Extra-P Meets Hatchet: Towards Modeling in Performance Analytics

SIAM Conf on Parallel Processing for Scientific Computing ‘22

CASC
Center for Applied
Scientific Computing

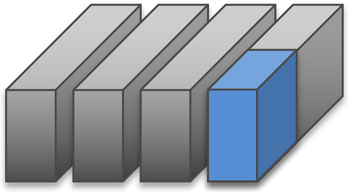
Feb 25, 2022

Sergei Shudler (presented by David Boehme)
Postdoctoral Researcher

LLNL-PRES-831250
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

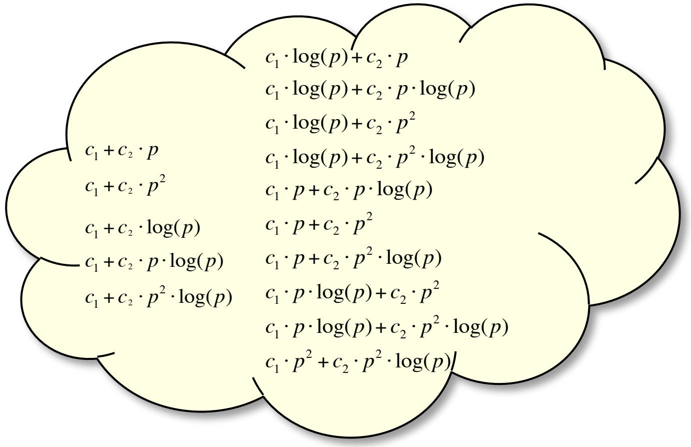
Automatic Performance Modeling



Small-scale measurements

$$f(x_1,\dots,x_m) = \sum_{k=1}^{n} c_k \prod_{l=1}^{m} x_l^{i_{kl}} \cdot \log_2^{j_{kl}}(x_l)$$

Performance model normal form (PMNF)



$c_1 + c_2 \cdot p$

$c_1 + c_2 \cdot p^2$

$c_1 + c_2 \cdot \log(p)$

$c_1 + c_2 \cdot p \cdot \log(p)$

$c_1 + c_2 \cdot p^2 \cdot \log(p)$

$c_1 \cdot \log(p) + c_2 \cdot p$

$c_1 \cdot \log(p) + c_2 \cdot p \cdot \log(p)$

$c_1 \cdot \log(p) + c_2 \cdot p^2$

$c_1 \cdot \log(p) + c_2 \cdot p^2 \cdot \log(p)$

$c_1 \cdot p + c_2 \cdot p \cdot \log(p)$

$c_1 \cdot p + c_2 \cdot p^2$

$c_1 \cdot p + c_2 \cdot p^2 \cdot \log(p)$

$c_1 \cdot p \cdot \log(p) + c_2 \cdot p^2$

$c_1 \cdot p \cdot \log(p) + c_2 \cdot p^2 \cdot \log(p)$

$c_1 \cdot p^2 + c_2 \cdot p^2 \cdot \log(p)$

Generation of candidate models
and selection of best fit

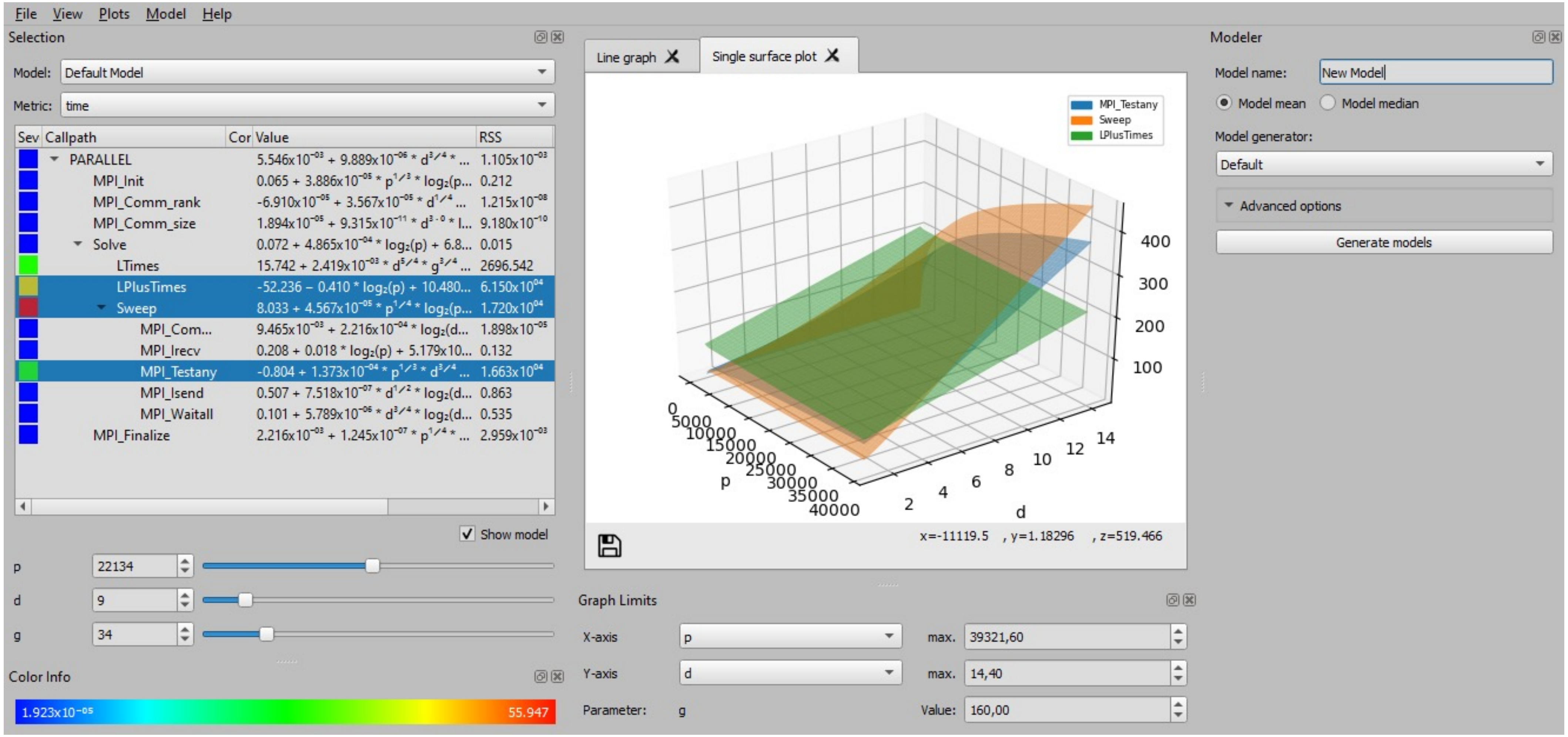
Kernel	Model
sweep → MPI_Recv	$4.03\sqrt{p}$
sweep	582.19
Kripke	$5.4 \cdot d \cdot g$

