Compas'2017 : Parallélisme / Architecture / Système
Sophia Antipolis, France, du 27 au 30 juin 2017

Ordonnancement multi-objectifs de workflows dans le cloud : un modèle plus réaliste avec tâches de durée stochastique

Aurélie Kong Win Chang*

Université de Lyon, ENS de Lyon, Inria, CNRS, Université Claude-Bernard Lyon 1, LIP.

Résumé
La souplesse en terme de disponibilité de ressources que permet le cloud rend possible une adaptation de l'ordonnancement des tâches qui y sont exécutées face à l'imprévisibilité de certains paramètres. Cependant, les méthodes d'ordonnancement existantes utilisent des modèles trop simplifiés : les workflows sont totalement déterministes ou leur structure n'est pas considérée, ou encore la modélisation du cloud ignore certains aspects capitaux de cette plateforme. Cet article propose un modèle prenant en compte le fait que le nombre d'instructions constituant une tâche peut ne pas être déterministe sans pour autant sacrifier totalement la complexité de la plateforme ou la structure du workflow. Nous proposons en outre quelques pistes pour l'élaboration d'une méthode d'ordonnancement multi-objectifs reposant sur ce modèle.

Mots-clés : ordonnancement multi-objectifs, cloud, workflows de tâches

1. Introduction

Le cloud est assimilable à une ressource de calcul d'une grande souplesse d'utilisation et peu coûteuse à l'entretien du point de vue des utilisateurs. Cette ressource est de plus en plus attractive tant au niveau des prix que de la fiabilité. Nous nous intéressons ici à un type de cloud dit *IaaS* (Infrastructure as a Service), dans lequel le fournisseur de cloud propose à l'utilisateur l'accès à l'équivalent virtuel de machines physiques, le laissant libre d'utiliser à sa guise ce qu'il a acheté. Cette liberté d'utilisation implique que c'est à l'utilisateur de définir comment gérer ces ressources, en particulier pour l'ordonnancement des tâches qu'il souhaite effectuer sur le cloud. Ce sera donc à lui de choisir, compte tenu de l'offre proposée par le fournisseur et des workflows de tâches qu'il veut soumettre, à quelles machines attribuer quelles tâches et à quel moment commencer le calcul des tâches et le chargement des données impliquées.

Des solutions d'ordonnancement pour répondre à la problématique de la minimisation du prix de l'exécution d'un ou plusieurs workflows existent [2, 4, 6] mais les modèles considérés pour ordonnancer les tâches soit ne prennent pas en compte la stochasticité inhérente au cloud issue du grand nombre de variables imprévisibles qui y sont impliquées, soit sont excessivement simplifiés. Dans le cadre d'une recherche de respect d'une *deadline* fixée à l'avance par l'utilisateur, ceci peut donner naissance à des algorithmes trop pessimistes, dans les cas où de nombreux mécanismes de la plateforme et informations tirées du workflow auraient pu être mis à

*. Un remerciement tout particulier à Yves Caniou, Eddy Caron et Yves Robert, mes encadrants de thèse, pour leurs conseils et relecture de ces travaux.

profit ; ou au contraire à des algorithmes trop optimistes dans le cas où des optimisations sont effectuées sans prendre en compte le risque qu'une tâche puisse demander plus de temps que prévu initialement.

Nous proposons ici un modèle pour étudier l'ordonnancement de workflows en gérant le non déterminisme de la durée des tâches, dans un cloud de type IaaS, avec deux objectifs : le respect d'une *deadline* stricte définie initialement par l'utilisateur, et le respect d'un budget. Après un bref récapitulatif en Section 2 des travaux similaires, nous présenterons notre modèle en détail et nous en justifierons ses différents aspects en Section 3, puis en Section 4 nous présenterons les bases d'une méthode d'ordonnancement que nous souhaitons élaborer à partir de ce modèle.

2. État de l'art

La stochasticité dans le cadre de workflows de tâches a été étudiée dans plusieurs travaux, mais ces derniers ne prennent généralement pas en compte la structure du workflow, préférant se placer dans un contexte *online* avec des arrivées de tâches imprévisibles, à des dates aléatoires comme dans [3]. Quelques travaux ont proposé des algorithmes tenant compte d'une incertitude sur la durée de calcul des tâches ([4], qui fait du *bin packing* dynamique ou [7], qui fait de l'ordonnancement en milieu hétérogène) mais l'absence de prise en compte de la structure du workflow et, d'une manière générale, les simplifications faites au modèle étudié ne leur permettent pas d'anticiper efficacement l'exécution des tâches futures.

