Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2493407 A1 2004/02/19

(21) 2 493 407

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(86) Date de dépôt PCT/PCT Filing Date: 2003/07/28
(87) Date publication PCT/PCT Publication Date: 2004/02/19
(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 (02/09855) FR

(51) Cl.Int.$^{7}$/Int.Cl.$^{7}$ G06F 11/20

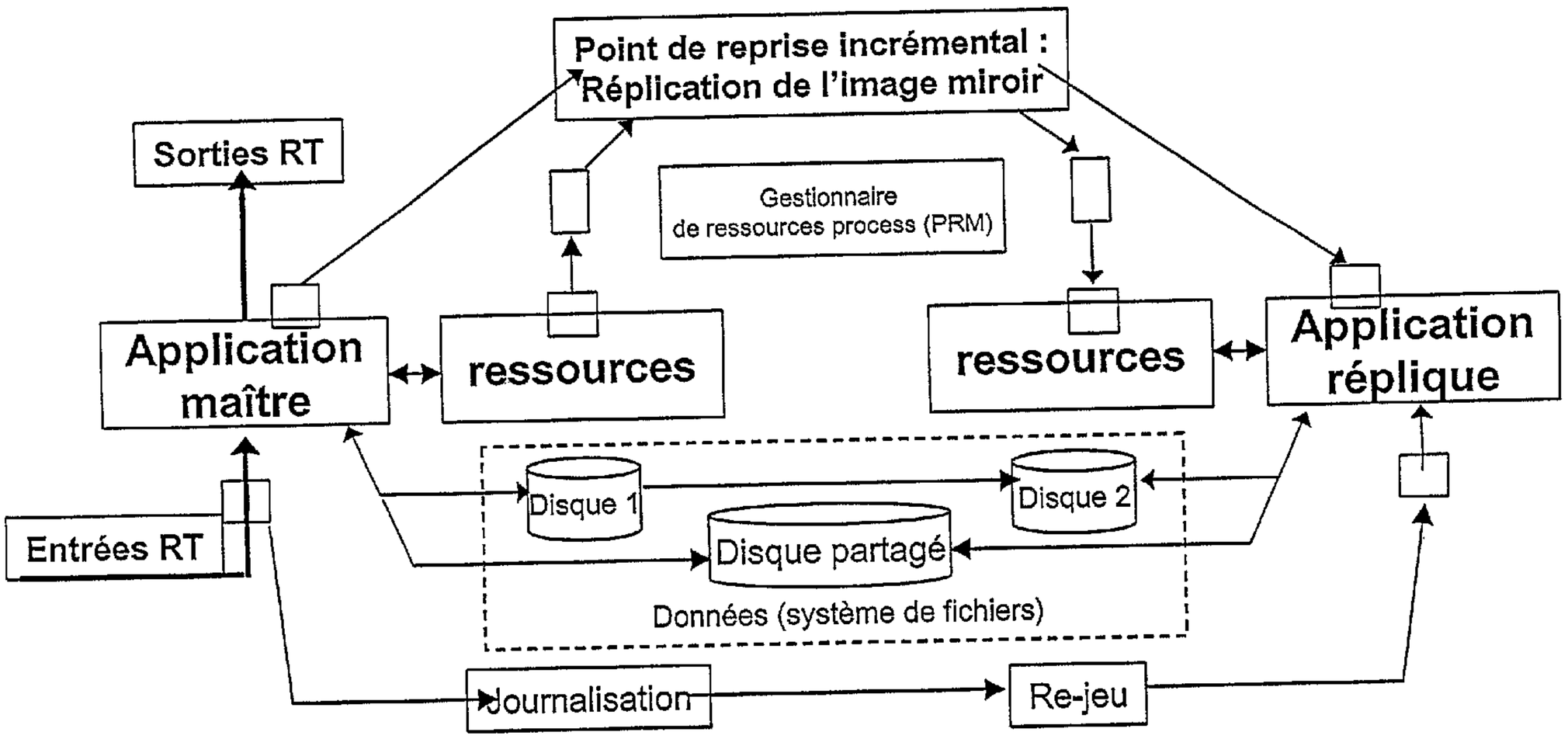
(71) Demandeur/Applicant:
MEIOSYS, FR

(72) Inventeurs/Inventors:
VERTES, MARC, FR;
DUFOUR, LAURENT, FR;
RICHARD, FRANCOIS, FR;
KURZ, GREGORY, FR

(74) Agent: FETHERSTONHAUGH & CO.

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

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




(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 n.oelig.ud primaire ou opérationnel et étant destinée à être répliquée sur au moins un autre ordinateur du cluster constituant un n.oelig.ud secondaire, comprenant une réplication 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 oeuvre.

(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 brevets[7] : 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 Croutte, 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).

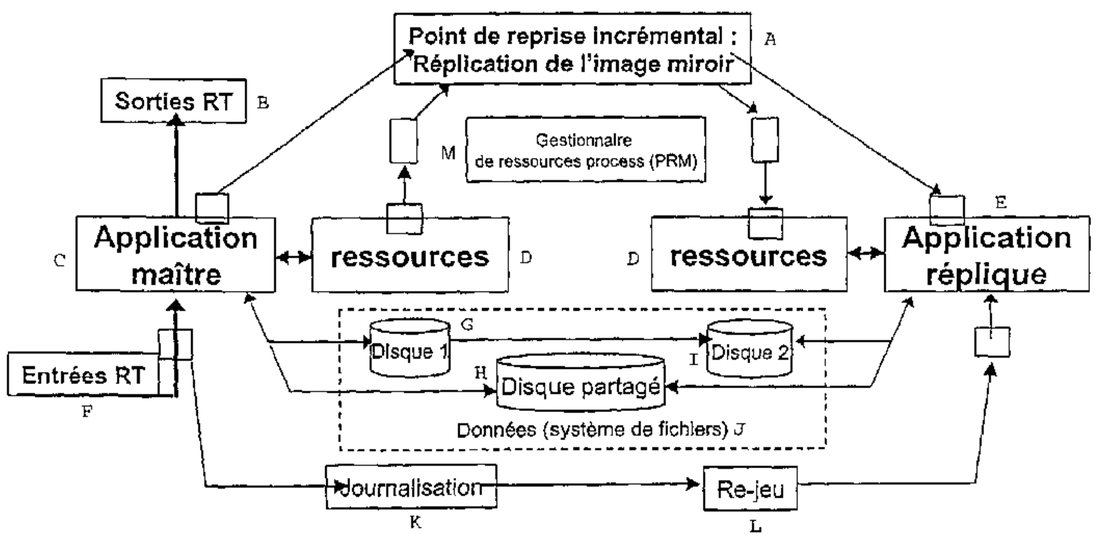
(74) Mandataires : **ALLANO, Sylvain** etc.; Pontet Allano & Associés S.E.L.A.R.L., 25, rue Jean-Rostand, Parc Club Orsay Université, F-91893 Orsay Cedex (FR).