Calotoiu et al.: Using Automated Performance Modeling to Find Scalability Bugs in Complex Codes (*SC'13*)

Extra-P

- Generates performance models
- Input is performance profiles (e.g., Score-P) or text data
- Both weak and strong scaling models are supported
- Offers both command line and GUI interfaces
- Github repo: https://github.com/Extra-p/extrap

Extra-P GUI



Model: Default Model

Metric: time

Sev	Callpath	Cor	Value	RSS
🟦	PARALLEL		$5.546x10^{-03} + 9.889x10^{-06} * d^{3/4} *$ ...	$1.105x10^{-03}$
🟦	MPI_Init		$0.065 + 3.886x10^{-05} * p^{1/3} * \log_2(p$...	0.212
🟦	MPI_Comm_rank		$-6.910x10^{-05} + 3.567x10^{-05} * d^{1/4}$ ...	$1.215x10^{-08}$
🟦	MPI_Comm_size		$1.894x10^{-05} + 9.315x10^{-11} * d^{3 \cdot 0} *$ l...	$9.180x10^{-10}$
🟦	Solve		$0.072 + 4.865x10^{-04} * \log_2(p) + 6.8$...	0.015
🟩	LTimes		$15.742 + 2.419x10^{-03} * d^{5/4} * g^{3/4}$ ...	2696.542
🟨	LPlusTimes		$-52.236 - 0.410 * \log_2(p) + 10.480$...	$6.150x10^{04}$
🟥	Sweep		$8.033 + 4.567x10^{-05} * p^{1/4} * \log_2(p$...	$1.720x10^{04}$
🟦	MPI_Com...		$9.465x10^{-03} + 2.216x10^{-04} * \log_2(d$...	$1.898x10^{-05}$
🟦	MPI_Irecv		$0.208 + 0.018 * \log_2(p) + 5.179x10$...	0.132
🟩	MPI_Testany		$-0.804 + 1.373x10^{-04} * p^{1/3} * d^{3/4}$ ...	$1.663x10^{04}$
🟦	MPI_Isend		$0.507 + 7.518x10^{-07} * d^{1/2} * \log_2(d$...	0.863
🟦	MPI_Waitall		$0.101 + 5.789x10^{-06} * d^{3/4} * \log_2(d$...	0.535
🟦	MPI_Finalize		$2.216x10^{-03} + 1.245x10^{-07} * p^{1/4} *$ ...	$2.959x10^{-03}$

