Towards Radix Sorting in the Chapel Standard Library

Michael Ferguson

CHIUW 2019

Outline

- Interface Design for Radix Sort
- Exploring Parallel Radix Sort Algorithms
- Comparing Single-Locale Performance
- Distributed Sorting

© 2019 Cray Inc.

Interface Design for Radix Sort

© 2019 Cray Inc.

Interface Design for Radix Sort

- Programming languages usually offer a sort library
- These normally include the ability to specify a comparison function
  - but that limits implementations to comparison sorting algorithms

- Some sorting libraries also allow specifying a key to sort by
  - easy to sort by a specific field in a structure
  - but no help for variable length keys

- Would like to improve on this situation
  - to enable radix sorting in most cases
  - including the common case of sorting by variable-length strings

© 2019 Cray Inc.

The .key method

```
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());
```

© 2019 Cray Inc.

The .keyPart method

```
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());
```

© 2019 Cray Inc.

How .keyPart supports variable length keys

- Say we are sorting strings
- Which comes first?
  - "badminton"
  - "bad"
- keyPart returns a tuple to indicate the ordering here
- Tuple consists of (section, part)
  - (-1, part) ⇢ sort this key before those with more data
  - (0, part) ⇢ sort based on key data in part
  - (1, part) ⇢ sort this key after those with more data

© 2019 Cray Inc.

Exploring Parallel Radix Sort Algorithms

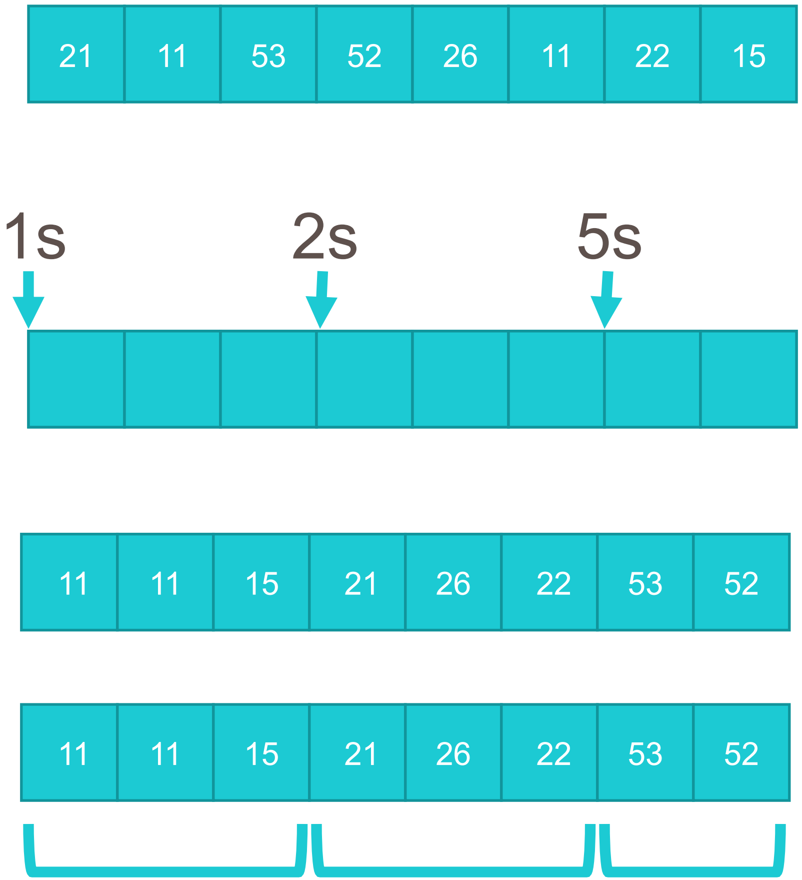
© 2019 Cray Inc.

