The Why of Go

21st Century Programming Languages Track, Qcon SF

Carmen Andoh

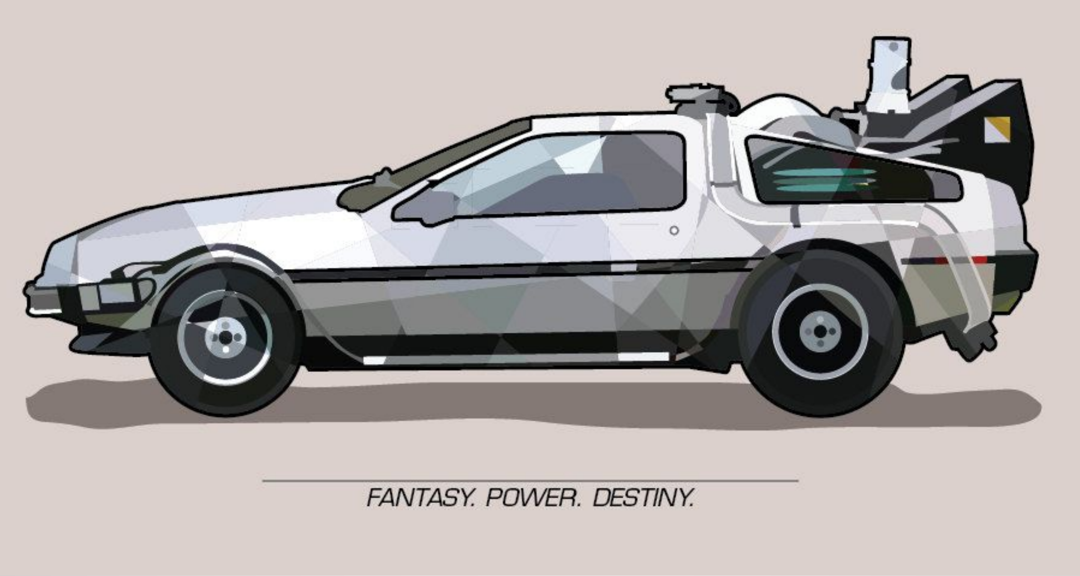
1983

RESTRICTED AREA
NO TRESPASSING
U.S. GOVERNMENT
PROPERTY
STRANGER
THINGS

Thank you, fellow time travelers from the Future

- *Dave Cheney*
- *Alan Donovan*
- *Steve Francia*
- *Jérôme Pettazoni*

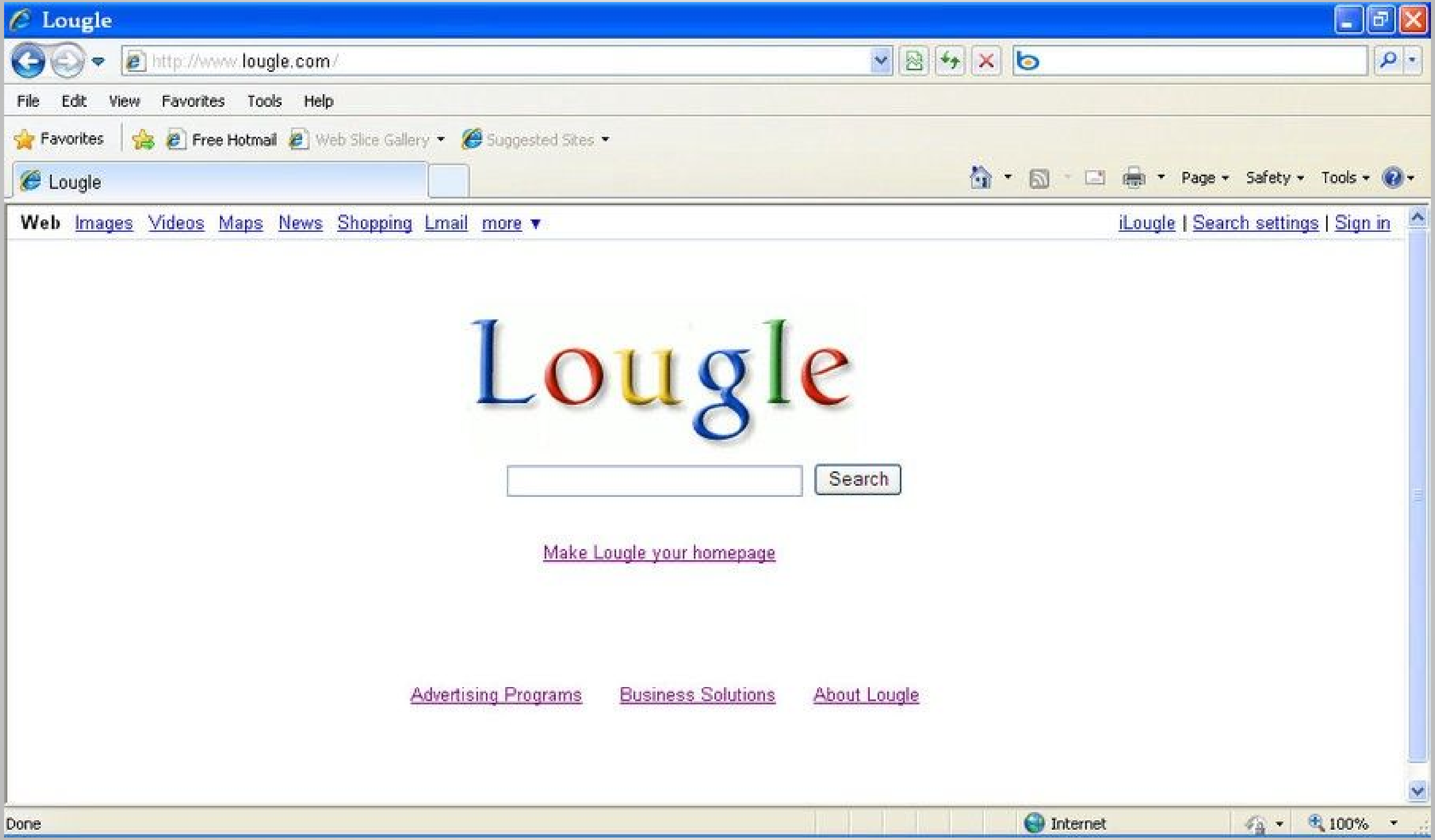
Back to 1983 … (from 1985, but whetev, 80s rule)



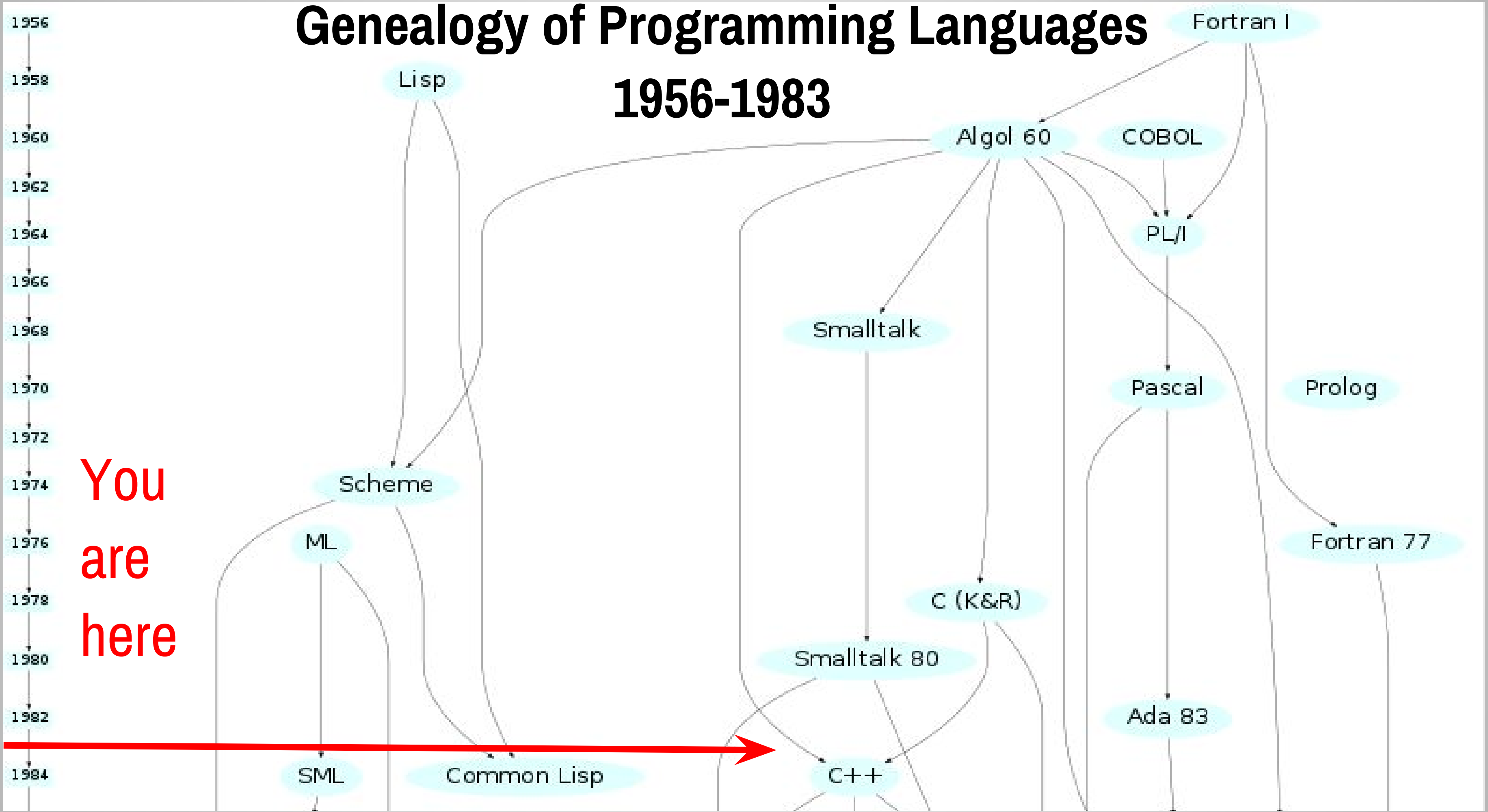
The 80s
ARE COOL
IN 2017

Google | google.com/datacenters

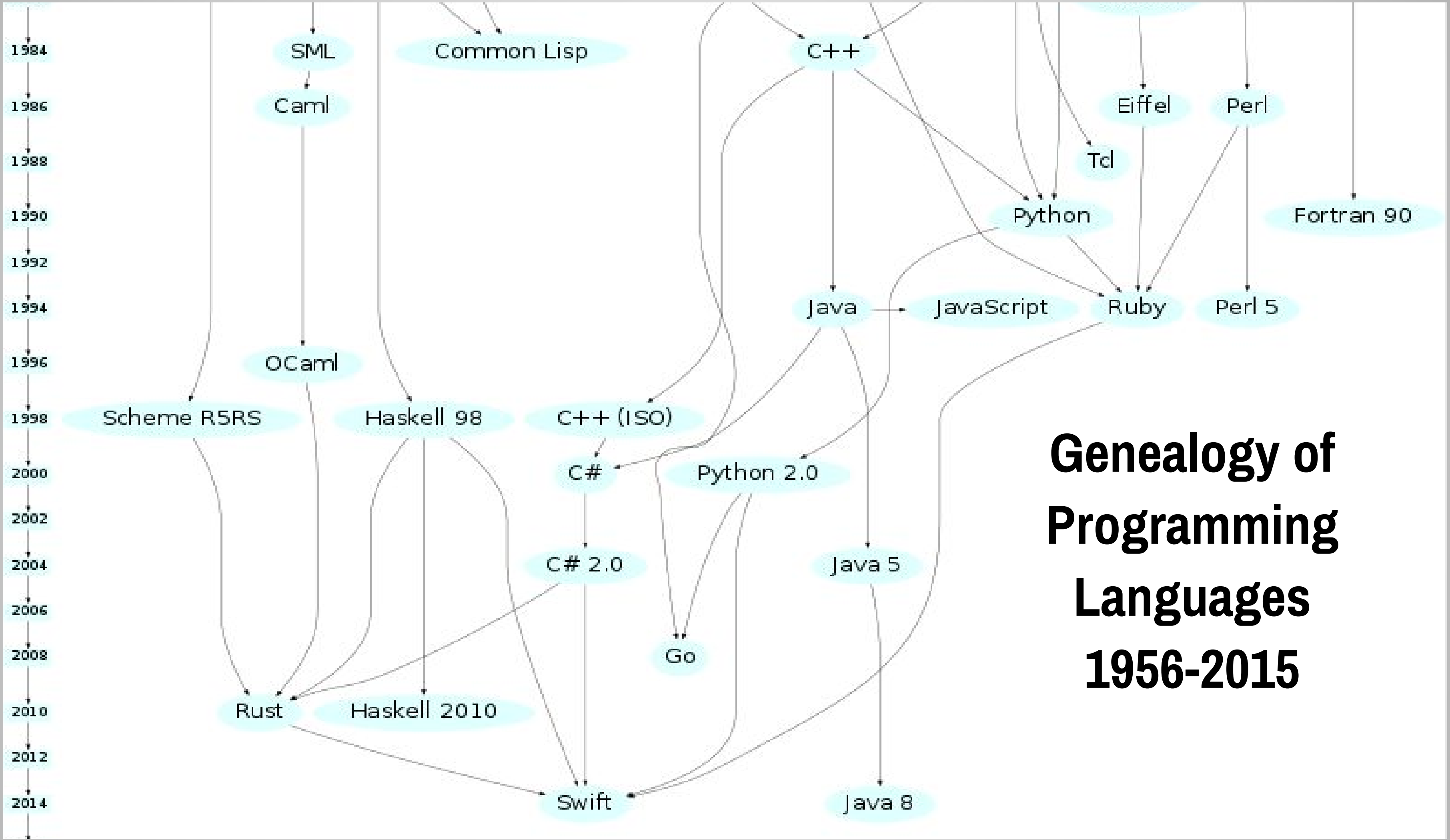
SAMSUNG
12:45
2:45
08:16
06:23
07:06
11:00
2:00
01:10
LG
oppo


Lougle
http://www.lougle.com/
File
Edit
View
Favorites
Tools
Help
Favorites
Free Hotmail
Web Slice Gallery
Suggested Sites
Lougle
Page
Safety
Tools
Web
Images
Videos
Maps
News
Shopping
Lmail
more ▾
iLougle | Search settings | Sign in
Lougle
Search
Make Lougle your homepage
Advertising Programs
Business Solutions
About Lougle
Done
Internet
100%

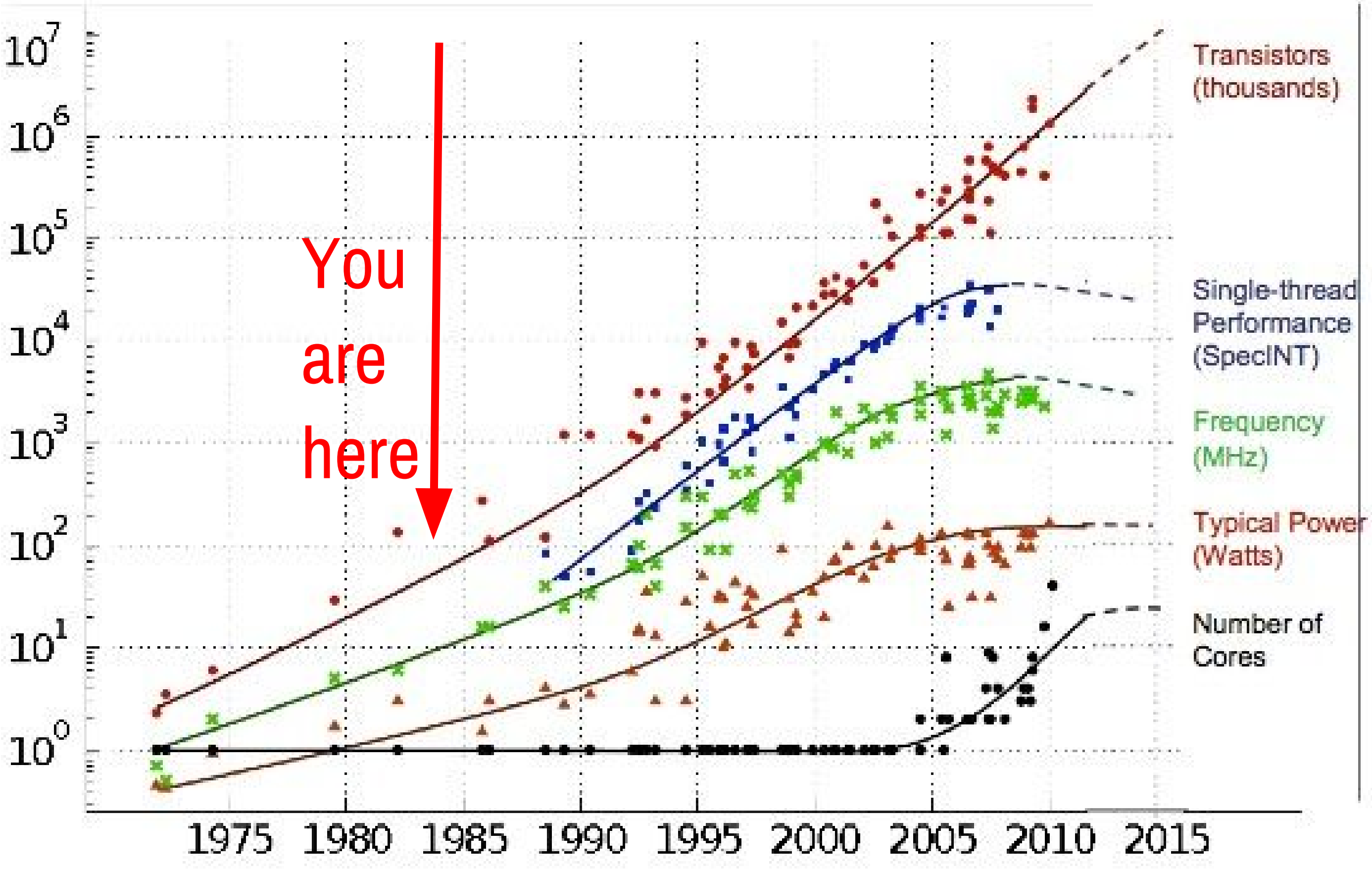
Genealogy of Programming Languages 1956-1983



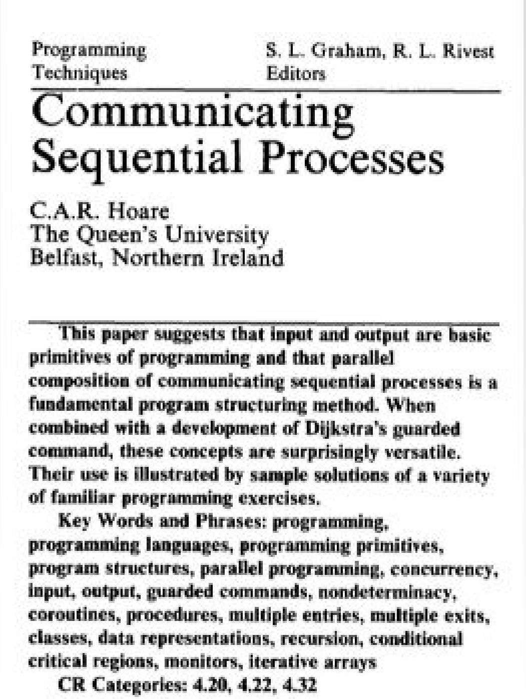
You are here



Genealogy of Programming Languages 1956-2015



Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten
Dotted line extrapolations by C. Moore



Programming Techniques

S. L. Graham, R. L. Rivest Editors

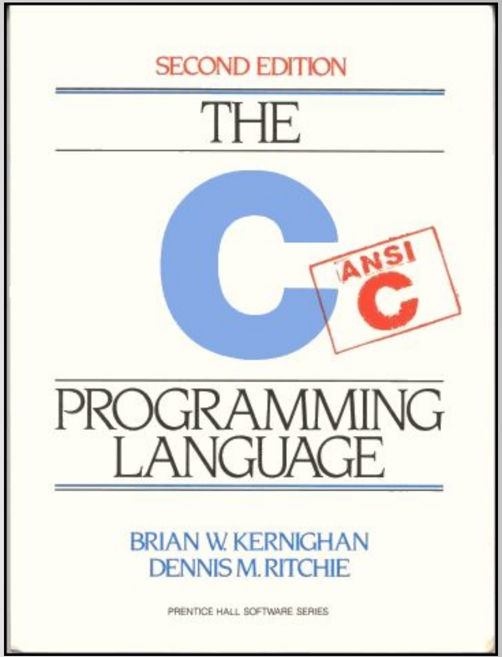
Communicating Sequential Processes

C.A.R. Hoare
The Queen's University
Belfast, Northern Ireland

This paper suggests that input and output are basic primitives of programming and that parallel composition of communicating sequential processes is a fundamental program structuring method. When combined with a development of Dijkstra's guarded command, these concepts are surprisingly versatile. Their use is illustrated by sample solutions of a variety of familiar programming exercises.

Key Words and Phrases: programming, programming languages, programming primitives, program structures, parallel programming, concurrency, input, output, guarded commands, nondeterminacy, coroutines, procedures, multiple entries, multiple exits, classes, data representations, recursion, conditional critical regions, monitors, iterative arrays

CR Categories: 4.20, 4.22, 4.32



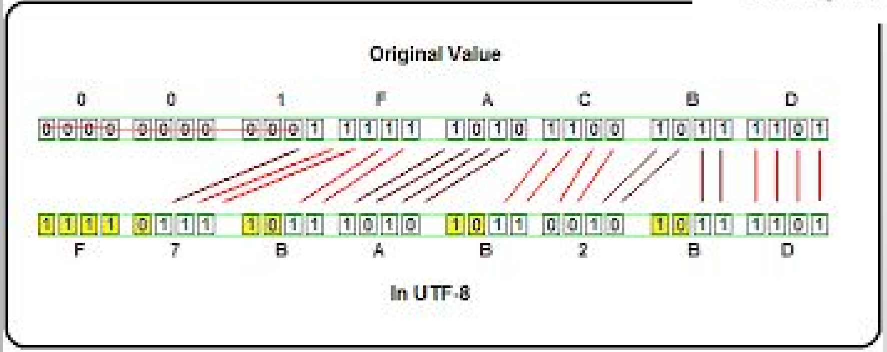
SECOND EDITION

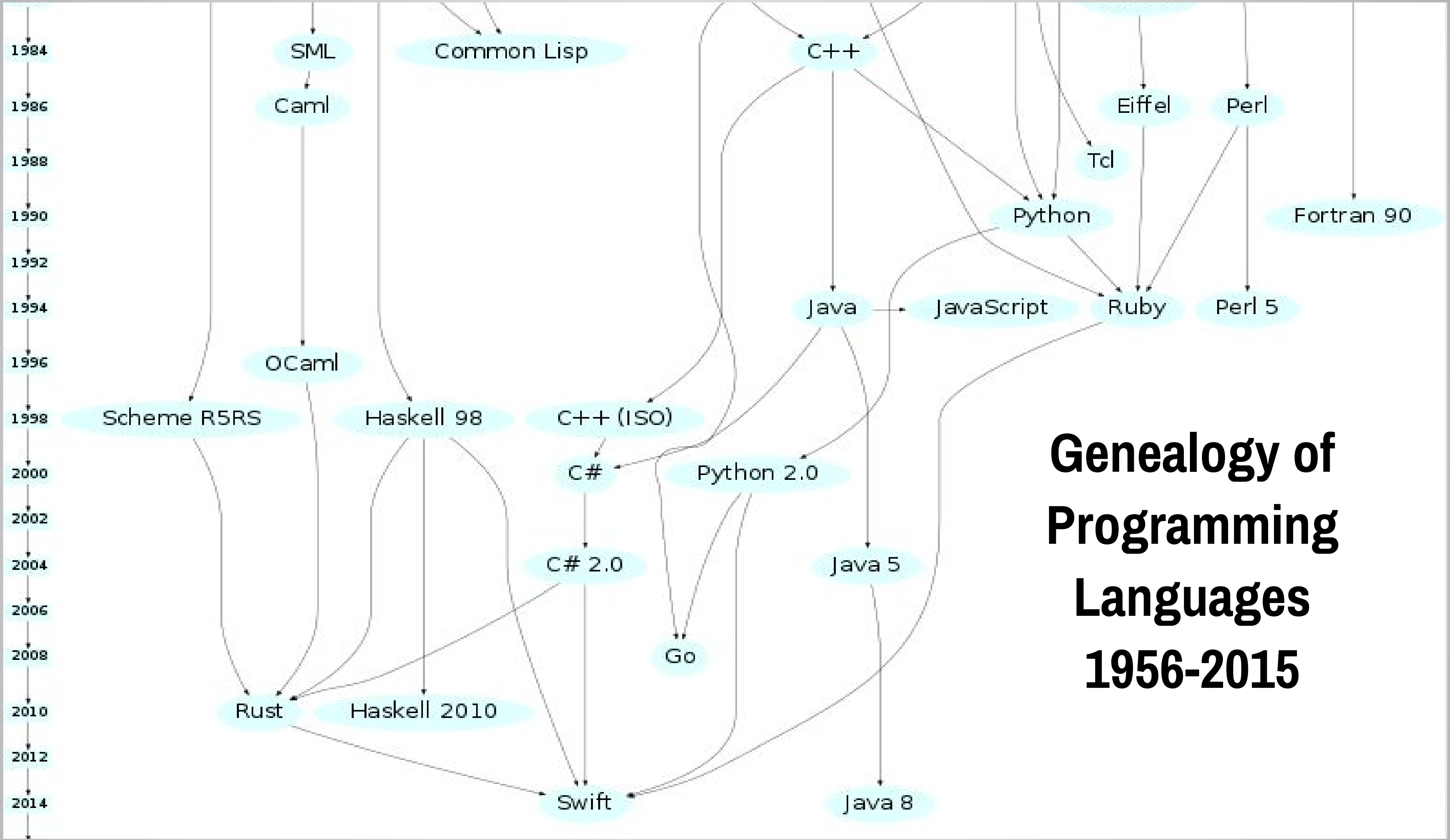
THE C PROGRAMMING LANGUAGE

ANSI C

BRIAN W. KERNIGHAN
DENNIS M. RITCHIE

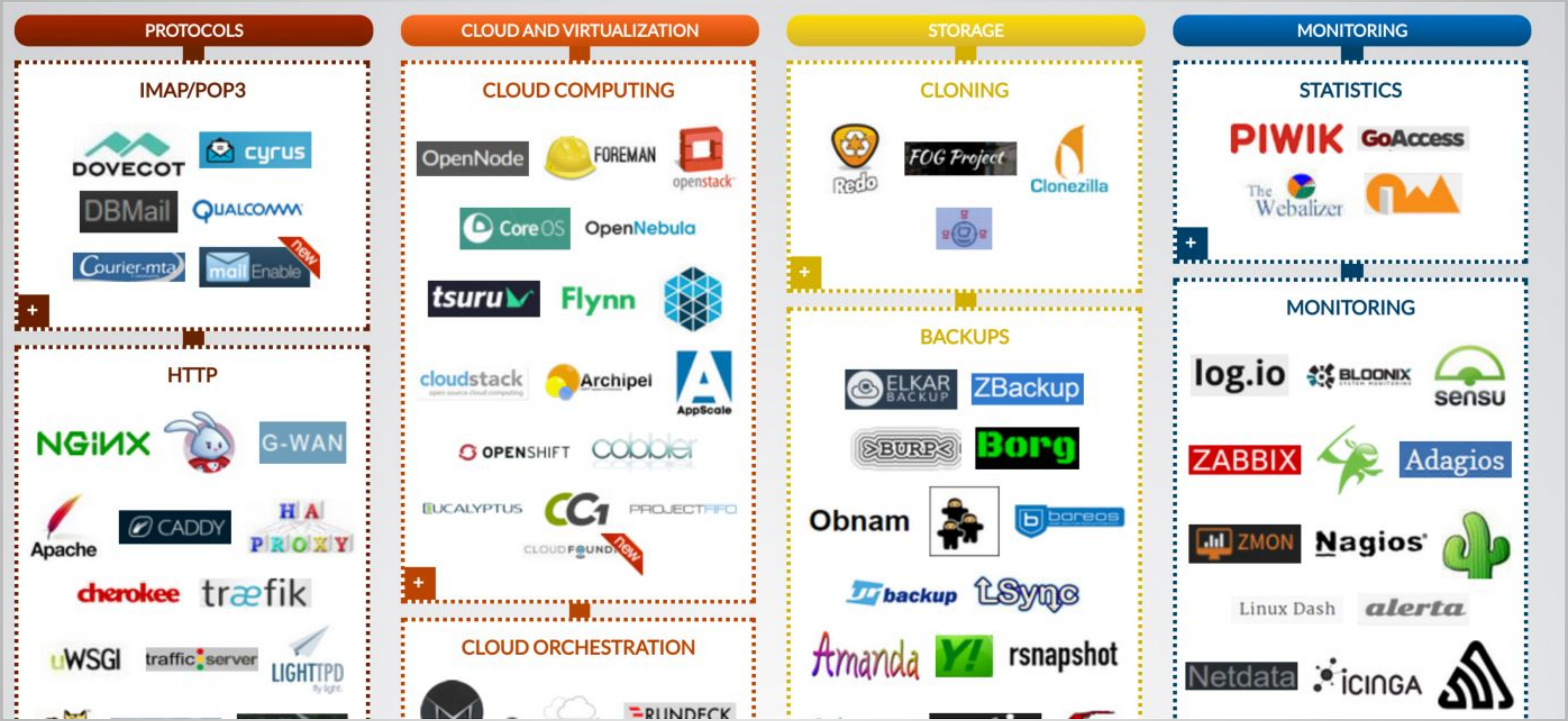
PRENTICE HALL SOFTWARE SERIES





Genealogy of Programming Languages 1956-2015

"What's that?" snapped the King.
And he looked down the stack.
And he saw at the bottom, a turtle named Mack.



