Introduction

Canonical has been made aware of a sensible performance degradation of the Python 2.7 interpreter on some Openstack workloads since moving to Xenial. (LP #1638695) James Page has also heard about remarks on the fact that Openstack deployments on Xenial were sensibly slower than those on Trusty. This **publicly visible document** tries to identify areas where the Python 2.7 performance on Xenial could be improved.

Overview

While we have not identified a single cause that would explain why the interpreter on Xenial would be slower, it appears that disable Link-Time Optimization (LTO) has a positive impact on performances by either reducing the performance gap, making it almost identical and in a few tests, improving performances. Only a few tests show even more degradation in performances.

Test environment

All tests have been run on bare metal (Intel(R) Xeon(R) CPU E5-2640 0 @ 2.50GHz) with 24 execution threads.

Builds were done in schroots with the following GCC compiler versions :

- Trusty : 4.8.2, Python 2.7-2.7.6
- Xenial : 5.4.0, Python 2.7-2.7.13

The interpreter builds were done using the following commands :

```
$ pull-lp-source python2.7 [trusty|xenial]
$ cd python2.7-*
$ DEB_BUILD_OPTIONS=parallel=$(nproc) DEB_BUILD_OPTIONS="nocheck
nobench" fakeroot debian/rules binary
```

The debian packages were sent to LXC containers and installed as normal packages.

The python performance framework is installed using :

```
$ sudo apt install -y python-pip python-virtualenv python2.7-dev
libpython2.7-dev
```

```
$ python -mpip install performance
```

Tests are run using the following syntax :

```
$ pyperformance run -o {pyversion}_{release}-{context}.json
```

Analysis

After multiple iteration with Matthias Klose, a set of tests was prepared to compare the impact of the **Link Time Optimization (LTO)** and **Profile-Guided Optimization (PGO)** on the performance of the interpreter.

Comparing a Trusty build without PGO to the stock version on Xenial shows that all but two performance benches become slower on Trusty. All the other benchmarks run faster on Xenial when PGO is disabled on Trusty. So we went ahead and prepared the following builds :

- Xenial without PGO
- Xenial without LTO

Those were compared with the Stock versions of the Trusty package in the official archive.

While disabling PGO on Xenial leads to slower benches in all but three benchmarks, disabling LTO either reduces the gap in a sensible number of benchmarks or make some benchmarks run faster :

- 74 % are improved, run faster or no longer have a difference
- 26 % run slower

Here is a breakdown of the changes :

Benchmarks that run faster without LTO :

	2.7.6 Trusty stock .vs 2.7.12 Xenial stock			2.7.6 Trusty stock .vs 2.7.12 Xenial NOLTO		
	Impact	Variance	Δ	Impact	Variance	Δ
### hg_startup ###	slower	(t=-8.53)	1.04	faster	(t=19.81)	1.10
### logging_silent ###	slower	(t=-5.84)	1.09	faster	(t=2.42)	1.04

Benchmarks that no longer have a difference with Trusty (were slower) without LTO :

	2.7.6 Trusty stock .vs 2.7.12 Xenial stock			2.7.6 Trusty stock .vs 2.7.12 Xenial NOLTO		
	Impact	Variance	Δ	Impact	Variance	Δ

### call_method ###	slower	(t=-8.45)	1.08			1.00
### call_method_slots ###	slower	(t=-7.82)	1.06			1.00
### call_method_unknown ###			1.01			1.02
### call_simple ###	slower	(t=-3.41)	1.05			1.02
### meteor_contest ###	slower	(t=-5.15)	1.04			1.01
### nbody ###	slower	(t=-14.45)	1.13			1.02
### sqlalchemy_declarative ###	slower	(t=-4.38)	1.03			1.02

Slower benchmarks improved by disabling LTO :