Algorithms Explored

- Two most-significant digit first counting radix sorts
  - Recursive algorithm with serial bucketize inspired by [1]
  - Two-array algorithm with parallel bucketize

[1] Peter M McIlroy, Keith Bostic, and M Douglas McIlroy. 1993. Engineering radix sort. Computing systems 6, 1 (1993), 5–27.

© 2019 Cray Inc.



Count 1st digit:

3x '1' 3x '2' 2x '5'

Scan to find bin starts

Bucketize: move into bins

Continue with next digit within each bin

© 2019 Cray Inc.

Recursive Algorithm

```
proc recursiveSort(start, end, A, digit) {
  // local arrays for byte counts and offsets
  var counts, offsets : [0..#256] int;
  parallelCountAndScan(...)
  sequentialInPlaceBucketize(...);  // repeated swapping of current item
  forall bins do
    // recursively calls algorithm
    recursiveSort(binStart, binEnd,
                  A, digit+1);
}
```

Drawbacks:

- Limited parallel speedup
- Lots of array allocations
- Not a stable sort

© 2019 Cray Inc.

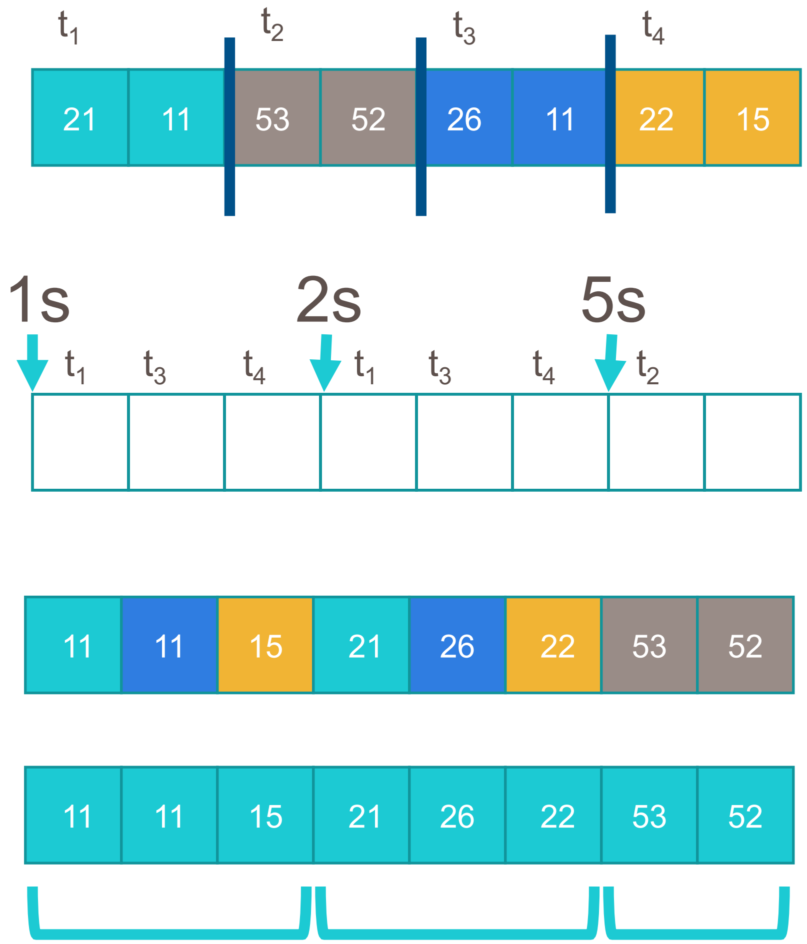
Iterative Algorithm

```
var counts, offsets : [0..#256] int; // just one of each per sort call
proc twoArraySort(start, end, A, Scratch, digit) {
  bigTasks.push( ... );
  while !bigTasks.isEmpty() {
    task = bigTasks.pop();
    parallelCountAndScan(...);
    parallelBucketizeToScratch(...)
    for bins do append task to bigTasks or smallTasks
  }
  forall tasks in smallTasks do baseCaseSort(...)
}
```

Drawbacks:

Uses 2n space

© 2019 Cray Inc.



