Python at Warp Speed

Andreas Schreiber
Department for Intelligent and Distributed Systems
German Aerospace Center (DLR), Cologne/Berlin
Introduction

Scientist, Head of department

Deutsches Zentrum für Luft- und Raumfahrt
German Aerospace Center

Co-Founder, Data Scientist, and Patient

Communities

Quantified Self Meetup Cologne
cologne
PyHPC2016
Python at Warp Speed

- High-Performance Computing
- Distributed Computing
- Quantum Computing
Algorithmic View
Input $x$ – Algorithm $A$ – Output $y$
High-Performance Computing
High-Performance Computing

High raw computing power for large science applications

- Huge performance on a single / multi-core processors
- Huge machines with up to millions of cores
Sunway TaihuLight

10,649,600 Cores, 93 PetaFLOPS
Tianhe-2 (天河二号)
3.120.000 Cores, 33,8 PetaFLOPS
Titan
560,640 Cores, 17,5 PetaFLOPS
Programming HPC

Technologies
- MPI (Message Passing Interface)
- OpenMP (Open Multi-Processing)
- OpenACC (Open Accelerators)
- Global Arrays Toolkit
- CUDA (GPGPUs)
- OpenCL (GPGPUs)

Languages
- Fortran
- Fortran
- C/C++
- Python
Performance Fortran vs. Python
Helicopter Simulation

Fortran Code

• Developed 1994-1996
• parallelized with MPI
• Performance optimization 2013/14 with MPI und OpenACC

Performance Comparison

• Multi-core CPUs
  • Cython mit OpenMP
  • Python bindings for Global Array Toolkit
• GPGPUs
  • NumbaPro
Core Computation Loops in Pure Python

```python
for iblades in range(numberOfBlades):
    for iradial in range(1, dimensionInRadialDirection):
        for iazimutal in range(dimensionInAzimualDirectionTotal):
            for i1 in range(len(vx[0])):
                for i2 in range(len(vx[0][0])):
                    for i3 in range(len(vx[0][0][0])):
                        # wilin-Aufruf 1

for iblades in range(numberOfBlades):
    for iradial in range(dimensionInRadialDirection):
        for iazimutal in range(1, dimensionInAzimualDirectionTotal):
            for i1 in range(len(vx[0])):
                for i2 in range(len(vx[0][0])):
                    for i3 in range(len(vx[0][0][0])):
                        # wilin-Aufruf 2

for iDir in range(3):
    for i in range(numberOfBlades):
        for j in range(dimensionInRadialDirection):
            for k in range(dimensionInAzimualDirectionTotal):
                x[iDir][i][j][k] = x[iDir][i][j][k] +
                dt * vx[iDir][i][j][k]
```
Performance Fortran vs. Python

Single Core (Xeon E5645, 6 Cores)

<table>
<thead>
<tr>
<th>Language</th>
<th>GFlops</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fortran</td>
<td>2.51</td>
</tr>
<tr>
<td>Cython</td>
<td>1.09</td>
</tr>
<tr>
<td>Numba</td>
<td>0.27</td>
</tr>
<tr>
<td>Numpy</td>
<td>0.46</td>
</tr>
<tr>
<td>Python</td>
<td>0.01</td>
</tr>
</tbody>
</table>
Performance Fortran vs. Python

Multi-Core (Xeon E5645, 6 Cores)

GFlops

Fortran | Cython | Global Arrays
---|---|---
13.64 | 5.78 | 1.38
Performance Fortran vs. Python

GPGPU (NVIDIA Tesla C2075, 448 CUDA-Cores)

<table>
<thead>
<tr>
<th></th>
<th>Fortran</th>
<th>Numba</th>
</tr>
</thead>
<tbody>
<tr>
<td>GFlops</td>
<td>69.77</td>
<td>7.79</td>
</tr>
</tbody>
</table>
Performance-Productivity-Space

- Pure Python
- NumPy / Numba
- Cython
- C++ / FORTRAN
Productivity vs. Performance of Python

Python’s productivity is great

- Allows to write code quickly
- Wide range of applications

Python’s performance still needs improvements

- Code optimization with tools for profiling, code examination, …
- Optimized libraries for parallel processing with MPI etc.

