Level 2 Timing

Mike Smyth

L2 Algorithm Meeting

February 14 - 15, 2012

Copyright 2012 California Institute of Technology.
Government sponsorship acknowledged.

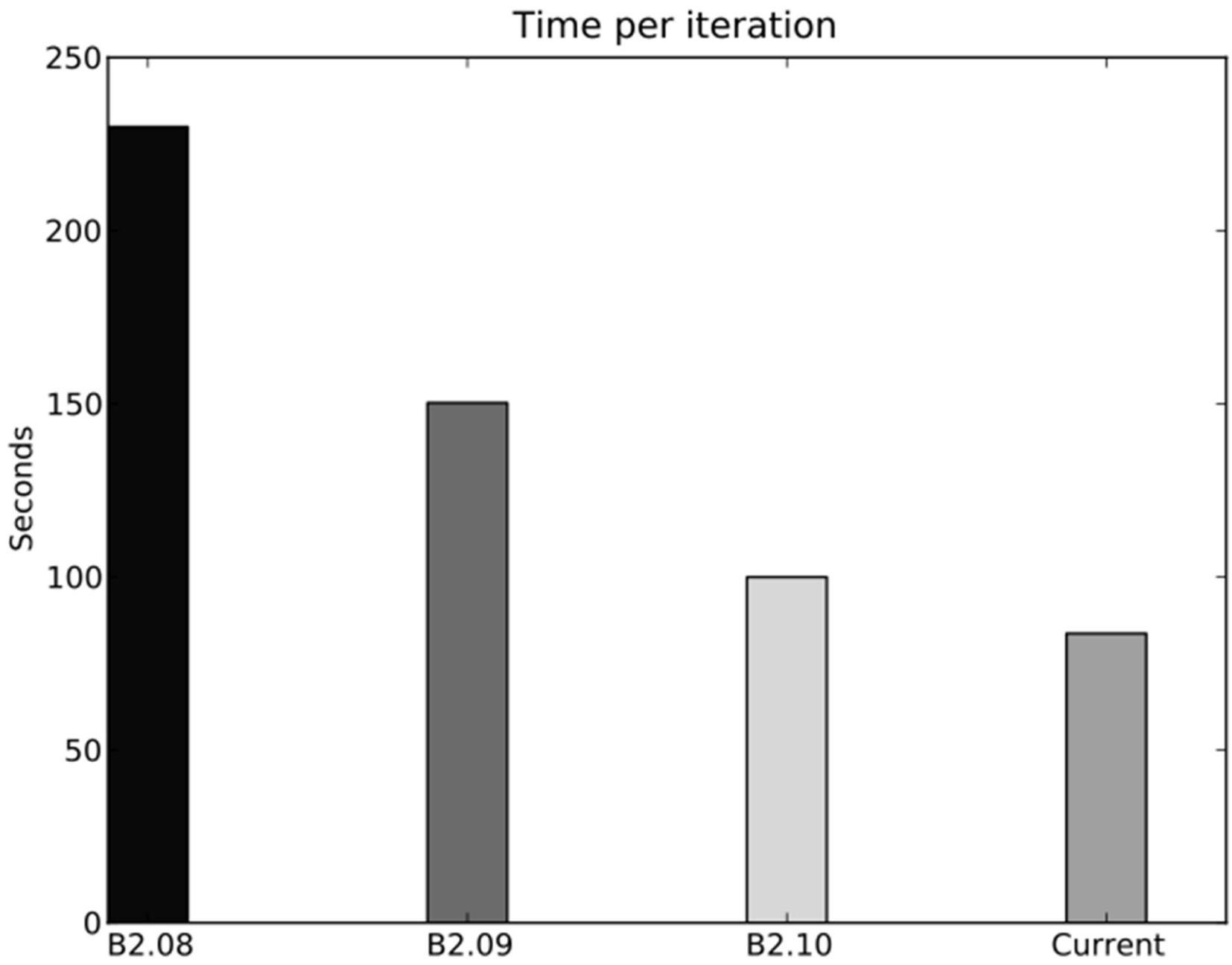
The research was carried out at the
Jet Propulsion Laboratory, California Institute of Technology,
under a contract with the National Aeronautics and
Space Administration.

