Analyzing Performance Profiles Using Hatchet

Abhinav Bhatele, Department of Computer Science

UCDAVIS

Lawrence Livermore National Laboratory

UNIVERSITY OF MARYLAND

THE UNIVERSITY OF TENNESSEE KNOXVILLE

THE UNIVERSITY OF ARIZONA

Code-centric performance analysis

- Understanding performance bottlenecks is critical to optimizing (parallel) software
- Profiling and tracing tools help identify parts of code that consume the most time

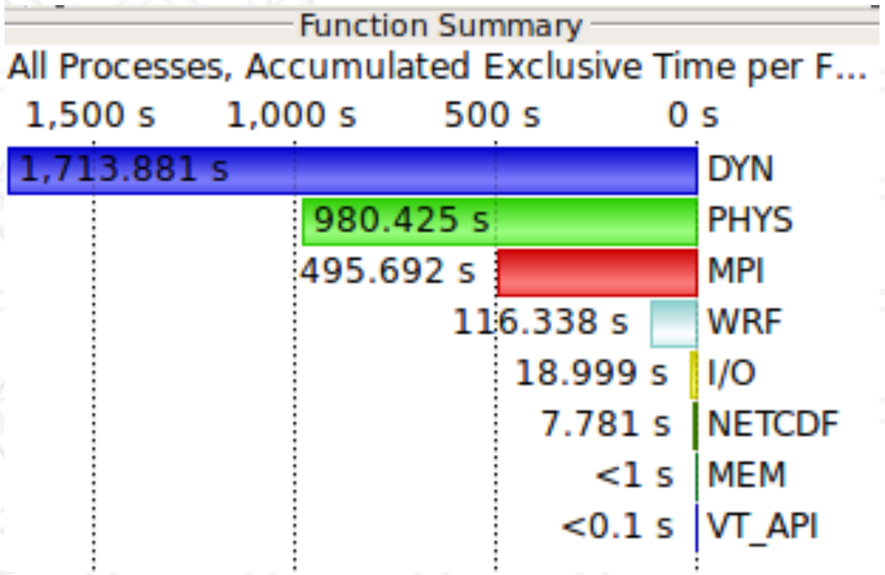
gprof Data

Enter filter text

4 bytes per bucket, each sample counts as 10.000ms

Name (location)	Samples	Calls	Time/Call	% Time
Summary	2228			100.0%
calc.c	590			26.48%
copy.c	0			0.0%
diag.c	25			1.12%
main.c	0			0.0%
time.c	653			29.31%
tstep.c	958			43.0%
tstep	958	10000	957.999us	43.0%
tstep (tstep.c:47)	1			0.04%
tstep (tstep.c:48)	62			2.78%
tstep (tstep.c:49)	46			2.06%
tstep (tstep.c:50)	46			2.06%
tstep (tstep.c:51)	48			2.15%
tstep (tstep.c:58)	101			4.53%
tstep (tstep.c:59)	135			6.06%
tstep (tstep.c:60)	120			5.39%
tstep (tstep.c:61)	126			5.66%
tstep (tstep.c:66)	3			0.13%
tstep (tstep.c:67)	108			4.85%
tstep (tstep.c:68)	63			2.83%
tstep (tstep.c:69)	43			1.93%
tstep (tstep.c:70)	56			2.51%
worker.c	2			0.09%

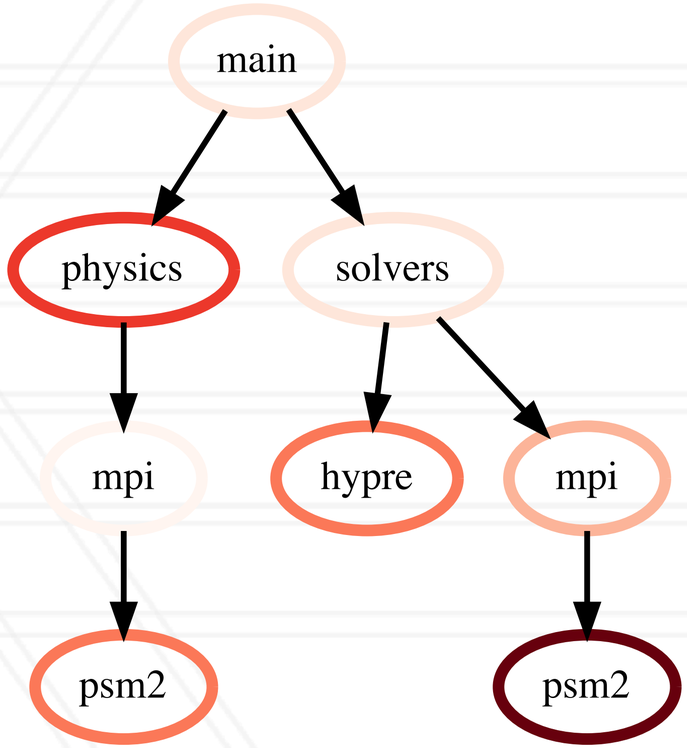
Gprof data in hpctView



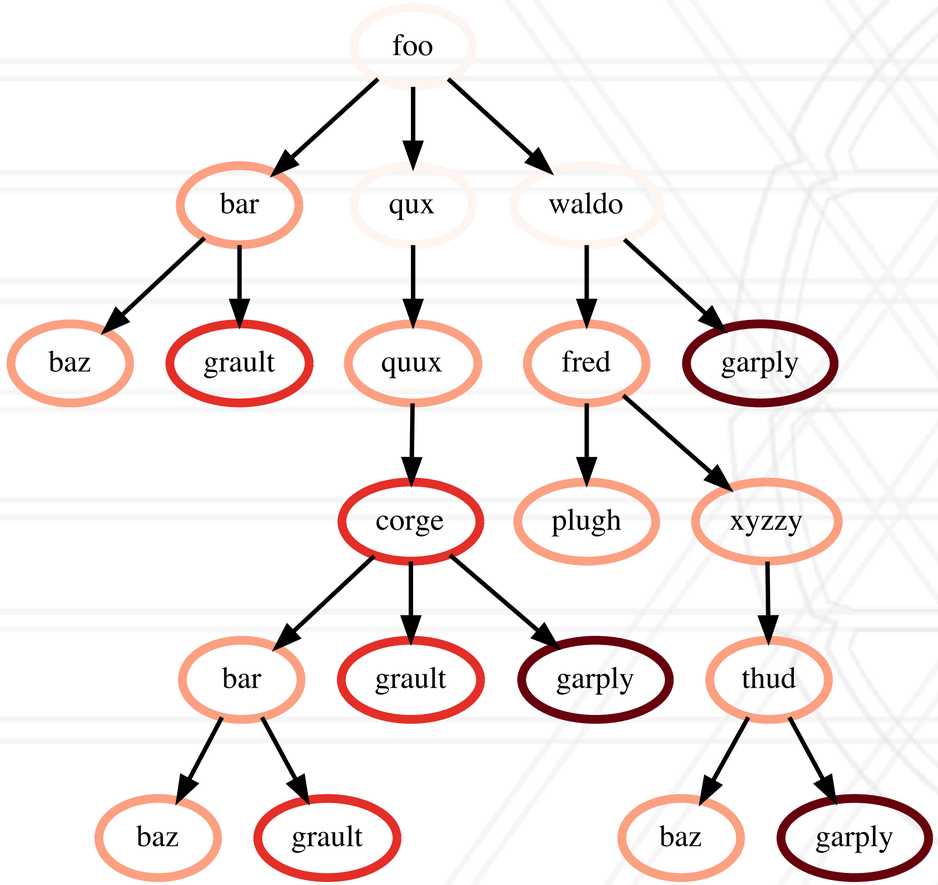
Trace data in Vampir

Attributing performance (time) to different calling contexts

- Requires reasonable understanding of code structure
- Sophisticated profilers can attribute time to calling contexts: where was a function called from?
- Many measurement tools:
  - HPCToolkit, Caliper, Score-P, TAU, Gprof, Callgrind, perf, Timemory
  - cProfile, pyinstrument

