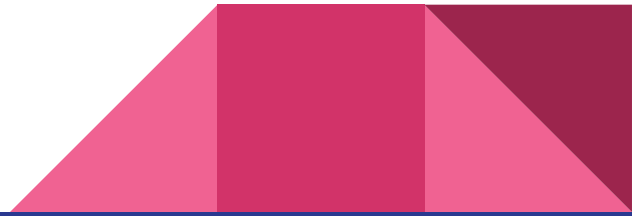
SR05 - Projet - Soutenance

Plateforme d'achat en ligne

Groupe 23 : HU Ruiqi, LI Chenxin, TIAN Linxiao, WANG Hongzhe

1. Introduction
2. Architecture du programme
3. Fonctionnment
4. Utilisation
5. Démonstration



Introduction

Scène de la conception : Plateforme d'achat en ligne

Données partagées entre les différents sites : Quantité d'articles restants

Deux fonctions principales :

Modification la copie des données & diffusion(Shopping) : Algorithme de file d'attente distribuée qui organise l'exclusion mutuelle (Estampilles)

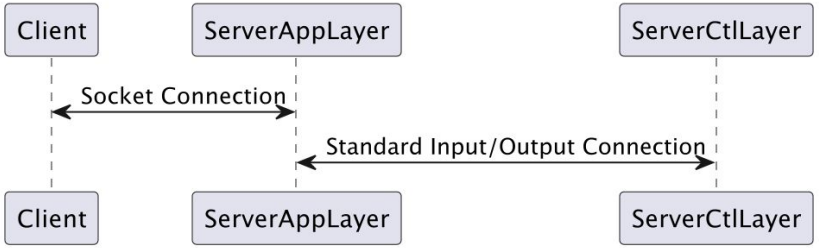
Sauvegarde distribuée obsolète(voir les enregistrements d'achat) : Algorithme avec marqueur pour créer des instantanés (Horloges vectorielles)

Version Web du client pour se connecter à chaque site

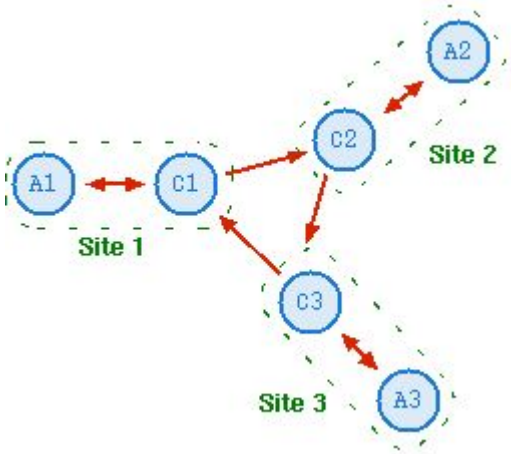
Architecture de site : Application/Contrôle