(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 :** CONTINUITE DE FONCTIONNEMENT PAR REPLICATION D'UN LOCIGIEL 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.

**(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).

**Publiée :**

— *avec rapport de recherche internationale*

— *avec revendications modifiées*

**(88) Date de publication du rapport de recherche internationale:** 2 septembre 2004

**Date de publication des revendications modifiées:** 16 décembre 2004

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

La présente invention concerne un procédé pour répliquer une application logicielle dans une architecture multi-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 réplication selon l'invention, ainsi qu'un système multi-ordinateurs implémentant ce procédé de 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 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.

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.

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 fourni par l'application en cours d'exécution au moment de la défaillance.

En particulier, se pose le problème de la réplication d'une application logicielle au sein d'une architecture multi-ordinateurs, cette réplication devant assurer une 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 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 réplication des ressources associées à ladite application logicielle.

Cet objectif principal est atteint avec un tel procédé de réplication 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 dynamique des ressources et dépendances mises en œuvre.

On peut en outre avantageusement prévoir que ce procédé de réplication 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 nécessaire de répliquer.

Il est important de noter que dans le procédé de réplication 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 réplication 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 plusieurs nœuds secondaires.

Le procédé de réplication 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 secondaires,
- restauration sur le ou les nœuds secondaires.

Le procédé de réplication selon l'invention peut être avantageusement mis en oeuvre pour une optimisation automatique de ressources informatiques par partage de charge 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.

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 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), 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 :

- réplication 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.

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 réplication de l'application et aptes à relayer sans discontinuité cette application en cas de détection de défaillance ou d'événement affectant le nœud principal.

La réplication mise en œuvre dans le procédé de réplication selon l'invention est avantageusement de type holistique. On dispose ainsi d'un clonage de l'application au 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 (périodiquement ou sur évènements caractéristiques). Ces clones contiennent toutes les ressources et informations requises par l'application pour fournir son service.

Le procédé de réplication selon l'invention permet en outre de superviser l'état de toutes les ressources nécessaires au bon fonctionnement de l'application. Quand la dégradation rédhibitoire de l'une d'entre elles est détectée, le procédé de réplication 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 mis 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 réplication 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 réplication selon l'invention résout 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 réplication implémentée dans le procédé de réplication 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 résidait dans le problème de l'intrusivité. Le procédé de réplication 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 puissent être migrés.

Il est à noter que pour la mise en œuvre du procédé de réplication 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 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 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 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 réplication selon l'invention ;

-la figure 2 illustre schématiquement les principes de réplication de données mis en œuvre dans le procédé de réplication selon l'invention ;

-la figure 3 représente un exemple d'architecture logicielle mettant en œuvre le procédé de réplication 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 réplication selon l'invention ;

-la figure 5 illustre schématiquement le mécanisme de copie sur écriture (« copy on write ») mis en œuvre dans le procédé de réplication selon l'invention ;

-la figure 6 illustre schématiquement le mécanisme d'incrémentation pour réplication mis en œuvre dans le procédé de réplication selon l'invention ; et

-la figure 7 illustre schématiquement le mécanisme de commutation mis en œuvre dans le procédé de réplication selon l'invention.

On va d'abord décrire, en référence aux figures précitées, le fonctionnement du mécanisme de réplication holistique mis en œuvre dans le procédé de réplication 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.

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 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.

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 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 transmettrent 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 « 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 réplication selon l'invention à chaque instant des 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 réalisation de continuité de fonctionnement selon l'invention, consiste à répliquer les ressources sur un ou plusieurs nœuds secondaires. Ce mécanisme de réplication des ressources est réalisé en trois étapes :

- capture des ressources sur le nœud primaire,
- 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 ainsi que sa pile d'appel,
- des ressources systèmes (inter process communication, connexion réseau, etc.), et
- des données écrites sur disques.

Le mécanisme de réplication des ressources assure que 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 réplication selon l'invention garantit que l'application puisse continuer de vivre sur le secondaire sans perdre son contexte : 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'autre ressources matérielles.

Les ressources mises en œuvre par l'application sont diverses et variées (multi process, système d'exploitation, etc.). Elle vivent 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 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 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 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 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 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 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 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.

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 réplication 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 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 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
- 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 des informations sauvegardées.

DUMP:
RM_PRE_DUMP
RM_DUMP
RM_POST_DUMP
RM_ABORT_DUMP
RESTORE:
RM_PRE_RESTORE,
RM_RESTORE,
RM_POST_RESTORE,
RM_ABORT_RESTORE,
SWITCH:
RM_PRE_SWITCH,
RM_SWITCH,
RM_POST_SWITCH,
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 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 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 appliqués pour les identifiants de ressources système suivants :

- Processus
- PIPE
- FIFO
- IPC System V
    - Mémoires partagées
    - Sémaphores
    - Message queues
- Socket AF UNIX
- 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 noyau dynamiques assurant la non-préemptivité de requêtes qui leurs sont faites, sur des systèmes mono et multiprocesseurs, et avec une capacité à instrospecter les différentes routines permettant de manipuler ces identifiants.

A titre d'exemple non limitatif, une routine *getpid* est 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, ...).

Le procédé de réplication selon l'invention inclut aussi un module de réplication 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

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)

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.

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 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 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 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.

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.

On va maintenant décrire le mécanisme de synchronisation mis en œuvre dans le procédé de réplication 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 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 décomposée en 2 phases afin de limiter la taille du log de CFOR.

1. Création d'un instantané (snapshot) sur le nœud opérationnel OP
2. 1ère 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 2éme instantané (snapshot) sur le nœud opérationnel OP
5. 2éme 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 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 analysant dynamiquement l'organisation mémoire interne du processus et en isolant les différentes zones :

- texte
- données
- pile d'exécution

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 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 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.

