PLATINUM SPONSORS

stackify

PAIGE TECHNOLOGIES
Our relationships, Your success

Alfresco
doGreatWork()

Balance Innovations

Ingenuity Consulting Partners, Inc.
"Innovation for Your Success"

Sprint

GOLD SPONSORS

KEYHOLE SOFTWARE

UMB

ADVANTAGE TECH

CENTRIQ TRAINING

AdventureTech
www.adventuretechgroup.com

ADAPTIVE SOLUTIONS GROUP

kansas city PMI
KC MID AMERICA CHAPTER

VERSIONONE
Agile Made Easier

perceptivesoftware
from Lexmark

DST SYSTEMS

CARFAX

telerik
deliver more than expected

SILVER SPONSORS

discount ASP.net

Microsoft

ComponentOne
a division of GrapeCity

Why not reinvent the wheel or your knowledge is limited

Seiya Kawashima
skawashima@uchicago.edu
Department of Radiology at The University of Chicago

About this session

a) Who's Seiya Kawashima ?

b) Why do you need to reinvent the wheel ?

c) How much do you need to reinvent the wheel ?

d) What does "right algorithms + right data structures = optimized solutions" mean ?

Who's Seiya Kawashima ?

a) Application Programmer

b) USER of JQuery, JavaScript library

c) Work on open source projects and try to contribute to open source projects, see the details at https://github.com/seiyak

Why do you need to reinvent the wheel ?

a) USEing APIs doesn't necessarily mean that you understand the internals

b) Are able to understand things internally, not just USE them from APIs externally

c) Spend some extra time but GAIN knowledge which is useful in the future. Who doesn't want to learn as programmers ?

d) Are examined about the knowledge in job interviews

Job interview subjects from GXXgle

a) Algorithm Complexity, Design, & Analysis: It's fairly critical that you understand big-O complexity analysis. Again run some practice problems to get this down in application. Sample topics: big-O analysis, sorting and hashing, handling obscenely large amounts of data. Also see topics listed under 'Coding'.

b) Sorting: Know how to sort. Don't do bubble-sort. You should know the details of at least one n*log(n) sorting algorithm, preferably two (say, quicksort and merge sort). Merge sort can be highly useful in situations where quicksort is impractical, so take a look at it.

c) Hashtables: Arguably the single most important data structure known to mankind. You absolutely should know how they work. Be able to implement one using only arrays in your favorite language, in about the space of one interview.

d) Trees: Know about trees; basic tree construction, traversal and manipulation algorithms. Familiarize yourself with binary trees, n-ary trees, and trie-trees. Be familiar with at least one type of balanced binary tree, whether it's a red/black tree, a splay tree or an AVL tree, and know how it's implemented. Understand tree traversal algorithms: BFS and DFS, and know the difference between inorder, postorder and preorder.

e) Graphs: Graphs are really important at Google. There are 3 basic ways to represent a graph in memory (objects and pointers, matrix, and adjacency list); familiarize yourself with each representation and its pros & cons. You should know the basic graph traversal breadth-first search and depth-first search. Know their computational complexity, their tradeoffs, and how to implement them in real code. If you get a chance, try to study up on fancier algorithms, such as Dijkstra and A*.

f) Other data structures: You should study up on as many other data structures and algorithms as possible. You should especially know about the most famous classes of NP-complete problems, such as traveling salesman and the knapsack problem, and be able to recognize them when an interviewer asks you them in disguise. Find out what NP-complete means.

g) Mathematics: Some interviewers ask basic discrete math questions. This is more prevalent at Google than at other companies because we are surrounded by counting problems, probability problems, and other Discrete Math 101 situations. Spend some time before the interview refreshing your memory on (or teaching yourself) the essentials of combinatorics and probability. You should be familiar with n-choose-k problems and their ilk – the more the better.

h) Operating Systems: Know about processes, threads and concurrency issues. Know about locks and mutexes and semaphores and monitors and how they work. Know about deadlock and livelock and how to avoid them. Know what resources a processes needs, and a thread needs, and how context switching works, and how it's initiated by the operating system and underlying hardware. Know a little about scheduling. The world is rapidly moving towards multi-core, so know the fundamentals of "modern" concurrency constructs.

