PyCon China 2020

PyConChina 2020
PYTHON 中国开发者大会 2020

Python For Good

基数树路由

Aber

社区环境背景

最新编程语言热度排名榜单

Nov 2020	Nov 2019	Change	Programming Language	Ratings	Change
1	2	↑	C	16.21%	+0.17%
2	**3**	↑	**Python**	**12.12%**	**+2.27%**
3	1	↓	Java	11.68%	-4.57%
4	4		C++	7.60%	+1.99%
5	5		C#	4.67%	+0.36%
6	6		Visual Basic	4.01%	-0.22%
7	7		JavaScript	2.03%	+0.10%
8	8		PHP	1.79%	+0.07%
9	16	⇈	R	1.64%	+0.66%
10	9	↓	SQL	1.54%	-0.15%
11	14	↑	Groovy	1.51%	+0.41%
12	21	⇈	Perl	1.51%	+0.68%
13	20	⇈	Go	1.36%	+0.51%
14	10	⇊	Swift	1.35%	-0.31%
15	11	⇊	Ruby	1.22%	-0.04%
16	15	↓	Assembly language	1.17%	+0.14%
17	19	↑	MATLAB	1.10%	+0.21%
18	13	⇊	Delphi/Object Pascal	0.86%	-0.28%
19	12	⇊	Objective-C	0.84%	-0.35%
20	32	⇈	Transact-SQL	0.82%	+0.44%

社区环境背景

性能优化

- The Python interpreter now uses a 1 optimizations possible. (Contributed Victor Stinner in bpo-26647 and bpo-
- The `asyncio.Future` class now has INADA Naoki in bpo-26081.)
- The `asyncio.Task` class now has an 28544.)
- Various implementation improvement 30 times performance improvements
- The ASCII decoder is now up to 60 ti (Contributed by Victor Stinner in bpo-
- The ASCII and the Latin1 encoders (Contributed by Victor Stinner in bpo-
- The UTF-8 encoder is now up to 75 `surrogatepass` (Contributed by Victor
- The UTF-8 decoder is now up to 15 t (Contributed by Victor Stinner in bpo-
- `bytes % args` is now up to 2 times fa
- `bytearray % args` is now between 2.
- Optimize `bytes.fromhex()` and by (Contributed by Victor Stinner in bpo-
- Optimize `bytes.replace(b'', b'.')` by Josh Snider in bpo-26574).
- Allocator functions of the `PyMem_Mal` allocator instead of `malloc()` functi smaller or equal to 512 bytes wit (Contributed by Victor Stinner in bpo-
- `pickle.load()` and `pickle.loads()` (Contributed by Victor Stinner in bpo-
- Passing keyword arguments to a arguments. Now in extension func significantly decreased. (Contributed
- Optimized `glob()` and `iglob()` fun (Contributed by Serhiy Storchaka in b
- Optimized globbing in `pathlib` by us Serhiy Storchaka in bpo-26032).
- `xml.etree.ElementTree` parsing, iter (Contributed by Serhiy Storchaka in b
- Creation of `fractions.Fraction` insta by Serhiy Storchaka in bpo-25971.)

性能优化

通过移植更多代码来使用 `METH_FASTCALL` 的
开销。 （由 Victor Stinner 在 bpo-29300、b

通过各种优化方式，使 Python 在 Linux 上
Stinner, INADA Naoki 在 bpo-29585 中，以

由于避免创建绑定方法案例的字节码更改，
在 bpo-26110 中贡献。）

对 `asyncio` 模块里面的一些常用函数做了显

- `asyncio.get_event_loop()` 函数已经
  在 bpo-32296 中贡献。）
- `asyncio.Future` 回调管理已经过优化。
- `asyncio.gather()` 的执行速度现在加
- 当 *delay* 参数为零或负值时 `asyncio.s`
  32351 中贡献。）
- asyncio 调试模式的执行开销已获减轻。

