Arkouda Set Operations Optimizations

Ben McDonald and Elliot Ronaghan

CHIUW 2021

June 4th, 2021

Overview

- Set Operations
- NumPy and Arkouda
- Arkouda design
- Optimization of set operations
- Performance results

What are set operations?

- Like you learned about in math: union, intersect, difference, xor
- Commonly used in Arkouda and data science in general

```
>>> import arkouda as ak
>>> ak.connect()
>>> a = ak.array([1,2,3])
>>> b = ak.array([2,3,4])
>>> ak.intersect1d(a,b)
array([2, 3])
>>> ak.union1d(a,b)
array([1, 2, 3, 4])
>>> ak.setxor1d(a,b)
array([1, 4])
>>> ak.setdiff1d(a,b)
array([1])
>>>
```

Background: NumPy, Data Science

NumPy

- Widely used Python package for array operations
- Uses pre-compiled C code

Data science

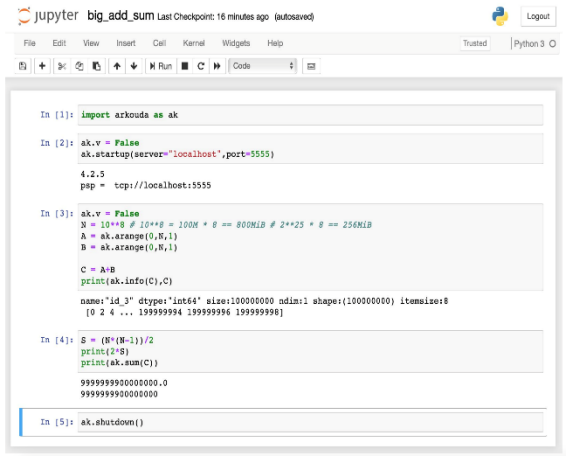
- Analyzing large sets of data
- Provides statistics, trends, information

Background: Arkouda

- Open-source Python package backed by Chapel
- Similar functionality to NumPy, but scalable
- Enables interactive supercomputing for data science
- Interoperates with NumPy

Design of Arkouda

Python3 Client



Arkouda User

Client to server communication

Chapel Server

```
File Edit Options Buffers Tools chpl Help
  // returns the exclusive-or of 2 arrays
  proc setxor1d(a: [] int, b: [] int, assume_unique: bool) {
    //if not unique, unique sort arrays then perform operation
    if (!assume_unique) {
      var a1  = uniqueSort(a, false);
      var b1  = uniqueSort(b, false);
      return  setxor1dHelper(a1, b1);
    }
    return setxor1dHelper(a,b);
  }

  // Gets xor of 2 arrays
  // first concatenates the 2 arrays, then
  // sorts and removes all values that occur
  // more than once
  proc setxor1dHelper(a: [] ?t, b: [] t) {
    const aux = radixSortLSD_keys(concatset(a,b));
    const ref D = aux.domain;

    // Concatenate a `true` onto each end of the array
    var flag = makeDistArray(aux.size+1, bool);
    const ref fD = flag.domain;

    flag[fD.low] = true;
    flag[fD.low+1..fD.high-1] = aux[..D.high-1] != aux[D.low+1..];
    flag[fD.high] = true;

    var mask;
    {
      mask = sliceTail(flag) & sliceHead(flag);
    }

    var ret = boolIndexer(aux, mask);

    return ret;
  }

  // returns the set difference of 2 arrays
  proc setdiff1d(a: [] int, b: [] int, assume_unique: bool) {
    //if not unique, unique sort arrays then perform operation
    if (!assume_unique) {
      var a1  = uniqueSort(a, false);
      var b1  = uniqueSort(b, false);
      return setdiff1dHelper(a1, b1);
    }
    return setdiff1dHelper(a,b);
  }

  // Gets diff of 2 arrays
  // first checks membership of values in
  // fist array in second array and stores
-UU-:----F1  ArraySetops.chpl   34% L53   Git-master  (Chapel/*l Abbrev) --
```

