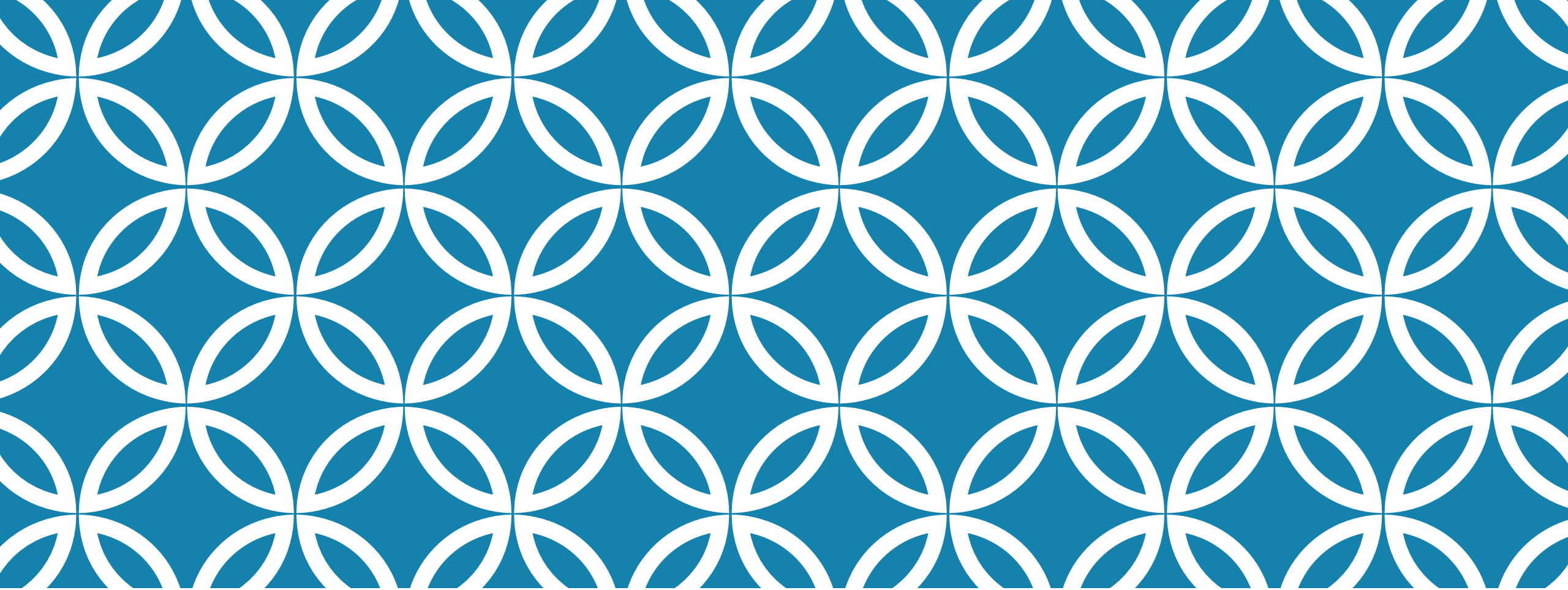
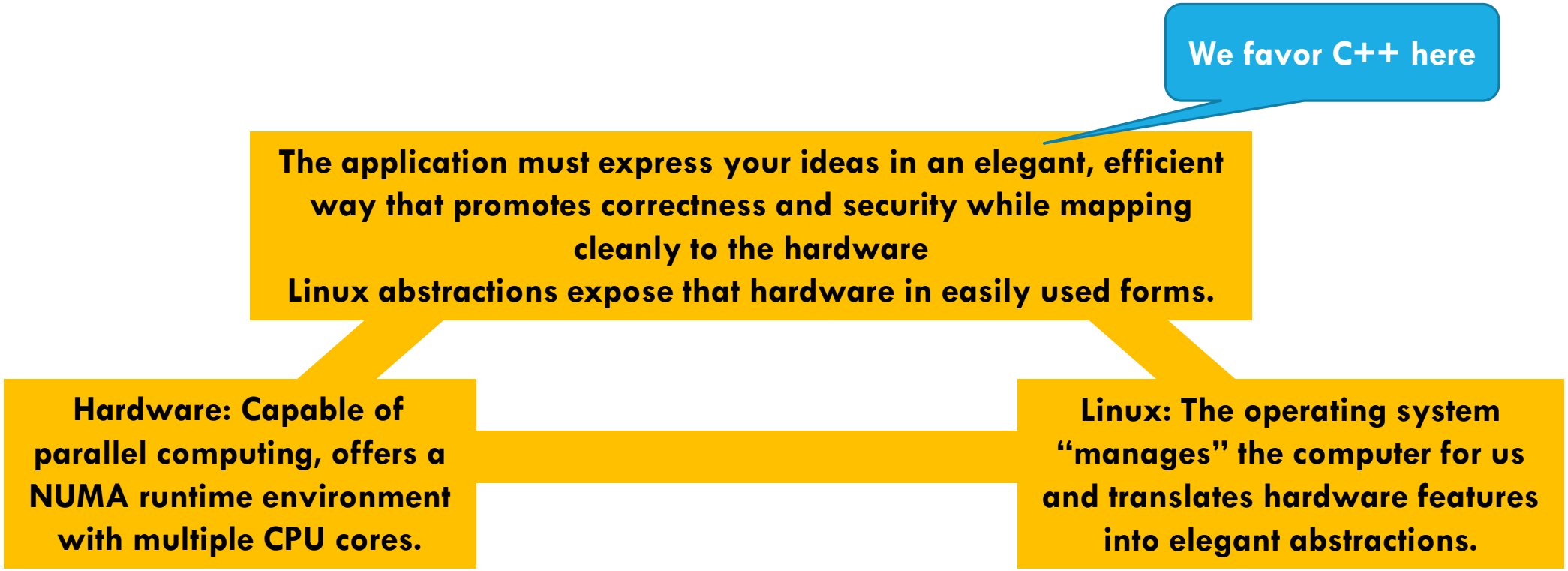


WELCOME TO CS4414 SYSTEMS PROGRAMMING

Professor Ken Birman

Lecture 1

"IDEA MAP" FOR THE WHOLE SEMESTER



WHEN YOU WRITE A PROGRAM, SHOULD YOU CARE HOW IT GETS EXECUTED?

