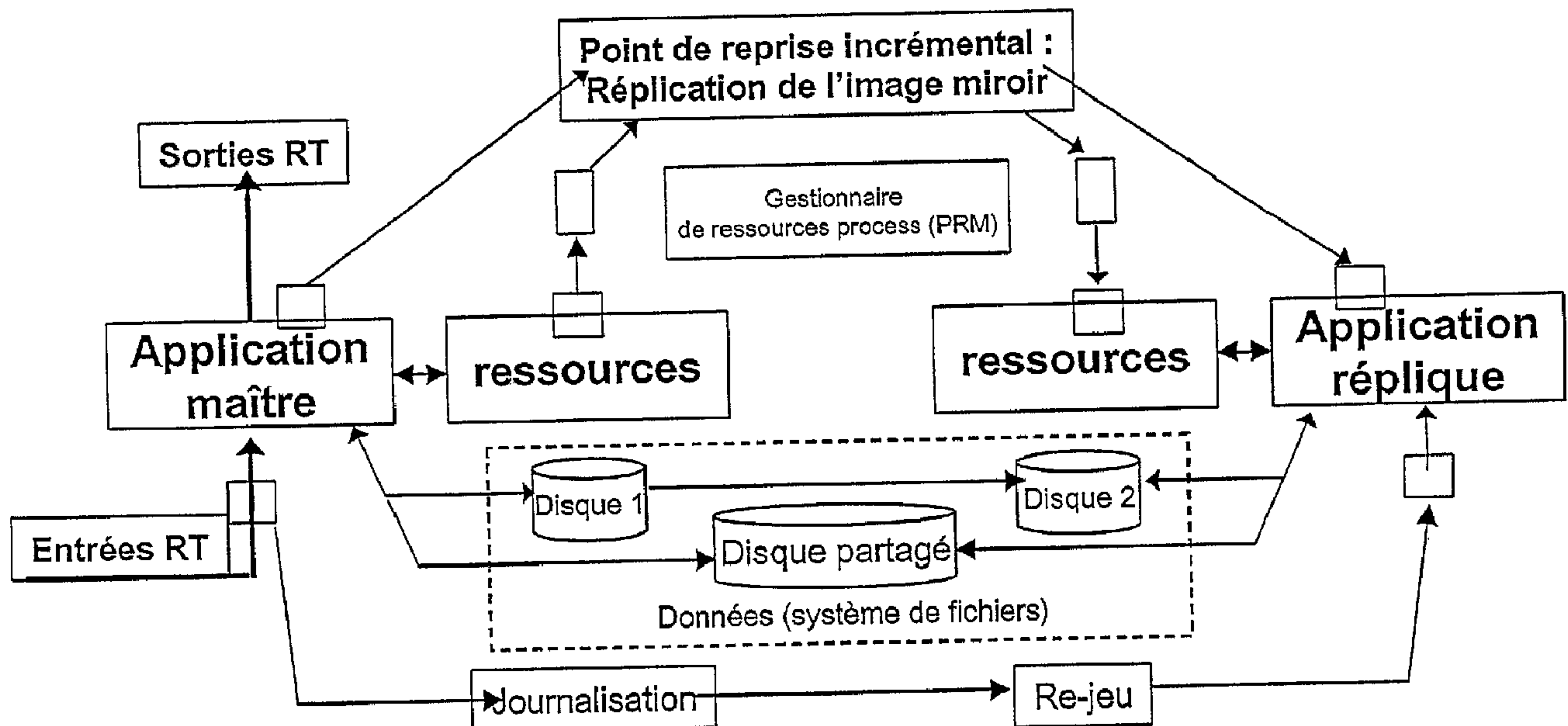




(86) Date de dépôt PCT/PCT Filing Date: 2003/07/28
 (87) Date publication PCT/PCT Publication Date: 2004/02/19
 (45) Date de délivrance/Issue Date: 2010/05/11
 (85) Entrée phase nationale/National Entry: 2005/01/26
 (86) N° demande PCT/PCT Application No.: FR 2003/002371
 (87) N° publication PCT/PCT Publication No.: 2004/015574
 (30) Priorité/Priority: 2002/08/02 (FR02/09855)

(51) Cl.Int./Int.Cl. *G06F 11/20* (2006.01)
 (72) Inventeurs/Inventors:
 VERTES, MARC, FR;
 DUFOUR, LAURENT, FR;
 RICHARD, FRANCOIS, FR;
 KURZ, GREGORY, FR
 (73) Propriétaire/Owner:
 INTERNATIONAL BUSINESS MACHINES
 CORPORATION, US
 (74) Agent: WANG, PETER

(54) Titre : PROCÉDE DE REPLICATION D'UNE APPLICATION LOGICIELLE DANS UNE ARCHITECTURE MULTI-ORDINATEURS, PROCÉDE POUR REALISER UNE CONTINUITÉ DE FONCTIONNEMENT METTANT EN OEUVRE CE PROCÉDE DE REPLICATION, ET SYSTÈME MULTI-ORDINATEURS AINSI ÉQUIPE
 (54) Title: METHOD FOR REPLICATING A SOFTWARE APPLICATION IN A MULTI-COMPUTER ARCHITECTURE, METHOD FOR CARRYING OUT FUNCTIONAL CONTINUITY IMPLEMENTING SAID REPLICATION METHOD, AND MULTICOMPUTER SYSTEM PROVIDED THEREWITH



(57) Abrégé/Abstract:

Procédé pour répliquer une application logicielle dans une architecture multi-ordinateurs (cluster), cette application logicielle étant préalablement exécutée sur un premier ordinateur du cluster constituant un nœud primaire ou opérationnel et étant destinée à être répliquée sur au moins un autre ordinateur du cluster constituant un nœud secondaire, comprenant une répliquée des ressources associées à cette application logicielle. Ce procédé comprend une mise à jour au fil de l'eau des ressources répliquées par un mécanisme d'introspection dynamique prévu pour fournir la structure de l'application à répliquer, ainsi que le graphe dynamique des ressources et dépendances mises en œuvre.

(12) DEMANDE INTERNATIONALE PUBLIÉE EN VERTU DU TRAITÉ DE COOPÉRATION
EN MATIÈRE DE BREVETS (PCT)(19) Organisation Mondiale de la Propriété
Intellectuelle
Bureau international(43) Date de la publication internationale
19 février 2004 (19.02.2004)

PCT

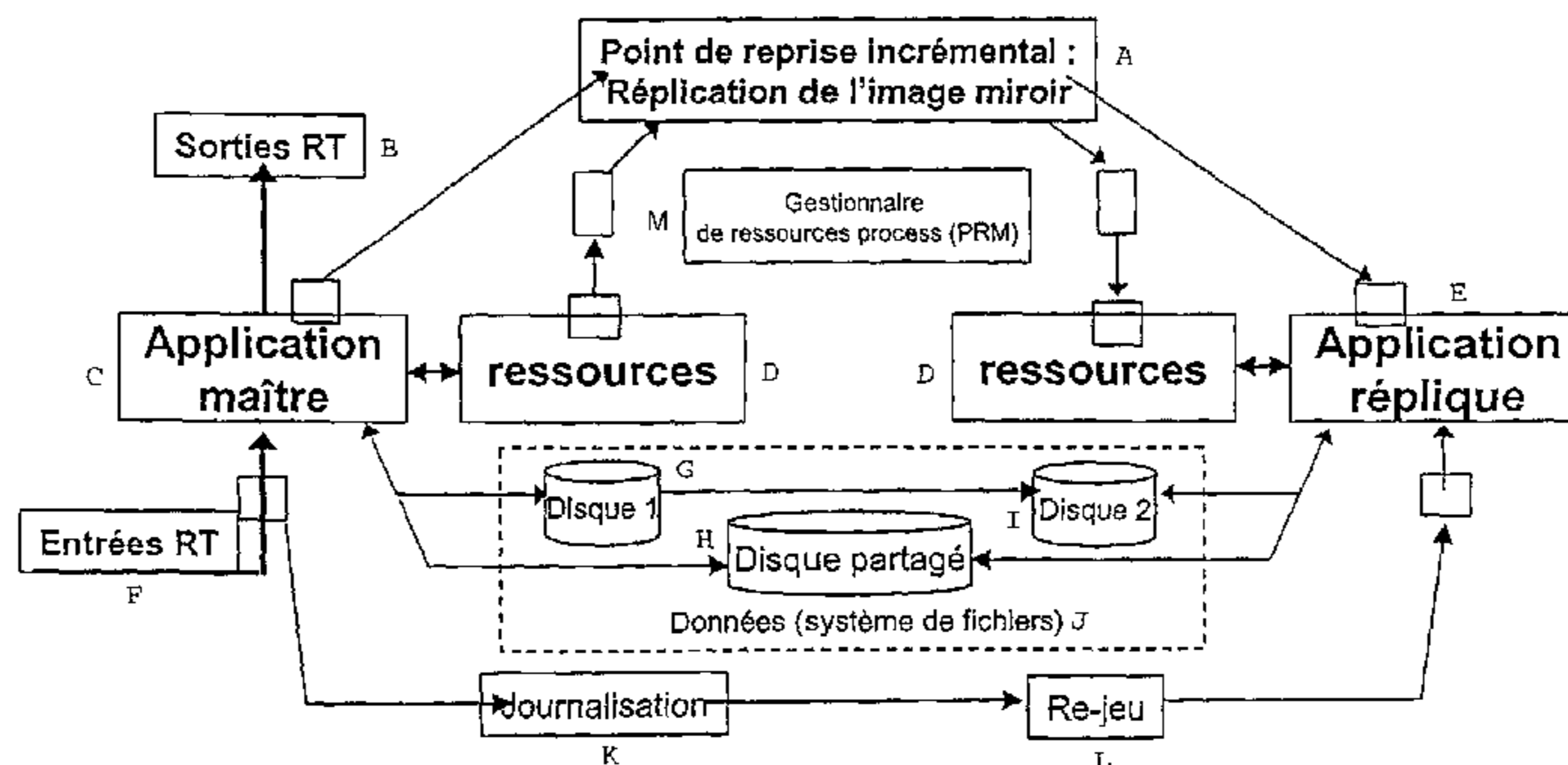
(10) Numéro de publication internationale
WO 2004/015574 A3