Par contre, l'ordonnancement de workflows de tâches déterministes dans le cloud a été bien plus étudié [2], et ce sous la forme de nombreuses variantes, dont celles consistant à respecter une *deadline* tout en prenant en compte le coût monétaire de l'exécution du workflow. Dans [10], les auteurs partent du présupposé pessimiste qu'en général, les ressources allouées sont insuffisantes pour que toutes les tâches de tous les workflows puissent être calculées, et envisagent le respect de la *deadline* comme une maximisation du nombre d'ensembles de tâches qu'ils peuvent exécuter intégralement, compte tenu du temps et du budget impartis. Bien que leur modèle soit plutôt complet (cloud hétérogène, VMs nécessitant un temps de démarrage, ...), l'absence de prise en compte des temps de transfert des fichiers d'une tâche à une autre pourrait conduire à des ordonnancements finalement non viables pour des workflows demandant de transférer des volumes de données conséquents. [1] et [9], quant à eux, cherchent à ce que toutes les tâches se terminent avant la *deadline*, tout en minimisant le coût de l'exécution des workflows, mais à l'instar de [10], les VMs modélisées sont bien plus limitées que dans la réalité dans le sens où elles ne peuvent exécuter qu'une tâche à la fois. De plus, elles ne peuvent charger les fichiers nécessaires à l'exécution d'une tâche tout en effectuant l'exécution d'une autre. Ces simplifications risquent de mener à une solution pessimiste qui aura tendance à fortement exagérer les temps nécessaires pour exécuter les différentes tâches, et aboutir à une augmentation démesurée du coût d'exécution total, ainsi que du temps nécessaire à exécuter le workflow.

On trouve également dans [8] des algorithmes d'ordonnancement, avec un modèle de plateforme plutôt fin et un temps de calcul de tâches non déterminé. Partant du principe que les ressources allouées en termes de temps et de budget seront insuffisantes pour exécuter la totalité des workflows, les auteurs cherchent à maximiser le nombre de workflows calculés intégralement. Cependant le modèle des VMs a été simplifié, et les temps de transfert ne sont pris en compte que comme partie intégrante du temps d'exécution des tâches, et sont donc considérés comme indépendants du placement de ces tâches, ce qui risque de fortement diminuer l'intérêt d'utiliser de telles méthodes dans le cadre de workflows manipulant de grosses quantités de

données. En outre, la seule concession faite vis-à-vis du non déterminisme du temps de calcul des tâches est la conservation d'une marge arbitrairement fixée à 10% du coût horaire de calcul des VMs. Toutes ces limitations nous conduisent à proposer notre nouveau modèle.

3. Modélisation

3.1. Workflow

La structure de workflow scientifique présentée ici s'inspire de [5]. Ainsi, un workflow de tâches est représenté sous la forme d'un graphe orienté acyclique $G = (V, E)$, où V représente les tâches qu'il nous faut ordonnancer et E décrit les dépendances entre tâches liées aux transferts des données. Les tâches sont considérées comme séquentielles et non préemptives. Nombreux sont les algorithmes d'ordonnancement supposant de connaître exactement le nombre d'instructions composant une tâche, voire même le temps exact nécessaire à la complétion de la tâche. Cependant, cette hypothèse n'est pas toujours réaliste. Le nombre d'instructions d'une tâche peut dépendre directement du contenu des données traitées, contenu non connu à l'avance, par exemple dans le cas de certains traitements d'images. Dans ce modèle, une tâche est donc constituée d'un nombre inconnu d'instructions, dont nous connaissons cependant une estimation. En l'absence de connaissances suffisamment précises sur l'origine des variations pour pouvoir faire un bon choix de modèle, nous avons choisi de le simplifier en estimant que tous les paramètres déterminant le nombre d'instructions d'une tâche sont indépendants entre eux, et que le résultat qui en découle tend à suivre une loi normale de moyenne $\overline{w_{real,i}}$ et d'un écart-type qu'on peut estimer (par exemple par échantillonnage) . À chaque dépendance $(T_i, T_j) \in E$ est associé un fichier de taille connue $size(f_{i,j})$. Une tâche est considérée comme calculable si elle n'a aucun prédécesseur dans le graphe des dépendances, ou si tous ses prédécesseurs ont été calculés, c'est-à-dire si tous les fichiers nécessaires à l'exécution de la tâche dite calculable sont disponibles.