PROTOCOLS

IMAP/POP3

HTTP

CLOUD AND VIRTUALIZATION

CLOUD COMPUTING

CLOUD ORCHESTRATION

STORAGE

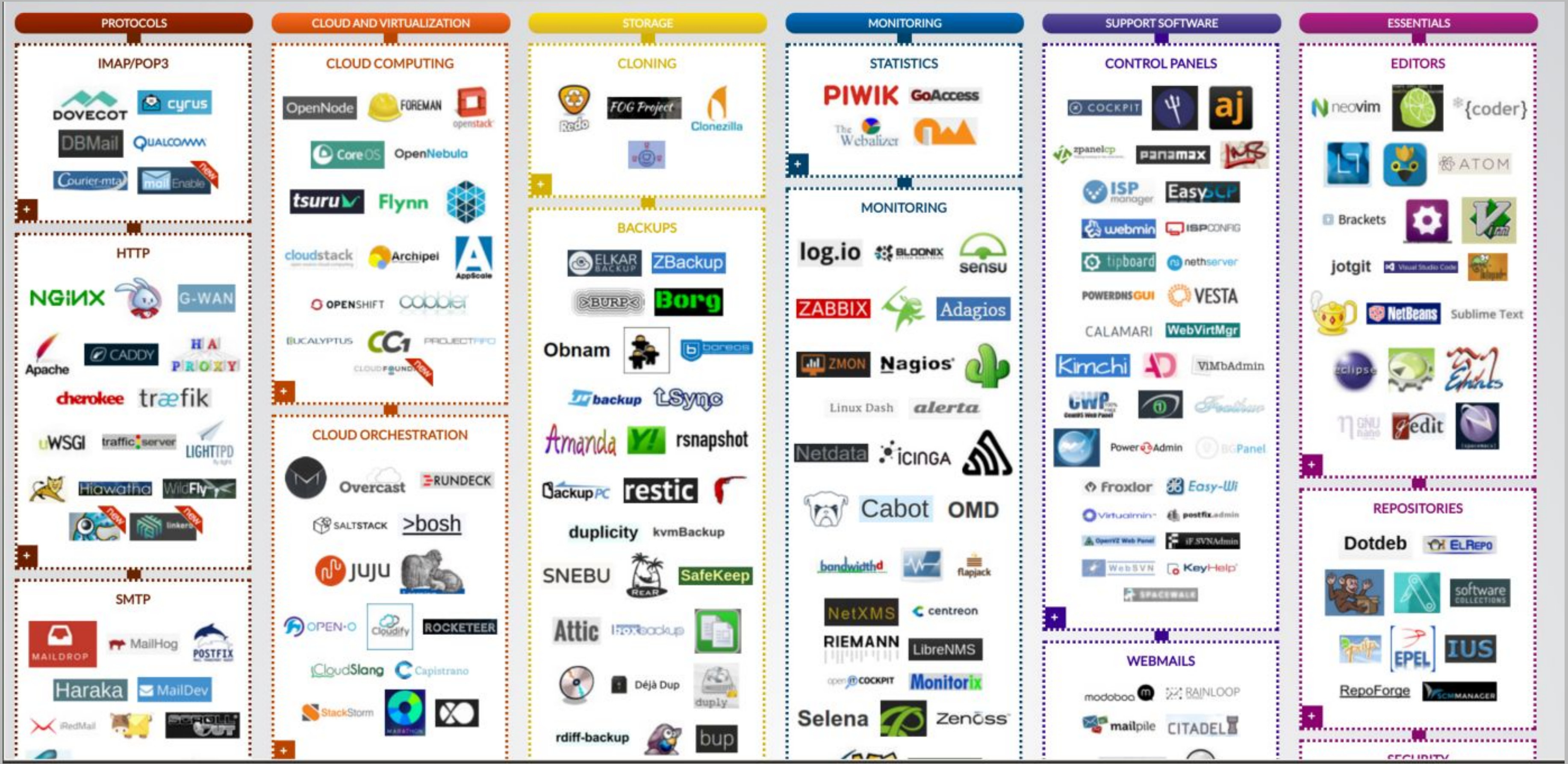
CLONING

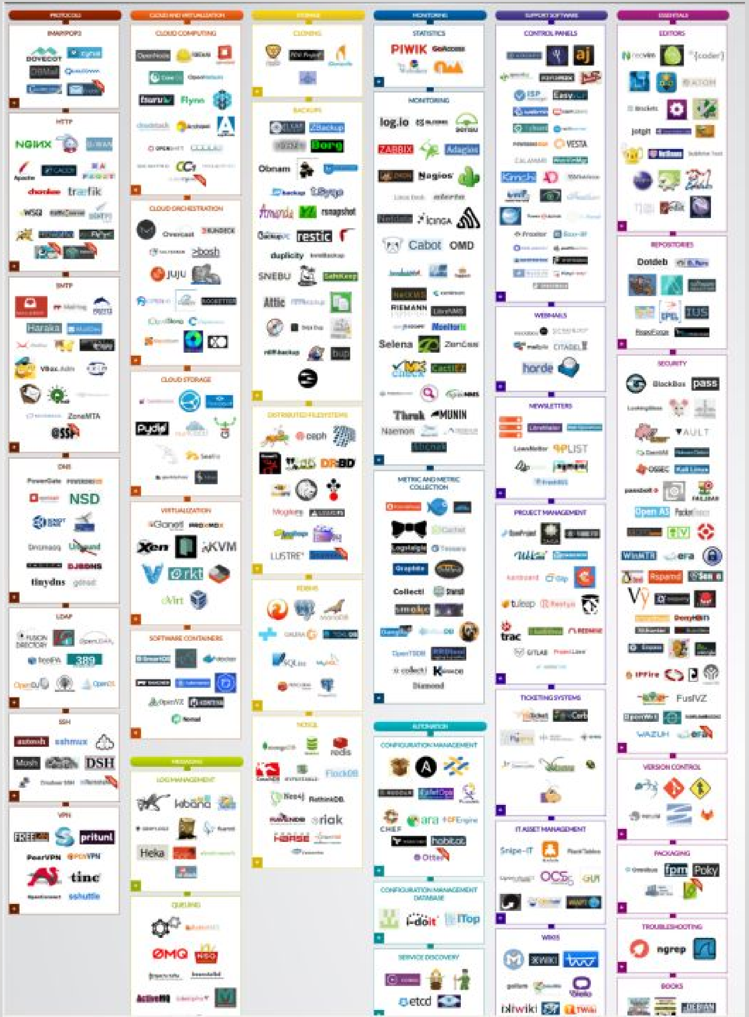
BACKUPS

MONITORING

STATISTICS

MONITORING

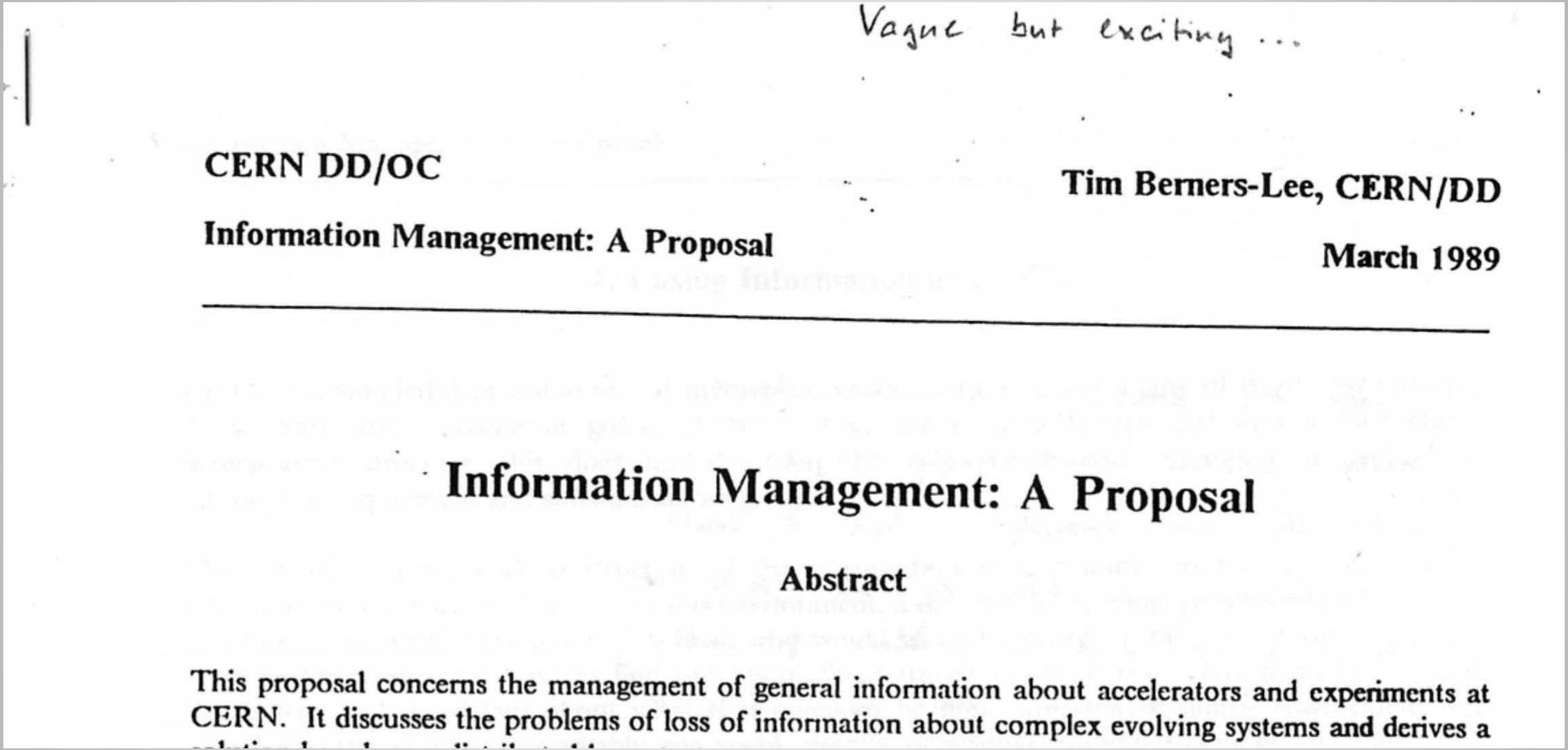

PROTOCOLS
IMAP/POP3
DOVECOT
cyrus
DBMail
QUALCOMM
Courier-mta
mail Enable
new
HTTP
NGINX
G-WAN
Apache
CADDY
HA PROXY
cherokee
træfik
uWSGI
traffic server
LIGHTTPD
Hiawatha
WildFly
new
linkerd
new
SMTP
MAILDROP
MailHog
POSTFIX
Haraka
MailDev
iRedMail
CLOUD AND VIRTUALIZATION
CLOUD COMPUTING
OpenNode
FOREMAN
openstack
CoreOS
OpenNebula
tsuru
Flynn
cloudstack
Archipel
AppScale
OPENSHIFT
cobbler
EUCALYPTUS
CC1
PROJECTFIFO
CLOUD FOUNDRY
new
CLOUD ORCHESTRATION
Overcast
RUNDECK
SALTSTACK
>bosh
juju
OPEN-O
Cloudify
ROCKETEER
CloudSlang
Capistrano
StackStorm
MARATHON
STORAGE
CLONING
Redo
FOG Project
Clonezilla
BACKUPS
ELKAR BACKUP
ZBackup
BURP
Borg
Obnam
bareos
backup
LSyncd
Amanda
rsnapshot
BackupPC
restic
duplicity
kvmBackup
SNEBU
REAR
SafeKeep
Attic
boxbackup
Déjà Dup
duply
rdiff-backup
bup
MONITORING
STATISTICS
PIWIK
GoAccess
The Webalizer
MONITORING
log.io
BLOONIX
sensu
ZABBIX
Adagios
ZMON
Nagios
Linux Dash
alerta
Netdata
icinga
Cabot
OMD
bandwidthd
flapjack
NetXMS
centreon
RIEMANN
LibreNMS
open COCKPIT
Monitorix
Selena
Zenoss
SUPPORT SOFTWARE
CONTROL PANELS
COCKPIT
aj
zpanelcp
panamax
ISP manager
EasySCP
webmin
ISPCONFIG
tipboard
nethserver
POWERDNSGUI
VESTA
CALAMARI
WebVirtMgr
Kimchi
ViMbAdmin
CWP
Feathur
PowerAdmin
BCPanel
Froxlor
Easy-Wi
Virtualmin
postfix.admin
OpenVZ Web Panel
iF.SVNAdmin
WebSVN
KeyHelp
SPACEWALK
WEBMAILS
modoboa
RAINLOOP
mailpile
CITADEL
ESSENTIALS
EDITORS
neovim
{coder}
ATOM
Brackets
jotgit
Visual Studio Code
NetBeans
Sublime Text
eclipse
GNU nano
gedit
REPOSITORIES
Dotdeb
ELRepo
software collections
EPEL
IUS
RepoForge
SCMMANAGER
SECURITY


PROTOCOLS
IMAP/POP3
HTTP
SMTP
DNS
LDAP
SSH
VPN
CLOUD AND VIRTUALIZATION
CLOUD COMPUTING
CLOUD ORCHESTRATION
CLOUD STORAGE
VIRTUALIZATION
SOFTWARE CONTAINERS
MESSAGING
LOG MANAGEMENT
QUEUING
STORAGE
CLONING
BACKUPS
DISTRIBUTED FILESYSTEMS
RDBMS
NOSQL
MONITORING
STATISTICS
MONITORING
METRIC AND METRIC COLLECTION
AUTOMATION
CONFIGURATION MANAGEMENT
CONFIGURATION MANAGEMENT DATABASE
SERVICE DISCOVERY
SUPPORT SOFTWARE
CONTROL PANELS
WEBMAILS
NEWSLETTERS
PROJECT MANAGEMENT
TICKETING SYSTEMS
IT ASSET MANAGEMENT
WIKIS
ESSENTIALS
EDITORS
REPOSITORIES
SECURITY
VERSION CONTROL
PACKAGING
TROUBLESHOOTING
BOOKS

GNU

GNU's Not Unix Project

Started in 1983



Vague but exciting ...

CERN DD/OC

Information Management: A Proposal

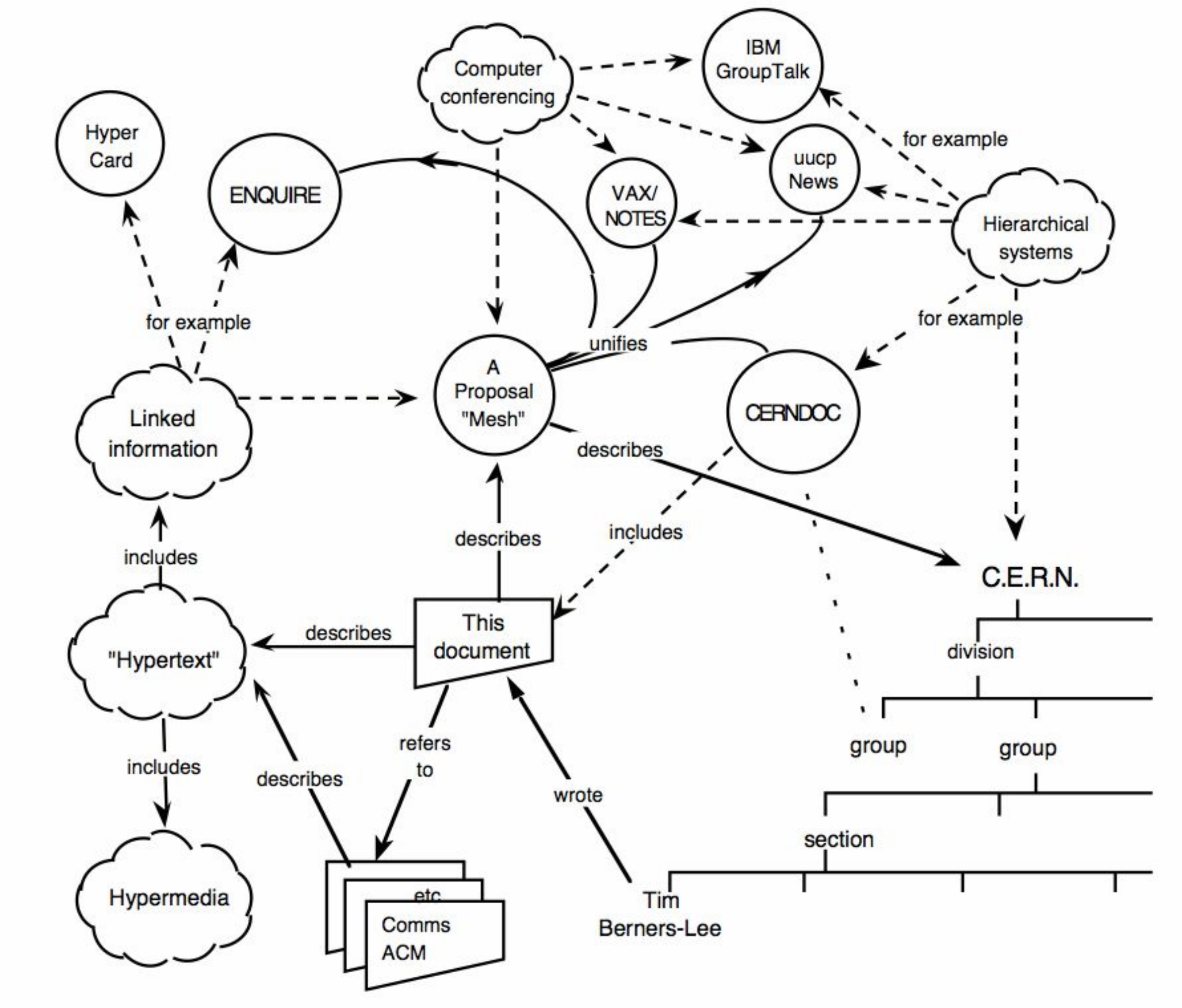
Tim Berners-Lee, CERN/DD

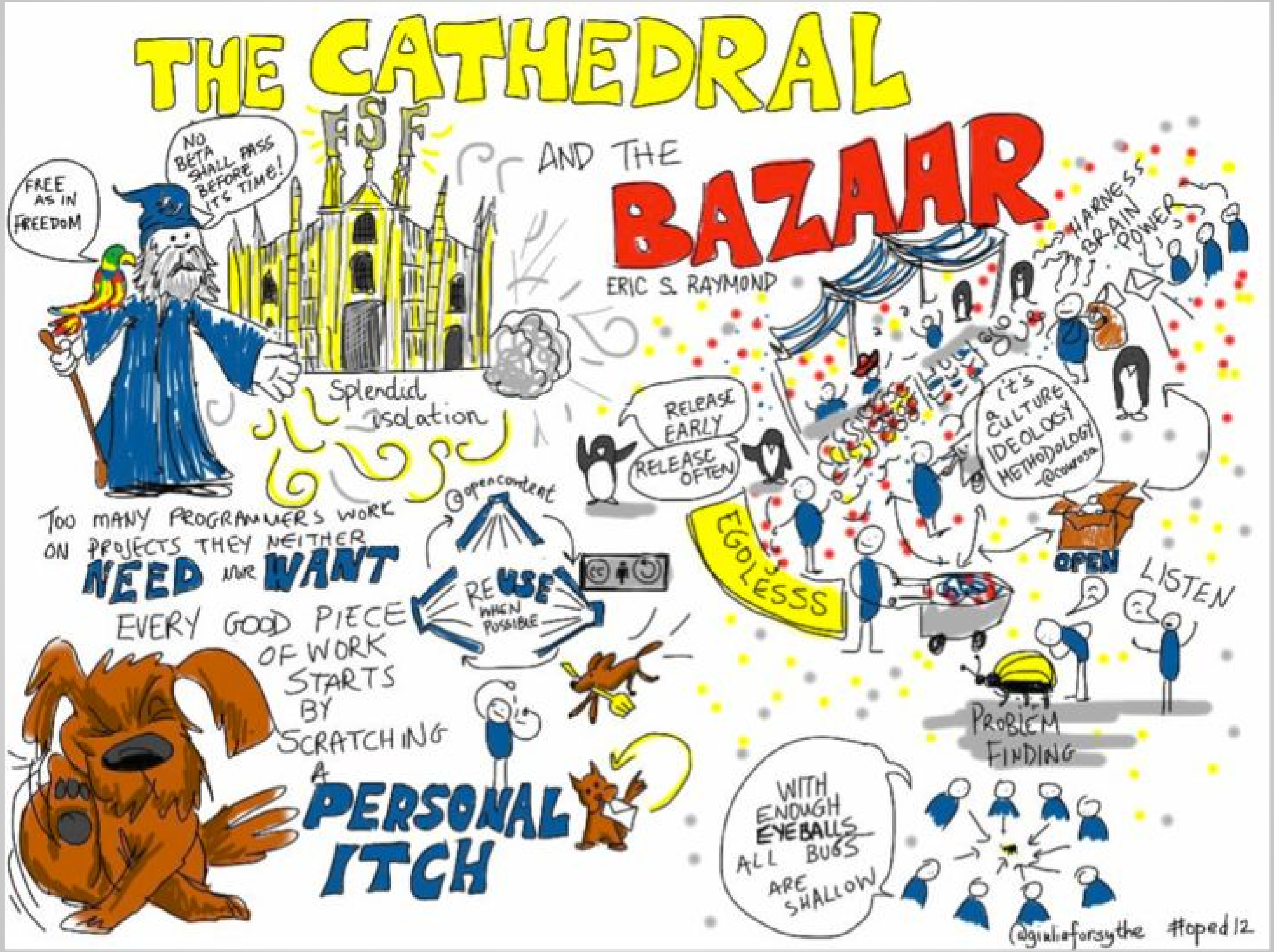
March 1989

Information Management: A Proposal

Abstract

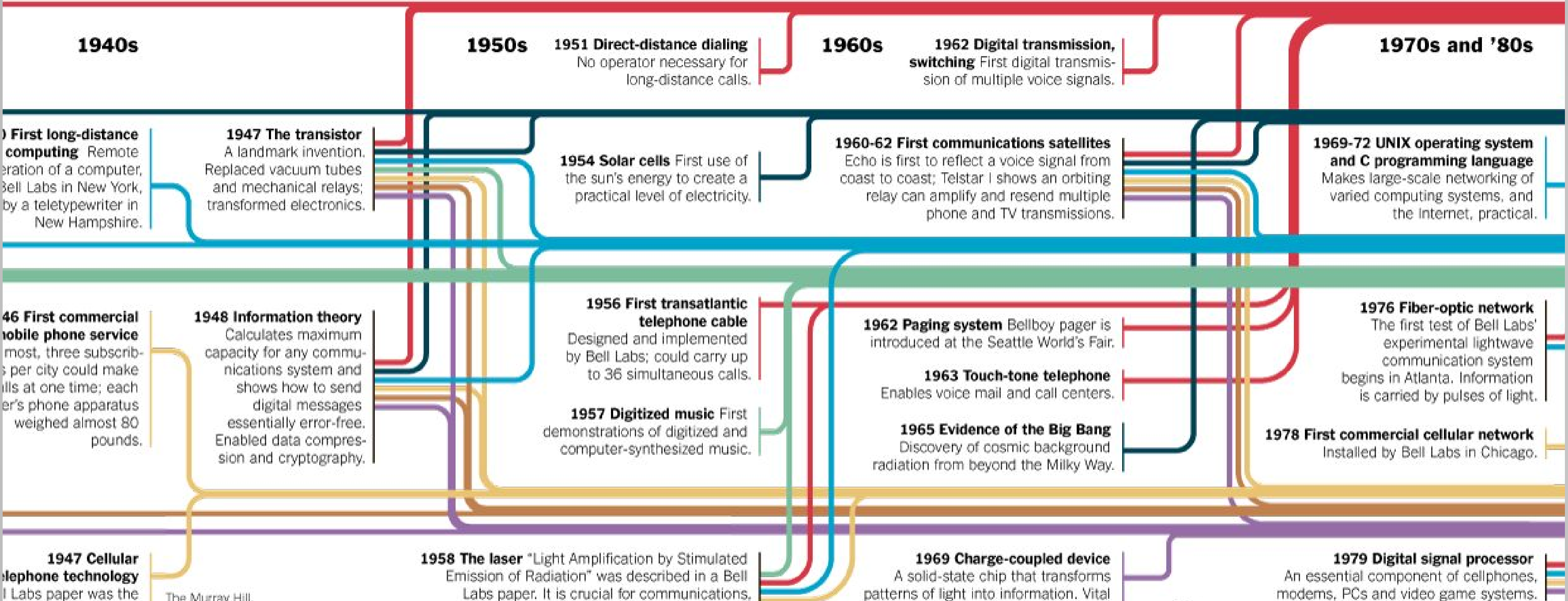
This proposal concerns the management of general information about accelerators and experiments at CERN. It discusses the problems of loss of information about complex evolving systems and derives a


Computer conferencing
IBM GroupTalk
Hyper Card
ENQUIRE
VAX/ NOTES
uucp News
for example
Hierarchical systems
for example
unifies
for example
Linked information
A Proposal "Mesh"
CERNDOC
describes
includes
describes
includes
C.E.R.N.
division
"Hypertext"
describes
This document
group
group
includes
refers to
describes
wrote
section
Hypermedia
etc
Comms ACM
Tim Berners-Lee


THE CATHEDRAL
FSF
AND THE
BAZAAR
ERIC S. RAYMOND
FREE AS IN FREEDOM
NO BETA SHALL PASS BEFORE IT'S TIME!
Splendid isolation
HARNESS BRAIN POWER
RELEASE EARLY
RELEASE OFTEN
it's a CULTURE IDEOLOGY METHODOLOGY -@cowrosa
EGOLESSS
OPEN
LISTEN
PROBLEM FINDING
TOO MANY PROGRAMMERS WORK ON PROJECTS THEY NEITHER NEED NOR WANT
@opencontent
REUSE WHEN POSSIBLE
EVERY GOOD PIECE OF WORK STARTS BY SCRATCHING A PERSONAL ITCH
WITH ENOUGH EYEBALLS ALL BUGS ARE SHALLOW
@giuliaforsythe
#oped12

Bell Labs: A Hive of Invention

A selection of its most important innovations in the decades leading up to the breakup of its parent company, AT&T, in 1984, and how they helped lead to some of the latest technologies.



1940s

) First long-distance computing Remote eration of a computer, Bell Labs in New York, by a teletypewriter in New Hampshire.

1947 The transistor A landmark invention. Replaced vacuum tubes and mechanical relays; transformed electronics.

46 First commercial obile phone service most, three subscrib- per city could make lls at one time; each er's phone apparatus weighed almost 80 pounds.

1948 Information theory Calculates maximum capacity for any communications system and shows how to send digital messages essentially error-free. Enabled data compression and cryptography.

1947 Cellular lephone technology l Labs paper was the propose a network of interlocking cell sites g users as they move, g their calls from one te to another without pping the connection.

The Murray Hill, N.J., buildings opened in 1941.

1950s

1951 Direct-distance dialing No operator necessary for long-distance calls.

1954 Solar cells First use of the sun's energy to create a practical level of electricity.

1956 First transatlantic telephone cable Designed and implemented by Bell Labs; could carry up to 36 simultaneous calls.

1957 Digitized music First demonstrations of digitized and computer-synthesized music.

1958 The laser "Light Amplification by Stimulated Emission of Radiation" was described in a Bell Labs paper. It is crucial for communications, surgical and DVD technologies.

1960s

1962 Digital transmission, switching First digital transmission of multiple voice signals.

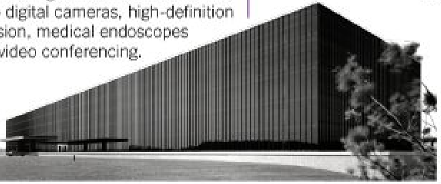
1960-62 First communications satellites Echo is first to reflect a voice signal from coast to coast; Telstar I shows an orbiting relay can amplify and resend multiple phone and TV transmissions.

1962 Paging system Bellboy pager is introduced at the Seattle World's Fair.

1963 Touch-tone telephone Enables voice mail and call centers.

1965 Evidence of the Big Bang Discovery of cosmic background radiation from beyond the Milky Way.

1969 Charge-coupled device A solid-state chip that transforms patterns of light into information. Vital to digital cameras, high-definition television, medical endoscopes and video conferencing.



Bell Labs opened in Holmdel, N.J., in 1962. It was vacated in 2007.

1970s and '80s

1969-72 UNIX operating system and C programming language Makes large-scale networking of varied computing systems, and the Internet, practical.

1976 Fiber-optic network The first test of Bell Labs' experimental lightwave communication system begins in Atlanta. Information is carried by pulses of light.

1978 First commercial cellular network Installed by Bell Labs in Chicago.

1979 Digital signal processor An essential component of cellphones, modems, PCs and video game systems.

1980 Digital cellular phone Better sound quality, greater channel capacity, lower cost.

1982 Fractional quantum hall effect Discovery of a new state of subatomic matter that wins the Nobel Prize.

LEFT AND CENTER PHOTOS COURTESY OF ALCATEL-LUCENT USA INC. AND THE AT&T ARCHIVES ANI

Bell Labs: A Hive of Invention

A selection of its most important innovations in the decades leading up to the breakup of its parent company, AT&T, in 1984, and how they helped lead to some of the latest technologies.

1940s

) **First long-distance computing** Remote eration of a computer, Bell Labs in New York, by a teletypewriter in New Hampshire.

1947 The transistor A landmark invention. Replaced vacuum tubes and mechanical relays; transformed electronics.

46 First commercial nobile phone service most, three subscrib- s per city could make lls at one time; each er's phone apparatus weighed almost 80 pounds.

1948 Information theory Calculates maximum capacity for any communications system and shows how to send digital messages essentially error-free. Enabled data compression and cryptography.

1947 Cellular elephone technology l Labs paper was the propose a network of interlocking cell sites g users as they move, g their calls from one te to another without pping the connection.

The Murray Hill, N.J., buildings opened in 1941.

1950s

1951 Direct-distance dialing No operator necessary for long-distance calls.

1954 Solar cells First use of the sun's energy to create a practical level of electricity.

1956 First transatlantic telephone cable Designed and implemented by Bell Labs; could carry up to 36 simultaneous calls.

1957 Digitized music First demonstrations of digitized and computer-synthesized music.

1958 The laser "Light Amplification by Stimulated Emission of Radiation" was described in a Bell Labs paper. It is crucial for communications, surgical and DVD technologies.