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 transfert terminé, l'image mémoire maintenue par les mécanismes de « Copy On Write » est supprimée.

Le procédé de réplication 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 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.

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 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 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 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) 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
- 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 dans une architecture multi-ordinateurs (cluster), ladite application logicielle étant préalablement exécutée 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 réplication des ressources associées à ladite application logicielle, caractérisé en ce que les ressources répliquées incluent :

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

et en ce qu'il comprend en outre une mise à jour au fil de l'eau desdites ressources répliquées par un mécanisme d'introspection dynamique prévu pour fournir la structure de l'application à répliquer, et un graphe dynamique des ressources et dépendances mises en œuvre.

2. Procédé de réplication selon la revendication 1, caractérisé en ce qu'il comprend 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 nécessaire de répliquer.

3. Procédé de réplication selon l'une des revendications 1 ou 2, caractérisé en ce qu'il comprend en outre un mécanisme de génération de point de reprise (« checkpointing »), par lequel les ressources à répliquer sont répliquées sur un ou plusieurs nœuds secondaires.

4. Procédé de réplication selon la revendication 3, caractérisé en ce qu'il comprend trois étapes :

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

5. Procédé de réplication selon l'une quelconque des revendications précédentes et la revendication 3, caractérisé en ce qu'il comprend en outre un mécanisme d'optimisation du mécanisme de génération de point de reprise.

6. Procédé de réplication selon la revendication 5, caractérisé en ce que le mécanisme de « checkpointing » est incrémental.

7. Procédé de réplication selon l'une des revendications 5 ou 6, caractérisé en ce que le mécanisme de « checkpointing » est discriminant.

8. Procédé de réplication selon l'une des revendications 5 à 7, caractérisé en ce que le mécanisme de « checkpointing » 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).

9. 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 un nœud opérationnel (OP) sur lequel l'application est exécutée et un nœud dit de stand-by (SB).

10. 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 ou opérationnel, les autres ordinateurs dudit cluster étant appelés nœuds secondaires, ce procédé mettant en œuvre le procédé de réplication selon l'une quelconque des revendications précédentes,

caractérisé en ce qu'il comprend les étapes suivantes :

- réplication 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 opérationnel, basculement de service vers l'un au moins desdits clones.

11. Procédé de continuité de fonctionnement selon la revendication 10, caractérisé en ce que la réplication de l'application est de nature holistique.

12. Procédé de continuité de fonctionnement selon l'une revendication 10 ou 11, caractérisé en ce qu'il comprend en outre une mise à jour des clones de l'application.

13. Procédé de continuité de fonctionnement selon l'une des revendications 10 à 12 , caractérisé en ce qu'il comprend en outre une supervision de l'état de ressources nécessairement au fonctionnement de l'application.

14. Procédé de continuité de fonctionnement selon l'une des revendications 10 à 13, caractérisé en ce qu'il comprend en

outre, à la suite d'une détection d'une défaillance ou d'un événement affectant le nœud opérationnel, une étape pour élire, parmi des clones installés sur des nœuds secondaires, un clone pour être substitué à l'application initiale, le nœud sur lequel ledit clone élu est installé devenant le nouveau nœud opérationnel.

15. Procédé de continuité de fonctionnement selon l'une des revendications 10 à 14 , caractérisé en ce qu'il comprend en outre un enregistrement sur chaque clone de messages reçus par le nœud primaire ou opérationnel, ces messages étant réinjectés dans le clone élu nouvel opérationnel en cas de basculement.

16. Système multi-ordinateurs prévu pour exécuter sur au moins desdits ordinateurs au moins une application logicielle, implémentant le procédé pour réaliser une continuité de fonctionnement selon l'une quelconque des revendications 11 à 15.

17. Application du procédé de réplication selon l'une quelconque des revendications 1 à 9, pour une optimisation automatique de ressources informatiques par partage de charge par répartition dynamique de processus.

18. Application du procédé de réplication selon l'une quelconque des revendications 1 à 9, pour une maintenance non interruptive par relocation à la demande de processus au travers d'un réseau de ressources informatiques.

19. Application du procédé de réplication selon l'une quelconque des revendications 1 à 9, pour une préservation de contexte applicatif dans des applications mobiles.