Excited to see advancements by community and companies
Workshop „Python for High-Performance and Scientific Computing“ (PyHPC)

Annual scientific workshop, in conjunction with Supercomputing conference

State-of-the-art in

• Hybrid programming
• Comparison with other languages for HPC
• Interactive parallel computing
• High-performance computing applications
• Performance analysis, profiling, and debugging

PyHPC 2016

• 6th edition, Nov 14, 2016, Salt Lake City
• http://www.dlr.de/sc/pyhpc2016
Tools Example

*Intel® VTune™ Amplifier for Profiling*

Tools Example
mpi4py with Intel MPI Library and Cython

```python
from mpi4py import MPI
comm = MPI.COMM_WORLD
size = comm.Get_size()
rank = comm.Get_rank()
name = MPI.Get_processor_name()
if rank == 0:
    print "Rank %d of %d running on %s"
    % (rank, size, name)
    for i in xrange(1, size):
        rank, size, name = comm.recv(source=i, tag=1)
        print "Rank %d of %d running on %s"
        % (rank, size, name)
else:
    comm.send((rank, size, name), dest=0, tag=1)
```

http://pythonhosted.org/mpi4py/
Distributed Computing
Distributed Computing

Driven by data science, machine learning, predictive analytics, ...

- Tabular data
- Time Series
- Stream data
- Connected data

Scaling up with increased data size from laptops to clusters

- Many-Task-Computing
- Distributed scheduling
- Peer-to-peer data sharing
Space Debris: Object Correlation from Sensor Data and Real-Time Collision Detection
29,000 Objects Larger than 10 cm
750,000 Objects Larger than 1 cm
150M Objects Larger than 1 mm
Space Debris: Graph of Computations

- Organisation
  - Sensor Error Statistic
  - Observation Error
  - Correction of Observations
- Sensor
  - Observation
  - Manoeuvre
- Observation
  - Correlation
  - Satellite Launch
  - Fragmentation
  - Re-Entry
- Manoeuvre
  - Object
- Correlation
  - Orbit Determination
  - Orbit Propagation
  - Orbit Modelling
  - Time & Coordinate System
  - Space Weather
- Satellite Launch
  - Orbit
  - TLE Error Statistic
  - TLE
  - DSST
  - Osculating Elements
  - Ephemeris
- Fragmentation
  - CA (Close Approach)
- Re-Entry
  - CA Analysis / Warning
  - CA Detection
  - CA Threshold

DLR.de • Chart 28 > PyCon DE 2016 > Andreas Schreiber • Python at Warp Speed > 30.10.2016
Graphs

Directed Acyclic Graph (DAG)

Python has great tools to execute graphs on distributed resources
from pyspark import SparkContext

logFile = "myfile.txt"
sc = SparkContext("local", "Simple App")
logData = sc.textFile(logFile).cache()

numAs = logData.filter(lambda s: 'a' in s).count()
numBs = logData.filter(lambda s: 'b' in s).count()

print("Lines with a: %i, lines with b: %i" % (numAs, numBs))

https://spark.apache.org/
https://spark.apache.org/docs/0.9.0/python-programming-guide.html
import dask.dataframe as dd
df = dd.read_csv('2015-**-*.csv')
df.groupby(df.user_id).value.mean().compute()

d = {'x': 1,
     'y': (inc, 'x'),
     'z': (add, 'y', 10)}

http://dask.pydata.org/
import tensorflow as tf

with tf.Session() as sess:
    with tf.device("/gpu:1"):
        matrix1 = tf.constant([[3., 3.]])
        matrix2 = tf.constant([[2.], [2.]])
        product = tf.matmul(matrix1, matrix2)

https://www.tensorflow.org/
Quantum Computing
Optimization Problems in Aeronautics & Space

Design optimization and robust design

- Space systems and air transportation systems
- Design and evaluation of systems with consideration of uncertainties
Optimization Problems in Aeronautics & Space

Machine Learning

- Deep learning, pattern recognition, clustering, images recognition, stream reasoning

Anomaly detection

- Monitoring of space systems

Mission planning

- Optimization in relation to time, resource allocation, energy consumption, costs etc.

