The Why of Go v2

Brave New World Edition

GOTO Copenhagen 2018

Carmen Andoh

@carmatrocity

Bryan Cantrill
@bcantrill

Follow

How about a conference called "In Retrospect" in which presenters revisit talks they've given years prior -- and describe how their thinking has evolved since?

10:01 PM - 28 Jun 2018

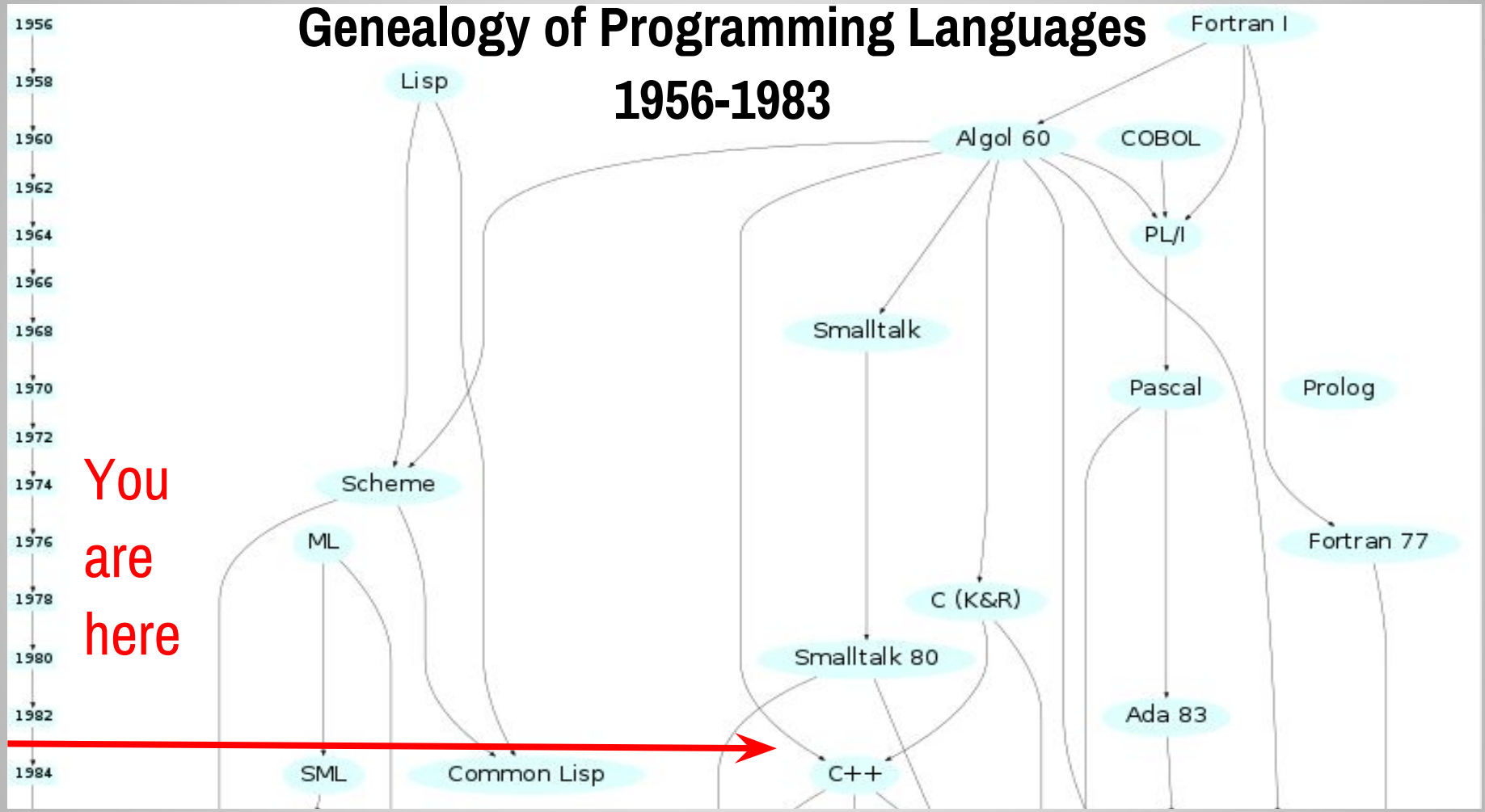
1,031 Retweets **5,696** Likes

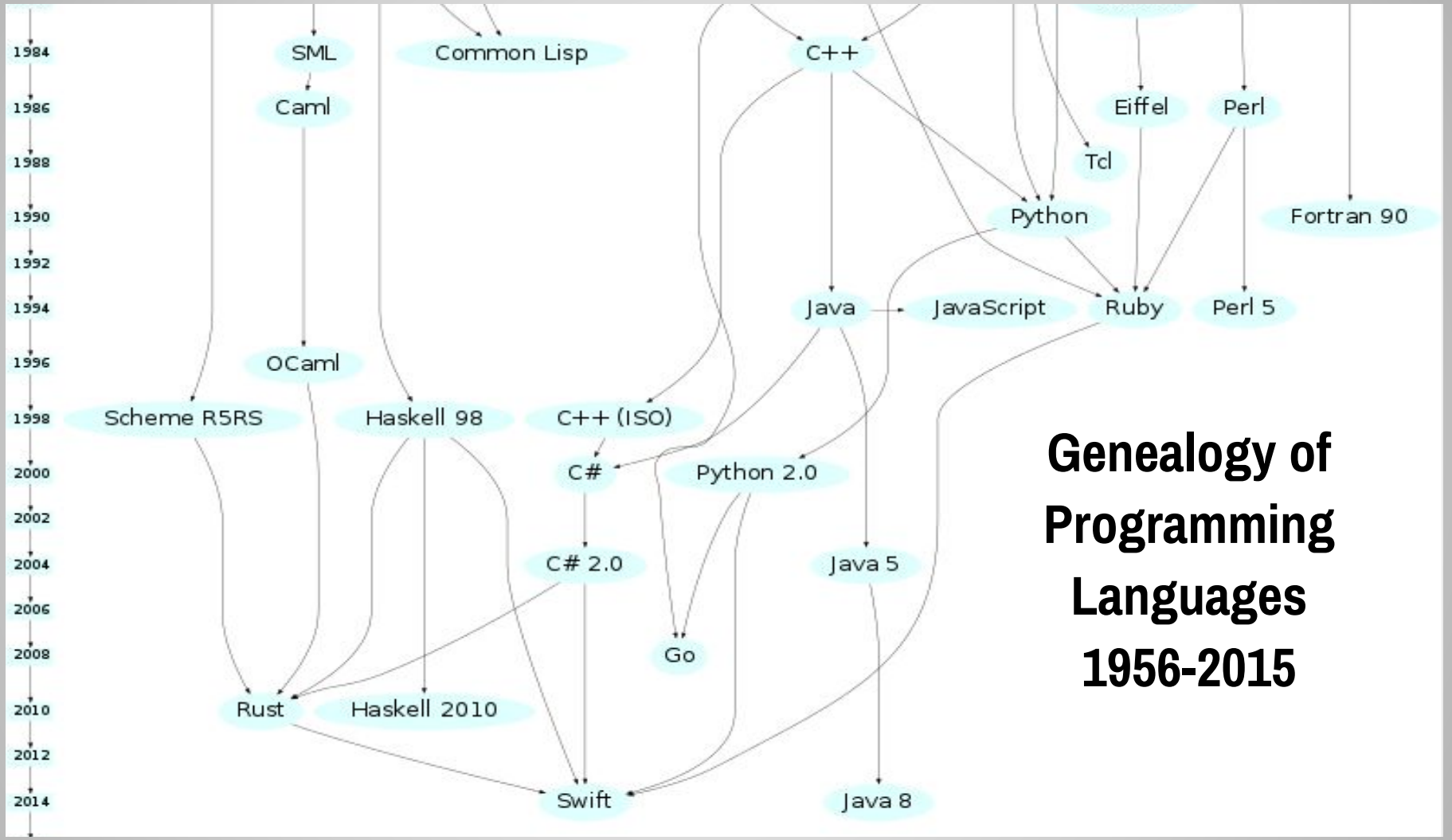
145 1.0K 5.7K

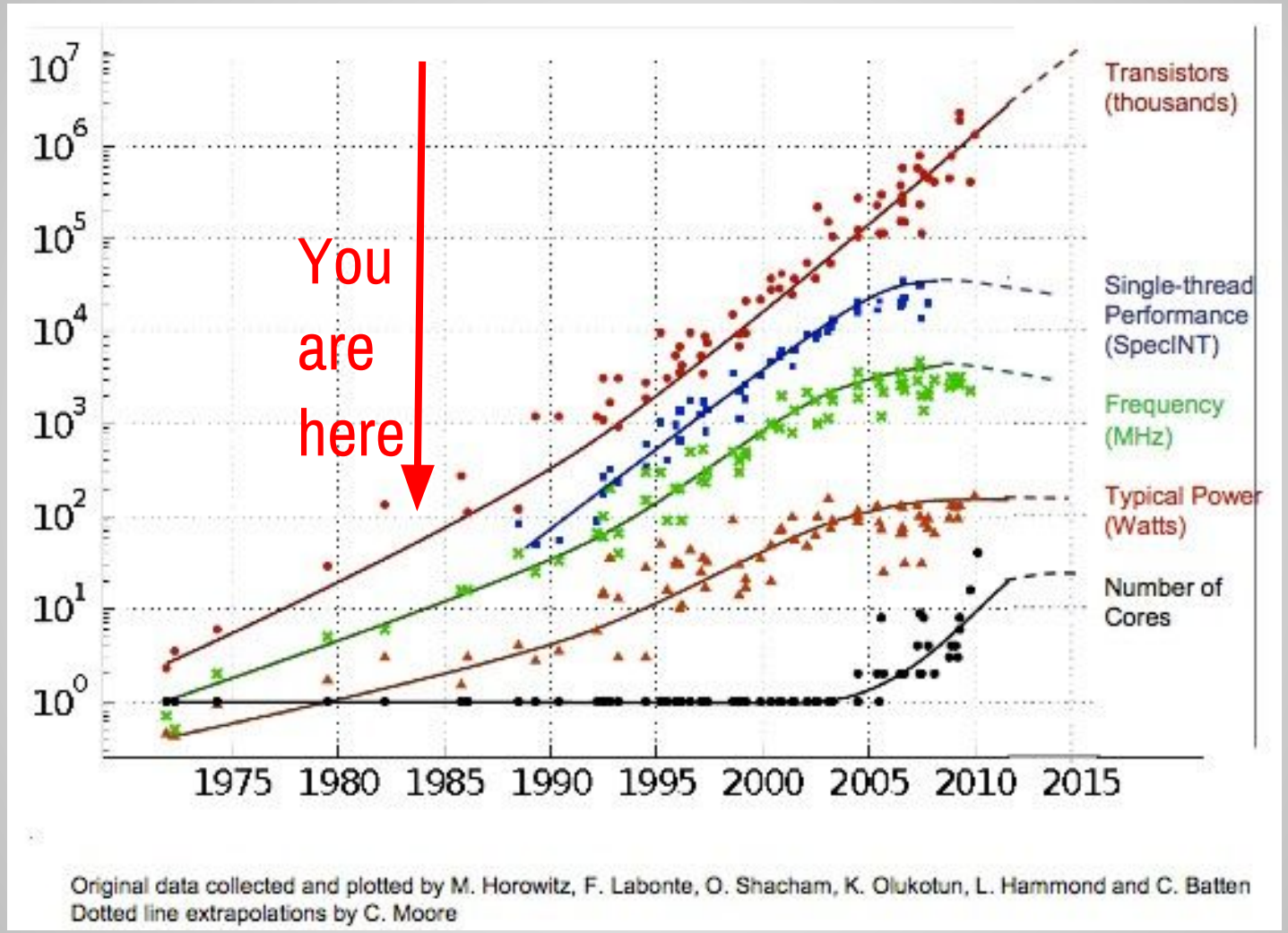
“Imagine you are a software engineer from the year 1983 who happened upon a time machine and traveled to today.”

(80s Dolorian time machines are the raddest time machines)

Genealogy of Programming Languages 1956-1983




1984
1986
1988
1990
1992
1994
1996
1998
2000
2002
2004
2006
2008
2010
2012
2014
SML
Caml
Common Lisp
C++
Eiffel
Perl
Tcl
Python
Fortran 90
Java
JavaScript
Ruby
Perl 5
OCaml
Scheme R5RS
Haskell 98
C++ (ISO)
C#
Python 2.0
C# 2.0
Java 5
Go
Rust
Haskell 2010
Swift
Java 8
Genealogy of Programming Languages 1956-2015


You are here
Transistors (thousands)
Single-thread Performance (SpecINT)
Frequency (MHz)
Typical Power (Watts)
Number of Cores
10^7
10^6
10^5
10^4
10^3
10^2
10^1
10^0
1975 1980 1985 1990 1995 2000 2005 2010 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

Unix®

Operating System

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

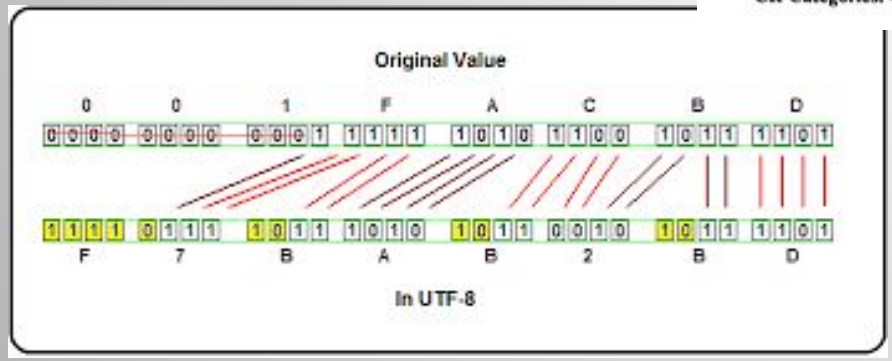
CR Categories: 4.20, 4.22, 4.32

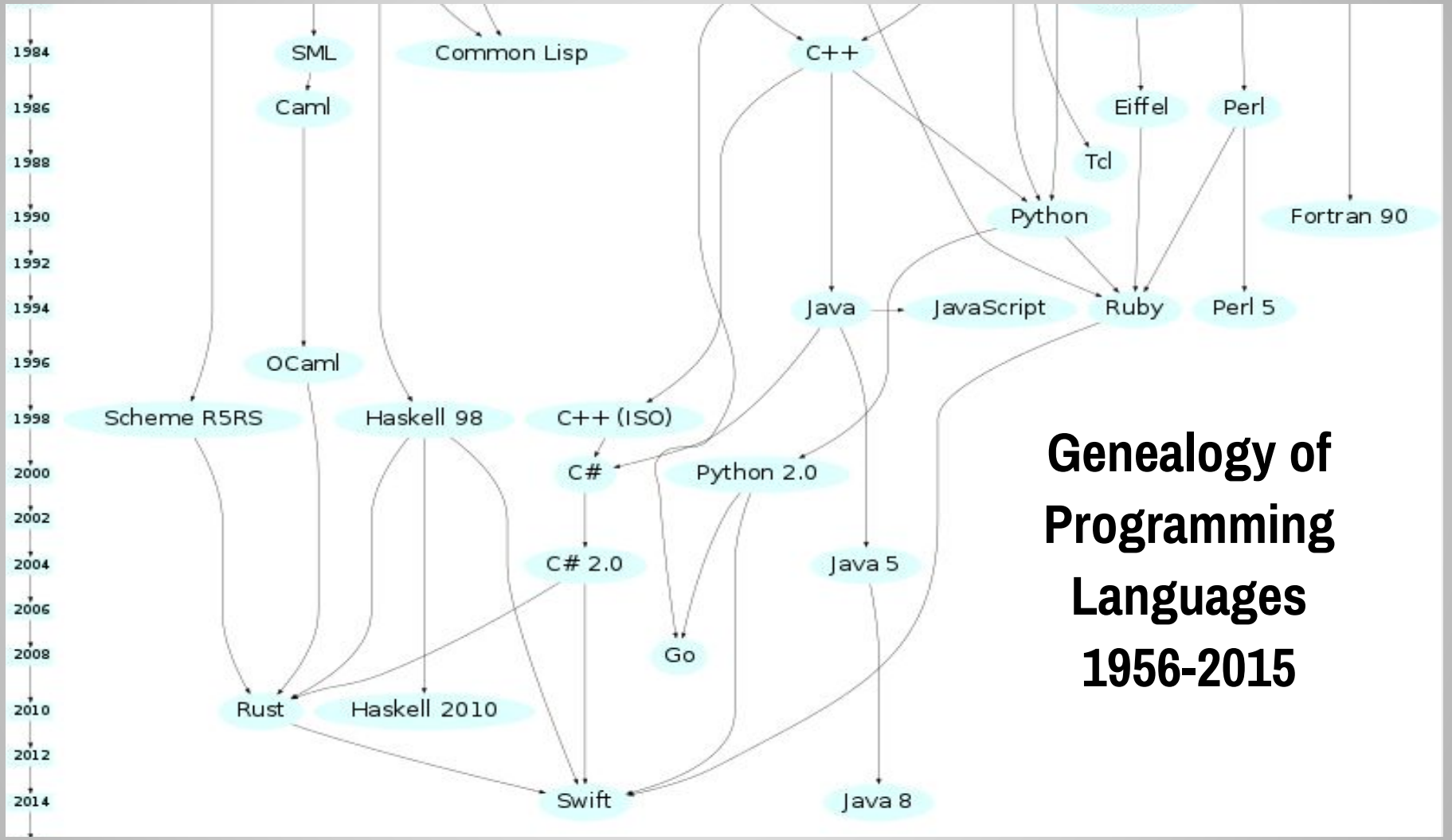
SECOND EDITION

THE C ANSI C PROGRAMMING LANGUAGE

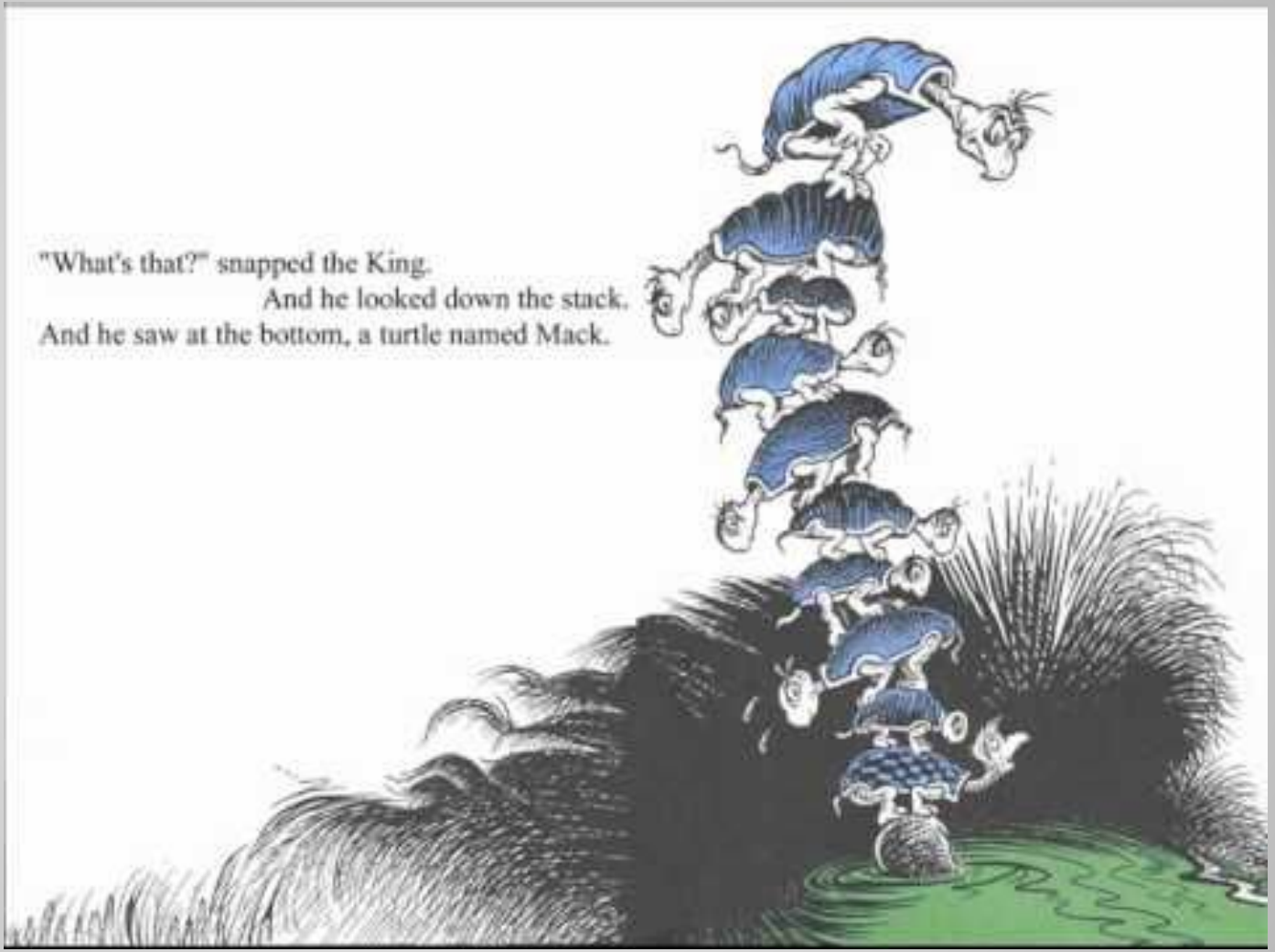
BRIAN W. KERNIGHAN
DENNIS M. RITCHIE

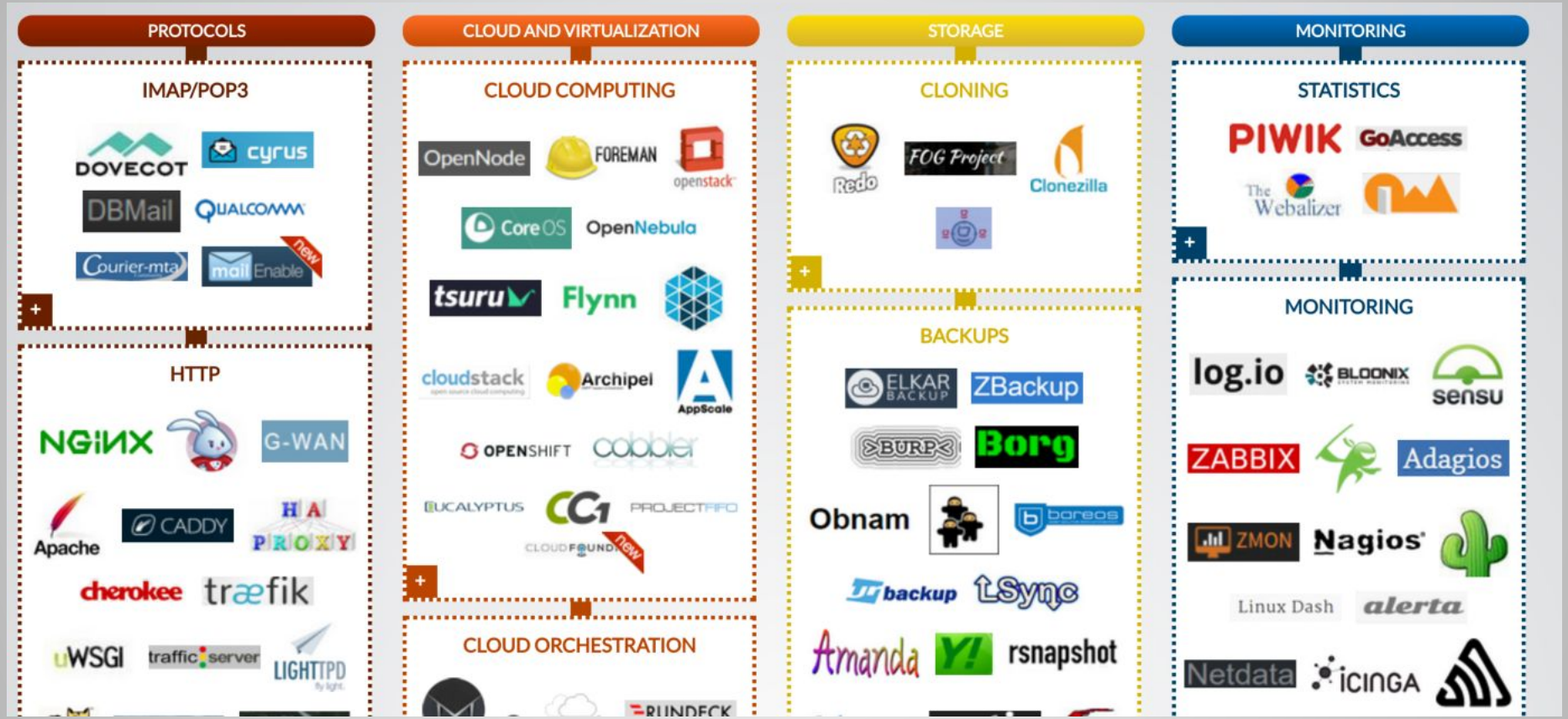
PRENTICE HALL SOFTWARE SERIES




1984
1986
1988
1990
1992
1994
1996
1998
2000
2002
2004
2006
2008
2010
2012
2014
SML
Caml
Common Lisp
C++
Eiffel
Perl
Tcl
Python
Fortran 90
Java
JavaScript
Ruby
Perl 5
OCaml
Scheme R5RS
Haskell 98
C++ (ISO)
C#
Python 2.0
C# 2.0
Java 5
Go
Rust
Haskell 2010
Swift
Java 8
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
DOVECOT
cyrus
DBMail
QUALCOMM
Courier-mta
mail Enable
new
+
HTTP
NGINX
G-WAN
Apache
CADDY
HAPROXY
cherokee
træfik
uWSGI
traffic server
LIGHTTPD
CLOUD AND VIRTUALIZATION
CLOUD COMPUTING
OpenNode
FOREMAN
openstack
CoreOS
OpenNebula
tsuru
Flynn
cloudstack
Archipel
AppScale
OPENSHIFT
cobbler
EUCALYPTUS
CC1
PROJECTFIFO
CLOUDFOUNDRY
new
+
CLOUD ORCHESTRATION
RUNDECK
STORAGE
CLONING
Redo
FOG Project
Clonezilla
+
BACKUPS
ELKAR BACKUP
ZBackup
BURP
Borg
Obnam
bareos
backup
tSync
Amanda
Y!
rsnapshot
MONITORING
STATISTICS
PIWIK
GoAccess
The Webalizer
+
MONITORING
log.io
BLOONIX
sensu
ZABBIX
Adagios
ZMON
Nagios
Linux Dash
alerta
Netdata
ICINGA


PROTOCOLS
IMAP/POP3
DOVECOT
cyrus
DBMail
QUALCOMM
Courier-mta
mail Enable
HTTP
NGINX
G-WAN
Apache
CADDY
HAPROXY
cherokee
træfik
uWSGI
traffic server
LIGHTTPD
Hiawatha
WildFly
tinkerbell
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
PROJECTPIFO
CLOUD FOUNDRY
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
XBURP
Borg
Obnam
bareos
backup
tSync
Amanda
rsnapshot
BackupPC
restic
duplicity
kvmBackup
SNEBU
REAR
SafeKeep
Attic
HashBackup
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
COCKPIT
Monitorix
Selena
Zenoss
SUPPORT SOFTWARE
CONTROL PANELS
COCKPIT
aj
zpanelcp
panamax
ISP manager
EasyDCP
webmin
ISPCONFIG
tipboard
nethserver
POWERDNSGUI
VESTA
CALAMARI
WebVirtMgr
Kimchi
ViMbAdmin
CWP
Froxlor
Easy-WI
Virtualmin
postfixadmin
OpenVZ Web Panel
iF.SVNAdmin
WebSVN
KeyHelp
SPACEWALE
WEBMAILS
modoboa
RAINLOOP
mailpile
CITADEL
ESSENTIALS
EDITORS
neovim
{coder}
ATOM
Brackets
jotgit
NetBeans
Sublime Text
eclipse
Emacs
GNU nano
gedit
REPOSITORIES
Dotdeb
ELRepo
software COLLECTIONS
EPEL
IUS
RepoForge
SCHMANAGER


PROTOCOLS
IMAP/POP3
Dovecot
Courier
Cyrus
MDaemon
Kerio
HTTP
NGINX
G-WAN
Apache
CACKE
Varnish
Cherokee
traefik
lighttpd
Hiawatha
Resin
SMTP
Postfix
exim
Haraka
Mailtrain
Sendmail
qmail
VBox.Adm
ZoneMTA
OSSP
DNS
PowerDNS
PowerDNS.rs
NSD
Knot
BIND
Unbound
Dnsmasq
djbdns
tinydns
gdnsd
LDAP
FUSION DIRECTORY
OpenLDAP
FreeIPA
389
OpenDJ
OpenDS
SSH
autossh
sshmux
Mosh
DSH
Dropbear SSH
lsshtunnel
VPN
FREE/S/WAN
SoftEther
pritunl
PeerVPN
OpenVPN
tinc
sshuttle
CLOUD AND VIRTUALIZATION
CLOUD COMPUTING
OpenStack
CloudStack
OpenNebula
Eucalyptus
Flynn
Deis
Cloudbreak
Apcera
OPENSHIFT
CLOUDIFY
Cloud Foundry
CLOUD ORCHESTRATION
Overcast
RUNDECK
ebash
juju
Rocket
CloudGlue
Crowbar
Mistral
CLOUD STORAGE
Pydio
SeaFile
ownCloud
Tahoe-LAFS
VIRTUALIZATION
oGanes
PROXMOX
Xen
KVM
orkt
rkt
vSphere
SOFTWARE CONTAINERS
Docker
Rocket
OpenVZ
LXD
Nomad
MESSAGING
LOG MANAGEMENT
Kibana
Graylog
Fluentd
Heka
Logstash
QUEUEING
ZeroMQ
NSQ
ActiveMQ
Beanstalkd
Apollo
Kafka
BACKUPS
rdiff-backup
Bacula
Bareos
Obnam
Duplicity
Attic
SNEBU
restic
rsnapshot
Amanda
Borg
ZBackup
Duplicati
CLONING
Clonezilla
FOG Project
DISTRIBUTED FILESYSTEMS
Ceph
GlusterFS
DRBD
MogileFS
Lustre
XtreemFS
MooseFS
PERMS
FreeIPA
OpenLDAP
Samba
NoSQL
Redis
Riak
Cassandra
Hypertable
FlockDB
Neo4j
RethinkDB
MariaDB
MySQL
PostgreSQL
MONITORING
STATISTICS
PIWIK
GeoAccess
Webalizer
AWStats
MONITORING
log.io
Munin
Zabbix
Adagios
Nagios
Icinga
Cabot
OMD
Riemann
LibreNMS
MonitoRix
Selena
ZenCase
Cacti
Observium
Thruk
Naemon
METRIC AND METRIC COLLECTION
Collectd
Logstash
Graphite
Tessera
Cubism
Ganglia
Graphene
Collecti
Statsd
Kibana
Grafana
Cube
Oculus
Tasseo
Seyren
Riemann
Bosun
OpenTSDB
KairosDB
Prometheus
Diamond
AUTOMATION
CONFIGURATION MANAGEMENT
Ansible
Salt
Puppet
CFEngine
Chef
ara
Fabric
Capistrano
Otter
CONFIGURATION MANAGEMENT DATABASE
I-doit
ITop
SERVICE DISCOVERY
etcd
SUPPORT SOFTWARE
CONTROL PANELS
aj
ISP
Easy
VESTA
Kloxo
Webmin
Froxlor
cPanel
Plesk
WEBMAILS
CITADEL
horde
NEWSLETTERS
LibreMailer
PHPList
PROJECT MANAGEMENT
Redmine
Trac
GITLAB
TICKETING SYSTEMS
OTRS
IT ASSET MANAGEMENT
Snipe-IT
OCS
GLPI
WIKIS
MediaWiki
DokuWiki
Foswiki
Gollum
Ghost
Oleko
ikiwiki
TWiki
ESSENTIALS
EDITORS
Neovim
(coder)
Atom
Brackets
jotgit
Sublime Text
Brackets
REPOSITORIES
Dotdeb
EPEL
SECURITY
BlackBox
pass
Looking Glass
AULT
OSSEC
Kali Linux
Open AS
PacketFence
WinMTR
era
Rspamd
Sec
DenyHosts
IPFire
FusIVZ
OpenWrt
WAZUH
VERSION CONTROL
PACKAGING
fpm
Poky
TROUBLESHOOTING
ngrep
BOOKS
O'REILLY

GNU

GNU’s Not Unix Project

Started in 1983

Vague but exciting ...