Count 1st digit:

3x '1' 3x '2' 2x '5'

Scan to find bin starts

Bucketize: move into bins

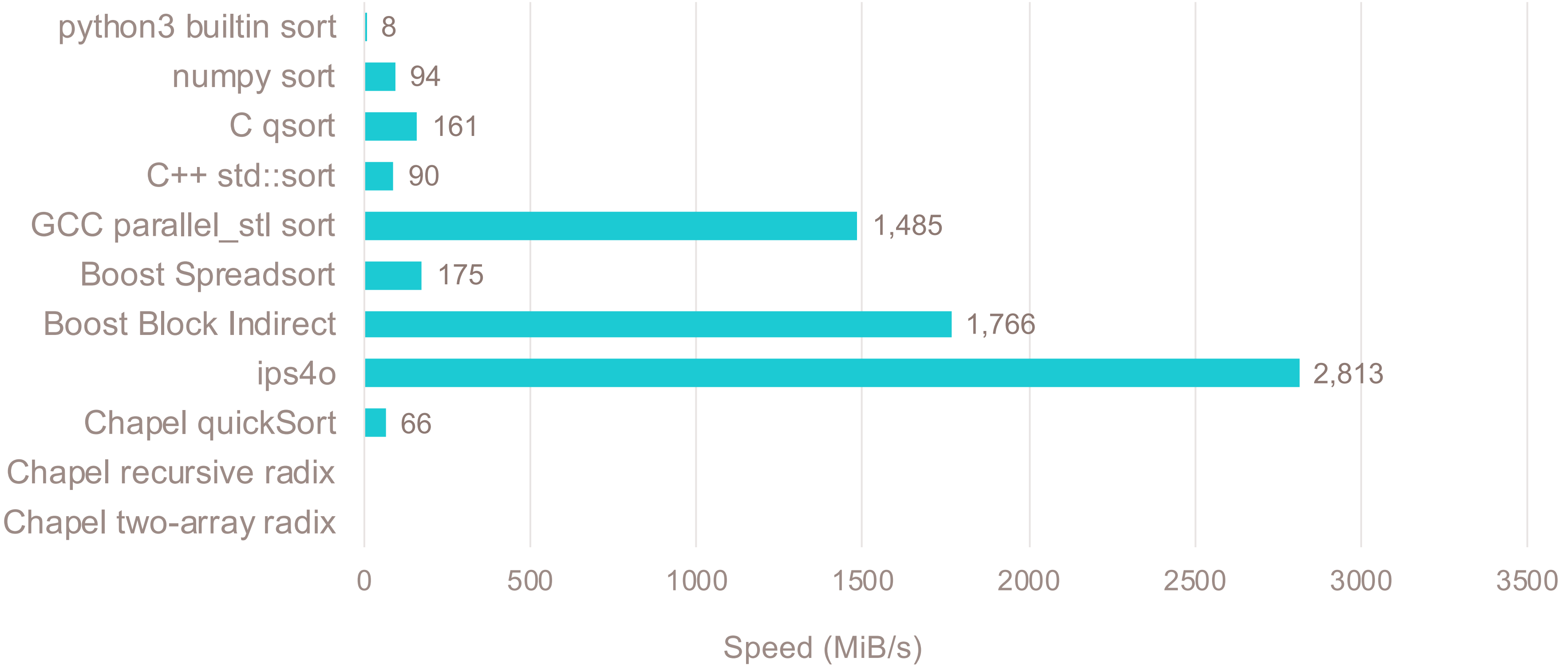
Continue with next digit within each bin

© 2019 Cray Inc.

