Library and Array Improvements

Chapel version 1.19

March 21, 2019

chapel_info@cray.com
chapel-lang.org
@ChapelLanguage

Outline

- Radix Sorting
- Hashed Distribution
- Remote Subdomain Queries
- Distribution Convenience
- Filter/Map/Consume
- 'Random' Improvements
- 'LinearAlgebra' Improvements

Radix Sorting

Radix Sorting: Background

- Sort module is currently a package module
  - Because its interface is not finalized
  - Because the implementation is incomplete
    - It has lacked sorting algorithms with competitive performance
- The sort() function is called from standard modules
  - e.g., for associative domain's sorted() iterator
- The sort() function can accept a comparator
  - Is element A less than, equal to, or greater than element B?
  - Alternatively, what is the 'key' to sort by?

Radix Sorting: This Effort

- Rails Girls Summer of Code project studied radix sorting in Chapel
  - Generated several implementations
  - Led to a straw-man interface proposal
- Extended the sort() comparator API to allow keyPart() for radix sorting
- Added a parallel, in-place radix sort to the Sort module
- sort() now calls radix sort if comparators allow it

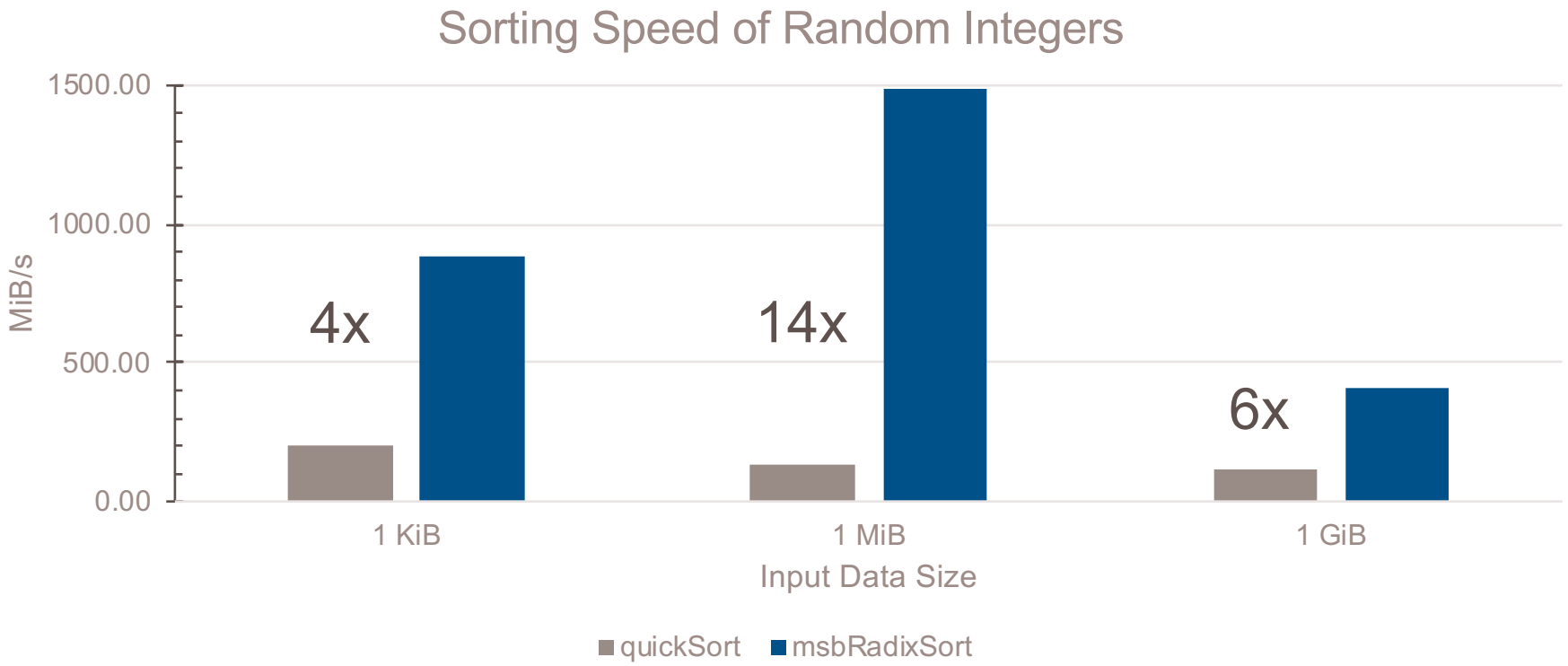
Radix Sorting: Example

```
use Sort;
record MyRecord { var key: int; var value: int; }
record MyKeyComparator {
  proc key(element: MyRecord) {
    return element.key; // now uses radix sorting for integral keys
  }
}
config const n = 10000;
var A: [1..n] MyRecord = [i in 1..n] new MyRecord(i, i*i);
sort(A, new MyKeyComparator());
```

