Journal : Pythran - 0.9.0 - kozhamzer

Posté par **serge_sans_paille** (**page perso**) le 07/11/18 à 07:22. **Licence CC by-sa.**
Tags : **pythran**

Petite mise en bouche en alexandrin pour annoncer une nouvelle version de Pythran, compilateur pour le Python scientifique.

> *L'automne arrive, avec son lot de feuilles mortes*
> *Pythran jamais ne dérive, il fallait qu'il sorte*
> *Pas de grande nouveauté, juste le nécessaire*
> *Tout est vectorisé, vraiment il faut le faire !*

Le logiciel est comme toujours disponible sur **pypi**, **anaconda** et **Github**.

Comme annoncé dans les vers ci-dessus, la grande nouveauté vient du passage de **boost.simd** à **xsimd**. J'ai détaillé dans un billet un peu plus long que cette annonce les raisons et les implications de ce passage, si ça vous intéresse, c'est sur **pythran stories**, site dont je vous recommande bien évidement la lecture.

Bonne nouvelle, la sauce commence à prendre ! Je reçois de plus en plus de demandes de correctifs de bugs. Parait-il que c'est bon signe :-) Et puis j'ai reçu il y a peu ce mail qui ne peut que flatter le nombril

> *Hi Serge,*
>
> *We work in Facebook's Stream Processing team, and we rely on Pythran to compile source scripts of streaming applications written in Python into C++ which is executed in the runtime engine.*

Pythran serait donc une brique du flux de traitement décrit dans **ce papier**, c'est assez fou !

Et sous l'impulsion de **@wolfv** est né un **petit bout de code** qui permettra peut-être un jour d'appeler Pythran comme compilateur à la volée, à la numba, avec un `@pythran.jit`. Pythran n'est vraiment pas taillé pour ça question temps de compilation, mais est-ce grave ?

Appel à contribution

Posté par **Nonolapéro** le 07/11/18 à 07:49. Évalué à 4 (+2/-0).

Puisque ça parle de Python pour la science. Je fais un peu de pub pour la dépêche en cours de rédaction sur le thème. Pour contribuer ça se passe dans ce coin_r-> **https://linuxfr.org/redaction/news/python-pour-les-sciences**

Expliciter l'intérêt

Posté par **freejeff** le 07/11/18 à 09:03. Évalué à 6 (+4/-0). Dernière modification le 07/11/18 à 09:04.

Salut,

J'ai du bien lire le pythran stories pour comprendre l'intérêt de ce travail et je pense que c'est vraiment intéressant.

Par exemple sur ce bout de code qui est entièrement vectorisé, qui devrait donc être optimal pour l'utilisation de numpy :

```
import numpy as np
def normalize_complex_arr(a):
    a_oo = a - a.real.min() - 1j*a.imag.min() # origin offsetted
    return a_oo/np.abs(a_oo).max()
```

Je ne me serais pas attendu à une amélioration due à l'utilisation de pythran ; en effet ; on est dans le cas où l'on utilise des fonctions purement numpy comme : np.complex,np.min,np.max,np.abs. Il n'y a aucune boucle et aucune condition, on pourrait donc se dire que compiler ce programme n'est absolument pas nécessaire !!

Et pourtant ...

Temps execution :
Numpy : 3.19 ms
pythran sans vectorisation : 2.84 ms --> c'est déjà étonnant !
pythran avec vectorisation : 723 us soit un speedup de 4.4x !

Et je pense qu'ici, c'est juste l'aspect vectorisé même pas de parallélisation ! Je me demande quel serait le résultat en utilisant OpenMP sur un simple quad cœur.

Peux tu expliquer pourquoi il y a un speed-up sur pythran sans vectorisation ? Est-ce du au fait que tu fais moins de tests de dépassements pour les slices ?

Du coup, on passe de : je fais des choses compliquées, il faudrait que je fasse une lib avec pythran à Quoi que je fasse qui prenne du temps cela vaut toujours le coup de le faire avec pythran !

Honnêtement, je n'avais pas intégré cela. C'est vraiment révolutionnaire. Cela place python devant MatLab en terme de simplicité.

Pour rappel depuis de très longues années MATLab a permis de nettes améliorations de performances en utilisant la simple commande "mcc -m hello.m -a ./testdir", il était possible d'améliorer considérablement la vitesse de son programme, et cela était bien plus difficile sous python, il fallait utiliser cython, ce qui revenait à récrire totalement son programme. La numba et pythran sont arrivés, il permettent tous les deux des améliorations hallucinantes des performances avec des approches différentes. Pythran est beaucoup plus proche de ce que fait MATLab avec mcc et numba avec ses décorateurs et sa compilation à la volée permet de se poser encore moins de questions.

On arrive donc aujourd'hui à un écosystème aussi performant (voir plus sur certains points) qu'un monstre sacré dans le domaine scientifique comme MATLab.

C'est franchement une excellente nouvelle !

Peut être que dans quelques années il y aura une implémentation numpy API/ABI compatible purement en pythran !

Bon par contre On va tous t'en vouloir, car tu fais diminuer nettement le temps des pauses café !

Franchement, je suis vraiment scotché !