Un nouveau langage, Go. Pour quoi faire ?

Un nouveau langage, Go. Pour quoi faire ?

- **Un langage pour dominer le monde**
- **Ou accompagner une rupture, un changement d'époque…**

Un nouveau langage, Go. Pour quoi faire ?

- **l'OS (Unix), le langage (C) [Ritchie, Thompson, Kerninghan]**
  - Portabilité
  - Adaptabilité
  - Compilateur simple et performant
    - Une recette pour dominer le monde, du PDP-11 au téléphone portable !
- **Aujourd'hui, quelle évolution ?**
  - Les langages de programmation ont évolués
  - Ont-ils évolués dans le bon sens ?
  - Faut-il de nouveaux langages pour dominer le monde
    - Ou tout simplement résoudre nos problèmes d'aujourd'hui et de demain...

Un nouveau langage, Go. Pour quoi faire ?

- **Intéressons nous à ce langage de programmation, Go**
- **Il est improbable…**
- **L'évolution de C existe**
  - Elle s'appelle C++
  - C++ intègre tout et sait tout faire
  - C++ génère du code très performant
  - C++ compile de plus en plus vite
  - C++ fait du parallélisme
  - C++ fait de la méta-programmation
  - C++ domine l'industrie
    - Ah ben non, plein de gens programment encore en C
    - Et d'autres programment en Fortran, Python, Java, C#, Delphi, etc...

Un nouveau langage, Go. Pour quoi faire ?

- **Pourquoi différents langages de programmation ?**
  - Parce qu'on a pas trouvé le langage idéal, celui qui est parfait pour toutes les situations
  - Un langage de programmation est un compromis
    - Plusieurs langages / différentes positions dans l'espace
    - Facile / sûr / performant / lisible / extensible / simple / adapté / élégant / pur
  - Un état de l'art : un contexte d'utilisation du langage
    - Hier, un processeur un cœur pour plusieurs dizaines d'utilisateurs
    - Aujourd'hui : plusieurs dizaines de cœurs pour un seul utilisateur
    - Demain ?
  - Un savoir : en langages et génie logiciel
    - Nouveaux concepts
    - Nouvelles exigences

Un nouveau langage, Go. Pour quoi faire ?

- **Please remember that the design of a programming language consists not just in a laundry list of features, but also in making judicious choices of what to *omit* and, more importantly, in establishing design principles that are easy to understand [Steele, 2003]**

Un nouveau langage, Go. Pour quoi faire ?

- **Depuis C, quelles sont les évolutions ?**
  - Logiciels plus complexes
  - Concepts supportant des architectures logicielles plus vastes
    - Modules, Objets, Types de données abstraits
  - Concepts rendant ces architectures plus extensibles
    - Fermetures (Closures)
  - Concepts permettant de structurer ces architectures
    - Design patterns : patrons de conception
- **C et proches restent**
  - Code efficace
  - Proche de la machine
  - Mais incapable d'assumer la complexité actuelle (et peu sûr!)

Un nouveau langage, Go. Pour quoi faire ?

