Шляхі акселерацыі выканання вылічальнай нагрузкі на Python

Антон Літвіненка,
Кіеў, Украіна

Сімвальная матрыца спін-гамільтаніяна для асі *z* для комплексу Cu_2

$$
\left(\begin{array}{c|cccc}
z & |+\frac{1}{2};+\frac{1}{2}\rangle & |+\frac{1}{2};-\frac{1}{2}\rangle & |-\frac{1}{2};+\frac{1}{2}\rangle & |-\frac{1}{2};-\frac{1}{2}\rangle \\
\hline
\langle+\frac{1}{2};+\frac{1}{2}| & \mu H_z g_z - \frac{1}{2}J_z & 0 & 0 & 0 \\
\langle+\frac{1}{2};-\frac{1}{2}| & 0 & \frac{1}{2}J_z & -J_z & 0 \\
\langle-\frac{1}{2};+\frac{1}{2}| & 0 & -J_z & \frac{1}{2}J_z & 0 \\
\langle-\frac{1}{2};-\frac{1}{2}| & 0 & 0 & 0 & -\mu H_z g_z - \frac{1}{2}J_z
\end{array}\right)
$$

“Традыцыйныя” метады акселерацыі

- Самая крытычная да рэсурсаў частка функцыянальнасці можа быць напісана на кампіляванай мове праграмавання;
- Прэкампіляцыя ў байт-код;
- Выкарыстанне адных канструкцый мовы замест іншых, аналагічных, але хутчэйшых

Прыклады замены канструкцый

- `map(operator.add, l1, l2)`

  ```
  map(lambda x,y: x+y, l1, l2)
  ```

- `"".join(map(lambda x:"".join(x),a))`

  ```
  for i in range(n):
    for j in range(n):
      collect_str+=a[i][j]
  ```

http://www.xakep.ru/magazine/xa/123/102/1.asp

Статычная кампіляцыя праграм на Python

- Статычная (розныя падмноствы мовы Python):
  - Shedskin
  - PyPy (падмноства RPython)