CERN DD/OC

Tim Berners-Lee, CERN/DD

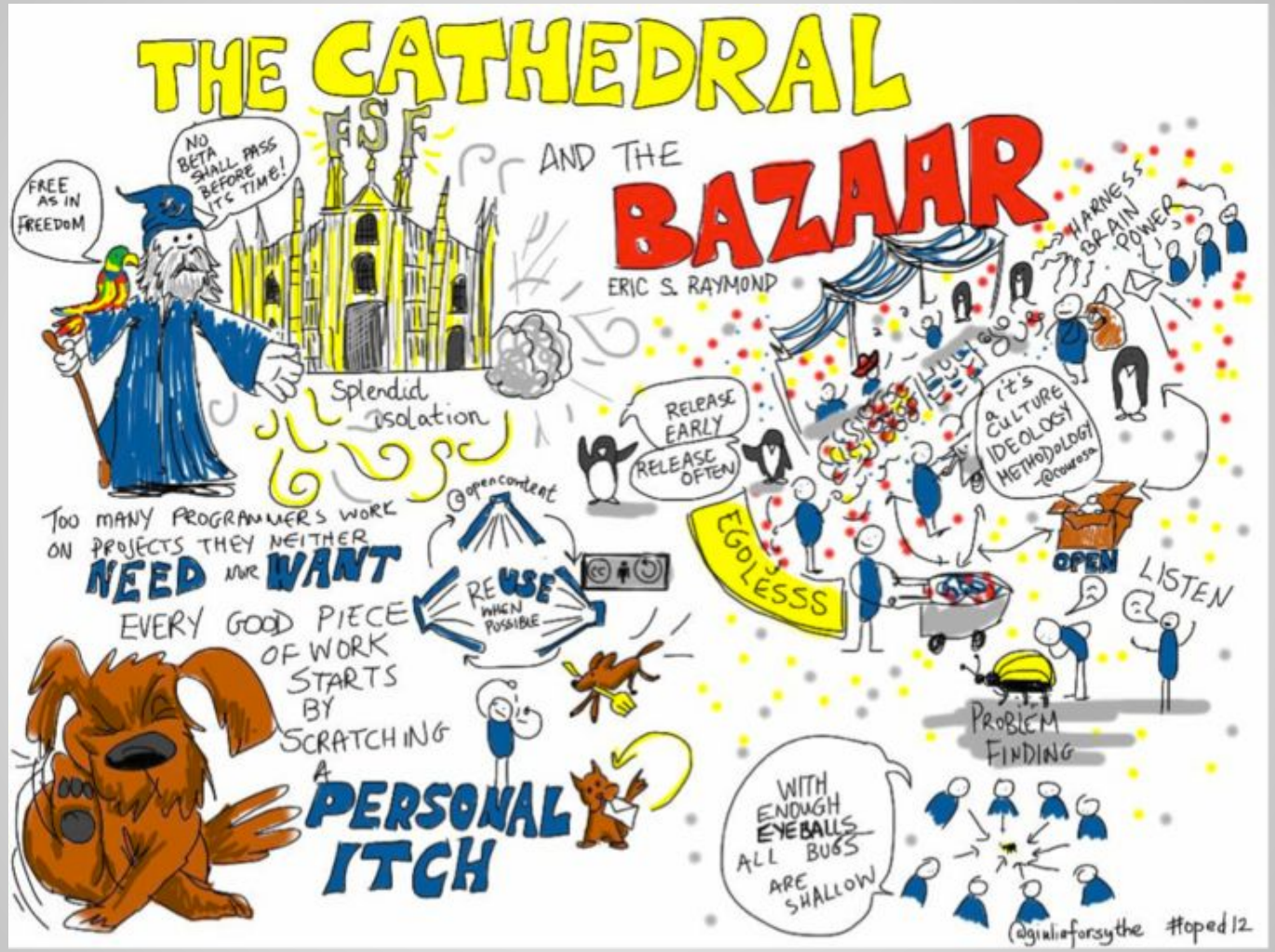
Information Management: A Proposal

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


THE CATHEDRAL
AND THE
BAZAAR
ERIC S. RAYMOND
FSF
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
RE USE WHEN POSSIBLE
EVERY GOOD PIECE OF WORK STARTS BY SCRATCHING A PERSONAL ITCH
WITH ENOUGH EYEBALLS ALL BUGS ARE SHALLOW
@gialiaforsythe #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.

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

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 opping the connection.

1950s

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

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.

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.

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

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 AN

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.

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.

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 opping the connection.

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

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.

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.

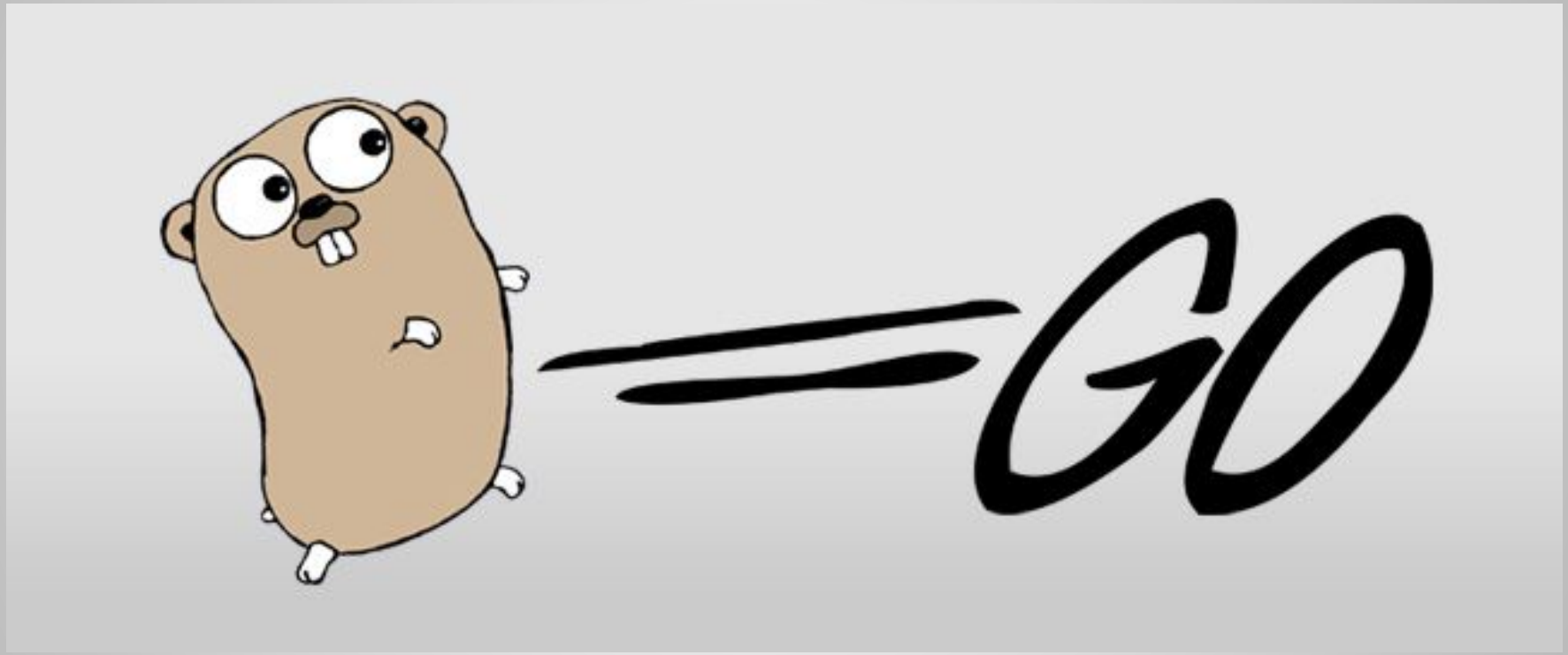
1978 First commercial cellular network Installed by Bell Labs in Chicago.

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.

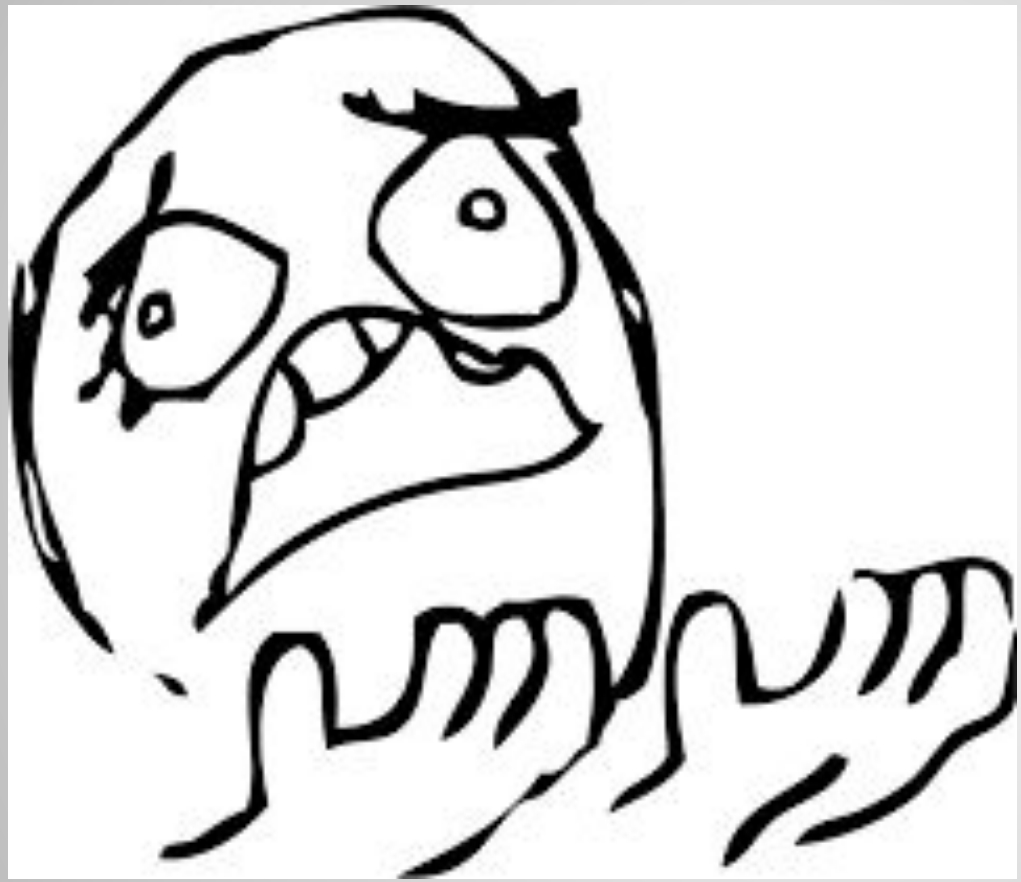
LEFT AND CENTER PHOTOS COURTESY OF ALCATEL-LUCENT USA INC. AND THE AT&T ARCHIVES AN


GO

Criticisms of Go

- 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.

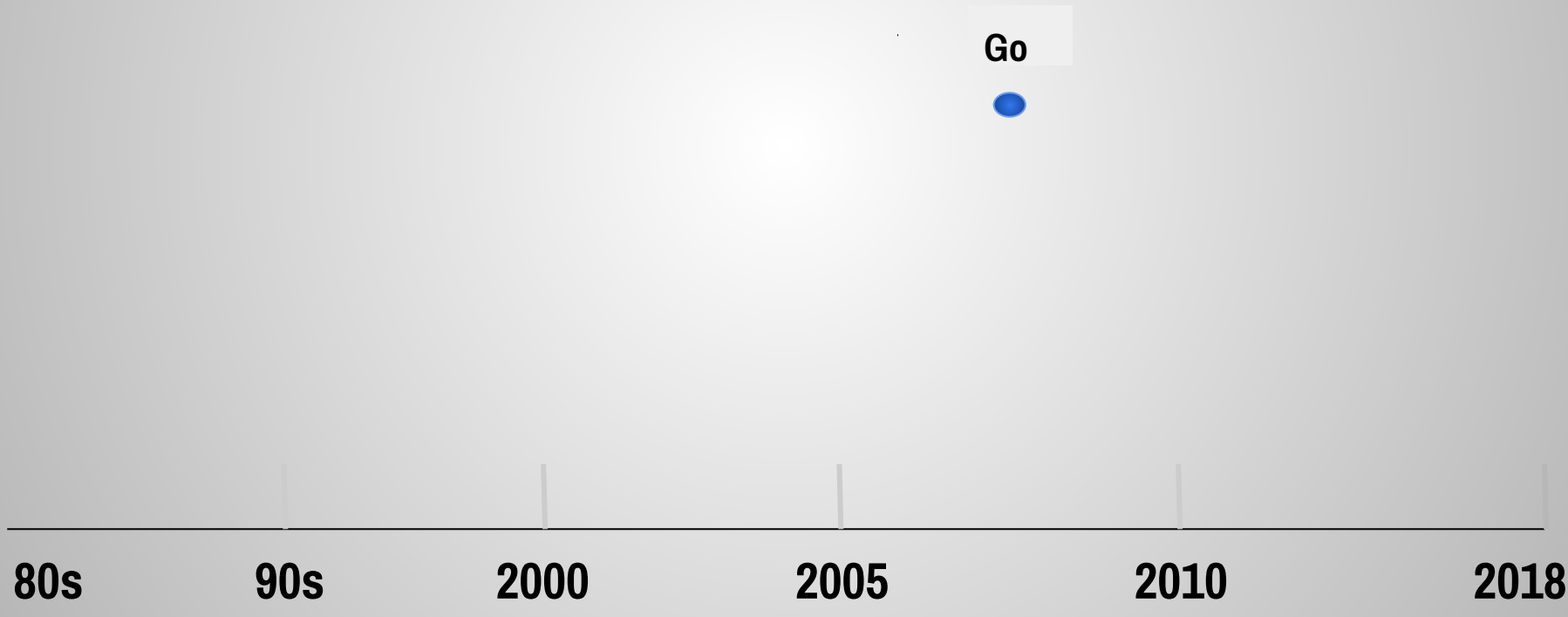
Moreover, the scale has changed: today's server programs comprise tens of millions of lines of code, are worked on by hundreds or even thousands of programmers, and are updated literally every day. To make matters worse, build time have stretched to many minutes, even hours, even languages,on large compilation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

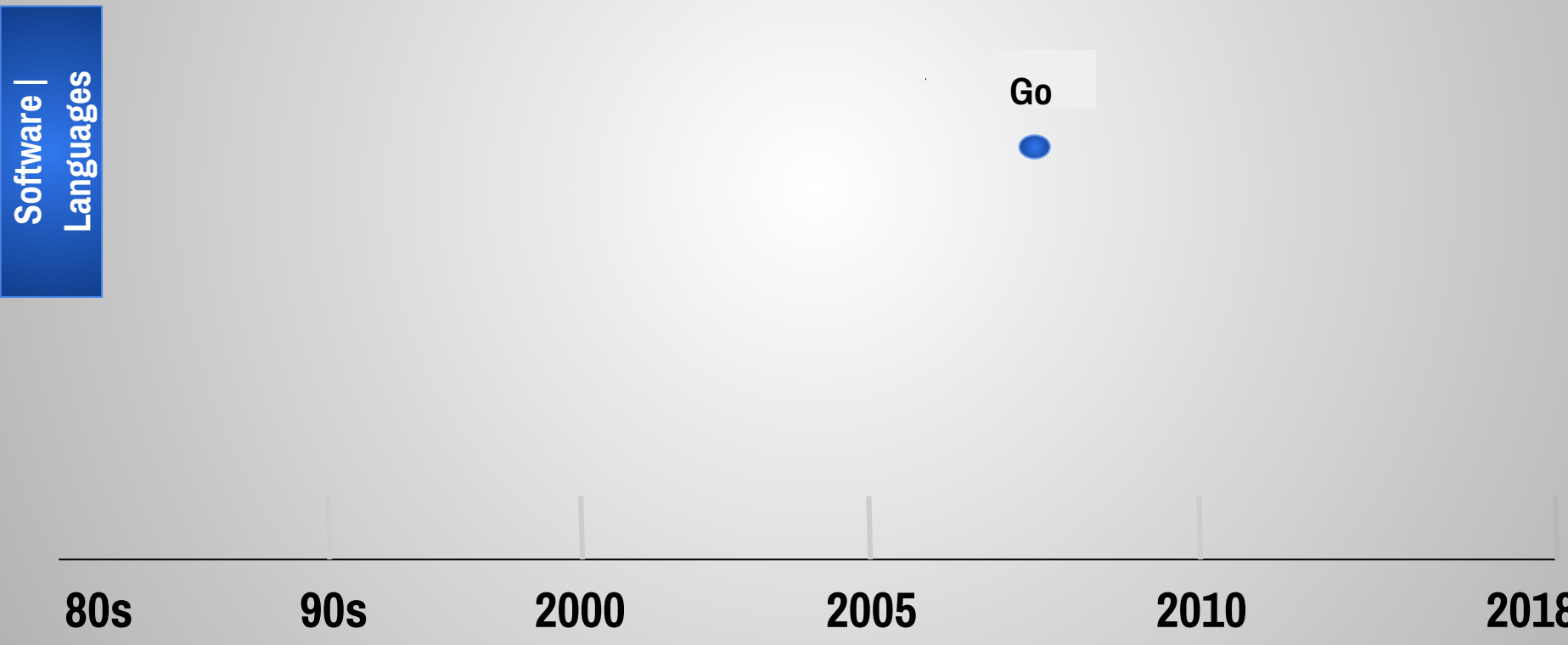
Moreover, the scale has changed: today's server programs comprise tens of millions of lines of code, are worked on by **hundreds or even thousands of programmers**, and are updated literally every day. To make matters worse, build time has stretched to many minutes, even hours, on large compilation clusters"

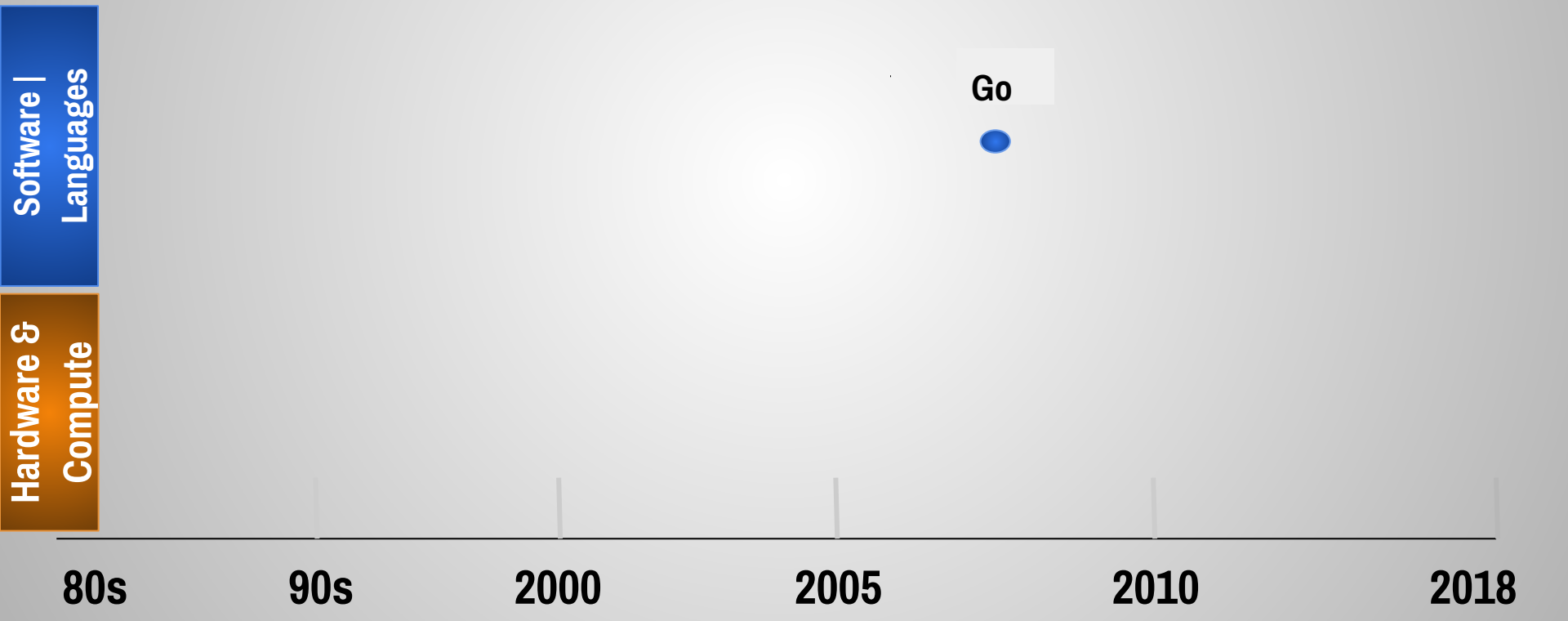
Moreover, the scale has changed: today's server programs comprise tens of millions of lines of code, are worked on by **hundreds or even thousands of programmers**, and are updated literally every day. To make matters worse, build time has stretched to many minutes, even hours, on **large compilation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

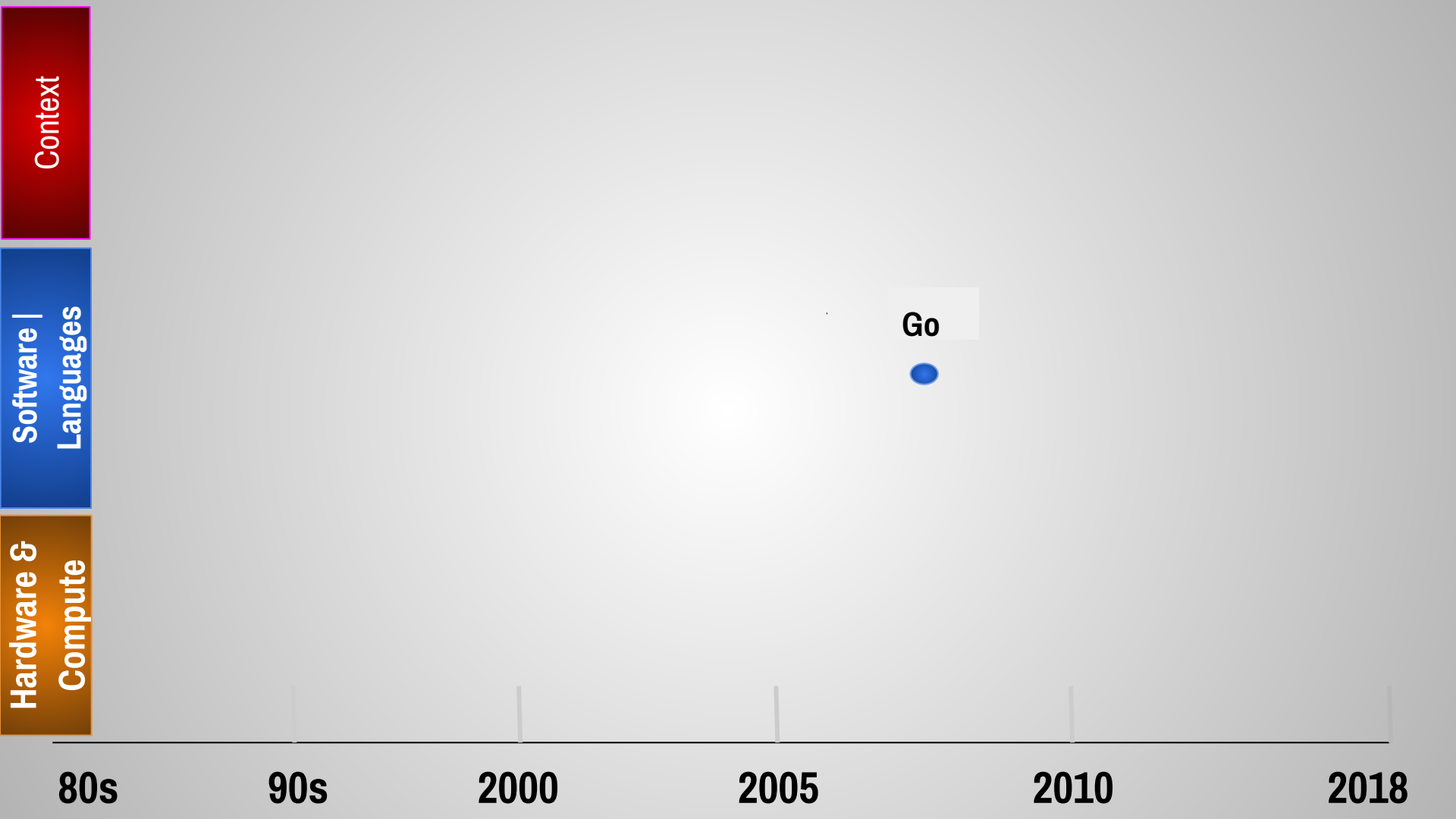
- multicore processors
- networked systems
- massive computation clusters
- web programming model