作为 PEP 560 工作 的结果，`typing` 的导入
Ivan Levkivskyi 在 bpo-32226 中贡献。）

`sorted()` 和 `list.sort()` 已经过优化，在通
28685 中贡献。）

`dict.copy()` 的执行速度现在加快了 5.5 倍。

当 *name* 未找到并且 *obj* 未重载 `objec`
`getattr()` 现在会比原来快大约 4 倍。 （由

在字符串中搜索特定的 Unicode 字符（例如
也只会慢上 3 倍。 （由 Serhiy Storchaka 在

`collections.namedtuple()` 工厂对象已经
Zijlstra 在 bpo-28638 中贡献，进一步的改进

现在 `date.fromordinal()` 和 `date.fromtime`
bpo-32403 中贡献。）

由于使用了 `os.scandir()`，现在 `os.fwalk()`
贡献。）

由于使用了 `os.scandir()` 函数，`shutil.rm`
在 bpo-28564 中贡献。）

性能优化

- `subprocess` 模块现在能在某些情况下使用
  macOS 和 Linux（使用 glibc 2.24 或更新版
  - *close_fds* 为假值；
  - *preexec_fn*, *pass_fds*, *cwd* 和 *start_n*
  - *executable* 路径包含一个目录。

（由 Joannah Nanjekye 和 Victor Stinner 在
- `shutil.copyfile()`, `shutil.copy()`, `shuti`
  macOS 上会使用平台专属的 "fast-copy" 系
  从而避免在进行 "`outfd.write(infd.read()`
  `shutil.copyfile()` 会使用更大的默认缓
  `shutil.copyfileobj()` 版本。 在同一分区
  macOS 上为 +50%，在 Windows 上为 +40
  效拷贝操作 一节。 （由 Giampaolo Rodolà
- `shutil.copytree()` 会根据其所用缓存的 `os`
  包含 8000 文件的目录的速度提升在 Linux
  享目录则为 +30%。 此外 `os.stat()` 系统
  系统上会特别快速。 （由Giampaolo Rodol
- `pickle` 模块使用的默认协议现在为 Protocc
  可用的 Protocol 3 更好的性能和更小的数据
- 从 `PyGC_Head` 移除了一个 `Py_ssize_t` 成员。
  字节。 （由 Inada Naoki 在 bpo-33597 中
- `uuid.UUID` 现在会使用 `__slots__` 以减少内
  贡献。）
- `operator.itemgetter()` 的性能提升了 33%
  况新增了一条快速路径（这是标准库中的典
- 加快了在 `collections.namedtuple()` 中的
  的实例变量查找形式。 （由 Raymond He
  32492 中贡献。）
- 如果输入的可迭代对象的长度已知 (即输入对
  这使得所创建的列表资源占用平均减少了 1
  中贡献。）
- 类变量写入速度加倍。 当一个非冗余属性
  Behnel, Pablo Galindo Salgado, Raymon
  36012 中贡献。）
- 减少了传递给许多内置函数和方法的参数
  -50%。 （由 Serhiy Storchaka 在 bpo-2386
- `LOAD_GLOBAL` 指令现在会使用新的 "per opc
  Selivanov 和 Inada Naoki 在 bpo-26219 中

性能优化

- 优化了在推导式中为临时变量赋值的惯用方式。 现在推导式中的 `for y in [expr]` 会与简单赋值语句 `y = expr` 一样快速。 例如:

  ```
  sums = [s for s in [0] for x in data for s in [s + x]]
  ```

  不同于 `:=` 运算符，这个惯用方式不会使变量泄露到外部作用域中。

  （由 Serhiy Storchaka 在 bpo-32856 中贡献。）
- 优化了多线程应用中的信号处理。 如果一个线程不是获得信号的主线程，字节码求值循环不会在每条字节码指令上被打断以检查无法被处理的挂起信号。 只有主解释器的主线程能够处理信号。

  在之前版本中，字节码求值循环会在每条指令上被打断直到主线程处理了信号。 （由 Victor Stinner 在 bpo-40010 上贡献。）
- 在 FreeBSD 上使用 `closefrom()` 优化了 `subprocess` 模块。 （由 Ed Maste, Conrad Meyer, Kyle Evans, Kubilay Kocak 和 Victor Stinner 在 bpo-38061 中贡献。）
- `PyLong_FromDouble()` 对于匹配 `long` 的值执行速度现在加快了 1.87 倍。 （由 Sergey Fedoseev d bpo-37986 中贡献。）
- 多个 Python 内置类型 (`range`, `tuple`, `set`, `frozenset`, `list`, `dict`) 现在通过使用 **PEP 590** 向量调用协议得到加速。 （由 Dong-hee Na, Mark Shannon, Jeroen Demeyer 和 Petr Viktorin 在 bpo-37207 中贡献。）
- 当另一集合远大于基础集合的情况下优化了 `difference_update()` 的性能。 （由 Evgeny Kapun 提议，由 Michele Orrù 在 bpo-8425 中贡献代码。）
- Python 的小对象分配器 (`obmalloc.c`) 现在允许（至多）一个空位可用于立即重用，而不必将其返回给 OS。 这可以防止简单循环中的多余消耗，在每次迭代中可以创建和销毁全新的空位。 （由 Tim Peters 在 bpo-37257 中贡献。）
- 浮点数运算中的 floor division 现在会有更好的性能。 并且此运算的 `ZeroDivisionError` 的消息也已更新。 （由 Dong-hee Na 在 bpo-39434 中贡献。）
- 使用 UTF-8 和 ascii 编解码器解码短 ASCII 字符串现在会加快大约 15%。 （由 Inada Naoki 在 bpo-37348 中贡献。）

