PYTHON/NUMPY
A "KLASYCZNE" PROGRAMOWANIE W NAUKACH ATMOSFERYCZNYCH - NOWE MOŻLIWOŚCI I PROBLEMY

Dorota Jarecka

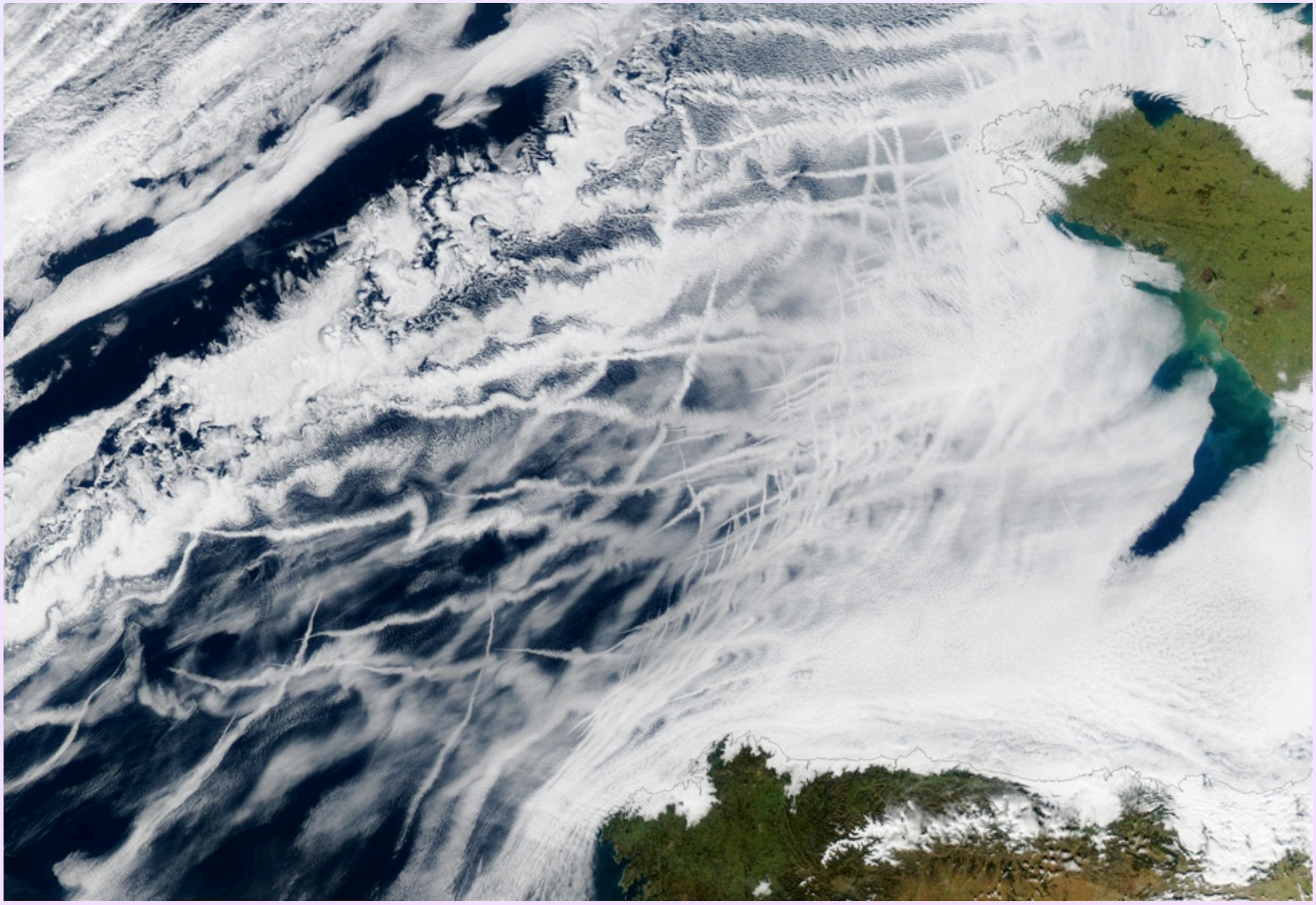
Sylwester Arabas, Anna Jaruga

Instytut Geofizyki, Wydział Fizyki, Uniwersytet Warszawski

Maciej Fijałkowski

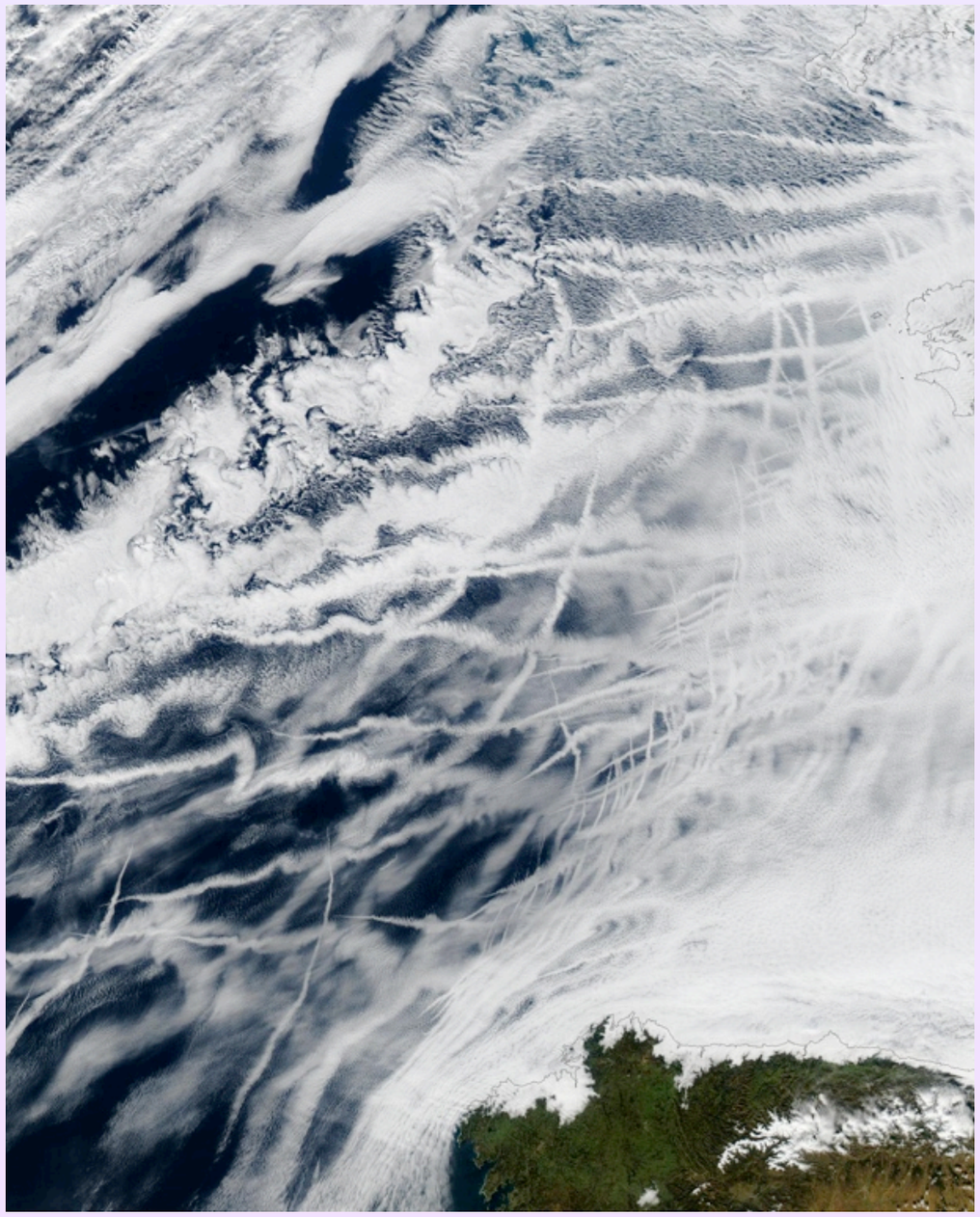
PyPy Team

DLACZEGO CIĄGLE ZAJMUJEMY SIĘ CHMURAMI..?



zdjęcie NASA/MODIS

DLACZEGO CIĄGLE ZAJMUJEMY SIĘ CHMURAMI..?