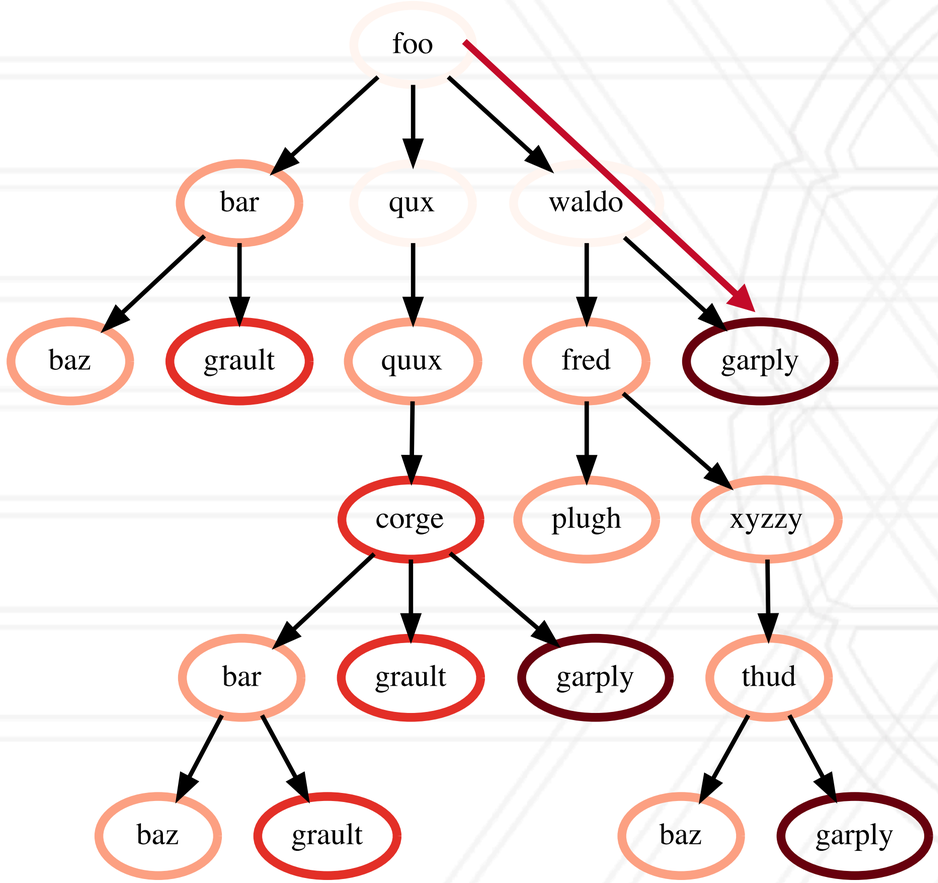


Calling context trees, call graphs, …



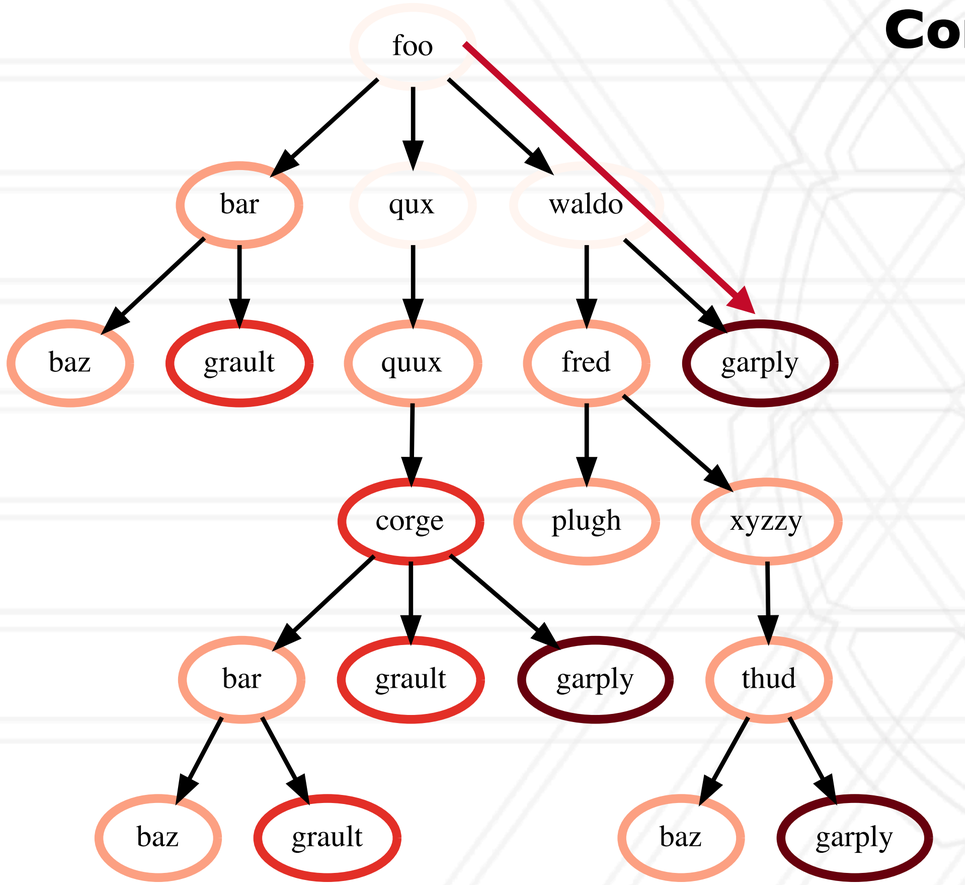
Calling context tree (CCT)

Calling context trees, call graphs, …



Calling context tree (CCT)

Calling context trees, call graphs, …

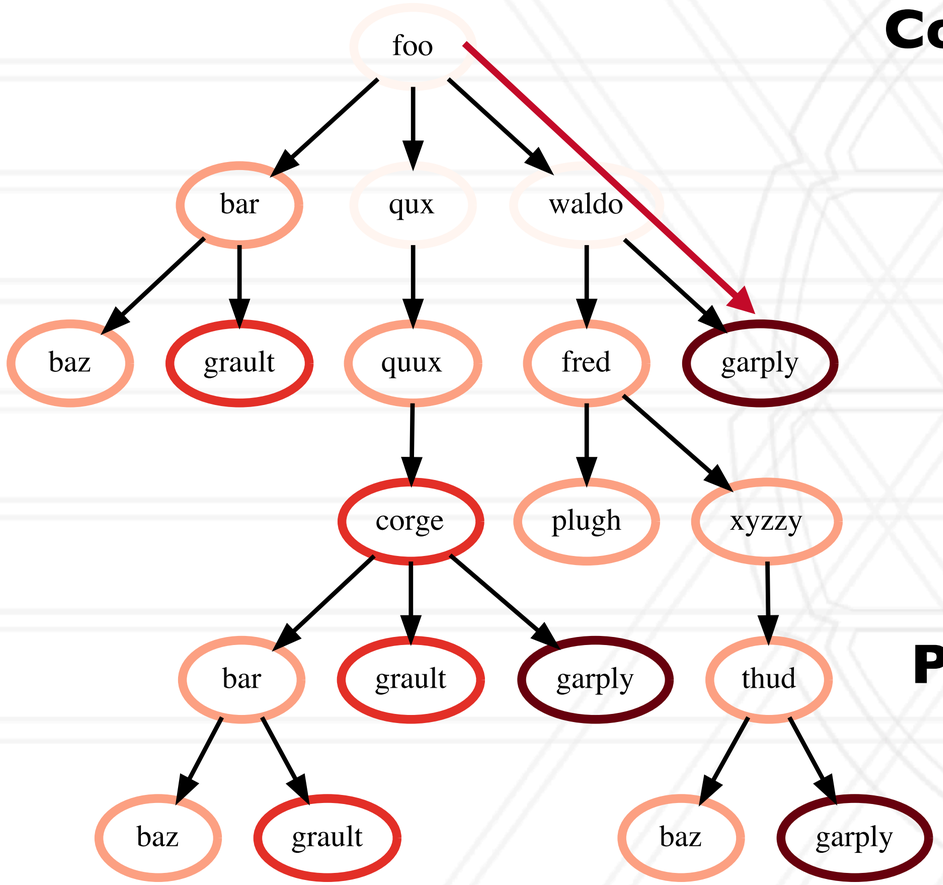


Calling context tree (CCT)

Contextual information

```
File
Line number
Function name
Callpath
Load module
Process ID
Thread ID
```

Calling context trees, call graphs, …



Calling context tree (CCT)

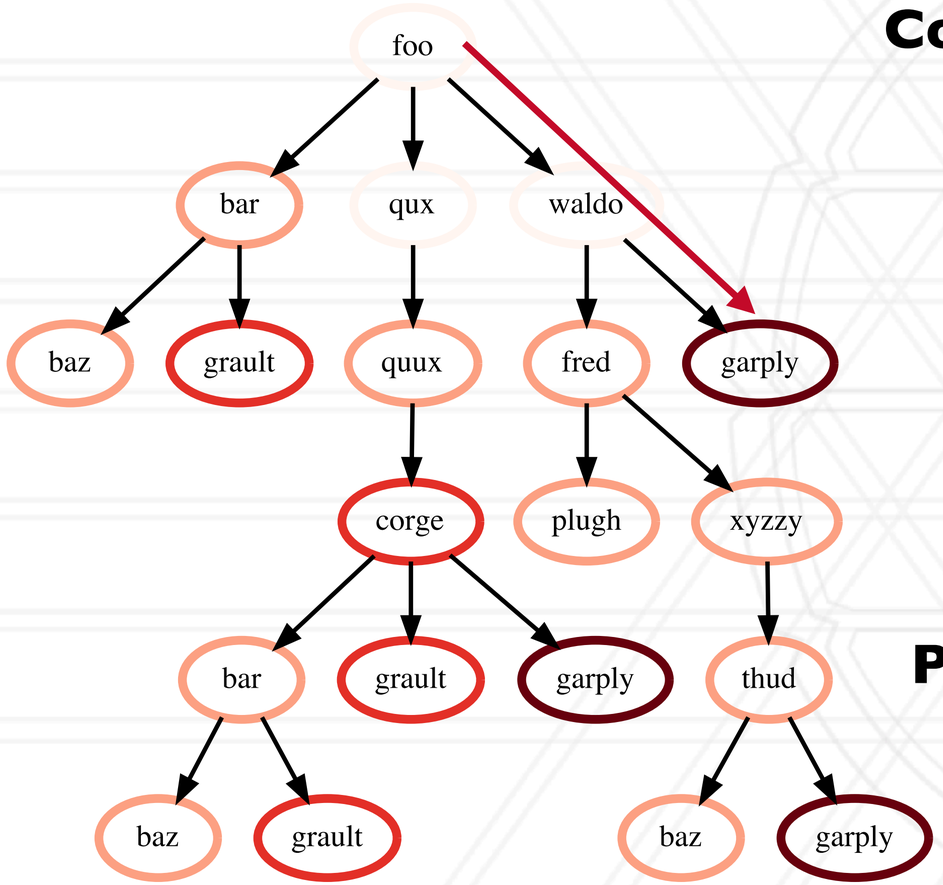
Contextual information

- File
- Line number
- Function name
- Callpath
- Load module
- Process ID
- Thread ID