1960s

1962 Digital transmission, switching First digital transmission of multiple voice signals.

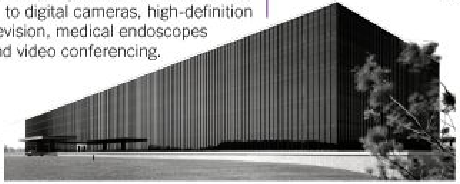
1960-62 First communications satellites Echo is first to reflect a voice signal from coast to coast; Telstar I shows an orbiting relay can amplify and resend multiple phone and TV transmissions.

1962 Paging system Bellboy pager is introduced at the Seattle World's Fair.

1963 Touch-tone telephone Enables voice mail and call centers.

1965 Evidence of the Big Bang Discovery of cosmic background radiation from beyond the Milky Way.

1969 Charge-coupled device A solid-state chip that transforms patterns of light into information. Vital to digital cameras, high-definition television, medical endoscopes and video conferencing.



Bell Labs opened in Holmdel, N.J., in 1962. It was vacated in 2007.

1970s and '80s

1969-72 UNIX operating system and C programming language Makes large-scale networking of varied computing systems, and the Internet, practical.

1976 Fiber-optic network The first test of Bell Labs' experimental lightwave communication system begins in Atlanta. Information is carried by pulses of light.

1978 First commercial cellular network Installed by Bell Labs in Chicago.

1979 Digital signal processor An essential component of cellphones, modems, PCs and video game systems.

1980 Digital cellular phone Better sound quality, greater channel capacity, lower cost.

1982 Fractional quantum hall effect Discovery of a new state of subatomic matter that wins the Nobel Prize.

LEFT AND CENTER PHOTOS COURTESY OF ALCATEL-LUCENT USA INC. AND THE AT&T ARCHIVES ANI

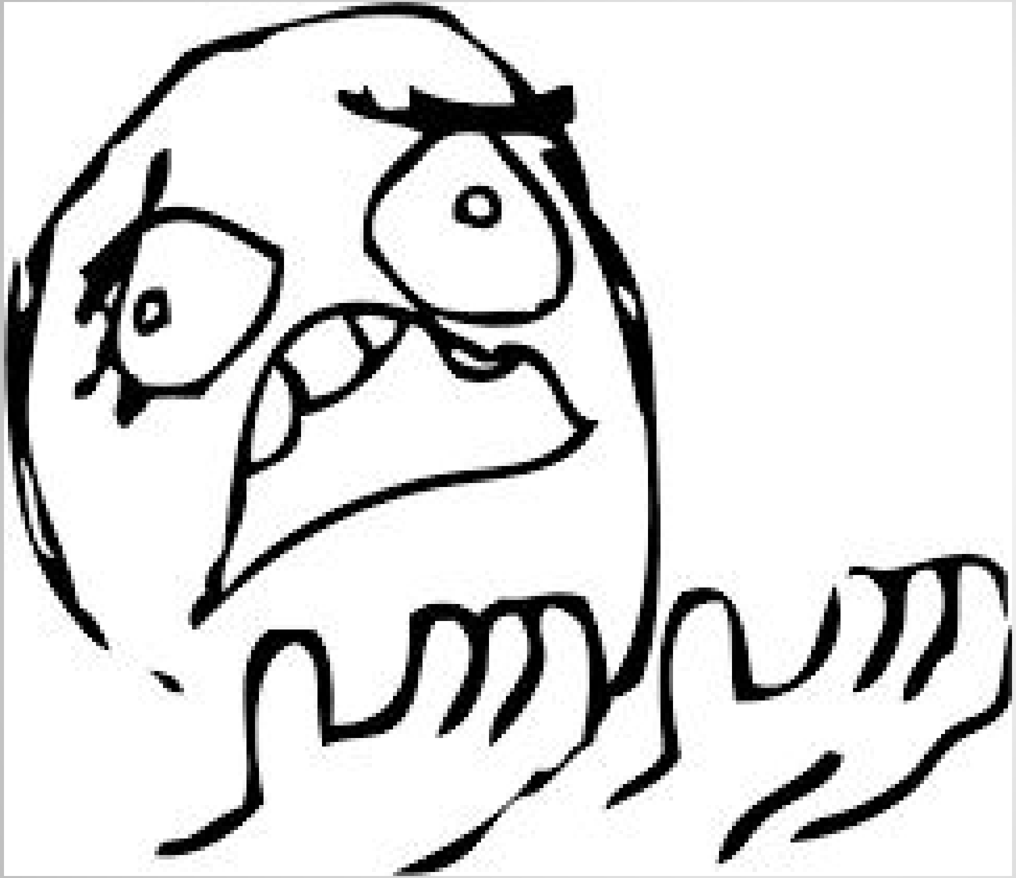
GO

Golang
IS COOL
IN 2017

- too simple / lack of syntactic sugar
- no generics
- bad dependency management
- stuck in 70/80's
- error handling

- no unused imports
- too opinionated
- too verbose
- no ternary operator
- no macros or templates

https://github.com/ksimka/go-is-not-good



WHERE
WE'RE
GOING,
WE DON'T
NEED
ROADS

“Go programming language was conceived as an answer to some of the problems we were seeing developing software infrastructure at Google. The computing landscape today is almost unrelated to the environment in which the languages being used, mostly C++, Java, and Python, had been created. The problems introduced by **multicore processors**, **networked systems**, **massive computation clusters**, and the **web programming model** were being worked around rather than addressed head-on.

“Go programming language was conceived as an answer to some of the problems we were seeing developing software infrastructure at Google. The computing landscape today is almost unrelated to the environment in which the languages being used, mostly C++, Java, and Python, had been created. The problems introduced by **multicore processors**, **networked systems**, **massive computation clusters**, and the **web programming model** were being worked around rather than addressed head-on.

“Go programming language was conceived as an answer to some of the problems we were seeing developing software infrastructure at Google. The computing landscape today is almost unrelated to the environment in which the languages being used, mostly C++, Java, and Python, had been created. The problems introduced by **multicore processors**, **networked systems**, **massive computation clusters**, and the **web programming model** were being worked around rather than addressed head-on.

“Go programming language was conceived as an answer to some of the problems we were seeing developing software infrastructure at Google. The computing landscape today is almost unrelated to the environment in which the languages being used, mostly C++, Java, and Python, had been created. The problems introduced by **multicore processors**, **networked systems**, **massive computation clusters**, and the **web programming model** were being worked around rather than addressed head-on.

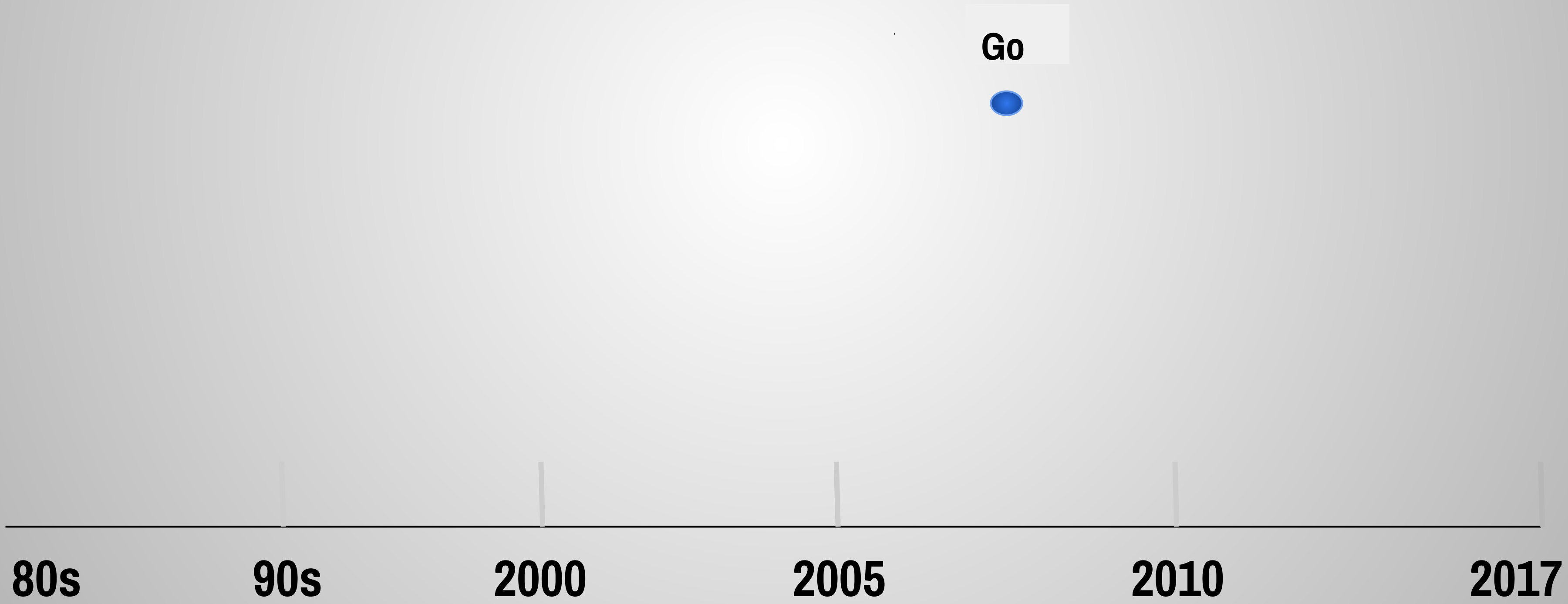
“Go programming language was conceived as an answer to some of the problems we were seeing developing software infrastructure at Google. The computing landscape today is almost unrelated to the environment in which the languages being used, mostly C++, Java, and Python, had been created. The problems introduced by **multicore processors**, **networked systems**, **massive computation clusters**, and the **web programming model** were being worked around rather than addressed head-on.

Moreover, the scale has changed: today’s server programs comprise tens of millions of lines of code, are worked on by hundreds or even thousands of programmers, and are updated literally every day. To make matters worse, build time have stretched to many minutes, even hours, even languages,on large compilation clusters,” Moreover, the scale has changed: today’s server programs comprise tens of millions of lines of code, are worked on by **hundreds or even thousands of programmers**, and are updated literally every day. To make matters worse, build time has stretched to many minutes, even hours, on large compilation clusters”

Moreover, the scale has changed: today's server programs comprise tens of millions of lines of code, are worked on by **hundreds or even thousands of programmers**, and are updated literally every day. To make matters worse, build time has stretched to many minutes, even hours, on **large compilation clusters**"

- multicore processors
- networked systems
- massive computation clusters
- web programming model

- hundreds or even thousands of programmers
- large compilation clusters



Go

80s

90s

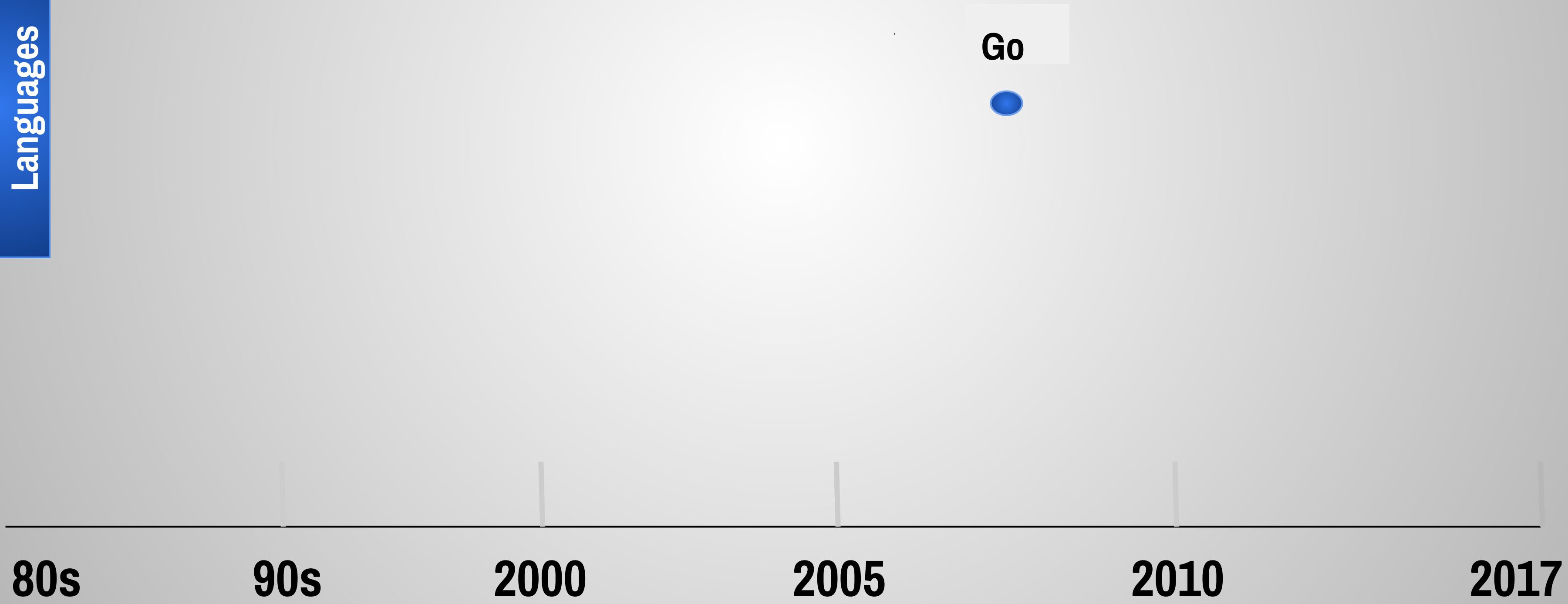
2000

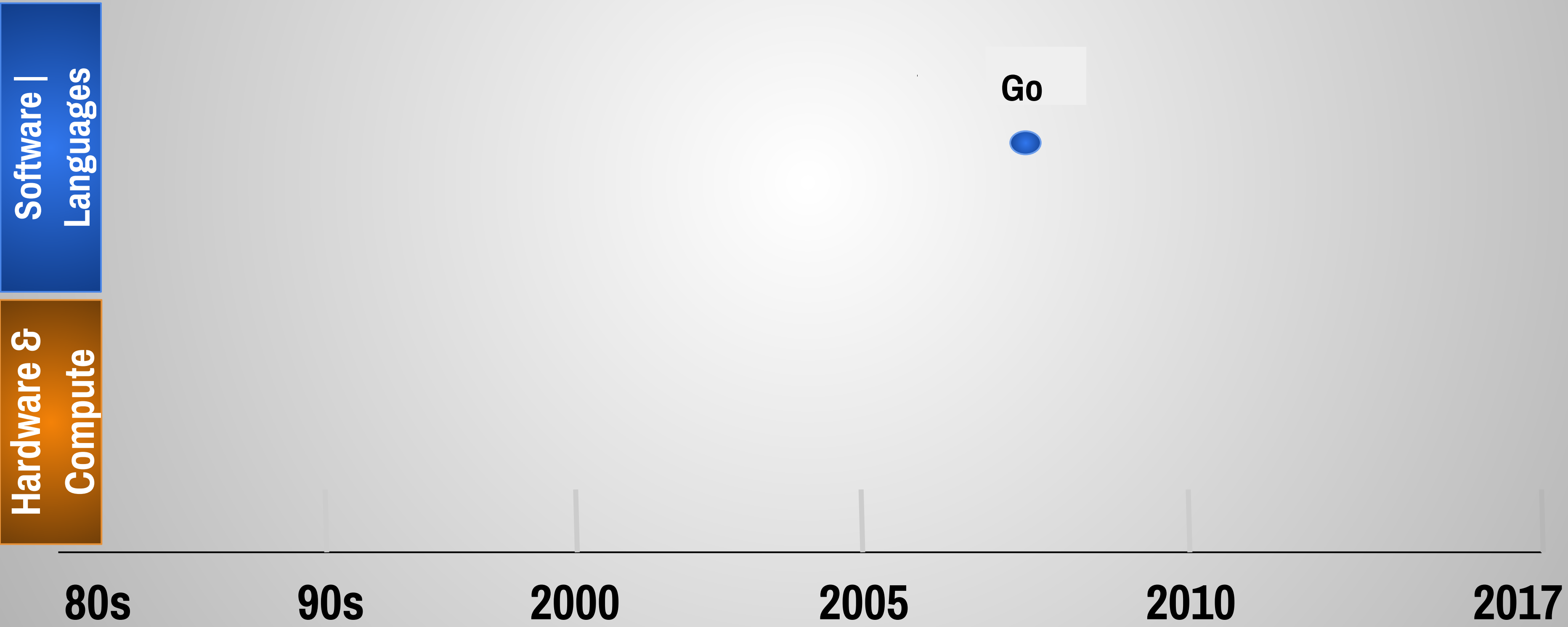
2005

2010

2017

Software | Languages





Software | Languages

Hardware & Compute

Go

80s

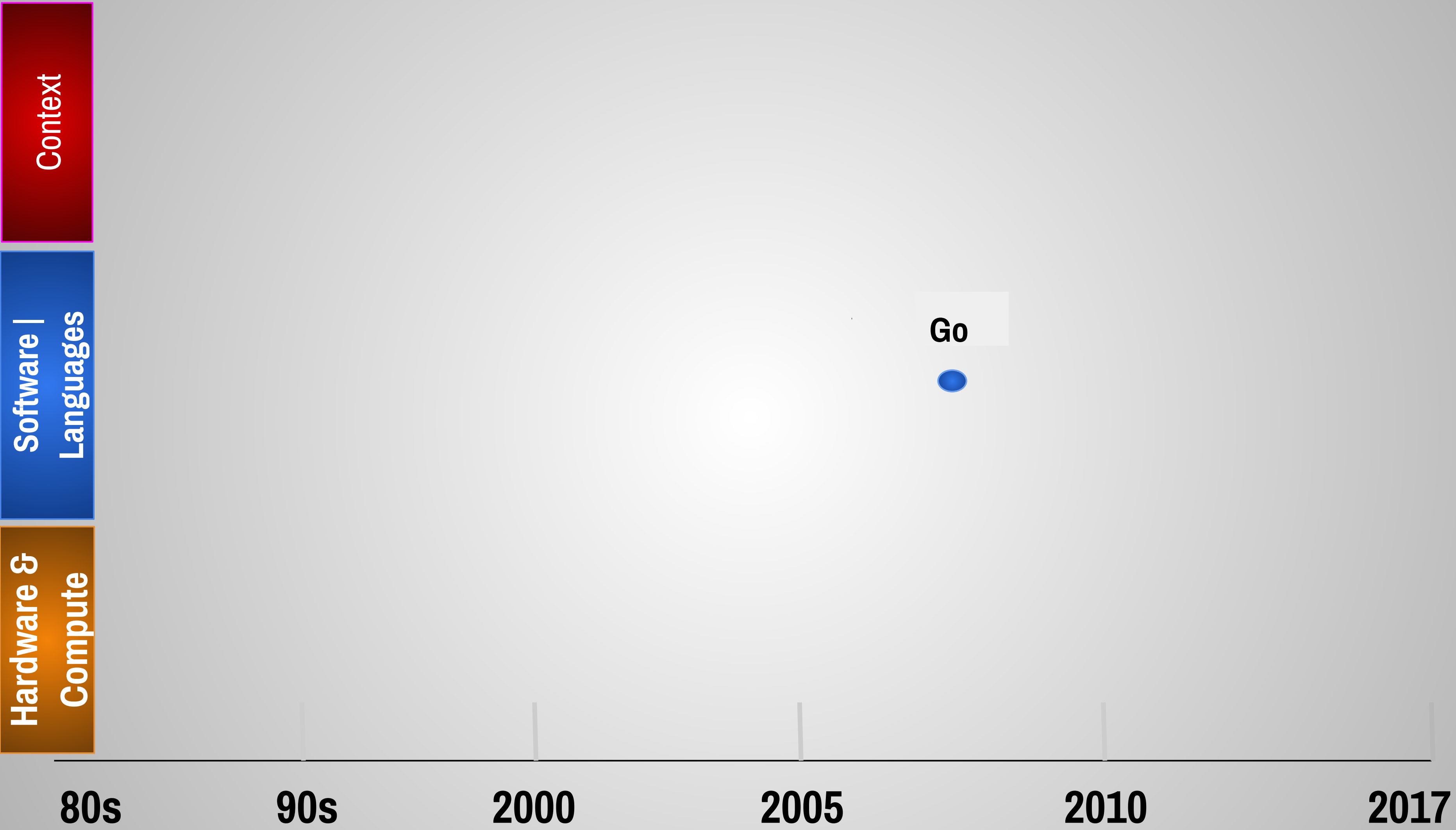
90s

2000

2005

2010

2017



Context

Software | Languages

Hardware & Compute

Go

80s

90s

2000

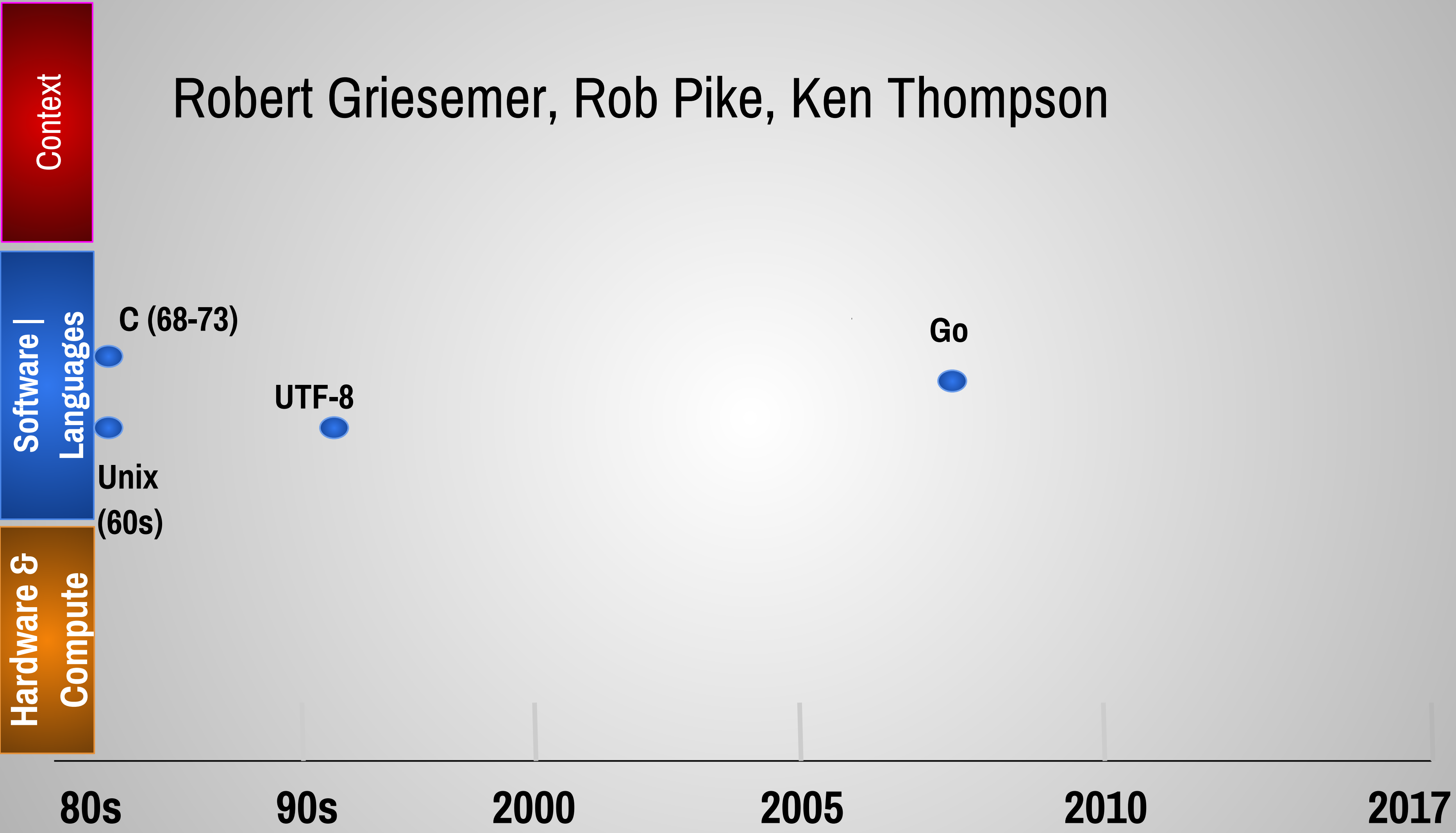
2005

2010

2017

Robert Griesemer, Rob Pike and Ken Thompson

Robert Griesemer, Rob Pike, Ken Thompson



Context

Software | Languages

Hardware & Compute

C (68-73)

Unix (60s)

UTF-8

Go

80s

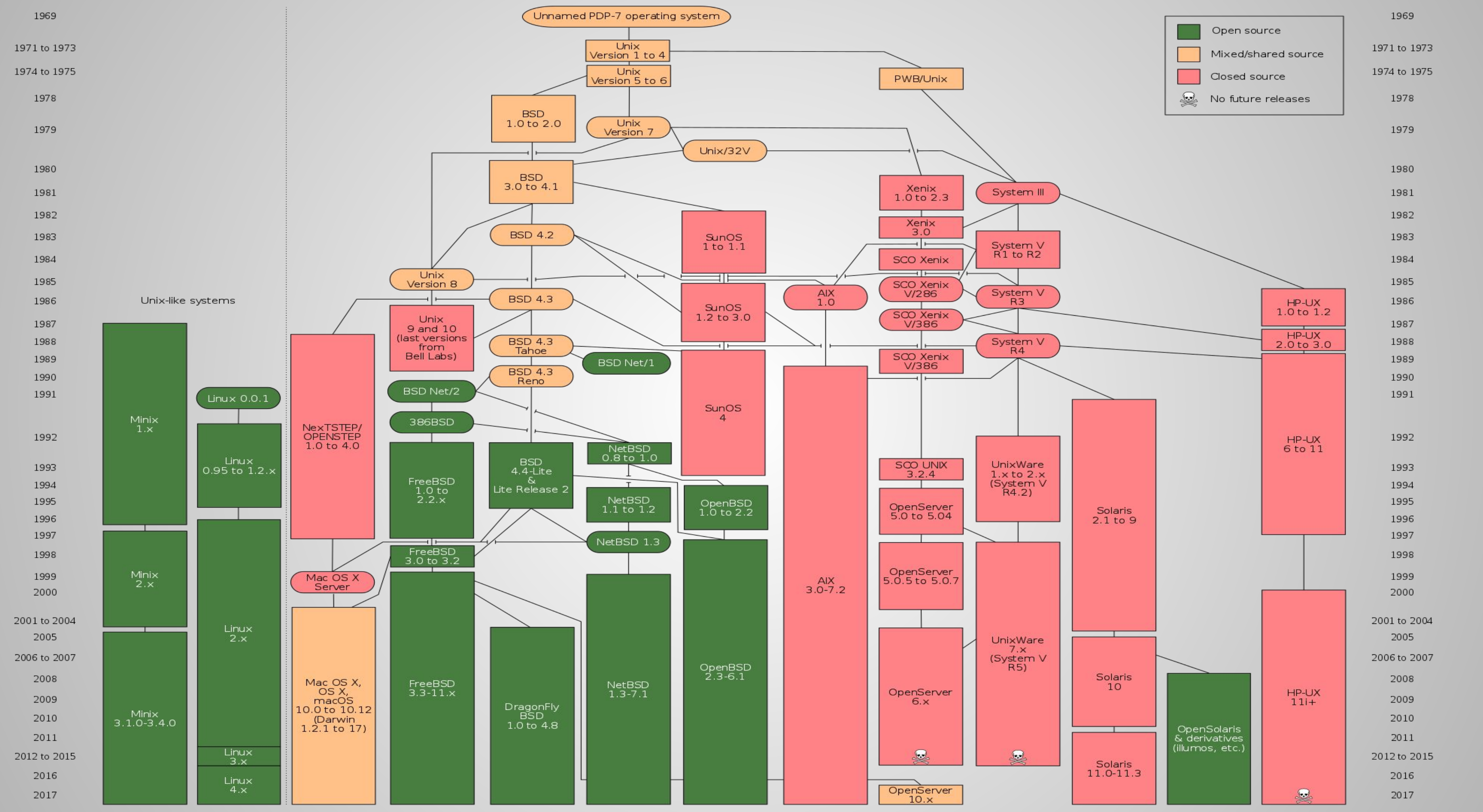
90s

2000

2005

2010

2017



🟩 Open source

🟧 Mixed/shared source

🟥 Closed source

☠ No future releases

Unix-like systems

1969
1971 to 1973
1974 to 1975
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001 to 2004
2005
2006 to 2007
2008
2009
2010
2011
2012 to 2015
2016
2017

Unnamed PDP-7 operating system

Unix Version 1 to 4

Unix Version 5 to 6

PWB/Unix

BSD 1.0 to 2.0

Unix Version 7

Unix/32V

BSD 3.0 to 4.1

Xenix 1.0 to 2.3

System III

Xenix 3.0

SunOS 1 to 1.1

BSD 4.2

System V R1 to R2

SCO Xenix

Unix Version 8

AIX 1.0

SCO Xenix V/286

System V R3

BSD 4.3

HP-UX 1.0 to 1.2

SunOS 1.2 to 3.0

Unix 9 and 10 (last versions from Bell Labs)

SCO Xenix V/386

HP-UX 2.0 to 3.0

BSD 4.3 Tahoe

System V R4

BSD Net/1

SCO Xenix V/386

BSD 4.3 Reno

Minix 1.x

Linux 0.0.1

BSD Net/2

NexTSTEP/ OPENSTEP 1.0 to 4.0

SunOS 4

386BSD

HP-UX 6 to 11

Linux 0.95 to 1.2.x

NetBSD 0.8 to 1.0

SCO UNIX 3.2.4

FreeBSD 1.0 to 2.2.x

BSD 4.4-Lite & Lite Release 2

UnixWare 1.x to 2.x (System V R4.2)

NetBSD 1.1 to 1.2

OpenBSD 1.0 to 2.2

OpenServer 5.0 to 5.04

Solaris 2.1 to 9

NetBSD 1.3

FreeBSD 3.0 to 3.2

Minix 2.x

OpenServer 5.0.5 to 5.0.7

Mac OS X Server

Linux 2.x

AIX 3.0-7.2

Minix 3.1.0-3.4.0

Mac OS X, OS X, macOS 10.0 to 10.12 (Darwin 1.2.1 to 17)

FreeBSD 3.3-11.x

DragonFly BSD 1.0 to 4.8

NetBSD 1.3-7.1

OpenBSD 2.3-6.1

OpenServer 6.x

UnixWare 7.x (System V R5)

Solaris 10

OpenSolaris & derivatives (illumos, etc.)

