SCONS

Build your software, better.

Why SCons is not slow

Dirk Bächle

PyCon.FR 2014

1. Intro to SCons
2. Problem
3. Analysis
4. Results

```
#include "foo.h"
int main(void)
{
  return 0;
}
```

```
#define FOO "foo"
```

```
env = Environment()

env.Program('main.cpp')
```

```
> ls
foo.h  main.cpp  SConstruct
> scons
g++ -o main.o -c main.cpp
g++ -o main main.o
```

Posix

```
> scons
g++ -o main.o -c main.cpp
g++ -o main main.o
```

Windows

```
C:\> scons
cl /Fomain.obj /c main.cpp /TP /nologo
link /TP /nologo /OUT:main.exe main.obj
```

```
> scons
scons: `.' is up to date.
> (edit foo.h: Kommentar)
> scons
g++ -o main.o -c main.cpp
>
> (edit main.cpp: Neue Funktion)
> scons
g++ -o main.o -c main.cpp
g++ -o main main.o
```

Implicit dependencies

```
> scons --tree=all main
scons: `main' is up to date.
+-main
  +-main.o
  | +-main.cpp
  | +-foo.h
  | +-/usr/bin/g++
  +-/usr/bin/g++
```

Picking a different compiler

```
env = Environment()
env['CC'] = '/opt/bin/mygcc'
env.Program('main','main.cpp')
# or
env.Replace(CC='/opt/bin/mygcc')
env.Program('main','main.cpp')
# or even
env.Program('main','main.cpp',
            CC='/opt/bin/mygcc')
```

A simple LaTeX example

```
env = Environment()
env.PDF(['mybook.tex']+
        Glob('images/*.eps'))
```

```
> scons
epstopdf images/circle.eps --outfile=images/circle.pdf
cd . && pdflatex mybook.tex
cd . && bibtex mybook
cd . && pdflatex mybook.tex
cd . && pdflatex mybook.tex
```

Implicit dependencies

```
> scons --tree=derived mybook.pdf
scons: `mybook.pdf' is up to date.
+-mybook.pdf
  +-mybook.tex
  +-chap1.tex
  +-chap2.tex
  +-images/circle.pdf
  | +-images/circle.eps
  | +-/usr/bin/epstopdf
  +-images/rectangle.png
  +-sources.bib
```

C/C++, Asm, Fortran, Java, D,

TeX/LaTeX, DocBook, gettext,

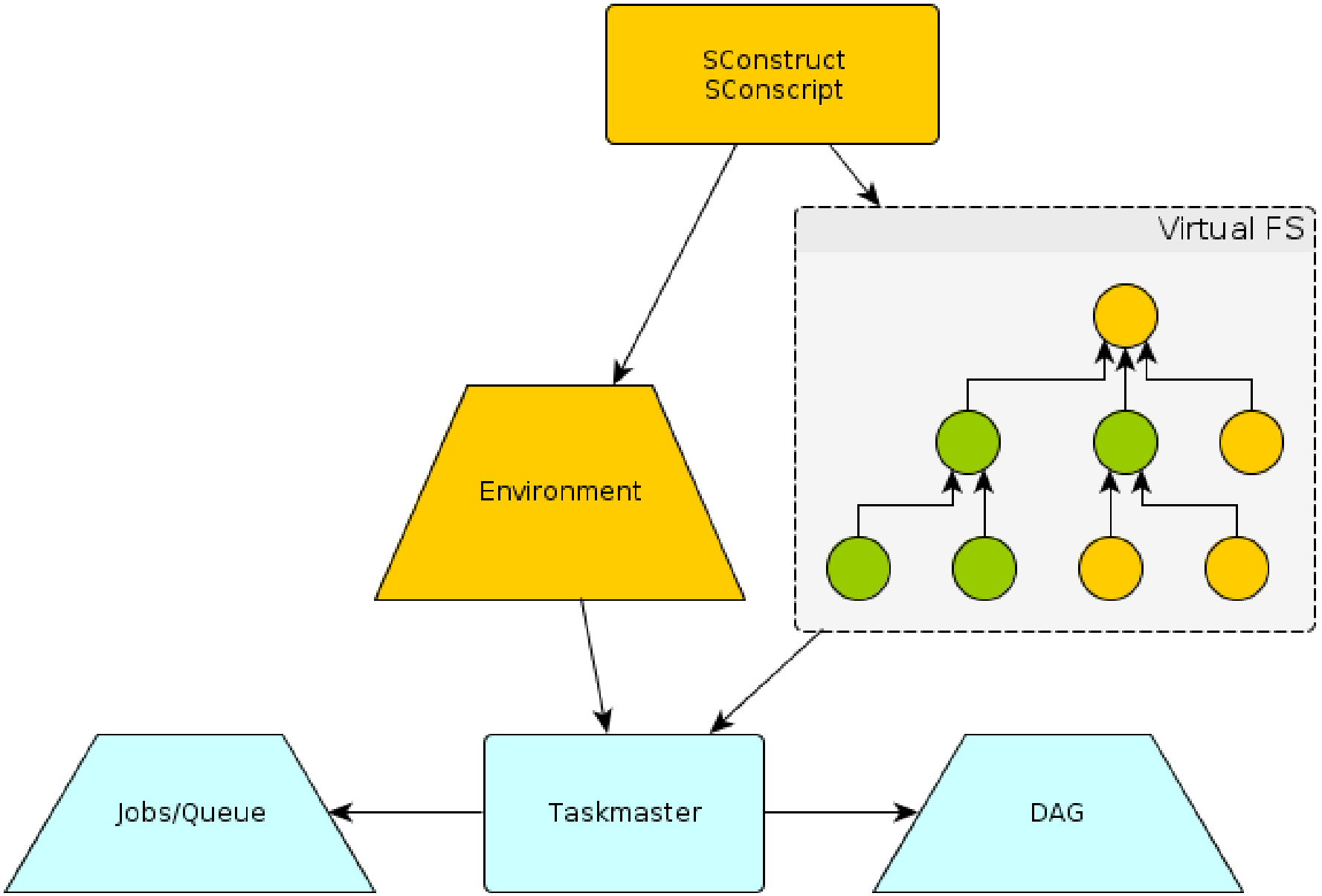
M4, lex/yacc, Qt3, SWIG

Install, Zip, Tar, Rpm, Msi

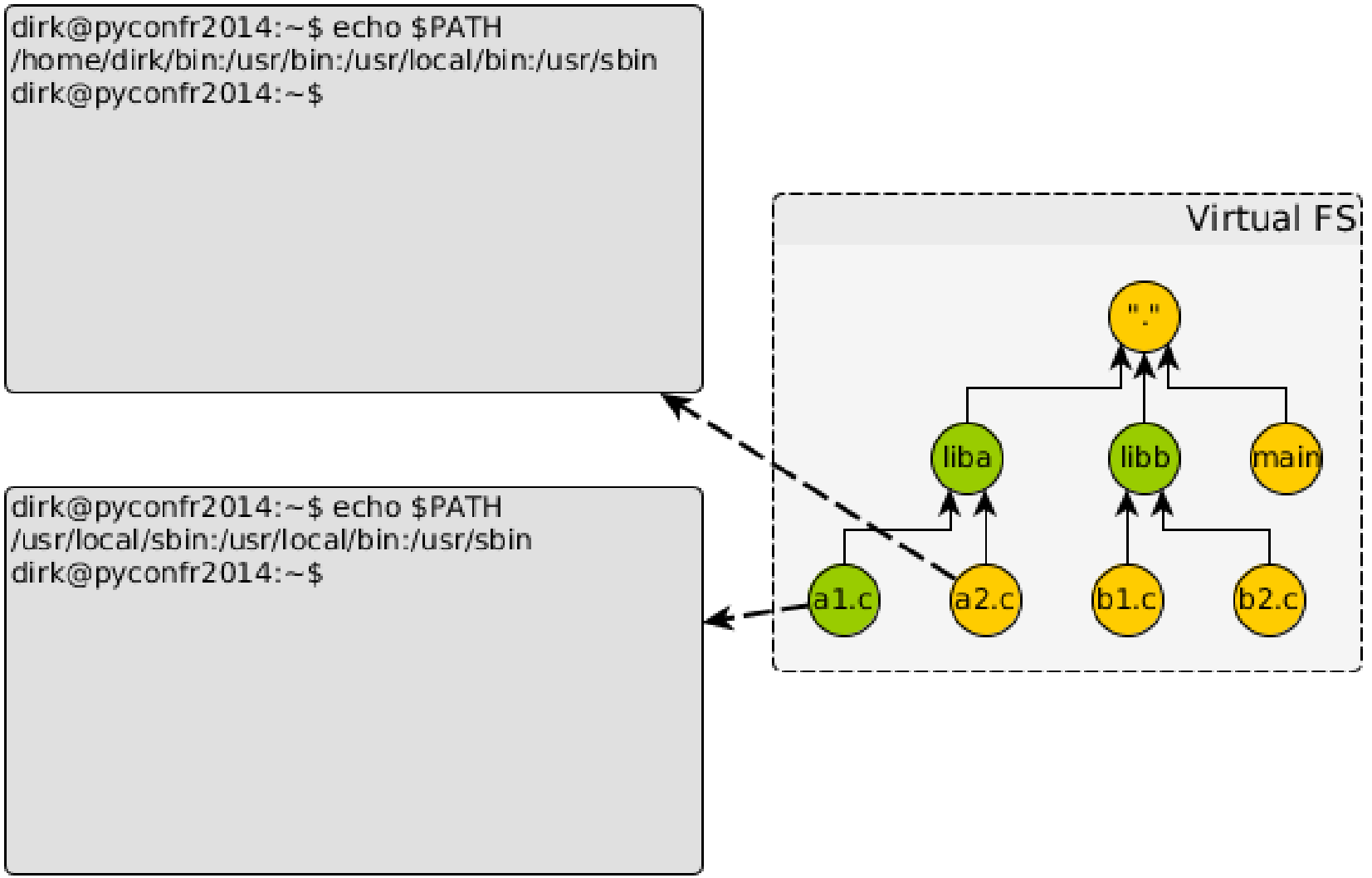
Chapel, C#, CPython, Doxygen, Eiffel, Erlang, Gnuplot, Go, GObject, Haskell, MFObject, OCaml, protobuf, Qt4+Qt5, reST, RightNow, Sphinx, Vala, X10

