SYNTHÈSE ET CONCLUSION

DÉFINITION DES POINTS DE CONTRÔLE

OBJECTIF

• relancer une exécution à partir de points de reprise
• calculer des propriétés définies sur des états globaux

CONTEXTE

• Tout processus a la possibilité de définir certains de ses états locaux comme des points de contrôle, pour des raisons qui lui sont propres :
occurrence d'une échéance éventuellement périodique,
réception d'un signal particulier,
passage à vrai d'un prédicat local

• Si les processus définissent leurs points de contrôle indépendamment les uns des autres, il est possible qu'il n'y ait pas cohérence

• d'où une coordination est nécessaire entre processus :
Les processus définissent des états locaux comme *points de contrôle spontanés*
Un mécanisme de coordination les oblige à définir d'autres états locaux comme *points de contrôle forcés*

CLASSES DE PROTOCOLES DE DÉFINITION DE POINTS DE CONTRÔLE

• *protocoles à synchronisation explicite* : la coordination utilise des messages de contrôle qui sont ajoutés à ceux de l'application
exemple : utiliser un protocole de détermination d'état global

• *protocoles à synchronisation implicite* : la coordination utilise les messages de l'application pour véhiculer des informations de contrôle
exemples : sauvegarde adaptative, stratégie de Russel

• Nota : même avec coordination, il peut rester encore des cycles en zigzag qu'on n'a pas su détecter. Il faut ensuite faire un retour arrière vers un état global cohérent construit avec les états locaux disponibles.

EXEMPLE DE PROTOCOLE À SYNCHRONISATION EXPLICITE

MARQUEUR

• On utilise des messages de contrôle appelés marqueurs
• Tout marqueur mk possède la propriété suivante : sur le canal où il est émis, tous les messages émis avant mk sont délivrés avant mk et tous les messages émis après mk sont délivrés après mk.

PROTOCOLE

soit lc_i l'estampille du dernier point de contrôle local de Pi

- (R1) Point de contrôle spontané : chaque fois que Pi prend un tel point de contrôle, il incrémente lc_i de 1 , attribue la valeur de lc_i à $c_{i,x}$.t et envoie des marqueurs $mk(lc_i)$ à chacun des processus

- (R2) Point de contrôle forcé : lorsque Pi reçoit $mk(lc)$, si $lc_i < lc$ il prend un point de contrôle forcé $c_{i,x}$, recale lc_i ($lc_i := lc$) et attribue la valeur de lc à $c_{i,x}$.t ; de plus il diffuse à son tour des marqueurs comme dans le cas R1

Ce protocole satisfait le théorème précédent, il n'y a pas de point de contrôle inutile (critère P1).
Tous les points de contrôle locaux de même estampille définissent un point de contrôle global cohérent (critère P2) . Un site qui reçoit copie des points de contrôle des autres sites doit regrouper ces copies par estampille et peut utiliser comme point de reprise le dernier groupe complet, soit GL (pour une estampille donnée, il a tous les points de contrôles locaux). Le site n'a pas à remonter plus en amont que GL. Le site n'a pas à sauvegarder les groupes en amont de GL.

Le coût en nombre de messages est proportionnel au nombre de canaux.

Ce protocole dérive directement de celui de Chandy Lamport 1985

EXEMPLE DE PROTOCOLE À SYNCHRONISATION IMPLICITE

PRINCIPE

faire transporter les estampilles par les messages de l'application

PROTOCOLE

- (R1) Point de contrôle spontané : chaque fois que Pi prend un tel point de contrôle, il incrémente lc_i et attribue la valeur de lc_i à $c_{i,x}.t$

- (R2) Point de contrôle forcé :

• Tout message *m* transporte la valeur courante lc_j de son émetteur Pj (soit *m.lc* cette valeur)

• lors de la réception d'un message *m* , Pi test si $lc_i < m.lc$. Si tel est le cas, Pi recale lc_i à la valeur de *m.lc* et définit l'état courant comme point de contrôle $c_{i,x}$ (avec $c_{i,x}.t := lc_i$)

Il n'y a pas de Z-cycle, donc pas de point de contrôle inutile (critère P1)
On a un algorithme pour trouver le dernier point de contrôle global (critère P2)
Ce protocole a été introduit par [Briatico 84], voir aussi [Hélary 97]