HP-UX 11i+

Linux 3.x

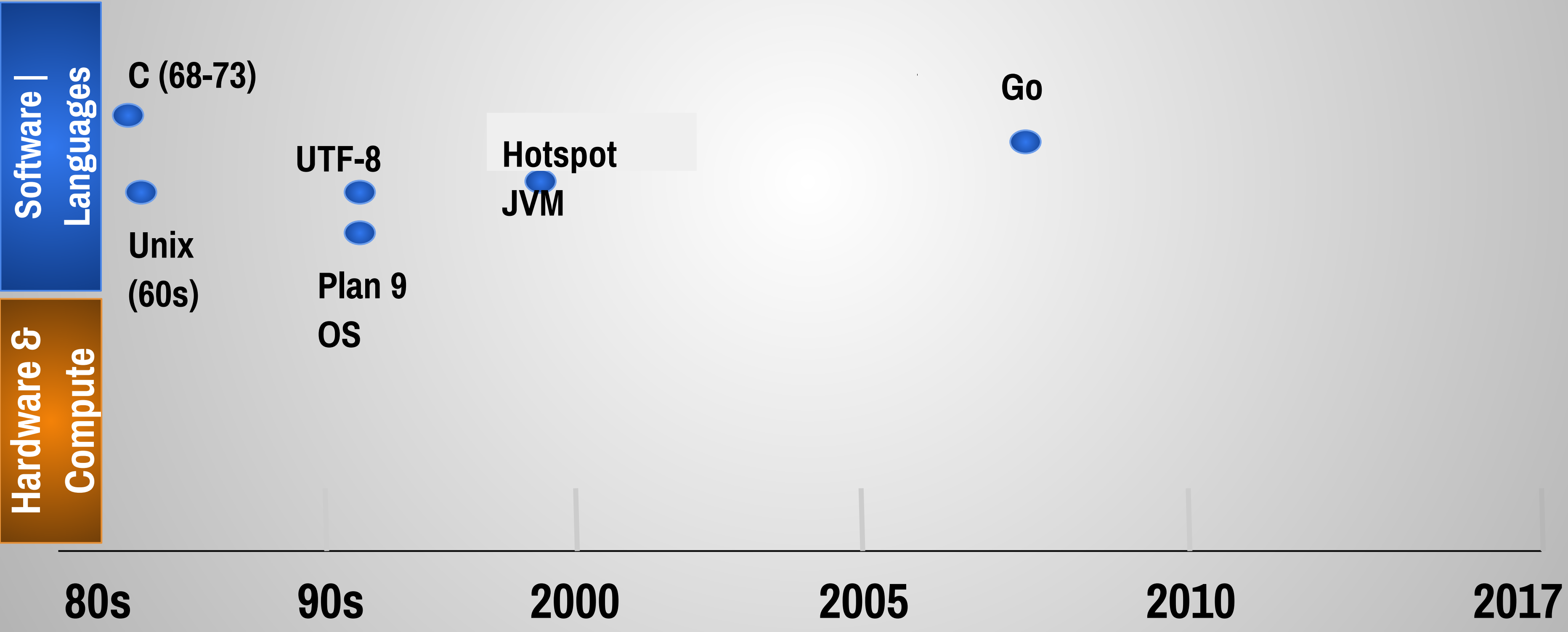
Solaris 11.0-11.3

Linux 4.x

OpenServer 10.x

Context

Robert Griesemer, Rob Pike, Ken Thompson



Software | Languages

Hardware & Compute

C (68-73)

Unix (60s)

UTF-8

Plan 9 OS

Hotspot JVM

Go

80s

90s

2000

2005

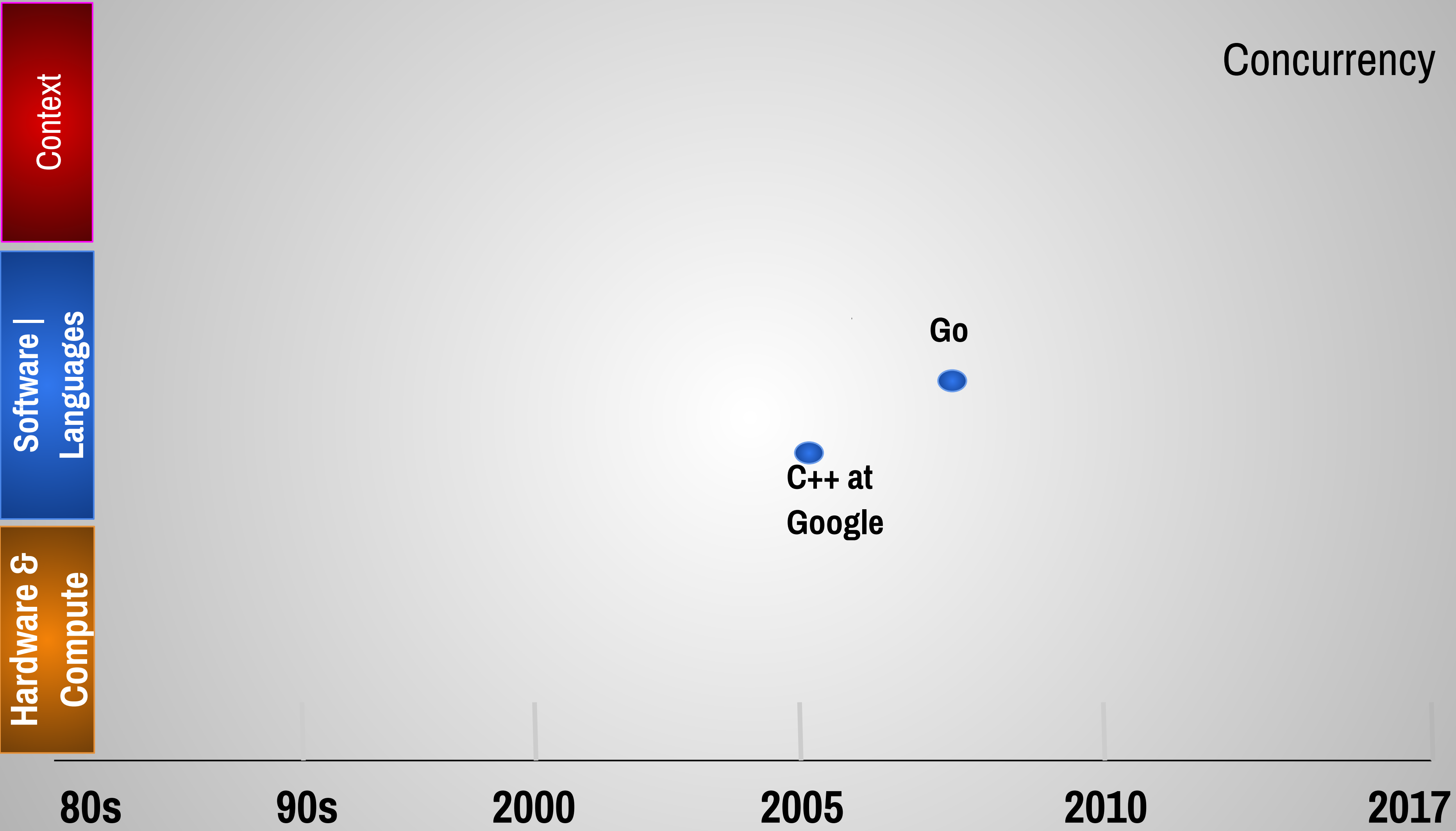
2010

2017

Go’s 21st Century Characteristics

- Concurrency
- Distributed Systems
- Garbage Collection
- Memory Locality
- Readability

Concurrency



Context

Software | Languages

Hardware & Compute

Go

C++ at Google

80s

90s

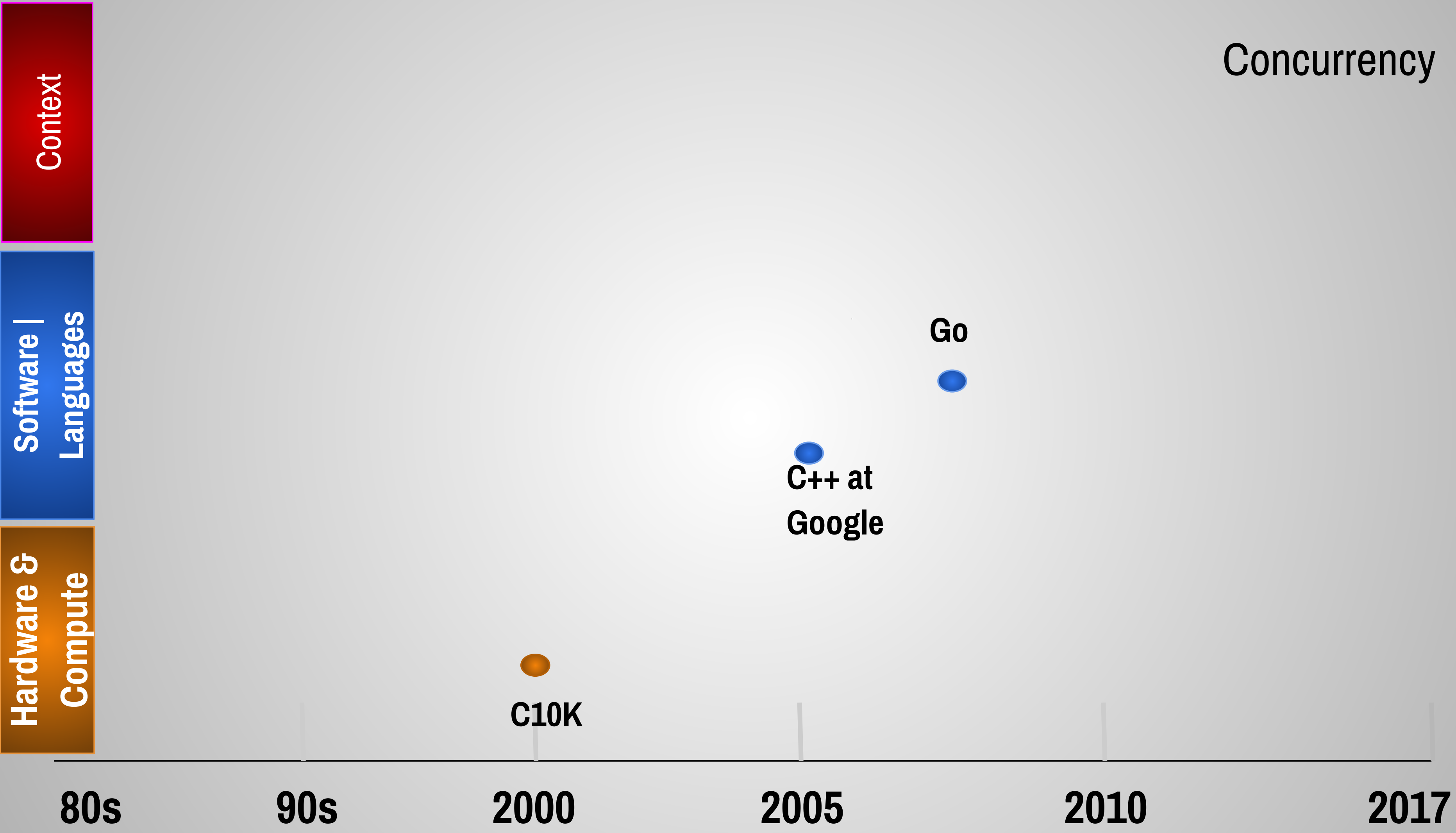
2000

2005

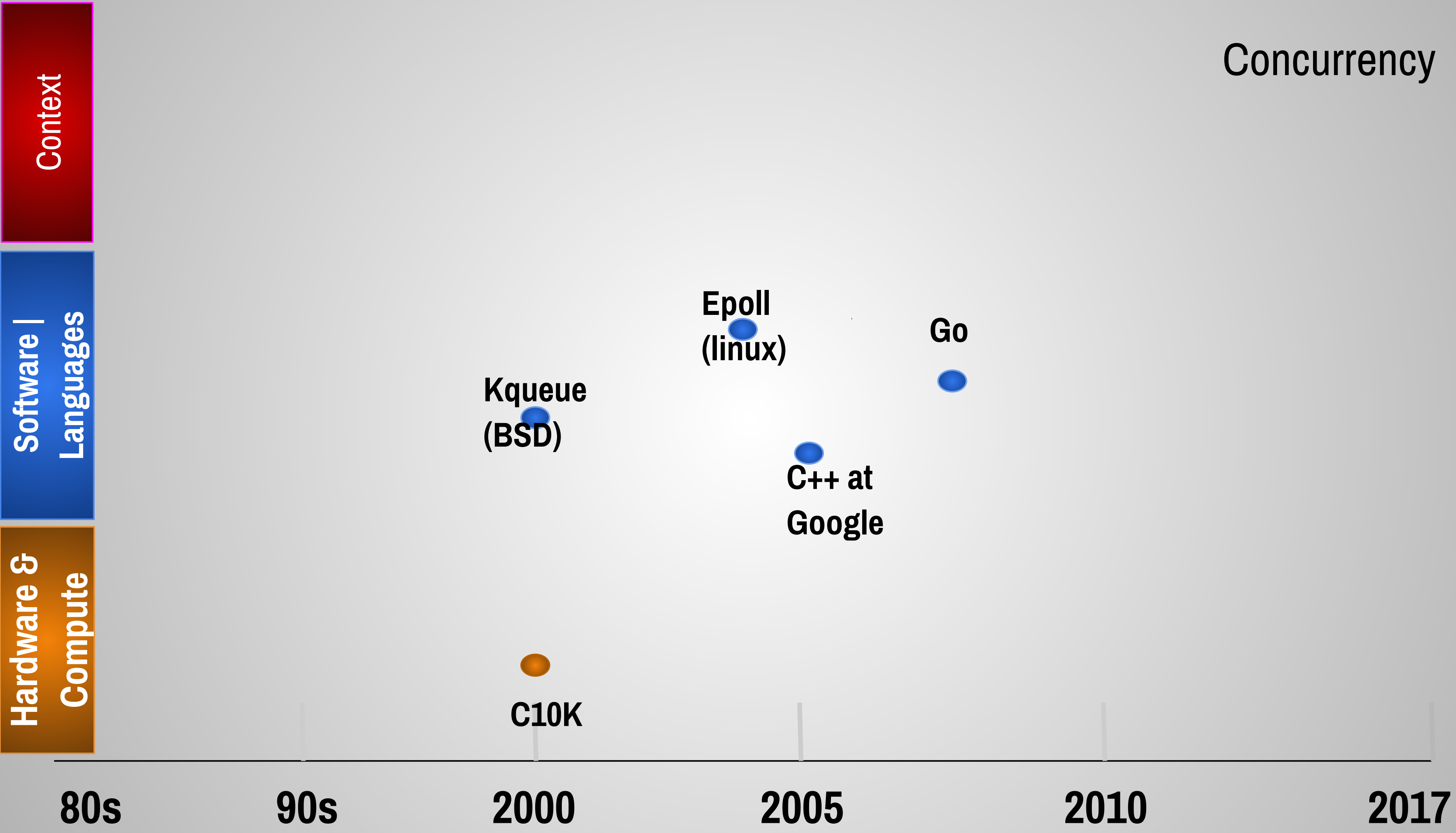
2010

2017

Concurrency



Concurrency



Context

Software | Languages

Hardware & Compute

Kqueue (BSD)

Epoll (linux)

