

# Table des matières

---

|                                                               |          |
|---------------------------------------------------------------|----------|
| À propos de l'auteur .....                                    | xxiv     |
| Avant-propos .....                                            | xxv      |
| 1. Pourquoi cet ouvrage .....                                 | xxv      |
| 2. À qui s'adresse ce livre .....                             | xxv      |
| 3. Structure de l'ouvrage .....                               | xxvi     |
| 4. Note sur la terminologie .....                             | xxvii    |
| 5. Approche méthodologique .....                              | xxvii    |
| 6. Prérequis .....                                            | xxvii    |
| 7. Versions et compatibilité .....                            | xxviii   |
| <b>Introduction .....</b>                                     | <b>1</b> |
| 1. Genèse d'un langage pragmatique .....                      | 1        |
| 2. Philosophie : faire simple, faire bien .....               | 2        |
| 3. Différences fondamentales avec d'autres langages .....     | 2        |
| Comparaison avec C .....                                      | 2        |
| Comparaison avec C++ .....                                    | 3        |
| Comparaison avec Java .....                                   | 3        |
| Comparaison avec Rust .....                                   | 3        |
| <b>Fondements du runtime .....</b>                            | <b>5</b> |
| <b>1. Architecture et modèle de concurrence .....</b>         | <b>6</b> |
| 1.1. Un paradigme unique dans l'écosystème des langages ..... | 6        |
| Compilation native et runtime embarqué .....                  | 7        |
| L'innovation de Go : réconcilier les opposés .....            | 8        |
| 1.2. Le modèle G-M-P (Goroutine-Machine-Processor) .....      | 10       |
| Les trois composants du modèle .....                          | 10       |
| Relations et interactions entre G, M et P .....               | 12       |
| 1.3. GOMAXPROCS et gestion des processeurs .....              | 13       |
| Valeur par défaut et détection automatique .....              | 13       |
| Configuration par variable d'environnement .....              | 14       |
| 1.4. Configuration dynamique de GOMAXPROCS .....              | 14       |
| Mécanismes de modification à l'exécution .....                | 14       |
| Reconfiguration à chaud via signaux système .....             | 15       |
| Configuration dynamique avec phases distinctes .....          | 16       |
| 1.5. GOMAXPROCS : performances et optimisation .....          | 17       |
| Deux dimensions de limitation CPU .....                       | 19       |

|                                                                                |           |
|--------------------------------------------------------------------------------|-----------|
| Détection et solutions pratiques .....                                         | 20        |
| <b>2. Démarrage du runtime et initialisation .....</b>                         | <b>22</b> |
| 2.1. Signification et convention de nommage <i>rt0</i> .....                   | 22        |
| 2.2. Configuration de la stack et structures fondamentales .....               | 24        |
| 2.3. Détection et configuration du processeur .....                            | 26        |
| 2.4. Gestion CGO et configuration TLS .....                                    | 28        |
| 2.5. Liaison des structures M et G .....                                       | 29        |
| 2.6. Démarrage du scheduler et exécution de <i>main</i> .....                  | 30        |
| 2.7. Initialisation des packages .....                                         | 31        |
| 2.8. Traitement des fonctions <i>init()</i> .....                              | 34        |
| 2.9. Génération des symboles d'initialisation par le compilateur .....         | 35        |
| 2.10. Exécution finale et lancement de <i>main</i> .....                       | 37        |
| 2.11. Récapitulatif des étapes de bootstrap .....                              | 37        |
| 2.12. Débogage du démarrage .....                                              | 38        |
| Syntaxe et utilisation .....                                                   | 38        |
| Traçage de l'initialisation avec <i>inittrace</i> .....                        | 39        |
| <b>3. Modèle mémoire de Go .....</b>                                           | <b>42</b> |
| 3.1. Concepts fondamentaux et règles de base .....                             | 42        |
| Un piège classique en programmation .....                                      | 43        |
| Race conditions : quand l'ordre devient imprévisible .....                     | 44        |
| Happens-before : les règles du jeu .....                                       | 44        |
| Comment le runtime implémente ces garanties .....                              | 45        |
| Sequential consistency par goroutine .....                                     | 47        |
| 3.2. Relations happens-before détaillées .....                                 | 48        |
| Happens-before et initialisation .....                                         | 48        |
| Happens-Before et création/démarrage de goroutines .....                       | 49        |
| Happens-before avec les opérations atomiques .....                             | 50        |
| Happens-before, <i>context</i> et annulation .....                             | 50        |
| Happens-before, finalizers et <i>KeepAlive</i> .....                           | 51        |
| Happens-before pour les mutex et channels .....                                | 51        |
| Tableau de référence des relations happens-before .....                        | 51        |
| <b>4. Allocation et gestion de la mémoire .....</b>                            | <b>54</b> |
| 4.1. Allocateur mémoire de Go (spans, arenas, heaps) .....                     | 54        |
| Stratégie d'allocation système .....                                           | 57        |
| Classes de taille et fragmentation interne .....                               | 57        |
| Sélection et dimensionnement des spans .....                                   | 58        |
| Hiérarchie d'allocation : <i>mcache</i> , <i>mcentral</i> , <i>mheap</i> ..... | 59        |
| Chemins d'allocation : rapide vs lent .....                                    | 61        |