3.2. Plateforme

Notre modèle de cloud (Figure 1) repose sur deux types d'éléments : un centre de stockage et des unités de calcul, et est largement inspiré de l'offre actuelle en matière de cloud public par trois grands fournisseurs (Google cloud, Amazon EC2, OVH). Les fournisseurs de cloud proposant un service de tolérance aux pannes permettant une disponibilité des ressources très élevée et une redondance de données suffisante (la disponibilité des machines au cours du dernier mois est généralement au-delà de 99,97 %[2]), le centre de stockage et les unités de calcul sont considérés comme fiables et non sujets aux pannes.

Le centre de stockage est unique, et est utilisé par toutes les unités de calcul. Il s'agit du lieu de transit des données échangées entre unités de calcul, celles-ci n'interagissant pas directement entre elles, et du lieu de stockage des données obtenues à la fin de l'exécution du workflow juste avant que l'utilisateur ne les télécharge. Par souci de simplicité, la bande passante du centre de stockage est considérée comme suffisante pour répondre pleinement aux demandes des unités de calcul, les limitations en terme de transfert de données provenant donc de la taille de la bande passante allouée à ces dernières. De même, on considèrera que le centre de stockage a la capacité de traiter simultanément toutes les requêtes qui lui sont soumises sans surcoût.
Les unités de calcul sont des Machines Virtuelles (ou VM pour *Virtual Machine*). Elles se subdivisent en différents types définis au préalable par le fournisseur, caractérisés par un ensemble de paramètres. Certains fournisseurs peuvent proposer des paramètres qui leur sont bien spé-

2. `https://cloudharmony.com/status`

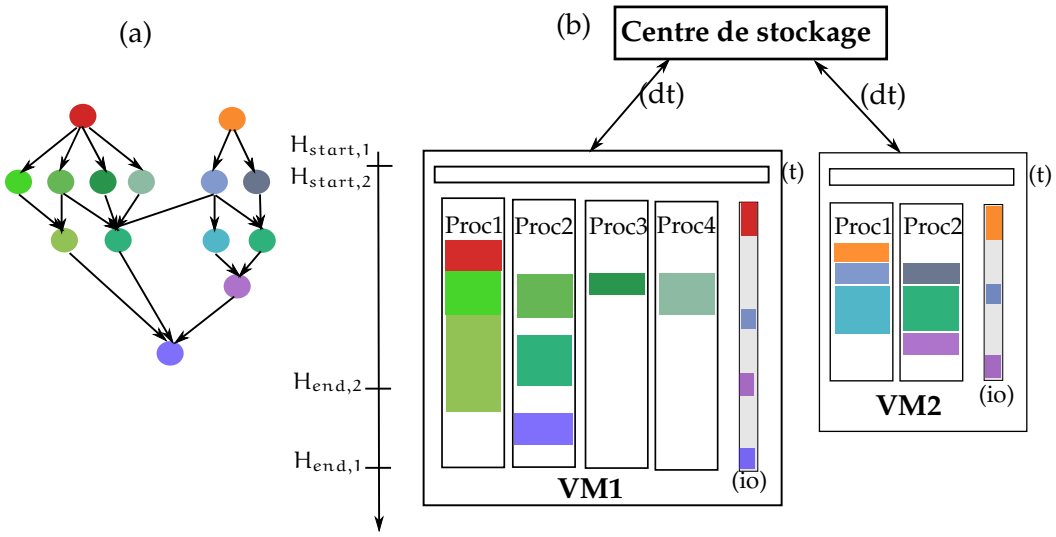

FIGURE 1 – Workflow (a) et plateforme (b) : machines virtuelles (VM), leurs processeurs (Proc1 à Proc4), leur temps de transfert de fichiers (io), leur temps de démarrage (t) de durée d_{ini}, leurs interactions avec le centre de stockage commun (dt)