- hundreds or even thousands of programmers
- large compilation clusters


Go
80s
90s
2000
2005
2010
2018


Software | Languages
Go
80s
90s
2000
2005
2010
2018


Software | Languages
Hardware & Compute
Go
80s
90s
2000
2005
2010
2018


Context
Software | Languages
Hardware & Compute
Go
80s
90s
2000
2005
2010
2018

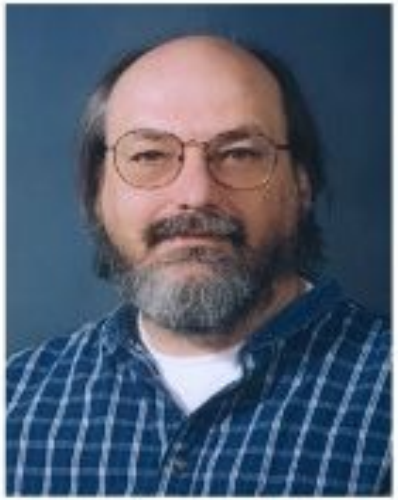
Designers

Robert Griesemer

V8 JavaScript engine, Java HotSpot VM

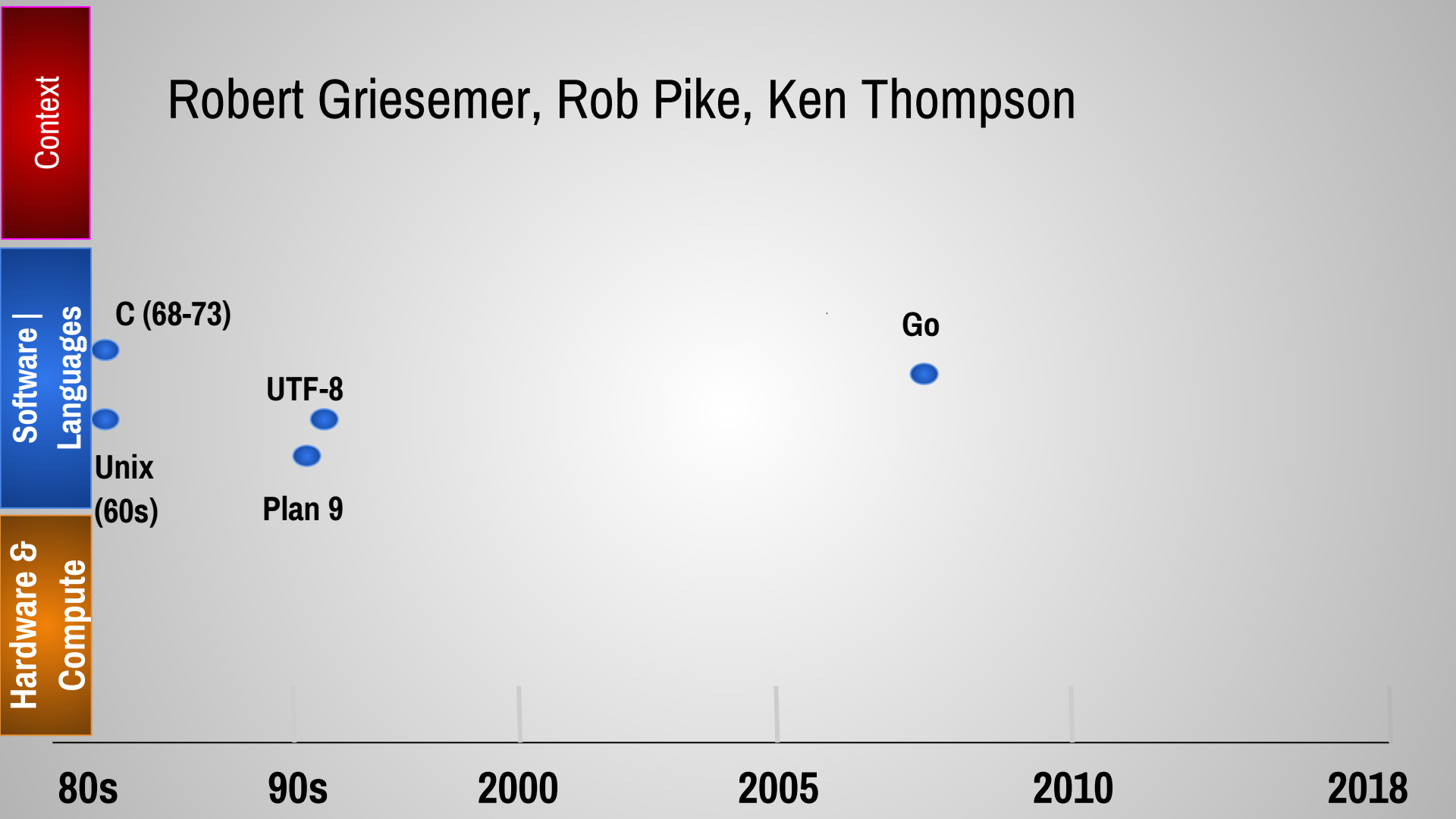
Rob Pike

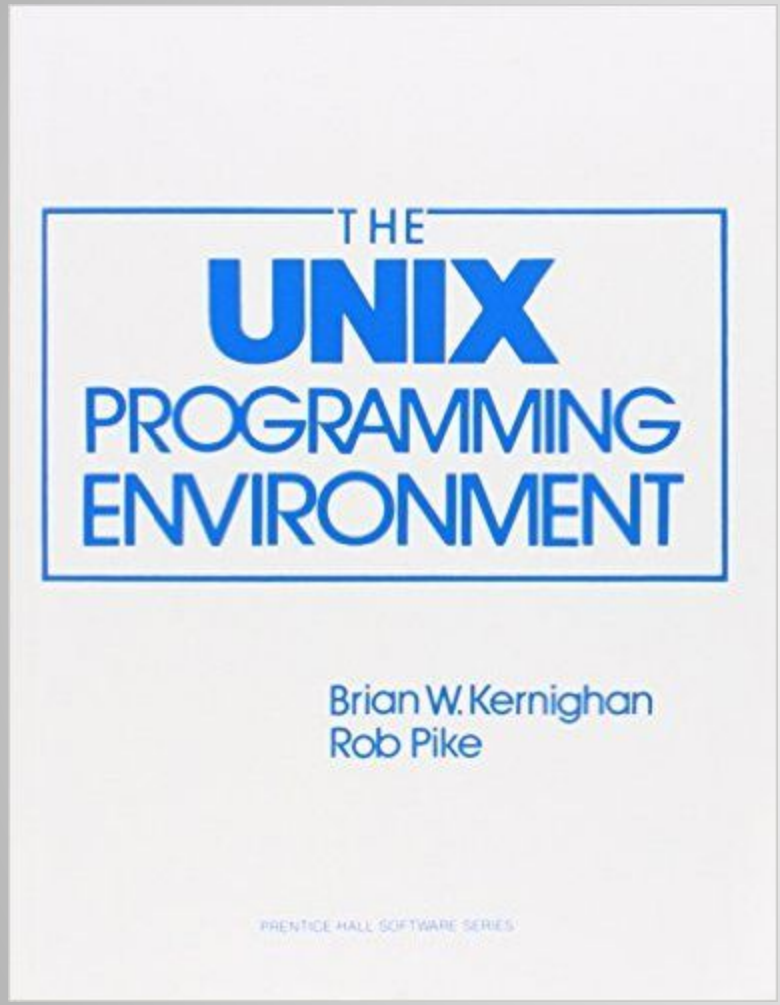
UNIX, Plan 9, UTF-8

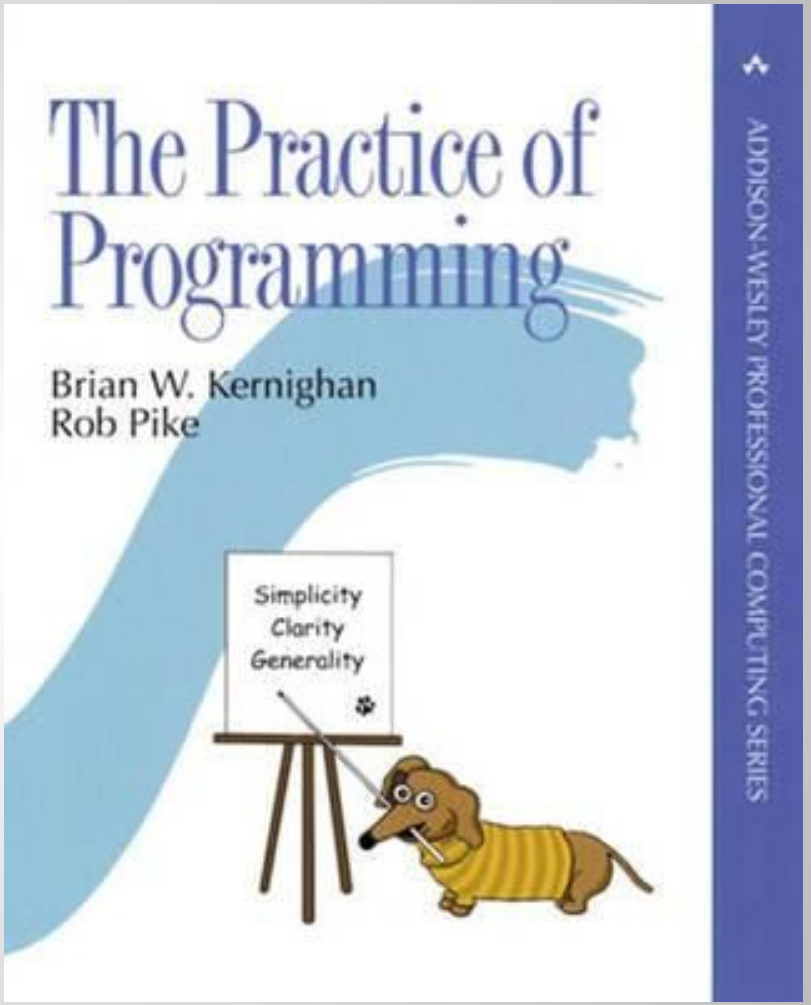


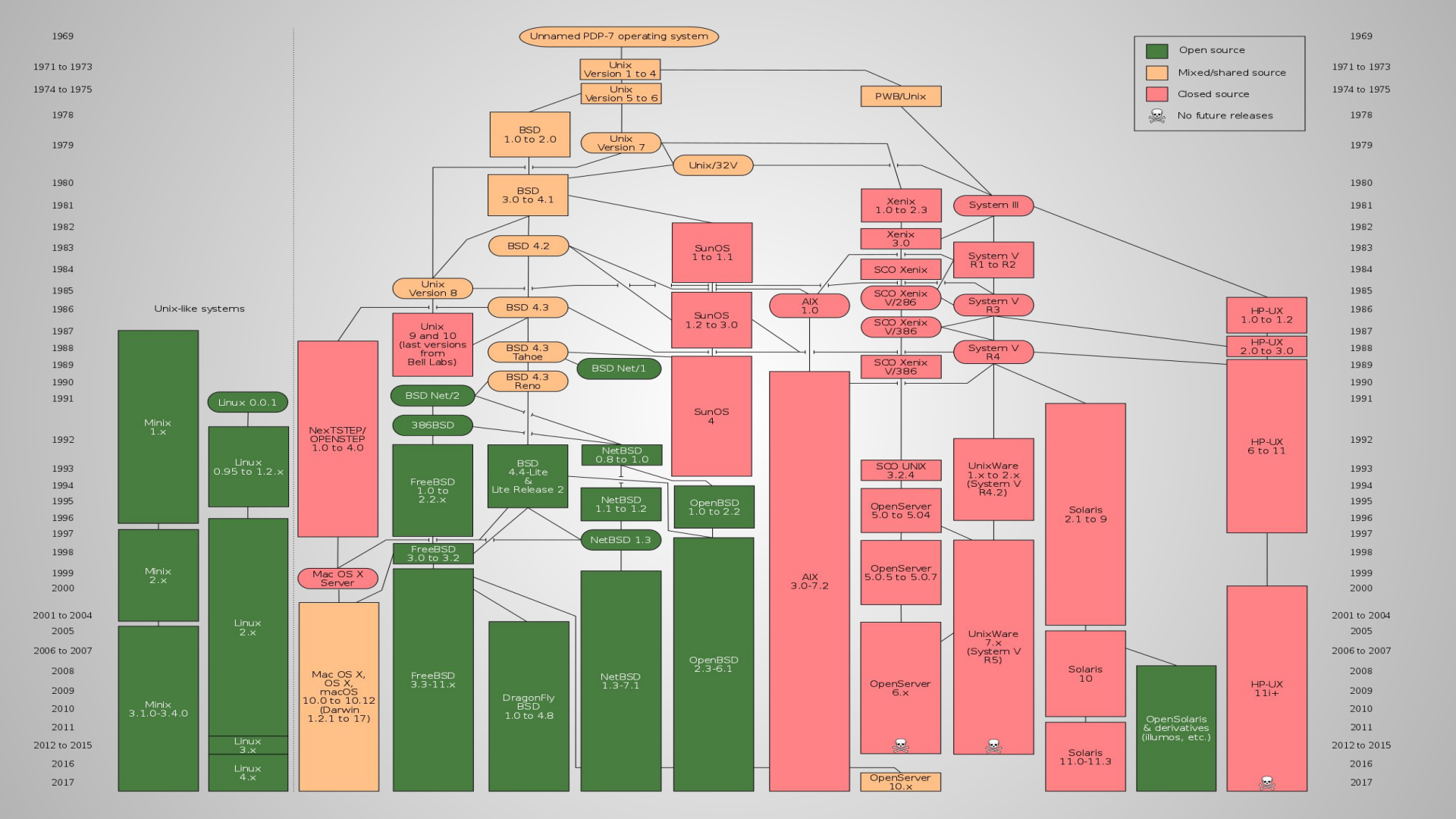
Ken Thompson

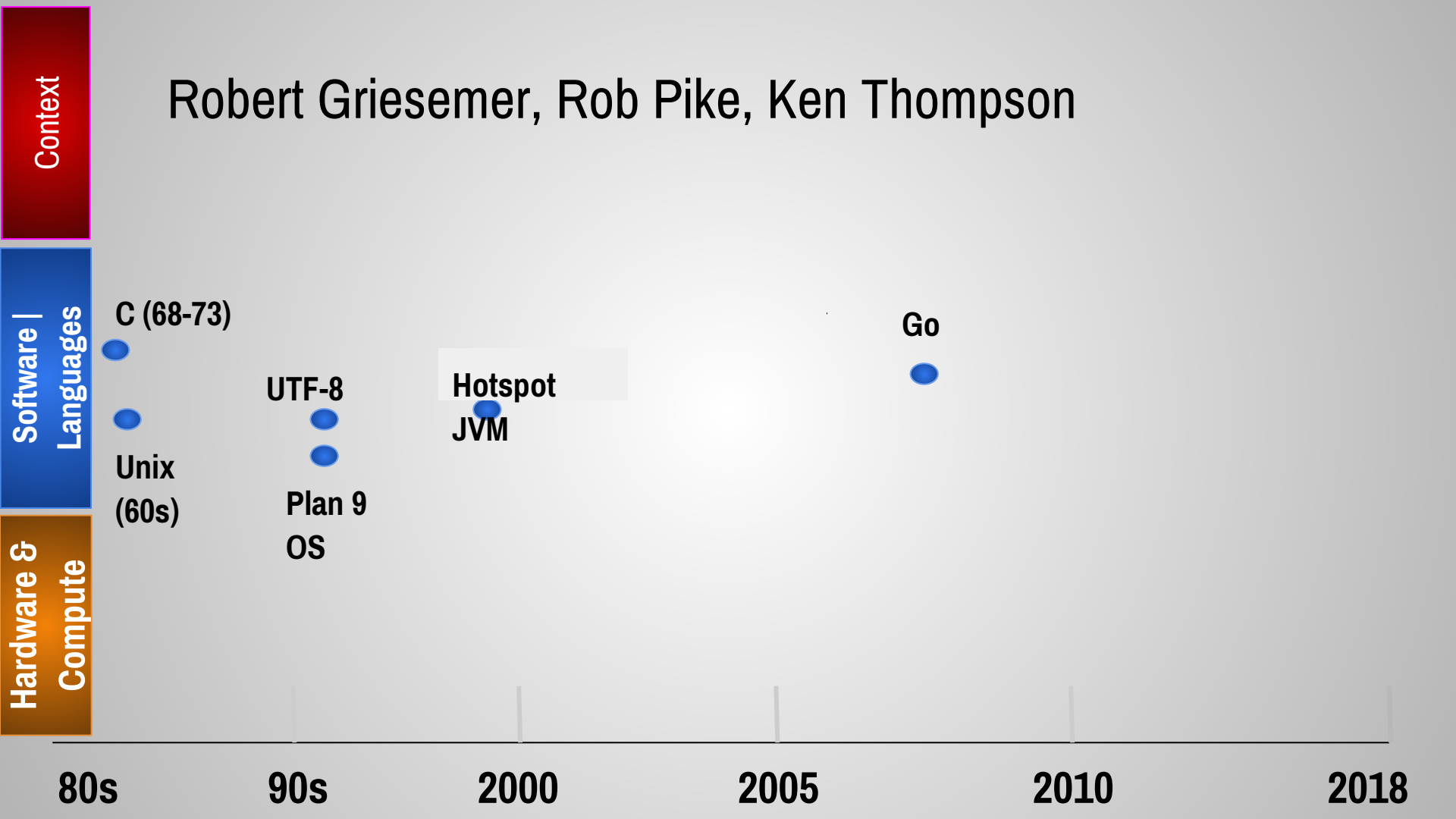
UNIX, Plan 9, B language, UTF-8

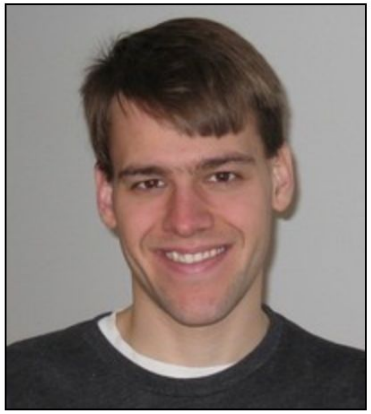

Context
Software | Languages
Hardware & Compute
Robert Griesemer, Rob Pike, Ken Thompson
C (68-73)
Unix (60s)
UTF-8
Plan 9
Go
80s
90s
2000
2005
2010
2018


THE
UNIX
PROGRAMMING
ENVIRONMENT
Brian W. Kernighan
Rob Pike
PRENTICE HALL SOFTWARE SERIES


The Practice of Programming
Brian W. Kernighan
Rob Pike
Simplicity
Clarity
Generality
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES


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
Unix-like systems
Minix 1.x
Minix 2.x
Minix 3.1.0-3.4.0
Linux 0.0.1
Linux 0.95 to 1.2.x
Linux 2.x
Linux 3.x
Linux 4.x
NexTSTEP/ OPENSTEP 1.0 to 4.0
Mac OS X Server
Mac OS X, OS X, macOS 10.0 to 10.12 (Darwin 1.2.1 to 17)
Unnamed PDP-7 operating system
Unix Version 1 to 4
Unix Version 5 to 6
Unix Version 7
Unix Version 8
Unix 9 and 10 (last versions from Bell Labs)
BSD 1.0 to 2.0
BSD 3.0 to 4.1
BSD 4.2
BSD 4.3
BSD 4.3 Tahoe
BSD 4.3 Reno
BSD Net/1
BSD Net/2
386BSD
FreeBSD 1.0 to 2.2.x
FreeBSD 3.0 to 3.2
FreeBSD 3.3-11.x
BSD 4.4-Lite & Lite Release 2
DragonFly BSD 1.0 to 4.8
NetBSD 0.8 to 1.0
NetBSD 1.1 to 1.2
NetBSD 1.3
NetBSD 1.3-7.1
Unix/32V
SunOS 1 to 1.1
SunOS 1.2 to 3.0
SunOS 4
OpenBSD 1.0 to 2.2
OpenBSD 2.3-6.1
PWB/Unix
AIX 1.0
AIX 3.0-7.2
Xenix 1.0 to 2.3
Xenix 3.0
SCO Xenix
SCO Xenix V/286
SCO Xenix V/386
SCO UNIX 3.2.4
OpenServer 5.0 to 5.04
OpenServer 5.0.5 to 5.0.7
OpenServer 6.x
OpenServer 10.x
System III
System V R1 to R2
System V R3
System V R4
UnixWare 1.x to 2.x (System V R4.2)
UnixWare 7.x (System V R5)
Solaris 2.1 to 9
Solaris 10
Solaris 11.0-11.3
OpenSolaris & derivatives (illumos, etc.)
HP-UX 1.0 to 1.2
HP-UX 2.0 to 3.0
HP-UX 6 to 11
HP-UX 11i+
Open source
Mixed/shared source
Closed source
No future releases


Context
Software | Languages
Hardware & Compute
Robert Griesemer, Rob Pike, Ken Thompson
C (68-73)
Unix (60s)
UTF-8
Plan 9 OS
Hotspot JVM
Go
80s
90s
2000
2005
2010
2018



Russ Cox

rsc@swtch.com

c/o Google
5 Cambridge Center
Cambridge, MA 02142

A C, an E-flat, and a G walk into a bar.
The bartender says, "Sorry, but we don't serve minors."

What's grey? A melted penguin.

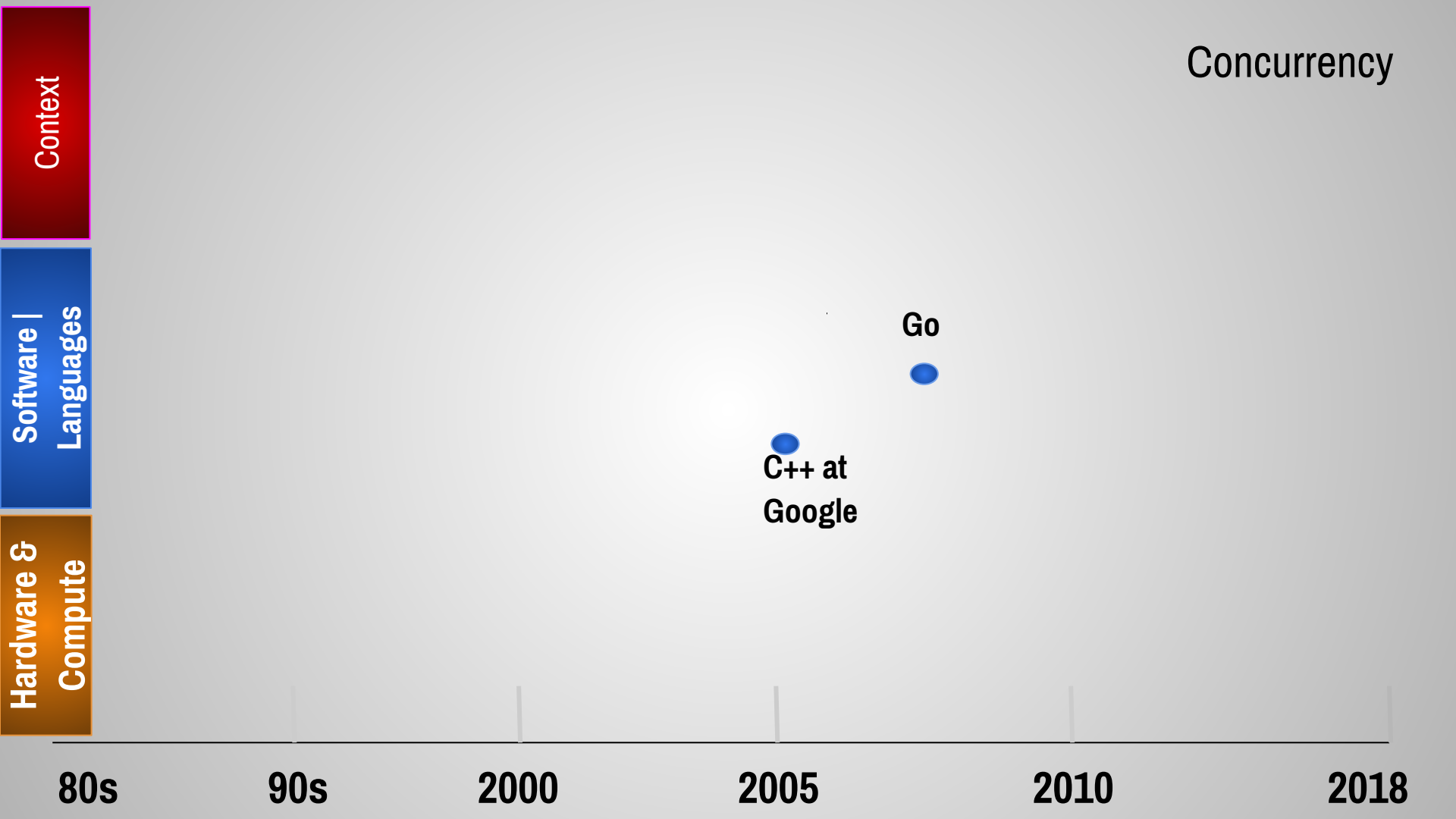
Go's 21st Century "5th Gen" Characteristics

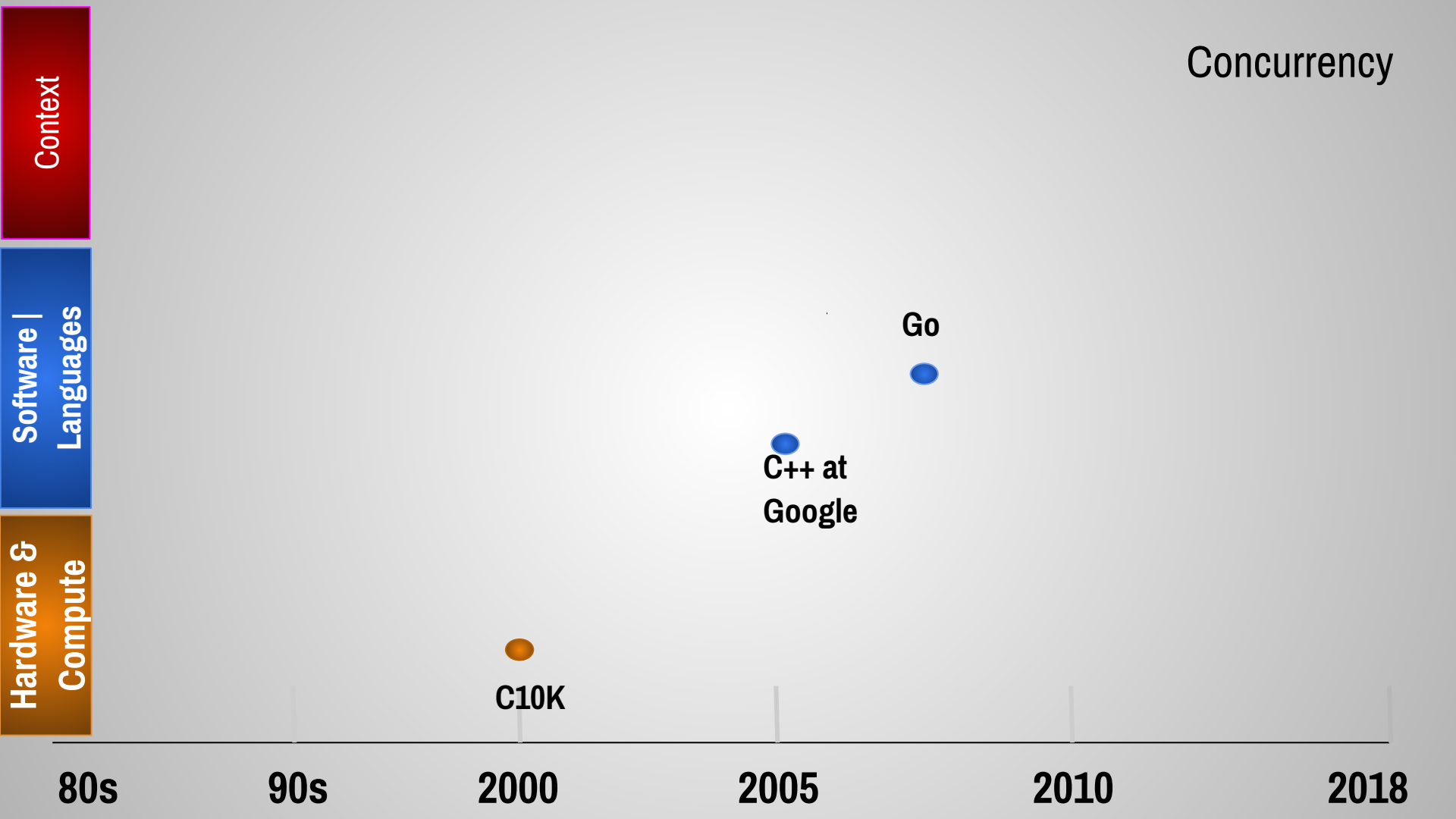
- Concurrency
- Distributed Systems
- Garbage Collection
- Memory and Data Locality
- Readability
- Simplicity

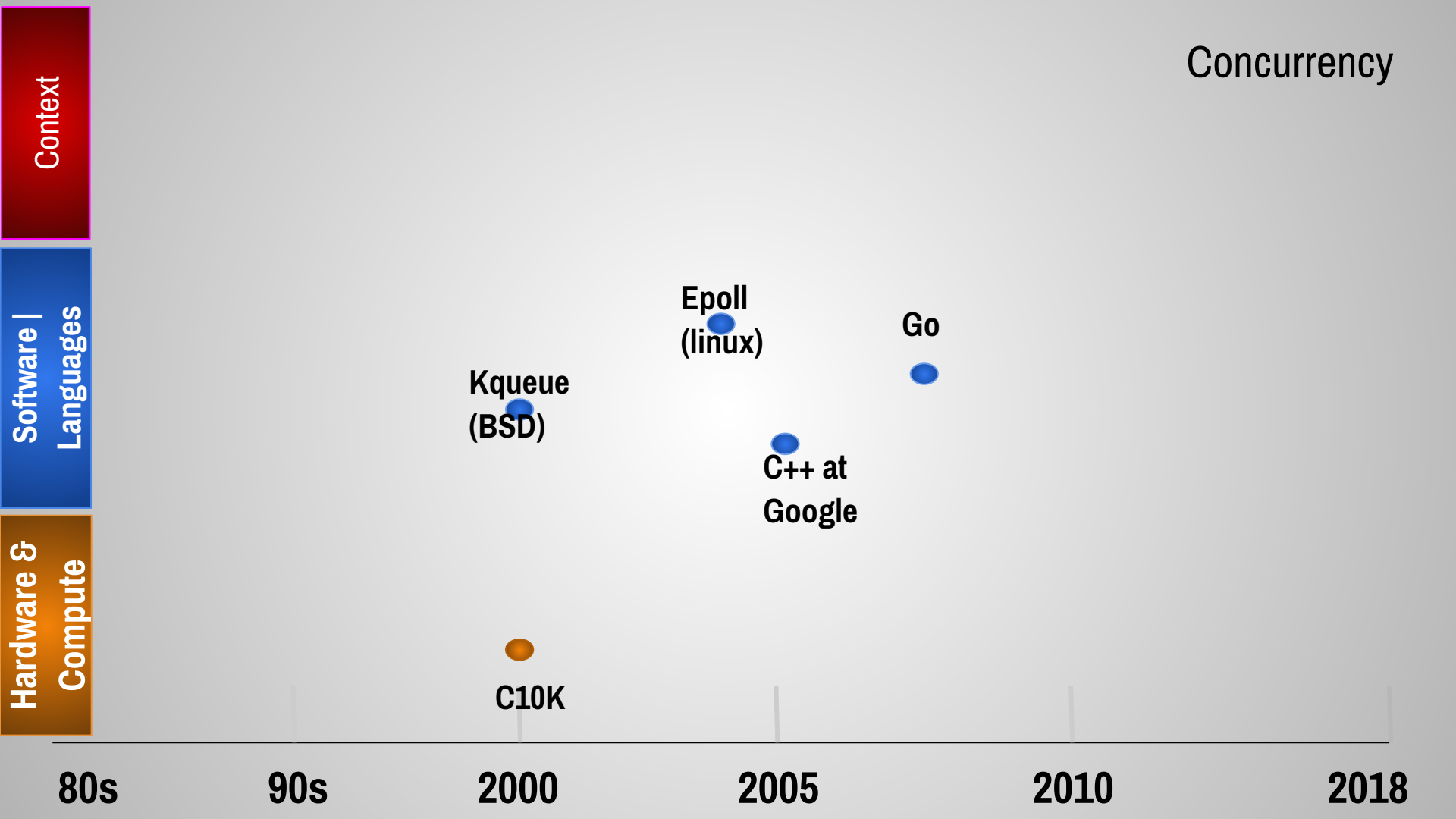
Concurrency

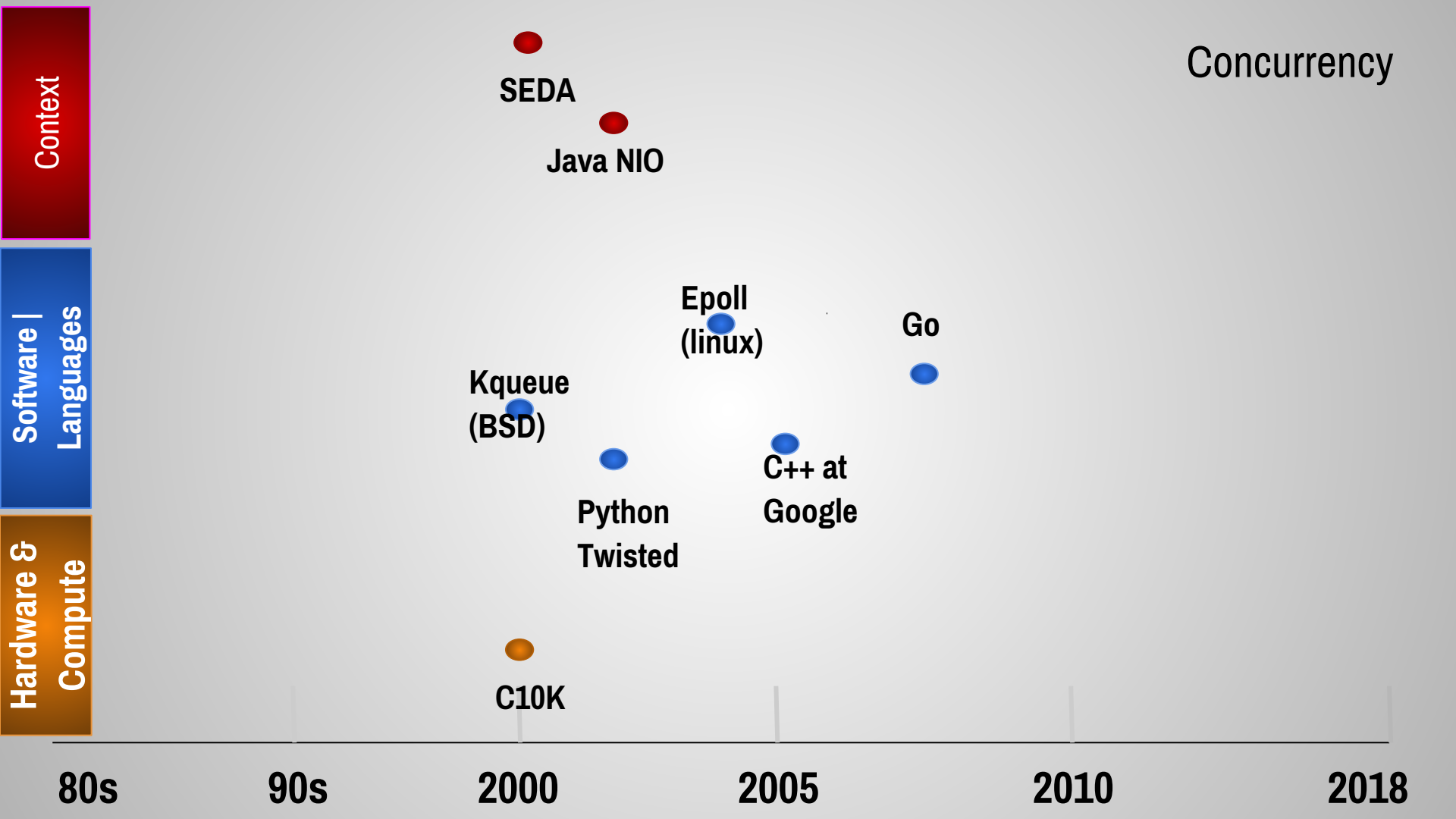
Why Massive Concurrency?

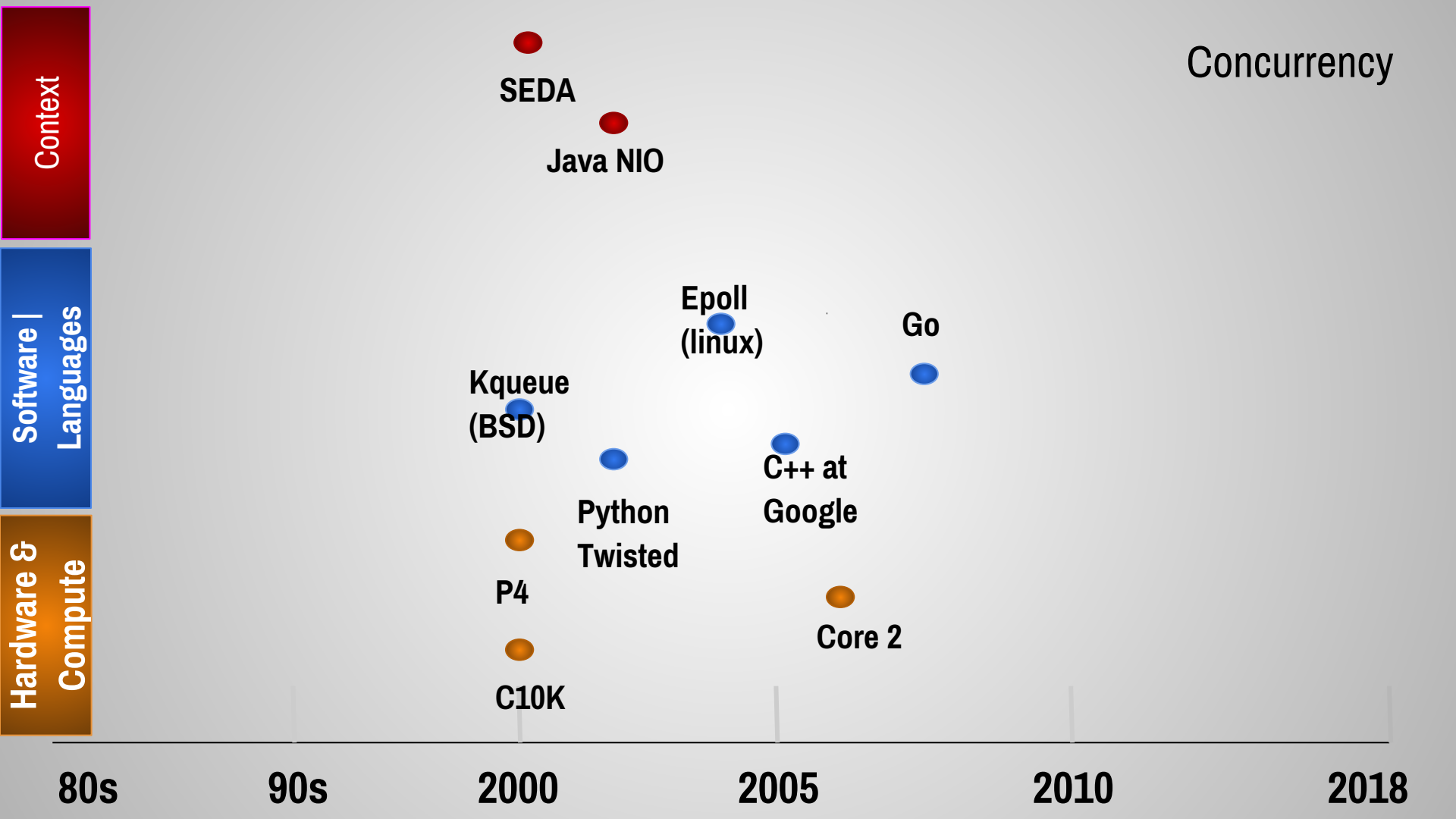
Why goroutines (green threads) ?

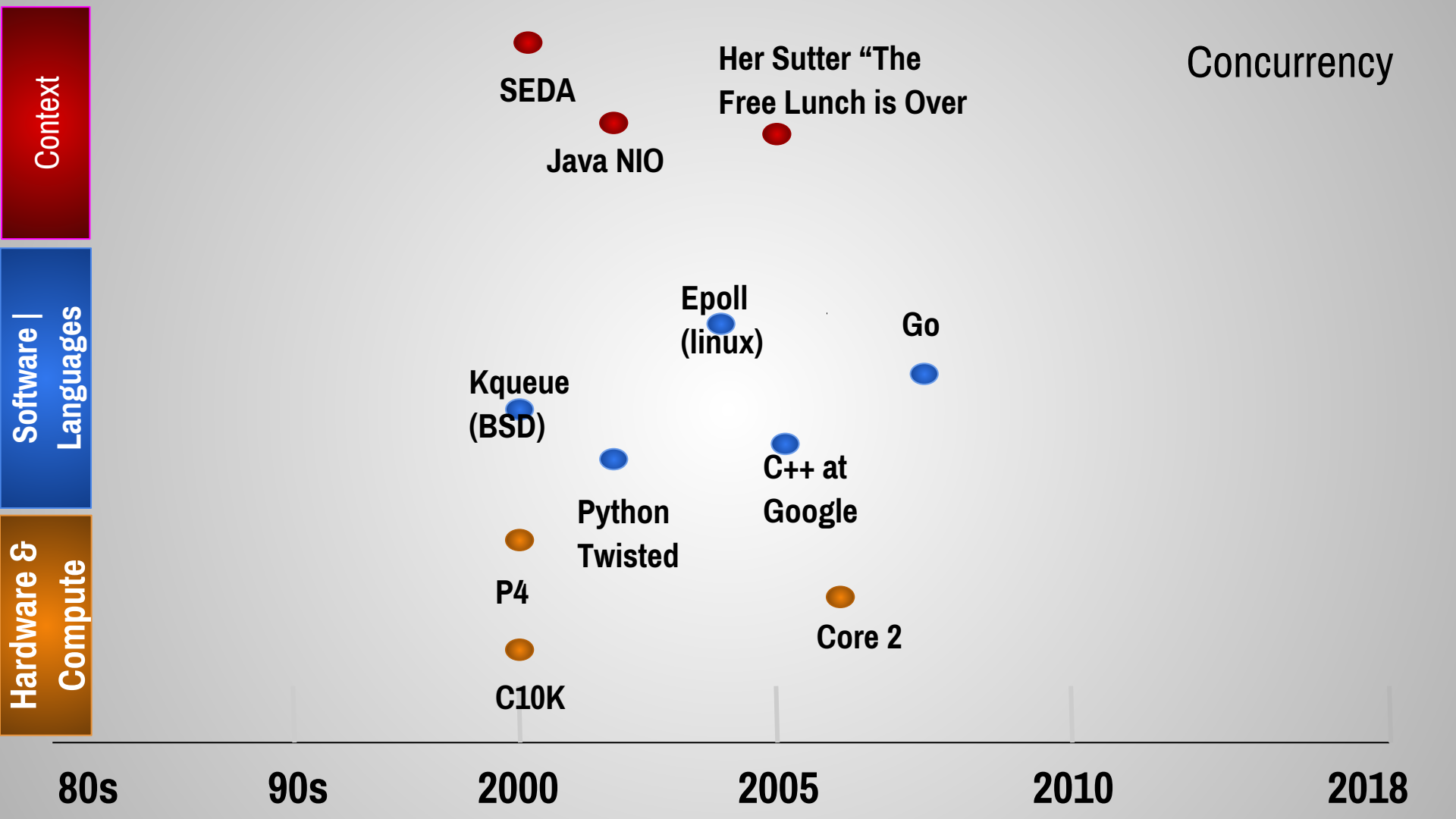

Context
Software | Languages
Hardware & Compute
Concurrency
Go
C++ at Google
80s
90s
2000
2005
2010
2018


Context
Software | Languages
Hardware & Compute
Concurrency
Go
C++ at Google
C10K
80s
90s
2000
2005
2010
2018


Context
Software | Languages
Hardware & Compute
Concurrency
Epoll (linux)
Go
Kqueue (BSD)
C++ at Google
C10K
80s
90s
2000
2005
2010
2018


Context
Software | Languages
Hardware & Compute
Concurrency
SEDA
Java NIO
Epoll (linux)
Go
Kqueue (BSD)
C++ at Google
Python Twisted
C10K
80s
90s
2000
2005
2010
2018


Context
Software | Languages
Hardware & Compute
Concurrency
SEDA
Java NIO
Epoll (linux)
Go
Kqueue (BSD)
C++ at Google
Python Twisted
P4
Core 2
C10K
80s
90s
2000
2005
2010
2018


Context
Software | Languages
Hardware & Compute
Concurrency
SEDA
Java NIO
Her Sutter "The Free Lunch is Over
Epoll (linux)
Go
Kqueue (BSD)
C++ at Google
Python Twisted
P4
Core 2
C10K
80s
90s
2000
2005
2010
2018