	2.7.6 Trusty stock .vs 2.7.12 Xenial stock			2.7.6 Trusty stock .vs 2.7.12 Xenial NOLTO		
	Impact	Variance	Δ	Impact	Variance	Δ
### chameleon ###	slower	(t=-9.65)	1.15	slower	(t=-9.50)	1.13
### chaos ###	slower	(t=-17.26)	1.20	slower	(t=-14.04)	1.18
### crypto_pyaes ###	slower	(t=-3.36)	1.09	slower	(t=-2.79)	1.11
### django_template ###	slower	(t=-20.92)	1.13	slower	(t=-13.59)	1.10
### fannkuch ###	slower	(t=-39.81)	1.13	slower	(t=-5.97)	1.02
### float ###	slower	(t=-18.75)	1.18	slower	(t=-18.24)	1.19
### genshi_text ###	slower	(t=-30.07)	1.28	slower	(t=-17.61)	1.22
### genshi_xml ###	slower	(t=-6.26)	1.07	slower	(t=-5.02)	1.07
### hexiom ###	slower	(t=-12.78)	1.17	slower	(t=-5.71)	1.07
### html5lib ###	slower	(t=-12.22)	1.10	slower	(t=-7.64)	1.07
### json_loads ###	slower	(t=-5.56)	1.09	slower	(t=-4.39)	1.04
### logging_format ###	slower	(t=-17.84)	1.21	slower	(t=-15.65)	1.25
### logging_simple ###	slower	(t=-8.38)	1.08	slower	(t=-5.62)	1.09
### mako ###	slower	(t=-5.06)	1.10	slower	(t=-4.28)	1.02
### nqueens ###	slower	(t=-10.40)	1.17	slower	(t=-4.44)	1.08
### regex_compile ###	slower	(t=-146.20)	1.18	slower	(t=-133.42)	1.15
### scimark_fft ###	slower	(t=-115.03)	1.18	slower	(t=-63.89)	1.09
### scimark_monte_carlo ###	slower	(t=-11.41)	1.17	slower	(t=-5.54)	1.08
### scimark_sor ###	slower	(t=-81.49)	1.16	slower	(t=-20.80)	1.06
### spambayes ###	slower	(t=-3.43)	1.05	slower	(t=-2.16)	1.04
### spectral_norm ###	slower	(t=-7.81)	1.12	slower	(t=-3.73)	1.06
### sympy_expand ###	slower	(t=-78.89)	1.12	slower	(t=-46.19)	1.08
### sympy_str ###	slower	(t=-42.48)	1.11	slower	(t=-15.95)	1.07
### sympy_sum ###	slower	(t=-11.17)	1.09	slower	(t=-7.86)	1.04
### telco ###	slower	(t=-87.57)	1.10	slower	(t=-62.27)	1.06
### unpickle_list ###			1.02	slower	(t=-3.14)	1.06
### unpickle_pure_python ###	slower	(t=-7.34)	1.12	slower	(t=-6.37)	1.11
### xml_etree_generate ###	slower	(t=-15.41)	1.12	slower	(t=-10.75)	1.09

Benchmarks that runs slower without LTO :

	2.7.6 Trusty stock .vs 2.7.12 Xenial stock			2.7.6 Trusty stock .vs 2.7.12 Xenial NOLTO		
	Impact	Variance	Δ	Impact	Variance	Δ
### 2to3 ###	slower	(t=-22.01)	1.14	slower	(t=-33.47)	1.12
### go ###	slower	(t=-74.78)	1.13	slower	(t=-83.41)	1.11
### hg_startup ###	slower	(t=-8.53)	1.04	faster	(t=19.81)	1.10
### json_dumps ###	slower	(t=-9.06)	1.10	slower	(t=-12.62)	1.11
### pathlib ###	slower	(t=-2.14)	1.11	slower	(t=-11.03)	1.26
### pickle ###	slower	(t=-10.89)	1.12	slower	(t=-18.10)	1.15
### pickle_dict ###		0	1.01	slower	(t=-19.55)	1.17
### pickle_list ###		0	1.03	slower	(t=-8.95)	1.10
### pickle_pure_python ###	slower	(t=-3.71)	1.04	slower	(t=-5.76)	1.10
### pidigits ###	slower	(t=-12.78)	1.12	slower	(t=-13.49)	1.13
### pyflate ###	slower	(t=-42.02)	1.15	slower	(t=-50.72)	1.11
### raytrace ###	slower	(t=-77.91)	1.24	slower	(t=-205.37)	1.23
### regex_effbot ###	slower	(t=-5.92)	1.05	slower	(t=-10.81)	1.09
### richards ###	faster	(t=-2.61)	1.07	slower	(t=-5.07)	1.05
### scimark_lu ###	slower	(t=-34.99)	1.25	slower	(t=-64.57)	1.19
### scimark_sparse_mat_mult ###	slower	(t=-10.26)	1.13	slower	(t=-34.03)	1.34
### unpickle_list ###		0	1.02	slower	(t=-3.14)	1.06
### xml_etree_iterparse ###		0	1.04	slower	(t=-2.50)	1.05

The complete result numbers can be seen in the Complete test results of the PGO/LTO comparison spreadsheet.

