



Lecture 12: Analysis/Visualization Tools

Abhinav Bhatele, Department of Computer Science

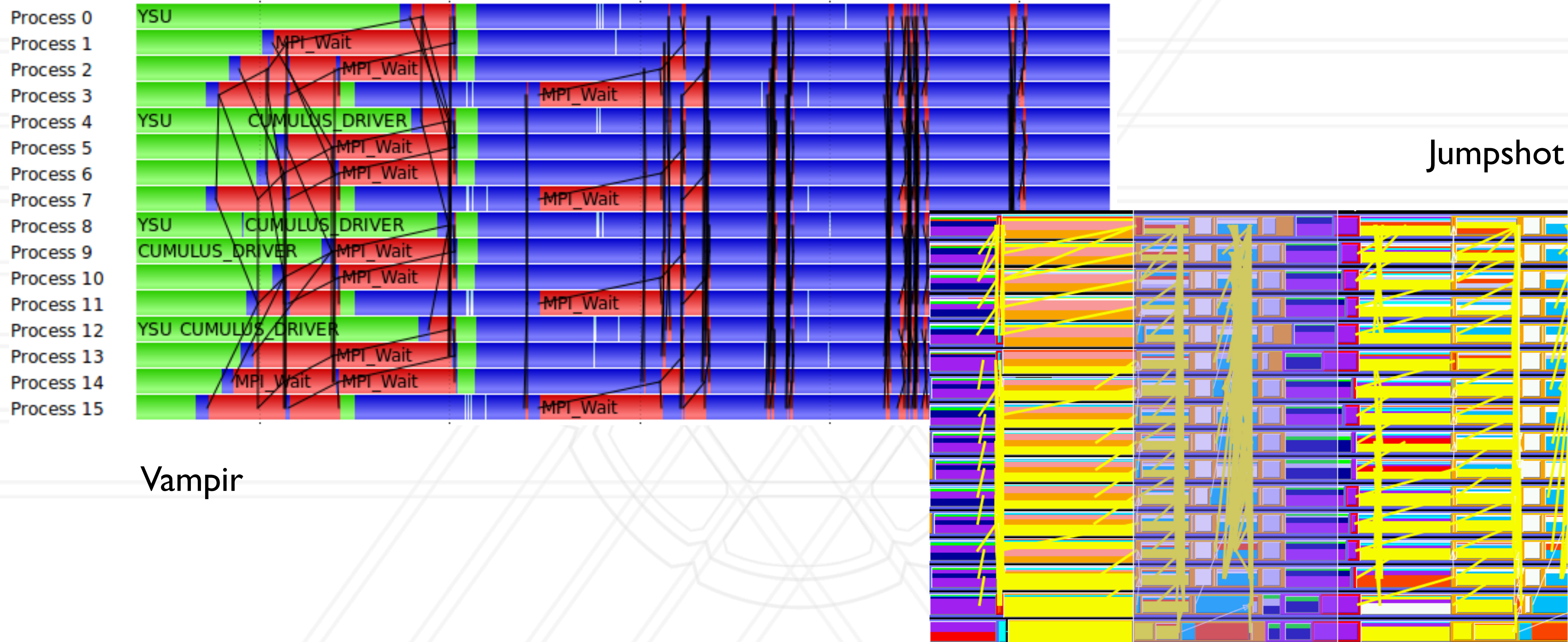


UNIVERSITY OF
MARYLAND

Summary of last lecture

- Performance analysis
 - Identify performance bottlenecks, anomalies
 - Measurement, analysis, visualization tools
- Tracing and profiling
- Calling context trees, graphs