Most people are familiar with Java and Python

Java has lots of data types (and lots of fancy syntax!), generics, other elaborate language features and compiles to a mix of machine code and programming language runtime logic.

Python is easier: No need to fuss with data types, easy to create arrays and transform all the objects with just one step.

WHEN YOU WRITE A PROGRAM, SHOULD YOU CARE HOW IT GETS EXECUTED?

Which is better?

1) Java
2) Python

... why?

Python is easier: No need to fuss with data types, easy to create arrays and transform all the objects with just one step.

CONSIDERATIONS PEOPLE OFTEN CITE

Expressivity and Efficiency: Can I code my solution elegantly and easily? Will my solution perform well?

Correctness: If I end up with buggy code, I'll waste time (and my boss won't be happy). A language should facilitate correctness.

Productivity: A language is just a tool. The easier it is to do the job (which is to solve some concrete problem), the better!

A SUBTLE CONSIDERATION: MODULARITY AND COMPOSITIONALITY

Don't fix things that already work. Ideally, we want the system to provide lots of pre-packaged solutions for common tasks.

As a systems person, I'm very focused on this idea of pre-packaged modular solutions.

Modern machine learning forces us to think in these terms!

MICROSOFT FARMBEATS EXAMPLE

How many programs are in use here?

… hundreds!

A modern computing applications is a *software ecosystem*

… THIS IS THE COMPLICATION

As we deal with larger and larger scale, the "modules" won't be simple things like a library that deals with managing a sorted list

We may need to "compose" entire programs or even systems, which will need to share files or perhaps "objects".

… the programming language is just a part of this ecology

DRILL-DOWN CONSIDERATIONS

We want our solutions to perform well and "scale well".

For many tasks this involves working on the "cloud" (big remote data centers, like AWS or Microsoft Azure or Google).

In the cloud you rent the machines you need, as needed, but pay for what you use. So performance ≅ $$$.

DRILL-DOWN CONSIDERATIONS

Which *performs* better?

1) Java
2) Python
3) … something else?

… why?

REASONS WE CARE ABOUT PERFORMANCE

Modern forms of computing are very power-hungry! And this is causing growing impact on the global "electricity footprint" associated with popular ways of solving problems.

Future of civilization might depend on whether your code can minimize the amount of electricity it consumes!

Roughly 1% of global electric use, doubling roughly every 2 years!