The Concurrency Revolution

By Herb Sutter, February 01, 2005

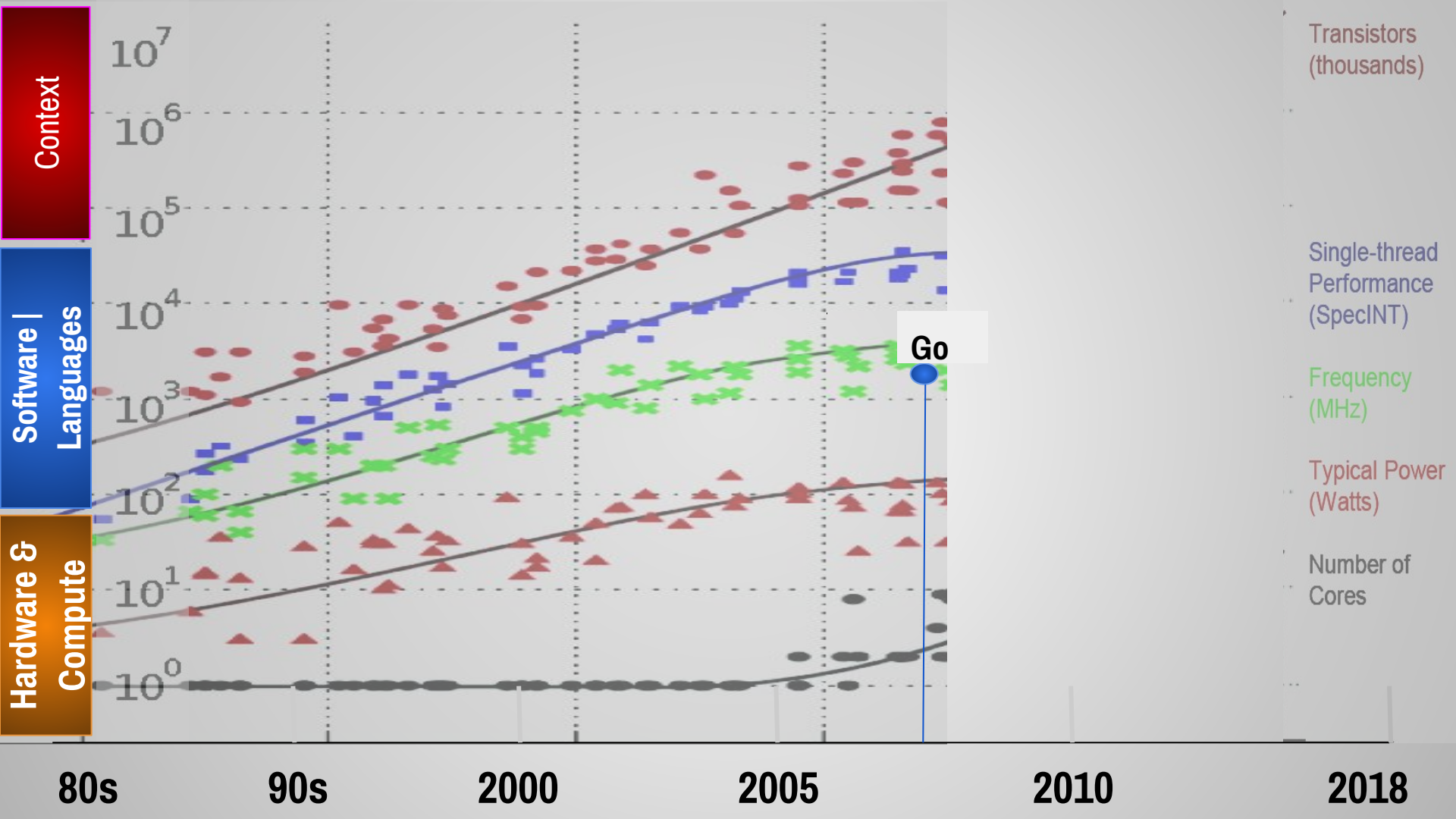
Post a Comment

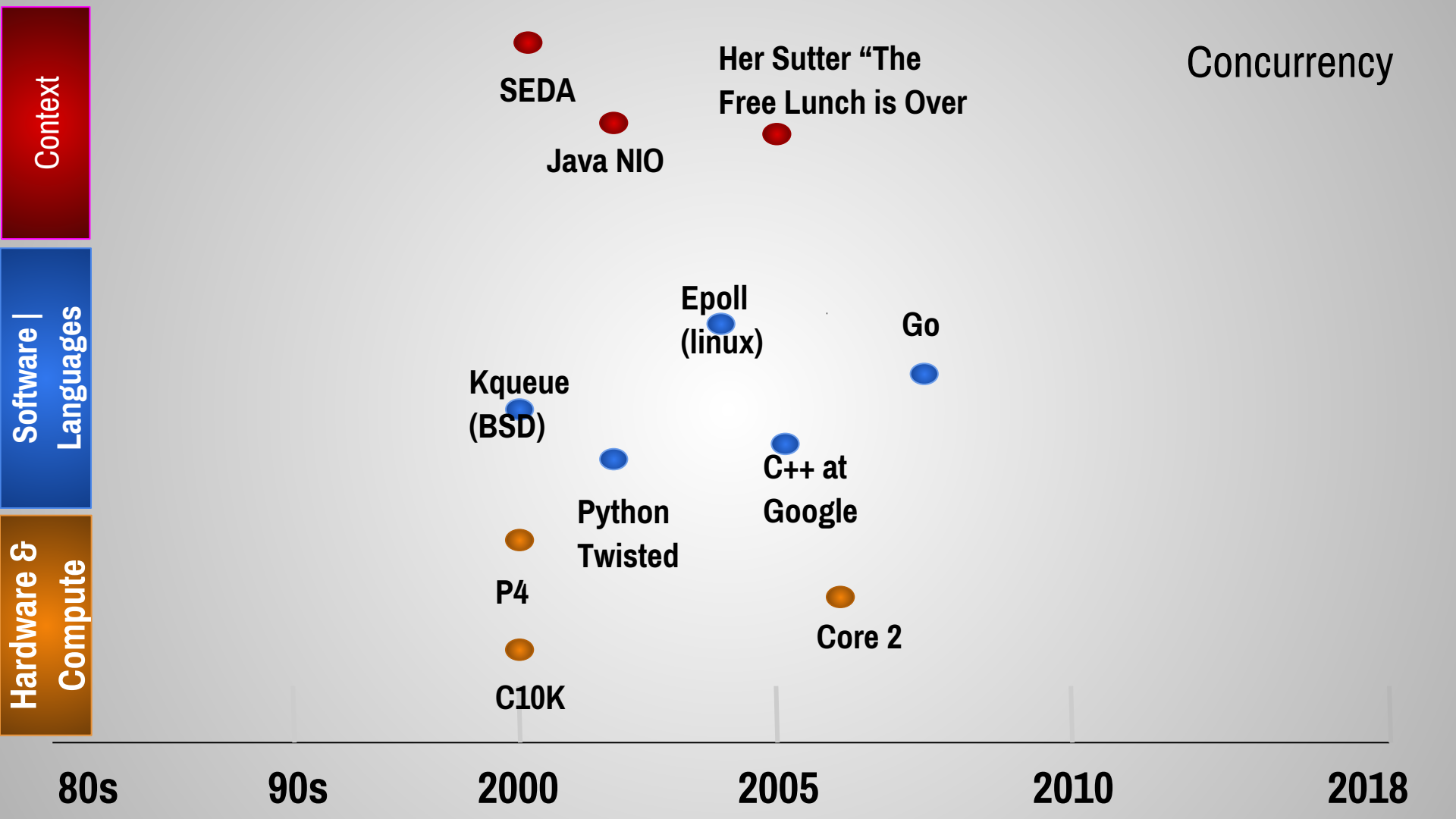
Will concurrency be the next revolution in software development?

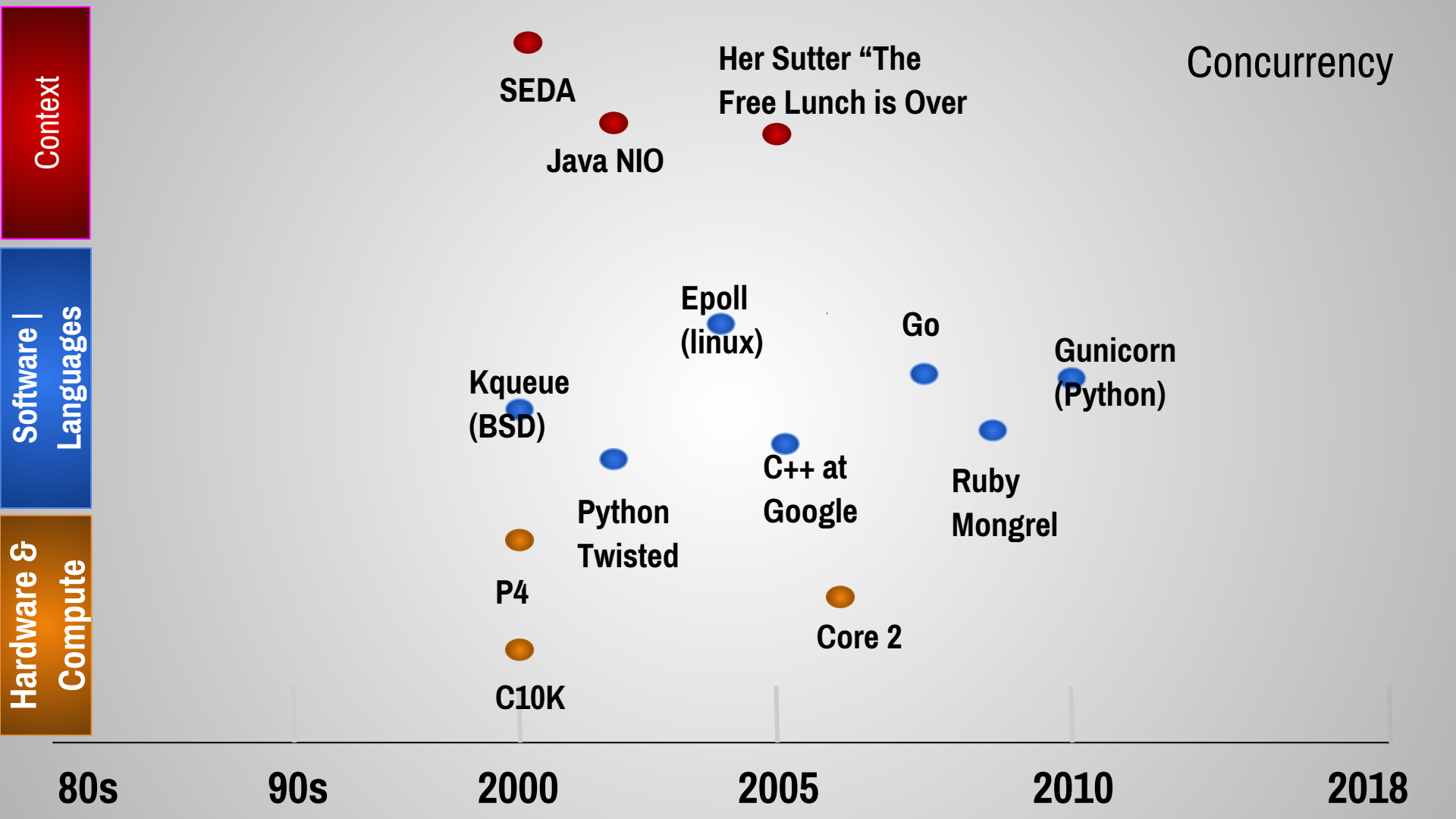
Herb Sutter (http://www.gotw.ca/) chairs the ISO C++ Standards committee and is an architect in Microsoft's Developer Division. His most recent books are Exceptional C++ Style *and* C++ Coding Standards *(Addison-Wesley).*

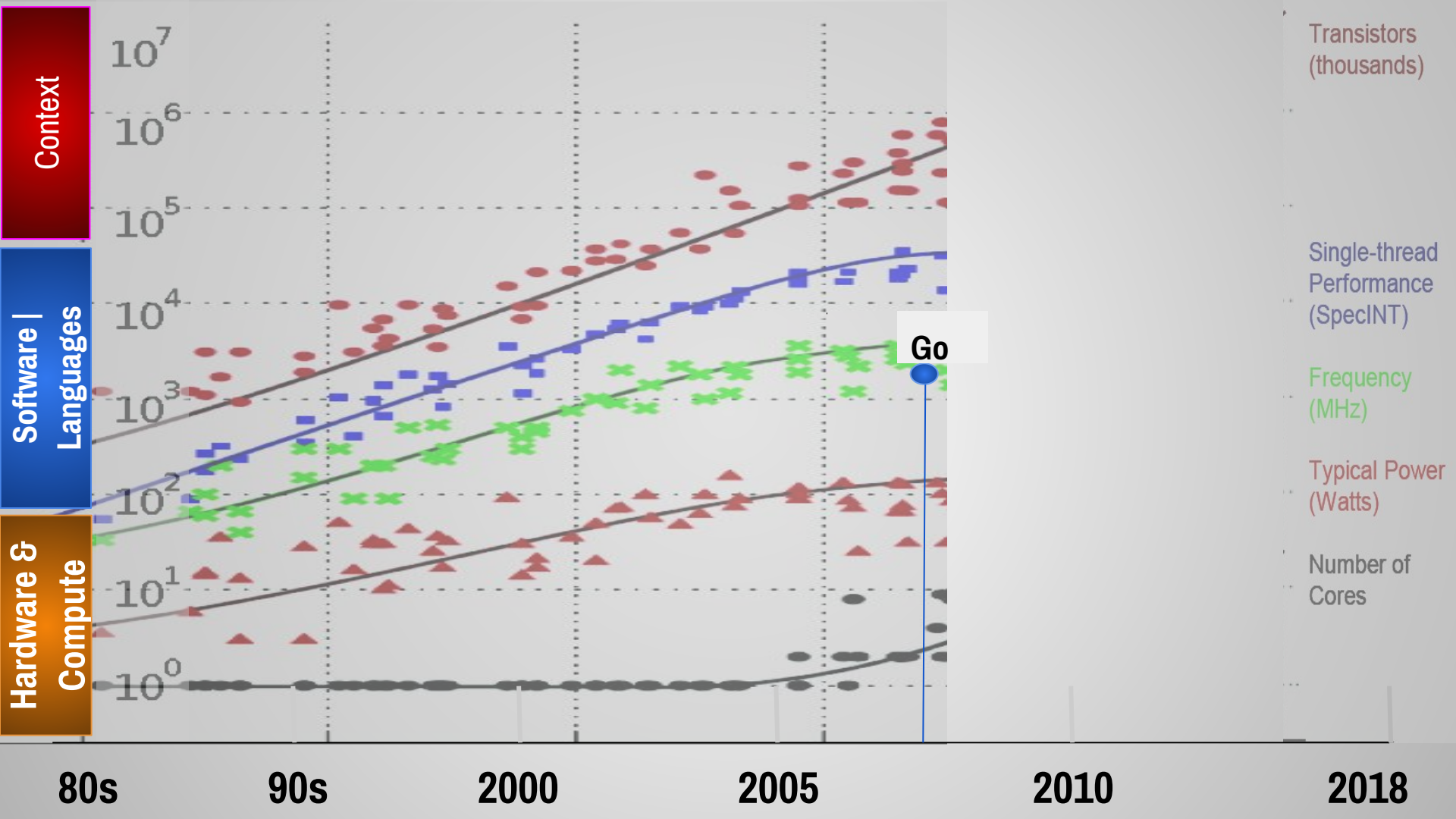
We appear to be at a major turning point in the way we develop software.

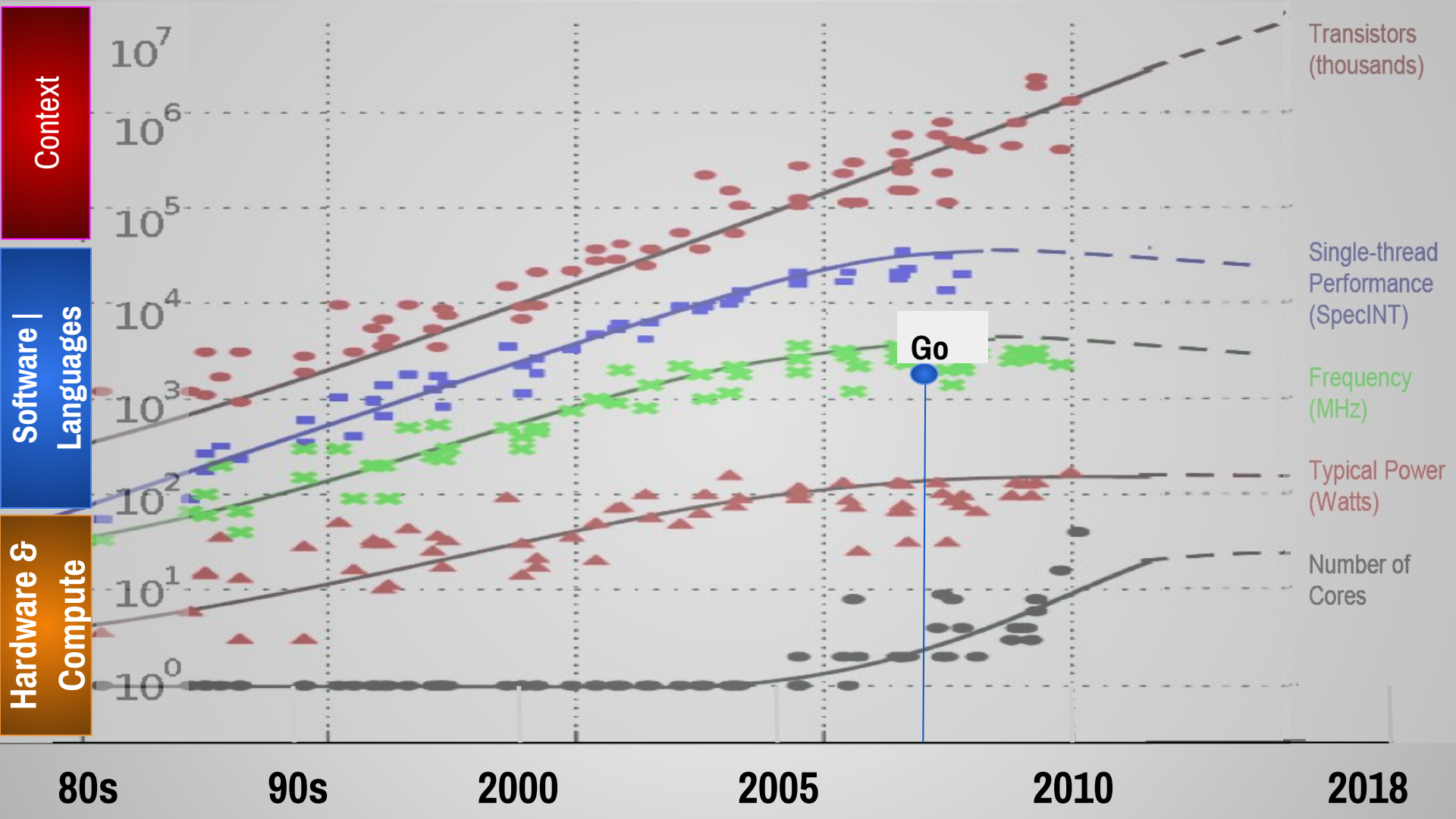
The major processor architectures, from Intel and AMD to Sparc and PowerPC, have run out of room with most of their traditional approaches to boosting CPU performance. Instead of driving clock speeds and straight-line instruction throughput ever higher, they are turning *en masse* to hyperthreading and multicore architectures. That puts us at a fundamental turning point in software development because for years, we've enjoyed a free lunch as faster computers directly made our applications faster too, and that will largely not be true any more. Most of the coming gains won't be picked up directly by the majority of today's applications.


Context
Software | Languages
Hardware & Compute
10^7
10^6
10^5
10^4
10^3
10^2
10^1
10^0
Go
Transistors (thousands)
Single-thread Performance (SpecINT)
Frequency (MHz)
Typical Power (Watts)
Number of Cores
80s
90s
2000
2005
2010
2018

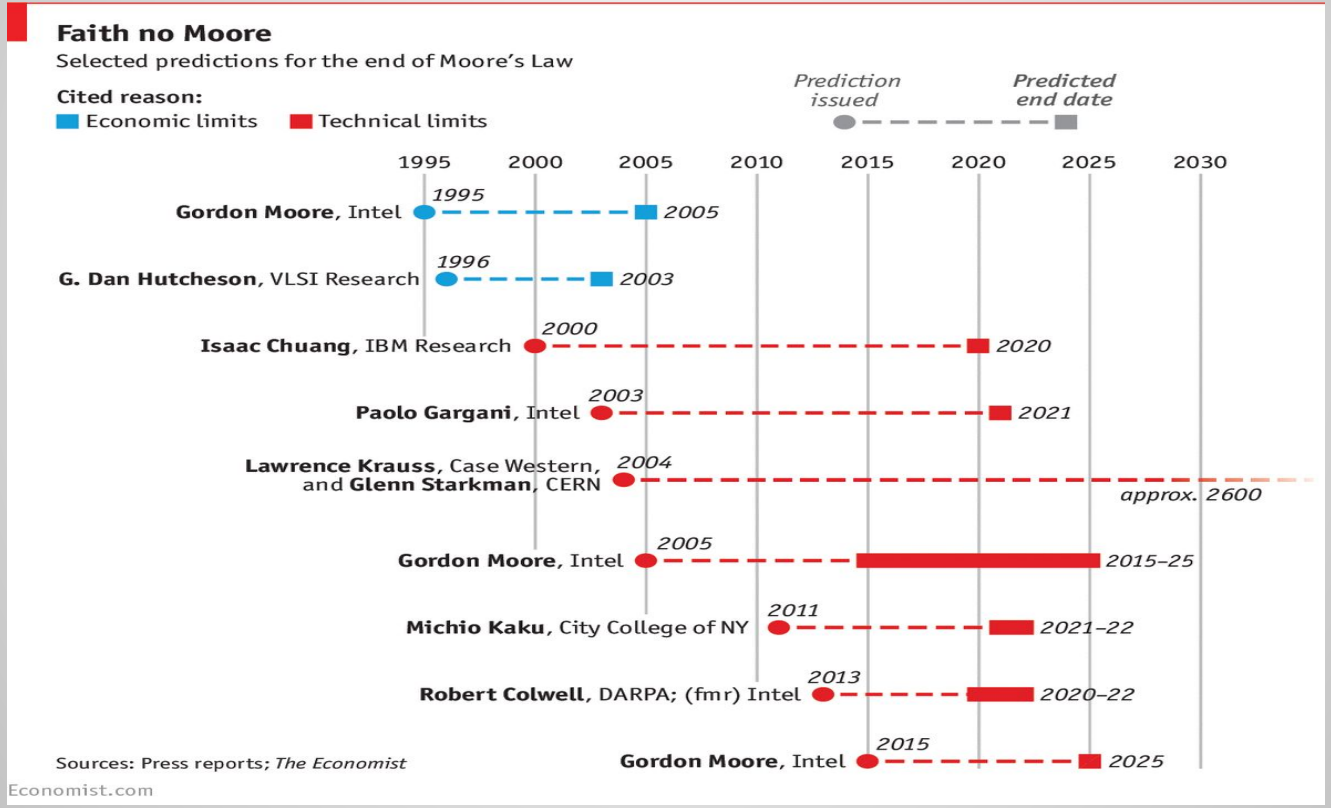

Context
Software | Languages
Hardware & Compute
Concurrency
SEDA
Java NIO
Her Sutter "The Free Lunch is Over
Epoll (linux)
Go
Kqueue (BSD)
C++ at Google
Python Twisted
P4
Core 2
C10K
80s
90s
2000
2005
2010
2018

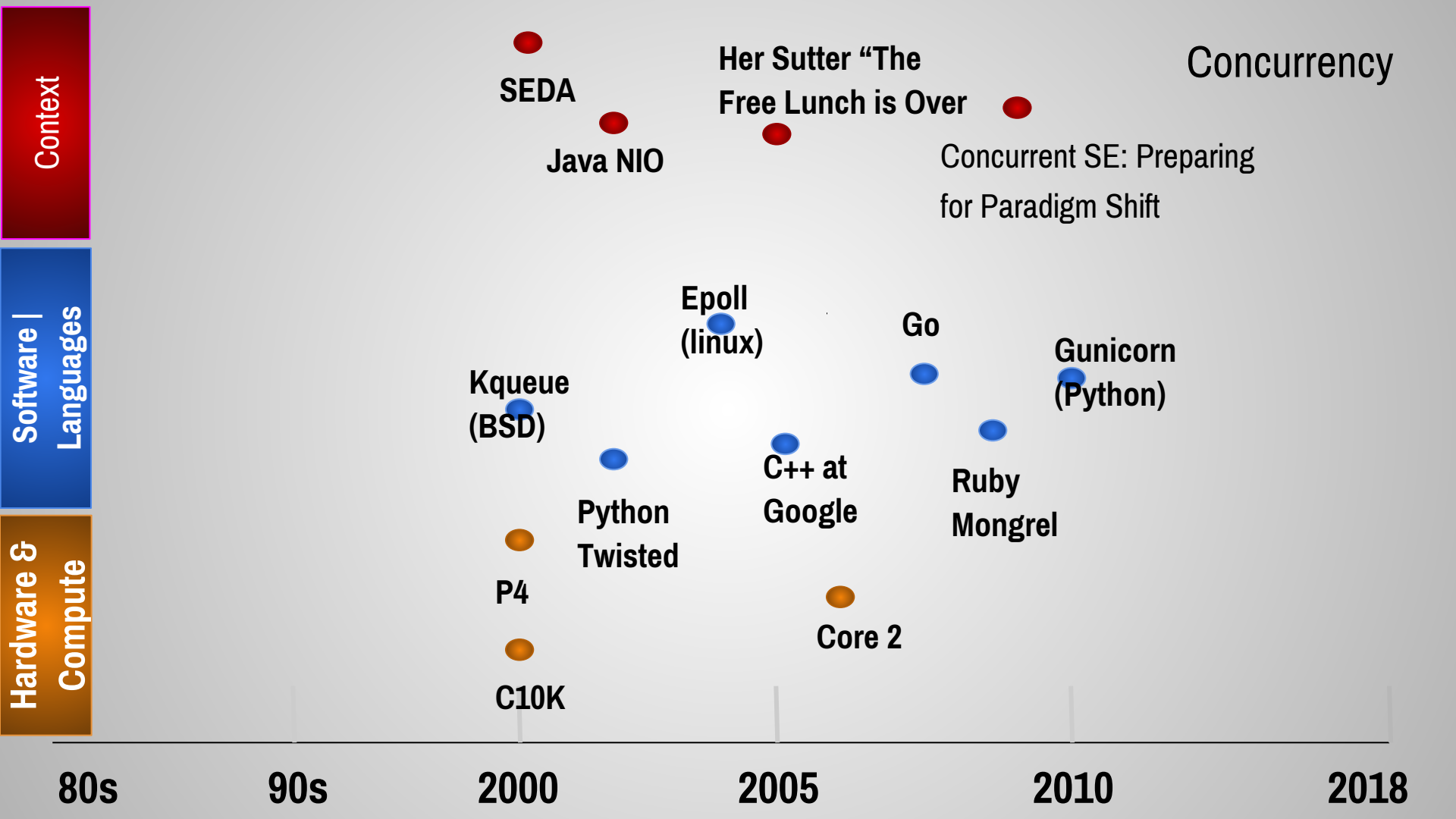

Context
Software | Languages
Hardware & Compute
Concurrency
SEDA
Java NIO
Her Sutter "The Free Lunch is Over
Epoll (linux)
Kqueue (BSD)
Python Twisted
P4
C10K
C++ at Google
Core 2
Go
Ruby Mongrel
Gunicorn (Python)
80s
90s
2000
2005
2010
2018

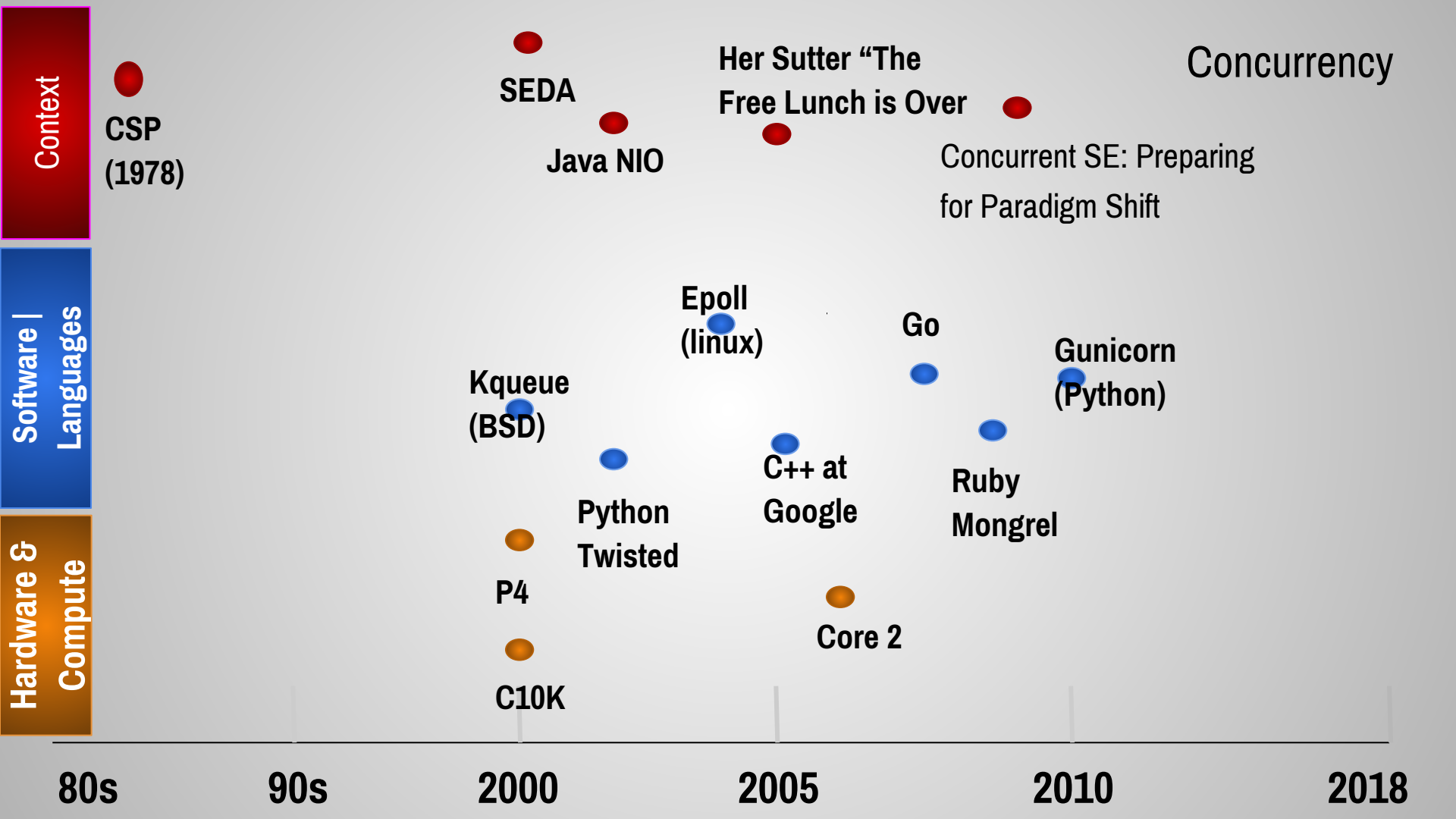

Context
Software | Languages
Hardware & Compute
10^7
10^6
10^5
10^4
10^3
10^2
10^1
10^0
Go
Transistors (thousands)
Single-thread Performance (SpecINT)
Frequency (MHz)
Typical Power (Watts)
Number of Cores
80s
90s
2000
2005
2010
2018


Context
Software | Languages
Hardware & Compute
10^7
10^6
10^5
10^4
10^3
10^2
10^1
10^0
Transistors (thousands)
Single-thread Performance (SpecINT)
Frequency (MHz)
Typical Power (Watts)
Number of Cores
Go
80s
90s
2000
2005
2010
2018

Predictions for the end of Moore's Law




Context
Software | Languages
Hardware & Compute
Concurrency
SEDA
Java NIO
Her Sutter "The Free Lunch is Over
Concurrent SE: Preparing for Paradigm Shift
Epoll (linux)
Kqueue (BSD)
Python Twisted
P4
C10K
C++ at Google
Core 2
Go
Ruby Mongrel
Gunicorn (Python)
80s
90s
2000
2005
2010
2018


Context
Software | Languages
Hardware & Compute
Concurrency
CSP (1978)
SEDA
Java NIO
Her Sutter "The Free Lunch is Over
Concurrent SE: Preparing for Paradigm Shift
Epoll (linux)
Kqueue (BSD)
Python Twisted
P4
C10K
C++ at Google
Core 2
Go
Ruby Mongrel
Gunicorn (Python)
80s
90s
2000
2005
2010
2018

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

1. Introduction

Among the primitive concepts of computer programming, and of the high level languages in which programs are expressed, the action of assignment is familiar and well understood. In fact, any change of the internal state of a machine executing a program can be modeled as an assignment of a new value to some variable part of that machine. However, the operations of input and output, which affect the external environment of a machine, are not nearly so well understood. They are often added to a programming language only as an afterthought.

Among the structuring methods for computer programs, three basic constructs have received widespread recognition and use: A repetitive construct (e.g. the **while** loop), an alternative construct (e.g. the conditional **if..then..else**), and normal sequential program composition (often denoted by a semicolon). Less agreement has been reached about the design of other important program structures, and many suggestions have been made: Subroutines (Fortran), procedures (Algol 60 [15]), entries (PL/I), coroutines (UNIX [17]), classes (SIMULA 67 [5]), processes and monitors (Concurrent Pascal [2]), clusters (CLU [13]), forms (ALPHARD [19]), actors (Hewitt [1]).

The traditional stored program digital computer has been designed primarily for deterministic execution of a single sequential program. Where the desire for greater speed has led to the introduction of parallelism, every attempt has been made to disguise this fact from the programmer, either by hardware itself (as in the multiple function units of the CDC 6600) or by the software (as in an I/O control package, or a multiprogrammed operating system). However, developments of processor technology suggest that a multiprocessor machine, constructed from a number of similar self-contained processors (each with its own store), may become more powerful, capacious, reliable, and economical than a machine which is disguised as a monoprocessor.

In order to use such a machine effectively on a single task, the component processors must be able to communicate and to synchronize with each other. Many methods of achieving this have been proposed. A widely adopted method of communication is by inspection and updating of a common store (as in Algol 68 [18], PL/I, and many machine codes). However, this can create severe problems in the construction of correct programs and it may lead to expense (e.g. crossbar switches) and unreliability (e.g. glitches) in some technologies of hardware implementation. A greater variety of methods has been proposed for synchronization: semaphores [6], events (PL/I), conditional critical regions [10], monitors and queues (Concurrent Pascal [2]), and path expressions [3]. Most of these are demonstrably adequate for their purpose, but there is no widely recognized criterion for choosing between them.