☑ Show model

p: 22134

d: 9

g: 34

Model name: New Model

● Model mean ○ Model median

Model generator: Default

X-axis: p max. 39321,60

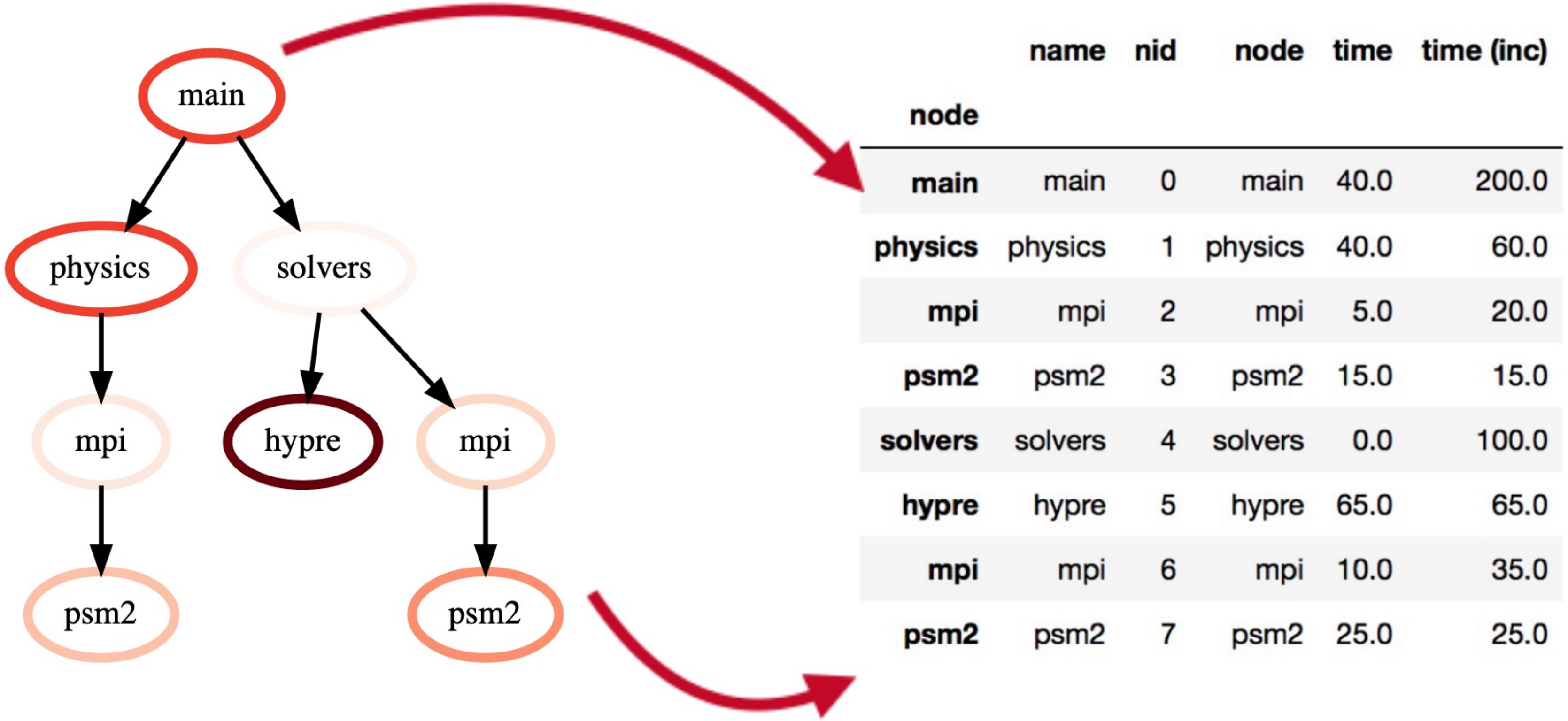
Y-axis: d max. 14,40

Parameter: g Value: 160,00

Hatchet

- A (programmatic) performance analysis tool
- Python-based data science applied to performance analysis
- Supports structured and hierarchical data
- Allows to compare multiple execution profiles, automate analysis in Python scripts
- Github repo: https://github.com/LLNL/Hatchet
- Publications:
  - Bhatele, Brink, Gamblin: “Hatchet: Pruning the Overgrowth in Parallel Profiles” (SC’19)
  - Brink et al.: “Usability and Performance Improvements in Hatchet” (ProTools’20)

Hatchet’s GraphFrame



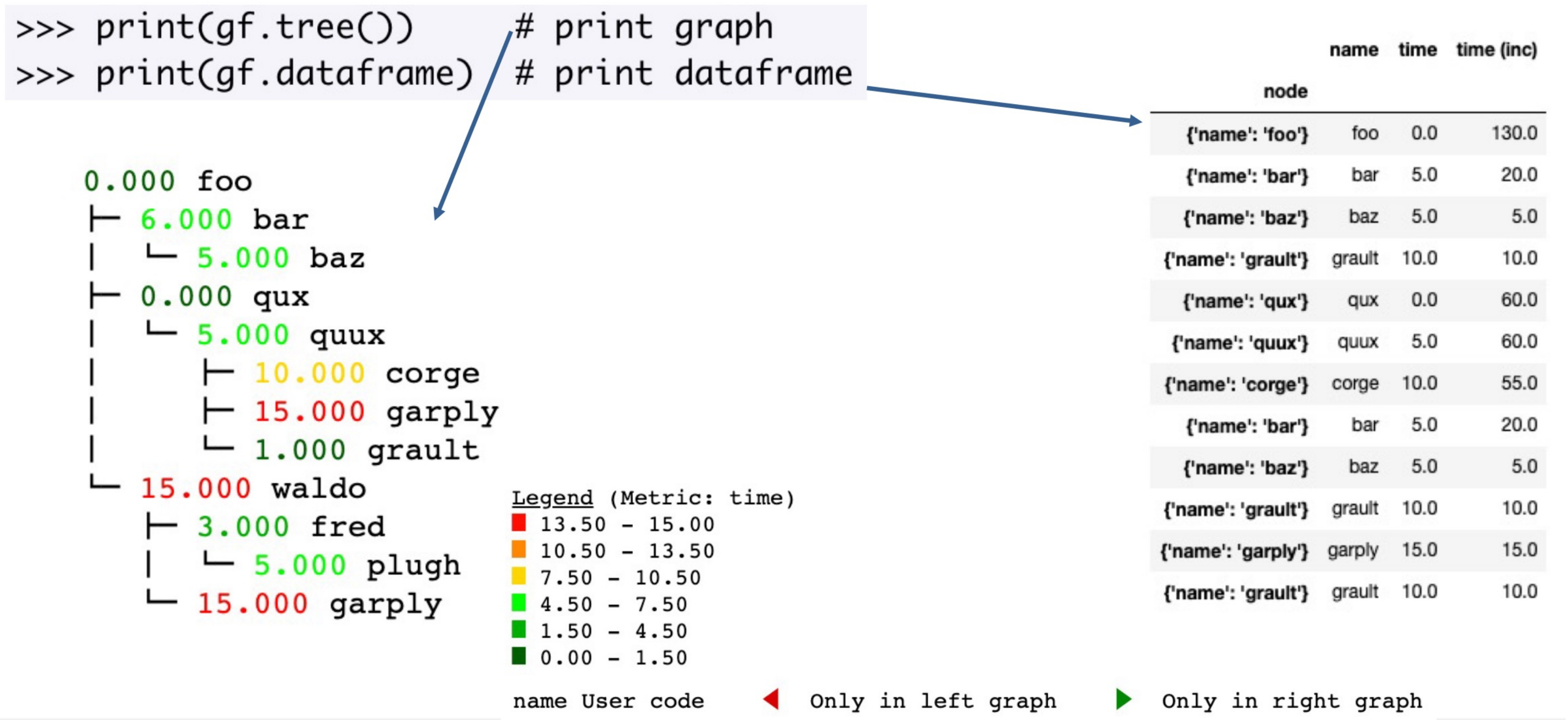
	name	nid	node	time	time (inc)
node					
main	main	0	main	40.0	200.0
physics	physics	1	physics	40.0	60.0
mpi	mpi	2	mpi	5.0	20.0
psm2	psm2	3	psm2	15.0	15.0
solvers	solvers	4	solvers	0.0	100.0
hypre	hypre	5	hypre	65.0	65.0
mpi	mpi	6	mpi	10.0	35.0
psm2	psm2	7	psm2	25.0	25.0

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)