https://venturebeat.com July 15, 2020

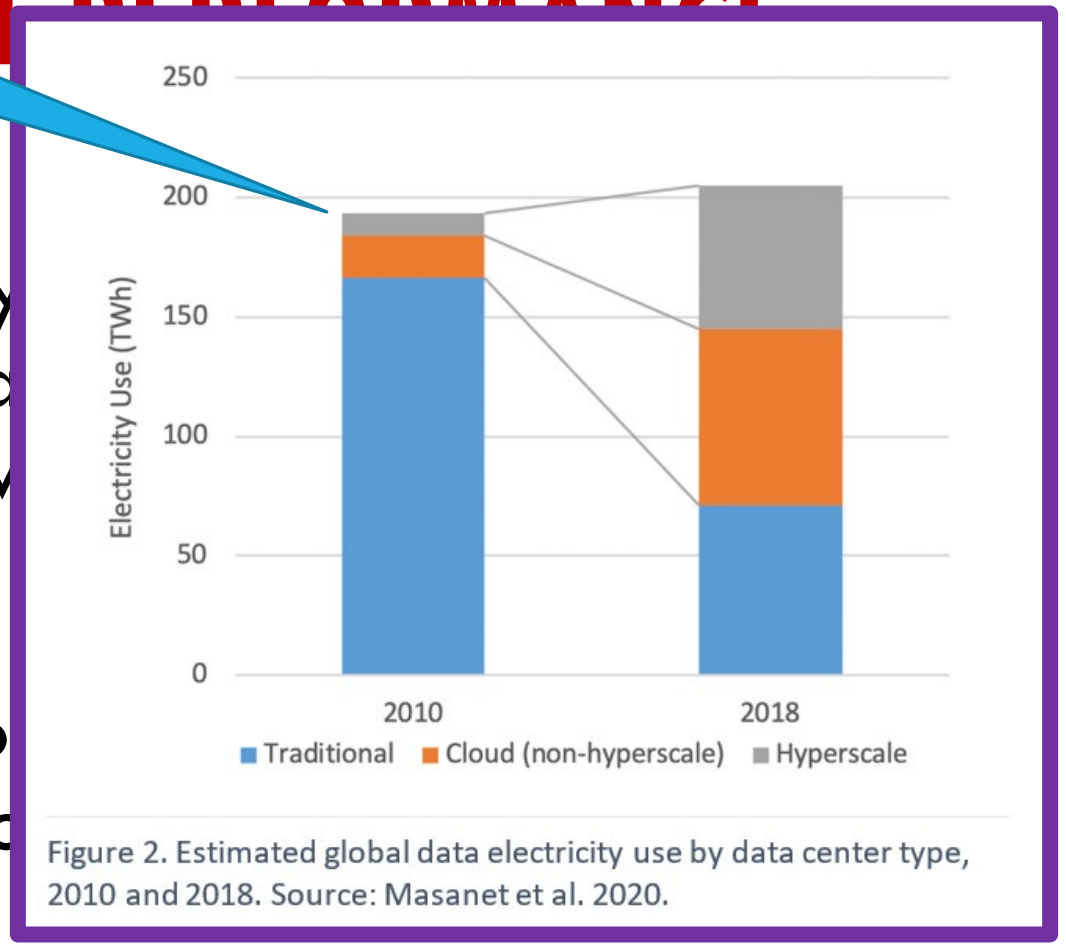
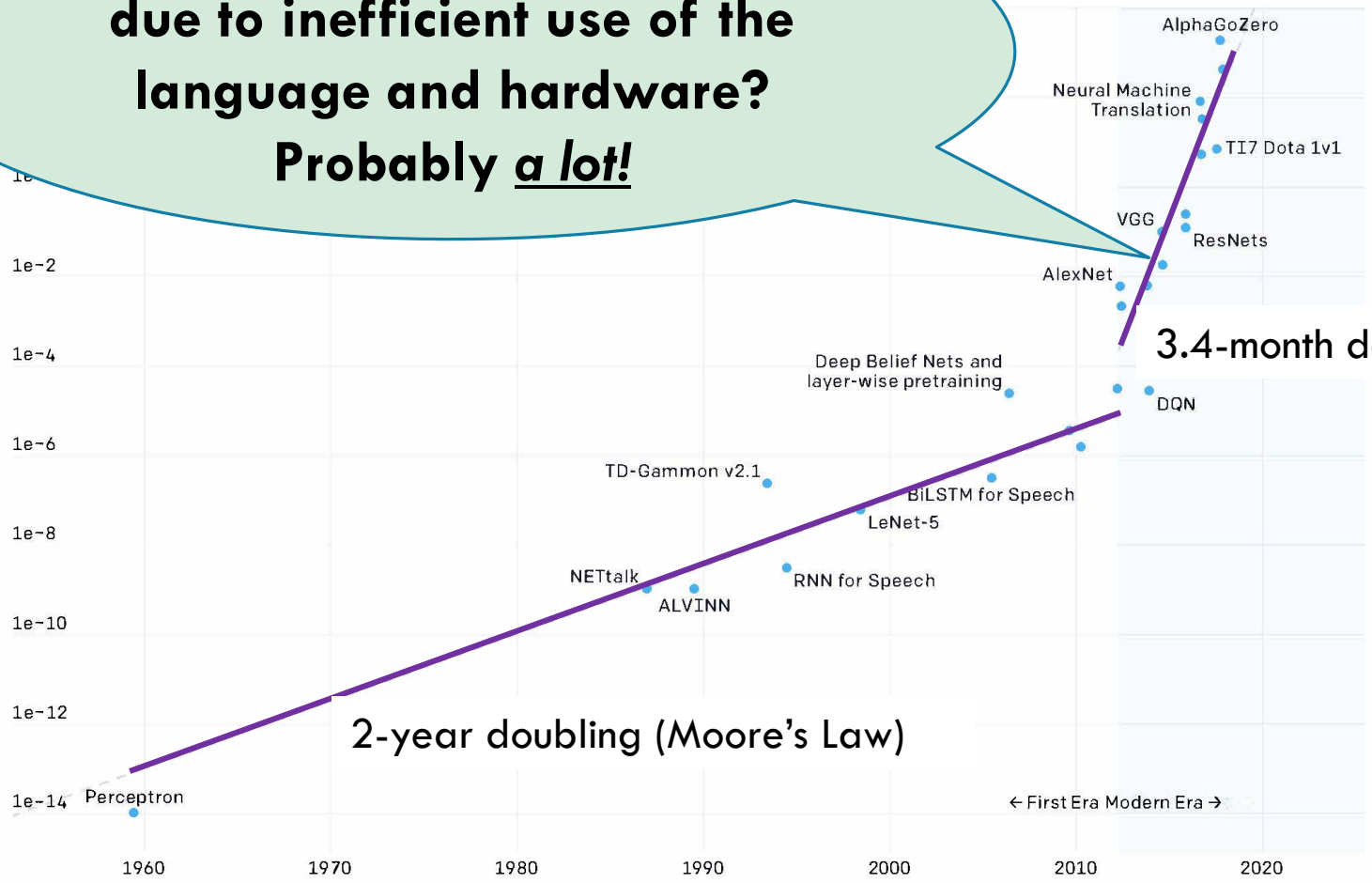


Figure 2. Estimated global data electricity use by data center type, 2010 and 2018. Source: Masanet et al. 2020.

https://energyinnovation.org/2020/03/17/how-much-energy-do-data-centers-really-use/

COMPUTE TIME TO TRAIN ML MODELS

How much of this is really due to inefficient use of the language and hardware? Probably *a lot!*



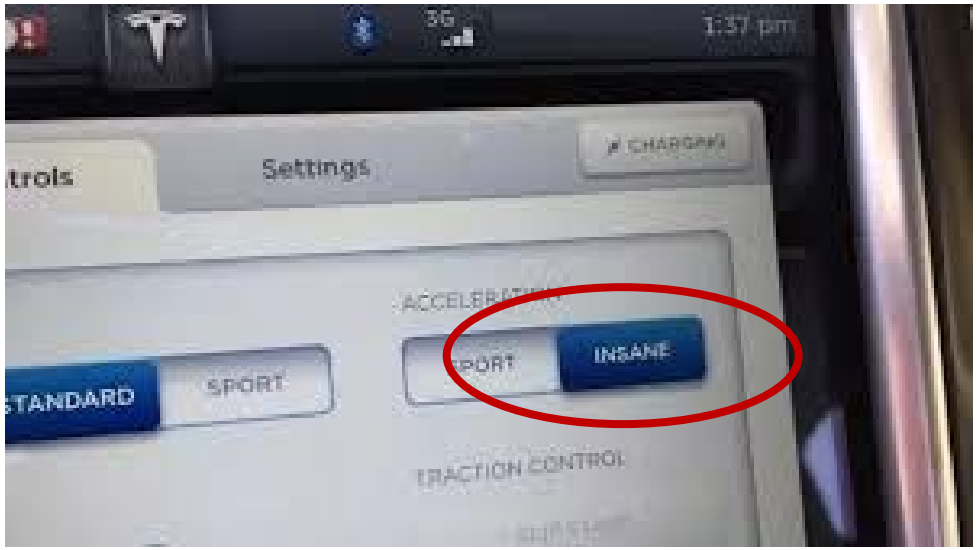
3.4-month doubling

2-year doubling (Moore's Law)

SOME CARS HAVE INSANE SPEED BUTTONS…

Guess what? So do computers!