MPI trace visualization



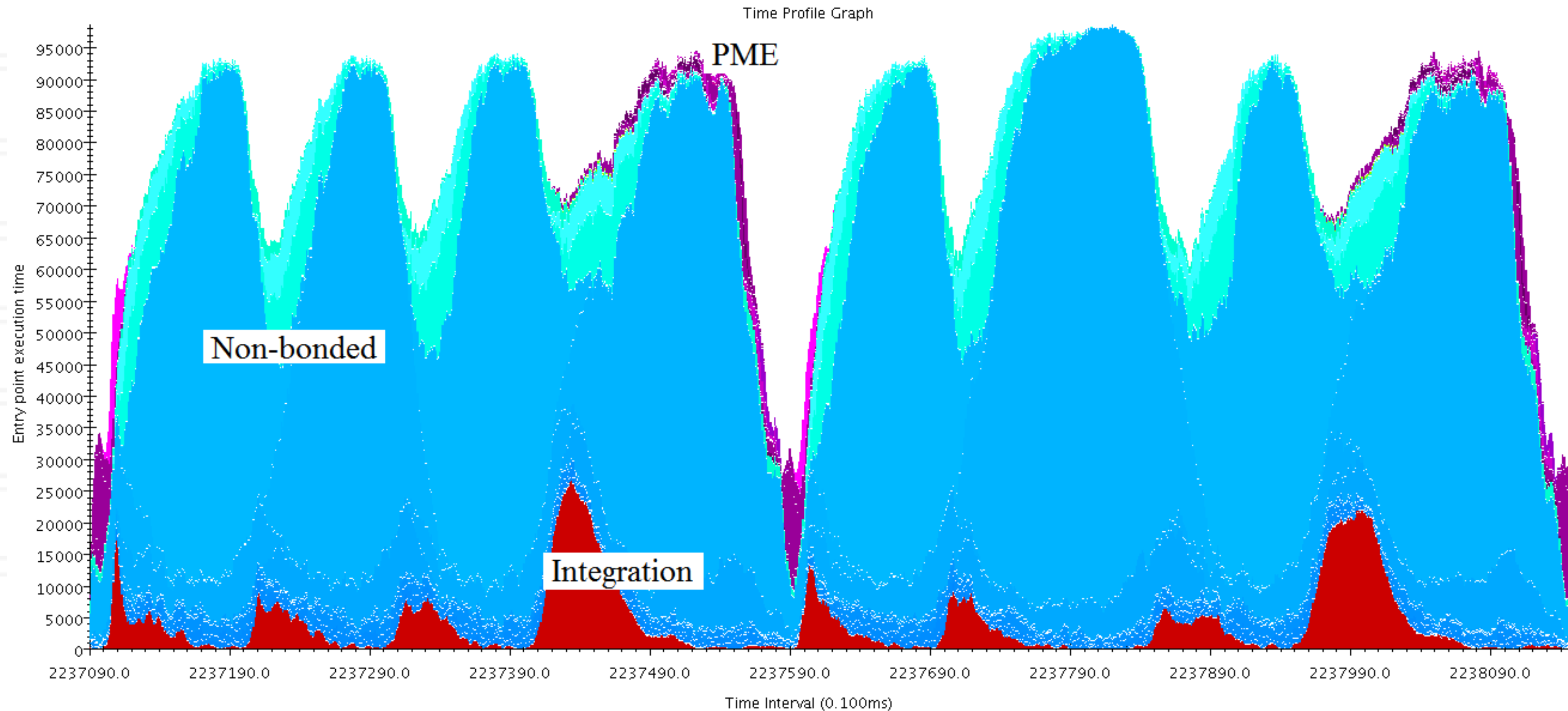
Vampir

Jumpshot

Projections Performance Analysis Tool

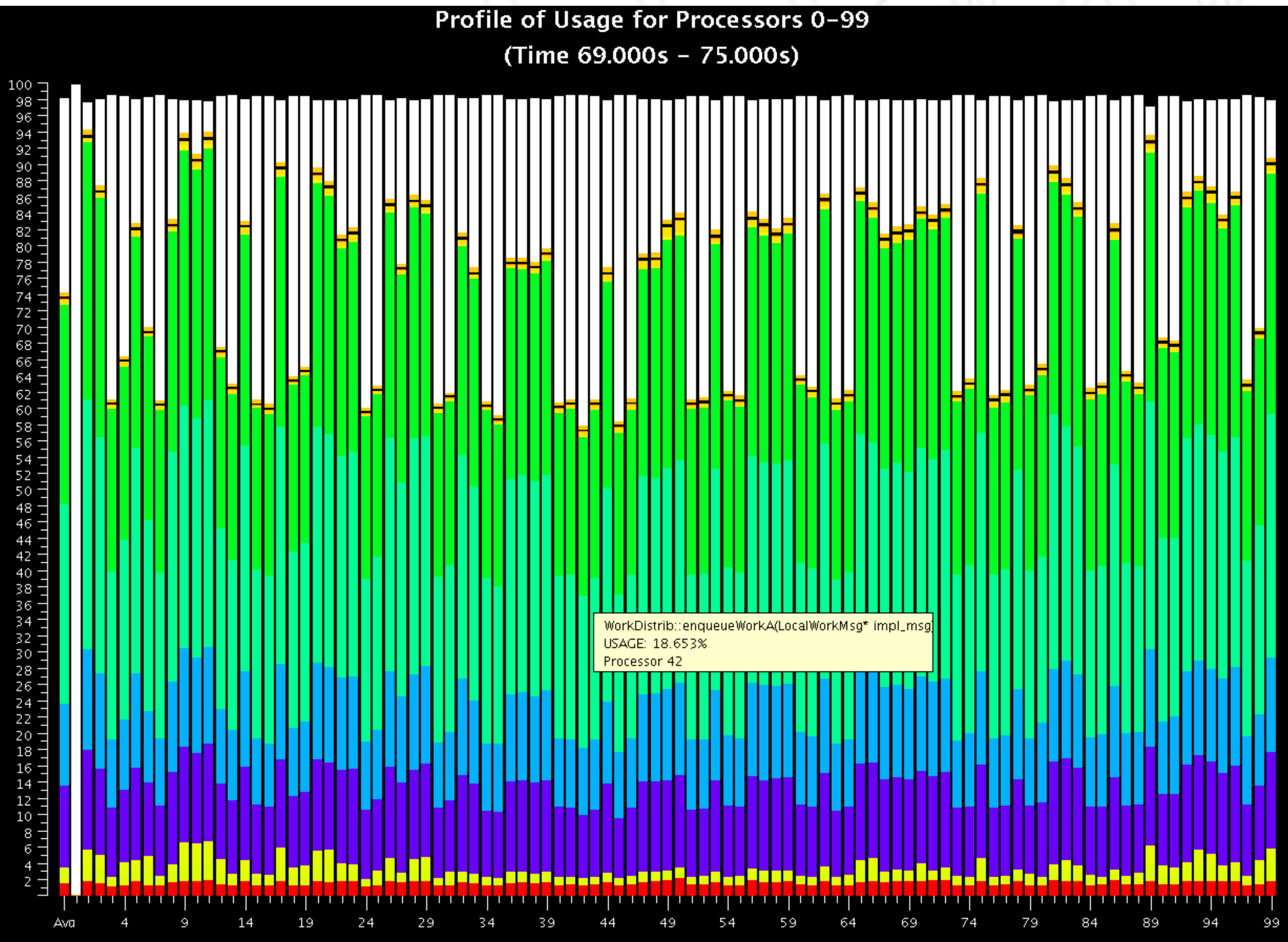
- For Charm++/Adaptive MPI programs
- Instrumentation library
 - Records data at the granularity of chares (Charm++ objects)

