(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
15 décembre 2016 (15.12.2016)

WIPO | PCT

(10) Numéro de publication internationale
WO 2016/198762 A1

(51) Classification internationale des brevets :
G06F 9/50 (2006.01) *G06F 11/34* (2006.01)

(21) Numéro de la demande internationale :
PCT/FR2016/051266

(22) Date de dépôt international :
27 mai 2016 (27.05.2016)

(25) Langue de dépôt : français

(26) Langue de publication : français

(30) Données relatives à la priorité :
1555267 9 juin 2015 (09.06.2015) FR

(71) Déposant : **ORANGE** [FR/FR]; 78 rue Olivier de Serres, 75015 Paris (FR).

(72) Inventeurs : **MONIN, Wei**; 68 impasse des Corjons, 38410 Saint Martin D'uriage (FR). **VACHET, Guy**; 5 passage de la Teille, 38240 Meylan (FR). **DILLENSEGER, Bruno**; 2 rue Charles Piot, 38320 Eybens (FR). **ETCHEVERS, Xavier**; 20 avenue Marius Cottier, 38700 Corenc (FR).

(74) Mandataire : **ORANGE/IPL**; LECOMTE Isabelle, 38-40 rue du Général Leclerc, 92794 Issy Moulineaux Cedex 9 (FR).

(81) États désignés *(sauf indication contraire, pour tout titre de protection nationale disponible)* : AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

[Suite sur la page suivante]

(54) **Title** : METHOD AND SYSTEM FOR DETERMINING A TARGET CONFIGURATION OF SERVERS FOR DEPLOYMENT OF A SOFTWARE APPLICATION

(54) **Titre** : PROCÉDÉ ET SYSTÈME DE DÉTERMINATION D'UNE CONFIGURATION DE SERVEURS CIBLE POUR UN DÉPLOIEMENT D'UNE APPLICATION LOGICIELLE

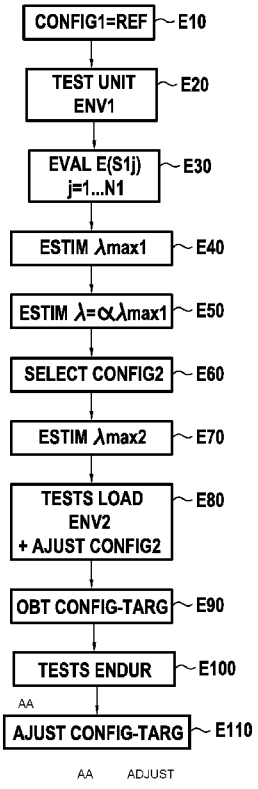


FIG.4

(57) **Abstract** : The method comprises: the obtaining (E20) for at least one service offered by the application, via at least one unit test carried out on a reference configuration of servers, of a mean execution time of a request invoking this service for each server; the selecting (E60) of an initial configuration of servers supporting a target load, while taking account of the times obtained, of the target load and of an admissible load of the reference configuration, and by using a numerical model reflecting at least one constraint of deployment of the application on the initial configuration; and the determining (E90) on the basis of the initial configuration of a target configuration via load-holding tests (E80) parameterized as a function of a maximum load of the initial configuration and of the target load, and by using a profile of requests invoking said at least one service representative of a use of the application during deployment, the determination comprising, as a function of the result of each test, an adjustment of resource(s) of the initial configuration so as to obtain on completion of the tests a target configuration satisfying a maximum response time and the target load.

(57) **Abrégé** : Le procédé comprend: l'obtention (E20) pour au moins un service offert par l'application,

[Suite sur la page suivante]

WO 2016/198762 A1

(84) États désignés *(sauf indication contraire, pour tout titre de protection régionale disponible)* : ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), eurasien (AM, AZ, BY, KG, KZ, RU, TJ, TM), européen (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Publiée :

— *avec rapport de recherche internationale (Art. 21(3))*

via au moins un test unitaire réalisé sur une configuration de serveurs de référence, d'un temps d'exécution moyen d'une requête invoquant ce service pour chaque serveur; la sélection (E60) d'une configuration de serveurs initiale supportant une charge cible, en tenant compte des temps obtenus, de la charge cible et d'une charge admissible de la configuration de référence, et en utilisant un modèle numérique reflétant au moins une contrainte de déploiement de l'application sur la configuration initiale; et la détermination (E90) à partir de la configuration initiale d'une configuration cible via des tests de tenue en charge (E80) paramétrés en fonction d'une charge maximale de la configuration initiale et de la charge cible, et en utilisant un profil de requêtes invoquant ledit au moins un service représentatif d'une utilisation de l'application en déploiement, la détermination comprenant, en fonction du résultat de chaque test, un ajustement de ressource(s) de la configuration initiale pour obtenir à l'issue des tests une configuration cible vérifiant un temps de réponse maximal et la charge cible.

PROCÉDÉ ET SYSTÈME DE DÉTERMINATION D'UNE CONFIGURATION DE SERVEURS CIBLE POUR UN DÉPLOIEMENT D'UNE APPLICATION LOGICIELLE

Arrière-plan de l'invention

L'invention se rapporte au domaine général des applications logicielles.

Elle concerne plus particulièrement le dimensionnement des ressources permettant l'exécution de telles applications.

L'invention s'applique ainsi de façon privilégiée mais non limitative lorsque des applications logicielles sont déployées dans un système informatique en nuage aussi plus communément appelé « cloud » en anglais.

La gestion de l'élasticité des applications logicielles déployées en cloud est une problématique importante, notamment dans les systèmes de type « PaaS » (« Platform as a Service » en anglais ou Plateforme en tant que service). Ces systèmes, principalement destinés aux entreprises, visent à proposer des services techniques qui facilitent la construction, la mise en œuvre et l'administration des applications logicielles afin de recentrer l'attention des utilisateurs sur leurs applications. La finalité d'un système PaaS est de masquer la complexité liée à l'instanciation et à l'exploitation des applications. Pour cela, le système PaaS propose des briques techniques, accessibles au travers d'interfaces de haut niveau via un réseau de télécommunications, permettant de prendre en charge un panel important des services liés à l'administration des applications, notamment l'adaptation dynamique aux variations de charge, la sûreté de fonctionnement et la sécurité. Le périmètre fonctionnel de ces briques correspond généralement à celui rempli par un intergiciel (e.g. serveur HTTP, serveur d'application, base de données, interpréteurs de script, canevas logiciels divers...), à savoir des opérations qui peuvent être mutualisées entre plusieurs applications.

Dans ce contexte, la gestion de l'élasticité des applications logicielles déployées en cloud a pour but de fournir et d'ajuster (i.e. adapter) les ressources nécessaires au bon fonctionnement de ces applications logicielles, en termes notamment de volumétrie, de disponibilité, de temps de réponse et de fiabilité, tout en satisfaisant les contraintes spécifiques à ces applications logicielles (ex. coût, placement des machines virtuelles, droit d'utilisation, etc.). En vue d'optimiser un rapport qualité/coût, la gestion de l'élasticité doit ainsi :

- permettre de déterminer (i.e. dimensionner) les ressources nécessaires au déploiement d'une application logicielle ainsi que la configuration optimale de ces ressources, en fonction notamment d'exigences contractualisées (par exemple sous la forme d'accord de niveau de service ou encore SLA pour « Service Level Agreement » en anglais) et des ressources engagées pour d'autres applications logicielles en cours d'exécution ; et
- être capable d'ajouter ou de supprimer dynamiquement des ressources durant l'exécution de l'application logicielle, afin notamment d'absorber des « rafales » ponctuelles (instantanées) de requêtes dues à des pics d'utilisation de l'application logicielle ou au partage de

l'environnement d'exécution avec d'autres applications logicielles, et de respecter ainsi les exigences contractualisées dans les SLA (ex. en termes de temps de réponse de l'application logicielle). Pour ce faire, la gestion de l'élasticité doit être en mesure de déterminer rapidement la nouvelle configuration de ressources et préciser quand doit intervenir l'opération de reconfiguration.

Les techniques de dimensionnement des ressources sont donc au cœur de la gestion de l'élasticité. Elles nécessitent généralement une bonne connaissance de l'application logicielle, notamment en termes de consommation de ressources par les instances de déploiement de l'application (ex. serveurs), mais également de profils de requêtes utilisateurs (ex. répartition par service, débits des requêtes, temps d'inter-arrivées entre les requêtes, etc.).

Dans l'état actuel de la technique, il n'existe pas de procédé structuré connu permettant de déterminer facilement les ressources nécessaires (ainsi que la configuration de ces ressources) pour le déploiement ou l'exécution d'une application logicielle. En effet, aujourd'hui, on dimensionne et on ajuste uniquement de façon expérimentale avec des outils de tests et/ou de supervision les ressources nécessaires à l'exécution d'une application logicielle à partir d'une étude de pré-métrologie. Cette façon de procéder implique un investissement très important en termes de coût et de temps : de nombreuses configurations doivent être testées avant de trouver une configuration optimale, nécessitant la mise en place d'équipements physiques et de logiciels, l'exécution de nombreux tests, etc. Une telle approche n'est donc pas toujours envisageable en pratique du fait de contraintes de temps imposées au déploiement de l'application logicielle par exemple ou de contraintes imposées en termes de retour sur investissement.

Objet et résumé de l'invention

L'invention permet de remédier notamment à cet inconvénient en proposant un procédé de détermination d'une configuration de serveurs dite cible pour un déploiement d'une application logicielle apte à offrir au moins un service, ce procédé comprenant :

— une étape d'obtention, pour au moins un service offert par l'application logicielle, au moyen d'au moins un test unitaire réalisé sur une configuration de serveurs dite de référence apte à exécuter l'application logicielle, d'un temps d'exécution moyen d'une requête invoquant ce service pour chaque serveur de la configuration de référence ;

— une étape de sélection d'une configuration de serveurs dite initiale pour le déploiement de l'application logicielle apte à supporter une charge cible déterminée pour l'application logicielle, cette étape de sélection tenant compte des temps d'exécution moyens obtenus lors dudit au moins un test unitaire, de la charge cible et d'une charge admissible déterminée pour la configuration de référence, et utilisant un modèle numérique reflétant au moins une contrainte de déploiement de l'application logicielle imposée à la configuration de serveurs initiale ; et

— une étape de détermination, à partir de la configuration initiale, d'une configuration de serveurs cible destinée à être utilisée pour le déploiement de l'application logicielle, ladite étape de détermination comprenant la réalisation d'une pluralité de tests de tenue en charge paramétrés en fonction d'une charge maximale théorique estimée pour la configuration initiale et de la charge cible, et utilisant au moins un profil de requêtes invoquant ledit au moins un service offert par l'application logicielle, ce profil étant représentatif d'une utilisation de l'application logicielle lors de son déploiement, ladite étape de détermination comprenant en outre, en fonction du résultat de chaque test de tenue en charge, un ajustement d'au moins une ressource de la configuration initiale de sorte que la configuration cible obtenue à l'issue de ladite pluralité de tests de tenue en charge soit apte à vérifier un temps de réponse maximal fixé pour l'application logicielle et à supporter ladite charge cible.

Corrélativement, l'invention vise aussi un système de détermination d'une configuration cible pour un déploiement d'une application logicielle, le système comprenant :

— un premier outil de test permettant l'exécution d'au moins un test unitaire sur l'application logicielle lorsque celle-ci est exécutée sur une configuration de serveurs dite de référence ;

— un module d'obtention, configuré pour commander l'exécution d'au moins un test unitaire par le premier outil de test sur ladite configuration de référence et pour obtenir, pour au moins un service offert par l'application logicielle, un temps d'exécution moyen d'une requête invoquant ce service pour chaque serveur de la configuration de référence ;

— un module de sélection, configuré pour sélectionner une configuration de serveurs initiale pour le déploiement de l'application logicielle apte à supporter une charge cible déterminée pour l'application logicielle, ledit module de sélection étant configuré pour tenir compte des temps d'exécution moyens obtenus lors dudit au moins un test unitaire, de la charge cible et d'une charge admissible déterminée pour la configuration de référence, et pour utiliser un modèle numérique reflétant au moins une contrainte de déploiement de l'application logicielle imposée à la configuration de serveurs initiale ;

— un second outil de test permettant l'exécution de tests de tenue en charge sur ladite application logicielle lorsque celle-ci est exécutée sur la configuration de serveurs initiale sélectionnée par le module de sélection ; et

— un module de détermination configuré pour déterminer à partir de la configuration initiale une configuration de serveurs cible destinée à être utilisée pour le déploiement de l'application logicielle, ledit module de détermination étant configuré pour commander l'exécution d'une pluralité de tests de tenue en charge par le second outil de test, lesdits tests de tenue en charge étant paramétrés par le module de détermination en fonction d'une charge maximale théorique estimée pour la configuration initiale et de la charge cible, et utilisant au moins un profil de requêtes invoquant ledit au moins un service offert par l'application logicielle, ce profil étant représentatif d'une utilisation de l'application logicielle lors de son déploiement, ledit

module de détermination étant en outre configuré pour ajuster, en fonction du résultat de chaque test de tenue en charge, au moins une ressource de la configuration initiale de sorte que la configuration cible obtenue à l'issue de ladite pluralité de tests de tenue en charge soit apte à vérifier un temps de réponse maximal fixé pour l'application logicielle et à supporter ladite charge cible.

Par serveur, on entend ici tout type de machine physique ou virtuelle, apte à exécuter tout ou partie d'une instance de l'application logicielle. L'application logicielle est destinée à être déployée sur une configuration (i.e. un agencement) de serveurs formée d'un ou de plusieurs serveurs reliés entre eux et vérifiant des contraintes de déploiement imposées pour la mise en production de l'application logicielle (ex. déploiement multi-sites, utilisation de machines spécifiques, etc.).

L'invention propose donc un procédé de dimensionnement des ressources nécessaires au déploiement d'une application logicielle dans un environnement de production qui s'appuie sur l'exécution structurée de deux types de tests, à savoir des tests unitaires sans accès concurrent aux ressources puis des tests de tenue en charge, judicieusement paramétrés. Ces différents tests sont réalisés sur des configurations de serveurs différentes, à savoir :

— pour les tests unitaires, sur une configuration de serveurs dite de référence, prédéterminée et relativement simple, ce qui permet d'obtenir des résultats en termes de consommation de ressources ; et

— pour les tests de tenue en charge, sur une configuration de serveurs de production dite initiale dont l'architecture reflète les contraintes imposées par le déploiement de l'application logicielle en environnement de production, et sélectionnée grâce à l'exploitation des résultats des tests unitaires réalisés sur la configuration de référence combinée à un modèle numérique permettant de mettre en correspondance les ressources des deux configurations de serveurs.

L'invention permet ainsi de déterminer de façon simple et efficace une configuration de production cible adaptée en termes de ressources au déploiement de l'application logicielle dans un environnement réel d'exécution et vérifiant les différentes exigences contractuelles qui lui sont imposées par l'intermédiaire par exemple d'un accord SLA tel que précédemment décrit. Cette configuration cible est obtenue en ajustant de façon itérative à l'issue de chaque test de tenue en charge les ressources de la configuration initiale de sorte à vérifier ces exigences.

L'ajustement des ressources d'une configuration de serveurs comprend, au sens de l'invention, le maintien, l'ajout, et/ou le retrait de ressources pour au moins un serveur de la configuration, et/ou le maintien ou la mise à jour de paramètres de cette configuration (comme par exemple le nombre de serveurs considérés dans cette configuration). Cet ajustement conduit à une configuration de production « optimale », c'est-à-dire à une configuration minimale de serveurs, permettant de garantir le respect des exigences fixées pour l'application logicielle en termes de volumétrie (i.e. de charge cible), de disponibilité et de temps de réponse de bout en bout de l'application logicielle aux requêtes des utilisateurs.

Comme mentionné précédemment, la configuration de serveurs de référence utilisée pour réaliser les tests unitaires est choisie généralement de complexité relativement réduite par rapport à la configuration de serveurs sur laquelle est destinée à être effectivement déployée l'application logicielle. Cette configuration de serveurs de référence peut typiquement se limiter à une seule machine (i.e. un serveur unique), par exemple la machine sur laquelle est développée l'application. Les tests unitaires réalisés à partir de cette configuration de référence visent à caractériser, pour chaque type de requête identifié comme pertinent au regard des services offerts par l'application logicielle (ex. requêtes de type souscription, résiliation, consultations d'objets multimédia, traitement, etc.), les ressources consommées par celle-ci sur chaque serveur de la configuration de référence pour traiter ce type de requête quand il n'y a pas d'accès concurrent à ces ressources. Autrement dit, il s'agit là de quantifier l'utilisation au niveau de chaque serveur de la configuration de référence des processeurs (ou unités de traitement centrales ou encore CPU pour « Central Processing Unit » en anglais), de la mémoire (ex. disque, RAM (Random Access Memory)) et/ou encore des ressources réseaux (ex. connecteurs réseaux) sollicitées lors du traitement des requêtes.

De tels tests unitaires sont connus en soi et sont classiquement mis en œuvre lors du développement et du débogage des applications logicielles. Durant ces tests, on s'assure qu'il n'y a pas d'accès concurrent aux ressources dont on cherche à quantifier la consommation pour traiter les requêtes envoyées à l'application logicielle. A cet effet, les requêtes peuvent être par exemple envoyées une par une à l'application logicielle exécutée par la configuration de référence (une requête étant envoyée à l'issue du traitement de la précédente), ou N requêtes (N nombre entier) peuvent être envoyées de façon déterministe à l'application logicielle en s'assurant qu'il n'y ait pas d'accès concurrent aux ressources utilisées pour leur traitement.

L'invention propose de tirer profit de ces tests unitaires simples à réaliser pour récolter diverses métriques de consommation de ressources liées au traitement intrinsèque de chaque requête par chaque serveur de la configuration de référence, et plus particulièrement pour chaque service représentatif offert par l'application logicielle, pour obtenir le temps d'exécution moyen d'une requête invoquant ce service par chaque serveur de la configuration de référence. D'autres métriques peuvent bien entendu être obtenues lors de ces tests unitaires, comme par exemple le nombre de connecteurs réseau utilisés simultanément, le volume de RAM utilisé sur chaque serveur, la durée et le volume de disque occupé, la durée s'écoulant entre la fin d'exécution de la requête sur un serveur et le début d'exécution sur le serveur suivant, etc. Les métriques ainsi récoltées servent d'entrées, conformément à l'invention, à des modèles numériques reflétant diverses exigences imposées en termes d'architecture notamment ou de types de machines à la configuration de production (c'est-à-dire diverses contraintes de déploiement de l'application logicielle) et sont exploitées pour sélectionner une configuration de production initiale satisfaisant une volumétrie (charge en termes de débit de requêtes) cible fixée pour l'application logicielle. Les modèles numériques considérés modélisent par exemple les contraintes (exigences) de

déploiement de l'application logicielle à partir d'au moins une file d'attente ou d'un réseau de files d'attente.

Il convient de noter que la sélection de la configuration de production initiale se fait sur la base de résultats de tests unitaires conduits dans un environnement d'exécution idéal où les requêtes ne subissent aucune concurrence pour l'accès aux ressources. Les tests unitaires permettent donc de déterminer uniquement quels sont les besoins de l'application logicielle en ressources CPU, mémoire, et/ou réseau pour traiter intrinsèquement chaque requête. La configuration de production initiale sélectionnée conformément à l'invention ne tient donc compte que d'une volumétrie à respecter et du temps d'exécution intrinsèque des requêtes en découlant. Toutefois ces hypothèses ne sont pas représentatives des conditions réelles d'utilisation de l'application logicielle dans lesquelles les requêtes des utilisateurs arrivent généralement de façon non déterministe, typiquement par rafales et avec des débits parfois très différents du débit moyen des requêtes. Les tests unitaires ne fournissent donc aucune information sur le temps de réponse à proprement parler de l'application logicielle dans un environnement de production, de sorte qu'un ajustement des ressources de la configuration de production initiale sélectionnée à l'issue des tests unitaires est proposé par l'invention pour tenir compte des exigences imposées à l'application logicielle en termes notamment de temps de réponse maximal de bout en bout.

Ces ajustements sont encadrés par des tests de tenue en charge dûment paramétrés et qui permettent de dimensionner les ressources nécessaires à la mise en production de l'application logicielle. Ces tests de tenue en charge prennent en effet en compte non seulement les contraintes en termes de volumétrie de l'application logicielle, mais également le profil des requêtes invoquant les services offerts par celle-ci et en particulier la nature des temps d'inter-arrivées des requêtes (i.e. la distribution d'arrivée de ces requêtes au niveau de l'application logicielle). Ainsi, un profil de requêtes invoquant un service comprend par exemple, pour au moins une période de temps déterminée d'un cycle d'activité de l'application logicielle ou d'un cycle d'activité de l'application logicielle :

- un débit moyen des requêtes invoquant ce service sur ladite période de temps ;
- une proportion de requêtes invoquant ce service parmi les requêtes invoquant des services offerts par l'application logicielle ; et
- une distribution des durées d'inter-arrivées entre deux requêtes invoquant ce service sur ladite période de temps.

En d'autres mots, les tests de tenue en charge prennent en compte un profil de requêtes réaliste représentatif de l'utilisation de l'application logicielle dans un environnement réel d'exécution (i.e. similaire ou tout du moins proche de celui auquel on s'attend lors de la mise en production de l'application logicielle).

Il convient de noter que la configuration de production initiale sélectionnée par l'invention n'est pas nécessairement strictement conforme à la configuration de serveurs qui sera déployée ultérieurement en production et testée dans un environnement réel d'exécution. Elle peut

être notamment de complexité plus réduite, mais elle est sélectionnée préférentiellement de sorte à être représentative des différentes fonctionnalités et problématiques attendues lors de la mise en production de l'application logicielle. Ainsi, la configuration de serveurs initiale est choisie préférentiellement de sorte à être représentative de la complexité logicielle, technique et fonctionnelle de l'environnement de production de l'application logicielle (ex. présence d'un nœud maître et de plusieurs nœuds secondaires qui communiquent entre eux, modélisation de la communication inter-sites, prise en compte des problèmes de sécurité, etc.). De cette sorte, on s'assure que le dimensionnement des ressources réalisé par l'invention est pertinent et exploitable pour faciliter le déploiement en production de l'application et limiter les tests réalisés à grande échelle.

Plusieurs configurations de serveurs cibles peuvent être identifiées à partir de l'invention. La sélection de l'une de ces configurations peut ensuite être réalisée à partir de tests expérimentaux réalisés dans l'environnement de production. Toutefois le nombre de configurations de serveurs à tester est considérablement réduit par rapport à l'état de la technique.