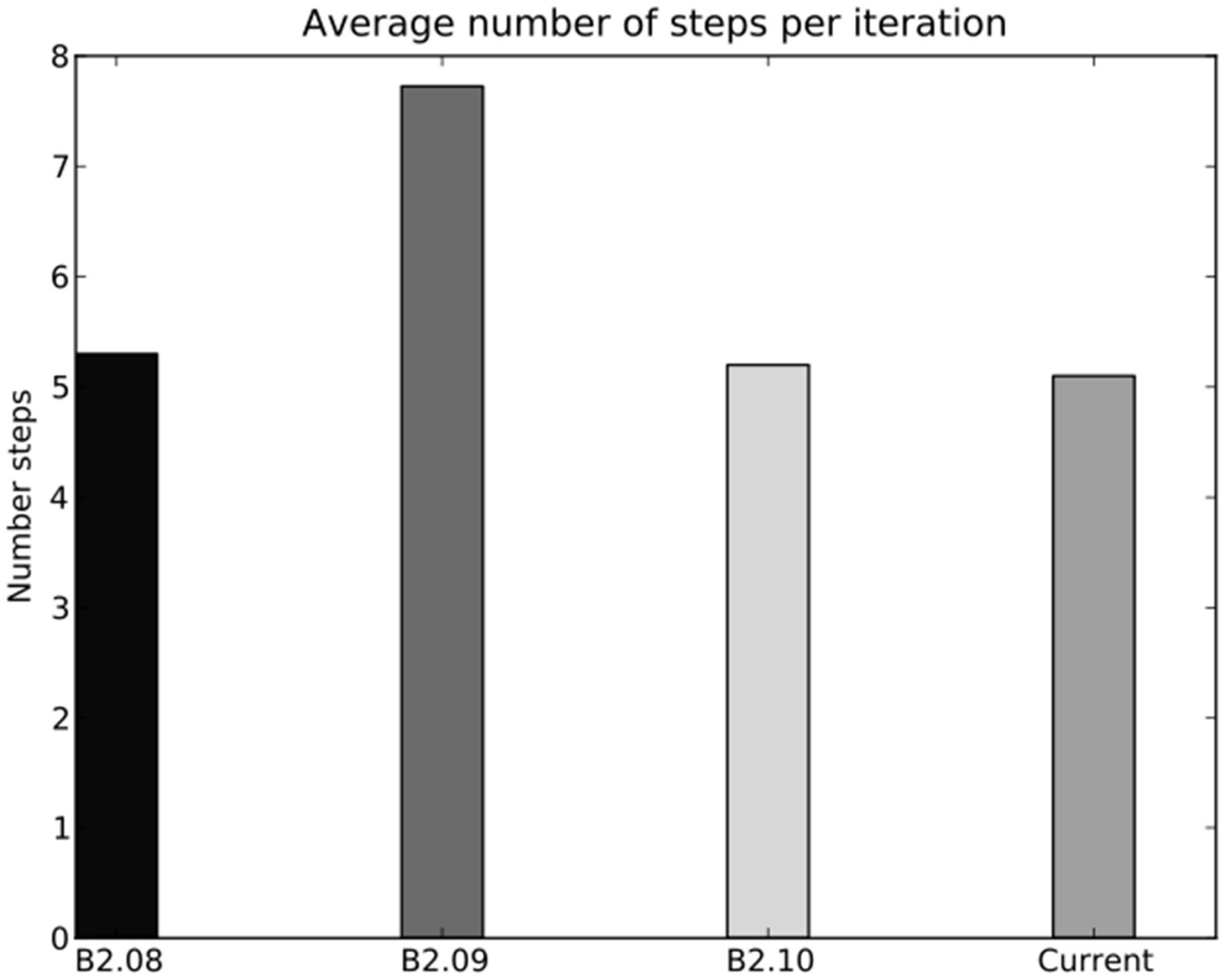
Timing per iteration

- Timings between version change for two reason
  - Number of iterations per retrieval changes
  - Time per iteration changes
- Both are important, but useful to separate
  - Time per iteration driven more by software, so more direct measure of software speed up
  - Number iterations driven by other things
    - Tolerances for successful fit
    - How good the initial guess is

