

[Suite sur la page suivante]



AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, 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, MZ, NA, 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.

FR, GB, GR, HU, IE, IT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Publiée :

— sans rapport de recherche internationale, sera republiée dès réception de ce rapport

(84) **États désignés** (sauf indication contraire, pour tout titre de protection régionale disponible) : ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), eurasién (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), européen (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI,

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) **Abrége :** L'invention concerne un procédé de gestion de l'exécution d'au moins un programme (P) associé à une liste de données d'entrée (D) nécessaires à l'exécution de ce programme (P). Le programme est adapté pour une exécution distribuée sur plusieurs calculateurs (14_i) grâce à la distribution de ses données d'entrée, et comporte une étape (18) de réception d'une requête (R₁(P, D)) d'exécution du programme (P). Le procédé comporte en outre les étapes suivantes - On sélectionne (26) un calculateur (14_e) parmi une pluralité de calculateurs (14_i) selon un critère de disponibilité, a priori, à exécuter cette requête (R₁(P, D)); - On teste (31) si le calculateur sélectionné (14_e) a les ressources suffisantes pour exécuter la totalité de la requête (R₁(P, D)); - Si non, on sélectionne (30) une partie (D₁) des données d'entrée sur laquelle on estime que le calculateur sélectionné (14_e) a les ressources suffisantes pour exécuter le programme (P), et l'on génère (32, 34) à partir de la requête (R₁(P, D)), une première sous-requête (R_{s1}(P, D₁)) d'exécution du programme (P) sur cette partie (D₁) sélectionnée des données d'entrée et au moins une seconde sous-requête (R_{s2}(P, D₂)) d'exécution du programme (P) sur le reste (D₂) des données d'entrée.

-1-

Procédé de gestion de l'exécution d'au moins un programme sur plusieurs calculateurs

La présente invention concerne un procédé de gestion de l'exécution d'au moins un
5 programme sur plusieurs calculateurs.

Plus précisément l'invention concerne un procédé de gestion de l'exécution d'au moins un programme associé à une liste de données d'entrée nécessaires à l'exécution de ce programme, le programme étant adapté pour une exécution distribuée par un gestionnaire de traitement sur plusieurs calculateurs grâce à la distribution de ses
10 données d'entrée. Ce procédé comporte une étape de réception d'une requête d'exécution du programme.

On sait que la taille et la complexité des programmes d'ordinateurs augmentent et qu'il est donc nécessaire d'en optimiser le temps de calcul.

Une solution pour augmenter la vitesse d'exécution de ces calculs consiste à répartir
15 l'exécution d'un programme sur plusieurs calculateurs selon une méthode connue sous le nom de parallélisation. Chaque calculateur possède un ou plusieurs processeurs pour l'exécution du programme. Cette méthode suppose que les données d'entrées du programme sont décorrélées, de façon à pouvoir les traiter séparément. Ainsi, il est possible de distribuer l'exécution de ce programme sur plusieurs calculateurs, chacun
20 d'eux effectuant partiellement le calcul en ne tenant compte que d'une partie des données d'entrée. Ensuite les différents résultats donnés par chacun des calculateurs sont combinés pour former un résultat final.

Le problème d'une telle distribution de l'exécution d'un programme consiste en ce qu'elle ne tient en général pas compte d'informations sur les capacités de calcul
25 respectives des calculateurs. Ainsi, si l'un des calculateurs a une capacité de calcul inférieure aux autres, il risque d'être entièrement sollicité lorsqu'un calcul complexe est lancé, au point d'arriver à saturation et de retarder ainsi le résultat final de ce calcul, alors que d'autres calculateurs de capacité plus grande ne sont que partiellement sollicités.

L'invention a pour but de remédier à cet inconvénient en fournissant un procédé de
30 gestion de l'exécution d'un programme, capable de réguler la distribution de l'exécution du programme sur les différents calculateurs en profitant au mieux des capacités des calculateurs disponibles.

A cet effet, l'invention a pour objet un procédé de gestion de l'exécution d'au moins un programme associé à une liste de données d'entrée nécessaires à l'exécution de ce
35 programme, le programme étant adapté pour une exécution distribuée par un gestionnaire de traitement sur plusieurs calculateurs grâce à la distribution de ses données d'entrée,

-2-

comportant une étape de réception d'une requête d'exécution du programme, caractérisé en ce qu'il comporte en outre les étapes suivantes :

- On sélectionne un calculateur parmi une pluralité de calculateurs selon un critère de disponibilité, a priori, à exécuter cette requête ;
- 5 - On teste si le calculateur sélectionné a les ressources suffisantes pour exécuter la totalité de la requête ;
- Si non, on sélectionne une partie des données d'entrée sur laquelle on estime que le calculateur sélectionné a les ressources suffisantes pour exécuter le programme, et l'on génère à partir de la requête, une première
- 10 sous-requête d'exécution du programme sur cette partie sélectionnée des données d'entrée et au moins une seconde sous-requête d'exécution du programme sur le reste des données d'entrée.

