

PEP 3128: Python BList

CSCI 3155

Fadhil Suhendi, Zhilli Yang, Thomas Kelly

Introduction of BList

PEP 3128:
Python BList

CSCI 3155

- New Data Structure for Python
- Based on principles of B+tree
- Blist has array-like and tree-like aspects
- Replace existing list type
- This data structure was rejeted

Motivation

PEP 3128:
Python BList

CSCI 3155

- For large inputs
- Offers array-like performance on small list
- It also offers asymptotic performance for deleting and inserting

Comparison with List

PEP 3128:
Python BList

CSCI 3155

- Comparison based on performance

PEP 3128:
Python BList

CSCI 3155

Comparison with List

PEP 3128:
Python BList

CSCI 3155

- List is using dynamically array type
- Blist is using a flexible, tree structure
- Blist uses twice memory usage as array-list
- Blist root node has at least 2 childrens
- List has fewer than 2 childrens

About B+tree

PEP 3128:
Python BList

CSCI 3155

- N-ary tree
- Large number of children per node
- No data in internal nodes
- Leaves nodes are in the same level
- Data pointers

ABout B+tree

PEP 3128:
Python BList

CSCI 3155

Implementation

PEP 3128:
Python BList

CSCI 3155

- Without using blist

```
items = [5,6,2]  
more_items = function_that_returns_a_list()
```

- Using blist:

```
from blist import blist  
items = ([5,6,2])  
more_items = blist(function_that_returns_a_list())
```

Examples of blist

PEP 3128:
Python BList

CSCI 3155

```
from blist import *  
x = blist([0]) # x is a blist with one element  
x *= 2**29 # x is a blist with > 500 million elements  
x.append(5) # append to x  
y = x[4:-234234] # Take a 500 million element slice from x  
del x[3:1024] # Delete a few thousand elements from x  
  
from blist import sortedlist  
my_list = sortedlist([3, 7, 2, 1]) #sortedlist([1, 2, 3, 5, 7])  
my_list.add(5) #sortedlist([1, 2, 3, 5, 7])
```

Pros

PEP 3128:
Python BList

CSCI 3155

- BList is useful for intermediate-users, but not for beginners users.
- Blist has a better performance than array-based list
- Matching the list-API reduces the learning curve to near-zero

Cons

PEP 3128:
Python BList

CSCI 3155

- Performance & desirability in real-world applications has yet to be proven – more testing is required
- Increasing the number of data types available to developers makes their job more difficult
- Adding BList to Python's default library may have an impact on extension modules, requiring extensive debugging
- There are some important use cases where BLists are actually slower than regular Lists
- The array-based List code is easy to maintain & simple to understand, while BLists would present a learning curve for many developers

Performance of Blist: Simple Addition

PEP 3128:
Python BList

CSCI 3155

Performance of Blist: For Loop Iteration

PEP 3128:
Python BList

CSCI 3155

Performance of Blist: Sorting a Reversed List

PEP 3128:
Python BList

CSCI 3155

Performance of Blist: FIFO Operations

PEP 3128:
Python BList

CSCI 3155

... So why was it rejected?

PEP 3128:
Python BList

CSCI 3155

- Various reasons, but mostly because it isn't very user-friendly
- "There is too much value in a simple C API, low space overhead for small lists, good performance [in] common use cases, and having performance that is easily understood. The BList implementation lacks these virtues and it trades-off a little performance in common cases ... for much better performance in uncommon cases." – Raymond Hettinger
- So even though it outperforms List on many different tasks involving very large numbers of elements, Python rejected BList because it was too difficult for inexperienced users to use & understand
- This seems to suggest that Python devs prefer user-friendly code over more powerful code – "If it ain't broke, don't fix it!"

Other reasons. . .

PEP 3128:
Python BList

CSCI 3155

- So even though it outperforms List on many different tasks involving very large numbers of elements, Python rejected BList because it was too difficult for inexperienced users to use & understand
- This seems to suggest that Python devs prefer user-friendly code over more powerful code – “If it ain’t broke, don’t fix it!”

Summary

PEP 3128:
Python BList

CSCI 3155

- Blist is a new data structure
- The strongest part of BList is about performances
- The weakness part of BList is about memory usage

Conclusion

PEP 3128:
Python BList

CSCI 3155

- In 2008 BList was rejected by community. However, due to the powerful performance of Blist, BList is still released as an extension module of python programming language. If many python users will use Blist, it is possible that in the future, Blist will be considered to replace the the array-based list.

References

PEP 3128:
Python BList

CSCI 3155

- <https://www.python.org/dev/peps/pep-3128/#id6>
- <https://pypi.python.org/pypi/blist/>
- <http://legacy.python.org/dev/peps/pep-3128/>
- <http://stutzbachenterprises.com/performance-blist>
- <http://byumcl.bitbucket.org/bootcamp2013/labs/performance>