PyData Array Recommendations

Thomas Dimitri

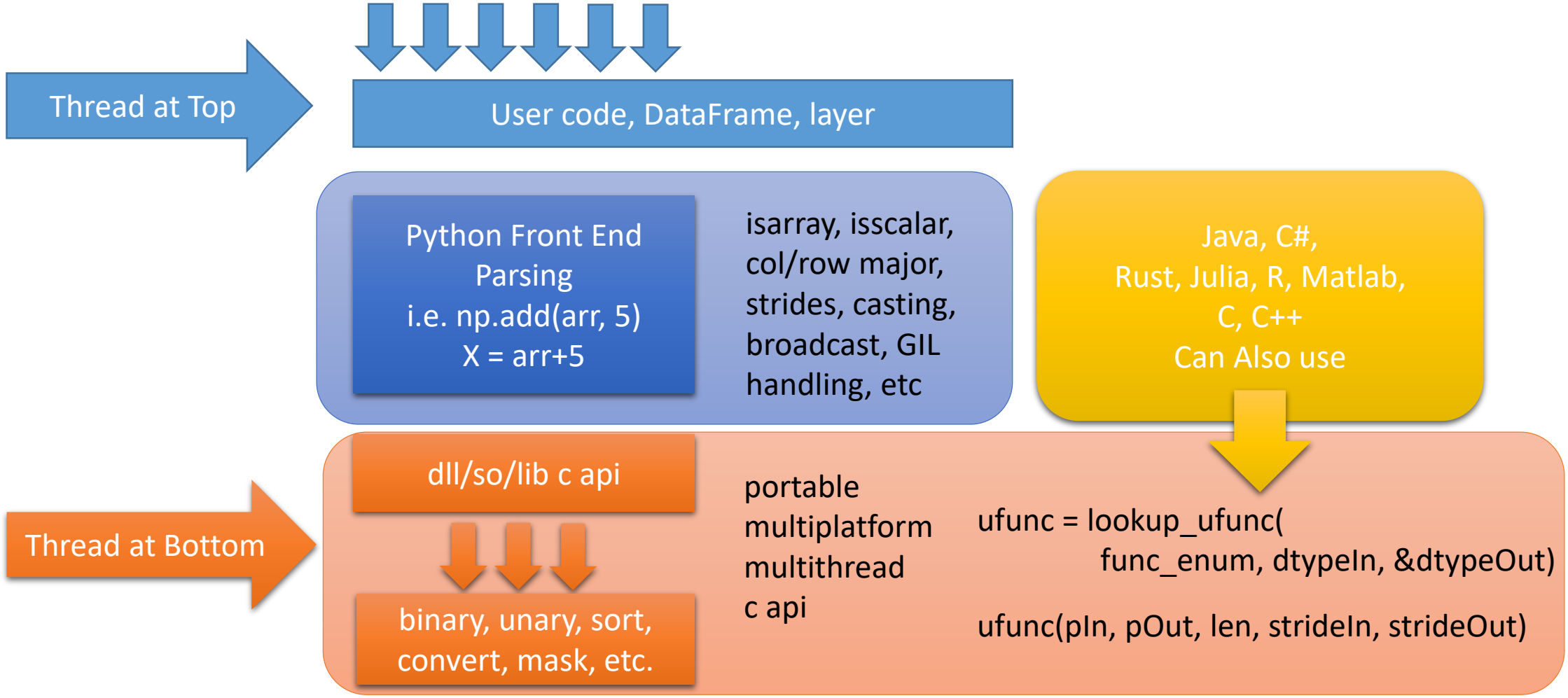
Background

- 30+ years professional software engineer – network protocols, user interfaces, compilers, file formats, pci cards, bios, trading engines, scanners, architectural software, ran 2 software companies, software patents, c/c++, java, c#, python
- 14 year consultant for SIG currently tasked with unifying large data analytics and framework
- 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, portable FileFormat with stacking

Array Recommendations (in order of importance)

- Multithread back end engine and separate out portable c lib
- Introduce multikey categorical flavors with grouping routines
- New loops: groupby loops/ partition loops/ closer JIT coupling
- Subclass before making another dtype: Rework Date/Time classes
- Ledger + other ways to analyze performance
- Introduce new routines (hashing) + Invalids
- Dtypes for variable length string: ucs1,2,4. Bitmask.
- Hooks for display/storage/setname/getname