Performance Metrics

- Time
- Flops
- Cache misses

Calling context trees, call graphs, …



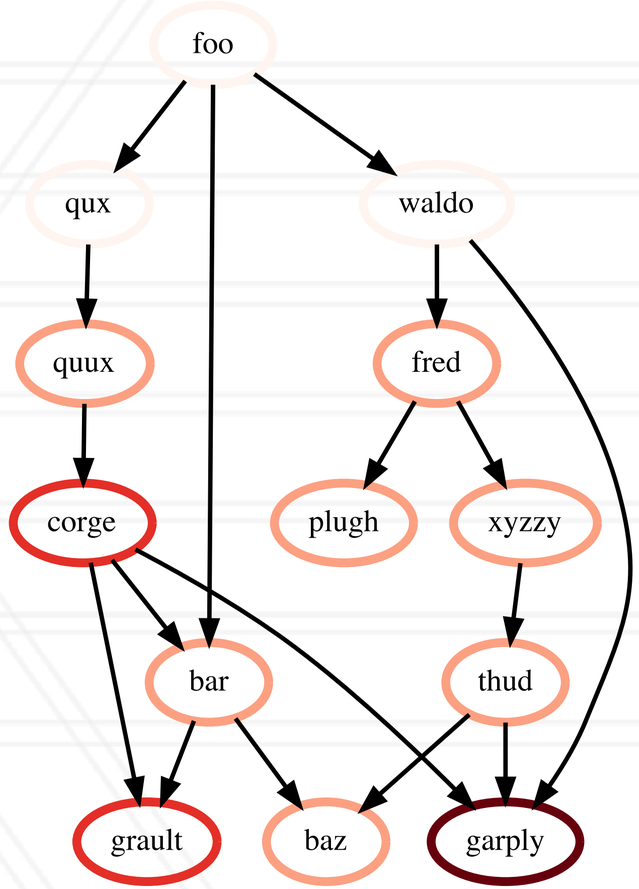
Calling context tree (CCT)

Contextual information

- File
- Line number
- Function name
- Callpath
- Load module
- Process ID
- Thread ID

Performance Metrics

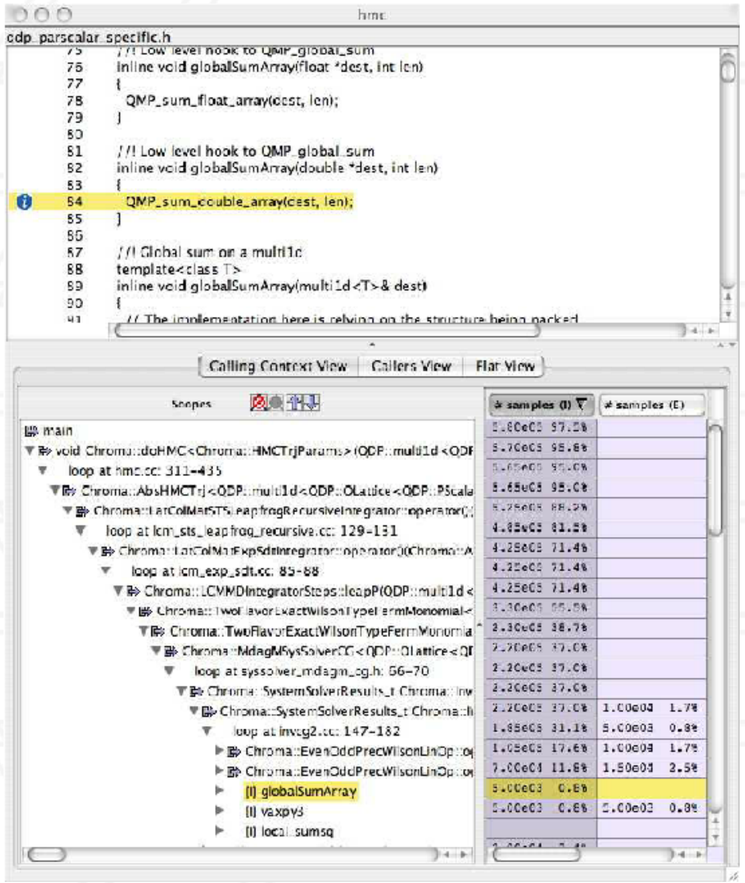
- Time
- Flops
- Cache misses



Call graph

Limitations of current *analysis* tools

- Support their own unique format(s)
- Limited support for saving or automating analysis
- Most tools only support viewing one dataset at a time
- Lack capabilities to sub-select and focus on specific parts

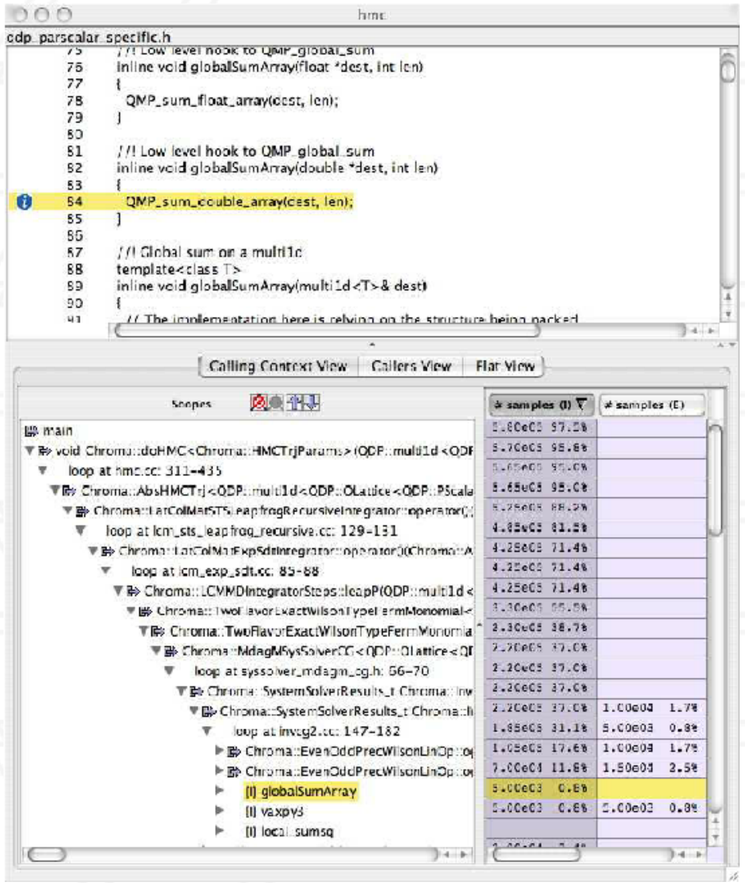


hpcviewer's GUI

Limitations of current *analysis* tools

- Support their own unique format(s)
- Limited support for saving or automating analysis
- Most tools only support viewing one dataset at a time
- Lack capabilities to sub-select and focus on specific parts

Do not enable programmatic analysis of the data by the end user



hpcviewer's GUI

The main idea behind Hatchet

- A Python-based library to enable programmatic analysis
- Creates an in-memory representation of the graph
- Leverage pandas which supports multi-dimensional tabular datasets
  - Use graph as structured index to index pandas dataframes
- A set of operators to sub-select and/or aggregate profile data
- A set of operators to compare multiple datasets

Pandas and dataframes

Pandas and dataframes

- Pandas is an open-source Python library for data analysis

Pandas and dataframes

- Pandas is an open-source Python library for data analysis
- Dataframe: two-dimensional tabular data structure
  - Supports many operations borrowed from SQL databases

Columns

Rows

	node	name	time (inc)	time
0	{'name': 'main'}	main	200.0	10.0
1	{'name': 'physics'}	physics	60.0	40.0
2	{'name': 'mpi'}	mpi	20.0	5.0
3	{'name': 'psm2'}	psm2	15.0	30.0
4	{'name': 'solvers'}	solvers	100.0	10.0
5	{'name': 'hypre'}	hypre	65.0	30.0
6	{'name': 'mpi'}	mpi	35.0	20.0
7	{'name': 'psm2'}	psm2	25.0	60.0

Pandas and dataframes

- Pandas is an open-source Python library for data analysis
- Dataframe: two-dimensional tabular data structure
  - Supports many operations borrowed from SQL databases

Index

Columns

Rows

	node	name	time (inc)	time
0	{'name': 'main'}	main	200.0	10.0
1	{'name': 'physics'}	physics	60.0	40.0
2	{'name': 'mpi'}	mpi	20.0	5.0
3	{'name': 'psm2'}	psm2	15.0	30.0
4	{'name': 'solvers'}	solvers	100.0	10.0
5	{'name': 'hypre'}	hypre	65.0	30.0
6	{'name': 'mpi'}	mpi	35.0	20.0
7	{'name': 'psm2'}	psm2	25.0	60.0

Pandas and dataframes

- Pandas is an open-source Python library for data analysis
- Dataframe: two-dimensional tabular data structure
  - Supports many operations borrowed from SQL databases
- MultiIndex enables working with high-dimensional data in a 2D data structure

Index

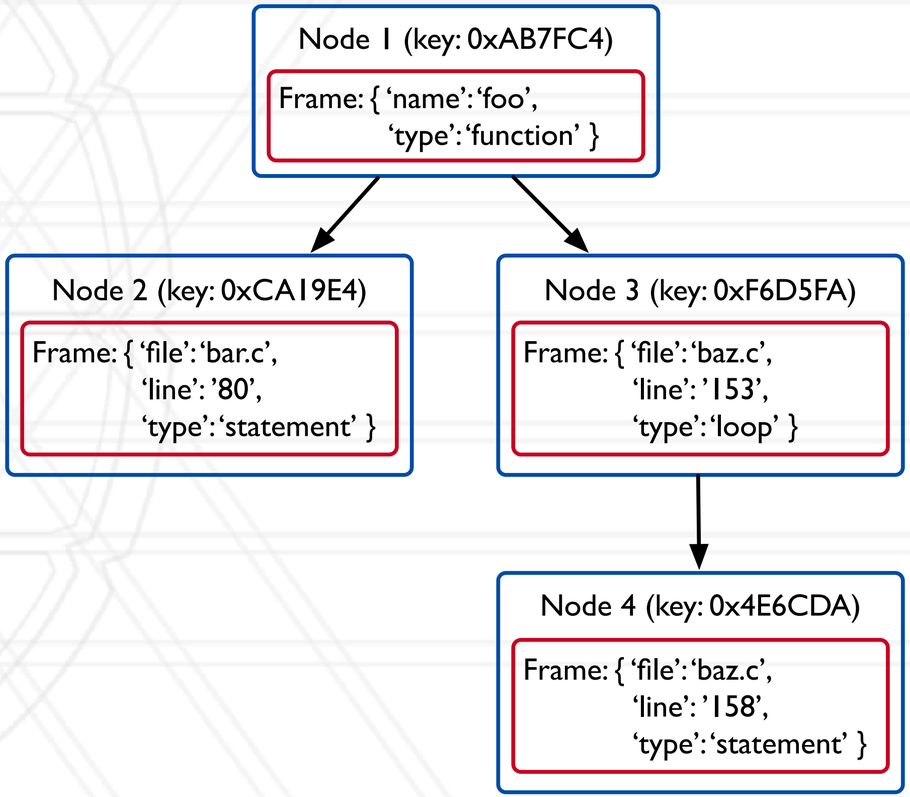
Columns

Rows

	node	name	time (inc)	time
0	{'name': 'main'}	main	200.0	10.0
1	{'name': 'physics'}	physics	60.0	40.0
2	{'name': 'mpi'}	mpi	20.0	5.0
3	{'name': 'psm2'}	psm2	15.0	30.0
4	{'name': 'solvers'}	solvers	100.0	10.0
5	{'name': 'hypre'}	hypre	65.0	30.0
6	{'name': 'mpi'}	mpi	35.0	20.0
7	{'name': 'psm2'}	psm2	25.0	60.0

