Analyzing Performance Profiles Using Hatchet Abhinav Bhatele, Department of Computer Science













Code-centric performance analysis

prof Data	Q Enter filter	Q Enter filter text				
bytes per bucket, each sample count	ts 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



Understanding performance bottlenecks is critical to optimizing (parallel) software

• Profiling and tracing tools help identify parts of code that consume the most time





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























Contextual information

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







Contextual information

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

Performance Metrics

- Time
- Flops
- Cache misses









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



00	0	hmc	
dp p	arscalar	specific.h	
	15	//! Low level nobk to QMP_global_sum	
	75	inline void globalSumArray(float *dest, int len)	
	77		
	78	QMP_sum_float_array(dest, len);	
	79		
	50		
	81	//! Low level hook to OMP_global_sum	
	82	inline void globalSumArray(double *dest, int len)	
	53		
0	54	QMP_sum_double_array(dest, len);	
-	85		
	86		
	57	/// Global sum on a multild	
	88	template <class t=""></class>	
	89	inline void globalSumArray(multi1d <t>& dest)</t>	
	90		
	41	// The implementation here is relying on the structure being nacked) 4 . 6

Calling Context View Callers View Flat View

Scopes 🕺 🕅 🖓	⇒ samples (I) ∇	# samples (E)
👺 main	5.80e05 97.5%	1
▼ ₩ void Chroma::doHMC <chroma::hmctrjparams>(OCP::multi1d<odf< td=""><td>5.70005 95.8%</td><td></td></odf<></chroma::hmctrjparams>	5.70005 95.8%	
loop at hmc.cc: 311-435	5.65405 95.08	
TRy Chroma::AbsHMCTrj < QDP::multi1d < QDP::OLattice < QDP::PScala	5.65005 95.08	
Text Chromatti at ColMatSTSLeap frogRecursive Integrator toperator();	5.25e05 88.2%	
Ioop at icm_sts_leapfrog_recursive.cc: 129-131	4.85003 81.58	
🔻 🖶 Chromatti atCol NatExpSdtintegrator:::operator:)(ChromattA	4.25605 71.48	
Ioop at icm_exp_sdt.cc: 85-88	4.25eC5 71.48	
▼ B⇒ Chroma:: LCMMDIntegratorSteps::leapP(QDP::multi1d <	4.25eC5 71.4%	
▼ 勝 Chroma:: TwoFlavorExactWilsonTypeFermMonomial<	3.30e05 55.5%	
TIP: Chroma::TwoFlavorExactWilsonTypeFermMonomia ^	2.30005 38.78	
▼ 卧 Chroma::MdagM5ysSolverCG < QDP::OLattice < QL	2.20e05 37.0%	
loop at syssolver_mdagm_cg.h: 56-70	2.20e05 37.0%	
T B: Chroma: SystemSolverResults_t Chroma: Inv	2.2CeC5 37.C%	
🔻 🔛 Chroma:::SystemSolverResults_t Chroma::h	2.2CeC5 37.0%	1.00604 1.78
▼ loop at inveg2.cc: 147-182	1.85605 31.1%	5.0003 0.8%
Chroma::EvenOddPrecWilsonLinOp::or	1.05eC5 17.6%	1.00604 1.75
B) Chroma::EvenOddPrecWilsonLinOp.com	7.0CeC4 11.8%	1.50604 2.5%
[I] globalSumArray	5.00e03 0.8%	
[I] vaxpy3	5.0CeC3 C.8%	0.00e03 0.8%
► [I] local sumsq		
	2) 4 - 5

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



DO	0	hme	
dp p	arscalar	.specific.h	
	15	//! Low level nook to QMP_global_sum	
	75	inline void globalSumArray(float *dest, int len)	
	77		
	78	QMP_sum_float_array(dest, len);	
	79		
	50		
	81	//! Low level hook to OMP global sum	
	82	inline void globalSumArray(double *dest, int len)	
	53		
0	54	QMP_sum_couble_array(cest, len);	
-	85		
	86		
	57	/// Global sum on a multild	
	88	template <class t=""></class>	
	89	inline void globalSumArrav(multi1d <t>& dest)</t>	
	90		
	41	// The implementation here is relying on the structure being packed	
			34.4