Nsis, Corba, PyUic, Rpc, Lyx, Pyrex, Gch, CxxTest, Cheetah, FltkFluid, InnoSetup, Cuda, WiX, NDDS4

Down the rabbit hole...

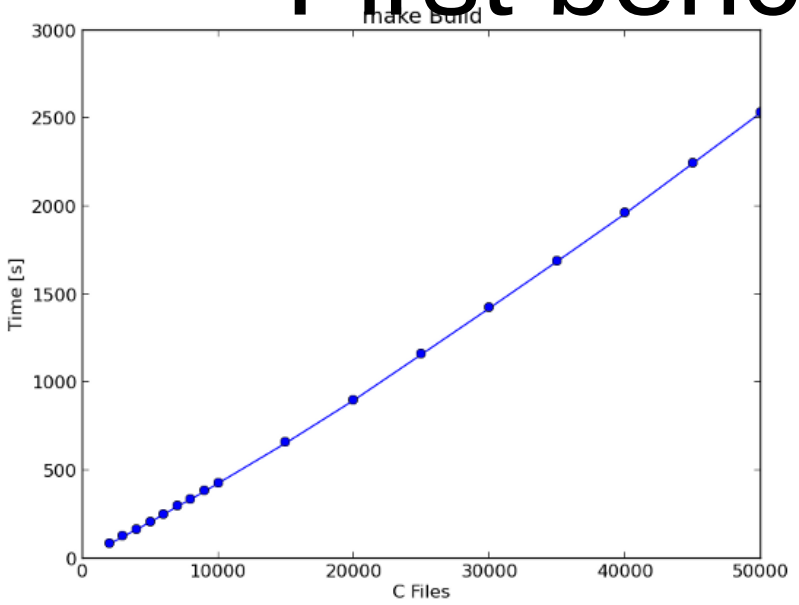


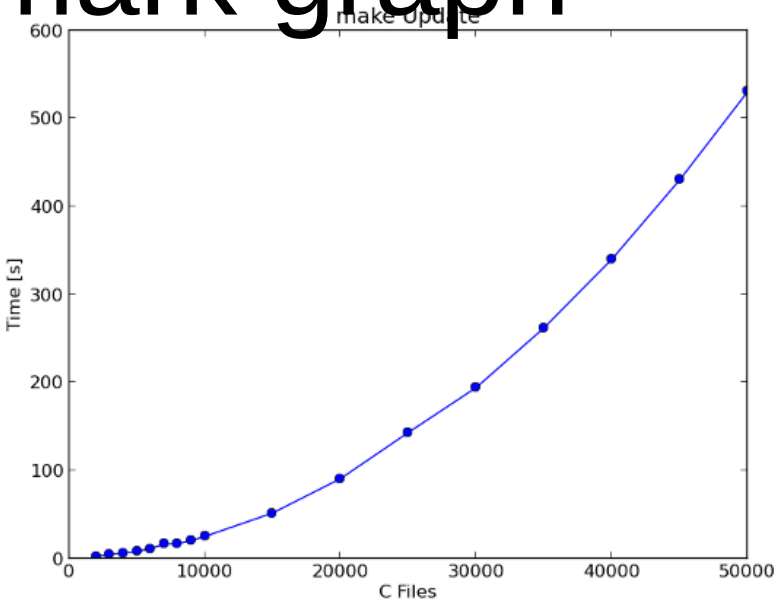
Actions as parameterized templates

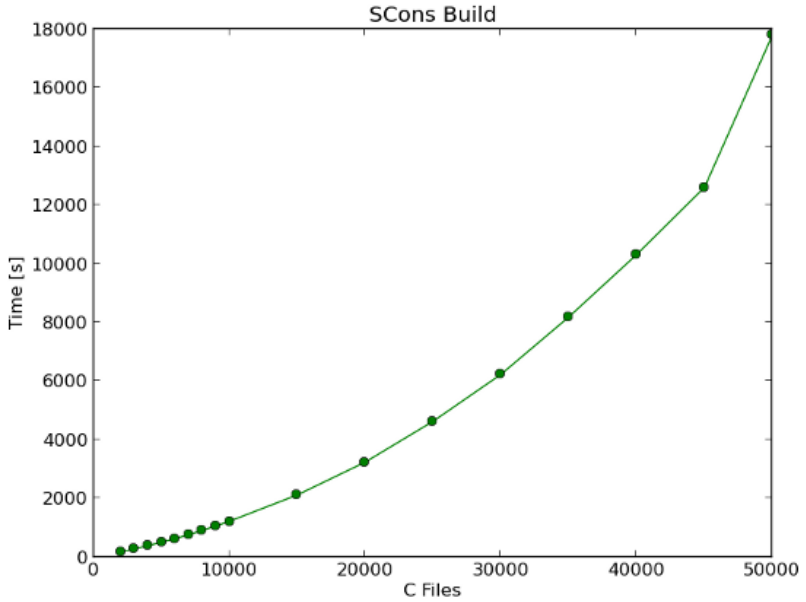


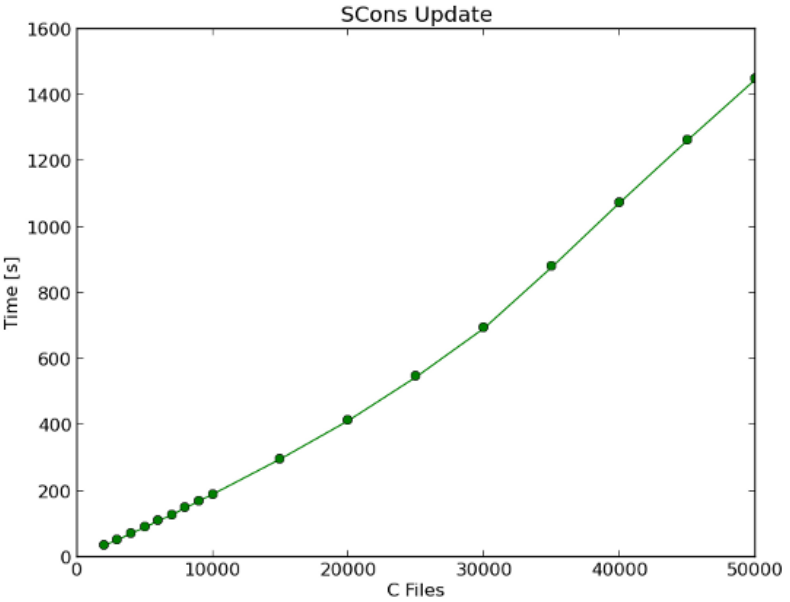
Compile action : `'$CC -o $TARGET -c $CFLAGS $CCFLAGS $SOURCES'`