Canonical data model: *a structured index*

- Structured index is basically an in-memory graph
- Each node is assigned a unique key, which enables using the nodes as the index in the dataframe
- Each node has a *Frame* that describes the code it represents
  - a set of key/value pairs
- Frames don't have a rigid schema
- *Nodes* define the structure and connectivity

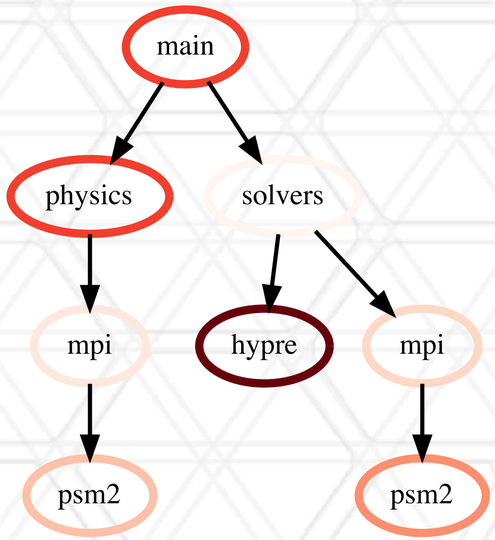


Central data structure: a *GraphFrame*

- Consists of a structured index graph object and a pandas dataframe
- Graph stores caller-callee relationships
- Dataframe stores all numerical and categorical data

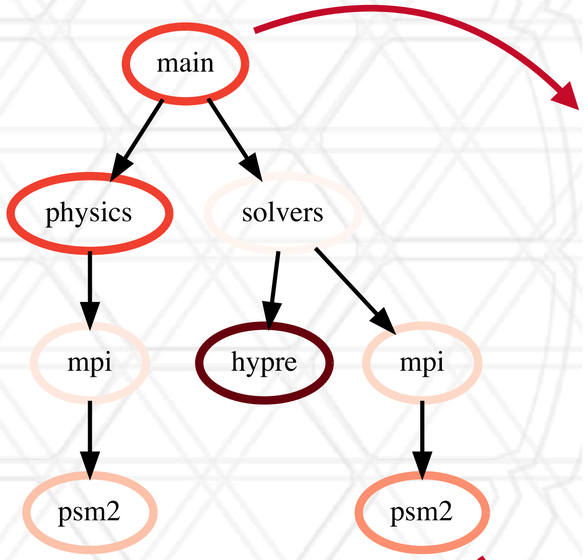
Central data structure: a *GraphFrame*

- Consists of a structured index graph object and a pandas dataframe
- Graph stores caller-callee relationships
- Dataframe stores all numerical and categorical data



Central data structure: a *GraphFrame*

- Consists of a structured index graph object and a pandas dataframe
- Graph stores caller-callee relationships
- Dataframe stores all numerical and categorical data



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

Use of MultiIndex

- When metrics are per MPI process, thread etc….

node	rank	time (inc)	time	nid	rank	file	line	module
{'type': 'function', 'name': '<program root>'}	0	999238.0	0.0	2	0	<unknown file>	0	/collab/usr/global/tools/hpctoolkit/chaos_5_x8...
	1	999390.0	0.0	2	1	<unknown file>	0	/collab/usr/global/tools/hpctoolkit/chaos_5_x8...
{'type': 'function', 'name': '27:MPI_Init'}	0	0.0	0.0	6	0	<unknown file>	0	/collab/usr/global/tools/hpctoolkit/chaos_5_x8...
	1	0.0	0.0	6	1	<unknown file>	0	/collab/usr/global/tools/hpctoolkit/chaos_5_x8...
{'type': 'function', 'name': 'main'}	0	999238.0	0.0	4	0	./src/cpi.c	19	cpi
	1	999390.0	0.0	4	1	./src/cpi.c	19	cpi

Use of MultiIndex

- When metrics are per MPI process, thread etc….

Index

node	rank	time (inc)	time	nid	rank	file	line	module
{'type': 'function', 'name': '<program root>'}	0	999238.0	0.0	2	0	<unknown file>	0	/collab/usr/global/tools/hpctoolkit/chaos_5_x8...
	1	999390.0	0.0	2	1	<unknown file>	0	/collab/usr/global/tools/hpctoolkit/chaos_5_x8...
{'type': 'function', 'name': '27:MPI_Init'}	0	0.0	0.0	6	0	<unknown file>	0	/collab/usr/global/tools/hpctoolkit/chaos_5_x8...
	1	0.0	0.0	6	1	<unknown file>	0	/collab/usr/global/tools/hpctoolkit/chaos_5_x8...
{'type': 'function', 'name': 'main'}	0	999238.0	0.0	4	0	./src/cpi.c	19	cpi
	1	999390.0	0.0	4	1	./src/cpi.c	19	cpi

Immutable semantics for graph nodes

- Having direct references to graph nodes in the dataframe is risky
  - In particular when graph nodes are shared by multiple graphframes
- Any operation that modifies graph nodes in place creates a new GraphFrame and a new graph index
- Implemented using copy-on-write semantics

Reading in an input dataset

Installing hatchet

```
$ pip install hatchet
```

- HPCToolkit, Caliper, gprof
- Pyinstrument, cprofile
- String literal
- In progress: Timemory, TAU, cube

https://github.com/hatchet/hatchet

Reading in an input dataset

Installing hatchet 

```
$ pip install hatchet
```

- HPCToolkit, Caliper, gprof
- Pyinstrument, cprofile
- String literal
- In progress: Timemory, TAU, cube

https://github.com/hatchet/hatchet

```
import hatchet as ht

if __name__ == "__main__":
    dirname = "hpctoolkit-database"
    gf = ht.GraphFrame.from_hpctoolkit(dirname)
```

Reading in an input dataset

Installing hatchet

```
$ pip install hatchet
```

- HPCToolkit, Caliper, gprof
- Pyinstrument, cprofile
- String literal
- In progress: Timemory, TAU, cube

https://github.com/hatchet/hatchet

Contribute a reader to hatchet!

```
import hatchet as ht

if __name__ == "__main__":
    dirname = "hpctoolkit-database"
    gf = ht.GraphFrame.from_hpctoolkit(dirname)
```

Dataframe operation: filter

```
filtered_gf = gf.filter(lambda x: x['time'] > 10.0)
```

Dataframe operation: filter

```
filtered_gf = gf.filter(lambda x: x['time'] > 10.0)
```

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

Dataframe operation: filter

```
filtered_gf = gf.filter(lambda x: x['time'] > 10.0)
```

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

Dataframe operation: drop_index_levels

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

Dataframe operation: drop_index_levels

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

		nid	rank	time	time (inc)	name
node	rank					
{'type': 'function', 'name': u'main'}	0	0.0	0	83309.0	50905781.0	main
	1	0.0	1	83920.0	50905819.0	main
	2	0.0	2	83023.0	50905849.0	main
	3	0.0	3	83258.0	50905840.0	main

Dataframe operation: drop_index_levels

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

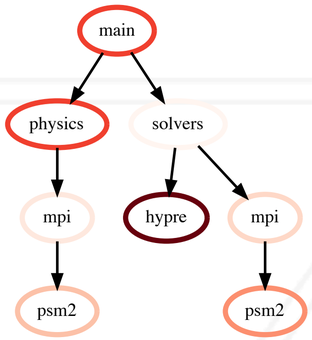
node	rank	nid	rank	time	time (inc)	name
{'type': 'function', 'name': u'main'}	0	0.0	0	83309.0	50905781.0	main
	1	0.0	1	83920.0	50905819.0	main
	2	0.0	2	83023.0	50905849.0	main
	3	0.0	3	83258.0	50905840.0	main

node	node	name
{'type': 'function', 'name': u'main'}	{'type': 'function', 'name': u'main'}	main
{'type': 'function', 'name': u'lulesh.cycle'}	{'type': 'function', 'name': u'lulesh.cycle'}	lulesh.cycle
{'type': 'function', 'name': u'TimeIncrement'}	{'type': 'function', 'name': u'TimeIncrement'}	TimeIncrement
{'type': 'function', 'name': u'LagrangeLeapFrog'}	{'type': 'function', 'name': u'LagrangeLeapFrog'}	LagrangeLeapFrog

Graph operation: squash

```
filtered_gf = gf.filter(lambda x: x['time'] > 10.0)
```

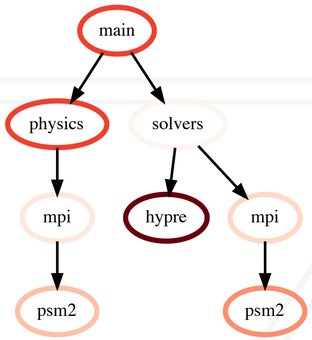
node	name	nid	node	time	time (inc)
main	main	0	main	40.0	200.0
physics	physics	1	physics	40.0	60.0
mpi	mpi	2	mpi	5.0	20.0
psm2	psm2	3	psm2	15.0	15.0
solvers	solvers	4	solvers	0.0	100.0
hypre	hypre	5	hypre	65.0	65.0
mpi	mpi	6	mpi	10.0	35.0
psm2	psm2	7	psm2	25.0	25.0