Timings

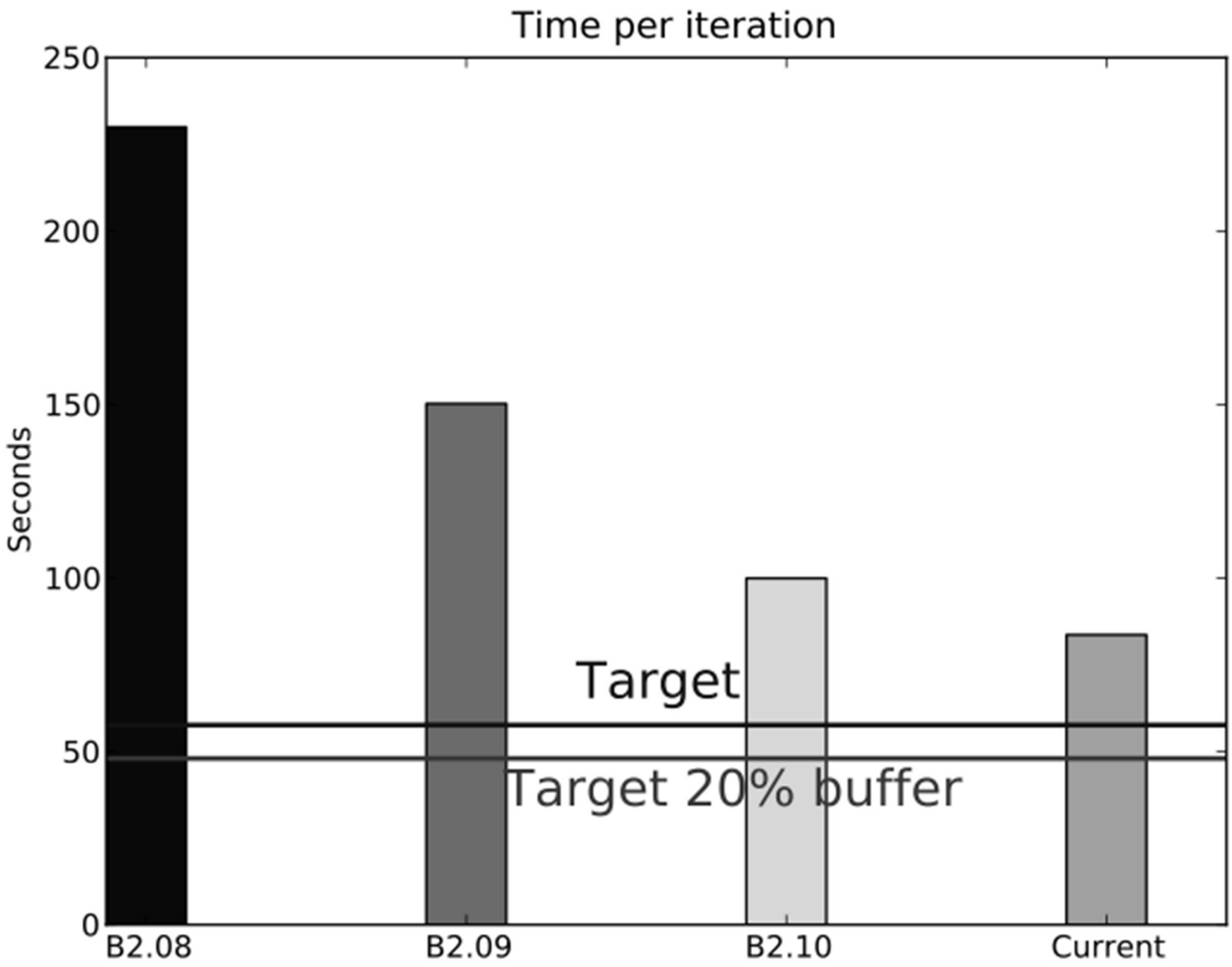
- Started measuring time per iteration in B2.08, so don't have timings from before this
- These give a "best case". Lan typically sees about a 20% penalty for her real world production runs (other things running, IO contention, etc.)
- Dug up older numbers, can't directly compare, but fun to look at
  - Time per sounding B2.06.01 - 174 minutes
  - Time per sounding B2.06.02 (switch to LIDORT, ifort) - 72 minutes
  - B2.07 – 16 minutes


Time per iteration
Seconds
250
200
150
100
50
0
B2.08
B2.09
B2.10
Current


Average number of steps per iteration
Number steps
0
1
2
3
4
5
6
7
8
B2.08
B2.09
B2.10
Current