L'invention se distingue donc de l'état actuel de la technique en ce qu'elle offre une solution qui permet, avant la mise en production de l'application logicielle, de présélectionner et de dimensionner, au moyen de modèles numériques, une ou plusieurs configurations de serveurs appropriées qui servent de points de départ pour le déploiement réel de l'application. Ainsi, contrairement à l'état de la technique, la sélection de la configuration optimale ne s'appuie pas uniquement sur des tests expérimentaux réalisés en production dans un environnement réel d'exécution et qui peuvent s'avérer, comme mentionné précédemment, hasardeux, coûteux et chronophages. L'invention offre au contraire une possibilité de pré-provisionner les ressources nécessaires à l'exécution d'une application logicielle, ce qui permet d'effectuer de tels tests expérimentaux uniquement sur un nombre réduit de configurations en vue de sélectionner une configuration optimale. On obtient ainsi grâce à l'invention un gain substantiel en complexité et en temps.

L'invention permet donc, par rapport aux solutions existantes basées essentiellement sur des observations expérimentales, d'assurer la maîtrise et l'efficacité de la gestion de l'élasticité, et de réduire les coûts d'investissement pour des applications logicielles et des systèmes de type PaaS.

Il convient de noter par ailleurs que de façon avantageuse, la mise en œuvre de l'invention ne requiert pas de connaissance détaillée des composants logiciels constituant l'application logicielle ni de l'enchaînement d'exécution de ces composants. L'application logicielle peut avantageusement être considérée comme une « boîte noire » offrant un ou plusieurs services en réponse à des requêtes d'utilisateur vérifiant un certain profil, et répartie ou instanciée sur un ou plusieurs serveurs. Les métriques de consommation des ressources issues des tests unitaires et qui sont ensuite utilisées pour sélectionner une configuration de serveurs initiale sur laquelle sont

effectués les tests de tenues en charge sont des métriques tous composants confondus mesurées sur chaque serveur de la configuration de référence.

Dans un mode particulier de réalisation de l'invention, le procédé de détermination comprend en outre une étape d'estimation d'une charge maximale théorique pour la configuration de référence et la charge admissible déterminée pour la configuration de référence résulte du produit d'un paramètre α compris entre 0 et 1 par la charge maximale théorique estimée pour la configuration de référence.

Par exemple, la charge maximale théorique pour la configuration de référence est estimée en appliquant une condition de stabilité de l'application logicielle à au moins un temps d'exécution moyen d'une requête de l'application logicielle dérivé pour ledit au moins un serveur de la configuration de référence à partir des temps d'exécution moyens obtenus pour ce serveur pour les services offerts par l'application logicielle.

Une telle condition de stabilité est vérifiée notamment si pour chaque serveur de la configuration de serveurs considérée pour exécuter l'application logicielle, le taux (débit) d'arrivée des requêtes est strictement inférieur au taux d'utilisation de ce serveur. Le temps de réponse du serveur peut être utilisé pour évaluer son taux d'utilisation. Cette condition de stabilité permet d'évaluer facilement une charge théorique maximale pour l'application logicielle aux alentours de laquelle l'application logicielle n'est plus capable de traiter les requêtes. Cette charge théorique maximale est utilisée pour dimensionner la configuration de serveurs initiale.

Par ailleurs, les inventeurs ont observé qu'en amont de cette charge théorique maximale, le temps de réponse moyen d'un serveur en fonction de la charge de l'application logicielle suit généralement un premier régime linéaire en pente douce suivi d'une montée brutale définissant un deuxième régime linéaire à l'approche de la saturation du serveur. Le paramètre α compris entre 0 et 1 utilisé pour définir la charge admissible de la configuration initiale est choisi préférentiellement pour que la charge admissible considérée se situe dans une zone de transition se trouvant juste avant le deuxième régime linéaire et dans laquelle le temps de réponse de bout en bout de l'application logicielle reste inférieur à un temps de réponse maximal fixé pour l'application logicielle (typiquement dans l'accord SLA). Dans cette zone « optimale », le temps de réponse versus la charge est contenu et reste inférieur au temps de réponse maximal autorisé pour l'application logicielle. On note que la valeur du paramètre α peut dépendre de la distribution des temps d'inter-arrivées des requêtes, comme détaillé ultérieurement. Toutefois une valeur de 0.8 ou 0.85 (ou aux alentours de ces deux valeurs) convient généralement assez bien pour mettre en œuvre l'invention.

Dans un mode particulier de réalisation, au cours de l'étape de détermination, l'ajustement d'au moins une ressource de la configuration initiale à l'issue d'un test de tenue en charge est réalisé en fonction d'une différence entre un temps de réponse de l'application logicielle évalué lors du test de tenue en charge et le temps de réponse maximal fixé pour l'application logicielle.

Cette façon d'ajuster les ressources en tenant compte du temps de réponse maximal fixé pour l'application logicielle et le temps de réponse obtenu de bout en bout avec la configuration de serveurs en cours de test permet de converger rapidement vers un dimensionnement approprié des ressources.

Dans un mode de réalisation, l'étape de détermination comprend au moins :

- un premier test de tenue en charge réalisé avec une première charge inférieure à une valeur minimum entre une moitié de la charge maximale théorique estimée pour la configuration initiale et le produit de la charge cible par un nombre réel prédéterminé inférieur ou égal à 1 ;
- un deuxième test de tenue en charge réalisé avec une deuxième charge inférieure à la première charge ; et
- un troisième test de tenue en charge réalisé avec une troisième charge égale à la charge cible.

Ce protocole permet de limiter les tests de tenue en charge réalisés sur la configuration de serveurs initiale et d'aboutir rapidement à une configuration de serveurs cible dérivée de la configuration initiale. Par configuration dérivée de la configuration initiale on entend au sens de l'invention que la configuration est la même que la configuration initiale si celle-ci est considérée comme correctement dimensionnée ou qu'elle est obtenue après avoir ajusté des ressources de la configuration initiale.

Dans ce mode de réalisation, l'étape de détermination peut comprendre en outre une estimation à l'issue du deuxième test, d'un temps de réponse de l'application logicielle avec la charge cible à partir d'un temps de réponse évalué à l'issue du premier test réalisé avec la première charge sur une configuration testée dérivée de la configuration de serveur initiale et d'un temps de réponse de l'application logicielle évalué à l'issue du deuxième test réalisé avec la deuxième charge sur ladite configuration testée, l'ajustement de ressources étant réalisé à l'issue du deuxième test en fonction d'une différence entre le temps de réponse estimé pour la charge cible et le temps de réponse maximal fixé pour l'application logicielle.

Typiquement, à l'issue du deuxième test :

- la configuration de serveurs testée peut être considérée comme sous-dimensionnée si le temps de réponse estimé pour la charge cible est supérieur au temps de réponse maximal ; et/ou
- la configuration de serveurs testée peut être considérée comme surdimensionnée si le temps de réponse estimé pour la charge cible est inférieur au produit d'un nombre réel prédéterminé γ compris entre 0 et 1 et du temps de réponse maximal, par exemple 0.9 ; et/ou
- la configuration de serveurs testée peut être considérée comme correctement dimensionnée sinon.

L'ajustement des ressources est ensuite réalisé en conséquence et le ou les tests de tenue en charge précédents sont réitérés sur la configuration ajustée pour valider cet ajustement et/ou procéder à un nouvel ajustement, et ce, jusqu'à déterminer la configuration cible apte à être déployée et qui répond aux exigences fixées pour la mise en production de l'application logicielle en termes de charge et de temps de réponse de bout en bout. Typiquement, la configuration cible

correspond à une configuration de serveurs testée pour laquelle à l'issue du troisième test, un temps de réponse de l'application logicielle évalué pour la charge cible sur cette configuration de serveurs testée est inférieur au temps de réponse maximal.

Dans un mode de réalisation particulier de l'invention, le procédé de détermination comprend en outre une étape de validation de la configuration cible au moyen d'un test d'endurance réalisé pendant une durée d'exécution prédéterminée de l'application logicielle.

Il s'agit par ce test d'endurance de tester l'application logicielle et la configuration de serveurs cible déterminée aux tests unitaires et aux tests de tenue en charge sur une durée suffisamment longue afin de prendre en compte, pour le dimensionnement des ressources, des évènements rares dont la probabilité d'apparition est relativement faible. Le résultat de ce test d'endurance permet de réajuster le cas échéant les ressources de la configuration cible en vue du déploiement de l'application logicielle. En effet, les métriques et les modèles numériques utilisés conformément à l'invention pour déterminer la configuration cible reposent sur certaines hypothèses d'abstraction sur le comportement de l'application logicielle, et notamment d'indépendance de certains composants ou de distribution des requêtes. Le test d'endurance permet d'affiner ces modèles (et d'ajuster si besoin le dimensionnement des ressources), et dans une certaine mesure, de mettre en évidence les écarts entre la réalité et les modèles numériques utilisés établis, de les évaluer et de juger de leurs pertinences.

Dans un mode particulier de réalisation, l'application logicielle est caractérisée par un cycle d'activité comprenant une pluralité d'intervalles, chaque intervalle étant associé à un débit moyen d'arrivée des requêtes représentatif sur cet intervalle, et dans lequel lesdites étapes d'obtention, de sélection et de détermination sont mises en œuvre pour au moins un intervalle de ladite pluralité d'intervalles.

Ce mode de réalisation permet de s'adapter aux applications logicielles pour lesquelles on a une distribution dynamique des requêtes, autrement dit pour lesquelles le débit des arrivées de requêtes de services varie de manière significative sur des intervalles de temps et ce de façon répétitive sur le cycle d'activité.

Selon un autre aspect, l'invention vise également un procédé de gestion d'élasticité d'une configuration de serveurs apte à exécuter une application logicielle, ce procédé comprenant :

- une étape de détermination d'une configuration cible de serveurs pour le déploiement de l'application logicielle comprenant l'exécution d'un procédé de détermination selon l'invention ; et
- une étape d'exécution de l'application logicielle sur ladite configuration cible de serveurs comprenant :
    - une surveillance d'au moins une métrique de supervision de cette configuration cible ; et

- en fonction de ladite au moins une métrique, un ajustement des ressources de la configuration cible, cet ajustement comprenant un maintien et/ou un ajout et/ou un retrait dynamique de ressources à la configuration cible.

L'invention vise aussi un système de gestion d'élasticité d'une configuration de serveurs apte à exécuter une application logicielle comprenant :

— un système selon l'invention de détermination d'une configuration cible de serveurs pour le déploiement de l'application logicielle comprenant l'exécution d'un procédé de détermination ;
— un module de déclenchement d'une exécution de l'application logicielle sur la configuration cible de serveurs ;
— un module de surveillance d'au moins une métrique de supervision de cette configuration cible lors de l'exécution de l'application logicielle ; et
— un module d'ajustement configuré pour ajuster les ressources allouées à la configuration cible en fonction de ladite au moins une métrique, ce module d'ajustement étant apte à maintenir les ressources allouées ou à ajouter et/ou retirer dynamiquement des ressources à la configuration cible.

Le procédé et le système de gestion d'élasticité bénéficient des mêmes avantages cités précédemment que le procédé et le dispositif de détermination selon l'invention.

Dans un mode particulier de réalisation, les différentes étapes du procédé de détermination et/ou du procédé de gestion d'élasticité sont déterminées par des instructions de programmes d'ordinateurs.

En conséquence, l'invention vise aussi un programme d'ordinateur sur un ou plusieurs support(s) d'informations, ce programme étant susceptible d'être mis en œuvre dans un système de détermination ou plus généralement dans un ordinateur, ce programme comportant des instructions adaptées à la mise en œuvre des étapes d'un procédé de dimensionnement tel que décrit ci-dessus.

L'invention vise également un programme d'ordinateur sur un ou plusieurs support(s) d'informations, ce programme étant susceptible d'être mis en œuvre dans un système de gestion d'élasticité ou plus généralement dans un ordinateur, ce programme comportant des instructions adaptées à la mise en œuvre des étapes d'un procédé de gestion d'élasticité tel que décrit ci-dessus.

Ce programme peut utiliser n'importe quel langage de programmation, et être sous la forme de code source, code objet, ou de code intermédiaire entre code source et code objet, tel que dans une forme partiellement compilée, ou dans n'importe quelle autre forme souhaitable.

L'invention vise aussi un support d'informations lisible par un ordinateur, et comportant des instructions d'un programme d'ordinateur tel que mentionné ci-dessus.

Le support d'informations peut être n'importe quelle entité ou dispositif capable de stocker le programme. Par exemple, le support peut comporter un moyen de stockage, tel qu'une

ROM, par exemple un CD ROM ou une ROM de circuit microélectronique, ou encore un moyen d'enregistrement magnétique, par exemple une disquette (floppy disc) ou un disque dur.

D'autre part, le support d'informations peut être un support transmissible tel qu'un signal électrique ou optique, qui peut être acheminé via un câble électrique ou optique, par radio ou par d'autres moyens. Le programme selon l'invention peut être en particulier téléchargé sur un réseau de type Internet.

Alternativement, le support d'informations peut être un circuit intégré dans lequel le programme est incorporé, le circuit étant adapté pour exécuter ou pour être utilisé dans l'exécution du procédé en question.

On peut également envisager, dans d'autres modes de réalisation, que le procédé de détermination, le procédé de gestion d'élasticité, le système de détermination et le système de gestion d'élasticité selon l'invention présentent en combinaison tout ou partie des caractéristiques précitées.

Brève description des dessins

D'autres caractéristiques et avantages de la présente invention ressortiront de la description faite ci-dessous, en référence aux dessins annexés qui en illustrent un exemple de réalisation dépourvu de tout caractère limitatif. Sur les figures :

- la figure 1 représente, de façon schématique, un système de détermination conforme à l'invention, dans un mode particulier de réalisation ;
- la figure 2 représente un outil de test permettant la réalisation de tests de tenue en charge sur une configuration de serveurs de référence sur laquelle est déployée une application logicielle et pouvant être utilisé comme composant du système de détermination de la figure 1 ;
- les figures 3A et 3B illustrent, pour deux services distincts d'une application logicielle, un cycle d'activité de cette application logicielle et la répartition des nombres d'arrivées de requêtes sur ce cycle d'activité ;
- la figure 4 représente, sous forme d'ordinogramme, les principales étapes d'un procédé de détermination conforme à l'invention dans un mode particulier de réalisation dans lequel il est mis en œuvre par le système de détermination de la figure 1 ;
- les figures 5A à 5D illustrent l'évolution du temps de réponse d'un serveur en fonction de la charge de l'application logicielle, et pour différents profils de requêtes ;
- les figures 6A à 6D illustrent plusieurs modèles de systèmes de files d'attente pouvant être utilisés pour déterminer la configuration initiale selon l'invention ;
- la figure 7 détaille, sous forme d'ordinogramme, les principales étapes mises en œuvre par le système de détermination de la figure 1 dans un mode particulier de l'invention pour déterminer la configuration de serveurs cible à partir de la configuration initiale ;
- les figures 8A à 8D illustrent différents cas de figure conduisant à un ajustement des ressources de la configuration initiale conformément à l'invention ;

- la figure 9 représente, sous forme d'ordinogramme, les principales étapes d'un procédé de gestion de l'élasticité conforme à l'invention, dans un mode particulier de réalisation ; et
- la figure 10 illustre le traitement mis en œuvre dans un mode particulier de réalisation de l'invention lorsque l'application logicielle présente un cycle d'activité périodique.

Description détaillée de l'invention

La **figure 1** représente, dans son environnement, un système 1 de détermination d'une configuration de machines pour le déploiement d'une application logicielle APP, conforme à l'invention, dans un mode particulier de réalisation. Le système 1 propose, par le biais de la configuration ainsi déterminée (désignée par configuration cible dans la suite de la description), un dimensionnement des ressources nécessaire à la mise en production de l'application logicielle APP, autrement dit à son déploiement dans un environnement réel d'exécution.

Aucune limitation n'est attachée à la nature de l'application logicielle APP. Il peut s'agir d'une application permettant l'accès à ou la génération de contenus multimédia, le référencement de produits, l'accès à des équipements distants, etc.

Cette application logicielle APP est apte à offrir un ou plusieurs services UC1,UC2,...,UCN (N entier supérieur ou égal à 1) à des utilisateurs, sur réception de requêtes invoquant ces services émises par ces derniers via des terminaux (ex. via un terminal mobile, un ordinateur, etc.).

Dans l'exemple envisagé ici, l'application logicielle APP est destinée à être déployée (i.e. mise en production) dans un système informatique en nuage ou cloud (non représenté), sur une configuration cible de machines (serveurs) virtuelles hébergées par le cloud. Dans un tel contexte, la gestion de l'élasticité des ressources mises à disposition par le cloud pour l'exécution des applications logicielles qu'il héberge est très importante : le cloud cherche en effet à optimiser le partage des ressources communes aux applications logicielles qu'il héberge en vue de réduire notamment le coût de leur déploiement, tout en garantissant la qualité des services offerts par ces applications à leurs utilisateurs. Les ressources partagées peuvent être de différentes natures : il peut s'agir notamment de ressources de type processeurs ou CPU de machines (serveurs) virtuelles et/ou physiques, de ressources mémoires (ex. RAM, disque, etc.), ou encore de ressources réseaux (ex. connecteurs réseaux, etc.).

La gestion de l'élasticité dans le cloud a donc pour fonction de fournir aux différentes applications logicielles les ressources précitées nécessaires à leur exécution et de garantir leur bon fonctionnement selon un accord de niveau de service (ou SLA) convenu au préalable. Chaque SLA définit des contraintes spécifiques à l'application logicielle auquel il se rapporte, notamment en termes de volumétrie, de disponibilité, de temps de réponse, de fiabilité mais également de coût, de placement des machines (virtuelles dans le contexte du cloud) sur lesquelles sont déployées l'application, de droits d'utilisation et d'abonnements, etc. Autrement dit, chaque SLA associé à une application logicielle définit un certain nombre d'exigences qui se doivent d'être respectées par le cloud. La gestion de l'élasticité a pour rôle de déterminer les ressources nécessaires au

déploiement de l'application logicielle APP dans le cloud afin de répondre aux exigences fixées par ce SLA et ce, tout en prenant en compte les ressources engagées pour d'autres applications logicielles hébergées par le cloud et en cours d'exécution.

Dans l'exemple envisagé ici, la gestion de l'élasticité du cloud hébergeant l'application logicielle APP s'appuie sur le système de détermination 1. Plus particulièrement, celui-ci permet une gestion efficace de l'élasticité du cloud en offrant la possibilité de procéder à un pré-provisionnement des ressources nécessaires à l'exécution de l'application APP avant sa mise en production via la détermination d'une configuration adaptée au déploiement de l'application APP dans un environnement réel d'exécution (aussi désignée par configuration « cible »).

A cet effet, conformément à l'invention, le système de détermination 1 s'appuie sur différents environnements de tests de l'application logicielle APP dans lesquels l'application est déployée sur différentes configurations de serveurs, et sur différents modules (logiciels ici) mettant en œuvre des algorithmes ou des modèles de calcul programmés exploitant les résultats obtenus dans ces environnements de tests.

Plus précisément, dans le mode de réalisation décrit ici, le système de détermination 1 comprend :

- un outil logiciel (ou environnement) de test ENV1, construit pour réaliser des tests unitaires sur l'application logicielle APP lorsque celle-ci est déployée sur une première configuration de serveurs CONFIG1 dite de référence, constituée d'un nombre N1 de serveurs reliés entre eux (N1 désignant un entier supérieur ou égal à 1) ;
- un outil logiciel (environnement) de test ENV2, construit pour réaliser des tests de tenue en charge de l'application logicielle APP lorsque celle-ci est déployée sur une deuxième configuration de serveurs CONFIG2 dite initiale, constituée d'un nombre N2 de serveurs reliés entre eux (N2 désignant un entier supérieur ou égal à 1), et tenant compte de contraintes de déploiement de l'application logicielle APP lors de sa mise en production ;
- un module 2 de récolte de diverses métriques lors des tests unitaires réalisés dans l'environnement de test ENV1, apte à évaluer à partir de ces métriques une consommation des ressources par les serveurs de la configuration CONFIG1 pour le traitement de requêtes invoquant des services offerts par l'application logicielle APP. Plus précisément, le module 2 est apte à obtenir ici, à partir des métriques récoltées, un temps d'exécution moyen d'une requête invoquant un service offert par l'application logicielle APP pour chaque serveur de la configuration de référence CONFIG1 et pour chaque service offert par l'application logicielle APP. Il est également configuré pour estimer à partir des temps d'exécution moyens obtenus, une charge théorique maximale $\lambda max1$ pouvant être supportée par la configuration de référence CONFIG1 ;
- un module 3 de sélection de la configuration de serveurs initiale CONFIG2 sur laquelle sont réalisés les tests de tenue en charge via l'environnement de test ENV2 et qui sert de point de départ au dimensionnement des ressources nécessaires pour la mise en production de

l'application logicielle APP. Cette configuration CONFIG2 est sélectionnée par le module 3 de sorte à supporter une charge cible déterminée λtarg. Pour déterminer la configuration CONFIG2, le module 3 est configuré de sorte à tenir compte des temps d'exécution moyens et de la charge théorique maximale λmax1 déterminés par le module 2, et à utiliser un modèle numérique reflétant les contraintes de déploiement de l'application logicielle en environnement réel d'exécution (ex. types de machines utilisées, déploiement multi-sites, connexions réseau requises, etc.). Le module 3 est également configuré pour déterminer une charge maximale théorique λmax2 pour la configuration initiale CONFIG2 ainsi sélectionnée ; et

— un module 4 d'ajustement des ressources et des paramètres des serveurs de la configuration CONFIG2, configuré pour déterminer à partir de la configuration initiale CONFIG2 une configuration de serveurs cible désignée par CONFIG_TARG pour la mise en production de l'application logicielle APP. Cette configuration cible est déterminée par le module 4 en réalisant plusieurs tests en tenue de charge sur la configuration initiale CONFIG2 grâce à l'outil de test ENV2 : le module 4 réajuste le cas échéant à l'issue de chacun des tests, en fonction des résultats obtenus (en terme de temps de réponse moyen notamment de l'application logicielle sur la configuration testée), les ressources et les paramètres des serveurs de la configuration initiale CONFIG2 jusqu'à déterminer une configuration qui permette de garantir un temps de réponse maximal TRmax fixé pour l'application logicielle APP et qui supporte la charge cible λtarg.

Dans le mode de réalisation décrit ici, par souci de simplification, la configuration de référence CONFIG1 est constituée d'un serveur unique (i.e. N1=1), par exemple, le serveur S11 sur lequel a été déployée l'application APP. La généralisation de l'invention à une configuration de référence plus complexe comprenant une pluralité de serveurs reliés entre eux et sur lesquels sont déployées différentes unités de déploiement de l'application logicielle APP est évoquée ultérieurement.

L'outil de test ENV1 est un environnement logiciel de test similaire à ceux construits classiquement pour le débogage des applications logicielles et connus de l'homme du métier. Il n'est par conséquent pas décrit en détail ici.