Time Profile

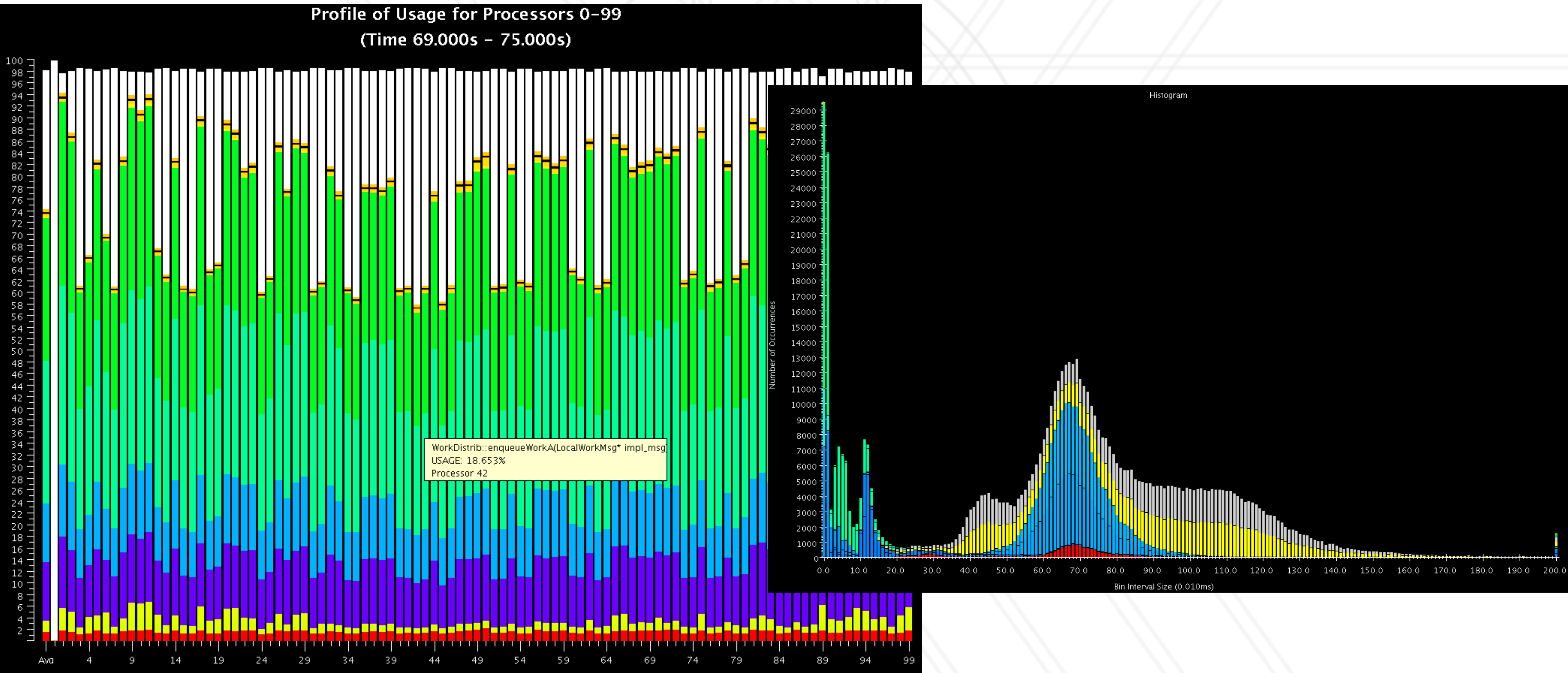


<https://charm.readthedocs.io/en/latest/projections/manual.html>

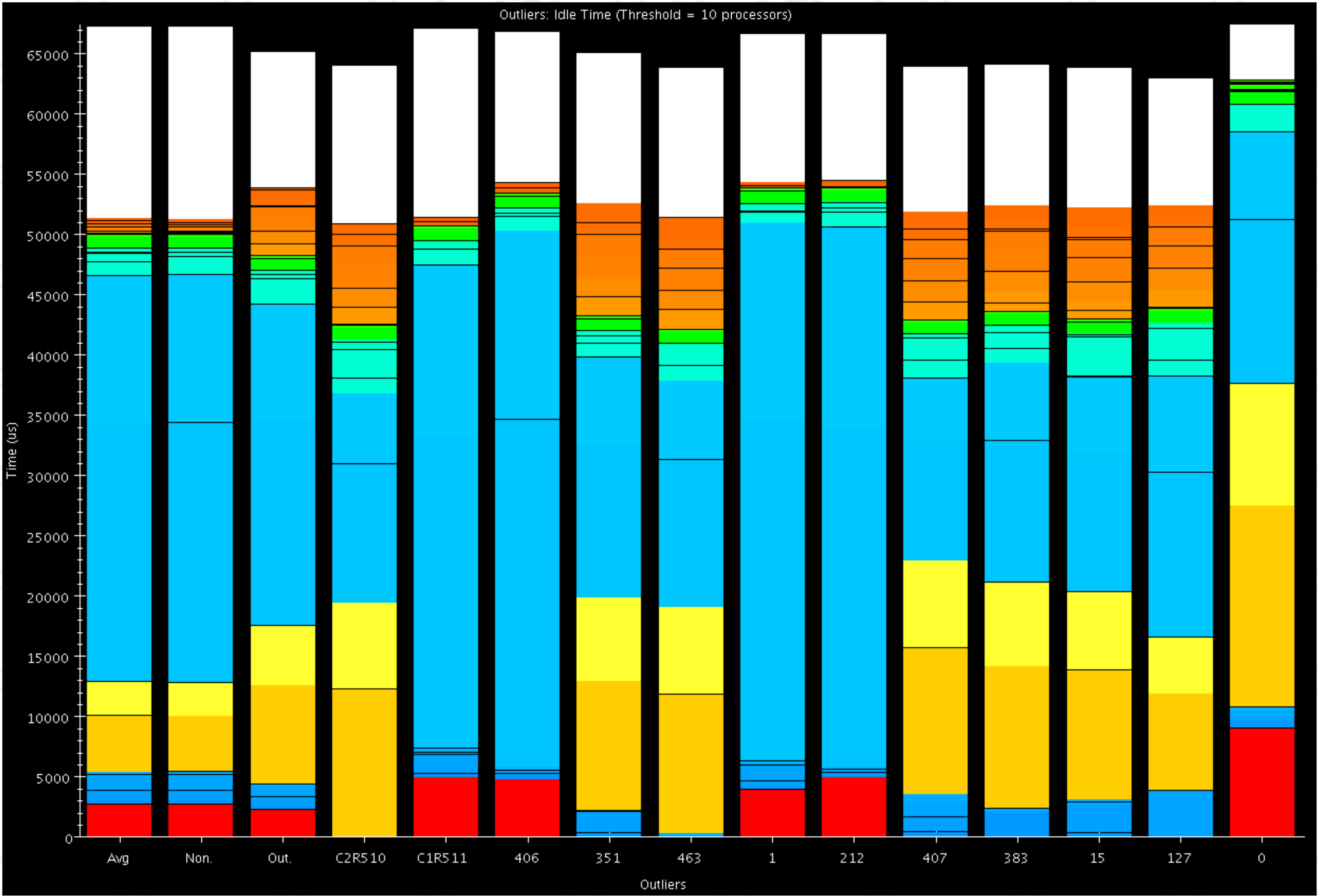
Usage Profile & Histogram View



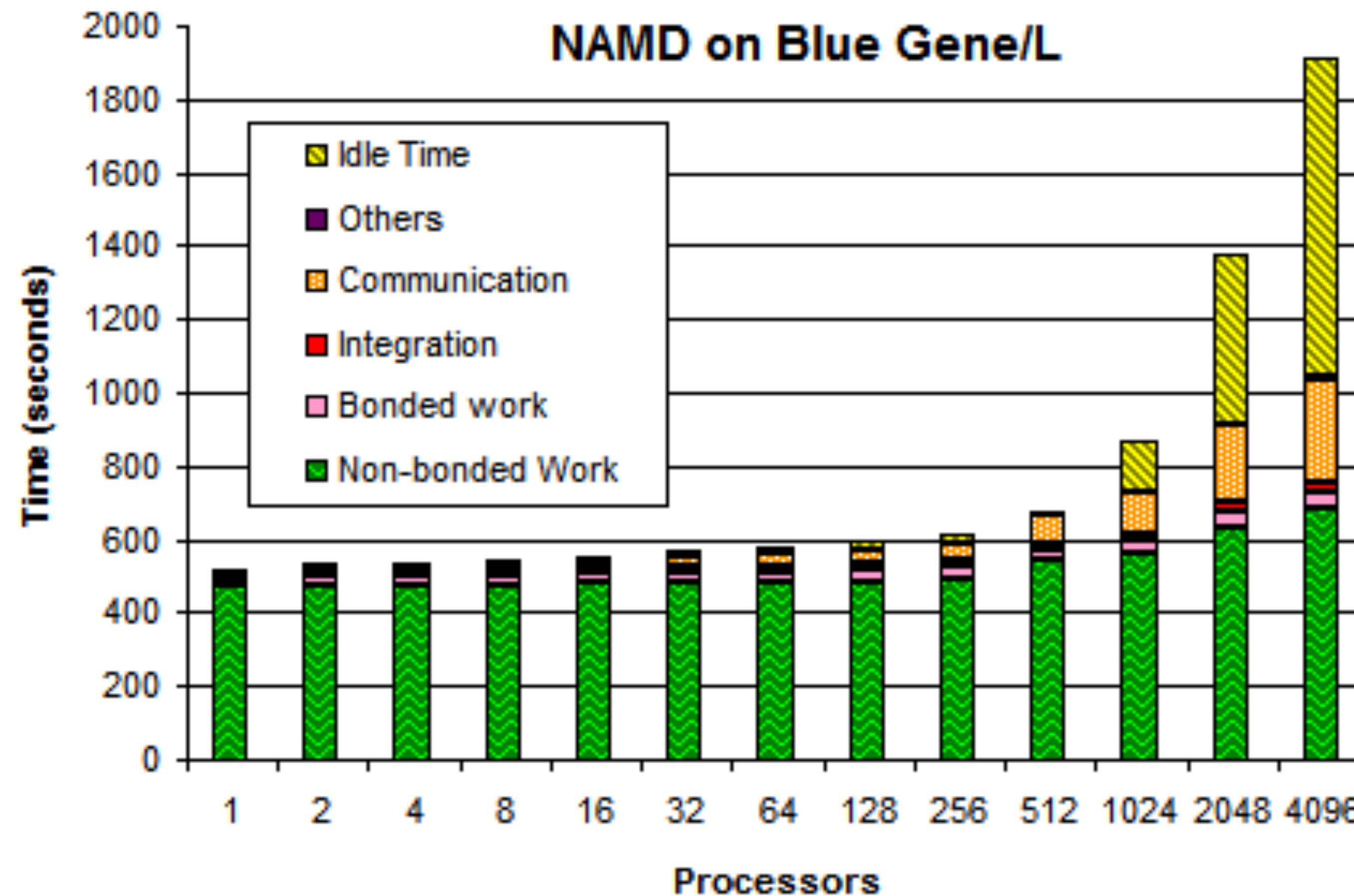
Usage Profile & Histogram View



Outlier Analysis



Scripting for multi-run comparisons




Hatchet

- Hatchet enables programmatic analysis of parallel profiles
- Leverages pandas which supports multi-dimensional tabular datasets
- Create a structured index to enable indexing pandas dataframes by nodes in a graph