C++ at Google

Go

C10K

80s

90s

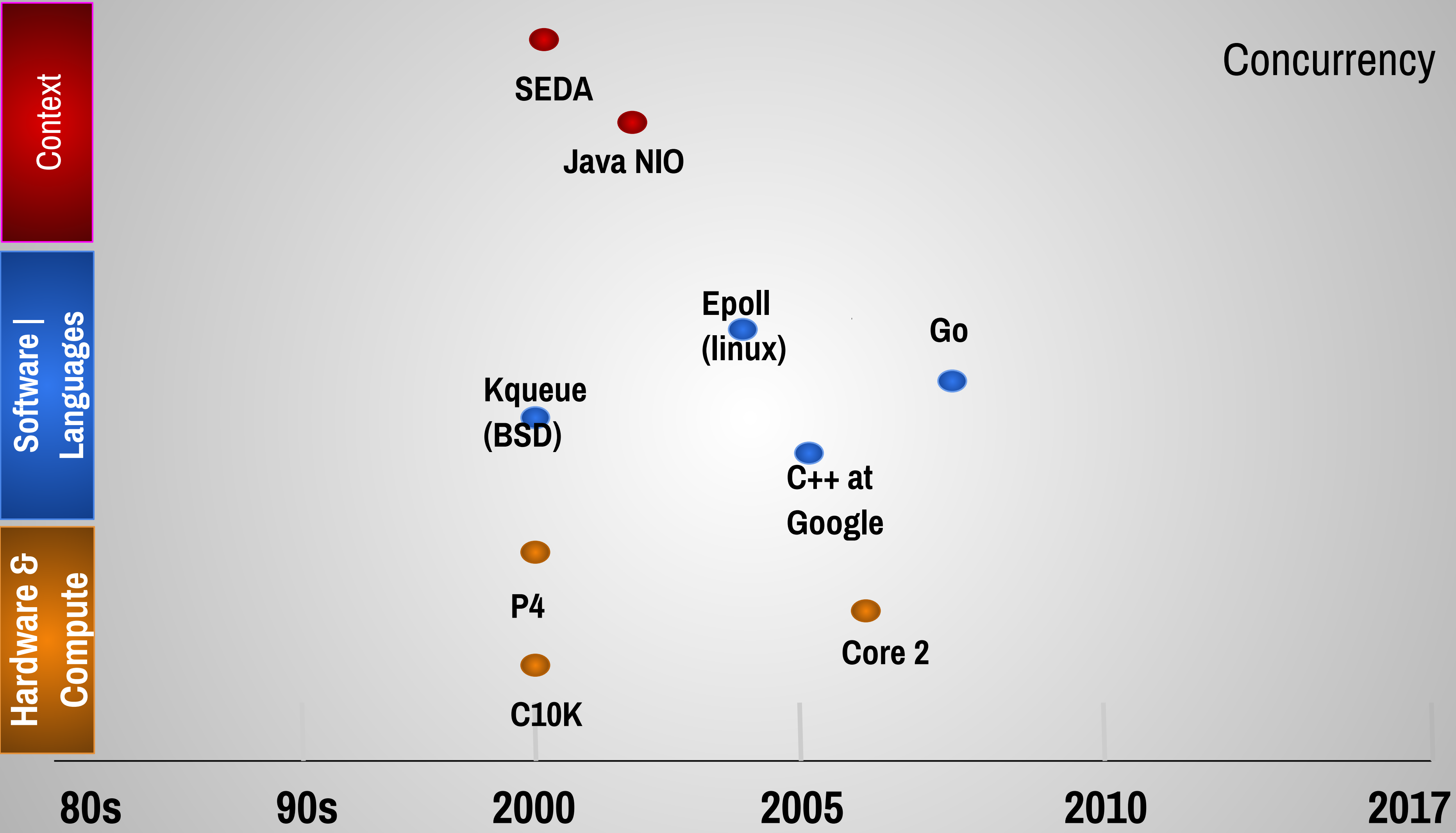
2000

2005

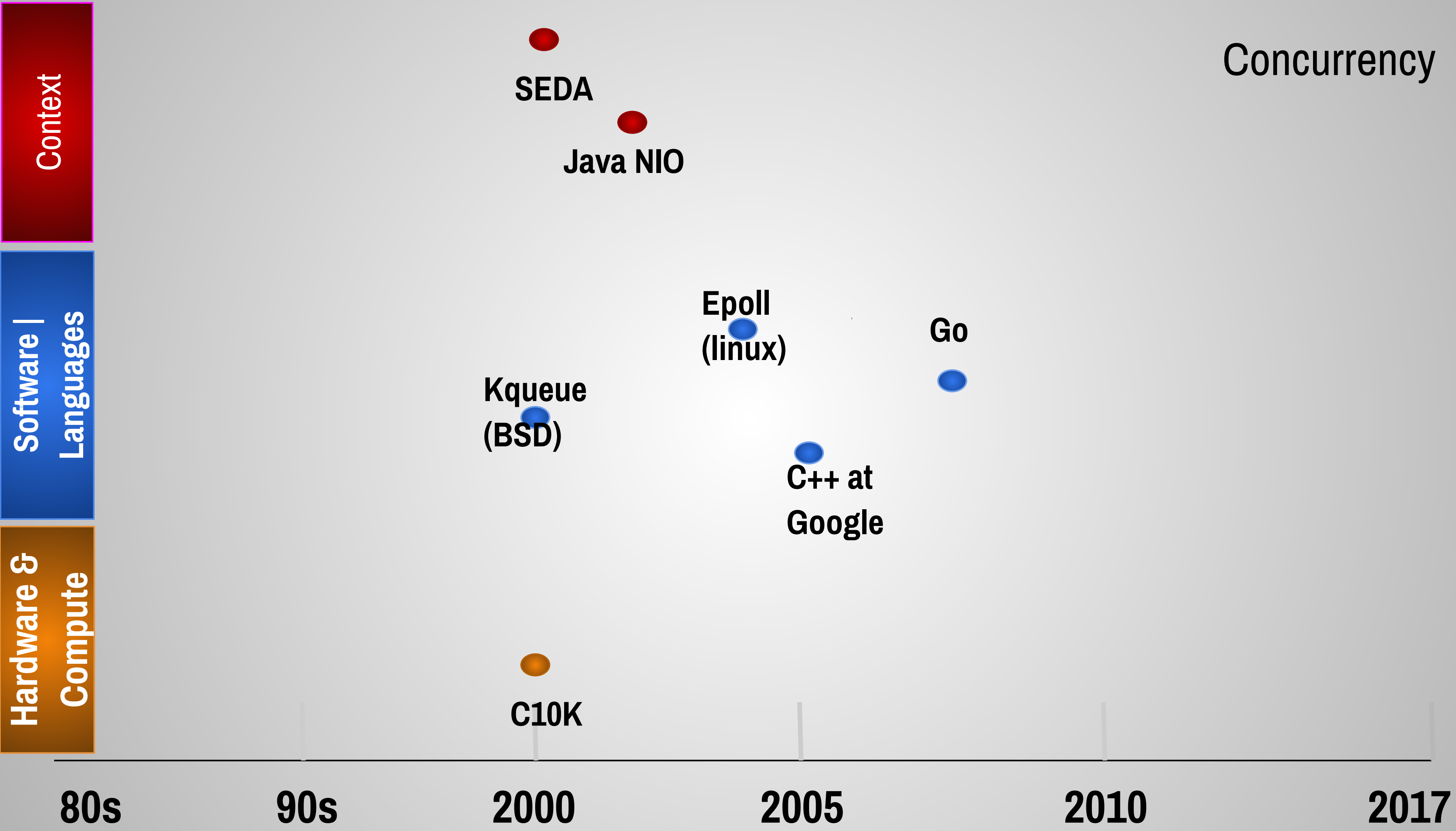
2010

2017

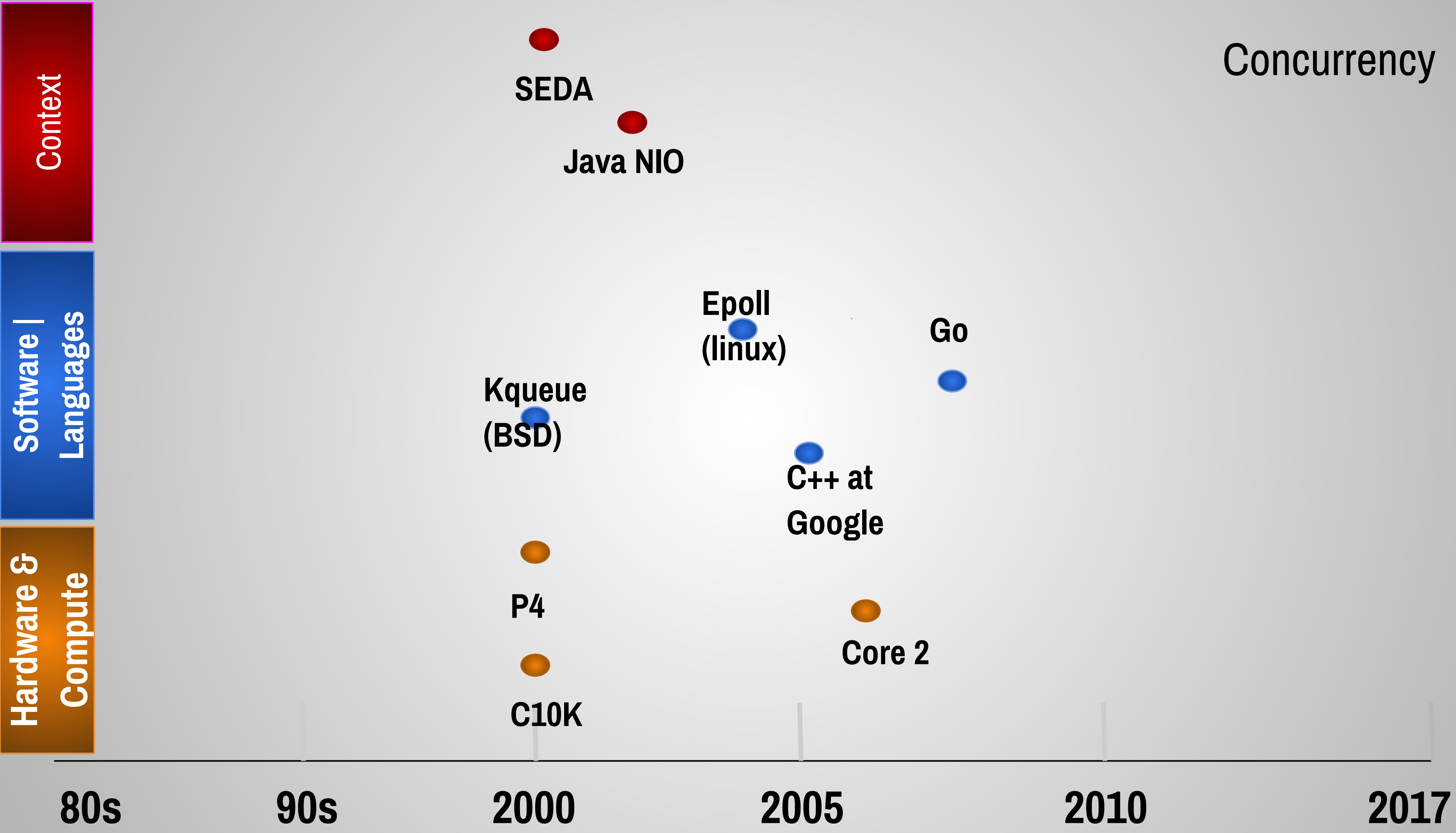
Concurrency



Concurrency



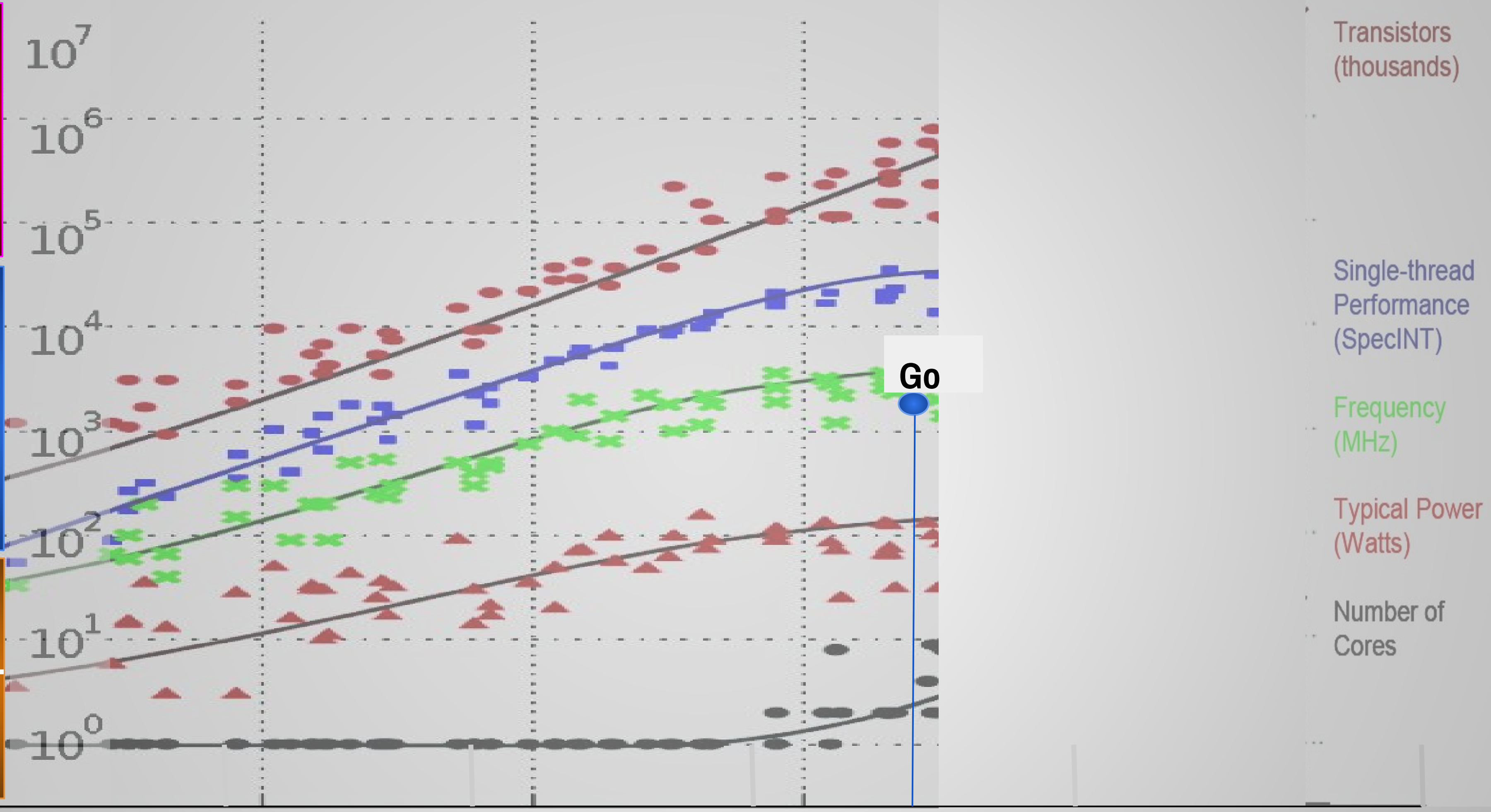
Concurrency



Context

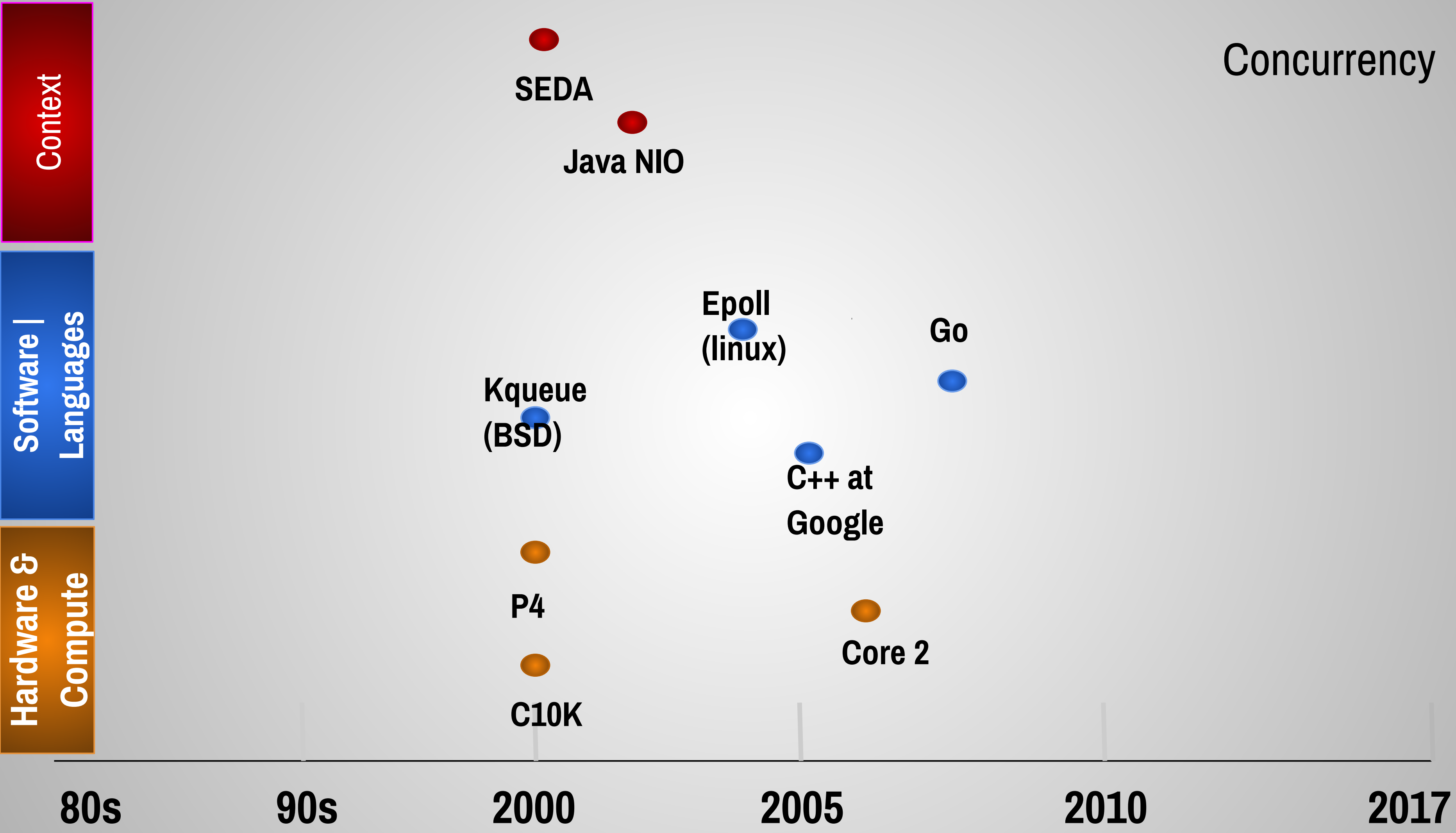
Software | Languages

Hardware & Compute

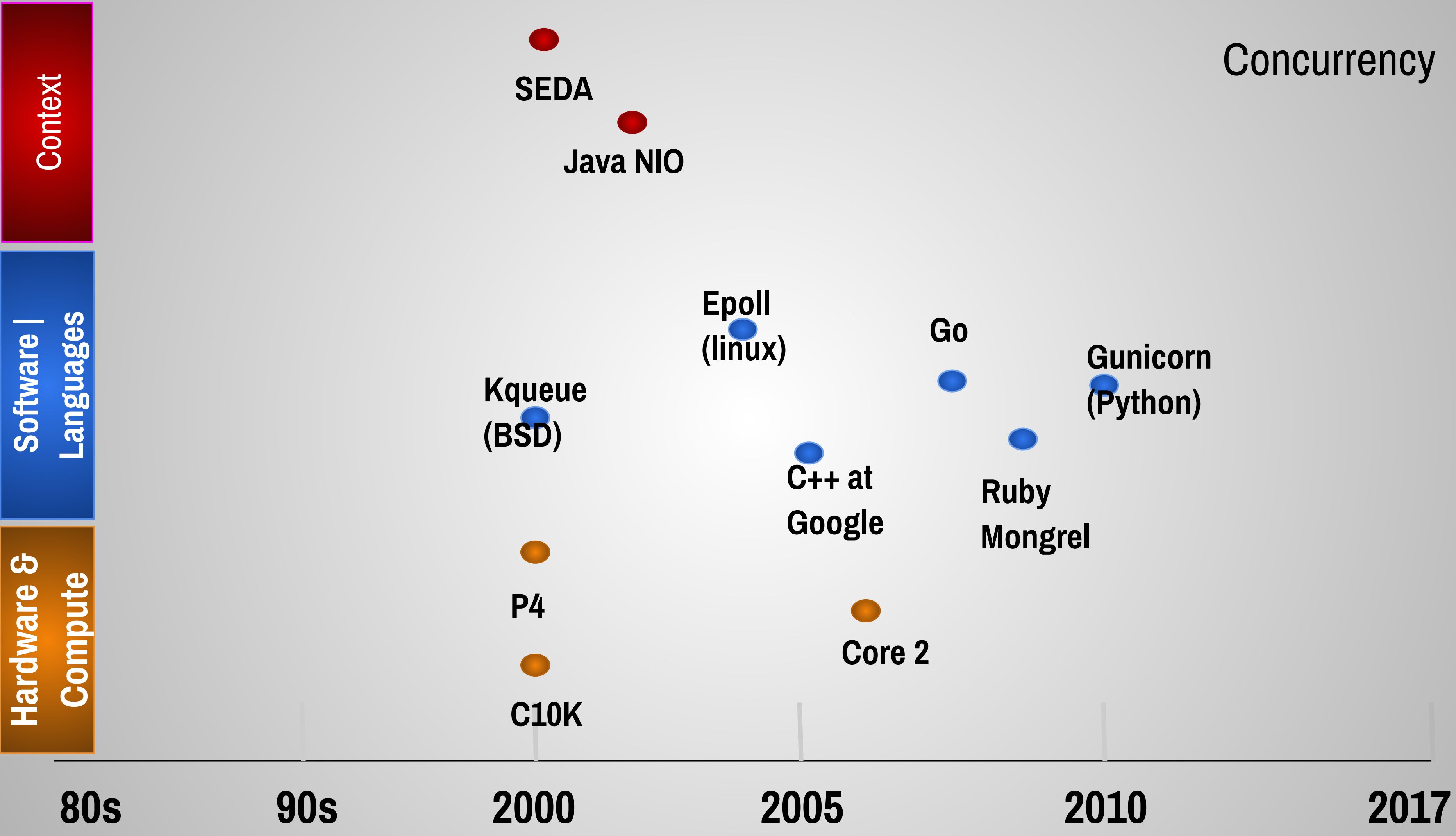


80s 90s 2000 2005 2010 2017

Concurrency



Concurrency



Context

Software | Languages

Hardware & Compute

SEDA

Java NIO

Kqueue (BSD)

Epoll (linux)

C++ at Google

Go

Ruby Mongrel

Gunicorn (Python)

P4

C10K

Core 2

80s

90s

2000

2005

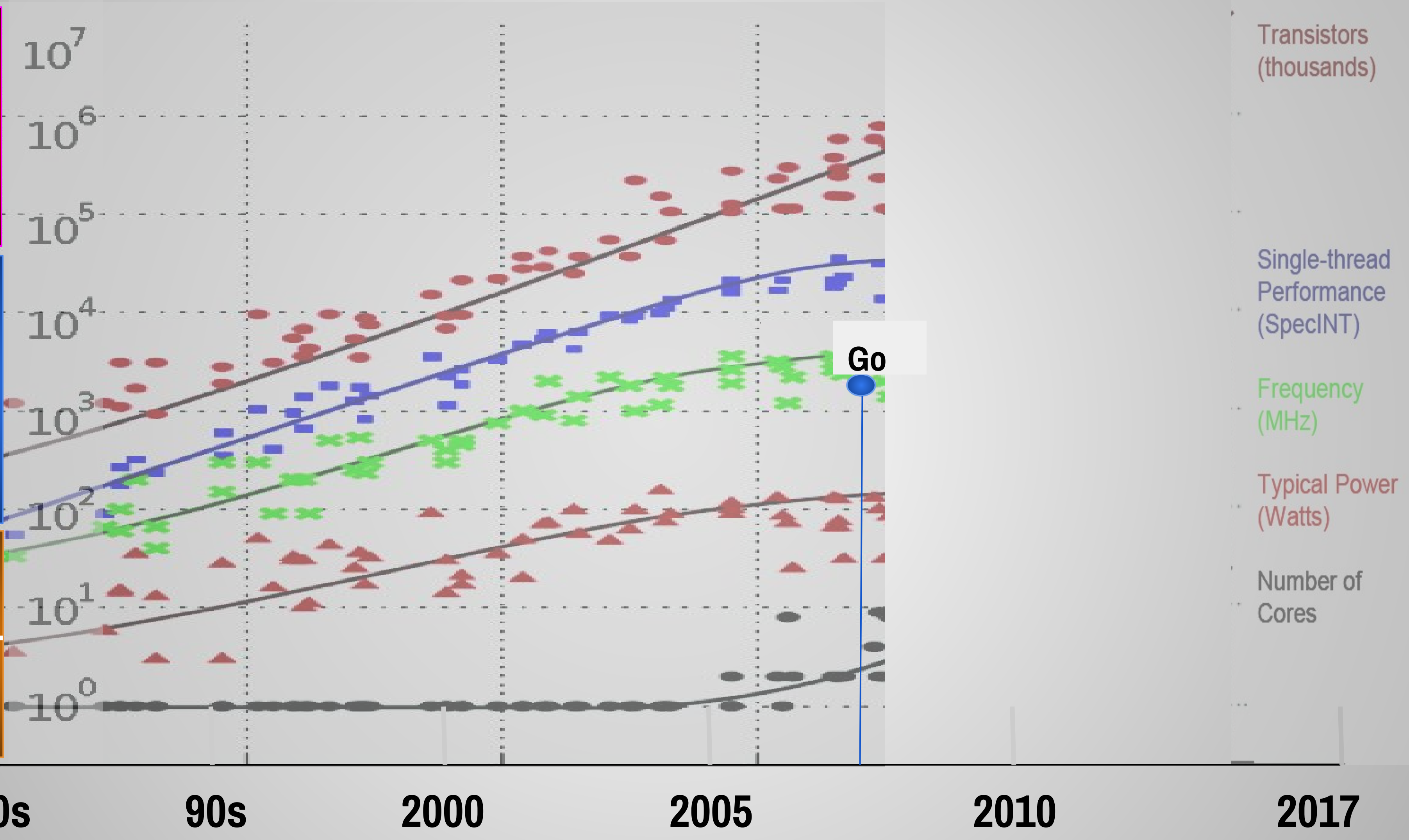
2010

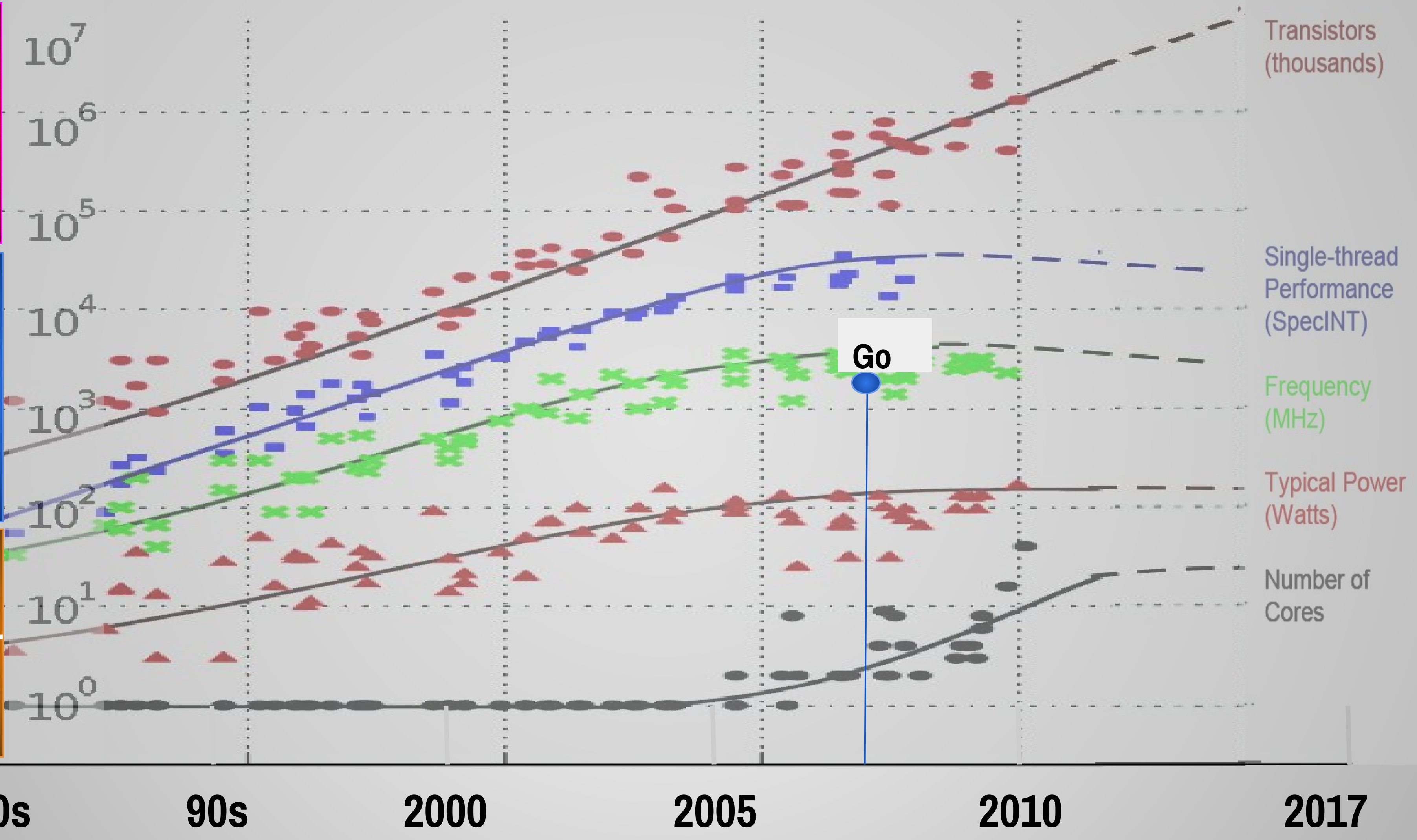
2017

Context

Software | Languages

Hardware & Compute





Context

Software | Languages

Hardware & Compute

Events, Threads and Goroutines

Nginx - event loop plus state machine model

Events, Threads and Goroutines

Nginx - event loop plus state machine model

App logic

Events, Threads and Goroutines

Nginx - event loop plus state machine model

~~App logic~~

Events, Threads and Goroutines

Nginx - event loop plus state machine model

App

Events, Threads and Goroutines

Nginx - event loop plus state machine model

Events, Threads and Goroutines

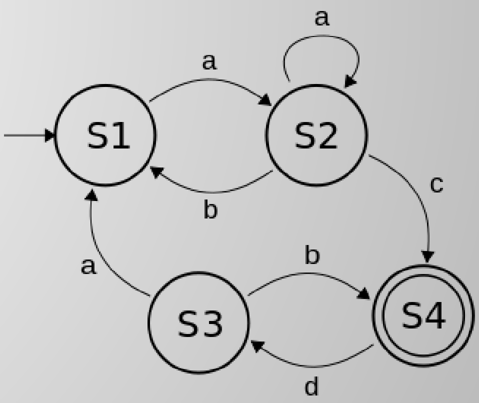
Nginx - event loop plus state machine model

Events, Threads and Goroutines

Nginx - event loop plus state machine model

NGINX

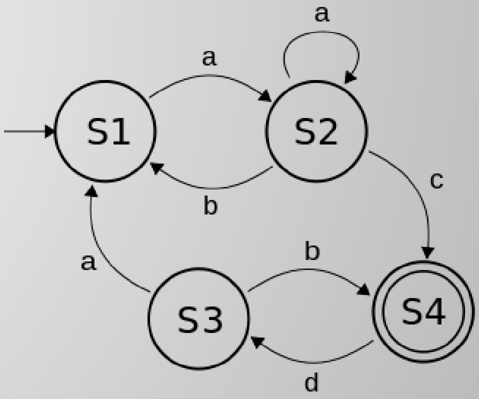
App



Events, Threads and Goroutines

Nginx - event loop plus state machine model

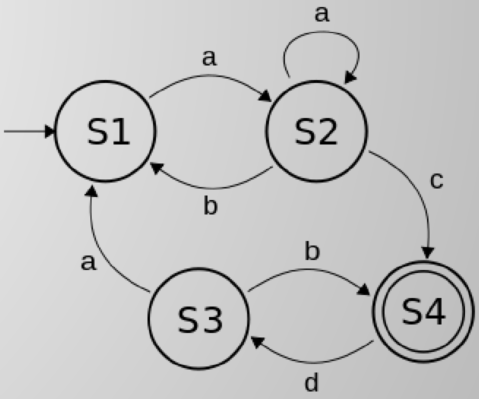
App



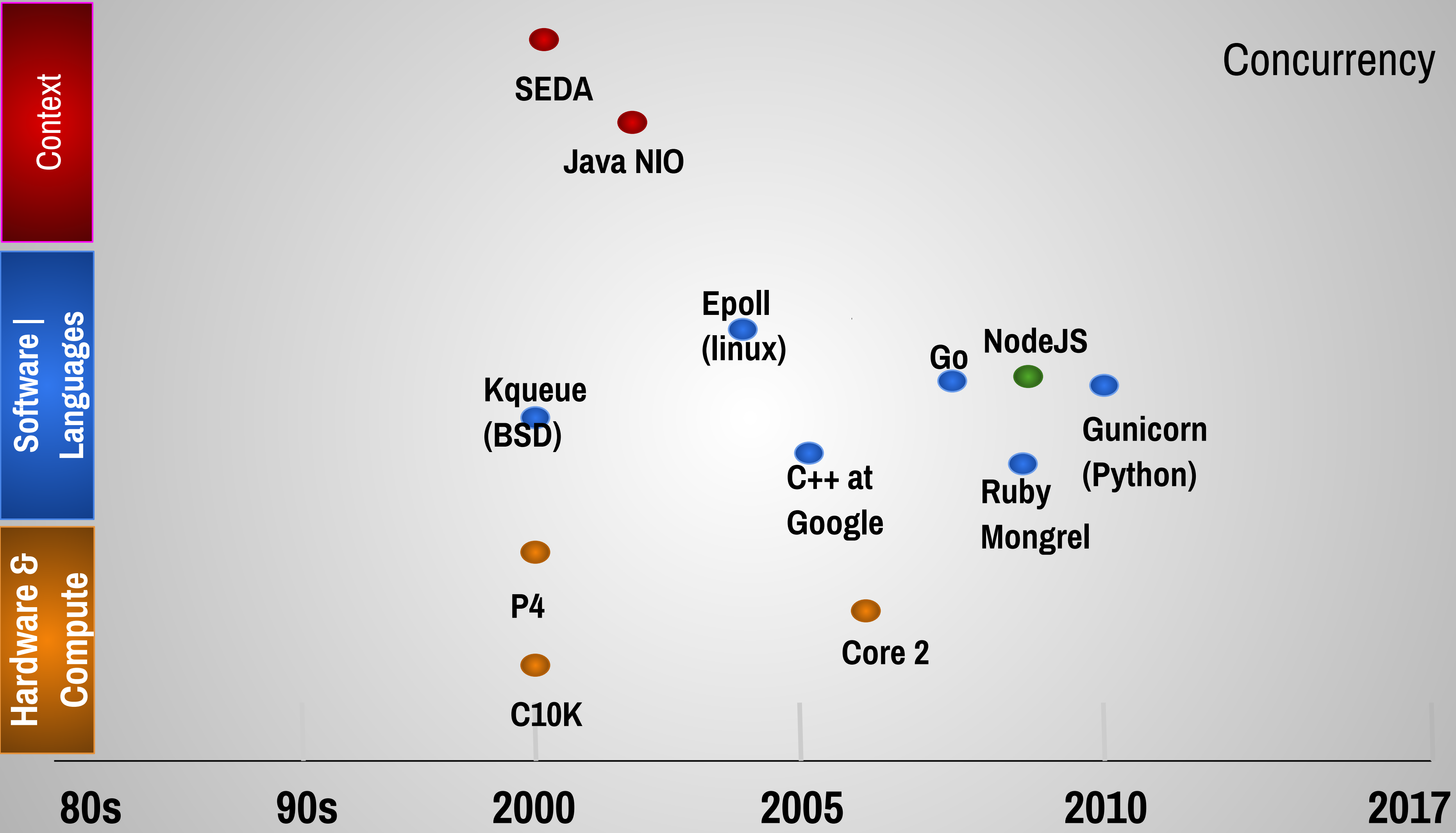
Events, Threads and Goroutines

Nginx - event loop plus state machine model

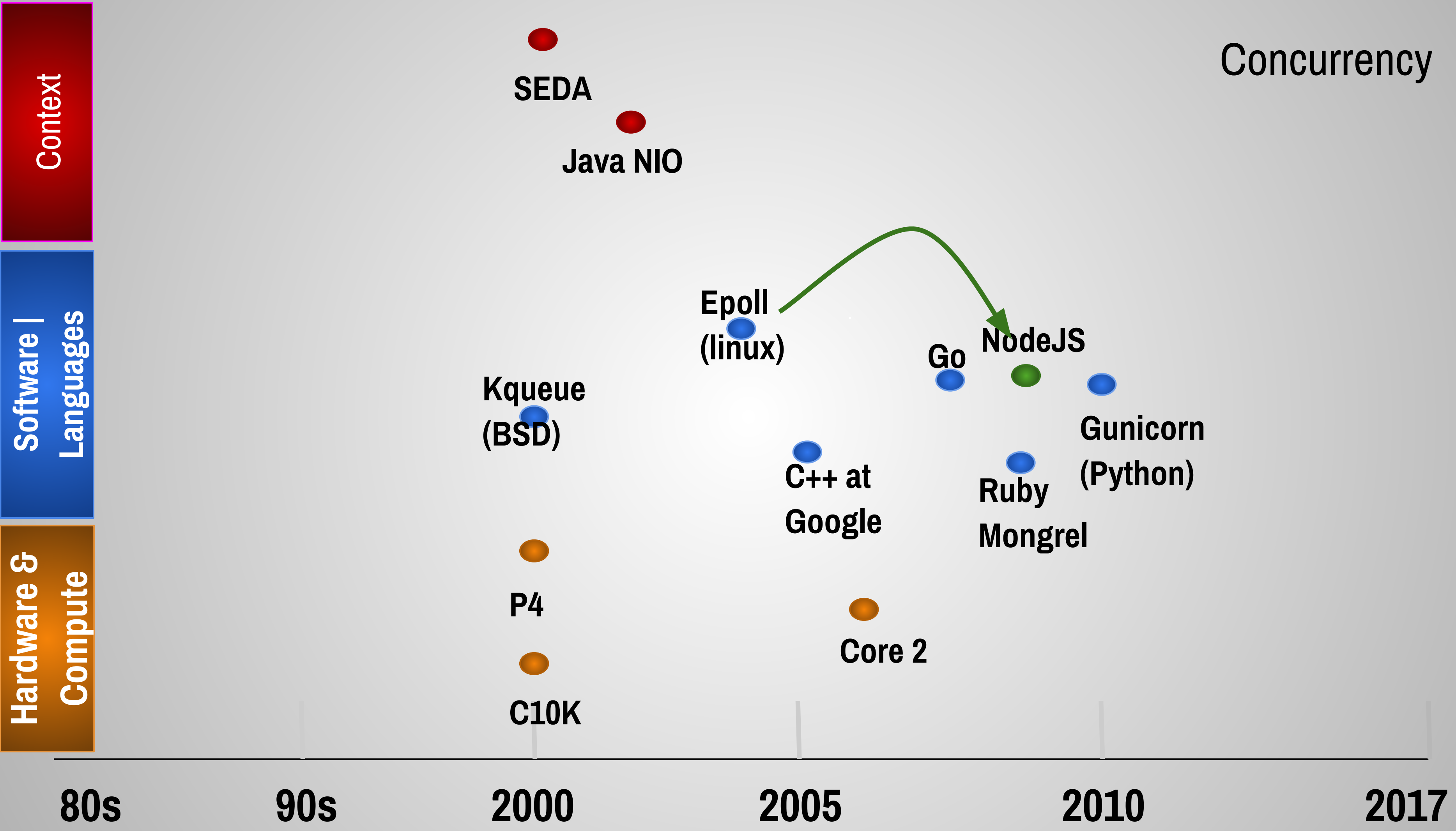
App



Concurrency



Concurrency



Context

Software | Languages

Hardware & Compute

SEDA

Java NIO

Epoll (linux)

Kqueue (BSD)

C++ at Google

Go

NodeJS

Gunicorn (Python)