Profiling analysis

In order to try to understand the differences induced by the LTO and PGO optimization, we have run profiling on two benchmarks that no longer show a difference when the LTO optimization is turned off on Xenial. The profiling was done on :

- Trusty stock interpreter
- Trusty interpreter built with PGO disabled
- Trusty interpreter built with LTO disabled
- Xenial stock interpreter
- Xenial interpreter built with PGO disabled
- Xenial interpreter built with LTO disabled

The increase in the number of instructions when PGO is disabled is much more noticeable on Trusty for both benchmarks. We also see that the profiling that is the closest to the Trusty stock is when LTO is disabled on Xenial

	call_method	Δ incr. From Trusty	tornado_http	Δ incr. From Trusty
Trusty Stock	9249989848		4466075788	
Trusty noPGO	10336086734	**11.74 %**	5071025478	**13.55 %**
Trusty noLTO	9596171515	3.74 %	4679410729	4.78 %
Xenial Stock	10138757758	9.61 %	4846064277	8.51 %
Xenial noPGO	10138756042	9.61 %	4846048609	8.51 %
Xenial noLTO	9507074251	**2.78 %**	4623572401	**3.53 %**

Details of the profiling for each of these combinations can be found at the end of the document.

Conclusion

At the time of writing, we do not have an explanation on the reasons why the 2.7.2-2.7.13 python interpreter currently available in Xenial has worse performances than the one in Trusty. The PGO optimization done on Trusty could be more efficient, since when disabling this optimization, Trusty becomes less performant than Xenial.

Factual evidence from the runs of the **pyperformance** bench shows that the majority of the benchmarks do perform better when the LTO optimization is disabled. This is not systematic but almost 75% of the benchmarks gain in performance over the current python 2.7 interpreter currently available on Xenial.

Profiling results

The profiling data was collected using the callgrind tool of the valgrind toolset. A typical command use would be :

```
valgrind --tool=callgrind --trace-children=yes
~/venv/cpython2.7-d8da6ef977cc/bin/python2.7 -m performance run
-bcall_method --inside-venv
```

Call_method benchmark

Trusty

Trusty Stock		Trusty noPGO		Trusty noLTO	
Instructions	Function	Instructions	Function	Instructions	Function
4918198605	PyEval_EvalFrameEx'2	5362602828	PyEval_EvalFrameEx'2	4930764066	PyEval_EvalFrameEx'2
1180507700	0x00000000005368f0	1250195637	0x0000000000585e90	1059123292	0x0000000000486370
1109707368	PyObject_GetAttr	890479191	PyFrame_New	879185389	PyObject_GetAttr
823065734	PyFrame_New	836574419	PyObject_GenericGetAttr	823049135	PyFrame_New
552755137	0x00000000004a5c90	552808267	0x000000000049ef00	552712907	0x0000000000477410
525836692	0x00000000004bedf0	539318922	0x0000000000493710	539326714	PyMethod_New
12732711	PyParser_AddToken	488401927	_PyType_Lookup	472694320	_PyType_Lookup
6120934	PyDict_GetItem	258028053	PyObject_GetAttr	148897549	PyObject_GC_UnTrack
4700333	0x00000000004bc0e0	12247026	0x00000000005937f0	40448799	0x00000000004877d0
4647564	0x00000000004afe90'2	11727983	PyParser_AddToken	10899345	PyParser_AddToken
3724240	PyObject_Free	6409083	PyDict_GetItem	10348722	PyDict_GetItem
3526112	PyDict_SetItem	4948165	PyObject_Malloc	4662730	0x00000000004cd110
3407575	0x0000000000571fd0	4746403	0x00000000004f7b70'2	4637942	0x000000000050b720'2
3304198	PyObject_Hash	4525638	PyNode_AddChild	4432451	PyObject_Free
3055436	0x00000000005495a0	4429698	PyObject_Free	4094020	PyDict_SetItem
2796306	0x0000000000535070	3582953	0x000000000041aff0	3629015	PyNode_AddChild
		3240230	PyObject_Hash	3408211	0x00000000004a2260
				3232715	PyObject_Malloc
				3206499	PyObject_Hash
				3055436	0x000000000050b610
9249989848		10336086734		9596171515	