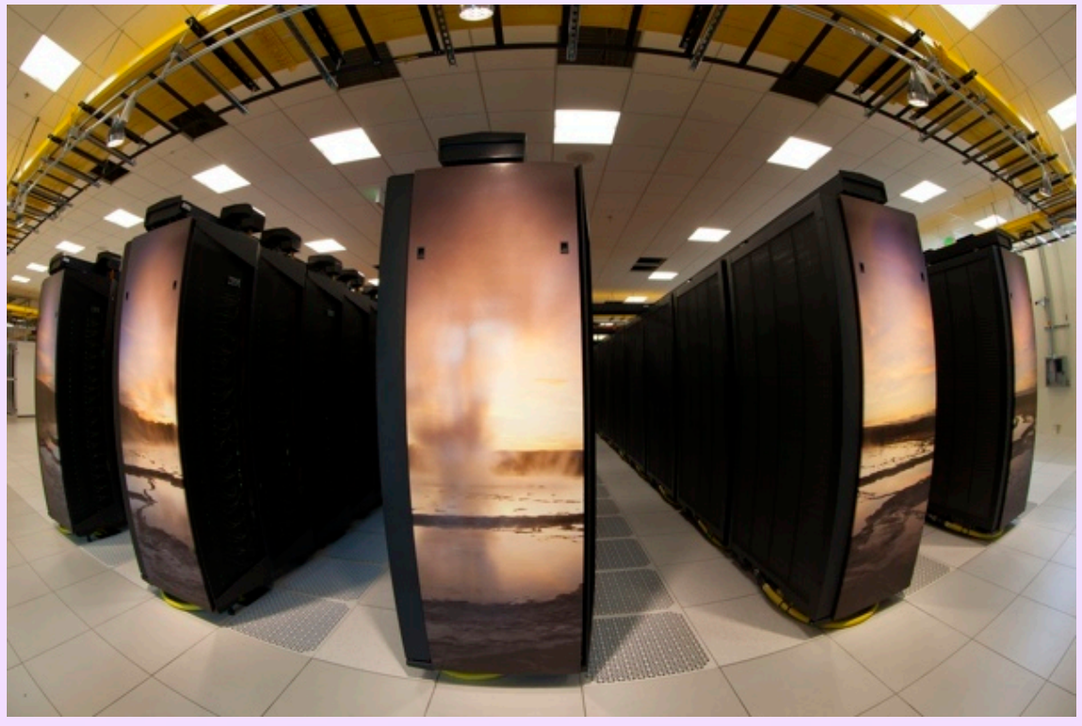
zdjęcie NASA/MODIS

- chmury składają się z kropelek wody powstających na drobinach zanieczyszczeń (tzw. aerozolach)
- własności fizyko-chemiczne zanieczyszczeń wpływają na rozmiar i ilość kropelek chmurowych
- rozkład rozmiarów kropelek wpływa m.in. na:
  - albedo (zdolność chmury do odbijania promieniowania) - patrz rysunek
  - zdolność chmury do wytworzenia deszczu
- chmury są istotnym elementem modeli pogody i klimatu

DO CZEGO WYKORZYSTUJEMY MODELE NUMERYCZNE?

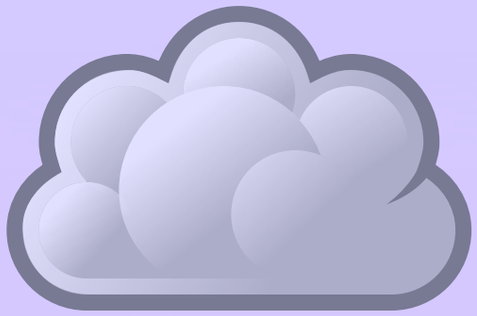
- Do badania zjawisk fizycznych, których nie można zbadać eksperymentalnie
- Do przewidywania przyszłości:
  - prognozy pogody
  - przyszłe zmiany klimatyczne