Cet environnement ENV1 permet au module 2 de récolter diverses métriques, pour chaque type de requêtes identifié comme pertinent pour les services offerts par l'application logicielle APP, chaque type de requête considéré étant associé à un unique service offert par l'application. Ainsi, dans l'exemple envisagé ici, le module 2 récolte à partir des tests unitaires menés dans l'environnement de test ENV1 une métrique représentative de la consommation moyenne de CPU par l'application logicielle APP sur le serveur S11 de la configuration de référence CONFIG1 pour le traitement d'une requête de chaque type de requêtes retenu, autrement dit pour chaque service. Cette métrique est définie ici comme le temps d'exécution moyen d'une requête du service envisagé par le serveur de la configuration de référence CONFIG1 (autrement dit le temps de traitement moyen d'une telle requête par le serveur ou encore le temps de réponse moyen du

serveur à une telle requête). Le temps d'exécution d'une requête par un serveur est défini comme la somme des durées d'exécution des processus de l'application logicielle APP (ou encore des tâches ou fils, plus communément désignés par « threads » en anglais) associés à la requête après/avant des entrées/sorties.

En variante, le module 2 peut récolter une métrique représentative d'un taux moyen d'utilisation de CPU sur le serveur considéré, le taux d'utilisation de CPU étant défini comme le pourcentage de temps où le ou les processeurs (CPU) du serveur est (sont) actif(s) par rapport à la durée d'observation lors du traitement de la requête. Cette métrique peut également être évaluée à partir du temps moyen d'exécution d'une requête par le serveur considéré, comme détaillé ultérieurement.

Dans le mode de réalisation décrit ici, on se limite à des métriques relatives à l'utilisation du ou des processeurs (CPU) des serveurs et au dimensionnement de ces processeurs pour déterminer la configuration cible CONFIG_TARG. Cette ressource est en effet la plus critique et la plus sensible lorsque l'on envisage une gestion dynamique des ressources dans un système de type cloud comparativement par exemple à la mémoire.

Toutefois, cette hypothèse n'est pas limitative et le dimensionnement d'autres ressources peut être envisagé grâce à l'invention, comme notamment le dimensionnement de ressources mémoire (ex. disque, RAM) et/ou réseaux. A cet effet, d'autres métriques peuvent être obtenues lors des tests unitaires réalisés dans l'outil de test ENV1 comme notamment :

- une métrique représentative de la consommation de RAM, définie à partir du volume de RAM utilisé pour le traitement de la requête ;
- des métriques représentatives d'une consommation de disque, définies à partir de la durée et du volume de disque occupé lors du traitement de la requête ;
- une métrique représentative d'une consommation réseau définie comme la durée s'écoulant entre la fin d'exécution de la requête sur un serveur et le début d'exécution de la requête sur le serveur suivant (qui pourrait être le même serveur) ;
- etc.

Les tests unitaires réalisés avec l'outil de test ENV1 s'appuient sur des scenarios (c'est-à-dire des cas d'utilisation de l'application) consistant à émettre chaque requête après la réception et le traitement de la requête précédente de sorte à ce qu'il n'y ait pas d'accès concurrent aux ressources (CPU ici) dont on cherche à quantifier la consommation. Chaque scénario s'ensuit du déroulement séquentiel ou en parallèle des fonctions composant l'application logicielle (aussi désigné par flot ou graphe d'exécution ou encore « workflow » en anglais).

En variante, ces scenarios peuvent consister à émettre un nombre entier K de requêtes de façon déterministe ou périodique si le temps de traitement d'une requête n'est pas significatif, et sans accès concurrent aux ressources CPU.

Les requêtes considérées dans les scenarios mis en œuvre par l'outil de test ENV1 correspondent à des requêtes considérées comme pertinentes au regard des services offerts par

l'application logicielle APP. La description des services offerts par l'application logicielle APP et des types de requêtes invoquant ces services est obtenue par exemple de l'administrateur et/ou le développeur de l'application logicielle APP, comme indiqué plus en détail ultérieurement.

Les tests unitaires déroulés dans l'environnement de test ENV1 permettent donc de quantifier une consommation de ressources au niveau de chaque serveur de la configuration de référence considérée pour le traitement des requêtes mais sans accès concurrent à ces ressources : ils ne fournissent par conséquent aucune information pertinente à proprement parler en termes de temps de réponse des serveurs dans un environnement réel d'exécution. Ces informations de temps de réponse, dont la connaissance est utilisée ici pour dimensionner les ressources de la configuration de production de l'application logicielle APP pour atteindre une charge cible λtarg donnée (c'est-à-dire une volumétrie imposée à l'application logicielle APP), sont obtenues dans un second temps grâce à l'outil de test ENV2 qui permet de réaliser des tests de tenue en charge sur une autre configuration de serveurs (i.e. la configuration initiale CONFIG2) construite par le module 3 à partir des résultats de tests unitaires obtenus par le module 2.

Dans le mode de réalisation décrit ici, la configuration de serveurs initiale CONFIG2 est choisie par le module 3 de sorte à être représentative des différentes contraintes et fonctionnalités attendues lors de la mise en production de l'application logicielle, et notamment de la complexité logicielle, technique et fonctionnelle de l'environnement d'exécution réel de l'application logicielle (aussi appelé environnement de production). Ainsi notamment, si l'on envisage un déploiement multi-sites de l'application logicielle, la configuration de serveurs initiale CONFIG2 doit comprendre une pluralité de nœuds matérialisant la composante « multi-sites », un nœud central orchestrant les données véhiculées par la pluralité de nœuds, et modéliser les communications entre les différents nœuds sur lesquels s'appuie l'application logicielle. La configuration de serveurs initiale peut toutefois comprendre un nombre N2 de serveurs réduit par rapport au nombre de sites sur lesquels on envisage de déployer réellement l'application en vue de limiter notamment le coût, la complexité et le temps des tests à réaliser. De façon similaire, si lors de la mise en production de l'application logicielle on envisage une fonctionnalité de type pare-feu, cette fonctionnalité doit être reflétée au niveau de la configuration de serveurs initiale. Le type de machines utilisées (ex. mono ou multiprocesseurs, capacités maximales des machines, etc.) peut également être une contrainte imposée par l'environnement de production et qui doit être reflétée dans la configuration initiale.

Ces différentes contraintes imposées par l'environnement de production sont matérialisées conformément à l'invention par un modèle numérique MOD, qui s'appuie ici sur le formalisme des files d'attentes ou des réseaux de files d'attente. Différents exemples de modèles numériques sont illustrés ultérieurement. Le module 3 sélectionne alors la configuration initiale CONFIG2 en utilisant ce modèle numérique MOD et en tenant compte d'une part, des résultats des tests unitaires réalisés sur la configuration de référence CONFIG1 (et notamment des temps d'exécution moyens obtenus pour chaque serveur de la configuration de référence pour chaque

type de requête considéré (i.e. pour chaque service), et de la charge maximale théorique) qui servent de référence pour dimensionner les ressources de la configuration initiale CONFIG2, et d'autre part, de la charge cible λtarg fixée pour l'application logicielle APP (c'est-à-dire de la volumétrie imposée à l'application logicielle APP).

La configuration initiale CONFIG2 sélectionnée par le module 3 est donc un échantillon réduit de la configuration de serveurs sur laquelle est destinée à être mise en production l'application logicielle APP. Cette configuration initiale CONFIG2 est utilisée conformément à l'invention comme point de départ par le module 4 du système de détermination 1 pour le dimensionnement des ressources nécessaires à l'exécution de l'application logicielle APP en environnement de production.

Plus précisément, le module 4 est configuré pour procéder à un ajustement des ressources de la configuration initiale CONFIG2 en fonction de résultats de tests de tenue en charge réalisés à l'aide de l'outil de test ENV2 en vue de déterminer une configuration de serveurs « cible » CONFIG_TARG pour déployer l'application logicielle APP dans un environnement de production. Ces tests permettent d'évaluer la capacité de traitements (ex. taux d'utilisation des CPU ou de la mémoire, temps de réponse, etc.) des serveurs de la configuration CONFIG2 sur laquelle est déployée l'application logicielle APP, pour différentes valeurs de débit de requêtes (i.e. différentes charges de l'application logicielle) avec des flux de requêtes utilisateurs simulés représentatifs d'une utilisation réelle de l'application.

A cet effet, l'outil de test ENV2 tient compte des instances de déploiement de l'application logicielle APP sur les serveurs de la configuration CONFIG2, de même que de la topologie reliant les instances, du périmètre de l'application logicielle APP par rapport à son environnement (i.e. applications ou produits logiciels tiers, terminaux, etc.), des protocoles de communication mis en œuvre, etc. Il tient compte également des caractéristiques logicielles et matérielles de la configuration CONFIG2, et notamment du paramétrage des machines physiques et des machines (serveurs) virtuelles, des espaces de stockage, des aspects réseaux, et du paramétrage des produits intermédiaires (ou « middleware » en anglais) comme les serveurs d'application, les bases de données, les bus et les protocoles de communication, etc.

La **figure 2** illustre schématiquement un exemple d'un tel environnement de test ENV2ex élaboré pour tester une application logicielle déployée sur une configuration de serveurs comprenant un serveur de présentation 6, un serveur d'application 7 et un serveur de données 8. Une instance de l'application logicielle est installée sur le serveur d'application 7.

Dans l'exemple illustré à la figure 2, des requêtes HTTP (HyperText Transfer Protocol) sont envoyées par un dispositif d'injection de charge 9 à l'application logicielle APP pour traitement. Les serveurs 6, 7 et 8 sont reliés à une zone de stockage 10, via le protocole NFS (Network File System). L'application logicielle APP interagit avec des dispositifs tiers, tels que par exemple un serveur d'une zone d'échange mutualisée 11 et un serveur rebond 13, et différents systèmes externes 12 et 14 modélisés à l'aide de simulateurs partenaires.

De manière générale, pour réaliser des tests en tenue de charge de l'application logicielle APP, l'outil de test ENV2 comprend notamment :

— des modules ou dispositifs d'injection de charge, aptes à générer et à fournir à l'application logicielle APP différents niveaux de charge générés par des utilisateurs virtuels (chaque niveau de charge correspondant à un nombre d'utilisateurs virtuels différents), et à mesurer des temps de réponse moyens ou médians aux requêtes émises par ces utilisateurs virtuels. Ces modules d'injection de charge sont ici des modules CLIF s'appuyant sur un modèle de composante fractale et décrits notamment dans le document de B. Dillenseger intitulé « CLIF, a framework based on Fractal for flexible, distributed load testing », Annales des Télécommunications,vol. 64, n°1-2, février 2009 ; et

— des sondes permettant d'observer différentes métriques de consommation de ressources (CPU, mémoire, réseaux, etc.) par l'application logicielle sur les serveurs de la configuration testée.

L'outil de test ENV2 est configuré pour injecter, via ses modules d'injection de charge, de façon automatique, sur une durée fixe et selon une politique d'injection prédéterminée décrite plus en détail ultérieurement, différents niveaux de charge (i.e. différents nombres de requêtes par unité de temps croissants) à la configuration de serveurs testée et supportant l'application logicielle APP. L'outil de test ENV2 est programmé à cet effet pour observer, après injection de chaque niveau de charge, le comportement de la configuration de serveurs testée, et décider en fonction de ce comportement du prochain niveau de charge à injecter. Le processus est réitéré jusqu'à l'atteinte d'un critère prédéterminé.

L'homme du métier est invité à se référer au document de N. Salmi et al. intitulé « Model-based performance anticipation in multi-tiers autonomic systems : methodology and experiments », International Journal on Advances in Networks and Services, vol 3 no 3 & 4, 2010 (http://www.iariajournals.org/networks_and_services/tocv3n34.html), section III, pour plus de détails sur l'automatisation du processus d'injection de charge mis en œuvre par l'outil de test ENV2.

Il convient toutefois de noter qu'à l'instar du mode de fonctionnement décrit dans le document de N. Salmi et al., l'outil de test ENV2 utilise, conformément à l'invention, des scénarios d'injection de charge s'appuyant sur des profils de requêtes utilisateurs représentatifs de l'utilisation réelle ou tout du moins attendue de l'application logicielle APP en environnement de production (ou s'en approchant le plus possible). De tels profils peuvent être obtenus par exemple du développeur et/ou de l'administrateur de l'application logicielle APP. Ils comprennent notamment des informations relatives à la fréquence d'arrivée des requêtes, et plus particulièrement :

— à la répartition des différents types de requêtes (i.e. proportion de chaque type de requêtes) ;

— aux débits moyens d'arrivée des différents types de requêtes associés aux services offerts par l'application logicielle APP ; et

— aux distributions des durées (temps) d'inter-arrivées entre ces requêtes (ex. distributions déterministes ou de type rafales telles que par exemple une distribution aléatoire ou exponentielle).

Des informations relatives aux débits minimum et maximum peuvent également être contenues dans ces profils.

Ainsi, les requêtes injectées par l'outil de test ENV2 ne sont plus systématiquement espacées de façon déterministe (ex. périodique) ou successives comme pour les tests unitaires réalisés par l'outil de test ENV1, mais des « grumeaux » de requêtes peuvent se produire et entraîner des concurrences d'accès à des ressources partagées, ce qui peut provoquer un effondrement momentané de l'application logicielle APP (i.e. l'application logicielle est saturée et ne répond plus). En particulier, si un goulot d'étranglement se forme sur un serveur (correspondant à un débit crête de requêtes très supérieur au débit moyen) et persiste, le risque d'écroulement de l'application logicielle peut devenir non négligeable et le temps de réponse de celle-ci se dégrader rapidement. Les comportements observés en termes de temps de réponse et de consommation de ressources de l'application logicielle à l'aide de cet outil de test ENV2 peuvent donc être très différents de ceux que l'on peut observer lors des tests unitaires réalisés à l'aide de l'outil de test ENV1.

On note que l'application logicielle APP peut connaître des variations d'utilisation reproductibles sur une fenêtre temporelle, communément appelée « cycle d'activité ». Dans ce cas, une connaissance détaillée des informations de débits et de distributions des temps d'inter-arrivées des requêtes sur différentes périodes de temps identifiées sur le cycle d'activité pour chaque service offert par l'application logicielle peut être envisagée. Le découpage en périodes de temps est réalisé préférentiellement de sorte à garantir que sur une période de temps donnée, le débit moyen des requêtes soit représentatif des valeurs réelles de débits observées. Ce découpage est alors pris en compte au niveau de l'outil de tests ENV2.

A titre d'exemple, les **figures 3A et 3B** illustrent, pour deux services distincts d'une application logicielle, la répartition des nombres d'arrivées de requêtes sur un cycle d'activité de 24h.

Sur la figure 3A, on peut distinguer 2 à 5 périodes de temps sur lesquelles les débits moyens sont assez représentatifs par rapport aux valeurs réelles de débits observées (et très différents du débit moyen global sur le cycle d'activité de 24h). Par exemple un découpage en deux périodes de temps peut consister en une première période de 00h à 8h environ et une seconde période de 8h à 00h.

Sur la figure 3B, un découpage en 4 périodes de temps peut être envisagé comme consistant en une première période entre 0h et 5h30, une deuxième période entre 5h30 et 10h, une troisième période entre 10h et 12h30 et une quatrième période entre 12h30 et 24h.

Les sondes de l'outil de test ENV2 sont configurées pour récolter diverses métriques, et notamment ici :

— le temps de réponse de bout en bout des requêtes utilisateur en fonction de débits d'arrivée moyens et de la distribution des inter-arrivées de requêtes utilisateur de l'application logicielle APP ; et

— pour chaque serveur de la configuration CONFIG2, le taux d'utilisation du ou des processeurs de ce serveur, ou le temps d'exécution de la requête par le serveur.

En variante, d'autres métriques peuvent être récoltées dans cet environnement de test et notamment pour chaque serveur, la distribution du nombre de connecteurs réseau utilisés, la distribution du volume de RAM utilisé, le taux d'occupation de RAM et/ou de pool de « threads » d'entrée/sortie de chaque serveur, etc.

Comme mentionné précédemment, le module 4 du système de détermination 1 est configuré pour procéder à un ajustement des ressources et des paramètres des serveurs de la configuration initiale CONFIG2 en fonction de résultats de tests de tenue en charge réalisés à l'aide de l'outil de test ENV2. Cet ajustement qui est réalisé en s'appuyant ici sur plusieurs tests de tenue en charge judicieusement paramétrés en termes de charge injectée pour limiter la complexité d'implémentation de ces tests, permet au module 4 de déterminer une configuration de serveurs « cible » (configuration CONFIG_TARG) pour le déploiement de l'application logicielle APP dans un environnement de production qui vérifie le temps de réponse maximal TRmax fixé pour l'application logicielle et supporte la charge cible λtarg.

Dans le mode de réalisation décrit ici, l'environnement de test ENV2 permet également de réaliser des tests d'endurance ou de disponibilité sur la configuration de serveurs CONFIG_TARG après sa détermination par le module 4. Ces tests sont destinés à valider le dimensionnement et le paramétrage des ressources réalisés à partir des tests unitaires et des tests de tenue en charge, ces derniers étant supposés correspondre à un déploiement réel mais à échelle réduite. Il s'agit, grâce aux tests d'endurance, de tester l'application logicielle et la configuration de serveurs cible CONFIG_TARG sur une durée suffisamment longue (typiquement plusieurs jours) afin de prendre en compte des événements rares ou dont la probabilité d'apparition est faible.

Les scenarios de tests considérés pour ces tests d'endurance consistent à répartir les requêtes utilisateur suivant les différents types de services invoqués, et à émettre ces requêtes selon une certaine charge estimée à partir des tests de tenue en charge pour la configuration cible CONFIG_TARG et selon des distributions d'inter-arrivées des requêtes prédéfinies : déterministe, aléatoire, en rafales, etc. Les métriques récoltées à l'issue de ces tests sont similaires à celles récoltées à l'issue des tests en tenue de charge (temps de réponse de bout en bout de l'application, taux d'utilisation de ressources (CPU, RAM, pool de threads, etc.) de chaque serveur, etc.).

Dans le mode de réalisation décrit ici, les outils de test ENV1 et ENV2 et les modules 2 à 4 du système de détermination 1 sont des modules logiciels définis à l'aide d'instructions de programmes d'ordinateur stockés en mémoire d'une ou de plusieurs machines physiques. Chacune

de ces machines a ici l'architecture matérielle d'un ordinateur et comprend notamment un processeur, une mémoire morte, une mémoire vive, une mémoire non volatile et un module de communication avec notamment d'autres machines ou ordinateurs. La mémoire morte de chaque machine constitue un support d'enregistrement conforme à l'invention, lisible par le processeur de la machine et sur lequel est enregistré un programme d'ordinateur conforme à l'invention comportant des instructions pour l'exécution d'une ou de plusieurs étapes du procédé de dimensionnement selon l'invention.

Nous allons maintenant décrire, en référence à la **figure 4**, les principales étapes d'un procédé de détermination selon l'invention, tel qu'il est mis en œuvre dans un mode particulier de réalisation par le système de détermination 1 illustré à la figure 1.

On suppose en préliminaire de la mise en œuvre de ce procédé qu'un certain nombre d'informations relatives à l'application logicielle APP et à son déploiement dans un environnement de production sont fournies au système de détermination 1, par exemple par l'administrateur ou le développeur de l'application logicielle. Ces informations sont stockées par exemple dans une mémoire non volatile du système de détermination 1.

Ainsi, notamment, est fourni au système de détermination 1 l'accord de niveau de service (SLA) défini pour l'application logicielle APP. Comme mentionné précédemment, ce SLA définit les exigences en termes de qualité des services utilisateurs UC1,...,UCN offerts par l'application logicielle APP, et en particulier, en termes de volumétrie (ex. charge cible λtarg), de disponibilité et de temps de réponse maximal TRmax attendu de bout en bout. On suppose ici que les services UC1,...,UCN sont des services considérés comme représentatifs en matière de consommation de ressources, c'est-à-dire dont la consommation de ressources pour la fourniture de ces services par l'application logicielle APP est considérée comme significative par l'administrateur ou développeur de l'application logicielle. Les services UC1,..,UCN dépendent bien entendu de l'application logicielle considérée.

Le système de détermination 1 dispose également comme mentionné précédemment, des profils des requêtes utilisateurs représentatifs de l'utilisation réelle ou tout du moins attendue de l'application logicielle APP en environnement de production (ou s'en approchant le plus possible). Via ces profils, le système de détermination 1 connait pour chacun des services UC1,...,UCN, un pourcentage (i.e. proportion) de requêtes utilisateurs invoquant ce service parmi les requêtes adressées à l'application logicielle APP. On note p_i, le pourcentage des requêtes utilisateurs destinées à l'application logicielle APP et associées au service UCi avec $\sum_{i=1}^{N} p\_i = 1$.

Ces requêtes utilisateurs peuvent être de nature (type) différente selon le service qu'elles invoquent. Elles déclenchent des séquences d'exécution de l'application logicielle et entrainent des consommations de ressources logicielles et matérielles. En associant chaque requête à un service, l'invention permet d'établir un modèle de consommation de ressources de

l'application logicielle en tenant compte des exigences utilisateur pour chacun de ces services. A titre illustratif, on peut avoir par exemple les types de requêtes suivants :

- pour le service UC1 : commandes/souscriptions ;
- pour le service UC2 : résiliations ;
- pour le service UC3 : consultations passant en commandes ;
- pour le service UC4 : traitements différés dans la nuit ;
- etc.

Si on désigne par λ le débit moyen des requêtes associées aux services principaux de l'application logicielle, le système de détermination 1 déduit de la connaissance de λ et de la répartition des requêtes par service, le débit moyen λi des requêtes associées au service UCi, i= 1, ..., N, en utilisant la relation :

$$\lambda i = p\_i \times \lambda$$

Les informations sur la répartition des requêtes sur les N services UC1,...,UCN, et sur les débits moyens λ_i sont données par les profils des requêtes utilisateurs de l'application logicielle APP stockés au niveau du système de détermination 1.

Chaque profil de requêtes associé à un service comprend également, comme évoqué précédemment, une information représentative de la distribution des temps d'inter-arrivées des requêtes invoquant ce service. Une telle information précise notamment si les temps d'inter-arrivées des requêtes pour un service donné suivent une distribution déterministe ou au contraire de type rafales, telle que par exemple une distribution aléatoire ou exponentielle, en spécifiant les paramètres d'une telle distribution pour qu'elle corresponde au délai moyen associé au service considéré (ex. durée « on » pendant laquelle l'application reçoit des requêtes et durée « off », etc.). Ces profils de requêtes sont utilisés dans les scénarios de test implémentés par les outils de tests ENV1 et ENV2.