Threading Model



Threading Model for Arrays

- Multicore/Shared memory same computer, same process
- Possible tuning: Thread count, NUMA, Process Affinity
- Wakeup (futex) – important: selective thread wake up
  - Some routines just need 2 worker threads, more threads is not going to speed anything up for absolute value. Memory bound problem.
- Calibration of worker threads
- Array Recycling (nec for speed or page faults)

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)
- In many low level routines want to get sum of Boolean mask (np.add.reduce(mask)) – should really just call SumBoolean()
- Threading while threading (have to know inside threaded routine)
- Be able to turn on “Ledger” to see what is getting called, performance

Array Routines We Threaded

- Basic Math both Unary and Binary (abs, add, sqrt, etc.)
- 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, multithreaded multikey indirect merge sort)
- putmask, where, searchsorted, hstack, interp
- + Added Hashing, ismember, group creation

Example dir of FastArray
which subclasses np.ndarray (inherits)

FastArray

'abs','all','any','apply','apply_numba','argmax','argmin','argpartition','argpartition2','argsort','astype',
'base','between','byteswap','choose','clip','clip_lower','clip_upper','compress','conj','conjugate','copy','copy_invalid','count',
'crc','ctypes','cummax','cummin','cumprod','cumsum','data','describe','diagonal','diff','differs','display_query_property',
'dot','dtype','dump','dumps','duplicated','ema_decay','fill','fill_backward','fill_forward','fill_invalid','fillna','flags','flat',
'flatten','get_name','getfield','imag','info','inv','iscomputable','isfinite','isin','isinf','isna','isnan','isnanorzero',
'isnormal','isnotfinite','isnotinf','isnotnan','isnotnormal','issorted','item','itemset','itemsize','map','map_old','max','mean',
'median','min','move_argmax','move_argmin','move_max','move_mean','move_median','move_min','move_rank','move_std','move_sum',
'move_var','nanargmax', 'nanargmin','nanmax','nanmean','nanmin','nanrankdata','nanstd','nansum','nanvar','nbytes','ndim',
'newbyteorder','nonzero','normalize_minmax', 'normalize_zscore','notna','numbastring','nunique','partition','partition2','prod','ptp',
'push','put','rankdata','ravel','real','register_function','repeat','replace','replacena','reshape','resize','rolling_mean','rolling_nanmean',
'rolling_nanstd','rolling_nansum','rolling_nanvar','rolling_std','rolling_sum',
'rolling_var','round','sample','save','searchsorted','set_name','setfield','setflags','shape','shift','sign','size','sort',
'squeeze','std','str','str_append','strides','sum','swapaxes','take','tile','timewindow_prod','timewindow_sum','tobytes','tofile',
'tolist','tostring','trace','transitions','transpose','trunc','unique','var','view','where'

Categorical Class (not a new dtype)

- Subclasses from (FastArray, GroupByOps)
- Take most common categorical, of single string array
  - mycat=Cat(stringarray)
  - Often looks like string, sometimes integer bins
  - We chain ._fa or _.np to view FastArray or np.ndarray if we want to force integer bin mode
  - Can chain .str for string or .date for Dateclass to
- Creation can take two paths:
  - Via a sort or lexsort
  - Via hashing (faster for low cardinality)
- Once uniques are discovered, the high/low unique ratio count known important for some algorithms
- Have to be able to filter out (we reserve 0) so base 1 indexing

Categorical has Grouping object also inherits from GroupByOps

- Grouping object:

'apply','apply_helper','as_filter','base_index','catinstance','copy','copy_from','count','gbkeys','get_name','ifirstgroup','ifirstkey','igroup',
'igroupreverse','ikey','ilastkey','inextkey','iprevkey','iscategorical','isdirty','isdisplaysorted','isenum','isin','ismember','ismultikey','isordered',
'isortrows','issinglekey','ncountgroup','ncountkey','newclassfrominstance','newgroupfrominstance','onedict','pack_by_group','packed',
'possibly_recast','register_functions','regroup','set_dirty','set_name','shrink','sort','unique_count','uniquedict','uniquelist'

dir of Categorical class (inherits from FastArray and GroupByOps)

