Benchmarking Perl

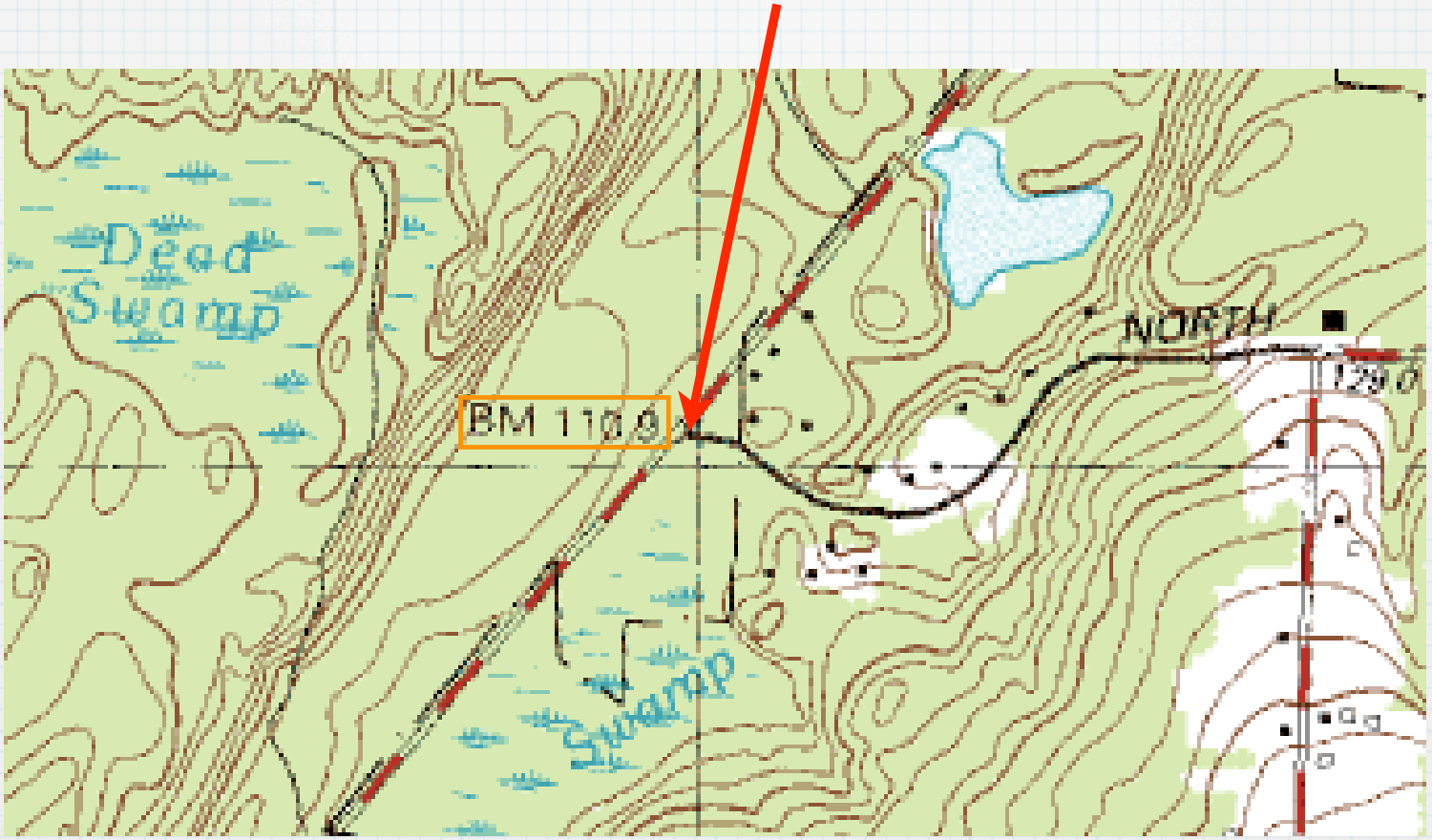
brian d foy
Stonehenge Consulting Services
March 14, 2006
Chicago UniForum