Optimization of Set Operations

- **Previously**: written purely on the client/Python side
- **Problem**: server communication taking up bulk of execution time
- **Goal**: move operations to purely server-side operations to minimize communication
- **Result**: saw performance improvements ranging from ~10%-90%

Chapel vs. Python

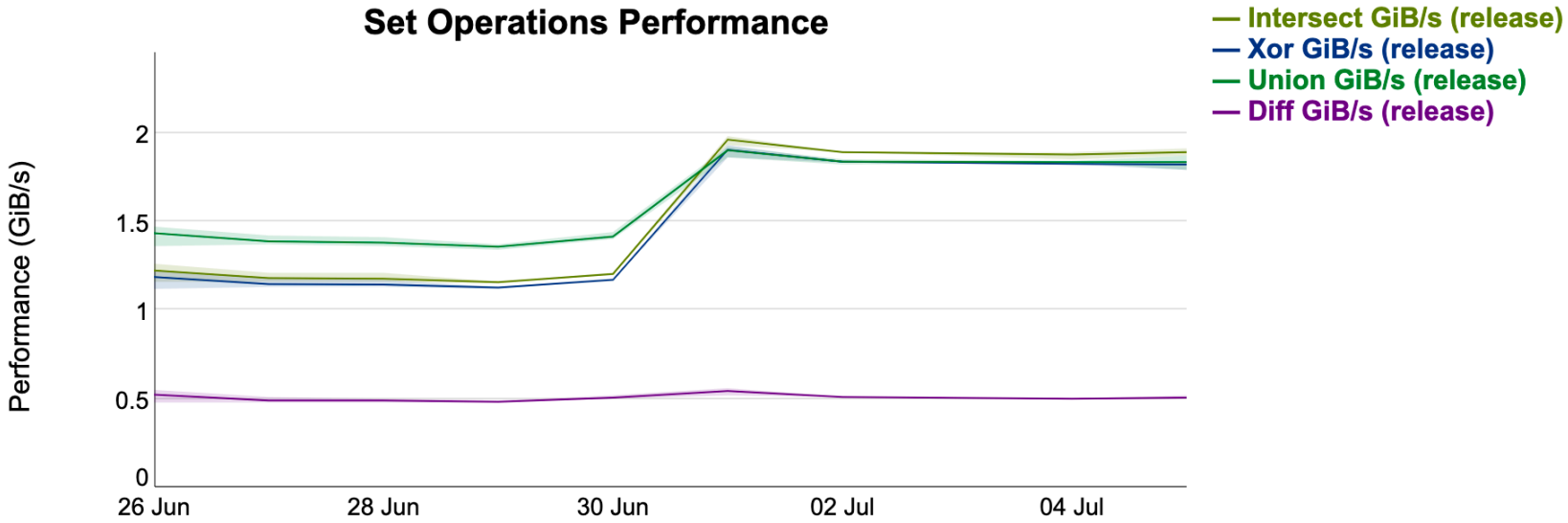
```python
def intersect1d(pda1 : pdarray, pda2 : pdarray,
                                assume_unique : bool=False) -> pdarray:
    if not assume_unique:
        pda1 = unique(pda1)
        pda2 = unique(pda2)
    aux = concatenate((pda1, pda2), ordered=False)
    aux_sort_indices = argsort(aux)
    aux = aux[aux_sort_indices]
    mask = aux[1:] == aux[:-1]
    int1d = aux[:-1][mask]
    return int1d
```

Python intersect

```
proc intersect1d(a: [] int, b: [] int, assume_unique: bool) {
  if (!assume_unique) {
        a  = uniqueSort(a, false);
        b = uniqueSort(b, false);
  }
  var aux = radixSortLSD_keys(concatset(a,b));
  const ref head = aux[..aux.domain.high-1];
  const ref tail = aux[aux.domain.low+1..];
  const mask = head == tail;
  return boolIndexer(head, mask);
}
```