Frost
IBM Blue Gene/L

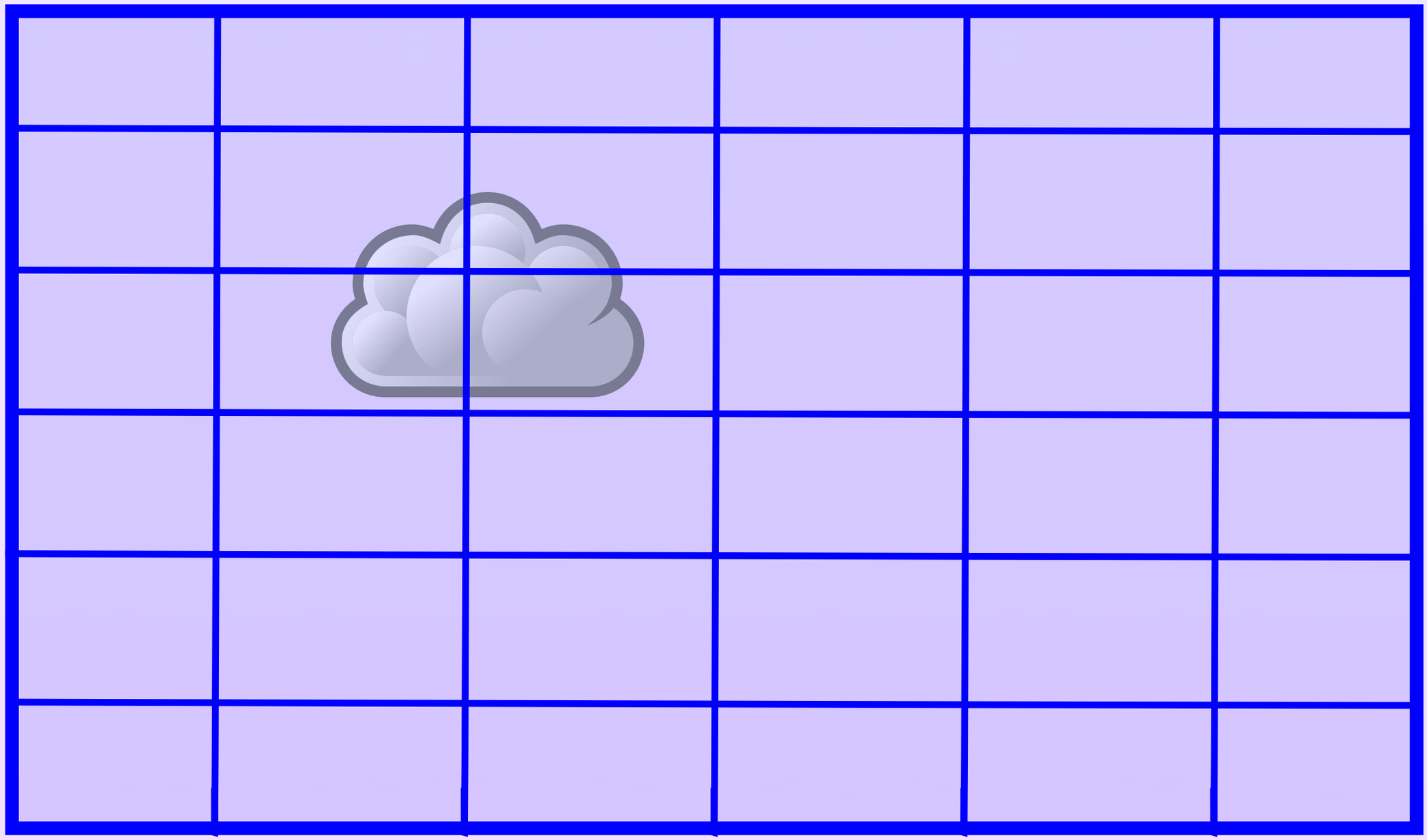


Yellowstone
IBM iDataPlex,

CHMURA OBSERWOWANA

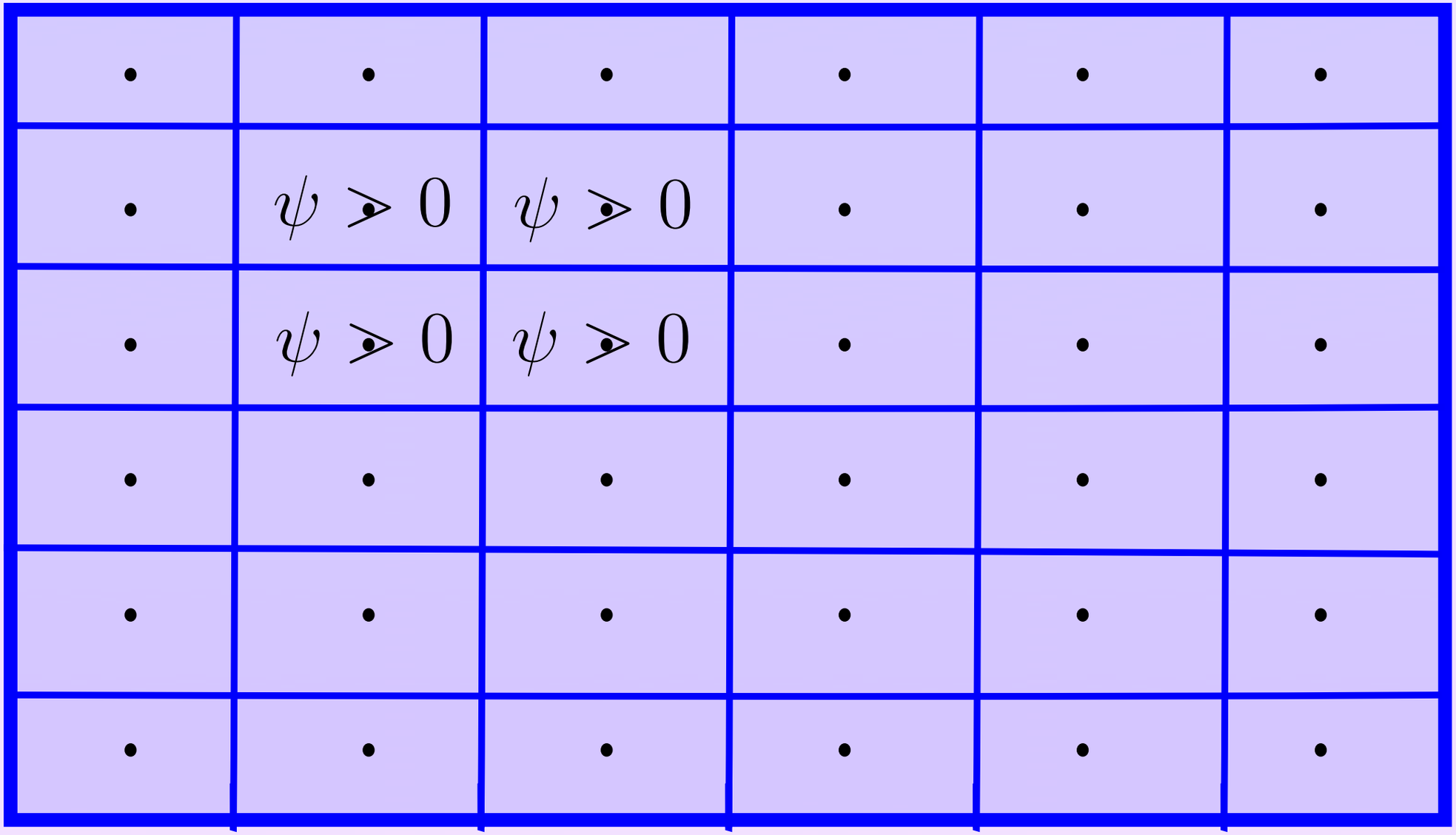


CHMURA MODELOWANA



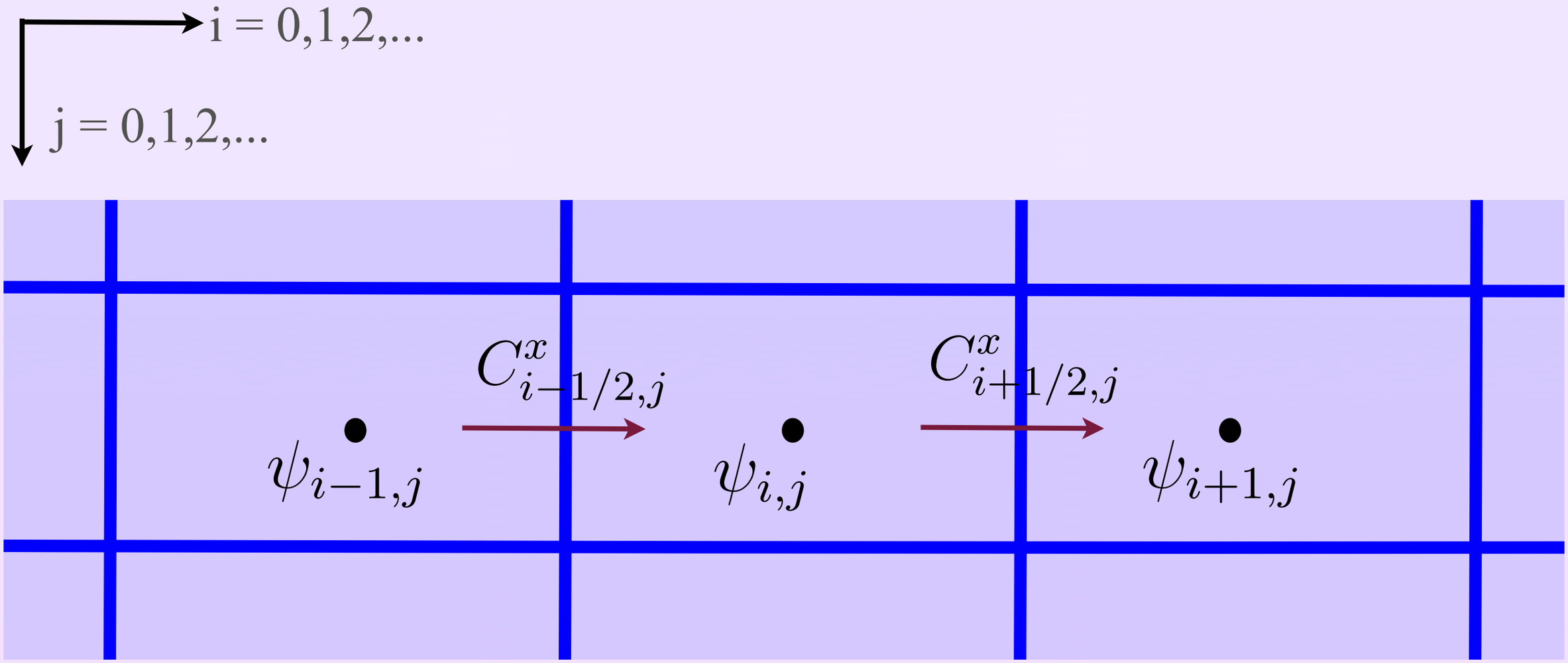
Badany obszar tworzy domenę modelu, którą dzieli się na tzw. oczka modelu stosując np. siatkę kartezjańską