Calling Context View Callers View Flat View

Scopes 🕺 🕂 🕂	⇒ samples (I) ∇	# samples (E)	
# main	5.80eC5 57.5%	1	1
▼ ₩> void Chroma::doHMC <chroma::hmctrjparams>(QDP::multi1d<qdf< td=""><td>5.7CeC5 95.8%</td><td></td><td></td></qdf<></chroma::hmctrjparams>	5.7CeC5 95.8%		
Ioop at hmc.cc: 311-435	5.65405 95.08		
The Chroma::AbsHMCTrj <qdp::multi1d<qdp::olattice<qdp::pscala< td=""><td>5.65005 95.08</td><td></td><td></td></qdp::multi1d<qdp::olattice<qdp::pscala<>	5.65005 95.08		
Text ChromatiliatColMatST5LeaptrogRecursiveIntegrator::operator()	5.25e05 88.2%		
Ioop at icm_sts_leapfrog_recursive.cc: 129-131	4.05eC3 01.50		
▼ B→ Chromat LatColNatExpSdtintegrator::operator()(Chromat:A	4.25608 71.48		1
Ioop at icm_exp_sdt.cc: 85-88	4.25e00 71.4%		
▼ ₽> Chroma::LCMMDIntegratorSteps::leapP(ODP::multi1d <	4.25eC5 71.48		
▼ ⊯ Chroma:: 1 wolf lavorExactWilsonTypeFermMonomial<	3.30e05 55.5%		
▼除 Chroma::TwoFlavorExactWilsonTypeFermMonomia	2.3CeC5 38.78		
▼ 卧 Chroma ::MdagM5ysSolverCG < QDP::Olattice < QI	2.20e05 37.0%		
loop at syssolver_mdagm_cg.h: 56–70	2.2CeC5 37.CB		
T 🕏 Chroma: SystemSolverResults_t Chroma: Inv	2.2CeC5 37.C8		
🔻 🔛 Chroma:::SystemSolverResults_t Chroma::h	2.2CeC5 37.0%	1.00e04 1.7	8
▼ loop at inveg2.ec: 147-182	1.85eC5 31.1%	5.C0e03 0.8	8
Chroma::EvenOddPrecWilsonLinOp::or	1.05eC5 17.6%	1.00604 1.7	
► B Chroma::EvenOddPrecWilsonLinOp::or	7.0CeC4 11.8%	1.50e04 2.5	8
[I] globalSumArray	5.00e03 0.8%		
► (I) vaxpy3	5.00e03 C.8%	5.00e03 0.8	8
► (I) local sumsq			
	200.00 2 40		

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 is an open-source Python library for data analysis





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



C		
CO	um	INS

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



	Inde	x // C	olumns		
		node	name	time (inc)	time
	0	{'name': 'main'}	main	200.0	10.0
	7 1	{'name': 'physics'}	physics	60.0	40.0
	2	{'name': 'mpi'}	mpi	20.0	5.0
Rows	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 is an open-source Python librar for data analysis
- Dataframe: two-dimensional tabular da structure
 - Supports many operations borrowed from SQL databases
- MultiIndex enables working with highdimensional data in a 2D data structur



	Index	x // //C	olumns		
^y		node	name	time (inc)	time
	0	{'name': 'main'}	main	200.0	10.0
	1	{'name': 'physics'}	physics	60.0	40.0
lla	2	{'name': 'mpi'}	mpi	20.0	5.0
Rov	VS 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
e	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

physics

mpi

psm2

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



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



se of MultiIndex								
Vhen metric	s ar	re per	MPI F	orc	cess	thread etc	/	
node	rank	time (inc)	time	nid	rank	file	line	module
{'type': 'function', 'name': ' <program root>'}</program 	0	9992 38.0	0.0	2	0	<unknown file=""></unknown>	0	/collab/usr/global/tools/hpctoolkit/chaos_5_x8
	1	9993 90.0	0.0	2	1	<unknown file=""></unknown>	0	/collab/usr/global/tools/hpctoolkit/chaos_5_x8
{'type': 'function', 'name': '27:MPI_Init'}	0	0.0	0.0	6	0	<unknown file=""></unknown>	0	/collab/usr/global/tools/hpctoolkit/chaos_5_x8
	1	0.0	0.0	6	1	<unknown file=""></unknown>	0	/collab/usr/global/tools/hpctoolkit/chaos_5_x8
{'type': 'function', 'name': 'main'}	0	9992 38.0	0.0	4	0	./src/cpi.c	19	ср
	1	9993 90.0	0.0	4	1	./src/cpi.c	19	cp