|                                                              |           |
|--------------------------------------------------------------|-----------|
| Types d'objets et stratégies spécialisées .....              | 62        |
| Métadonnées de types et localisation des pointeurs .....     | 63        |
| Intégration avec le garbage collector .....                  | 65        |
| Heap système unifié .....                                    | 65        |
| Analyse du gaspillage mémoire .....                          | 66        |
| 4.2. Architecture et gestion de la stack .....               | 67        |
| Concept et rôle de la stack .....                            | 67        |
| Stack système face à l'implémentation Go .....               | 68        |
| Anatomie de la stack .....                                   | 69        |
| Analyse du comportement de la stack .....                    | 70        |
| Croissance de la stack .....                                 | 71        |
| Réduction de la stack .....                                  | 72        |
| 4.3. Escape analysis en détail .....                         | 73        |
| Règles d'échappement .....                                   | 74        |
| Analyser l'escape analysis .....                             | 74        |
| Impact sur les performances .....                            | 76        |
| Stratégies d'optimisation .....                              | 77        |
| <b>5. Garbage collector et récupération mémoire .....</b>    | <b>82</b> |
| 5.1. Algorithme à trois couleurs ( <i>tricolor</i> ) .....   | 82        |
| Principe général .....                                       | 82        |
| Fonctionnement de l'algorithme .....                         | 82        |
| Exécution concurrente .....                                  | 84        |
| Impact de GOMAXPROCS sur le garbage collector .....          | 84        |
| Concurrence et invariant à trois couleurs .....              | 86        |
| Avantages de l'algorithme à trois couleurs .....             | 88        |
| Limitations .....                                            | 88        |
| Implémentation dans Go .....                                 | 88        |
| Green Tea : marquage par page (Go 1.26+) .....               | 90        |
| Bénéfices de l'architecture du garbage collector .....       | 92        |
| Conclusion sur le garbage collector .....                    | 92        |
| 5.2. Le scavenger : restitution de mémoire au système .....  | 93        |
| Problématique : mémoire virtuelle vs mémoire résidente ..... | 93        |
| Fonctionnement du scavenger .....                            | 94        |
| Heuristiques de déclenchement .....                          | 94        |
| 5.3. Tuning du GC (GOGC, GOMEMLIMIT) .....                   | 94        |
| Variables d'environnement pour le contrôle GC .....          | 95        |
| Mécanisme GOGC en détail .....                               | 95        |
| Risques de la modification dynamique du GC .....             | 95        |
| GOMEMLIMIT : contrôle par limite mémoire .....               | 97        |

|                                                          |            |
|----------------------------------------------------------|------------|
| Stratégies de tuning par cas d'usage .....               | 98         |
| Contrôle de la taille des stacks .....                   | 99         |
| Restitution forcée de mémoire .....                      | 100        |
| <b>6. Observabilité et optimisation mémoire .....</b>    | <b>102</b> |
| 6.1. Métriques de monitoring .....                       | 102        |
| Cycles du garbage collector .....                        | 102        |
| Mémoire heap .....                                       | 103        |
| Allocations globales .....                               | 104        |
| Structures internes .....                                | 104        |
| Interprétation et corrélations des métriques .....       | 105        |
| 6.2. Débogage du garbage collector .....                 | 106        |
| Variables de debug disponibles .....                     | 106        |
| Analyse du garbage collector .....                       | 106        |
| Contrôle de la gestion mémoire système .....             | 107        |
| 6.3. Profiling de la mémoire avancé .....                | 108        |
| Configuration de l'échantillonnage .....                 | 108        |
| Activation et collecte .....                             | 109        |
| Analyse avec go tool pprof .....                         | 111        |
| <b>7. Architecture et ordonnancement .....</b>           | <b>113</b> |
| 7.1. Anatomie du scheduler .....                         | 113        |
| Qu'est-ce qu'un scheduler ? .....                        | 113        |
| Architecture des files .....                             | 114        |
| Mécanismes de parking et unparking .....                 | 115        |
| États des goroutines .....                               | 115        |
| Intégration avec le modèle G-M-P .....                   | 116        |
| 7.2. Scheduling coopératif vs préemptif .....            | 116        |
| Scheduling coopératif : les premiers temps .....         | 117        |
| L'émergence de la préemption .....                       | 117        |
| Limitations par plateforme .....                         | 118        |
| Comparaison des approches .....                          | 118        |
| Impact de la préemption sur l'équité du scheduling ..... | 121        |
| Comportement hybride moderne .....                       | 122        |
| Implications pour les développeurs .....                 | 122        |
| 7.3. Appels système et goroutines bloquantes .....       | 123        |
| Typologie des appels système .....                       | 123        |
| Traitement des appels bloquants .....                    | 124        |
| Chemin optimiste : appels système rapides .....          | 124        |
| Mécanisme de hand-off .....                              | 125        |

|                                                     |            |
|-----------------------------------------------------|------------|
| Compromis et coût du hand-off .....                 | 126        |
| Gestion des threads système .....                   | 127        |
| Contrôle défensif du nombre de threads .....        | 127        |
| Impact sur les performances .....                   | 128        |
| 7.4. Mécanismes de synchronisation bas niveau ..... | 129        |
| Futex : Fast Userspace Mutexes .....                | 129        |
| Équivalents sur autres systèmes .....               | 129        |
| Base commune : les futex comme fondation .....      | 130        |
| Interaction avec le scheduler .....                 | 130        |
| Gestion des timeouts .....                          | 130        |
| Détection de deadlocks .....                        | 131        |
| <b>8. Composants du runtime et monitoring .....</b> | <b>132</b> |
| 8.1. Timers et ordonnancement temporel .....        | 132        |
| Architecture des timers .....                       | 134        |
| Mécanismes d'expiration .....                       | 134        |
| Optimisations et performances .....                 | 135        |
| 8.2. Network poller (epoll/kqueue) .....            | 135        |
| Architecture du network poller .....                | 135        |
| Intégration avec le scheduler .....                 | 136        |
| Polling et intégration système .....                | 137        |
| Optimisations et tuning .....                       | 138        |
| 8.3. Sysmon et mécanismes d'interruption .....      | 139        |
| Architecture de sysmon .....                        | 139        |
| Surveillance des goroutines longues .....           | 139        |
| Déclenchement du garbage collector .....            | 140        |
| Intervention réseau de secours .....                | 140        |
| Détection des deadlocks .....                       | 140        |
| Impact sur les performances .....                   | 141        |
| 8.4. Gestion des signaux système .....              | 141        |
| Goroutine isolée et verrouillage .....              | 141        |
| Gestionnaire de signaux universel .....             | 141        |
| Canaux de contrôle et coordination .....            | 142        |
| Protocole non-bloquant .....                        | 142        |
| Goroutine de distribution .....                     | 143        |
| 8.5. Work stealing et load balancing .....          | 144        |
| Principe du work stealing .....                     | 144        |
| Algorithme de recherche de travail .....            | 144        |
| Équilibrage de charge par priorité .....            | 145        |
| Optimisations du vol .....                          | 145        |

