CSSE 221 Day 20

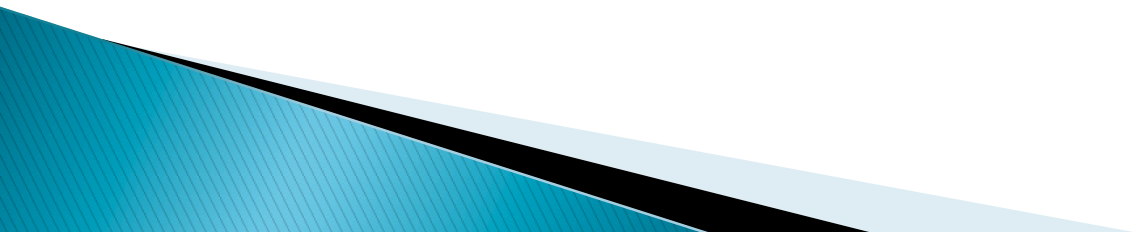
Function Objects and the Comparator Interface
Merge Sort
Fork/Join Framework

Checkout *ForkJoinIntro* project from SVN

Questions

Today's Plan

- Merge sort overview
- Introduction to function objects, **Comparator**
- Parallelism with the Fork/Join Framework



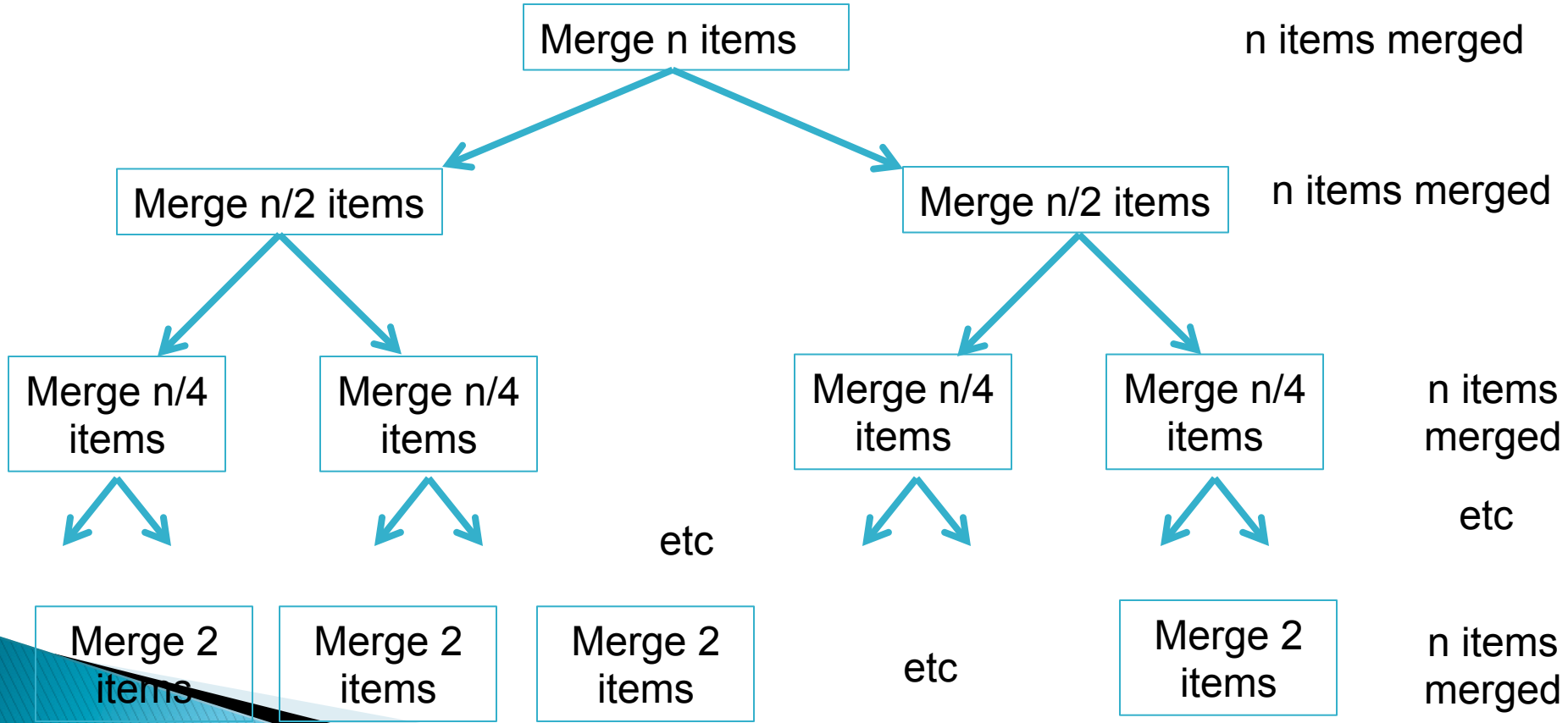
Merge Sort Overview

- Basic recursive idea:
  - If list is length 0 or 1, then it's already sorted
  - Otherwise:
    - Divide list into two halves
    - Recursively sort the two halves
    - **Merge** the sorted halves back together