cifiques, tels que le nombre de règles utilisables pour pouvoir gérer la charge du réseau [3]. Nous n'avons conservé que les paramètres communs aux différents fournisseurs considérés : une VM de type k a un nombre $| \mathcal{H} | = n_{p,k}$ de processeurs, chaque processeur permettant de calculer une tâche à la fois ; une vitesse de calcul s_k correspondant au nombre d'instructions par unité de temps qu'un processeur de la VM est capable d'effectuer ; un coût horaire $c_{h,k}$ et un coût initial $c_{ini,k}$. De manière générale, toutes ces VMs ont besoin d'un temps initial de démarrage d_{ini} avant de pouvoir commencer à calculer. Ce temps de démarrage n'est pas compté dans le coût horaire lié à l'utilisation de la VM, dont nous aborderons le calcul dans la Section 3.4.

Notre plateforme est ainsi dotée d'un ensemble de n VMs de k types possibles. Un certain nombre de simplifications ont été effectuées dans le cadre de ce modèle. Étant donné que la stochasticité qui nous intéresse ici est celle liée à la durée des tâches, la bande passante est la même pour toutes les VMs, en entrée comme en sortie, et ne varie pas. Nous considérons en outre que la capacité de stockage des données des VMs est suffisante pour les tâches qu'on souhaite leur allouer ainsi que les données à stocker, une VM ne connaîtra donc pas de problème de dépassement de mémoire et les augmentations de temps total d'exécution du workflow seront effectivement liées au caractère stochastique de la durée des tâches. Le temps nécessaire au démarrage est le même pour toutes les VMs. Nous partons également du principe que les transferts de fichiers sont indépendants et n'influent pas sur la vitesse de traitement des tâches.

Nous utilisons un système d'approvisionnement à la demande : il est possible de déployer une nouvelle VM pendant une exécution de workflow si besoin est, ainsi toutes les VMs ne commenceront pas nécessairement en même temps. Une VM v est donc démarrée au temps $H_{start,v}$, et s'arrête une fois que tous les fichiers de la dernière tâche qui lui a été allouée ont été envoyés au centre de stockage commun, au temps $H_{end,v}$. Les VMs sont allouées sur des plages de temps continues. Faire des allocations discontinues implique de libérer une VM, puis d'en réserver une autre plus tard, et donc de devoir envoyer les données de la dernière tâche

3. https://cloud.google.com/compute/pricing

calculée par la VM qui a été fermée au centre de stockage, puis de recharger et redémarrer une VM neuve plus tard. Cette possible souplesse dans l'utilisation des ressources nous permet de considérer des ordonnancements dynamiques, plus à même de faire face à l'imprévisibilité du temps de calcul des tâches.

3.3. Modèle d'exécution du workflow

L'exécution du workflow est conditionnée par une règle simple : pour qu'une tâche puisse être lancée sur l'un des processeurs d'une VM, il faut que le processeur soit disponible et que la tâche soit calculable. Par « disponible » nous entendons qu'aucune tâche n'est en cours d'exécution sur ce processeur et que l'ordonnanceur ne l'a pas réservé dans le but de parer à un imprévu. Une tâche est quant à elle dite « calculable » si tous les fichiers qui lui sont nécessaires ont été intégralement chargés sur la VM où elle est censée être exécutée et si toutes les tâches dont l'exécution a été prévue avant la sienne ont été exécutées. Tant que ces deux conditions ne sont pas vérifiées, le calcul de la tâche ne peut commencer. Étant donnés un cloud et un premier ordonnancement qui sera amené à évoluer au fil de l'exécution, on lancera pour commencer les différentes VMs nécessaires au calcul des tâches calculables prévues. Une fois les VMs lancées, le téléchargement des fichiers nécessaires au calcul des différentes tâches attribuées aux diverses VMs pourra débuter. Dès qu'une tâche prévue est calculable et que le processeur sur lequel elle doit s'exécuter est disponible, son calcul est lancé. L'algorithme d'ordonnancement sera réexécuté tout au long de l'exécution du workflow pour réajuster, si besoin est, le placement de chaque tâche et sa priorité sur la VM d'allocation. À noter que si tous les fichiers nécessaires à l'exécution d'une tâche sont sur une VM, il est inutile de les charger depuis le centre de stockage central ; tous les processeurs d'une VM ont accès aux données qui y sont stockées.