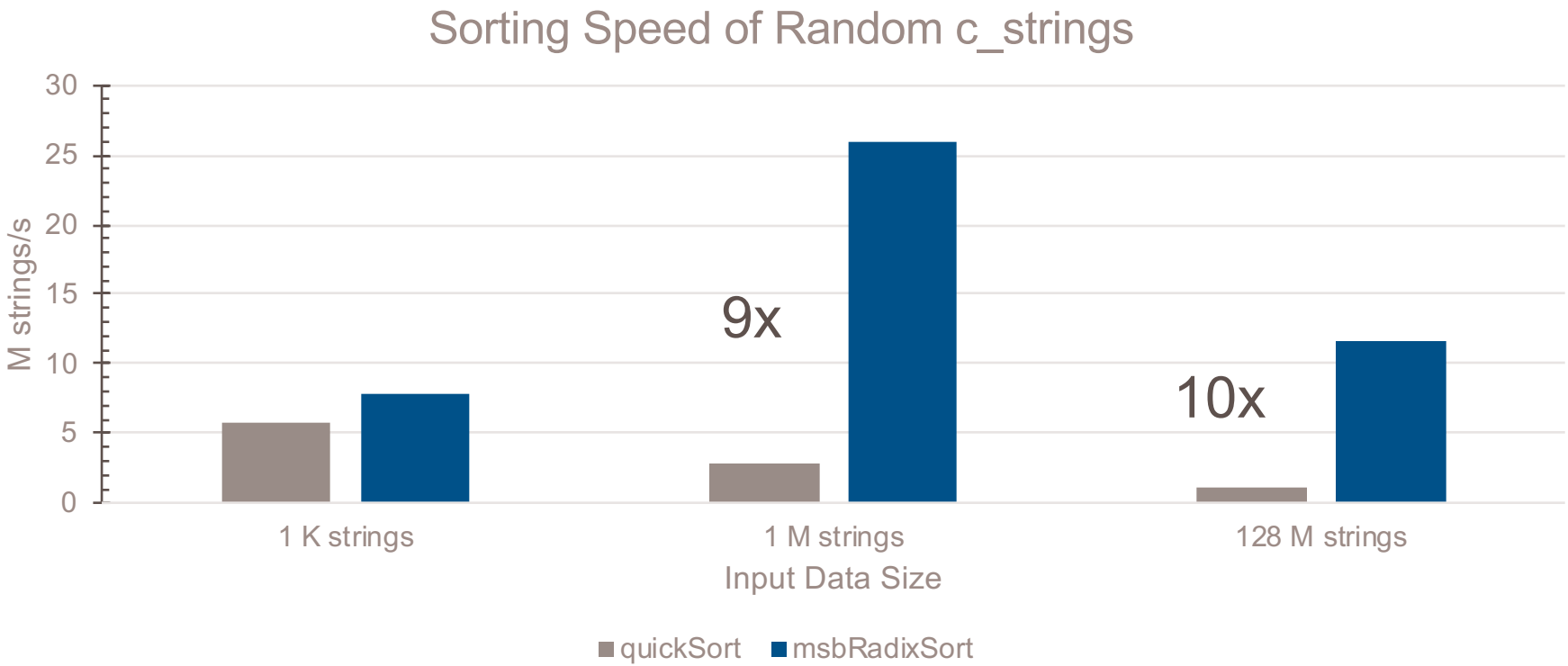

Radix Sorting: Example

```
use Sort;
record MyRecord { var key: c_string; var value: int; }
record MyKeyPartComparator { }
  proc keyPart(element: MyRecord, i: int) {
    var byte = element.key[i-1]; // compute the current key byte
    // has the end been reached? Note, c_strings have a 0 terminator