Дае магчымасць стварыць вельмі хуткі код

Абмяжоўвае мажлівасці мовы, складана застасаваць да ўжо гатовых праэктаў

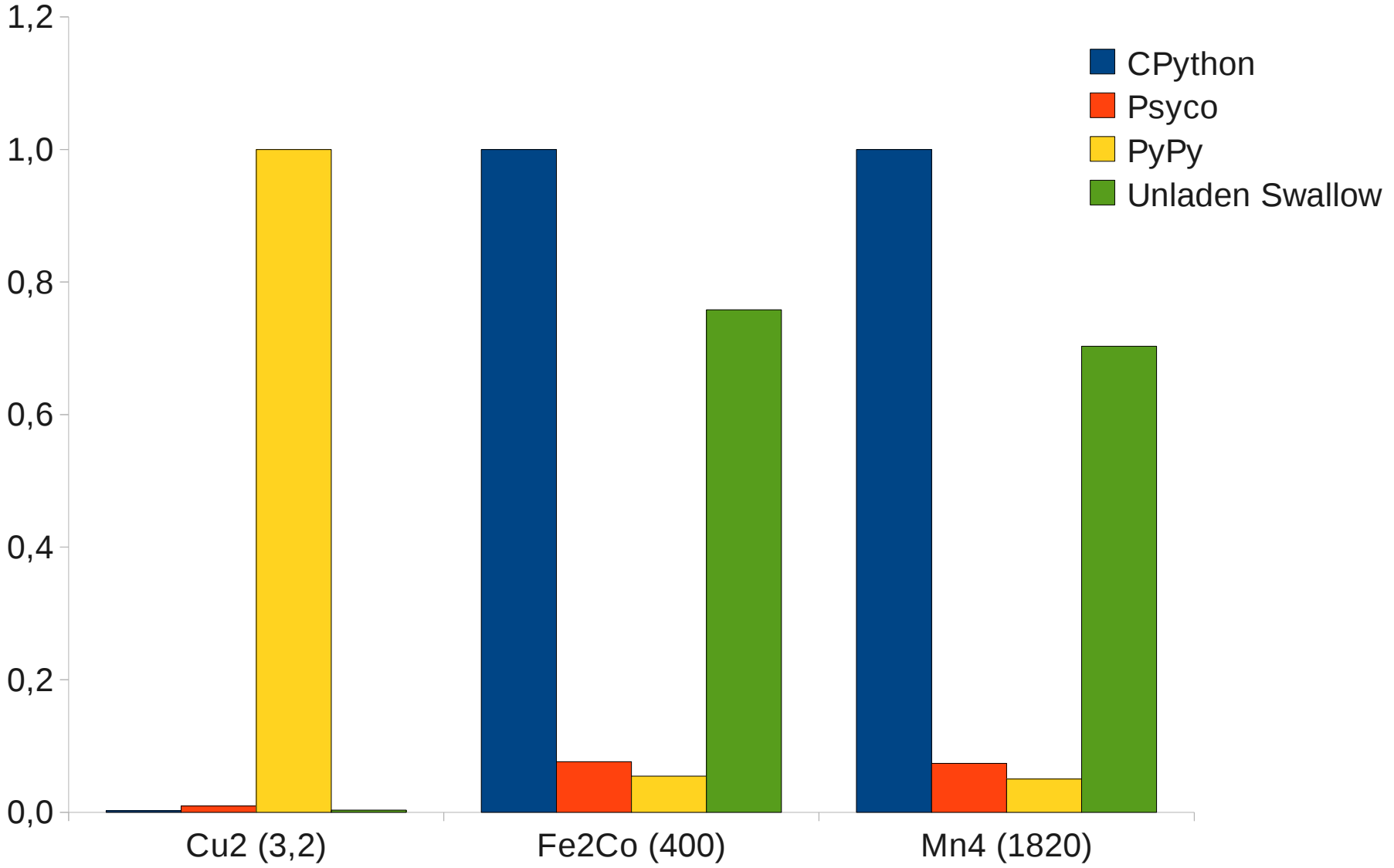
JIT-кампіляцыя

Psyco (модуль CPython)	PyPy (асобны інтэрпрэтатар)	Unladen Swallow (аптымізаваны CPython).
Прасты ў выкарыстанні, адносна хуткі, сумяшчальны з большасцю іншых модуляў	Хуткі (хутчэйшы сярод іншых), даступны пад большасць платформ, лёгкі ў развіцці	Прасты ў выкарыстанні (увогуле не патрабуе праўкі коду)
Складаны ў развіцці, адсутнічае версія для 64-бітных платформ	Ідэалагічна праблемнае ўзаемадзеянне з бібліятэкамі на C	Малаэфектыўны