3.4. Modèle de coût

Le modèle de coût utilisé est simplifié pour représenter une certaine généricité des offres proposées actuellement chez les différents fournisseurs de cloud considérés [4] [5] [6].
Le coût total de l'exécution du workflow correspond à la somme des coûts liés à l'utilisation des VMs à laquelle est ajouté le coût d'utilisation du centre de stockage $\mathcal{C}_s$. Le coût C_v associé à l'utilisation d'une VM v de type k_v se calcule comme suit : $C_v = (H_{end,v} - H_{start,v}) \times c_{h,k_v} + c_{ini,k_v}$.
En ce qui concerne le coût d'utilisation du centre de stockage, il se base sur un coût horaire c_{sh} auquel est ajouté un coût de transfert c_{ft} de fichiers calculé à partir de la taille des données transférées depuis l'extérieur du cloud vers le cloud (size($d_{out,cloud}$),) et depuis le cloud vers l'extérieur du cloud (size($d_{cloud,out}$)) : $C_s = (\text{size}(d_{out,cloud}) + \text{size}(d_{cloud,out})) \times c_{sh} + c_{ft}$.
Le coût total de l'exécution de l'intégralité du workflow est donc, en notant R_{VM} l'ensemble des VMs qui ont été réservées au cours de l'exécution du workflow : $C_{wf} = \sum_{v \in R_{VM}} C_v + C_s$.

3.5. Objectif

Étant donnés une durée limite $\mathcal{D}$ et un budget $\mathcal{B}$, l'objectif est de respecter cette durée limite tout en ne dépassant pas le budget octroyé. En notant $H_{start,first}$ l'instant de réservation de la première VM et $H_{end,last}$ l'instant où les fichiers de la dernière tâche calculée ont fini d'être envoyés au centre de stockage central, on obtient donc :

$$\mathcal{D} \geq H_{end,last} - H_{start,first} \qquad \text{où } \mathcal{B} \geq C_{wf} \tag{1}$$

4. https://aws.amazon.com/ec2/pricing/on-demand/
5. https://cloud.google.com/compute/pricing
6. https://www.ovh.com/fr/public-cloud/instances/tarifs/

Une variante plus complexe étant de trouver l'ordonnancement permettant de minimiser le temps de calcul tout en respectant le budget : $\min(H_{end,last} - H_{start,first})$ où $\mathcal{B} \geq C_{wf}$

4. Élaboration d'une méthode d'ordonnancement

Nous proposons ce modèle afin de permettre son utilisation pour élaborer des méthodes d'ordonnancement plus performantes, dans le cas fréquent des workflows pour lesquels le temps de calcul des tâches n'est pas déterministe. Même si la conception de ces algorithmes reste ouverte, nous pouvons à titre d'exemple dès à présent présenter les grandes lignes d'une proposition de solution. Afin de profiter de la grande souplesse en matière de disponibilité des ressources proposée par le cloud, notre algorithme sera dynamique, et exécuté régulièrement. Deux types d'événements pourront enclencher cette exécution : la fin de l'exécution d'une tâche et le dépassement du temps moyen d'exécution attendu d'une valeur seuil qu'il conviendra de déterminer. Nous chercherons à exploiter les structures du workflow et de la plateforme afin de déterminer quels choix d'affectations, pour chaque tâche, permettra d'aboutir à la solution la plus appropriée et dans quel ordre effectuer les transferts de fichiers impliqués. Il pourra par exemple être intéressant de réserver des marges suffisamment larges si les incertitudes concernant les temps de calcul sont très élevées, ou utiliser les tâches de faible variance de temps de calcul dans les cas où les marges sont plus limitées. Cette détermination du meilleur choix nécessitera de déterminer sous quelles conditions il deviendra intéressant de dévier d'une solution précédente, en mettant à profit ce qui est su des tâches déjà exécutées, des temps de calcul des tâches en cours d'exécution et des variances connues pour celles-ci, du budget restant et du temps restant avant d'atteindre la *deadline*.