- A set of operators to filter, prune and/or aggregate structured data

Dataframe operation: filter

	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

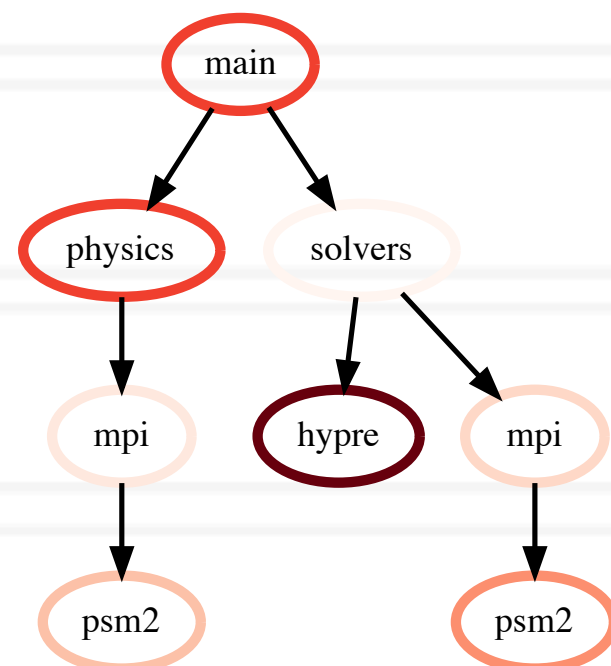


	name	nid	node	time	time (inc)
node					
main	main	0	main	40.0	200.0
physics	physics	1	physics	40.0	60.0
psm2	psm2	3	psm2	15.0	15.0
hypre	hypre	5	hypre	65.0	65.0
psm2	psm2	7	psm2	25.0	25.0

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

Graph operation: squash

	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
hypr	hypr	5	hypr	65.0	65.0
mpi	mpi	6	mpi	10.0	35.0
psm2	psm2	7	psm2	25.0	25.0