Comparing Single-Locale Performance

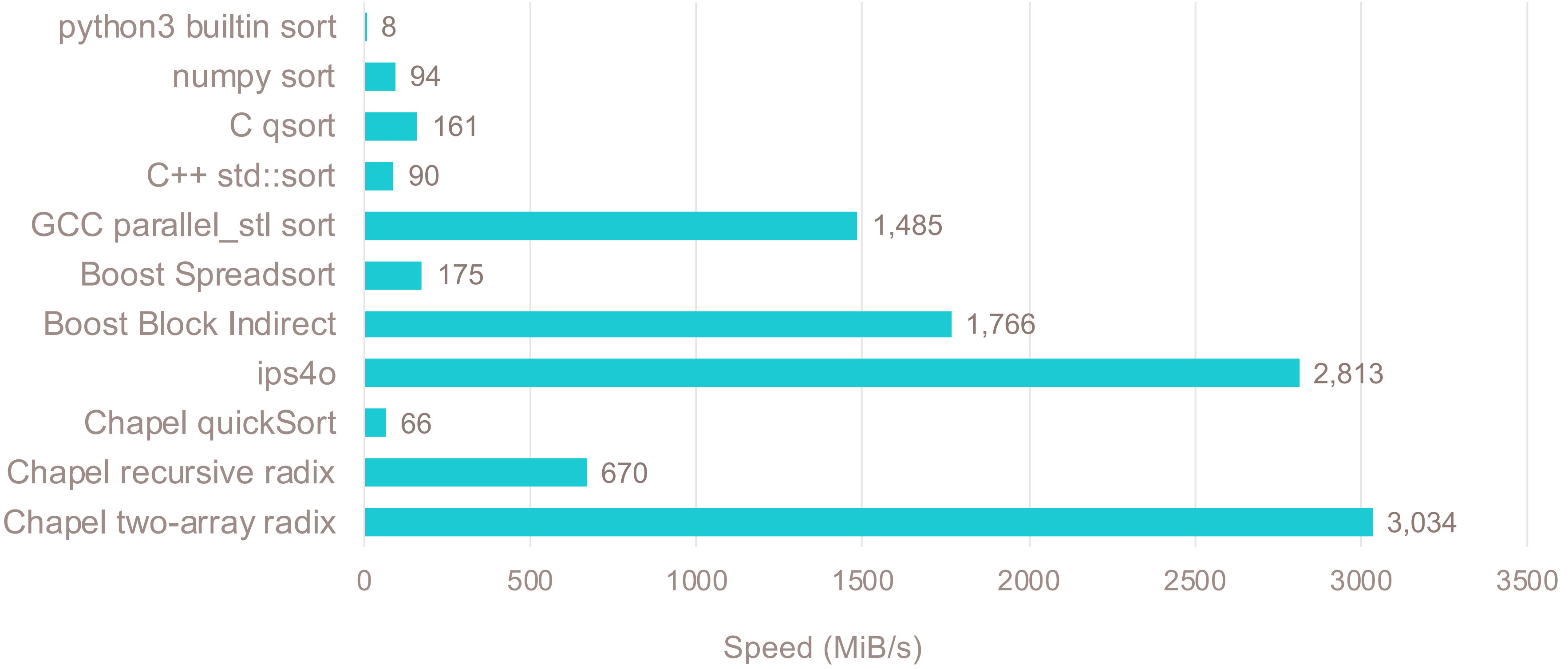
© 2019 Cray Inc.

Sorting 1GiB of random uint(64): Broadwell



© 2019 Cray Inc.

Sorting 1GiB of random uint(64): Broadwell



© 2019 Cray Inc.