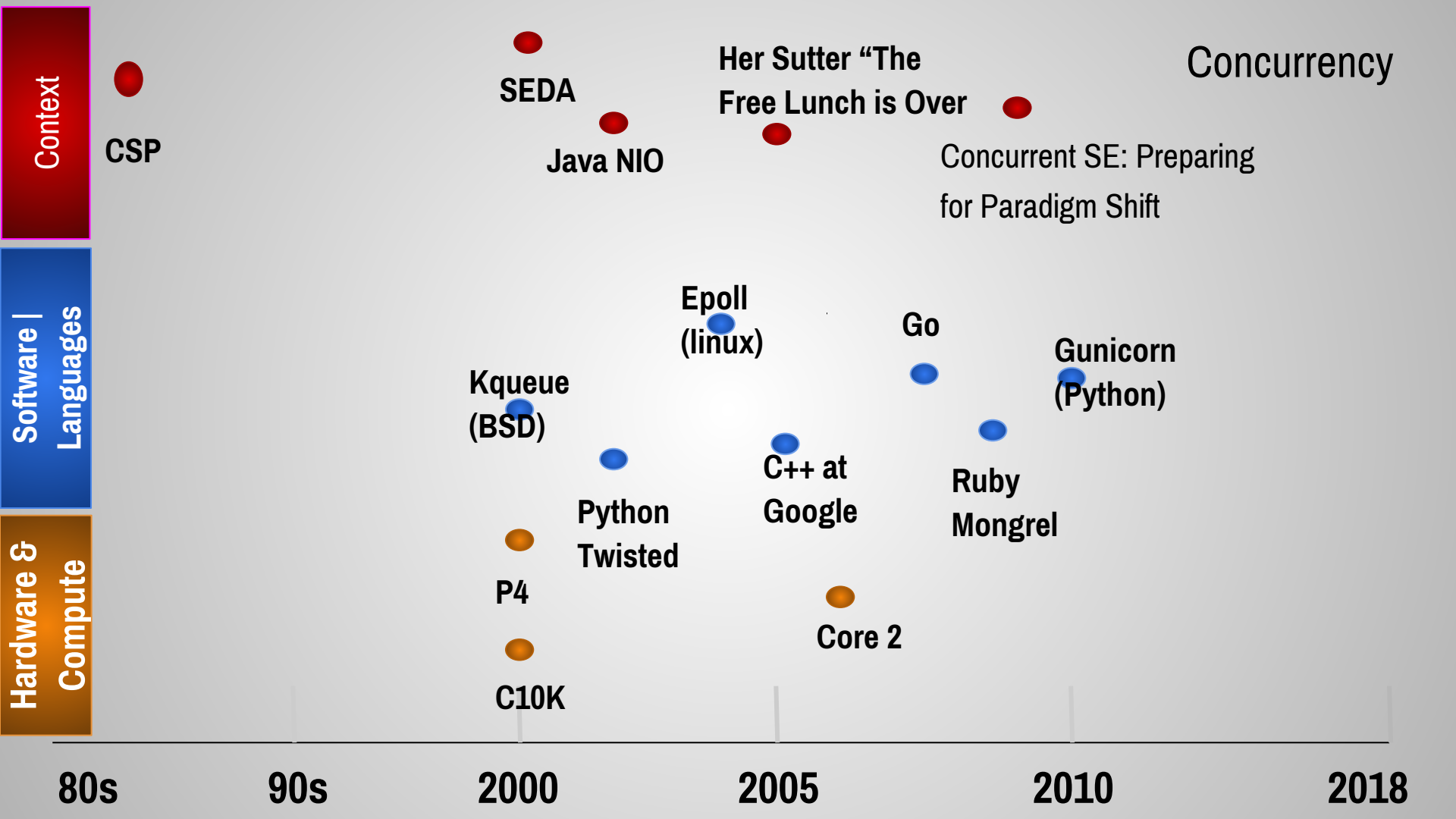
This paper makes an ambitious attempt to find a single simple solution to all these problems. The essential proposals are:
(1) Dijkstra's guarded commands [8] are adopted (with a slight change of notation) as sequential control structures, and as the sole means of introducing and controlling nondeterminism.
(2) A parallel command, based on Dijkstra's *parbegin* [6], specifies concurrent execution of its constituent sequential commands (processes). All the processes start simultaneously, and the parallel command ends only when they are all finished. They may not communicate with each other by updating global variables.
(3) Simple forms of input and output command are introduced. They are used for communication between concurrent processes.

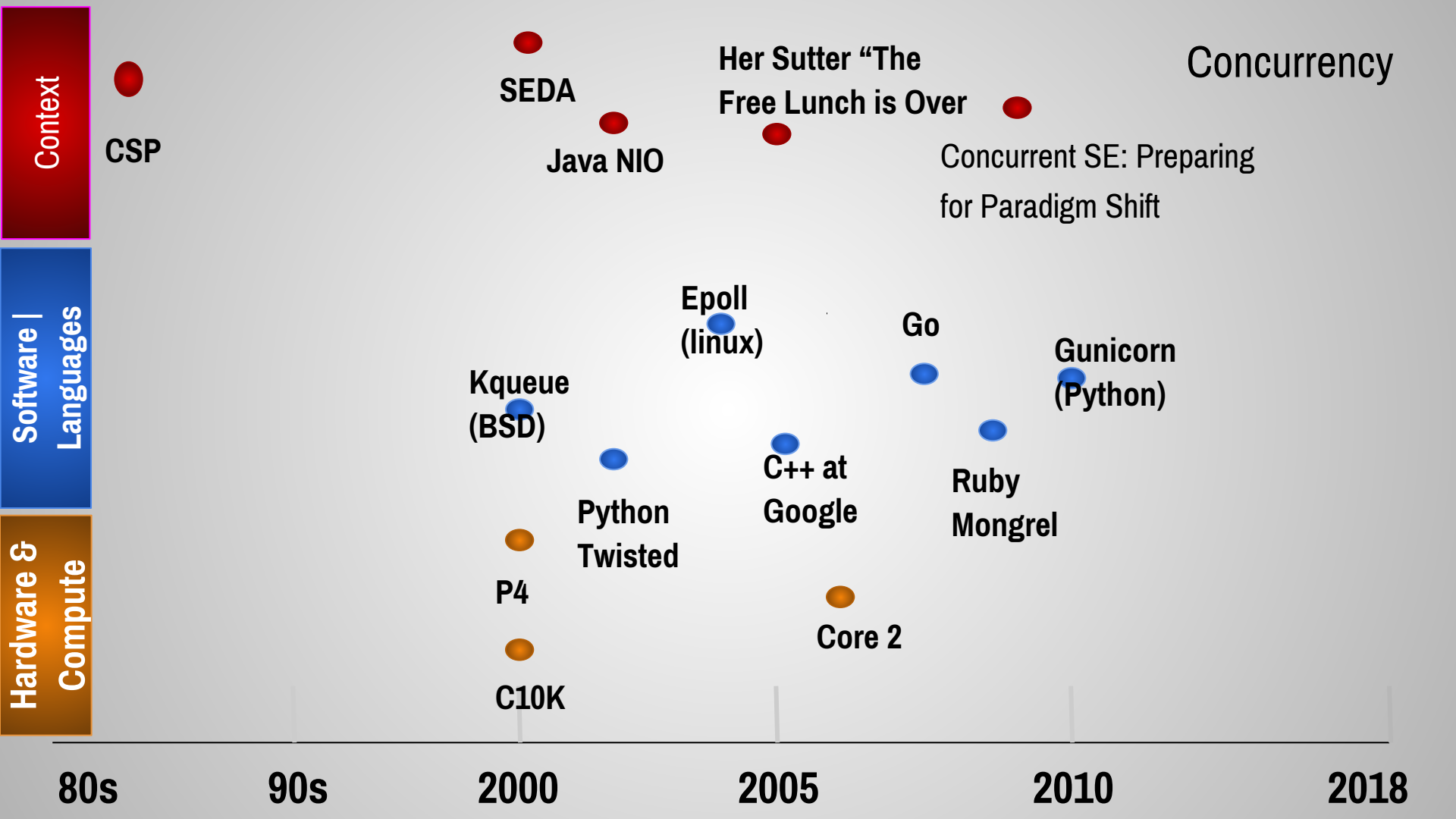
This research was supported by a Senior Fellowship of the Science Research Council.

Author's present address: Programming Research Group, 45, Banbury Road, Oxford, England.

Why build concurrency on the ideas of CSP?

Concurrency and multi-threaded programming have over time developed a reputation for difficulty. [...]One of the most successful models for providing high-level linguistic support for concurrency comes from Hoare's Communicating Sequential Processes, or CSP. ***Occam and Erlang are two well known languages that stem from CSP.*** Go's concurrency primitives derive from a different part of the family tree whose main contribution is the powerful notion of channels as first class objects. Experience with several earlier languages has shown that the CSP model fits well into a procedural language framework.


Context
Software | Languages
Hardware & Compute
Concurrency
CSP
SEDA
Java NIO
Her Sutter "The Free Lunch is Over
Concurrent SE: Preparing for Paradigm Shift
Epoll (linux)
Kqueue (BSD)
Python Twisted
P4
C10K
C++ at Google
Core 2
Go
Ruby Mongrel
Gunicorn (Python)
80s
90s
2000
2005
2010
2018


Context
Software | Languages
Hardware & Compute
Concurrency
CSP
SEDA
Java NIO
Her Sutter "The Free Lunch is Over
Concurrent SE: Preparing for Paradigm Shift
Epoll (linux)
Kqueue (BSD)
Python Twisted
P4
C10K
C++ at Google
Core 2
Go
Ruby Mongrel
Gunicorn (Python)
80s
90s
2000
2005
2010
2018

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

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

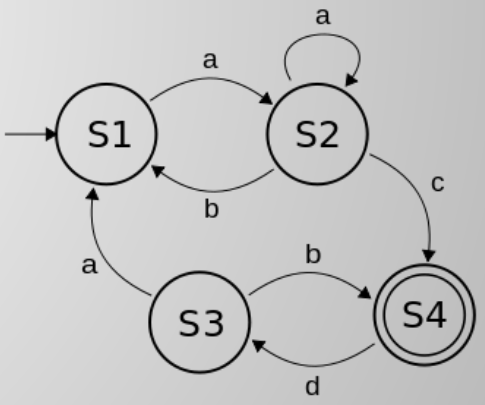
Nginx - event loop plus state machine model

App

Events, Threads and Goroutines

Nginx - event loop plus state machine model

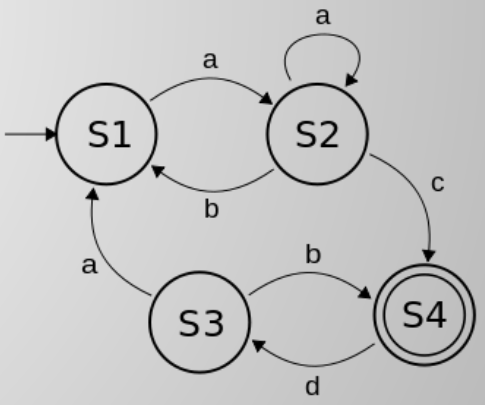
App



Events, Threads and Goroutines

Nginx - event loop plus state machine model

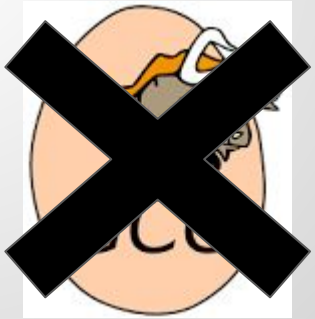
App

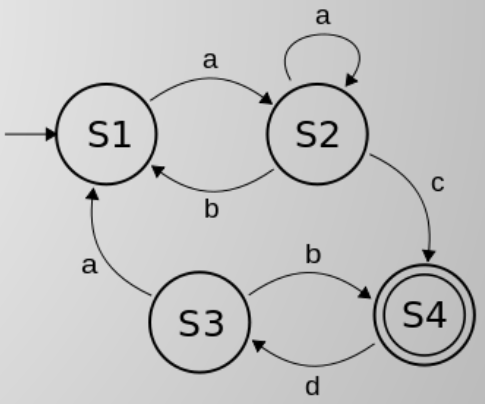


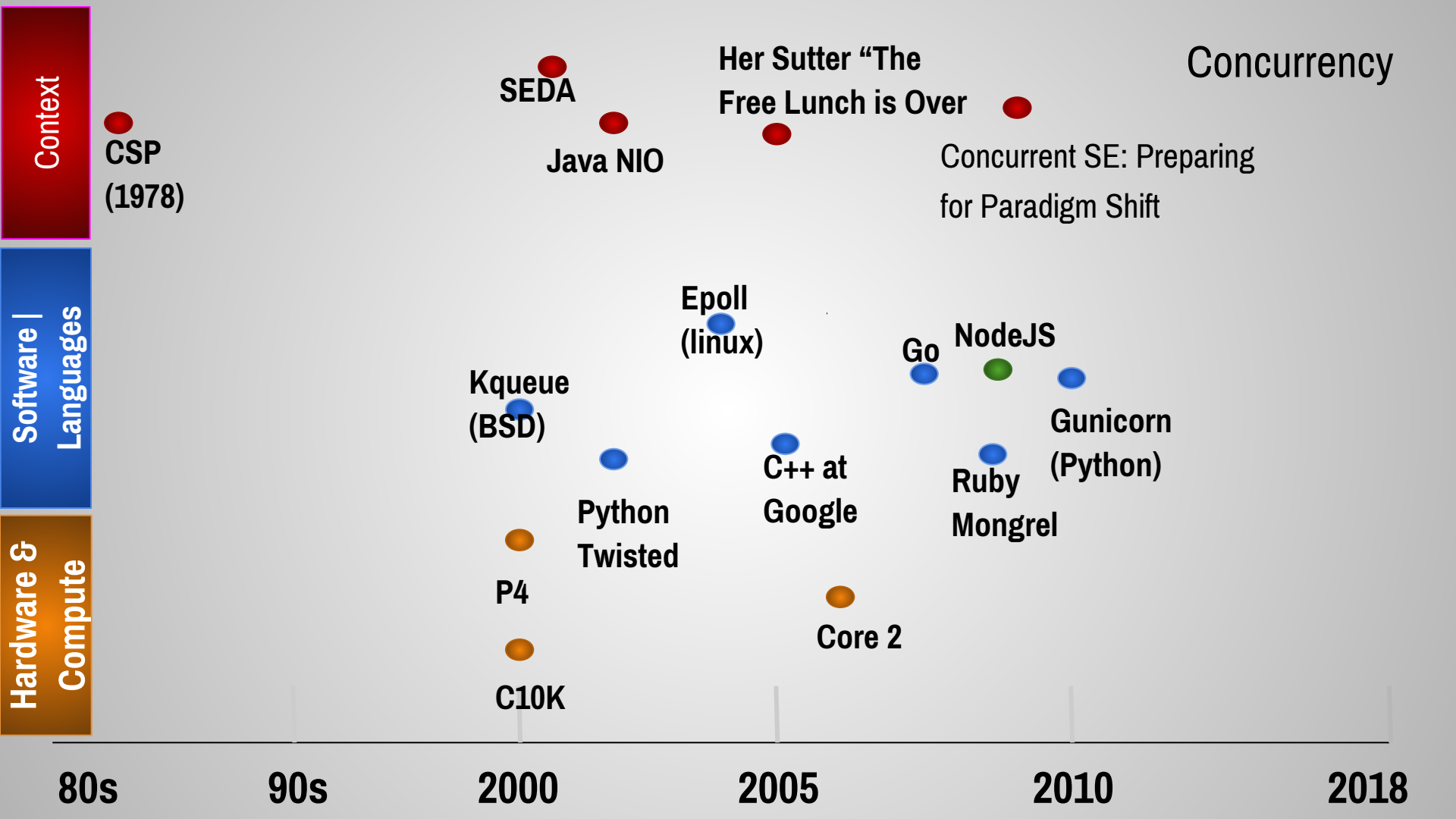
Events, Threads and Goroutines

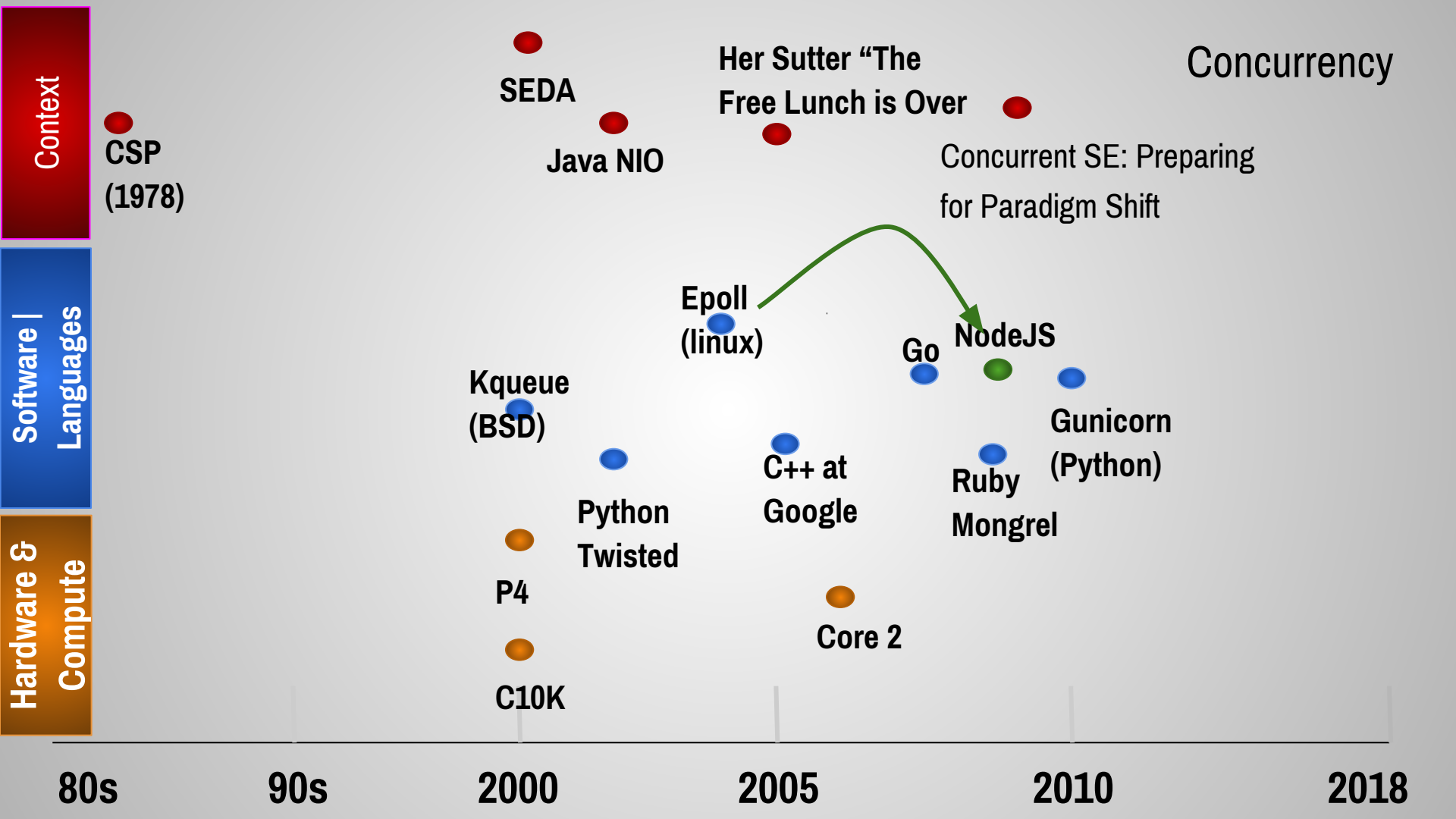
Nginx - event loop plus state machine model

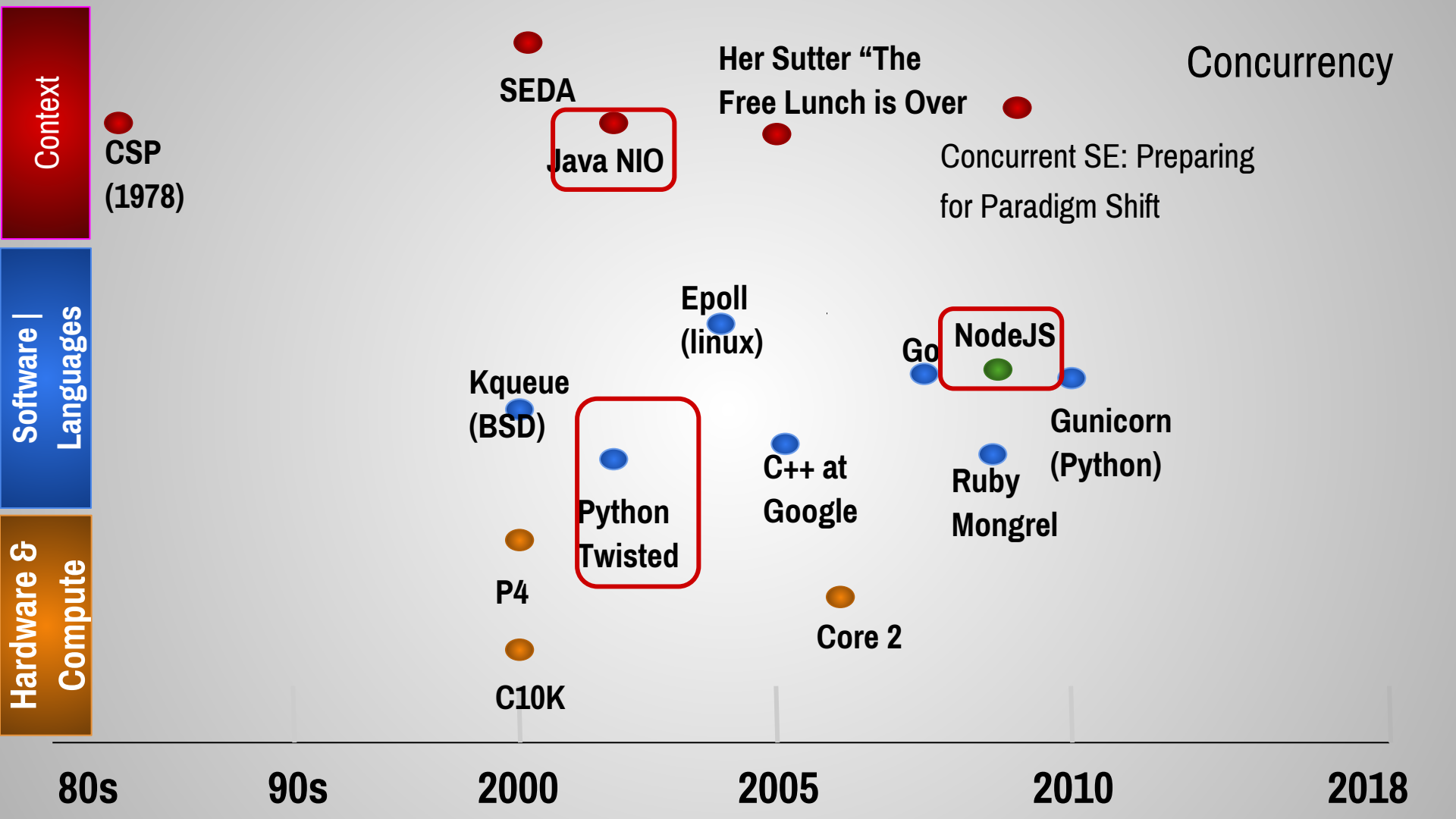
App



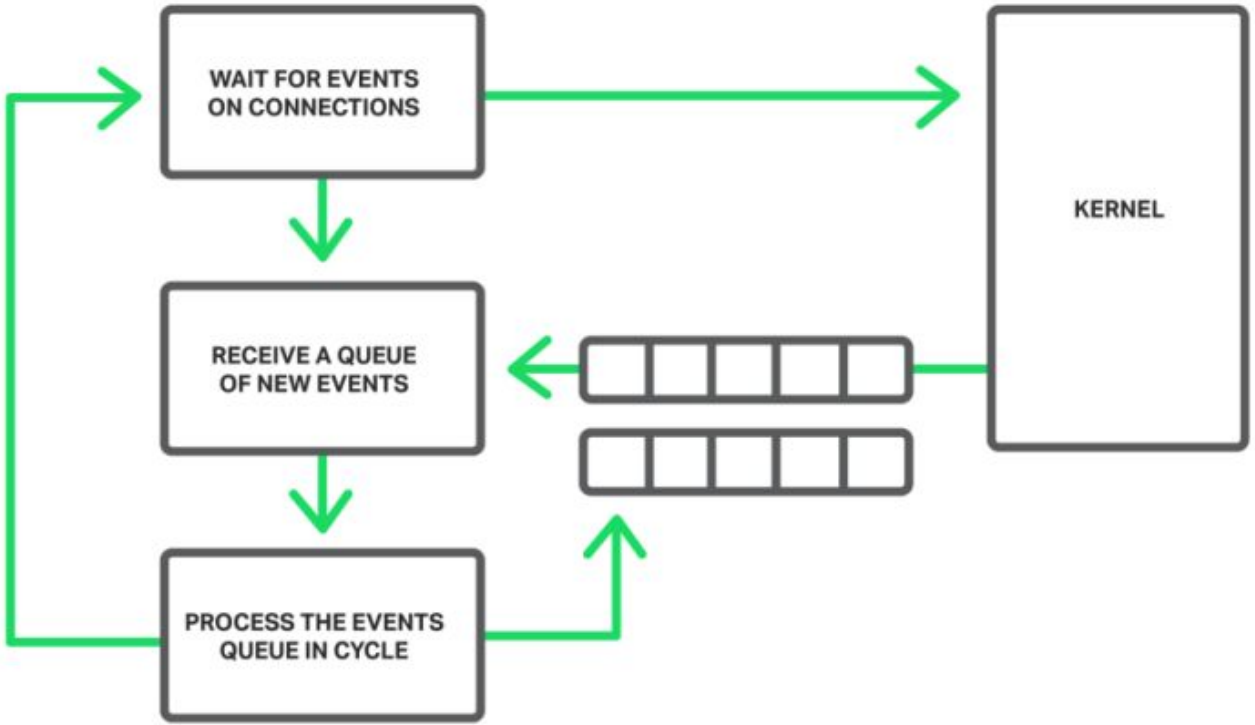


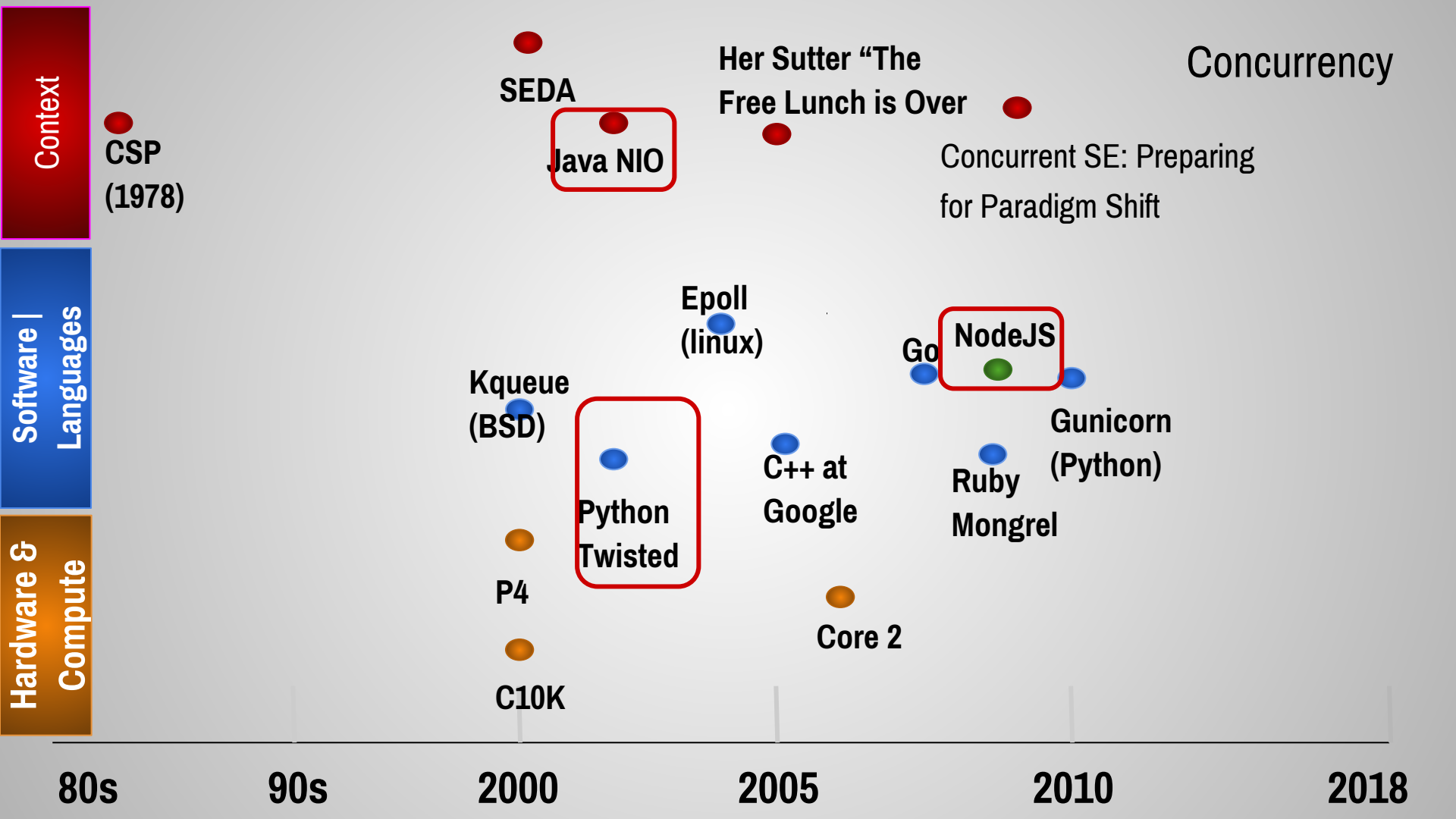

Context
Software | Languages
Hardware & Compute
Concurrency
CSP (1978)
SEDA
Java NIO
Her Sutter "The Free Lunch is Over
Concurrent SE: Preparing for Paradigm Shift
Epoll (linux)
Kqueue (BSD)
Python Twisted
P4
C10K
C++ at Google
Core 2
Go
NodeJS
Gunicorn (Python)
Ruby Mongrel
80s
90s
2000
2005
2010
2018

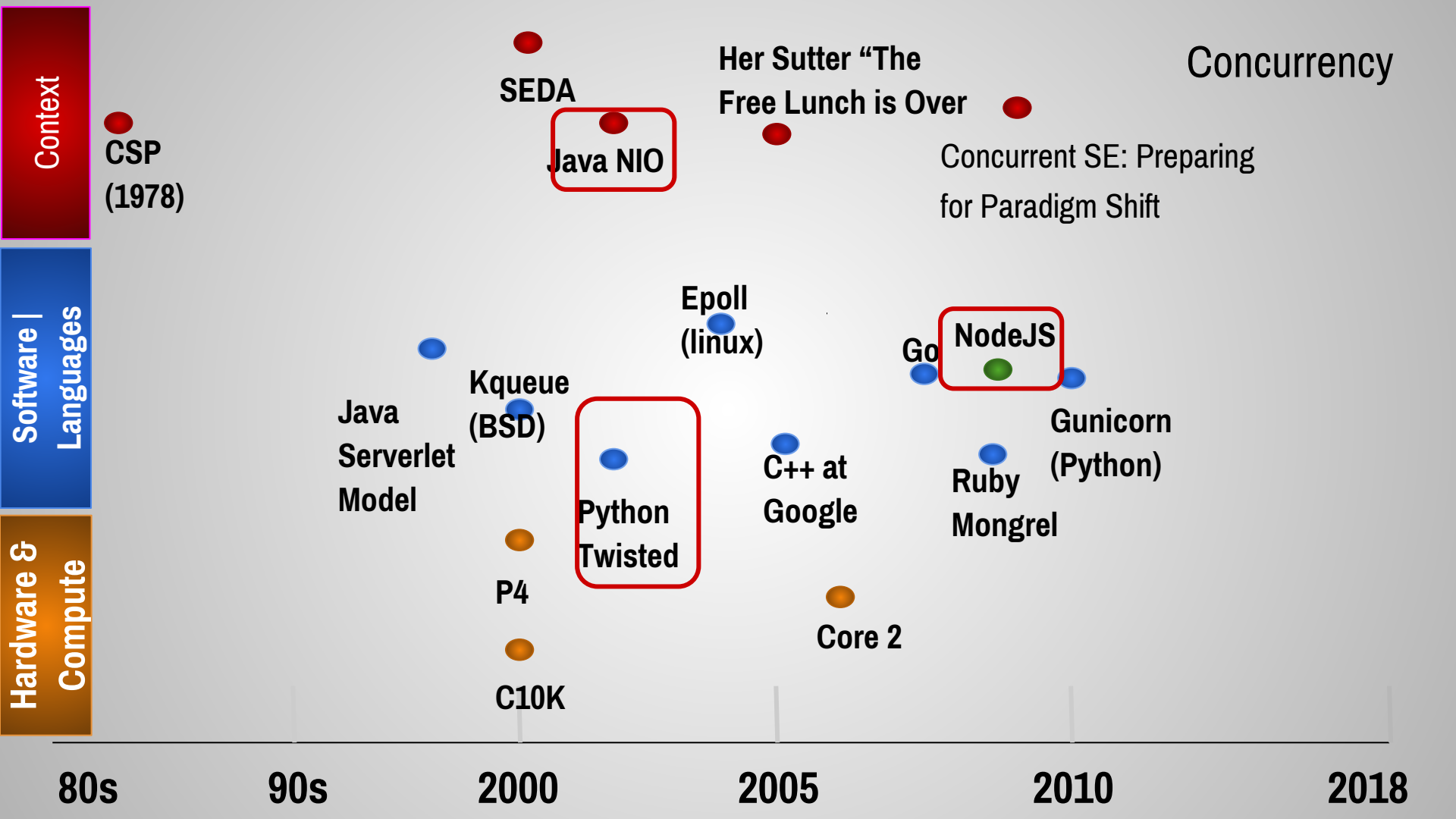

Context
Software | Languages
Hardware & Compute
Concurrency
CSP
(1978)
SEDA
Java NIO
Her Sutter "The
Free Lunch is Over
Concurrent SE: Preparing
for Paradigm Shift
Epoll
(linux)
Kqueue
(BSD)
Python
Twisted
P4
C10K
C++ at
Google
Core 2
Go
NodeJS
Ruby
Mongrel
Gunicorn
(Python)
80s
90s
2000
2005
2010
2018