|                                                       |            |
|-------------------------------------------------------|------------|
| Impact sur les performances .....                     | 145        |
| 8.6. Mécanismes d'ordonnancement des goroutines ..... | 146        |
| Algorithme de sélection .....                         | 146        |
| Gestion des goroutines nouvellement prêtes .....      | 147        |
| Changement de contexte et transition .....            | 147        |
| Cas particulier des goroutines liées .....            | 148        |
| Optimisations de l'ordonnancement .....               | 148        |
| 8.7. Avantages du scheduler Go .....                  | 149        |
| Synthèse des paradigmes .....                         | 149        |
| Avantages du modèle hybride .....                     | 150        |
| Compromis et limitations .....                        | 151        |
| Positionnement pragmatique .....                      | 151        |
| 8.8. Débogage du scheduler .....                      | 152        |
| Variables de debug disponibles .....                  | 152        |
| Monitoring du scheduler .....                         | 152        |
| Débogage réseau et DNS .....                          | 154        |
| Extension de schedtrace avec scheddetail .....        | 156        |
| <i>dontfreezetheworld</i> .....                       | 157        |
| <b>Types et structures de données .....</b>           | <b>158</b> |
| <b>9. Fondements des structures de données .....</b>  | <b>159</b> |
| 9.1. Hiérarchie mémoire et localité .....             | 159        |
| Principe de localité .....                            | 159        |
| Lignes de cache et performance .....                  | 161        |
| Coût des défauts de cache .....                       | 162        |
| Représentation mémoire des types primitifs .....      | 162        |
| Alignement mémoire et padding .....                   | 163        |
| Règles d'alignement et insertion de padding .....     | 165        |
| Stratégies d'optimisation .....                       | 169        |
| 9.2. Sémantique et performance des pointeurs .....    | 171        |
| Intégrité mémoire et sécurité .....                   | 172        |
| Stratégies d'utilisation des pointeurs .....          | 173        |
| 9.3. Compromis entre mémoire et vitesse .....         | 174        |
| Principe du compromis .....                           | 174        |
| Exemple concret : cache de hash de fichiers .....     | 175        |
| Critères de décision .....                            | 176        |
| Métriques de mesure .....                             | 177        |
| 9.4. <i>Zero values</i> et leur utilisation .....     | 177        |

|                                                                 |            |
|-----------------------------------------------------------------|------------|
| Définition et garanties .....                                   | 178        |
| Conception pour l'utilisabilité immédiate .....                 | 179        |
| Pièges courants .....                                           | 179        |
| <b>10. Slices et arrays .....</b>                               | <b>181</b> |
| 10.1. Représentation mémoire et slice header .....              | 181        |
| Tableaux système : la réalité matérielle .....                  | 181        |
| Arrays Go : sécurité et performance .....                       | 182        |
| Slices : l'abstraction dynamique .....                          | 184        |
| Comportements particuliers du compilateur .....                 | 184        |
| 10.2. Backing arrays et partage mémoire .....                   | 186        |
| Architecture du backing array .....                             | 186        |
| Syntaxe de slicing étendue .....                                | 188        |
| Stratégies d'allocation .....                                   | 188        |
| Cycle de vie et fuites de mémoire .....                         | 189        |
| Détection de chevauchement et protections .....                 | 189        |
| Copie de slice headers et partage du backing array .....        | 190        |
| Indépendance lors des réallocations .....                       | 190        |
| Éviter l'implicite .....                                        | 191        |
| 10.3. Impact sur les performances .....                         | 191        |
| Mécanismes d'expansion .....                                    | 191        |
| Optimisations et performance .....                              | 192        |
| 10.4. Techniques sans copie .....                               | 193        |
| Slicing et isolation .....                                      | 193        |
| Conversion entre types compatibles .....                        | 194        |
| Réutilisation de backing arrays .....                           | 194        |
| 10.5. Bonnes pratiques de développement .....                   | 194        |
| Préalouer avec une capacité généreuse .....                     | 195        |
| Limiter l'usage des sous-slices .....                           | 195        |
| Copier explicitement les données critiques .....                | 196        |
| Documenter les mutations partagées .....                        | 196        |
| Préférer les capacités explicites pour un contrôle précis ..... | 196        |
| Éviter le reslicing expansif .....                              | 196        |
| Éviter les pointeurs vers des éléments de slices .....          | 196        |
| Manipulations bas niveau dangereuses .....                      | 197        |
| <b>11. Strings .....</b>                                        | <b>198</b> |
| 11.1. Strings : représentation mémoire et propriétés .....      | 198        |
| Représentations des chaînes : panorama des approches .....      | 198        |
| Spécificités de Go face aux autres approches .....              | 199        |