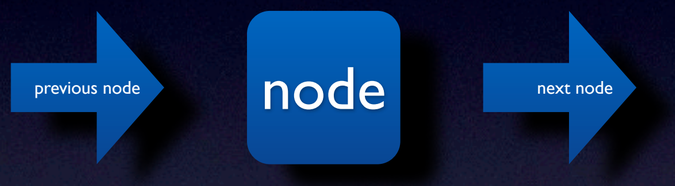
i) Coding: You should know at least one programming language really well. You will be expected to write some code in at least some of your interviews. You will be expected to know a fair amount of detail about your favorite programming language. Sample topics: construct / traverse data structures, implement system routines, distill large data sets to single values, transform one data set to another.

j) System Design Sample topics: features sets, interfaces, class hierarchies, designing a system under certain constraints, simplicity and robustness, tradeoffs.

k) Open-Ended Discussion, Sample topics: biggest challenges faced, best/worst designs seen, performance analysis and optimization, testing, ideas for improving existing products.

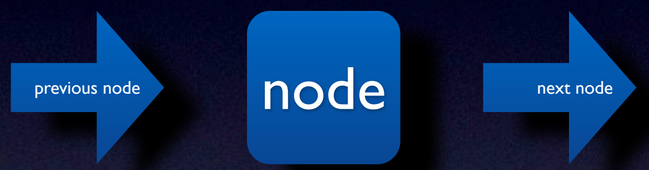
Examples of Data Structures

Linked List



Examples of Data Structures

Linked List



Hash Map

Examples of Data Structures

Linked List

previous node

node

next node

Hash Map

key

index

value

Tree

parent

left child

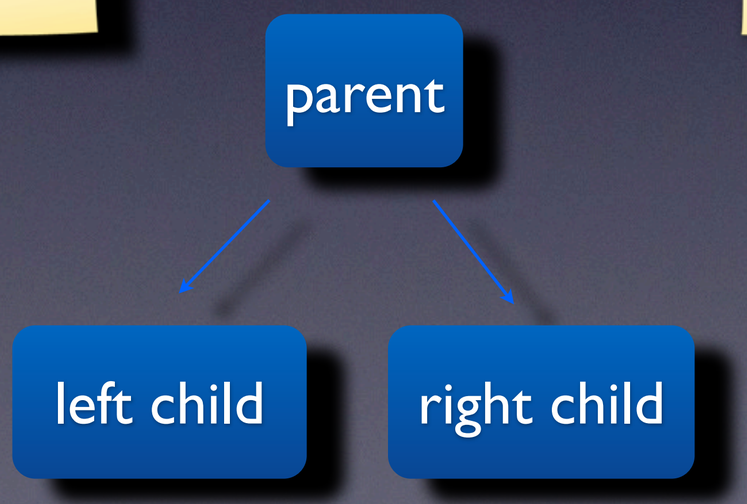
right child

Examples of Data Structures

Linked List

Hash Map

Tree



Set

Examples of Java APIs

a) Apache MINA

Multipurpose Infrastructure for Network Application

Nicely hide complexity for non blocking network Application

Easy to use

b) java.util.concurrent package

No fear to write applications with threads

Easy to use

Resources:

Apache MINA logo

http://upload.wikimedia.org/wikipedia/commons/3/34/Apache-MINA-logo.png

Examples of Java APIs

```
ServerSocketChannel so = null;
SocketChannel sc = null;

while ( true ) {
    try {
        if ( key.selector().select() <= 0 ) {
            continue;
        }

        Iterator itr = key.selector().selectedKeys().iterator();
        while ( itr.hasNext() ) {
            SelectionKey ready = itr.next();
            itr.remove();

            if ( ready.isValid() && ready.isReadable() ) {
                sc = (SocketChannel) ready.channel();
                 //Do something when read is ready
            }
            else if ( ready.isValid() && ready.isAcceptable() ) {

                so = (ServerSocketChannel) key.channel();

                sc = so.accept();
                sc.configureBlocking( false );
                sc.register( selector, SelectionKey.OP_READ | SelectionKey.OP_CONNECT );
            }
        }
    }
    catch ( IOException e ) {
        log.error( e );
    }
    catch ( Exception ex ) {
        log.error( ex );
    }
}
```