以下是对从 Python 3.4 到 Python 3.9 的提升提升情况的总结:

```
Python version                      3.4     3.5     3.6     3.7     3.8     3.9
--------------                      ---     ---     ---     ---     ---     ---

Variable and attribute read access:
    read_local                      7.1     7.1     5.4     5.1     3.9     3.9
    read_nonlocal                   7.1     8.1     5.8     5.4     4.4     4.5
    read_global                    15.5    19.0    14.3    13.6     7.6     7.8
    read_builtin                   21.1    21.6    18.5    19.0     7.5     7.8
    read_classvar_from_class       25.6    26.5    20.7    19.5    18.4    17.9
    read_classvar_from_instance    22.8    23.5    18.8    17.1    16.4    16.9
    read_instancevar               32.4    33.1    28.0    26.3    25.4    25.3
    read_instancevar_slots         27.8    31.3    20.8    20.8    20.2    20.5
    read_namedtuple                73.8    57.5    45.0    46.8    18.4    18.7
```

新的挑战

WSGI	Django、Bottle、Flask……
ASGI	Starlette（FastAPI）、Quart……
HTTP	aiohttp、Sanic……

Fastest?

路由代码样例


```python
urlpatterns = [
    path('', homepage),
    path('about', about),
]
```

Django

```python
routes = [
    Route("/", endpoint=homepage),
    Route("/about", endpoint=about),
]
```

Starlette

```python
@app.route("/")
def homepage():
    return "Hello, world"

@app.route("/about")
def about():
    return "about page"
```

Bottle/Flask

Or More framework

传统匹配方式

Request

r"/"	...
r"/user/(?P<username>[^/]+)"	...
r"/user/(?P<username>[^/]+)/profile"	...
r"/about"	...
r"/sitemap.xml"	...
r"/article/(?P<slug[^/]+>)"	...
.....	...

其他语言的Web框架

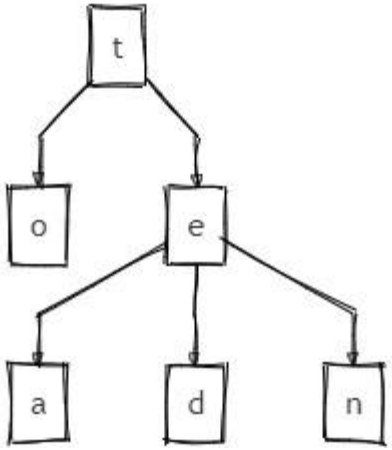
Web Frameworks based on HttpRouter

If the HttpRouter is a bit too minimalistic for you, you might try one of the following more high-level 3rd-party web frameworks building upon the HttpRouter package:

- Ace: Blazing fast Go Web Framework
- api2go: A JSON API Implementation for Go
- Gin: Features a martini-like API with much better performance
- Goat: A minimalistic REST API server in Go
- goMiddlewareChain: An express.js-like-middleware-chain
- Hikaru: Supports standalone and Google AppEngine
- Hitch: Hitch ties httprouter, httpcontext, and middleware up in a bow
- httpway: Simple middleware extension with context for httprouter and a server with gracefully shutdown support
- kami: A tiny web framework using x/net/context
- Medeina: Inspired by Ruby's Roda and Cuba
- Neko: A lightweight web application framework for Golang
- pbgo: pbgo is a mini RPC/REST framework based on Protobuf
- River: River is a simple and lightweight REST server
- siesta: Composable HTTP handlers with contexts
- xmux: xmux is a httprouter fork on top of xhandler (net/context aware)

.....