    var done = if byte != 0 then 0 else -1;
    return (done, byte);
} }
var A:[1..n] MyRecord = ...;
sort(A, new MyKeyPartComparator());
```


Radix Sorting: Impact

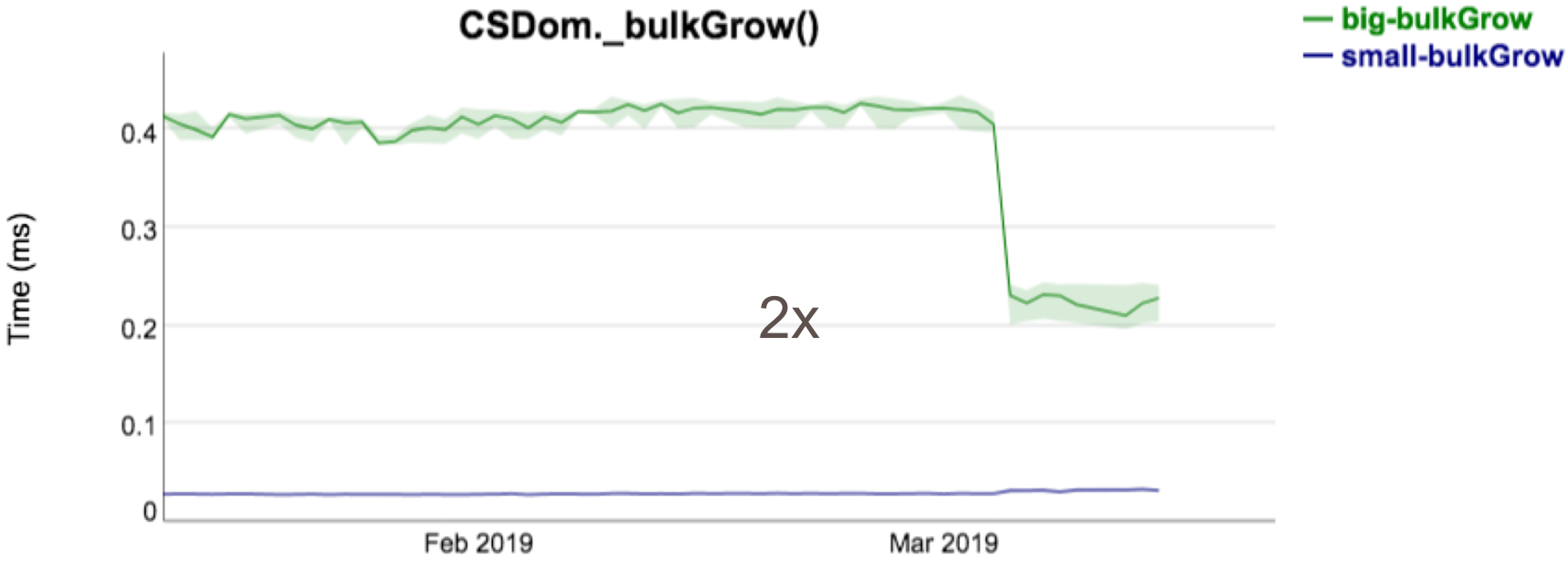


Radix Sorting: Impact



Radix Sorting: Impact

- Significantly improved performance for a sparse domain benchmark:



