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

<table>
<thead>
<tr>
<th>Database:</th>
<th>/var/log/nagios/opsview-trunk/opsview-core/cover_db</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report Date:</td>
<td>2013-03-20 17:18:52</td>
</tr>
<tr>
<td>Perl Version:</td>
<td>v5.10.1</td>
</tr>
<tr>
<td>OS:</td>
<td>linux</td>
</tr>
</tbody>
</table>

### Thresholds

- < 75%
- < 90%
- < 100%
- = 100%

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

<table>
<thead>
<tr>
<th>line</th>
<th>stmt</th>
<th>bran</th>
<th>cond</th>
<th>sub</th>
<th>pod</th>
<th>time</th>
<th>code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>package Opsview::Statistics;</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td>21788</td>
<td>use strict;</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td>3</td>
<td>use warnings;</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td>15744</td>
<td>use Moose;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>76608</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>has schema =&gt; (</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>is =&gt; 'rw',</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>isa =&gt; 'DBIx::Class::Schema'</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>);</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td>20</td>
<td>sub host_count {</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(shift)-&gt;schema-&gt;resultset('Hosts')-&gt;count;</td>
</tr>
</tbody>
</table>
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
Executed 13 statements in 708μs

### Subroutines

<table>
<thead>
<tr>
<th>Calls</th>
<th>P</th>
<th>F</th>
<th>Exclusive Time</th>
<th>Inclusive Time</th>
<th>Subroutine</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1.10ms</td>
<td>21.7ms</td>
<td>Opsview::Statistics::BEGIN05</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>45μs</td>
<td>72.5ms</td>
<td>Opsview::Statistics::host_count</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>30μs</td>
<td>36μs</td>
<td>Opsview::Statistics::BEGIN03</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>17μs</td>
<td>17μs</td>
<td>Opsview::Statistics::schema (xsub)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>12μs</td>
<td>32μs</td>
<td>Opsview::Statistics::BEGIN04</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0s</td>
<td>0s</td>
<td>Opsview::Statistics::monitoringclusternodes_count</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0s</td>
<td>0s</td>
<td>Opsview::Statistics::monitoringservers_count</td>
</tr>
</tbody>
</table>

Call graph for these subroutines as a [Graphviz dot language file](https://graphviz.org/).

### Code

1
2
3

<table>
<thead>
<tr>
<th>Line</th>
<th>Statement</th>
<th>Time on line</th>
<th>Calls</th>
<th>Time in subs</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>43μs</td>
<td>2</td>
<td>41μs</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>35μs</td>
<td>2</td>
<td>52μs</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>520μs</td>
<td>2</td>
<td>22.5ms</td>
<td></td>
</tr>
</tbody>
</table>

# spent 1.10ms (1.10+20.6) within Opsview::Statistics::BEGIN05 which was called:
# once (1.10ms+20.6ms) by Test::Opsview::Statistics::BEGIN@2.69 at line 5
use warnings;
# 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)
Class::C3::Componentised::ensure_class_loaded

Called 558 times from 7 places in 5 files
Inclusive time: 1.05s, 14.61%
Exclusive time: 113ms, 1.59%
Recursion: max depth 4, recursive inclusive time 255ms
Profiling

¬ Development – Devel::NYTProf

• Line, subroutine and block profiles
• HTML reports (treemaps, links to source code)
• Call-graphs via KcacheGrind (relationships between subroutines)
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
#!/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

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?