Le procédé selon l'invention propose, d'une part, le choix d'un calculateur pour exécuter un calcul partiel contributif du résultat final, et d'autre part, l'extraction, pour ce

15 calculateur sélectionné, d'un sous-ensemble optimal des données d'entrée qui seront traitées par celui-ci afin qu'il ne soit pas mis en état de saturation. Ainsi, l'élection d'un calculateur, combinée à un dimensionnement a priori du calcul qui sera effectué par ce calculateur permet un équilibrage de charge assurant de ne jamais dépasser les limites d'aucun des calculateurs sollicités. Un tel procédé élimine donc tout risque de saturation

20 des calculateurs, quel que soit le nombre de requêtes à exécuter.

Un procédé de gestion de l'exécution d'un programme selon l'invention peut en outre comporter l'une ou plusieurs des caractéristiques suivantes :

- on exécute la première sous-requête sur le calculateur sélectionné ;
- le critère de disponibilité d'un calculateur à exécuter une requête est
- 25 directement lié à la comparaison du nombre de processus, issus du gestionnaire de traitement, en cours d'exécution sur ce calculateur au nombre de processus, issus du gestionnaire de traitement, exécutables simultanément sur ce calculateur ;
- le critère de disponibilité d'un calculateur à exécuter une requête est le
- 30 rapport du nombre de processus issus du gestionnaire de traitement en cours d'exécution sur ce calculateur sur le nombre de processus issus du gestionnaire de traitement exécutables simultanément sur ce calculateur, et le calculateur sélectionné est celui pour lequel ce rapport est le plus faible ;
- le critère de disponibilité d'un calculateur à exécuter une requête est pondéré
- 35 par l'estimation d'un coefficient de charge dépendant des ressources du

-3-

calculateur, notamment le nombre de processus en cours d'exécution sur le calculateur au moment du test ou la mémoire vive utilisée ;

- lors de l'étape de test du calculateur sélectionné, on compare le nombre de données d'entrée de la requête à un nombre maximal estimé de données d'entrées que le calculateur sélectionné est a priori capable de traiter ;
- le nombre maximal de données d'entrées que le calculateur sélectionné est a priori capable de traiter est estimé à partir des paramètres suivants : le nombre de données d'entrée de la requête, la somme sur tous les calculateurs du nombre de processus, issus du gestionnaire de traitement, exécutables simultanément sur chaque calculateur, un nombre de processus, issus du gestionnaire de traitement, exécutables simultanément sur le calculateur sélectionné, et un coefficient d'étalement défini par le gestionnaire de traitement et permettant de diviser en un nombre plus ou moins grand de sous-requêtes l'exécution de la requête ;
- le nombre maximal de données d'entrées que le calculateur sélectionné est a priori capable de traiter est estimé à partir de la formule suivante :

$$NE_e = E\left(NE_R \cdot \frac{NP_e}{\sum_{i=1}^n NP_i} \cdot CE + 1\right)$$

- où : E() est la fonction Partie Entière, NE_e est le nombre de données d'entrées que le calculateur sélectionné est a priori capable de traiter, NE_R est le nombre de données d'entrée de la requête, NP_e est le nombre de processus issus du gestionnaire de traitement exécutables simultanément

sur le calculateur sélectionné, $\sum_{i=1}^n NP_i$ est la somme sur tous les calculateurs du nombre de processus issus du gestionnaire de traitement exécutables simultanément sur chaque calculateur, et CE est le coefficient d'étalement.

- les valeurs du coefficient d'étalement et du nombre de processus, issus du gestionnaire de traitement, exécutables simultanément sur un calculateur sont modifiables par l'utilisateur, et/ou de façon dynamique ;
- la partie des données d'entrée sur laquelle est générée la première sous-requête correspond au nombre maximal de données d'entrées que le calculateur sélectionné est a priori capable de traiter ;
- on sélectionne la requête parmi une liste de requêtes classées par ordre de priorité ;
- on renvoie dans la liste des requêtes la seconde sous-requête ; et

-4-

- on affecte à chaque requête de la liste un coefficient de priorité, et les requêtes de coefficients de priorité égaux sont classées par ordre d'arrivée dans la liste.

L'invention sera mieux comprise à la lecture de la description qui va suivre, donnée
5 uniquement à titre d'exemple et faite en se référant aux dessins annexés dans lesquels :

- la figure 1 représente un schéma structurel d'un système de gestion pour la mise en œuvre d'un procédé selon l'invention ;
- la figure 2 illustre les étapes successives d'un procédé de gestion de l'exécution d'un programme selon l'invention ;
- 10 - la figure 3 représente un schéma fonctionnel de la combinaison des résultats partiels d'une requête exécutée selon le procédé de gestion de la figure 2 ;

On a représenté sur la figure 1 un système comportant un terminal utilisateur 10, un gestionnaire de traitement 12 et des calculateurs $14_1, \dots, 14_n$.