Use of MultiIndex

• When metrics are per MPI process, th





	/	
file	line	module
<unknown file=""></unknown>	0	/collab/usr/global/tools/hpctoolkit/chaos_5_x8
./src/cpi.c	19	cpi
./src/cpi.c	19	cpi
	file <unknown file=""> <unknown file=""> <unknown file=""> ile></unknown></unknown></unknown>	fileline <unknown file="">0<unknown file="">0<unknown file="">0<unknown file="">10<unknown file="">19./src/cpi.c19</unknown></unknown></unknown></unknown></unknown>



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 in

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



\$ pip install hatchet

https://github.com/hatchet/hatchet



Reading in an input dataset

Installing hatchet

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



\$ pip install hatchet

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

\$ pip install hatchet Installing hatchet

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

Contribute a reader to hatchet!



https://github.com/hatchet/hatchet

import hatchet as ht

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



Dataframe operation: filter





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





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



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	5090 5781.0	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)

		nid	rank	time	time (inc)	nan	ne	
node	rank							
{'type': 'function', 'name': u'main'}	0	0.0	0	83309.0	5090 5781.0	ma	ain	
	1	0.0	1	83920.0	50905819.0	ma	ain	
	2	0.0	2	83023.0	50905849.0	ma	ain	
	3	0.0	3	83258.0	50905840.0	ma	ain	
				n	ode	name		
node								
{'type': 'function', 'name': u'main'}	{'typ	be': 'fur	nction',	'name': u'm	ain'}	main		
{'type': 'function', 'name': u'lulesh.cycle'}		{'t	ype': 'fu	unction', 'na u'lulesh.cy	me': cle'}	lulesh.cycle		
{'type': 'function', 'name': u'TimeIncrement'}		{'t	ype': 'fu u'	unction', 'na TimeIncrem	me': ent'}	TimeIncrement		
{'type': 'function', 'name': u'LagrangeLeapFrog'}		{'t	ype': 'fu u'Lagr	unction', 'na angeLeapFr	me': 'og'}	LagrangeLeapFrog		

		nid	rank	time	time (inc)	na	me
node	rank						
{'type': 'function', 'name': u'main'}	0	0.0	0	83309.0	50905781.0	r	ain
	1	0.0	1	83920.0	50905819.0	r	ain
	2	0.0	2	83023.0	50905849.0	m	ain
	3	0.0	З	83258.0	50905840.0	r	ain
				n	ode	name	
node							
{'type': 'function', 'name': u'main'}	{'type	e': 'fund	ction', '	name': u'm	ain'}	main	
{'type': 'function', 'name': u'lulesh.cycle'}		{'ty	pe': 'fu	nction', 'na u'lulesh.cy	me': cle'}	lulesh.cycle	
{'type': 'function', 'name': u'TimeIncrement'}		{'ty	pe': 'fu u'T	nction', 'na TimeIncrem	me': ent'}	TimeIncrement	
{'type': 'function', 'name': u'LagrangeLeapFrog'}		{'ty	pe': 'fu u'Lagra	nction', 'na angeLeapFr	me': og'}	LagrangeLeapFrog	





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





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

							name	nid	node	time	time (inc)						
	name	nid	node	time	time (inc)	node											
nada	name	ma	noue	une	une (me)	main	main	0	main	40.0	200.0						
node				10.0		physics	physics	1	physics	40.0	60.0		name	nid	node	time	time (inc)
main	main	0	main	40.0	200.0	mpi	mpi	2	mpi	5.0	20.0	node					
physics	physics	1	physics	40.0	60.0	psm2	psm2	3	psm2	15.0	15.0	main	main	0	main	40.0	200.0
mpi	mpi	2	mpi	5.0	20.0	solvers	solvers	4	solvers	0.0	100.0	physics	physics	1	physics	40.0	60.0
psm2	psm2	3	psm2	15.0	15.0	bypro	bypro	5	bypro	65.0	65.0	nsm2	nsm2	3	nsm2	15.0	15.0
solvers	solvers	4	solvers	0.0	100.0	nypre	nypre	5	nypre	05.0	05.0	burne	burgers	5	burgers	05.0	0.0
hypre	hypre	5	hypre	65.0	65.0	mpi	mpi	6	mpi	10.0	35.0	nypre	nypre	5	nypre	65.0	65.0
mni	mpi	6	mni	10.0	35.0	psm2	psm2	7	psm2	25.0	25.0	psm2	psm2	7	psm2	25.0	25.0
inpi	mpi	-	mpi	10.0	00.0	nom0		7		25.0	05.0						
psm2	psm2	7	psm2	25.0	25.0	psinz	psm2		psm2	25.0	25.0						