Graph operation: squash

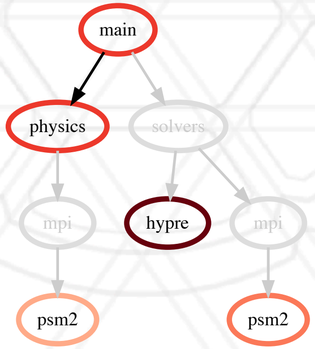
```
filtered_gf = gf.filter(lambda x: x['time'] > 10.0)
```

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

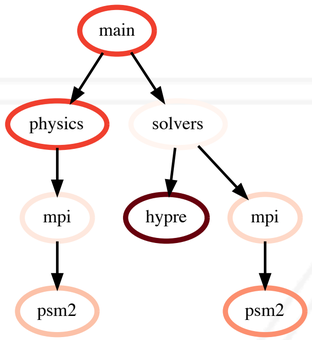


Graph operation: squash

```
filtered_gf = gf.filter(lambda x: x['time'] > 10.0)
```

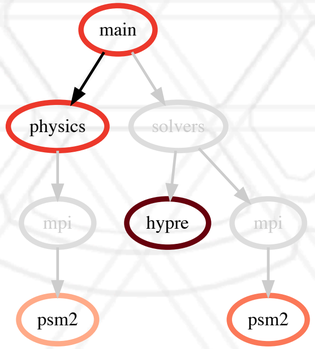
```
squashed_gf = filtered_gf.squash()
```

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

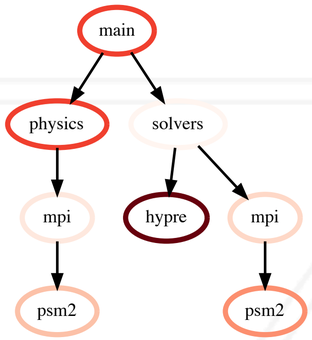


Graph operation: squash

```
filtered_gf = gf.filter(lambda x: x['time'] > 10.0)
```

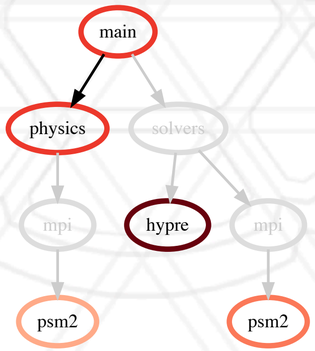
```
squashed_gf = filtered_gf.squash()
```

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



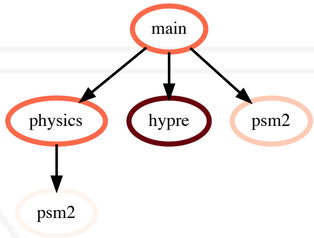
filter

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

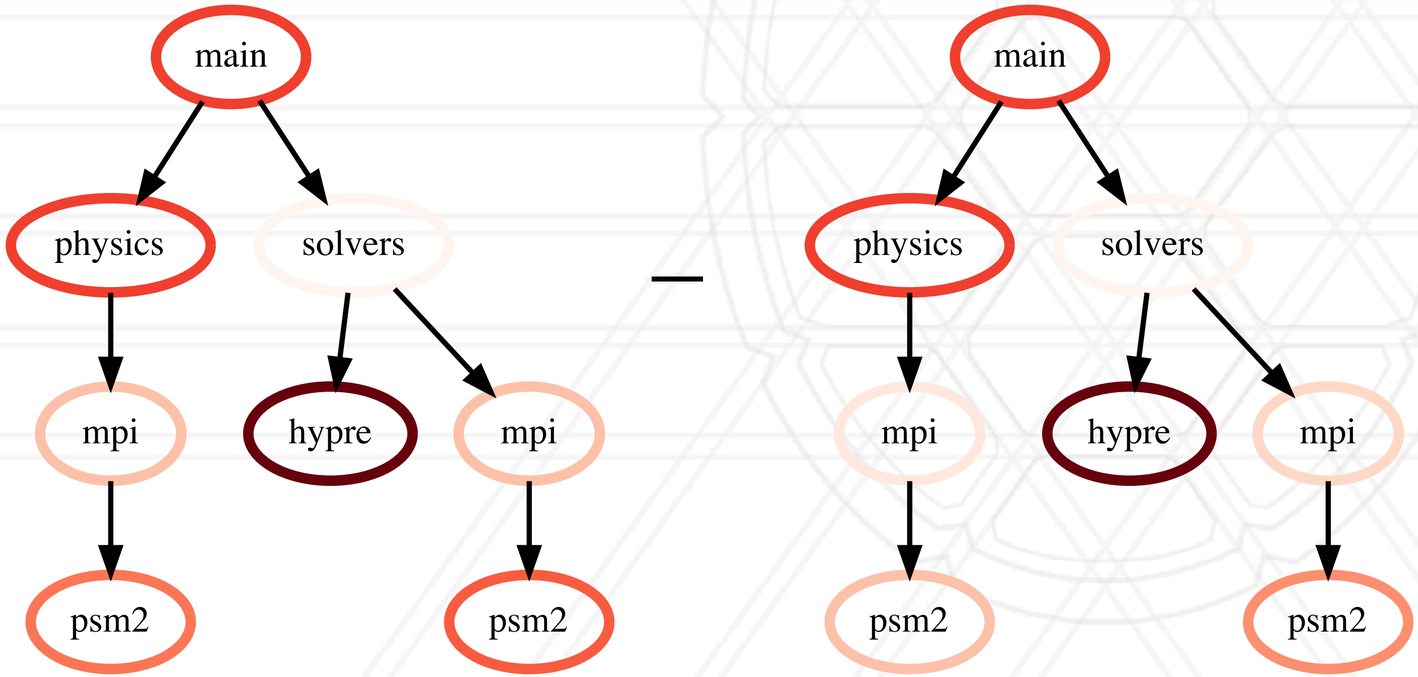


Graphframe operation: subtract

```
gf1 = ht.GraphFrame.from_literal( ... )
gf2 = ht.GraphFrame.from_literal( ... )
gf2 -= gf1
```

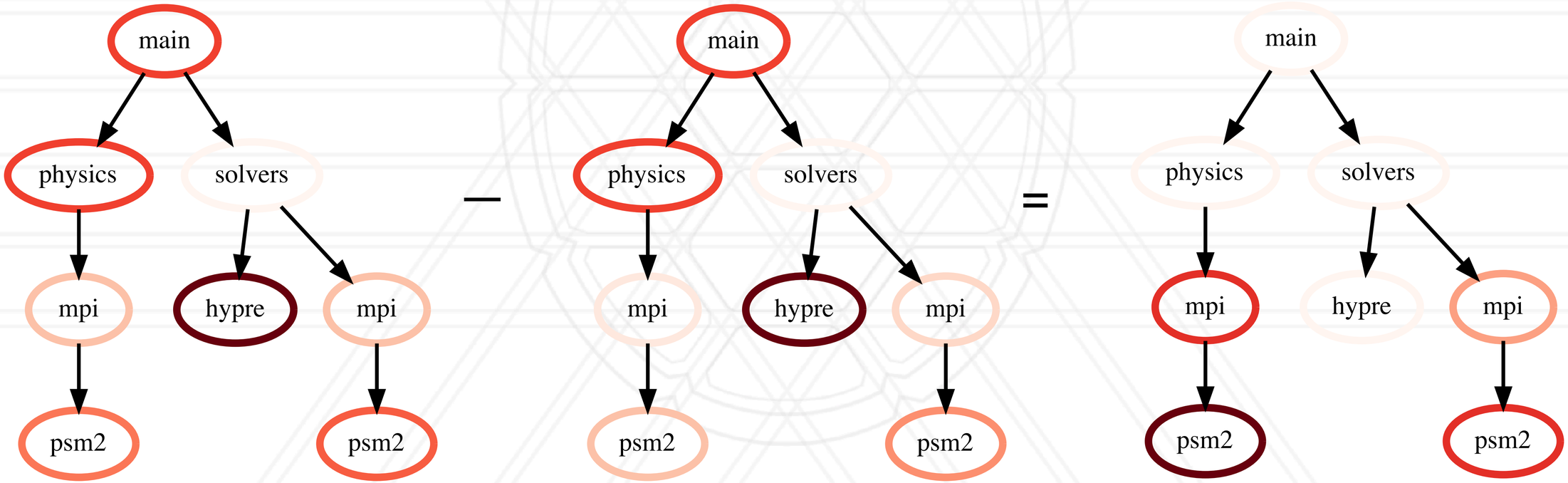
Graphframe operation: subtract

```
gf1 = ht.GraphFrame.from_literal( ... )
gf2 = ht.GraphFrame.from_literal( ... )
gf2 -= gf1
```