Le système de détermination 1 dispose également d'informations sur les contraintes de déploiement imposées lors de la mise en production de l'application logicielle APP. Ces contraintes comprennent notamment le type d'architecture de déploiement envisagée (présentée par exemple sous forme d'un graphe de nœuds associés aux unités de déploiement de l'application APP). Elles peuvent porter également sur des caractéristiques des serveurs (ex. types de machines envisagées), par exemple sur la capacité et la possibilité d'extension de CPU ou de RAM d'un serveur, sur l'utilisation de serveurs mono ou multiprocesseurs ou encore d'un cluster de serveurs, sur le type de configuration des unités de déploiements de l'application APP, etc. Ces informations peuvent être notamment fournies dans l'accord SLA.

D'autres contraintes et/ou informations peuvent être fournies au système de détermination 1 comme par exemple des contraintes en matière de placement des serveurs, etc. Toutefois par souci de simplification ici, nous ne considérerons pas de telles contraintes additionnelles.

Conformément à l'invention, le système de détermination 1 permet de dimensionner les ressources nécessaires à l'exécution de l'application logicielle APP sur une configuration de serveurs, en vue de son déploiement dans un environnement réel d'exécution, la configuration de serveurs dûment dimensionnée étant apte à vérifier le temps de réponse global maximal TRmax et à supporter la charge cible λtarg.

A cet effet, le système de détermination 1 s'appuie sur des tests unitaires réalisés sur une configuration de serveurs « minimale » de référence (à savoir la configuration CONFIG1) pour obtenir une estimation de la consommation de ressources de l'application logicielle APP. Cette consommation de ressources est caractérisée ici :

— d'une part, par les temps d'exécution moyens d'une requête de l'application logicielle sur chacun des serveurs $S11, \ldots, S1\mathrm{N1}$ de la configuration CONFIG1 sur lesquels sont déployées les unités de déploiement de l'application logicielle APP. Ces temps d'exécution moyens sont désignés dans la suite de la description par $E(S1\mathrm{j}),\ \mathrm{j} = 1, \ldots, \mathrm{N1}$; et

— d'autre part, par la charge maximale λmax1 pouvant être supportée par l'application logicielle APP1 dans la configuration de serveurs CONFIG1 (autrement dit le débit maximum de l'application logicielle APP dans la configuration de serveurs CONFIG1).

Le choix de la configuration CONFIG1 a déjà été détaillé précédemment. Le système de détermination 1 dispose donc, en préalable de l'exécution des tests unitaires, d'une description de la configuration de serveurs CONFIG1 et des caractéristiques des serveurs impliqués dans cette configuration (étape E10). Comme mentionné précédemment, à titre illustratif, on choisit ici une configuration CONFIG1 minimale constituée d'un unique serveur mono-processeur désigné par S11 (N1=1). Toutefois cette hypothèse n'est en aucun cas limitative, la configuration de référence CONFIG1 pouvant comprendre plusieurs serveurs reliés entre eux.

Conformément à l'invention, le système de détermination 1 via son outil de test ENV1 réalise des tests unitaires sur la configuration de serveurs CONFIG1 à partir de scénarios définis pour chaque type de requête identifié comme pertinent pour les services UC1,...,UCN offerts par l'application logicielle APP (étape E20). Comme mentionné précédemment, à l'issue de ces tests unitaires, le module 2 du système de détermination 1 récolte des métriques quantifiant la consommation de ressources par l'application logicielle sur le serveur S11 lors du traitement des requêtes de l'App. Dans les scénarios de test mis en œuvre, les requêtes sont envoyées à l'application logicielle de façon déterministe de sorte à ne pas avoir d'accès concurrents aux ressources (ex. CPU ici) dont on cherche à évaluer la consommation à l'aide des métriques récoltées.

Dans l'exemple envisagé ici, les métriques obtenues par le module 2 durant les tests unitaires correspondent pour chaque type de requête associé à un service UCi offert par l'application logicielle, à la consommation de CPU notée E(UCi,S11) par l'application logicielle sur le serveur S11 pour exécuter une requête de ce type. Chaque métrique E(UCi,S11) est plus précisément ici une mesure du temps d'exécution moyen d'une requête associée au service UCi sur

le serveur S11 (ou de façon équivalente, une mesure du taux d'utilisation moyen du CPU du serveur S11 pour traiter cette requête).

Le module 2 évalue ensuite (étape E30), à partir des métriques E(UCi,S11), i=1,...,N, récoltées pour chaque type de requête (autrement dit pour chaque service), un temps d'exécution moyen E(S11) d'une requête par l'application logicielle APP sur le serveur S11 (i.e. tous services confondus) suivant la relation :

$$E(S11) = \sum_{i=1}^{N} p\_i \times E(UCi, S11)$$

Puis le module 2 évalue à partir du temps d'exécution moyen E(S11), le taux d'utilisation $\mu(S11)$ du serveur S11. Pour un serveur S11 monoprocesseur exécutant une unique instance de l'application APP, le module 2 utilise à cet effet la relation suivante :

$$\mu(S11) = \frac{1}{E(S11)}$$

On note que lorsque la configuration CONFIG1 comprend plusieurs serveurs distincts S1j, j=1,...,N1 avec N1>1 (autrement dit, l'application logicielle APP est composée de plusieurs unités de déploiement distribuées sur différents serveurs), un temps d'exécution moyen E(S1j) d'une requête par l'application logicielle APP est évalué par le module 2 pour chaque serveur S1j, j=1,...,N1 de la configuration CONFIG1 à partir des métriques récoltées lors des tests unitaires menés dans l'environnement de test ENV1. De même, un taux d'utilisation (ou taux de service) moyen $\mu(S1j)$ est dérivé pour chaque serveur S1j, j=1,...,N1 à partir de ce temps d'exécution moyen de façon similaire à celle décrite précédemment.

A partir de ces taux d'utilisation du (ou des serveurs) de la configuration CONFIG1, le module 2 estime ici la charge maximale théorique λmax1 pour la configuration de référence CONFIG1 (étape E40). A cet effet, l'invention s'appuie sur une condition dite de stabilité de l'application logicielle selon laquelle pour que l'application logicielle APP soit stable, il faut que le taux d'arrivée λ des requêtes sur un serveur (qui définit ici la charge de l'application logicielle) soit strictement inférieur au taux d'utilisation de ce serveur. Autrement dit dans l'exemple du serveur unique S11 de la configuration CONFIG1 :

$$\frac{\lambda}{\mu(S11)} < 1 \Leftrightarrow \lambda < \frac{1}{E(S11)} \quad (1)$$

Cette hypothèse de stabilité de l'application logicielle APP et la relation (1) qui en découle permettent d'estimer une borne supérieure λmax1 de la charge théorique maximale de l'application logicielle dans la configuration de référence CONFIG1 à partir des taux d'utilisation des serveurs de la configuration CONFIG1.

Dans le mode de réalisation décrit ici, par souci de simplification, le module 2 utilise comme borne supérieure λmax1 de la charge théorique maximale :

$$\lambda max1 = \frac{1}{E(S11)} \quad (2)$$

Toutefois, cette hypothèse n'est pas limitative et dans un autre mode de réalisation, le module 2 peut prendre comme borne supérieure λmax1 de la charge théorique maximale une valeur $\frac{1}{E(S11)} - \epsilon$ avec ϵ nombre réel strictement positif.

De manière générale, si la configuration CONFIG1 comprend N1 serveurs S11, ..., SN1, en appliquant l'hypothèse de stabilité de l'application logicielle APP, on obtient :

$$\frac{\lambda(S1j)}{\mu(S1j)} < 1, \text{ pour tout j } = 1, \ldots, \mathrm{N1} \qquad (3)$$

où $\lambda(S1j)$ représente le taux d'arrivée (charge) de requêtes de l'application logicielle APP sur le serveur $\mathrm{S1j}$, et $\mu(S1j)$ représente le taux d'utilisation du serveur $\mathrm{S1j}$, j= 1, ..., N1. Ces taux $\lambda(S1j)$ et $\mu(S1j)$ peuvent être obtenus par le module 2 en utilisant les équations suivantes :

$$\begin{cases} \lambda(S1j) = \sum_{i=1}^{N} \beta_{ij} \times \lambda \mathrm{i} \\ \mu(S1j) = \frac{1}{E(S1j)} \\ E(S1j) = \sum_{i=1}^{N} p\_i \times E_i(S1j) \end{cases} \qquad (4)$$

avec $\beta_{ij} = 1$ si une requête du service UCi entraîne l'exécution d'un programme sur le serveur S1j, sinon $\beta_{ij} = 0$, et $\mathrm{E_i}(S1j)$ désigne le temps d'exécution moyen d'une requête du service UCi sur le serveur S1j.

L'inégalité (3) et les relations (4) permettent alors au module 2 d'estimer la borne supérieure λmax1 de la charge maximale théorique à partir de l'inégalité suivante :

$$\lambda \leq \frac{1}{\max \left\{ (\sum_{i=1}^{N} \beta_{ij} \times \mathrm{p\_i}) \times \mathrm{E}(S1j) \right\}_{j=1}^{N1}} \qquad (5)$$

Comme mentionné précédemment pour la configuration comprenant un unique serveur, par souci de simplification, le module 2 utilise ici comme borne supérieure :

$$\lambda\mathrm{max1} = \frac{1}{\max \left\{ (\sum_{i=1}^{N} \beta_{ij} \times \mathrm{p\_i}) \times \mathrm{E}(S1j) \right\}_{j=1}^{N1}} \qquad (6)$$

ou en variante :

$$\lambda\mathrm{max1} = \frac{1}{\max \left\{ (\sum_{i=1}^{N} \beta_{ij} \times \mathrm{p\_i}) \times \mathrm{E}(S1j) \right\}_{j=1}^{N1}} - \epsilon$$

avec ϵ nombre réel strictement positif.

La borne supérieure λmax1 ainsi déterminée à l'aide de l'égalité (2) ou (6) est utilisée par la suite par le module 2, et plus généralement par le système de détermination 1, comme estimation de la charge théorique maximale de l'application logicielle dans la configuration de référence CONFIG1.

On note que l'inégalité (5) ci-dessus met en évidence l'influence de la puissance et du nombre de serveurs dans la configuration de serveurs CONFIG1 sur la charge maximale admissible par l'application logicielle. Ainsi, au regard de cette inégalité, il convient, pour utiliser les serveurs S1j, j=1,...,N1 de la configuration CONFIG1 le plus équitablement possible, de s'assurer que les charges de ces serveurs S1j qui sont égales à $(\sum_{\mathrm{i=1}}^{\mathrm{N}} \beta_{\mathrm{ij}} \times \mathrm{p\_i}) \times \mathrm{E}(S1j)$, j=1,...,N1 soient les plus

proches possibles les unes des autres. De cette sorte, on peut optimiser le coût des serveurs dans le déploiement de l'application logicielle APP y compris sur la configuration de serveurs de référence CONFIG1.

La charge maximale théorique $\lambda max1$ estimée par le module 2 donne une borne supérieure pour les valeurs de débits de requêtes pouvant être supportées par l'application logicielle APP dans la configuration de serveurs de référence CONFIG1. Dans le mode de réalisation décrit ici, le module 2 détermine alors, à partir de cette borne supérieure, une valeur de débit (i.e. une charge) λ dite admissible pour la configuration CONFIG1 (étape E50) selon :

$$\lambda=\alpha\times\lambda max1 \qquad (6)$$

avec α nombre réel tel que $0<\alpha<1$.

Les figures **5A à 5D** illustrent l'impact du choix du paramètre α sur les performances de l'application logicielle APP.

La figure 5A représente schématiquement l'allure générale du temps de réponse moyen (exprimé en millisecondes) d'une application logicielle APP déployée sur un nombre NS de serveurs, $NS\geq1$, en fonction du débit moyen de requêtes injectées (exprimé en nombre de requêtes par seconde). Une telle figure peut être obtenue en réalisant des tests de tenue en charge par exemple à l'aide de l'environnement de test ENV2.

Sur cette figure, on peut identifier trois zones principales, notées respectivement Z1, ZOpt et Z2, et telles que :

- lorsque les valeurs de charge se situent dans la zone Z1, les temps de réponse moyens de l'application logicielle APP suivent un premier régime linéaire dans lequel tous les serveurs de la configuration considérée se comportent normalement et ont la possibilité de traiter davantage de requêtes. Autrement dit, la zone Z1 matérialise une zone où les serveurs de la configuration considérée sont sous-utilisés (zone de sous-utilisation des ressources) ;
- lorsque les valeurs de charge se situent dans la zone Z2, les temps de réponse moyens de l'application logicielle APP suivent un second régime linéaire dans lequel ils se dégradent plus rapidement que l'accroissement de la charge correspondant. Dans cette zone dite de congestion, une saturation des serveurs et de l'application logicielle peut se produire, autrement dit, certains au moins des serveurs de la configuration testée sont surchargés ;
- lorsque les valeurs de charge se situent dans la zone ZOpt, les temps de réponse moyens de l'application logicielle APP se trouvent au voisinage du changement de régimes. Ils doivent bien entendu restés bornés par la valeur TRmax du temps de réponse maximal jugé acceptable pour l'application logicielle. Dans cette zone, la capacité maximum de certains serveurs de la configuration de serveurs peut être atteinte.

Il convient de noter que la fin du premier régime linéaire (et donc la zone Z1) est beaucoup plus facile à caractériser que la valeur de charge marquant le début de la surcharge des serveurs de l'application logicielle (début de la zone Z2) car cette valeur est particulièrement instable. Les inventeurs ont constaté par expérience que la charge « limite » matérialisant la fin de la zone

optimale ZOpt peut être considérée comme matérialisant également le début de la zone Z2, du fait qu'au-delà de cette charge limite, la saturation des serveurs et de l'application logicielle devient imminente. Cette charge limite peut être considérée comme la valeur de charge correspondant au temps de réponse maximal TRmax jugé acceptable pour l'application logicielle APP. On peut par ailleurs considérer que la fin de la zone Z2 correspond à la charge théorique maximale $\lambda max1$ donnée par les bornes supérieures des inégalités (2) et (5).

On suppose ici que le paramètre α est choisi pour que la charge λ se trouve dans la zone ZOpt. Ceci permet d'optimiser l'utilisation des serveurs sur lesquelles est déployée l'application logicielle tout en respectant le temps de réponse maximal exigé.

Comme l'illustrent les figures 5B à 5D, on note que la distribution des temps d'inter-arrivées des requêtes a une influence sur la position du point d'intersection des régimes linéaires des zones Z1 et Z2. Les figures 5B à 5D représentent respectivement des résultats de trois séries de tests de tenue en charge réalisés sur une application logicielle déployée sur une configuration de serveurs, ces résultats illustrant les temps de réponse moyens obtenus en fonction des débits de requêtes moyens envoyés à l'application logicielle et en faisant varier les inter-arrivées des requêtes suivant une distribution déterministe (cf. figure 5B), une distribution exponentielle (figure 5C) ou une distribution en rafale (figure 5D). On peut constater sur ces figures que moins le trafic de requêtes est régulier (autrement dit, plus il s'éloigne d'une distribution déterministe), plus le changement de régimes apparait tôt en terme de charge. Sur cet exemple, on observe que, le paramètre α peut être choisi égal à environ 0.95 pour des inter-arrivées déterministes, à 0.80 pour des inter-arrivées exponentielles et à 0.50 ou 0.60 pour des inter-arrivées en rafale.

Au vu de ces constats, on suppose ici que le module 2 sélectionne une valeur du paramètre α comprise entre 0.80 et 0.95. Il obtient ainsi la valeur de charge admissible λ à partir de l'équation (6) (étape E50).

En variante, le module 2 sélectionne une valeur du paramètre α en tenant compte des distributions d'inter-arrivées des requêtes spécifiées dans les profils de requêtes associés aux différents services UC1,...,UCN, par exemple en se basant sur les valeurs précitées (0.95 pour des inter-arrivées déterministes, 0.80 pour des inter-arrivées exponentielles).

Les différents éléments estimés par le module 2 à partir de la configuration de référence CONFIG1 et des tests unitaires (i.e. charge maximale, charge admissible, etc.) servent alors conformément à l'invention de données de référence pour tout dimensionnement de serveurs dans un futur environnement de production de l'application logicielle APP dans lequel des exigences en termes de caractéristiques des serveurs et de volumétrie sont prédéfinies par le SLA. Le dimensionnement des serveurs de cet environnement de production de l'application logicielle APP se fait en deux temps par le système de détermination 1 :

(1) sélection par le module 3 d'une configuration de serveurs initiale (à savoir la configuration CONFIG2) pour le déploiement de l'application logicielle comprenant la détermination du nombre

de serveurs de cette configuration et de leurs paramètres en fonction de la volumétrie exigée dans le SLA (i.e. de la charge cible λtarg) (étape E60), et l'estimation de la charge théorique maximale pour cette configuration CONFIG2 (étape E70) ; et

(2) ajustement par le module 4 du nombre et des paramètres des serveurs de la configuration CONFIG2 pour satisfaire le temps de réponse maximal TRmax de bout en bout exigé dans le SLA (étape E80), et obtention de la configuration de serveurs cible CONFIG_TARG (étape E90).

Nous allons maintenant détailler davantage les étapes E60-E90 mises en œuvre par les modules 3 et 4 conduisant à l'obtention de la configuration de serveurs cible CONFIG_TARG.

Pour sélectionner la configuration initiale CONFIG2 (étape E60), le module 3 tient compte de la charge (volumétrie) cible λtarg et des caractéristiques des serveurs imposées par les contraintes de déploiement de l'application logicielle APP. Plus précisément, il s'appuie sur la charge maximale λmax1 et la charge admissible λ évaluées par le module 2 à partir des résultats des tests unitaires conduits sur la configuration de serveurs CONFIG1 et sur des règles de correspondance établies entre les paramètres des serveurs de la configuration CONFIG1 et les paramètres des serveurs de la configuration CONFIG2, pour le choix du nombre et des paramètres des serveurs à considérer dans la configuration CONFIG2. Ces règles de correspondance sont établies ici en s'appuyant sur la théorie des files d'attente (ou des réseaux de files d'attente). Elles permettent au module 3 de déterminer la puissance et le nombre de serveurs ou clusters de serveurs S2j, j=1,…,N2 à considérer dans la configuration initiale CONFIG2 en tenant compte de leurs caractéristiques (monoprocesseur, multiprocesseurs, etc.).

De façon connue, selon la théorie des files d'attente, chaque système de file d'attente est un formalisme composé d'une file d'attente et d'un serveur, et est caractérisé principalement par :

- un processus aléatoire d'arrivées de requêtes, défini par un débit d'arrivée des requêtes et une distribution des temps d'inter-arrivées des requêtes ;
- un processus de service défini par une distribution statistique du temps de service ;
- une politique d'ordonnancement des requêtes du serveur (ex. FIFO (First In First Out), PS, RR, etc.) ;
- un nombre de serveurs ; et
- une capacité (taille) de la file d'attente.

Dans le mode de réalisation décrit ici, on adopte cette modélisation par file d'attente en faisant les hypothèses suivantes pour l'application logicielle APP :

- un processeur (CPU) du serveur est modélisé par un serveur d'une file d'attente ;
- la durée d'exécution d'une requête correspond au temps de service ;
- le stockage/mémoire du serveur est modélisé par la file d'attente ; et
- les arrivées des requêtes sont modélisées par le processus d'arrivées.

Ces hypothèses permettent une modélisation simple d'une application logicielle s'exécutant sur un ou plusieurs serveurs monoprocesseur, multi-fils ou multiprocesseurs. Ainsi, à titre illustratif :

- la **figure 6A** représente une modélisation par un système de file d'attente d'une application logicielle s'exécutant sur un serveur monoprocesseur ;
- la **figure 6B** représente une modélisation par un système de file d'attente d'une application logicielle s'exécutant sur un serveur multiprocesseur comprenant m processeurs identiques ; et
- la **figure 6C** représente une modélisation par un système de file d'attente d'une application logicielle s'exécutant sur un cluster de n serveurs multiprocesseur identiques.

D'autres configurations peuvent bien entendu être envisagées. Notamment, les unités de déploiement de l'application logicielle peuvent être distribuées et déployées sur des serveurs ou clusters de serveurs reliés entre eux via un réseau de télécommunications (NW), et être modélisées alors par un réseau de files d'attente dont un exemple est illustré à la **figure 6D**.

Le module 3 associe donc un modèle de système ou de réseau de files d'attente à la configuration CONFIG2, en fonction des contraintes imposées pour le déploiement. Il convient de noter que cette association peut être réalisée en amont du procédé de détermination selon l'invention, par exemple dès lors que les contraintes de déploiement de l'application logicielle APP sont connues.

Puis, à partir de ce modèle, le module 3 établit une relation (c'est-à-dire une règle de correspondance) entre les temps d'exécution moyens des serveurs ou cluster(s) de serveurs de la configuration CONFIG2 avec ceux du ou des serveurs de la configuration CONFIG1. Pour les exemples de modèles envisagés sur les figures 6A à 6C et pour une configuration de référence CONFIG1 comprenant un unique serveur S11, le module 3 établit par exemple les relations suivantes (modèle numérique MOD au sens de l'invention) en tenant compte de la charge cible λtarg et de la charge admissible λ estimée par le module 2 pour la configuration de référence CONFIG1 :

- Pour une application logicielle s'exécutant sur une configuration CONFIG2 constituée d'un unique serveur S21 monoprocesseur (cf. modélisé sur la figure 6A), le serveur S21 doit être choisi tel que (étape E60) :

$$E(S21) = \frac{\lambda}{\lambda\text{targ}} \times E(S11)$$

où E(S21) désigne le temps d'exécution moyen d'une requête de l'application logicielle APP sur le serveur S21. Cela signifie par exemple, que si la charge cible $\lambda\text{targ} = 2 \times \lambda$, il faut que le serveur S21 de la configuration initiale CONFIG2 soit deux fois plus puissant en termes de CPU que le serveur S11 de la configuration de référence CONFIG1. Ainsi, la même requête pourra être exécutée par le serveur S21 en deux fois moins de temps, conduisant à un taux d'utilisation deux fois supérieur en termes de CPU.

A partir de cette relation et en faisant une hypothèse de stabilité de l'application logicielle APP telle qu'évoquée précédemment, le module 3 estime une charge maximale théorique λmax2 pour la configuration CONFIG2 qui est telle que (étape E70) :

$$\lambda\text{targ} < \lambda\text{max2}$$

avec

$$\lambda\text{max2} = \frac{1}{E(S21)}$$

— Pour une application logicielle s'exécutant sur une configuration CONFIG2 constituée d'un serveur S21 multiprocesseur ayant m processeurs identiques (cf. modélisé sur la figure 6B), le serveur S21 doit être choisi tel que (étape E60) :

$$\text{E(S21)}=\frac{E(S_{proc})}{m} \leq \frac{\lambda}{\lambda\text{targ}} \times E(S11)$$

où E(S21) désigne le temps d'exécution moyen d'une requête de l'application logicielle APP sur le serveur S21 et $E(S_{proc})$ désigne le temps d'exécution de la requête par un processeur.