In CS4414 we'll push the button.



(in ways that are correct, secure, natural, elegant)

SMART USE OF THE "PLATFORM" IS HOW!

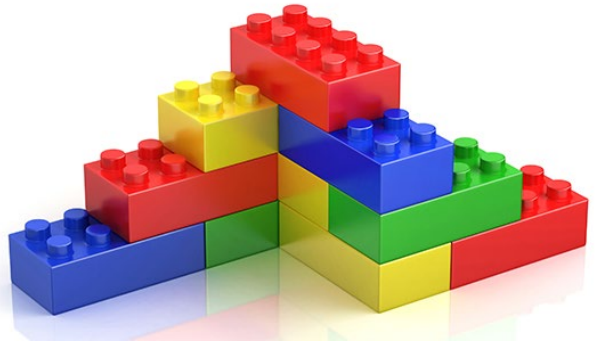
In CS4414 we will be learning about the Linux operating system. Linux is universal these days.

We will use C++ as our programming language.

And we'll learn to write code in smart ways that use the hardware and software "ideally" to get the best possible speed.

WHY LINUX? DOES THE O/S EVEN MATTER?

When building "interesting" applications we often put a few building blocks together, Lego style.



Linux is full of small, easily used building blocks for common tasks, and has easy ways to connect things to make a bigger application from little pieces.

Productivity rises because you often don't need to build new code – you can just use these existing standard programs in flexible ways.

LINUX AND THE HARDWARE: TWO SIDES OF THE SYSTEM ARCHITECTURE

We will be learning about the modern computer hardware, not so much from an internals perspective, but as users.

Linux lets you design applications that correspond closely to the hardware. But then we need a programming language that lets us talk directly to the operating system and the hardware.

WHY ARE PYTHON AND JAVA EXPENSIVE?

Python: Interpreted

Compiles to a high-level representation that enables an "interpretive" execution model.

In fact, Python is like a "general machine" controlled by your code: Python itself runs on the hardware. Then your code runs on Python!

Gradual typing: Python is very laissez-faire and can't optimize for specific data types.

Java: Runtime overheads

Compiles (twice: to byte code, then via JIT) but rarely exploits full power of hardware. Limited optimizations, parallelism

Dynamic types and polymorphism are costly.

Everything is an object, causing huge need for copying and garbage collection.

It feels as if your programs run inside layers and layers of "black boxes"

HOW DOES C++ AVOID THESE PITFALLS?

C++ objects are a compile-time feature. At runtime, all the type-related work is finished: no runtime dynamics.

The compiler "inline expands" and optimizes heavily. You help it.

Computers execute billions of instructions per second, yet we can write code that will minimize the instructions and shape the choices.

Parallelism is easy, and the compiler automatically leverages modern hardware features to ensure that you will have highly efficient code.

LET'S DRILL DOWN ON SPEED

For some situations, C++ can be *thousands of times faster* than Python or Java, on a single machine!

- Typically these are cases where the application has a lot of *parallelism* that the program needs to exploit.
- For example, identifying animals in a photo entails a lot of steps that involve pixel-by-pixel analysis of the image
- But in fact we can get substantial speedups just scanning large numbers of big files… hence our word-count demo

LET'S DRILL DOWN ON SPEED

We said that Python is slowest, Java is pretty good, but C++ can beat both. C++ knocks the socks off Java for parallel tasks.

What would be a good way to "see that in action"?

A small example: "word count" in Python, Java and C++

WORD COUNT TASK

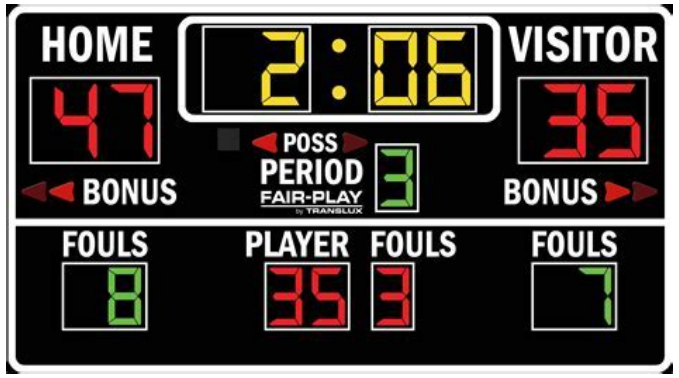
Basically, we take our input files and "parse" them into words. All three languages have prebuilt library methods for this. Discard non-words (things like punctuation marks).

Keep a sorted list of words. As we see a word, we look it up and increment a count for that word (adding it if needed).

At the end, print out a nicely formatted table of the words/counts in descending order by count, alphabetic order for ties

PAUSE HERE FOR A LITTLE DEMO

THE SCOREBOARD



#1-A: Ken's C++ Faster, but more complex…

real 4.645s
user 14.779s
sys 1.983s

#1-B (Sagar's code, shorter & better use of C++…)

real 0m8.200s
user 0m49.295s
sys 0m2.145s

#2 Lucy's Python version

real 1m30.857s
user 1m30.276s
sys 0.572s

This was only 19 lines of code!