Timing Goal

- Code is already fast enough for GOSAT – no processing issues
- Requirement for OCO is 5 minutes/sounding (latest number from Brian Chafin)
- GOSAT and OCO timings aren't identical, but should be close
  - Almost the same high resolution spectral range
  - High res calculations dominate, so time per iteration should be almost the same
  - Not as sure that the number of iterations will be same, but that is our best guess
  - 5 minute per GOSAT sounding close to meeting OCO requirement
  - Want to do timings with real OCO data when we can

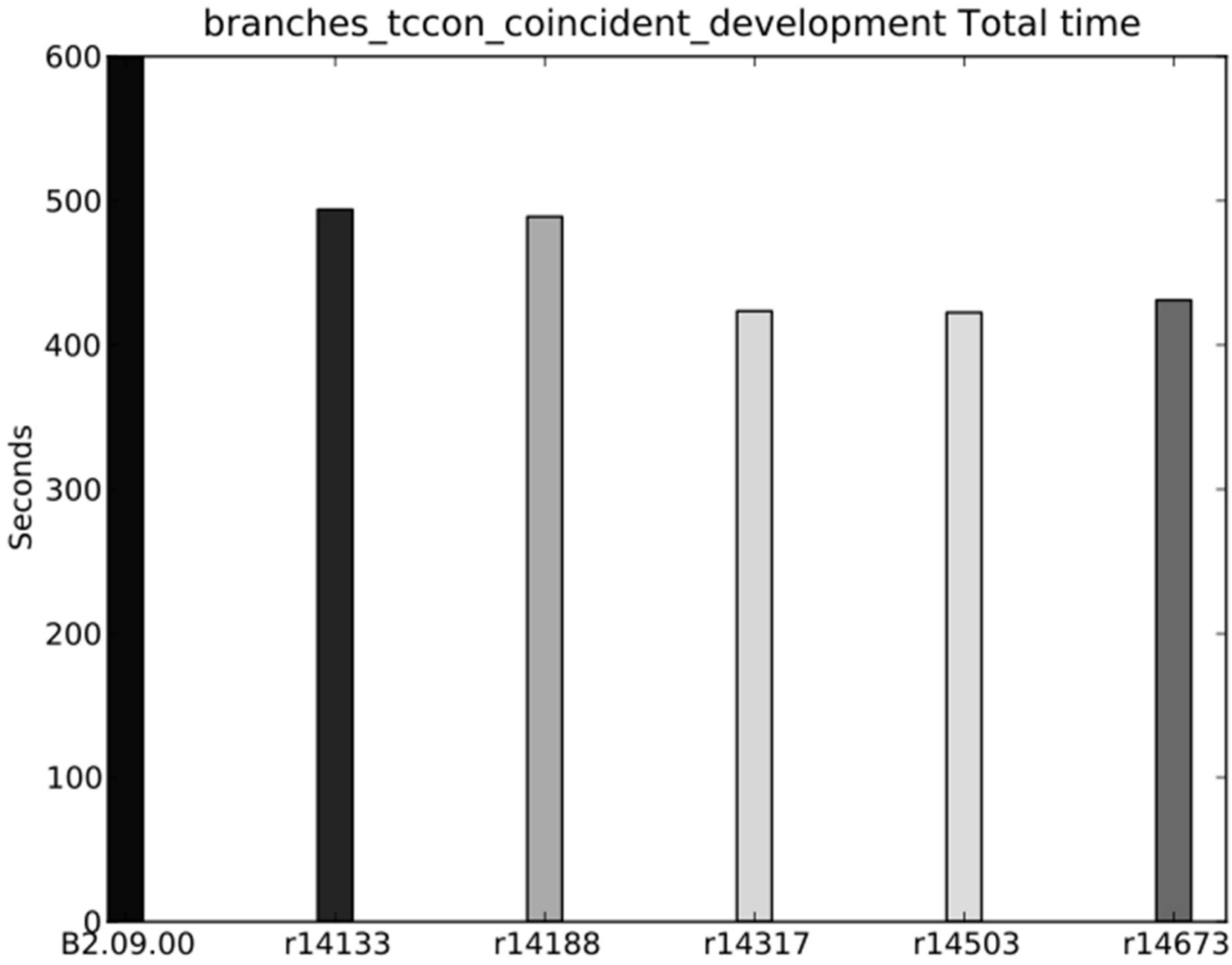

Time per iteration
Seconds
250
200
150
100
50
0
Target
Target 20% buffer
B2.08
B2.09
B2.10
Current