'abs','agg','aggregate','align','all','any','apply','apply_nonreduce','apply_numba','apply_reduce','argmax',
'argmin','argpartition','argpartition2','argsort','as_filter','as_singlekey','as_string_array','astype','auto_add_off','auto_add_on','base','base_index','between',
'byteswap','categories','categories_equal','category_add','category_array','category_codes','category_dict','category_mapping','category_mode',
'category_remove','category_replace','choose','clip',
'clip_lower','clip_upper','compress','conj','conjugate','contains_np_arrays','copy','copy_invalid','count','count_uniques','crc','ctypes','cumcount',
'cummax','cummin','cumprod','cumsum','data','describe','diagonal','diff','differs','display_convert_func','display_query_properties','doc','dot',
'dtype','dump','dumps','duplicated','ema_decay','ema_normal','ema_weighted','expand_any','expand_array','expand_dict','fill','fill_backward','fill_forward',
'fill_invalid','fillna','filter','filtered_name','filtered_set_name','filtered_string','findnth','first','first_bool','first_fancy','flags','flat','flatten',
'from_bin','from_category','gb_keychain','get_groupings','get_header_names','get_name','getfield','groupby_data','groupby_data_clear',
'groupby_data_set','groupby_reset','grouping','grouping_dict',
'groups','head','hstack','ifirstkey','ikey','ilastkey','imag','info','inv','invalid_category','invalid_set','iscomputable','isenum',
'isfiltered','isfinite','isin','isinf','ismultikey','isna','isnan','isnanorzero','isnormal','isnotfinite','isnotinf','isnotnan','isnotnormal',
'issinglekey','issorted','item','itemset','itemsize','iter_groups','key_from_bin','last','lock','map','map_old','mapping_add','mapping_new',
'mapping_remove','mapping_replace','max','mean','median','min','mode','move_argmax','move_argmin','move_max','move_mean','move_median','move_min',
'move_rank','move_std','move_sum','move_var','nan_index','nanargmax','nanargmin','nanmax','nanmean','nanmedian','nanmin','nanrankdata','nanstd',
'nansum','nanvar','nb_ema','nb_min','nb_sum','nb_sum_punt','nbytes','ndim','newbyteorder','newclassfrominstance','ngroup','nonzero','normalize_minmax',
'normalize_zscore','notna','nth','null','numbastring','nunique','ohlc','one_hot_encode','ordered','partition','partition2','prod','ptp',
'push','put','rank','rankdata','ravel','real','register_function','register_functions','repeat','replace','replacena','resample','reshape','resize',
'rolling_count','rolling_diff','rolling_mean','rolling_nanmean','rolling_nanstd','rolling_nansum','rolling_nanvar','rolling_shift','rolling_std','rolling_sum','rolling_var','round',
'sample','save','searchsorted','sem','set_name','setfield','setflags','shape','shift','shift_cat','shrink','sign','size',
'sort','sort_gb','sorted','squeeze','std','str','str_append','strides','sum','swapaxes','tail','take','tile',
'timewindow_prod','timewindow_sum','tobytes','tofile','tolist','tostring','trace','transform','transitions','transpose','trimbr','trunc','unique',
'unique_count','unique_repr','unlock','var','view','where'