On entend ici par « gestionnaire de traitement 12 » un serveur de gestion de
15 traitement. Mais on pourrait aussi désigner par cette expression un logiciel installé sur un tel serveur.

Le terminal utilisateur 10 est par exemple un micro-ordinateur de type classique adapté pour émettre des requêtes d'exécution de programmes vers le gestionnaire de traitement 12.

20 Le gestionnaire de traitement 12 est lui aussi un micro-ordinateur de type classique, comportant des moyens de stockage, tels qu'une mémoire 15, pour le stockage au moins temporaire d'une liste d'attente 17 de requêtes en provenance d'un ou plusieurs terminaux utilisateurs tels que le terminal 10, et pour le stockage de résultats 50 au moins partiels provenant de l'exécution de requêtes issues de la liste 17.

25 Chaque calculateur 14_i est relié au gestionnaire de traitement 12 et est muni d'au moins un processeur. Ainsi, comme on le voit sur la figure 1, le calculateur 14_1 comporte un seul processeur 16_1 , le calculateur 14_2 comporte quatre processeurs $16_2, 16_3, 16_4, 16_5$ et le calculateur 14_n comporte deux processeurs 16_{m-1} et 16_m . Les calculateurs ne sont donc pas tous identiques : certains sont monoprocesseurs, tandis que d'autres sont
30 biprocesseurs ou multiprocesseurs.

Comme on le voit sur la figure 2, lors d'une première étape de saisie 18, un utilisateur rentre sur le terminal utilisateur 10 une requête $R_1(P, D)$ d'exécution d'un programme P associé à une liste D de données d'entrée.

Cette requête $R_1(P, D)$ est ensuite envoyée au gestionnaire de traitement 12 et l'on
35 passe à une étape 22 de classement cette requête R_1 dans la liste d'attente 17 de requêtes.

La liste d'attente 17 comporte par exemple d'autres requêtes déjà reçues par le gestionnaire de traitement 12, soit du même terminal utilisateur 10, soit d'un autre terminal utilisateur quelconque. Ces requêtes peuvent être par exemple des requêtes d'exécution du même programme P associé à une autre liste de données, ou bien des requêtes d'exécution d'autres programmes associés à d'autres données d'entrée.

Pour le bon fonctionnement du procédé selon l'invention, il est important de noter que les données d'entrée figurant dans la liste D de données d'entrée sont décorréliées. La décorrélation de ces données d'entrée signifie qu'il est possible d'effectuer des exécutions du programme P sur chacune d'entre elles de façon séparée. La liste D de données d'entrée peut comporter plusieurs centaines de milliers de données d'entrée, celles-ci étant simples ou multivaluées.

La décorrélation de ces données d'entrée est une condition sine qua non de la distribution de l'exécution du programme P sur plusieurs calculateurs 14_i .

Lors de l'étape 22 de classement, on classe les requêtes de la liste d'attente 17 par ordre de priorité d'exécution. La priorité des requêtes de la liste d'attente 17 peut être définie de différentes façons, telles que par un système de définition a priori des priorités qui consisterait à rentrer un numéro de priorité lors de l'étape 18 de saisie de chaque requête, ou encore, telles que par un principe de traitement du type FIFO consistant à traiter les requêtes par ordre d'arrivée.

Après l'étape 22 de classement de la liste d'attente 17, on passe à une étape 24 d'élection d'une requête prioritaire, correspondant à la première requête de la liste d'attente 17.

Pour simplifier les notations dans la suite de la description, on suppose que la requête prioritaire est la requête précédemment reçue par le gestionnaire de traitement 12 lors de l'étape 18, c'est-à-dire la requête $R_1(P, D)$. La requête prioritaire aurait bien sûr pu être l'une des autres requêtes contenues dans la liste 17.

Une fois que l'on connaît la requête prioritaire à exécuter, on passe à une étape 26 d'élection d'un calculateur parmi tous les calculateurs 14_i , pour traiter au moins une partie de cette requête prioritaire. Le calculateur élu 14_e correspond au calculateur le plus disponible parmi tous les calculateurs 14_i .

Pour choisir le calculateur élu 14_e lors de l'étape 26, le gestionnaire de traitement 12 calcule pour chacun des calculateurs 14_i la valeur np_i/NP_i où np_i représente le nombre de processus en cours d'exécution sur le calculateur 14_i et NP_i le nombre de processus exécutables simultanément sur ce calculateur 14_i . Un processus correspond à l'exécution d'un programme par un calculateur sur des données d'entrées associée à ce programme.

-6-

Le calculateur élu 14_e est le calculateur 14_i pour lequel la valeur np/NP_i est la plus petite.

