DAUPHINE UNIVERSITÉ PARIS | MIDO MATHEMATIQUES ET INFORMATIQUE DE LA DECISION ET DES ORGANISATIONS

# SYSTEMES ET ALGORITHMES REPARTIES

**Implementation algorithme Producteurs/Consommateur avec élection**

**Réalisé par** :

Abdelkafi Said Sofiane

Benajim Othman

Aminata DIA

Ndèye Amy Diakhaté DIOP

Samir AMRANI

Année universitaire 2018-2019

# Introduction

Le problème de producteurs/consommateur est un problème classique de synchronisation, dont nous donnons ici une solution. Le problème est celui du passage de valeurs entre des threads avec des buffers de capacité limitée. L'un des threads retire et consomme les valeurs de son buffer que les autres lui ont envoyé.

Le système est composé d'un nombre fixe de machines, tous producteurs. Ensuite il est question de déléguer le role de consommateur à un des producteurs via un algorithme d'élection.

Dans ce rapport, vous trouverez la description du projet, une architecture générale du systeme constituée par le diagramme de class et le diagramme.

# I. Fonctionnement du systéme

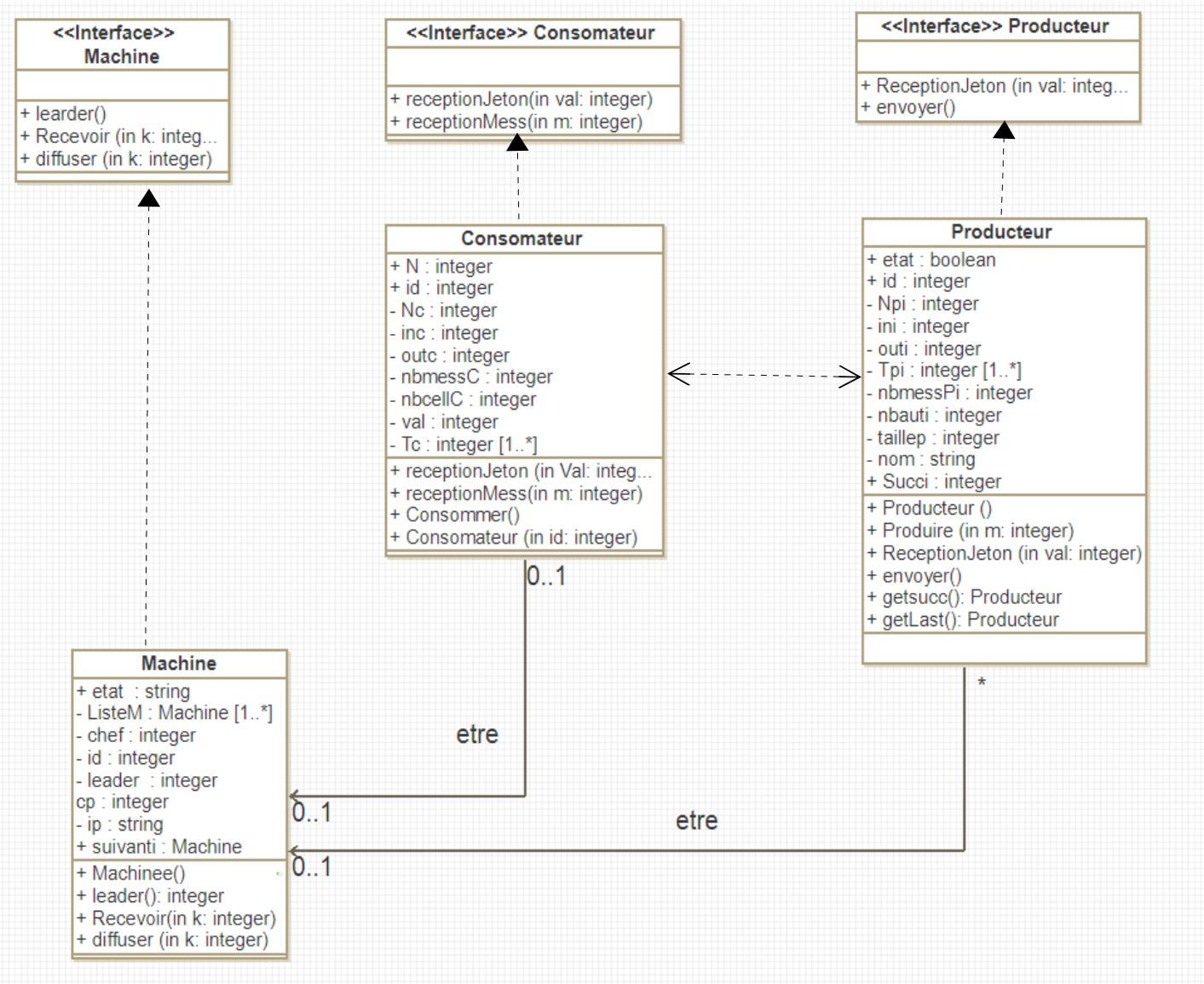
Au démarrage de l'application, on choisit le nombre de machines qui va composer notre système. Puis automatiquement, l'algorithme d'élection est lancé et permet de désigner le consommateur. Dans notre cas c'est l'algorithme de Chang et Roberts. Le principe de l'algorithme est d'élire la machine avec l'identifiant le plus petit. Pour cela, chaque machine possède en attribut son identifiant, l'identifiant de son successeur et celui du serveur élu. Initialement, chaque serveur initialise l'identifiant du serveur élu avec son propre identifiant.
Au départ, un serveur démarre la procédure d'élection en envoyant son identifiant à son successeur.
Celui-ci va comparer l'identifiant du serveur actuellement élu chez lui avec l'identifiant reçu.
Si l'identifiant reçu est inferieur à l'identifiant du serveur élu, alors l'identifiant du serveur élu est mis à jour et est envoyé à son successeur. Sinon, il ne le met pas à jour mais l'envoie à son successeur. Le processus continue ainsi jusqu'à ce qu'un serveur reçoive son propre identifiant , ce qui signifie qu'il est le serveur. Comme aucun autre serveur n'avait d'identifiant inferieur au sien, son identifiant a été transmis de machine en machine jusqu'à lui. A ce stade, nous savons donc que cette machine est élu.

