Subject: Python for your applications
Posted by bytefield on Sat, 03 Jan 2009 23:31:42 GMT
View Forum Message <> Reply to Message

Hello,
I was thinking to use python with c++ programs, so I could benefit from python as scripting language, unicode support, sockets, etc.
I've tried some small test to see how python work with. Python 2.5.2 filled my 3 Gigs of memory in seconds and also 500 Mb from swap until i've killed it. With python 3 it finished the execution using just 2.5 MB of memory but it's time is not what i want .
Python timecan@can-laptop:~$ time ./pytest.py

```
real 6m38.723s
user 6m28.628s
sys 0m0.600s
```

C++ timecan@can-laptop:~$ time ./cpptest

```
real 0m4.837s
user 0m4.696s
sys 0m0.020s
```

```
#!/usr/bin/env python3
# pytest.py
def fun(x):
 return x + 1

for x in range(0,1000000000):
 y = fun(x)
```

```
// cpptest.cpp
int fun(int x)
{
 return x + 1;
}

int main()
{
 for(int y, x = 0; x < 1000000000; x++)
  y = fun(x);
 return 0;
}
```

So, now after seeing those results i'm thinking not to use python embed in application. I'm curious what will be the results for usc? Hope better

P.S.: I think the test are right, i've tried to not let C++ compiler to optimize it. If someone can provide other test which prove that python is though good as scripting side of a application please post them and try to convince me about the reality

Subject: Re: Python for your applications
Posted by Sender Ghost on Sun, 04 Jan 2009 22:05:09 GMT
View Forum Message <> Reply to Message

Hello, Andrei.

I tested your test case on Windows XP 32 bit. My results are:
1. For 10000000
MSVC++ v9.0 SP1: 16 msec
Python v2.6.1: 3 sec, 156 msec
Python v3.0: 3 sec, 344 msec
2. For 100000000
MSVC++ v9.0 SP1: 47 msec
Python v2.6.1: 31 sec, 828 msec
Python v3.0: 33 sec, 578 msec
3. For 1000000000
MSVC++ v9.0 SP1: 422 msec
Python v2.6.1: MemoryError
Python v3.0: 5 min, 25 sec, 781 msec.

Memory consumption of Python v2.6.1 are large (about 1.5 Gb of 3.25 Gb accessible memory for this test case). Python v3.0 - about 268 Kb.

Because in Windows I haven't the time program for benchmarking I created following gettime program

```
// gettime.cpp
#include <Core/Core.h>

using namespace Upp;

String FormatElapsedTime(double run)
{
 String rtime;
 int msec = int(run) % 1000;
 run /= 1000;
 double hrs = floor(run / 3600);

 if (hrs > 0) rtime << NFormat("%0n hours, ", hrs);

 int minsec = fround(run - 3600 * hrs);
 int min = minsec / 60, sec = minsec % 60;
 if (min || hrs) rtime << min << " min, ";
 if (sec || min) rtime << sec << " sec, ";

 rtime << msec << " msec";
 return rtime;
}

CONSOLE_APP_MAIN
{
 const Vector<String>& cmdline = CommandLine();

 if (cmdline.GetCount() == 0)
 {
  Cout() << "Not enough command line arguments\n";
  exit(0);
 }

 StringBuffer sb;

 for (int i = 0, n = cmdline.GetCount(); i < n; ++i)
 {
  sb << cmdline[i];
  if (i < n - 1) sb << " ";
 }

 String output;
 TimeStop ts;
 int exitCode = Sys(String(sb), output);
 double elapsed = ts.Elapsed();

 Cout() << "Elapsed: " << FormatElapsedTime(elapsed) << EOL
  << "Exit code: " << AsString(exitCode) << EOL;

 if (!output.IsEmpty()) Cout() << "Output:\n" << output;
}
```

Then I compiled test case python script to optimized bytecode using following batch file (also with benchmarking):

```
rem benchmark.bat
SETLOCAL
SET Python26=C:\Python26
SET Python30=C:\Python30
SET DirectoryToCompile=%~p0

gettime cpptest
```

```
%Python26%\python -O %Python26%\Lib\compileall.py %DirectoryToCompile%
gettime %Python26%\python pytest.pyo
del *.pyo

%Python30%\python -O %Python30%\Lib\compileall.py %DirectoryToCompile%
gettime %Python30%\python pytest.pyo
del *.pyo
```

You can try Lua programming language. I created similar test case:

```
-- luatest.lua
function fun(x)
 return x + 1
end

for x = 0, 1000000000 do
 y = fun(x)
end
```

My results are:
1. For 10000000
Lua v5.1.4: 1 sec, 171 msec
2. For 100000000
Lua v5.1.4: 11 sec, 469 msec
3. For 1000000000
Lua v5.1.4: 1 min, 54 sec, 391 msec.