- (51) Classification internationale des brevets⁷ : G06F 11/20
- (21) Numéro de la demande internationale : PCT/FR2003/002371
- (22) Date de dépôt international : 28 juillet 2003 (28.07.2003)
- (25) Langue de dépôt : français
- (26) Langue de publication : français
- (30) Données relatives à la priorité : 02/09855 2 août 2002 (02.08.2002) FR
- (71) Déposant (pour tous les États désignés sauf US) : MEIOSYS [FR/FR]; Centre Industriel d'Innovation de Basso Cambo, 42, avenue du Général de Crouette, F-31100 Toulouse (FR).
- (72) Inventeurs; et
- (75) Inventeurs/Déposants (pour US seulement) : VERTES, Marc [FR/FR]; 57, rue de Terris, F-31830 Plaisance du Touch (FR). DUFOUR, Laurent [FR/FR]; 3, impasse des Noisetiers, F-31830 Plaisance du Touch (FR). RICHARD, François [FR/FR]; 2, impasse des Garrabiers, F-31120 Lacroix Falgarde (FR). KURZ, Gregory [FR/FR]; 56, route de Tarbes, F-31170 Tournefeuille (FR).
- (74) Mandataires : ALLANO, Sylvain etc.; Pontet Allano & Associés S.E.L.A.R.L., 25, rue Jean-Rostand, Parc Club Orsay Université, F-91893 Orsay Cedex (FR).
- (81) États désignés (national) : AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX,

[Suite sur la page suivante]

(54) Title: FUNCTIONAL CONTINUITY BY REPLICATING A SOFTWARE APPLICATION IN A MULTI-COMPUTER ARCHITECTURE

(54) Titre : CONTINUITÉ DE FONCTIONNEMENT PAR REPLICATION D'UN LOGICIEL DANS UNE ARCHITECTURE MULTI-ORDINATEURS



- A INCREMENTAL RESTART POINT: MIRROR IMAGE REPLICATION
B RT OUTPUTS
C MASTER APPLICATION
D RESOURCES
E REPLICATED APPLICATION
F RT INPUTS
G DISK 1
H SHARED DISK
I DISK 2
J DATA (FILE SYSTEM)
K LOGGING
L REPLAY
M PROCESS RESOURCE MANAGER (PRM)

(57) Abstract: Disclosed is a method for replicating a software application in a multi-computer architecture (cluster). Said software application is executed on a first computer of said cluster, which represents a primary or operational node, and is replicated on at least one other computer of the cluster, which represents a secondary node, comprising replication of the resources associated with said software application. The inventive method comprises streamlined updating of the replicated resources by means of a dynamic introspection mechanism supplying the structure of the application that is to be replicated and the dynamic graph of the implemented resources and dependencies.

[Suite sur la page suivante]

WO 2004/015574 A3

WO 2004/015574 A3

MZ, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

— *avec revendications modifiées*

(84) États désignés (régional) : brevet ARIPO (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), brevet eurasien (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), brevet européen (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), brevet OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

(88) Date de publication du rapport de recherche internationale:

2 septembre 2004

Date de publication des revendications modifiées:

16 décembre 2004

Publiée :

— *avec rapport de recherche internationale*

En ce qui concerne les codes à deux lettres et autres abréviations, se référer aux "Notes explicatives relatives aux codes et abréviations" figurant au début de chaque numéro ordinaire de la Gazette du PCT.

(57) Abrégé : Procédé pour répliquer une application logicielle dans une architecture multi-ordinateurs (cluster), cette application logicielle étant préalablement exécutée sur un premier ordinateur du cluster constituant un nœud primaire ou opérationnel et étant destinée à être répliquée sur au moins un autre ordinateur du cluster constituant un nœud secondaire, comprenant une réplification des ressources associées à cette application logicielle. Ce procédé comprend une mise à jour au fil de l'eau des ressources répliquées par un mécanisme d'introspection dynamique prévu pour fournir la structure de l'application à répliquer, ainsi que le graphe dynamique des ressources et dépendances mises en œuvre.

«Procédé de répllication d'une application logicielle dans une
architecture multi-ordinateurs, procédé pour réaliser une
continuité de fonctionnement mettant en œuvre ce procédé de
5 répllication, et système multi-ordinateurs ainsi équipé »

La présente invention concerne un procédé pour répliquer
une application logicielle dans une architecture multi-
10 ordinateurs (cluster). Elle vise également un procédé pour
réaliser une continuité de fonctionnement d'une application
logicielle au sein d'un cluster d'ordinateurs, qui met en
œuvre le procédé de répllication selon l'invention, ainsi
qu'un système multi-ordinateurs implémentant ce procédé de
15 continuité de fonctionnement.

Le domaine de l'invention est celui des clusters
d'ordinateurs formés de plusieurs ordinateurs collaborant
entre eux. Ces clusters sont par exemple prévus pour exécuter
des applications logicielles. Ainsi, à un instant donné, une
20 application est exécutée sur l'un des ordinateurs du cluster,
appelé nœud primaire ou opérationnel (OP), tandis que les
autres ordinateurs du cluster sont appelés nœuds secondaires
ou « stand-by » (SB), dans un contexte d'architecture
redondante.

25 Or, l'exploitation de tels clusters montre que se posent
des problèmes de fiabilité qui peuvent être dus à des
défaillances du matériel ou du système d'exploitation, à des
erreurs humaines, ou à la défaillance des applications elles-
mêmes.

30 Pour résoudre ces problèmes de fiabilité, il existe
actuellement des mécanismes, dits de haute disponibilité, qui
sont mis en œuvre sur la plupart des clusters actuels et qui
sont basés sur un redémarrage automatique à froid de

l'application sur un nœud de secours parmi l'un des nœuds secondaires du cluster.

Or ces mécanismes basés sur un redémarrage automatique ne permettent pas d'assurer une continuité totale du services
5 fourni par l'application en cours d'exécution au moment de la défaillance.

En particulier, se pose le problème de la réplification d'une application logicielle au sein d'une architecture multi-ordinateurs, cette réplification devant assurer une
10 continuité totale de service.

Un objectif principal de la présente invention est donc de proposer un procédé pour répliquer une application logicielle dans une architecture multi-ordinateurs (cluster), ladite application logicielle étant préalablement exécutée
15 sur un premier ordinateur dudit cluster constituant un nœud primaire et étant destinée à être répliquée sur au moins un autre ordinateur dudit cluster constituant un nœud secondaire, comprenant une réplification des ressources associées à ladite application logicielle.

Cet objectif principal est atteint avec un tel procédé de réplification caractérisé en ce qu'il comprend une mise à jour au fil de l'eau des ressources répliquées par un mécanisme d'introspection dynamique prévu pour fournir la structure de l'application à répliquer, ainsi que le graphe
20 dynamique des ressources et dépendances mises en œuvre.

On peut en outre avantageusement prévoir que ce procédé de réplification comprenne en outre une création et une maintenance d'un arbre de dépendance, qui fournit à chaque instant des informations sur les ressources qu'il est
30 nécessaire de répliquer.

Il est important de noter que dans le procédé de réplification selon l'invention, le nombre de nœuds secondaires ou stand-by concernés peut être quelconque.

Dans un mode de réalisation préféré de l'invention, le procédé de répllication selon l'invention comprend en outre un mécanisme dit de « point de reprise (« checkpointing ») par lequel les ressources à répliquer sont répliquées sur un ou
5 plusieurs nœuds secondaires.

Le procédé de répllication selon l'invention peut avantageusement comprendre les trois étapes suivantes:

- capture des ressources sur le nœud primaire,
- transfert par le réseau vers un ou plusieurs nœuds
10 secondaires,
- restauration sur le ou les nœuds secondaires.

Le procédé de répllication selon l'invention peut être avantageusement mis en oeuvre pour une optimisation automatique de ressources informatiques par partage de charge
15 par répartition dynamique de processus. Il peut aussi être utilisé pour une maintenance non interruptive par relocation à la demande de processus au travers d'un réseau de ressources informatiques, ou pour une préservation de contexte applicatif dans des applications mobiles.

20 Un autre but de la présente invention est de proposer un procédé pour réaliser une continuité de fonctionnement d'une application logicielle dans une architecture multi-ordinateurs (cluster), cette application étant exécutée à un instant donné sur l'un des ordinateurs du cluster, appelé
25 nœud principal ou opérationnel, les autres ordinateurs dudit cluster étant appelés nœuds secondaires.