CHMURA MODELOWANA



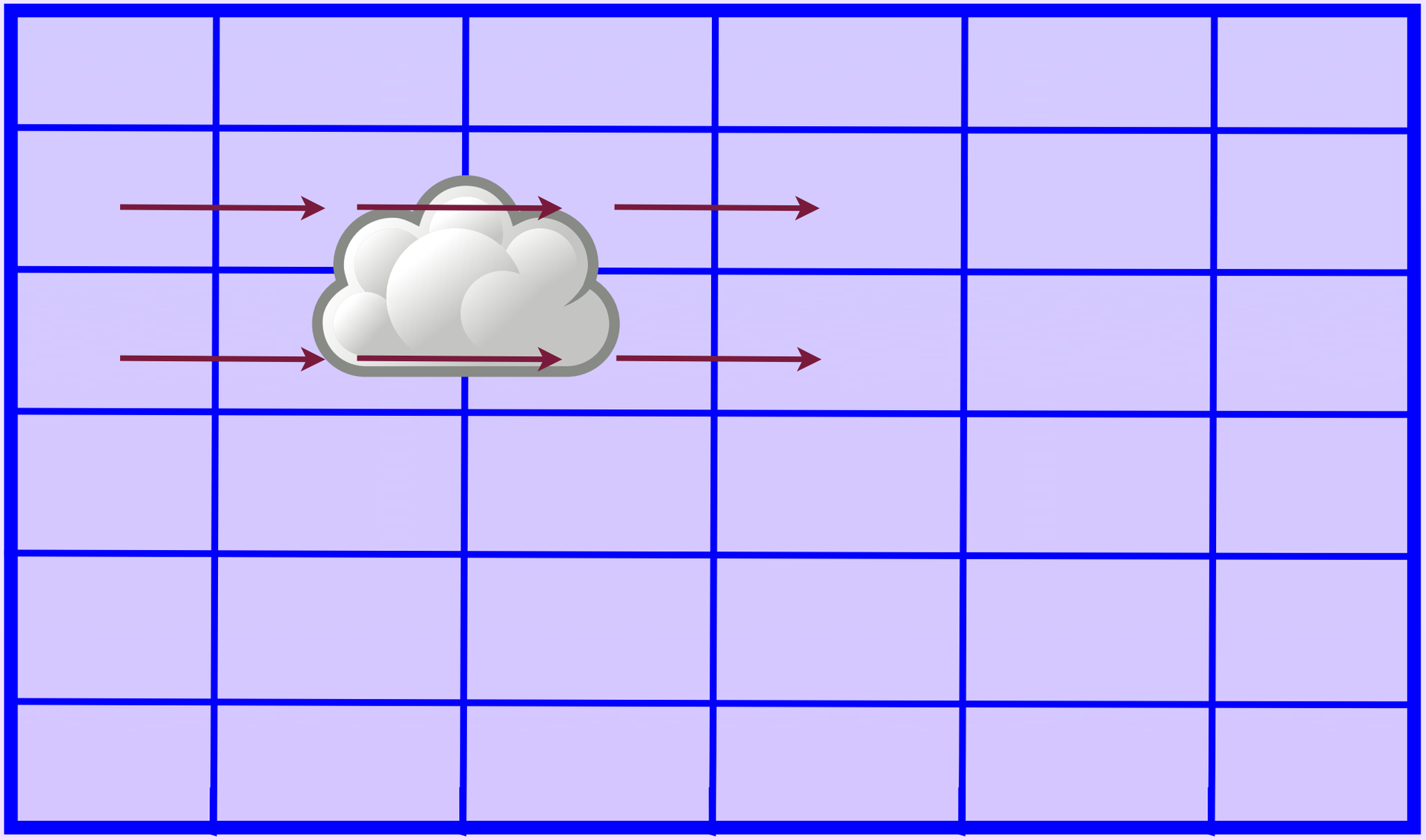
W każdym oczku modelu rozwiązuje się równania na wielkości fizyczne ψ, np. temperaturę, zawartość ciekłej wody

SIATKA ARAKAWA - C



- Skalary liczone w środku oczka modelu
- Prędkości liczone na brzegach oczka modelu

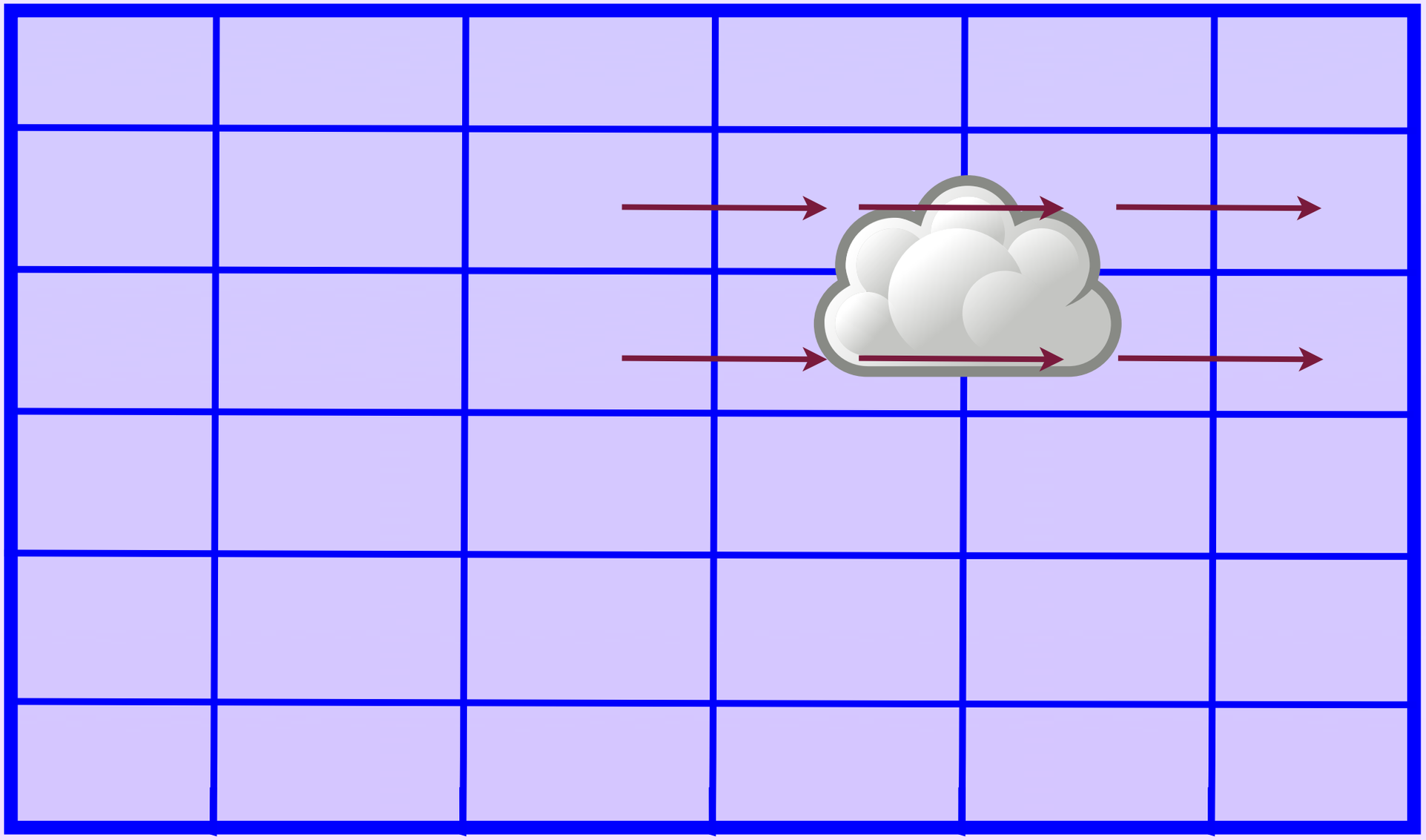
PROCES ADWEKCJI



Równanie adwekcji dla skalara:

$$\partial_t \psi = -\nabla \cdot (\vec{v}\psi)$$

PROCES ADWEKCJI



Równanie adwekcji dla skalara:

$$\partial_t \psi = -\nabla \cdot (\vec{v}\psi)$$

MODELE NUMERYCZNE - CHARAKTERYSTYKA

Moje doświadczenia:

- FORTRAN 77
- ~100 kLOC latami dopisywane do jednego pliku bez użycia bibliotek
- współpraca wielu grup badawczych bez użycia repozytorium
- dokumentacja w artykułach naukowych

MODELE NUMERYCZNE - CHARAKTERYSTYKA

Moje doświadczenia:

- FORTRAN 77
- ~100 kLOC latami dopisywane do jednego pliku bez użycia bibliotek
- współpraca wielu grup badawczych bez użycia repozytorium
- dokumentacja w artykułach naukowych

Obserwacje NASA:

The current Earth system models codify the accumulation of more than 40 years of scientific discovery and knowledge. As the knowledge grows, their size also grows. A typical Earth system model consists of more than a million lines of code. (...)

As both the complexity and societal importance of Earth system models continues to grow, the quality of the underlying software implementation is too often not keeping up with modern software engineering methodology, leading to inadequate software infrastructure, and opening development and testing gaps that raise risk and significantly limit return on the science research investment. (...)

MODELE NUMERYCZNE - CZY MOŻE BYĆ LEPIEJ..?

Sugestie NASA:

In order to enable distributed model development and to provide a means to encourage open and rapid contribution, modern software engineering methodologies must be practiced to develop modular, agile, easily maintainable, and extensible code.

