Julia vs. C

Berj Chilingirian (berjc@mit.edu)
Varun Mohan (vmohan@mit.edu)
William Qian (wqian94@mit.edu)

Goal

To understand why Julia may be slower than C in serial and parallel worlds.

Why?

As a developer, need to understand the fundamental benefits/costs of choosing a language.

But also...

Julia isn’t perfect ***yet***; maybe we can find something that can be improved.

Experimental Design

(1) Select code from Julia Base library.
(2) Translate directly to C and compare.
(3) Optimize both in the same way and compare.
(4) Parallelize and compare.

Selection Phase

MergeSort implementation in Base.Sort

Reasons:

(1) NOT embarrassingly parallel
(2) Can use SharedArrays (experimental feature)
(3) Everyone knows MergeSort

Base Comparison Phase

Translate Julia implementation directly to C.

For 10 trials with array of 2^{23} 32-bit elements

Julia	C
0.86s	0.80s

C is only 7.5% faster than Julia

Reasons for Performance Differences

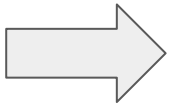
- Vectorization
  - Automatic vectorization in GCC O3 optimizations
  - Performs 16/32 byte move in one instruction
- Branching
  - cmov and bithacks in C vs. cmp and jmp instructions in Julia

Optimization Comparison Phase

Perform the same optimizations on both codes.

Use copy! instead of while loop...

```
i, j = 1, lo
while j <= half
    t[i] = v[j]
    i += 1
    j += 1
end
```



```
copy!(t, lo, v, lo, m - lo + 1)
```

Uses memmove (gets vectorized)

Optimization Comparison Phase

Perform the same optimizations on both codes.

For 10 trials with array of 2^{23} 32-bit elements

Julia	C
0.83s	0.80s

C is only 3.75% faster than Julia

Insight

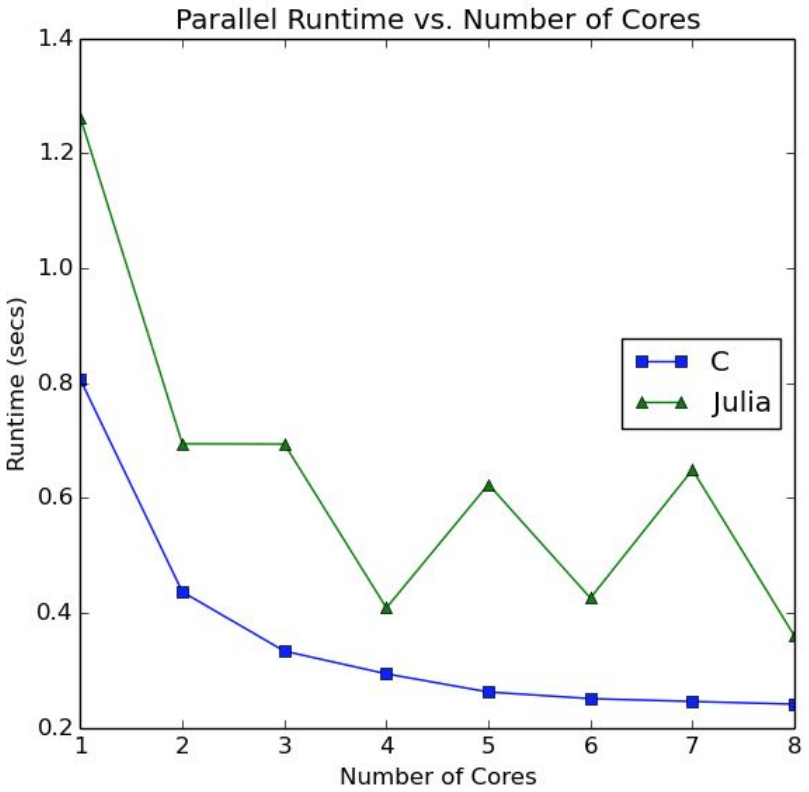
Julia can be even faster (with very little effort!)

Parallel Comparison Phase - Issues with Julia

- Using @everywhere not easy
- similar() with SharedArray doesn’t work
- Harder to parallelize Julia code compared to C code

Parallel Comparison Phase - Results

Parallelize C with Cilk and Julia with @spawnat and SharedArray.



Reasons for Performance Difference

- Work-Stealing
  - Julia workers performed no additional work after sorting provided subarray
- SharedArray is very expensive
  - Use mmap on data region

Conclusion

Julia is fast…
But parallelizing non-trivial code is challenging

Future Work

- Implement work-stealing in Julia
- Improvements to JIT compiler (HotSpot-like optimizations)
- Find places where Julia can be faster