Examples of Java APIs

```
import org.apache.mina.core.service.IoAcceptor;
import org.apache.mina.transport.socket.nio.NioSocketAcceptor;

IoAcceptor acceptor = new NioSocketAcceptor();
acceptor.setHandler(  new YourApplicationLogic() );
```

```
import org.apache.mina.core.service.IoHandler;

public class YourApplicationLogic implements IoHandler{}
```

Or

```
import org.apache.mina.core.service.IoHandlerAdapter;

public class YourApplicationLogic extends IoHandlerAdapter{}
```

How much do you need to reinvent the wheel ?

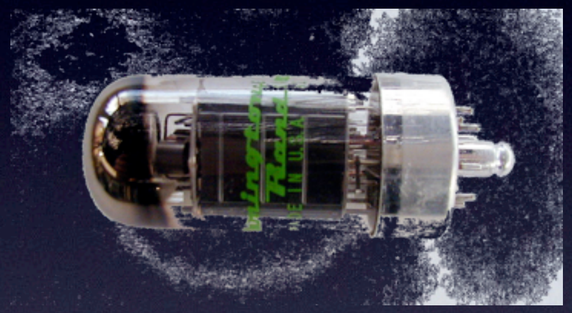
Resources:

History of Computer

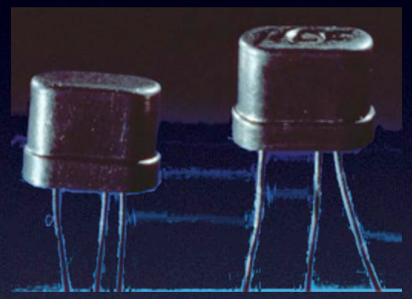
http://people.uncw.edu/tompkinsj/112/texnh/handouts/historyHandout.html

No GIF
http://thisblogisforwomen.com/wp-content/uploads/2012/02/no.gif?w=300

How much do you need to reinvent the wheel ?



Vacuum tube ?



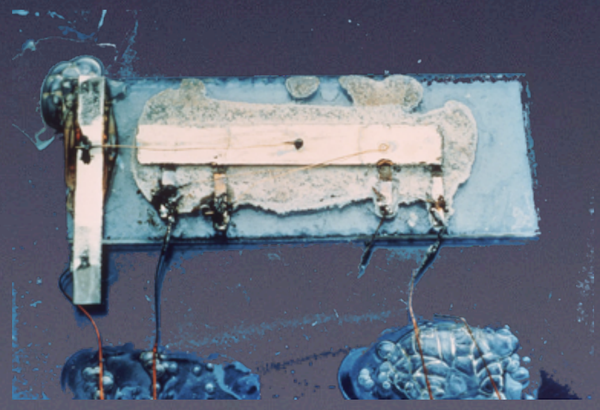
Transistor ?

Resources:

History of Computer

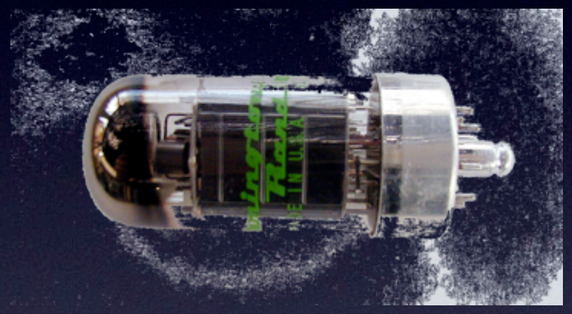
http://people.uncw.edu/tompkinsj/112/texnh/handouts/historyHandout.html

No GIF
http://thisblogisforwomen.com/wp-content/uploads/2012/02/no.gif?w=300

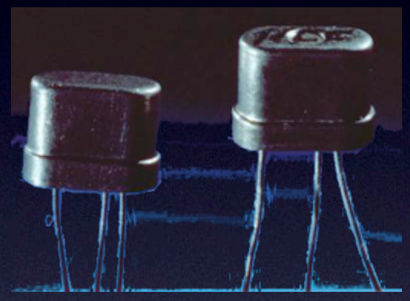


Integrated Circuit ?