Cat example 1

- Example Cat
- isin filter
- can trim
- very fast str ops
- has groupbyops

```
In [41]: symbol
Categorical([GOLD, GOLD, GOLD, GOLD, GOLD, ..., QURE, QURE, QURE, QURE, QURE]) Length: 82212305
  FastArray([1249, 1249, 1249, 1249, 1249, ..., 2336, 2336, 2336, 2336, 2336], dtype=int16) Base Index: 1
  FastArray([b'A', b'AA', b'AAL', b'AAN', b'AAOI', ..., b'ZUO', b'ZVO', b'ZYME', b'ZYNE', b'ZYXI'], dtype='|S16') Unique count: 3201

In [42]: symbol[['GOLD','AAPL']]
FastArray([ True,  True,  True, ..., False, False, False])

In [43]: symbol.shrink(['AAPL','IBM','QURE'])
Categorical([Filtered, Filtered, Filtered, Filtered, Filtered, ..., QURE, QURE, QURE, QURE, QURE]) Length: 82212305
  FastArray([0, 0, 0, 0, 0, ..., 3, 3, 3, 3, 3]) Base Index: 1
  FastArray([b'AAPL', b'IBM', b'QURE'], dtype='|S4') Unique count: 3

In [44]: symbol.str.lower
FastArray([b'gold', b'gold', b'gold', ..., b'qure', b'qure', b'qure'],
          dtype='|S16')

In [45]: symbol.sum(arange(len(symbol)))
*b7o_Header_Symbol             col_0
------------------   --------------
A                       966489024152
AA                     2063505633016
AAL                   26129178449080
AAN                     316412972621
AAOI                    252219031198
AAON                     16269136409
AAP                     866498166991
AAPL                 287286361753363
AAT                        137698846
```

Cat example 2

can flip view

get uniques

Grouping object

igroup is fancy index

has apply power

```
In [53]: symbol._np
array([1249, 1249, 1249, ..., 2336, 2336, 2336], dtype=int16)

In [54]: symbol.categories()
FastArray([b'Filtered', b'A', b'AA', ..., b'ZYME', b'ZYNE', b'ZYXI'],
          dtype='|S16')

In [55]: symbol.grouping

_iKey: [1249 1249 1249 ... 2336 2336 2336]
iFirstKey: None
iLastKey: None
iNextKey: None
_unique_count: 3201
_grouping_dict: None
_grouping_unique_dict: {'symbol': FastArray([b'A', b'AA', b'AAL', ..., b'ZYME', b'ZYNE', b'ZYXI'],
          dtype='|S16')}
_enum: None
_categorical: False
isenum: False
isdirty: False
_catinstance: None

_packed: True
iGroup: [34051548 34051549 34051550 ... 40172985 40172986 40172987]
iFirstGroup: [       0        0    17272 ... 82202231 82208329 82210691]
nCountGroup: [    0 17272 38129 ...  6098  2362  1614]

_gbkeys: None
_sort_display: False
Ordered: True
_base_index: 1

In [56]: symbol.apply_reduce(lambda x:x.mean(), {'arange':arange(len(symbol)), 'ones':ones(len(symbol))})
*b7o_Header_Symbol      arange   ones
------------------   ---------   ----
A                    5.596e+07   1.00
AA                   5.412e+07   1.00
AAL                  5.046e+07   1.00
AAN                  4.919e+07   1.00
AAOI                 4.725e+07   1.00
AAON                 5.959e+07   1.00
AAP                  5.322e+07   1.00
AAPL                 4.676e+07   1.00
```

New Loops

- Existing:
  - Contig math vector loops (stride == itemsize)
    - Able to hit mm_256_add_ routines, etc.
  - Strided loops
    - Can also use vector routines via gather intrinsic
- New: Groupby Loops
  - See “igroup” – fancy index in order of uniques
  - For example if ‘AAPL’ is first category then it might exist in 10,000 rows scattered about 1 million row array. Similar to what lexsort returns now to sum up all ‘AAPL’ can just use the index to get to the data (no need to copy or reshuffle data)
- New: Partition Loops
  - Often used after sort_inplace. Now groups are in order – might be many groups, perhaps 1 million… want to ‘fake’ slice arrays fast to do operations
- Loop over JIT kernels
  - Numba has pgroup and threads.. Can use that to create own groupby kernel

Suggested Additions to Array

- Hashing Class
  - Example crc
  - Like np.add.reduce… np.hash.reduce
  - ismember function (see matlab routine)
- Searchsorted with kwarg ‘exact’
  - Must be exact match or give other index, like -1

Ledger

- Need to show example
- Records all operations, can time them
- Can see when upcasting occurs
- Can see which routines take longest
- Built into portable C code (so all languages can use)

New dtypes

- Variable length strings
- Bitmask Boolean
- Otherwise use classes to build up from common dtypes

Display/Serialization Hooks

- Known __ attributes to get/set meta data for serialization
- Hooks to help with autocomplete
- Or other display – left justified, abbreviated, etc.