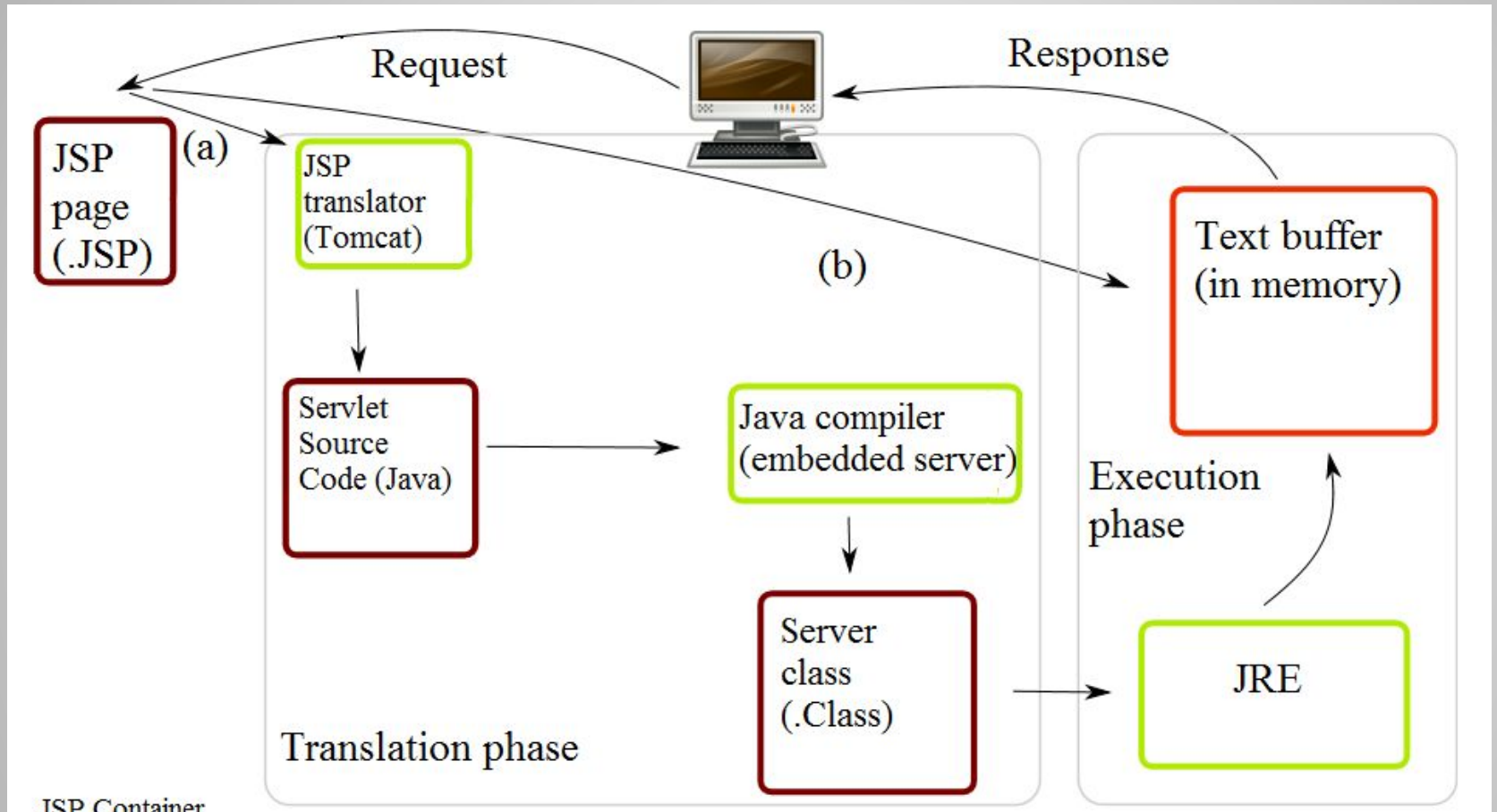

Context
Software | Languages
Hardware & Compute
Concurrency
CSP (1978)
SEDA
Java NIO
Her Sutter "The Free Lunch is Over
Concurrent SE: Preparing for Paradigm Shift
Epoll (linux)
Kqueue (BSD)
Python Twisted
P4
C10K
C++ at Google
Core 2
Go
NodeJS
Gunicorn (Python)
Ruby Mongrel
80s
90s
2000
2005
2010
2018

NGINX EVENT LOOP

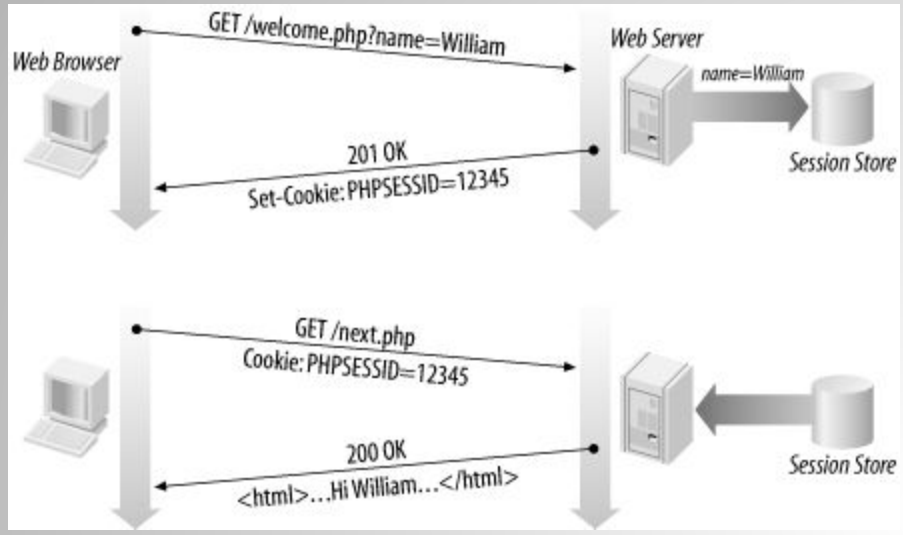


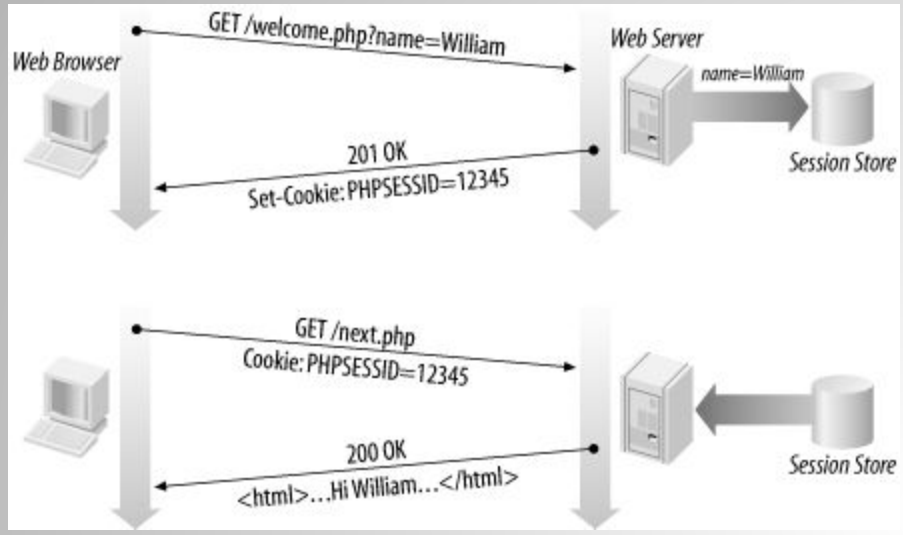

Context
Software | Languages
Hardware & Compute
Concurrency
CSP (1978)
SEDA
Java NIO
Her Sutter "The Free Lunch is Over
Concurrent SE: Preparing for Paradigm Shift
Epoll (linux)
Kqueue (BSD)
Python Twisted
P4
C10K
C++ at Google
Core 2
Go
NodeJS
Gunicorn (Python)
Ruby Mongrel
80s
90s
2000
2005
2010
2018


Context
Software | Languages
Hardware & Compute
Concurrency
CSP (1978)
SEDA
Java NIO
Her Sutter "The Free Lunch is Over
Concurrent SE: Preparing for Paradigm Shift
Epoll (linux)
Java Serverlet Model
Kqueue (BSD)
Python Twisted
P4
C10K
C++ at Google
Core 2
Go
NodeJS
Ruby Mongrel
Gunicorn (Python)
80s
90s
2000
2005
2010
2018

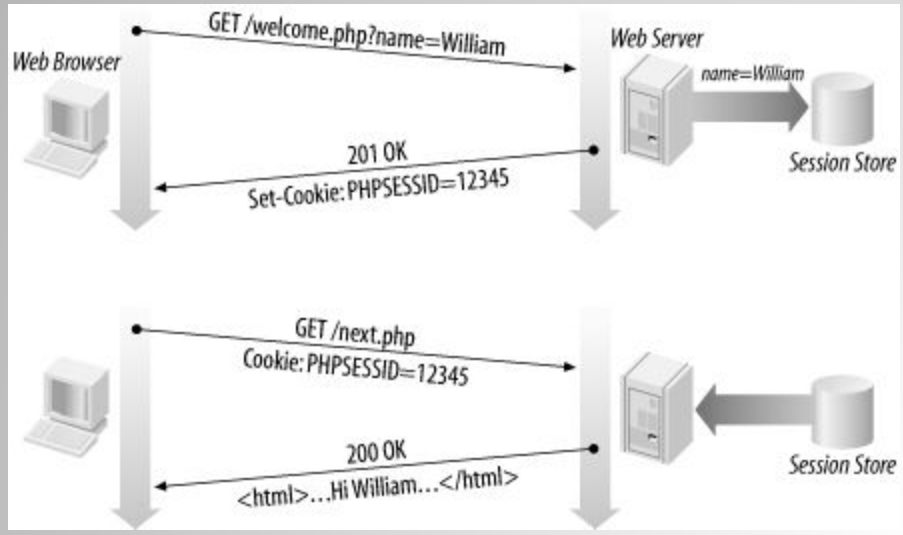


JSP Container
(a) Translation occurs at this point, if JSP has been changed or is new.
(b) If not, translation is skipped.

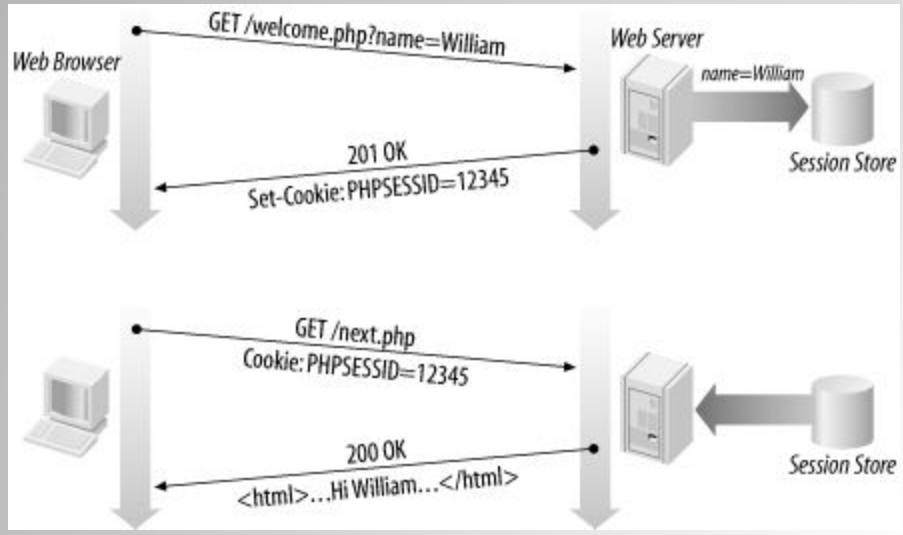

GET /welcome.php?name=William
Web Browser
Web Server
name=William
Session Store
201 OK
Set-Cookie: PHPSESSID=12345
GET /next.php
Cookie: PHPSESSID=12345
Session Store
200 OK
<html>…Hi William…</html>


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
200 OK
<html>…Hi William…</html>
Session Store

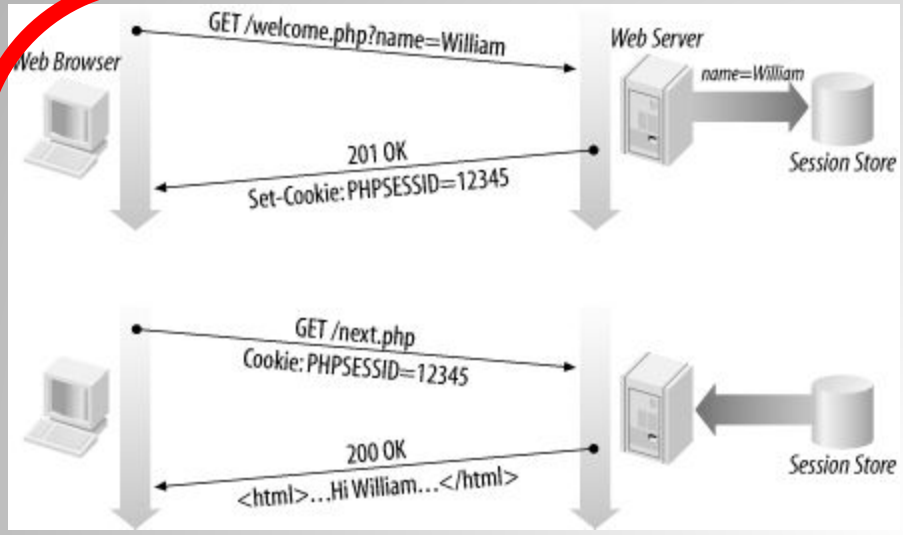

WWW.


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
200 OK
<html>…Hi William…</html>
Session Store


WWW.

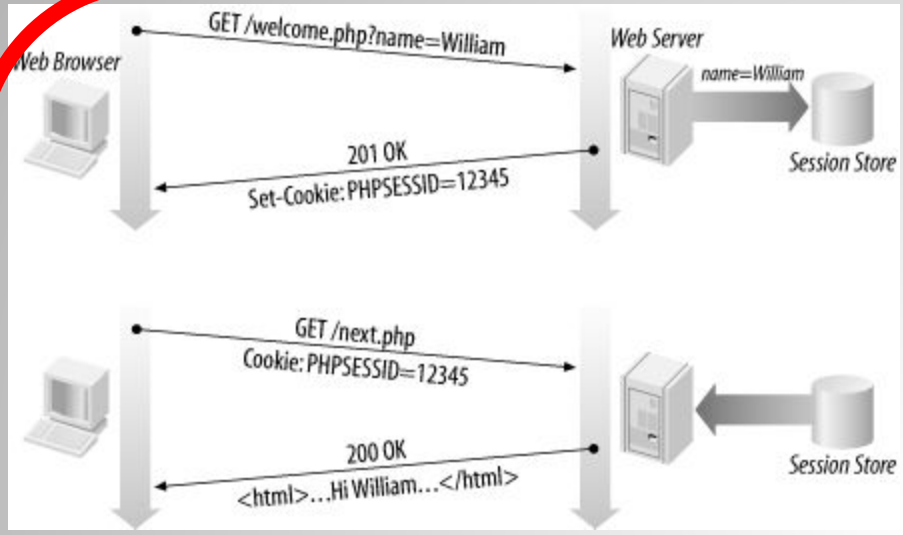

GET /welcome.php?name=William
Web Browser
Web Server
name=William
Session Store
201 OK
Set-Cookie: PHPSESSID=12345
GET /next.php
Cookie: PHPSESSID=12345
200 OK
<html>…Hi William…</html>
Session Store


WWW.


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


WWW.

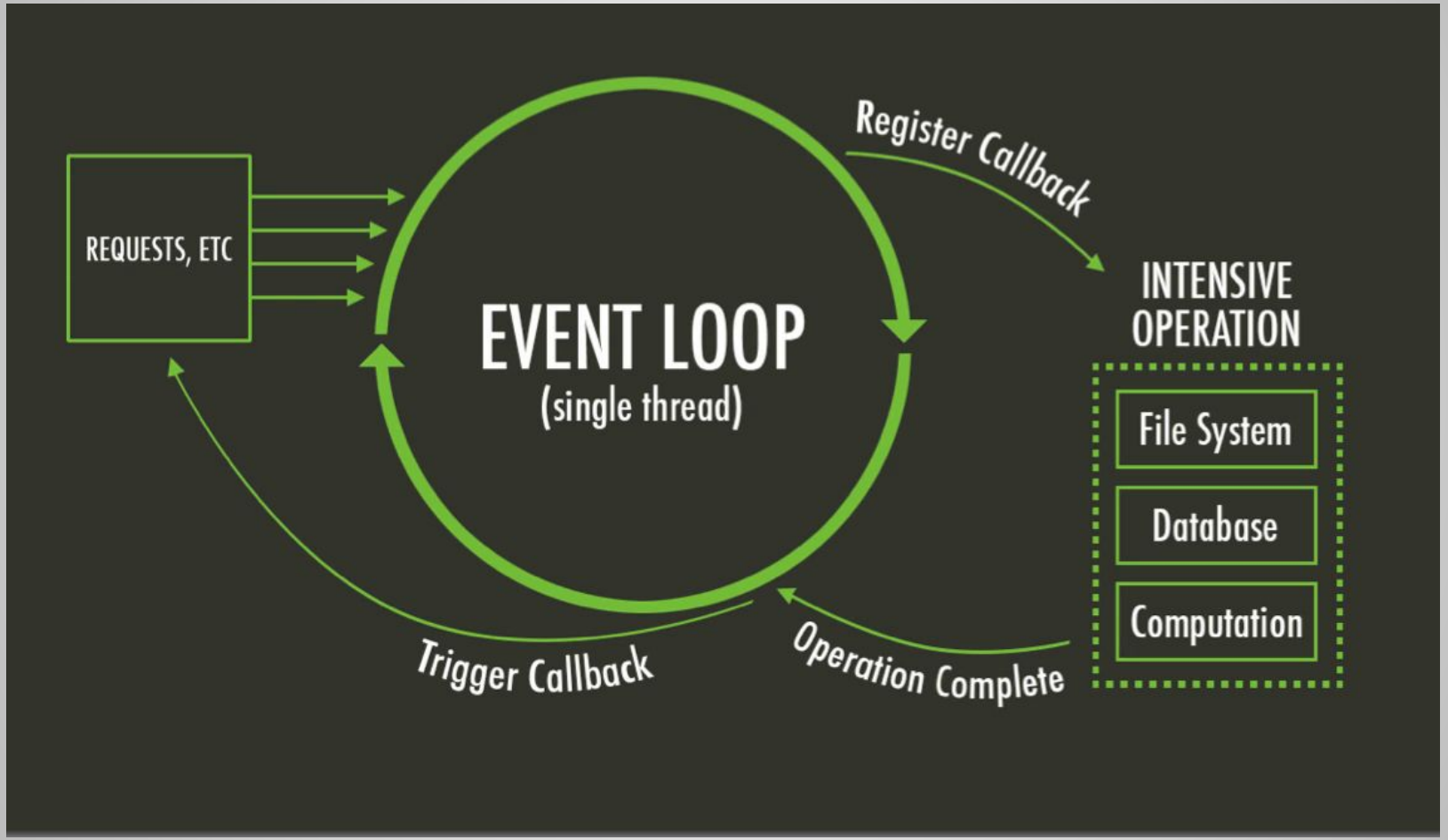
App

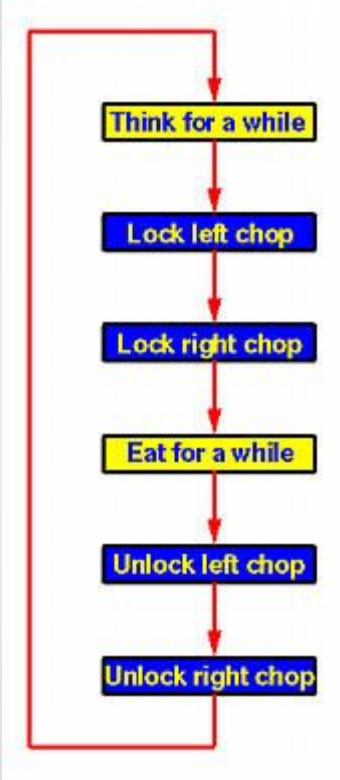

GET /welcome.php?name=William
Web Browser
Web Server
name=William
Session Store
201 OK
Set-Cookie: PHPSESSID=12345
GET /next.php
Cookie: PHPSESSID=12345
200 OK
<html>…Hi William…</html>
Session Store

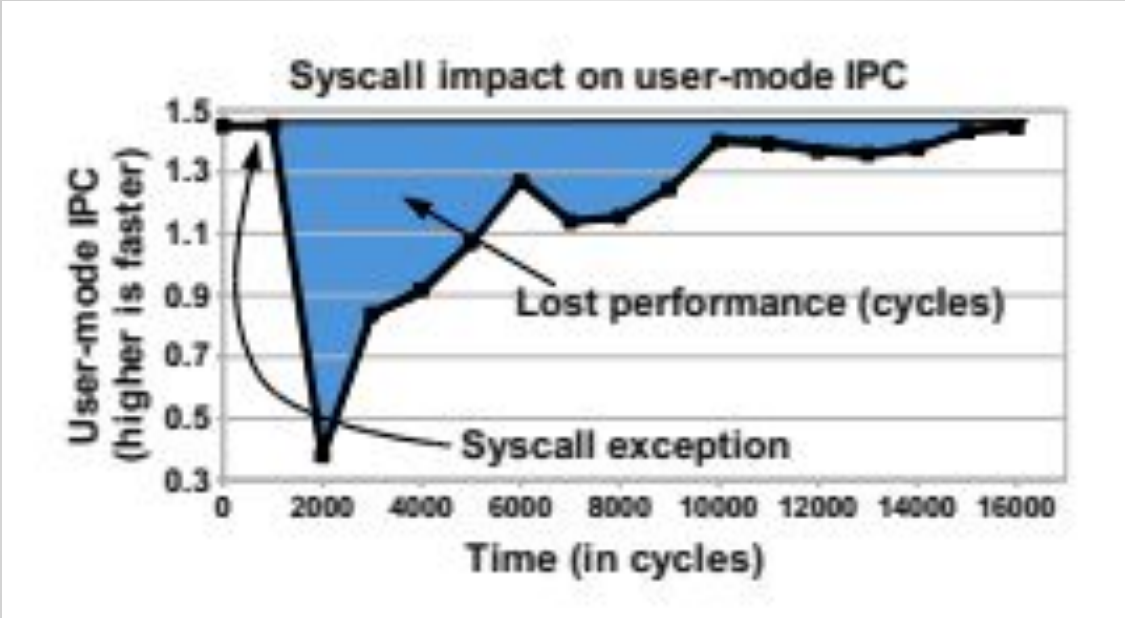

WWW.

= Threads

App


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

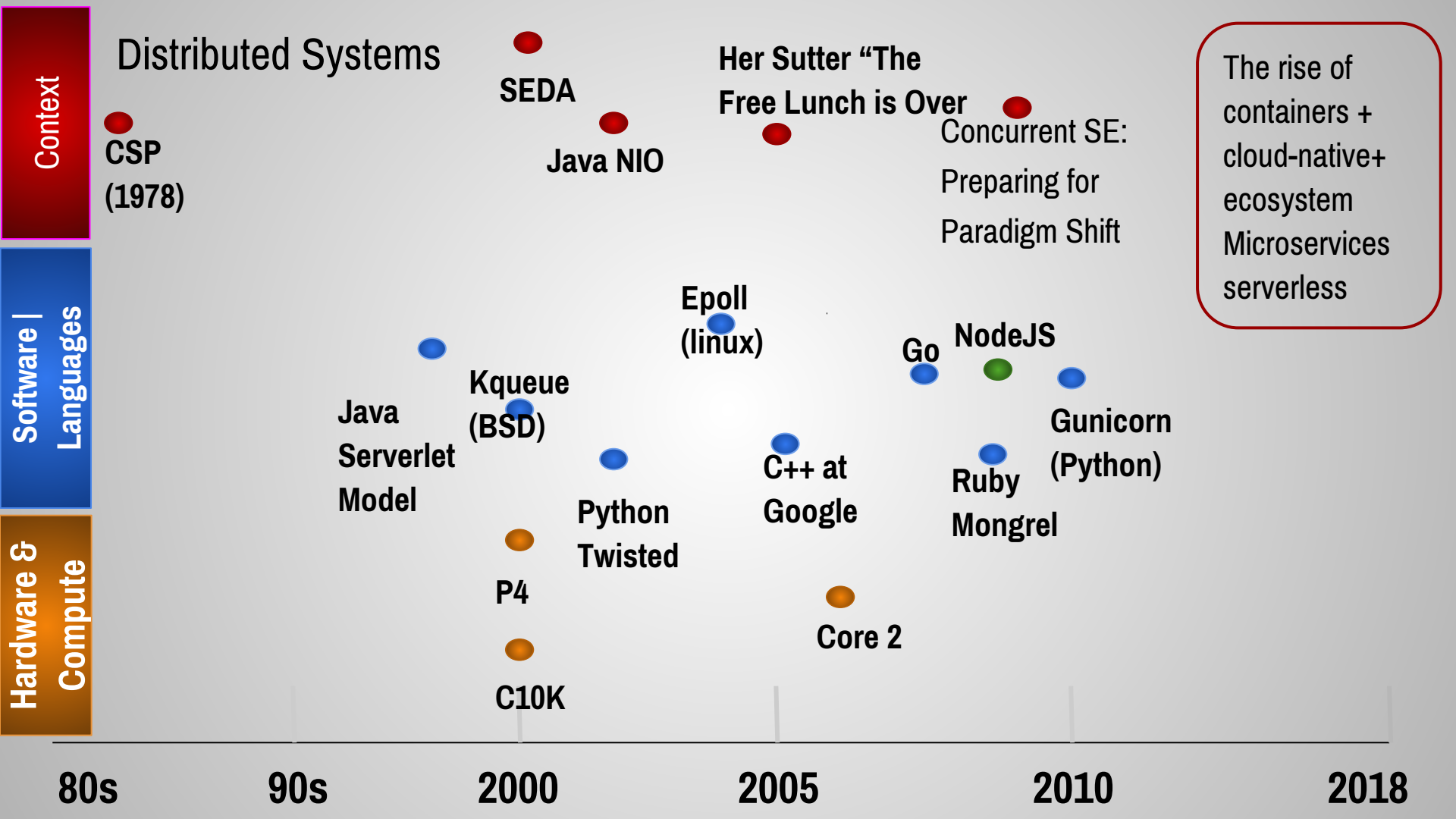

Think for a while
Lock left chop
Lock right chop
Eat for a while
Unlock left chop
Unlock right chop

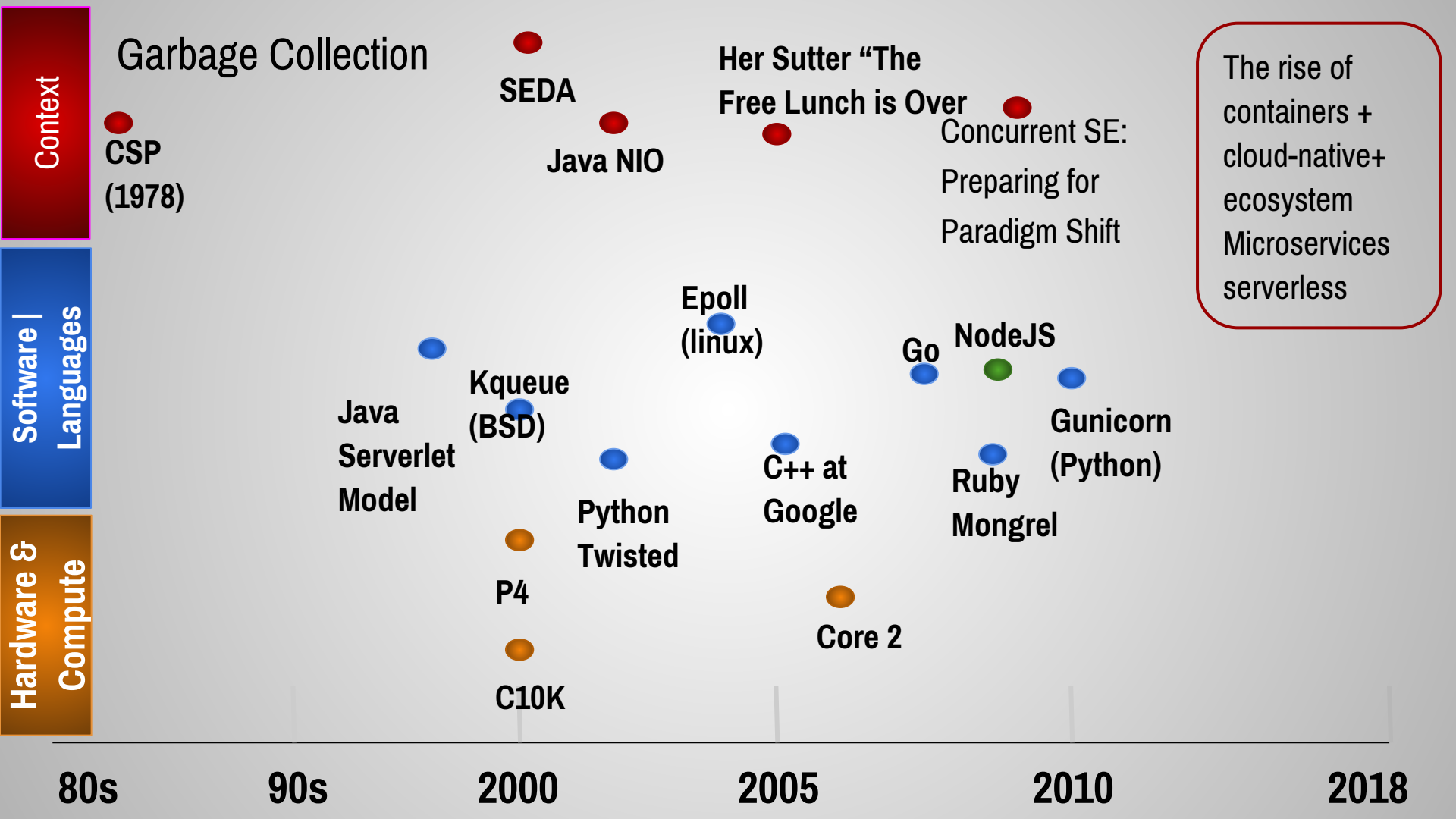

Syscall impact on user-mode IPC
User-mode IPC
(higher is faster)
1.5
1.3
1.1
0.9
0.7
0.5
0.3
0
2000
4000
6000
8000
10000
12000
14000
16000
Time (in cycles)
Lost performance (cycles)
Syscall exception

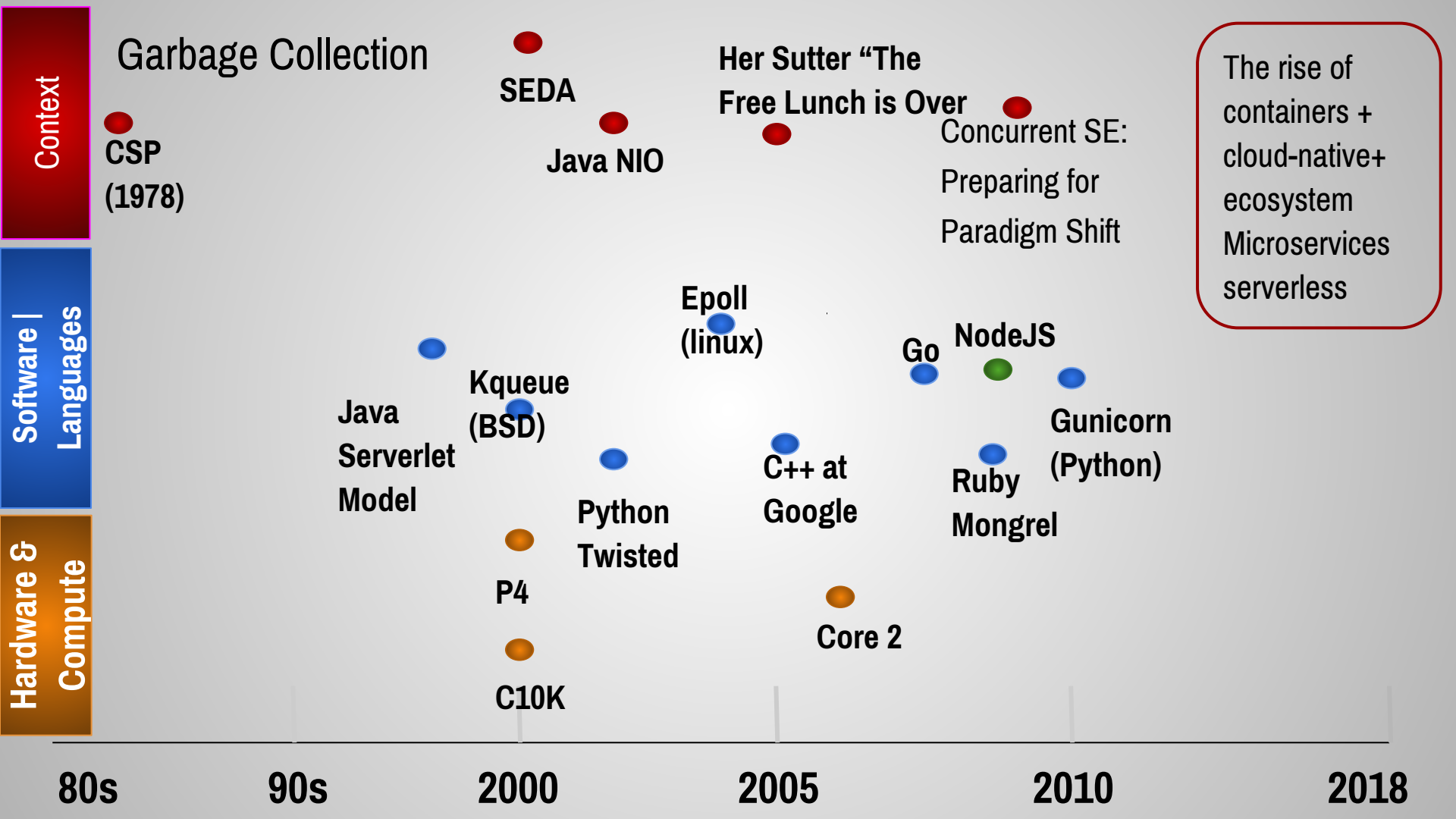
CONCURRENCY - Conclusions

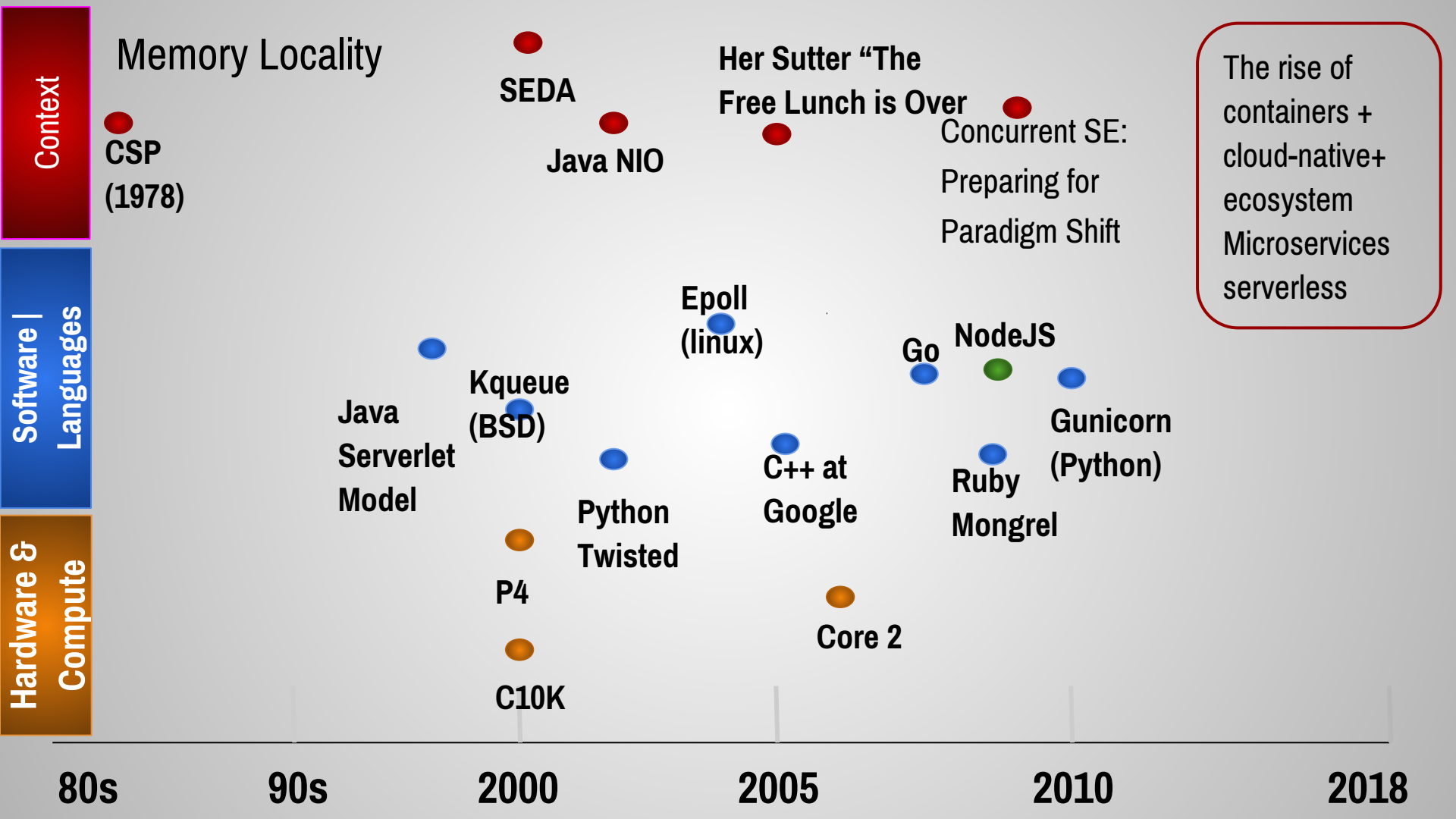
- goroutines are a success: give you the same expressiveness of a traditional imperative programming language while hiding the event driven nature of their interactions with the outside world from the programmer.