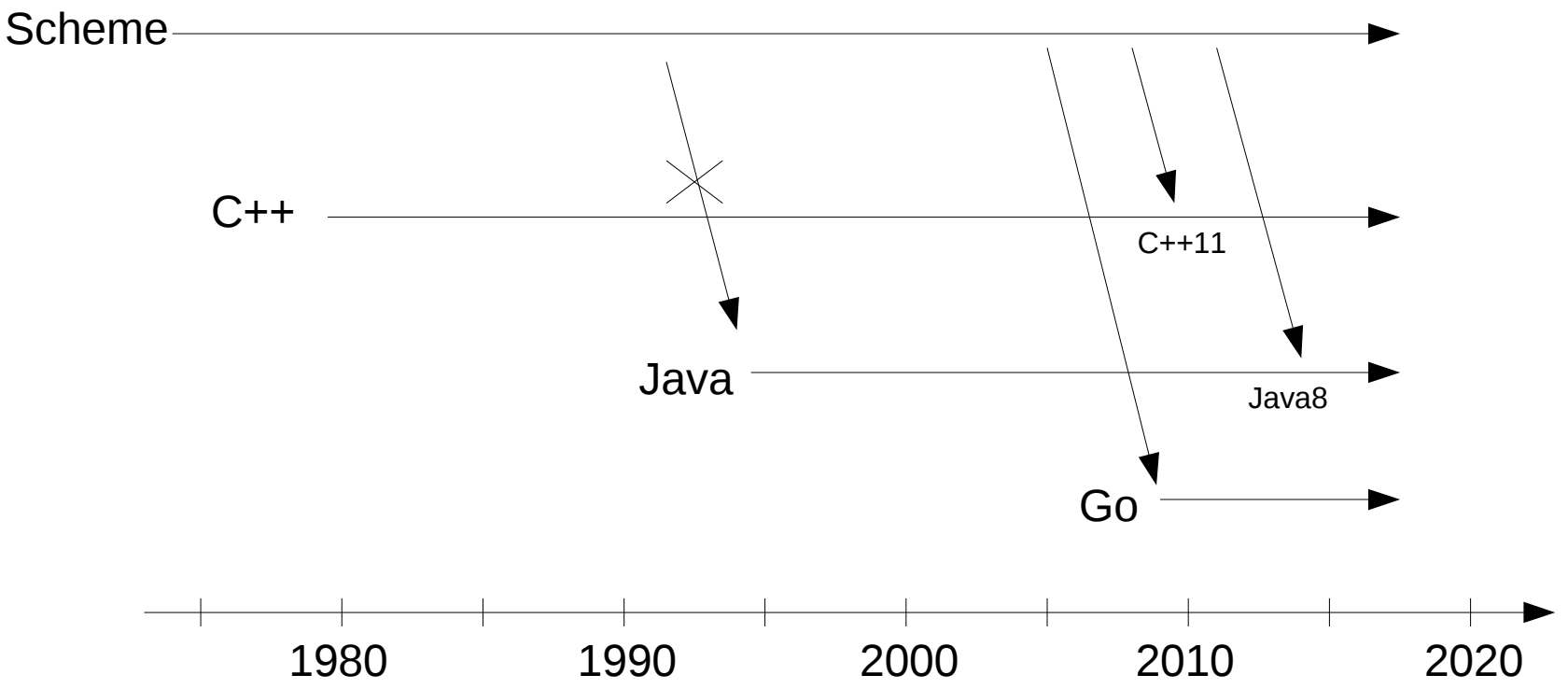
- **Importance d'un langage de programmation**
  - Nous isole du matériel, de la machine.
  - Nous donne des outils / concepts pour exprimer des algorithmes
  - Nous donne des outils pour maîtriser la complexité
  - Nous donne des outils pour exploiter la puissance du matériel
    - Parallélisme, multi-many cœurs
- **Langage de programmation : l'IHM du programmeur**
  - Forme / Contraint la manière dont on résout nos problèmes
  - Rend facile certaines formes / structures / manières de travailler
  - Rend difficile des formes que le concepteur ne souhaite pas
    - Pas toujours conscient

Un nouveau langage, Go. Pour quoi faire ?

- **Revenons à Go [Griesemer, Pike, Thompson]**
  - Go n'est pas un langage à la pointe de la recherche
    - Tout ce que Go propose et contient existe déjà dans d'autres langages
    - D'autres communautés se chargent de repousser les limites !
  - Mais : le pouvoir d'un langage de programmation est dans
    - Le choix des fonctions / concepts qu'il intègre
    - La manière dont elles sont intégrées
- **Et tirer profit des leçons de l'histoire**
  - En fait, nous avons déjà suffisamment d'histoire pour pouvoir dire ce qui marche bien, ce qui marche moins bien.
  - Et suffisamment de science pour dire ce qui est essentiel

• Un nouveau langage, Go. Pour quoi faire ?

- **Exemple : les closures (Go fonctions anonymes)**
  - Définis avec Scheme [Steele ~75]. Considérées comme essentielles



Un nouveau langage, Go. Pour quoi faire ?

- **Un résumé des choix de Go en un transparent**
  - Proche de C (avec des { })
  - Modularité par composition (suffit pour programmer « large »)
  - Lambdas/Fermetures sont essentielles
  - Goroutines/CSP pour le parallélisme multi-coeur et many-coeur
    - Avec une pile adaptée pour rendre plus performant
  - Types statiques pour la performance et la structure
  - Interfaces pour rendre les types extensibles
  - Duck typing !
  - Les types complexes sont trop complexes
  - Les cas simples sont statiques ;
    - Les cas complexes sont résolus à l'exécution

Un nouveau langage, Go. Pour quoi faire ?

- **Un résumé des choix de Go en ~~un~~ deux transparents**
  - La mémoire est gérée automatiquement (GC)
  - Pas d'exceptions, mais
    - « panic »
    - Et des retours d'erreurs
  - De l'assignation multiple
  - Des outils avancés
    - Compilateur, parseur, réflexion.
  - De la compilation rapide !
    - Surtout pour des applications complexes avec beaucoup de modules !
  - De la simplification de la syntaxe
    - Déclarations de types, inférence de types, non-redondance si possible

Un nouveau langage, Go. Pour quoi faire ?

- **Intérêt de Go ?**
  - Rend ce que Go cible
    - Simple à programmer
    - Aisé à maîtriser
  - Ce que Go cible
    - Est le monde actuel
    - Multi-coeur, many-coeur
    - Cloud / déploiement simplifié
    - Gestion mémoire simplifiée
- **En résumé**
  - Go est beaucoup plus proche des langages des années 1980 (simplicité)
  - Avec tous les vrais bénéfices de la recherche depuis
    - Closure, composition, csp, méta-programmation

Un nouveau langage, Go. Pour quoi faire ?