Cet autre objectif est atteint avec un procédé pour réaliser une continuité de fonctionnement d'une application logicielle dans une architecture multi-ordinateurs (cluster),
30 cette application étant exécutée à un instant donné sur l'un des ordinateurs du cluster, appelé nœud principal, les autres ordinateurs dudit cluster étant appelés nœuds secondaires.

Suivant l'invention, le procédé comprend les étapes suivantes :

- 5 - répllication de l'application sur au moins des nœuds secondaires, de façon à réaliser au moins un clone de ladite application,
- mise à jour au fil de l'eau dudit ou desdits clones, et
- en cas de détection d'une défaillance ou d'un événement affectant ledit nœud principal, basculement de service vers l'un au moins desdits clones.

10 Ainsi, avec le procédé de réalisation de continuité de fonctionnement selon l'invention, il est désormais possible de disposer de nœuds secondaires pourvus de clones d'application résultant de la répllication de l'application et aptes à relayer sans discontinuité cette application en cas
15 de détection de défaillance ou d'événement affectant le nœud principal.

La répllication mise en œuvre dans le procédé de répllication selon l'invention est avantageusement de type holistique. On dispose ainsi d'un clonage de l'application au
20 fil de l'eau, avec une mise à jour de ces clones, de façon déterministe et complète.

Ces clones sont dits « chauds », c'est-à-dire qu'ils sont la réplique exacte de l'application et de tout son contexte opératoire. Ils sont mis à jour régulièrement
25 (périodiquement ou sur événements caractéristiques). Ces clones contiennent toutes les ressources et informations requises par l'application pour fournir son service.

Le procédé de répllication selon l'invention permet en outre de superviser l'état de toutes les ressources
30 nécessaires au bon fonctionnement de l'application. Quand la dégradation rédhibitoire de l'une d'entre elles est détectée, le procédé de répllication selon l'invention prévoit une

élection d'un clone comme nouveau primaire et lui ordonne de prendre la main.

Cette élection est appelée basculement et est transparente pour le reste du monde qui communique avec l'application : bien que le nœud primaire soit mis hors service, le service fourni par l'application n'est pas interrompu car il est repris avec tout son contexte par le clone élu.

On peut ainsi garantir que tout message transmis par le reste du monde à l'application sera traité, soit par le nœud primaire (pré-basculement), soit par le clone (post-basculement). Pour ce faire, le procédé de réplication selon l'invention peut en outre comporter un enregistrement sur chaque clone (en plus du mécanisme de clonage périodique) de tous les messages reçus par le primaire depuis la dernière mise à jour des clones. Ces messages seront réinjectés dans le clone élu nouveau primaire en cas de basculement.

La réplication holistique reprend des mécanismes déjà mis en œuvre dans des systèmes existants de migration de process. Cependant, la conception et l'utilisation qui en sont faites dans le procédé de réplication selon l'invention diffèrent de tous les travaux antérieurs connus.

Le procédé de réalisation de continuité de fonctionnement selon l'invention met ainsi en œuvre une réplication transparente, holistique et optimisée, dédiée à la continuité de service par délocalisation de l'application et virtualisation des ressources.

Avec ce procédé, on résout plusieurs limitations des implémentations classiques qui les rendaient inopérantes pour une utilisation en tolérance aux pannes au sein d'une architecture multi-ordinateurs en cluster.

Une première limitation résidait dans le problème de l'indépendance entre nœud primaire et nœud secondaire. Dans

les systèmes classiques, la réplique d'une ressource d'un nœud primaire vers un nœud secondaire présuppose et nécessite la présence opérationnelle du nœud primaire pendant tout le procédé. Le procédé de réplique selon l'invention résout
5 cette limitation, puisque le clone peut à tout instant vivre de façon autonome même en cas de disparition du primaire. Cette dé-corrélation primaire / secondaire est un pré requis pour la tolérance aux pannes.

Comme la réplique implémentée dans le procédé de
10 réplique selon l'invention est holistique, on capture l'intégralité cohérente de ressources asynchrones interdépendantes. Dans les procédés de l'art antérieur, seuls étaient capturés les états de ressources indépendantes.

Une autre limitation des procédés de l'art antérieur
15 résidait dans le problème de l'intrusivité. Le procédé de réplique selon l'invention est non intrusif sur le code source : les instances antérieures nécessitent de modifier le code source (ou de le concevoir explicitement) pour que les processus informatiques créés et les ressources utilisées
20 puissent être migrés.

Il est à noter que pour la mise en œuvre du procédé de réplique selon l'invention, on peut avantageusement utiliser des techniques d'ingénierie logicielle non intrusive dynamique, qui ont fait l'objet d'une demande de brevet
25 publiée le 2 août 2002 sous le numéro FR2820221. Ces techniques d'ingénierie logicielle permettent de manipuler des applications dans leur représentation binaire (exécutable), de façon à rendre le procédé de réalisation de continuité de fonctionnement selon l'invention transparent
30 pour l'application et donc générique.

Suivant un autre aspect de l'invention, il est proposé un système multi-ordinateurs prévu pour exécuter sur au moins des ordinateurs au moins une application logicielle,

implémentant le procédé pour réaliser une continuité de fonctionnement selon l'invention.

D'autres avantages et caractéristiques de l'invention apparaîtront à l'examen de la description détaillée d'un mode
5 de mise en œuvre nullement limitatif, et des dessins annexés sur lesquels :

- la figure 1 illustre la fonction de miroir dynamique mise en œuvre dans le procédé de répllication selon l'invention ;
- 10 -la figure 2 illustre schématiquement les principes de répllication de données mis en œuvre dans le procédé de répllication selon l'invention ;
- la figure 3 représente un exemple d'architecture logicielle mettant en œuvre le procédé de répllication
15 selon l'invention, pour la supervision et la détection de défaillance ;
- la figure 4 illustre schématiquement des principes de supervision mis en œuvre dans le procédé de répllication selon l'invention ;
- 20 -la figure 5 illustre schématiquement le mécanisme de copie sur écriture (« copy on write ») mis en œuvre dans le procédé de répllication selon l'invention ;
- la figure 6 illustre schématiquement le mécanisme d'incrémentation pour répllication mis en œuvre dans le
25 procédé de répllication selon l'invention ; et
- la figure 7 illustre schématiquement le mécanisme de commutation mis en œuvre dans le procédé de répllication selon l'invention.

On va d'abord décrire, en référence aux figures
30 précitées, le fonctionnement du mécanisme de répllication holistique mis en œuvre dans le procédé de répllication selon l'invention.

Pour que l'application puisse correctement tourner sur un nœud secondaire dans le cas d'un basculement, il est nécessaire que l'ensemble des ressources requises par cette application soit également répliqué sur le nœud secondaire.

5 Si ces ressources sont des ressources à état, i.e. qu'elles varient au fil de l'exécution de l'application et contribuent à son contexte global, alors leur état doit également être capturé et répliqué de façon cohérente.

L'ensemble de ces ressources est découvert à
10 l'initialisation de l'application puis maintenu à jour au fil de l'eau par des mécanismes d'introspection dynamique qui permettent d'obtenir automatiquement la structure de l'application à protéger, ainsi que le graphe dynamique des ressources et dépendances mises en œuvre.

15 Ces mécanismes s'appuient sur les caractéristiques réflexives des binaires, sur les mécanismes d'héritage des systèmes d'exploitation et sur la surveillance, via une instrumentation binaire, des mécanismes - incluant les appels système - qui contribuent à modifier l'état de ses
20 ressources.

En référence à la figure 4, dans un exemple de mise en œuvre du procédé de réplication selon l'invention, des pilotes (drivers) d'introspection et de surveillance assurent une surveillance sur tous les nœuds du cluster et
25 transmettent des données de surveillances à la base d'information de gestion MIB du système. Cette base MIB est sollicitée à la fois dans la gestion du cluster sur le nœud opérationnel pour les déclenchements de point de reprise (checkpoint) et dans la gestion du cluster sur des nœuds
30 « back-up ». La base MIB est également sollicitée par le gestionnaire de supervision avec une base MIB synthétique et est accédée par l'administrateur système auquel sont associées des interfaces utilisateur graphiques (GUI).

Le résultat de ce travail de découverte et d'introspection au fil de l'eau est la création et la maintenance d'un « arbre de dépendance », qui fournit au procédé de répllication selon l'invention à chaque instant des
5 informations sur les ressources qu'il est nécessaire de répliquer. L'existence de ce graphe garantit la complétude et la cohérence des clones.

Un autre mécanisme de point de reprise (« checkpointing »), mis en œuvre dans le procédé de
10 réalisation de continuité de fonctionnement selon l'invention, consiste à répliquer les ressources sur un ou plusieurs nœuds secondaires. Ce mécanisme de répllication des ressources est réalisé en trois étapes :

- capture des ressources sur le nœud primaire,
- 15 - transfert par le réseau vers un ou plusieurs nœuds secondaires, et
- restauration sur le ou les nœuds secondaires.

Les ressources répliquées incluent :

- la mémoire virtuelle de chaque processus concerné
20 ainsi que sa pile d'appel,
- des ressources systèmes (inter process communication, connexion réseau, etc.), et
- des données écrites sur disques.