Selon un autre mode de réalisation, il est possible de choisir le calculateur le plus disponible selon d'autres critères tels que l'estimation d'un coefficient de charge, calculé dynamiquement sur chaque calculateur en tenant compte des ressources de chaque calculateur telles que le nombre de processus actifs sur le calculateur en cours d'exécution au moment du test ou la mémoire vive utilisée. Dans ce cas, l'étape 26 d'élection du calculateur consiste à toujours choisir le calculateur le moins chargé.

Une fois que le calculateur élu 14_e est déterminé, on passe à des étapes destinées à déterminer la façon de distribuer l'exécution de la requête prioritaire $R_1(P, D)$.

Ainsi, lors d'une étape 27, on compare les deux paramètres suivants :

- NE_{min} , représentant le nombre minimal de données d'entrées devant être traitées sur tout calculateur 14_i lors de l'exécution d'un programme, ce paramètre permettant de ne pas distribuer la requête prioritaire sur plus d'un calculateur si celle-ci est jugée petite a priori (c'est à dire acceptable en totalité par le calculateur le plus petit) ;
- NE_R correspondant au nombre total de données d'entrée que la requête prioritaire $R_1(P, D)$ doit traiter, c'est à dire au nombre de données d'entrées figurant dans la liste D.

Si $NE_R \leq NE_{min}$, c'est-à-dire si la requête prioritaire $R_1(P, D)$ doit traiter un nombre de données d'entrée plus faible que le nombre minimal de données d'entrées devant être traitées sur tout calculateur 14_i , cela signifie que le calculateur élu 14_e peut traiter l'intégralité des données d'entrée de la requête prioritaire $R_1(P, D)$. On passe alors à une étape 28 d'exécution de la requête $R_1(P, D)$ par le calculateur 14_e et le procédé est simultanément reporté à l'étape 24 d'élection d'une autre requête prioritaire à traiter. L'étape 28 d'exécution de la requête prioritaire $R_1(P, D)$ est suivie éventuellement d'une étape 29 de traitement de résultats, comportant par exemple le stockage, le formatage et l'affichage des résultats sur l'interface du terminal utilisateur 10.

Si $NE_R > NE_{min}$, on passe alors à une étape 30 de calcul du nombre NE_e maximal de données d'entrée que le calculateur élu 14_e est a priori capable de traiter.

Pour estimer NE_e , le gestionnaire de traitement 12 prend en compte différents paramètres :

- $\sum_{i=1}^n NP_i$ représentant la somme, sur tous les calculateurs 14_i , du nombre de processus, issus du gestionnaire de traitement 12, exécutables simultanément sur chaque calculateur 14_i ;

-7-

- NP_e correspondant au nombre de processus, issus du gestionnaire de traitement, exécutables simultanément sur le calculateur élu 14_e , et dont la valeur peut être modifiée par l'utilisateur et/ou de façon dynamique ; et
- CE représentant un coefficient d'étalement défini par le gestionnaire de traitement 12 et dont la valeur peut être modifiée par l'utilisateur, et/ou de façon dynamique, ce coefficient d'étalement permettant, pour l'ensemble des calculateurs, de regrouper en un nombre plus au moins grand de sous-requêtes l'exécution de la requête prioritaire $R_1(P, D)$.

Le coefficient d'étalement CE est tel que :

$$0 \leq CE \leq \frac{\sum_{i=1}^n NP_i}{\min(NP_i)}.$$

A partir de ces paramètres, le gestionnaire estime NE_e grâce à la formule :

$$NE_e = E\left(NE_R \cdot \frac{NP_e}{\sum_{i=1}^n NP_i} \cdot CE + 1\right)$$

Où $E()$ représente la fonction « Partie Entière ».

On passe ensuite à une étape de test 31 consistant à comparer le nombre NE_e calculé lors de l'étape 30 au nombre total NE_R de données d'entrée que la requête prioritaire $R_1(P, D)$ doit traiter. Cette étape permet de déterminer si le calculateur élu 14_e est apte à exécuter la requête prioritaire $R_1(P, D)$ dans sa totalité.

Si $NE_R \leq NE_e$, c'est-à-dire si les ressources disponibles du calculateur élu 14_e sont suffisantes pour exécuter la requête prioritaire $R_1(P, D)$ dans sa totalité, on passe alors à l'étape 28 d'exécution complète de cette requête prioritaire et le procédé est simultanément reporté à l'étape 24 d'élection d'une autre requête prioritaire à traiter.

Si $NE_R > NE_e$, c'est-à-dire si le calculateur élu 14_e n'a pas les ressources suffisantes pour exécuter la requête prioritaire $R_1(P, D)$ dans sa totalité, on passe alors à deux étapes 32 et 34 de génération de sous-requêtes à partir de la requête prioritaire $R_1(P, D)$.