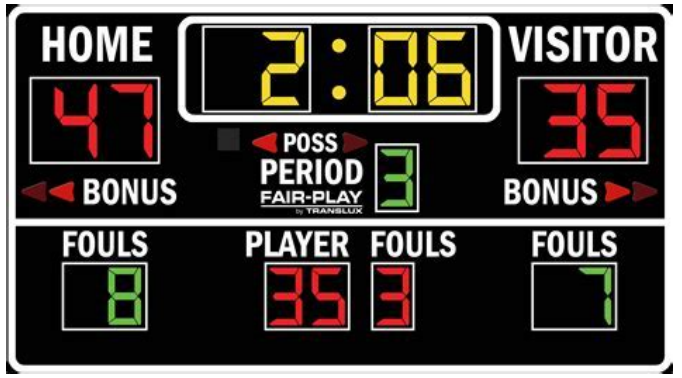
#3 Lucy's Java version (no threads)

real 1m49.373s
user 3m16.950s
sys 8.742s

#4: Pure Linux (buggy sort order)

real 2m38.965s
user 2m43.999s
sys 27.084s

THE SCORE



C++ version was 34x faster than Linux, 20x faster than Java or Python

#1-A: Ken's C++ Faster, but more complex…

real 4.645s
user 14.779s
sys 1.983s

#1-B (Sagar's code, shorter & better use of C++…)

real 0m8.200s
user 0m49.295s
sys 0m2.145s

#2 Lucy's Python version

real 1m30.857s
user 1m30.276s
sys 0.572s

This was only 19 lines of code!

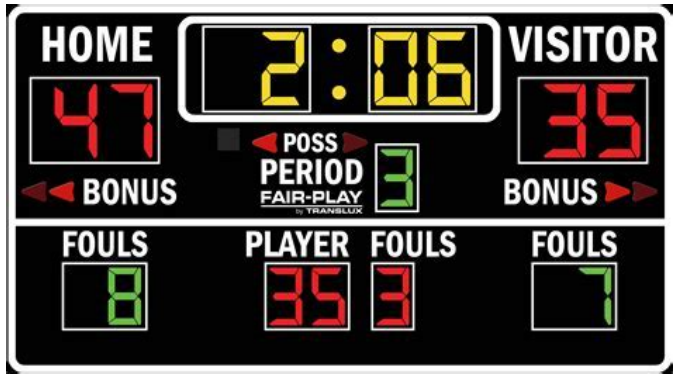
#3 Lucy's Java version (no threads)

real 1m49.373s
user 3m16.950s
sys 8.742s

#4: Pure Linux (buggy sort order)

real 2m38.965s
user 2m43.999s
sys 27.084s

THE SCORE



Notice that the user time is 3x larger than the real time. Puzzle: how can this be true?

#1-A: Ken's C++ Faster, but more complex…

real 4.645s
user 14.779s
sys 1.983s

#1-B (Sagar's code, shorter & better use of C++…)

real 0m8.200s
user 0m49.295s
sys 0m2.145s

#2 Lucy's Python version

real 1m30.857s
user 1m30.276s
sys 0.572s

This was only 19 lines of code!

#3 Lucy's Java version (no threads)

real 1m49.373s
user 3m16.950s
sys 8.742s

#4: Pure Linux (buggy sort order)

real 2m38.965s
user 2m43.999s
sys 27.084s

HOW CAN A PROGRAM DO 14.7779s OF COMPUTING IN 4.645s?

A 3-horsepower system

Could this just be a measurement mistake in Linux?

… in fact, if a process is using more than one thread it can harness more than one CPU at the same time.

With 3 CPUs running continuously at full speed, it can do 3x more work than the elapsed wall-clock time!

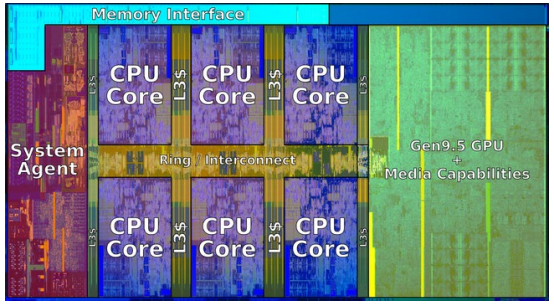
DIDN'T I PROMISE "THOUSANDS X" HOW DID THAT DROP TO 30X?

We counted words in text files: Limited parallelism.

The C++ program is able to process them in parallel side-by-side streams, which was how we got the speedup.

With image processing or machine learning (tensor arithmetic), the value of parallel processing is dramatically larger.