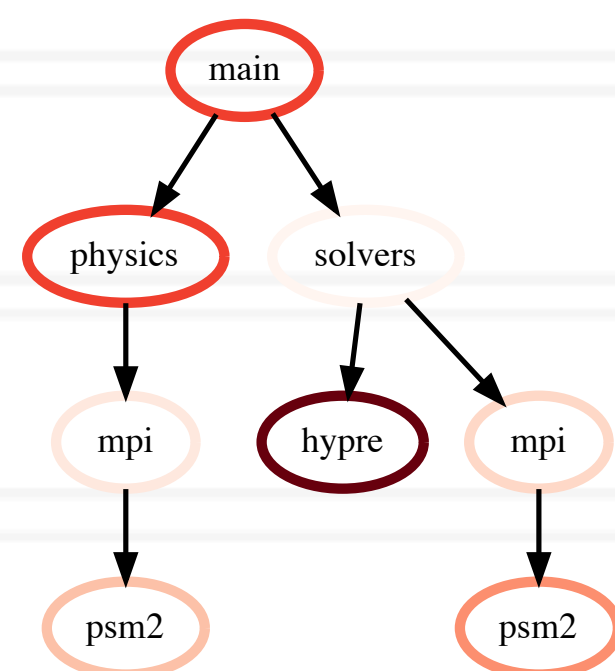


```
1 gf = GraphFrame( ... )
2 filtered_gf = gf.filter(lambda x: x['time'] > 10.0)
```

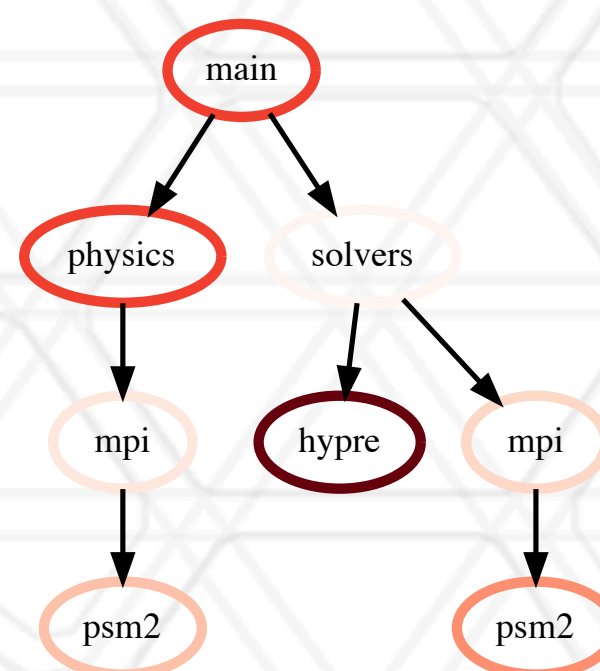
Graph operation: squash

	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

	name	nid	node	time	time (inc)
node					
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



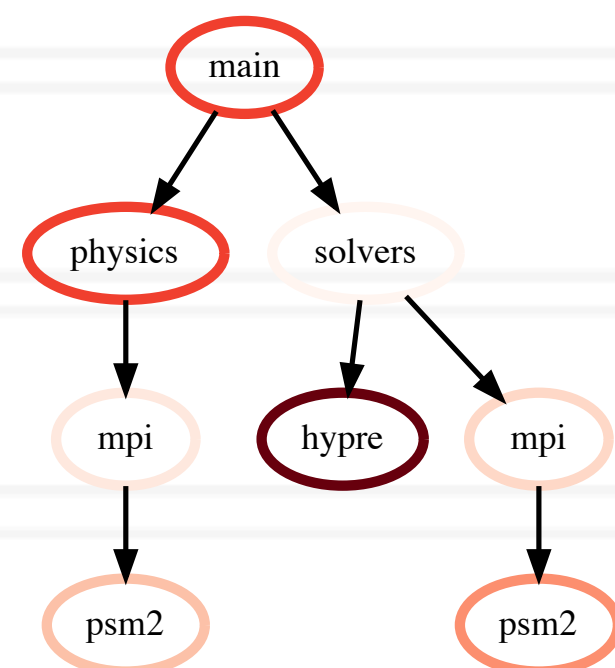
→
filter



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

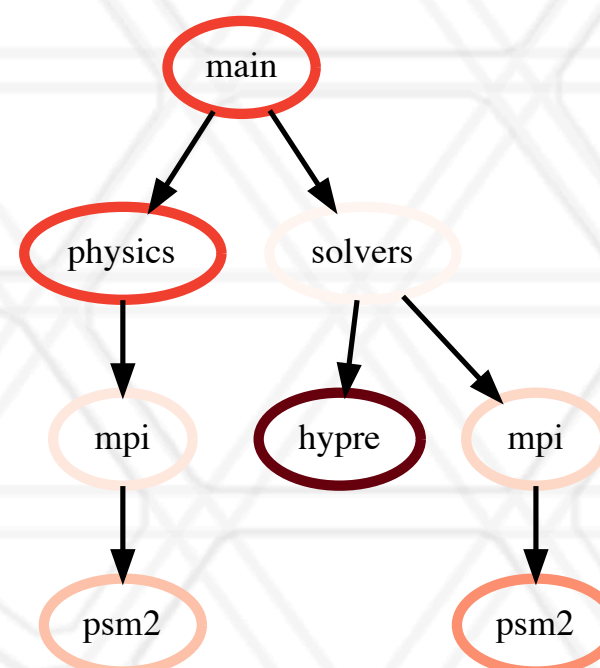
Graph operation: squash

	name	nid	node	time	time (inc)
node					
	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



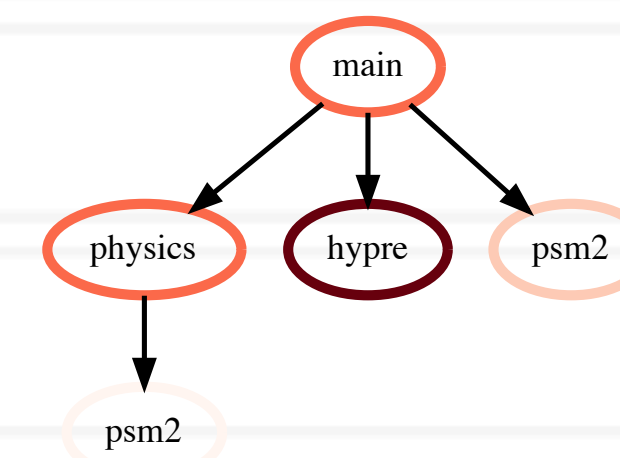
filter

	name	nid	node	time	time (inc)
node					
	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