Aprés définition du consommateur, la production et la consommation commencent. Le envoi en premier lieu le jeton qui contient le nombre de cases vides dans son buffer. Pour chaque Producteur, chaque valeur produite est dans un premier temps stockée dans son tampon localement. Il vérifie auparavant que son tampon ne soit pas plein ; si c'est le cas, il se met en attente jusqu'à ce qu'une case du tampon se vide. Pour envoyer les messages produits au consommateur, un producteur doit avoir des autorisations qui correspondent à des cases reservés dans le buffer du consommateur. Inversement, le consommateur retire une par une les valeurs de son tampon. Il vérifie à chaque fois que le tampon ne soit pas vide ; si c'est le cas, il se met en attente jusqu'a ce que le tampon cesse d'être vide.

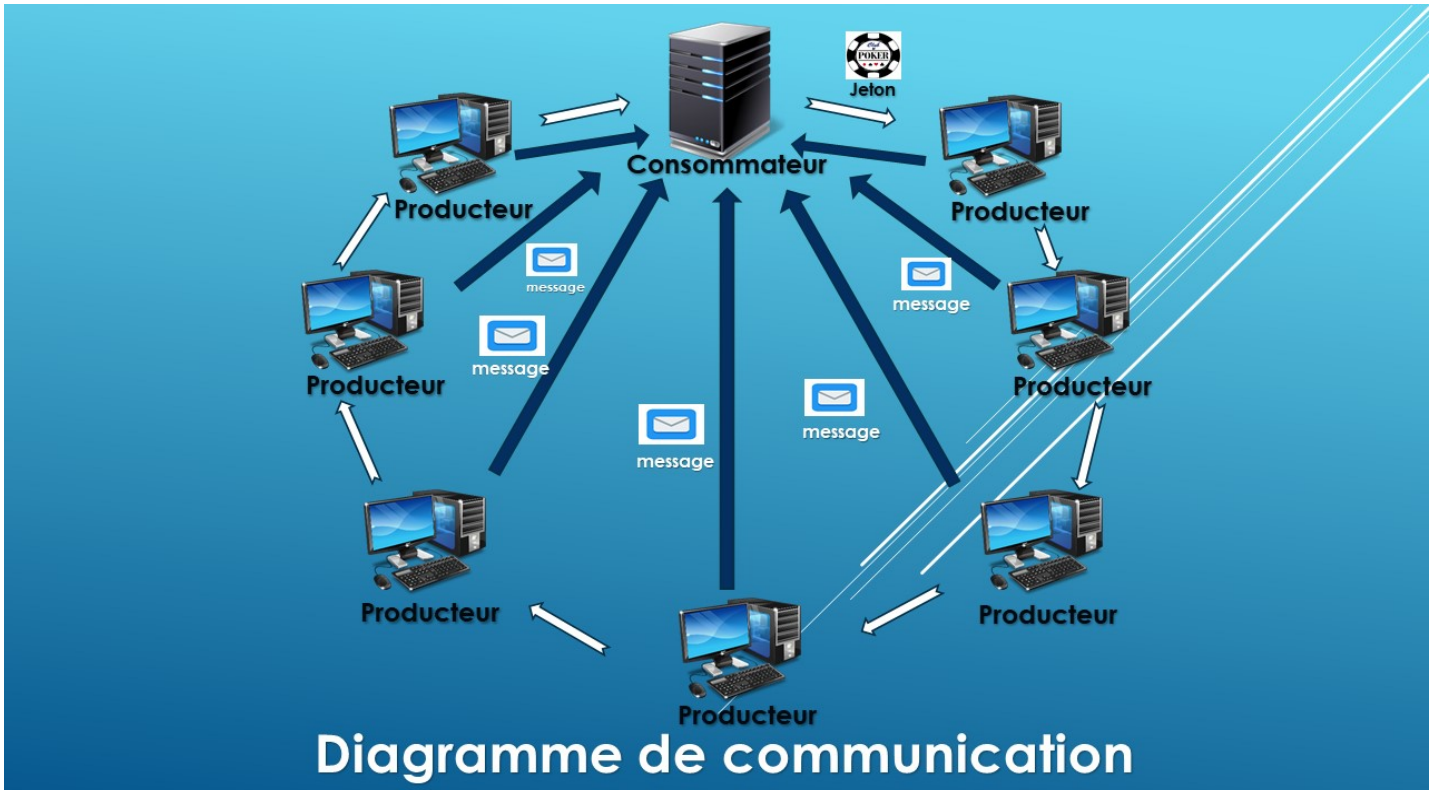
Concernant les producteurs, il est impossible qu'ils réservent au meme moment une meme case dans le buffer du consommateur. De se fait on synchronise l'accés de chaque producteur au buffer du consommateur.

## II. Architecture générale du système

- **Diagramme de classes Producteurs consommateur**



- **Diagramme de communication du systeme**




## III. Les pseudo-code de l'application

- **Algorithme d'election :Chang et Roberts**

Cet algorithme s'applique à un anneau unidirectionnel. Chaque site n'émet des messages que vers le site suivant sur l'anneau. Après fermeture de l'anneau, les machines présentes dans le système doivent procéder à l'élection de la machine Consommateur. L'un des enjeux de ce projet a été l'implémentation de cet algorithme.

- **Variables du site i**

• **suivanti** : constante contenant l'identité du site successeur de i sur l'anneau.

• **étati** : état du service. Cette variable prend une valeur parmi l'ensemble des valeurs (repos,en_cours,terminé). Cette variable est initialisée à repos.

• **chefi** : identité du site élu.