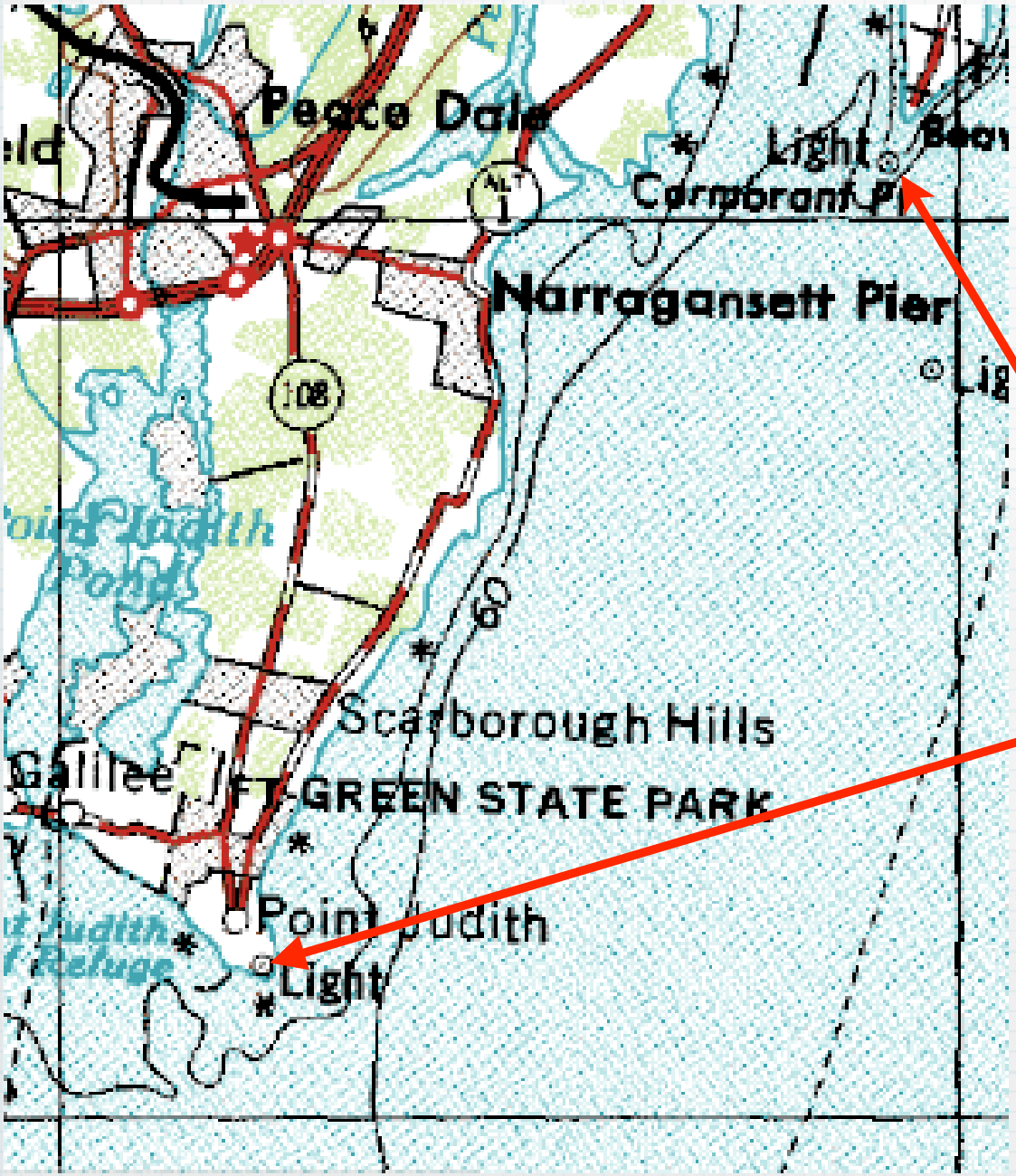
Know where you are

"A benchmark is a point of reference for a measurement. The term originates from the chiseled horizontal marks that surveyors made into which an angle-iron could be placed to bracket (bench) a leveling rod, thus ensuring that the leveling rod can be repositioned in the exact same place in the future."

http://en.wikipedia.org/wiki/Benchmark

Single Points





Multiple Points

Nobody is ferpect

* It's only relative
* It has to be the same answer next time
* If something changes, the reference is meaningless

Computer Benchmarks

- Standard Performance Evaluation Corporation (SPEC)
  - CPU (2000, 95, 92)
  - Graphics
  - Web servers

http://www.spec.org/

All Things Being Equal...