First benchmark graph









Benchmark structure

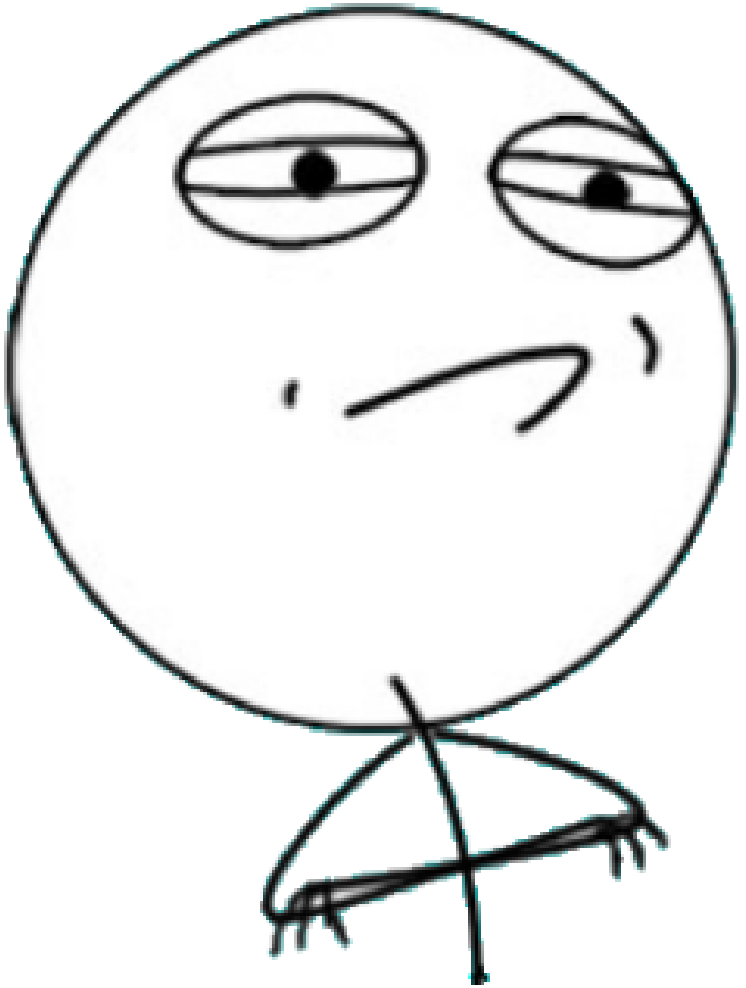
```
.
├── SConstruct
├── 500 C files (→ static lib + main, 20 each)
├── Lup_001
│   └── 500 headers
├── D1_001
│   ├── 500 C files (→ static lib + main, 20 each)
│   └── Lup_d1_001
│       └── 500 headers
├── D1_002
│   └── …
…
```

Benchmark sources

```
//////// Header: ////////
#ifndef f00249_sconsbld
#define f00249_sconsbld "sconsbld"

#include "stdio.h"

#endif

//////// Source: ////////
#include <f00249_sconsbld.h>
#include <omega.h>
extern int printr_f00250_sconsbld
(char * fname);
printr_f00249_sconsbld (char * fname)
{
    printr_f00250_sconsbld (fname);
    return (0);
}
```

```
//////// Header: ////////
class class_0 {
public:
    class_0();
    ~class_0();
};

//////// Source: ////////
#include "class_0.h"
...several more includes

class_0::class_0() {}
class_0::~class_0() {}
```

CHALLENGE ACCEPTED



(Source: http://memekid.com/meme-faces-challenge-accepted.htm)

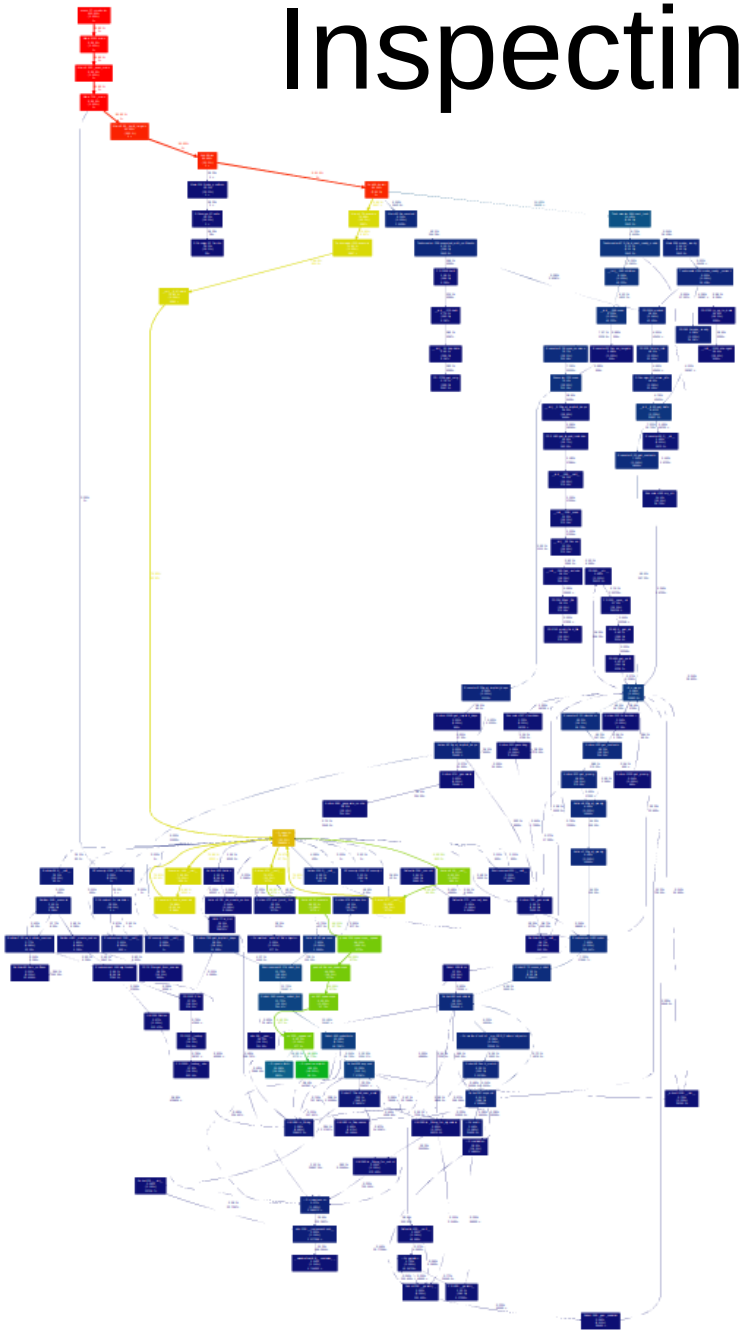
Basic method

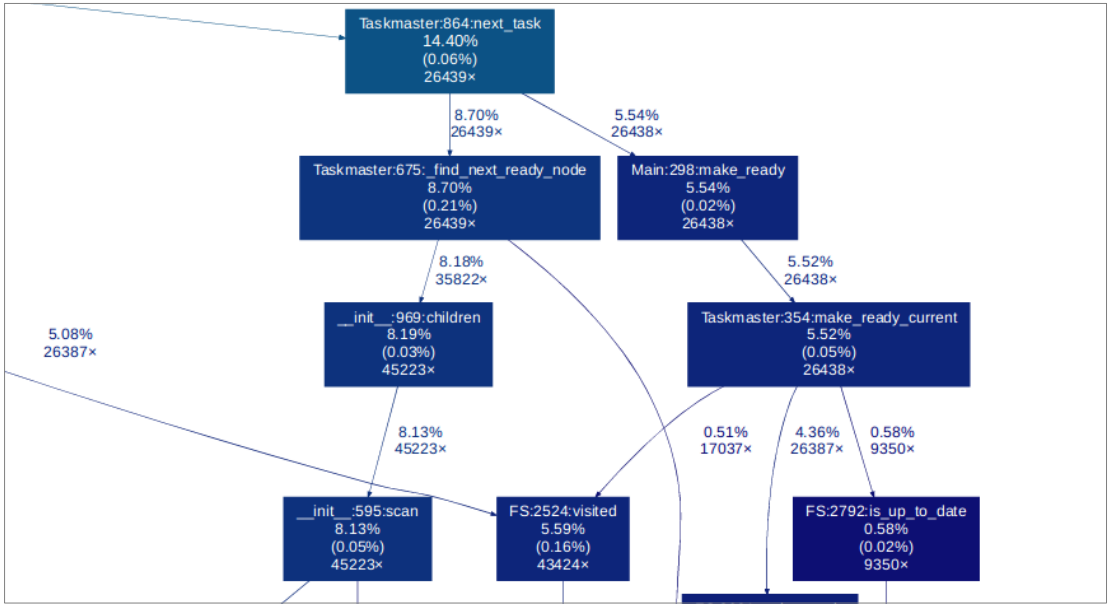
Profile! Measure! Document the results! Profile! Measure! Document the results! Profile! Measure! Document the results! Profile! Measure! Document the results! Profile! Measure! Document the results! Profile! Measure! Document the results! Profile! Measure! Document the results! Profile! Measure! Document the results! Profile! Measure! Document the results! Profile! Measure! Document the results! ….