Hatchet tutorial:
https://hatchet.readthedocs.io/en/latest/publications.html

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
```

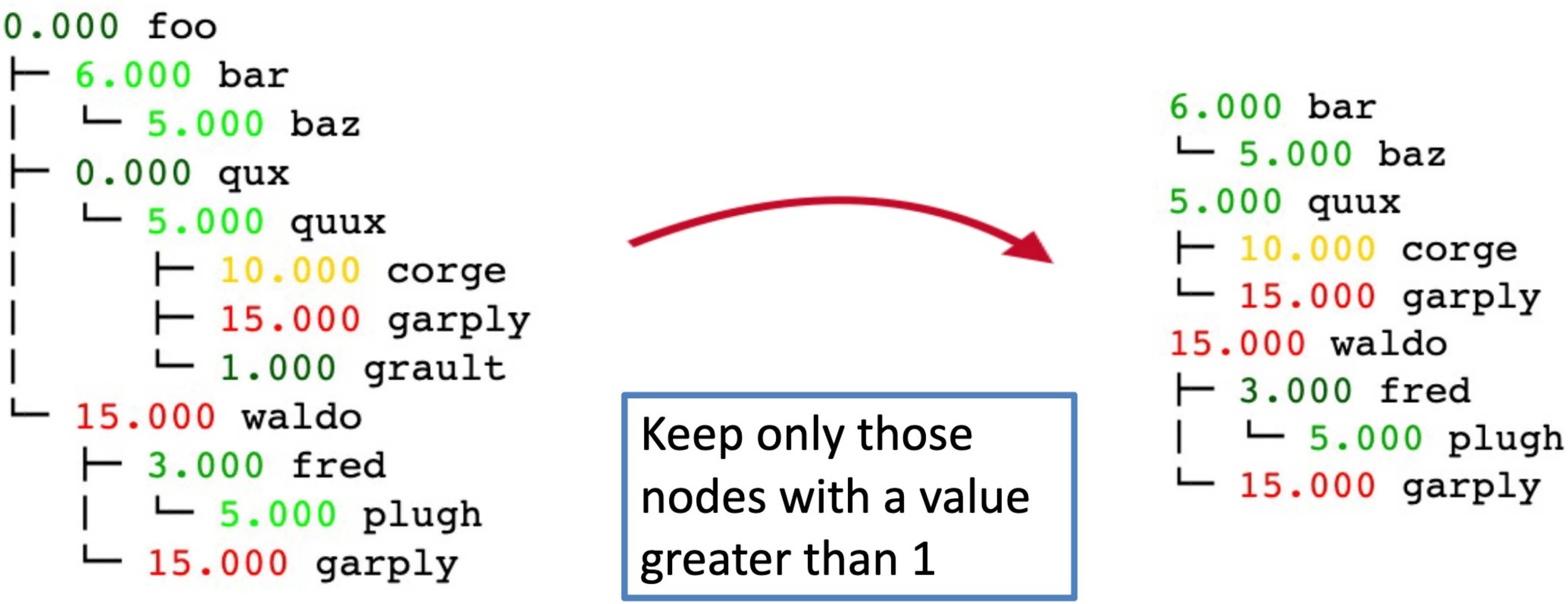
```
Legend (Metric: time)
🟥 13.50 - 15.00
🟧 10.50 - 13.50
🟨 7.50 - 10.50
🟩 4.50 - 7.50
🟩 1.50 - 4.50
🟩 0.00 - 1.50

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

node	name	time	time (inc)
{'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

Hatchet tutorial:
https://hatchet.readthedocs.io/en/latest/publications.html

Programmatic analysis

```
>>> 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
```

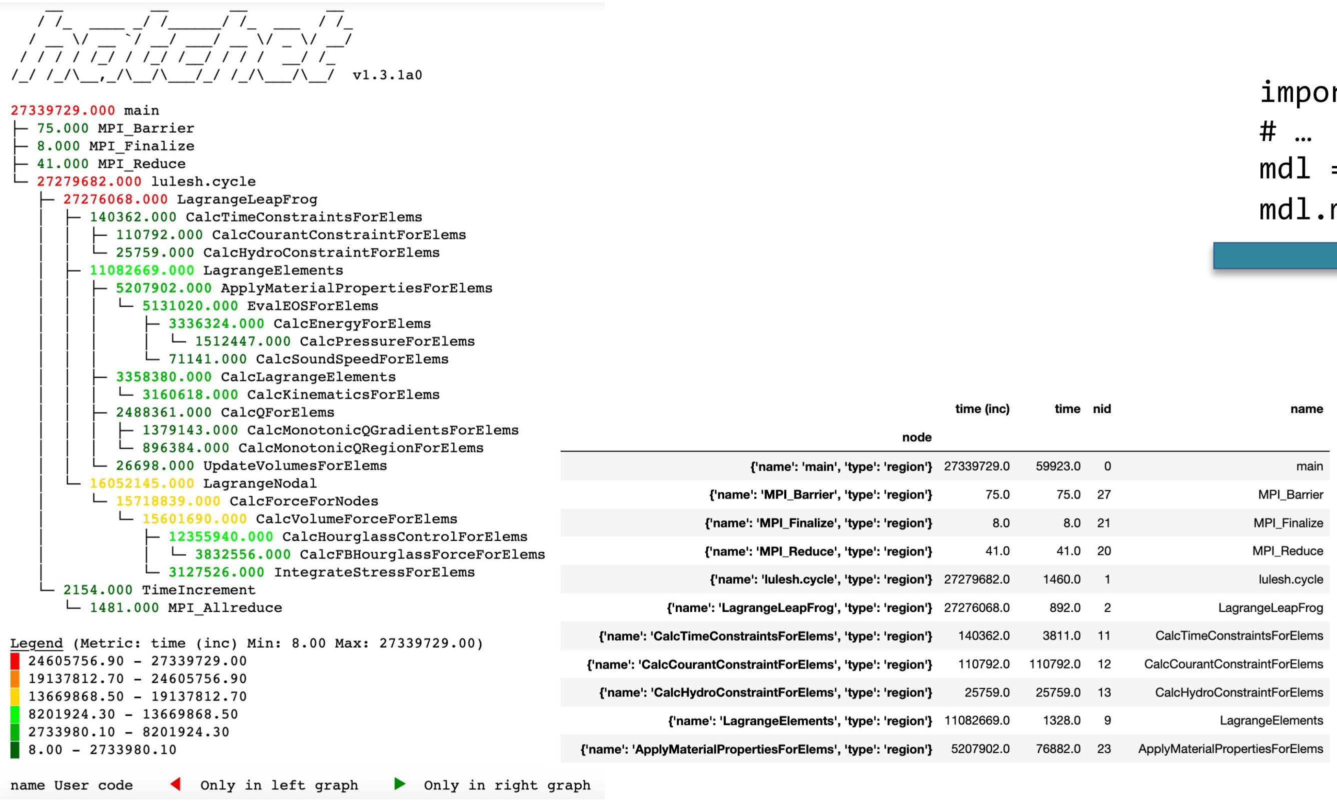
```
6.000 bar
└─ 5.000 baz
5.000 quux
├─ 10.000 corge
└─ 15.000 garply
15.000 waldo
├─ 3.000 fred
|   └─ 5.000 plugh
└─ 15.000 garply
```

