Modèles temporels d'évaluation des protocoles de gestion des données

Hamza Chaker, Loïc Cudennec, Safae Dahmani, Guy Gogniat, Cédric Maignan, Martha Johanna Sepùlveda

Compas'2016 : Parallélisme/ Architecture / Système
Lorient, France, du 5 au 8 juillet 2016

Modèles temporels d'évaluation des protocoles de gestion des données

Hamza Chaker[1], Loïc Cudennec[2], Safae Dahmani[3], Guy Gogniat[1], Cédric Maignan[1], Martha Johanna Sepùlveda[4]

[1] Université de Bretagne-Sud, Lorient, France [2] CEA, LIST, Saclay, France
[3] Sorbonne Universités, UPMC Paris 06, UMR 7606, LIP6, Paris [4] Technical University of Munich, Germany

---

Résumé
Les modèles et protocoles de cohérence des données sont responsables de la gestion des accès aux données partagées, et leur choix peut influencer les performances générales du système. Nous avons proposé dans des travaux précédents une chaîne de compilation permettant d'automatiser le processus du choix des protocoles. Le système est chargé de la prise de décision hors-ligne sur le choix et le paramétrage des protocoles. Dans ce processus, la phase d'évaluation des différentes solutions est une des principales briques de décision. L'analyse des performances doit être suffisamment précise pour orienter au mieux la prise de décision tout en étant rapide pour pouvoir s'intégrer dans la chaîne de compilation proposée. Afin de répondre à ces besoins, nous proposons deux modèles temporels d'évaluation des protocoles de cohérence. Le premier modèle permet de calculer les temps d'accès aux données en nombre de cycles. Le deuxième modèle est une extension basée sur un modèle TLM (*Transaction Level Modeling*) de réseau sur puce (*NoC*) qui permet d'estimer les temps d'accès, de prendre en charge l'effet de contention du réseau et de faire varier les paramètres de l'architecture (ex. différentes topologies du *NoC*).

Mots-clés : Protocoles de cohérence, Chaine de décision, Évaluation des performances, Modèles temporels.

---

1. Introduction

La programmation des systèmes hautement parallèles peut s'avérer être très complexe. Elle nécessite une gestion efficace de tous les aspects concurrentiels du calcul parallèle, notamment les conflits d'accès à la mémoire. Dans une architecture à mémoires réparties les accès aux données par plusieurs cœurs en parallèle mènent à des conflits d'accès. Ces problèmes sont soit gérés par le matériel, solution complexe et qui ne passe pas à l'échelle, ou bien par l'utilisateur, qui est amené à prendre des décisions sur les points de synchronisation et sur les accès atomiques dans son programme, ce qui n'est pas toujours facile à faire. Une autre solution, basée sur une mémoire virtuellement partagée, consiste à créer un espace d'adressage global entre toutes les mémoires réparties de l'architecture. Cette vision partagée de la mémoire sur puce permet de simplifier la gestion des accès aux données de façon transparente par rapport à l'utilisateur grâce à des mécanismes de cohérence. L'efficacité du système à gérer les accès aux données

partagées dépend de l'efficacité de ces mécanismes. L'étude de différents mécanismes et protocoles de cohérence a permis de montrer qu'il n'existe pas un protocole de cohérence qui soit adapté à tous les contextes applicatifs et à toutes les architectures cibles. Les systèmes existants proposent deux principales techniques. Une première technique déléguant au programmeur la décision sur le protocole de cohérence en fonction de ses besoins applicatifs. Les choix de l'utilisateur sont souvent limités par les contraintes de l'architecture. De plus, lors du développement de son programme, il ne possède pas toujours les moyens de vérifier l'efficacité de ses choix à l'exécution (difficulté d'accès à l'architecture par exemple). La deuxième technique consiste à implémenter un ou plusieurs protocoles de cohérence dans le système et d'adapter en ligne le choix des protocoles en fonction de l'évolution de l'exécution. Cette technique représente un surcoût élevé ce qui peut dégrader les performances du système. Nous avons proposé dans des travaux précédents une plateforme de compilation multi-protocolaire qui, pendant la compilation, permet de choisir et de configurer une combinaison de protocoles de cohérence en prenant en compte l'application et son environnement d'exécution. Dans le processus du choix et de configuration des protocoles de cohérence de données la phase d'évaluation des solutions est une des principales briques du processus de décision. L'outil d'estimation des performances des solutions doit être suffisamment précis pour orienter la décision sur la configuration des protocoles et suffisamment rapide pour s'intégrer dans la chaîne de compilation. L'objectif de ces travaux est de proposer une méthode d'évaluation qui répond à ces critères. Les modèles d'évaluation proposés sont basés sur une étude analytique des communications générées par les instances de protocoles. Un premier modèle consiste à évaluer les coûts en nombre de cycles d'accès aux données pour chacun des schémas de communication possibles. Cependant, il ne prend pas en compte les problèmes liés à la surcharge du réseau et la latence qui peut en résulter. Afin d'améliorer la précision de ce modèle, une version étendue a été développée. Elle se base sur un simulateur TLM (*Transaction Level Modeling* de réseau-sur-puce (NoC) permettant de modéliser des scénarios de contention plus proches de ce qui peut se passer sur le réseau réel.