Le mécanisme de répllication des ressources assure que
25 l'ensemble des ressources nécessaires à l'application est transféré de façon complète et cohérente (d'où holistique).

La mise en œuvre du procédé de répllication selon l'invention garantit que l'application puisse continuer de vivre sur le secondaire sans perdre son contexte :
30 l'application est délocalisée, le matériel et le système d'exploitation sous-jacent sont virtualisés, l'application se comportant indépendamment de sa localisation physique.

En cas de basculement, l'application n'est pas considérée comme stoppée : elle continue de tourner dans son contexte, mais sur d'autres ressources matérielles.

Les ressources mises en œuvre par l'application sont
5 diverses et variées (multi process, système d'exploitation, etc.). Elle vit de façon asynchrone, sur un environnement non déterministe.

Le procédé de réplication selon l'invention met en œuvre un algorithme de « checkpointing » (génération de points de
10 reprise) asynchrone : une barrière de synchronisation est transmise à toutes les ressources et le procédé de réplication selon l'invention garantit que la capture d'état est complète et cohérente.

On va maintenant décrire une technique d'optimisation
15 mise en œuvre dans le procédé de réplication selon l'invention. La capture déterministe et complète de l'état de toutes les ressources est coûteuse pour les performances du système. Or, un faible impact sur les performances de l'application est un pré requis pour l'acceptabilité par le
20 marché du produit et donc in fine pour son utilité. Plusieurs techniques d'optimisation ont donc été conçues et développées pour minimiser cet impact.

Premièrement, le point de reprise quasi synchrone est une optimisation des mécanismes de génération de point de
25 reprise (checkpointing) classiques : il offre la cohérence de capture des algorithmes synchrones, sans pour autant nécessiter l'arrêt total du système pendant la capture que nécessitent les algorithmes asynchrones.

La période du point de reprise est ajustable, de sorte à
30 optimiser le compromis entre le temps de reprise après basculement (potentiellement d'autant plus long que la période entre deux points de reprise est longue) et la quantité d'information d'état à capturer et à transférer.

Par ailleurs, le point de reprise est incrémental : seules les différences d'état entre deux points de reprise sont transmises, comme l'illustre l'exemple fonctionnel de la figure 1. Dans cet exemple, un point de reprise incrémental
5 est effectué à partir de l'application maître pour obtenir l'application réplique et un disque partagé entre les deux nœuds primaire et secondaire est mis en œuvre.

Ainsi, le premier point de reprise est cher en performance (initialisation des clones) mais les suivants
10 sont d'impact faible.

Le point de reprise est également discriminant : l'analyse intelligente du graphe de dépendance permet de limiter au strict nécessaire la quantité d'information à transmettre.

15 Enfin, des mécanismes de « Copy On Write » (copie sur écriture) offerts par le système d'exploitation sont mis en œuvre pour séparer le temps de capture du temps de transfert, en référence à la figure 5 qui illustre un exemple de mise en œuvre d'un mécanisme de copie sur écriture dans un procédé de
20 réplification selon l'invention, à la suite du déclenchement d'un point de reprise. Dans cet exemple, le mécanisme de copie sur écriture intervient sur des blocs de données (mémoire ou disque) pour une nouvelle référence, après une requête en écriture issue d'un Utilisateur via un process ou
25 un i-node (nœud d'index). Seuls les blocs de données modifiés sont répliqués, comme l'illustre la figure 6.

On va maintenant décrire un exemple de mise en œuvre du mécanisme de génération de point de reprise quasi-synchrone au sein du procédé de réalisation de continuité de
30 fonctionnement selon l'invention. Ce mécanisme inclut :

- une barrière de synchronisation de processus (« Process Synchronisation Barrier » : PSB),

- une gestion de ressources (« Ressource Management » : RM),
- une gestion de ressources système (« System Resources Management » : (SRM), et
- 5 - une gestion de ressources de processus (« Process Resources Management » (PRM).

La barrière de synchronisation de processus (PSB) est un mécanisme permettant de synchroniser le blocage des processus composant une application tout en respectant la gestion des entrées/sorties en cours, dans le but de pouvoir prendre à un instant T une "photographie" non floue de l'état du système et de l'application.

La gestion de ressources (RM) est un ensemble d'automates génériques permettant de mettre en oeuvre le séquencement des différentes phases du checkpointing vis à vis des différentes ressources nécessaires pour répliquer une application d'une machine sur une autre:

La gestion de ressources système (SRM) inclut des mécanismes permettant de gérer les différentes routines de gestion des ressources systèmes utilisées par une application (ensemble de processus) lors des différentes phases du mécanisme de génération de point de reprise.

La gestion de ressources de processus (PRM) inclut des mécanismes permettant de gérer les différentes routines de gestion des ressources utilisées par un processus lors des différentes phases du mécanisme de génération de point de reprise. Ce code est chargé dynamiquement à l'intérieur des processus applicatifs lors de leur lancement.

Il existe aujourd'hui trois phases principales qui sont elles même découpées en différentes phases nécessaires pour la captures des ressources utilisées par l'application.

La raison d'être de ces différentes sous-phases est de minimiser les temps de blocages applicatif liés à la récupération/restauration des différentes ressources applicatives et système ainsi que de garantir la cohérence
5 des informations sauvegardées.

DUMP:

RM_PRE_DUMP

RM_DUMP

RM_POST_DUMP

10 RM_ABORT_DUMP

RESTORE:

RM_PRE_RESTORE,

RM_RESTORE,

RM_POST_RESTORE,

15 RM_ABORT_RESTORE,

SWITCH:

RM_PRE_SWITCH,

RM_SWITCH,

RM_POST_SWITCH,

20 RM_ABORT_SWITCH,

On va maintenant décrire une virtualisation des ressources systèmes dans le cadre du procédé de réplication selon l'invention. Certaines ressources systèmes UNIX sont
25 caractérisées par un identifiant unique propre à chaque machine. Ces identifiants sont stockés sous forme de variables par les applications, ce qui leur permet de pouvoir les référencer. Lors de la migration d'une application d'une machine à une autre la mémoire des applications est
30 intégralement transférée, y compris les données relatives aux ressources systèmes. Pour pouvoir garantir l'unicité du référencement des ressources système par une application

META-CLUSTER met en oeuvre des mécanismes de virtualisation de ces ressources, permettant de maintenir des identifiants uniques entre les différentes machines composant un CLUSTER.

Ces mécanismes de virtualisation sont aujourd'hui
5 appliqués pour les identifiants de ressources système suivants :

- Processus
- PIPE
- FIFO
- 10 - IPC System V
 - Mémoires partagées
 - Sémaphores
 - Message queues
- Socket AF UNIX
- 15 - Threads

Les mécanismes de virtualisation assurent donc l'unicité du référencement d'une ressource système au sein du cluster ainsi que sa translation vers une ressource système sur chaque machine. Ils sont implémentés sous la forme de modules
20 noyau dynamiques assurant la non-préemptivité de requêtes qui leurs sont faites, sur des systèmes mono et multiprocesseurs, et avec une capacité à introspecter les différentes routines permettant de manipuler ces identifiants.

A titre d'exemple non limitatif, une routine *getpid* est
25 instrumentée par META lors de la prise en charge de l'application par META-CLUSTER et son utilisation par l'application retourne un *CLUSTER_PID* qui pourra ensuite être utilisé par l'application sur toute routine prenant un PID comme paramètres (*kill*, *waitpid*, ...).

30 Le procédé de répllication selon l'invention inclut aussi un module de répllication de fichiers de données applicatives entre le nœud opérationnel et le nœud stand-by, désigné sous le terme de CFOR (Cluster File system Optimized Replication :

Réplication Optimisée de système de Fichier pour Cluster), en en référence à la figure 2.

Le module CFOR réalise ainsi les fonctions suivantes :

- a) écriture de données
- 5 b) modification du Log (journal)
- c) ordre de réplication
- d) synthèse construite à partir des données
- e) synthèse de transfert
- f) mise à jour du système de fichiers (FS)

10 Le fonctionnement de ce module de réplication est le suivant : entre chaque copie (dump), CFOR construit à la volée un journal cumulatif et synthétique des modifications apportées au système de fichiers par les applications contrôlées par le cluster.

15 Les modifications du système de fichier sont extraites à la volée par instrumentation dans les processus applicatifs des divers appels systèmes: write(2), mkdir(2), rmdir(2), unlink(2)... Le principe consiste à ne mémoriser que les actions sans les données. Ainsi si une application écrit 2Mo
20 dans un fichier, on ne mémorise que l'action "fichier, écriture, début, fin", les 2Mo de données ayant été sauvegardés par l'OS sur le disque, il n'est pas nécessaire de les dupliquer ailleurs.

Les écritures multiples sont synthétisées au fil de
25 l'eau. Si une application effectue les actions suivantes:

- 1.ouverture du fichier 'toto'
- 2.écriture de 30000 octets à l'offset 30 dans le fichier toto
- 3.écriture de 20000 octets à l'offset 20 dans le fichier
30 toto
- 4.fermeture du fichier 'toto'

Le log CFOR résultant sera:

•fichier toto, 20 ? 30030

Au moment de la copie (dump), les données structurelles (metadata) ainsi que le contenu des modifications sont enregistrés dans un fichier séparé.

