Optimizing Perl programs

Alex Burzyński

Thames Valley Perl Mongers

20th March 2012

Optimizing is fun:

Optimizing is fun:

- Feels like beating world records

Optimizing is fun:

- Feels like beating world records
- It's relatively easy

Optimizing is fun:

- Feels like beating world records
- It's relatively easy
  - tons of *“fast templating systems”*

Optimizing is fun:

- Feels like beating world records
- It's relatively easy
  - tons of *“fast templating systems”*
- Refactoring for *“experts”*

Important part is to get maximum results with minimum effort

Where to start?

➔ Get your app working first...

Where to start?

- Get your app working first...
  - unit tests (`Devel::Cover`)

Coverage Summary

Database:	/var/log/nagios/opsview-trunk/opsview-core/cover_db
Report Date:	2013-03-20 17:18:52
Perl Version:	v5.10.1
OS:	linux
Thresholds:	< 75% < 90% < 100% = 100%

file	stmt	bran	cond	sub	pod	time	total
lib/ClassDBIExtras.pm	32.4	3.6	40.0	43.5	n/a	1.2	29.3
lib/Opsview.pm	53.3	0.0	0.0	57.9	88.9	0.1	51.0
lib/Opsview/Agent.pm	60.0	n/a	n/a	66.7	100.0	0.0	64.3
lib/Opsview/AgentPlugin.pm	66.7	n/a	n/a	66.7	0.0	0.0	61.5
lib/Opsview/Auditlog.pm	54.5	0.0	n/a	50.0	50.0	0.4	47.4
lib/Opsview/Base.pm	42.9	0.0	0.0	50.0	0.0	0.0	32.0
lib/Opsview/Base/Contact.pm	11.4	0.0	n/a	27.3	87.5	0.0	15.3
lib/Opsview/Base/Icon.pm	42.9	n/a	n/a	33.3	0.0	0.0	33.3
lib/Opsview/Checktype.pm	85.7	n/a	n/a	66.7	100.0	0.0	81.8
lib/Opsview/Config.pm	25.7	0.0	0.0	21.4	1.3	0.1	15.3
lib/Opsview/Config/Web.pm	48.0	0.0	n/a	80.0	0.0	0.0	45.7
lib/Opsview/Connections.pm	40.9	0.0	0.0	66.7	n/a	0.0	38.1
lib/Opsview/DBIx/Class.pm	11.9	0.0	0.0	36.8	0.0	0.1	9.7
lib/Opsview/DBIx/Class/Common.pm	45.0	0.0	0.0	60.0	0.0	0.0	33.3
lib/Opsview/Exceptions.pm	100.0	n/a	n/a	100.0	n/a	0.1	100.0
lib/Opsview/Externalcommand.pm	7.7	0.0	0.0	13.3	66.7	0.0	10.4

File Coverage

File:	lib/Opsview/Statistics.pm
Coverage:	66.7%

line	stmt	bran	cond	sub	pod	time	code
1							package Opsview::Statistics;
2							
3	1 1 1			1		21788 1 25	use strict;
4	1 1 1			1		3 1 25	use warnings;
5	1 1 1			1		15744 76608 15	use Moose;
6							
7							has schema => (
8							is => 'rw',
9							isa => 'DBIx::Class::Schema'
10							);
11							
12							sub host_count {
13	1			1	0	20	(shift)->schema->resultset('Hosts')->count;
14							}

Where to start?

- Get your app working first...
  - unit tests (`Devel::Cover`)
- Gather lots of real-life input data

Where to start?

- Get your app working first...
  - unit tests (`Devel::Cover`)
- Gather lots of real-life input data
  - avoid optimizing for rare events

Where to start?

- Get your app working first...
  - unit tests (`Devel::Cover`)
- Gather lots of real-life input data
  - avoid optimizing for rare events
- **Profile!**

Profiling

- Development – `Devel::NYTProf`

Profiling

- Development – `Devel::NYTProf`
  - Line, subroutine and block profiles

Filename	/var/log/nagios/opsview-trunk/opsview-core/t/../lib/Opsview/Statistics.pm
Statements	Executed 13 statements in 708µs

Subroutines

Calls	P	F	Exclusive Time	Inclusive Time	Subroutine
1	1	1	1.10ms	21.7ms	Opsview::Statistics::BEGIN@5
1	1	1	45µs	72.5ms	Opsview::Statistics::host_count
1	1	1	30µs	36µs	Opsview::Statistics::BEGIN@3
1	1	1	17µs	17µs	Opsview::Statistics::schema (xsub)
1	1	1	12µs	32µs	Opsview::Statistics::BEGIN@4
0	0	0	0s	0s	Opsview::Statistics::monitoringclusternodes_count
0	0	0	0s	0s	Opsview::Statistics::monitoringservers_count

Call graph for these subroutines as a Graphviz dot language file.