Nous pouvons supposer, compte tenu du modèle, qu'un changement d'ordonnancement aura majoritairement des répercussions dûes aux temps de chargement des fichiers nécessaires au calcul des tâches à calculer attribuées aux VMs impliquées. Il faudra notamment évaluer l'éventuel retard de début d'exécution des tâches réattribuées, issu du téléchargement des données nécessaires si ces dernières ne sont pas présentes sur les VMs les recevant, mais aussi les temps d'envoi du résultat de ces tâches, en profitant par exemple des moments où la bande passante des VMs n'est pas utilisée.

Une fois la méthode d'ordonnancement déterminée, nous la comparerons à d'autres méthodes de la littérature mises au point sur des modèles les plus proches possibles ([1] pour des workflows déterministes, [7] pour un algorithme dédié à des workflows non détermnistes) en obervant sur les résultats de simulations effectuées sur des workflows issus de la littrature [5] divers critères déterminant ce qui pour nous garantit la qualité d'une solution : le nombre de fois, pour un nombre d'exécutions donné, où la *deadline* est dépassée, celui où le budget est dépassé et le temps d'exécution des workflows, le tout sur différents types de workflows.

5. Conclusion

Nous avons proposé un modèle du problème d'ordonnancement de workflows de tâches dans un cloud de type IaaS se voulant plus réaliste que ceux traditionnellement utilisés dans la littérature. Ce modèle nous oriente vers un type bien précis d'ordonnancement dynamique, en se servant des informations disponibles pour prédire dans une certaine mesure ce qui se passera par la suite et réorganiser les placements au besoin. Nous souhaitons mettre à profit ces observations pour proposer de nouvelles méthodes d'ordonnancement, qui pourraient, à terme, constituer une avancée solide vers la prise en compte de la stochasticité du problème.

Bibliographie

1. Abrishami (S.), Naghibzadeh (M.) et Epema (D. H. J.). – Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds. *Future Generation Comp. Syst.*, vol. 29, n1, 2013, pp. 158–169.
2. Alkhanak (E. N.), Lee (S. P.), Rezaei (R.) et Parizi (R. M.). – Cost optimization approaches for scientific workflow scheduling in cloud and grid computing : A review, classifications, and open issues. *Journal of Systems and Software*, vol. 113, 2016, pp. 1–26.
3. Ghaderi (J.), Shakkottai (S.) et Srikant (R.). – Scheduling storms and streams in the cloud. *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 1, n4, août 2016, pp. 14 :1–14 :28.
4. Gupta (V.) et Radovanovic (A.). – Lagrangian-based online stochastic bin packing. *SIGMETRICS Perform. Eval. Rev.*, vol. 43, n1, juin 2015, pp. 467–468.
5. Juve (G.), Chervenak (A. L.), Deelman (E.), Bharathi (S.), Mehta (G.) et Vahi (K.). – Characterizing and profiling scientific workflows. *Future Generation Comp. Syst.*, vol. 29, n3, 2013, pp. 682–692.
6. Kllapi (H.), Sitaridi (E.), Tsangaris (M. M.) et Ioannidis (Y.). – Schedule optimization for data processing flows on the cloud. – In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, SIGMOD '11*, SIGMOD '11, pp. 289–300, New York, NY, USA, 2011. ACM.
7. Maguluri (S. T.) et Srikant (R.). – Scheduling jobs with unknown duration in clouds. *IEEE/ACM Trans. Netw.*, vol. 22, n6, décembre 2014, pp. 1938–1951.
8. Malawski (M.), Juve (G.), Deelman (E.) et Nabrzyski (J.). – Cost- and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds. – In *SC Conference on High Performance Computing Networking, Storage and Analysis, SC '12, Salt Lake City, UT, USA - November 11 - 15, 2012*, p. 22, 2012.
9. Rodriguez (M. A.) et Buyya (R.). – Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds. *IEEE Trans. Cloud Computing*, vol. 2, n2, 2014, pp. 222–235.
10. Shi (J.), Luo (J.), Dong (F.) et Zhang (J.). – A budget and deadline aware scientific workflow resource provisioning and scheduling mechanism for cloud. – In *Proceedings of the IEEE 18th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2014, Hsinchu, Taiwan, May 21-23, 2014*, pp. 672–677, 2014.