|                                                                     |            |
|---------------------------------------------------------------------|------------|
| Vérifications de bornes et sécurité mémoire .....                   | 200        |
| 11.2. Méthodes d'encodage des caractères .....                      | 200        |
| 11.3. Encodage et représentation des caractères en Go .....         | 203        |
| 11.4. Manipulation des strings UTF-8 en Go .....                    | 204        |
| 11.5. Structure interne et représentation mémoire .....             | 206        |
| 11.6. Immutabilité des strings Go et implications .....             | 208        |
| Fondements théoriques de l'immutabilité .....                       | 208        |
| Partage du backing array .....                                      | 209        |
| Optimisations du compilateur .....                                  | 210        |
| 11.7. Contourner l'immutabilité avec le package <i>unsafe</i> ..... | 211        |
| Motivation et cas d'usage .....                                     | 211        |
| Représentation mémoire et accès direct .....                        | 211        |
| Risques et limitations critiques .....                              | 212        |
| Alternatives recommandées .....                                     | 213        |
| Conclusion et recommandations .....                                 | 213        |
| <b>12. Maps – tables de hachage internes .....</b>                  | <b>215</b> |
| 12.1. Architecture générale des maps en Go .....                    | 215        |
| Sémantique de valeur et représentation compilateur .....            | 215        |
| Structure interne hmap .....                                        | 216        |
| Buckets et organisation des données .....                           | 217        |
| 12.2. Algorithmes de hachage intégrés .....                         | 219        |
| Fonctions de hachage spécialisées .....                             | 219        |
| Optimisation AES pour les chaînes .....                             | 219        |
| 12.3. Gestion des collisions et facteur de charge .....             | 220        |
| Résolution des collisions .....                                     | 220        |
| Facteur de charge et déclencheurs de croissance .....               | 220        |
| 12.4. Algorithmes de croissance et rétrécissement .....             | 221        |
| Mécanisme de croissance .....                                       | 221        |
| Optimisation pour de petites maps .....                             | 222        |
| Absence de rétrécissement .....                                     | 223        |
| 12.5. Ordre d'énumération et non-déterminisme .....                 | 224        |
| 12.6. Accès concurrent et race conditions .....                     | 225        |
| Problématique des accès non synchronisés .....                      | 225        |
| Solutions de synchronisation .....                                  | 225        |
| 12.7. Bonnes pratiques de développement .....                       | 226        |
| Initialiser avec une capacité .....                                 | 226        |
| Minimiser les accès .....                                           | 226        |
| Utiliser des pointeurs .....                                        | 227        |
| Choisir la bonne taille de clés .....                               | 227        |

|                                                                    |            |
|--------------------------------------------------------------------|------------|
| <b>13. Interfaces et système de types .....</b>                    | <b>229</b> |
| 13.1. Concept fondamental des interfaces .....                     | 229        |
| L'interface vide et son détournement .....                         | 230        |
| Les interfaces à l'exécution .....                                 | 230        |
| 13.2. Représentation mémoire des interfaces .....                  | 231        |
| Structure de données des interfaces .....                          | 231        |
| Tables de méthodes et résolution d'appels .....                    | 231        |
| Table globale et cache des conversions .....                       | 232        |
| Optimisations de la résolution d'appels .....                      | 234        |
| 13.3. Assertions de type et performance .....                      | 234        |
| Deux formes d'assertion .....                                      | 234        |
| Détection de la forme par le compilateur .....                     | 234        |
| Coûts selon le type cible .....                                    | 235        |
| Recommandations d'usage .....                                      | 235        |
| Interface nil et pièges subtils .....                              | 235        |
| Solutions et bonnes pratiques .....                                | 237        |
| Ordre de vérification dans les switch de type .....                | 237        |
| 13.4. Interface vide et overhead .....                             | 238        |
| Mécanisme d'allocation .....                                       | 238        |
| Impact des interfaces sur l'escape analysis .....                  | 239        |
| Stratégies d'optimisation .....                                    | 239        |
| 13.5. Composition d'interfaces .....                               | 240        |
| Incorporation d'interfaces ( <i>embedding</i> ) .....              | 240        |
| Détection de capacités .....                                       | 241        |
| 13.6. Architectures modulaires .....                               | 243        |
| Registry pattern .....                                             | 243        |
| Factory pattern avec interfaces .....                              | 244        |
| Middleware pattern .....                                           | 245        |
| Plugin dynamique avec build constraints .....                      | 246        |
| Pattern de configuration par interface .....                       | 246        |
| 13.7. Optimisations avancées .....                                 | 247        |
| Piège des pointeurs vers interfaces .....                          | 247        |
| Réduction de l'overhead des interfaces .....                       | 248        |
| Profiling des interfaces .....                                     | 248        |
| <b>14. Types paramétrés : polymorphisme à la compilation .....</b> | <b>249</b> |
| 14.1. Syntaxe et concepts fondamentaux .....                       | 249        |
| 14.2. Contraintes de type .....                                    | 250        |
| Les contraintes sont des interfaces .....                          | 250        |

|                                                                    |     |
|--------------------------------------------------------------------|-----|
| Contraintes prédéfinies .....                                      | 251 |
| L'opérateur tilde (~) .....                                        | 251 |
| Combinaison de méthodes et de types .....                          | 251 |
| Contraintes auto-référentes (Go 1.26+) .....                       | 252 |
| 14.3. Implémentation : GCShape stenciling avec dictionnaires ..... | 252 |
| L'approche Go : GCShape stenciling .....                           | 253 |
| Les dictionnaires .....                                            | 253 |
| Impact sur les performances .....                                  | 254 |
| 14.4. Observer la compilation des generics .....                   | 254 |
| Mise en place de l'exemple .....                                   | 254 |
| Visualiser les instanciations .....                                | 255 |
| Examiner le code généré .....                                      | 255 |
| Taille des fonctions générées .....                                | 257 |
| Instanciation cross-package .....                                  | 257 |
| Quand le dictionnaire intervient .....                             | 258 |
| 14.5. Limitations actuelles .....                                  | 259 |
| Pas de méthodes paramétrées .....                                  | 259 |
| Pas de spécialisation .....                                        | 259 |
| Corps requis à la compilation .....                                | 259 |
| 14.6. Conclusion .....                                             | 259 |