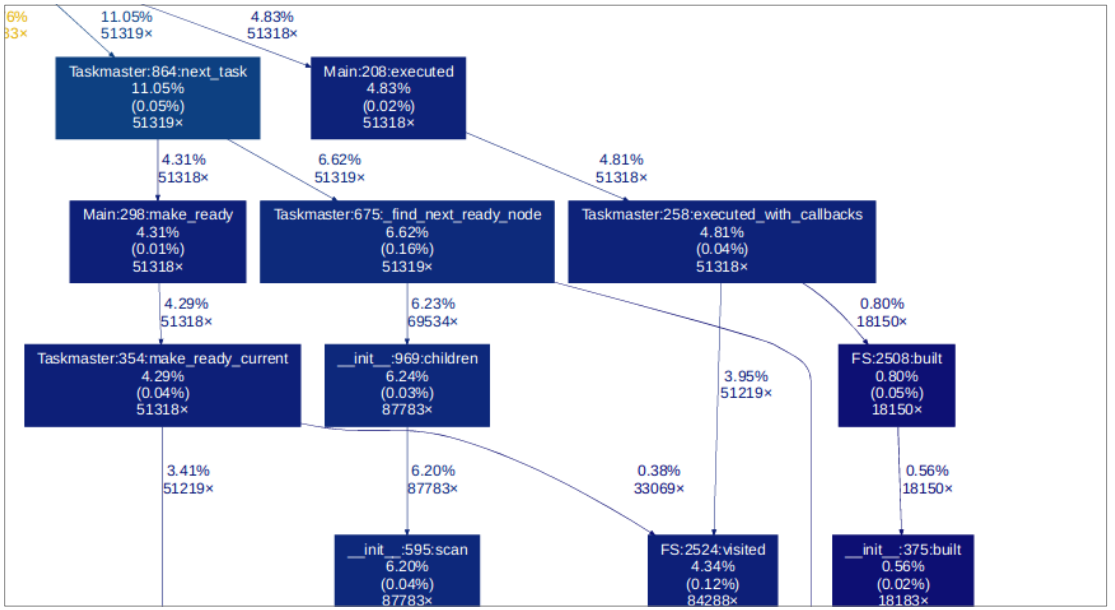
Our chief weapons...

(Source: *The Spanish Inquisition*, jumperbean2, https://www.youtube.com/watch?v=Tym0MObFpTI)