Xenial

Xenial Stock		Xenial noPGO		Xenial noLTO	
Instructions	Function	Instructions	Function	Instructions	Function
5293380177	PyEval_EvalFrameEx'2	5293380790	PyEval_EvalFrameEx'2	4638372768	PyEval_EvalFrameEx'2
1058376436	0x00000000004d9030	1058376436	0x0000000000585e90	1032105069	0x0000000000486370
850012013	PyFrame_New	850012013	PyFrame_New	930988654	PyObject_GetAttr
823053334	_PyObject_GenericGetAttrWithDict	823053388	PyObject_GenericGetAttr	836521421	PyFrame_New
525847184	PyMethod_New	525847184	0x000000000049ef00	539330714	0x0000000000477410
471794110	0x00000000004e1030	471794110	0x0000000000493710	471820360	PyMethod_New
461203120	_PyType_Lookup	461203154	_PyType_Lookup	459027697	_PyType_Lookup
312611371	PyObject_GetAttr	312611394	PyObject_GetAttr	231463205	PyObject_GC_UnTrack
148905231	PyObject_GC_UnTrack	148905242	0x00000000005937f0	148905033	0x00000000004877d0
40444971	0x00000000004e09c0	40444971	PyParser_AddToken	40447959	PyParser_AddToken
12120701	PyDict_GetItem	12120701	PyDict_GetItem	26987922	PyDict_GetItem
10987593	PyParser_AddToken	10987593	PyObject_Malloc	10987547	0x00000000004cd110
6574452	PyObject_Malloc	6569654	0x00000000004f7b70'2	10817485	0x000000000050b720'2
5006318	0x00000000004bdc60'2	5006318	PyNode_AddChild	6371045	PyObject_Free
4842496	PyDict_SetItem	4842666	PyObject_Free	4810558	PyDict_SetItem
4519155	PyDict_Next	4519155	0x000000000041aff0	4434281	PyNode_AddChild
3835459	PyObject_Free	3835456	PyObject_Hash	4103768	0x00000000004a2260
3270809	PyObject_Hash			3779149	PyObject_Malloc
3257249	PyNode_AddChild			3406572	PyObject_Hash
				3363736	0x000000000050b610
10138757758		10138756042		9507074251	

Tornado_http benchmark

Trusty

Trusty Stock		Trusty noPGO		Trusty noLTO	
Instructions	Function	Instructions	Function	Instructions	Function
1320294609	PyEval_EvalFrameEx'2	1371457976	PyEval_EvalFrameEx'2	1299680772	PyEval_EvalFrameEx'2
323040823	PyObject_GetAttr	386292180	0x0000000005937f0	295910797	PyDict_GetItem
260442702	PyDict_GetItem	238083036	_PyType_Lookup	243986925	PyObject_GetAttr
153555524	0x0000000005368f0	215228199	PyObject_GenericGetAttr	195617523	_PyType_Lookup
135777548	0x0000000000535070	181639902	PyEval_EvalCodeEx'2	165398776	PyEval_EvalCodeEx'2
124016854	PyEval_EvalCodeEx'2	181167997	PyDict_GetItem	140421739	0x0000000000486370
115765355	0x000000000053a740	165016601	0x0000000000585e90	123049394	0x00000000004c1250
115091581	PyObject_GenericSetAttr	140713310	0x0000000000585a20	112130233	0x00000000004a0650
113029910	PyFrame_New	106981401	PyFrame_New	104918436	PyFrame_New
71175250	PyTuple_New	84603895	PyObject_GetAttr	98640611	PyDict_SetItem
68938352	PyString_FromFormatV	73924922	PyDict_SetItem	77597623	PyObject_SetAttr
58004336	_PyType_Lookup	73705581	PyTuple_New	72610634	PyTuple_New
51218282	/build/eglibc-..._memcpy_sse2_unaligned	68869545	PyString_FromFormatV	69402995	PyString_FromFormatV
49049259	0x0000000000568850'2	65249860	PyObject_GenericSetAttr	51465761	/build/eglibc-..._memcpy_sse2_unaligned
45554925	PyObject_SetAttr	45004103	0x000000000049ef00	43886449	0x0000000000477410
44930268	0x00000000004a5c90	44485698	0x00000000004181c8	43347698	PyObject_Free
42356835	0x0000000000571fd0	44339024	0x000000000041aff0	43295505	PyMethod_New
42155692	0x00000000004bedf0	43325872	PyObject_Free	42356829	0x00000000004a2260
39006304	PyCFunction_Call	43114865	/build/eglibc-..._memcpy_sse2_unaligned	41778364	PyObject_GC_UnTrack
38476239	/build/eglibc-...:memset	34898824	0x000000000057c620	38679170	/build/eglibc-...memset
37723384	PyObject_Hash	33623779	0x000000000053c970	36741093	PyObject_Hash
34076833	PySys_SetObject	33073764	/build/eglibc..._malloc	36716624	PyObject_Malloc
33188645	0x000000000054d870	31123485	PyObject_CallFunctionObjArgs	33001042	/build/eglibc-..._malloc
32988753	/build/eglibc-..._malloc	(lower values truncated)		32426504	0x0000000000529be0
(lower values truncated)				30320307	PyString_InternInPlace
				(lower values truncated)	
4466075788		5071025478		4679410729	