En effet, la liste D des données d'entrée associée à la requête prioritaire $R_1(P, D)$ ne comportant que des données d'entrée décorréliées, il est possible de diviser la requête en plusieurs sous-requêtes portant sur un sous-ensemble de l'ensemble de données d'entrée D.

L'étape 32 consiste à générer une première sous-requête $R_{s1}(P, D_1)$ que l'un des processeurs du calculateur élu 14_e est capable de traiter. Cette première sous-requête $R_{s1}(P, D_1)$ est telle que la liste D_1 comporte NE_e données d'entrée, NE_e ayant été estimé au cours de l'étape 30.

L'étape 34 consiste à générer une seconde sous-requête $R_{s2}(P, D_2)$, destinée à être exécutée avec les données d'entrée qui ne sont pas comprises dans la liste D_1 de données d'entrées sur laquelle sera exécutée la sous-requête $R_{s1}(P, D_1)$. D_2 est le complément de D_1 dans D et comporte $NE_R - NE_e$ données d'entrée.

5 Après l'étape 34, la seconde sous-requête $R_{s2}(P, D_2)$ générée est envoyée dans la liste d'attente 17 des requêtes. Le procédé est alors reporté à l'étape de classement 22 des requêtes de la liste d'attente 17.

Après l'étape 32, la première sous-requête $R_{s1}(P, D_1)$ générée est envoyée au calculateur élu 14_e qui l'exécute lors d'une étape d'exécution 40 et le procédé est en
10 même temps reporté à l'étape 24 d'élection d'une autre requête prioritaire à traiter.

Ensuite, lors d'une étape 42, le calculateur 14_e envoie les résultats issus de l'étape 40 au gestionnaire de traitement 12.

Cette étape 42 d'envoi des résultats est suivie par une étape 44 de stockage par le gestionnaire de traitement 12 des résultats de l'exécution de la première sous-requête
15 $R_{s1}(P, D_1)$ dans la mémoire 15.

Comme on le voit sur la figure 3, l'exécution d'une requête telle que la requête $R_1(P, D)$ saisie lors de l'étape 18, peut aboutir, après l'application d'un procédé tel que celui décrit sur la figure 2, à plusieurs étapes 44, 46₁, 46₂, ..., 46_q de stockage de résultats intermédiaires. Les résultats stockés au cours des étapes 44, 46₁, 46₂, ..., 46_q sont les
20 résultats de l'exécution de toutes les sous-requêtes générées à partir de la requête $R_1(P, D)$ afin de distribuer l'exécution de cette requête $R_1(P, D)$ sur plusieurs calculateurs 14_i .

Après les étapes 44, 46₁, 46₂, ..., 46_q de stockage des résultats par le gestionnaire de distribution 12, on passe à une étape 48 de combinaison de tous ces résultats en un résultat final 50 de la requête $R_1(P, D)$.

25 Le résultat final 50 de la requête $R_1(P, D)$ est ensuite envoyé au terminal utilisateur 10 qui traite ces résultats 50 sur son interface lors d'une étape 52 de traitement de résultats comportant notamment le formatage et l'affichage des résultats.

Selon un autre mode de réalisation de l'invention, le système de gestion peut exécuter plusieurs programmes $P_1, P_2, \dots, P_z$ transmis sous forme de requêtes par
30 plusieurs terminaux utilisateurs.

De façon optionnelle, il est possible de doter le système d'une fonction d'information de l'avancement de l'exécution des requêtes. Un système de messagerie peut rendre compte à chaque client utilisateur, dans le cas où le système comporte plusieurs terminaux utilisateurs 10, de l'état d'avancement de ses requêtes ou de manière globale,
35 de l'état d'avancement de toutes les requêtes à un client administrateur.

Enfin, il est possible d'associer d'autres fonctions classiques à ce système de gestion telles qu'un démarrage de l'exécution des requêtes en différé, une gestion de pannes, des mécanismes de synchronisation de requêtes, ou encore une fonction d'arrêt automatique permettant d'arrêter une requête quel que soit son statut, en cours d'exécution ou en file d'attente.

5

Il apparaît clairement qu'un procédé selon l'invention permet de distribuer l'exécution du programme P sur plusieurs calculateurs 14_i, les étapes 26, 27, 30 et 31 de calcul du mode de distribution permettant de distribuer ces exécutions de façon optimale afin de réduire le temps global d'exécution.