- **Algorithme du site i**

Si aucun processus d'élection n'a atteint un site **Si**, alors le service initialise un tel processus. Le service d'un site attend que le processus d'élection soit terminé pour renvoyer l'identité du site élu.

**Leader ()**

```
Début
    Si (étati==repos) Alors
        étati=en_cours;
        chefi=i;
        envoyer_à(suivanti,(req,i));
    Finsi
        Attendre(étati==terminé) ;
        renvoyer(chefi);
Fin
```

Si le site est au repos, la réception d'une requête provoque le changement d'état et la retransmission de la requête. Dans le cas où le processus reçoit une "meilleure" requête, il en tient compte et retransmet aussi la requête. Enfin si une requête parvient à son initiateur, ce site est élu et il avertit les autres sites.

**Sur_réception_de(j,(req,k))**

```
Début
    Si (étati==repos || k<chefi) Alors
        étati=en_cours; chefi=k;
        envoyer_à(suivanti,(req,k));
    Sinon
        si (i==k) Alors
            étati=terminé;
            envoyer_à(suivanti,(conf,i));
        fsi
    Finsi
```

Fin

**Sur_réception_de(j,(conf,k))**

Début

Si (i!=k) Alors

envoyer_à(suivanti,(conf,k));

étati=terminé;

Finsi

Fin

Cette primitive permet de remettre l'état des sites non élus à « terminé ».

- **Algorithme producteurs consommateur**

➢ **Les variables d'un producteur Pi**

- Ti[0..Ni-1] : tableau contenant les messages produits parle producteur Pi
- **ini** : indice d'insertion dans Ti, initialisé à 0.
- **outi** : indice d'extraction de Ti, initialisé à 0.
- **Nbmessi** : nombre de messages stockés dans Ti, initialisé à 0.
- **Nbauti** : nombre d'autorisations d'envoi de messages initialisé à 0.
- **Succi** : identificateur du site successeur du site Pi. Si i < p alors Succi prend la valeur i+1 sinon la valeur de l'identificateur du site consommateur.

➢ **Algorithme d'un producteur Pi**

**Produire(m)**

Début

Attendre (Nbmessi < Ni ) ;

Ti [ini] = m;

ini = (ini + 1) % Ni ;

nbmessi++ ;

Fin

**Sur_réception_de(j, (jeton, val) )**

```
Début
    tempi = Min( (nbmessi - nbauti) , val );
    nbauti += tempi;
    val - = tempi;
    Envoyer_à (Succi, (jeton, val) );
Fin
```

**Facteur ()**

```
Début
  Tant que (vrai)
    Attendre ( Nbauti > 0 );
    Envoyer_à( C, (app, Ti[outi] ) );
    outi = (outi + 1) % Ni;
    nbauti--;
    nbmessi--;
  Fin tant que
Fin
```

- **Les variables du consommateur C:**

- **TC [0...N-1] :** un tableau contenant les messages des sites producteurs
- **inC** : indice d'insertion dans TC , initialisé à 0.
- **outC** : indice d'extraction de TC , initialisé à 0.
- **NbmessC** : le nombre de messages stockés dans TC et non encore consommés, initialisé à 0.
- **NbcellC** : le nombre de cellules libérées entre deux passages du jeton, initialisé à 0.

- **Algorithme du consommateur**

**Consommer(m)**

```
Début
    Attendre ( NbmessC > 0 ) ;
    m = TC [outC] ;
```

```
        outC= (outC + 1) % N ;
        NbmessC -- ;
       NbcellC ++ ;
 Fin
Sur_réception_de (j, (app, m) )
 Début
        TC [inC] = m ;
        inC = (inC + 1) % N ;
        NbmessC ++ ;
 Fin
Sur_réception_de (j, (jeton, val ) )
   Début
        val += nbcellC;
        nbcellC = 0;
        envoyer_à(1,(jeton,val));
 Fin
```

## IV. Choix techniques

### ○ Algorithme d'élection

Notre topologie réseau étant en anneaux, notre choix s'est porté sur l'algorithme Chang et Roberts qui est un algorithme d'élection dans un anneau unidirectionnel ou chaque site n'émet des messages que vers le site suivant sur l'anneau et permet d'elire un leader correspondant au site qui possede le plus petit identifiant.
Cette choix est basée sur les propriétés de sureté et de vicacité de l'algorithme.