* There are lies, damned lies, and benchmarks
* Everyone has an agenda
* You don't run testbeds as production
* Skepticism wins the day

Language Shootout

Benchmarking programming languages?

How can we benchmark a programming language?
We can't - we benchmark programming language implementations.

How can we benchmark language implementations?
We can't - we measure particular programs.

http://shootout.alioth.debian.org/

Performance

A major factor in determining the overall productivity of a system, performance is primarily tied to availability, throughput and response time.

http://www.comptia.org/sections/ssg/glossary.aspx

Performance (2)

A performance comprises an event in which generally one group of people behave in a particular way for another group of people

http://en.wikipedia.org/wiki/Performance

Performance (3)

"Your investment's activity over time. Past performance does not guarantee future results."

My accountant

How many
metrics
can you
name?

Availability Disk Use
Concurrent Users CPU Time
Completion Time Memory Use
Uptime Bandwidth Use
Network Lag Responsiveness
Binary Size

Programmer Time

Power

Speed

Ease of use

Pick Any Two

You haven't said "Perl"

* Perl is just a programming language
* It's a High Level Language
* Measure Perl the same as other things
* Measure Perl against itself
* Compare the results

TIMTOWTMI

It's not just speed

Memory Use

```
use Devel::Size qw(size total_size);

my $size = size("A string");

my @foo = (1, 2, 3, 4, 5);
my $other_size = size(\@foo);

my $foo = {
          a => [1, 2, 3],
          b => {a => [1, 3, 4]}
        };
my  $total_size = total_size($foo);
```

Memory Use (2)

```
use Devel::Peek;
Dump( $a );
Dump( $a, 5 );
DumpArray( 5, $a, $b, ... );
mstat "Point 5";
```

Need PERL_DEBUG_MSTATS

Database query time

```
$ DBI_PROFILE=2 db_script
```

```
  DBI::Profile: 0.001015s (5 calls) db_script @ YYYY-MM-DD HH:MM:SS
  '' =>
      0.000024s / 2 = 0.000012s avg (first 0.000015s, min 0.000009s,
max 0.000015s)
  'SELECT mode,size,name FROM table' =>
      0.000991s / 3 = 0.000330s avg (first 0.000678s, min 0.000009s,
max 0.000678s)
```

Benchmark.pm comes with Perl...

...and it sux...

Sux : Sucks :: Kwalitee : Quality

Common misuse

```
use Benchmark 'cmpthese';

my @long  = ('a' .. 'z', '');

my $iter = shift || -1;

cmpthese(
    $iter,{
        long_block_ne  => q{grep {$_ ne ''} @long},
        long_block_len => q{grep {length}   @long},
        long_bare_ne   => q{grep $_ ne '',  @long},
        long_bare_len  => q{grep length,    @long},
    }
);
```

http://www.perlmonks.org/index.pl?node_id=536503

What's wrong with this picture?

	Rate	bare_ne	block_len	block_ne	bare_len
long_bare_ne	3635361/s	--	-6%	-6%	-8%
long_block_len	3869054/s	6%	--	-0%	-2%
long_block_ne	3872708/s	7%	0%	--	-2%
long_bare_len	3963159/s	9%	2%	2%	--