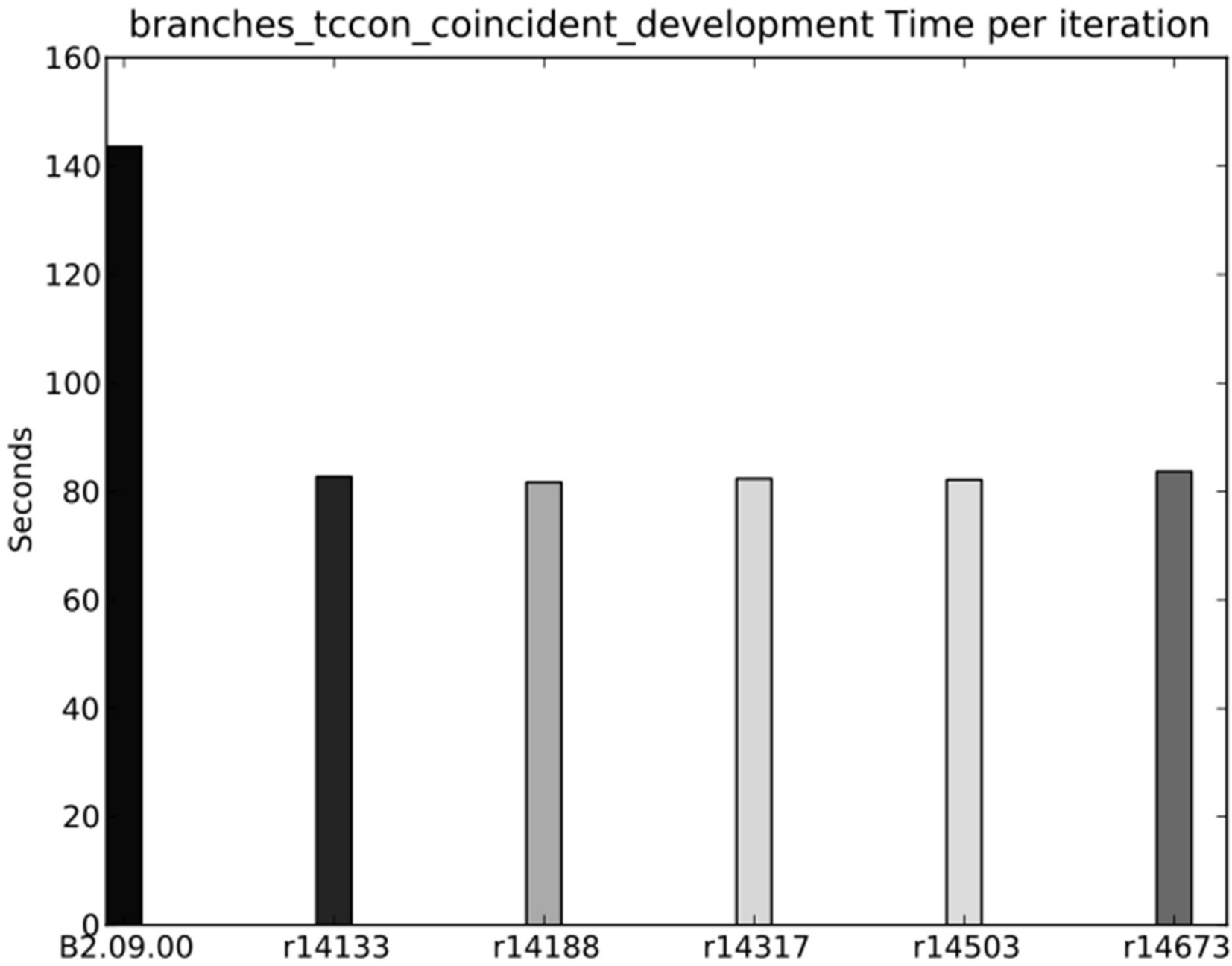
Review Regression Tests

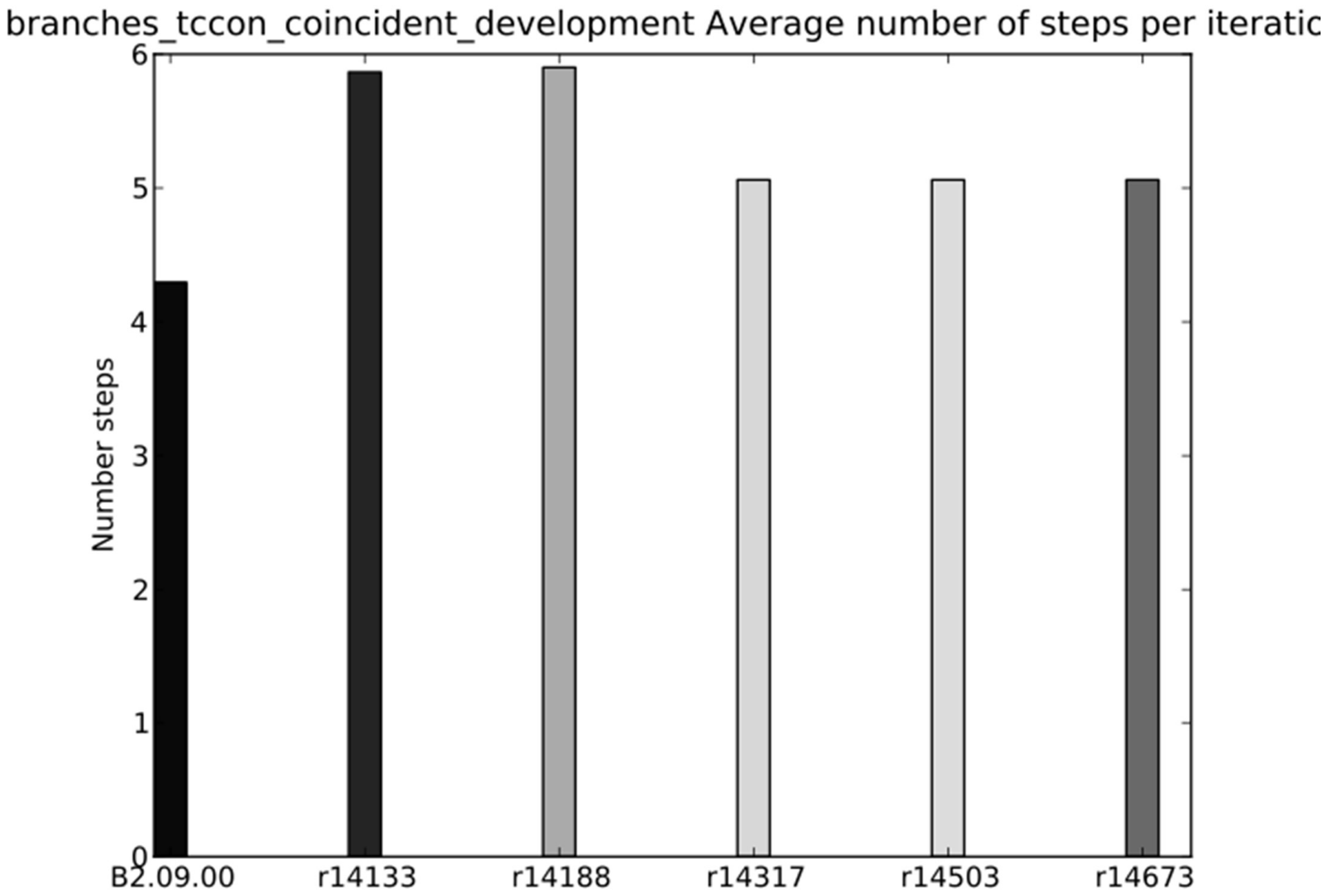
- Last Algorithm meeting, James introduced buildbot
  - Regularly, automatically run various tests
  - "Regression tests" run once a week, or on demand for important changes
  - 425 sounding coincident with TCCON, plus some orbit simulator data
  - Generates automatic plots for monitoring changes in software that affect XCO_2 and other parameters