Ruby Mongrel

P4

C10K

Core 2

80s

90s

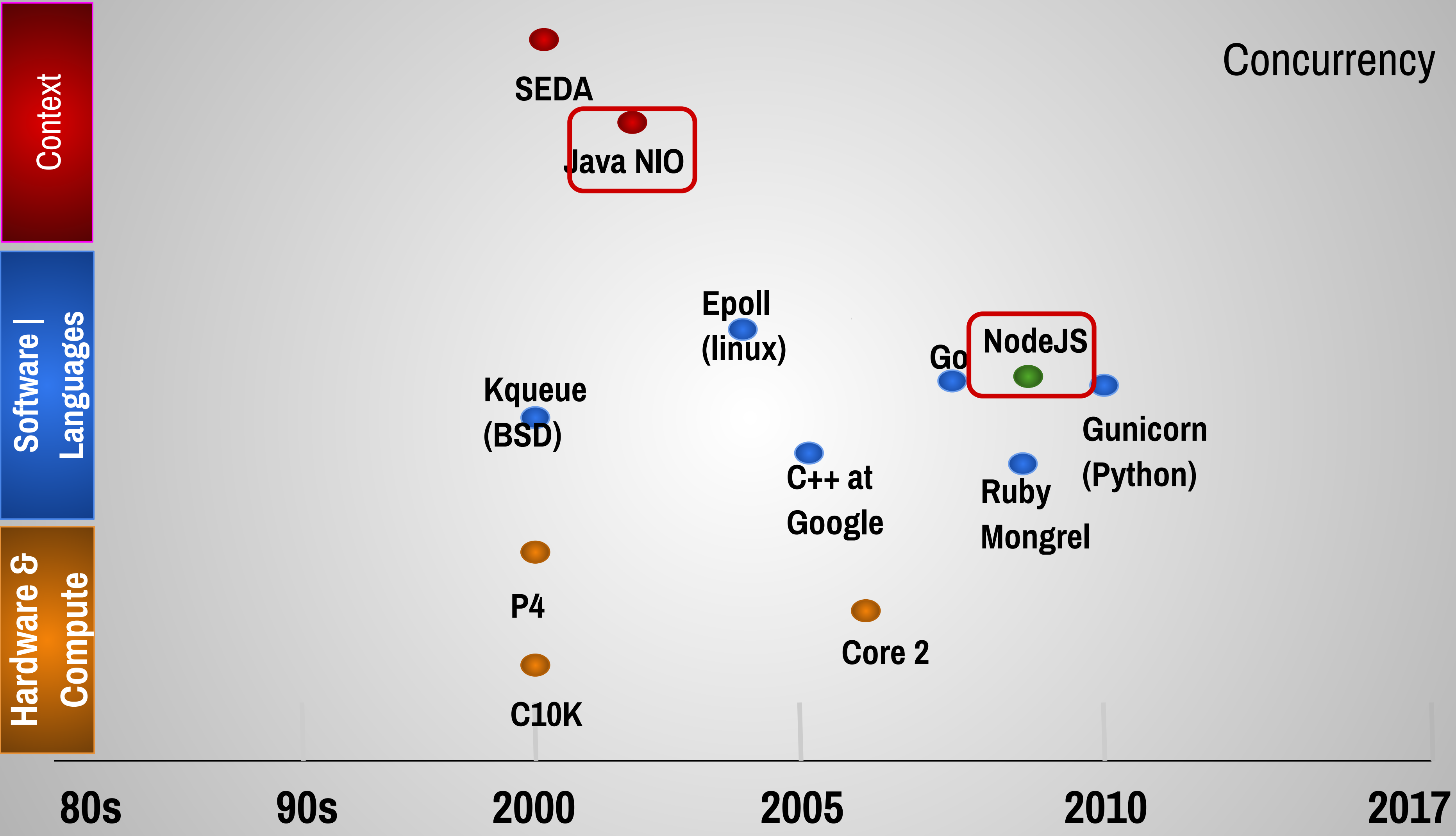
2000

2005

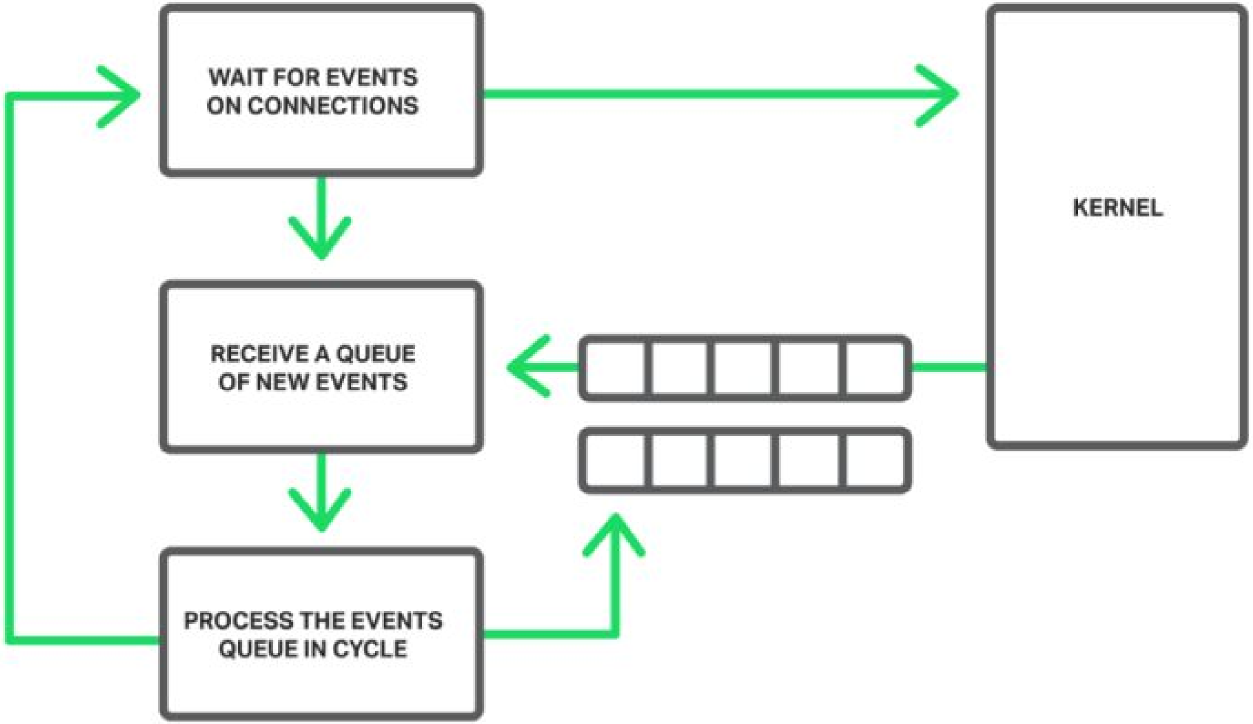
2010

2017

Concurrency



NGINX EVENT LOOP

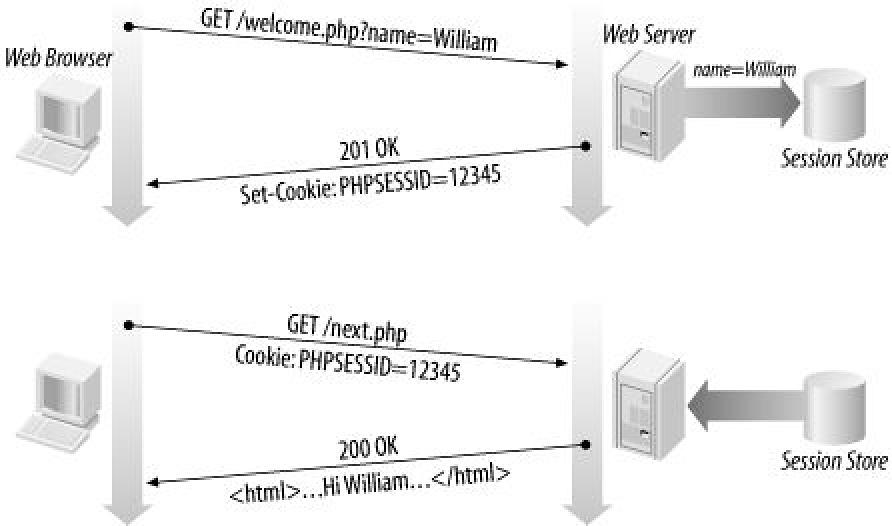


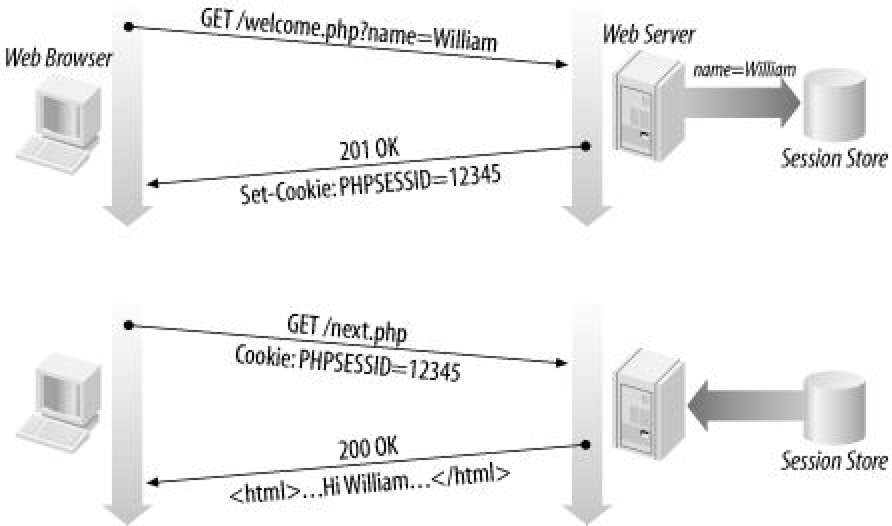
WAIT FOR EVENTS ON CONNECTIONS

RECEIVE A QUEUE OF NEW EVENTS

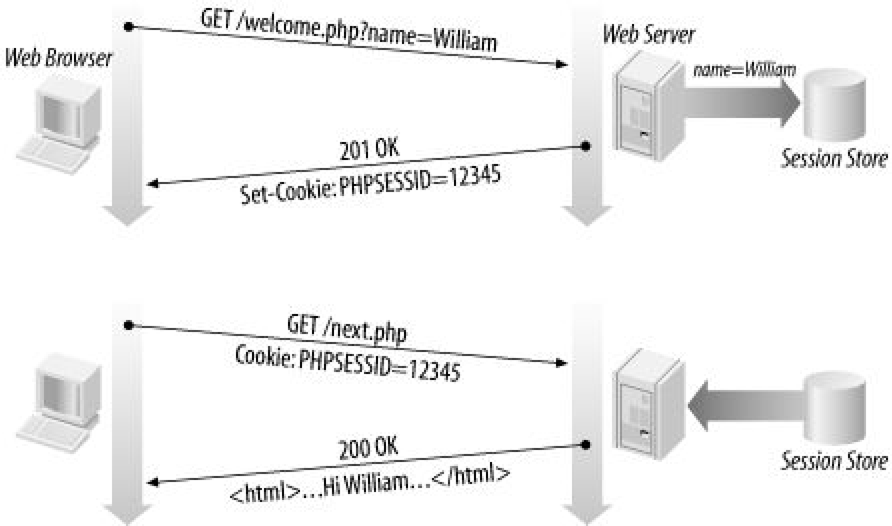
PROCESS THE EVENTS QUEUE IN CYCLE

KERNEL

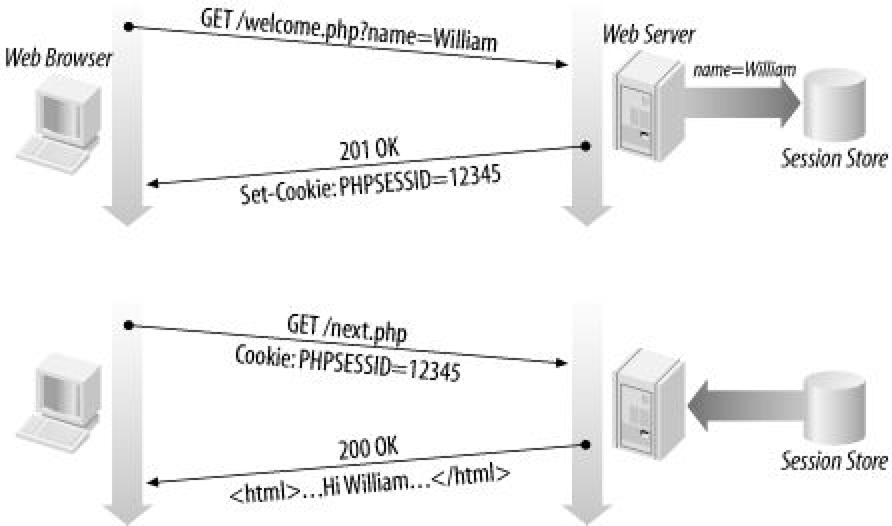

Web Browser
GET /welcome.php?name=William
Web Server
name=William
Session Store
201 OK
Set-Cookie: PHPSESSID=12345
GET /next.php
Cookie: PHPSESSID=12345
Session Store
200 OK
<html>…Hi William…</html>


Web Browser
GET /welcome.php?name=William
Web Server
name=William
Session Store
201 OK
Set-Cookie: PHPSESSID=12345
GET /next.php
Cookie: PHPSESSID=12345
Session Store
200 OK
<html>…Hi William…</html>

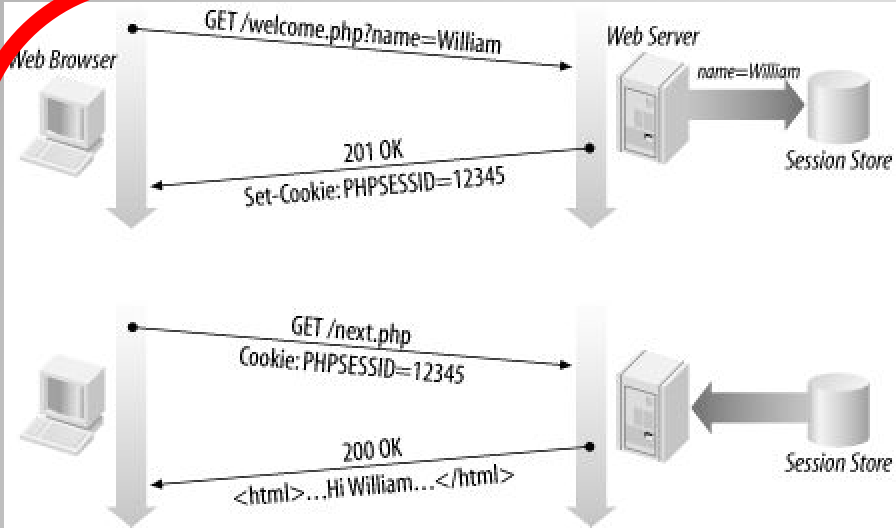
www.


Web Browser
GET /welcome.php?name=William
Web Server
name=William
Session Store
201 OK
Set-Cookie: PHPSESSID=12345
GET /next.php
Cookie: PHPSESSID=12345
Session Store
200 OK
<html>…Hi William…</html>

www.

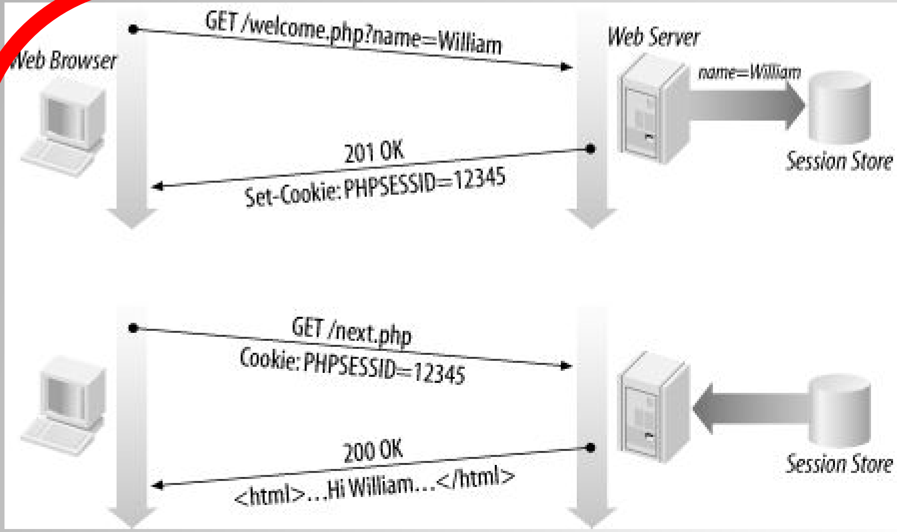

Web Browser
Web Server
GET /welcome.php?name=William
name=William
Session Store
201 OK
Set-Cookie: PHPSESSID=12345
GET /next.php
Cookie: PHPSESSID=12345
Session Store
200 OK
<html>…Hi William…</html>

www.


Web Browser
GET /welcome.php?name=William
Web Server
name=William
Session Store
201 OK
Set-Cookie: PHPSESSID=12345
GET /next.php
Cookie: PHPSESSID=12345
Session Store
200 OK
<html>...Hi William...</html>

www.

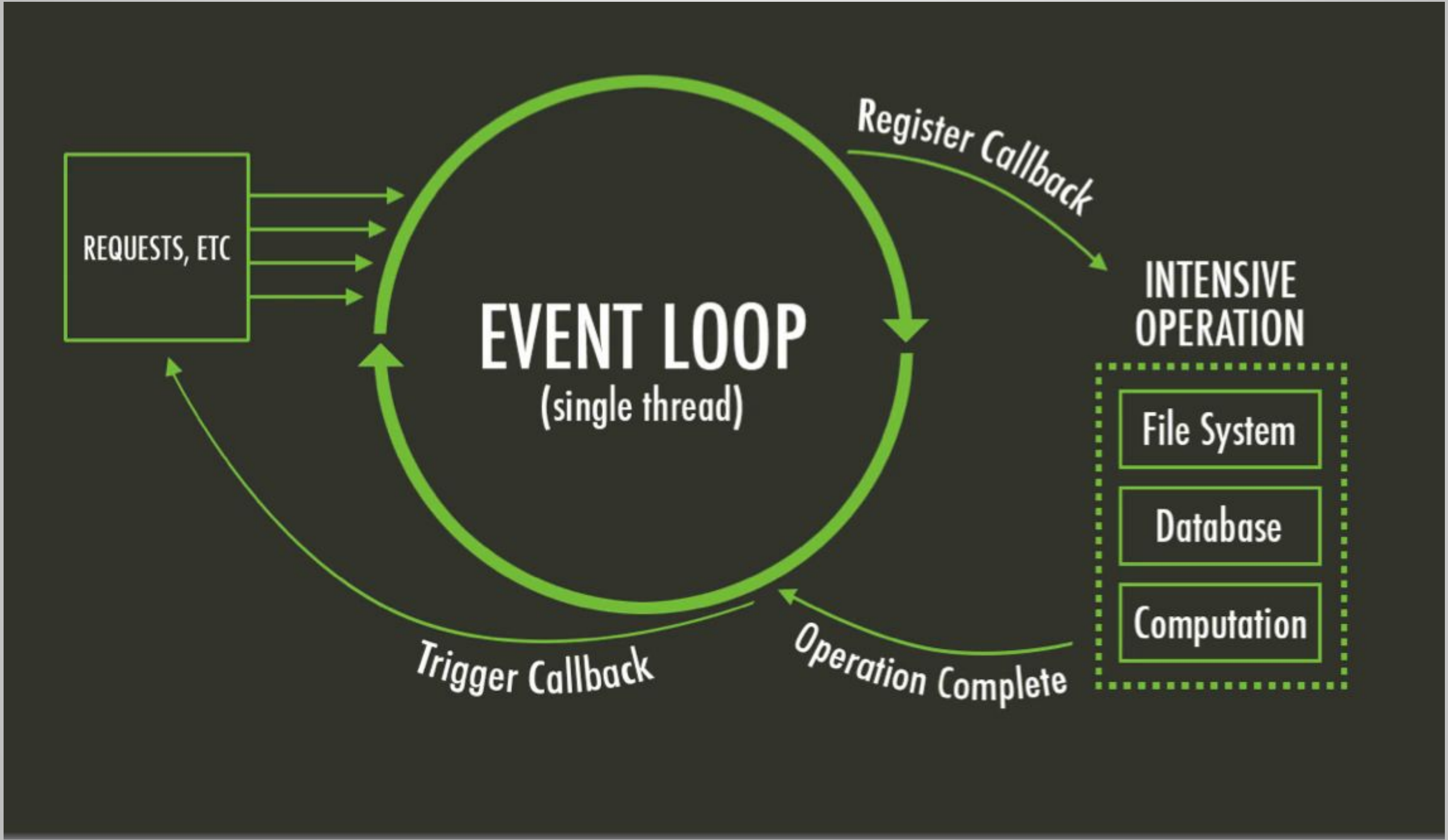
App

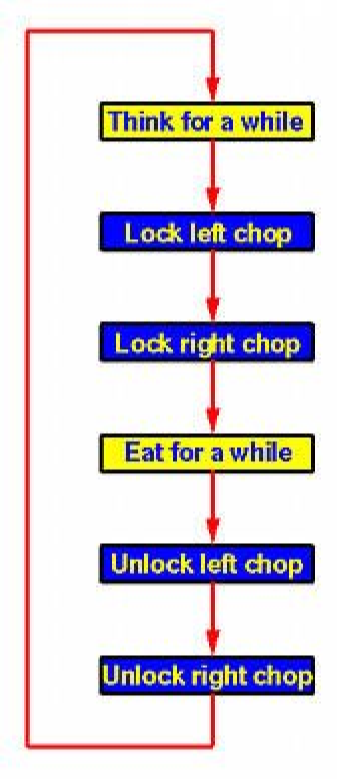

Web Browser
GET /welcome.php?name=William
Web Server
name=William
Session Store
201 OK
Set-Cookie: PHPSESSID=12345
GET /next.php
Cookie: PHPSESSID=12345
Session Store
200 OK
<html>...Hi William...</html>

www.

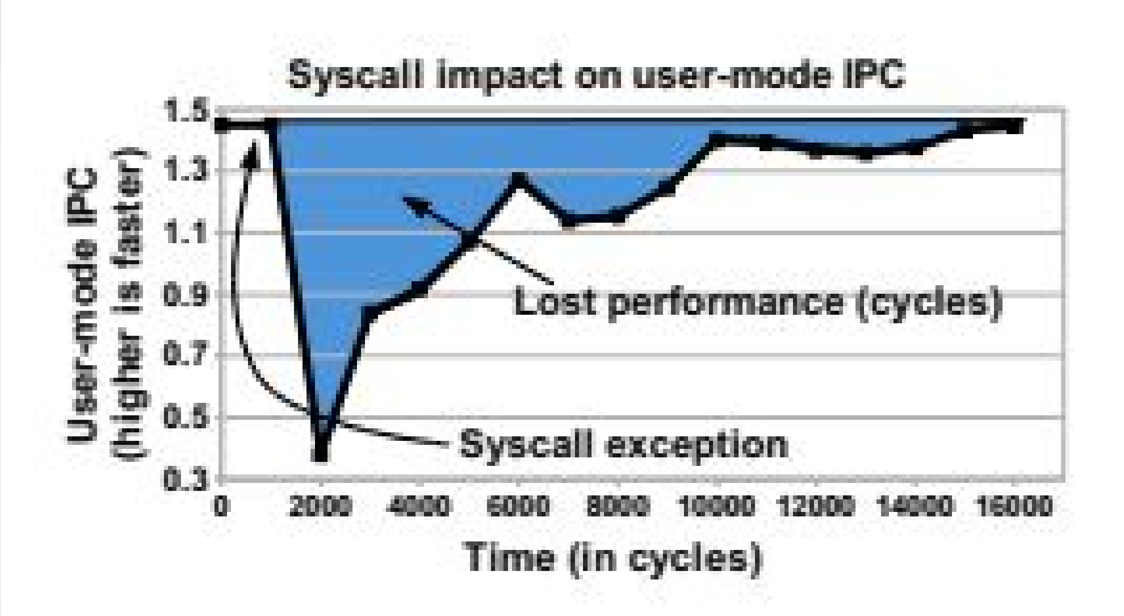
= Threads

App

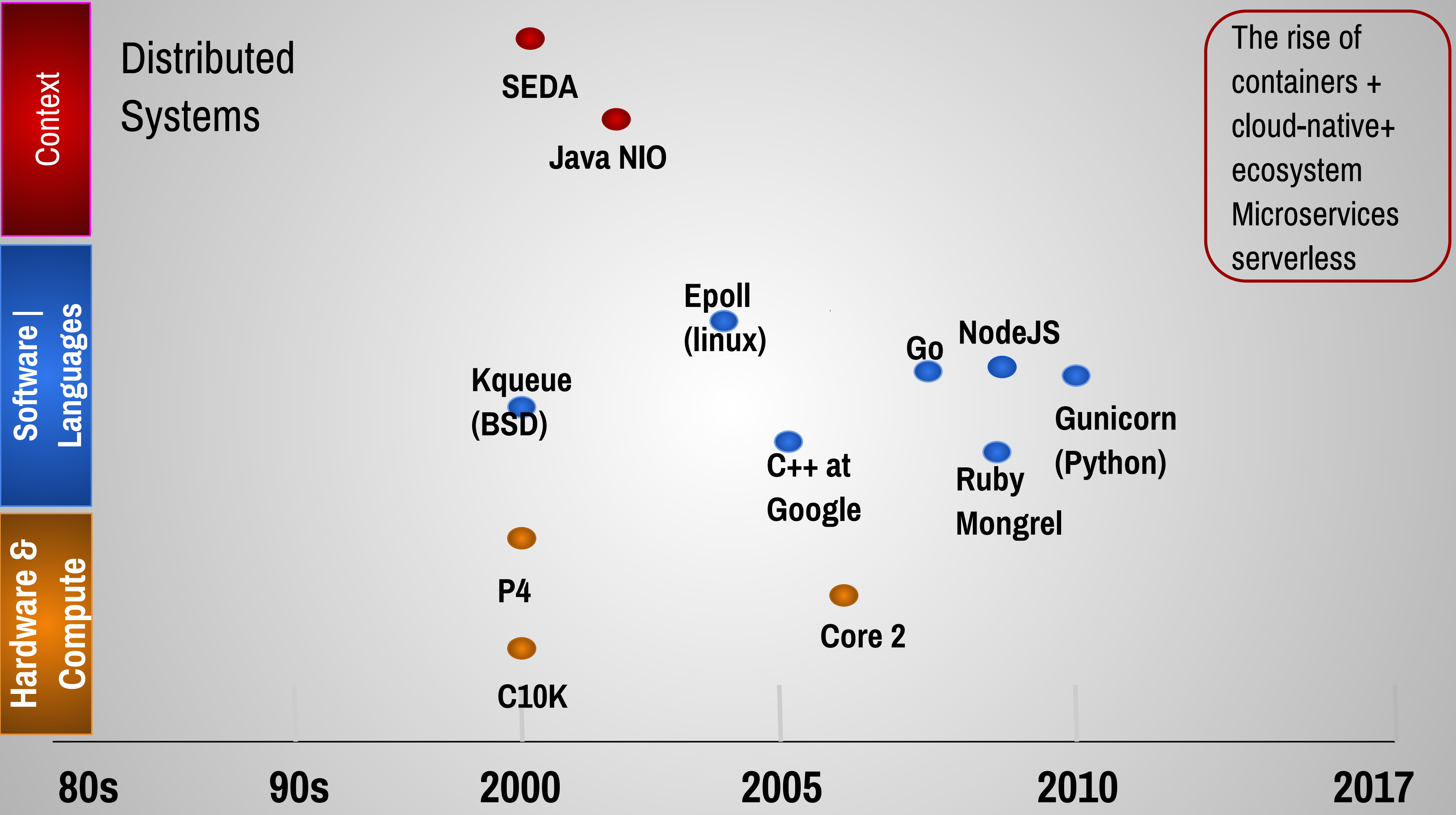

REQUESTS, ETC
EVENT LOOP
(single thread)
Register Callback
INTENSIVE OPERATION
File System
Database
Computation
Operation Complete
Trigger Callback



Think for a while
Lock left chop
Lock right chop
Eat for a while
Unlock left chop
Unlock right chop


Syscall impact on user-mode IPC
User-mode IPC (higher is faster)
1.5
1.3
1.1
0.9
0.7
0.5
0.3
0
2000
4000
6000
8000
10000
12000
14000
16000
Time (in cycles)
Lost performance (cycles)
Syscall exception

Distributed Systems



Context

SEDA

Java NIO

The rise of containers + cloud-native+ ecosystem Microservices serverless

Software | Languages

Epoll (linux)

Go

NodeJS

Kqueue (BSD)

Gunicorn (Python)

C++ at Google

Ruby Mongrel

Hardware & Compute

P4

Core 2

C10K

80s

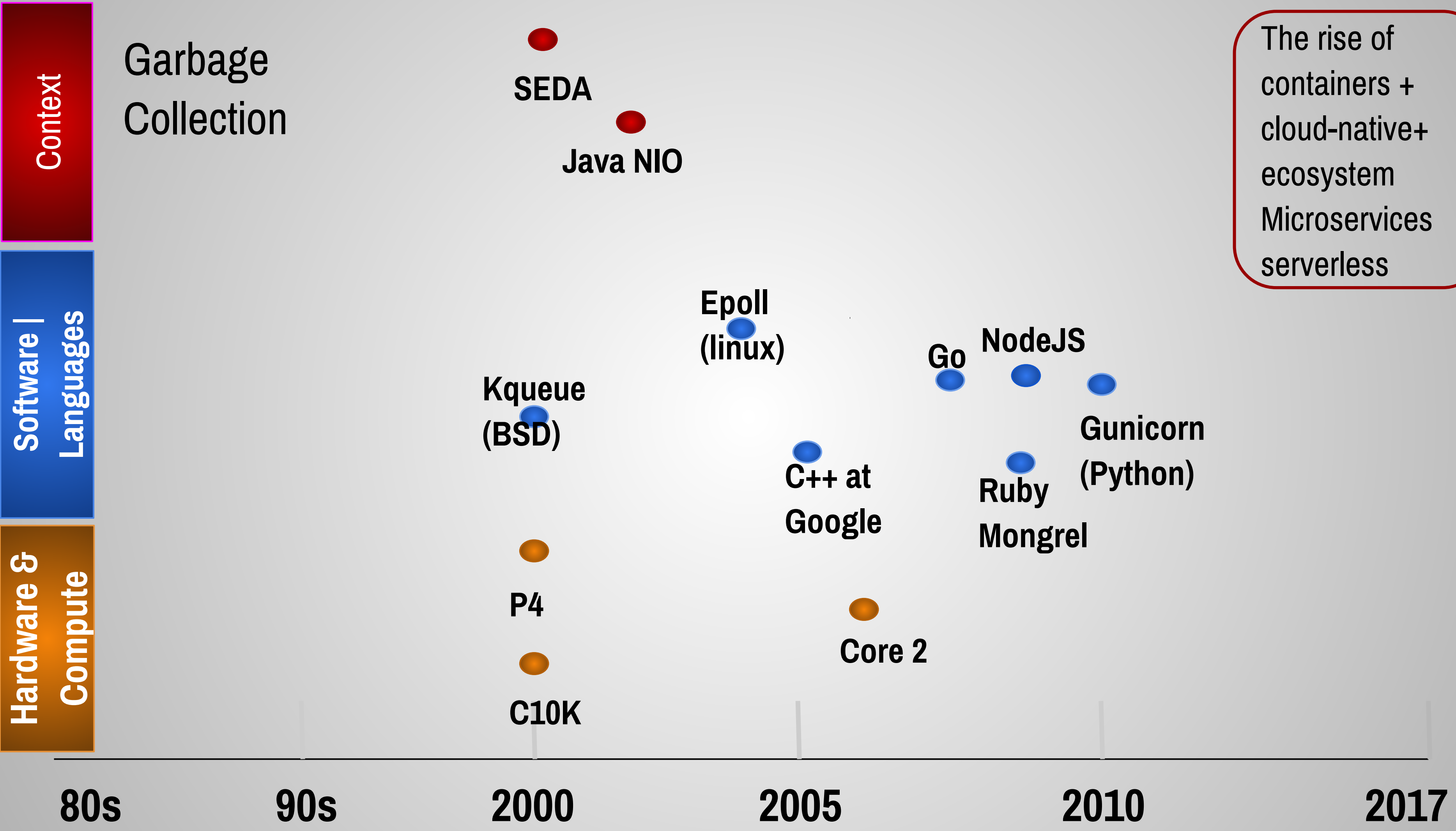
90s

2000

2005

2010

2017



Context

Software | Languages

Hardware & Compute

Garbage Collection

SEDA

Java NIO

Epoll (linux)

Kqueue (BSD)

C++ at Google

Go

NodeJS

Ruby Mongrel

Gunicorn (Python)