Mac OS X.4.5
15" G4 Powerbook
perl5.8.4

	Rate	bare_len	bare_ne	block_ne	block_len
long_bare_len	2805822/s	--	-0%	-1%	-3%
long_bare_ne	2805822/s	0%	--	-1%	-3%
long_block_ne	2840569/s	1%	1%	--	-2%
long_block_len	2885232/s	3%	3%	2%	--

```
This is perl, v5.8.4 built for darwin-2level

Summary of my perl5 (revision 5 version 8 subversion 4) configuration:
  Platform:
    osname=darwin, osvers=7.3.1, archname=darwin-2level
    uname='darwin albook.local 7.3.1 darwin kernel version 7.3.1: mon mar 22
21:48:41 pst 2004; root:xnuxnu-517.4.12.obj~2release_ppc power macintosh powerpc '
    config_args=''
    hint=recommended, useposix=true, d_sigaction=define
    usethreads=undef use5005threads=undef useithreads=undef usemultiplicity=undef
    useperlio=define d_sfio=undef uselargefiles=define usesocks=undef
    use64bitint=undef use64bitall=undef uselongdouble=undef
    usemymalloc=n, bincompat5005=undef
  Compiler:
    cc='cc', ccflags ='-pipe -fno-common -DPERL_DARWIN -no-cpp-precomp -fno-strict-
aliasing',
    optimize='-Os',
    cppflags='-no-cpp-precomp -pipe -fno-common -DPERL_DARWIN -no-cpp-precomp -fno-
strict-aliasing'
    ccversion='', gccversion='3.3 20030304 (Apple Computer, Inc. build 1640)',
gccosandvers=''
    intsize=4, longsize=4, ptrsize=4, doublesize=8, byteorder=4321
    d_longlong=define, longlongsize=8, d_longdbl=define, longdblsize=8
    ivtype='long', ivsize=4, nvtype='double', nvsize=8, Off_t='off_t', lseeksize=8
    alignbytes=8, prototype=define
  Linker and Libraries:
    ld='env MACOSX_DEPLOYMENT_TARGET=10.3 cc', ldflags =''
    libpth=/usr/lib
    libs=-ldbm -ldl -lm -lc
    perllibs=-ldl -lm -lc
    libc=/usr/lib/libc.dylib, so=dylib, useshrplib=false, libperl=libperl.a
    gnulibc_version=''
  Dynamic Linking:
    dlsrc=dl_dyld.xs, dlext=bundle, d_dlsymun=undef, ccdlflags=' '
    cccdlflags=' ', lddlflags=' -bundle -undefined dynamic_lookup'
```

Millions
of times
a second?

Do something useful

```
use Benchmark 'cmpthese';

our @long  = ('a' .. 'z', '');

my $iter = shift || -1;

cmpthese(
    $iter,{
        long_block_ne  => q{my @array = grep {$_ ne ''} @long},
        long_block_len => q{my @array = grep {length}   @long},
        long_bare_ne   => q{my @array = grep $_ ne '',  @long},
        long_bare_len  => q{my @array = grep length,    @long},
    }
);
```

These numbers make sense

	Rate	block_ne	block_len	bare_ne	bare_len
long_block_ne	31210/s	--	-3%	-3%	-5%
long_block_len	32119/s	3%	--	-0%	-2%
long_bare_ne	32237/s	3%	0%	--	-2%
long_bare_len	32755/s	5%	2%	2%	--

Don't Trust!
Verify!

```
use Benchmark 'cmpthese';

our @long  = ('a' .. 'z', 0 .. 10_000, '');

my $iter = shift || -1;

cmpthese(
    $iter,{
        long_block_ne  => q{my @array = grep {$_ ne ''} @long},
        long_block_len => q{my @array = grep {length}   @long},
        long_bare_ne   => q{my @array = grep $_ ne '',  @long},
        long_bare_len  => q{my @array = grep length,    @long},
    }
);
```

It takes longer to do more

	Rate	bare_ne	block_ne	block_len	bare_len
long_bare_ne	59.8/s	--	-1%	-2%	-3%
long_block_ne	60.4/s	1%	--	-1%	-3%
long_block_len	60.9/s	2%	1%	--	-2%
long_bare_len	61.9/s	4%	3%	2%	--

Theory of Measurement

* Observation changes the universe
* Nothing is objective
* Tools have inherent uncertainities