## Mécanismes avancés du langage ..... 261

|                                                          |     |
|----------------------------------------------------------|-----|
| 15. Switch et sélection de cas .....                     | 262 |
| 15.1. Les formes du switch en Go .....                   | 262 |
| 15.2. Compilation et représentation mémoire .....        | 263 |
| Switch sur types scalaires .....                         | 263 |
| Switch sur chaînes de caractères .....                   | 267 |
| Switch sur types composites comparables .....            | 270 |
| Switch sans expression : évaluation séquentielle .....   | 271 |
| 15.3. Switch de type et interrogation d'interfaces ..... | 271 |
| Compilation des switch de type .....                     | 272 |
| 15.4. Patterns optimisables vs non-optimisables .....    | 276 |
| Expressions dynamiques dans les case .....               | 276 |
| Switch sans expression .....                             | 277 |
| Ordre imposé avec dépendances .....                      | 277 |
| 15.5. Recommandations pratiques .....                    | 278 |
| Quand privilégier le switch vs if/else ? .....           | 278 |
| Organisation des cas pour la performance .....           | 278 |

|                                                                            |            |
|----------------------------------------------------------------------------|------------|
| Compromis lisibilité/performance .....                                     | 279        |
| <b>16. Fonctions anonymes et closures .....</b>                            | <b>280</b> |
| 16.1. Mécanisme de capture des variables .....                             | 280        |
| Fonctions anonymes : des fonctions à part entière .....                    | 280        |
| Isolement des stack frames .....                                           | 281        |
| Structure d'une closure .....                                              | 282        |
| Capture par valeur .....                                                   | 283        |
| Capture par référence .....                                                | 284        |
| Comprendre la capture .....                                                | 285        |
| 16.2. Impact sur les performances .....                                    | 286        |
| 16.3. Bonnes pratiques .....                                               | 286        |
| <b>17. Defer : garanties d'exécution .....</b>                             | <b>288</b> |
| 17.1. Sémantique et ordre d'exécution .....                                | 289        |
| 17.2. Defer et stack frame .....                                           | 290        |
| 17.3. Évaluation des arguments .....                                       | 290        |
| 17.4. Defer dans une boucle .....                                          | 291        |
| 17.5. Gestion des erreurs avec defer .....                                 | 293        |
| 17.6. Compromis entre simplicité et lisibilité .....                       | 294        |
| 17.7. Représentation mémoire .....                                         | 295        |
| <b>18. Panic et recover : gestion des conditions exceptionnelles .....</b> | <b>296</b> |
| 18.1. Pourquoi Go n'a pas d'exceptions .....                               | 296        |
| L'ambiguïté du terme <i>exception</i> .....                                | 296        |
| Flux de contrôle caché .....                                               | 297        |
| Difficulté de raisonnement .....                                           | 298        |
| L'approche de Go .....                                                     | 298        |
| Avantages des erreurs explicites .....                                     | 299        |
| Inconvénients des erreurs explicites .....                                 | 299        |
| 18.2. Panic : conditions d'arrêt exceptionnelles .....                     | 300        |
| Déclenchement explicite .....                                              | 300        |
| Quand utiliser panic ? .....                                               | 300        |
| Panics générés par le runtime .....                                        | 301        |
| Propagation du panic .....                                                 | 302        |
| Panics du runtime et sûreté mémoire .....                                  | 303        |
| 18.3. Recover : interception et reprise .....                              | 308        |
| Sémantique de recover .....                                                | 308        |
| Compromis et risques de recover .....                                      | 308        |
| Panic et flux de contrôle .....                                            | 309        |
| Re-panic et chaînage de panics .....                                       | 310        |

|                                                                        |            |
|------------------------------------------------------------------------|------------|
| Le cas particulier de <i>panic(nil)</i> .....                          | 311        |
| 18.4. Implémentation technique .....                                   | 312        |
| Structure <i>_panic</i> .....                                          | 312        |
| Liaison avec les <i>defer</i> .....                                    | 313        |
| Dépilage de la pile .....                                              | 313        |
| Interaction avec le scheduler .....                                    | 313        |
| 18.5. Conclusion .....                                                 | 314        |
| <b>19. Itérateurs par fonction .....</b>                               | <b>315</b> |
| 19.1. Syntaxe et sémantique de base .....                              | 316        |
| 19.2. Mécanismes internes .....                                        | 317        |
| Découpage du code en deux fonctions .....                              | 317        |
| Structure d'état et protocole .....                                    | 319        |
| Détection des violations .....                                         | 319        |
| 19.3. Interaction avec <i>defer</i> et <i>panic</i> .....              | 320        |
| Comportement des <i>defer</i> .....                                    | 320        |
| Gestion dans le runtime .....                                          | 321        |
| 19.4. Conclusion .....                                                 | 322        |
| <b>20. Unsafe : contourner la sécurité des types .....</b>             | <b>323</b> |
| 20.1. Pourquoi ce package existe .....                                 | 323        |
| 20.2. Problèmes de portabilité .....                                   | 324        |
| 20.3. Les primitives fondamentales .....                               | 326        |
| Type <i>unsafe.Pointer</i> .....                                       | 326        |
| Fonctions d'introspection .....                                        | 327        |
| 20.4. Les six patterns valides .....                                   | 329        |
| Pattern 1 : Conversion entre types de pointeurs .....                  | 329        |
| Pattern 2 : Conversion vers <i>uintptr</i> (sans retour) .....         | 330        |
| Pattern 3 : Arithmétique de pointeur .....                             | 330        |
| Pattern 4 : Conversion pour <i>syscall.Syscall</i> .....               | 330        |
| Pattern 5 : Conversion depuis <i>reflect.Value</i> .....               | 330        |
| Pattern 6 : <i>SliceHeader</i> et <i>StringHeader</i> (déprécié) ..... | 330        |
| 20.5. Autres cas d'usage légitimes .....                               | 331        |
| Conversions sans copie .....                                           | 331        |
| Interfaçage avec C via CGO .....                                       | 331        |
| Optimisations mesurées .....                                           | 331        |
| 20.6. Vérification et outils de débogage .....                         | 332        |
| go vet .....                                                           | 332        |
| Flag <i>-d=checkptr</i> .....                                          | 332        |
| 20.7. Alternatives et bonnes pratiques .....                           | 333        |