Keep only those nodes with a value greater than 1

Hatchet tutorial:
https://hatchet.readthedocs.io/en/latest/publications.html

Integrating Extra-P in Hatchet

- Code in Github: https://github.com/sshudler/hatchet/tree/modeling

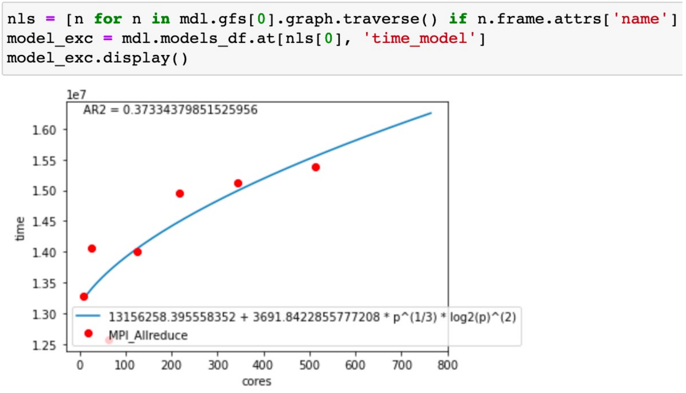


```
    __          __       __         __
   / /_  ____ _/ /______/ /_  ___  / /_
  / __ \/ __ `/ __/ ___/ __ \/ _ \/ __/
 / / / / /_/ / /_/ /__/ / / /  __/ /_
/_/ /_/\__,_/\__/\___/_/ /_/\___/\__/  v1.3.1a0

27339729.000 main
├─ 75.000 MPI_Barrier
├─ 8.000 MPI_Finalize
├─ 41.000 MPI_Reduce
└─ 27279682.000 lulesh.cycle
   ├─ 27276068.000 LagrangeLeapFrog
   │  ├─ 140362.000 CalcTimeConstraintsForElems
   │  │  ├─ 110792.000 CalcCourantConstraintForElems
   │  │  └─ 25759.000 CalcHydroConstraintForElems
   │  ├─ 11082669.000 LagrangeElements
   │  │  ├─ 5207902.000 ApplyMaterialPropertiesForElems
   │  │  │  └─ 5131020.000 EvalEOSForElems
   │  │  │     ├─ 3336324.000 CalcEnergyForElems
   │  │  │     │  └─ 1512447.000 CalcPressureForElems
   │  │  │     └─ 71141.000 CalcSoundSpeedForElems
   │  │  ├─ 3358380.000 CalcLagrangeElements
   │  │  │  └─ 3160618.000 CalcKinematicsForElems
   │  │  ├─ 2488361.000 CalcQForElems
   │  │  │  ├─ 1379143.000 CalcMonotonicQGradientsForElems
   │  │  │  └─ 896384.000 CalcMonotonicQRegionForElems
   │  │  └─ 26698.000 UpdateVolumesForElems
   │  └─ 16052145.000 LagrangeNodal
   │     └─ 15718839.000 CalcForceForNodes
   │        └─ 15601690.000 CalcVolumeForceForElems
   │           ├─ 12355940.000 CalcHourglassControlForElems
   │           │  └─ 3832556.000 CalcFBHourglassForceForElems
   │           └─ 3127526.000 IntegrateStressForElems
   └─ 2154.000 TimeIncrement
      └─ 1481.000 MPI_Allreduce

Legend (Metric: time (inc) Min: 8.00 Max: 27339729.00)
█ 24605756.90 - 27339729.00
█ 19137812.70 - 24605756.90
█ 13669868.50 - 19137812.70
█ 8201924.30 - 13669868.50
█ 2733980.10 - 8201924.30
█ 8.00 - 2733980.10

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

	time (inc)	time	nid	name
node				
{'name': 'main', 'type': 'region'}	27339729.0	59923.0	0	main
{'name': 'MPI_Barrier', 'type': 'region'}	75.0	75.0	27	MPI_Barrier
{'name': 'MPI_Finalize', 'type': 'region'}	8.0	8.0	21	MPI_Finalize
{'name': 'MPI_Reduce', 'type': 'region'}	41.0	41.0	20	MPI_Reduce
{'name': 'lulesh.cycle', 'type': 'region'}	27279682.0	1460.0	1	lulesh.cycle
{'name': 'LagrangeLeapFrog', 'type': 'region'}	27276068.0	892.0	2	LagrangeLeapFrog
{'name': 'CalcTimeConstraintsForElems', 'type': 'region'}	140362.0	3811.0	11	CalcTimeConstraintsForElems
{'name': 'CalcCourantConstraintForElems', 'type': 'region'}	110792.0	110792.0	12	CalcCourantConstraintForElems
{'name': 'CalcHydroConstraintForElems', 'type': 'region'}	25759.0	25759.0	13	CalcHydroConstraintForElems
{'name': 'LagrangeElements', 'type': 'region'}	11082669.0	1328.0	9	LagrangeElements
{'name': 'ApplyMaterialPropertiesForElems', 'type': 'region'}	5207902.0	76882.0	23	ApplyMaterialPropertiesForElems

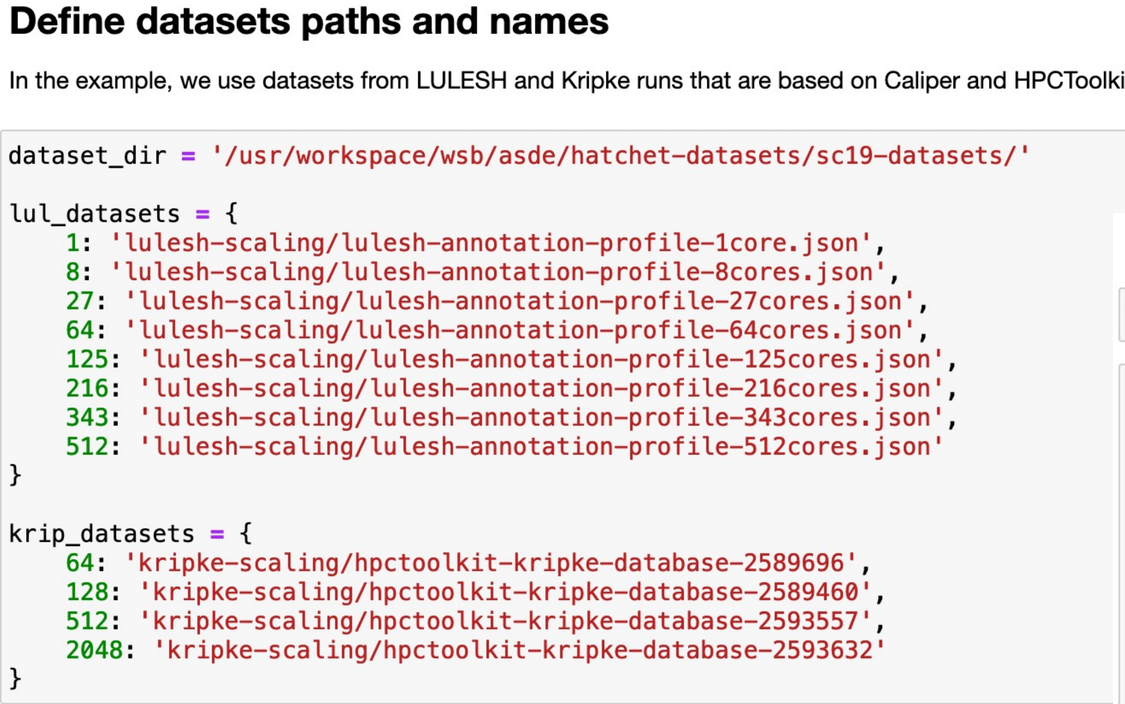
Hatchet GraphFrame (graph + dataframe)