前缀树

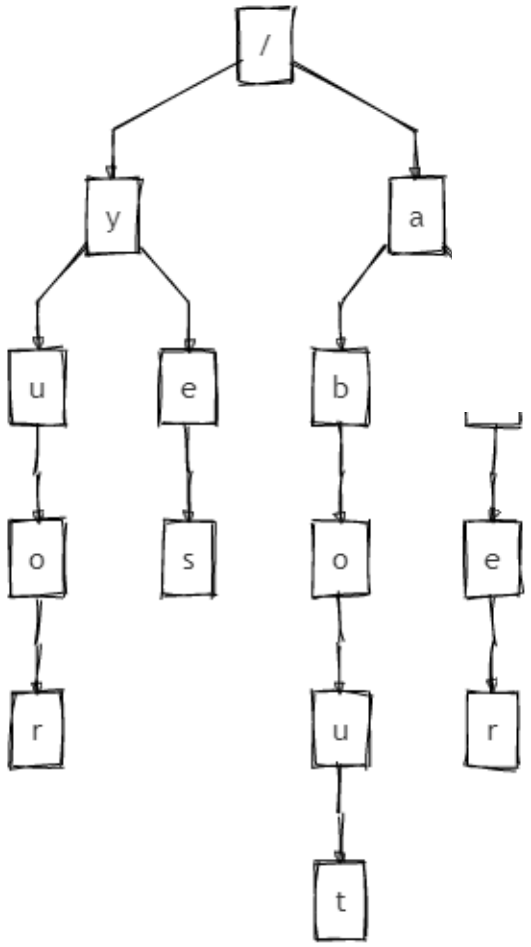


包含四个单词(to/tea/ted/ten)的前缀树

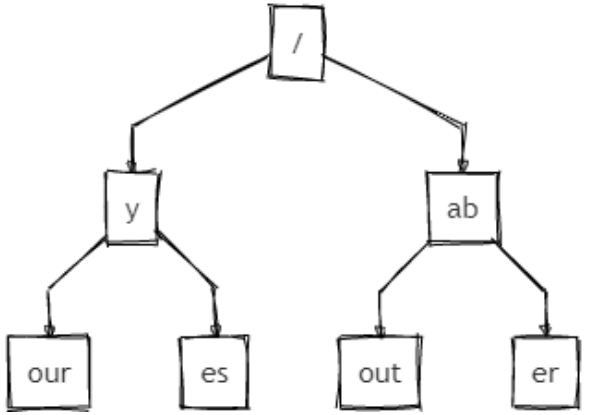
在计算机科学中，前缀树(trie)又称字典树，是一种有序树，用于保存关联数组，其中的键通常是字符串。