10

REVENDICATIONS

1. Procédé de gestion de l'exécution d'au moins un programme (P) associé à une liste de données d'entrée (D) nécessaires à l'exécution de ce programme (P), le programme étant adapté pour une exécution distribuée par un gestionnaire de traitement (12) sur plusieurs calculateurs (14_i) grâce à la distribution de ses données d'entrée, comportant une étape (18) de réception d'une requête (R₁(P, D)) d'exécution du programme (P), caractérisé en ce qu'il comporte en outre les étapes suivantes :
- On sélectionne (26) un calculateur (14_e) parmi une pluralité de calculateurs (14_i) selon un critère de disponibilité, a priori, à exécuter cette requête (R₁(P, D)) ;
 - On teste (31) si le calculateur sélectionné (14_e) a les ressources suffisantes pour exécuter la totalité de la requête (R₁(P, D)) ;
 - Si non, on sélectionne (30) une partie (D₁) des données d'entrée sur laquelle on estime que le calculateur sélectionné (14_e) a les ressources suffisantes pour exécuter le programme (P), et l'on génère (32, 34) à partir de la requête (R₁(P, D)), une première sous-requête (R_{s1}(P, D₁)) d'exécution du programme (P) sur cette partie (D₁) sélectionnée des données d'entrée et au moins une seconde sous-requête (R_{s2}(P, D₂)) d'exécution du programme (P) sur le reste (D₂) des données d'entrée.
2. Procédé de gestion de l'exécution d'au moins un programme (P) selon la revendication 1, caractérisé en ce que l'on exécute la première sous-requête (R_{s1}(P, D₁)) sur le calculateur sélectionné (14_e).
3. Procédé de gestion de l'exécution d'au moins un programme (P) selon la revendication 1 ou 2, caractérisé en ce que le critère de disponibilité d'un calculateur (14_i) à exécuter une requête (R₁(P, D)) est directement lié à la comparaison du nombre (np_i) de processus, issus du gestionnaire de traitement (12), en cours d'exécution sur ce calculateur (14_i) au nombre (NP_i) de processus, issus du gestionnaire de traitement (12), exécutables simultanément sur ce calculateur (14_i).
4. Procédé de gestion de l'exécution d'au moins un programme (P) selon la revendication 3, caractérisé en ce que le critère de disponibilité d'un calculateur (14_i) à exécuter une requête (R₁(P, D)) est le rapport du nombre (np_i) de processus, issus du gestionnaire de traitement (12), en cours d'exécution sur ce calculateur (14_i) sur le nombre (NP_i) de processus, issus du gestionnaire de traitement (12), exécutables

simultanément sur ce calculateur (14_i), et en ce que le calculateur sélectionné (14_e) est celui pour lequel ce rapport est le plus faible.

5. Procédé de gestion de l'exécution d'au moins un programme (P) selon la revendication 3 ou 4, caractérisé en ce que le critère de disponibilité d'un calculateur (14_i) à exécuter une requête (R₁(P, D)) est pondéré par l'estimation d'un coefficient de charge dépendant des ressources du calculateur (14_i), notamment le nombre de processus en cours d'exécution sur le calculateur au moment du test ou la mémoire vive utilisée.

6. Procédé de gestion de l'exécution d'au moins un programme (P) selon l'une quelconque des revendications 1 à 5, caractérisé en ce que, lors de l'étape de test (26) du calculateur sélectionné (14_e), on compare le nombre (NE_R) de données d'entrée de la requête (R₁(P, D)) à un nombre (NE_e) maximal estimé de données d'entrées que le calculateur sélectionné (14_e) est a priori capable de traiter.

7. Procédé de gestion de l'exécution d'au moins un programme (P) selon la revendication 6, caractérisé en ce que le nombre (NE_e) maximal de données d'entrées que le calculateur (14_e) sélectionné est a priori capable de traiter est estimé à partir des paramètres suivants :

- le nombre (NE_R) de données d'entrée de la requête (R₁(P, D)) ;
- la somme ($\sum_{i=1}^n NP_i$) sur tous les calculateurs (14_i) du nombre (NP_i) de processus, issus du gestionnaire de traitement (12), exécutables simultanément sur chaque calculateur (14_i) ;
- un nombre (NP_e) de processus, issus du gestionnaire de traitement (12), exécutables simultanément sur le calculateur sélectionné (14_e) ; et
- un coefficient d'étalement (CE) défini par le gestionnaire de traitement (12) et permettant de diviser en un nombre plus ou moins grand de sous-requêtes l'exécution de la requête (R₁(P, D)).

8. Procédé de gestion de l'exécution d'au moins un programme (P) selon la revendication 7, caractérisé en ce que le nombre (NE_e) maximal de données d'entrées que le calculateur (14_e) sélectionné est a priori capable de traiter est estimé à partir de la formule suivante :

$$NE_e = E\left(NE_R \cdot \frac{NP_e}{\sum_{i=1}^n NP_i} \cdot CE + 1\right), \text{ où } E() \text{ représente la fonction Partie Entière.}$$

9. Procédé de gestion de l'exécution d'au moins un programme (P) selon la revendication 8, caractérisé en ce que les valeurs du coefficient d'étalement (CE) et du nombre (NP_e) de processus, issus du gestionnaire de traitement (12), exécutables