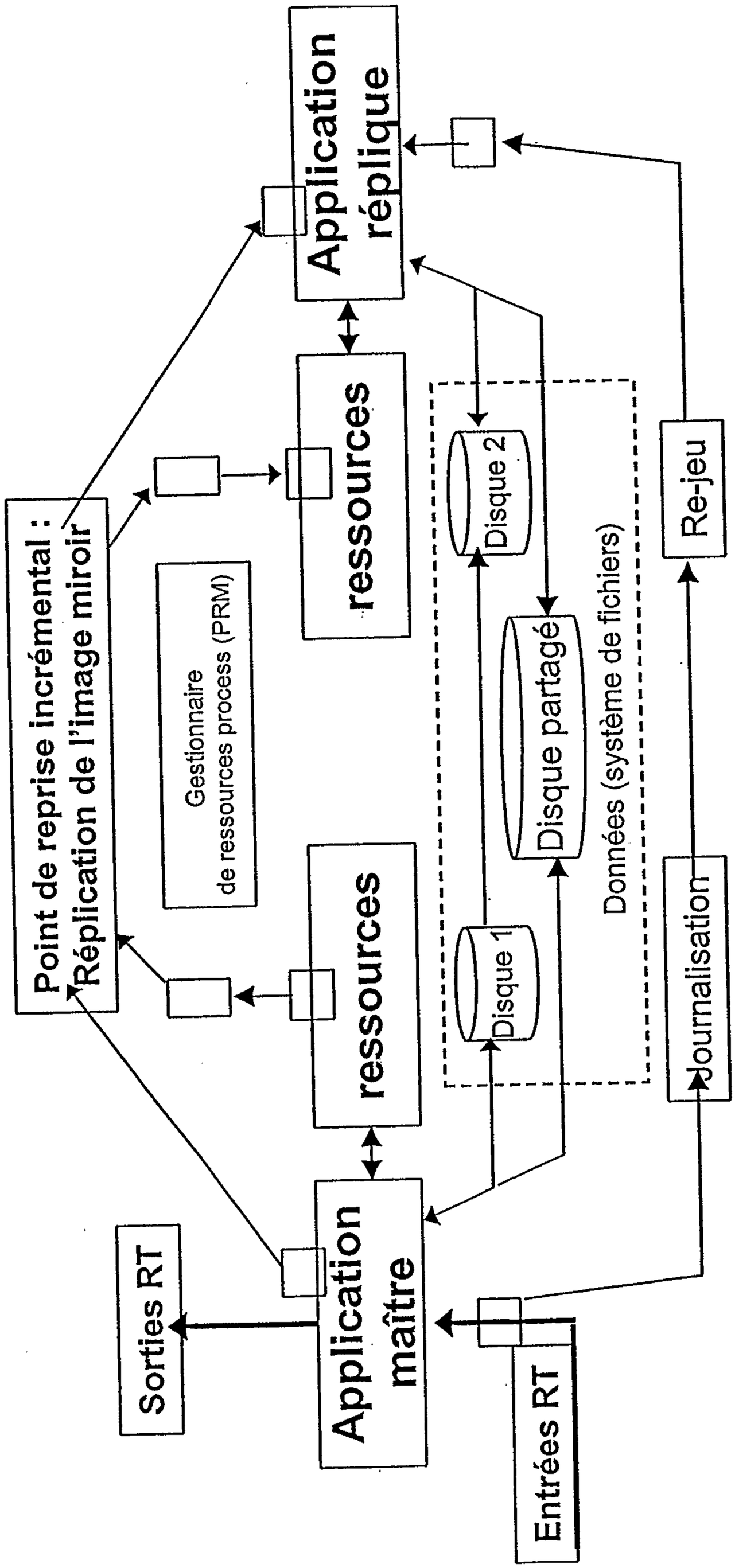

Point de reprise incrémental :
Réplication de l'image miroir
Gestionnaire de ressources process (PRM)
ressources
Application répliquée
Disque 2
Disque partagé
Re-jeu
Données (système de fichiers)
ressources
Disque 1
Journalisation
Sorties RT
Application maître
Entrées RT

FIG.1

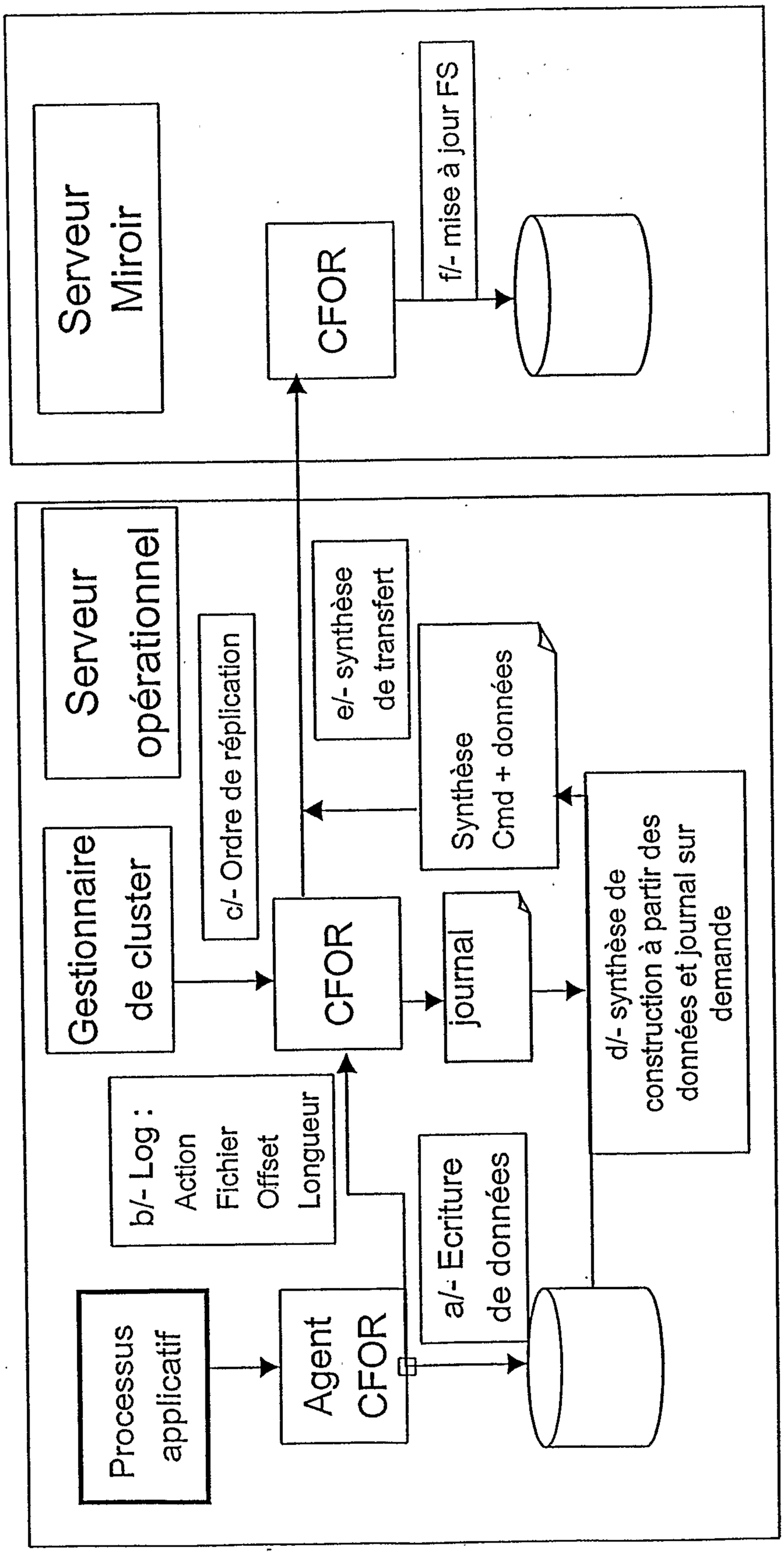

Serveur Miroir
CFOR
f/- mise à jour FS
Serveur opérationnel
e/- synthèse de transfert
c/- Ordre de réplication
Synthèse Cmd + données
Gestionnaire de cluster
journal
CFOR
d/- synthèse de construction à partir des données et journal sur demande
b/- Log : Action Fichier Offset Longueur
a/- Ecriture de données
Processus applicatif
Agent CFOR