New timing test

- Added two new tests
  - Timing tests based on regression tests
  - Detailed timing for a single run (valgrind/callgrind)
- Regression tests tend to run a bit faster than average production tests
  - About 5 – 10% faster
  - Still good test, particularly for comparison with changes
  - Run once a week, or on demand. Part of normal regression tests


branches_tccon_coincident_development Total time
Seconds
600
500
400
300
200
100
0
B2.09.00
r14133
r14188
r14317
r14503
r14673


branches_tccon_coincident_development Time per iteration
Seconds
160
140
120
100
80
60
40
20
0
B2.09.00
r14133
r14188
r14317
r14503
r14673


branches_tccon_coincident_development Average number of steps per iteratic
Number steps
0
1
2
3
4
5
6
B2.09.00
r14133
r14188
r14317
r14503
r14673

Callgrind

- Instruments code, track every function call
  - Very detailed timing
  - Runs much slower, takes about 4 hours per sounding
  - Run on a single TCCON sounding for timing
  - Produce file that is then analyzed by the GUI tool kcachgrind

File View Go Settings Help

Instruction Fetch

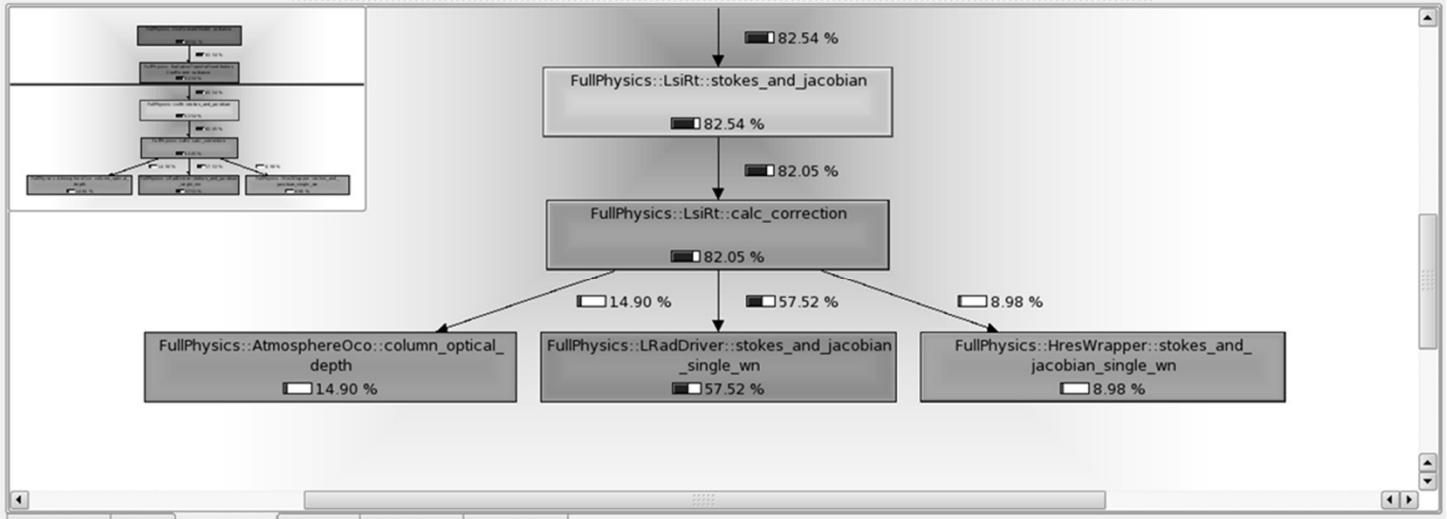
Search: (No Grouping)