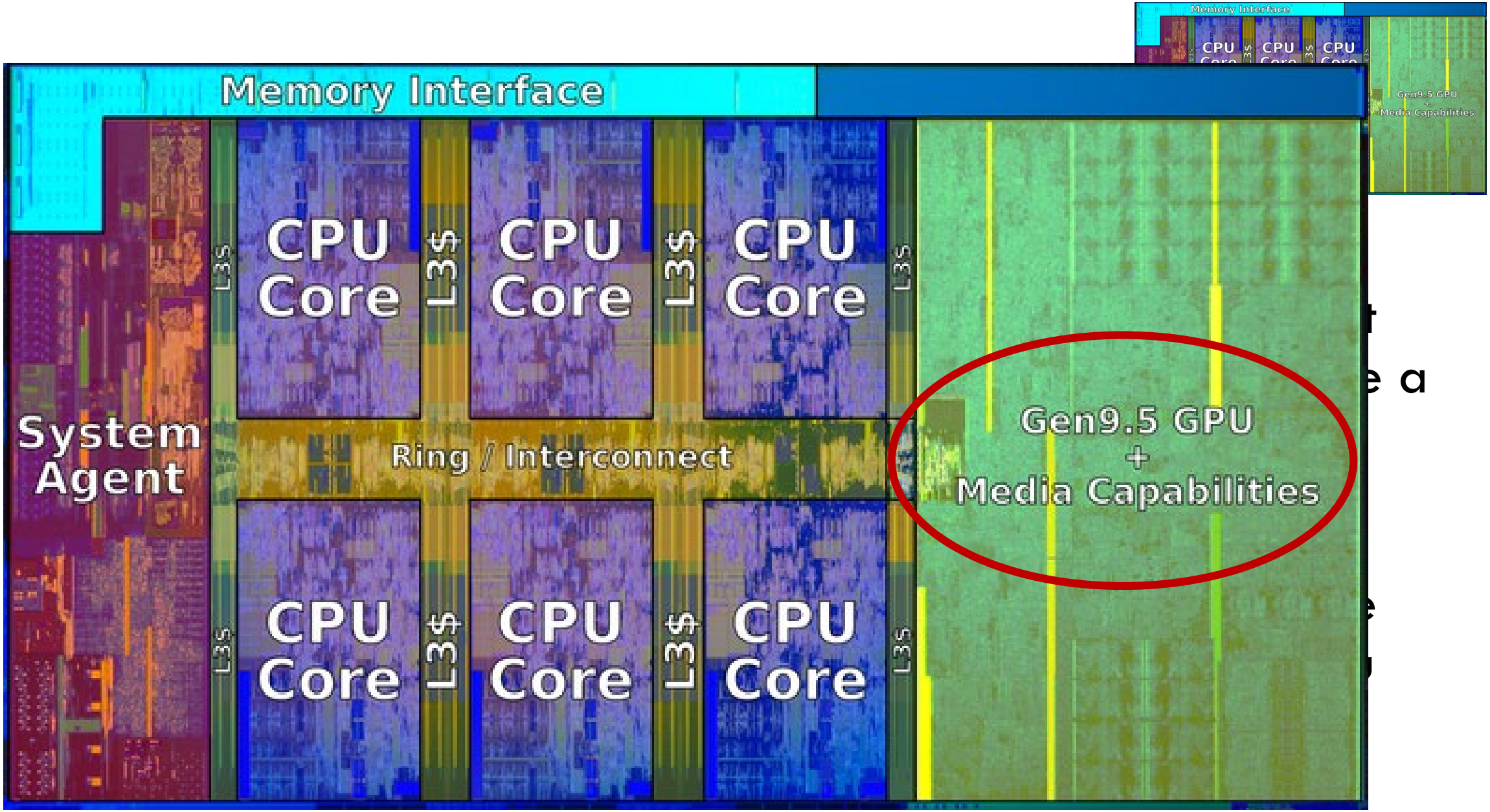
PARALLELISM IN YOUR COMPUTER



6-core Intel chip with GPU

A modern computer has multiple smaller computers (cores) that all run on the same computer memory (RAM). It may also have a special form of "accelerator" called a GPU.

To leverage this power, the computer offers special hardware instructions. The compiler can use those for big speedups. You can also use "threads": a method of having more than one computational activity running within a single program.


Memory Interface
CPU Core
CPU Core
CPU Core
L3$
L3$
L3$
L3$
System Agent
Ring / Interconnect
CPU Core
CPU Core
CPU Core
L3$
L3$
L3$
L3$
Gen9.5 GPU
+
Media Capabilities

TOPIC FROM BEYOND CS4414: CLOUD COMPUTING TAKES THIS FURTHER

For compute-intensive tasks, companies set up an account on a big commercial data center called a cloud. "Rent, don't own".

You can easily run a program on hundreds of thousands of computers, each instance processing different input files.

- So a single job might have millions of threads working in parallel, and each using parallel computing instructions!
- … the hard part is when they need to combine their results.

MODERN CLOUD COMPUTING DATA CENTER

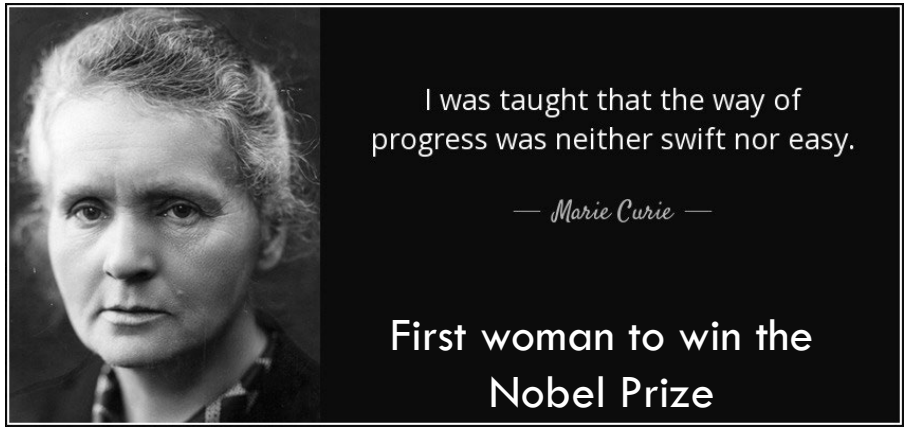
ISSUES INTRODUCED BY LARGE SCALE

Program design: When we use networking to let processes talk to one-another, they need ways to share data, cooperate, coordinate, recover from failures.

Algorithmic: Just like in a single machine, there are more efficient and less efficient data structures and "protocols"

Security: At scale it is common to run into new kinds of attacks, and we need different styles of defense. Sensitive data is an especially important concern (like private information).

AND YET…



Many jobs today involve larger scale computing platforms, even if you don't work "for" a cloud company.

A factor of 10x can be dramatic if the code is used by Google to respond to a search or is part of the Facebook image feed.

C++ is the "way of progress" for demanding tasks.

LEARNING LINUX AND C++

You came into this class comfortable in an object oriented language, and learned data structures, so C++ should be easy to learn.