Radix Sorting: Next Steps

- Explore ways to achieve better performance for heavily skewed data
  - Need to improve parallel load balance
- Investigate alternative parallelization strategies
  - The 'count' and 'shuffle' functions are currently serial
- Support distributed radix sorting

Hashed Distribution

HashedDist: Background

- Distributed associative arrays are important for certain applications
  - e.g. when counting or assigning unique numbers to strings in distributed data

- A prototype distribution for associative arrays was already implemented
  - Used in earlier label propagation study
  - Never promoted out of the test system

HashedDist: This Effort

- Added a new module, ‘HashedDist’, and a new distribution, ‘Hashed’
  - Based on the prototype that was in the testing system
- The ‘Hashed’ distribution:
  - Maps an associative domain and its arrays to a set of target locales
  - Maps each index to a locale based upon its value
  - Can be customized by providing a mapper

HashedDist: This Effort

```
use HashedDist;
var D: domain(string) dmapped Hashed(idxType=string);
// Now D is a distributed associative domain (set) of strings. Add some elements:
D += "one"; D += "two";

var A: [D] int;
// Now A is a distributed associative array (map) from string to int
// Let's iterate over it across all Locales
forall (key, value) in zip(D, A) {
  // do something with the (key, value) pair
}
```


HashedDist: Impact, Next Steps

Impact: Distributed associative arrays and domains are now available

Next Steps:

- Get feedback from users of ‘HashedDist’ and improve the interface
- Improve the implementation
  - Make the domain map implementation complete
  - Support adding indices in bulk

Remote Subdomain Queries

Remote Subdomains: Background

Background:

- Chapel supports subdomain queries on distributed domains/arrays:

```
const myInds = A.getLocalSubdomain();
```

- However, these queries have only been for the current locale ('here')
  - Thus, to query for a remote locale, on-clauses had to be used:

```
var remoteInds: (A.getLocalSubdomain()).type;
on remoteLocale do
  remoteInds = A.getLocalSubdomain();
```

  - Yet, many distributions can compute such queries without communicating

Remote Subdomains: This Effort

This Effort:

- Added support for remote subdomain queries:

```
proc <domain>.localSubdomain(loc: locale = here);
proc <array>.localSubdomain(loc: locale = here);

iter <domain>.localSubdomains(loc: locale = here);
iter <array>.localSubdomains(loc: locale = here);
```

- Used an optional argument to preserve backward-compatibility

Remote Subdomains: Status, Next Steps

Status:

- Added (communication-free) implementations for most major domain maps:
  - Default / local layouts
  - Key distributions: Block, Stencil, Cyclic, Replicated, HashedDist
  - Array views

Next Steps:

- Extend to remaining domain maps: BlockCyclic, Block-Sparse, Dimensional
- Decide whether to retire the procedure forms of the queries
  - Realized that it's broken when a locale is oversubscribed in 'targetLocales'
  - This would permit 'hasSingleLocalSubdomain()' to be retired as well

Distribution Convenience Routines

Distribution Routines: Background, This Effort

Background: Creating distributed domains/arrays can be repetitive

- Block domains frequently declared over same indices as boundingBox

```
const D = {1..m, 1..n} dmapped Block(boundingBox={1..m, 1..n});
var A: [D] real;
```

- Cyclic domains frequently declared with startIdx == domain's low bound

This Effort: Provide convenience routines for Block and Cyclic domains/arrays

- Simplify the common cases

Distribution Routines: Impact, Next Steps

Impact: The common cases for Block and Cyclic are simplified

```
var BlkDom = newBlockDom({1..n, 1..m});
var CycDom = newCyclicDom({1..n, 1..m});
var BlkArr = newBlockArr({1..n, 1..m});
var CycArr = newCyclicArr({1..n, 1..m});
```