Xenial

Xenial Stock		Xenial noPGO		Xenial noLTO	
Instructions	Function	Instructions	Function	Instructions	Function
1391908925	PyEval_EvalFrameEx'2	1391906488	PyEval_EvalFrameEx'2	1199789852	PyEval_EvalFrameEx'2
285996948	PyDict_GetItem	285996879	PyDict_GetItem	317119225	0x00000000004d5100
279457640	PyEval_EvalCodeEx'2	279457438	PyEval_EvalCodeEx'2	245447198	_PyObject_GenericGetAttrWithDict
227086985	_PyObject_GenericGetAttrWithDict	227086769	_PyObject_GenericGetAttrWithDict	177181631	_PyType_Lookup
206168967	_PyType_Lookup	206168797	_PyType_Lookup	162996902	PyEval_EvalCodeEx'2
117220545	PyDict_SetItem	117220428	PyDict_SetItem	161825355	PyDict_GetItem
108069053	PyFrame_New	108068990	PyFrame_New	134864577	0x00000000004baed0
106247784	0x00000000004a04c0	106247511	0x00000000004a04c0	119819967	0x00000000004ef120
97498764	PyObject_GetAttr	97498672	PyObject_GetAttr	106620349	PyFrame_New
82903921	PyTuple_New	82903641	PyTuple_New	78177016	PyObject_GetAttr
66985448	PyString_FromFormatV	66985448	PyString_FromFormatV	77467845	PyDict_SetItem
65534475	0x00000000004b50a0	65534135	0x00000000004b50a0	71827339	PyTuple_New
61017767	_PyObject_GenericSetAttrWithDict	61017767	_PyObject_GenericSetAttrWithDict	69673240	PyString_FromFormatV
53349989	PyObject_Malloc	53350010	PyObject_Malloc	55007051	PyObject_GenericSetAttr
51321098	/build/glibc-...memcpy_sse2_unaligned	51320959	/build/glibc-...memcpy_sse2_unaligned	51206180	/build/glibc...memcpy_sse2_unaligned
44754003	PyDict_Next	44754003	PyDict_Next	44012369	0x00000000004d66a0
43372897	PyObject_CallFunctionObjArgs	43372773	PyObject_CallFunctionObjArgs	43088635	PyMethod_New
42572988	PyMethod_New	42572988	PyMethod_New	41790020	PyObject_Free
41382319	PyObject_GC_UnTrack	41382209	PyObject_GC_UnTrack	41382616	PyObject_GC_UnTrack
38667392	0x000000000055b8c0	38666831	0x000000000055b8c0	38454048	PyObject_SetAttr
38454192	PyObject_SetAttr	38454192	PyObject_SetAttr	37826355	/build/glibc-..._memset
37836729	PyObject_Free	37836382	PyObject_Free	37230569	0x00000000004ab110
37820442	/build/glibc-...memset	37820336	/build/glibc-...memset	36819380	PyObject_Malloc
35217932	PyObject_Hash	35217874	PyObject_Hash	34867758	PyObject_Hash
35124858	PyCFunction_Call	35124494	PyCFunction_Call	34590577	0x000000000054f120
34372905	PyString_InternInPlace	34372833	PyString_InternInPlace	33693018	/build/glibc-..._malloc
33935579	PyObject_Call'2	33935467	PyObject_Call'2	30763478	PyObject_CallFunctionObjArgs
33812803	/build/glibc-..._malloc	33810248	/build/glibc-..._malloc	30332058	PyString_InternInPlace
32521809	0x00000000004e1030	32521809	0x00000000004e1030	(lower values truncated)	
(lower values truncated)		(lower values truncated)			
4846064277		4846048609		4623572401	

More comparative tests can be found in the following spreadsheet which compares different 2.7.6 and 2.7.12 builds using gcc 4.8.2 and gcc 5.4