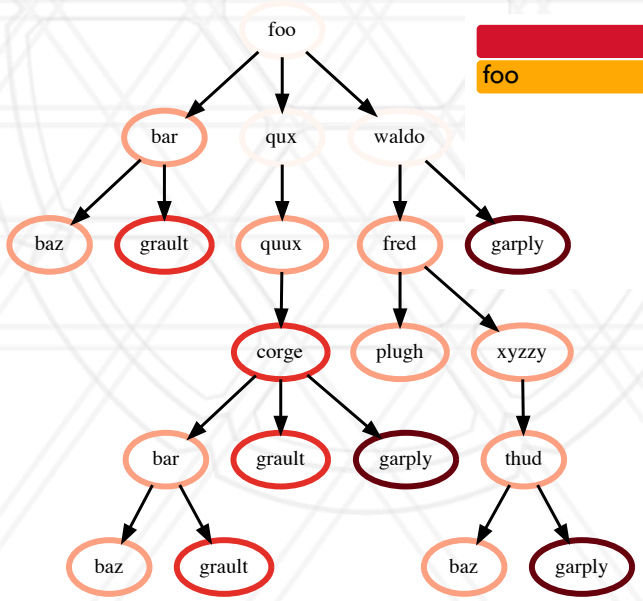
Matches a call path (1) rooted at a node with name "solvers", (2) followed by a node with a time metric value less than 50, and (3) followed by any number of children nodes.

Visualizing *small* graphs

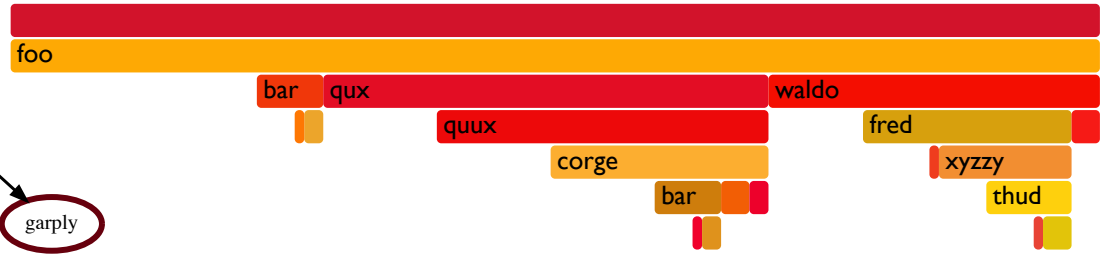
```
print(gf.tree())
```

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

```
with open("test.dot", "w") as dot_file:
    dot_file.write(gf.to_dot())
```




```
with open("test.txt", "w") as folded_stack:
    folded_stack.write(gf.to_flamegraph())
```



Flamegraph

CASC

Center for Applied
Scientific Computing

Lawrence Livermore
National Laboratory