How much do you need to reinvent the wheel ?



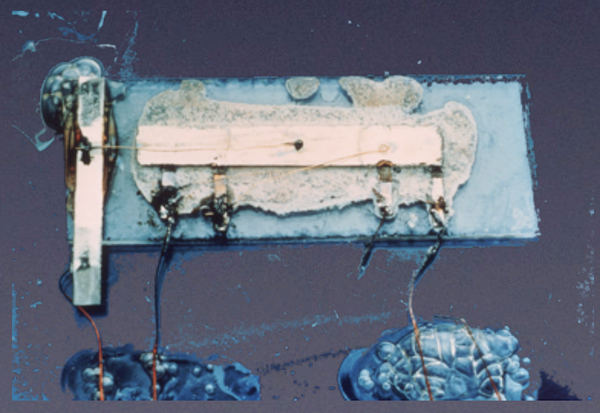
Maybe



Maybe

Vacuum tube ?

Transistor ?



Maybe

Integrated Circuit ?

Resources:

History of Computer

http://people.uncw.edu/tompkinsj/112/texnh/handouts/historyHandout.html

No GIF
http://thisblogisforwomen.com/wp-content/uploads/2012/02/no.gif?w=300

right algorithms + right data structures = optimized solutions

a) Based on “Algorithms + Data Structures = Programs” by Niklaus Wirth published in 1976

b) Need to carefully choose internal structures of your programs to meet your requirements

c) Would be very surprising if compilers change data structures from list to array as an optimization

right algorithms + right data structures = optimized solutions

a) org.apache.mina.http.HttpServerDecoder.decode()

Converts incoming strings to http header and http message for the further work

Uses heavily regular expressions to split the messages to extract data

b) sfs.async.handler.http.reader.HttpMessageReader.findEndOfMessage()

Converts incoming strings to http header and http message for the further work

Uses Boyer-Moore String search algorithms to split the messages to extract data

Resources:

Boyer-Moore Fast String Search
http://www.cs.utexas.edu/~moore/best-ideas/string-searching/