MODELE NUMERYCZNE - CZY MOŻE BYĆ LEPIEJ..?

Sugestie NASA:

In order to enable distributed model development and to provide a means to encourage open and rapid contribution, modern software engineering methodologies must be practiced to develop modular, agile, easily maintainable, and extensible code.

Dalsza część prezentacji :

- Wykorzystanie obiektowego programowania - OOP do odwzorowania matematycznego zapisu używanego w literaturze naukowej

Nasz projekt (http://arxiv.org/abs/1301.1334):

- stworzenie prototypu solwera równań adwekcyjnych z zastosowaniem technik programowania obiektowego (OOP) w trzech językach: **Python/NumPy**, **C++11/Blitz++** oraz **(OOP-) Fortran 2008**

- zachowanie tej samej struktury OOP we wszystkich implementacjach

- odwzorowanie "abstrakcji" i notacji matematycznej używanej w literaturze ("blackboard abstractions")

- dyskusja konsekwencji wyboru języka

IMPLEMENTACJA W **PYTHONIE** - ADWEKCJA 1D

$$\psi_i^{n+1} = \psi_i^n - \left(F\left[\psi_i^n, \psi_{i+1}^n, C_{i+1/2}\right] - F\left[\psi_{i-1}^n, \psi_i^n, C_{i-1/2}\right]\right)$$

$$F(\psi_L, \psi_R, C) = \frac{C + |C|}{2} \cdot \psi_L + \frac{C - |C|}{2} \cdot \psi_R$$

```
def adv_op(psi, C, i):
    return (
        f(psi[i    ], psi[i+one], C[i+hlf]) -
        f(psi[i-one], psi[i    ], C[i-hlf])
        )

def scalar_advection(psi, n, C, i):
    psi[n+1][i] = psi[n][i] - adv_op(psi[n], C, i)


def f(psi_l, psi_r, C):
    return (C + abs(C))/2 * psi_l + (C - abs(C))/2 * psi_r
```

- odwzorowanie zapisu matematycznego
- brak pętli
- "przeciążenie operatorów" dla połówkowych indeksów (siatka Arakawa-C)

IMPLEMENTACJA W **PYTHONIE** - ADWEKCJA 2D

$$\psi_{i,j}^{n+1} = \psi_i^n - \left(F\left[\psi_{i,j}^n, \psi_{i+1,j}^n, C_{i+1/2,j}^0\right] - F\left[\psi_{i-1,j}^n, \psi_{i,j}^n, C_{i-1/2,j}^0\right]\right)$$
$$- \left(F\left[\psi_{i,j}^n, \psi_{i,j+1}^n, C_{i,j+1/2}^1\right] - F\left[\psi_{i,j-1}^n, \psi_{i,j}^n, C_{i,j-1/2}^1\right]\right)$$

zwięzła notacja
z użyciem permutacji (**π**):

$$\sum_{d=0}^{1} \psi_{[i,j]+\pi_{1,0}^d} = \psi_{[i+1,j]} + \psi_{[i,j+1]}$$

odpowiednik w **NumPy**?
(dwukrotnie zmniejszyłby długość kodu)

$$\psi_{[i,j]}^{n+1} = \psi_{[i,j]}^n - \sum_{d=0}^{N-1}\left(F\left[\psi_{[i,j]}^n, \psi_{[i,j]+\pi_{1,0}^d}^n, C_{[i,j]+\pi_{1/2,0}^d}^d\right] - F\left[\psi_{[i,j]+\pi_{-1,0}^d}^n, \psi_{[i,j]}^n, C_{[i,j]+\pi_{-1/2,0}^d}^d\right]\right)$$

IMPLEMENTACJA W **PYTHONIE** - ADWEKCJA 2D

$$\psi^{n+1}_{[i,j]} = \psi^{n}_{[i,j]} - \sum_{d=0}^{N-1} \left(F\left[\psi^{n}_{[i,j]}, \psi^{n}_{[i,j]+\pi^{d}_{1,0}}, C^{d}_{[i,j]+\pi^{d}_{1/2,0}}\right] - F\left[\psi^{n}_{[i,j]+\pi^{d}_{-1,0}}, \psi^{n}_{[i,j]}, C^{d}_{[i,j]+\pi^{d}_{-1/2,0}}\right] \right)$$

```
def adv_op(d, psi, C, i, j):
    return (
        f(psi[pi(d, i,     j)], psi[pi(d, i+one, j)], C[pi(d, i+hlf, j)]) -
        f(psi[pi(d, i-one, j)], psi[pi(d, i,     j)], C[pi(d, i-hlf, j)])
        )

def scalar_advection(psi, n, C, i, j):
    psi[n+1][i,j] = psi[n][i,j] - (adv_op(0, psi[n], C[0], i, j) +
                                   adv_op(1, psi[n], C[1], j, i))

def pi(d, *idx):
    return (idx[d], idx[d-1])
```

- łatwo rozszerza się na dodatkowe wymiary

CZYTELNOŚĆ KODU: **PYTHON** VS. **C++** I **FORTRAN**

Odwzorowanie zapisu matematycznego:

- **Python/NumPy, C++11/Blitz++** i **Fortran 2008**: wszystkie oferują porównywalne możliwości

Łatwość użycia:

- **Python**: najprostsza składnia, zachęca do OOP, łatwy do nauczenia
- **NumPy** (w odróżnieniu od **Blitz++**) zostało zaadaptowane przez wiele innych bibliotek

Zwięzłość zapisu:

- **Python/NumPy:** 250LOC, **C++11/Blitz++:** 350LOC; **Fortran 2008**: 550 LOC

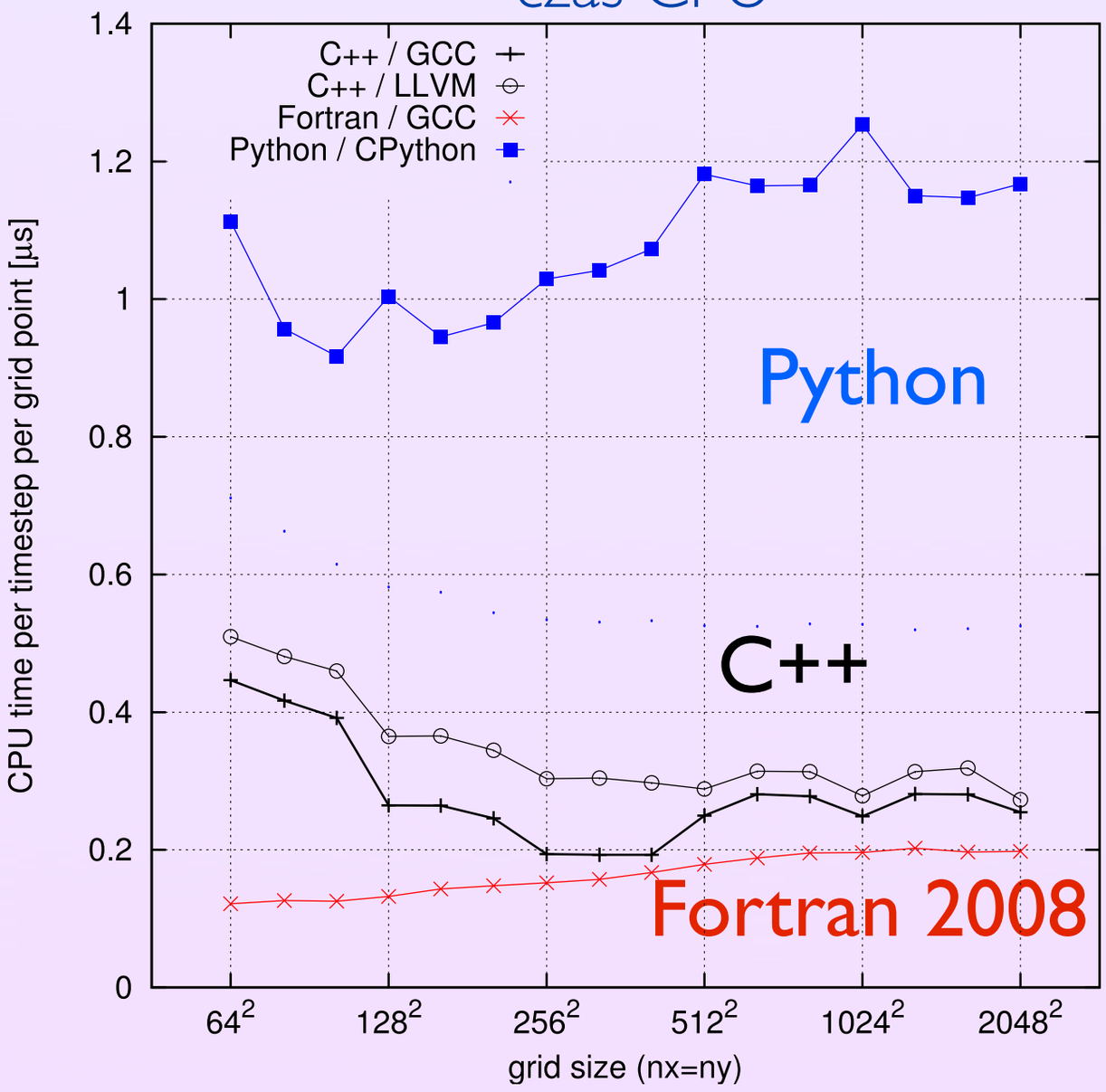
Wartości dodane:

- **Python** i **C++**: biblioteki OOP, narzędzia do testowania
  (**Fortran**: biblioteki i debuggery nie wspierają OOP)
- **Python** i **C++** (w odróżnieniu od **Fortrana**): języki ogólnego użytku

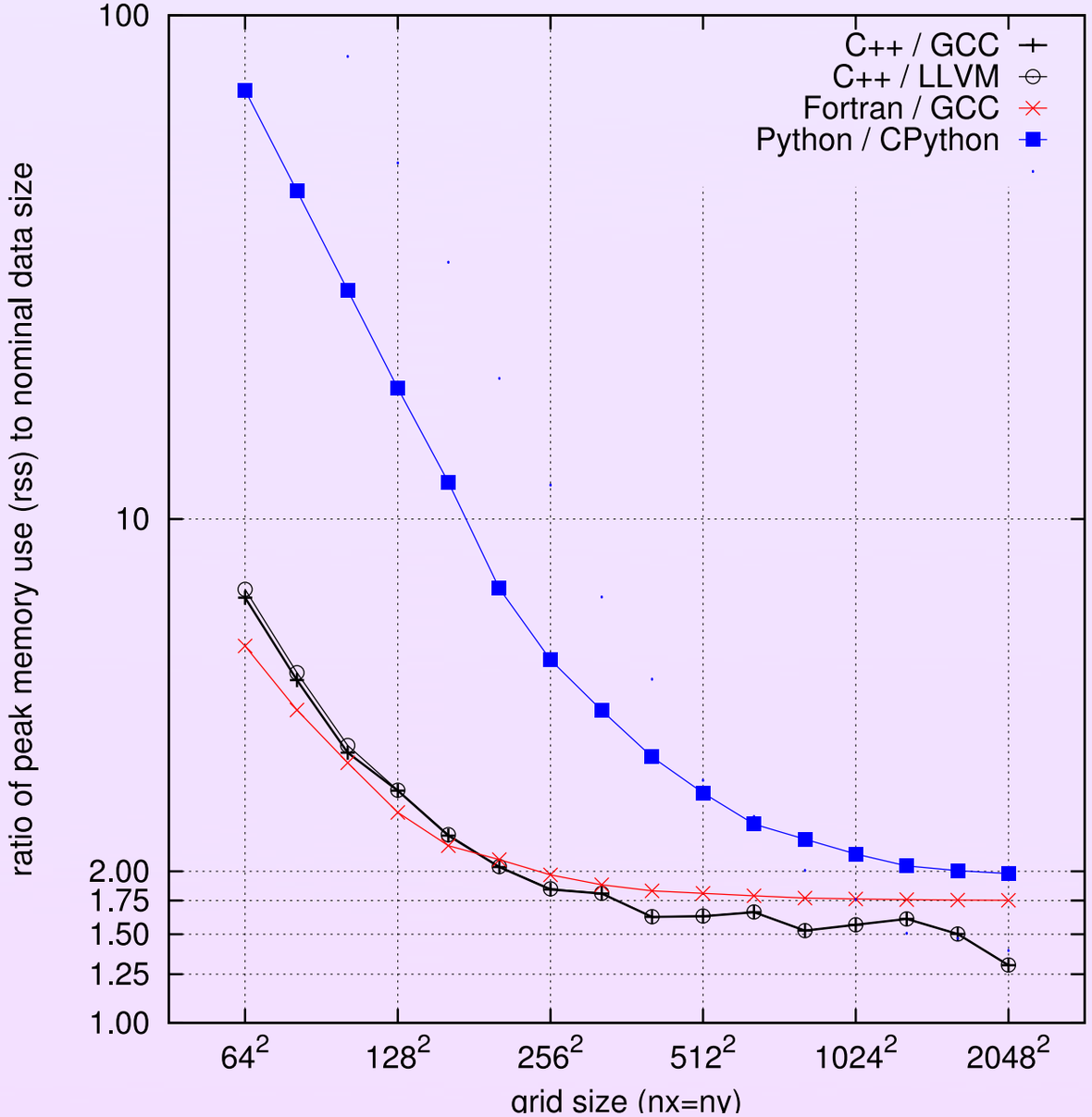
WYDAJNOŚĆ

- ustawienie: MPDATA
- testowany na 64-bit AMD CPU

czas CPU



maksymalne zużycie pamięci



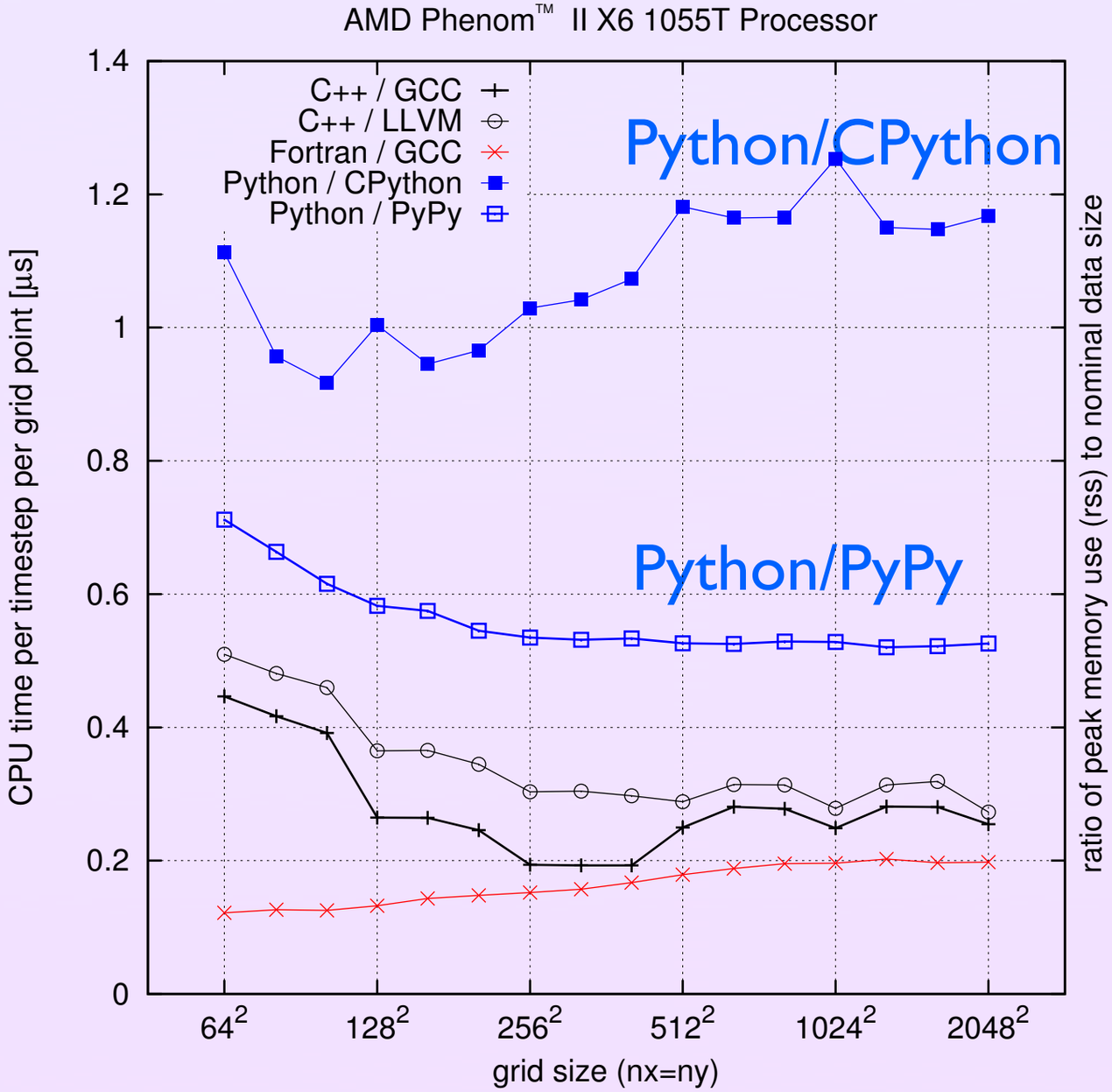
- **Python**: znacznie wolniejszy, duże zużycie pamięci