- **Go : Composition**
  - Composition
    - En lieu et place de l'héritage

```
type ReadWriter interface {
    Reader
    Writer
}

// ReadWriter stores pointers to a
Reader and a Writer.
// It implements io.ReadWriter.
type ReadWriter struct {
    *Reader  // *bufio.Reader
    *Writer  // *bufio.Writer
}
```

Un nouveau langage, Go. Pour quoi faire ?

- **Go : Retour multiple**
  - Simplifie le retour de résultats / l'association résultat / erreur

```
func nextInt(b []byte, i int) (int, int) {
    for ; i < len(b) && !isDigit(b[i]); i++ {
    }
    x := 0
    for ; i < len(b) && isDigit(b[i]); i++ {
        x = x*10 + int(b[i]) - '0'
    }
    return x, i
}
...
    for i := 0; i < len(b); {
        x, i = nextInt(b, i)
        fmt.Println(x)
    }
```

Un nouveau langage, Go. Pour quoi faire ?

- **Go : Panic**
  - Pas d'exceptions
  - Des erreurs / et un « panic »
  - Changement de mindset

```
// A toy implementation of cube root using Newton's method.
func CubeRoot(x float64) float64 {
    z := x/3   // Arbitrary initial value
    for i := 0; i < 1e6; i++ {
        prevz := z
        z -= (z*z*z-x) / (3*z*z)
        if veryClose(z, prevz) {
            return z
        }
    }
    // A million iterations has not converged; something is wrong.
    panic(fmt.Sprintf("CubeRoot(%g) did not converge", x))
}
```

Un nouveau langage, Go. Pour quoi faire ?

- **Go : Fonctions anonymes**

```
package main

import "fmt"

func adder() func(int) int {
	sum := 0
	return func(x int) int {
		sum += x
		return sum
	}
}

func main() {
	pos, neg := adder(), adder()
	for i := 0; i < 10; i++ {
		fmt.Println(
			pos(i),
			neg(-2*i),
		)
	}
}
```

Go Tour

Un nouveau langage, Go. Pour quoi faire ?

- **Go : Goroutine**
  - Et des canaux / Channels
  - Concepts pour gérer le parallélisme / la concurrence (CSP)

```
package main

import "fmt"

func sum(s []int, c chan int) {
	sum := 0
	for _, v := range s {
		sum += v
	}
	c <- sum // send sum to c
}

func main() {
	s := []int{7, 2, 8, -9, 4, 0}

	c := make(chan int)
	go sum(s[:len(s)/2], c)
	go sum(s[len(s)/2:], c)
	x, y := <-c, <-c // receive from c

	fmt.Println(x, y, x+y)
}
```

Go Tour

Un nouveau langage, Go. Pour quoi faire ?

- **Optimisation de la génération de binaire**
  - Rendre la compilation, l'édition de liens « rapides »
- **En assurant que les dépendances ne sont compilées qu'une fois**
  - Si A importe B, B importe C, alors
  - Compiler B intègre C
  - Compiler A intègre B (qui intègre déjà C)...

Un nouveau langage, Go. Pour quoi faire ?

- **Changer l'état d'esprit**
  - Retirer les exceptions,
  - Insister sur l'usage d'erreurs en retour multiple de fonctions
    - Transcription directe d'une approche C commune (-1 + errno)
    - Sucre syntaxique très efficace
  - Proposer un panic
    - Cas type : redémarrage d'un processus sur un gros serveur
  - Retirer l'héritage
    - Mal compris, souvent mal utilisé…

Un nouveau langage, Go. Pour quoi faire ?

- **Support industriel**
  - Google
  - Effort sérieux sur les outils
    - Qualité, présence de compilateurs
    - Attention aux performances de la compilation
    - Outils périphériques
    - Déploiement simplifié (compilation statique)
- **Qui amène ses questions**
  - La présence de Google est-elle un frein ?

Pour Conclure

- **Go propose un retour aux fondements**
  - Simplicité, maîtrise
- **En tenant compte du meilleur du domaine**
  - Progrès dans le génie logiciel, les langages de programmation
- **Et de l'expérience**
  - Usages négatifs de fonctionnalités (ex : héritage, absent de Go)
- **Et des nouveaux usages**
  - Machines parallèles, serveurs, multi/many coeurs

Et une requête

- **Les créateurs de langages de programmation**
  - Répondent à un besoin
  - Même si ce dernier n'est pas « cool » d'un point de vue recherche
- **La recherche en langages de programmation**
  - Doit répondre à ces besoins
  - Même si l'expression du besoin n'est pas « trendy »
    - Rejet des langages dynamiques
  - Les chercheurs ont les outils dont ont besoin les créateurs de langages
    - Pour faciliter l'implémentation
- **Go est un exemple de réussite de ce point de vue**
  - Bonne utilisation des outils / techniques
  - Avec une forte opinion !

Commissariat à l'énergie atomique et aux énergies alternatives
Institut List | CEA SACLAY NANO-INNOV | BAT. 861 – PC142
91191 Gif-sur-Yvette Cedex - FRANCE
www-list.cea.fr

Établissement public à caractère industriel et commercial | RCS Paris B 775 685 019