5 Ce fichier séparé est transmis sur le nœud stand-by et son exploitation permet de synchroniser l'arborescence de manière à ce qu'elle soit strictement identique à celle du nœud opérationnel au moment du dump.

10 On va maintenant décrire le mécanisme de synchronisation mis en œuvre dans le procédé de répllication selon l'invention.

Lors de l'apparition d'une machine SB, il faut synchroniser son système de fichiers (FS) par rapport à celui du nœud opérationnel OP. Cette synchronisation doit se faire
15 sans bloquer le nœud opérationnel OP donc elle se fait sur un système de fichiers en constante évolution. Afin d'éviter le problème des images floues, la synchronisation se fait au travers d'un instantané (snapshot) du système de fichiers (file system) du nœud opérationnel OP. La procédure est
20 décomposée en 2 phases afin de limiter la taille du log de CFOR.

- 1.Création d'un instantané (snapshot) sur le nœud opérationnel OP
- 25 2.lère synchro avec le nœud SB
- 3.Destruction de l'instantané (snapshot) sur le nœud opérationnel OP
- 4.Activation du log de CFOR et création d'un 2ème instantané (snapshot) sur le nœud opérationnel OP
- 30 5.2ème synchronisation avec le nœud SB (elle doit être la plus courte possible).

6. Suppression de l'instantané (snapshot) sur le nœud opérationnel OP, le nœud SB étant prêt à recevoir un première copie (dump) totale.

7. A la prochaine copie (dump), transfert du log de CFOR et
5 mise à jour du système de fichiers FS du nœud SB avec les données de CFOR

8. le cycle normal des copie/restauration (dump/restore) est en place.

La recopie de la mémoire d'un processus se fait en
10 analysant dynamiquement l'organisation mémoire interne du processus et en isolant les différentes zones :

- texte
- données
- pile d'exécution

15 L'analyse de la mémoire est effectuée sans intrusion dans le code utilisateur en s'appuyant sur les données fournies par le système d'exploitation (Operating System). Ces données sont capturées et analysées dans le contexte même du processus et permettent de créer la table des zones
20 mémoires utilisées.

Une fois l'analyse terminée, les agents d'interposition des appels système d'allocation/libération mémoire assurent le suivi de l'évolution de la table des zones mémoires.

Lors de la copie (dump), seules les zones mémoire
25 modifiables, c'est à dire accessibles en écriture sont transférées sur le nœud stand-by où elles seront recopiées. Ainsi le processus sur le nœud stand-by contient les mêmes zones mémoires avec les mêmes données que sur le nœud opérationnel.

30 La sauvegarde du contenu de la mémoire devant être atomique du point de vue d'un processus, elle doit se faire sans que le processus ne puisse en changer l'état, donc processus bloqué. Afin de ne pas bloquer le processus trop

longtemps, on s'appuie sur le mécanisme de « Copy On Write » du système d'exploitation (primitive (fork) par exemple) pour créer une copie de l'image mémoire du processus et on transfert cette image vers les nœuds stand-by. Une fois le
5 transfert terminé, l'image mémoire maintenue par les mécanismes de « Copy On Write » est supprimée.

Le procédé de répllication selon l'invention met aussi en œuvre un mécanisme de copie (dump) incrémentale qui s'appuie sur l'analyse mémoire, mais ajoute en plus un mécanisme de
10 protection des pages en écriture.

Une fois l'analyse des pages effectuée, toutes les pages accessibles en écriture sont protégées, c'est à dire qu'une écriture dans une de ces pages déclenche l'émission d'un signal de violation de protection de pages.

15 La protection s'appuie sur les mécanismes fournis par le système d'exploitation tel que l'appel système « mprotect ».

Lorsque l'application tente de modifier une donnée, la ou les pages contenant cette donnée sont marquées comme modifiées et déprotégées. Le déroulement du code applicatif
20 n'est pas impacté par l'adjonction de ces mécanismes (non intrusivité).

Lors d'une copie (dump) incrémentale, seules les pages modifiées depuis le précédent dump sont transférées. La copie (dump) terminée, toutes les pages modifiées sont re-protégées
25 afin de détecter les prochaines écritures. La copie (dump) incrémentale permet de réduire la taille des données à transférer vers le stand-by à chaque copie (dump).

La gestion des déclenchements des points de reprise peut être effectuée à partir de la base MIB qui reçoit, du nœud
30 primaire ou opérationnel, des informations sur les états du système et de l'application, des informations sur les événements et call-back sur l'application et des informations délivrées par un analyseur d'état synthétique, comme

l'illustre la figure 7. L'organisation du basculement de l'application du nœud primaire vers un nœud secondaire agit par exemple sur un dernier point de reprise EVENT, un dernier point de reprise PERIODIC, un enregistrement (logging)

5 d'entrée, et peut inclure :

- le choix d'un scénario de basculement,
- le choix d'un point de reprise,
- le déclenchement de la restauration,
- le déclenchement (ou non) du re-jeu du Log

10 - la notification du nouveau nœud opérationnel.

Bien sûr, l'invention n'est pas limitée aux exemples qui viennent d'être décrits et de nombreux aménagements peuvent être apportés à ces exemples sans sortir du cadre de l'invention.

REVENDICATIONS

1. Procédé pour répliquer une application logicielle comprenant un ensemble de processus dans une architecture multi-ordinateurs de type cluster, ladite application
5 logicielle étant préalablement exécutée sur un premier ordinateur dudit cluster constituant un noeud primaire et étant destinée à être répliquée sur au moins un autre ordinateur dudit cluster constituant un noeud secondaire, le procédé comprenant les étapes consistant à:
- découvrir les ressources requises par l'application, ces ressources incluant:
10
 - la mémoire virtuelle de chaque processus concerné ainsi que sa pile d'appel,
 - des ressources systèmes incluant au moins une structure logicielle incluse dans la mémoire de travail du noeud primaire, et
 - des données écrites sur disques;
 - 15 -créer et maintenir un graphe de dépendance contenant des informations sur lesdites ressources découvertes,
 - répliquer lesdites ressources découvertes du noeud primaire vers certains au moins desdits noeuds secondaires à partir du graphe de dépendances de manière à capturer de manière cohérente et complète au niveau de chaque noeud secondaire l'état
20 des ressources requises par l'application dans le noeud primaire, lesdites étapes de découverte des ressources, de création et de maintien du graphe de dépendance étant mises en oeuvre par un mécanisme d'introspection dynamique opérant dans le contexte de chaque processus, ledit mécanisme d'introspection étant apte à analyser dynamiquement l'organisation de la mémoire interne de chaque processus sans intrusion
25 dans le code utilisateur à partir de données fournies par le système d'exploitation.
2. Procédé de réplication selon la revendication 1, caractérisé en ce que l'étape de réplication met en oeuvre un mécanisme de génération de point de reprise (checkpointing), par lequel les ressources à répliquer sont répliquées sur un ou plusieurs
30 noeuds secondaires.
3. Procédé de réplication selon la revendication 2, caractérisé en ce que le mécanisme de génération de point de reprise comprend trois étapes:
- 35
 - capture des ressources sélectionnées sur le noeud primaire,
 - transfert par le réseau vers un ou plusieurs noeuds secondaires, et
 - restauration sur le ou les noeuds secondaires.

4. Procédé de réplication selon l'une quelconque des revendications 2 et 3, caractérisé en ce le mécanisme de génération de point de reprise est quasi-synchrone.
5. Procédé de réplication selon l'une des revendications 3 à 4, caractérisé en ce que le mécanisme de génération de point de reprise est incrémental.
6. Procédé de réplication selon l'une des revendications 3 à 5, caractérisé en ce que le mécanisme de génération de point de reprise est discriminant.
7. Procédé de réplication selon l'une des revendications 4 à 6, caractérisé en ce que le mécanisme de génération de point de reprise inclut au moins l'une des fonctions suivantes:
- une barrière de synchronisation de processus (PSB),
 - une gestion de ressources (RM),
 - une gestion de ressources système (SRM), et
 - une gestion de ressources de processus (PRM).
8. Procédé de réplication selon l'une quelconque des revendications précédentes, caractérisé en ce qu'il comprend en outre un mécanisme de réplication de fichiers de données applicatives entre le nœud primaire (OP) sur lequel l'application est exécutée et un desdits nœuds secondaires.
9. Procédé pour réaliser une continuité de fonctionnement d'une application logicielle dans une architecture multi-ordinateurs (cluster), cette application étant exécutée à un instant donné sur l'un des ordinateurs du cluster, appelé nœud primaire, les autres ordinateurs dudit cluster étant appelés nœuds secondaires, caractérisé en ce que, en cas de détection d'une défaillance ou d'un événement affectant ledit nœud principal, il met en oeuvre le procédé de réplication selon l'une quelconque des revendications précédentes pour un basculement de service vers lesdits nœuds secondaires.
10. Procédé de continuité de fonctionnement selon la revendication 9, caractérisé en ce que la réplication de l'application est de nature holistique.
11. Procédé de continuité de fonctionnement selon l'une des revendications 9 et 10, caractérisé en ce qu'il comprend en outre une mise à jour des clones de l'application.