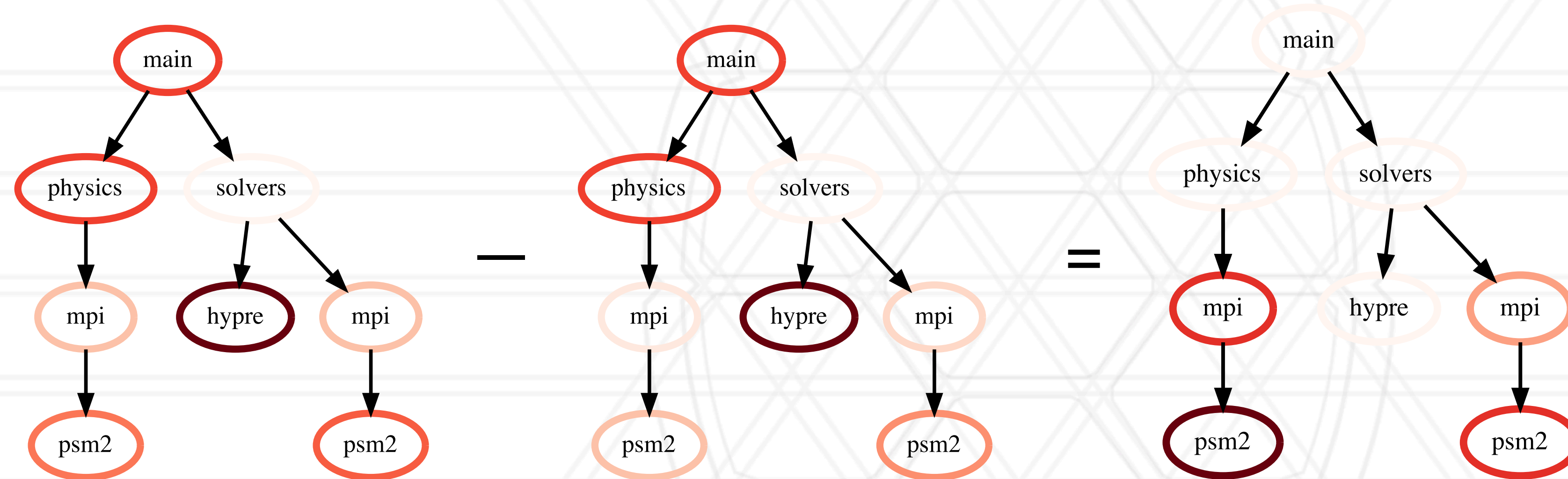
	name	nid	node	time	time (inc)
node					
	main	0	main	40.0	200.0
physics	physics	1	physics	40.0	60.0
psm2	psm2	3	psm2	15.0	15.0
hypre	hypre	5	hypre	65.0	65.0
psm2	psm2	7	psm2	25.0	25.0



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

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

Graphframe operation: subtract

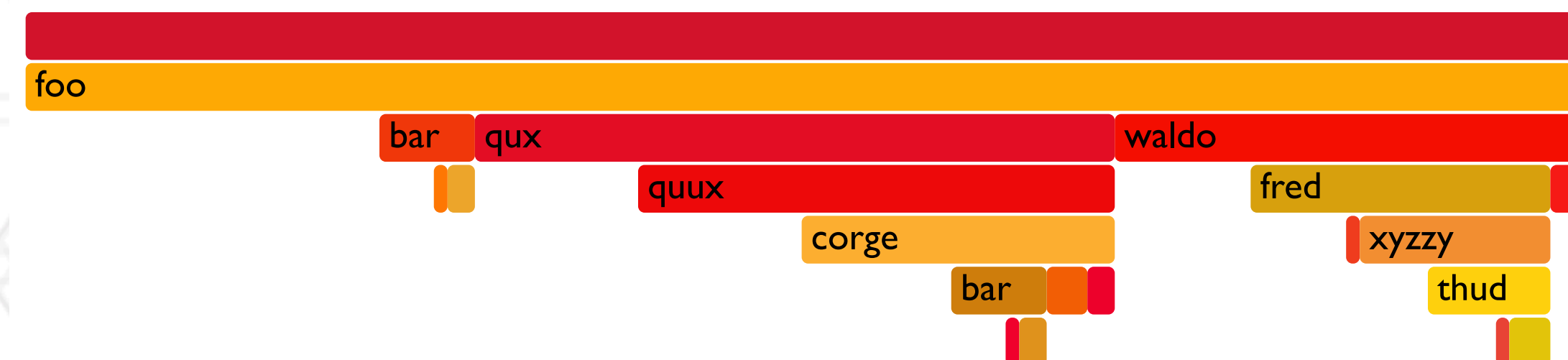
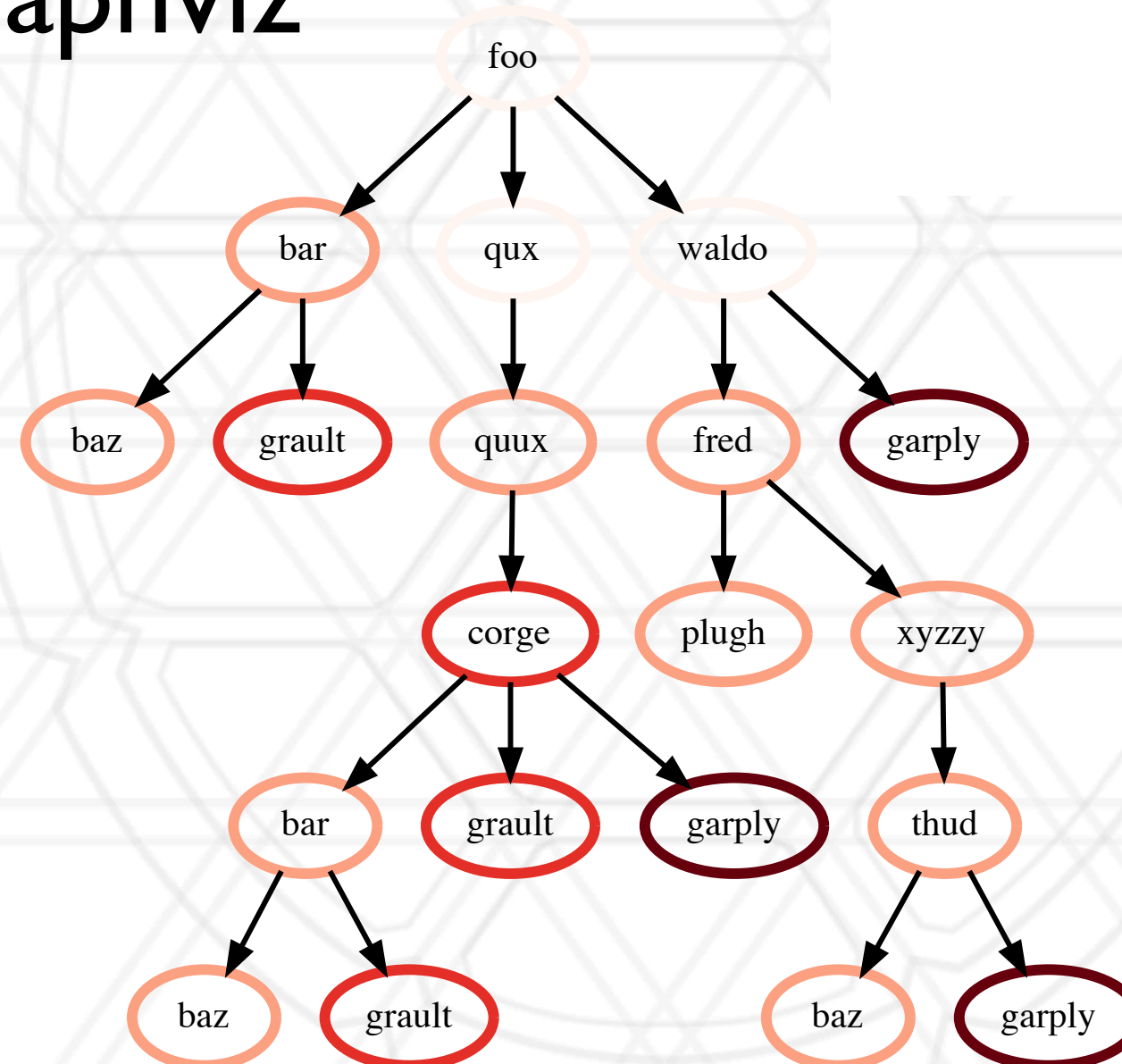


```
1 gf1 = GraphFrame( ... )
2 gf2 = GraphFrame( ... )
3
4 gf2 -= gf1
```