Line	Statements	Time on line	Calls	Time in subs	Code
1					`package Opsview::Statistics;`
2					

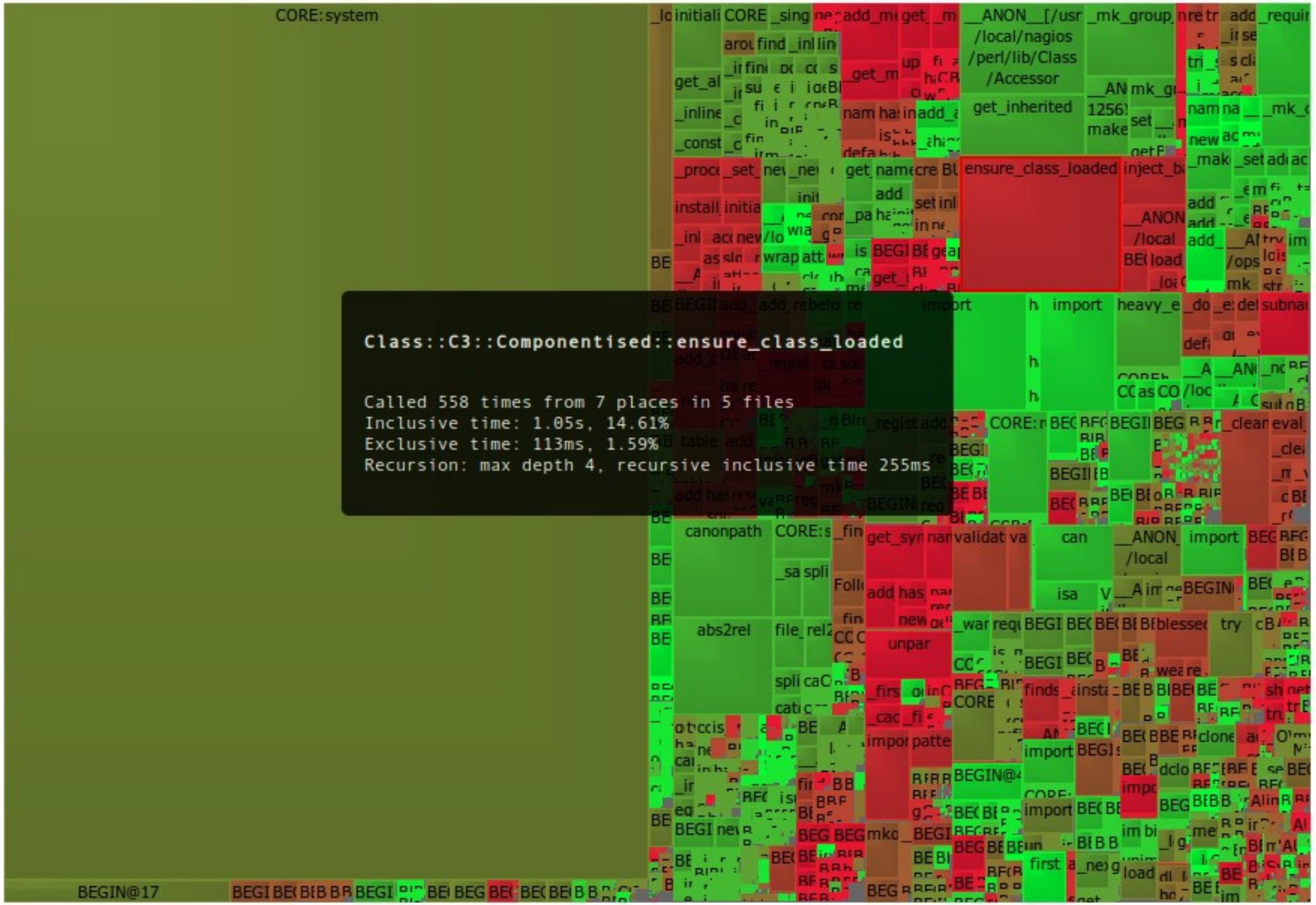
3	3	43µs	2	41µs	`# spent 36µs (30+6) within Opsview::Statistics::BEGIN@3 which was called:`
					`#  once (30µs+6µs) by Test::Opsview::Statistics::BEGIN@2.69 at line 3`
					`use strict;`
					`# spent  36µs making 1 call to Opsview::Statistics::BEGIN@3`
					`# spent  6µs making 1 call to strict::import`