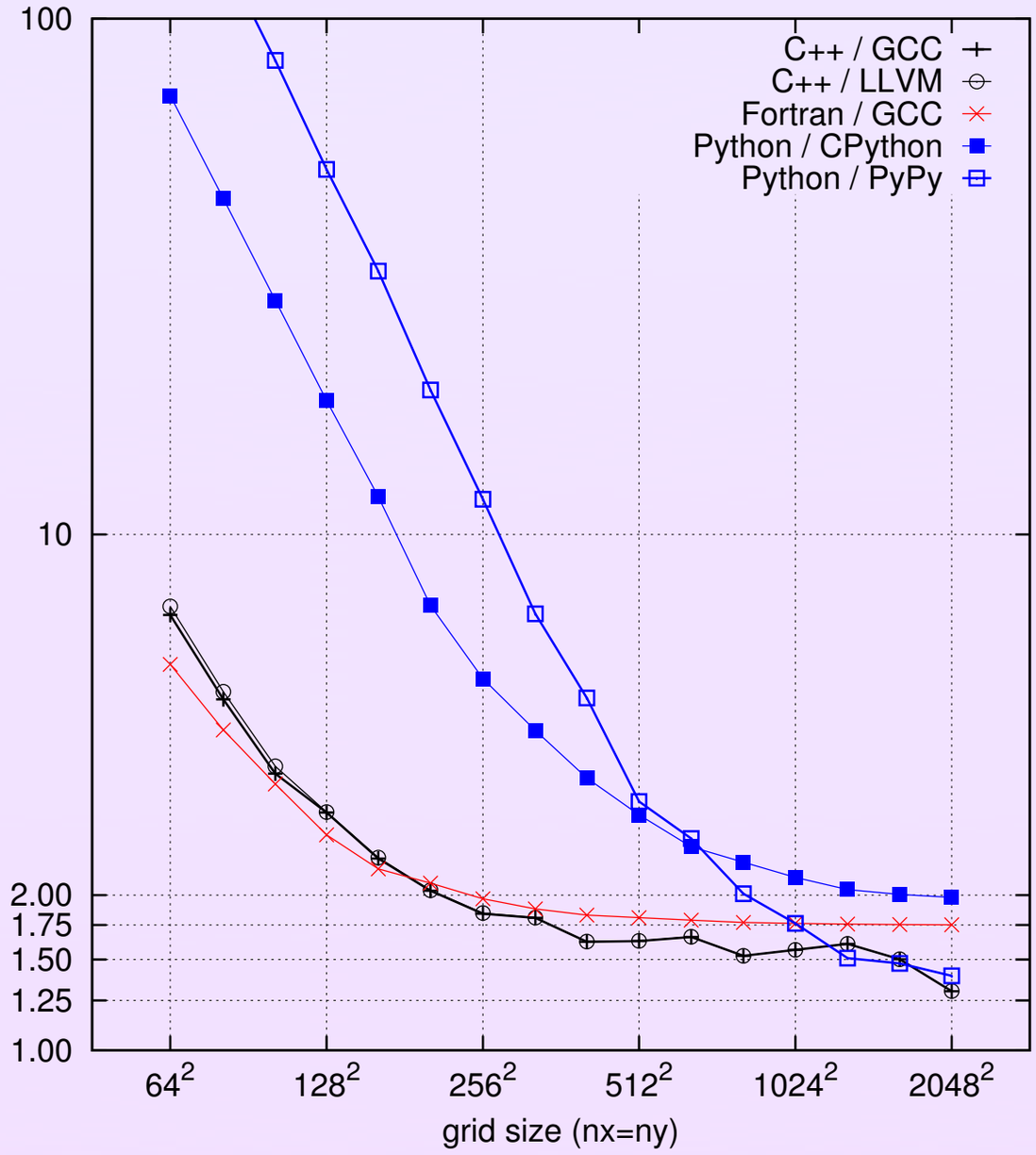
PRZYŚPIESZANIE **PYTHONA** - **PYPY**

- Czym jest **PyPy**?
  - interpreter **Pythona** i just-in-time compiler (JIT)
  - nacisk na poprawienie wydajności i zachowanie kompatybilności z oryginalnym interpreterem, z **CPythonem**
  - więcej informacji na www.pypy.org
- Dlaczego **PyPy**?
  - możliwość poprawy wydajności kodów opartych na **NumPy** (prędkość i zużycie pamięci)
  - nie ma potrzeby zmian w oryginalnym kodzie **Pythona**! Zachowuje OOP/składnie pythonową (w odróżnieniu od **Cython**, **numexpr, Numba...**)