Graphframe operation: subtract

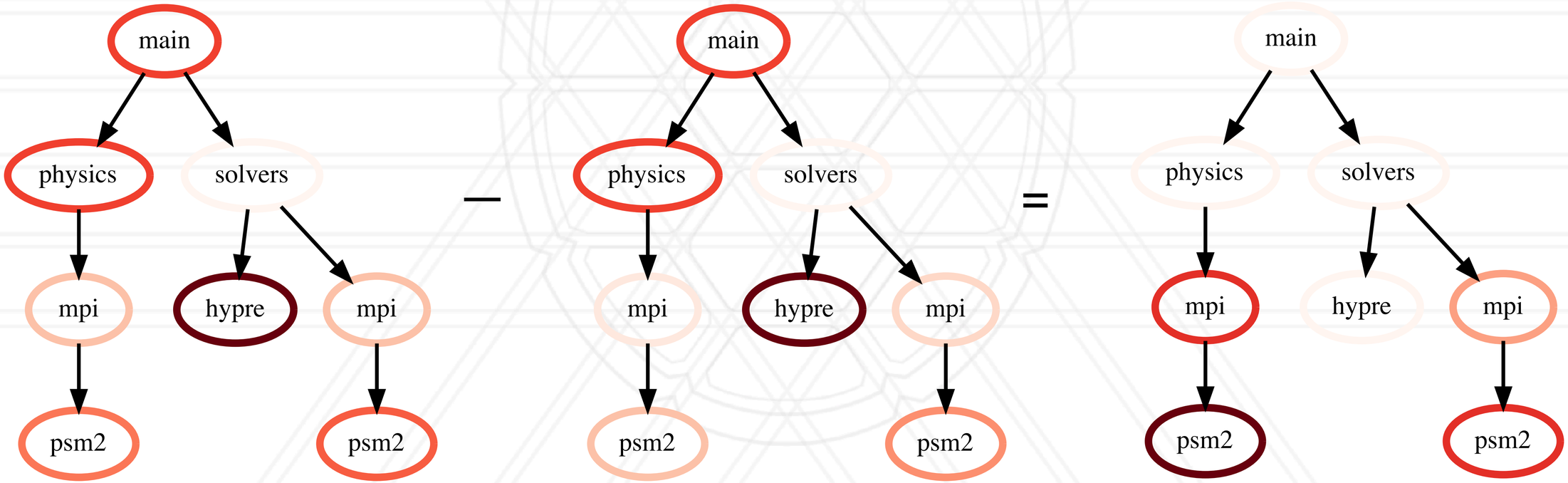
```
gf1 = ht.GraphFrame.from_literal( ... )
gf2 = ht.GraphFrame.from_literal( ... )
gf2 -= gf1
```



Graphframe operation: subtract

```
gf1 = ht.GraphFrame.from_literal( ... )
gf2 = ht.GraphFrame.from_literal( ... )
gf2 -= gf1
```

https://hatchet.readthedocs.io



Example 1: Generating a flat profile

```
gf = ht.GraphFrame.from_hpctoolkit('kripke')
gf.drop_index_levels()

grouped = gf.dataframe.groupby('name').sum()
sorted_df = grouped.sort_values(by=['time'], ascending=False)
print(sorted_df)
```

Example 1: Generating a flat profile

```
gf = ht.GraphFrame.from_hpctoolkit('kripke')
gf.drop_index_levels()

grouped = gf.dataframe.groupby('name').sum()
sorted_df = grouped.sort_values(by=['time'], ascending=False)
print(sorted_df)
```

Example 1: Generating a flat profile

```
gf = ht.GraphFrame.from_hpctoolkit('kripke')
gf.drop_index_levels()

grouped = gf.dataframe.groupby('name').sum()
sorted_df = grouped.sort_values(by=['time'], asce
print(sorted_df)
```

name	nid	time	time (inc)
<unknown file> [kripke]:0	17234	1.825282e+08	1.825282e+08
Kernel_3d_DGZ::scattering	60	7.669936e+07	7.896253e+07
Kernel_3d_DGZ::LTimes	30	5.010439e+07	5.240528e+07
Kernel_3d_DGZ::LPlusTimes	115	4.947707e+07	5.104498e+07
Kernel_3d_DGZ::sweep	981	5.018862e+06	5.018862e+06
memset.S:99	3773	3.168982e+06	3.168982e+06
memset.S:101	3970	2.120895e+06	2.120895e+06
Grid_Data::particleEdit	1201	1.131266e+06	1.249157e+06
<unknown file> [libpsm2.so.2.1]:0	324763	9.733415e+05	9.733415e+05
memset.S:98	3767	6.197776e+05	6.197776e+05

Example 2: Identifying load imbalance

```
gf1 = ht.GraphFrame.from_caliper('lulesh-512cores')
gf2 = gf1.copy()

gf1.drop_index_levels(function=np.mean)
gf2.drop_index_levels(function=np.max)

gf1.dataframe['imbalance'] = gf2.dataframe['time'].div(gf1.dataframe['time'])

sorted_df = gf1.dataframe.sort_values(by=['imbalance'], ascending=False)
print(sorted_df)
```

Example 2: Identifying load imbalance

```
gf1 = ht.GraphFrame.from_caliper('lulesh-512cores')
gf2 = gf1.copy()

gf1.drop_index_levels(function=np.mean)
gf2.drop_index_levels(function=np.max)

gf1.dataframe['imbalance'] = gf2.dataframe['time'].div(gf1.dataframe['time'])

sorted_df = gf1.dataframe.sort_values(by=['imbalance'], ascending=False)
print(sorted_df)
```

Example 2: Identifying load imbalance

```
gf1 = ht.GraphFrame.from_caliper('lulesh-512cores')
gf2 = gf1.copy()

gf1.drop_index_levels(function=np.mean)
gf2.drop_index_levels(function=np.max)

gf1.dataframe['imbalance'] = gf2.dataframe['time'].div(gf1.dataframe['time'])

sorted_df = gf1.dataframe.sort_values(by=['imbalance'], ascending=False)
print(sorted_df)
```

Example 2: Identifying load imbalance

```
gf1 = ht.GraphFrame.from_caliper('lulesh-512cores')
gf2 = gf1.copy()

gf1.drop_index_levels(function=np.mean)
gf2.drop_index_levels(function=np.max)

gf1.dataframe['imbalance'] = gf2.dataframe['time'].div(gf1.dataframe['time'])

sorted_df = gf1.dataframe.sort_values(by=['imbalance'], ascending=False)
print(sorted_df)
```

node	name	nid	time	time (inc)	imbalance
LagrangeNodal	LagrangeNodal	3.0	2.242594e+06	2.593621e+07	2.494720
main	main	0.0	1.106013e+05	5.357208e+07	2.161845
CalcForceForNodes	CalcForceForNodes	4.0	1.033639e+06	2.369361e+07	2.142526
CalcQForElems	CalcQForElems	16.0	3.351894e+06	6.649351e+06	2.037651
CalcEnergyForElems	CalcEnergyForElems	22.0	1.571996e+06	2.807323e+06	2.013174
CalcPressureForElems	CalcPressureForElems	23.0	1.235327e+06	1.235327e+06	2.005437

Example 3: Comparing two executions

```
gf1 = ht.GraphFrame.from_caliper('lulesh-1core.json')
gf2 = ht.GraphFrame.from_caliper('lulesh-27cores. json')

gf2.drop_index_levels()
gf3 = gf2 - gf1

sorted_df = gf3.dataframe.sort_values(by=['time'], ascending=False)
print(sorted_df)
```

Example 3: Comparing two executions

```
gf1 = ht.GraphFrame.from_caliper('lulesh-1core.json')
gf2 = ht.GraphFrame.from_caliper('lulesh-27cores. json')

gf2.drop_index_levels()
gf3 = gf2 - gf1

sorted_df = gf3.dataframe.sort_values(by=['time'], ascending=False)
print(sorted_df)
```

Example 3: Comparing two executions

```
gf1 = ht.GraphFrame.from_caliper('lulesh-1core.json')
gf2 = ht.GraphFrame.from_caliper('lulesh-27cores. json')

gf2.drop_index_levels()
gf3 = gf2 - gf1

sorted_df = gf3.dataframe.sort_values(by=['time'], ascending=False)
print(sorted_df)
```

node	name	nid	time	time (inc)
TimeIncrement	TimeIncrement	25.0	8.505048e+06	8.505048e+06
CalcQForElems	CalcQForElems	16.0	4.455672e+06	5.189453e+06
CalcHourglassControlForElems	CalcHourglassControlForElems	7.0	3.888798e+06	4.755817e+06
LagrangeNodal	LagrangeNodal	3.0	1.986046e+06	8.828475e+06
CalcForceForNodes	CalcForceForNodes	4.0	1.017857e+06	6.842429e+06

Example 4: Filtering by library

```
gf1 = GraphFrame.from_caliper('lulesh-27cores')
gf1.drop_index_levels()
filtered_gf1 = gf1.filter(lambda x: x['name'].startswith('MPI'))
squashed_gf1 = filtered_gf1.squash()

gf2 = GraphFrame.from_caliper('lulesh-512cores')
gf2.drop_index_levels()
filtered_gf2 = gf2.filter(lambda x: x['name'].startswith('MPI'))
squashed_gf2 = filtered_gf2.squash()

diff_gf = squashed_gf2 - squashed_gf1

sorted_df = diff_gf.dataframe.sort_values(by=['time'], ascending=False)
print(sorted_df)
```

Example 4: Filtering by library

```
gf1 = GraphFrame.from_caliper('lulesh-27cores')
gf1.drop_index_levels()
filtered_gf1 = gf1.filter(lambda x: x['name'].startswith('MPI'))
squashed_gf1 = filtered_gf1.squash()

gf2 = GraphFrame.from_caliper('lulesh-512cores')
gf2.drop_index_levels()
filtered_gf2 = gf2.filter(lambda x: x['name'].startswith('MPI'))
squashed_gf2 = filtered_gf2.squash()

diff_gf = squashed_gf2 - squashed_gf1

sorted_df = diff_gf.dataframe.sort_values(by=['time'], ascending=False)
print(sorted_df)
```

Example 4: Filtering by library

```
gf1 = GraphFrame.from_caliper('lulesh-27cores')
gf1.drop_index_levels()
filtered_gf1 = gf1.filter(lambda x: x['name'].startswith('MPI'))
squashed_gf1 = filtered_gf1.squash()

gf2 = GraphFrame.from_caliper('lulesh-512cores')
gf2.drop_index_levels()
filtered_gf2 = gf2.filter(lambda x: x['name
squashed_gf2 = filtered_gf2.squash()

diff_gf = squashed_gf2 - squashed_gf1

sorted_df = diff_gf.dataframe.sort_values(b
print(sorted_df)
```

node	node	time (inc)	name	nid	time
MPI_Allreduce	MPI_Allreduce	2.072371e+06	MPI_Allreduce	3	2.072371e+06
MPI_Finalize	MPI_Finalize	4.042198e+04	MPI_Finalize	0	4.042198e+04
MPI_Isend	MPI_Isend	1.753768e+04	MPI_Isend	15	1.753768e+04
MPI_Isend	MPI_Isend	7.718737e+03	MPI_Isend	13	7.718737e+03
MPI_Isend	MPI_Isend	7.542969e+03	MPI_Isend	7	7.542969e+03
MPI_Waitall	MPI_Waitall	4.573508e+03	MPI_Waitall	5	4.573508e+03
MPI_Barrier	MPI_Barrier	4.240952e+03	MPI_Barrier	12	4.240952e+03