4	3	35µs	2	52µs	`# spent 32µs (12+20) within Opsview::Statistics::BEGIN@4 which was called:`
					`#  once (12µs+20µs) by Test::Opsview::Statistics::BEGIN@2.69 at line 4`
					`use warnings;`
					`# spent  32µs making 1 call to Opsview::Statistics::BEGIN@4`
					`# spent  20µs making 1 call to warnings::import`

5	3	520µs	2	22.5ms	`# spent 21.7ms (1.10+20.6) within Opsview::Statistics::BEGIN@5 which was called:` `#  once (1.10ms+20.6ms) by Test::Opsview::Statistics::BEGIN@2.69 at line 5`

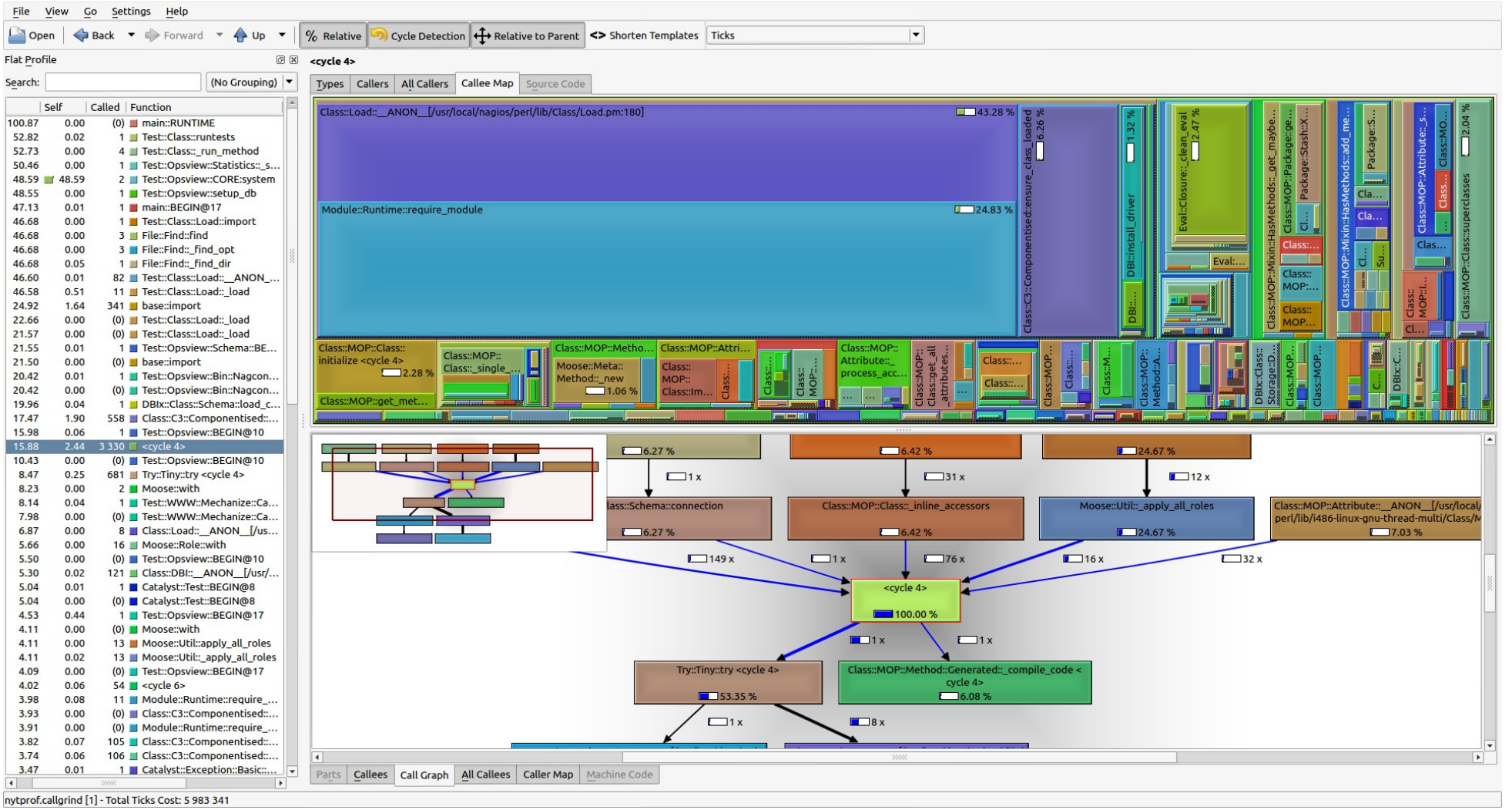
Profiling

- Development – `Devel::NYTProf`
  - Line, subroutine and block profiles
  - HTML reports (treemaps, links to source code)



