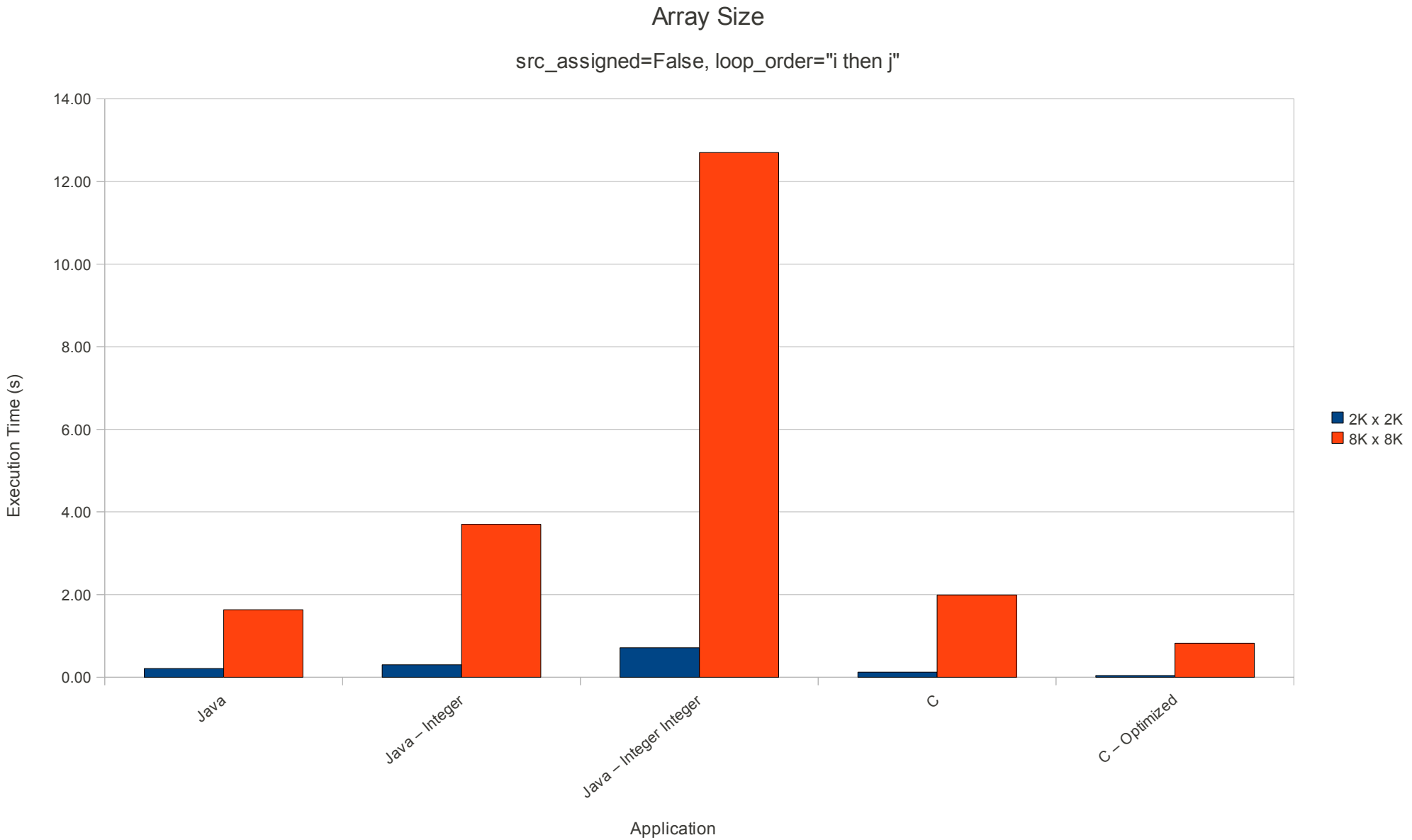
CSE 351
Section 1: HW 0 + Intro to C

Aaron Miller
David Cohen
March 31st, 2011

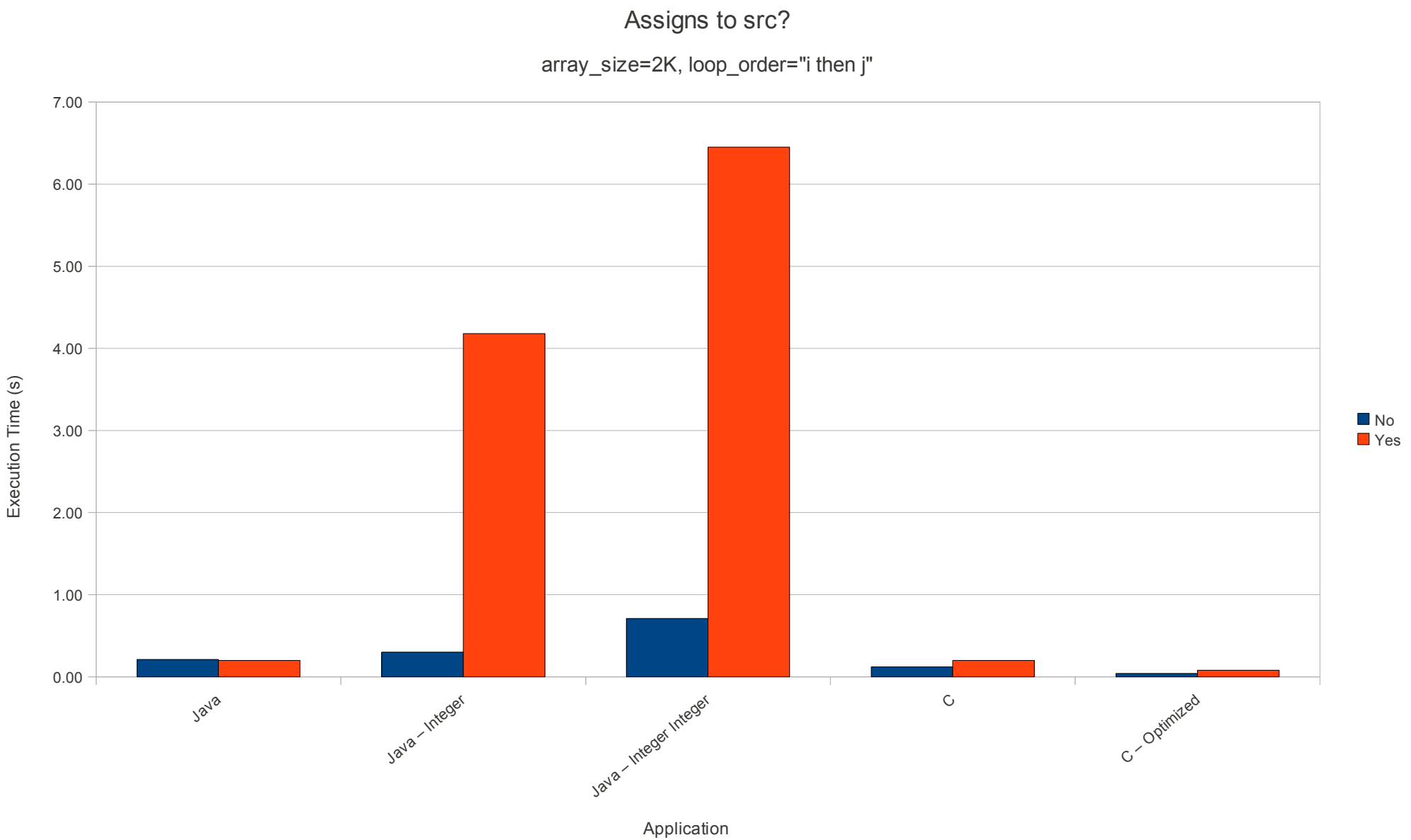
HW0

- Problems?
- Results Discussion
  - Array Size: 2K x 2K vs. 8K x 8K
  - Assigns to src?: Yes vs. No
  - Loop order: “i then j” vs. “j then i”
  - Language: Java vs. C
  - Compiler: C vs. Optimized C
- Surprises?
- Extra Credit

Our Results


Array Size
src_assigned=False, loop_order="i then j"
Execution Time (s)
14.00
12.00
10.00
8.00
6.00
4.00
2.00
0.00
Java
Java – Integer
Java – Integer Integer
C
C – Optimized
Application
2K x 2K
8K x 8K