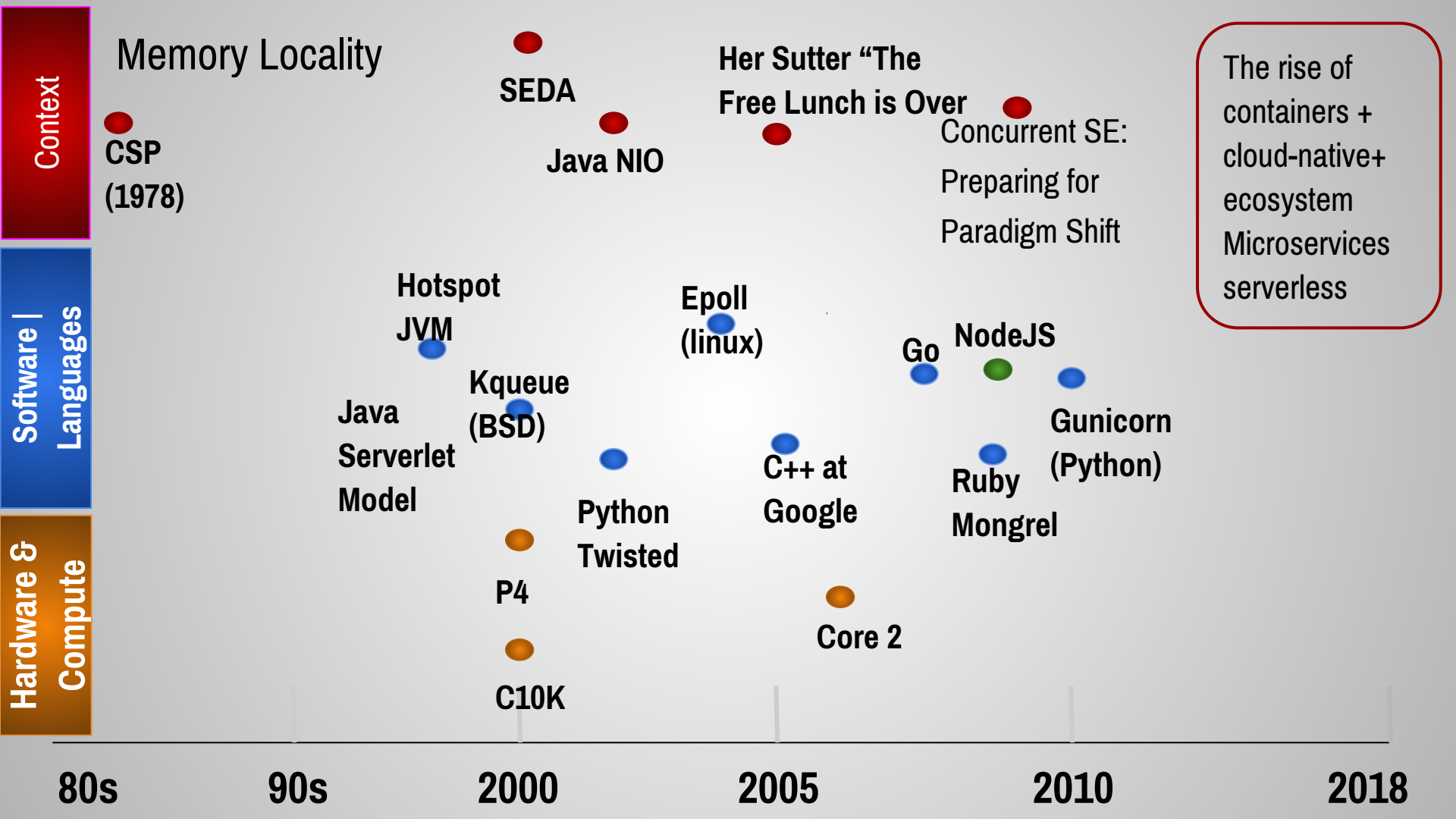
- when goroutines do need to coordinate, they do so in user space, rather than being forced to use expensive kernel interactions.


Context
Software | Languages
Hardware & Compute
Distributed Systems
CSP (1978)
SEDA
Java NIO
Her Sutter "The Free Lunch is Over
Concurrent SE: Preparing for Paradigm Shift
The rise of containers + cloud-native+ ecosystem Microservices serverless
Epoll (linux)
Go
NodeJS
Java Serverlet Model
Kqueue (BSD)
Gunicorn (Python)
C++ at Google
Ruby Mongrel
Python Twisted
P4
Core 2
C10K
80s
90s
2000
2005
2010
2018


Context
Software | Languages
Hardware & Compute
Garbage Collection
CSP (1978)
SEDA
Java NIO
Her Sutter "The Free Lunch is Over
Concurrent SE: Preparing for Paradigm Shift
The rise of containers + cloud-native+ ecosystem Microservices serverless
Epoll (linux)
Go
NodeJS
Java Serverlet Model
Kqueue (BSD)
Gunicorn (Python)
C++ at Google
Ruby Mongrel
Python Twisted
P4
Core 2
C10K
80s
90s
2000
2005
2010
2018


Context
Software | Languages
Hardware & Compute
Garbage Collection
CSP (1978)
SEDA
Java NIO
Her Sutter "The Free Lunch is Over
Concurrent SE: Preparing for Paradigm Shift
The rise of containers + cloud-native+ ecosystem Microservices serverless
Epoll (linux)
Go
NodeJS
Java Serverlet Model
Kqueue (BSD)
Gunicorn (Python)
C++ at Google
Ruby Mongrel
Python Twisted
P4
Core 2
C10K
80s
90s
2000
2005
2010
2018


Context
Software | Languages
Hardware & Compute
Memory Locality
CSP (1978)
SEDA
Java NIO
Her Sutter "The Free Lunch is Over
Concurrent SE: Preparing for Paradigm Shift
The rise of containers + cloud-native+ ecosystem Microservices serverless
Epoll (linux)
Go
NodeJS
Java Serverlet Model
Kqueue (BSD)
Gunicorn (Python)
C++ at Google
Ruby Mongrel
Python Twisted
P4
Core 2
C10K
80s
90s
2000
2005
2010
2018


Context
Software | Languages
Hardware & Compute
Memory Locality
CSP (1978)
SEDA
Java NIO
Her Sutter "The Free Lunch is Over
Concurrent SE: Preparing for Paradigm Shift
The rise of containers + cloud-native+ ecosystem Microservices serverless
Hotspot JVM
Epoll (linux)
Go
NodeJS
Kqueue (BSD)
Java Serverlet Model
Gunicorn (Python)
C++ at Google
Ruby Mongrel
Python Twisted
P4
Core 2
C10K
80s
90s
2000
2005
2010
2018

Memory & Data Locality

Python

```
% python
>>> from sys import getsizeof
>>> gocon = 2014
>>> getsizeof(gocon)
24
```

Memory & Data Locality

Java

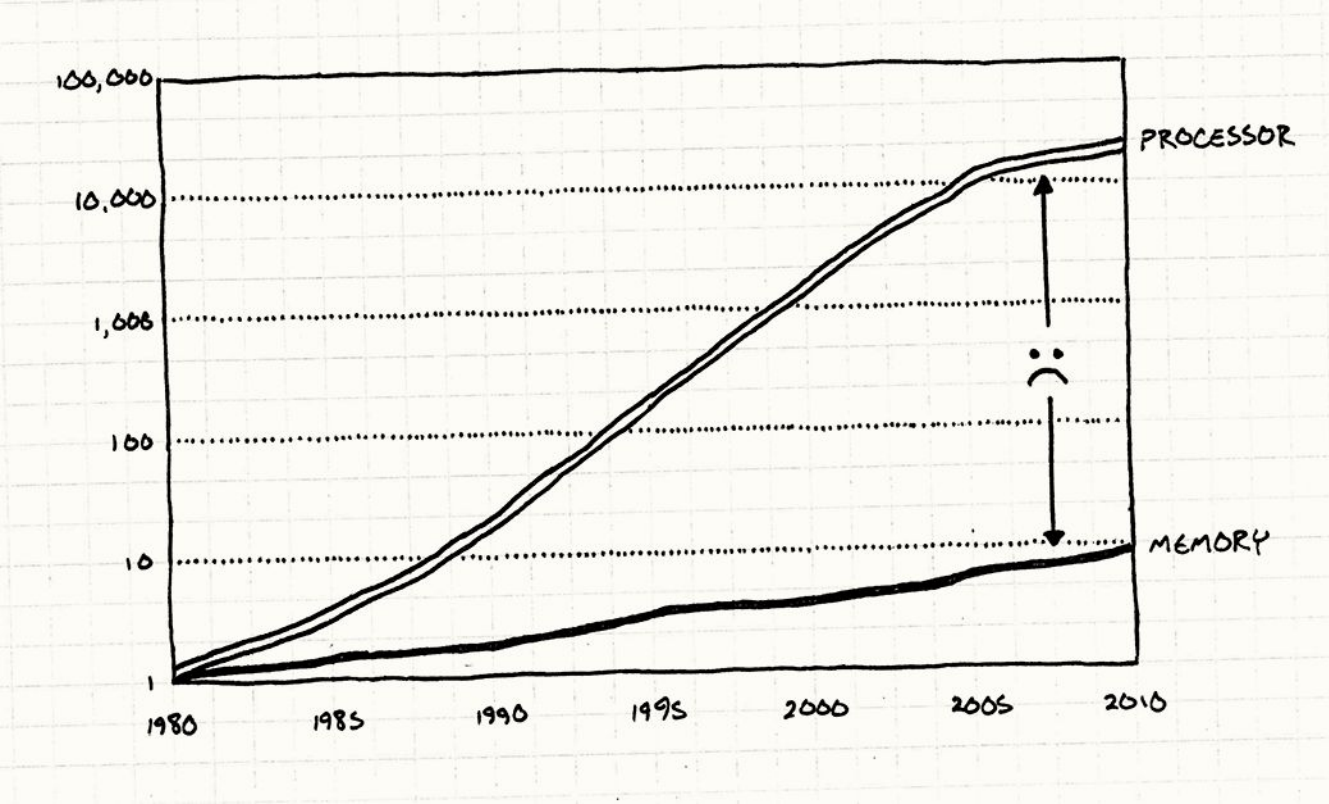
```
int gocon = 2014;
```

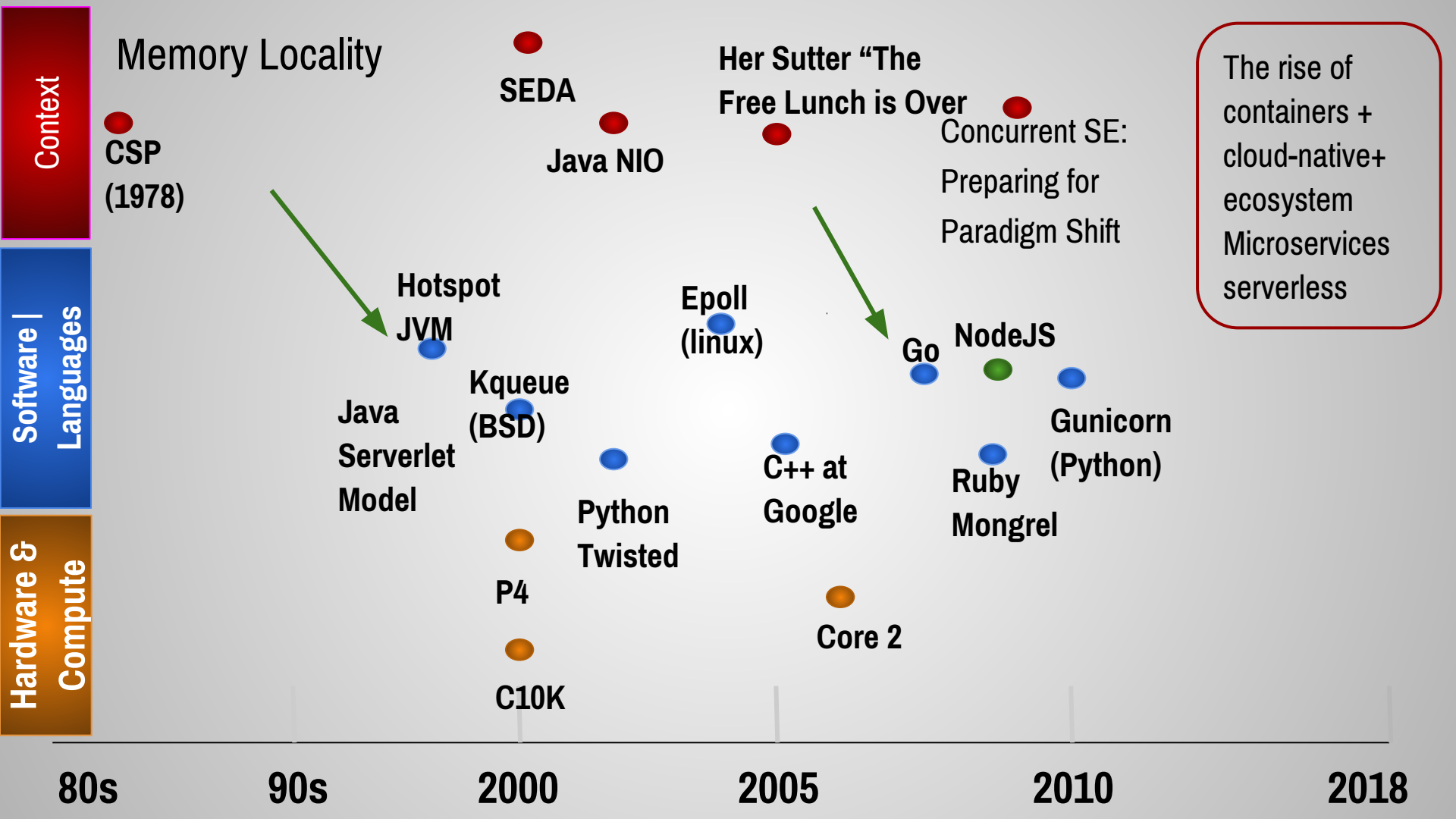
Memory & Data Locality

Java

```
// 16 bytes on 32 bit JVM
// 24 bytes on 64 bit JVM
Integer gocon = new Integer(2014);
```

Memory & Data Locality




Context
Software | Languages
Hardware & Compute
Memory Locality
CSP (1978)
SEDA
Java NIO
Her Sutter "The Free Lunch is Over
Concurrent SE: Preparing for Paradigm Shift
The rise of containers + cloud-native+ ecosystem Microservices serverless
Hotspot JVM
Java Serverlet Model
Kqueue (BSD)
Python Twisted
P4
C10K
Epoll (linux)
C++ at Google
Core 2
Go
NodeJS
Ruby Mongrel
Gunicorn (Python)
80s
90s
2000
2005
2010
2018

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

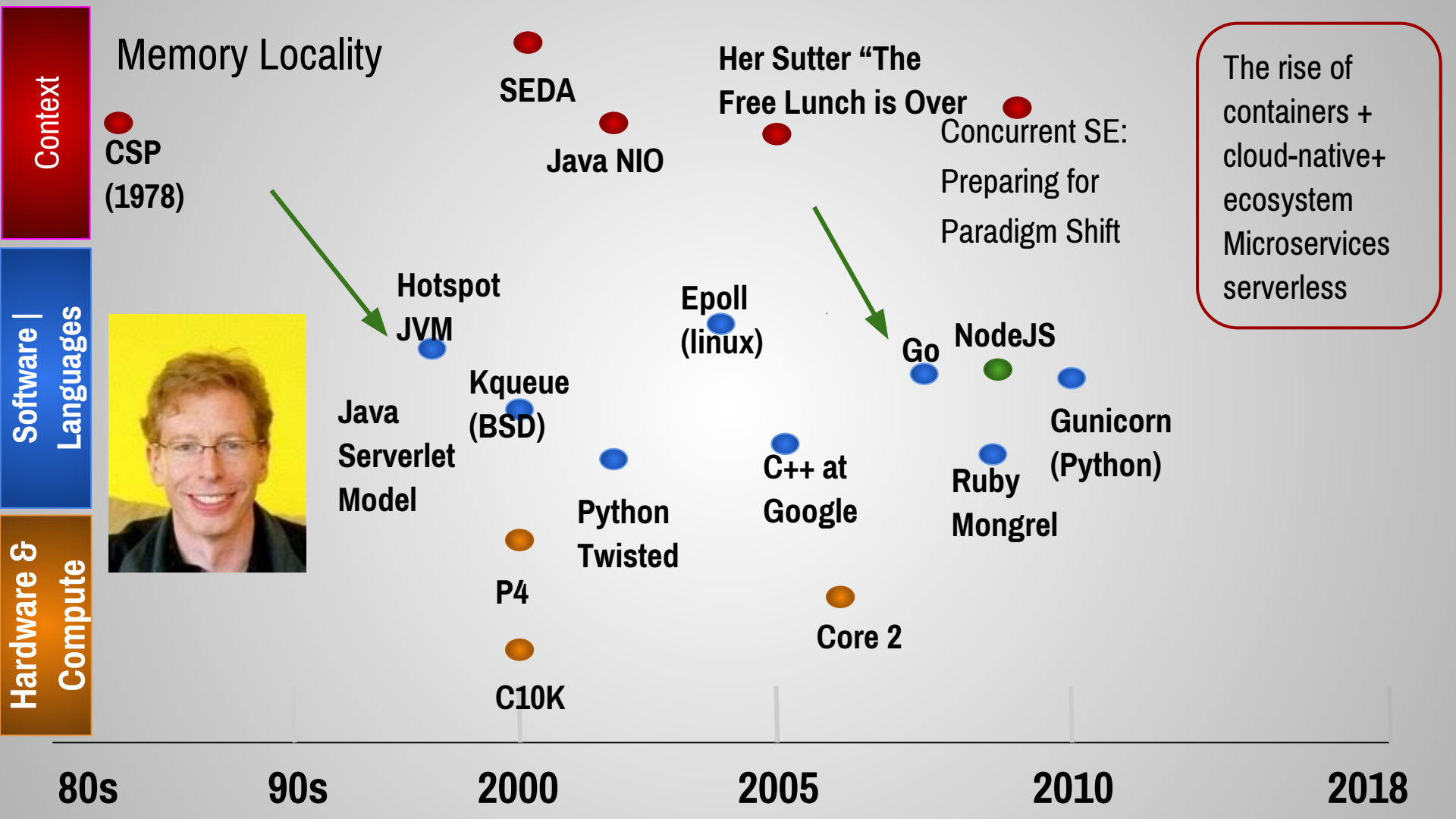
Everything Allocated

Can’t return multiple values (until 2018)

Go

Structs

True Value types


Context
Software | Languages
Hardware & Compute
Memory Locality
CSP (1978)
SEDA
Java NIO
Her Sutter "The Free Lunch is Over
Concurrent SE: Preparing for Paradigm Shift
The rise of containers + cloud-native+ ecosystem Microservices serverless
Hotspot JVM
Java Serverlet Model
Kqueue (BSD)
Python Twisted
P4
C10K
Epoll (linux)
C++ at Google
Core 2
Go
NodeJS
Ruby Mongrel
Gunicorn (Python)
80s
90s
2000
2005
2010
2018

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.

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	Go
No value types	Structs
Everything Allocated	True Value types
Can’t return multiple values	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 and Data 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)

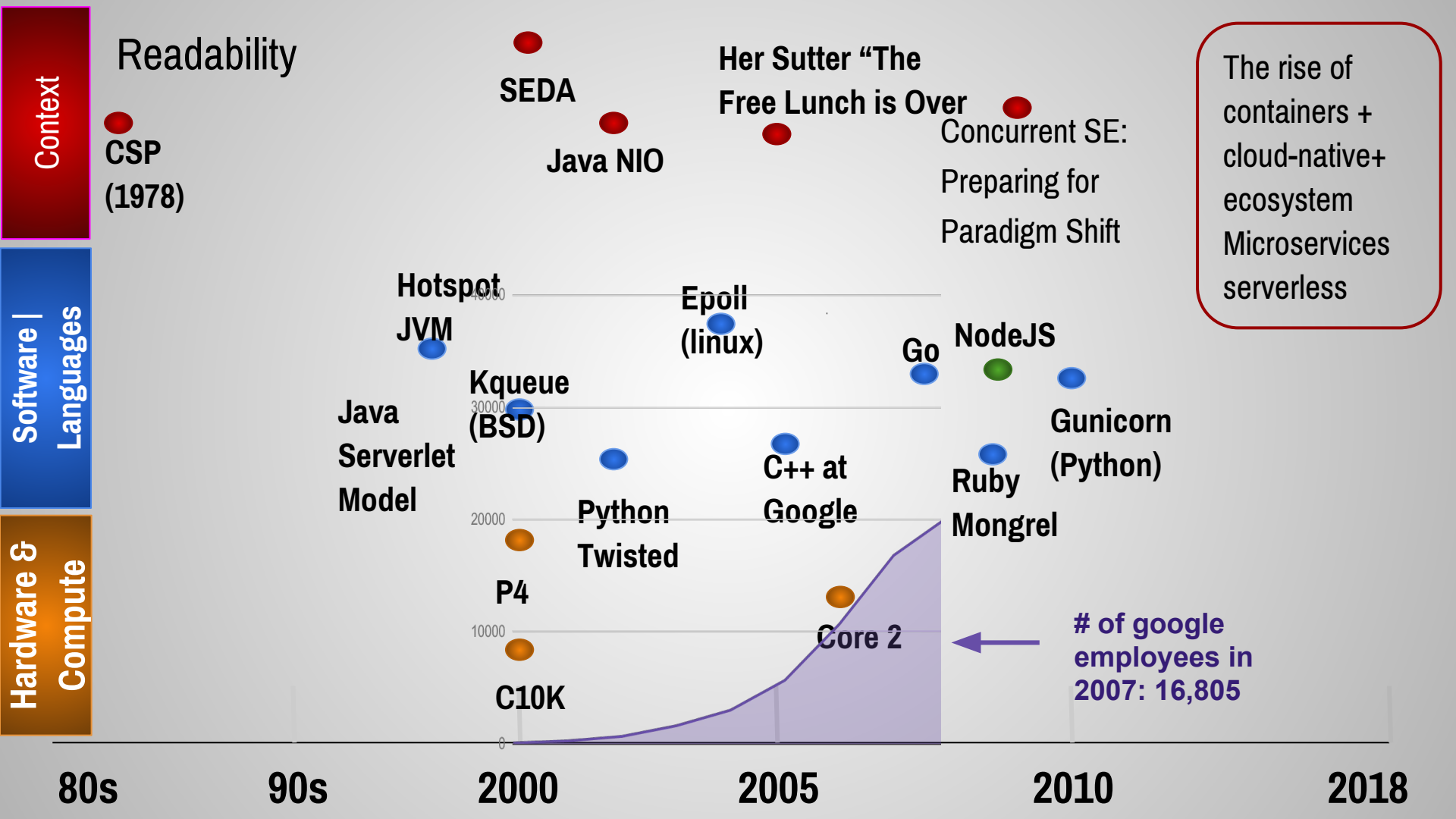
- Go gives programmers the tools to talk about memory efficiently ***if they need it.***
- Flexible
- Memory management (not an all-or-nothing like in C++ or Rust)

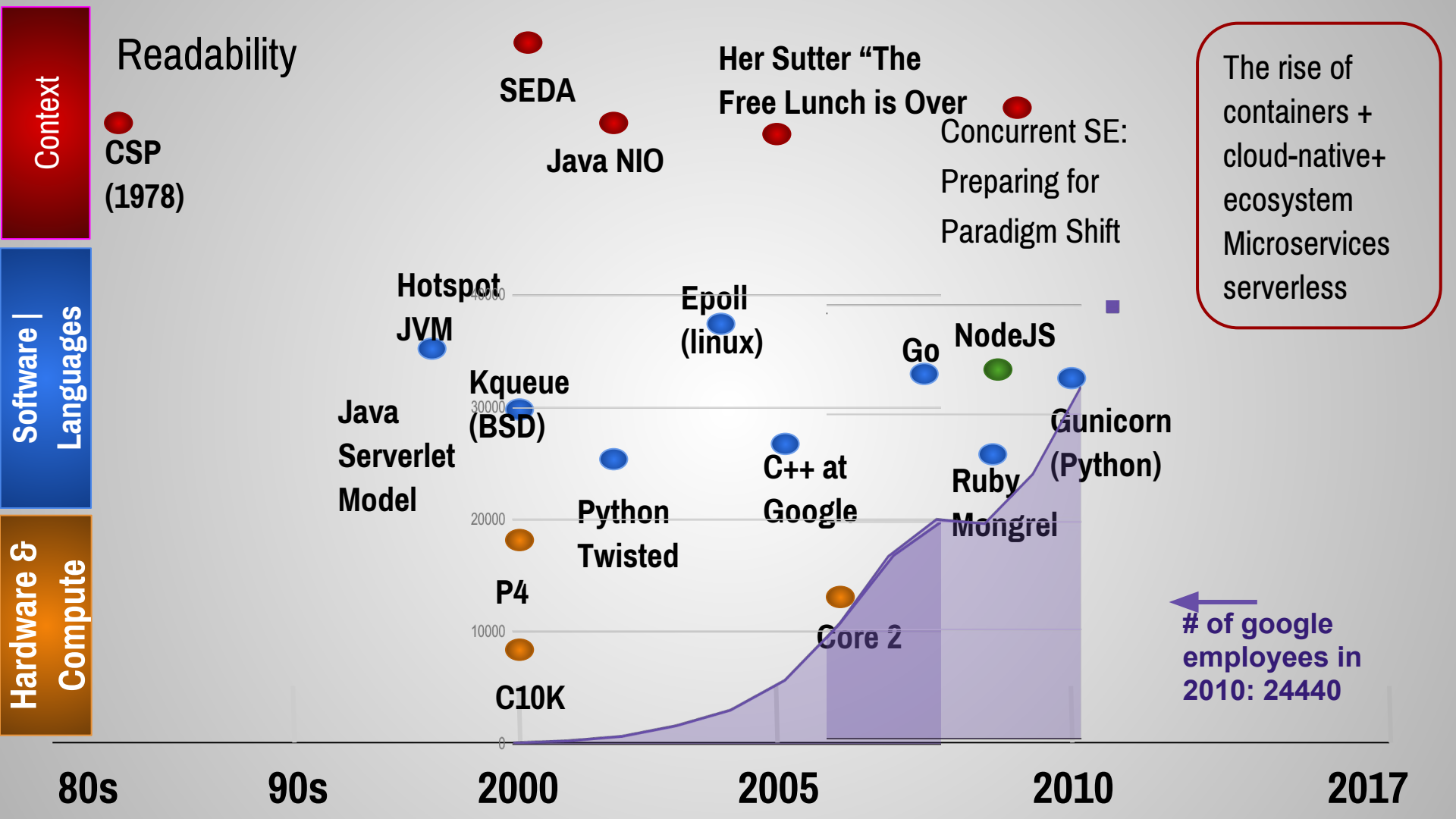
Readability

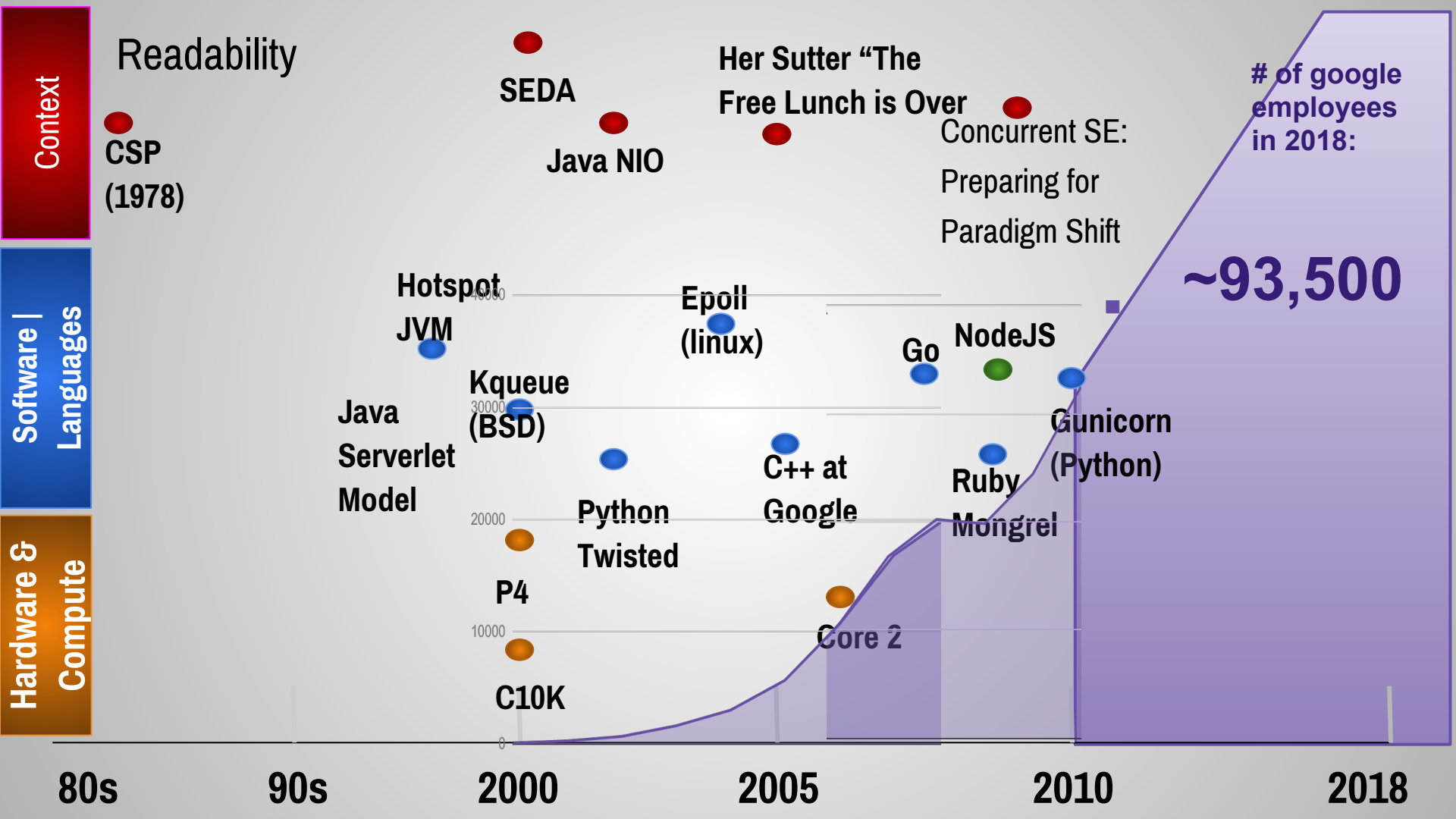
Readability is paramount—Rob Pike

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


Context
Software | Languages
Hardware & Compute
Readability
CSP (1978)
SEDA
Java NIO
Her Sutter "The Free Lunch is Over
Concurrent SE: Preparing for Paradigm Shift
The rise of containers + cloud-native+ ecosystem Microservices serverless
Hotspot JVM
Epoll (linux)
Go
NodeJS
Kqueue (BSD)
Java Serverlet Model
Python Twisted
C++ at Google
Ruby Mongrel
Gunicorn (Python)
P4
C10K
Core 2
# of google employees in 2007: 16,805
40000
30000
20000
10000
0
80s
90s
2000
2005
2010
2018


Context
Software | Languages
Hardware & Compute
Readability
CSP (1978)
SEDA
Java NIO
Her Sutter "The Free Lunch is Over
Concurrent SE: Preparing for Paradigm Shift
The rise of containers + cloud-native+ ecosystem Microservices serverless
Hotspot JVM
Epoll (linux)
Go
NodeJS
Kqueue (BSD)
Java Serverlet Model
Gunicorn (Python)
C++ at Google
Ruby Mongrel
Python Twisted
P4
Core 2
C10K
40000
30000
20000
10000
0
# of google employees in 2010: 24440
80s
90s
2000
2005
2010
2017