PSSG PARALLEL SOFTWARE AND SYSTEMS GROUP

1	
n	ain
sics	solvers
pi	hypre mpi
n2	psm2



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

							name	nid	node	time	time (inc)	
		mid	nede	.	time (in c)	node						
	name	nia	node	ume	time (inc)	main	main	0	main	40.0	200.0	
node						physics	physics	1	physics	40.0	60.0	
main	main	0	main	40.0	200.0	mpi	mpi	2	mpi	5.0	20.0	node
physics	physics	1	physics	40.0	60.0	psm2	psm2	3	psm2	15.0	15.0	main
mpi	mpi	2	mpi	5.0	20.0	solvers	solvers	4	solvers	0.0	100.0	physics
psm2	psm2	3	psm2	15.0	15.0	bypro	bypro	5	bypro	65.0	65.0	psm2
solvers	solvers	4	solvers	0.0	100.0	nypre	nypre	0	nypre	10.0	05.0	bypre
hypre	hypre	5	hypre	65.0	65.0	mpi	mpi	6	mpi	10.0	35.0	nypre
mpi	mpi	6	mpi	10.0	35.0	psm2	psm2	7	psm2	25.0	25.0	psmz
psm2	psm2	7	psm2	25.0	25.0	psm2	psm2	7	psm2	25.0	25.0	



PSSG PARALLEL SOFTWARE AND SYSTEMS GROUP

> .	10	.0)			<pre>squashed_gf = filtered_gf.squash()</pre>
name	nid	node	time	time (inc)	
main	0	main	40.0	200.0	
nysics	1	physics	40.0	60.0	
psm2	3	psm2	15.0	15.0	
hypre	5	hypre	65.0	65.0	
psm2	7	psm2	25.0	25.0	
ma	ain				
cs	hy	pre	mpi		
2			psm2		



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

							name	nid	node	time	time (inc)							name	nid	noo	de t	time tir	ne (inc)						
	name	nid	node	time	time (inc)	node											node												
node						main	main	0	main	40.0	200.0						main	main	0	ma	ain 4	40.0	200.0						
main	main	0	main	40.0	200.0	physics	physics	1	physics	40.0	60.0		name	nid	node	time	^{ti} physics	physics	1	physi	cs 4	40.0	60.0		name	nid	node	time	time (inc)
nhysics	nhysics	1	physics	40.0	60.0	mpi	mpi	2	mpi	5.0	20.0	node					mpi	mpi	2	m	ipi	5.0	20.0	node					
physics	priysics	۰ ۵	priysics	40.0 E 0	00.0	psm2	psm2	3	psm2	15.0	15.0	main	main	0	main	40.0	psm2	psm2	3	psn	n2 [·]	15.0	15.0	main	main	0	main	40.0	200.0
mpi	mpi	2	mpi	5.0	20.0	solvers	solvers	4	solvers	0.0	100.0	physics	physics	1	physics	40.0	solvers	solvers	4	solve	ers	0.0	100.0	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	3	psm2	15.0	hypre	hypre	5	hyp	ore (65.0	65.0	psm2	psm2	3	psm2	15.0	15.0
solvers	solvers	4	solvers	0.0	100.0	mpi	mpi	6	mpi	10.0	35.0	hypre	hypre	5	hypre	65.0	mpi	mpi	6	m	ipi [.]	10.0	35.0	hypre	hypre	5	hypre	65.0	65.0
hypre	hypre	5	hypre	65.0	65.0	psm2	psm2	7	psm2	25.0	25.0	psm2	psm2	7	psm2	25.0	psm2	psm2	7	psn	n2 2	25.0	25.0	psm2	psm2	7	psm2	25.0	25.0
mpi	mpi	6	mpi	10.0	35.0			-				1			7.4		1/1	<u> </u>	, i.,										
psm2	psm2	7	psm2	25.0	25.0	psm2	psm2	7	psm2	25.0	25.0																		
							nam	ne nic	d node	time	time (inc)			\mathbf{i}				-2	Ú.										
	(main				no	de C	: ٦	┶┙				mai	in				<i>))</i> \			~								
						ma	ain ma	in () mak	40.0	200.0							// 2	5(łU	U	Sr					main		
						physi	cs physic	cs 1	l physics	40.0	60.0	n pł	ame hid	noc	e tinte vers	time (inc				\sim						/	\prec		
	physics		solvers			m	npi m	ipi 2	2 mpi	5.0	20.0	node					_/X												
						psr	m2 psm	n2 3	3 psm2	15.0	15.0	main	main O	ma	n 40.0	200.0	- / /								ohysics	ľ	nypre	psm	2
			-	*		solve	e rs solve	ers 2	1 solvers	0.0	100.0	physics phy	vsics 1	physic	ife 20.0	mpi 60 (
	mpı		nypre	mp	01	hyp	bre hyp	re 5	5 hypre	65.0	65.0	psm2 p	sın2 3	psm	2 15.0	15.0									psm2				
						m	npi m	ipi f	6 mpi	10.0	35.0	hypre h	v re 5	hyp	re 65.0	65.0									1				
(psm2			psm	12	psr	m2 psm	n2 7	7 psm2	25.0	25.0	psm2 👔	sm2 7	psm	2 25.0	psm225													
				$\mathbf{>}$																									