Analyzing Merge Sort

If list is length 0 or 1,
then it's already sorted

- Otherwise:
  - Divide list into two halves
  - Recursively sort the two halves
  - **Merge** the sorted halves back together

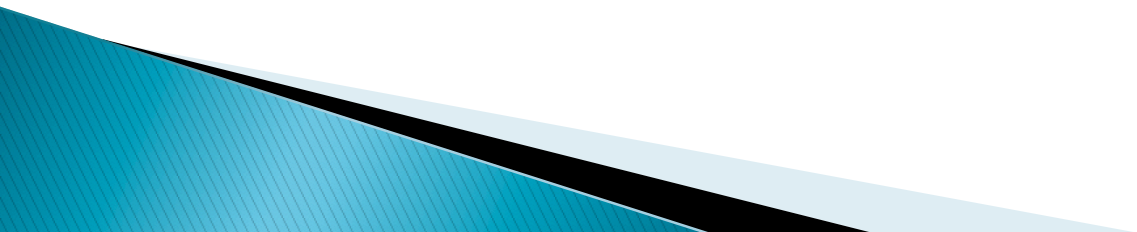


Function Objects

- Another way of creating reusable code

A Sort of a Different Order

- Java libraries provide efficient sorting algorithms
  - `Arrays.sort(…)` and `Collections.sort(…)`
- But suppose we want to sort by something other than the "natural order" given by `compareTo()`
- *Function objects* to the rescue!



Function Objects

- Objects defined to just “wrap up” functions so we can pass them to other (library) code
- For sorting we can create a function object that implements `Comparator`
- `Let's try it!`

Intro. to Fork-Join Parallelism

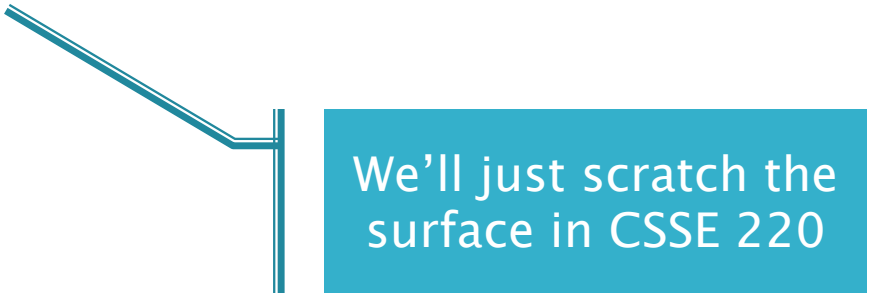
- Function objects and recursion meet multicore computers

Some slides and examples derived from Dan Grossman's materials at http://www.cs.washington.edu/homes/djg/teachingMaterials/

Changing a Major Assumption

- *Sequential programming*: one thing happens at a time
  - No longer the case!
- *Parallel programming*: multiple things happen simultaneously

- Major challenges and opportunities
  - Programming
  - Algorithms
  - Data

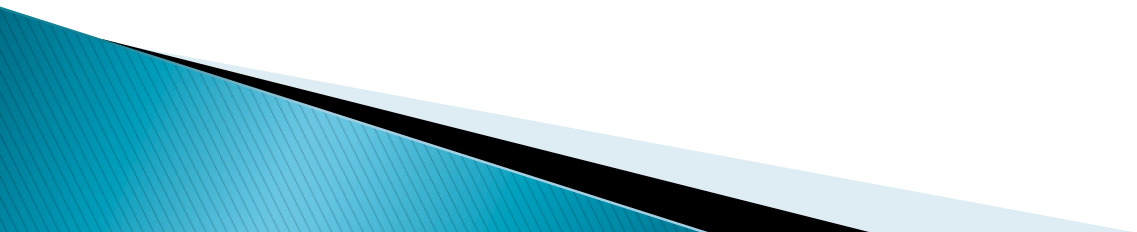


Simplified View of History

- Parallel code is often much harder to write than sequential
- Free ride from the CPEs
  - From 1980-2005 performance of same sequential code doubled every two years
- No one knows how to continue this!
  - Speed up clock rate?
    - Two much heat
    - Memory can't keep up
  - But the "wires" keep getting smaller, so…
    - Put multiple processors on same chip!