Context
Software | Languages
Hardware & Compute
Readability
CSP (1978)
SEDA
Java NIO
Her Sutter "The Free Lunch is Over
Concurrent SE: Preparing for Paradigm Shift
# of google employees in 2018:
~93,500
Hotspot JVM
Epoll (linux)
Go
NodeJS
Kqueue (BSD)
Java Serverlet Model
Gunicorn (Python)
C++ at Google
Ruby Mongrel
Python Twisted
P4
Core 2
C10K
40000
30000
20000
10000
0
80s
90s
2000
2005
2010
2018

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

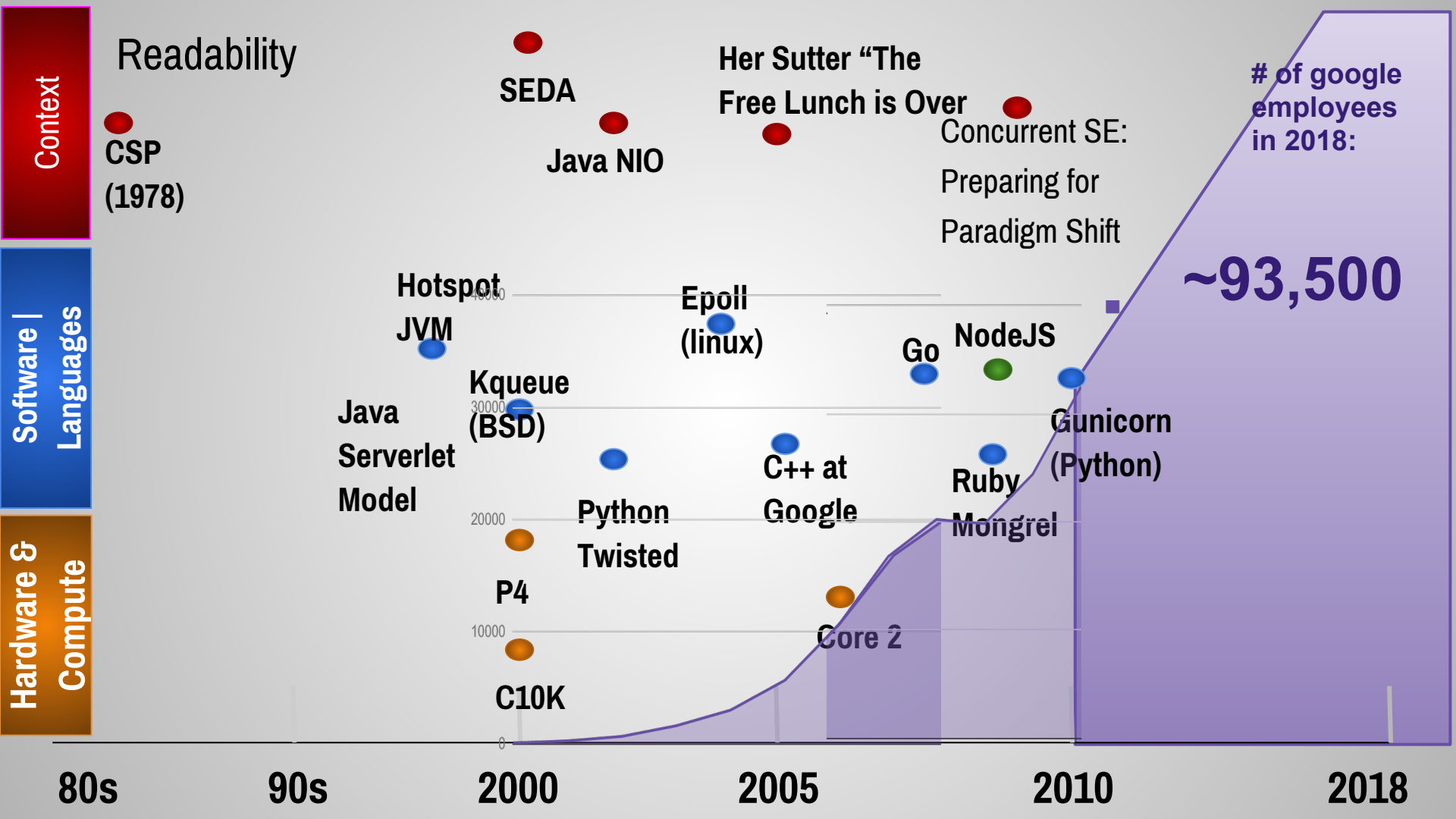
Readability

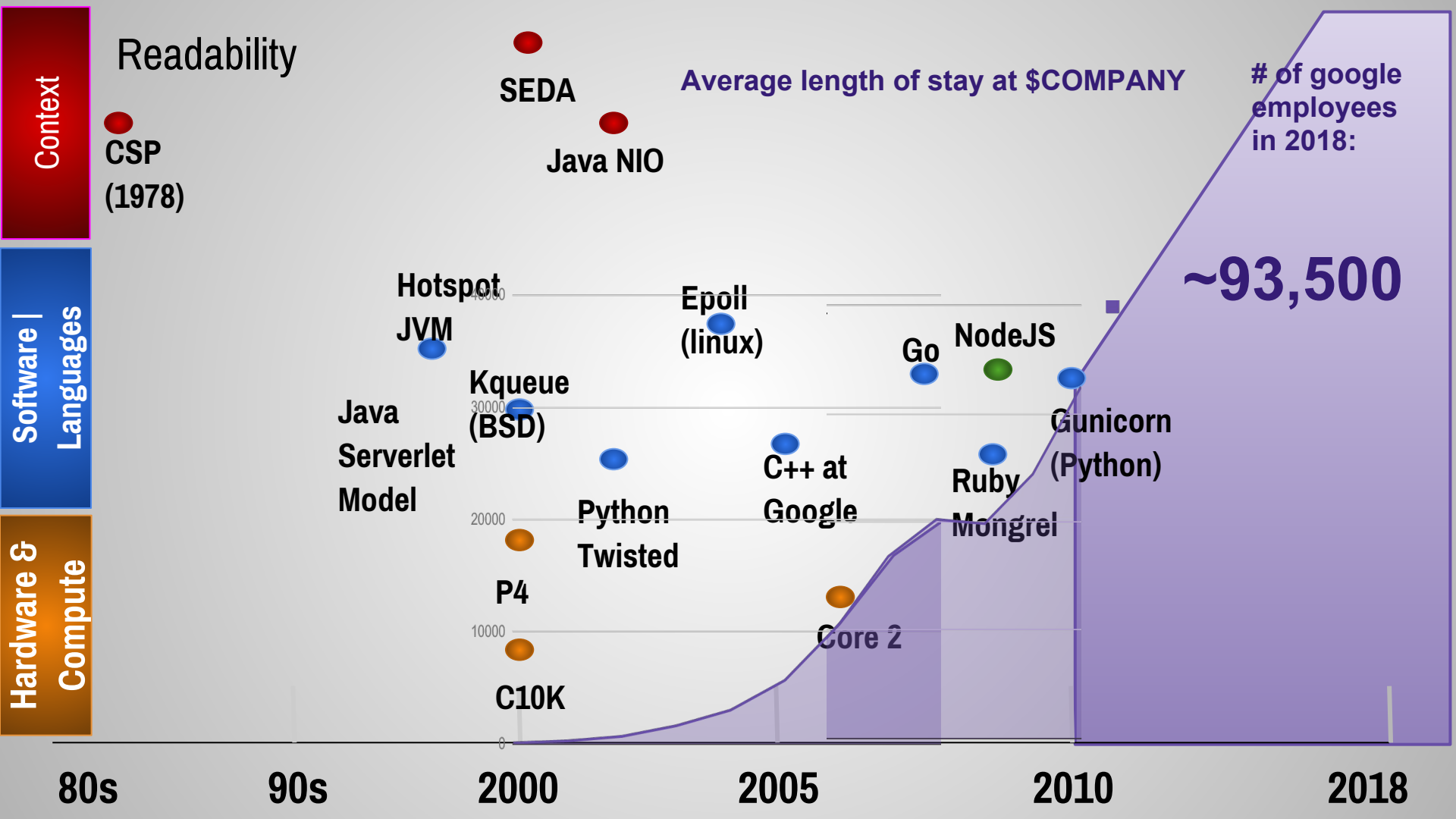
Programs which cannot be maintained will be rewritten

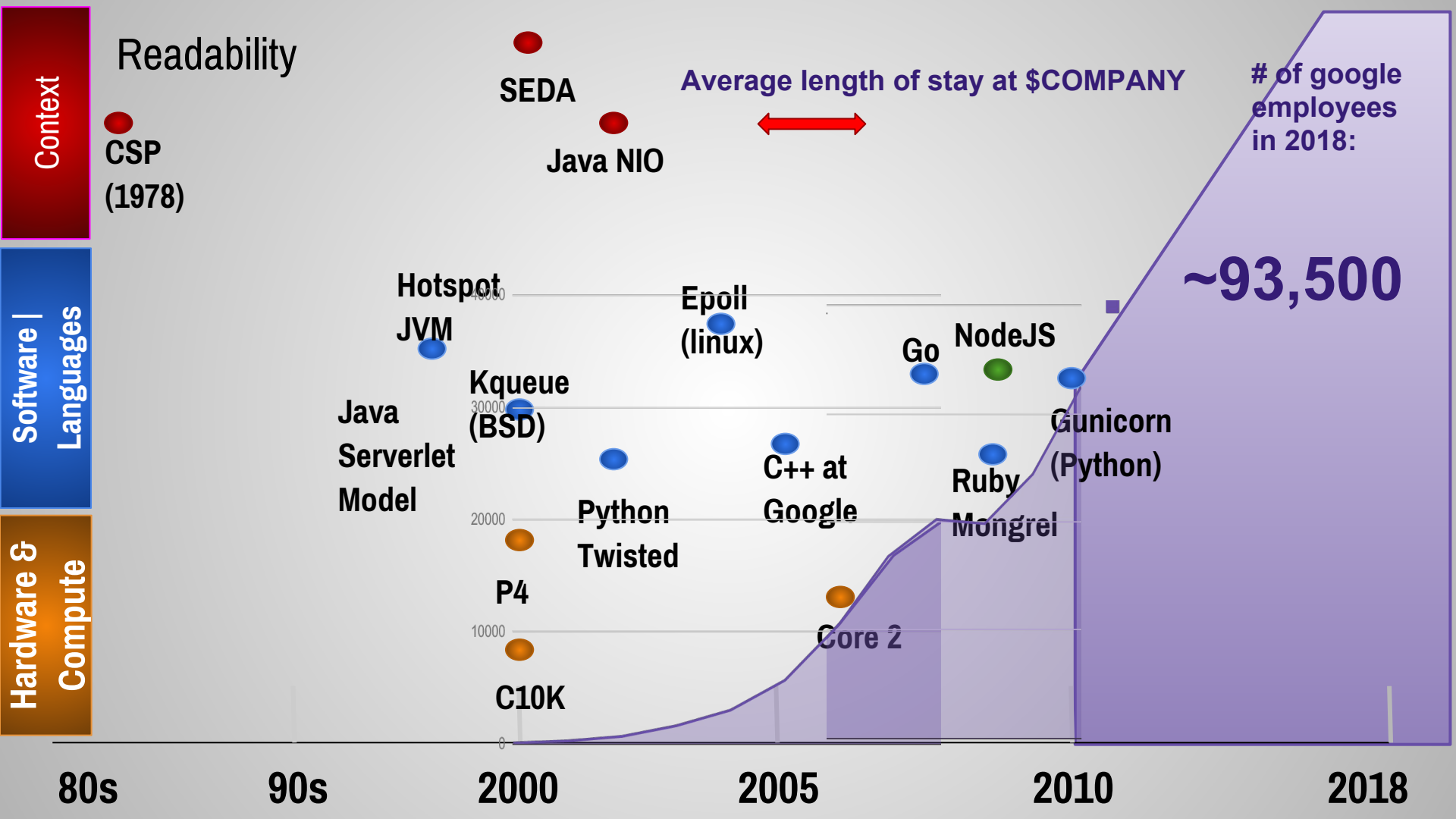
Readability

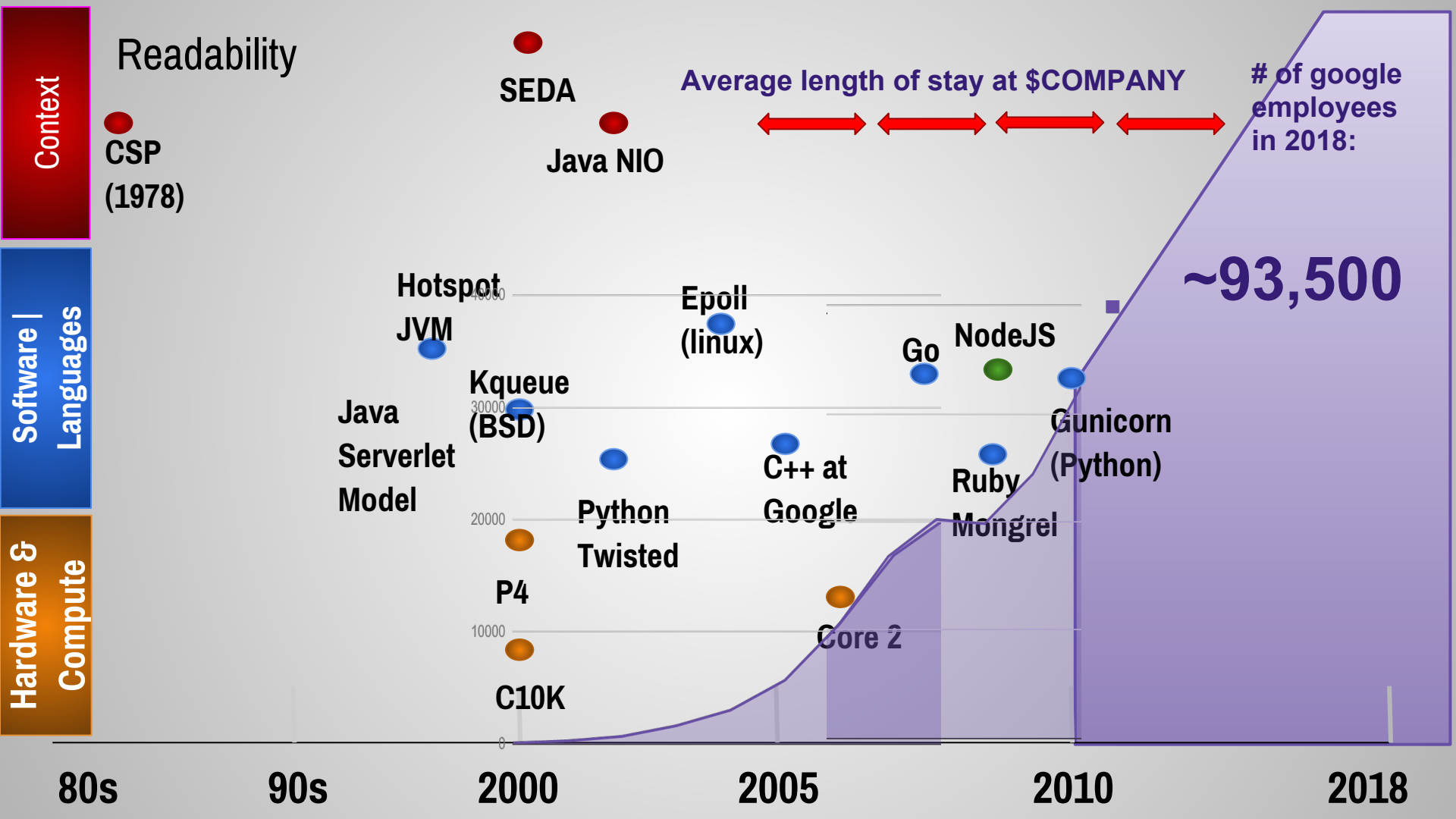
Programs which cannot be maintained will be rewritten

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

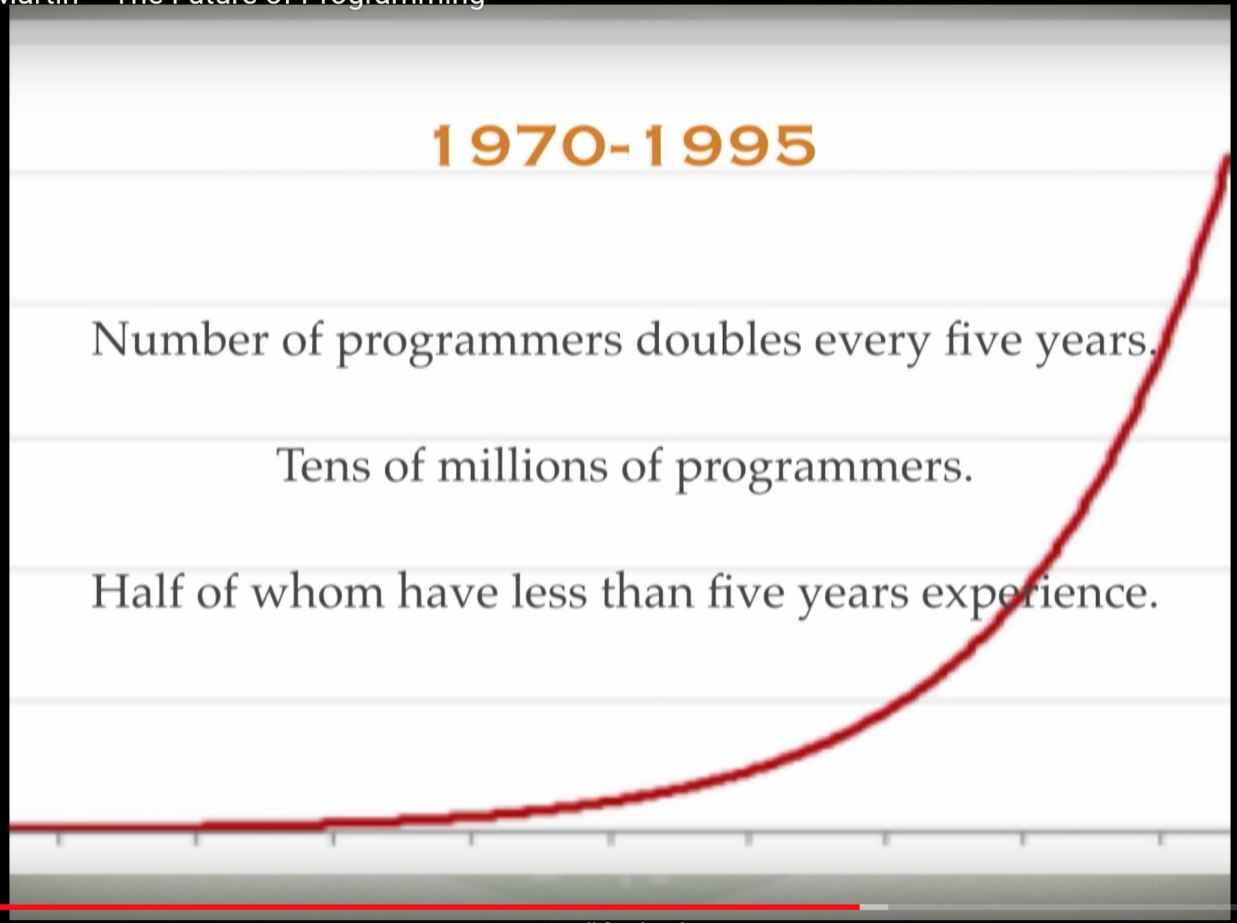

Context
Software | Languages
Hardware & Compute
Readability
CSP (1978)
SEDA
Java NIO
Her Sutter "The Free Lunch is Over
Concurrent SE: Preparing for Paradigm Shift
# of google employees in 2018:
~93,500
Hotspot JVM
Epoll (linux)
Go
NodeJS
Kqueue (BSD)
Java Serverlet Model
Gunicorn (Python)
C++ at Google
Ruby Mongrel
Python Twisted
P4
Core 2
C10K
40000
30000
20000
10000
0
80s
90s
2000
2005
2010
2018


Context
Software | Languages
Hardware & Compute
Readability
CSP
(1978)
SEDA
Java NIO
Average length of stay at $COMPANY
# of google employees in 2018:
~93,500
Hotspot
JVM
Epoll
(linux)
Go
NodeJS
Kqueue
(BSD)
Java
Serverlet
Model
Gunicorn
(Python)
C++ at
Google
Ruby
Mongrel
Python
Twisted
P4
Core 2
C10K
40000
30000
20000
10000
0
80s
90s
2000
2005
2010
2018


Context
Software | Languages
Hardware & Compute
Readability
CSP
(1978)
SEDA
Java NIO
Average length of stay at $COMPANY
# of google employees in 2018:
~93,500
Hotspot
JVM
Epoll
(linux)
Go
NodeJS
Kqueue
(BSD)
Java
Serverlet
Model
Gunicorn
(Python)
C++ at
Google
Ruby
Mongrel
Python
Twisted
P4
Core 2
C10K
40000
30000
20000
10000
0
80s
90s
2000
2005
2010
2018


Context
Software | Languages
Hardware & Compute
Readability
CSP
(1978)
SEDA
Java NIO
Average length of stay at $COMPANY
# of google employees in 2018:
~93,500
Hotspot
JVM
Epoll
(linux)
Go
NodeJS
Kqueue
(BSD)
Java
Serverlet
Model
Gunicorn
(Python)
C++ at
Google
Ruby
Mongrel
Python
Twisted
P4
Core 2
C10K
40000
30000
20000
10000
0
80s
90s
2000
2005
2010
2018

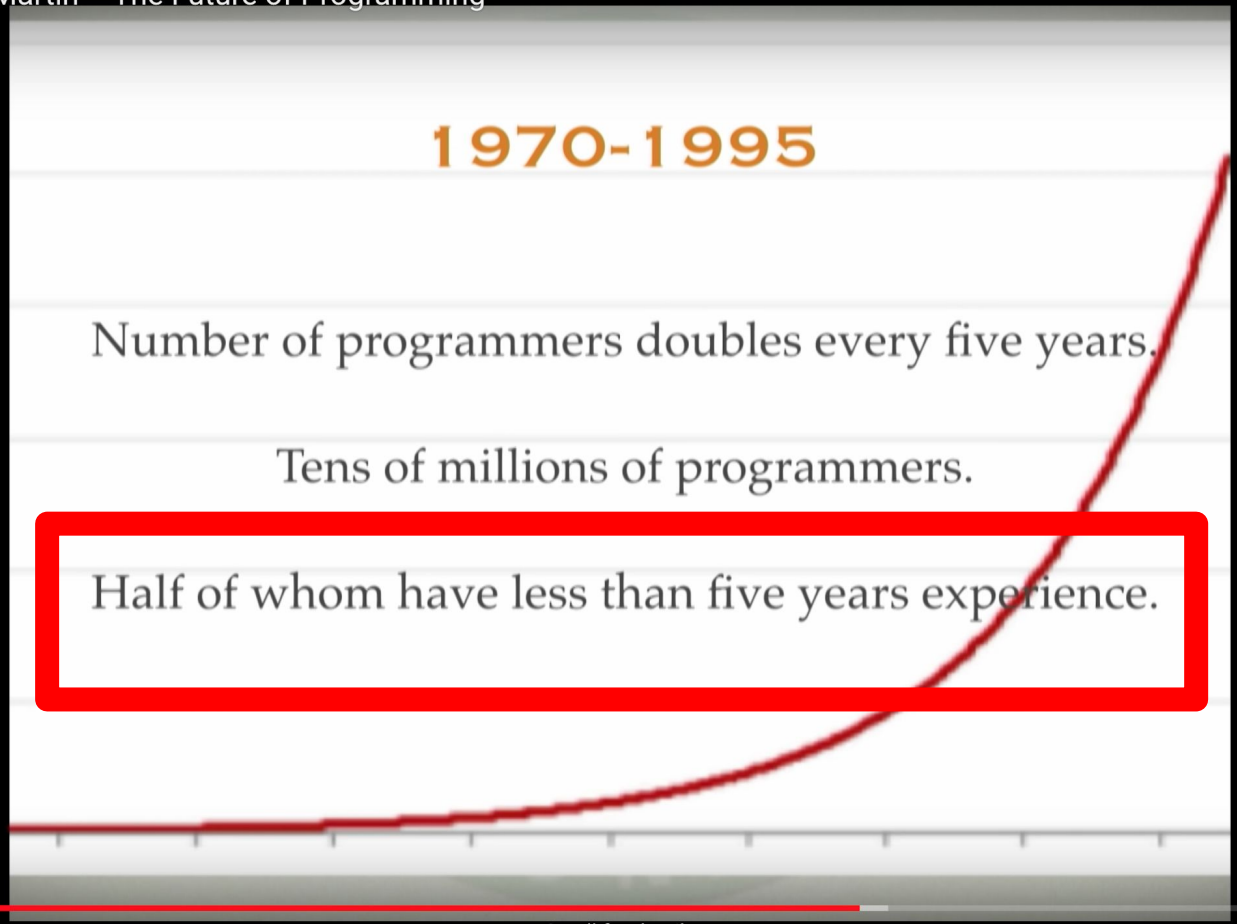
"Uncle" Bob Martin - "The Future of Programming"



51:05 / 1:18:20

Scroll for details

"Uncle" Bob Martin - "The Future of Programming"



51:05 / 1:18:20

Scroll for details

Readability

N+1 different opinions on what makes readability.

Familiarity - Go’s small language footprint (25 keywords)

Readability

Familiarity - Go’s small language footprint (25 keywords)

Ingroup knowledge

Readability

Familiarity - Go’s small language footprint (25 keywords)

Ingroup knowledge

Learners perspectives - First time seeing code

No “magic” - easy vs. Simple

Readability

Familiarity - Go’s small language footprint (25 keywords)

Ingroup knowledge

Learners perspectives - First time seeing code

No “magic” - easy vs. Simple

HTML w3org: 3.2. Priority of Constituencies: User > Author > Implementor

Readability

Personal Convenience.

Convention over Configuration

Readability

~~Personal Convenience.~~

Convention over Configuration

Readability

~~Personal Convenience.~~

~~Convention over Configuration~~

Simplicity

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

TIME TRAVELING EVERYWHERE

CI

Unit Tests

Refactoring

Design Patterns

Dependency Management

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

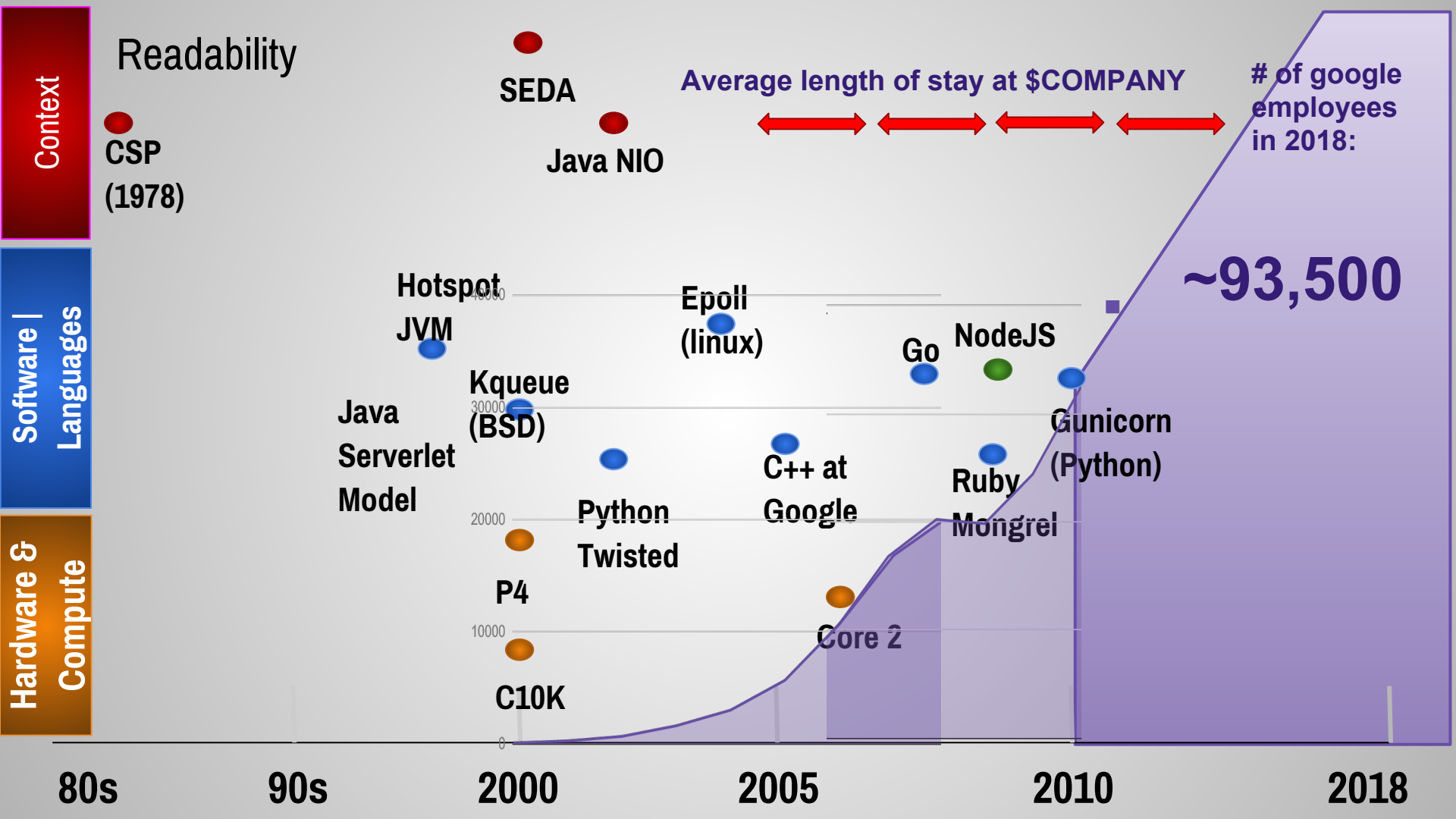

KEEP
LEFT
NO
ENTRY
NO
RIGHT
TURN


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
Readability
CSP
(1978)
SEDA
Java NIO
Average length of stay at $COMPANY
# of google employees in 2018:
~93,500
Hotspot
JVM
Epoll
(linux)
Go
NodeJS
Kqueue
(BSD)
Java
Serverlet
Model
Gunicorn
(Python)
C++ at
Google
Ruby
Mongrel
Python
Twisted
P4
Core 2
C10K
40000
30000
20000
10000
0
80s
90s
2000
2005
2010
2018

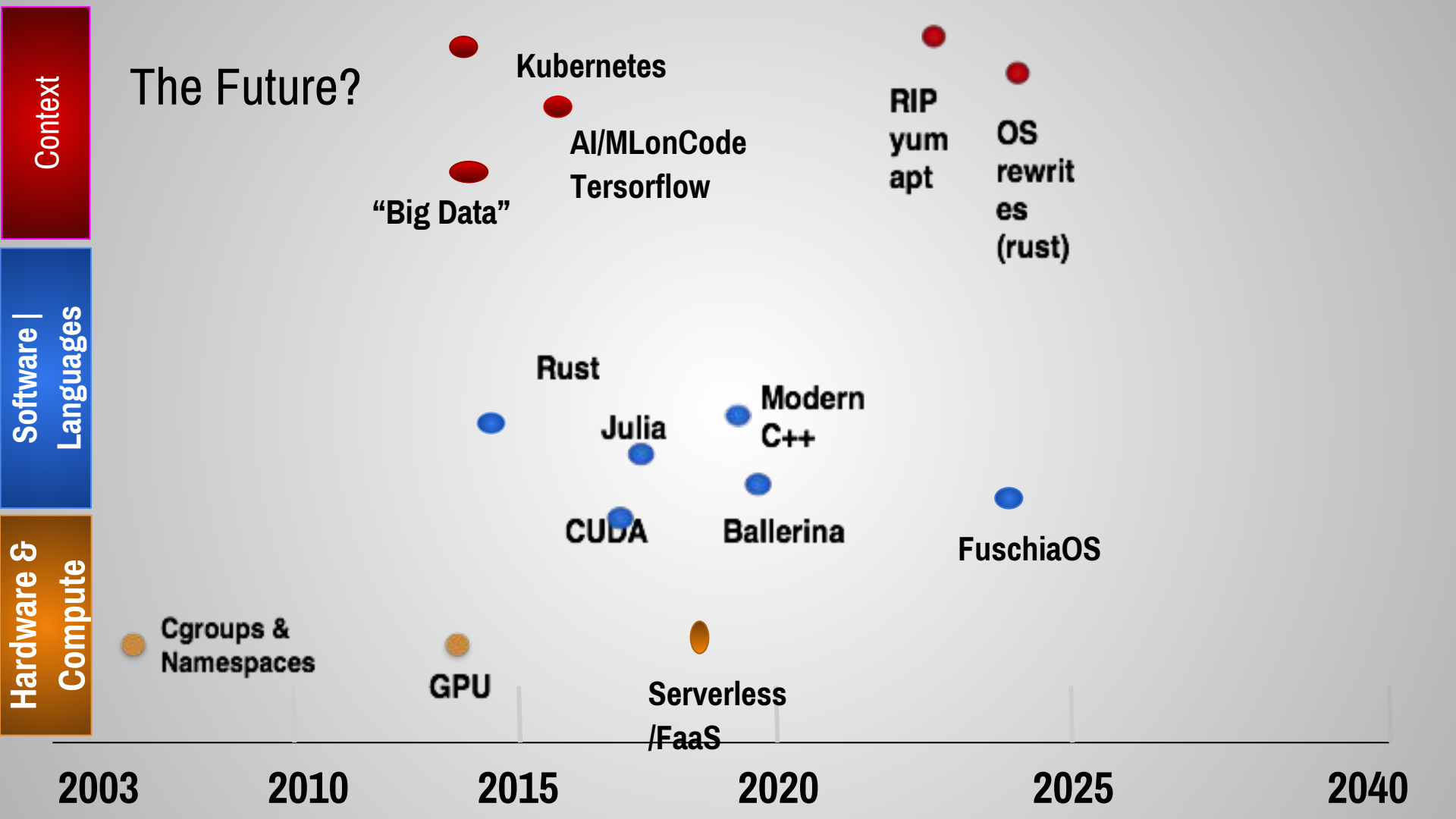
Context

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.


Context
Software | Languages
Hardware & Compute
The Future?
Kubernetes
AI/MLonCode
Tersorflow
"Big Data"
RIP
yum
apt
OS
rewrit
es
(rust)
Rust
Julia
Modern
C++
CUDA
Ballerina
FuschiaOS
Cgroups &
Namespaces
GPU
Serverless
/FaaS
2003
2010
2015
2020
2025
2040

Context

Software | Languages

Hardware & Compute

The Future?

...hang on for the ride

2017 2020 2025 2030 2035 2040

Thank you!

Carmen Andoh
@carmatrocity
GOTO Copenhagen

November 2018