P4

C10K

Core 2

80s

90s

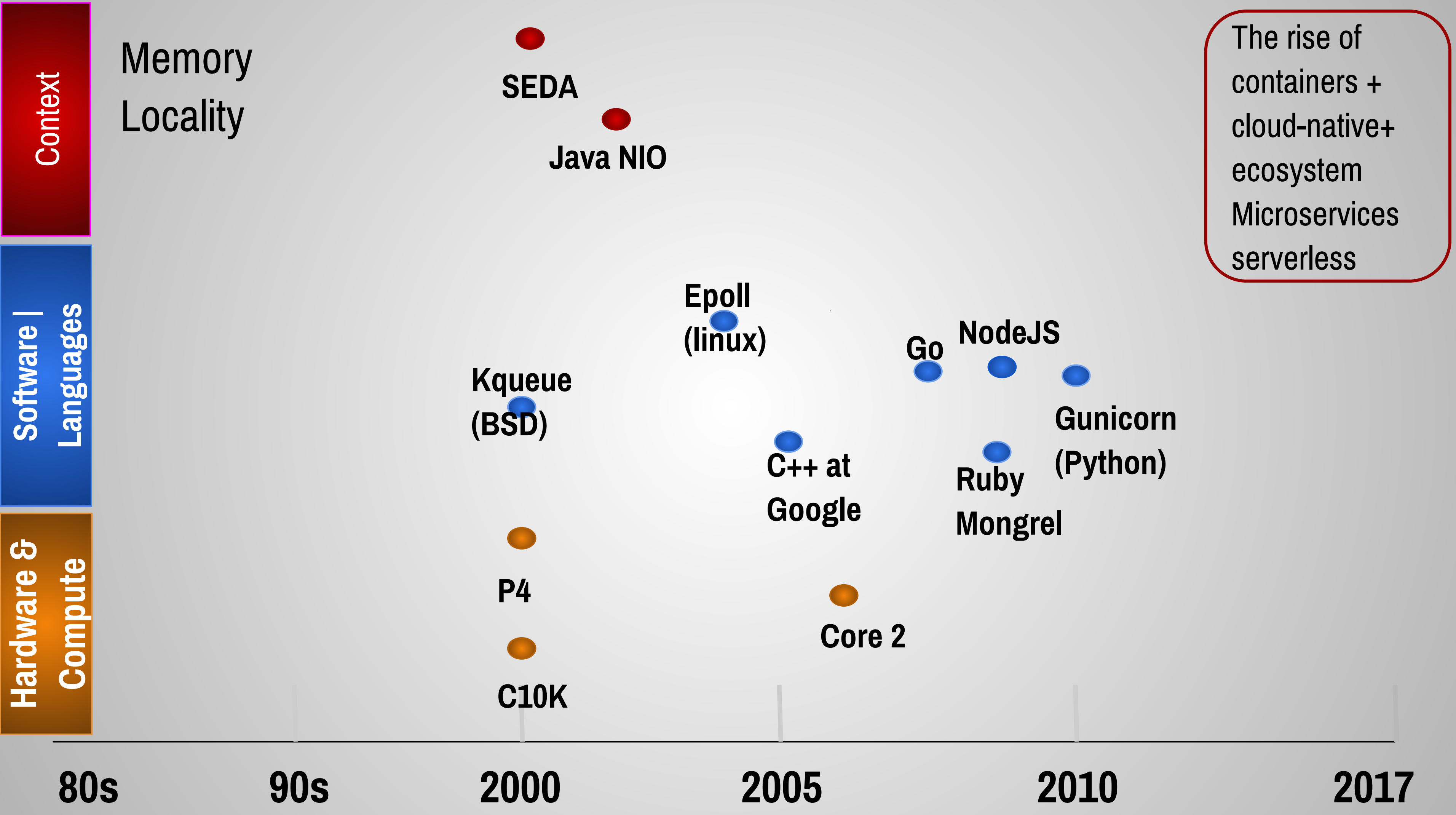
2000

2005

2010

2017

The rise of containers + cloud-native+ ecosystem Microservices serverless



Context

Memory Locality

SEDA

Java NIO

Software | Languages

Epoll (linux)

Kqueue (BSD)

C++ at Google

Go

NodeJS

Ruby Mongrel

Gunicorn (Python)

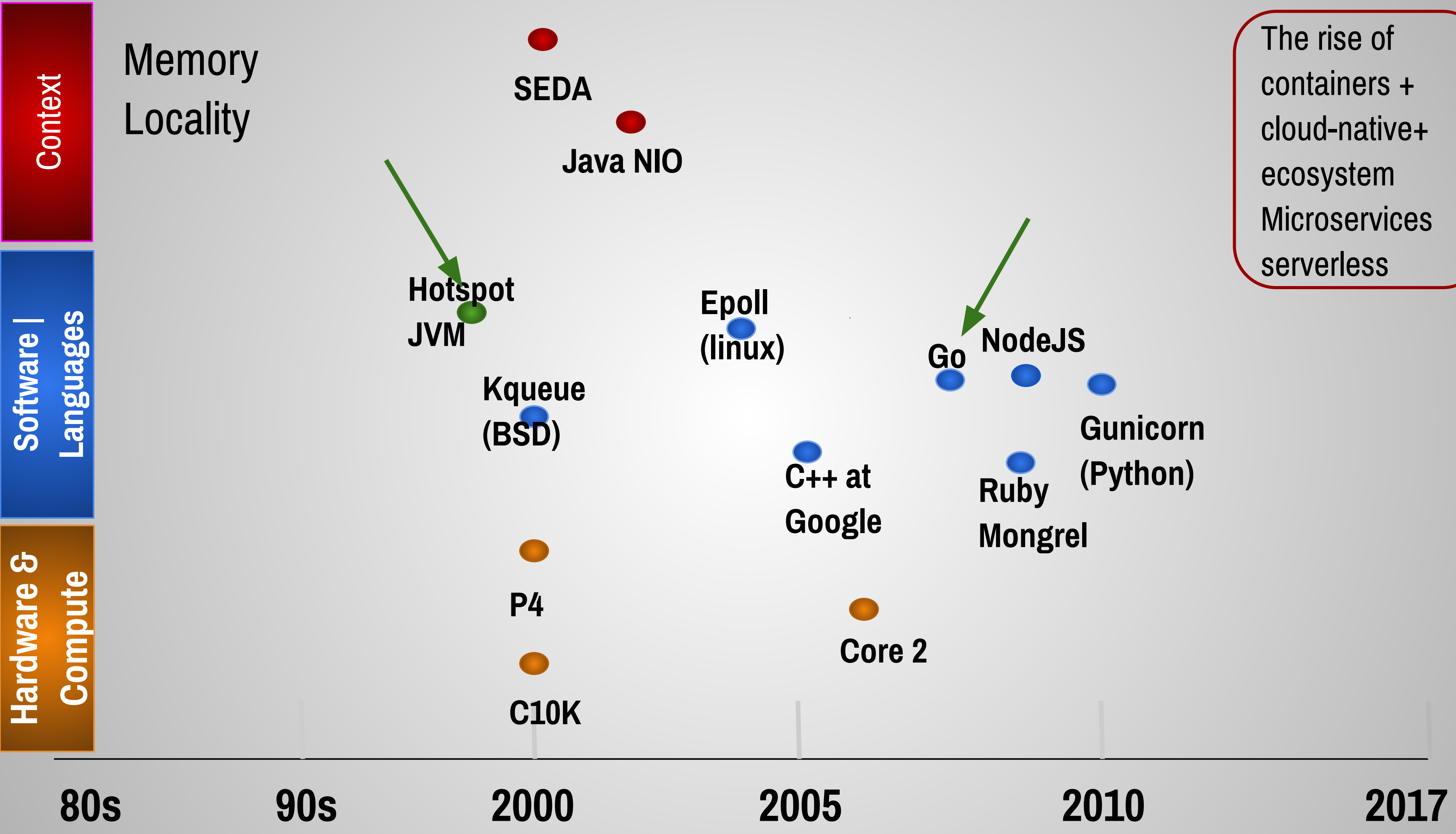
Hardware & Compute

P4

C10K

Core 2

The rise of containers + cloud-native+ ecosystem Microservices serverless



Context

Memory Locality

SEDA

Java NIO

Software | Languages

Hotspot JVM

Kqueue (BSD)

Epoll (linux)

C++ at Google

Go

NodeJS

Ruby Mongrel

Gunicorn (Python)

Hardware & Compute

P4

C10K

Core 2

80s

90s

2000

2005

2010

2017

The rise of containers +
cloud-native+
ecosystem
Microservices
serverless

Memory Locality

Memory Locality

Java

Memory Locality

Java

No value types

Everything Allocated

Memory Locality

Java

No value types

Everything Allocated

Go

Memory Locality

Java

No value types

Everything Allocated

Go

Structs

True Value types

Memory Locality

Java

No value types

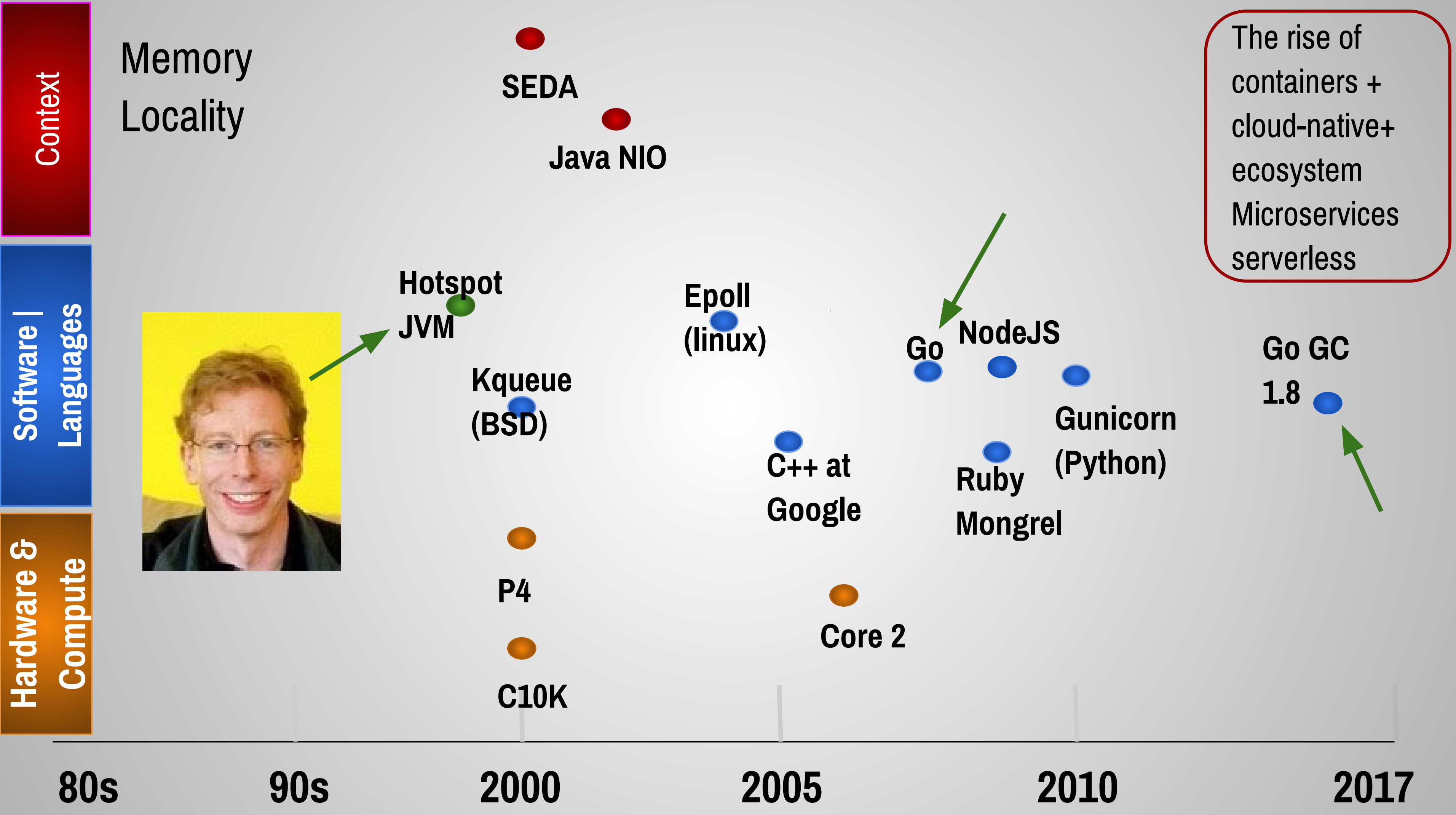
Everything Allocated

Can’t return multiple values

Go

Structs

True Value types



Context

Memory Locality

SEDA

Java NIO

The rise of containers + cloud-native+ ecosystem Microservices serverless

Software | Languages

Hotspot JVM

Kqueue (BSD)

Epoll (linux)

C++ at Google

Go

NodeJS

Ruby Mongrel

Gunicorn (Python)

Go GC 1.8

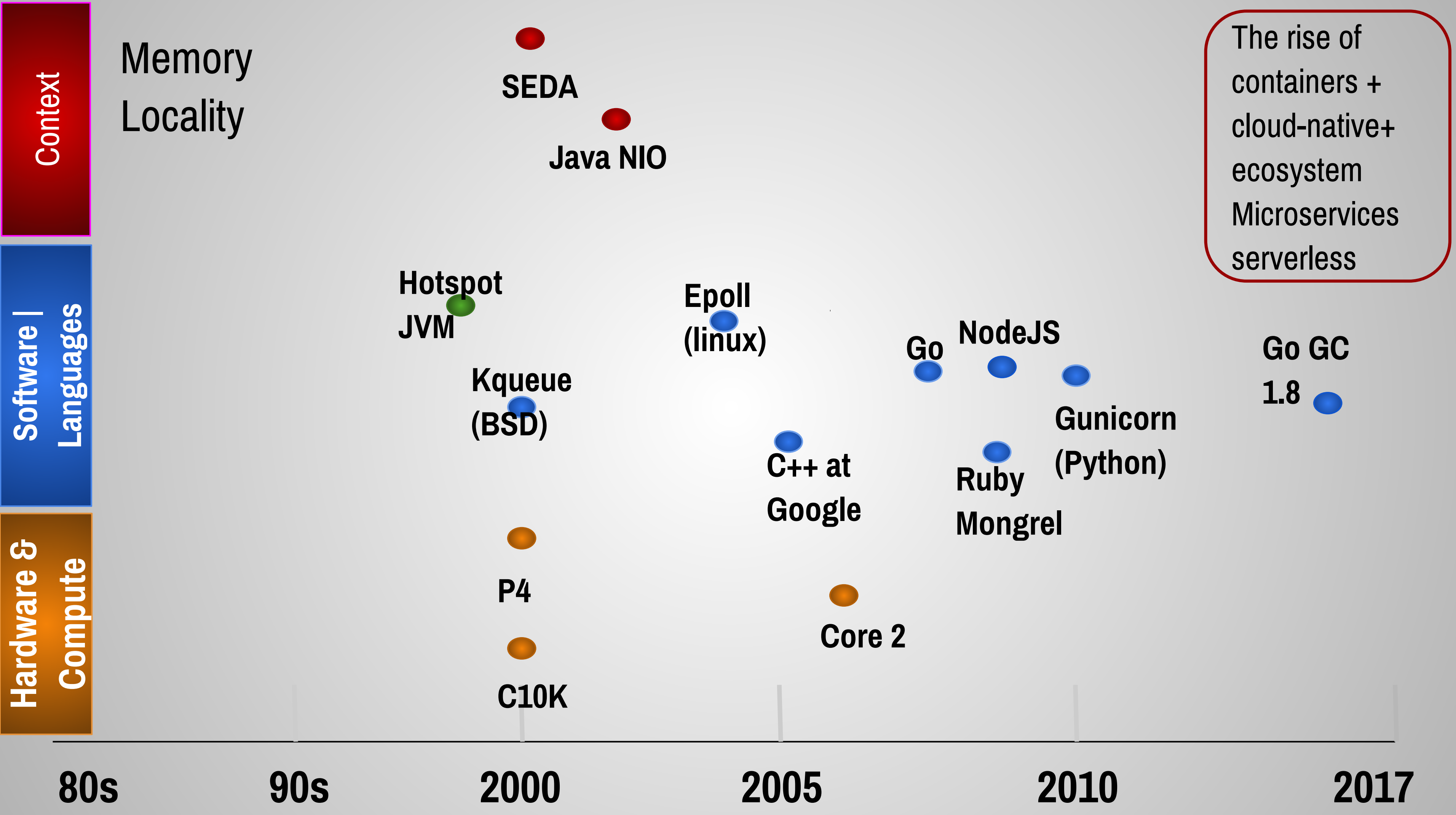
Hardware & Compute

P4

C10K

Core 2

When the three of us [Ken Thompson, Rob Pike, and Robert Griesemer] got started, it was pure research. The three of us got together and decided that we hated C++. [laughter] ... [Returning to Go,] we started off with the idea that all three of us had to be talked into every feature in the language, so there was no extraneous garbage put into the language for any reason.



Context

Software | Languages

Hardware & Compute

Memory Locality

SEDA

Java NIO

Hotspot JVM

Kqueue (BSD)

Epoll (linux)

C++ at Google

Go

NodeJS

Ruby Mongrel

Gunicorn (Python)

Go GC 1.8

P4

C10K

Core 2

The rise of containers + cloud-native+ ecosystem Microservices serverless

80s

90s

2000

2005

2010

2017

Memory Locality

Java

No value types

Everything Allocated

Can’t return multiple values

Go

Structs

True Value types

Memory Locality

Java	Go
No value types	Structs
Everything Allocated	True Value types
Can’t return multiple values	compact object layout
	No object headers

Memory Locality

Java

No value types

Everything Allocated

Can’t return multiple values

Go

UTF-8

Structs

True Value types

compact object layout

No object headers

Memory Locality

Java

UTF-16

No value types

Everything Allocated

Can’t return multiple values

Go

UTF-8

Structs

True Value types

compact object layout

No object headers

Memory Locality

Java **UTF-16**

No value types

Everything Allocated

Can’t return multiple values

Go **UTF-8**

Structs

True Value types

Compact object layout

No object headers

Lazy initialization of collections

Memory Locality (conclusion)

Memory Locality (conclusion)

- Go gives programmers the tools to talk about memory efficiently ***if they need it.***

Memory Locality (conclusion)

- Go gives programmers the tools to talk about memory efficiently ***if they need it.***
- Flexible

Memory Locality (conclusion)

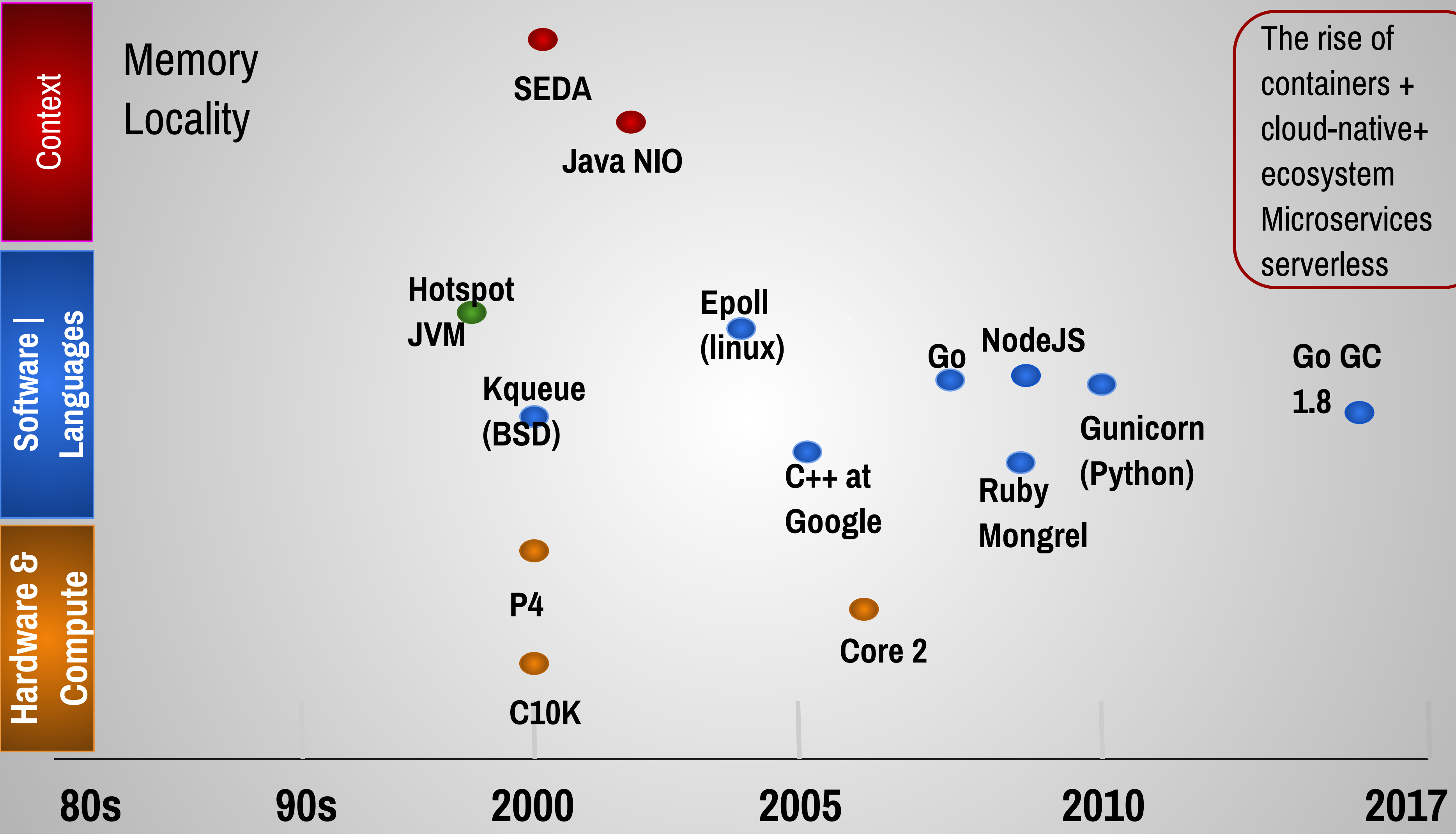
- Go gives programmers the tools to talk about memory efficiently ***if they need it.***
- Flexible
- Memory management (not an all-or-nothing like in C++ or Rust)

Readability

Readability

“ *Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it.* ”—Brian Kernighan

Memory Locality



Context

Software | Languages

Hardware & Compute

SEDA

Java NIO

Hotspot JVM

Kqueue (BSD)

Epoll (linux)

C++ at Google

Go

NodeJS

Ruby Mongrel

Gunicorn (Python)

Go GC 1.8

P4

C10K

Core 2

80s

90s

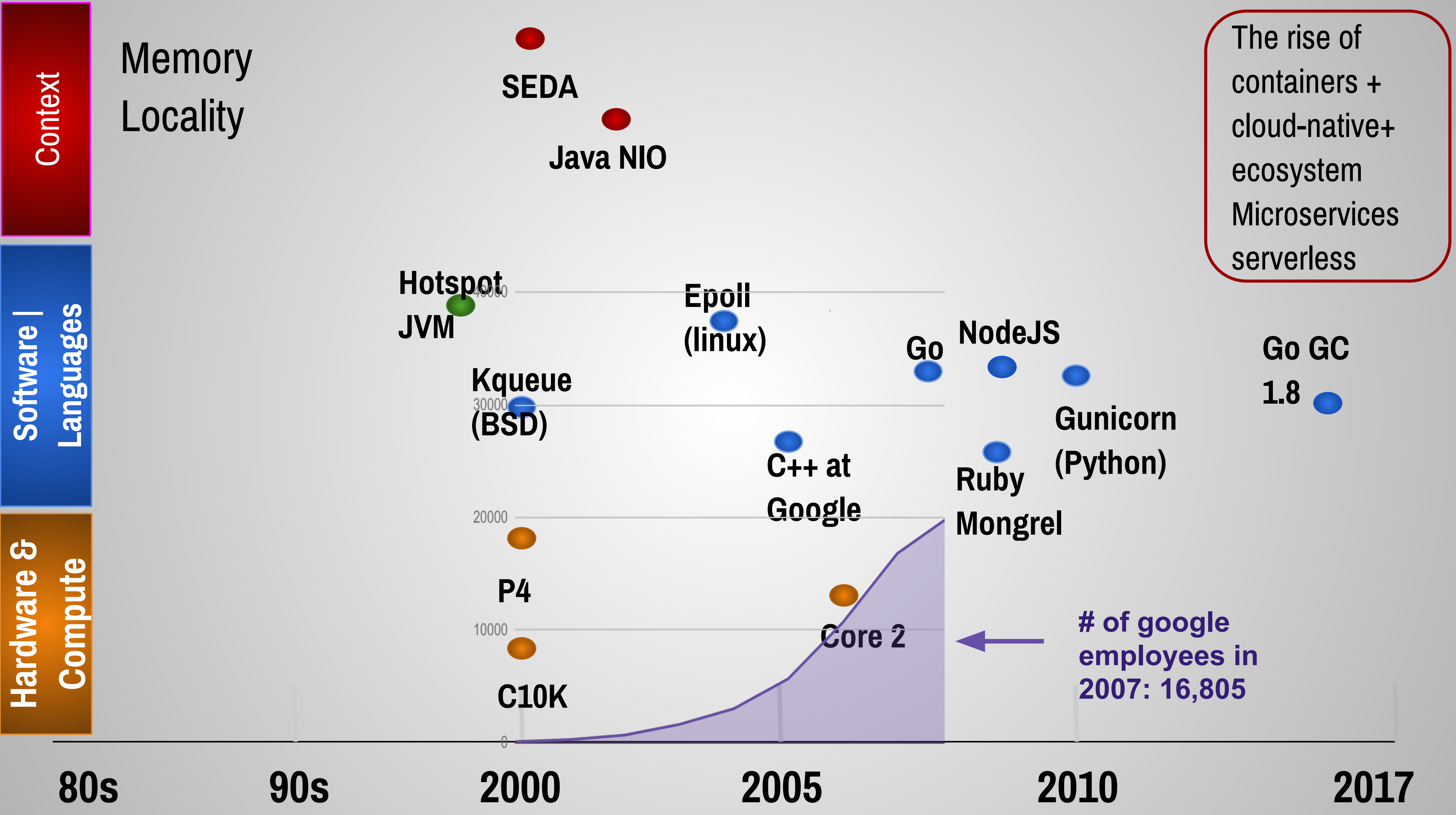
2000

2005

2010

2017

The rise of
containers +
cloud-native+
ecosystem
Microservices
serverless



Context

Software | Languages

Hardware & Compute

Memory Locality

SEDA

Java NIO

Hotspot JVM

Kqueue (BSD)

Epoll (linux)

C++ at Google

Go

NodeJS

Ruby Mongrel

Gunicorn (Python)

Go GC 1.8

P4

C10K

Core 2

# of google employees in 2007: 16,805

The rise of containers + cloud-native+ ecosystem Microservices serverless

80s

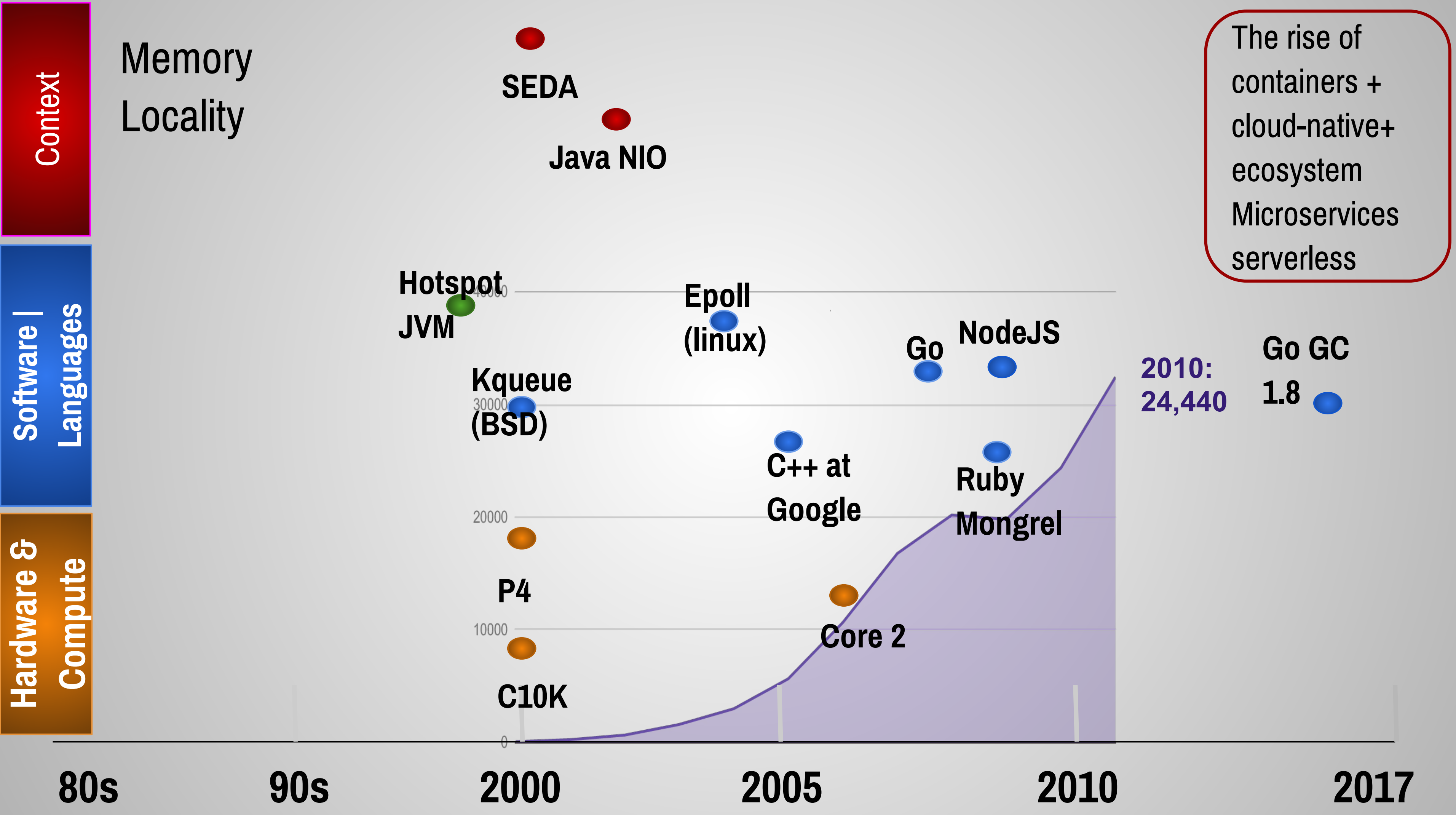
90s

2000

2005

2010

2017



Context

Memory
Locality

SEDA

Java NIO

The rise of
containers +
cloud-native+
ecosystem
Microservices
serverless

Software | Languages

Hotspot
JVM

Kqueue
(BSD)