Distributed Sorting

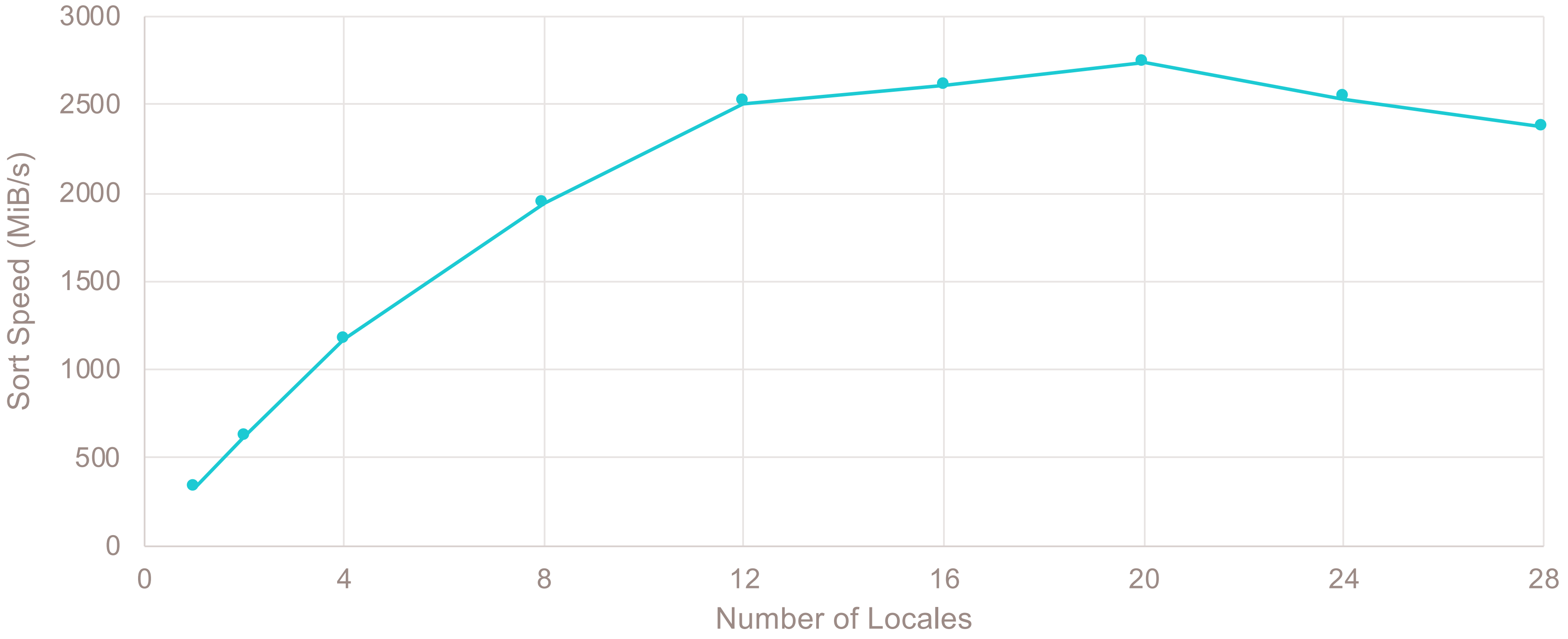
© 2019 Cray Inc.

Distributed Two-Array Algorithm

```
proc distSort(start, end, A, Scratch, digit) {
  while !distTasks.isEmpty() {
    countAndBucketizeLocalDataToScratch(...) on each locale
    for bins do append task to distTasks or localTasks
  }
  forall tasks in localTasks do twoArraySort(...)
}
```