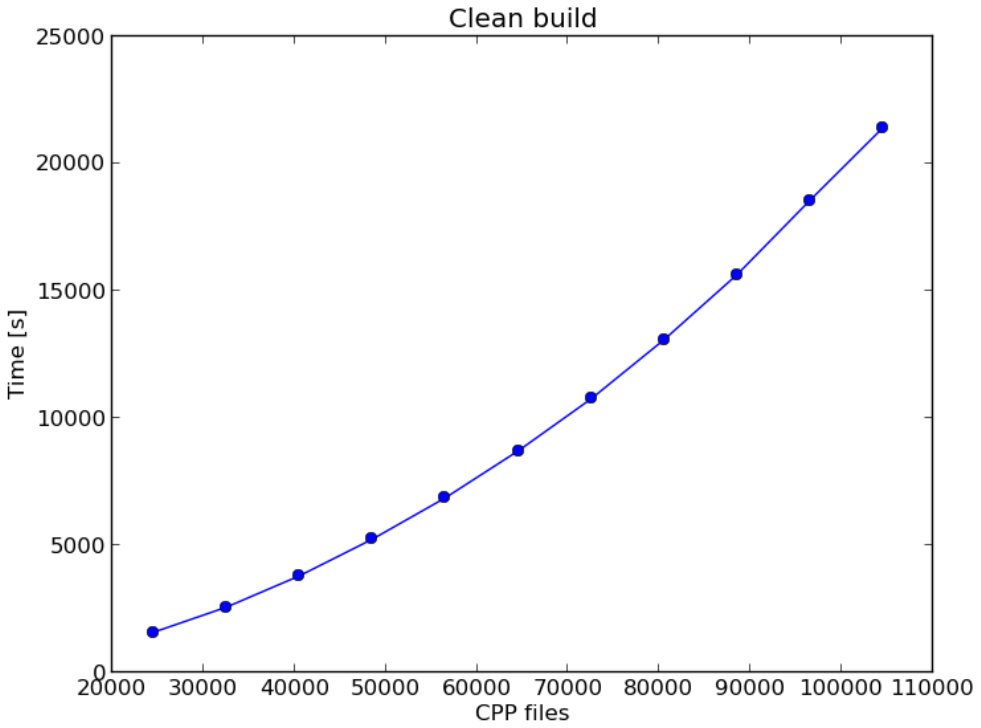
Inspecting the Taskmaster

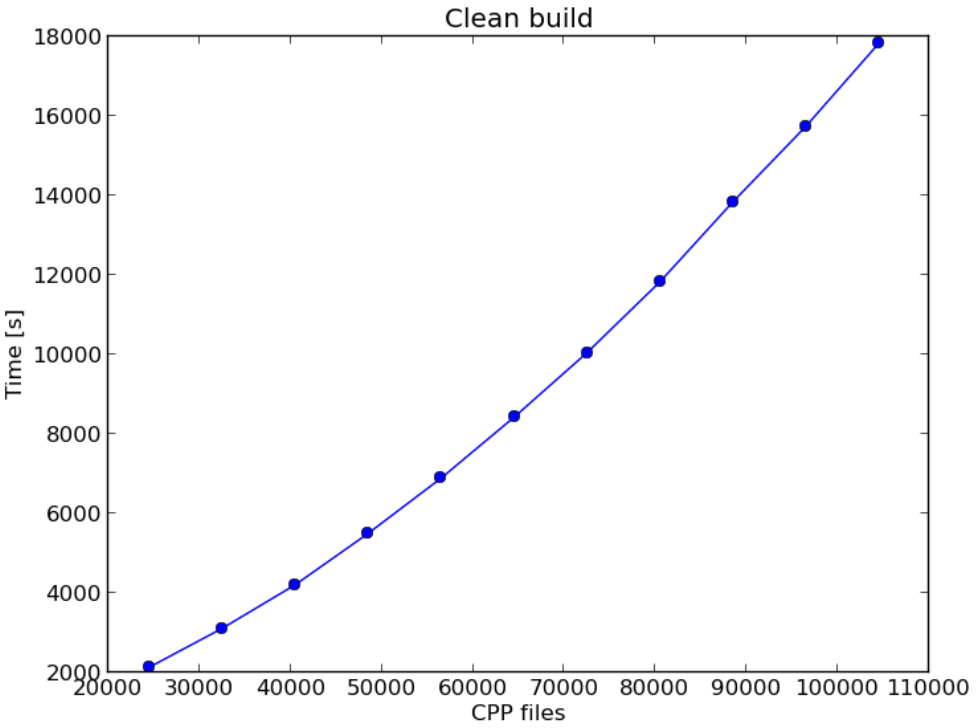






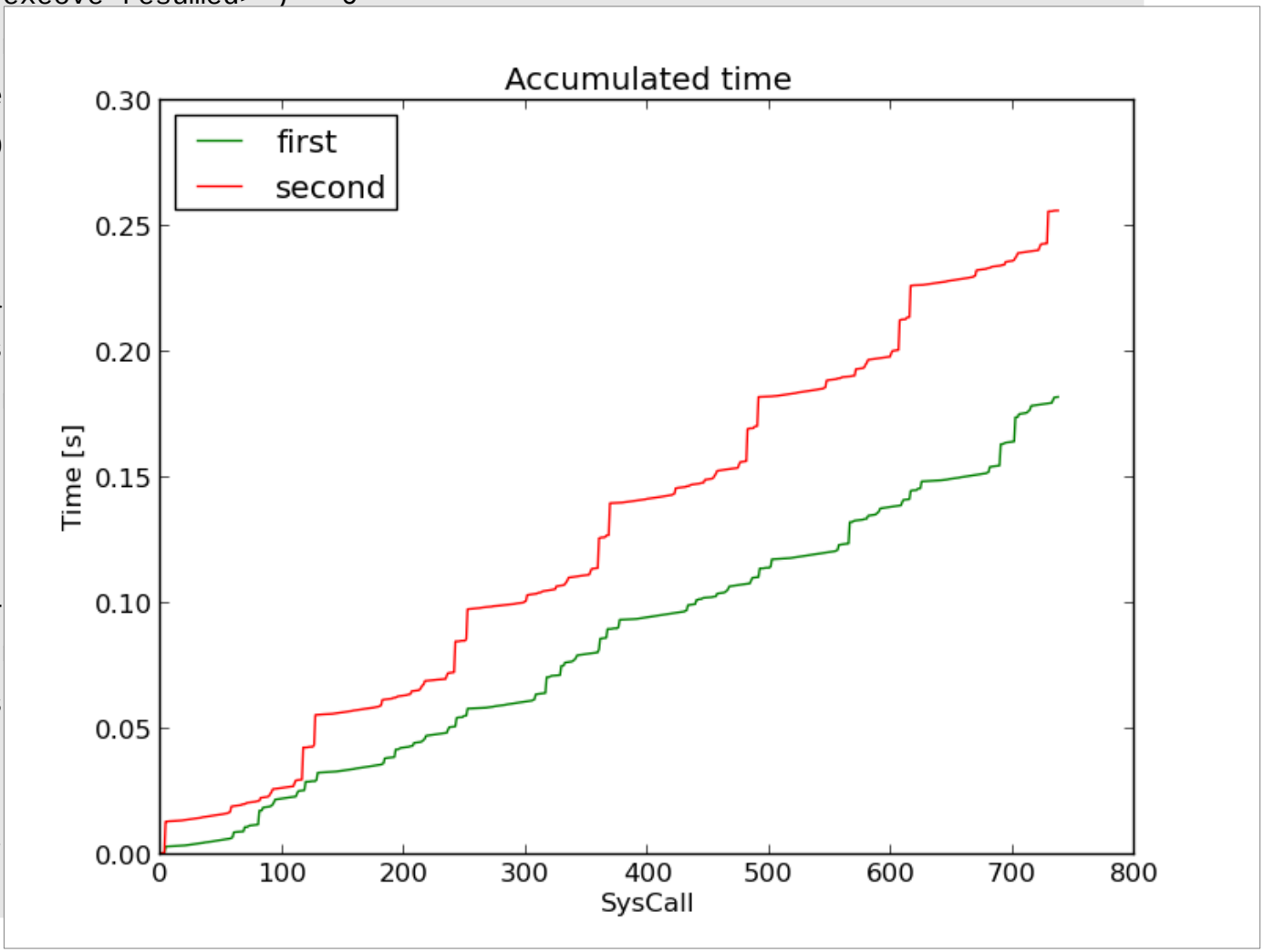
CPython vs PyPy





Let'strace it...

```
2659       0.000054 <... mmap resumed> ) = 0x7f9044857000
2659       0.000046 read(3, "", 1048576) = 0
2659       0.002740 mremap(0x7f9044857000, 1052672, 4096, MREMAP_MAYMOVE <unfinished ...>
2663       0.000026 <... execve resumed> ) = 0
2659       0.000013 <...
2659       0.000043 close
2663       0.000033 brk(0
2659       0.000010 <...
2663       0.000011 <...
2659       0.000023 munma
2663       0.000024 acces
2659       0.000015 <...
2663       0.000021 <...
2663       0.000038 mmap(


2659       0.000017 wait4
2663       0.000008 <...
2663       0.000050 acces
directory)
2663       0.000074 open(
2663       0.000064 fstat
2663       0.000074 mmap(
```

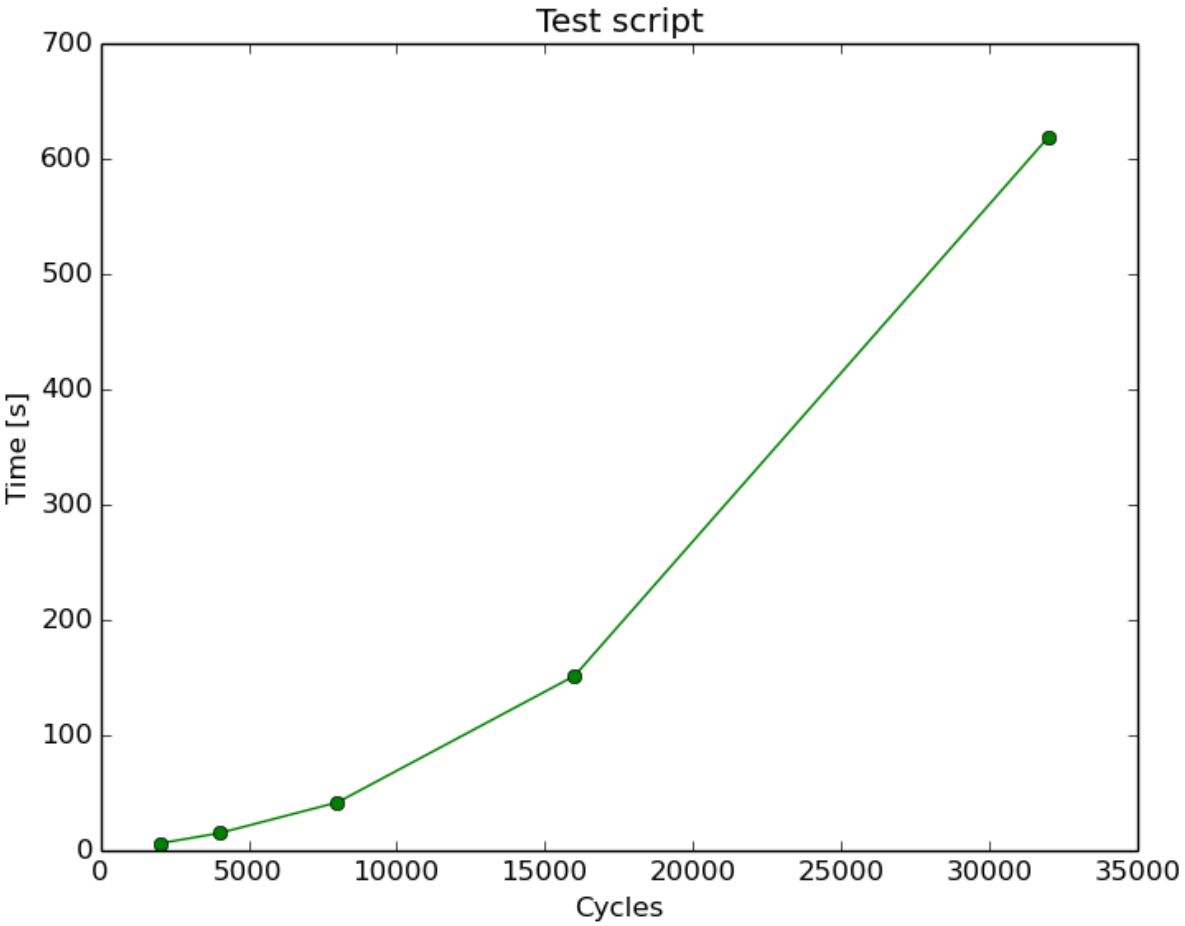


Test script

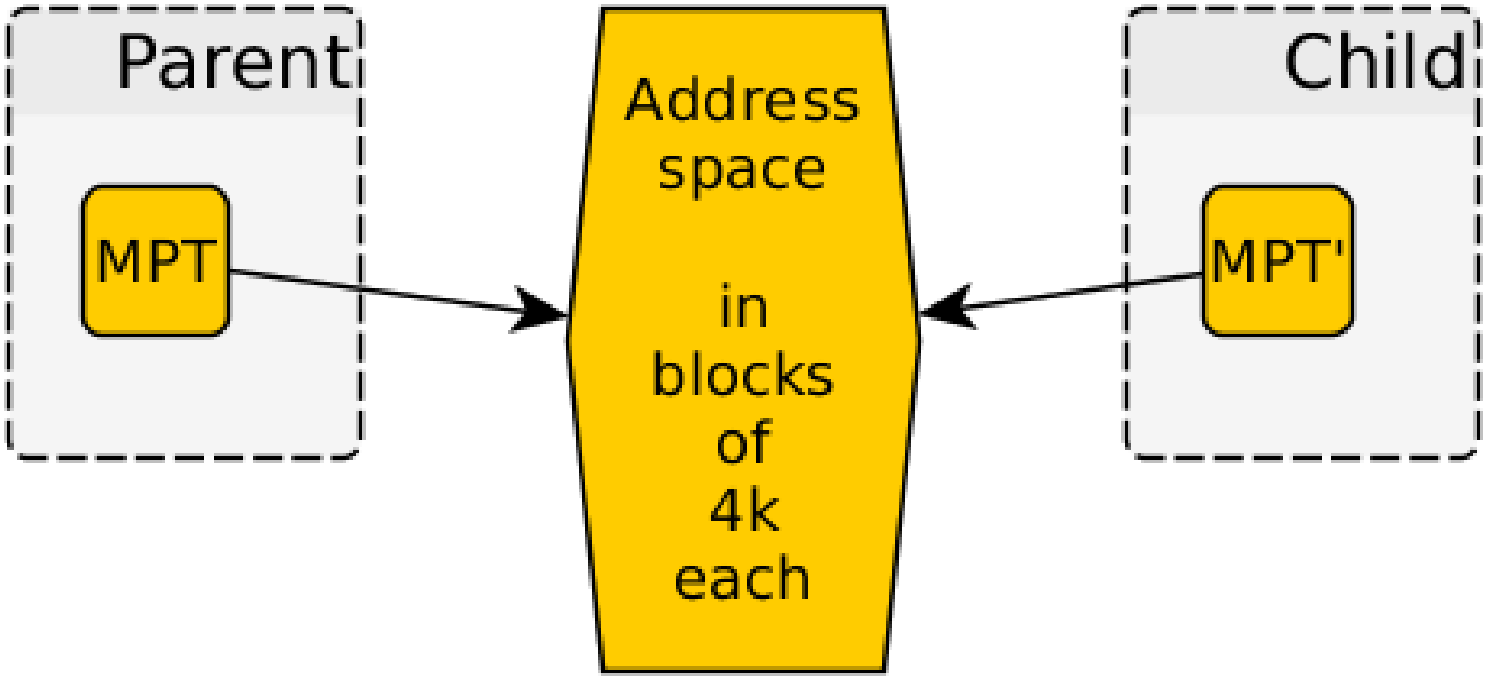
```
import os, sys

def main():
    cycles = 25000
    if len(sys.argv) > 1:
        cycles = int(sys.argv[1])

    m_list = []
    cnt = 0
    for i in xrange(cycles):
        cnt += 1
        args = ['echo', '%d/%d' % (cnt, cycles)]
        os.spawnvpe(os.P_WAIT, args[0], args, os.environ)
        signature ='A'*20000
        m_list.append(signature)

if __name__ == "__main__":
    main()
```

