PyData Array Recommendations

Thomas Dimitri

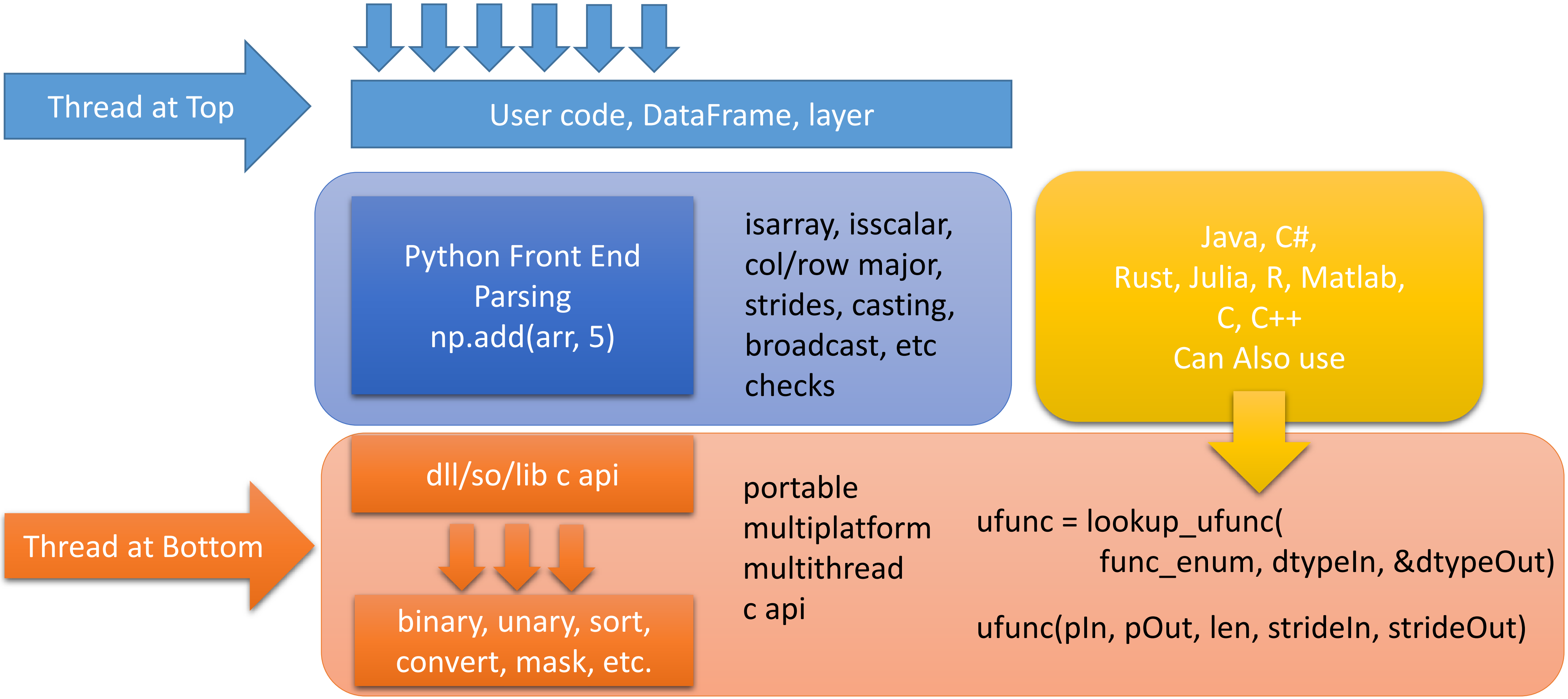
Background

- 30+ years professional software engineer – network protocols, user interfaces, compilers, file formats, pci cards, bios, trading engines, scanners, architectural software, 2 software companies, software patents, c/c++, java, c#, python
- 14 year consultant for SIG currently tasked with unifying large data analytics
- SIG has Matlab, Python (numpy, pandas), C++/C# algos
- Desire to move towards one platform over time
- Ongoing beta solution: riptide: Uses Datasets/Structs/NumpyArrays, Multithreaded Numpy, FileFormat with stacking and in memory file format for shared memory.

Array Recommendations (in order of importance)

- Multhread back end engine
- Introduce multikey categoricals with grouping routines
- New loops groupby loops/ partition loops/ closer JIT coupling
- Ledger + other ways to analyze performance
- Subclass before making another dtype: Rework Date/Time classes
- Introduce new routines (hashing) + Invalids
- Hooks for display/storage

Threading Model



Thread at Top

User code, DataFrame, layer

Python Front End
Parsing
np.add(arr, 5)

isarray, isscalar, col/row major, strides, casting, broadcast, etc checks

Java, C#,
Rust, Julia, R, Matlab,
C, C++
Can Also use

Thread at Bottom

dll/so/lib c api

binary, unary, sort, convert, mask, etc.

portable
multiplatform
multithread
c api

```
ufunc = lookup_ufunc(
            func_enum, dtypeIn, &dtypeOut)

ufunc(pIn, pOut, len, strideIn, strideOut)
```

Threading Model for Arrays

- Multicore/Shared memory same computer, same process
- NUMA
- Process Affinity
- Wakeup (futex) – important: selective thread wake up
- Calibration of worker threads

NOTE: Distributed/Cloud Computing (not covered)

Portable Multithreaded C

- Gold Standard and easy to achieve with simple interface decoupling
- Allows other platforms or languages to call same routines with same results and performance
- Prefer C interface over C++ (a C++ layer can wrap C interface)

Array Routines we Threaded

- Basic Math both Unary and Binary
- Comparisons
- Casting Conversions (i.e. int32 to float64)
- Boolean mask, Fancy Index mask set/get (i.e., a = b[filter])
- Reduce: all reduce functions sped up
- Sorting (espec lexsort)
- Putmask, searchsorted, hstack, interp
- + Added Hashing, ismember, group creation

Suggested Additions to Array

- Hashing Class

Ledger

Categorical

“groupby apply” – “cat apply”