FIG.2

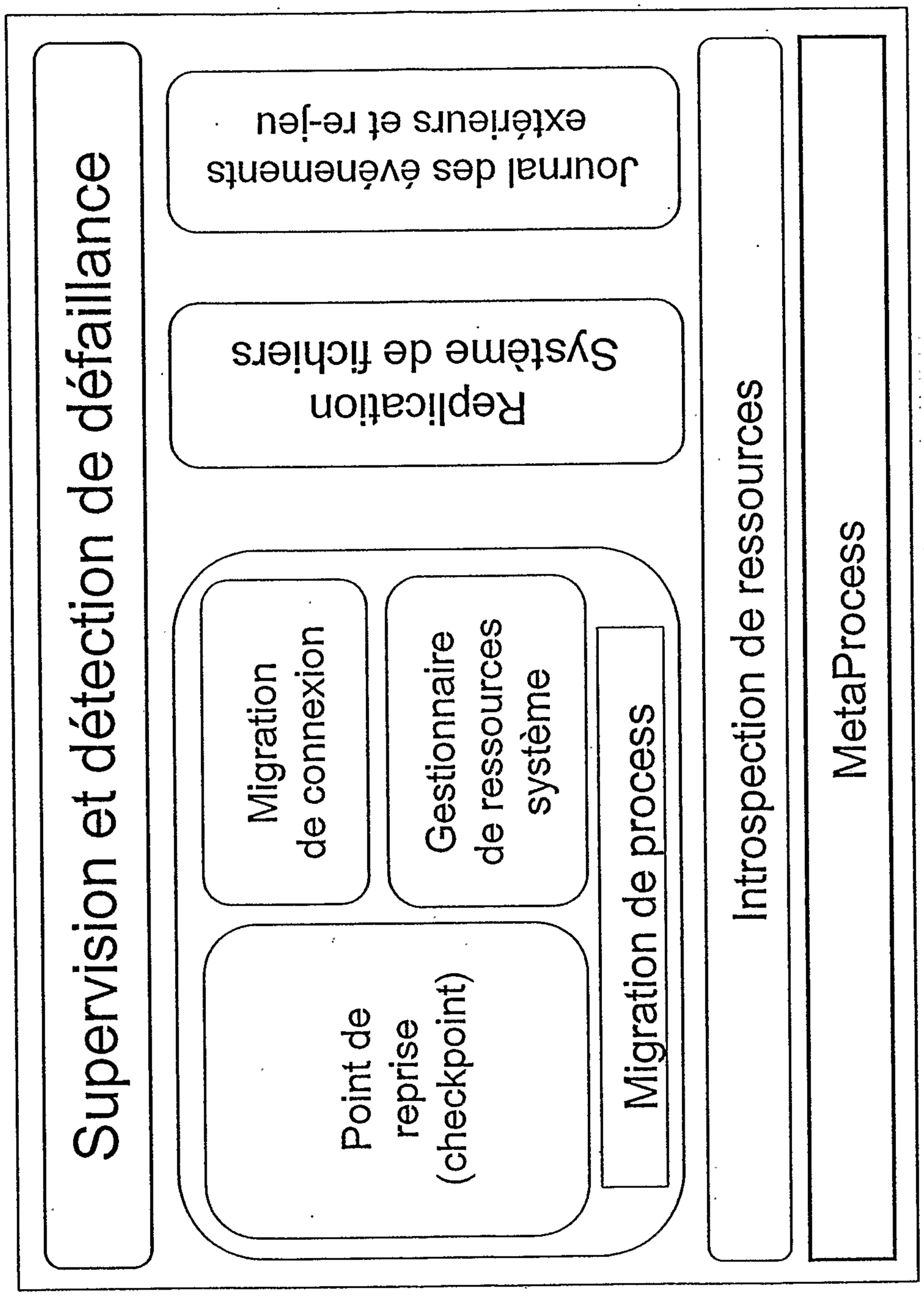

Supervision et détection de défaillance
Point de reprise (checkpoint)
Migration de connexion
Gestionnaire de ressources système
Migration de process
Introspection de ressources
MetaProcess
Journal des événements extérieurs et re-jeu
Réplication Système de fichiers