Precision
is
repeatability

Accuracy
is
getting the
right answer

precision != accuracy

you
want
both

Benchmark.pm's benchmark

* Uses a null loop as a control
* sub { }
* It's just a timer
* Subtracts the null loop time
* Introduces an error of about 7%

```
$ time perl script.pl

real    0m0.293s
user    0m0.130s
sys     0m0.036s
```

http://ppt.perl.org/

The map is not the terrain

```
my @selected = grep { ... } @array;
          () = grep { ... } @array;
my @selected =              @array;
my @selected =            ( ... );
my @selected                      ;
```

```
$ perl -MO=Concise -e 'grep { lc } @array'

a  <@> leave[1 ref] vKP/REFC ->(end)
1     <0> enter ->2
2     <;> nextstate(main 2 -e:1) v ->3
7     <|> grepwhile(other->8)[t4] vK/1 ->a
6        <@> grepstart K*/2 ->7
3           <0> pushmark s ->4
-           <1> null lK/1 ->4
-              <1> null sK/1 ->7
-                 <@> scope sK ->7
-                    <0> ex-nextstate v ->8
9                   <1> lc[t2] sK/1 ->-
-                      <1> ex-rv2sv sK/1 ->9
8                        <$> gvsv(*_) s ->9
5           <1> rv2av[t3] lKM/1 ->6
4              <$> gv(*array) s ->5
```

```
$ perl -MO=Concise -e '() = grep { lc } @array'

d  <@> leave[1 ref] vKP/REFC ->(end)
1     <0> enter ->2
2     <;> nextstate(main 2 -e:1) v ->3
c     <2> aassign[t5] vKS/COMMON ->d
-        <1> ex-list lK ->b
3           <0> pushmark s ->4
8           <|> grepwhile(other->9)[t4] lK/1 ->b
7              <@> grepstart lK*/2 ->8
4                 <0> pushmark s ->5
-                 <1> null lK/1 ->5
-                    <1> null sK/1 ->8
-                       <@> scope sK ->8
-                          <0> ex-nextstate v ->9
a                         <1> lc[t2] sK/1 ->-
-                            <1> ex-rv2sv sK/1 ->a
9                              <$> gvsv(*_) s ->a
6                 <1> rv2av[t3] lKM/1 ->7
5                    <$> gv(*array) s ->6
-        <1> ex-list lK ->c
b           <0> pushmark s ->c
-           <0> stub lPRM* ->-
```

```
$ perl -MO=Concise -e 'my @selected = grep { lc } @array'

e  <@> leave[1 ref] vKP/REFC ->(end)
1     <0> enter ->2
2     <;> nextstate(main 2 -e:1) v ->3
d     <2> aassign[t6] vKS ->e
-        <1> ex-list lK ->b
3           <0> pushmark s ->4
8           <|> grepwhile(other->9)[t5] lK/1 ->b
7              <@> grepstart lK*/2 ->8
4                 <0> pushmark s ->5
-                 <1> null lK/1 ->5
-                    <1> null sK/1 ->8
-                       <@> scope sK ->8
-                          <0> ex-nextstate v ->9
a                         <1> lc[t3] sK/1 ->-
-                            <1> ex-rv2sv sK/1 ->a
9                              <$> gvsv(*_) s ->a
6                 <1> rv2av[t4] lKM/1 ->7
5                    <$> gv(*array) s ->6
-        <1> ex-list lK ->d
b           <0> pushmark s ->c
c           <0> padav[@selected:2,3] lRM*/LVINTRO ->d
```

"Premature optimization is the root of all evil"

Tony Hoare

What do I benchmark?

Find the bad parts

* Profile the application first
* Find out who's taking all the time/ memory/network
* Compare situations
* Fix that first

```
perl -d:SmallProf script
```

Better

algorithms

do better

Summary

- Decide what is important to you
- Realize you have bias
- Report the situation
- Don't turn off your brain
- Make predictions that you can verify

Further Reading

- "Benchmarking", The Perl Journal #11, http://www.pair.com/~comdog/Articles/benchmark.1_4.txt
- "Wasting Time Thinking About Wasted Time", http://www.perlmonks.org/?node_id=393128
- "Profiling in Perl", http://www.ddj.com/documents/s=1498/ddj0104pl/