Réseau unidirectionnel en forme d'anneau de trois sites pour tester

Architecture du programme



Application



Réseaux

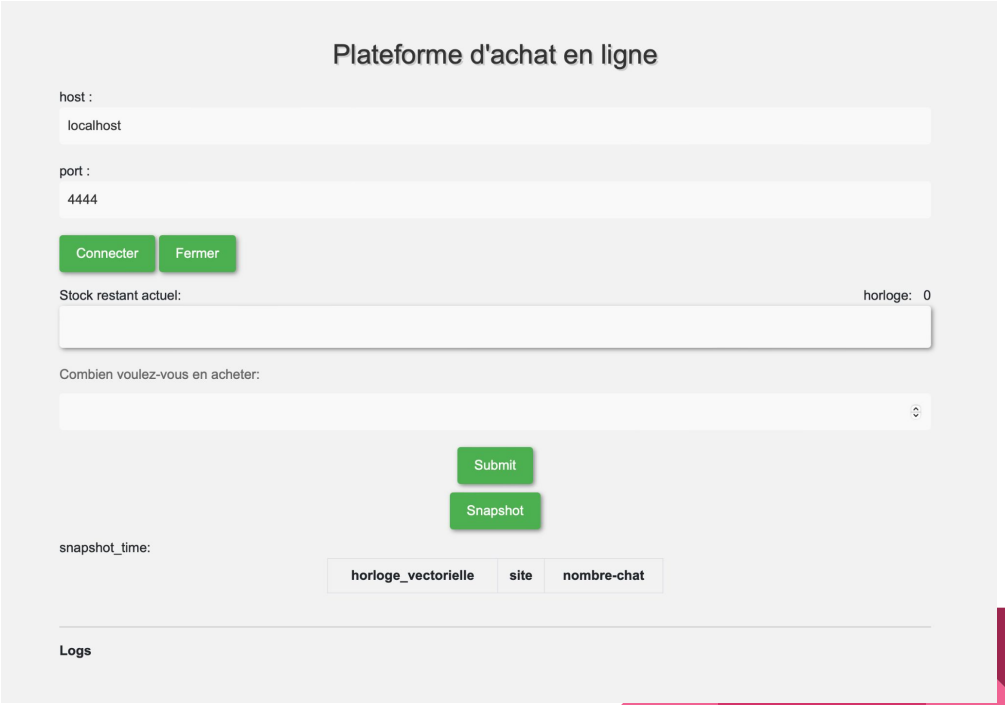
Fonctionnement - Interface visuelle utilisateur

Gestion de connexion

Gestion de donnée partagée

Snapshot

Impression des journaux



Fonctionnement - Donnée partagée entre les sites

```
type site struct {
    id          int
    logicalTime int
    tab         [N + 1][2]int
}
```

```
const (
    request messageType = iota
    release
    ack
    demandeSC
    finSC
    demandeSnap
    finSnap
)
```

```
type message struct {
    msgType     messageType
    logicalTime int
    sender      int
    receiver    int
    count       int
    h1          int
    h2          int
    h3          int
}
```

Dans la couche de contrôle

Fonctionnement - Donnée partagée entre les sites

```go
type myData struct {
    Number      string `json:"number"`
    Text        string `json:"text"`
    MyLock      string `json:"mylock"`
    Horloge     string `json:"horloge"`
    Snapshot    string `json:"snapshot"`
    Snapshottime string `json:"snapshot_time"`
}
```

```go
func ws_send(text, mylock string) {
    msg := &myData{
        Number:       strconv.Itoa(stock),
        Text:         text,
        MyLock:       mylock,
        Horloge:      strconv.Itoa(horloge),
        Snapshot:     "",
        Snapshottime: "",
    }
    err := ws.WriteJSON(msg)
    if err != nil {
        display_d("write:", string(err.Error()))
        return
    }
}
```

```go
var data myData
err = json.Unmarshal(message, &data)
if err != nil {
    l.Println("unmarshal:", err)
    return
}
```

Dans la communication de web-socket

Fonctionnement - Sauvegarde répartie datée

```
// Mettre à jour l'horloge vectorielle

arr := []int{0, msg.h1, msg.h2, msg.h3}

horloge_vec = calVec(horloge_vec, arr)

horloge_vec[s.id] += 1
```

Chaque fois que nous recevons un message de type demandeSnap, finSnap, request, release ou ack, nous mettons à jour notre vecteur horaire.

```
func calVec(x, y []int) []int {

    res := make([]int, 4)

    res[0] = 0

    res[1] = max(x[1], y[1])

    res[2] = max(x[2], y[2])

    res[3] = max(x[3], y[3])

    return res

}
```

Nous avons écrit la fonction calVec pour mettre à jour le vecteur d'horloge. les valeurs des horloges vectorielles des trois sites 1, 2 et 3 correspondent respectivement à horlog_vec[1], horlog_vec[2] et horlog_vec[3]. Le processus de modification est basé sur l'algorithme de mise à jour du vecteur d'horloge.