Profiling

- Development – `Devel::NYTProf`
  - Line, subroutine and block profiles
  - HTML reports (treemaps, links to source code)
  - Call-graphs via Kcachegrind (relationships between subroutines)


File View Go Settings Help
Open Back Forward Up % Relative Cycle Detection Relative to Parent <> Shorten Templates Ticks
Flat Profile
Search: (No Grouping)
Self Called Function
100.87 0.00 (0) main::RUNTIME
52.82 0.02 1 Test::Class::runtests
52.73 0.00 4 Test::Class::_run_method
50.46 0.00 1 Test::Opsview::Statistics::_s...
48.59 48.59 2 Test::Opsview::CORE:system
48.55 0.00 1 Test::Opsview::setup_db
47.13 0.01 1 main::BEGIN@17
46.68 0.00 1 Test::Class::Load::import
46.68 0.00 3 File::Find::find
46.68 0.00 3 File::Find::_find_opt
46.68 0.05 1 File::Find::_find_dir
46.60 0.01 82 Test::Class::Load::__ANON_...
46.58 0.51 11 Test::Class::Load::_load
24.92 1.64 341 base::import
22.66 0.00 (0) Test::Class::Load::_load
21.57 0.00 (0) Test::Class::Load::_load
21.55 0.01 1 Test::Opsview::Schema::BE...
21.50 0.00 (0) base::import
20.42 0.01 1 Test::Opsview::Bin::Nagcon...
20.42 0.00 (0) Test::Opsview::Bin::Nagcon...
19.96 0.04 1 DBIx::Class::Schema::load_c...
17.47 1.90 558 Class::C3::Componentised::...
15.98 0.06 1 Test::Opsview::BEGIN@10
15.88 2.44 3 330 <cycle 4>
10.43 0.00 (0) Test::Opsview::BEGIN@10
8.47 0.25 681 Try::Tiny::try <cycle 4>
8.23 0.00 2 Moose::with
8.14 0.04 1 Test::WWW::Mechanize::Ca...
7.98 0.00 (0) Test::WWW::Mechanize::Ca...
6.87 0.00 8 Class::Load::__ANON__[/us...
5.66 0.00 16 Moose::Role::with
5.50 0.00 (0) Test::Opsview::BEGIN@10
5.30 0.02 121 Class::DBI::__ANON__[/usr/...
5.04 0.01 1 Catalyst::Test::BEGIN@8
5.04 0.00 (0) Catalyst::Test::BEGIN@8
4.53 0.44 1 Test::Opsview::BEGIN@17
4.11 0.00 (0) Moose::with
4.11 0.00 13 Moose::Util::apply_all_roles
4.11 0.02 13 Moose::Util::_apply_all_roles
4.09 0.00 (0) Test::Opsview::BEGIN@17
4.02 0.06 54 <cycle 6>
3.98 0.08 11 Module::Runtime::require_...
3.93 0.00 (0) Class::C3::Componentised::...
3.91 0.00 (0) Module::Runtime::require_...
3.82 0.07 105 Class::C3::Componentised::...
3.74 0.06 106 Class::C3::Componentised::...
3.47 0.01 1 Catalyst::Exception::Basic::...
<cycle 4>
Types Callers All Callers Callee Map Source Code
Class::Load::__ANON__[/usr/local/nagios/perl/lib/Class/Load.pm:180] 43.28 %
Module::Runtime::require_module 24.83 %
Class::C3::Componentised::ensure_class_loaded 6.26 %
DBI::install_driver 1.32 %
Eval::Closure::_clean_eval 2.47 %
Class::MOP::Class::initialize <cycle 4> 2.28 %
Class::MOP::Class::get_met...
Class::MOP::Class::_single_...
Moose::Meta::Method::_new 1.06 %
Class::MOP::Class::Im...
Class::MOP::Attribute::_process_acc...
Class::MOP::Method::A...
Class::MOP::Mixin::HasMethods::add_me...
Class::MOP::Attribute::_s...
Class::MOP::Class::superclasses 2.04 %
6.27 %
6.42 %
24.67 %
1 x
31 x
12 x
Class::MOP::Class::_inline_accessors
6.42 %
Moose::Util::_apply_all_roles
24.67 %
7.03 %
149 x
1 x
76 x
16 x
32 x
<cycle 4>
100.00 %
Try::Tiny::try <cycle 4>
53.35 %
Class::MOP::Method::Generated::_compile_code < cycle 4>
6.08 %
8 x
Parts Callees Call Graph All Callees Caller Map Machine Code
nytprof.callgrind [1] - Total Ticks Cost: 5 983 341