Comme dans l'exemple ci-dessus, le module 3 estime alors une charge maximale théorique λmax2 pour la configuration CONFIG2 qui est dans ce cas telle que (étape E70) :

$$\lambda\text{targ} < \lambda\text{max2}$$

avec

$$\lambda\text{max2} = \frac{1}{E(S21)}=\frac{m}{E(S_{proc})}$$

— Pour une application logicielle s'exécutant sur une configuration CONFIG2 constituée d'un cluster de n serveurs S21 multiprocesseur identiques ayant m processeurs identiques (cf. modélisé sur la figure 6C), le cluster de serveurs S21 doit être choisi tel que (étape E60) :

$$\text{E(cluster)}=\frac{E(S_{proc})}{nm} \leq \frac{\lambda}{\lambda\text{targ}} \times E(S11)$$

où E(cluster) désigne le temps d'exécution moyen d'une requête de l'application logicielle APP sur le cluster de serveurs S21 et $E(S_{proc})$ désigne le temps d'exécution de la requête par un processeur d'un serveur S21, soit :

$$\text{E}(S_{proc}) \leq \frac{\lambda}{\lambda\text{targ}} \times E(S11) \times nm$$

Comme dans les deux exemples ci-dessus, le module 3 estime alors une charge maximale théorique λmax2 pour la configuration CONFIG2 qui est dans ce cas telle que (étape E70) :

$$\lambda\text{targ} < \lambda\text{max2}$$

avec

$$\lambda\text{max2} = \frac{1}{\text{E(cluster)}} = \frac{nm}{E(Sproc)}$$

D'autres modèles plus complexes peuvent bien entendu être envisagés pour modéliser la configuration de serveurs CONFIG2 en fonction des contraintes de déploiement imposées, à partir desquels sont dérivées des relations entre la charge cible λtarg, les temps d'exécution des

serveurs de la configuration CONFIG1 et les temps d'exécution des serveurs de la configuration CONFIG2.

Ainsi, par exemple, on considère une application logicielle APP offrant un service unique destinée à s'exécuter sur une configuration CONFIG2 constituée de plusieurs serveurs distribués S21, S22 et d'un cluster de serveurs comprenant n serveurs identiques S23, le serveur S21 étant un serveur multiprocesseur comprenant m processeurs identiques S_{proc}. Dans ce cas, le module 3 peut déterminer la puissance et le nombre de serveurs et/ou de processeurs de la configuration CONFIG2 à partir de résultats de tests unitaires réalisés sur une configuration de référence CONFIG1 implémentant trois unités de déploiement de l'application logicielle APP sur trois serveurs S11, S12 et S13 et en fonction de de la charge cible λtarg. Plus précisément, le module 3 choisit dans ce cas la puissance et le nombre de serveurs pour la configuration initiale CONFIG2 tels que (étape E60) :

$$\begin{cases} E(S21) = \dfrac{E(S_{proc})}{m} = \dfrac{\lambda}{\lambda\text{targ}} \times E(S11) \\ E(S22) = \dfrac{\lambda}{\lambda\text{targ}} \times E(S12) \\ E(cluster\ S23) = \dfrac{E(S23)}{n} = \dfrac{\lambda}{\lambda\text{targ}} \times E(S13) \end{cases}$$

Les mêmes notations qu'introduites précédemment sont utilisées.

Comme dans les trois exemples ci-dessus, le module 3 estime alors une charge maximale théorique pour la configuration CONFIG2 qui est dans ce cas telle que (étape E70) :

$$\lambda\text{targ} < \lambda\text{max2}$$

avec

$$\lambda\text{max2} = \min\left\{\frac{m}{E(S_{proc})}, \frac{1}{E(S22)}, \frac{n}{E(S23)}\right\}$$

Selon un autre exemple encore, on considère une application logicielle APP offrant une pluralité de services UC1,…,UCN, chaque service correspondant à un pourcentage de requêtes p_i, i=1,…,N, l'application logicielle étant destinée à s'exécuter sur une configuration CONFIG2 comprenant des serveurs ou clusters de serveurs distribués S21, S22 et S23, chacun de ces serveurs comprenant respectivement m1, m2 et m3 processeurs identiques notés S_{proc21}, S_{proc22} et S_{proc23}. La configuration CONFIG1 comprend trois serveurs S11, S12 et S13 implémentant trois unités de déploiement de l'application logicielle APP. Le module 3 choisit alors dans ce cas la puissance et le nombre de serveurs pour la configuration initiale CONFIG2 tels que (étape E60) :

$$\begin{cases} E(S21) = \dfrac{E(S_{proc21})}{m1} \leq \sum_{i=1}^{N} p\_i \times \dfrac{\lambda_i}{\lambda \mathrm{targ}_i} \times E_i(S11) \\ E(S22) = \dfrac{E(S_{proc22})}{m2} \leq \sum_{i=1}^{N} p\_i \times \dfrac{\lambda_i}{\lambda \mathrm{targ}_i} \times E_i(S12) \\ E(S23) = \dfrac{E(S_{proc23})}{m3} \leq \sum_{i=1}^{N} p\_i \times \dfrac{\lambda_i}{\lambda \mathrm{targ}_i} \times E_i(S13) \end{cases}$$

Les mêmes notations qu'introduites précédemment sont utilisées, les charges admissibles et cibles étant dans cet exemple déterminées par les tests unitaires sur les serveurs pour l'ensemble des services de l'application logicielle APP.

Comme dans les exemples ci-dessus, le module 3 estime alors une charge maximale théorique λmax2 pour la configuration CONFIG2 qui est dans ce cas telle que (étape E70) :

$$\lambda \mathrm{targ} < \lambda \mathrm{max2}$$

avec

$$\lambda \mathrm{max2} = \min\left\{\frac{m1}{E(S_{proc21})}, \frac{m2}{E(S_{proc22})}, \frac{m3}{E(S_{proc23})}\right\}.$$

Bien entendu, ces modèles ne sont donnés qu'à titre illustratif et ne sont en aucun cas exhaustifs.

La valeur de charge maximale théorique $\lambda \mathrm{max2}$ ainsi obtenue à l'étape E70 par le module d'estimation 3 pour la configuration initiale CONFIG2 sélectionnée à l'étape E60, et qui majore en particulier la charge cible $\lambda \mathrm{targ}$, est ensuite utilisée par le système de détermination 1, et plus particulièrement par le module 4, pour borner les tests de tenue en charge réalisés sur la configuration de serveurs initiale CONFIG2 à l'aide de l'outil de test ENV2.

Plus précisément, comme mentionné précédemment, le système de détermination 1 réalise ensuite, par l'intermédiaire de son outil de test ENV2 et de son module 4, des tests de tenue en charge sur la configuration de serveurs initiale CONFIG2 (étape E80). Ces tests ont pour objectif d'évaluer la capacité de traitements (ex. taux d'utilisation CPU, temps de réponse, etc.) des serveurs de l'application logicielle APP dans la configuration de serveurs CONFIG2 avec des flux de requêtes utilisateurs simulés représentatifs d'une utilisation réelle de l'application logicielle. Autrement dit, l'environnement de test ENV2 tient compte lors de ces tests non seulement du débit de requêtes demandé (et en particulier de la charge cible $\lambda \mathrm{targ}$ imposée par l'accord de niveau de service SLA), mais également de la variation des temps qui séparent deux arrivées de requêtes consécutives aussi appelé temps d'inter-arrivées des requêtes.

En effet, la configuration CONFIG2 sélectionnée par le module 3 est construite à partir de modèles numériques de dimensionnement basés uniquement sur le débit moyen des requêtes envoyées à l'application logicielle APP et sur des métriques de tests unitaires pour lesquelles il est accédé aux ressources des serveurs sans concurrence.

Or la variation des temps d'inter-arrivées des requêtes impacte de façon importante le temps de réponse de bout en bout de l'application logicielle comme mentionné précédemment, et

peut être source d'engorgement, voire d'arrêt de serveurs de l'application logicielle notamment si des grumeaux de requêtes se forment et persistent dans le temps lors de l'exécution de l'application logicielle.

Conformément à l'invention, la configuration initiale CONFIG2 sert donc avantageusement de point départ pour effectuer des tests de tenue en charge dans l'environnement ENV2 en s'appuyant sur des injections de requêtes proches du comportement des utilisateurs de l'application logicielle APP dans un environnement de production. Ce sont les résultats de ces tests qui permettent au module 4, au cours de l'étape E80, d'ajuster et/ou de réajuster le nombre et/ou les paramètres des serveurs, l'objectif étant d'obtenir une configuration de serveurs cible CONFIG_TARG destinée à la production (étape E90).

Autrement dit, durant ces tests de tenue en charge, contrairement aux tests unitaires précédemment réalisés sur la configuration de serveurs CONFIG1, les temps d'inter-arrivées des requêtes ne sont plus déterministes mais ils sont proches des distributions réellement rencontrées lors de l'utilisation de l'application logicielle dans un environnement réel d'exécution. Ces distributions ont été définies préalablement par l'administrateur de l'application logicielle et sont stockées comme mentionné précédemment en mémoire du système de détermination 1, dans des profils de requêtes utilisateurs définis pour chaque service UC1,...,UCN. Ainsi, en considérant de telles distributions réalistes, des grumeaux de requêtes peuvent se produire et entraîner des accès concurrents à des ressources partagées de la configuration CONFIG2, ce qui peut provoquer un effondrement momentané de l'application logicielle ou dégrader fortement son temps de réponse. Les comportements en termes de consommation de ressources de l'application logicielle APP sur les serveurs de la configuration CONFIG2 peuvent donc être très différents de ceux estimés précédemment par le module 3.

En particulier, la configuration CONFIG2 telle que sélectionnée par le module 3 pour supporter la charge cible λ_{targ} a été sélectionnée sans tenir compte des variations des temps d'inter-arrivées des requêtes de l'application logicielle APP. Rien ne garantit donc que cette charge cible se trouve en dehors de la zone de congestion Z2 décrite précédemment en référence à la figure 5A. Le module 4 ajuste et/ou réajuste la configuration de serveurs CONFIG2 au fil des tests de tenues en charge, autrement dit dimensionne les ressources de la configuration CONFIG2, de sorte à s'assurer que la charge cible λ_{targ} se trouve, pour la configuration ajustée, dans la zone Zopt, i.e. en dehors de la zone de congestion Z2. La limite supérieure en termes de charges de cette zone Zopt correspond à la charge conduisant au temps de réponse maximal TRmax fixé par l'accord SLA.

Dans le mode de réalisation décrit ici, pour réaliser ces ajustements au cours de l'étape E80, le module 4 du système de détermination 1 met en œuvre une politique d'injection de charges permettant de minimiser le nombre de tests réalisés dans l'environnement ENV2.

Plus précisément, les tests de tenue en charge sont réalisés en injectant en entrée de l'application logicielle et de la configuration CONFIG2 des charges selon un algorithme détaillé ci-

après. Ces charges sont inférieures en tout état de cause à la charge théorique maximale $\lambda\max 2$ déterminée par le module 3, cette charge $\lambda\max 2$ étant strictement supérieure à la charge cible λtarg. Ces injections de charge sont réalisées à l'aide des modules d'injection de charge de l'outil de test ENV2. Les temps d'inter-arrivées de requêtes injectées suivent une distribution proche d'une utilisation réelle de l'application logicielle et telle que spécifiée dans les profils de requêtes utilisateurs. Il convient de noter qu'avantageusement, l'invention ne requiert pas la connaissance des distributions de charge devant être injectées individuellement à chaque serveur de la configuration CONFIG2. L'invention se contente d'une connaissance globale des profils de requêtes pour l'application logicielle de bout en bout.

Pour chaque valeur de charge injectée lors de ces tests de tenue en charge, le module 4 obtient ici les métriques suivantes :

— temps de réponse de bout en bout de l'application logicielle APP ; et

— consommation des ressources (ex. CPU, RAM, pool, etc.) pour chacun des serveurs $S2j, j = 1, \ldots, N2$ de la configuration de serveurs testée (CONFIG2 ou CONFIG2 ajustée).

La **figure 7** illustre plus en détail, sous forme d'ordinogramme, l'algorithme mis en œuvre par le module 4 pour réaliser de manière optimisée au cours de l'étape E80 les tests de tenue en charge et l'ajustement des ressources de la configuration CONFIG2.

Conformément à cet algorithme, le module 4 choisit tout d'abord un débit de requêtes (charge) $\lambda test1$ tel que (étape E801) :

$$\lambda test1 = \min\left(\frac{\lambda\max 2}{2}, K1 \times \lambda\text{targ}\right)$$

où K1 désigne une constante réelle prédéterminée inférieure ou égale à 1. Par exemple K1=0.8.

Puis le module 4 réalise un premier test de tenue en charge sur la configuration initiale CONFIG2 issue de l'étape E60 en injectant via l'environnement de test ENV2 une charge égale à $\lambda test1$ (étape E802).

Le module 4 compare alors le temps de réponse TR($\lambda test1$) de bout en bout de l'application logicielle APP obtenu pour cette charge $\lambda test1$ par rapport au temps de réponse maximal attendu TRmax (étape test E803).

Si TR($\lambda test1$) est supérieur à TRmax (réponse oui à l'étape test E803), alors cela signifie qu'un ou plusieurs serveurs de la configuration CONFIG2 sont sous-dimensionnés et qu'il convient de procéder à un ajustement du nombre de serveurs et/ou de leurs paramètres et/ou ressources (étape E804). Cet ajustement est réalisé par le module 4 en tenant compte des consommations individuelles de chaque serveur de la configuration testée pour renforcer les ressources de ceux qui sont le plus chargés. La valeur du dépassement du temps de réponse TR($\lambda test1$) par rapport au temps de réponse maximal TRmax définit le ratio de cet ajustement.

A titre d'exemple, la **figure 8A** illustre un cas de dépassement de 25% du temps de réponse maximal TRmax, i.e.$(\text{TR}(\lambda test1)) - \text{TRmax}) / \text{TRmax}) = 25\%$. Dans ce cas, le module 4 applique une augmentation des ressources de la configuration CONFIG2 du même ordre, soit en

ajustant le nombre de serveurs de la configuration, soit en ajustant leurs paramètres (ex. puissance des processeurs, nombre de processeurs, etc.).

Lorsque le redimensionnement de la configuration CONFIG2 est terminé, les étapes précédentes sont réitérées sur la configuration ajustée (toujours notée CONFIG2 par souci de simplification), autrement dit, le module 4 reprend l'étape E802 et l'applique à la configuration ajustée.

Si TR($\lambda test1$) est inférieur ou égal à TRmax (réponse non à l'étape test E803), alors le module 4 sélectionne une nouvelle valeur de charge notée $\lambda test2$ telle que $\lambda test2 < \lambda test1$ (étape E805). Par exemple, le module 4 choisit ici une charge $\lambda test2 = K2 \times \lambda test1$ avec K2 constante réelle prédéterminée inférieure à 1. K2 est par exemple prise ici égale à 0.25. En variante, le module 4 choisit une charge $\lambda test2$ déterminée en fonction de la charge cible $\lambda targ$, par exemple égale à 10% de la charge cible λtarg.

Puis le module 4 réalise un deuxième test de tenue en charge sur la configuration CONFIG2, via l'environnement de test ENV2, en injectant la charge $\lambda test2$ à l'application logicielle APP (étape E806). Il obtient à l'issue de ce test le temps de réponse TR($\lambda test2$) de bout en bout de l'application logicielle APP.

Le module 4 détermine ensuite, à partir des charges $\lambda test1$ et $\lambda test2$ et des temps de réponse de l'application logicielle APP TR($\lambda test1$) et TR($\lambda test2$) obtenus pour ces charges, une estimation du temps de réponse de l'application logicielle pour la configuration de serveurs testée pour une charge égale à la charge cible λtarg (étape E807). Cette estimation est notée D(λtarg).

Dans le mode de réalisation décrit ici, cette estimation est obtenue par le module 4 en considérant la droite D passant par les points A1 et A2 de coordonnées respectives A1=($\lambda test1$,TR($\lambda test1$)) et A2=($\lambda test2$,TR($\lambda test2$)), et le point A de la droite D d'abscisse λtarg. Le module 4 prend comme estimation D(λtarg) l'ordonnée du point A. La **figure 8B** illustre cette droite sur un exemple ainsi que l'estimation D(λtarg) obtenue à partir de cette droite.

Il convient de noter que les constantes K1 et K2 sont choisies préférentiellement de sorte à obtenir deux points A1 et A2 suffisamment éloignés pour permettre la construction de la droite D. Les valeurs considérées dans l'exemple décrit ici, à savoir K1=0.8 et K2=0.25 permettent avantageusement d'obtenir un tel tracé, toutefois d'autres valeurs peuvent être envisagées en variante.

Puis le module 4 compare la valeur D(λtarg) au temps de réponse maximal TRmax spécifié dans l'accord SLA (étape test E808).

Si D(λtarg) est supérieur à TRmax (réponse oui à l'étape test E808), alors cela signifie qu'un ou plusieurs serveurs de la configuration CONFIG2 sont sous-dimensionnés et qu'il convient de procéder à un ajustement du nombre de serveurs et/ou de leurs paramètres et/ou ressources (étape E809). Cet ajustement est réalisé par le module 4, comme à l'étape E804, en tenant compte des consommations individuelles de chaque serveur de la configuration testée pour renforcer les

ressources de ceux qui sont le plus chargés. La valeur du dépassement du temps de réponse D(λtarg) par rapport au temps de réponse maximal TRmax définit le ratio de cet ajustement.

A titre d'exemple, la figure 8B illustre un cas de dépassement par D(λtarg) du temps de réponse maximal TRmax. Dans ce cas, le module 4 applique une augmentation des ressources de la configuration CONFIG2 du même ordre que $(\mathrm{D}(\lambda\mathrm{targ}) - \mathrm{TRmax}) / \mathrm{TRmax}$, soit en ajustant le nombre de serveurs de la configuration, soit en ajustant leurs paramètres (ex. puissance des processeurs, nombre de processeurs, etc.).

Lorsque le redimensionnement de la configuration CONFIG2 est terminé, les étapes précédentes sont réitérées sur la configuration ajustée (toujours notée CONFIG2 par souci de simplification), autrement dit, le module 4 reprend l'étape E802 et l'applique à la configuration ajustée.

Si au contraire D(λtarg) est inférieur ou égal à TRmax (réponse non à l'étape test E808), alors le module 4 examine si D(λtarg) est sensiblement inférieur à TRmax (étape test E810). Dans le mode de réalisation décrit ici, par sensiblement inférieur, on entend que D(λtarg) est inférieur à γ = 90% de la valeur de TRmax. Bien entendu, cette valeur γ prise égale à 90% ou de manière équivalente à 0.9 est paramétrable et n'est donné qu'à titre illustratif.

La **figure 8C** illustre un exemple où D(λtarg) est sensiblement inférieur à TRmax selon le critère précité.

Si un tel cas de figure se présente (réponse oui à l'étape test E810), cela signifie que la configuration CONFIG2 testée (qui correspond à la configuration CONFIG2 éventuellement ajustée et/ou réajustée) est surdimensionnée. Le module 4 procède donc à un (ré)ajustement des ressources de la configuration CONFIG2 en se basant comme à l'étape E809 sur la différence $(\mathrm{D}(\lambda\mathrm{targ}) - \mathrm{TRmax}) / \mathrm{TRmax}$ (étape E811).

Puis lorsque le redimensionnement de la configuration CONFIG2 est terminé, les étapes précédentes sont réitérées sur la configuration ajustée (toujours notée CONFIG2 par souci de simplification), autrement dit, le module 4 reprend l'étape E802 et l'applique à la configuration ajustée.

Sinon (réponse non à l'étape test E810), le module 4 sélectionne une nouvelle charge $\lambda test3$ égale cette fois-ci à la charge cible λtarg (étape E812) et réalise un nouveau test de tenue en charge sur la configuration CONFIG2 en injectant, via l'environnement de test ENV2, à l'application logicielle APP déployée sur cette configuration la charge $\lambda test3$ (étape E813).

Le module 4 compare ensuite le temps de réponse TR($\lambda test3$) de bout en bout de l'application logicielle APP obtenu pour cette charge $\lambda test3$ par rapport au temps de réponse maximal attendu TRmax (étape test E814).

Si TR($\lambda test3$) est supérieur à TRmax (réponse oui à l'étape test E814) comme illustré par exemple à la **figure 8D**, alors cela signifie qu'un ou plusieurs serveurs de la configuration CONFIG2 sont sous-dimensionnés et qu'il convient de procéder à un ajustement du nombre de serveurs et/ou de leurs paramètres et/ou ressources (étape E815). Cet ajustement est réalisé par

le module 4 de façon similaire aux étapes E804, E809, E811 en tenant compte des consommations individuelles de chaque serveur de la configuration testée pour renforcer les ressources de ceux qui sont le plus chargés. La valeur du dépassement du temps de réponse TR($\lambda test3$) par rapport au temps de réponse maximal TRmax définit le ratio de cet ajustement, de façon similaire à ce qui est fait aux étapes d'ajustement E804, E809 et E811.

Puis lorsque le redimensionnement de la configuration CONFIG2 est terminé, les étapes précédentes sont réitérées sur la configuration ajustée (toujours notée CONFIG2 par souci de simplification), autrement dit, le module 4 reprend l'étape E802 et l'applique à la configuration ajustée.

Sinon (réponse non à l'étape test E814), la configuration CONFIG2 est considérée comme correctement dimensionnée pour le déploiement de l'application logicielle APP dans un environnement de production (étape E816). La configuration ainsi ajustée constitue la configuration cible CONFIG_TARG obtenue par le système de détermination 1 et considérée pour la mise en production de l'application logicielle (étape E90).

L'algorithme mis en œuvre par le module 4 au cours de l'étape E80 et représenté à la figure 7 permet de réaliser un compromis entre le nombre de tests de tenue en charge effectués et l'ajustement de la configuration CONFIG2. Cet algorithme permet avantageusement de valider au plus tôt la configuration de serveurs sur laquelle déployer l'application logicielle APP tout en s'abstenant de réaliser un trop grand nombre de tests de tenue en charge du fait d'un incrément de charge entre chaque test trop faible. Cet algorithme converge très rapidement vers une configuration de production grâce aux informations acquises des modules 2 et 3 et en tirant parti des écarts de temps de réponse mesurés par rapport au temps maximal de réponse TRmax pour quantifier chaque ajustement de la configuration de serveurs testée.

Dans le mode de réalisation décrit ici, à la suite du dimensionnement (potentiellement itératif) des serveurs de la configuration CONFIG2 par le module 4 résultant en l'obtention de la configuration de serveurs cible CONFIG_TARG pour la mise en production de l'application logicielle, le système de détermination 1 valide la configuration obtenue en réalisant un test d'endurance sur celle-ci (étape E100). Ces tests d'endurance sont réalisés à l'aide de l'outil de test ENV2 sur une durée d'exécution prédéterminée suffisamment longue (ex. plusieurs jours) afin de prendre en compte des événements rares dont la probabilité d'apparition est faible. Les requêtes utilisateur sont réparties suivant les services UC1,…,UCN et émises avec une certaine charge admissible et des temps d'inter-arrivées suivant des distributions prédéfinies (ex. déterministe, aléatoire, en rafale, etc.).