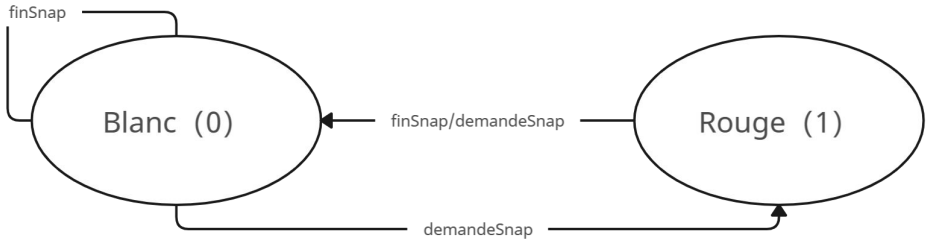
Fonctionnement - Sauvegarde répartie datée

```
snapshot=append(snapshot,
"*horloge_vectorielle:["+strconv.Itoa(horloge_vec[1])+",
"+strconv.Itoa(horloge_vec[2])+",
"+strconv.Itoa(horloge_vec[3])+"]*site:"+strconv.Itoa(s.id)+"*nombre_achat:
"+strconv.Itoa(count))
```

Réception: {"number":"2","text":"sauvegarde","mylock":"unlocked","horloge":"16","snapshot":"*horloge_vectorielle:[5, 2, 2]*site:1*nombre_achat:1@*horloge_vectorielle:[9, 4, 4]*site:1*nombre_achat:3@*horloge_vectorielle:[13, 6, 6]*site:1*nombre_achat:4","snapshot_time":"[14,6,6]"}

Nous avons ajouté une variable de tableau de chaînes d'instantanés à chaque site pour stocker les informations d'achat. Chaque fois que la couche de contrôle reçoit un message finSC, les informations sur l'achat sont ajoutées à la fin du tableau instantané, y compris l'horloge vectorielle actuelle, le site et la quantité achetée.

Fonctionnement - Sauvegarde répartie datée



Pour réaliser l'algorithme de capture d'écran, nous avons ajouté une variable couleur à chaque site, qui est blanche (0) par défaut. Lorsque le message demandeSnap est reçu :

1. Si le site actuel est blanc (0), il changera la couleur en rouge (1), enverra un message demandeSnap au prochain site. Il enverra également les instantanés précédemment stockés à la couche application et les affichera sur la page Web.
2. Si le site actuel est rouge (1), étant donné que notre configuration de site est une boucle unidirectionnelle, cela signifie que c'est ce site qui a envoyé la demande de capture de snapshot. Il enverra alors un message finSnap au prochain site et réinitialisera sa propre couleur à blanc (0).

Lorsqu'un message finSnap est reçu, seul si la couleur du propre site est rouge (1), elle sera modifiée en blanc (0) et un message finSnap sera envoyé au prochain site.

Avec cet algorithme, lors du processus de capture d'écran, l'interface utilisateur Web affiche les propres achats du client pour chaque station. Après avoir généré une capture d'écran, tous les sites reviennent à leur état initial blanc afin qu'il soit possible de capturer plusieurs fois des instantanés.

Utilisation

- **Back-end :** Compilez le programme de langage GO et exécutez-le sur le serveur.
  - Couche App : L'interface de l'application de couche supérieure, dans ce couche on peut spécifier l'IP, le port, le nombre de nœuds…
  - Couche Ctl : Implémentation de l'algorithme de base, nous pouvons contrôler des informations spécifiques telles que le nom du nœud.
  - Script de démarrage : Le script utilisé pour démarrer, créer un cluster simulé, communiquer par pipeline. On peut exécuter le script pour démarrer tous les processus(nœuds).

- **Front-end :** Ouvrez la page Web, sélectionnez l'adresse IP et le numéro de port de votre serveur, puis connectez-vous. Et ensuite, choisissez les actions que vous voulez.
  - **Acheter un produit :** Saisissez la quantité souhaitée et envoyez une demande d'achat
  - **Afficher les enregistrements d'achat :** historique des requêtes

Merci pour votre attention.

Démonstration