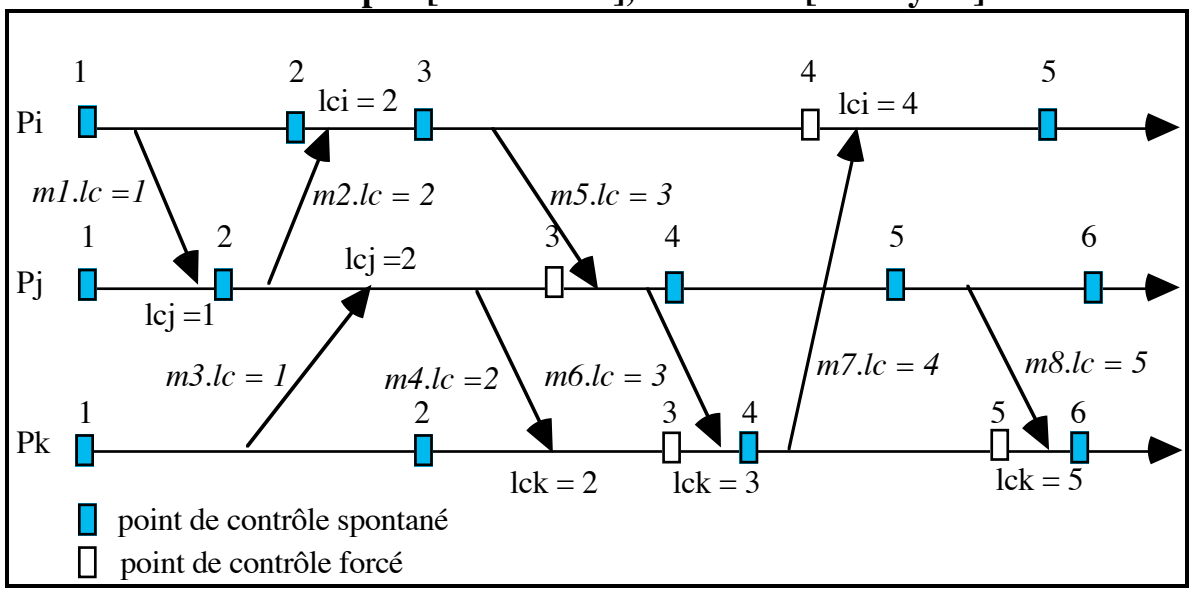

Fig.6. Points de contrôle forcés par synchronisation implicite simple

• Problèmes généraux :
- comment réduire le nombre des points de contrôle forcés sans laisser de Z-cycles?
- comment obtenir le dernier point de reprise global le plus efficacement possible?

STRATÉGIE DE RUSSELL

Cette stratégie élimine tous les Z-chemins , par ajout de point de contrôle forcés.

Soit *ckpt* le méta-événement qui consiste à définir un état local comme point de contrôle local et faisons abstraction des événements autres que *send, receive et ckpt*.
Si chaque processus obéit au comportement

(*receive* send* ckpt*)*

Propriété P1 : il ne peut pas y avoir de Z-chemin non causal

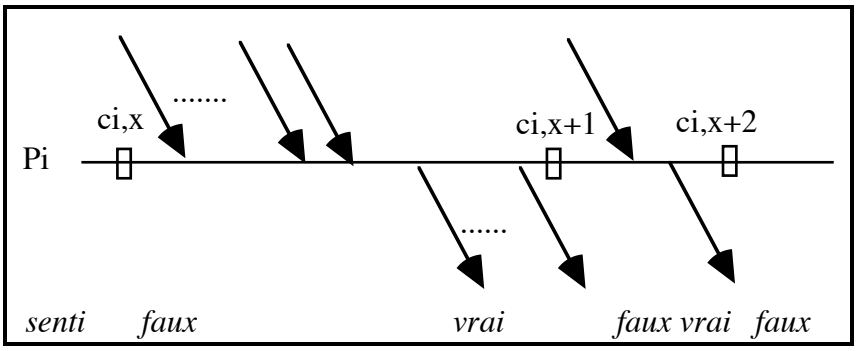

Fig.8. stratégie de Russell (1980)

PROTOCOLE DE RUSSELL

Chaque processus Pi est doté d'une variable booléenne $sent_i$ qui est initialisée à *faux* au début de chaque intervalle et qui est mise à *vrai* lorsque Pi envoie un message
- Point de contrôle forcé : à la réception d'un message , Pi prend un point de contrôle si la condition $sent_i$ est vraie (et ensuite remet $sent_i$ à *faux*)
- La prise de point de contrôle spontané n'appelle pas de traitement particulier si on veut seulement garantir l'absence de Z-chemin

Si on veut associer à tout point de contrôle local, le premier point de contrôle global cohérent, il faut gérer des horloges vectorielles afin de repérer les dépendances causales entre les points de contrôles locaux. Cela conduit à mettre à jour l'horloge lors de la prise de points de contrôle spontanés, à estampiller les messages et à mettre à jour l'horloge lors de la réception de message et lors de la prise de points de contrôle forcés.

Article de synthèse en français : J.M.HÉLARY, A.MOSTÉFAOUI, et M.RAYNAL. Points de contrôle cohérents dans les systèmes répartis : concepts et protocoles. *TSI (Technique et science informatique)* vol. 17(10), p. 1223-1245, octobre 1998.