Look ma,... not linear

Cycles	Elapsed
2000	00:05.83
4000	00:14.65
8000	00:41.42
16000	02:31.24
32000	10:46.85



The forking problem



Related talks

Subprocess to FFI: Memory, Performance, and Why You Shouldn't Shell Out

Christine Spang, PyCon 2014, Montréal

http://pyvideo.org/video/2640/subprocess-to-ffi-memory-performance-and-why-y

Fast Python, Slow Python

Alex Gaynor, PyCon 2014, Montréal

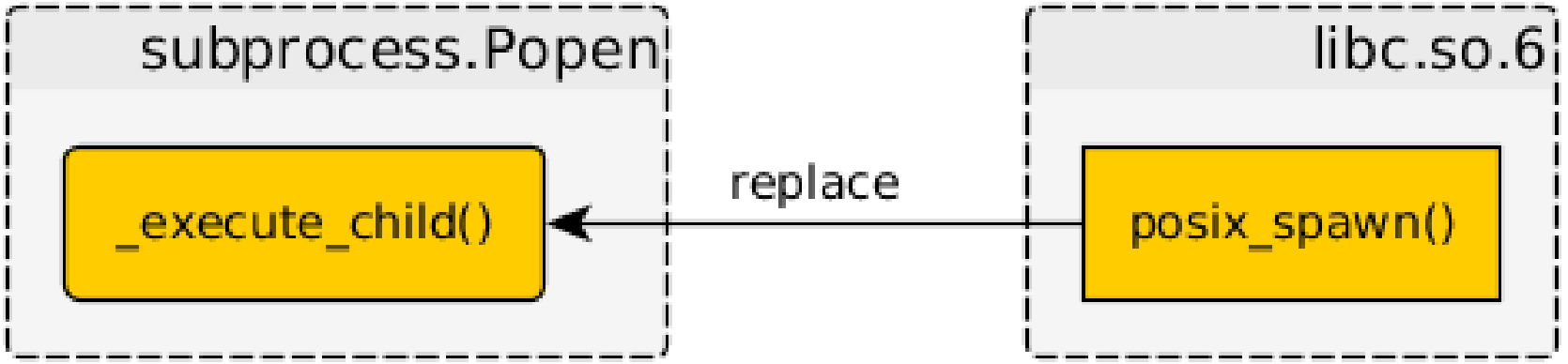
http://pyvideo.org/video/2627/fast-python-slow-python

Python is slow, make it faster with C

Ben Shaw, Kiwi PyCon 2014, Wellington

http://pyvideo.org/video/3226/python-is-slow-make-it-faster-with-c

Our solution

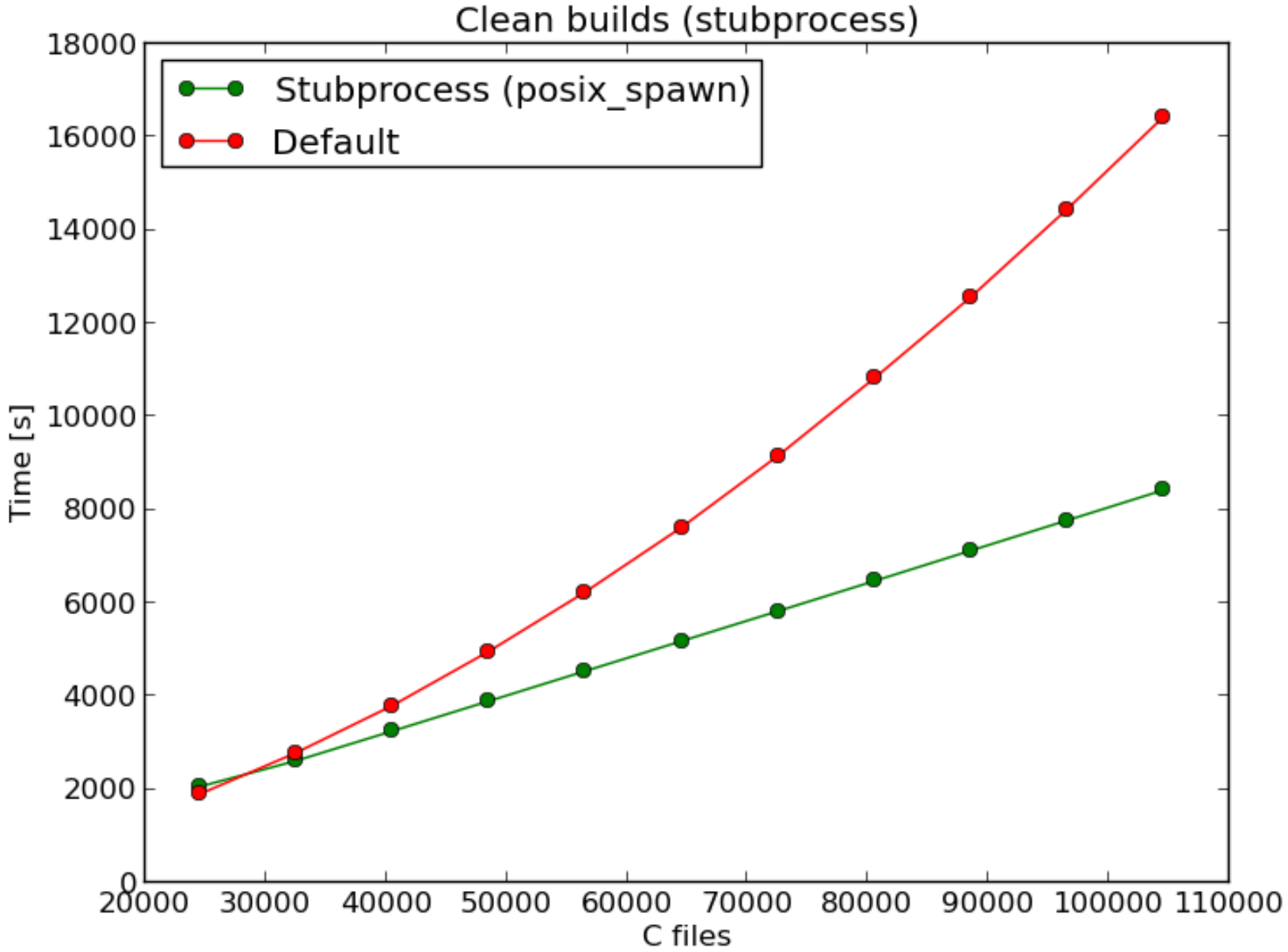


posix_spawn() = vfork() + execve()