|                                                                      |            |
|----------------------------------------------------------------------|------------|
| <b>21. Réflexion : introspection et manipulation dynamique .....</b> | <b>334</b> |
| 21.1. Concepts et cas d'usage .....                                  | 334        |
| Qu'est-ce que la réflexion ? .....                                   | 334        |
| À quoi sert la réflexion ? .....                                     | 334        |
| Quand utiliser la réflexion ? .....                                  | 335        |
| Quand éviter la réflexion ? .....                                    | 335        |
| 21.2. Types et valeurs en réflexion .....                            | 336        |
| <i>reflect.Type</i> .....                                            | 336        |
| Type descriptors et metadata .....                                   | 337        |
| Obtention des types au runtime .....                                 | 338        |
| <i>reflect.Value</i> .....                                           | 338        |
| Modification de valeurs .....                                        | 338        |
| 21.3. Construction dynamique de types .....                          | 340        |
| 21.4. Struct tags : métadonnées déclaratives .....                   | 341        |
| Nature et fonctionnement .....                                       | 341        |
| Lecture des tags avec réflexion .....                                | 342        |
| Patterns d'usage .....                                               | 342        |
| 21.5. Interactions avec <i>unsafe</i> .....                          | 343        |
| 21.6. Alternatives à la réflexion .....                              | 344        |
| Interfaces .....                                                     | 344        |
| Generics .....                                                       | 344        |
| Type switches .....                                                  | 344        |
| Génération de code .....                                             | 345        |
| 21.7. Coût et performance de la réflexion .....                      | 345        |
| 21.8. Conclusion .....                                               | 347        |
| <b>22. Compilation et optimisations .....</b>                        | <b>348</b> |
| 22.1. Le processus de compilation .....                              | 348        |
| De la source à l'AST .....                                           | 348        |
| Vérification des types .....                                         | 350        |
| De l'AST à la SSA .....                                              | 350        |
| Optimisations et génération du code .....                            | 351        |
| 22.2. Optimisations génériques .....                                 | 351        |
| Remplacement d'opérations coûteuses .....                            | 351        |
| Réorganisation d'expressions .....                                   | 352        |
| 22.3. Inlining .....                                                 | 353        |
| Mécanisme et heuristiques .....                                      | 353        |
| Contrôle de l'inlining .....                                         | 354        |
| Impact sur les performances .....                                    | 354        |

|                                                         |     |
|---------------------------------------------------------|-----|
| 22.4. Fonctions intrinsèques ( <i>intrinsic</i> ) ..... | 355 |
| Fonctions concernées .....                              | 355 |
| Code spécifique à l'architecture .....                  | 356 |
| 22.5. Élimination des vérifications de sécurité .....   | 358 |
| Principe général .....                                  | 358 |
| Élimination des vérifications de bornes .....           | 358 |
| Élimination des vérifications de pointeur nil .....     | 360 |
| 22.6. Constant folding et propagation .....             | 361 |
| 22.7. Dévirtualisation .....                            | 361 |
| 22.8. Escape analysis .....                             | 362 |
| 22.9. Élimination des sous-expressions communes .....   | 362 |
| Principe de base .....                                  | 362 |
| Conditions d'application .....                          | 363 |
| CSE locale et globale .....                             | 363 |
| Rôle de la forme SSA .....                              | 364 |
| Interaction avec d'autres optimisations .....           | 364 |
| 22.10. Élimination de code mort .....                   | 365 |
| Interaction avec les benchmarks .....                   | 366 |
| 22.11. GOSSAFUNC : observer les optimisations .....     | 367 |
| Utilisation .....                                       | 367 |
| Cas d'usage .....                                       | 367 |
| 22.12. Optimisation guidée par profil (PGO) .....       | 368 |
| Principe de fonctionnement .....                        | 368 |
| Collecte du profil .....                                | 368 |
| Compilation avec PGO .....                              | 371 |
| Optimisations activées par PGO .....                    | 372 |
| Stabilité du profil .....                               | 373 |
| Mesure des gains .....                                  | 374 |
| Bonnes pratiques .....                                  | 374 |
| 22.13. <code>go:generate</code> .....                   | 375 |
| Principes de conception .....                           | 375 |
| Syntaxe et utilisation .....                            | 375 |
| Cas d'usage typiques .....                              | 376 |
| Alternatives .....                                      | 377 |
| 22.14. Système de plugins .....                         | 377 |
| Fonctionnement de base .....                            | 378 |
| Limitations techniques .....                            | 379 |
| Alternatives recommandées .....                         | 379 |