前缀树常用于搜索提示。
如当输入一个或多个字符，可以自动搜索出可能的选择。当没有完全匹配的搜索结果，可以返回前缀最相似的可能。

前缀树



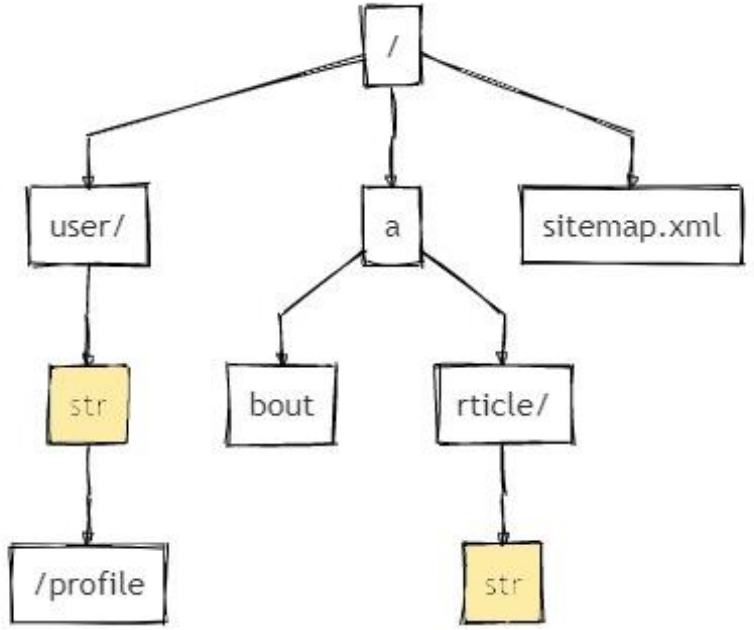
前缀树



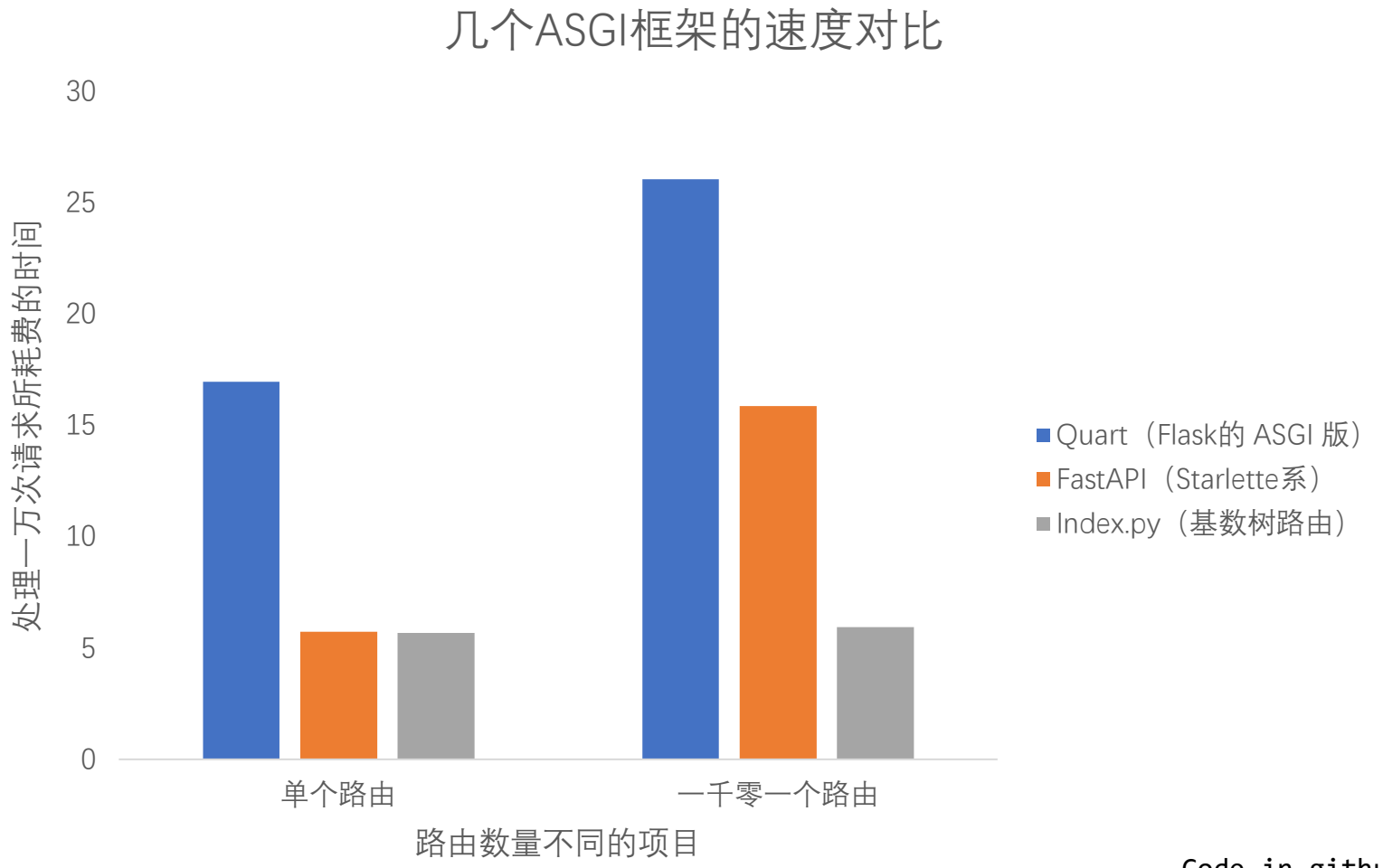
压缩前缀树(基数树)

基数树路由

r"/"
r"/user/(?P<username>[^/]+)"
r"/user/(?P<username>[^/]+)/profile"
r"/about"
r"/sitemap.xml"
r"/article/(?P<slug[^/]+>)"
……



实际项目速度对比



Code in github.com/abersheeran/asgi-benchmark

Fast web framework

THANK YOU