And memory consumption about 112 Kb.

Also for comparison purposes I created test cases for PHP

```
<?php
// phptest.php
function fun($x)
{
 return $x + 1;
}

for ($x = 0; $x < 1000000000; ++$x)
{
 $y = fun($x);
}
?>
```

and Perl:

```
# perltest.pl
```

```
sub fun
{
 my $x = shift;
 return $x + 1;
}

for my $x (0 .. 1000000000)
{
 my $y = fun($x);
}
```

with:

```
gettime php -n -f phptest.php
gettime perl perltest.pl
```

My results are:
1. For 10000000
PHP v5.2.8: 8 sec, 656 msec
Perl v5.10.0: 6 sec, 187 msec
2. For 100000000
PHP v5.2.8: 1 min, 15 sec, 78 msec
Perl v5.10.0: 1 min, 1 sec, 422 msec
3. For 1000000000
PHP v5.2.8: 12 min, 37 sec, 172 msec
Perl v5.10.0: 10 min, 14 sec, 157 msec.

Memory consumption:
PHP v5.2.8: 388 Kb
Perl v5.10.0: 268 Kb.

Added:
The correct cpptest.cpp file are follows:

```
int fun(int x)
{
 return x + 1;
}

int main()
{
 int y;

 for(int x = 0; x < 1000000000; ++x)
 {
  y = fun(x);
 }
 return y;
}
```

And what about C++ (or JavaScript) interpreter embedded into your program? Examples: Ch, CINT C/C++ Interpreter.

Added:
1. For 10000000
Cint v5.16.19: 5 sec, 703 msec
2. For 100000000
Cint v5.16.19: 47 sec, 562 msec
3. For 1000000000
Cint v5.16.19: 7 min, 44 sec, 719 msec.

Memory consumption of Cint v5.16.19: 260 Kb.

Updated and added:
Result table:

#	Name
1.	MSVC++ v9.0 SP1
2.	GNU GCC v4.2.4 TDM-1
3.	Python v2.6.1
4.	Python v2.6.1 with xrange
5.	Python v3.0
6.	Lua v5.1.4
7.	PHP v5.2.8
8.	Perl v5.10.0
9.	Cint v5.16.19

#	10000000	100000000	1000000000
1.	16 msec	47 msec	422 msec
2.	31 msec	31 msec	31 msec
3.	3 sec, 156 msec	31 sec, 828 msec	MemoryError
4.	3 sec, 860 msec	28 sec, 938 msec	4 min, 38 sec, 157 msec
5.	3 sec, 344 msec	33 sec, 578 msec	5 min, 25 sec, 781 msec
6.	1 sec, 171 msec	11 sec, 469 msec	1 min, 54 sec, 391 msec
7.	8 sec, 656 msec	1 min, 15 sec, 78 msec	12 min, 37 sec, 172 msec
8.	6 sec, 187 msec	1 min, 1 sec, 422 msec	10 min, 14 sec, 157 msec
9.	5 sec, 703 msec	47 sec, 562 msec	7 min, 44 sec, 719 msec

Subject: Re: Python for your applications
Posted by bytefield on Mon, 05 Jan 2009 08:59:29 GMT

View Forum Message <> Reply to Message

Thanks, I appreciate your efforts to test interpreted languages and I have to say that I'm impressed about Lua results. It's pity that massive usage of function calls is that expensive in python, i liked features provided by this languages. Perhaps python will work good too as scripting side of an app. or as standalone language while wrapping heavy parts written in C/C++ to gain full speed at functions calls, benefit from C/C++ compiler optimizations and run parts that usually you can't write in interpreted languages.
I've think at python languages as it is world wide used and his features doesn't force you to reinvent the wheel to have them.
Compared to python, lua have better performances but lack of some features available for python. For example is much simple so use boost python to embed C++ classes in python than using API provided by lua to embed C++ classes in it. The comparison could continue with lua std library vs. python std lib.
I don't know, now i'm hesitate to choose one instead of other, features or speed?

Subject: Re: Python for your applications
Posted by qapko on Mon, 05 Jan 2009 09:33:55 GMT
View Forum Message <> Reply to Message

Hi,
I'm using Python as embedded language for U++ projects and it's great! In U++/C++ I do only GUI and routines that must be optimized for speed. Configuration files and possibly all routines of the function engine of the application are in Python - application is thus configurable and modifiable in Pythonic way, which is simply enjoying. If you are interested in embedding Python in C++ you may look for PyCXX or some other project for wrapping Python API to objects/methods in C++. Also please look for the difference between range and xrange in Python 2.6. The things are not that bad at all when you know what you are doing

Gabi

Subject: Re: Python for your applications
Posted by mirek on Mon, 05 Jan 2009 11:30:31 GMT
View Forum Message <> Reply to Message

qapko wrote on Mon, 05 January 2009 04:33
Hi,
I'm using Python as embedded language for U++ projects and it's great!