**Concurrence maîtrisée ..... 380**

|                                                                            |            |
|----------------------------------------------------------------------------|------------|
| <b>23. Goroutines .....</b>                                                | <b>381</b> |
| 23.1. Concurrence et parallélisme : une distinction fondamentale .....     | 381        |
| 23.2. Multiplexage et non-déterminisme .....                               | 382        |
| 23.3. Cycle de vie et terminaison des goroutines .....                     | 383        |
| Panics et isolation des goroutines .....                                   | 383        |
| 23.4. Fuites de goroutines : détection et prévention .....                 | 387        |
| 23.5. Profiling des goroutines .....                                       | 389        |
| 23.6. Limitation du runtime : impossibilité de <code>fork()</code> .....   | 390        |
| Origine de la limitation .....                                             | 390        |
| Alternative : <code>ForkExec</code> .....                                  | 391        |
| Pattern pour les démons .....                                              | 392        |
| <b>24. Primitives de synchronisation .....</b>                             | <b>393</b> |
| 24.1. Fondements matériels des opérations atomiques .....                  | 393        |
| Le problème fondamental de la concurrence .....                            | 393        |
| Instructions atomiques matérielles .....                                   | 394        |
| Cohérence de cache et protocoles MESI .....                                | 395        |
| Memory barriers et ordering .....                                          | 396        |
| Coût des opérations atomiques .....                                        | 396        |
| 24.2. Package <code>sync/atomic</code> : interface Go .....                | 396        |
| Types supportés .....                                                      | 397        |
| Opérations de base .....                                                   | 397        |
| <code>atomic.Value</code> : types arbitraires .....                        | 399        |
| 24.3. Garanties <code>happens-before</code> des opérations atomiques ..... | 399        |
| 24.4. Performance et compromis .....                                       | 400        |
| Impact de la contention .....                                              | 400        |
| False sharing et padding .....                                             | 401        |
| 24.5. Programmation lock-free .....                                        | 403        |
| Compare-and-swap en pratique .....                                         | 403        |
| Stack lock-free .....                                                      | 406        |
| File MPSC (Multiple Producer, Single Consumer) .....                       | 408        |
| Considérations de performance .....                                        | 411        |
| 24.6. <code>sync.Mutex</code> : exclusion mutuelle fondamentale .....      | 411        |
| Synchronisation avec l'OS .....                                            | 412        |
| Garanties <code>happens-before</code> .....                                | 413        |
| Architecture interne .....                                                 | 414        |
| Interdiction de copie des mutex .....                                      | 415        |

|                                                            |            |
|------------------------------------------------------------|------------|
| 24.7. Hiérarchie des primitives de synchronisation .....   | 416        |
| 24.8. Conclusion .....                                     | 417        |
| <b>25. Fonctions avancées de synchronisation .....</b>     | <b>419</b> |
| 25.1. Lecture concurrente : <i>sync.RWMutex</i> .....      | 419        |
| Sémantique lecteurs-écrivains .....                        | 419        |
| Garanties happens-before .....                             | 420        |
| Architecture interne .....                                 | 420        |
| Coût des opérations .....                                  | 422        |
| Patterns d'utilisation .....                               | 422        |
| Performance et compromis .....                             | 423        |
| 25.2. Coordination de tâches : <i>sync.WaitGroup</i> ..... | 425        |
| Garanties happens-before .....                             | 425        |
| Pièges classiques .....                                    | 426        |
| 25.3. Initialisation unique : <i>sync.Once</i> .....       | 427        |
| Garanties happens-before .....                             | 427        |
| Architecture interne .....                                 | 428        |
| 25.4. Signalisation de condition : <i>sync.Cond</i> .....  | 430        |
| Garanties happens-before .....                             | 430        |
| Architecture et sémantique .....                           | 432        |
| 25.5. Map concurrente optimisée : <i>sync.Map</i> .....    | 434        |
| Garanties happens-before .....                             | 434        |
| Architecture interne .....                                 | 434        |
| Performance et compromis .....                             | 436        |
| 25.6. Réutilisation d'objets : <i>sync.Pool</i> .....      | 436        |
| 25.7. Impact sur les performances .....                    | 437        |
| <b>26. Channels : communication et signalisation .....</b> | <b>440</b> |
| 26.1. Implémentation interne des channels .....            | 440        |
| Structure de données fondamentale .....                    | 440        |
| Files d'attente des goroutines .....                       | 441        |
| Optimisations de copie mémoire .....                       | 441        |
| 26.2. Implications et performances .....                   | 442        |
| Channels sans buffer : synchronisation stricte .....       | 442        |
| Channels avec buffer : découplage temporel .....           | 442        |
| 26.3. Garanties de synchronisation et happens-before ..... | 445        |
| Channels sans buffer et point de rendez-vous .....         | 445        |
| Channels avec buffer et ordre FIFO .....                   | 446        |
| Fermeture et ordre de livraison .....                      | 446        |
| Implémentation et primitives de bas niveau .....           | 448        |

|                                                                     |            |
|---------------------------------------------------------------------|------------|
| 26.4. Transmission par copie ou par pointeur .....                  | 448        |
| 26.5. Sens des channels et sûreté des types .....                   | 449        |
| 26.6. Fermeture et patterns de signalisation .....                  | 450        |
| Fermeture d'un channel .....                                        | 450        |
| Diffusion par fermeture .....                                       | 451        |
| 26.7. Select statement : algorithme et optimisations .....          | 451        |
| Sémantique de base .....                                            | 451        |
| Algorithme interne du <i>select</i> .....                           | 452        |
| Select non-bloquant .....                                           | 453        |
| Optimisations pour les cas courants .....                           | 454        |
| Performance avec de nombreux cas .....                              | 454        |
| 26.8. Select dynamique avec réflexion .....                         | 454        |
| 26.9. Patterns avancés de communication .....                       | 456        |
| Pipeline pattern .....                                              | 456        |
| Request-Response avec channels .....                                | 457        |
| <b>27. Outils pour la programmation concurrente .....</b>           | <b>458</b> |
| 27.1. Le race detector .....                                        | 458        |
| Fonctionnement interne .....                                        | 458        |
| Utilisation .....                                                   | 461        |
| Data race par capture de variable de boucle .....                   | 461        |
| Limites du race detector .....                                      | 463        |
| 27.2. Profiling de la contention .....                              | 464        |
| Profil de blocage .....                                             | 464        |
| Profil des mutex .....                                              | 465        |
| Interprétation des résultats .....                                  | 465        |
| 27.3. Tester le code concurrent avec <i>testing/syncstest</i> ..... | 467        |
| Le concept de bulle .....                                           | 467        |
| L'horloge simulée .....                                             | 468        |
| Opérations durablement bloquantes .....                             | 469        |
| Vérifier qu'un événement ne se produit pas .....                    | 470        |
| Cycle de vie d'une bulle .....                                      | 471        |
| Isolation des ressources .....                                      | 471        |
| <b>Annexes .....</b>                                                | <b>473</b> |
| <b>Démonstrations techniques et benchmarks .....</b>                | <b>474</b> |
| 1. Validation de l'impact CGO sur GOMAXPROCS .....                  | 474        |
| 2. Mécanismes de synchronisation et performances .....              | 477        |
| Résultats .....                                                     | 478        |

