Parallel Programming with the Python Skeleton Library: New Data Structures

CS 485 – Undergraduate Research

Supervisor: Frédéric Loulergue

Fall 2019 or/and Spring 2020

Context

Low-level parallel programming for distributed memory architectures has been shown efficient to deal with large datasets but remains a difficult solution for most programers. The programmers have to face different constraints such as the explicit inter-processors communications or the distribution of data.

PySke [2] is a Python library that aims at easing parallel programming for casual users. The parallel implementation of computation patterns, called skeletons [1], are provided by the library to keep abstract the parallel aspects of a program. Their implementations in Python are made as high-order functions (i.e., functions that take other functions as input), to keep general as much as possible the computation pattern. The skeletons of PySke can be used as methods with a pointed notation, following an object-oriented paradigm, on parallel structures. These structures are equivalent to sequential structures but are distributed between several processors.

For example, an instance of the class `PList` represents a distributed list and can be created from a sequential one. PySke programs are very concise. For example the following code computes the variance of a discrete random variable `X` implemented as a parallel list:

```
n = X.length()
avg = X.reduce(add) / n
def f(x): return (x-avg) ** 2
var = X.map(f).reduce(add) / n
```

where `var` is given by the following mathematical formula:

$$\texttt{var} = \frac{1}{n}\sum_{k=0}^{n}(X_k - \mu)^2 \text{ where } \mu = \frac{1}{n}\sum_{k=0}^{n} X_k.$$

Such a program can be run on a multi-core workstation but also a on high-performance computing cluster such as NAU's Monsoon (2860 cores).

PySke source code is available at `https://pypi.org/project/pyske/`.

The Projects

The current version of PySke already provides two parallel data structures:

- parallel lists (class `PList`),
- parallel trees (class `PTree`).

These parallel data structure rely on sequential data structures:

- sequential lists (class `SList`),
- sequential linearized trees (class `LTree`).

There are opportunities for several projects to extent PySke with new data structures. Several students may develop different sets of data structures. For a given student, the list of new data structures to consider will depend on their interests and the number of units chosen for CS 485 (1 to 6).

Possible new data structures include:

- one dimensional parallel arrays based on the Python `array` module,
- one dimensional parallel arrays based on NumPy[1] arrays,
- parallel multidimensional arrays based on NumPy arrays,
- parallel dictionaries.

For each project, the deliverable will be:

1. a document on the overall design of the new data structures,
2. source code: implementation of the data structures and their skeletons,
3. user's documentation,
4. a document on performance experiments with at least 2 example applications (for one dimensional arrays these applications can be a port of the list applications).

For the development and experiments, students will be given access to SSERL[2] (room 227, SICCS) and its machines including the Titan workstation (256 Gb of memory and 32 cores) as well as Monsoon[3].

[1] `https://www.numpy.org`

[2] `https://sserl.github.io`

[3] `https://nau.edu/high-performance-computing`

Requirements

Minimum Requirements

- Good knowledge of programming with Python
- CS 396 with at least a C grade

Preferred Requirements

- CS 499 Introduction to Parallel Programming

References

[1] Murray Cole. *Algorithmic Skeletons: Structured Management of Parallel Computation.* MIT Press, 1989.

[2] Jolan Philippe. Systematic development of efficient programs on parallel data structures. Master's thesis, School of Informatics Computing and Cyber Systems, Northern Arizona University, May 2019.