Verification and validation

- Software, embedded systems
New Research Field: Quantum Computing

Discrete optimization is basis for many kinds of problems

→ Packaging
→ Partitioning
→ Mapping
→ Scheduling

NP-hard problems!

Hope: Quantum Computers solve those problems faster than classical computers
Bits and Qubits

Classical Bits

- “0” or “1”
- Electric voltage

Quantum bits (Qubits)

- Superposition of complex base states

\[ |\psi\rangle = c_0 |0\rangle + c_1 |1\rangle \]
Quantum Computer

To date, one Quantum Computer is available commercially

- D-Wave Systems, Inc., Canada
- System with ~2000 Qubits („D-Wave 2X“)
- Adiabatic QC

Images: © D-Wave Systems, Inc.
Inside D-Wave 2

Images: © D-Wave Systems, Inc.
'Spin-up' circulating current

'Spin-down' circulating current

Josephson junction
D-Wave Qubit Topology – Chimera Graph
“Programming” a Quantum Computer – Step 1

Rewrite the problem as discrete optimization problem

• **QUBO**: Quadratic unconstrained binary optimization

\[
E(q_1, ..., q_n) = \sum_{i=1}^{n} g_i q_i + \sum_{i,j=1, i>j}^{n} s_{ij} q_i q_j
\]

\[q_i = \begin{cases} 
0, & \text{for switch off} \\ 
1, & \text{for switch on} 
\end{cases} \]

\[g_i \text{ weights} \]

\[s_{ij} \text{ strength of coupling} \]
“Programming” a Quantum Computer – Step 2

Mapping to hardware topology

→ Chimera-QUBO
“Programming” a Quantum Computer – Step 3

Bringing the problem to the QC

- Copy weights and coupling strengths to physical Qubits
Starting the actual “computation”:

- Adiabatic process from start energy level to target energy level, which represents solution of the optimization problem
- Result are the voltages after this process
D-Wave Software Environment

- Optimization
  - QSage
- Constraint Satisfaction
  - ToQ
- Sampling, SAT, ML
  - "Translators"
    - Host Libraries
    - Intermediate Representation
    - Command Line Interface
    - Quantum Machine Instruction

"Virtual" QUBO

QUBO

qbsolv

DW

QMI

Target System

C, C++, MATLAB, Python
import dwave_sapi2.remote as remote
import dwave_sapi2.embedding as embedding
import dwave_sapi2.util as util
import dwave_sapi2.core as core

# print "Connect to DWave machine ...
# create a remote connection
try:
    conn = remote.RemoteConnection(myToken.myUrl,
                                    myToken.myToken)

    # get the solver
    solver = conn.get_solver('C12')
except:
    print 'Unable to establish connection'
exit(1)
Programming with Python

Prepare the Problem ("Embedding")

```python
hwa = get_hardware_adjacency(solver)

embeddings[eIndex] = embedding.find_embedding(J, hwa)

h_embedded[eIndex], j0, jc, new_embed = embedding.embed_problem(h, J, embeddings[eIndex], hwa)
J_embedded[eIndex] = jc
J_embedded[eIndex].update(j0)
embeddings[eIndex] = new_embed
```
Programming with Python
Solve the Problem (“Ising”)
Result Distribution
Summary

Python is or will become standard in programming for...

High Performance

Data Science

Future Architectures
Thank You!

Questions?

Andreas.Schreiber@dlr.de

www.DLR.de/sc | @onyame