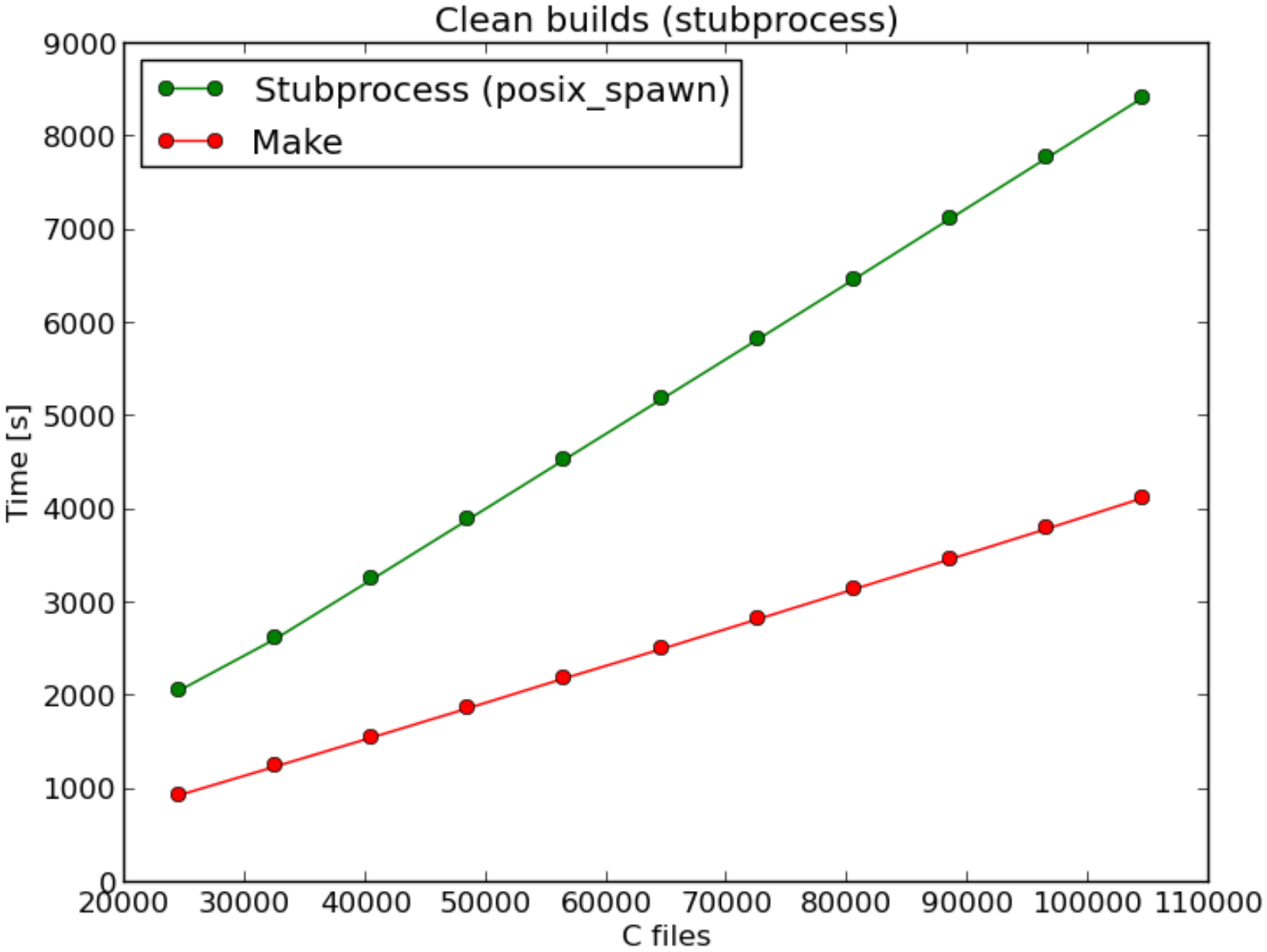
Simple script „`stubprocess.py`“, import after `subprocess`, supports „linux2“ and „darwin“ under Python 2.7.x.

All credits for this wrapper go to: Jason Kenny, Eugene Leskinen, and the rest of their Intel team!

Final results



Final results

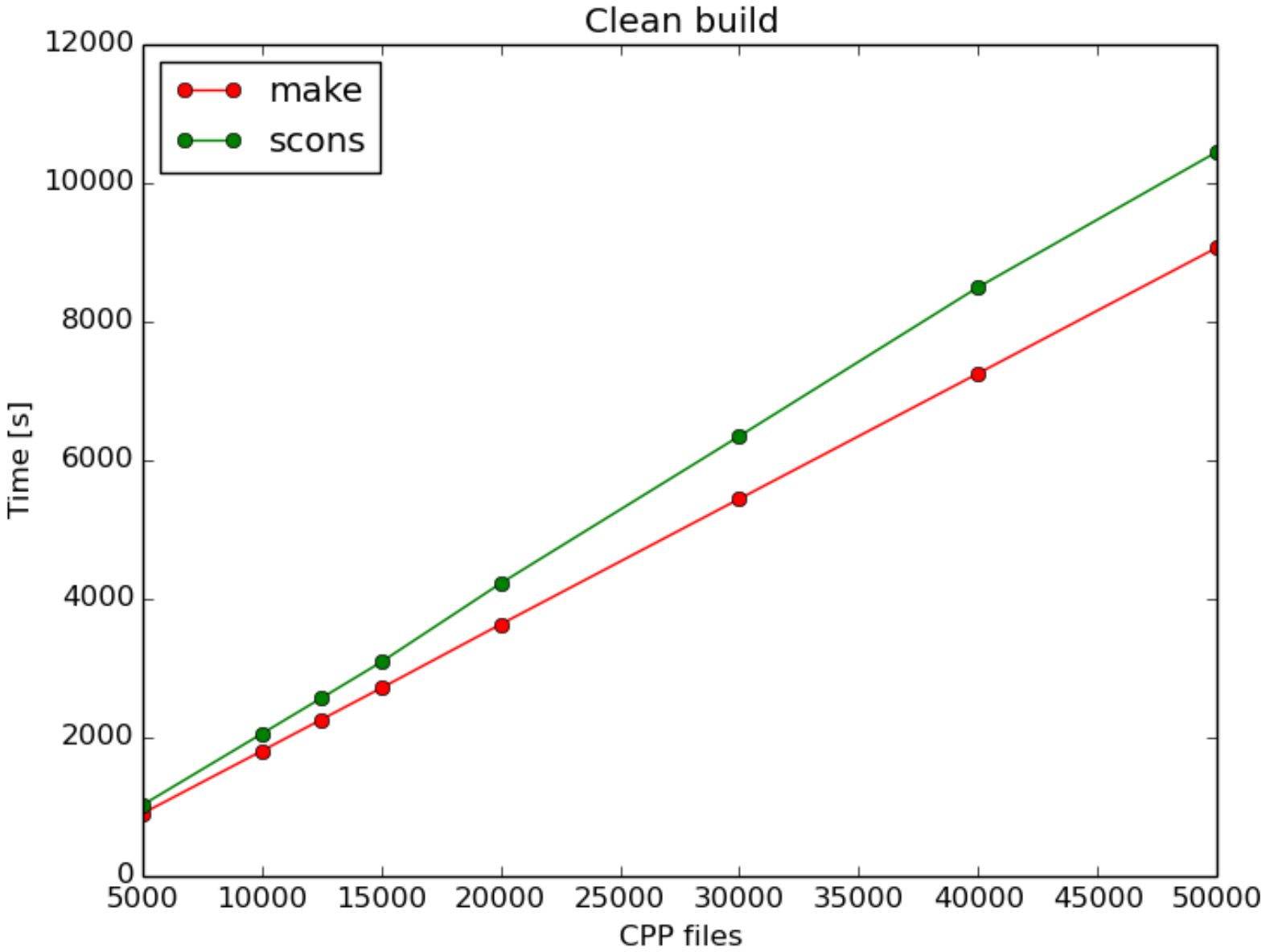


Patched sources

```
//////// Source: ////////
#include "class_0.h"
<several other includes>
using namespace std;

class_0::class_0() {}
class_0::~class_0() {}
class_0::class_0(const class_0 &elem) {
  data = elem.data;
}
class_0 &class_0::operator=(const class_0
&elem) {
  if (&elem == this) {
    return *this;
  }
  data = elem.data;
  return *this;
}
void class_0::clear() { data.clear(); }
void class_0::addData(const string &value) {
  data.push_back(value);
}
```

```
//////// Header: ////////
#ifndef class_0_h_
#define class_0_h_

#include <string>
#include <vector>
class class_0
{
public:
    class_0();
    ~class_0();
    class_0(const class_0 &elem);
    class_0 &operator=(const class_0 &elem);

    void addData(const std::string &value);
    void clear();
private:
    std::vector<std::string> data;
};

#endif
```

Final results (cont'd)



Resources

- Web: www.scons.org
- Wiki: www.scons.org/wiki
- Downloads, mailing lists, documentation, examples and recipes
- All the data: http://scons.org/wiki/WhySconsIsNotSlow
- Parts: http://parts.tigris.org/
- OpenHatch: http://openhatch.org/projects/SCons

- Slides: https://bitbucket.org/dirkbaechle/scons_talks
- Me: http://google.com/+DirkBächle (dl9obn@darc.de)