Incl.	Self	Called	Function
100.00	0.00	(0)	FullPhysics::ConnorSolve
100.00	0.00	(0)	main
100.00	0.00	(0)	(below main)
100.00	0.00	(0)	0x000000372e000a70
100.00	0.00	(0)	0x00000000004dd2e0
99.94	0.00	1	FullPhysics::ForwardMod
99.89	0.00	1	FullPhysics::ForwardMod
99.86	0.00	2	FullPhysics::OcoForward
82.57	0.03	2	FullPhysics::RadiativeTra
82.54	0.02	2	FullPhysics::LsiRt::stokes
82.05	0.02	2	FullPhysics::LsiRt::calc_c
66.50	0.03	12 853	FullPhysics::LRadDriver:
49.68	0.62	12 852	FullPhysics::LidortDriver:
45.82	0.00	12 852	FullPhysics::LidortDriver:
45.82	0.00	12 852	FullPhysics::Lidort_Lps_M
45.82	0.00	12 852	lps_master_lps_master_w
45.81	0.02	12 852	lidort_lps_master_module
42.83	0.27	25 785	lidort_lps_master_module
25.59	3.44	489 915	lidort_lp_bvproblem_mp_
17.18	0.00	1	FullPhysics::IlsInstrument
17.16	0.02	1	FullPhysics::IlsConvolutic
16.24	1.43	528 552	lidort_aux_mp_dgbtrs_
16.10	1.84	12 853	FullPhysics::LRadDriver:
15.36	0.00	140 840	FullPhysics::Atmosphere
14.90	0.00	25 606	FullPhysics::Atmosphere
14.23	0.00	12 812	FullPhysics::Atmosphere
12.46	0.00	12 812	FullPhysics::AbsorberAbs
12.45	3.77	12 812	FullPhysics::AbsorberAbs
10.53	10.26	2 978 127	lidort_aux_mp_dtbsv_
9.61	1.55	489 915	lidort_lp_wfatmos_mp_up
8.98	0.00	25	FullPhysics::HresWrappe
8.11	0.00	25 496	FullPhysics::Absco::absc
8.11	3.35	25 496	FullPhysics::Absco::absc
8.06	8.06	9 308 385	lidort_lp_wfatmos_mp_lp_
6.84	0.65	12 853	l_rad_first_driver2
6.28	1.51	979	FullPhysics::IlsTable::ils
6.16	5.41	489 915	lidort_lp_bvproblem_mp_
6.05	0.10	12 853	l_rad_first_enh_m_mp_l_r
5.97	2.73	50 148 078	void blitz::Array<double,
5.90	1.75	12 853	l_rad_first_enh_m_mp_l_s
5.21	2.62	1 959	FullPhysics::IlsConvolutic
5.12	5.12	979	blitz::Array<double, 2>&
4.58	4.58	21 895 607	lidort_aux_mp_dger_

FullPhysics::LsiRt::stokes_and_jacobian

Types | Callers | All Callers | Source | Callee Map

#	Ir	Source ('/home/smyth/Level2/lib/Implementation/lsi_rt.cc')
274		{
275		#ifndef BZ_ARRAY_STACK_TRAVERSAL_UNROLL
276	0.00	for (int i=0; i < ubound; ++i)
277		T_update::update(*data++, expr.fastRead(i));
278		#else
279		int n1 = ubound & 3;
...		...
351		#endif
352		}
353		else {
354	0.00	while (iter.data() != last)
355		{
356		T_update::update(*const_cast<T_numtype*>(iter.data()), *expr);
357		iter.advance();
0		--- Inlined from '/groups/algorithm/commonlib/3.01.00a/ifort/include/blitz/ops.h' ---
137		};
138		
139		BZ_DEFINE_BINARY_OP(Add,+)
140	0.00	BZ_DEFINE_BINARY_OP(Subtract,-)
141	0.00	BZ_DEFINE_BINARY_OP(Multiply,*)



Caller Map | Parts | Call Graph | Callees | All Callees | Assembler

Detailed Timing

- **For recent development version (post B2.10, r14926)**

Apply ILS	**13.97%**
Set up Atmosphere Layers	14.41%
RT (LSI, LRAD, LIDORT)	70.06%
High stream	9.52%
Low stream	60.54%
LRAD (both high and low stream)	17.14%
LIDORT (both high and low stream)	52.36%

Effect of different speed up

- **Can see where we would get the most bang for our buck**

½ time of low res LIDORT (e.g., use faster 2 streams)	**21% speed up overall**
½ number of low res points we retrieve (e.g., really good nonuniform sampling)	37% speed up
½ high res LIDORT (some clever trick to do this)	4.76%
½ ILS Calculation	6.98%
Reduce # iterations from 5 to 4	20%

Speed up needed

- To reach 5 minutes, need
  - Speed up of 30% for "best case"
  - Speed up of 43% to allow 20% penalty for "real world"

Planned Speed Ups

- Use dedicated 2 stream version of LIDORT
  - Don't have good timing numbers yet, we'll get this once included
  - Initial rough timings show something like a 30-40% speed up with respect to LIDORT
  - Gives roughly 10 – 15% speed up
  - We'll get better numbers once implemented

Planned Speedup

- Nonuniform sampling
  - Already done for A-Band
  - Plan to add for WC-Band and SC-Band