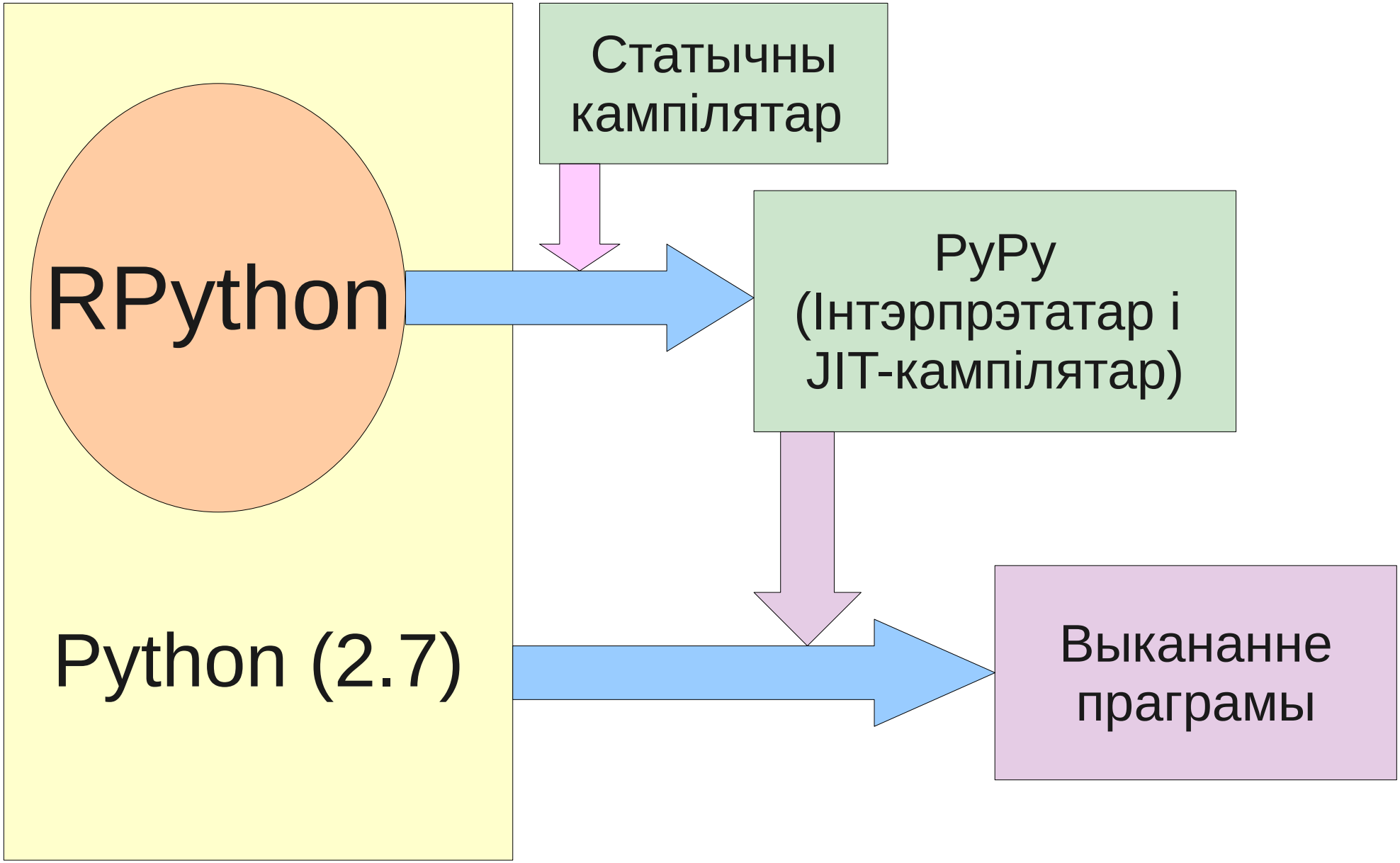
Параўнанне хуткасці выканання з праграмы Mjöllnir (час выканання у секундах)

Мадэль	Cu_2	Fe_2Co	Mn_4
Базісныя функцыі	4	432	625
CPython 2.6.2	0,0075±0,0002	400±7	1820±20
Psyco 1.6/CPython 2.6.2	0,0304±0,0002	30,5±0,3	134±1
PyPy 1.5	3,2±0,1	21,8±0,6	91±2
Unladen Swallow 2009Q3	0,010±0,005	303±4	1280±20

Параўнанне хуткасці выканання з выкарыстаннем праграмы Mjöllnir



Асаблівасці праэкту PyPy



Высновы

- для праграмы Mjöllnir найбольш эфектыўным акселератарам выканання на дадзены момант з’яўляецца PyPy і, з улікам імклівага развіцця праэкту, верагодным ёсць ягонае выкарыстанне ў будучыні як асноўнага (зараз ужываецца Psyco);
- Psyco дае блізкія вынікі;
- Unladen Swallow з’яўляецца неэфектыўным.

Падзякі

- *С.А. Літвіненку (за каштоўныя парады).*