Example 5: Scaling study

```
datasets = glob.glob('lulesh*.json')
datasets.sort()

dataframes = []
for dataset in datasets:
    gf = ht.GraphFrame.from_caliper(dataset)
    gf.drop_index_levels()

    num_pes = re.match('(.*)-(\d+)(.*)', dataset).group(2)
    gf.dataframe['pes'] = num_pes
    filtered_gf = gf.filter(lambda x: x['time'] > 1e6)
    dataframes.append(filtered_gf.dataframe)

result = pd.concat(dataframes)
pivot_df = result.pivot(index='pes', columns='name', values='time')
pivot_df.loc[:,:].plot.bar(stacked=True, figsize=(10,7))
```

Example 5: Scaling study

```
datasets = glob.glob('lulesh*.json')
datasets.sort()

dataframes = []
for dataset in datasets:
    gf = ht.GraphFrame.from_caliper(dataset)
    gf.drop_index_levels()

    num_pes = re.match('(.*)-(\d+)(.*)', dataset).group(2)
    gf.dataframe['pes'] = num_pes
    filtered_gf = gf.filter(lambda x: x['time'] > 1e6)
    dataframes.append(filtered_gf.dataframe)

result = pd.concat(dataframes)
pivot_df = result.pivot(index='pes', columns='name', values='time')
pivot_df.loc[:,:].plot.bar(stacked=True, figsize=(10,7))
```

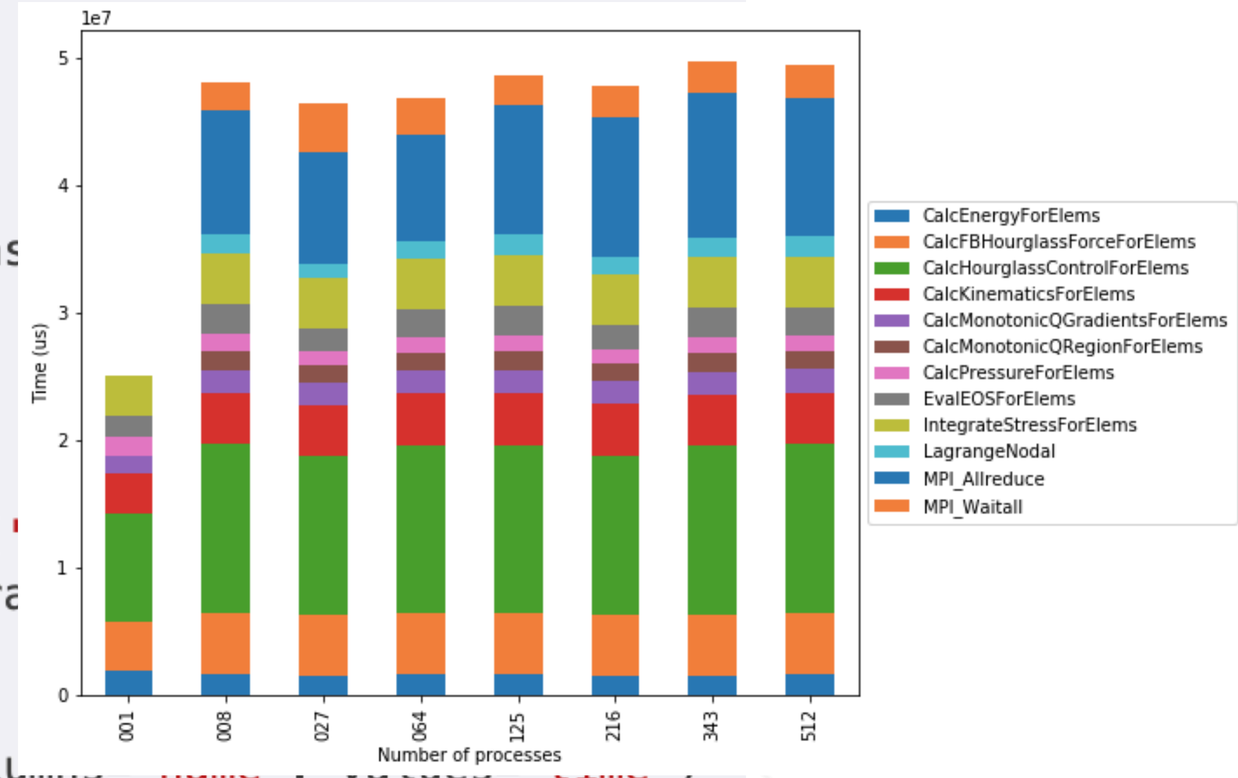
Example 5: Scaling study

```
datasets = glob.glob('lulesh*.json')
datasets.sort()

dataframes = []
for dataset in datasets:
    gf = ht.GraphFrame.from_caliper(datas
    gf.drop_index_levels()

    num_pes = re.match('(.*)-(\d+)(.*)',
    gf.dataframe['pes'] = num_pes
    filtered_gf = gf.filter(lambda x: x['
    dataframes.append(filtered_gf.datafra

result = pd.concat(dataframes)
pivot_df = result.pivot(index='pes', colu
pivot_df.loc[:,:].plot.bar(stacked=True, figsize=(10,7))
```



Visualizing *small* graphs

```
print(gf.tree(color=True))
```

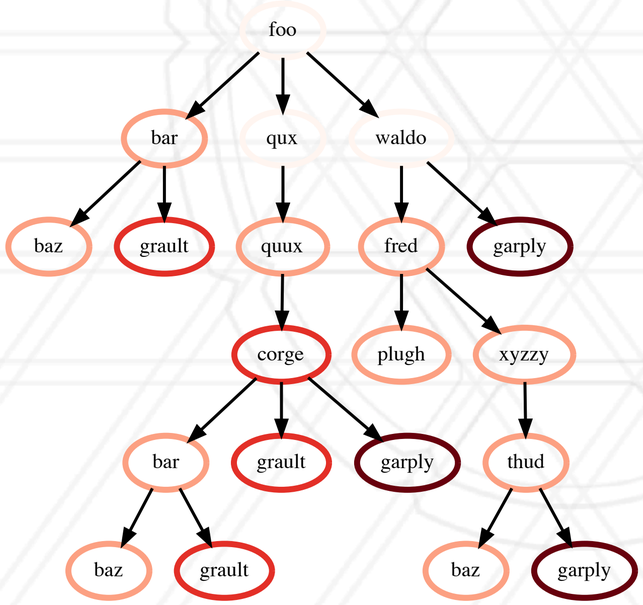
```
0.000 foo
├─ 5.000 bar
│  ├─ 5.000 baz
│  └─ 10.000 grault
├─ 0.000 qux
│  └─ 5.000 quux
│     └─ 10.000 corge
│        ├─ 5.000 bar
│        │  ├─ 5.000 baz
│        │  └─ 10.000 grault
│        ├─ 10.000 grault
│        └─ 15.000 garply
└─ 0.000 waldo
   ├─ 5.000 fred
   │  ├─ 5.000 plugh
   │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

Visualizing *small* graphs

```
print(gf.tree(color=True))
```

```
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())
```



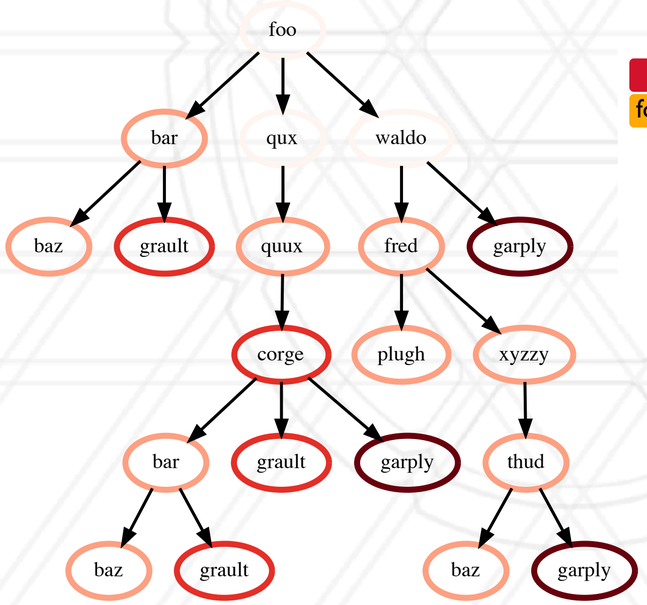
Graphviz DOT

Visualizing *small* graphs

```
print(gf.tree(color=True))
```

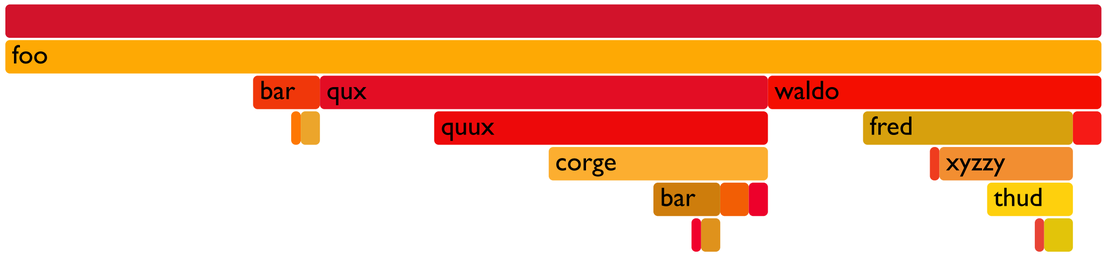
```
0.000 foo
├─ 5.000 bar
│  ├─ 5.000 baz
│  └─ 10.000 grault
├─ 0.000 qux
│  └─ 5.000 quux
│     └─ 10.000 corge
│        ├─ 5.000 bar
│        │  ├─ 5.000 baz
│        │  └─ 10.000 grault
│        ├─ 10.000 grault
│        └─ 15.000 garply
└─ 0.000 waldo
   ├─ 5.000 fred
   │  ├─ 5.000 plugh
   │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