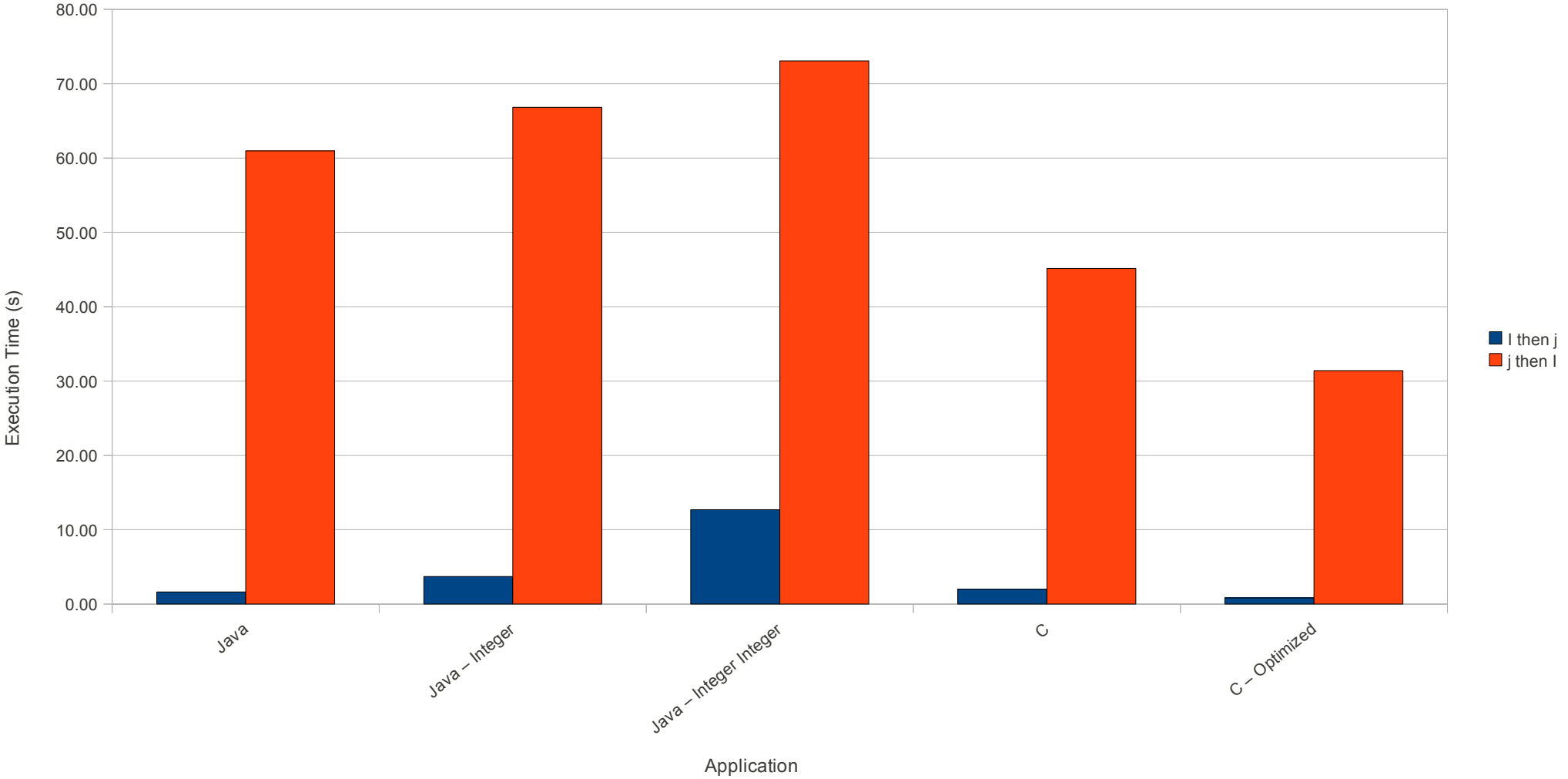
Our Results


Assigns to src?
array_size=2K, loop_order="i then j"
7.00
6.00
5.00
4.00
3.00
2.00
1.00
0.00
Execution Time (s)
Java
Java – Integer
Java – Integer Integer
C
C – Optimized
Application
No
Yes

Our Results

"i then j" vs. "j then i"

size-8K, src_assigned=False



Results - Summary

- Loop order matters!
  - Row-major storage of arrays in memory
- Multiplication isn't free
- Compilers help a lot
  - Static and dynamic optimization
- Java can be SLOW
  - Tip: use built-in System.arraycopy()
- Original claim: 21x speedup
  - Actual: 15x

Intro to C: Why C?

- Modern languages are still implemented in C
  - Java, Python, Perl, PHP, Ruby
- So are operating systems
- Convenient for understanding how a program executes on hardware
- Affords great performance and more control
  - “With great freedom comes great responsibility”
- 2nd most popular language today - TIOBE.com
  - You'll encounter it

Intro to C: Hello World

```
/* hello.c */
#include <stdio.h>

int main (int argc, char* argv[])
{
    printf("Hello, world!\n");

    return (0);
}
```

Intro to C: C vs. Java

- Classes / methods
- main()
  - Command-line arguments via String[] args
  - No return value
- Objects / References
- Simple import of library routines
- Arrays / Strings have length
- System.out.print/println()

- Functions
- main()
  - Count of command-line arguments and commands via “array” of “strings”
  - Has int return value
- Pointers
- Header files for importing library routines
- Strings: null-terminated sequence of chars
- printf()

Intro to C: Debugging

```
/* crash.c */
#include <stdio.h>


int main(int argc, char* argv[]) {
    int* blah = (int*)&main;
    printf("%x\n", blah);
    *blah = 0x3141592;
    return 0;
}
```

Intro to C: Debugging (con't)

- GDB (GNU DeBugger) is your friend
- Must compile w/ debugging symbols!
  - Use -ggdb switch:
    - $ gcc -o foo.exe -Wall -ggdb foo.c
- Important commands:
  - run
  - break <line# / symbol>
  - step
  - continue
  - info <locals / frame / register>
  - print, x
  - backtrace
  - help

Thanks!

Questions:
cse351-tas@cs.washington.edu