Strings += Performance

Ryan Dougherty

Quick Refresher

- Question: Are Strings in Java immutable?
- Answer: Yes!
- More on this later…

Background

- Junior in Computer Science + Math minor
- Honors Thesis
- Extension of previous thesis

Honors Thesis

- Each file: 448 * 304 cells, 4 bytes/cell
- 27 years * 52 weeks * 545 KB ~ 750 MB data
- 136k vertices → ~ 10^9 (billion) edges
- ~10^{12} (trillion) correlation computations
- Goal: make code run faster!

Original Code

```
byte[] byteArray = new byte[4];
try {
    File file = new File("binary_input_file");
    FileInputStream file_input = new FileInputStream(file);
    DataInputStream data_in = new DataInputStream(file_input);

    while (true) {
        try {
            for (int index = 0; index < 4; index++) {
                byteArray[index] = data_in.readByte();
            }
        } catch (EOFException eof) {
            break;
        }

        // transform byteArray to a float value
        float f = readFloatLittleEndian(byteArray);

        //... Do stuff with f...
    }
    data_in.close();
} catch (IOException e) {
    System.err.println(e.toString());
}
```

Execution Time?

- ~0.43 sec
- Not that fast, but a good baseline.

1 Code Change

- Floats easier than bytes
- Idea: write out all floats to file…How?
- Solution: Use a concatenated String!
- What was added in loop: s += f + “\n”; // s is a String, f is a float

Execution Time?

- ~72 sec
- ~170x slower!

Why?

- Question: 1 + 2 + 3 + … + n = ?
- = $n*(n+1)/2 = n^2/2 + n/2$
- $\in O(n^2)$

Why? (cont.)

- Reminder: s += f + “\n”; // s is a String, f is a float
- Reason 1: Immutable Strings!
- Reason 2: Compiler cannot guess!
- Each assignment creates a new String of 1 character, then 2, etc.
- O(n) loop → $O(n^2)$ loop

Solution…?

- java.lang.StringBuilder! (and its .append() method)
- S.B. is interesting, ask afterward…
- Execution time: ~0.52-0.55 sec → ~20-28% slower
- Much better than 17000% slower!

Summary

	Baseline	String concatenation	StringBuilder
Execution Time	0.43 s	~72 s	0.52 s
Memory Usage (normalized)	0 MB	+155.68 MB	+3.86 MB

Take-home Messages

- Your code does not live in a vacuum!
- View the Java docs! (for fun…?)
- Analyze your code!
- Use profilers!

Suggestions

- View (compiled) object code: “javap -c MyClass”
- Use S.B. (not String) for “unknowable” data values.

Questions?


WHENEVER I LEARN A NEW SKILL I CONCOCT ELABORATE FANTASY SCENARIOS WHERE IT LETS ME SAVE THE DAY.
OH NO! THE KILLER MUST HAVE FOLLOWED HER ON VACATION!
BUT TO FIND THEM WE'D HAVE TO SEARCH THROUGH 200 MB OF EMAILS LOOKING FOR SOMETHING FORMATTED LIKE AN ADDRESS!
IT'S HOPELESS!
EVERYBODY STAND BACK.
I KNOW REGULAR EXPRESSIONS.
PERL!
TAP TAP