Profiling

- Production – `DashProfiler`

Profiling

- Production – `DashProfiler`
  - Continuous monitoring

Profiling

- Production – `DashProfiler`
  - Continuous monitoring
  - Flexible configuration

Profiling

- Production – `DashProfiler`
  - Continuous monitoring
  - Flexible configuration

Profiling

- Production – `DashProfiler`
  - Continuous monitoring
  - Flexible configuration
  - Minimal code changes

```
Nagios config re-generated in 5.096 seconds

auto > nagconfgen.pl > ms-Cluster: dur=1.088865 count=1 (max=1.088865 avg=1.088865)
auto > nagconfgen.pl > ms-ClusterA: dur=0.972281 count=1 (max=0.972281 avg=0.972281)
auto > nagconfgen.pl > ms-Master Monitoring Server: dur=2.197775 count=1 (max=2.197775 avg=2.197775)
auto > nagconfgen.pl > ms-PassiveSlave: dur=0.655362 count=1 (max=0.655362 avg=0.655362)
auto > other > other: dur=0.412792 count=1 (max=0.412792 avg=0.412792)
nagios@ov-dev-alex:~/opsview-trunk/opsview-core$ vc
Index: bin-protected/nagconfgen.pl
===================================================================
--- nagconfgen.pl (revision 11840)
+++ nagconfgen.pl (working copy)
@@ -37,6 +37,8 @@
 use Data::Dumper;
 use JSON;

+use DashProfiler::Auto;
+
 my $opsview4_upgrade_config_generation_lock =
   "/tmp/opsview4_upgrade_config_generation.lock";
 my $only_opsview_host = 0;
@@ -272,6 +274,7 @@

         my $ms_name = $monitoringserver->name;
         plog "--> Writing config files for $ms_name";
+        my $dp_main = auto_profiler("ms-$ms_name");

         my @all_nodes;
         if ( $monitoringserver->is_slave ) {
nagios@ov-dev-alex:~/opsview-trunk/opsview-core$
```

Optimizing

- Cache

Optimizing

- Cache
  - functions

Optimizing

- Cache
  - functions, variables

Optimizing

- Cache
  - functions, variables, DBI's *_*cached*

Optimizing

- Cache
  - functions, variables, DBI's *_*cached*
- Refactor (`B::Concise` + `B::Deparse`) and `Benchmark`

```
alex@alex-pc-ubuntu:~/opsview$ cat fib.pl
#!/usr/bin/perl

use strict;
use warnings;

sub fib {
    my $n = shift;
    return $n if $n < 2;
    fib($n-1) + fib($n-2);
}

print fib( $ARGV[0] ), "\n";


alex@alex-pc-ubuntu:~/opsview$ perl -MO=Concise fib.pl 6
d  <@> leave[1 ref] vKP/REFC ->(end)
1     <0> enter ->2
2     <;> nextstate(main 5 fib.pl:12) v:*,&,{,$ ->3
c     <@> print vK ->d
3        <0> pushmark s ->4
a        <1> entersub[t4] lKS/TARG,3 ->b
-           <1> ex-list lK ->a
4              <0> pushmark s ->5
8              <2> aelem sKM/LVDEFER,2 ->9
6                 <1> rv2av sKR/3 ->7
5                    <#> gv[*ARGV] s ->6
7                 <$> const[IV 0] s ->8
-              <1> ex-rv2cv sK/2 ->-
9                 <#> gv[*fib] s ->a
b        <$> const[PV "\n"] s ->c
fib.pl syntax OK
```

Optimizing

- Cache
  - functions, variables, DBI's *_cached*
- Refactor (`B::Concise` + `B::Deparse`) and `Benchmark`
- Parallel execution

Optimizing

- Cache
  - functions, variables, DBI's *_*_cached_*
- Refactor (`B::Concise` + `B::Deparse`) and `Benchmark`
- Parallel execution
- Event-based programming

Optimizing

- Cache
  - functions, variables, DBI's *_*cached*
- Refactor (`B::Concise` + `B::Deparse`) and `Benchmark`
- Parallel execution
- Event-based programming
- Rewrite in C – Perl XS

Questions?