-12-

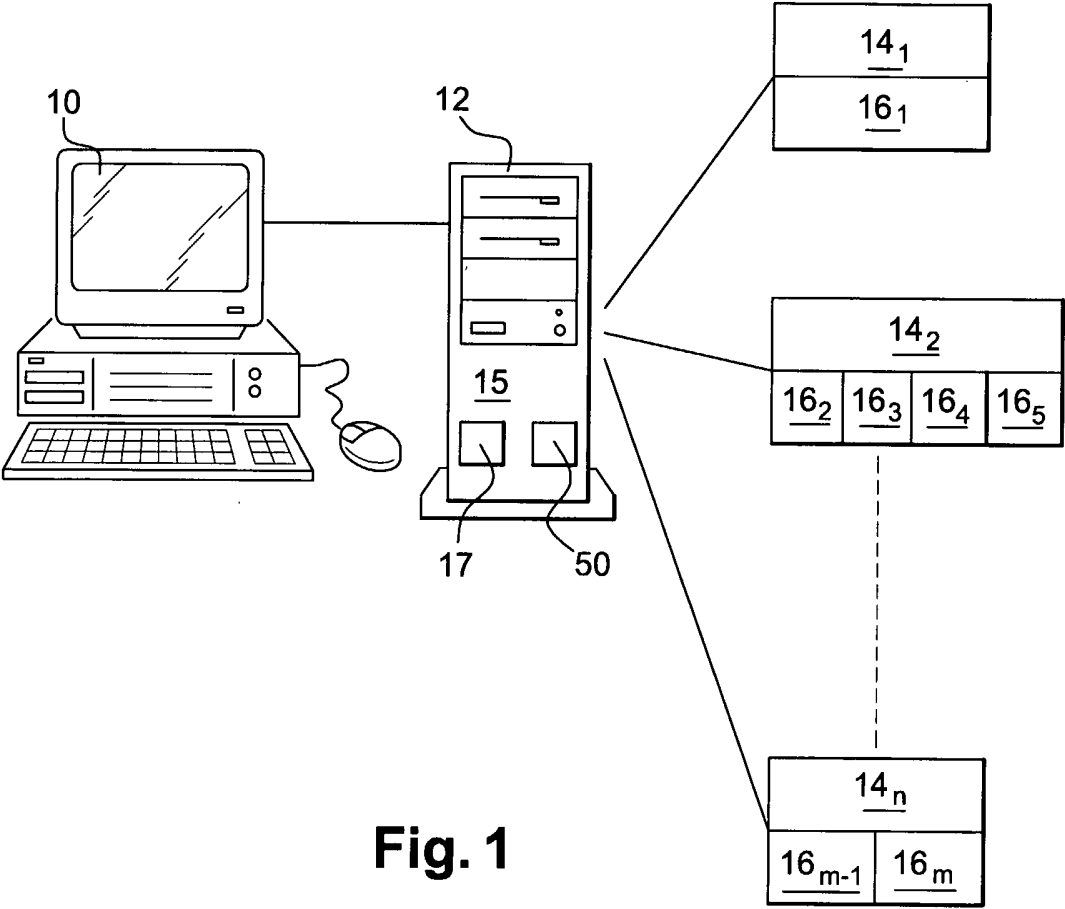
simultanément sur un calculateur (14_i) sont modifiables par l'utilisateur, et/ou de façon dynamique.

5 10. Procédé de gestion de l'exécution d'au moins un programme (P) selon l'une quelconque des revendications 6 à 9, caractérisé en ce que la partie (D₁) des données d'entrée sur laquelle est générée la première sous-requête (R_{s1}(P, D₁)) correspond au nombre (NE_e) maximal de données d'entrées que le calculateur sélectionné (14_e) est a priori capable de traiter.

10 11. Procédé de gestion de l'exécution d'au moins un programme (P) selon l'une quelconque des revendications 1 à 10, caractérisé en ce que l'on sélectionne (24) la requête (R₁(P, D)) parmi une liste de requêtes (17) classées (22) par ordre de priorité.

12. Procédé de gestion de l'exécution d'au moins un programme (P) selon la revendication 11, caractérisé en ce que l'on renvoie dans la liste des requêtes (17) la seconde sous-requête (R_{s2}(P, D₂)).

15 13. Procédé de gestion de l'exécution d'au moins un programme (P) selon la revendication 11 ou 12, caractérisé en ce que l'on affecte à chaque requête de la liste un coefficient de priorité, et en ce que les requêtes de coefficients de priorité égaux sont classées (22) par ordre d'arrivée dans la liste (17).