### ○ Moyen de communicaton

Pour la communication dans le système nous avons décidé d'utiliser RMI (Remote Method Invocation) , que ce soit pour l'élection ou pour l'application Producteurs/Consomateur. cette technologie présente dans le language java permet de passer des appels sur des machines distantes, les objets client peuvent ainsi accéder aux methodes des objets serveurs.
Notre application est basée sur deux couches, et l'utilisation de RMI permet de soulager les

machines clients de calculs lourds en consommation. Il s'agit d'un modèle Clients/serveur à deux niveaux.
Pour l'élection chaqu'une de nos machines fait appel aux methodes de l'objet suivant, on peut donc dire que chaque machine est client et serveur en meme temps pour le passage du jeton .
pour la partie Producteurs/Consomateur le passage du jeton repond à la meme logique que l'élection mais pour les production et l'envoie de messages, les producteurs sont les clients et ils envoient les messages produits au Consomateur et ce dernier est le serveur et consomme les messages reçus.

## V. Test

Dans cette partie, il est question de montrer les resultats obtenus. Pour lancer l'application, on utilise les commandes suivants :
pour la generation du stub et du squeleton

```
[Minatu@M1TDSI-Amina ~]$ rmic Consomateur Producteur Machinee
```

Ensuite on lance le serveur de nom avec :

```
[Minatu@M1TDSI-Amina ~]$ rmiregistry &
[1] 16199
[Minatu@M1TDSI-Amina ~]$ ps
  PID TTY          TIME CMD
15823 pts/0    00:00:00 bash
16199 pts/0    00:00:00 rmiregistry
16223 pts/0    00:00:00 ps
```

Enfin l'application est prés à etre executé.
Dés le lancement de l'algorithme, le systeme nous demande de choisir le nombre de machines.

```
[Minatu@M1TDSI-Amina src]$ java Maintest

-------------- BIENVENU --------------

Veuillez  saisir le nombre de machine
4

Creation de 4 machines
```

Ensuite le systeme désigne le consommateur.

```
-------------- ELECTION --------------
je suis 0
je suis 2
je suis 1
je suis 3
0 est le Consommateur
```

Et la production et la consommation commencent.

```
-------------- Production & Consommation --------------
je suis le Producteur 1
je suis le Producteur 2
je suis le Producteur 3
Producteur 1: production de 0
Producteur 3: production de 0
Producteur 2: production de 0
Producteur 1 : reception de jeton 5
Producteur 1: production de 1
Producteur 3: production de 1
Producteur 2: production de 1
Producteur 2 : reception de jeton 3
Producteur 3: production de 2
Producteur 2: production de 2
Producteur 3 : reception de jeton 0
Producteur 3: production de 3
Consommateur: reception du message 0
Consommateur: consommation de 0
Producteur 3: production de 4
Consommateur: reception du message 0
Consommateur: reception du message 1
Producteur 1: production de 2
Consommateur reception de jeton: 0
Producteur 1 : reception de jeton 1
Consommateur: consommation de 0
Consommateur: consommation de 1
Producteur 1: production de 3
Consommateur: reception du message 1
Producteur 2 : reception de jeton 0
Consommateur: consommation de 1
Consommateur: reception du message 2
Producteur 2: production de 3
Producteur 3 : reception de jeton 0
Consommateur: consommation de 2
```

## Conclusion

Ce projet entre dans le cadre de notre formation en systemes et algorithmes réparties. Il nous a permis de mettre en pratique les theories acquises en cours en s'inspirant des TP.
Durant la realisation de ce projet, des choix ont été fait comme le choix de RMI pour la communication entre les entités du systeme.