time (inc)

200.0

60.0

15.0

65.0

25.0

squashed_gf = filtered_gf.squash()





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













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').s sorted_df = grouped.sort_values(by=['tim print(sorted_df)



sum()		nid		time	time (inc)
ne'], asce	name				
	<unknown file=""> [kripke]:0</unknown>	17234	1.82528	2e+08	1.825282e+08
	Kernel_3d_DGZ::scattering	60	7.66993	6e+07	7.896253e+07
	Kernel_3d_DGZ::LTimes	30	5.01043	9e+07	5.240528e+07
	Kernel_3d_DGZ::LPlusTimes	115	4.94770	7e+07	5.104498e+07
	Kernel_3d_DGZ::sweep	981	5.01886	2e+06	5.018862e+06
	memset.S:99	3773	3.16898	2e+ 06	3.168982e+06
	memset.S:101	3970	2.12089	5e+06	2.120895e+06
	Grid_Data::particleEdit	1201	1.13126	6e+06	1.249157e+06
	<unknown file=""> [libpsm2.so.2.1]:0</unknown>	324763	9.73341	5e+05	9.733415e+05
	memset.S:98	3767	6.19777	e+05	6.197776e+05

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)





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)





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)





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)

Cal

Calc

CalcP



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
cForceForNodes	CalcForceForNodes	4.0	1.033639e+06	2.369361e+07	2.142526
CalcQForElems	CalcQForElems	16.0	3.351894e+06	6.649351e+06	2.037651
EnergyForElems	CalcEnergyForElems	22.0	1.571996e+06	2.807323e+06	2.013174
ressureForElems	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()
f_{gf3} = gf2 - gf1
```

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

node

TimeIncremen

CalcQForElems

CalcHourglassControlForElems

LagrangeNoda

CalcForceForNodes



•	name	nid	time	time (inc)
t	TimeIncrement	25.0	8.505048e+06	8.505048e+06
S	CalcQForElems	16.0	4.455672e+06	5.189453e+06
5	CalcHourglassControlForElems	7.0	3.888798e+06	4.755817e+06
l	LagrangeNodal	3.0	1.986046e+06	8.828475e+06
S	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	time (inc)	name	nid	time
-	node					
	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_lsend	MPI_Isend	1.753768e+04	MPI_lsend	15	1.753768e+04
)	MPI_lsend	MPI_Isend	7.718737e+03	MPI_lsend	13	7.718737e+03
	MPI_lsend	MPI_Isend	7.542969e+03	MPI_lsend	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 graf-





Visualizing small graf-



Performance improvements to Hatchet

HPCToolkit read()

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: <u>https://github.com/</u> 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