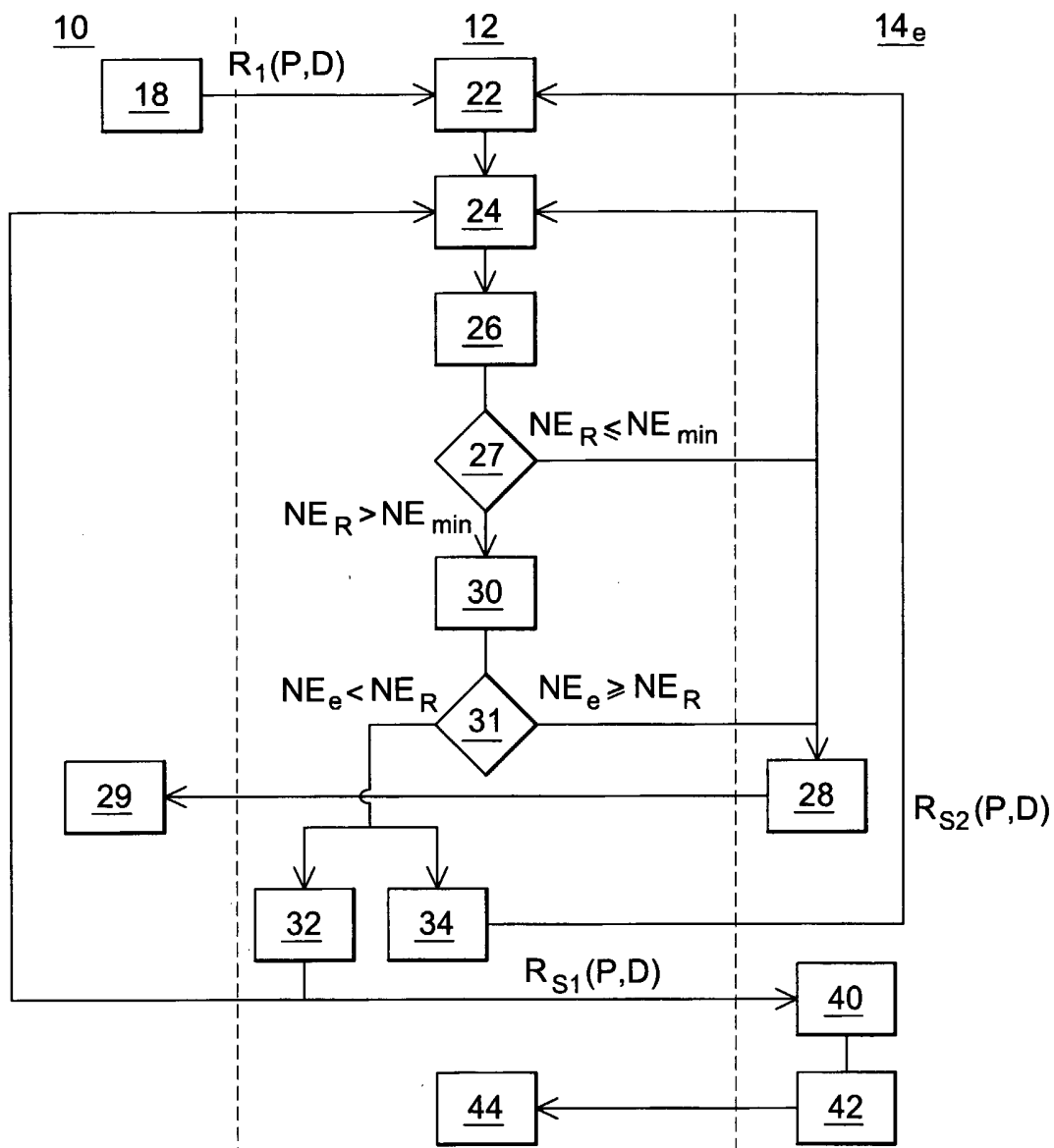


Fig. 2

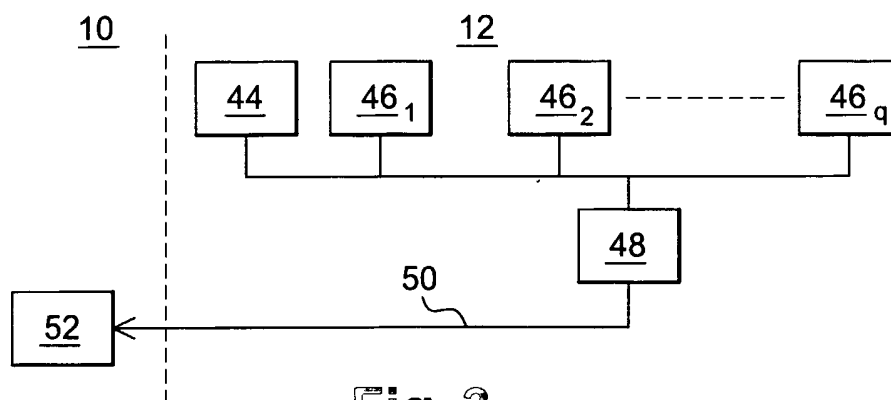


Fig. 3