FIG.3

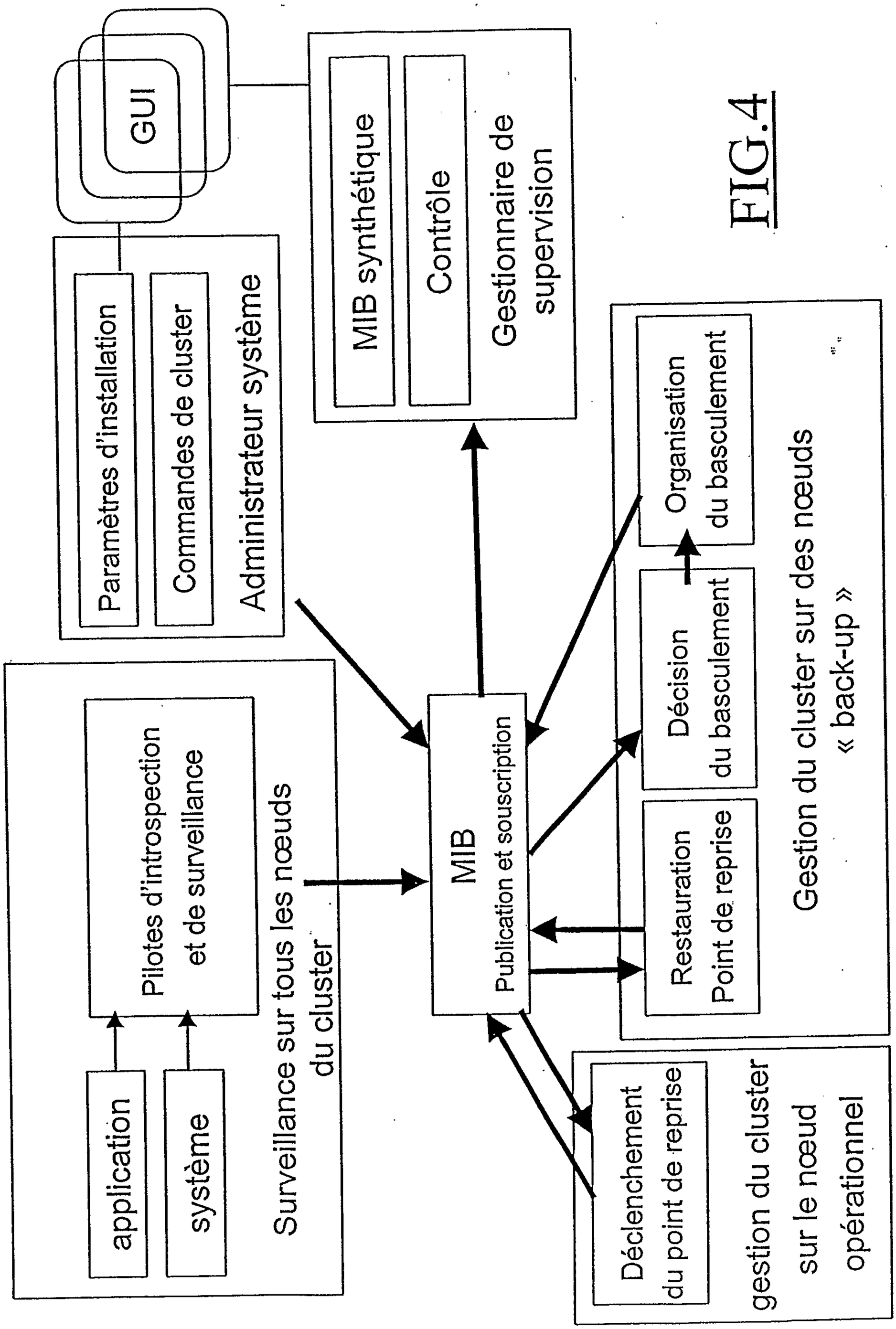

GUI
Administrateur système
Paramètres d'installation
Commandes de cluster
Gestionnaire de supervision
Contrôle
MIB synthétique
Gestion du cluster sur des nœuds « back-up »
Organisation du basculement
Décision du basculement
Restauration Point de reprise
MIB
Publication et souscription
Déclenchement du point de reprise
gestion du cluster sur le nœud opérationnel
Surveillance sur tous les nœuds du cluster
Pilotes d'introspection et de surveillance
application
système

FIG.4

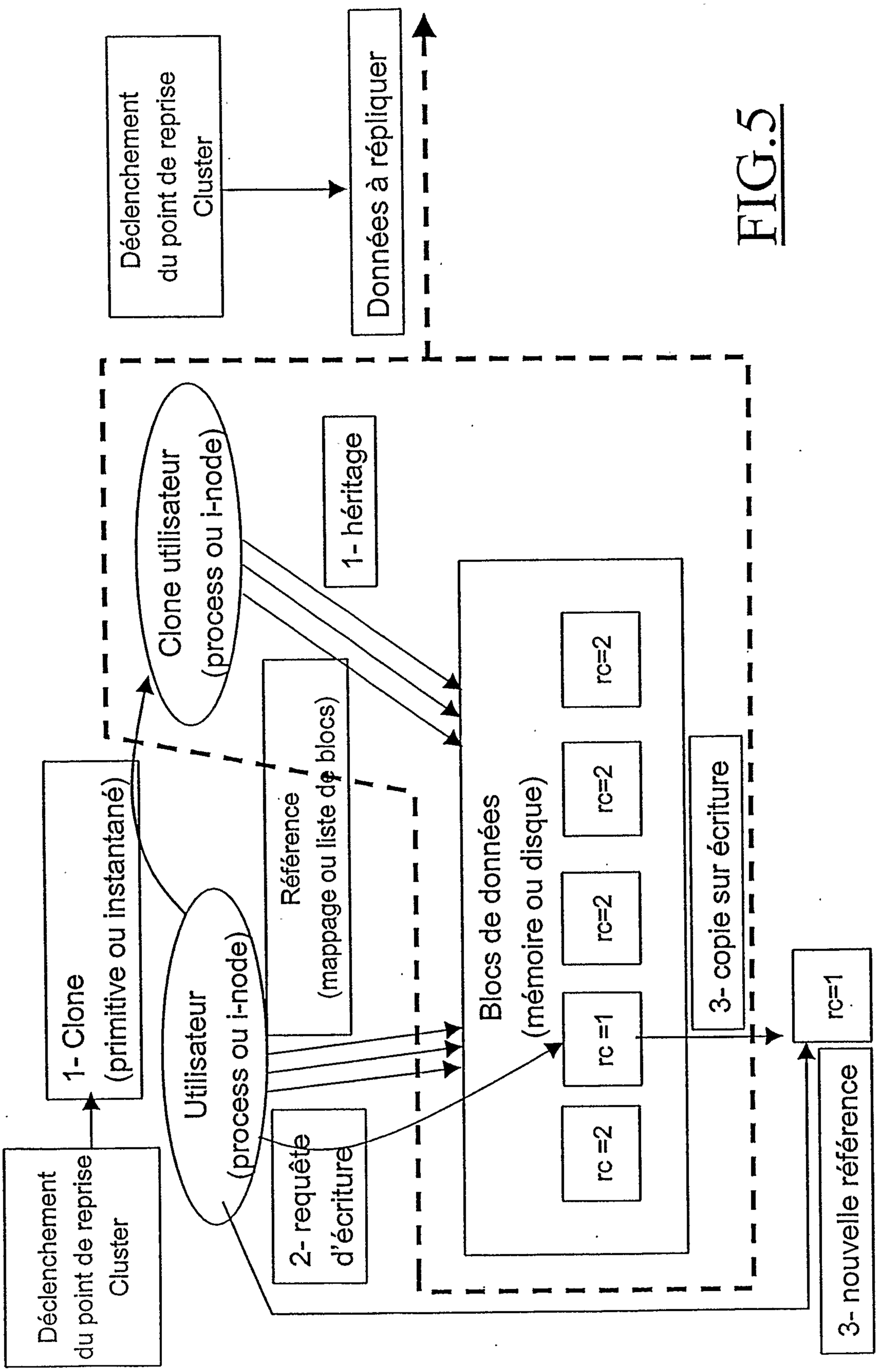

Déclenchement du point de reprise Cluster
Données à répliquer
1- héritage
Clone utilisateur (processus ou i-node)
1- Clone (primitive ou instantané)
Référence (mappage ou liste de blocs)
Utilisateur (processus ou i-node)
Blocs de données (mémoire ou disque)
rc=2
rc=2
rc=2
rc =1
rc =2
3- copie sur écriture
rc=1
2- requête d'écriture
3- nouvelle référence
Déclenchement du point de reprise Cluster