Graphviz DOT

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



Flamegraph

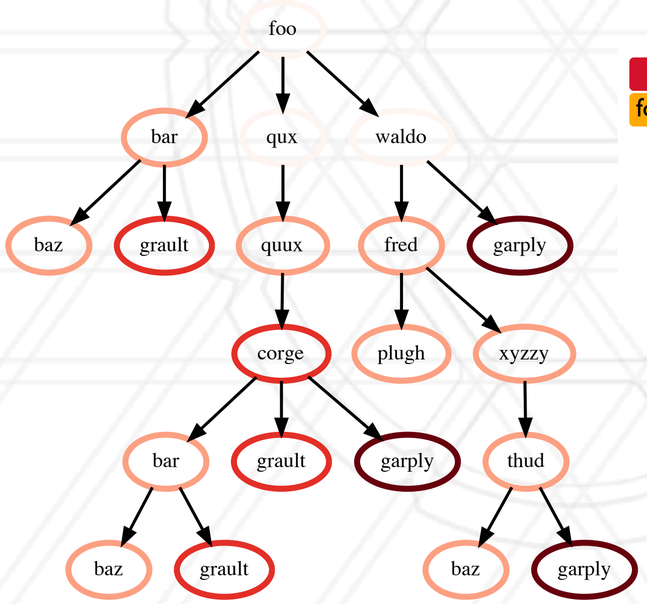
Visualizing *small* graphs

```
print(gf.tree(color=True))
```

https://hatchet.readthedocs.io

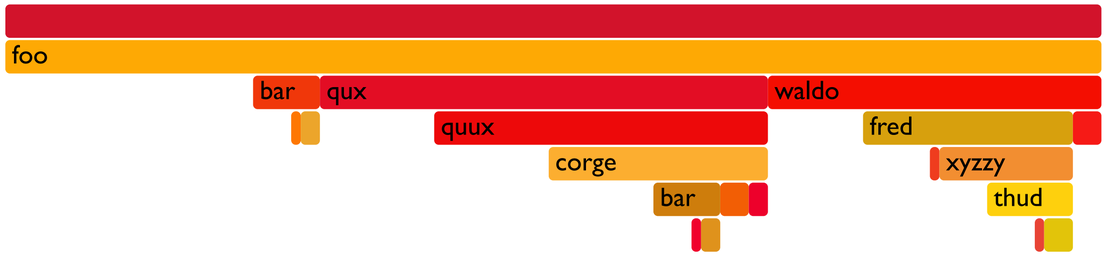
```
0.000 foo
├─ 5.000 bar
│  ├─ 5.000 baz
│  └─ 10.000 grault
├─ 0.000 qux
│  └─ 5.000 quux
│     └─ 10.000 corge
│        ├─ 5.000 bar
│        │  ├─ 5.000 baz
│        │  └─ 10.000 grault
│        ├─ 10.000 grault
│        └─ 15.000 garply
└─ 0.000 waldo
   ├─ 5.000 fred
   │  ├─ 5.000 plugh
   │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



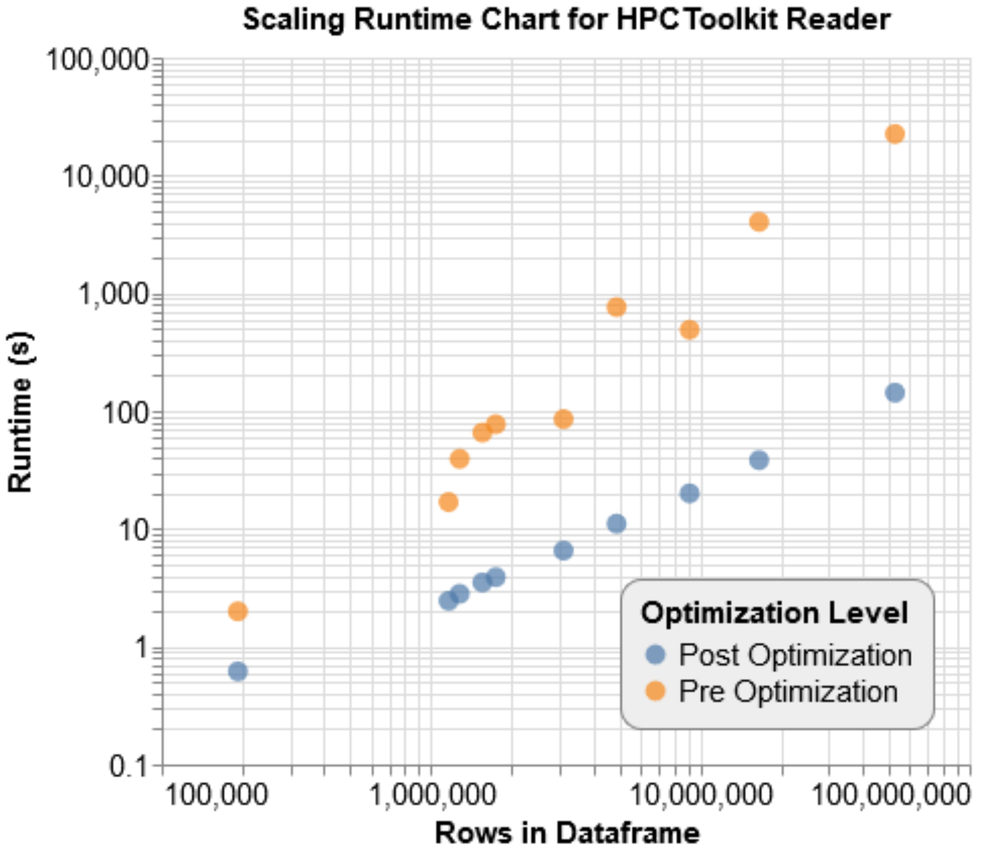
Graphviz DOT

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

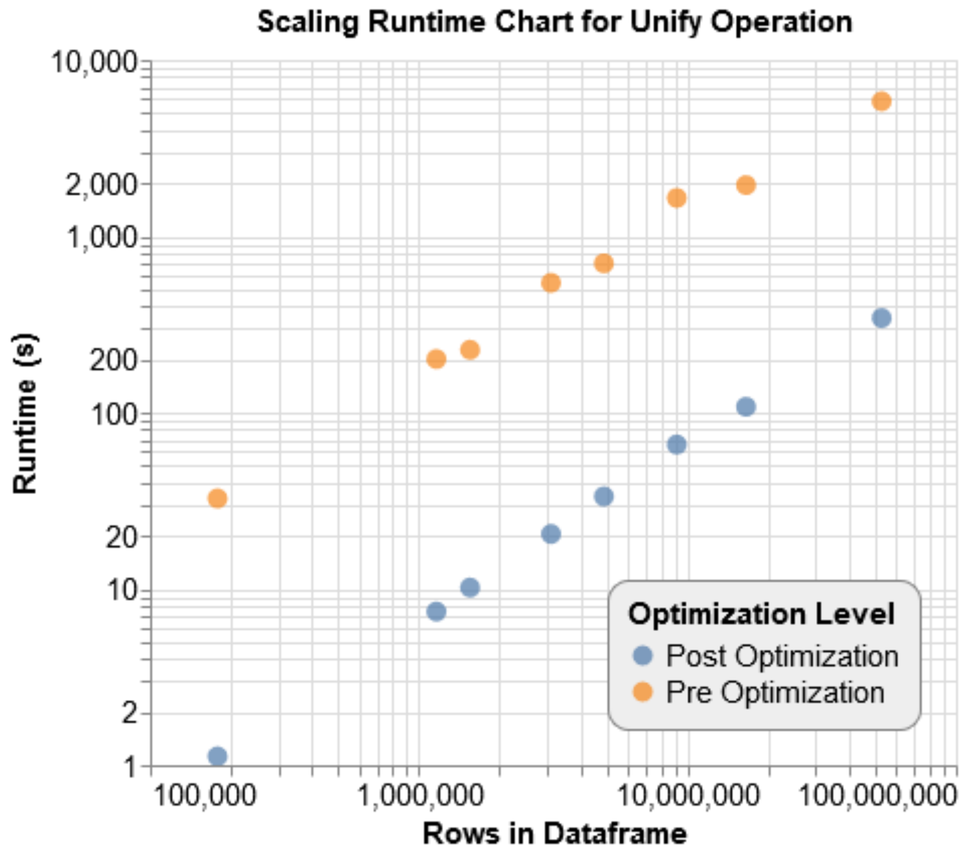


Flamegraph

Performance improvements to Hatchet



HPCToolkit `read()`



`unify()` operation

Conclusion and future work

- Leverage the power of pandas for performance analysis in HPC
- Enable using graph nodes to index a dataframe
- Programmatic analysis of hierarchical data from one or multiple executions
- Support for other profile formats, add an output format
- Implement a higher-level API for automating performance analysis

Conclusion and future work

- Leverage the power of pandas for performance analysis in HPC
- Enable using graph nodes to index a dataframe
- Programmatic analysis of hierarchical data from one or multiple executions
- Support for other profile formats, add an output format
- Implement a higher-level API for automating performance analysis

https://github.com/hatchet/hatchet
https://hatchet.readthedocs.io

Conclusion and future work

- Leverage the power of pandas for performance analysis in HPC
- Enable using graph nodes to index a dataframe
- Programmatic analysis of hierarchical data from one or multiple executions
- Support for other profile formats, add an output format
- Implement a higher-level API for automating performance analysis

Hatchet contributors: https://github.com/hatchet/hatchet/graphs/contributors

https://github.com/hatchet/hatchet
https://hatchet.readthedocs.io

UNIVERSITY OF
MARYLAND

Abhinav Bhatele

Parallel Software and Systems Group

5218 Brendan Iribe Center (IRB) / College Park, MD 20742

phone: 301.405.4507 / e-mail: bhatele@cs.umd.edu