```
import hatchet as ht
# …
mdl = ht.Modeling(…)
mdl.model_all()
```



```
nls = [n for n in mdl.gfs[0].graph.traverse() if n.frame.attrs['name']]
model_exc = mdl.models_df.at[nls[0], 'time_model']
model_exc.display()
```

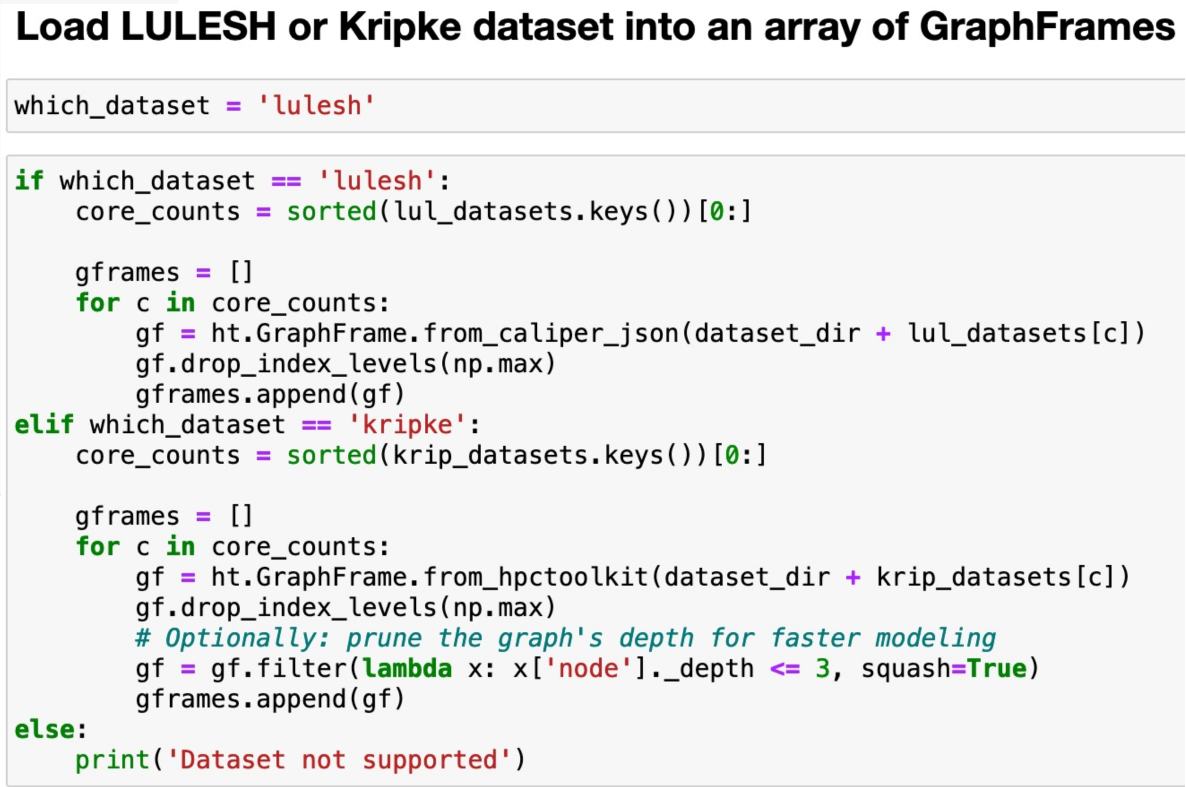
Input: Sequence of GraphFrames



Define datasets paths and names

In the example, we use datasets from LULESH and Kripke runs that are based on Caliper and HPCToolkit, respectively.

```
dataset_dir = '/usr/workspace/wsb/asde/hatchet-datasets/sc19-datasets/'

lul_datasets = {
    1: 'lulesh-scaling/lulesh-annotation-profile-1core.json',
    8: 'lulesh-scaling/lulesh-annotation-profile-8cores.json',
    27: 'lulesh-scaling/lulesh-annotation-profile-27cores.json',
    64: 'lulesh-scaling/lulesh-annotation-profile-64cores.json',
    125: 'lulesh-scaling/lulesh-annotation-profile-125cores.json',
    216: 'lulesh-scaling/lulesh-annotation-profile-216cores.json',
    343: 'lulesh-scaling/lulesh-annotation-profile-343cores.json',
    512: 'lulesh-scaling/lulesh-annotation-profile-512cores.json'
}