FIG.5

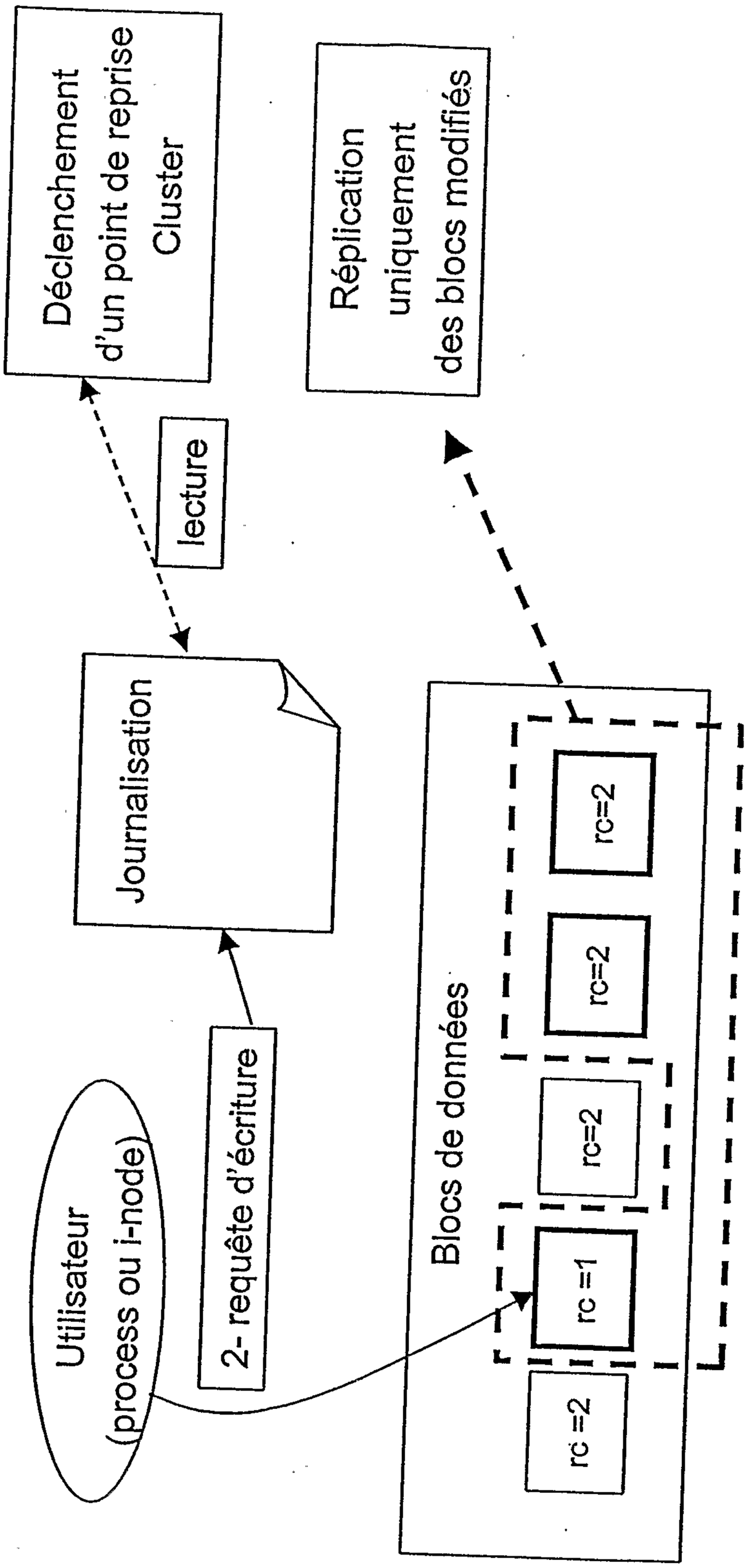

Déclenchement d'un point de reprise Cluster
Réplication uniquement des blocs modifiés
lecture
Journalisation
Utilisateur (process ou i-node)
2- requête d'écriture
Blocs de données
rc=2
rc=2
rc=2
rc =1
rc =2