Chapel intersect

Performance Results

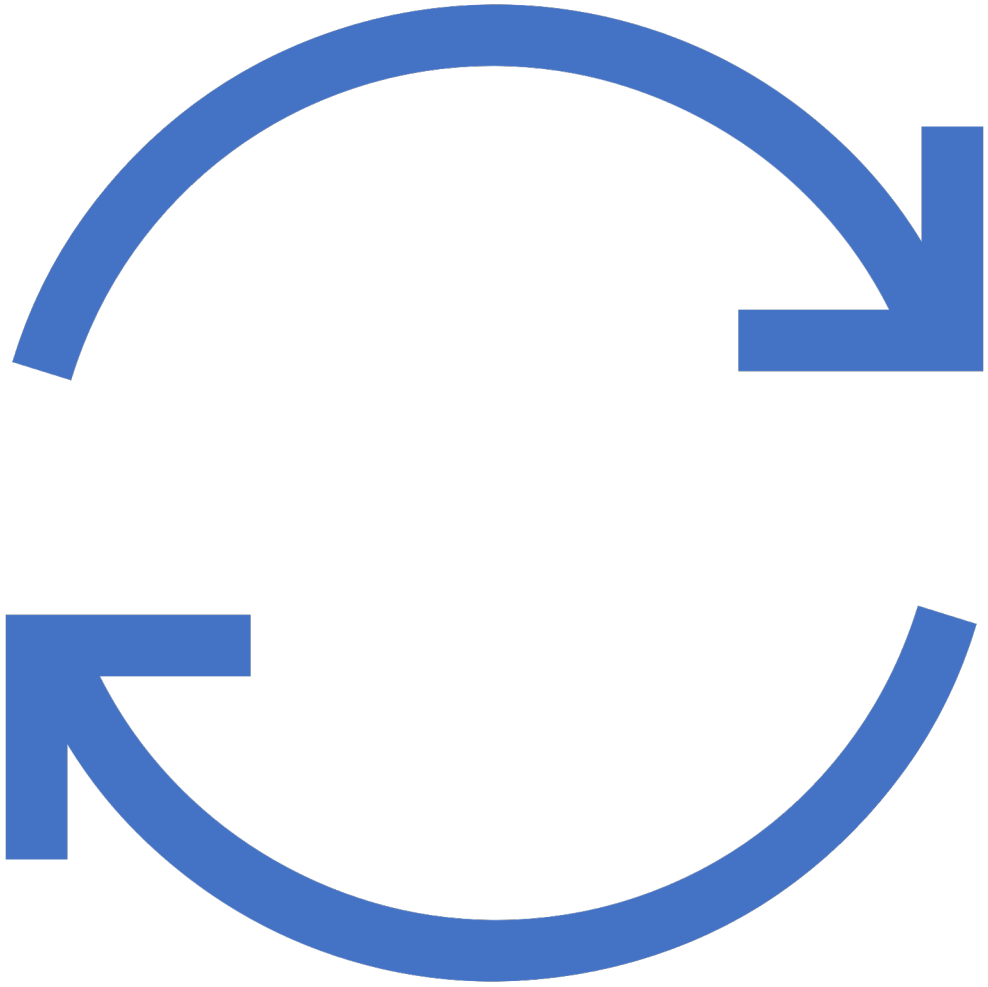


Arkouda Set Operations Performance using 16-locales of a Cray-XC

Operation	Before Changes (GiB/s)	After Changes (GiB/s)	Speedup
Intersect	1.07	2.01	88%
Union	1.12	1.95	74%
Exclusive or	1.05	1.90	81%
Set difference	0.45	0.49	8.9%

Conclusion

- Optimizations like this should be done selectively
- Goal of Arkouda is to enable data scientists to interactively utilize supercomputers



Acknowledgements

- Thank you, Chapel team
- Thank you, Arkouda team

Thank you!

Questions?