Should we offer something in U++ for python? Would you contribute?

Mirek

Subject: Re: Python for your applications
Posted by Sender Ghost on Mon, 05 Jan 2009 13:01:00 GMT
View Forum Message <> Reply to Message

bytefield wrote on Mon, 05 January 2009 13:59
I don't know, now i'm hesitate to choose one instead of other, features or speed?
Thanks, for your test case. It's your choice what to do right. For more detailed comparisons you can see The Computer Language Benchmarks Game site.

Subject: Re: Python for your applications
Posted by cocob on Tue, 06 Jan 2009 13:25:10 GMT
View Forum Message <> Reply to Message

Good idea, If you are interested i already have a upp package for python26.

cocob

Subject: Re: Python for your applications
Posted by tojocky on Tue, 06 Jan 2009 14:02:25 GMT
View Forum Message <> Reply to Message

cocob wrote on Tue, 06 January 2009 15:25Good idea, If you are interested i already have a upp package for python26.

cocob
Maybe you attach the example here?

Subject: Re: Python for your applications
Posted by Sender Ghost on Tue, 06 Jan 2009 21:20:19 GMT
View Forum Message <> Reply to Message

My first post on this thread was updated.
- Added GNU GCC v4.2.4 TDM-1 results for C++.
- PHP was running with -n parameter to exclude impact of php.ini configuration.
- Updated results with new gettime program.
- Added result table.

Subject: Re: Python for your applications
Posted by qapko on Wed, 07 Jan 2009 09:20:22 GMT
View Forum Message <> Reply to Message

luzr wrote on Mon, 05 January 2009 12:30

Should we offer something in U++ for python? Would you contribute?

Mirek

Maybe I can make some example application that will demonstrate binding Python engine with U++ GUI. Or did you mean something like U++/Python bindings package? I thing there is no need to provide it. I use only C++/Python bindings in my U++ projects and personally have no idea on what this U++/Python package should do. Have anyone here looked at PyCXX's web site? I was highly inspired by this package when starting using Python in U++.
If you have any questions on using Python in C++ or U++ I'm ready to give you the answers or at least provide some help.

Gabi

Subject: Re: Python for your applications
Posted by Sender Ghost on Wed, 07 Jan 2009 16:08:23 GMT
View Forum Message <> Reply to Message

luzr wrote on Mon, 05 January 2009 16:30
Should we offer something in U++ for python? Would you contribute?

Mirek
Hello, Mirek.

You can start from reading documentation for Python programming language, e.g. "Embedding Python in Another Application" (v2.6, v3.0). Also this approach works for Lua programming language: "Extending your Application" and Perl: "How to embed perl in your C program".

Subject: Re: Python for your applications
Posted by Mindtraveller on Wed, 07 Jan 2009 22:51:44 GMT
View Forum Message <> Reply to Message

qapko wrote on Wed, 07 January 2009 12:20Maybe I can make some example application that will demonstrate binding Python engine with U++ GUI.
Binding Python with U++ GUI is very interesting. It would be great to see it.

Subject: Re: Python for your applications
Posted by captainc on Thu, 08 Jan 2009 01:15:21 GMT
View Forum Message <> Reply to Message

I have also used embedded Python much in a very large project. It wasn't used with a gui though, just command line. I used the Paramiko SFTP module to provide as SFTP server and client as well as using Python as a way to extend the application's functionality (to perform data transformations). I also used Py2Exe to package the entire interpreter along with the program. My memory footprint was not over 10 MB and performance was good, and it was running in a Virtual Box VM.

I'll post some of the code when I get a chance again.

Just be careful with threading and embedded python, it can get tricky.

Subject: Re: Python for your applications
Posted by qapko on Fri, 23 Jan 2009 09:34:57 GMT
View Forum Message <> Reply to Message

Hello,
I'm sending my ConnSrv application for everybody interested in embedding Python in U++. It's quite simple server used for routing TCP/IP connections on the host PC. It uses threading and saves its configuration in Python source file. I hope you can run it on the PC without Python installed - it uses Python 2.5.2 and some core parts of it are included. It can be run as is, "connsrv.exe" file is in the "tst" directory.
I'm ready to respond if you have any questions about the way it works.

Have a nice day.
Gabi

File Attachments
1) connsrv.zip, downloaded 441 times

Subject: Re: Python for your applications
Posted by qapko on Fri, 23 Jan 2009 09:41:23 GMT
View Forum Message <> Reply to Message

Hi,
I'm sorry - in previous attached "connsrv.zip" there are not "python25.dll" and "mfc71.dll" that are needed for running the application without Python 2.5 installed. So here they are. You must copy them to "tst" directory.

Gabi

File Attachments
1) python_dlls.zip, downloaded 449 times