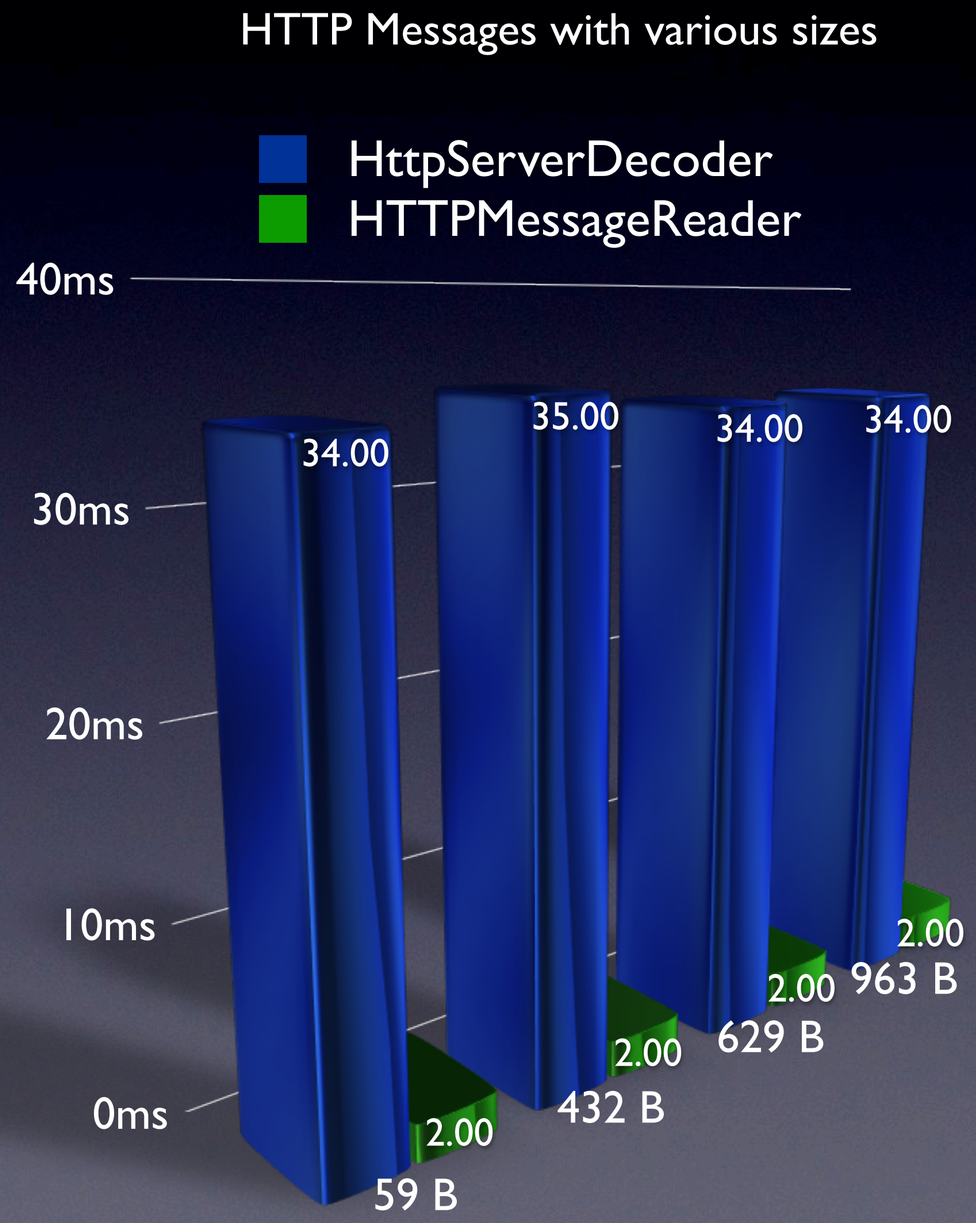
Boyer Moore Algorithm Understanding and Example ?
http://stackoverflow.com/questions/6207819/boyer-moore-algorithm-understanding-and-example

SFS, Seiya File System
https://github.com/seiyak/SFS---Seiya-File-System

right algorithms + right data structures = optimized solutions

- HttpServerDecoder
- HTTPMessageReader

right algorithms + right data structures = optimized solutions



right algorithms + right data structures = optimized solutions

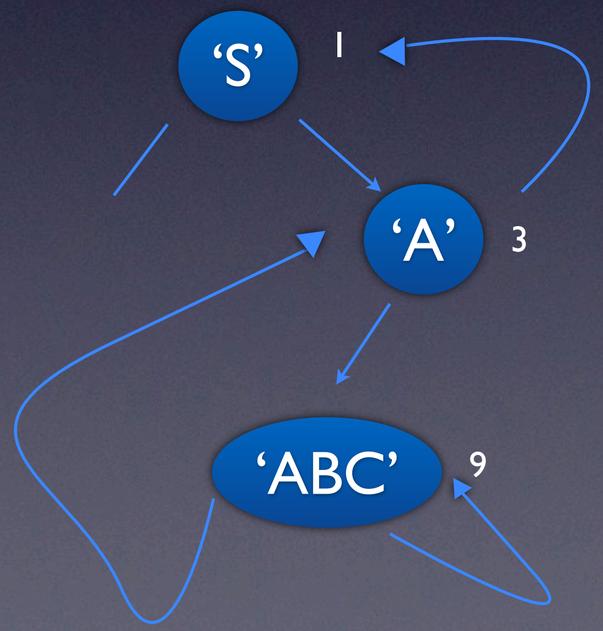
String search using a data structure

a) List

b) PATRICIA Trie, Practical Algorithm To Retrieve Information Coded In Alphanumeric, invented by Donald R. Morrison in 1968

List

PATRICIA Trie



Bit Patterns in ASCII code

	0	1	2	3	4	5	6	7	8	9
'S'	0	1	0	1	0	0	1	1	0	0
'A'	0	1	0	0	0	0	0	1	0	0
'ABC'	0	1	0	0	0	0	0	1	0	1

Demo

Resources:

Algorithms in C Third Edition, Chapter 15.3 Patricia Tries
http://www.amazon.com/Algorithms-Parts-1-4-Fundamentals-Structures/dp/0201314525

Introduction to Algorithms, Chapter 12-2 Radix trees
http://www.amazon.com/Introduction-Algorithms-Thomas-H-Cormen/dp/0262033844

Radix Search
http://www.mif.vu.lt/~algis/dsax/DsRadix.pdf

right algorithms + right data structures = optimized solutions

String prefix search using a data structure

List

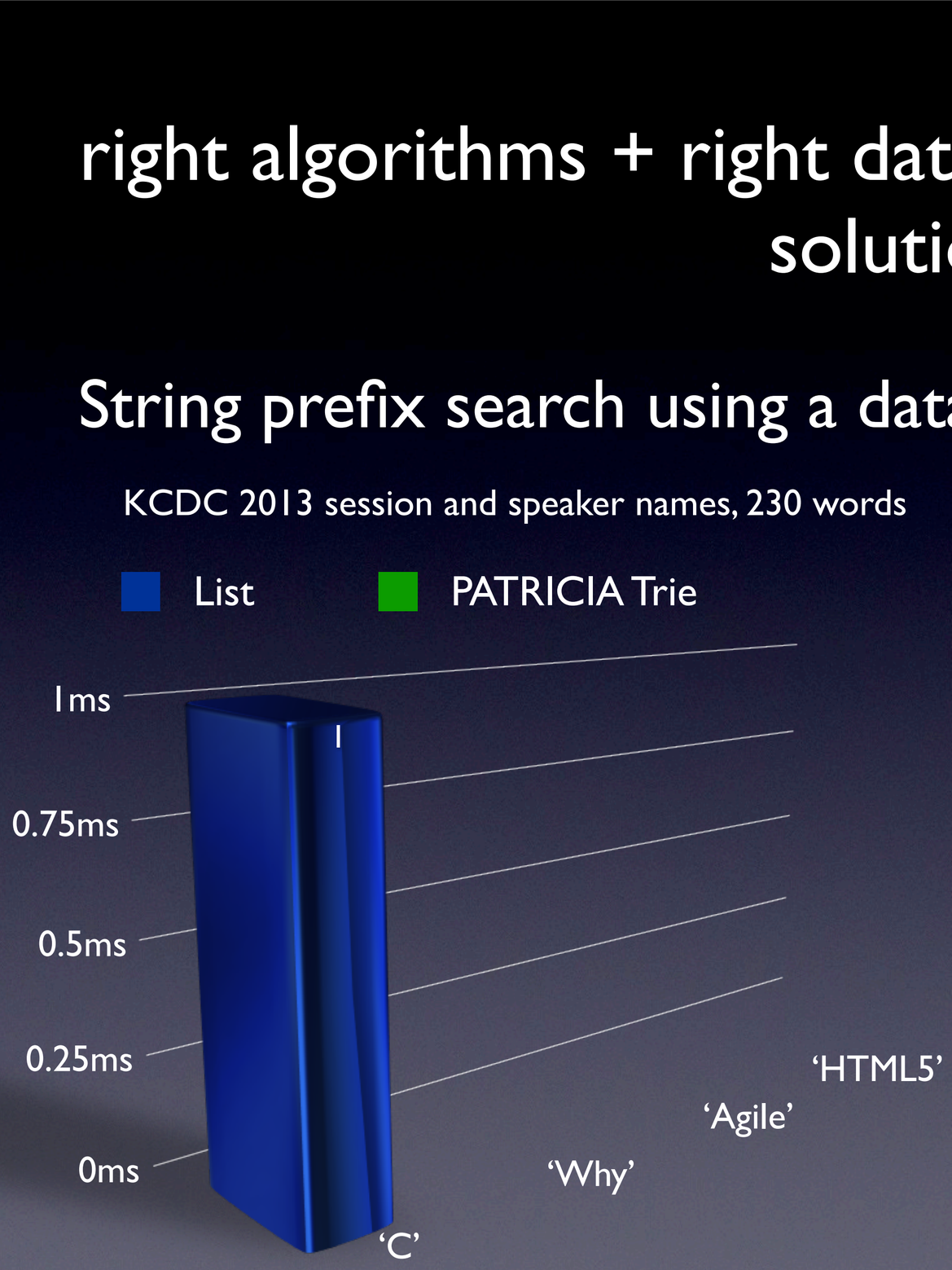
PATRICIA Trie

right algorithms + right data structures = optimized solutions