Les métriques observées durant ce test sont similaires à celles observées durant les tests en tenue de charge (temps de réponse de bout en bout des requêtes utilisateur, temps d'exécution moyen sur chaque serveur ou taux d'utilisation du CPU, etc.), et permettent de valider la configuration CONFIG_TARG et le dimensionnement de ses ressources ou de procéder si besoin à de nouveaux ajustements de façon similaire à ce qui a été fait durant l'étape E80 (étape E110).

A l'issue de l'étape E110, le système de dimensionnement 1 offre donc une configuration de serveurs CONFIG_TARG *a priori* dimensionnée de sorte à vérifier les contraintes imposées par l'accord SLA, c'est-à-dire dimensionnée pour supporter, dans un environnement réel de production, la charge cible λtarg tout en respectant le temps de réponse maximal TRmax.

Il convient de noter que la technique de pré-dimensionnement proposée par l'invention s'appuie sur la prise en compte de profils de requêtes réalistes et proches de l'utilisation réelle de l'application logicielle. Or, il est parfois difficile de disposer d'informations précises sur ces distributions pour des applications logicielles réparties dans des environnements de type cloud. En effet, les délais qui séparent deux arrivées consécutives de requêtes à l'entrée d'un serveur sont perturbés par des délais de réseaux traversés dans le cloud et de traitements d'autres serveurs, et subissent des gigues aléatoires. Des accumulations de requêtes en attente de traitement à l'entrée d'un serveur peuvent se produire et impacter significativement le temps de réponse. Par ailleurs, comme tout système logiciel a une capacité limitée d'un point de vue matériel, un écroulement du système peut se produire lorsque la quantité de traitements dépasse momentanément la capacité du serveur bien que le débit moyen soit respecté.

Par conséquent, dans un tel contexte, la configuration cible obtenue CONFIG_TARG à l'issue de l'étape E110 est préférentiellement considérée comme une configuration de déploiement initial permettant de satisfaire les exigences spécifiées dans le SLA, dans un procédé de gestion d'élasticité selon l'invention.

La **figure 9** représente, sous forme d'ordinogramme, dans un mode particulier de réalisation, les principales étapes d'un procédé de gestion d'élasticité selon l'invention qui utilise comme configuration de déploiement initiale, la configuration cible CONFIG_TARG déterminée par le système de détermination 1 (étape F10).

La configuration cible initiale CONFIG_TARG est ensuite déployée en production et l'application logicielle APP est exécutée sur la configuration cible (étape F20). Cette exécution est accompagnée de mécanismes de surveillance de l'exécution de l'application logicielle à l'aide de métriques de supervision connues en soi (étape F30), et de mécanismes de maintien, ajout ou de retrait de ressources en fonction des métriques de supervision et de règles prédéterminées (étape F40), de sorte à absorber des rafales momentanées durant l'exécution de l'application et garantir des temps de réponse conformes au SLA. Un tel mécanisme est décrit par exemple dans le document de L. Letondeur et al. intitulé « Planification pour la gestion automatique de l'élasticité d'applications dans le cloud », ComPAS'2014, 23-25 avril 2014.

Ce procédé de gestion d'élasticité peut ainsi être aisément mis en œuvre par un système de gestion d'élasticité conforme à l'invention (non représenté sur les figures) comprenant :

— le système 1 de détermination, qui détermine une configuration cible de serveurs pour le déploiement de l'application logicielle via l'exécution du procédé de détermination selon l'invention ;

— un module de déclenchement d'une exécution de l'application logicielle sur la configuration cible de serveurs ;
— un module de surveillance d'au moins une métrique de supervision de cette configuration cible lors de l'exécution de l'application logicielle ; et
— un module d'ajustement tel que celui décrit dans le document précidé de L. Letondeur et al. et configuré pour ajuster les ressources allouées à la configuration cible en fonction de ladite au moins une métrique, ce module d'ajustement étant apte à maintenir les ressources allouées ou à ajouter et/ou retirer dynamiquement des ressources à la configuration cible.

Dans le mode de réalisation et les exemples décrits précédemment, on a considéré que le débit moyen d'arrivée des requêtes de l'application logicielle APP était représentatif des valeurs de débit réellement rencontrées lors de l'exécution de l'application logicielle. Toutefois, comme déjà évoqué précédemment et illustré aux figures 3A et 3B, il se peut que pour certaines applications logicielles, le débit d'arrivée des requêtes de services varie significativement au cours du temps et ce de façon répétitive, définissant ainsi des cycles d'activités de l'application logicielle. Dans un tel contexte, le débit moyen d'arrivée des requêtes de l'application logicielle sur chaque cycle d'activité est assez peu significatif par rapport aux valeurs de débits réelles sur les différentes périodes composant le cycle d'activité.

Pour gérer une telle situation, dans un mode particulier de réalisation de l'invention, le cycle d'activité de durée T de l'application logicielle APP est préalablement découpé pour chaque service UCi, i=1,...,N en une pluralité Li d'intervalles de temps [Tj]_i, j=1,...,Li tels que les débits moyens correspondants, notés λ(UCi, [Tj]_i), soient suffisamment représentatifs des valeurs réelles de débit rencontrées sur les intervalles de temps [Tj]_i (c'est-à-dire typiquement tels que l'écart-type des débits moyens sur chaque intervalle soient faible, par exemple inférieur à un seuil prédéfini). Par ailleurs, les intervalles de temps [Tj]_i, j=1,...,Li sont définis ici pour chaque service i=1,...,N tels que :

$$T = [T1]\_i \cup [T2]\_i \cup \ldots \cup [TLi]\_i$$

Le nombre d'intervalles Li considéré pour le découpage d'un cycle d'activité peut varier d'un service à l'autre ou être identique pour tout ou partie des services UCi, i=1,...,N considérés.

Pour mieux illustrer le traitement effectué par le système de détermination 1 dans ce mode de réalisation, on considère ci-après, en référence à la **figure 10**, un exemple simple d'une application logicielle offrant deux services principaux UC1 et UC2 définis dans un accord SLA et caractérisés, sur un cycle d'activité de durée T, par le profil de requêtes suivant :

— pour le service UC1 : les arrivées des requêtes relatives au service UC1 sont réparties sur L1=3 intervalles de temps [T1]_1, [T2]_1 et [T3]_1 tels que $T = [T1]\_1 \cup [T2]\_1 \cup [T3]\_1$ et avec des débits moyens respectifs sur chacun de ces intervalles notés $\lambda 11 = \lambda(UC1, [T1]\_1)$, $\lambda 12 = \lambda(UC1, [T2]\_1)$ et $\lambda 13 = \lambda(UC1, [T3]\_1)$;

— pour le service UC2 : les arrivées des requêtes relatives au service UC2 sont réparties sur L2=4 intervalles de temps [T1]_2, [T2]_2, [T3]_2 et [T4]_2 tels que T = [T1]_2 ∪ [T2]_2 ∪ [T3]_2 ∪ [T4]_2, et avec des débits moyens respectifs sur ces intervalles notés λ21=λ(UC2, [T1]_2), $\lambda 22 = \lambda(\mathrm{UC2}, [\mathrm{T2}]\_2)$, λ23=λ(UC2, [T3]_2) et λ24=λ(UC2, [T4]_2).

On considère par ailleurs dans cet exemple N1=4 serveurs S11, S12, S13 et S14 dans la configuration de référence CONFIG1, les trois serveurs S11, S12, S13 étant impliqués dans l'exécution du service UC1 et les deux serveurs S11 et S14 étant impliqués dans l'exécution du service UC2.

Au regard de ces hypothèses, le cycle d'activité de l'application logicielle APP peut être décomposé, selon un nouveau découpage commun aux deux services UC1 et UC2 dérivé du découpage pour chacun des services, en L=6 intervalles de temps [T1], [T2], [T3], [T4], [T5] et [T6] comme illustré à la figure 10, sur lesquels les débits λ[Ti], i=1,...,6 d'arrivée de requêtes de l'application logicielle APP sont représentatifs et donnés par les relations (7) suivantes :

λ[T1]= λ11+λ21

λ[T2]= λ11+λ22

λ[T3]= λ12+λ22

λ[T4]= λ12+λ23

λ[T5]= λ13+λ24

λ[T6]= λ13+λ24

Pour chaque intervalle de temps $[Ti]$, i=1,...,6, la répartition des requêtes de l'application logicielle entre les deux services peut être déduite comme suit :

$$\begin{cases} p_1([Ti]) = \frac{\lambda 11}{\lambda[Ti]}, \ i = 1 \ et \ 2 \\ p_1([Ti]) = \frac{\lambda 12}{\lambda[Ti]}, \ i = 3 \ et \ 4 \\ p_1([Ti]) = \frac{\lambda_{13}}{\lambda[Ti]}, \ i = 5 \ et \ 6 \end{cases}$$

$$\begin{cases} p_2([T1]) = \frac{\lambda 21}{\lambda[T1]} \\ p_2([Ti]) = \frac{\lambda 22}{\lambda[\mathrm{Ti}]}, \ i = 2 \ et \ 3 \\ p_2([Ti]) = \frac{\lambda 23}{\lambda[\mathrm{Ti}]}, \ i = 4 \ et \ 5 \\ p_2([T6]) = \frac{\lambda 24}{\lambda[\mathrm{T6}]} \end{cases}$$

Ainsi on peut en déduire, pour chaque intervalle de temps [Tl], le taux des arrivées de requêtes de l'App :

$\lambda(\mathrm{UCi}, [Tl]) = p_i([Tl]) \times \lambda[Tl\ ]$ pour i=1,...,N et l=1,...,L

Les étapes du procédé de détermination telles que décrites précédemment sont ensuite appliquées par le système de détermination sur chacun des six intervalles [T1],...,[TL], avec L=6.

Autrement dit, pour chaque intervalle de temps [Tl], l=1,...,L, le système de détermination 1 effectue les étapes suivantes :

— tests unitaires via l'environnement de test ENV1 et détermination par le module 2, pour chaque serveur S1j, j=1...,N1=4 de la configuration de référence CONFIG1, du débit d'arrivée moyen des requêtes $\lambda(S1j)[Tl]$ de l'application logicielle APP et du temps d'exécution moyen d'une requête $E(S1j)[Tl]$ sur le serveur $S1j$ sur l'intervalle de temps [Tl], tous services confondus.

Dans l'exemple envisagé ici, on obtient en particulier, à partir des relations (7) indiquées ci-dessus appliquées individuellement pour chaque serveur de la configuration de référence CONFIG1, et en fonction de l'implication de chaque serveur de la configuration de référence dans l'exécution des services UC1 et UC2 :

- pour le serveur S11 impliqué dans l'exécution des deux services UC1 et UC2 :

$$\lambda(S11)[T1]= \lambda 11 + \lambda 21$$
$$\lambda(S11)[T2]= \lambda 11 + \lambda 22$$
$$\lambda(S11)[T3]= \lambda 12 + \lambda 22$$
$$\lambda(S11)[T4]= \lambda 12 + \lambda 23$$
$$\lambda(S11)[T5]= \lambda 13 + \lambda 23$$
$$\lambda(S11)[T6]= \lambda 13 + \lambda 24$$

et

$$E(S11)[Tl]= p1([Tl])\times E(UC1,S11)+ p2([Tl])\times E(UC2,S11),\ l=1,2,3,4,5,6.$$

- pour les serveurs S12 et S13 impliqués dans l'exécution du service UC1 seulement :

$$\lambda(S12/S13)[T1]= \lambda 11(S12/S13)$$
$$\lambda(S12/S13)[T2]= \lambda 11(S12/S13)$$
$$\lambda(S12/S13)[T3]= \lambda 12(S12/S13)$$
$$\lambda(S12/S13)[T4]= \lambda 12(S12/S13)$$
$$\lambda(S12/S13)[T5]= \lambda 13(S12/S13)$$
$$\lambda(S12/S13)[T6]= \lambda 13(S12/S13)$$

et

$$E(S12/S13)[Tl]= p1([Tl])\times E(UC1,S12/S13),\ l=1,2,3,4,5,6.$$

la notation S1i/S1j signifiant que les égalités s'appliquent aussi bien au serveur S1i qu'au serveur S1j ; et

- pour le serveur S14 impliqué dans l'exécution du service UC2 seulement :

$$\lambda(S14)[T1]= \lambda 21$$
$$\lambda(S14)[T2]= \lambda 22$$

$$\lambda(S14)[T3]= \lambda 22$$

$$\lambda(S14)[T4]= \lambda 23$$

$$\lambda(S14)[T5]= \lambda 23$$

$$\lambda(S14)[T6]= \lambda 24$$

et

$$E(S14)[Tl]= p2([Tl])\times E(UC2,S14),\ l=1,2,3,4,5,6.$$

— détermination par le module 2, en appliquant la condition de stabilité de l'application logicielle APP, d'une estimation $\lambda 1\max[\mathrm{Tl}]$ de la charge maximale théorique telle que :

$$\lambda 1\ \max[\mathrm{Tl}] = \frac{1}{\max\left\{\left(\sum_{i=1}^{N}\beta_{ij}\times \mathrm{pi}([\mathrm{Tl}])\right)\times \mathrm{E}(S1j)[Tl]\right\}_{j=1}^{4}}$$

Il convient de noter que certaines intervalles peuvent être fusionnés avec leurs intervalles voisins notamment lorsque :

— la durée de l'intervalle $[\mathrm{Tl}]$ est inférieure à un seuil, par exemple correspondant au temps de mise en place d'un serveur virtuel ; et/ou

— les débits et les temps d'exécution moyens sur deux intervalles consécutifs ont des valeurs proches.

Suite aux estimations réalisées par le module 2, et à cette phase de révision des intervalles Tl le cas échéant, les étapes E60 à E110 décrites précédemment sont mises en œuvre par les modules 3 et 4 individuellement pour chacun des intervalles de temps (éventuellement révisés)

On note que l'invention a été décrite dans un contexte de type cloud pour le dimensionnement de machines virtuelles permettant de supporter une application logicielle. Toutefois, l'invention s'applique à d'autres types d'environnement et au dimensionnement de machines matérielles et/ou virtuelles.

REVENDICATIONS

1. Procédé de détermination d'une configuration de serveurs dite cible pour un déploiement d'une application logicielle (APP) apte à offrir au moins un service, le procédé comprenant :

— une étape d'obtention (E20), pour au moins un service offert par l'application logicielle, au moyen d'au moins un test unitaire réalisé sur une configuration de serveurs dite de référence (CONFIG1) apte à exécuter l'application logicielle (APP), d'un temps d'exécution moyen d'une requête invoquant ce service pour chaque serveur (S1j) de la configuration de référence ;

— une étape de sélection (E60) d'une configuration de serveurs dite initiale (CONFIG2) pour le déploiement de l'application logicielle apte à supporter une charge cible (λtarg) déterminée pour l'application logicielle, cette étape de sélection tenant compte des temps d'exécution moyens obtenus lors dudit au moins un test unitaire, de la charge cible et d'une charge admissible déterminée (λ) pour la configuration de référence, et utilisant un modèle numérique reflétant au moins une contrainte de déploiement de l'application logicielle imposée à la configuration de serveurs initiale ; et

— une étape de détermination (E90), à partir de la configuration initiale, d'une configuration de serveurs cible (CONFIG_TARG) destinée à être utilisée pour le déploiement de l'application logicielle, ladite étape de détermination comprenant la réalisation (E80) d'une pluralité de tests de tenue en charge paramétrés en fonction d'une charge maximale théorique (λmax2) estimée pour la configuration initiale et de la charge cible, et utilisant au moins un profil de requêtes invoquant ledit au moins un service offert par l'application logicielle, ce profil étant représentatif d'une utilisation de l'application logicielle lors de son déploiement, ladite étape de détermination comprenant en outre, en fonction du résultat de chaque test de tenue en charge, un ajustement (E80) d'au moins une ressource de la configuration initiale de sorte que la configuration cible obtenue à l'issue de ladite pluralité de tests de tenue en charge soit apte à vérifier un temps de réponse maximal fixé pour l'application logicielle et à supporter ladite charge cible.

2. Procédé selon la revendication 1 comprenant en outre une étape d'estimation (E40) d'une charge maximale théorique (λmax1) pour la configuration de référence et dans lequel la charge admissible (λ) déterminée pour la configuration de référence résulte du produit d'un paramètre α compris entre 0 et 1 par la charge maximale théorique estimée pour la configuration de référence.

3. Procédé selon la revendication 2 dans lequel la charge maximale théorique (λmax1) pour la configuration de référence est estimée en appliquant une condition de stabilité de

l'application logicielle à au moins un temps d'exécution moyen d'une requête de l'application logicielle dérivé pour ledit au moins un serveur de la configuration de référence à partir des temps d'exécution moyens obtenus pour ce serveur pour les services offerts par l'application logicielle.

4. Procédé selon l'une quelconque des revendications 1 à 3 dans lequel au cours de l'étape de détermination, l'ajustement (E804,E809,E811,E815) d'au moins une ressource de la configuration initiale à l'issue d'un test de tenue en charge est réalisé en fonction d'une différence entre un temps de réponse de l'application logicielle évalué lors du test de tenue en charge et le temps de réponse maximal (TRmax) fixé pour l'application logicielle.

5. Procédé selon l'une quelconque des revendications 1 à 4 dans lequel l'étape de détermination comprend au moins :

— un premier test de tenue en charge (E802) réalisé avec une première charge inférieure à une valeur minimum entre une moitié de la charge maximale théorique estimée pour la configuration initiale et le produit de la charge cible par un nombre réel prédéterminé inférieur ou égal à 1 ;
— un deuxième test de tenue en charge (E806) réalisé avec une deuxième charge inférieure à la première charge ; et
— un troisième test de tenue en charge (E813) réalisé avec une troisième charge égale à la charge cible.

6. Procédé selon la revendication 5 dans lequel l'étape de détermination comprend en outre une estimation (E807) à l'issue du deuxième test, d'un temps de réponse de l'application logicielle avec la charge cible à partir d'un temps de réponse évalué à l'issue du premier test réalisé avec la première charge sur une configuration testée dérivée de la configuration de serveur initiale et d'un temps de réponse de l'application logicielle évalué à l'issue du deuxième test réalisé avec la deuxième charge sur ladite configuration testée, l'ajustement de ressources étant réalisé à l'issue du deuxième test en fonction d'une différence entre le temps de réponse estimé pour la charge cible et le temps de réponse maximal fixé pour l'application logicielle.

7. Procédé selon la revendication 6 dans lequel, à l'issue du deuxième test :

— la configuration de serveurs testée est considérée comme sous-dimensionnée si le temps de réponse estimé pour la charge cible est supérieur au temps de réponse maximal ; et/ou
— la configuration de serveurs testée est considérée comme surdimensionnée si le temps de réponse estimé pour la charge cible est inférieur au produit d'un nombre réel prédéterminé γ compris entre 0 et 1 et du temps de réponse maximal ; et/ou
— la configuration de serveurs testée est considérée comme correctement dimensionnée sinon.

8. Procédé selon la revendication 7 dans lequel γ est pris égal à 0.9.

9. Procédé selon l'une quelconque des revendications 5 à 8 dans lequel la configuration cible correspond à une configuration de serveurs testée pour laquelle à l'issue du troisième test, un temps de réponse de l'application logicielle évalué pour la charge cible sur cette configuration de serveurs testée est inférieur au temps de réponse maximal.

10. Procédé selon l'une quelconque des revendications 1 à 9 comprenant en outre une étape de validation (E100) de la configuration cible au moyen d'un test d'endurance réalisé pendant une durée d'exécution prédéterminée de l'application logicielle.

11. Procédé selon l'une quelconque des revendications 1 à 10 dans lequel le modèle numérique modélise ladite au moins une contrainte de déploiement de l'application logicielle à partir d'au moins une file d'attente ou d'un réseau de files d'attente.

12. Procédé selon l'une quelconque des revendications 1 à 11 dans lequel au moins un profil de requêtes comprend, pour au moins une période de temps déterminée d'un cycle d'activité de l'application logicielle :

- un débit moyen des requêtes invoquant ce service sur ladite période de temps ;
- une proportion de requêtes invoquant ce service parmi les requêtes invoquant des services offerts par l'application logicielle ; et
- une distribution des durées d'inter-arrivées entre deux requêtes invoquant ce service sur ladite période de temps.

13. Procédé selon la revendication 1 à 12 dans lequel ladite application logicielle est caractérisée par un cycle d'activité comprenant une pluralité d'intervalles, chaque intervalle étant associé à un débit moyen d'arrivée des requêtes représentatif sur cet intervalle, et dans lequel lesdites étapes d'obtention, de sélection et de détermination sont mises en œuvre pour au moins un intervalle de ladite pluralité d'intervalles.

14. Procédé de gestion d'élasticité d'une configuration de serveurs apte à exécuter une application logicielle, ce procédé comprenant :

- une étape de détermination (F10) d'une configuration cible de serveurs pour le déploiement de l'application logicielle comprenant l'exécution d'un procédé de détermination selon l'une quelconque des revendications 1 à 13 ; et
- une étape d'exécution (F20) de l'application logicielle sur ladite configuration cible de serveurs comprenant :

- une surveillance (F30) d'au moins une métrique de supervision de cette configuration cible ; et
- en fonction de ladite au moins une métrique, un ajustement (F40) des ressources de la configuration cible, cet ajustement comprenant un maintien et/ou un ajout et/ou un retrait dynamique de ressources à la configuration cible.

15. Système de détermination (1) d'une configuration cible pour un déploiement d'une application logicielle, le système comprenant :

— un premier outil de test (ENV1) permettant l'exécution d'au moins un test unitaire sur l'application logicielle lorsque celle-ci est exécutée sur une configuration de serveurs dite de référence ;

— un module d'obtention (2), configuré pour commander l'exécution d'au moins un test unitaire par le premier outil de test sur ladite configuration de référence et pour obtenir, pour au moins un service offert par l'application logicielle, un temps d'exécution moyen d'une requête invoquant ce service pour chaque serveur (S1j) de la configuration de référence ;

— un module de sélection (3), configuré pour sélectionner une configuration de serveurs initiale (CONFIG2) pour le déploiement de l'application logicielle apte à supporter une charge cible déterminée pour l'application logicielle, ledit module de sélection étant configuré pour tenir compte des temps d'exécution moyens obtenus lors dudit au moins un test unitaire, de la charge cible et d'une charge admissible déterminée pour la configuration de référence, et pour utiliser un modèle numérique reflétant au moins une contrainte de déploiement de l'application logicielle imposée à la configuration de serveurs initiale ;

— un second outil de test (ENV2) permettant l'exécution de tests de tenue en charge sur ladite application logicielle lorsque celle-ci est exécutée sur la configuration de serveurs initiale sélectionnée par le module de sélection ; et

— un module de détermination (4) configuré pour déterminer à partir de la configuration initiale une configuration de serveurs cible destinée à être utilisée pour le déploiement de l'application logicielle, ledit module de détermination étant configuré pour commander l'exécution d'une pluralité de tests de tenue en charge par le second outil de test, lesdits tests de tenue en charge étant paramétrés par le module de détermination en fonction d'une charge maximale théorique estimée pour la configuration initiale et de la charge cible, et utilisant au moins un profil de requêtes invoquant ledit au moins un service offert par l'application logicielle, ce profil étant représentatif d'une utilisation de l'application logicielle lors de son déploiement, ledit module de détermination étant en outre configuré pour ajuster, en fonction du résultat de chaque test de tenue en charge, au moins une ressource de la configuration initiale de sorte que la configuration cible obtenue à l'issue de ladite pluralité de tests de tenue en charge soit

apte à vérifier un temps de réponse maximal fixé pour l'application logicielle et à supporter ladite charge cible.