|                                                                        |     |
|------------------------------------------------------------------------|-----|
| Analyse .....                                                          | 478 |
| 3. Démonstration d'une race condition .....                            | 479 |
| 4. Démonstration des races conditions sur un seul thread .....         | 480 |
| 5. Impact de <code>runtime.Gosched()</code> sur les performances ..... | 482 |
| 6. Démonstration de la préemption des goroutines .....                 | 483 |
| 7. Benchmark des performances Map vs Slice .....                       | 485 |
| 8. Test de performance du hachage de chaînes .....                     | 487 |
| 9. Benchmark : sondage linéaire vs quadratique .....                   | 488 |
| 10. Dévirtualisation .....                                             | 492 |
| Mécanisme de base .....                                                | 492 |
| Analyse du code assembleur .....                                       | 493 |
| Conditions pour la dévirtualisation .....                              | 494 |
| Impact sur les performances .....                                      | 494 |
| 11. Localité mémoire et performance des accès .....                    | 494 |
| Analyse des résultats .....                                            | 496 |
| 12. Démonstration de la complexité $O(n^2)$ de la concaténation .....  | 496 |
| Code de test .....                                                     | 496 |
| Analyse de la complexité .....                                         | 498 |
| Alternative efficace .....                                             | 499 |
| 13. Compilation des instructions switch .....                          | 499 |
| Séquence linéaire : trois cas .....                                    | 499 |
| Recherche binaire : quatre cas .....                                   | 501 |
| Switch hiérarchique sur strings : longueurs mixtes .....               | 503 |
| Switch sur types composites : comparaisons séquentielles .....         | 505 |
| Conclusion .....                                                       | 508 |
| 14. Impact de l'ordre des cas sur les performances .....               | 509 |
| Code de test .....                                                     | 509 |
| Résultats .....                                                        | 511 |
| Analyse des résultats et recommandations .....                         | 512 |
| 15. Impact de la capture de variables par les closures .....           | 513 |
| Capture par référence et escape analysis .....                         | 513 |
| Coûts additionnels des closures .....                                  | 515 |
| 16. Impact de la Bounds Check Elimination .....                        | 516 |
| Repérer les vérifications de bornes .....                              | 517 |
| Analyse de l'assembleur .....                                          | 518 |
| Résultats .....                                                        | 519 |
| Analyse .....                                                          | 519 |
| Conclusion .....                                                       | 520 |
| 17. Démonstration PGO .....                                            | 520 |

|                                                               |            |
|---------------------------------------------------------------|------------|
| Code du benchmark .....                                       | 520        |
| Méthodologie .....                                            | 521        |
| Résultats Apple M4 Pro .....                                  | 522        |
| Résultats Intel Xeon 8171M .....                              | 522        |
| Analyse .....                                                 | 523        |
| 18. Benchmark des channels .....                              | 523        |
| Protocole de test .....                                       | 523        |
| Mesures de référence .....                                    | 526        |
| Impact de la contention .....                                 | 527        |
| Influence de la taille du channel .....                       | 527        |
| Influence du nombre de goroutines .....                       | 528        |
| Optimisation par batch .....                                  | 529        |
| Synthèse .....                                                | 530        |
| 19. Benchmark CAS et backoff exponentiel .....                | 531        |
| Protocole de test .....                                       | 531        |
| Résultats sans backoff .....                                  | 533        |
| Résultats avec backoff .....                                  | 534        |
| Analyse .....                                                 | 534        |
| Variante avec yield .....                                     | 535        |
| Équité et écart-type .....                                    | 536        |
| <b>Algorithmes .....</b>                                      | <b>537</b> |
| 1. Représentation des tas binaires par tableaux .....         | 537        |
| Propriétés et objectifs des tas binaires .....                | 537        |
| Principe de la représentation linéaire .....                  | 538        |
| Navigation dans la structure .....                            | 539        |
| Algorithmes fondamentaux .....                                | 540        |
| Comparaison avec les structures à pointeurs .....             | 541        |
| 2. Sondage quadratique dans les tables de hachage .....       | 541        |
| Problématique des collisions dans les tables de hachage ..... | 541        |
| Adressage ouvert vs chaînage séparé .....                     | 542        |
| Limitations du sondage linéaire .....                         | 542        |
| Principe du sondage quadratique .....                         | 543        |
| Analyse de la séquence de sondage .....                       | 543        |
| Implémentation pratique .....                                 | 544        |
| Avantages du sondage quadratique .....                        | 545        |
| Contraintes d'implémentation .....                            | 547        |
| 3. Optimisation de la division par constante .....            | 548        |
| Coût de la division sur CPU .....                             | 548        |
| Division par puissance de 2 .....                             | 548        |

|                                                                          |            |
|--------------------------------------------------------------------------|------------|
| Le Magic Number : méthode générale .....                                 | 549        |
| La solution : augmenter la précision avec le paramètre <i>shift</i> .... | 551        |
| Implémentation sur CPU .....                                             | 553        |
| Limites de l'algorithme .....                                            | 554        |
| Benchmark : validation expérimentale .....                               | 555        |
| Conclusion .....                                                         | 558        |
| <b>Glossaire .....</b>                                                   | <b>559</b> |
| <b>Index .....</b>                                                       | <b>569</b> |