Visualizing output

```
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
```

Graphviz



Flamegraph

Terminal output

Generating a flat profile

	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

```
1 gf = GraphFrame()
2 gf.from_hpctoolkit('kripke')
3
4 grouped = gf.dataframe.groupby('name').sum()
```

Generating a flat profile

	nid	time	time (inc)
name			
<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

	nid	time	time (inc)
module			
/Kripke/build-mvapich2.3/kripke	14366	1.825802e+08	5.847993e+08
/usr/lib64/gcc-4.9.3/hpctoolkit-develop- toolkit/ext-libs/libmonitor.so.0.0.0	2512	0.000000e+00	1.918548e+08
/usr/lib64/ld-2.17.so	9676	0.000000e+00	9.340625e+02
/usr/lib64/libc-2.17.so	37970	0.000000e+00	7.150550e+06
/usr/lib64/libdl-2.17.so	4427	0.000000e+00	2.804062e+02
/usr/lib64/libpsm2.so.2.1	433252	0.000000e+00	2.496037e+06
/usr/lib64/libpthread-2.17.so	2679	0.000000e+00	4.674375e+02
/usr/lib64/libstdc++.so.6.0.20	14945	0.000000e+00	3.898480e+05
/usr/lib64/libintel64_lin/libintlc.so.5	1215	0.000000e+00	9.357812e+01
/usr/lib64/libintel-18.0.1/lib/libmpi.so.12.1.1	126726	0.000000e+00	7.962225e+06

```
1 gf = GraphFrame()
2 gf.from_hpctoolkit('kripke')
3
4 grouped = gf.dataframe.groupby('name').sum()
```

```
1 gf = GraphFrame()
2 gf.from_hpctoolkit('kripke')
3
4 grouped = gf.dataframe.groupby('module').sum()
```

Degree of load imbalance

```
1 gf1 = GraphFrame()
2 gf1.from_caliper('lulesh-512cores')
3
4 gf2 = gf1.copy()
5
6 gf1.drop_index_levels(function=np.mean)
7 gf2.drop_index_levels(function=np.max)
8
9 gf1.dataframe['imbalance']
10    = gf2.dataframe['time'].div(gf1.dataframe['time'])
```

Degree of load imbalance

```
1 gf1 = GraphFrame()
2 gf1.from_caliper('lulesh-512cores')
3
4 gf2 = gf1.copy()
5
6 gf1.drop_index_levels(function=np.mean)
7 gf2.drop_index_levels(function=np.max)
8
9 gf1.dataframe['imbalance']
10    = gf2.dataframe['time'].div(gf1.dataframe['time'])
```

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

Comparing two profiles

```
1 gf1 = GraphFrame()
2 gf1.from_caliper('lulesh-27cores')
3
4 gf2 = GraphFrame()
5 gf2.from_caliper('lulesh-512cores')
6
7 filtered_gf1
8     = gf1.filter(lambda x: x['name'].startswith('MPI'))
9 filtered_gf2
10    = gf2.filter(lambda x: x['name'].startswith('MPI'))
11
12 squashed_gf1 = filtered_gf1.squash()
13 squashed_gf2 = filtered_gf2.squash()
14
15 diff_gf = squashed_gf2 - squashed_gf1
```