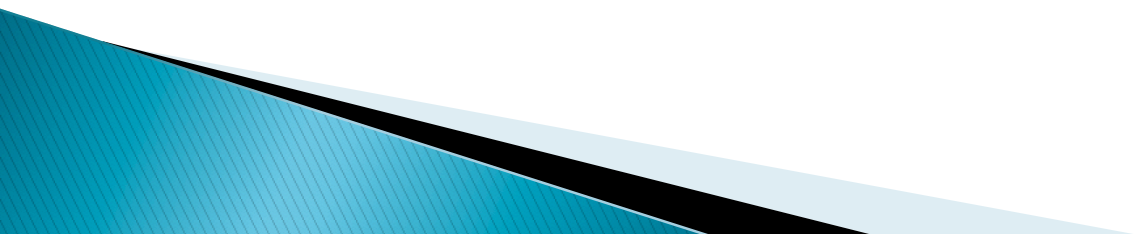
What do we do with all of them?

- Run multiple totally different programs
  - Operating system handles this
  - Uses *time-slicing* plus multiple cores
- Multiple things at once in one program
  - We'll play with this today!



Parallelism vs. Concurrency

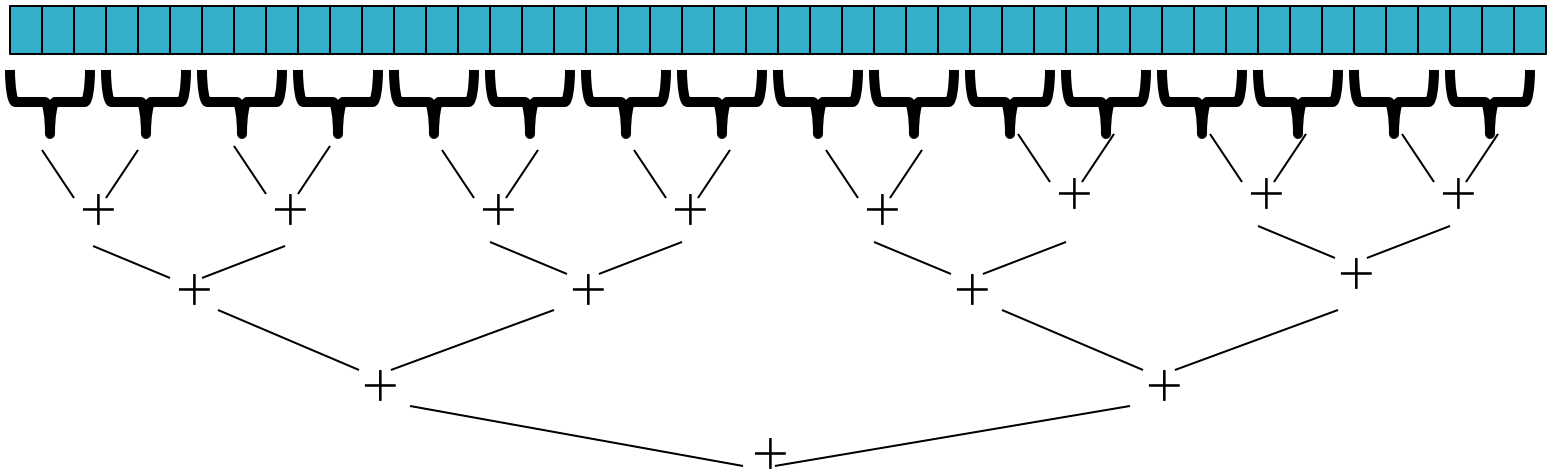
- *Parallelism*: Use more resources for a faster answer
- *Concurrency*: Correctly and efficiently allow simultaneous access to data



An analogy

- CS1 idea: Writing a program is like writing a recipe for a cook
- Parallelism: slicing lots of potatoes
- Concurrency: sharing stove burners

Parallelism Idea



- Example: Sum elements of a large array
- Use divide-and-conquer!
  - Parallelism for the recursive calls

Fork-Join Framework

- Specifically for recursive, divide-and-conquer parallelism
  - Is in Java 7 standard libraries, but available in Java 6 as a downloaded `.jar` file

- *Fork*: splitting off some code that can run in parallel with the original code
  - Like handing a potato to a helper

- *Join*: waiting for some forked code to finish
  - Like waiting for the potato slices from the helper

Getting good results in practice

- Set a *sequential threshold*
  - A size below which we just "slice 'em ourselves"
- Library needs to "warm up"
  - Java Virtual Machine optimizes as it runs
- Wait until your computer has more processors ☺

- Here there be dragons!
  - Memory-hierarchy issues
  - Race conditions
  - We're ignoring lots of gory details!

Fork-Join Lab

- Find a partner for ForkJoin Lab
- You'll:
  - Write some code
  - Run some experiments
  - Write a lab report

- This is an exciting lab:
  - Enjoy playing with the tools and ideas
  - Ask questions!

Follow the written lab instructions carefully. There's a lot more independent learning here than we've been doing so far.