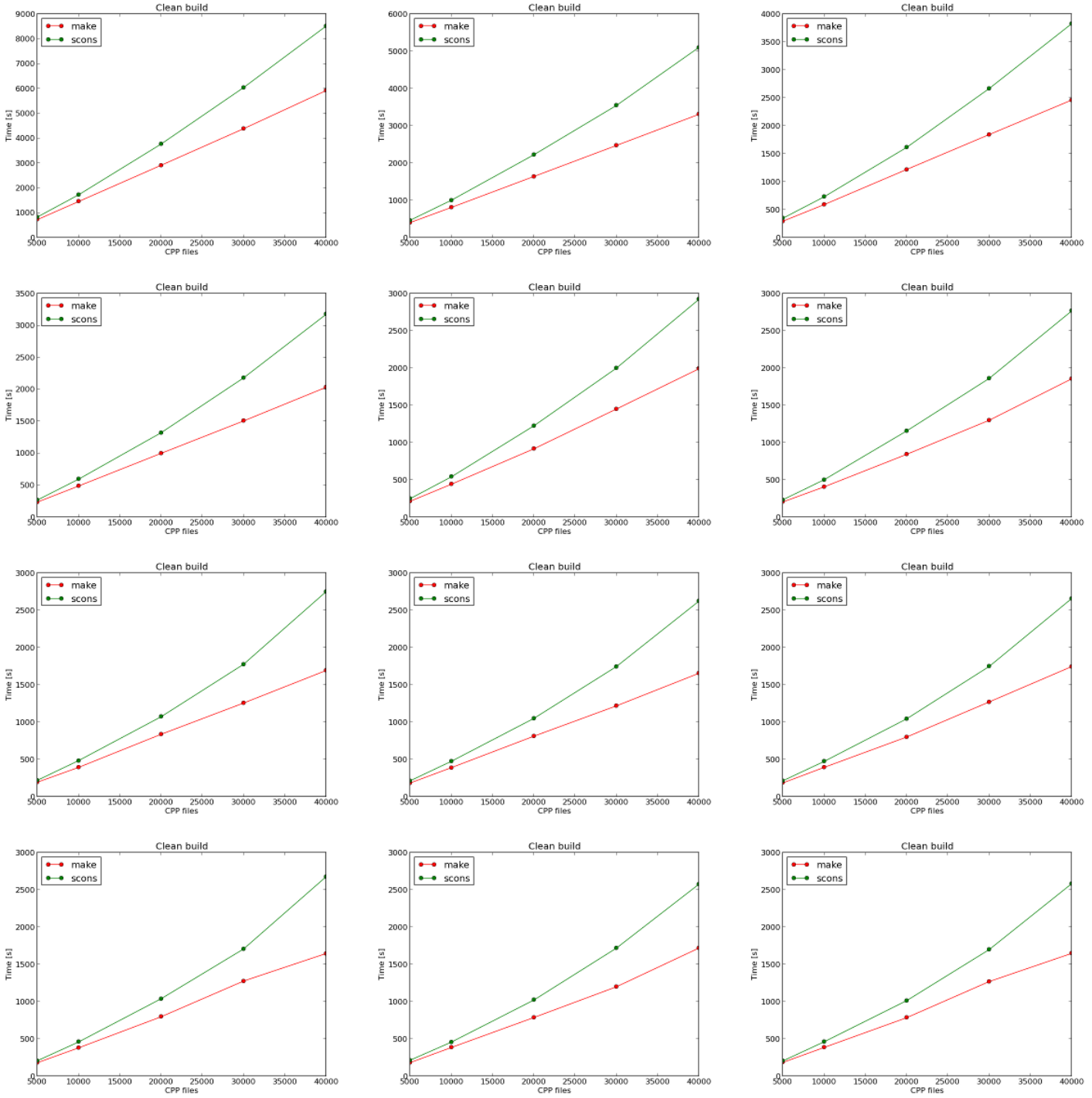
News

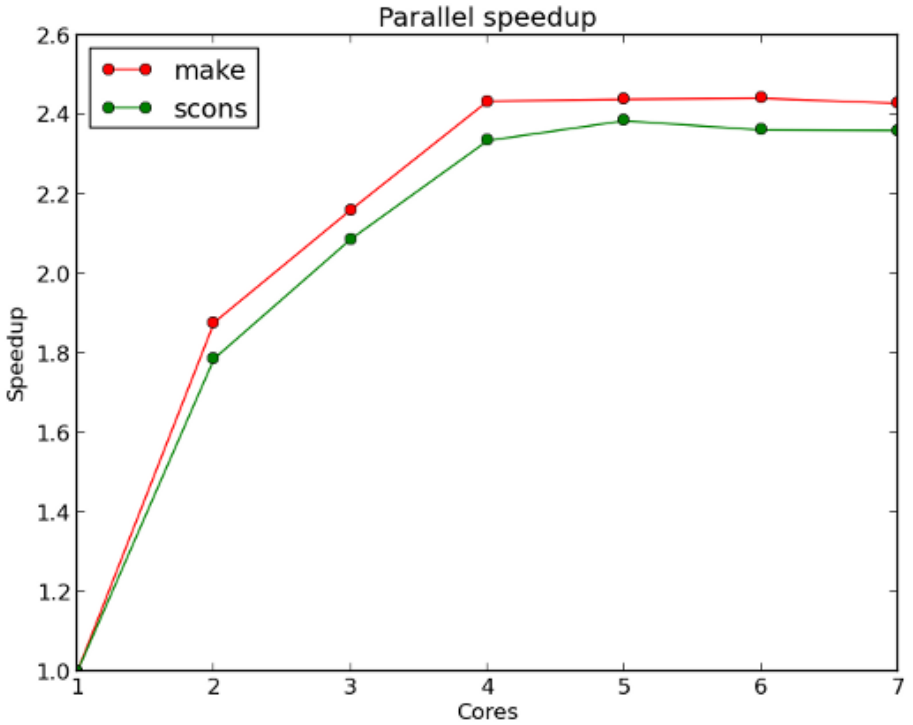
- SVN → hg
- Documentation toolchain in DocBook
- External tools (ToolsIndex)
- Python 2.6 is the new floor version (switching to 2.7 soon)
- Versioned SharedLibraries

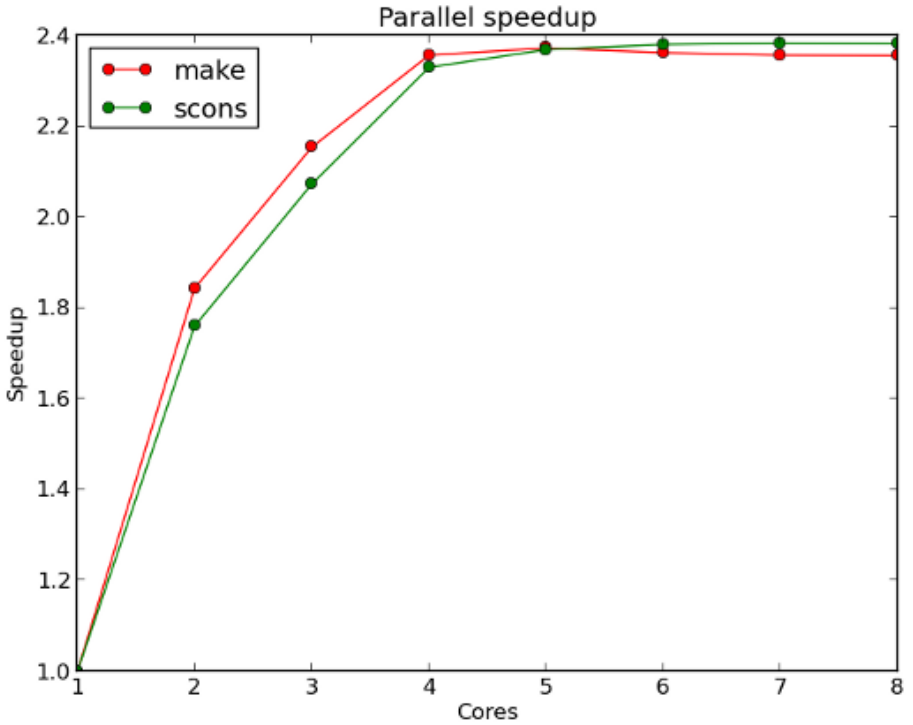
Roadmap

- Python 3
- Tools subsystem (adding „toolchains“)
- Migrating the bugtracker away from Tigris
- Bugs, documentation, …

(Source: https://www.flickr.com/photos/walkn/3526522573)


Clean build
make
scons
Time [s]
CPP files


Parallel speedup
make
scons
Speedup
Cores
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
1
2
3
4
5
6
7


Parallel speedup
make
scons
Speedup
Cores
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
1
2
3
4
5
6
7
8

Install Builder

```
env = Environment()
hello = env.Program('hello.c')
env.Install('/usr/bin', hello)
env.Alias('install', '/usr/bin')
```

```
env.Package(NAME = “hello“, VERSION = '1.0',
            PACKAGEVERSION = 0, PACKAGETYPE = 'rpm',
            LICENSE = 'gpl', SUMMARY = 'Hello test RPM',
            DESCRIPTION = 'A simple test',
            X_RPM_GROUP = 'Application/something',
            SOURCE_URL = 'http://hello.org')
```

Configure context

```
env = Environment()
conf = Configure(env)
if not conf.CheckCHeader('math.h'):
    print 'Math.h must be installed!'
    Exit(1)
if conf.CheckCHeader('foo.h'):
    conf.env.Append('-DHAS_FOO_H')
env = conf.Finish()
```

Command-line option (based on optparse)

```
AddOption('--prefix',
          dest='prefix',
          type='string',
          nargs=1,
          action='store',
          metavar='DIR',
          help='installation prefix')

env = Environment(PREFIX = GetOption('prefix'))

installed_foo = env.Install('$PREFIX/usr/bin',
'foo.in')
Default(installed_foo)
```