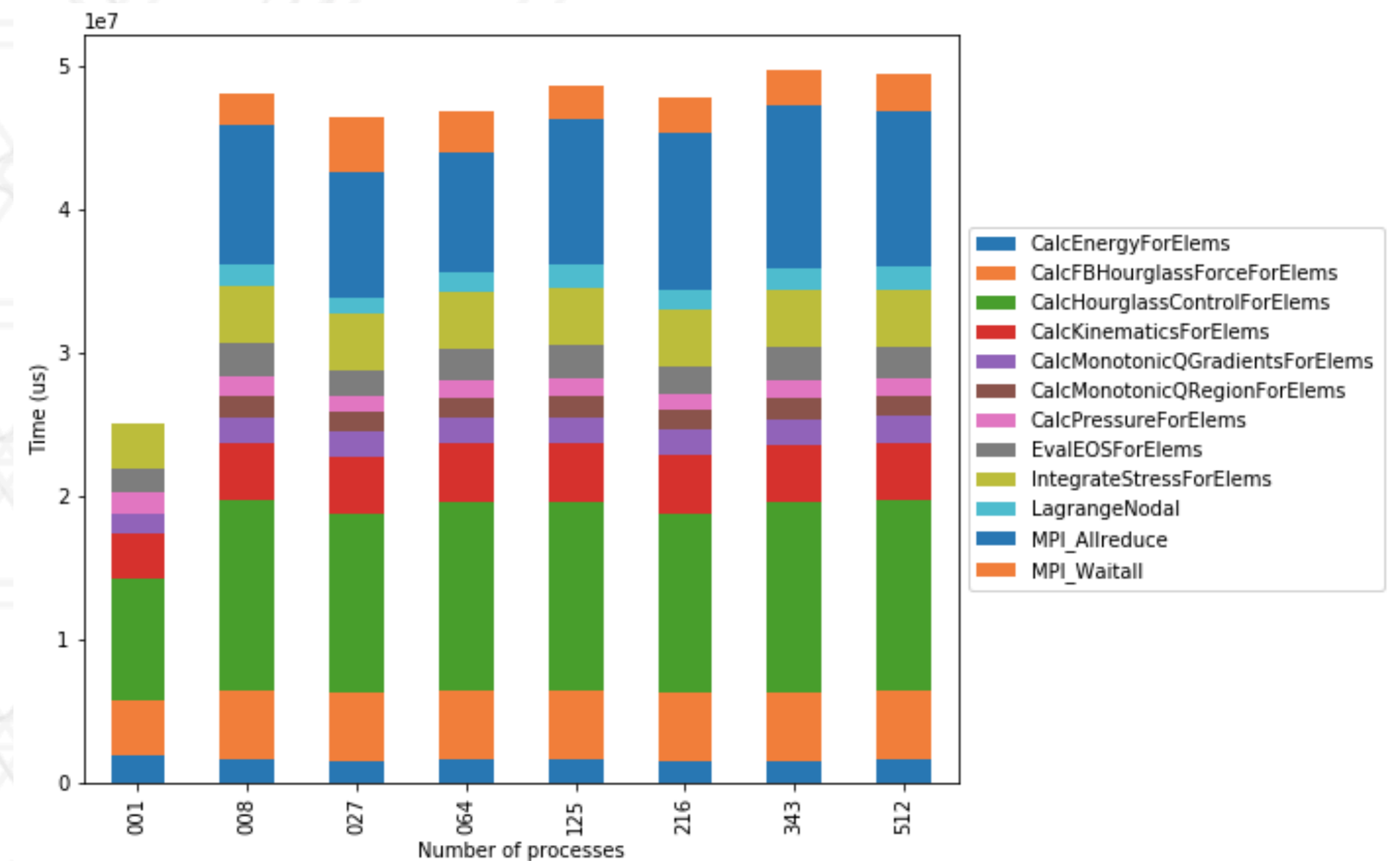
Comparing two profiles

```
1 gf1 = GraphFrame()
2 gf1.from_caliper('lulesh-27cores')
3
4 gf2 = GraphFrame()
5 gf2.from_caliper('lulesh-512cores')
6
7 filtered_gf1
8     = gf1.filter(lambda x: x['name'].startswith('MPI'))
9 filtered_gf2
10    = gf2.filter(lambda x: x['name'].startswith('MPI'))
11
12 squashed_gf1 = filtered_gf1.squash()
13 squashed_gf2 = filtered_gf2.squash()
14
15 diff_gf = squashed_gf2 - squashed_gf1
```

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

Comparing several profiles for scaling

```
1 datasets = glob.glob('lulesh*.json')
2 datasets.sort()
3
4 dataframes = []
5 for dataset in datasets:
6     gf = GraphFrame()
7     gf.from_caliper(dataset)
8     gf.drop_index_levels()
9
10    num_pes = re.match('(.*)-(\d+)(.*)', dataset).group(2)
11    gf.dataframe['pes'] = num_pes
12    filtered_gf = gf.filter(lambda x: x['time'] > 1e6)
13    dataframes.append(filtered_gf.dataframe)
14
15 result = pd.concat(dataframes)
16 pivot_df = result.pivot(index='pes', columns='name', values
17                          = 'time')
17 pivot_df.loc[:, :].plot.bar(stacked=True, figsize=(10,7))
```



Questions

Scaling Applications to Massively Parallel Machines Using Projections ...

- What is AMPI?
- Is there any standardized data format to store performance profiling/analysis results?
- Does Projections support heterogeneous systems (like a node with a CPU and multiple GPUs)?
- Performance analysis and tuning, in general, seems to incorporate a lot of experience and hand crafting. Are there tools that generate suggestions for possible code modifications based on the profiling result?
- Can we go over the load balancing? Why does the balance look a little worse (and the overall load higher) after refinement? The paper talks about quirks in background load leading to underutilization in a range of processors. What sorts of quirks can lead to this type of behavior?
- How do parallel simulators work? The paper mentions BigSim. Is this a popular one? Is it common for people to use a simulator before running on a large supercomputer?

Questions

Hatchet: Pruning the Overgrowth of Parallel Profiles

- What is the definition of reproducibility in performance analysis?
- A programmable tool is great to automate analysis, but I guess a dedicated interactive GUI is also very useful for some analysis. Are there plans to incorporate such elements?
- Which profiling tool is most recommended to generate profile data for the processing with Hatchet?
- Is the library open-sourced, or are there any plans?
- How is it that the drop_index_levels performance is able to remain basically constant until getting to about 256 processors? Also, what's with the strange shape of the filter performance graph? And is 512 processors as the max for the performance test for the tool a little on the low end? Would the analysis tool be usable to look at profiling results from a real or simulated run on a supercomputer?
- Hatchet is ~2.5k lines of code. What were some of the most complicated parts to implement? Could you go over the design of the code briefly

Questions?



UNIVERSITY OF
MARYLAND

Abhinav Bhatele

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

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