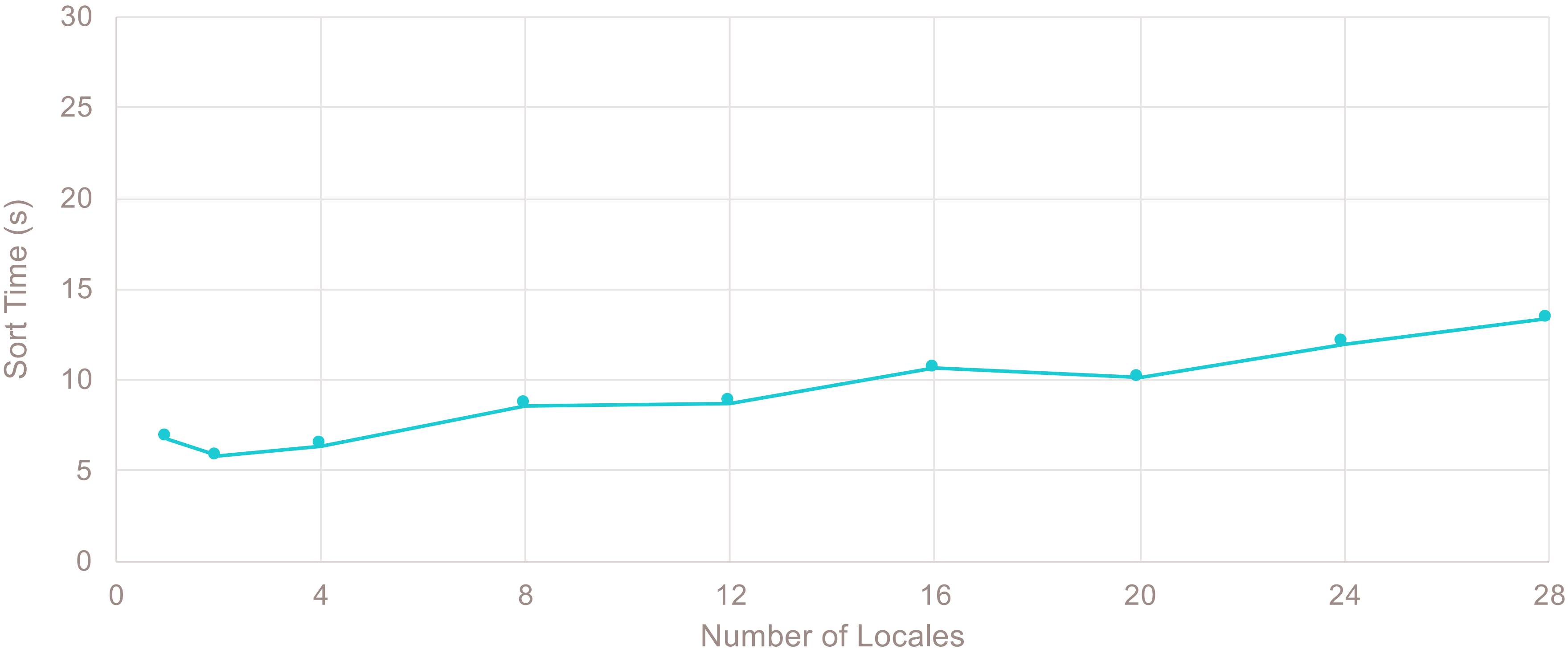
© 2019 Cray Inc.

Strong Scaling on Broadwell, sorting 100 M uint



© 2019 Cray Inc.

Weak Scaling on Broadwell sorting numLocales*100 M uint



© 2019 Cray Inc.

Future Work

- Put two-array sorting and distributed sorting on master
- Explore in-place one-pass parallel bucketizer as with ips4o [2]
- Support sample sort
  - when only comparison function is provided
  - for very data with skewed data distribution

Thanks to: Rupal Jain and Avneet Kaur (Rails Girls Summer of Code 2018)

References:

[1] Peter M McIlroy, Keith Bostic, and M Douglas McIlroy. 1993. Engineering radix sort. Computing systems 6, 1 (1993), 5–27.

[2] Michael Axtmann, Sascha Witt, Daniel Ferizovic, Peter Sanders. In-Place Parallel Super Scalar Sample Sort. arXiv:1705.02257v2 [cs.DC] 29 Jun 2017

© 2019 Cray Inc.

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.

© 2019 Cray Inc.

THANK YOU

QUESTIONS?

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

cray.com
@cray_inc
linkedin.com/company/cray-inc-