Epoll
(linux)

C++ at
Google

Go

NodeJS

Ruby
Mongrel

2010:
24,440

Go GC
1.8

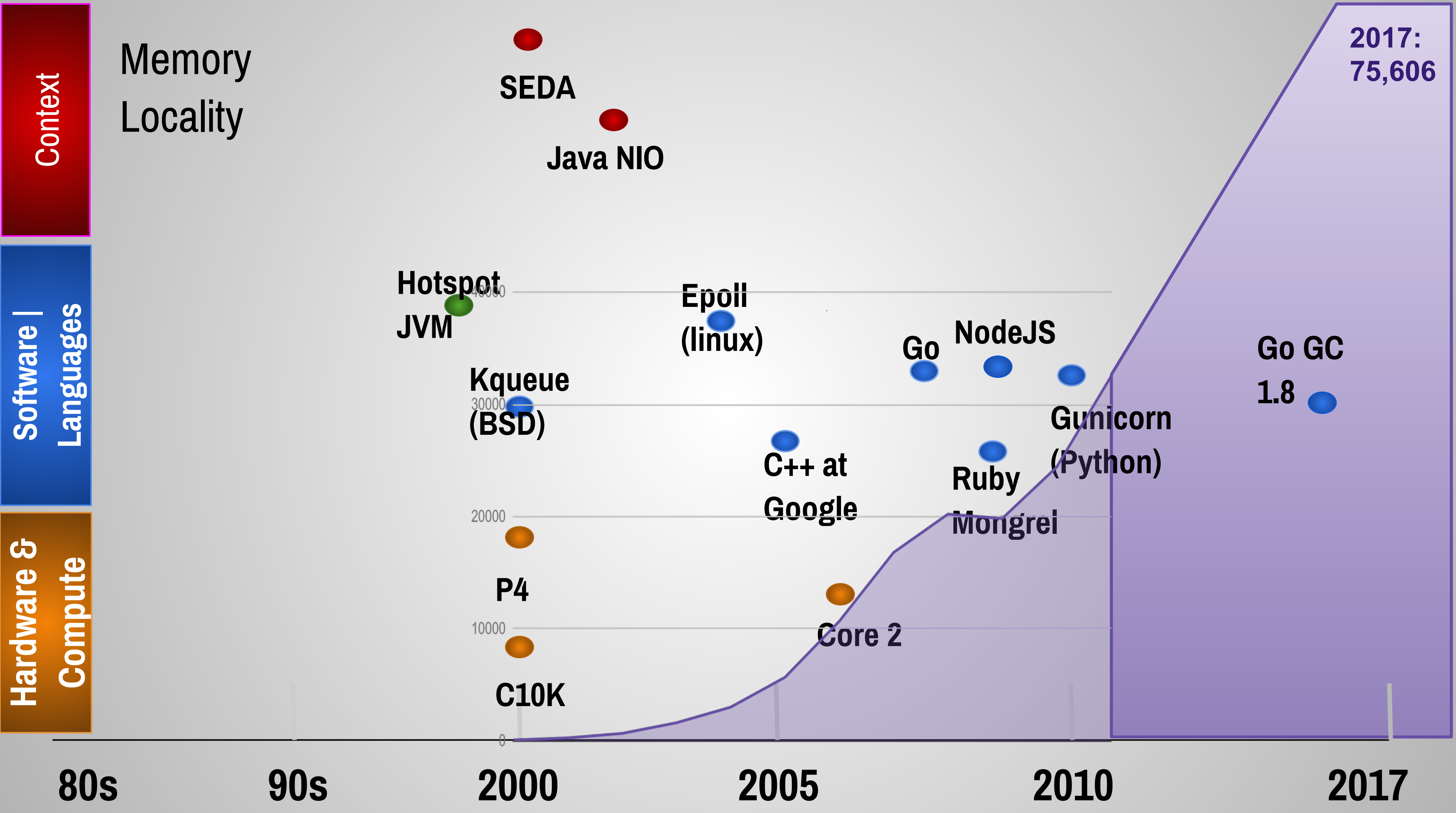
Hardware & Compute

P4

C10K

Core 2

Memory Locality



Readability

simplicity

Readability

simplicity

“simple is better”

Readability

simplicity

“simple is better”

“this is an insult to intelligent programmers”

Readability

simplicity

“simple is better”

“you’re trying to commodify programming and create a situation where our bosses can replace us at will”

“You’re not paid to program, you’re not even paid to maintain someone else’s program, you’re paid to deliver solutions to the business.”

- Dave Cheney

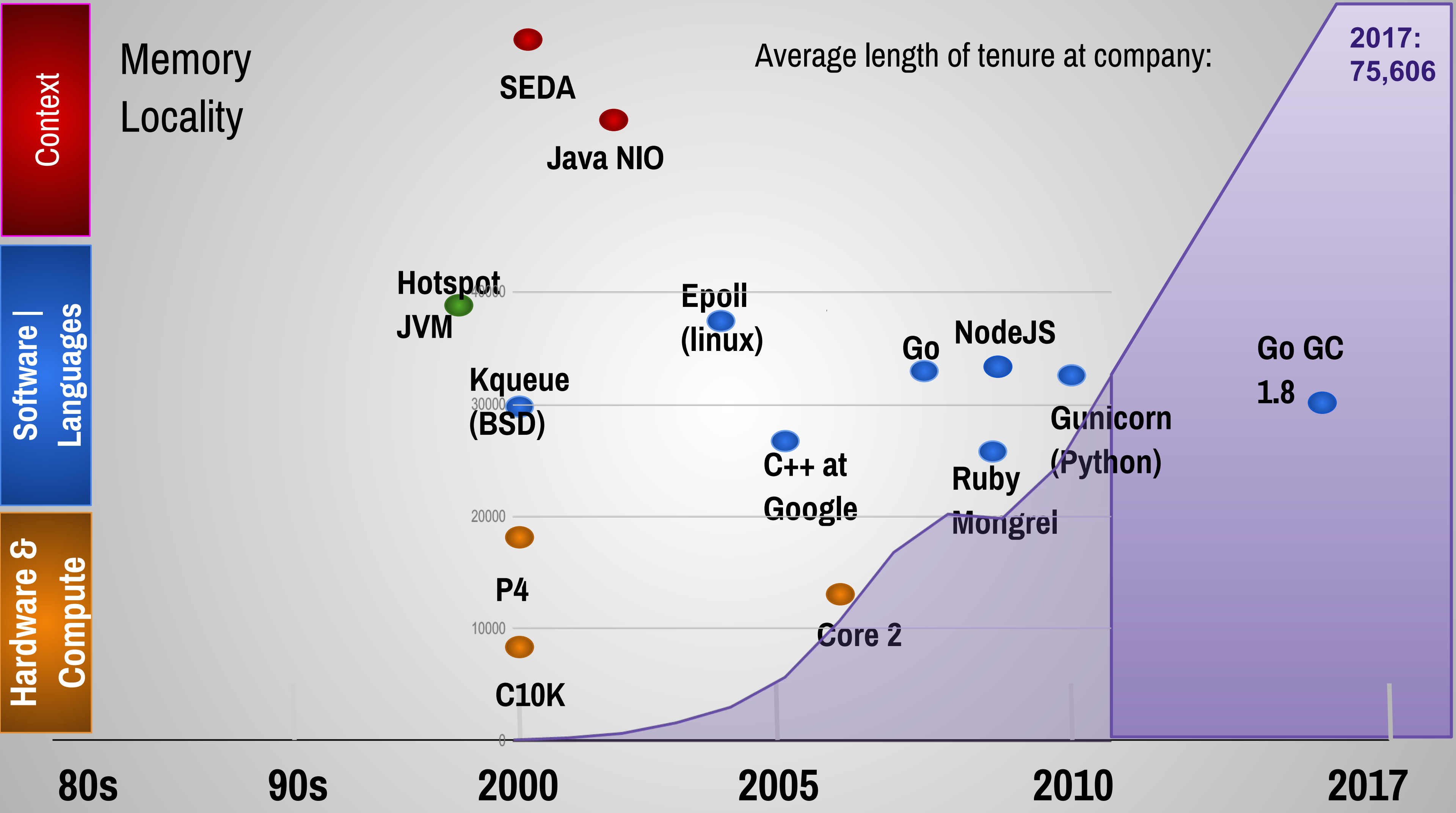
Readability

Programs which cannot be maintained will be rewritten

Readability

Programs which cannot be maintained will be rewritten

“If you can’t be replaced, you cannot be promoted”



Context

Memory Locality

SEDA

Java NIO

Average length of tenure at company:

2017: 75,606

Software | Languages

Hotspot JVM

Epoll (linux)

Go

NodeJS

Kqueue (BSD)

Go GC 1.8

Gunicorn (Python)

C++ at Google

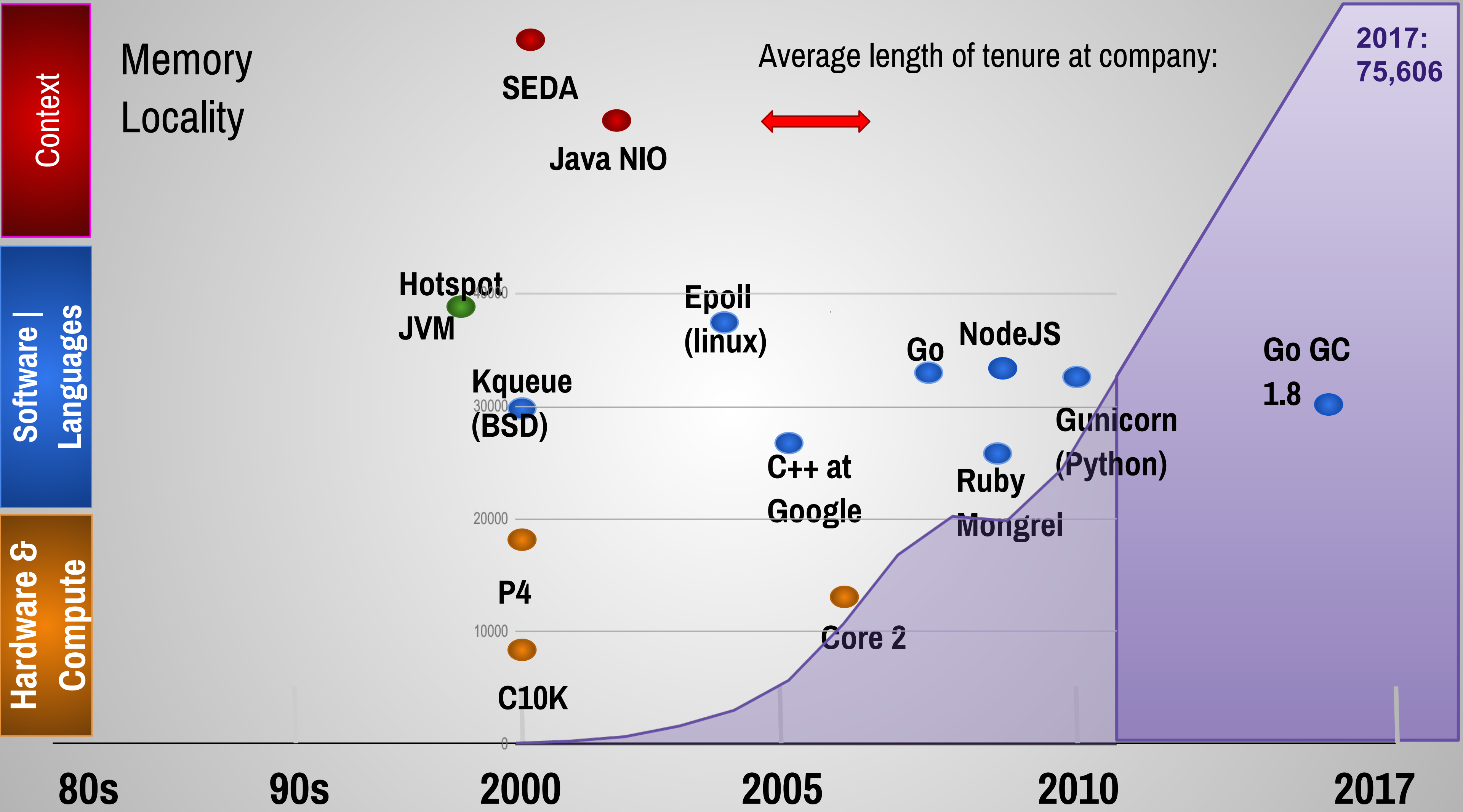
Ruby Mongrel

Hardware & Compute

P4

Core 2

C10K



Context

Memory

Locality

SEDA

Java NIO

Average length of tenure at company:

2017:
75,606

Software | Languages

Hotspot
JVM

Kqueue
(BSD)

Epoll
(linux)

C++ at
Google

Go

NodeJS

Ruby
Mongrel

Gunicorn
(Python)

Go GC
1.8

Hardware & Compute

P4

C10K

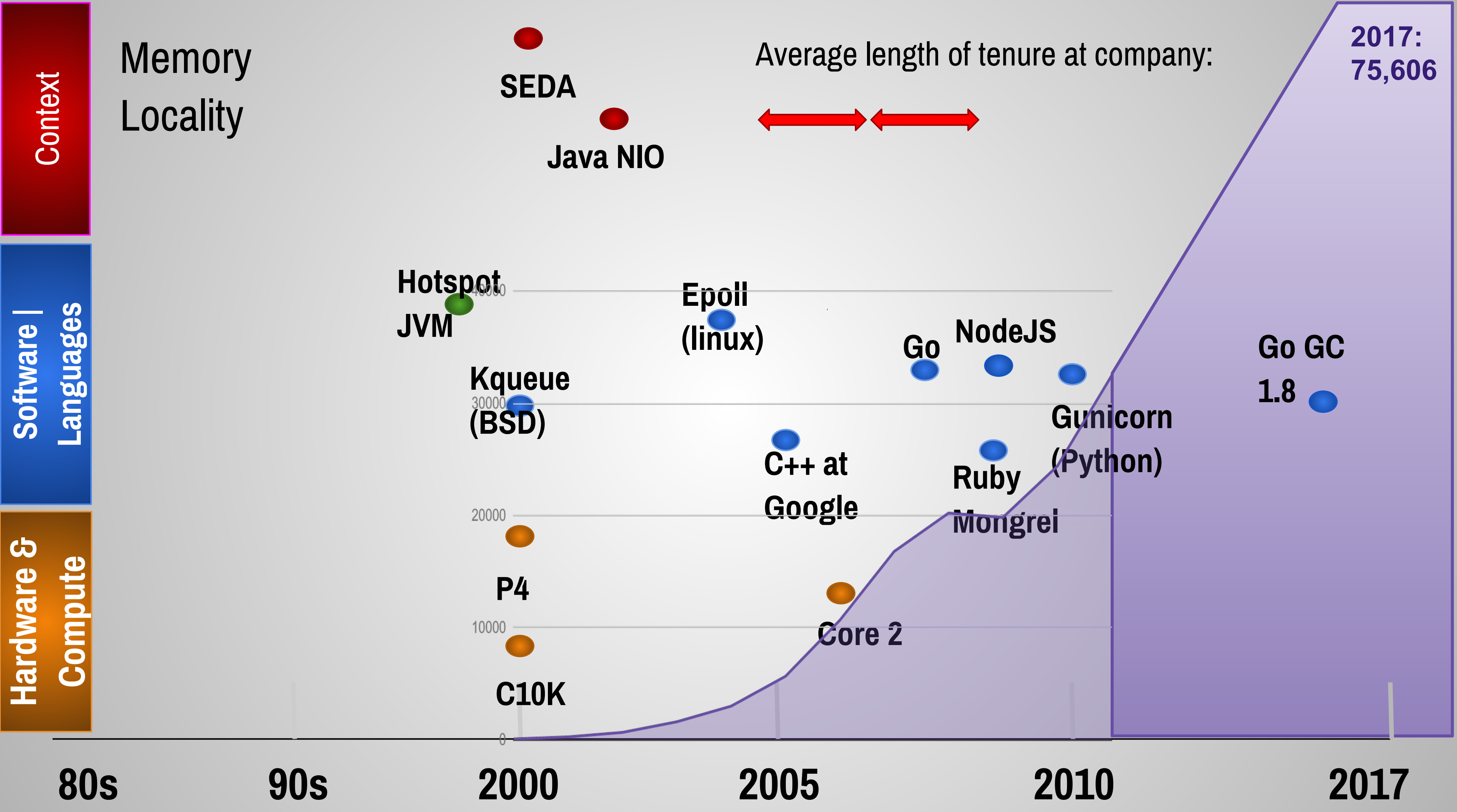
Core 2

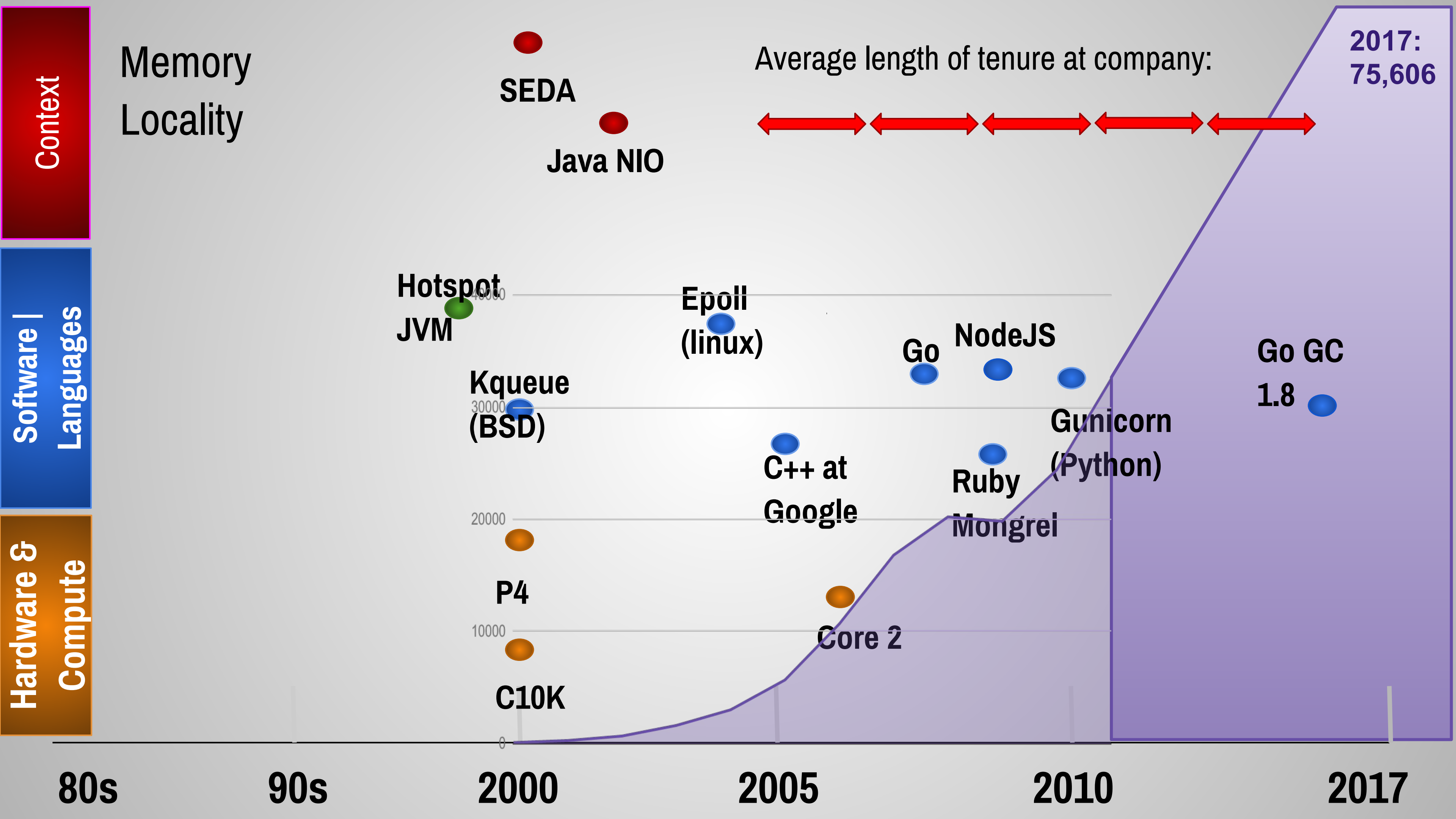
Memory

Locality

Average length of tenure at company:

2017:
75,606





Context

Memory Locality

SEDA

Java NIO

Average length of tenure at company:

2017: 75,606

Software | Languages

Hotspot JVM

Epoll (linux)

Go

NodeJS

Kqueue (BSD)

Go GC 1.8

Gunicorn (Python)

C++ at Google

Ruby Mongrel

Hardware & Compute

P4

Core 2

C10K

Software Engineering

Software Engineering

Software Engineering vs Programming

Software Engineering

Software Engineering vs Programming

Software Engineering = Programming integrated over time.

Software Engineering

Software Engineering vs Programming

Software Engineering = Programming integrated over time.

Engineering is what happens when things need to live longer and influence of time starts creeping in. -Titus Winters

Software Engineering

Software Engineering vs Programming

Software Engineering = Programming integrated over time.

Engineering is what happens when things need to live longer and influence of time starts creeping in. -Titus Winters

All this complexity is fundamentally a different flavor than programming.

Software Engineering

focus on sustaining engineering (readability)

Software Engineering

focus on sustaining engineering (readability)

continuance of many different engineers over a long period of time

Software Engineering

focus on sustaining engineering (readability)

continuance of many different engineers over a long period of time

clear module boundaries

Software Engineering

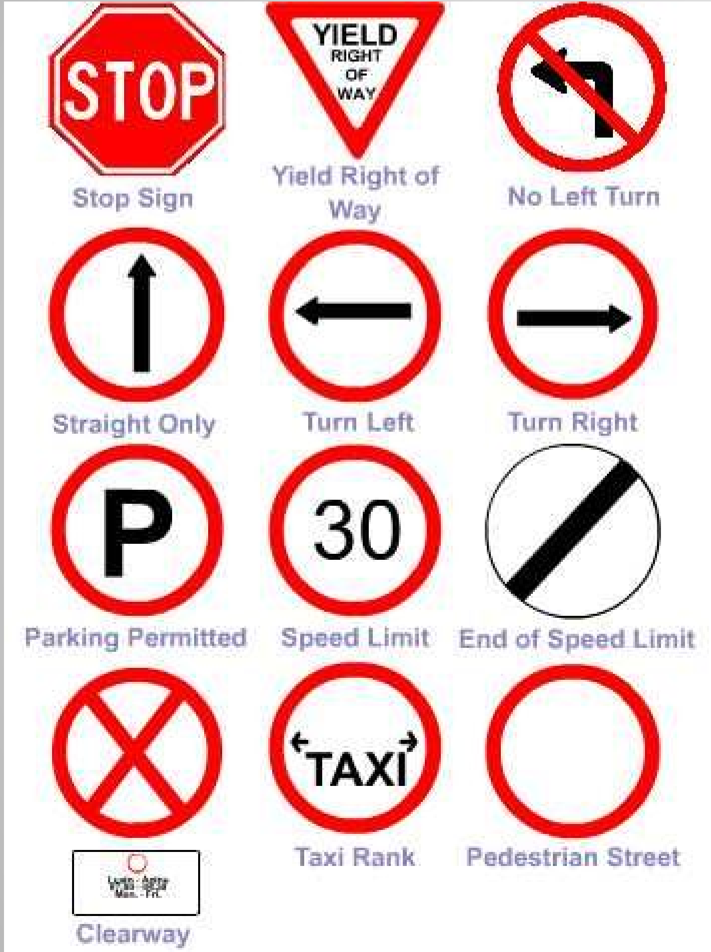
focus on sustaining engineering (readability)

continuance of many different engineers over a long period of time

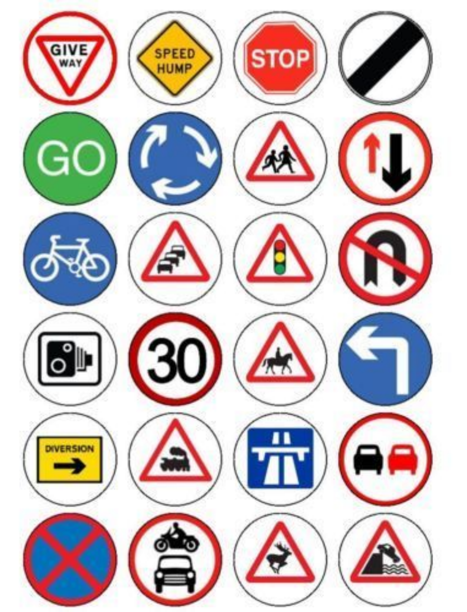
clear module boundaries

keeping import dependencies between packages linear, thus keeping compile times down.

Simplicity and the Greater Good


STOP
Stop Sign
YIELD RIGHT OF WAY
Yield Right of Way
No Left Turn
Straight Only
Turn Left
Turn Right
P
Parking Permitted
30
Speed Limit
End of Speed Limit
TAXI
Taxi Rank
Pedestrian Street
Clearway

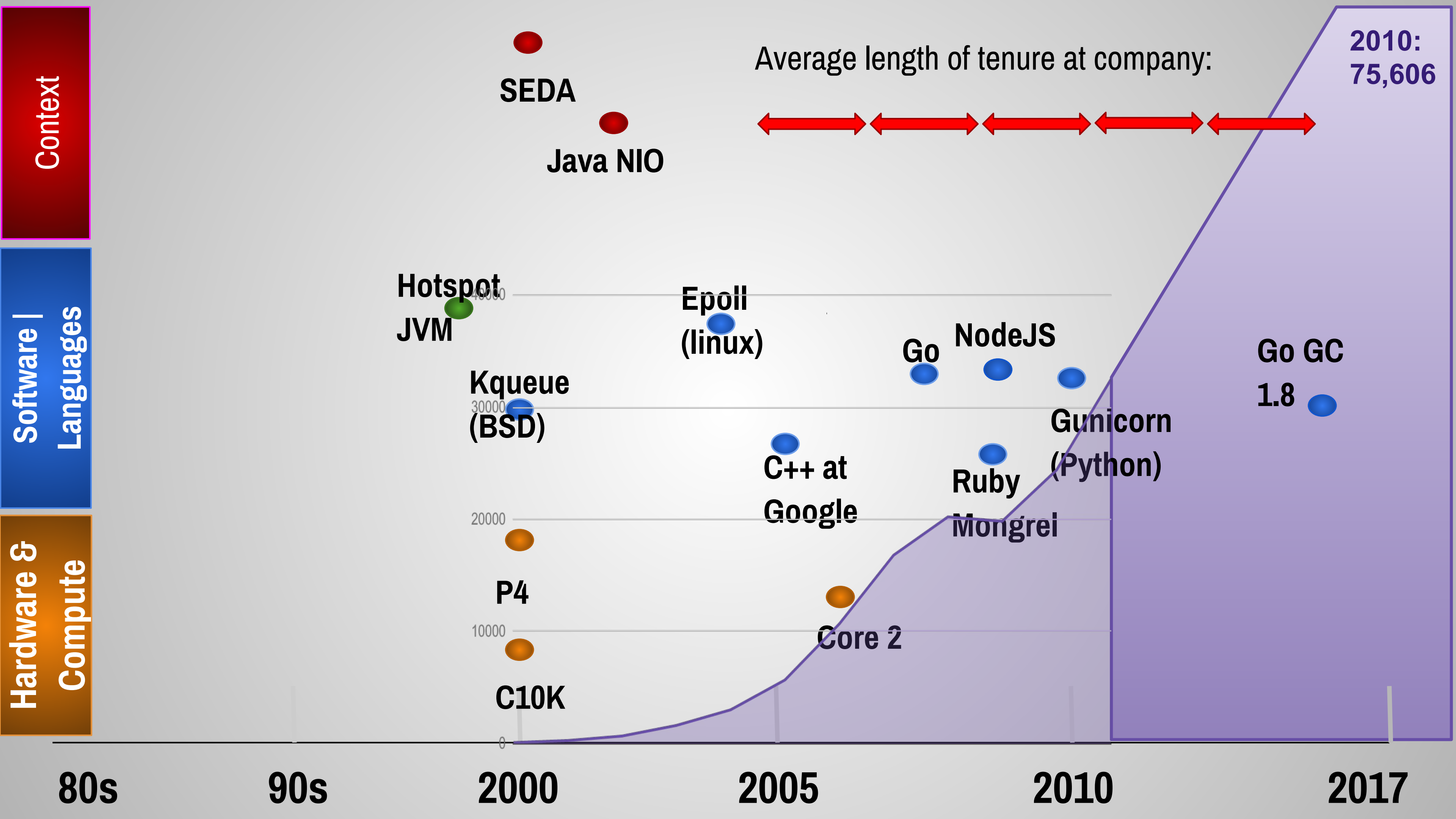
KEEP LEFT
NO ENTRY
NO RIGHT TURN


GIVE WAY
SPEED HUMP
STOP
GO
30
DIVERSION

“Simplicity is a great virtue but it requires hard work to achieve it and education to appreciate it. And to make matters worse: complexity sells better.”

— Edsger W. Dijkstra

The Future



Context

Software | Languages

Hardware & Compute

SEDA

Java NIO

Average length of tenure at company:

2010: 75,606

Hotspot JVM

Epoll (linux)

Go

NodeJS

Go GC 1.8

Kqueue (BSD)

Gunicorn (Python)

C++ at Google

Ruby Mongrel

P4

Core 2

C10K

Context

Software | Languages

Hardware & Compute

The Future?

Context

Software | Languages

Hardware & Compute

The Future?

The problems we have today were not there 20 years ago, nor will be problems we face 20 years from now.

2017 2020 2025 2030 2035 2040

Context

Software | Languages

Hardware & Compute

The Future?

...it may surprise you

2017 2020 2025 2030 2035 2040

Thank you!

Carmen Andoh @carmatrocity

QCon San Francisco

21st Century Languages Track

November 2017