String prefix search using a data structure

KCDC 2013 session and speaker names, 230 words

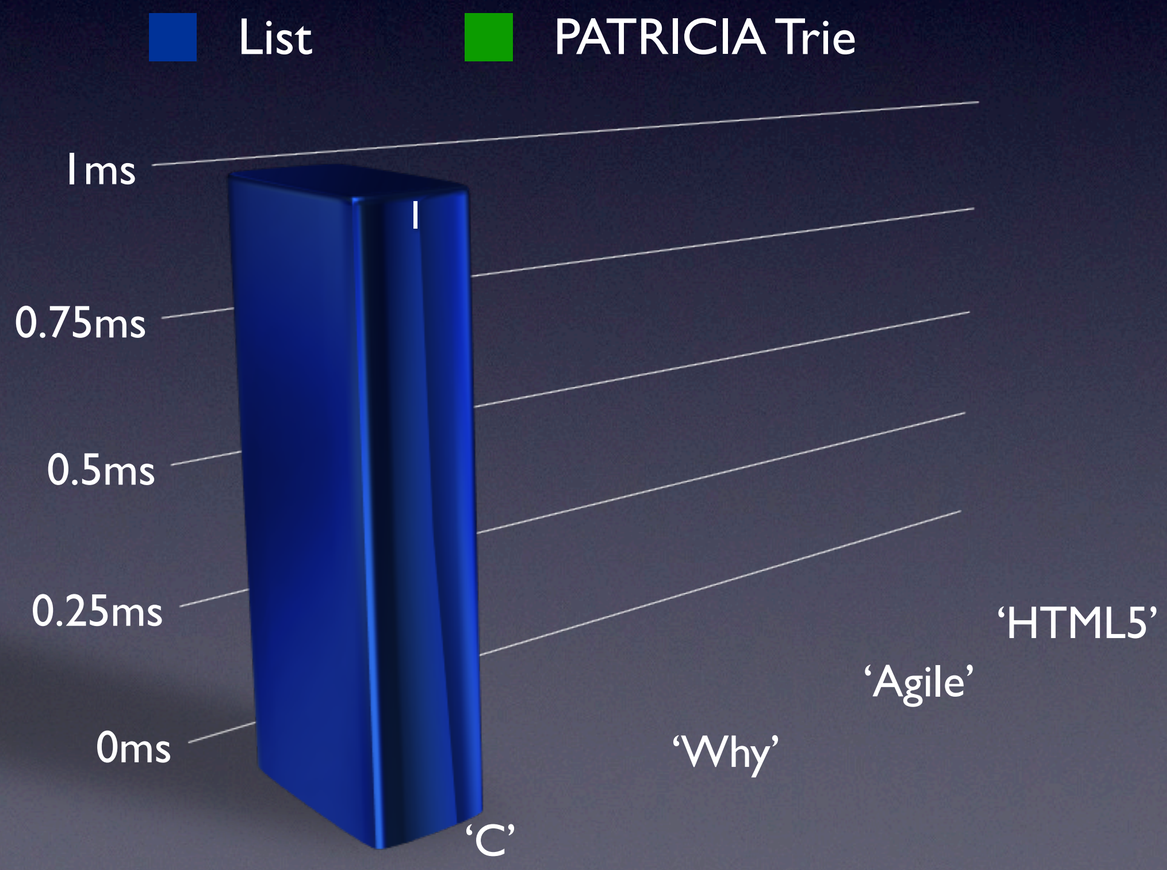
List PATRICIA Trie



right algorithms + right data structures = optimized solutions

String prefix search using a data structure

KCDC 2013 session and speaker names, 230 words



right algorithms + right data structures = optimized solutions

String prefix search using a data structure

KCDC 2013 session and speaker names, 230 words

List PATRICIA Trie

99,171 English words at /usr/share/dict/words

List PATRICIA Trie