- The operators and syntax and features will remind you of Java
- There are *extra* operators, and those we will teach you, and things like dynamic memory management, but we won't teach basics: those you need to learn in a hands-on way!
- Similarly, we will provide pointers to lists of Linux commands and bash syntax, but these are topics for experiential learning

EXAMPLE: HELLO WORLD IN C++

C++ "Hello World!" Program

```cpp
// Your First C++ Program

#include <iostream>

int main() {
    std::cout << "Hello World!";
    return 0;
}
```

Output

```
Hello World!
```

EXAMPLE: WORD COUNT IN C++

This is the “core” of the counting logic:

```
using          WC = std::map<std::string, int>;
WC              sub_count[MAXTHREADS];

inline void found(int& tn, char*& word)
{
      sub_count[tn][std::string(word)]++;
}
```

EXAMPLE: WORD COUNT IN C++

… and here is the core of the sorting logic:

```
struct SortOrder: public std::binary_function<std::pair<int, std::string>, std::pair<int, std::string>, bool>
{
    bool operator()(const std::pair<int, std::string>& lhs, const std::pair<int, std::string>& rhs) const
    {
        return lhs.first > rhs.first || (lhs.first == rhs.first && lhs.second < rhs.second);
    }
};

using SO = std::map<std::pair<int, std::string>, int, SortOrder>;
SO sorted_totals;
for(auto wc: totals)
{
        std::pair<int,std::string> new_pair(wc.second, wc.first);
        sorted_totals[new_pair] = wc.second;
}
```

FRIGHTENING SYNTAX!

But the course staff is used to C++ and we can help.

Once you really learn it, you'll start to find it much more natural.

The one big thing to know is that even a single misplaced character can trigger pages of compiler complaints, so you do need to understand exactly what you are "asking it to do!"

THE WAY OF PROGRESS BEING "NEITHER EASY NOR FAST" ASPECT?

C++ will eventually start to feel much easier, after a few weeks.

But you don't *automatically* get high speed. That takes more work – rewarding work, because you can measure the improvements as you make them, but it does take effort!

The key is to be able match your coding goals to the hardware.

WHEN I FIRST CODED MY SOLUTION, MY PROGRAM WAS VERY SHORT, BUT RATHER SLOW.

I added parallel threads – which complicated the solution but helped a lot. Then because file opening was slow, I added a thread to "preopen" files before they were needed.

The C++ library for file opening and reading files was a bottleneck, so I switched to calling Linux file open and Linux file read, directly. This gave an additional speedup

FINAL VERSION

With threads, my code was "part way" to the goal.

Thinking about bottlenecks, I decided to add one more parallel computing idea (we'll see it soon). Then I changed some of my very heavily-used methods to be "inlined"

This gave that 20x-30x compared to Python and Java.

ROLL YOUR OWN? OR LEARN SOME WEIRD LIBRARY INTERFACE?

In C++ we could certainly code everything from scratch.

But library solutions have the benefit of being standard, widely used, and heavily tested.

C++ libraries are also exceptionally performant, and for our course this is an important consideration!

REMAINDER OF WORD COUNT IN C++

I didn't show you:

- The standard includes required to import the std:: libraries
- The main program that calls the "getWordCount" method
- A bunch of little helper methods I wrote.

WHY WOULD PEOPLE CARE ABOUT WORD COUNTING IN A MODERN SYSTEM, LIKE FOR NLP?

Natural language programs are *based on word counts*, but :

- White space, punctuation, hyphenation is removed
- Conjunctions such as “a”, “and”, “or” are discarded
- Upper case is mapped to lower-case
- Stems are removed: “**fly**ing” might map to “fly”.

WHY DO NLP SYSTEMS CARE ABOUT STEMS?

When people do a web search such as “learn to fly small plane” or “birds that cannot fly” small style differences arise.

Stems have been shown to be much more stable and consistent

Web pages for flying schools would probably use “fly” very extensively. In contrast “that” and “to” are less relevant…

WHAT SHAPES THE PERFORMANCE OF "STEMMED, LOWER-CASE WORD COUNT"?

- Speed of reading the data, especially if the file is large. At Google, some AI systems that learn from web pages scan hundreds of billions of them.
- Splitting, stemming, mapping to lower case, discarding conjunctions
- If we use new variables to hold stemmed words, we'll need to "allocate" memory and initialize the object.
- We'll need to keep a sorted list of words, and look for each word as we discover it, and increase the associated counter.

GLIMPSE OF LINUX AND BASH

This bash command runs c++, telling it to optimize the code, warn in a "fussy" way.

```
% g++ -std=c++11  -O3 -Wall -Wpedantic -pthread  -o fast-wc fast-wc.cpp
```

This does a timed run of the program (fast-wc):