12. Procédé de continuité de fonctionnement selon l'une des revendications 9 à 11, caractérisé en ce qu'il comprend en outre une supervision de l'état de ressources nécessairement au fonctionnement de l'application.

5

13. Procédé de continuité de fonctionnement selon l'une des revendications 9 à 12, caractérisé en ce qu'il comprend en outre, à la suite d'une détection d'une défaillance ou d'un événement affectant le noeud primaire, une étape pour élire, parmi lesdits noeuds secondaires, un clone, le clone ainsi élu devenant le nouveau noeud primaire.

10

14. Procédé de continuité de fonctionnement selon l'une des revendications 9 à 13, caractérisé en ce qu'il comprend en outre un enregistrement sur chaque noeud secondaire de messages reçus par le noeud primaire, ces messages étant réinjectés dans le clone élu en cas de basculement.

1/7

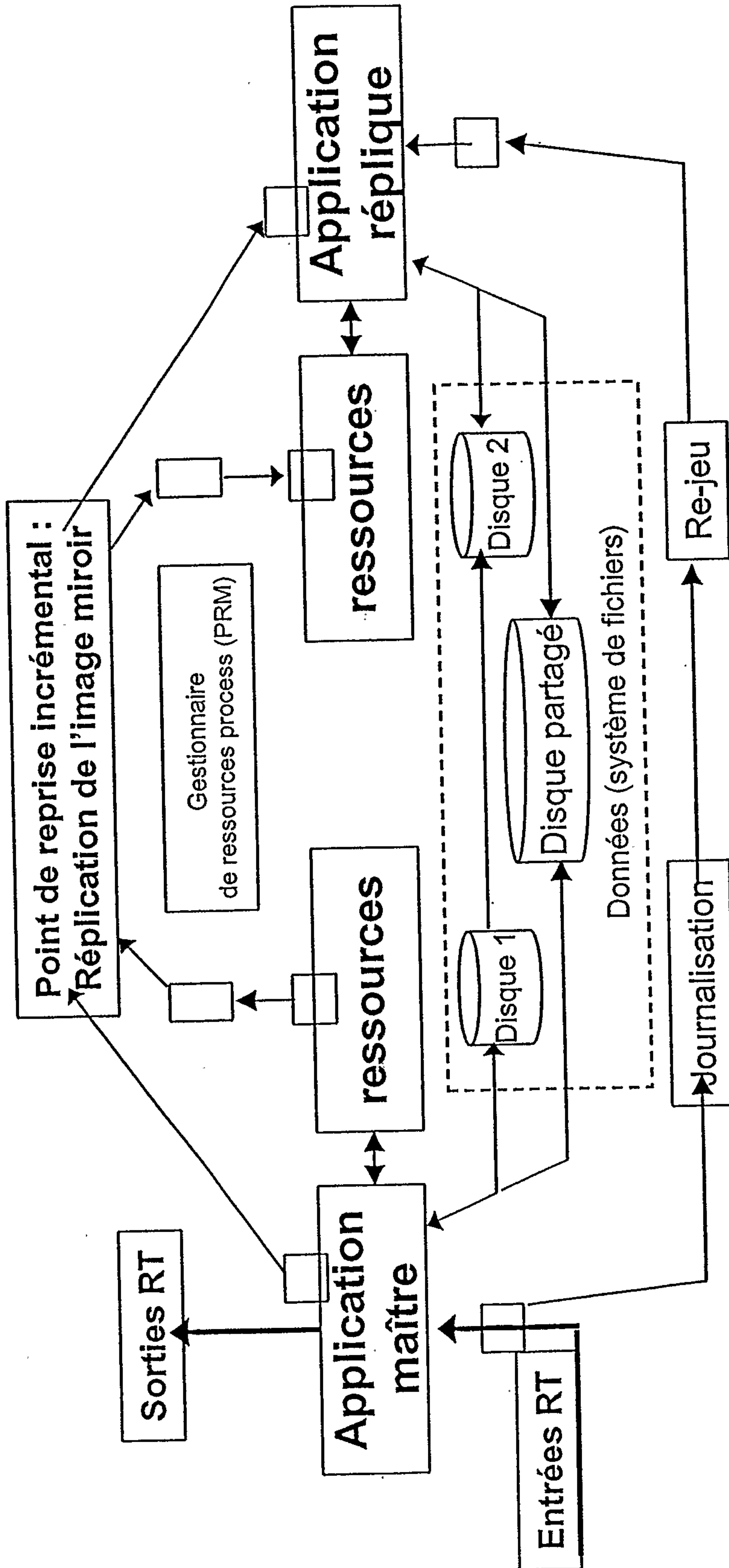


FIG.1

2/7

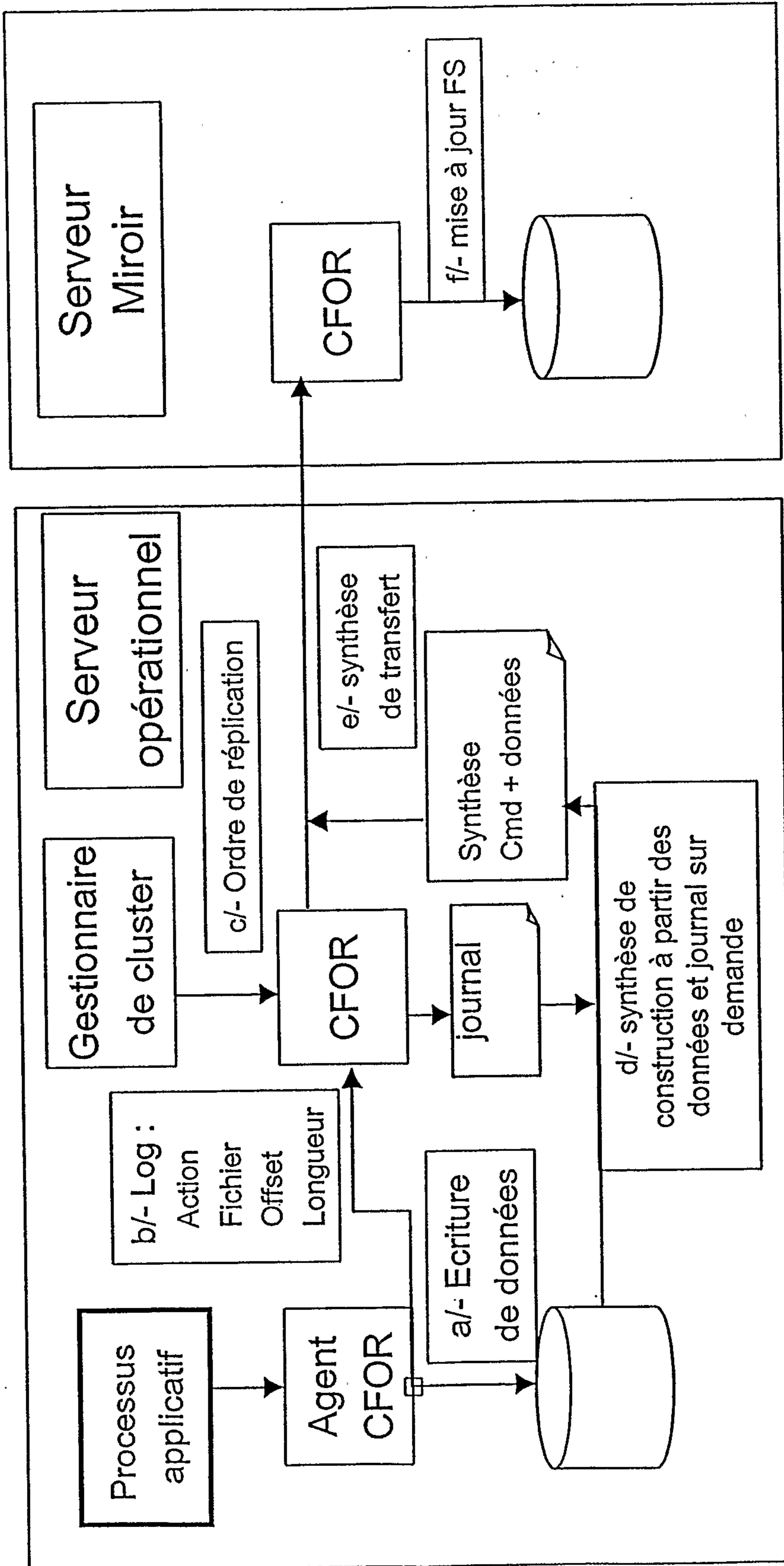


FIG.2

3/7

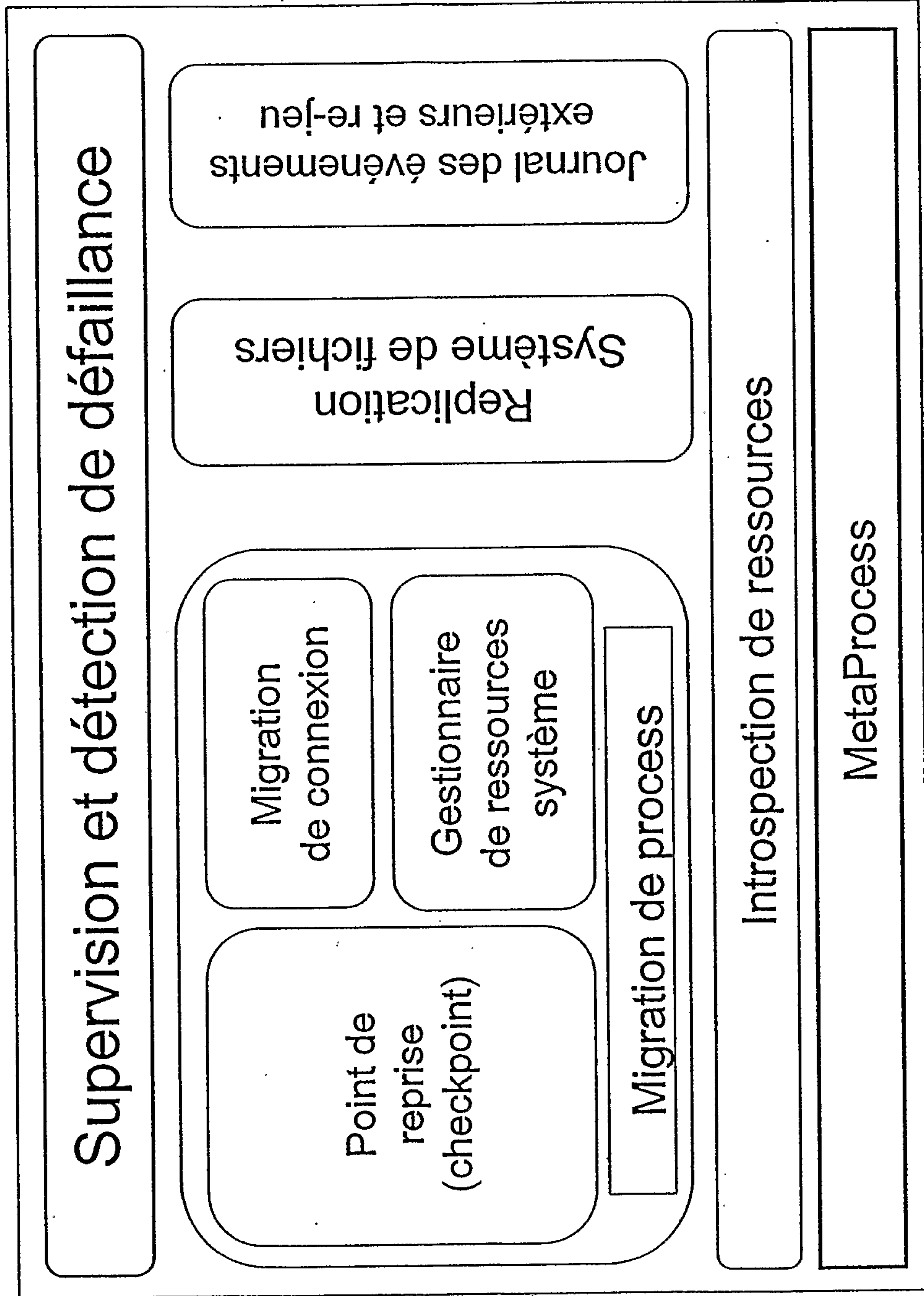


FIG.3

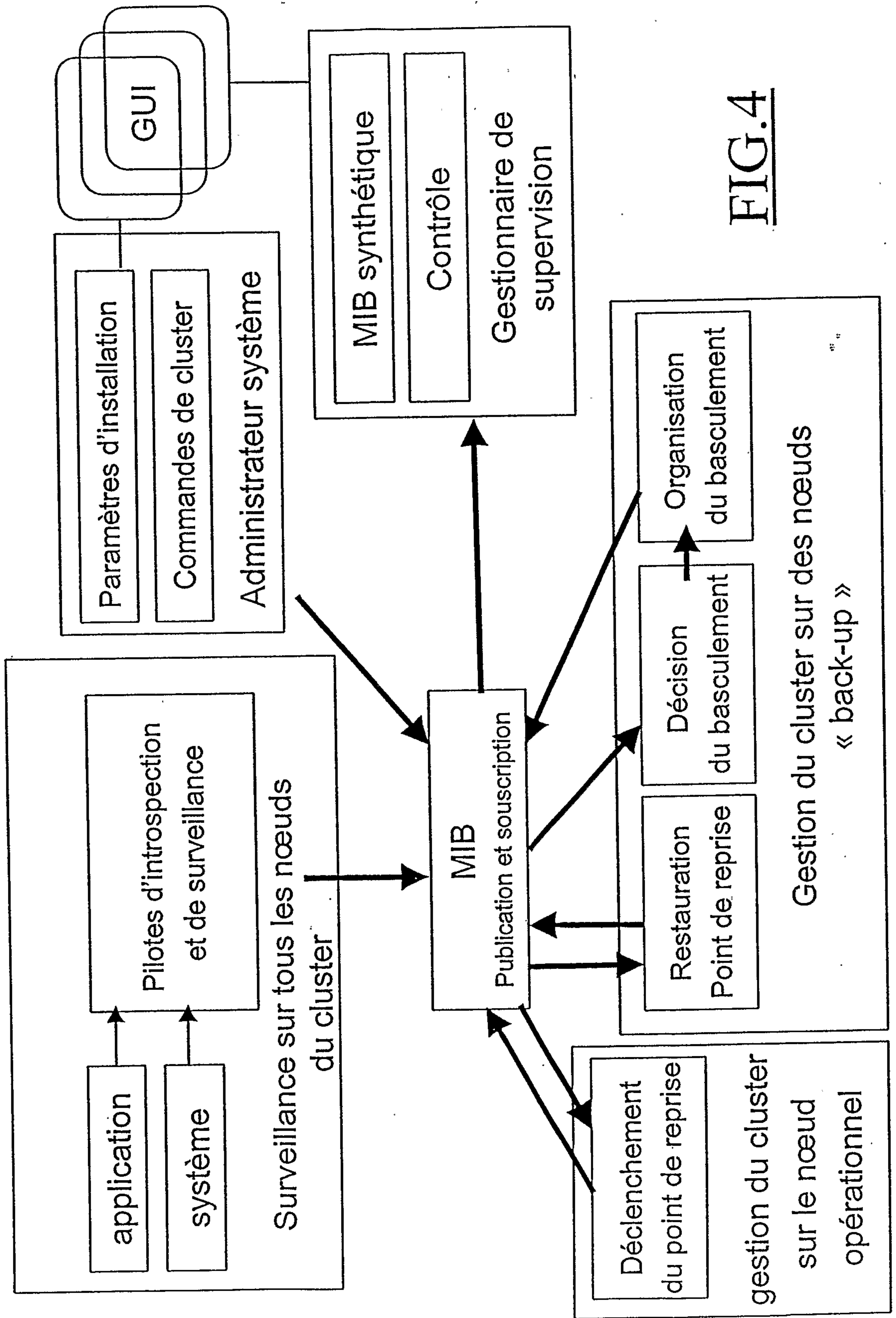


FIG.4

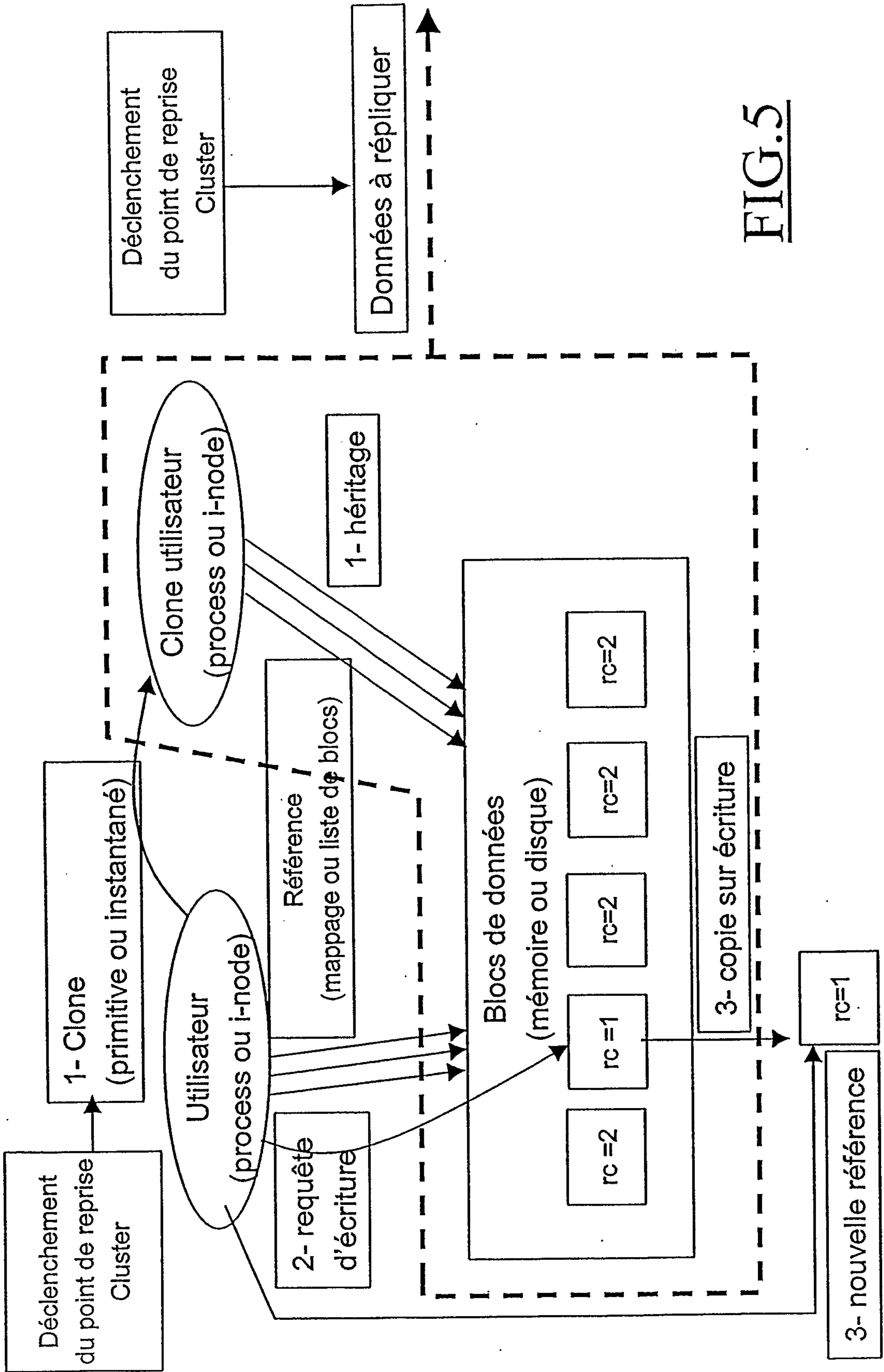


FIG.5

6/7

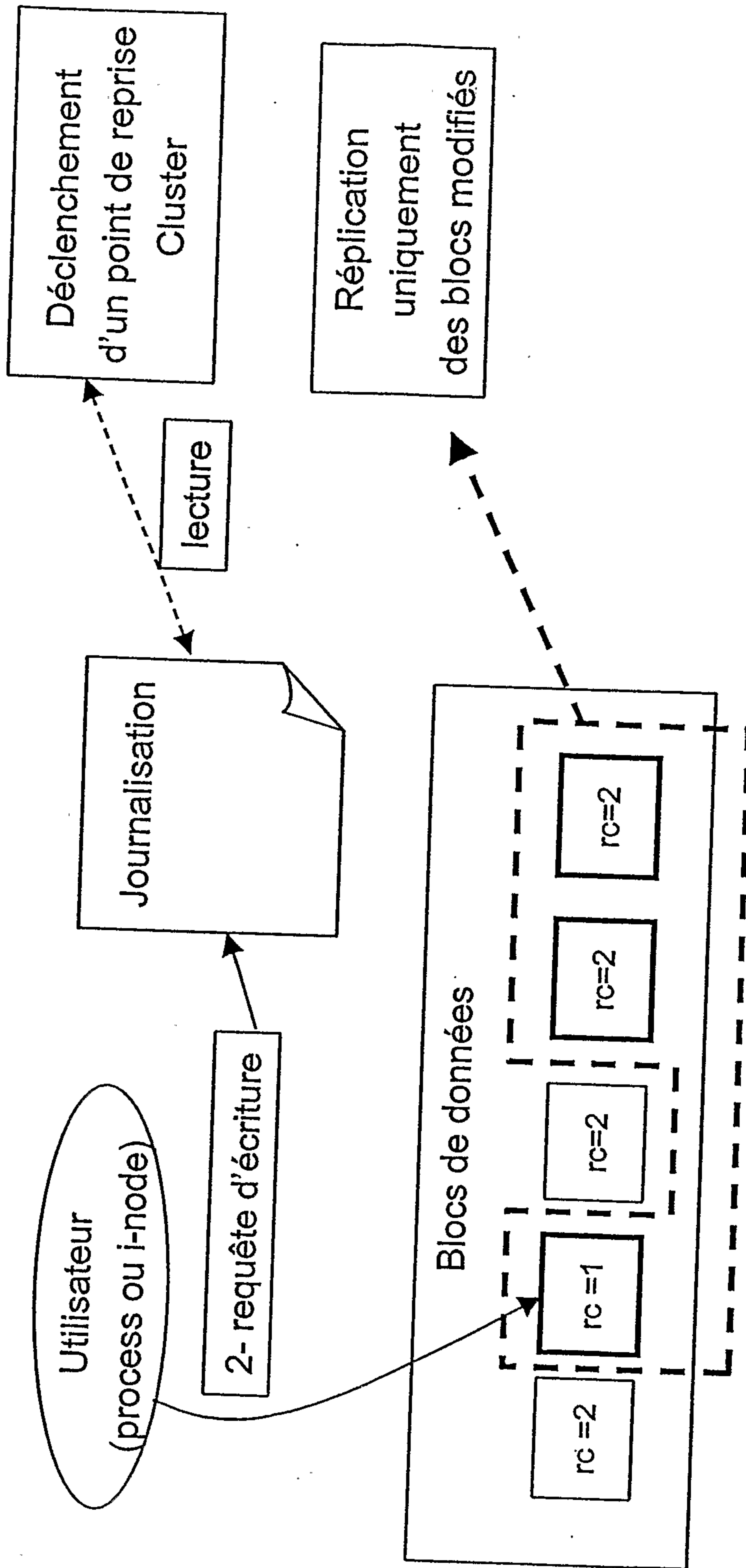


FIG.6

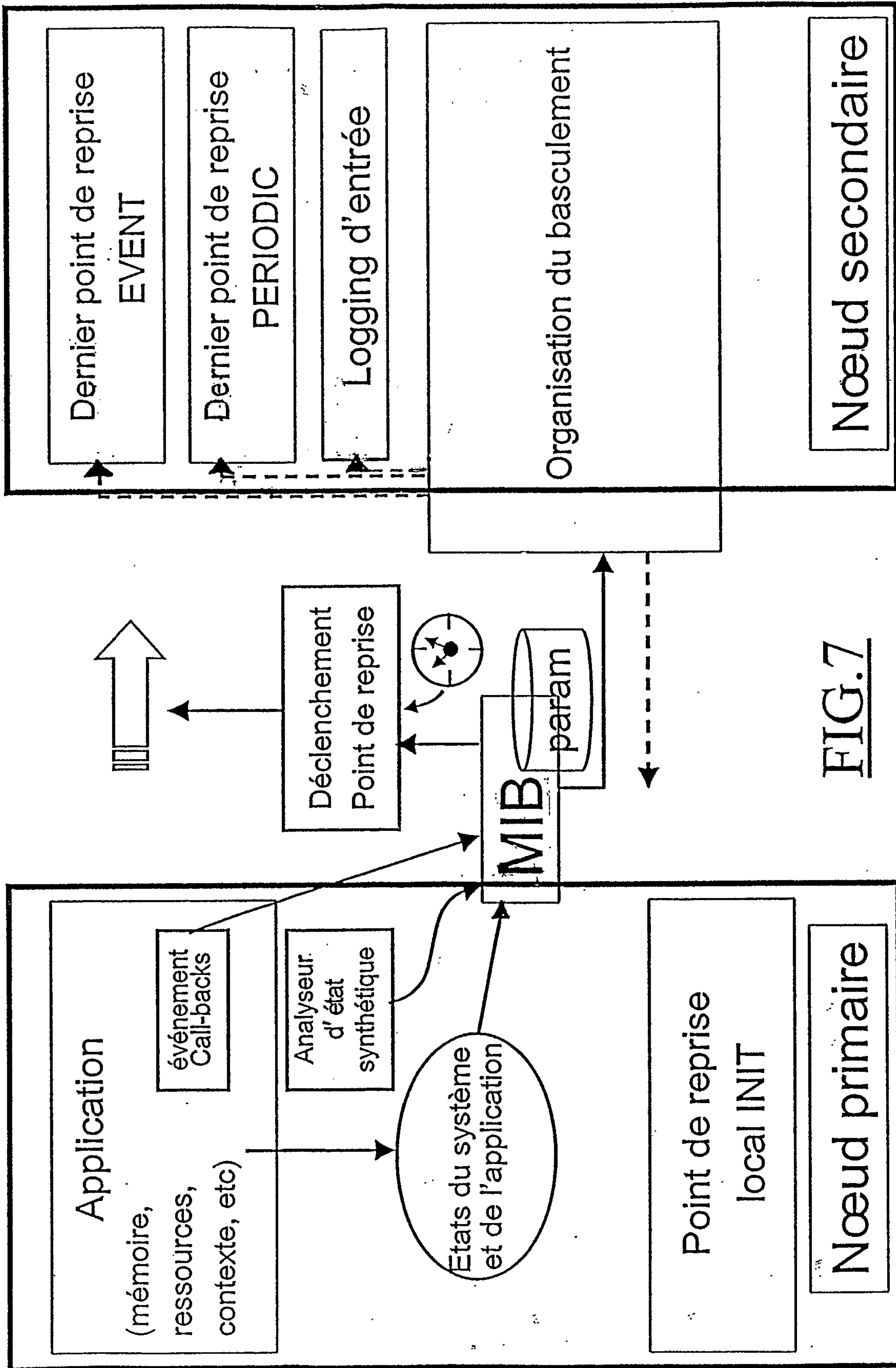


FIG. 7