FIG.6

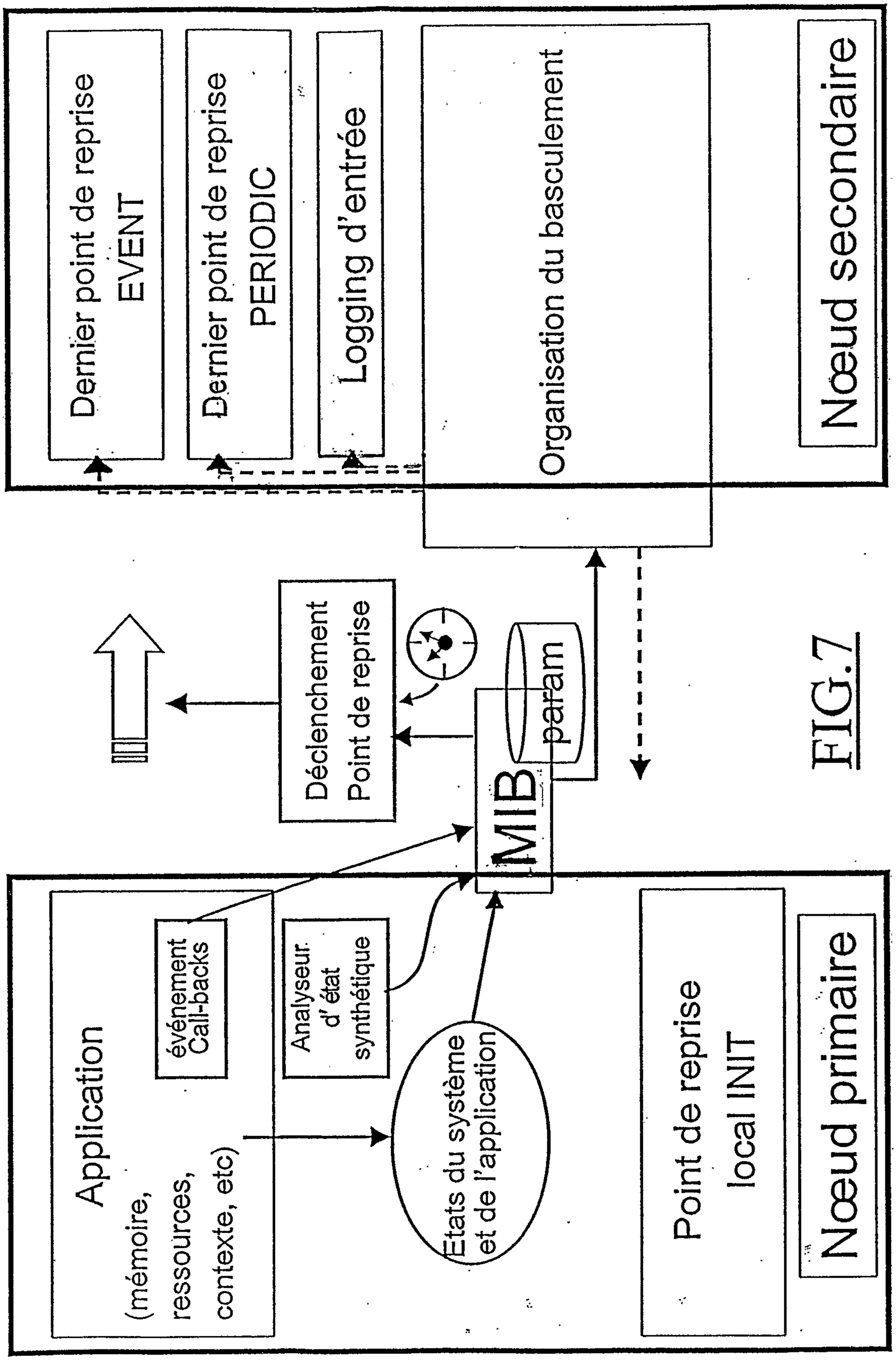
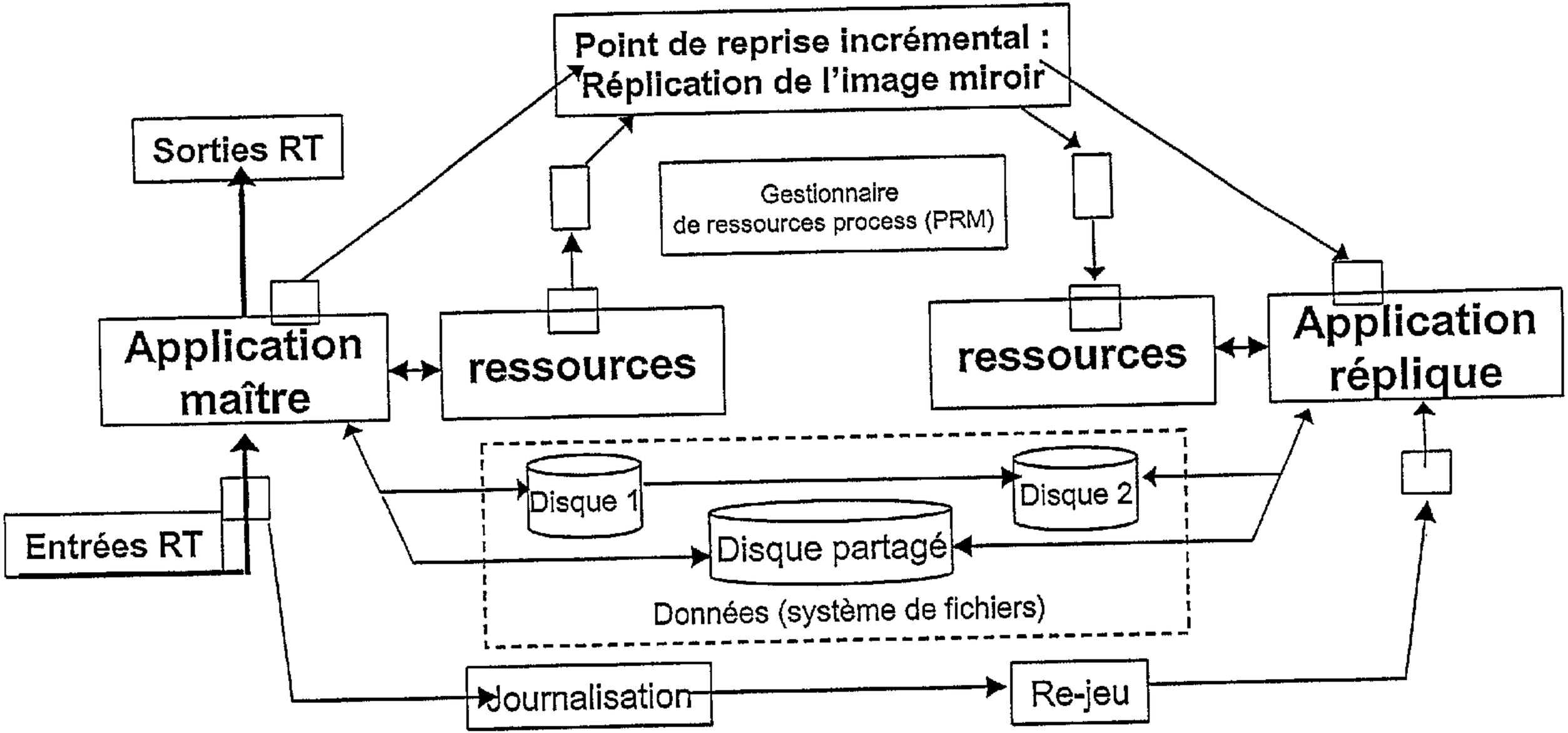

Nœud primaire
Point de reprise local INIT
Applications (mémoire, ressources, contexte, etc)
Etats du système et de l'application
événement Call-backs
Analyseur d' état synthétique
MIB
param
Déclenchement Point de reprise
Nœud secondaire
Organisation du basculement
Logging d'entrée
Dernier point de reprise PERIODIC
Dernier point de reprise EVENT

FIG.7


Point de reprise incrémental :
Réplication de l'image miroir
Sorties RT
Gestionnaire
de ressources process (PRM)
Application
maître
ressources
ressources
Application
réplique
Disque 1
Disque 2
Entrées RT
Disque partagé
Données (système de fichiers)
Journalisation
Re-jeu