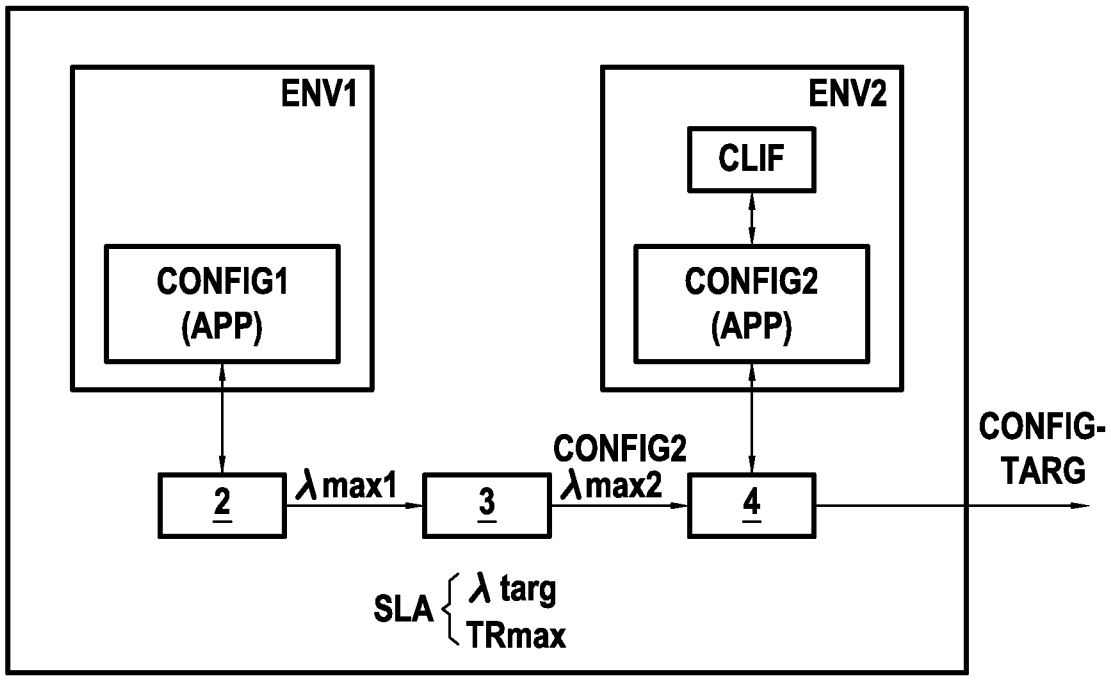

ENV1
CONFIG1
(APP)
ENV2
CLIF
CONFIG2
(APP)
2
λmax1
3
CONFIG2
λmax2
4
CONFIG-
TARG
SLA
λ targ
TRmax
1

FIG.1

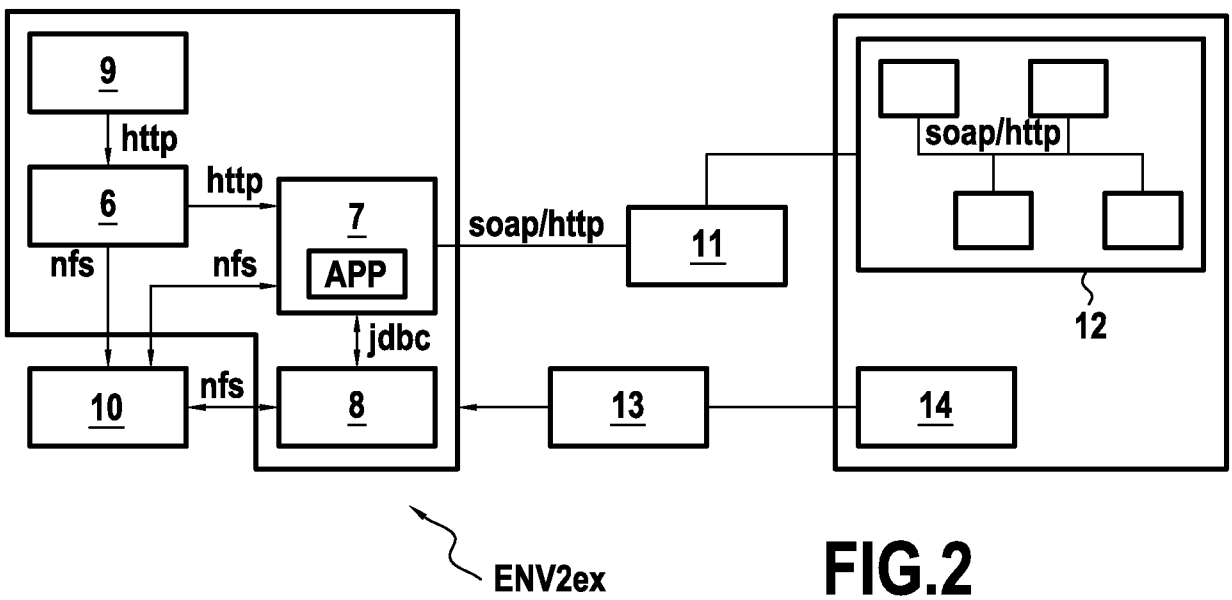

9
http
6
http
7
APP
soap/http
11
nfs
nfs
jdbc
soap/http
12
10
nfs
8
13
14
ENV2ex

FIG.2

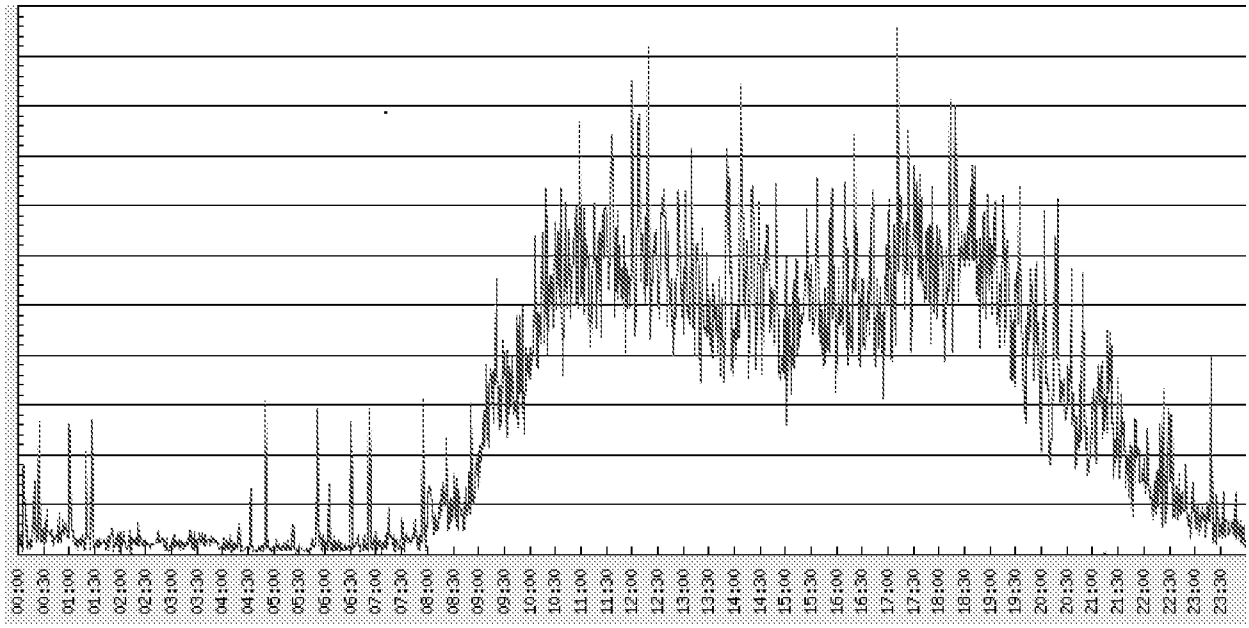

00:00
00:30
01:00
01:30
02:00
02:30
03:00
03:30
04:00
04:30
05:00
05:30
06:00
06:30
07:00
07:30
08:00
08:30
09:00
09:30
10:00
10:30
11:00
11:30
12:00
12:30
13:00
13:30
14:00
14:30
15:00
15:30
16:00
16:30
17:00
17:30
18:00
18:30
19:00
19:30
20:00
20:30
21:00
21:30
22:00
22:30
23:00
23:30

FIG.3A

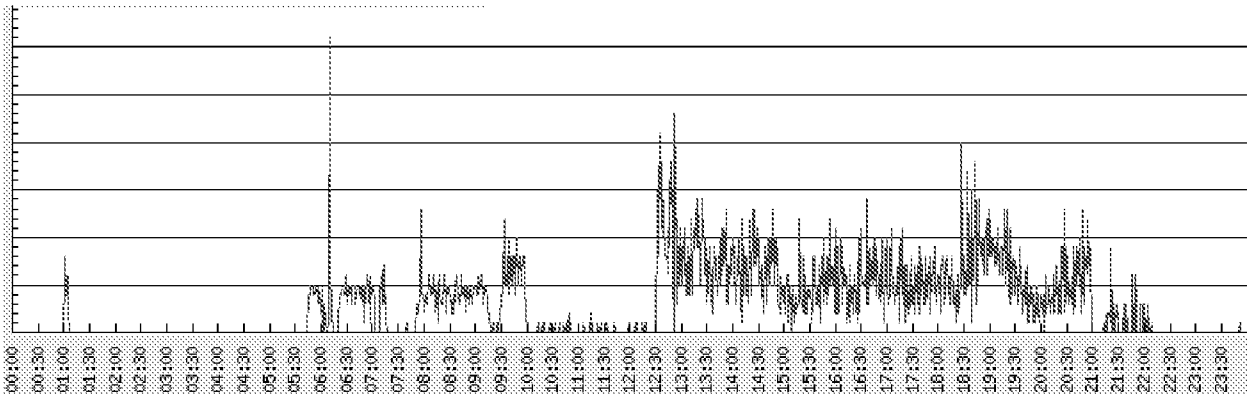

00:00
00:30
01:00
01:30
02:00
02:30
03:00
03:30
04:00
04:30
05:00
05:30
06:00
06:30
07:00
07:30
08:00
08:30
09:00
09:30
10:00
10:30
11:00
11:30
12:00
12:30
13:00
13:30
14:00
14:30
15:00
15:30
16:00
16:30
17:00
17:30
18:00
18:30
19:00
19:30
20:00
20:30
21:00
21:30
22:00
22:30
23:00
23:30

FIG.3B

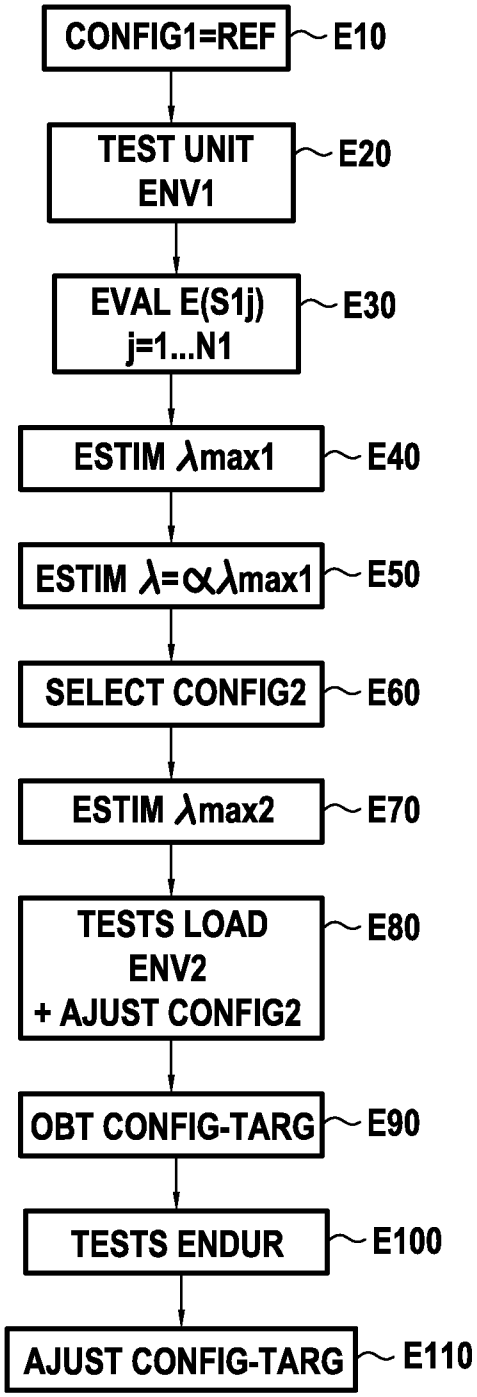

CONFIG1=REF
E10
TEST UNIT ENV1
E20
EVAL E(S1j) j=1...N1
E30
ESTIM λmax1
E40
ESTIM λ=αλmax1
E50
SELECT CONFIG2
E60
ESTIM λmax2
E70
TESTS LOAD ENV2 + AJUST CONFIG2
E80
OBT CONFIG-TARG
E90
TESTS ENDUR
E100
AJUST CONFIG-TARG
E110

FIG.4

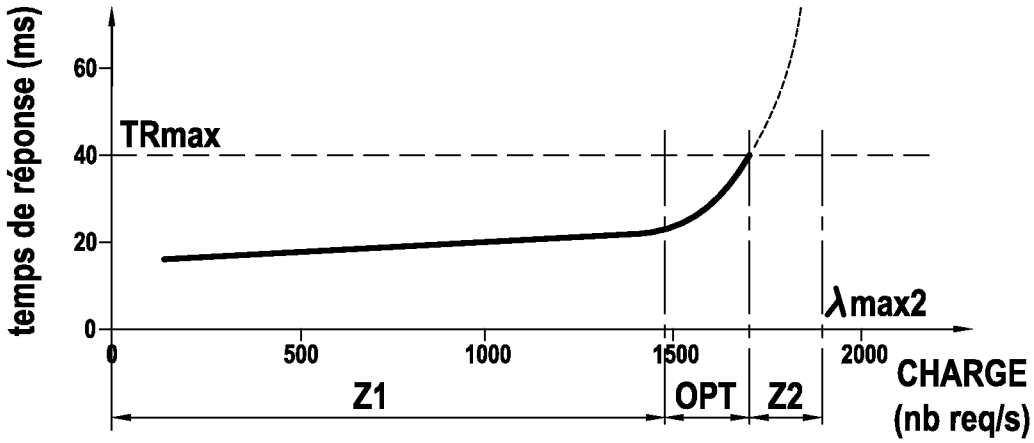


FIG.5A

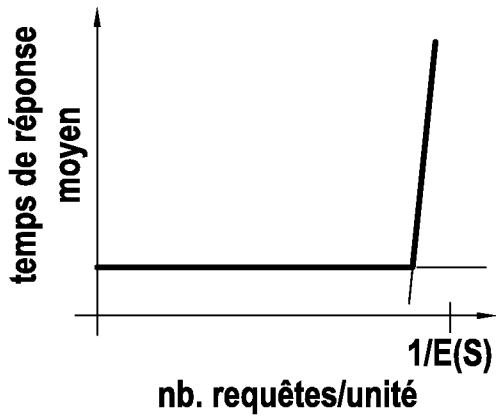


FIG.5B

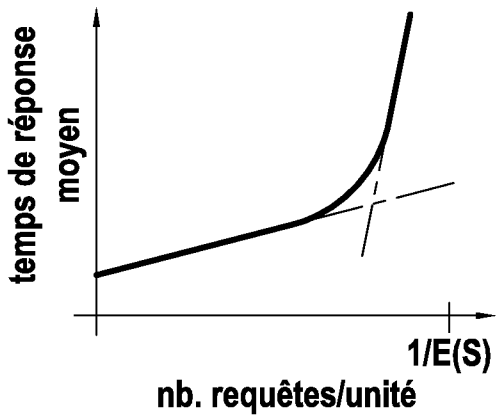


FIG.5C

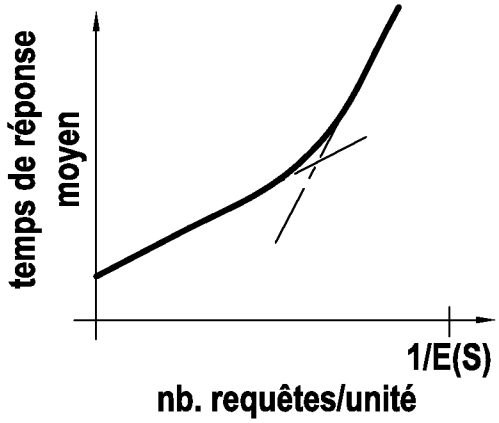


FIG.5D

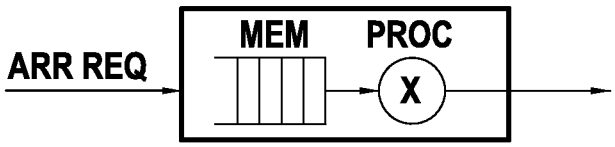


FIG.6A

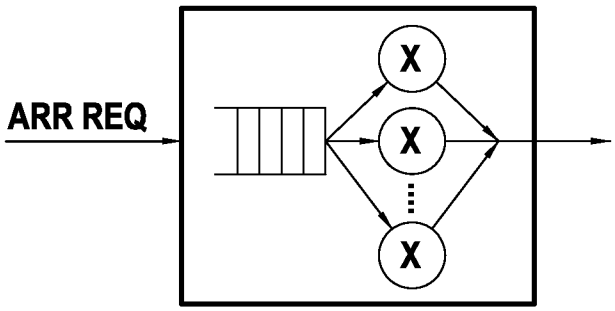


FIG.6B

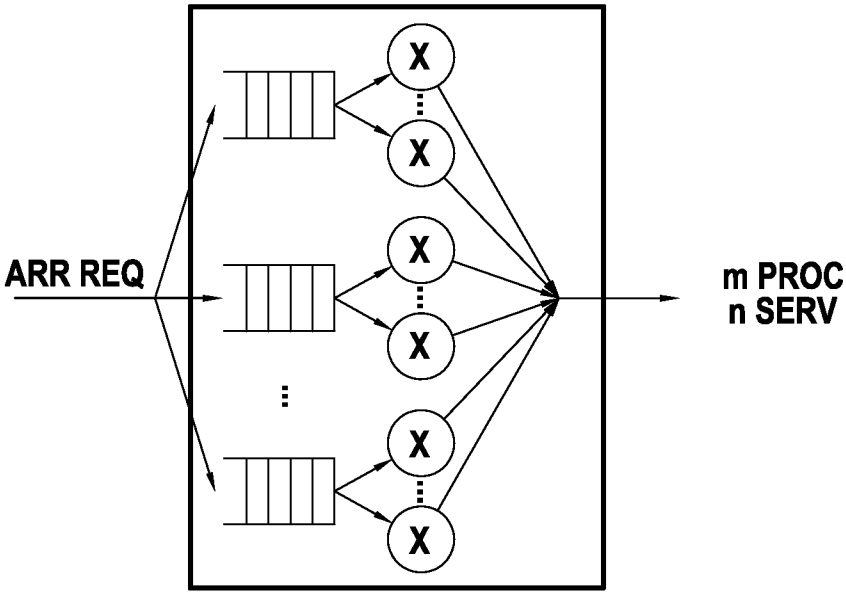
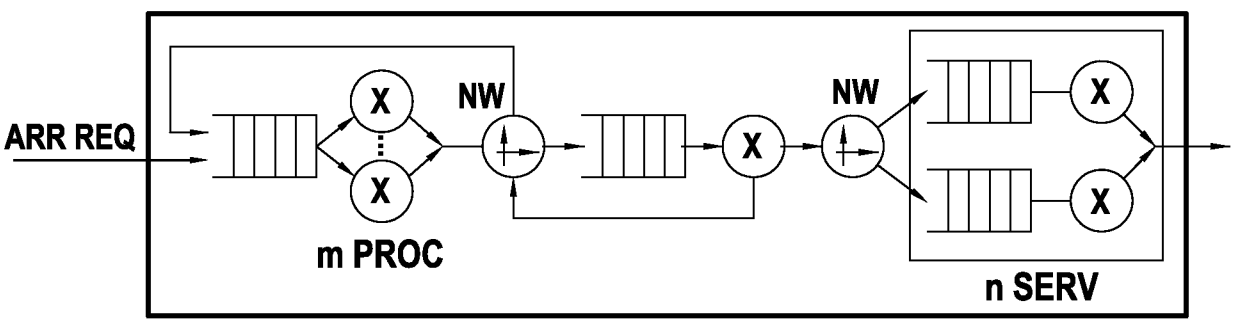