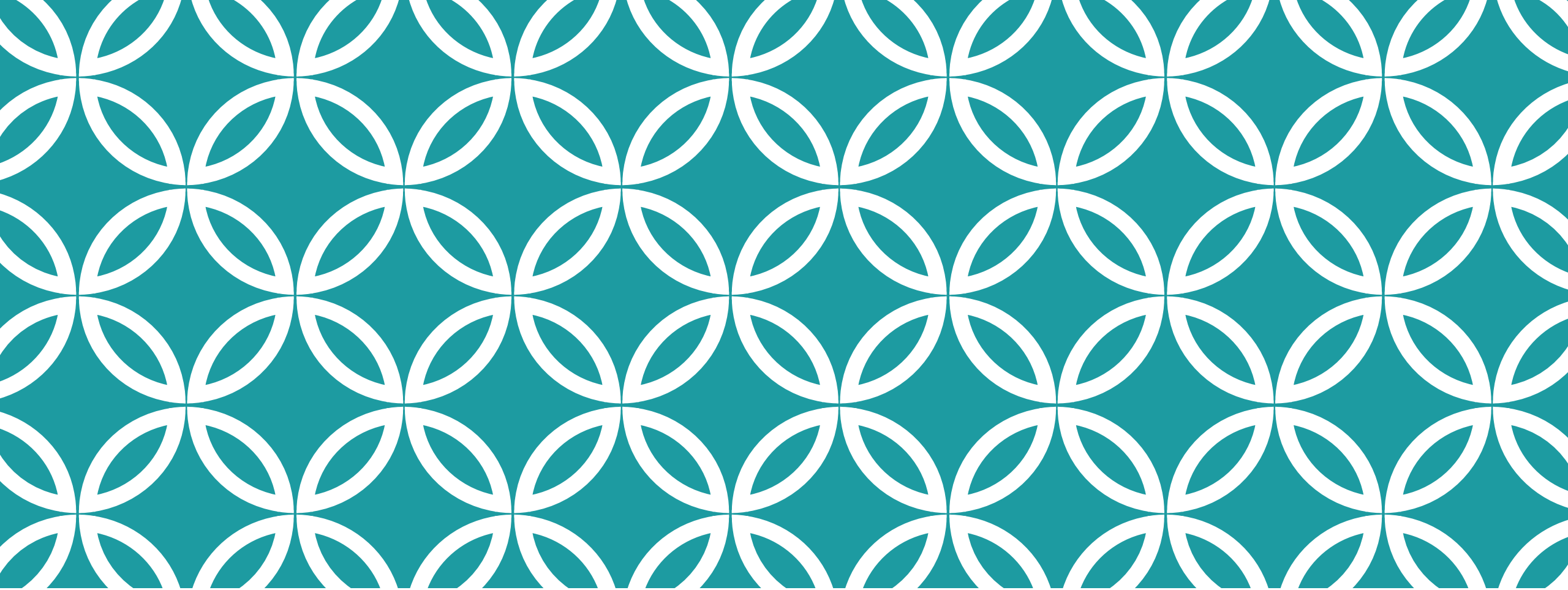
```
% time taskset 0xFF ./fast-wc -n4 -p
```

```
fast-wc with 4 cores, 50095 files, 16 blocks per read, parallel merge ON
                define   |  2008083
                struct   |  1694853
                     0   |  1268529
                    if   |  1172461
```

MORE FUN WITH BASH

In fact we can use bash to…

- Automatically open a file and have it look like "console input" in our C++ program.
- Put the program output into a file
- Run a program in the background
- Put the program output into a bash variable, and then pass that variable as a command-line argument to some other program
- … the list goes on for quite a while!



PRACTICAL CONSIDERATIONS

Organizational stuff

IS C++ HARD TO LEARN?

C++ is easy to learn at a basic level (in CS4414 we won't dive into the really fancy, obscure, C++ features).

C++ was designed to make the mapping to the hardware and to the machine instructions "evident". An interview with the inventor of C++.

Unlike that sense of black-boxes everywhere, with C++ you can know everything your code is doing, even to very low levels.

DEVELOPER TOOLS (IDE)

Ken and Sagar do many things at the "command line" layer of Linux. Neither uses any kind of IDE for C++ on Linux.

But there are good IDE options, like Eclipse and Visual Studio Code

Linux comes in many flavors. We prefer Ubuntu, the bash shell, and the "cmake" application builder. Sagar loves the emacs editor.

RESOURCES

The whole course closely tracks the main textbook by Bryant and O'Halloran at CMU.

- Main difference is that CMU CS213 views the class as an introduction to programming languages and compilers
- CS4414 has a systems-centric focus. But in fact the textbook includes this material, even if CMU's course brushes over it

To fill in gaps, we also recommend getting a C++ textbook and learning to access the online Linux materials

SETUP

No matter what you use, Linux requires some (annoying) setup, and you'll need to use online resources to get these right.

Otherwise things like console colors or the editor colors might be difficult to read. Linux still works, but could be hard to use. c++ might get confused and not find things like the std library.

Ample online instructions about every single step…

EQUIPMENT

CS4414 will be a very hands-on course, with a stream of homework assignments that you'll work on for one or two weeks at a time.

Recitation will introduce the things you need to know and drill down on "surprises" relative to Java and Python, but you really learn these things by doing.

You can use your own laptop, or remote login to a Cornell machine in CSUG or MEng lab, but you won't be able to work on an iPad or Android Tablet that lacks a keyboard and mouse.

YOU CAN LEARN IN GROUPS… BUT MUST WORK ALONE

We encourage study groups. Learning as a group is great!

But… *every single thing you do on a graded homework must be done by you, and not with any help from friends or CourseHero or other cheats. Graded work must be your individual work.*

We have outstanding TAs (ugrad and PhD) to help if you get stuck, and you can also post questions on Piazza (or answer them)

EXAMS (NONE), QUIZZES (SOME), HOMEWORK

In 2020, in-person exams are tricky to schedule and proctor and grade, so we are planning to assess purely on at-home work.

This will be mostly homeworks, but we may also have some quizzes that would include questions drawn from lecture material

You are required to watch every lecture either in real-time or offline. We are exploring ways to track this in Canvas.

CURVE

CS4414 is new, so there is no history to base a curve on.

Our feeling is that most students should easily be able to earn a grade in the range from B- to A+.

Grades below B- would only be used if a student really isn't doing well, especially if that person is also skipping lectures.

ACADEMIC INTEGRITY

We use automated tools to look for suspicious code.

Ask us for help if you fall behind or have trouble with a homework. We'll help, and you'll do well in the course.

If you cheat in CS4414 you trade a B- or better for an F. You can't drop a class once this happens or expunge that F, it goes on your record, and you could be expelled from the CS program.

DOWNSTREAM COURSES

CS4414 is a great preparation for other courses!

In AI and ML courses, you'll need to write really high quality, fast code. The experience you gain here will pay off there.

CS4414 feeds naturally into systems courses like O/S, programming languages and compilers, security, networking, databases, embedded systems and IoT, cloud computing…