Next Steps:

- Look for common usage patterns in other distributions
  - Provide similar convenience functions in those cases
- Continue to refine and improve these helper routines

Filter, Map, Consume on Iterators

Filter, Map, Consume: Background, This Effort

Background: Filter, Map, Consume are common patterns on stream-like data

- These operations are commonly supported in other languages, e.g. Python
- Would be useful for iterators since they yield streams of data

This Effort: Define methods on iterators implementing Filter, Map, and Consume

```
iter iterator.map(function): function.type
iter iterator.filter(function): iterator.type
iter iterator.consume(function): void
```


Filter, Map, Consume: Status, Next Steps

Status: Functional style operations are available for iterators

- Currently requires calling 'these()' to get an iterator from an iterable object

```
var r = 1..17 by 3;
proc even(i: int) return i % 2 == 0;
for i in r.these().filter(even) do … // 4, 10, 16
```

Next Steps:

- Add 'foldL' and 'foldR'
- Add parallel versions of these operations
- Make the functions directly available on iterable objects

Random Module Improvements

Random Module: Background, This Effort

Background: Random sampling was not available in ‘Random’ module

This Effort: Implemented choice() method for sampling from a 1D array

- Supports weighted sampling (*prob*) with or without replacement (*replace*)
- Supports returning a single value, or an N-dimensional array (*size*)

```
use Random;
var stream = makeRandomStream(int);
var ret = stream.choice([1,2,3], prob=[0.1, 0.3, 0.6],
                                 size={1..2, 1..2}, replace=true);
```

- Improved getNext() in order to support 'choice':
  - Added 'getNext(resultType, min, max)' overload to PCG random stream
  - Added bounds-checking to getNext() overloads with min/max arguments

Random Module: Impact, Next Steps

Impact: Improved Random module

- Random sampling is now supported in the Random module
- Extended getNext() functionality and added bounds checking

Next Steps: Extend sampling functionality and provide distribution sampling

- Support sampling from an N-dimensional array
- Support sampling bigint, imaginary, and complex types
- Optimize implementation for sampling from local and distributed arrays
- Support distribution-sampling like Gaussian, Binomial, Poisson, etc.

LinearAlgebra Module Improvements

LinearAlgebra Module: Background, This Effort

Background: LinearAlgebra module provides linear algebra routines in Chapel

This Effort: Made some quality of life improvements to the module

- Added checks for distributed arrays which are not yet supported
- Renamed eigvals() to eigs() since it returns eigenvalues and eigenvectors
  - Kept eigvals() for eigenvalues only
- Stopped transitively using BLAS and LAPACK with LinearAlgebra
  - Prevents a potential collision with 'BLAS.dot()'
- Removed previously deprecated features

LinearAlgebra Module: Status, Next Steps

Status:

- LinearAlgebra module is improved
  - Fewer confusing errors
  - Easier to use

Next Steps:

- Continue to improve LinearAlgebra module
  - Distributed support
  - GPU support
  - More linear algebra routines (native and BLAS/LAPACK)

For More Information

For a more complete list of library and array changes in the 1.19 release, refer to 'Standard Modules / Library', 'Package Modules' and 'Standard Domain Maps' sections in the CHANGES.md file.

SAFE HARBOR STATEMENT

This presentation may contain forward-looking statements that are based on our current expectations. Forward looking statements may include statements about our financial guidance and expected operating results, our opportunities and future potential, our product development and new product introduction plans, our ability to expand and penetrate our addressable markets and other statements that are not historical facts.

These statements are only predictions and actual results may materially vary from those projected. Please refer to Cray's documents filed with the SEC from time to time concerning factors that could affect the Company and these forward-looking statements.

THANK YOU

QUESTIONS?


chapel_info@cray.com


@ChapelLanguage


chapel-lang.org


cray.com


@cray_inc


linkedin.com/company/cray-inc-