krip_datasets = {
    64: 'kripke-scaling/hpctoolkit-kripke-database-2589696',
    128: 'kripke-scaling/hpctoolkit-kripke-database-2589460',
    512: 'kripke-scaling/hpctoolkit-kripke-database-2593557',
    2048: 'kripke-scaling/hpctoolkit-kripke-database-2593632'
}
```

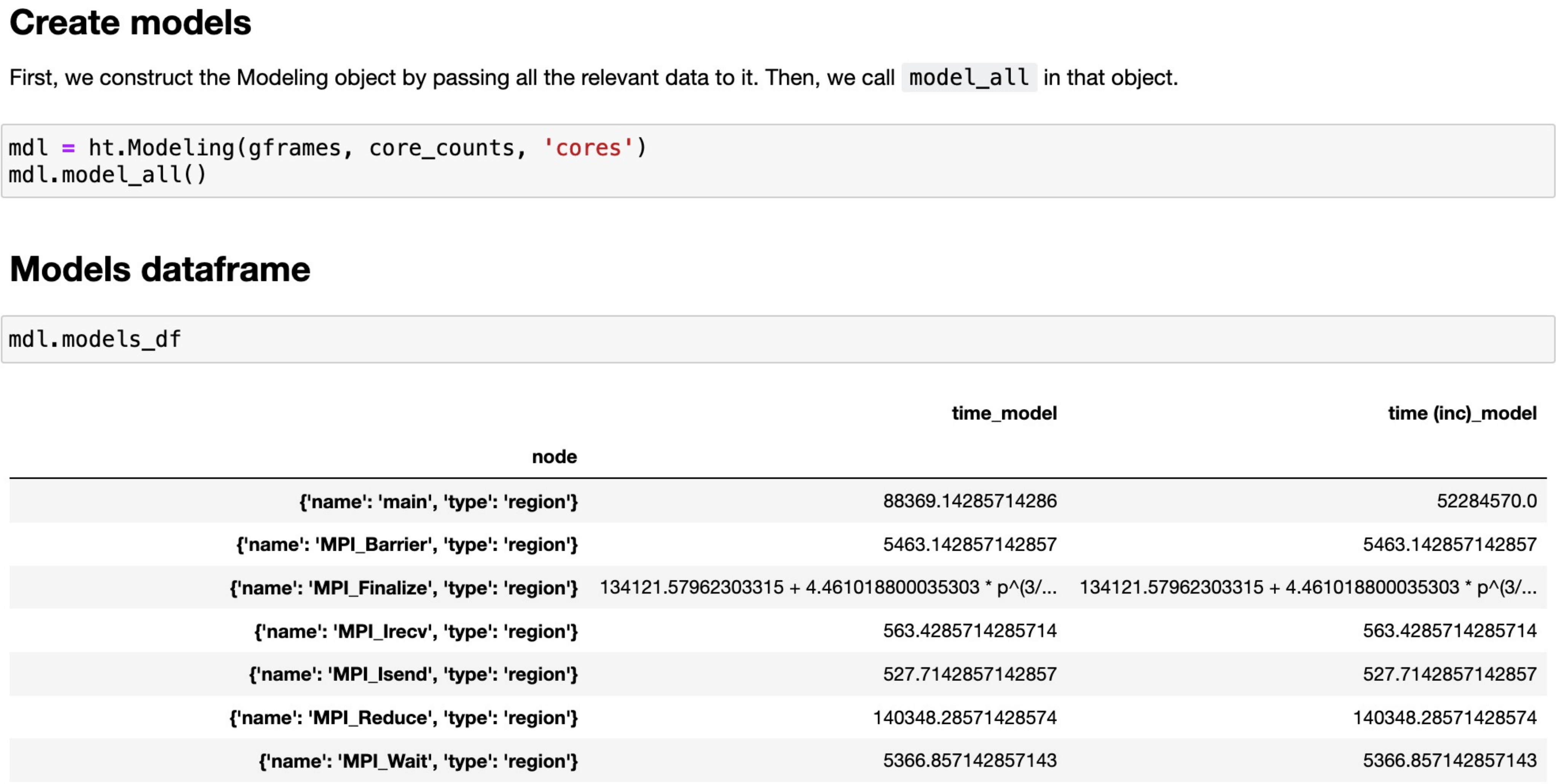


Load LULESH or Kripke dataset into an array of GraphFrames

```
which_dataset = 'lulesh'
```

```
if which_dataset == 'lulesh':
    core_counts = sorted(lul_datasets.keys())[0:]

    gframes = []
    for c in core_counts:
        gf = ht.GraphFrame.from_caliper_json(dataset_dir + lul_datasets[c])
        gf.drop_index_levels(np.max)
        gframes.append(gf)
elif which_dataset == 'kripke':
    core_counts = sorted(krip_datasets.keys())[0:]

    gframes = []
    for c in core_counts:
        gf = ht.GraphFrame.from_hpctoolkit(dataset_dir + krip_datasets[c])
        gf.drop_index_levels(np.max)
        # Optionally: prune the graph's depth for faster modeling
        gf = gf.filter(lambda x: x['node']._depth <= 3, squash=True)
        gframes.append(gf)
else:
    print('Dataset not supported')
```

Producing models



Create models

First, we construct the Modeling object by passing all the relevant data to it. Then, we call `model_all` in that object.

```
mdl = ht.Modeling(gframes, core_counts, 'cores')
mdl.model_all()
```

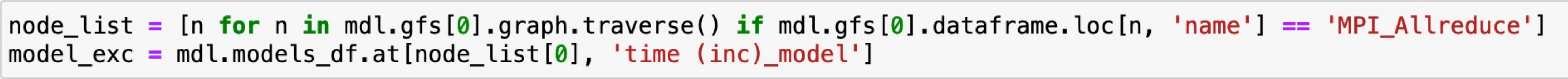
Models dataframe

```
mdl.models_df
```

node	time_model	time (inc)_model
{'name': 'main', 'type': 'region'}	88369.14285714286	52284570.0
{'name': 'MPI_Barrier', 'type': 'region'}	5463.142857142857	5463.142857142857
{'name': 'MPI_Finalize', 'type': 'region'}	134121.57962303315 + 4.461018800035303 * p^(3/...	134121.57962303315 + 4.461018800035303 * p^(3/...
{'name': 'MPI_Irecv', 'type': 'region'}	563.4285714285714	563.4285714285714
{'name': 'MPI_Isend', 'type': 'region'}	527.7142857142857	527.7142857142857
{'name': 'MPI_Reduce', 'type': 'region'}	140348.28571428574	140348.28571428574
{'name': 'MPI_Wait', 'type': 'region'}	5366.857142857143	5366.857142857143

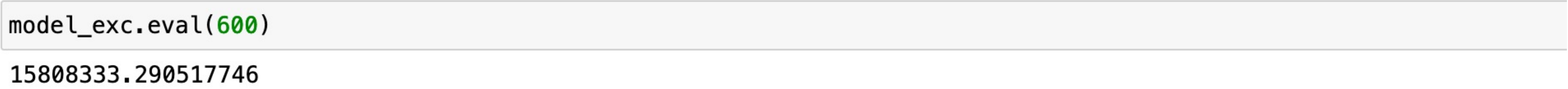
Operations on specific model

- Query the model:



```
node_list = [n for n in mdl.gfs[0].graph.traverse() if mdl.gfs[0].dataframe.loc[n, 'name'] == 'MPI_Allreduce']
model_exc = mdl.models_df.at[node_list[0], 'time (inc)_model']
```

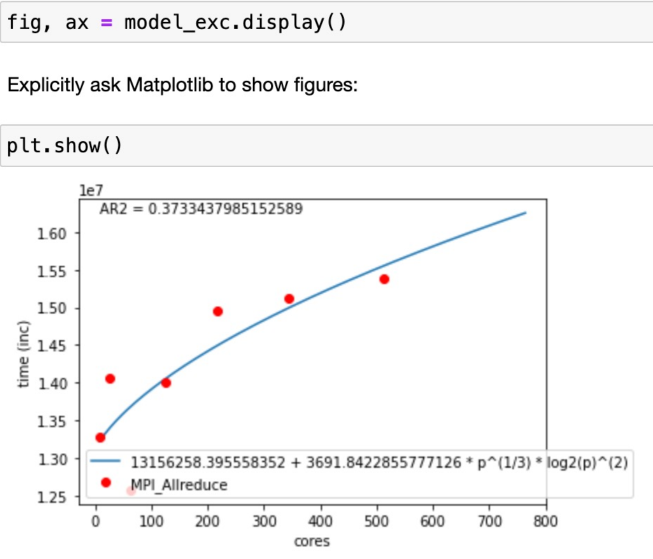
- Evaluate values:



```
model_exc.eval(600)
```

```
15808333.290517746
```

- Display the model:

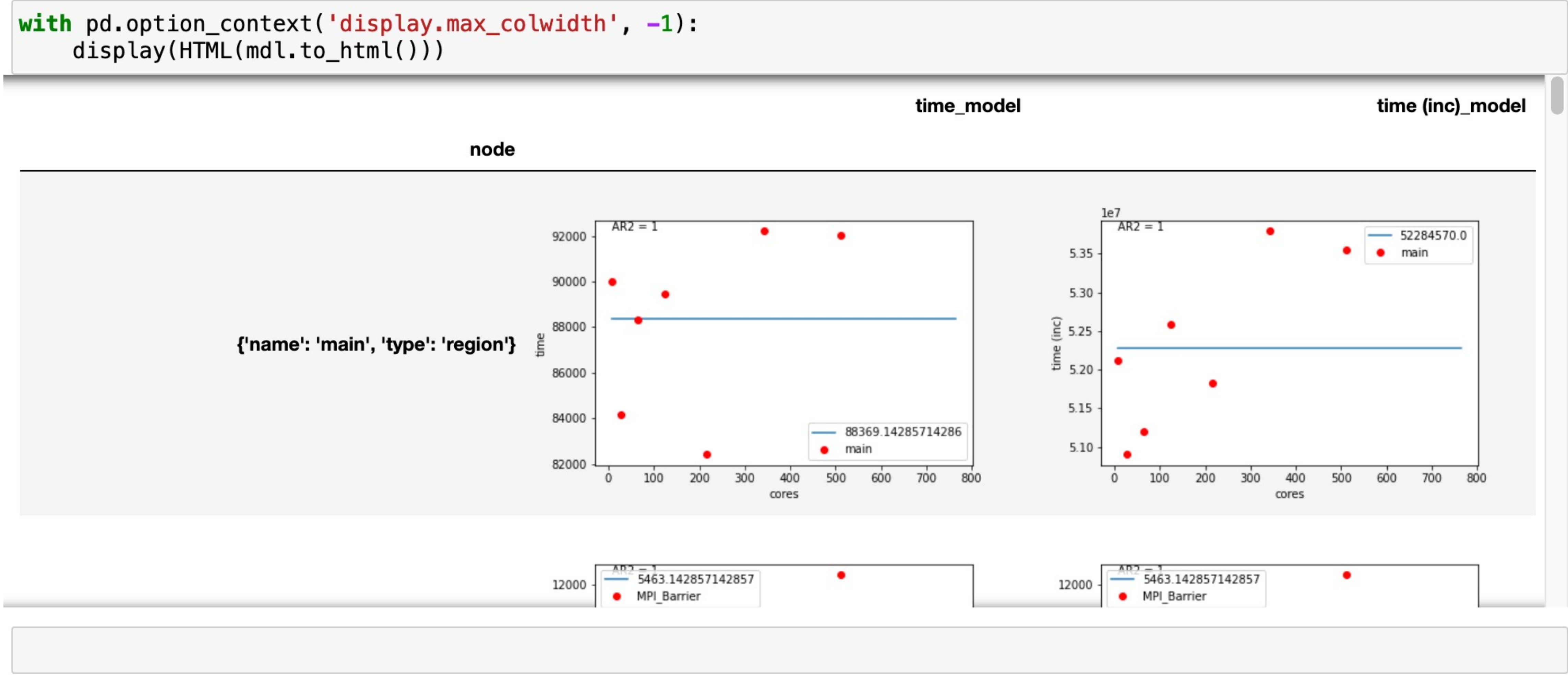


```
fig, ax = model_exc.display()
```

Explicitly ask Matplotlib to show figures:

```
plt.show()
```

Models dataframe with embedded plots



```
with pd.option_context('display.max_colwidth', -1):
    display(HTML(mdl.to_html()))
```

node	time_model	time (inc)_model
{'name': 'main', 'type': 'region'}		

Live demo with Binder

- https://mybinder.org/v2/gh/sshudler/hatchet.git/modeling
- docs/examples/tutorial/hatchet_modeling_demo.ipynb

Conclusion

- Enhance performance analytics with modeling capabilities
- Expand the Extra-P – Hatchet integration to 2+ parameters models
- Experiment with datasets on a longer timeline (SPOT)
- Gather some user feedback

Acknowledgements:
Stephanie Brink
Olga Pearce
David Boehme

CASC

Center for Applied Scientific Computing

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.