FIG.6C

FIG.6D



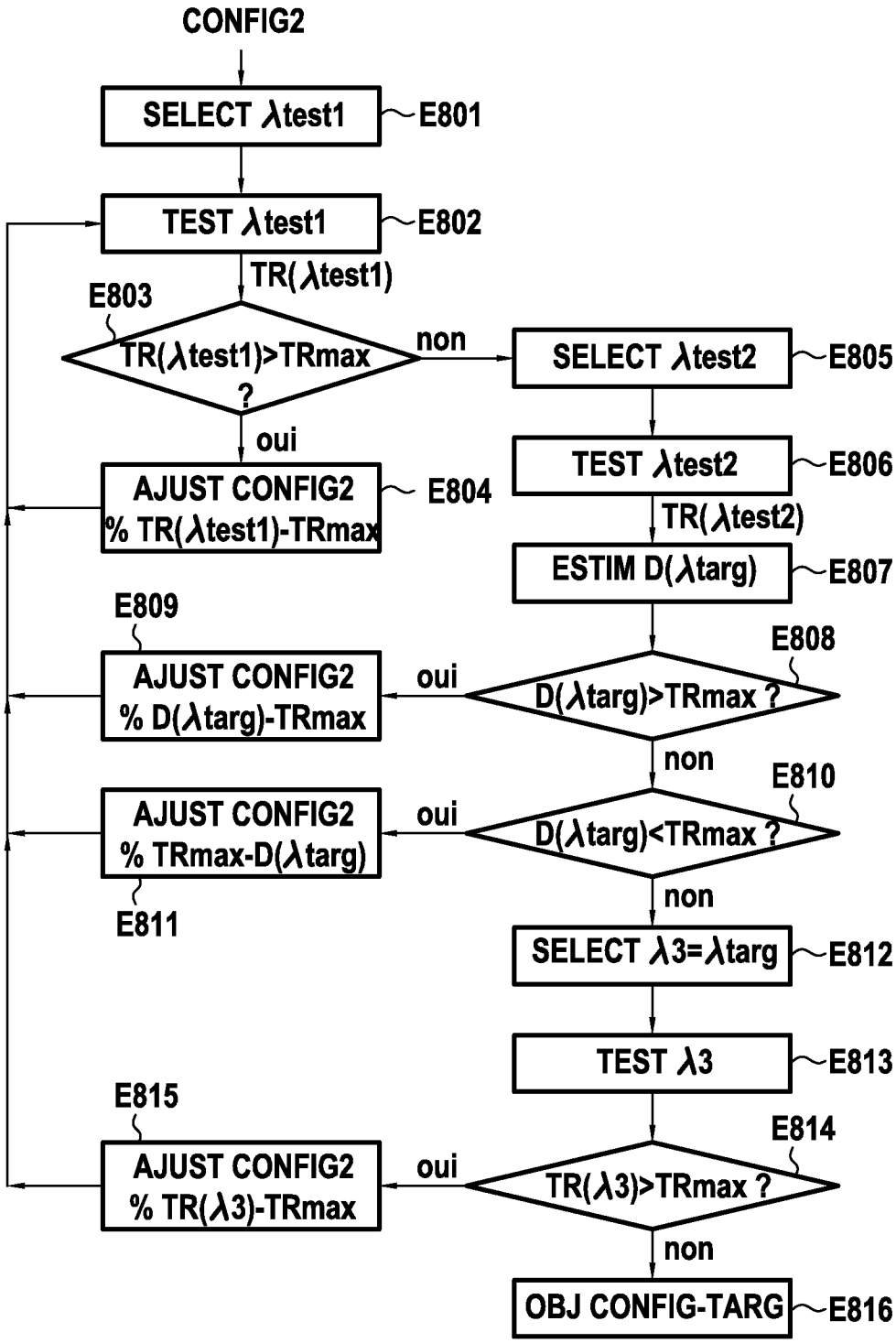

CONFIG2
SELECT λtest1
E801
TEST λtest1
E802
TR(λtest1)
E803
TR(λtest1)>TRmax ?
non
oui
SELECT λtest2
E805
TEST λtest2
E806
TR(λtest2)
ESTIM D(λtarg)
E807
AJUST CONFIG2
% TR(λtest1)-TRmax
E804
E809
E808
AJUST CONFIG2
% D(λtarg)-TRmax
oui
D(λtarg)>TRmax ?
non
E810
AJUST CONFIG2
% TRmax-D(λtarg)
oui
D(λtarg)<TRmax ?
E811
non
SELECT λ3=λtarg
E812
TEST λ3
E813
E815
E814
AJUST CONFIG2
% TR(λ3)-TRmax
oui
TR(λ3)>TRmax ?
non
OBJ CONFIG-TARG
E816

FIG.7

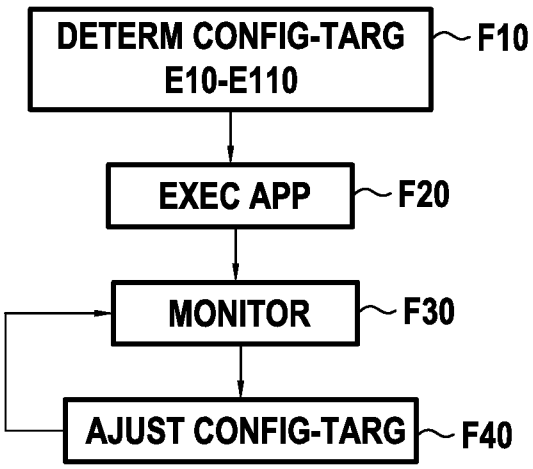

DETERM CONFIG-TARG
E10-E110
F10
EXEC APP
F20
MONITOR
F30
AJUST CONFIG-TARG
F40

FIG.9

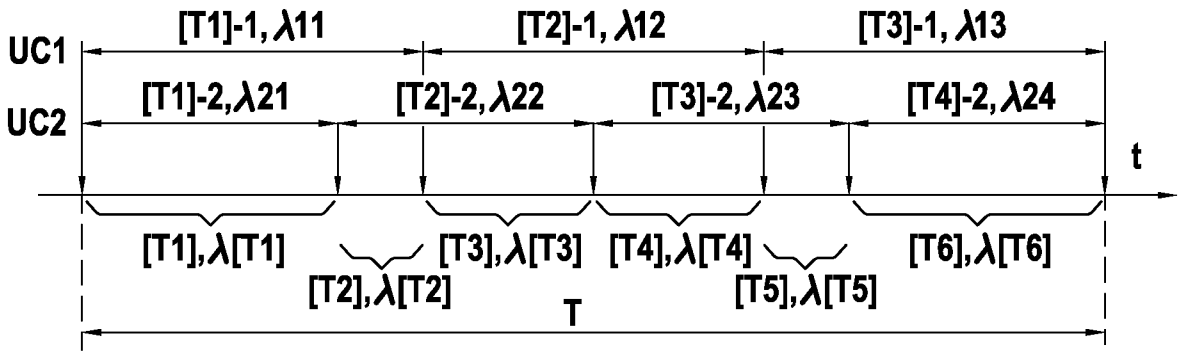

UC1
UC2
[T1]-1, λ11
[T2]-1, λ12
[T3]-1, λ13
[T1]-2,λ21
[T2]-2,λ22
[T3]-2,λ23
[T4]-2,λ24
t
[T1],λ[T1]
[T2],λ[T2]
[T3],λ[T3]
[T4],λ[T4]
[T5],λ[T5]
[T6],λ[T6]
T

FIG.10

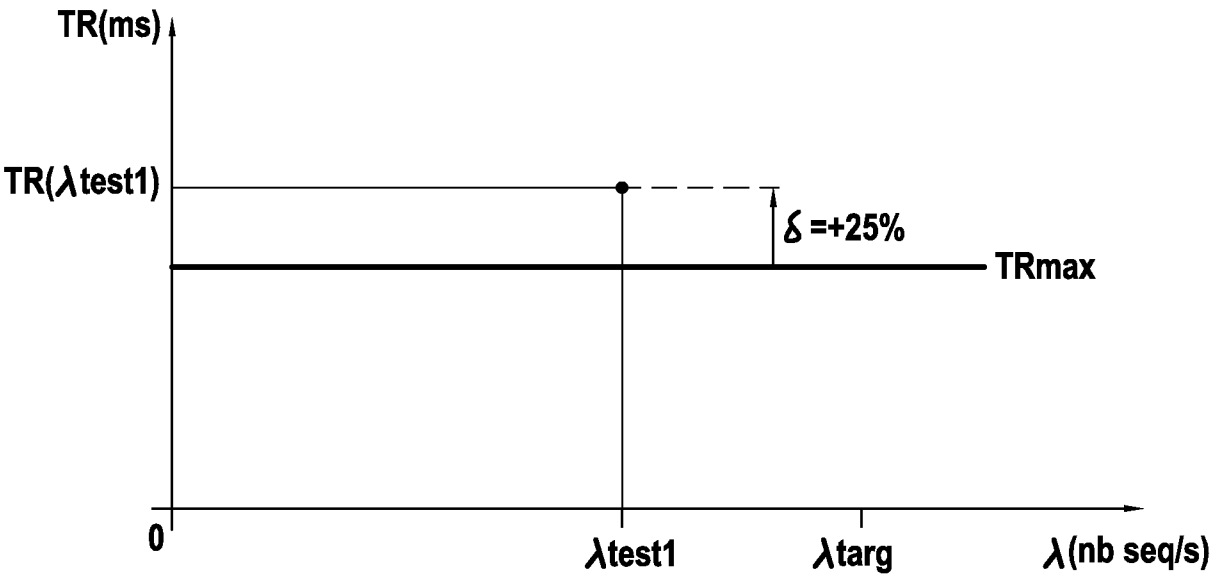

TR(ms)
TR(λtest1)
δ =+25%
TRmax
0
λtest1
λtarg
λ(nb seq/s)

FIG.8A

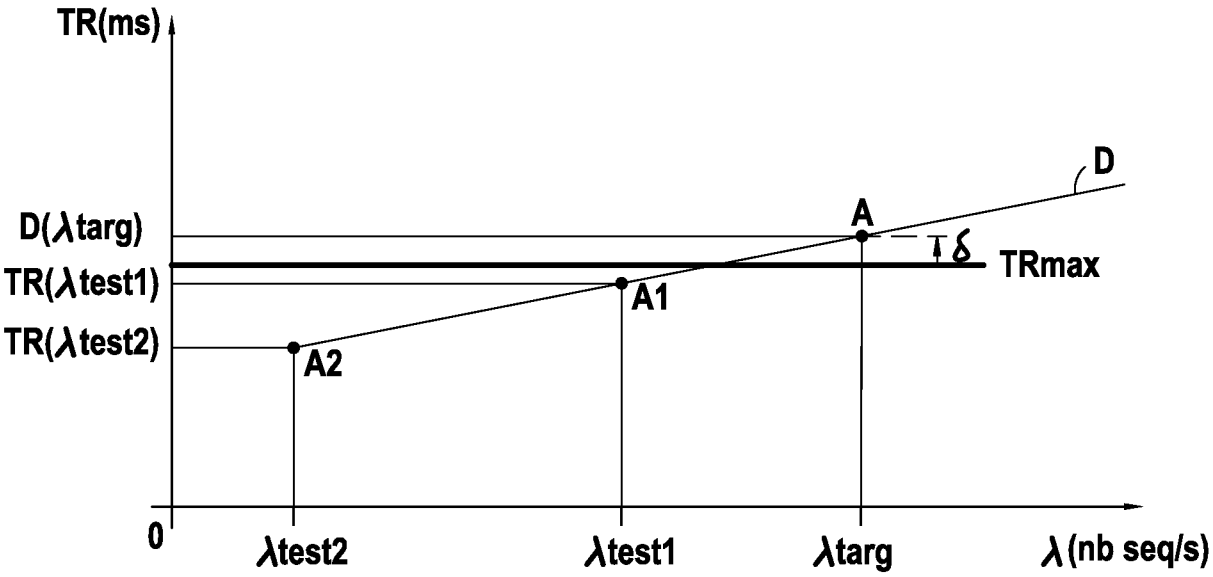

TR(ms)
D
D(λtarg)
A
δ
TRmax
TR(λtest1)
A1
TR(λtest2)
A2
0
λtest2
λtest1
λtarg
λ(nb seq/s)

FIG.8B

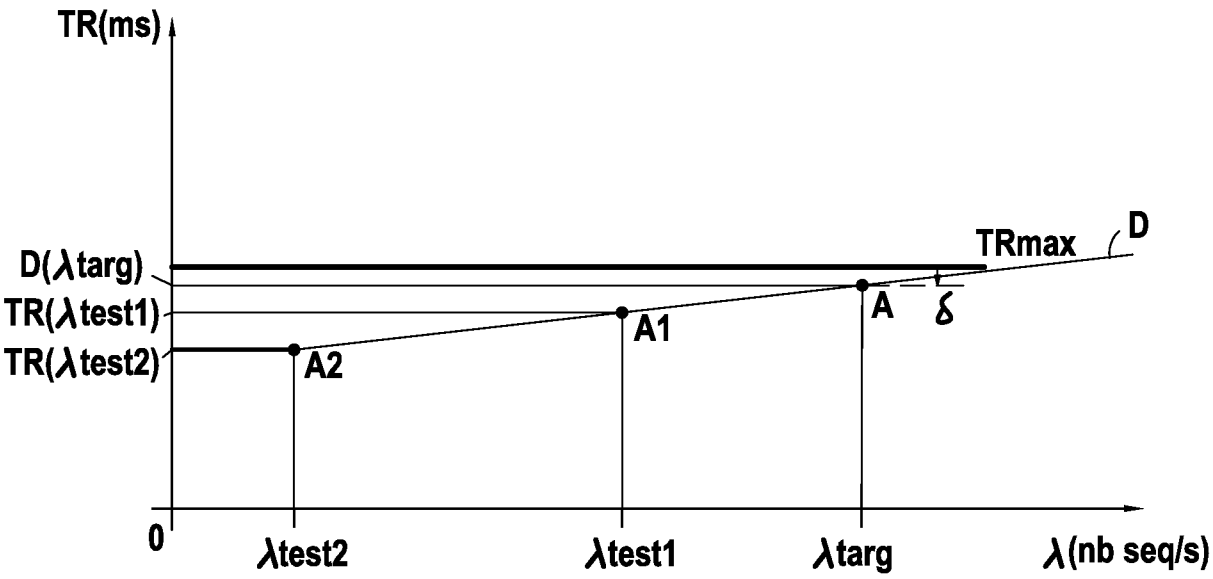

TR(ms)
D(λtarg)
TR(λtest1)
TR(λtest2)
TRmax
D
A
δ
A1
A2
0
λtest2
λtest1
λtarg
λ(nb seq/s)

FIG.8C

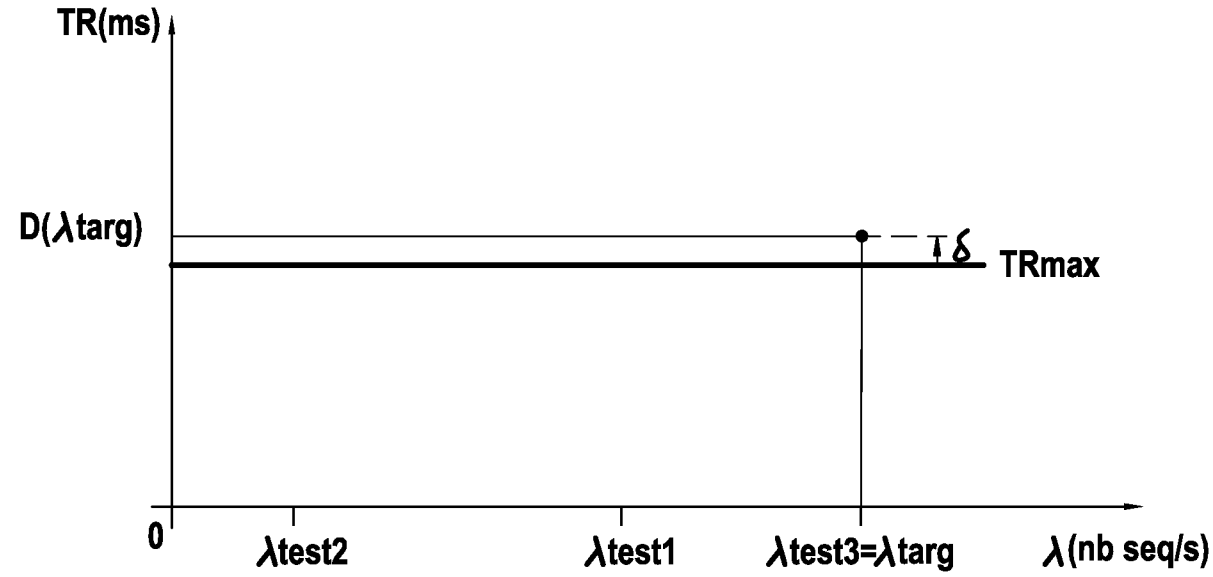

TR(ms)
D(λtarg)
δ
TRmax
0
λtest2
λtest1
λtest3=λtarg
λ(nb seq/s)

FIG.8D

INTERNATIONAL SEARCH REPORT

International application No
PCT/FR2016/051266

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F9/50 G06F11/34
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal, INSPEC, IBM-TDB, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	SUN YU ET AL: "A Model-Based System to Automate Cloud Resource Allocation and Optimization", 28 September 2014 (2014-09-28), CORRECT SYSTEM DESIGN; [LECTURE NOTES IN COMPUTER SCIENCE; LECT.NOTES COMPUTER], SPRINGER INTERNATIONAL PUBLISHING, CHAM, PAGE(S) 18 - 34, XP047299834, ISSN: 0302-9743 ISBN: 978-3-642-28871-5 pages 18-34, abstract page 18, paragraph 1 page 19, paragraph 3 - page 30, paragraph 2; figures 1-3 ----- -/--	1-15

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
26 July 2016	02/08/2016

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Lelait, Sylvain

1

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/FR2016/051266

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2008/155074 A1 (BACINSCHI RADIM [CA]) 26 June 2008 (2008-06-26) abstract page 1, paragraph 1 - page 1, paragraph 19 page 2, paragraph 24 - page 3, paragraph 32 page 3, paragraph 40 - page 3, paragraph 41	1-15
A	LOÏC LETONDEUR ET AL: "Planification pour la gestion autonomique de l'élasticité d'applications dans le cloud", ACTES DE LA CONFERENCE COMPAS'14, 23-25 AVRIL 2014,NEUCHÂTEL, SUISSE, 23 April 2014 (2014-04-23), pages 1-12, XP055257355, cited in the application abstract page 1, paragraph 1 - page 7, paragraph 1	1-15
A	YATAGHENE LYDIA ET AL: "A Queuing Model for Business Processes Elasticity Evaluation", 2014 INTERNATIONAL WORKSHOP ON ADVANCED INFORMATION SYSTEMS FOR ENTERPRISES, IEEE, 10 November 2014 (2014-11-10), pages 22-28, XP032699589, DOI: 10.1109/IWAISE.2014.12 abstract page 22, left-hand column, paragraph 1 - page 26, right-hand column, paragraph 5	1-15

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/FR2016/051266

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2008155074 A1	26-06-2008	US 2008155074 A1	26-06-2008
		WO 2008079739 A2	03-07-2008

RAPPORT DE RECHERCHE INTERNATIONALE

Demande internationale n°
PCT/FR2016/051266

A. CLASSEMENT DE L'OBJET DE LA DEMANDE
INV. G06F9/50 G06F11/34
ADD.

Selon la classification internationale des brevets (CIB) ou à la fois selon la classification nationale et la CIB

B. DOMAINES SUR LESQUELS LA RECHERCHE A PORTE

Documentation minimale consultée (système de classification suivi des symboles de classement)
G06F

Documentation consultée autre que la documentation minimale dans la mesure où ces documents relèvent des domaines sur lesquels a porté la recherche

Base de données électronique consultée au cours de la recherche internationale (nom de la base de données, et si cela est réalisable, termes de recherche utilisés)
EPO-Internal, INSPEC, IBM-TDB, WPI Data

C. DOCUMENTS CONSIDERES COMME PERTINENTS

Catégorie*	Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents	no. des revendications visées
A	SUN YU ET AL: "A Model-Based System to Automate Cloud Resource Allocation and Optimization", 28 septembre 2014 (2014-09-28), CORRECT SYSTEM DESIGN; [LECTURE NOTES IN COMPUTER SCIENCE; LECT.NOTES COMPUTER], SPRINGER INTERNATIONAL PUBLISHING, CHAM, PAGE(S) 18 - 34, XP047299834, ISSN: 0302-9743 ISBN: 978-3-642-28871-5 pages 18-34, abrégé page 18, alinéa 1 page 19, alinéa 3 - page 30, alinéa 2; figures 1-3 ----- -/--	1-15

[X] Voir la suite du cadre C pour la fin de la liste des documents

[X] Les documents de familles de brevets sont indiqués en annexe

* Catégories spéciales de documents cités:

"A" document définissant l'état général de la technique, non considéré comme particulièrement pertinent

"E" document antérieur, mais publié à la date de dépôt international ou après cette date

"L" document pouvant jeter un doute sur une revendication de priorité ou cité pour déterminer la date de publication d'une autre citation ou pour une raison spéciale (telle qu'indiquée)

"O" document se référant à une divulgation orale, à un usage, à une exposition ou tous autres moyens

"P" document publié avant la date de dépôt international, mais postérieurement à la date de priorité revendiquée

"T" document ultérieur publié après la date de dépôt international ou la date de priorité et n'appartenant pas à l'état de la technique pertinent, mais cité pour comprendre le principe ou la théorie constituant la base de l'invention

"X" document particulièrement pertinent; l'invention revendiquée ne peut être considérée comme nouvelle ou comme impliquant une activité inventive par rapport au document considéré isolément

"Y" document particulièrement pertinent; l'invention revendiquée ne peut être considérée comme impliquant une activité inventive lorsque le document est associé à un ou plusieurs autres documents de même nature, cette combinaison étant évidente pour une personne du métier

"&" document qui fait partie de la même famille de brevets

Date à laquelle la recherche internationale a été effectivement achevée	Date d'expédition du présent rapport de recherche internationale
26 juillet 2016	02/08/2016

Nom et adresse postale de l'administration chargée de la recherche internationale	Fonctionnaire autorisé
Office Européen des Brevets, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Lelait, Sylvain

1

C(suite). DOCUMENTS CONSIDERES COMME PERTINENTS

Catégorie*	Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents	no. des revendications visées
A	US 2008/155074 A1 (BACINSCHI RADIM [CA]) 26 juin 2008 (2008-06-26) abrégé page 1, alinéa 1 - page 1, alinéa 19 page 2, alinéa 24 - page 3, alinéa 32 page 3, alinéa 40 - page 3, alinéa 41	1-15
A	LOÏC LETONDEUR ET AL: "Planification pour la gestion autonomique de l'élasticité d'applications dans le cloud", ACTES DE LA CONFERENCE COMPAS'14, 23-25 AVRIL 2014,NEUCHÂTEL, SUISSE, 23 avril 2014 (2014-04-23), pages 1-12, XP055257355, cité dans la demande abrégé page 1, alinéa 1 - page 7, alinéa 1	1-15
A	YATAGHENE LYDIA ET AL: "A Queuing Model for Business Processes Elasticity Evaluation", 2014 INTERNATIONAL WORKSHOP ON ADVANCED INFORMATION SYSTEMS FOR ENTERPRISES, IEEE, 10 novembre 2014 (2014-11-10), pages 22-28, XP032699589, DOI: 10.1109/IWAISE.2014.12 abrégé page 22, colonne de gauche, alinéa 1 - page 26, colonne de droite, alinéa 5	1-15

RAPPORT DE RECHERCHE INTERNATIONALE

Renseignements relatifs aux membres de familles de brevets

Demande internationale n°
PCT/FR2016/051266

Document brevet cité au rapport de recherche	Date de publication	Membre(s) de la famille de brevet(s)	Date de publication
US 2008155074 A1	26-06-2008	US 2008155074 A1	26-06-2008
		WO 2008079739 A2	03-07-2008