WYDAJNOŚĆ - **PYPY**

Python/CPython

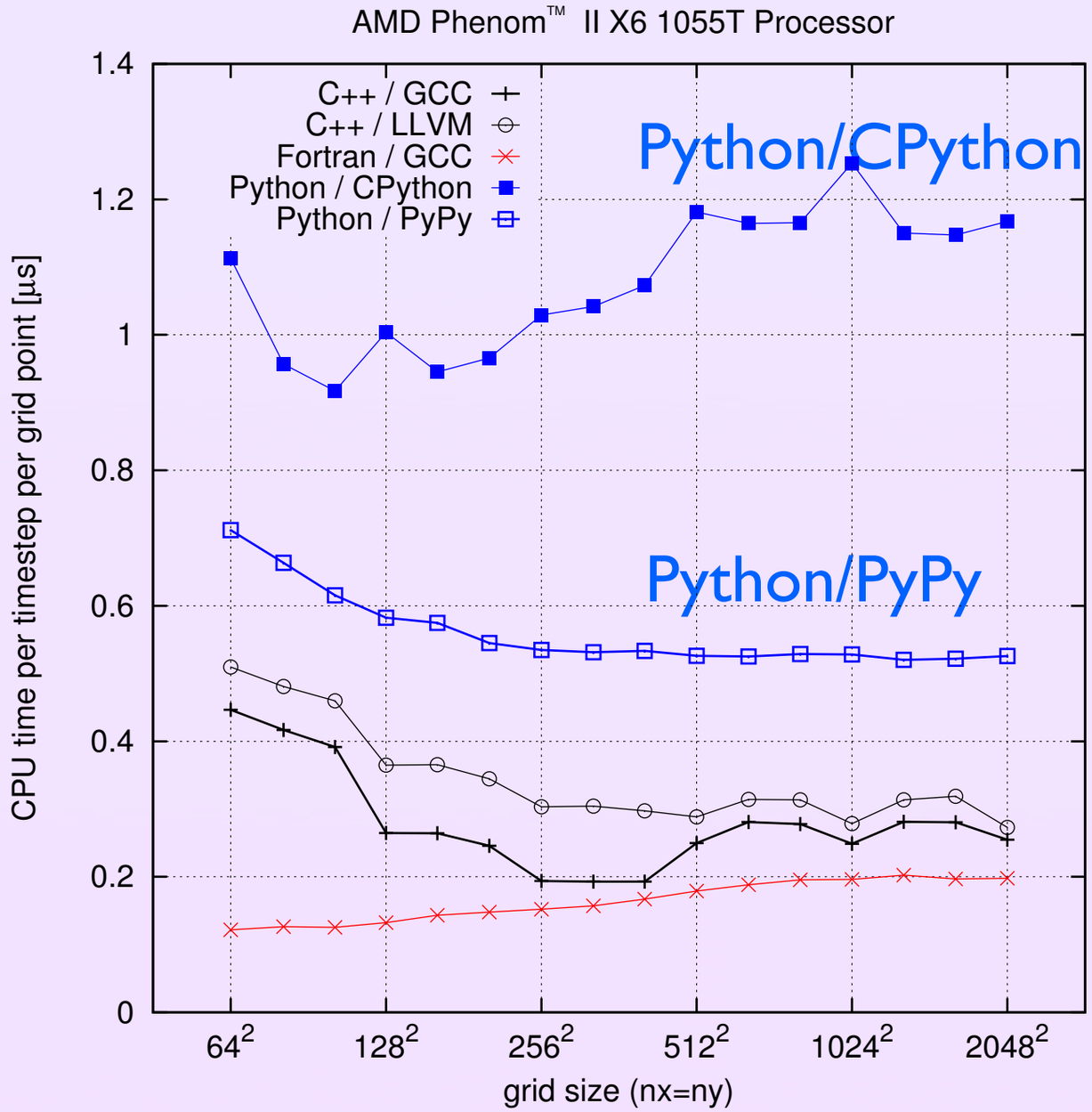


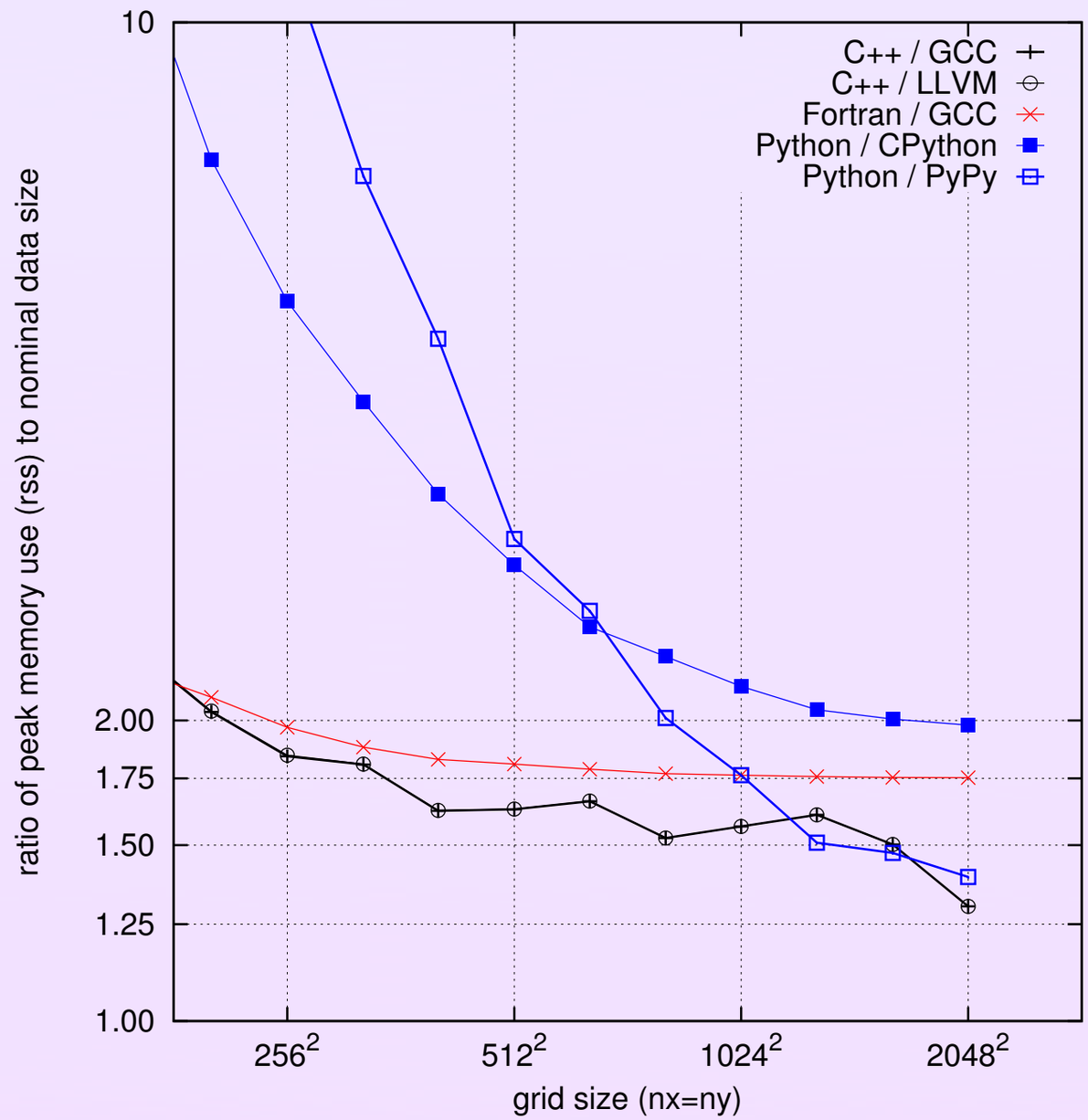


- **PyPy**: czas CPU wyraźnie zbliżył się do **C++** i **OOP-Fortrana**

WYDAJNOŚĆ - **PYPY**

Python/CPython





- **PyPy**: czas CPU wyraźnie zbliżył się do **C++** i **Fortrana 2008**; niższe zużycie pamięci niż **CPython i Fortran 2008**

DLACZEGO **PYPY** JEST SZYBSZE?

- Problem przy pisaniu operacji na tablicach bez użycia pętli: wyrażenia typu **a = b + c + d** prowadzą do powstawania tablic tymczasowych (np. na przechowanie wyniku **c+d**)

- Eliminowanie tablic tymczasowych:

  - **Fortran:** zależy od kompilatora

  - **C++**: zapewnione przy korzystaniu z biblioteki **Blitz++**

  - **CPython/NumPy**: brak wbudowanego mechanizmu

  - **PyPy**: eliminuje tablice tymczasowe poprzez tzw. "lazy evaluation" (eksperymentalna opcja w **PyPy 1.9**)

```
import numpypy
from _numpypy.pypy import set_invalidation
set_invalidation(False)
```

PYPY VS. CPYTHON, C++ I FORTRAN

Wydajność:

- **Python/NumPy** z **PyPy:**
  - wyeliminowanie tablic tymczasowych (podobnie jak w **C++/Blitz++**)
  - istotnie szybszy niż **CPython**
  - mniejsze zużycie pamięci niż **Fortran 2008** i **CPython/NumPy**

Po uwzględnieniu czasu programisty: **Python** z **PyPy** wygrywa! :)

... pomijamy porównanie możliwości obliczeń operacji wielowątkowych

Więcej wniosków w artykule - http://arxiv.org/abs/1301.1334

DZIĘKUJĘ ZA UWAGĘ!

Repozytorium:

http://github.com/slayoo/mpdata/

Artykuł (wszelkie uwagi mile widziane!!):

Sylwester Arabas, Dorota Jarecka, Anna Jaruga, Maciej Fijalkowski:

Object-oriented implementations of the MPDATA advection equation solver in C++, Python and Fortran

http://arxiv.org/abs/1301.1334

Podziękowania:

Thanks are due Piotr Smolarkiewicz and Hanna Pawłowska. Authors would like to thank Tobias Burnus, Julian Cummings, Patrik Jonsson and Arjen Markus for their feedback on questions posted to Blitz++ and gfortran mailing lists.

SA, AJ and DJ acknowledge funding from the Polish National Science Centre (project no. 2011/01/N/ST10/01483). Part of the work was carried out during a visit of SA to the Na- tional Center for Atmospheric Research (NCAR) in Boulder, Col- orado, USA. NCAR is operated by the University Corporation for Atmospheric Research. The visit was funded by the Foundation for Polish Science (START programme).

Development of NumPy support in PyPy was led by Alex Gaynor, Matti Picus and MF.