2. Description des modèles temporels

2.1. État de l'art et motivation

Le choix des protocoles de cohérence nécessite d'évaluer plusieurs configurations de protocoles pendant la compilation. Une première approche consiste à déployer toutes les solutions et de les tester sur une plateforme réelle. Cette solution n'est pas toujours possible pour deux raisons ; la première est le coût temporel que le déploiement et l'exécution peuvent générer, la deuxième est liée à la disponibilité de la plateforme d'exécution. Une deuxième approche est la simulation de type *SystemC* [1]. Cette simulation permet plusieurs niveaux de précision selon le niveau d'abstraction choisi. La vitesse des outils de simulation avec un niveau d'abstraction bas ne dépasse pas 10 MIPS (ex. 3 MIPS pour le simulateur SESAM [10]). D'autres simulateurs moins précis tels que *OVPsim* [2] promettent d'atteindre des centaines de MIPS en fonction du comportement de l'application et de la complexité des processeurs utilisés dans la plateforme. Le niveau d'abstraction et la complexité du système simulé jouent un rôle très important dans la détermination des performances de simulation. Cependant, la simulation des systèmes avec un grand nombre de cœurs reste un processus complexe et coûteux. Si la puce *MPPA 256* est capable d'atteindre des performances de 60 fps pour un encodeur vidéo *h264*, l'exécution de cette même application sur un simulateur de type *ISS* (*Instruction Set Simulator*) nécessite plusieurs minutes pour le calcul d'une seule image [3]. D'autres travaux proposent de la modélisation matérielle *FPGA (Field Programmable Gate Array)* [11, 7, 6]. Ce type de modélisation promet

une grande précision dans la conception du système. La principale difficulté d'une telle implémentation est la complexité de conception des circuits *FPGA*. Une telle précision n'est pas forcément nécessaire dans notre cas d'étude et peut être très coûteuse en temps de simulation ce qui ne correspond pas à nos besoins. Afin d'améliorer la qualité de notre évaluation sans sacrifier le temps de compilation, nous proposons un modèle temporel permettant d'obtenir plus d'informations sur les performances temporelles de nos protocoles.

2.2. Modèle d'analyse temporelle

Un modèle d'analyse temporel pour protocole de cohérence consiste à calculer pour chaque évènement (ici des accès effectués par les processus aux données partagées) un ensemble de statistiques concernant le comportement du système. Ces statistiques peuvent inclure le nombre, le type et la taille des messages générés par le protocole, le routage de ces messages sur la puce, le temps de transfert des messages en nombre de cycles du réseau. En comparaison avec un simulateur d'architecture complet, cette approche permet de simplifier de manière importante le calcul global en ne traitant que les évènements significatifs (aucun code utilisateur n'est exécuté et seule est considérée la liste des accès aux données partagées) et en ne déroulant les calculs que sur les éléments logiciels et matériels pertinents (nous ne considérons que le réseau sur puce et la hiérarchie des caches). Dans une première approche [4], le modèle temporel calcule le nombre de cycles du réseau nécessaires à la réalisation de chaque accès mémoire. L'analyse se base sur une trace d'accès mémoire obtenue par instrumentation du binaire de l'application avec l'aide de l'outil PinTool. Chaque entrée de la trace contient des informations sur le processus qui effectue l'accès, sur la donnée accédée et si c'est une lecture ou une écriture. A partir de là, pour chaque entrée, le modèle temporel calcule un certain nombre d'évènements en fonction de l'état actuel de la puce (taux d'occupation des caches principalement) et du protocole de cohérence choisi (les protocoles se comportent différement, notamment sur les messages échangés). Ces statistiques peuvent alors être analysées globalement (effectuer la somme des cycles réseaux requis pour accéder aux données) ou localement (effectuer l'analyse pour un cœur, un processus, une donnée partagée, un routeur). Ces analyses sont ensuite utilisées pour évaluer une fonction objectif qui doit permettre de comparer si le protocole de cohérence choisi est meilleur qu'un autre selon un ou des critères (performance, consommation, température). Afin de présenter plus simplement le modèle nous choisissons de fixer certains paramètres. Nous considérons un réseau sur puce dont la topologie est en maille *2D* et sur lequel est déployé un algorithme de routage *X-Y*. Sur cette architecture, il existe deux possibilités d'accès aux données : dans la mémoire sur puce (sur le cache d'un cœur) ou bien dans la mémoire externe. En fonction du modèle de cohérence et du comportement de l'application, les données peuvent migrer de cache en cache, de ou vers la mémoire externe. Le modèle considère 4 types d'accès. Accès local : la donnée est sur le cache local du cœur demandeur. Accès voisin : la donnée est disponible dans le cache du voisin direct. Accès distant dans la puce : la donnée est disponible sur le cache d'un cœur distant. Accès en mémoire externe : la donnée n'est pas disponible sur la puce. La requête d'accès est transférée vers la mémoire externe. Le modèle propose pour chaque type d'accès des formules de mise à jour des statistiques. Nous ne donnons ici que les formules correspondant au temps d'accès au cache voisin (l'ensemble du modèle est disponible dans [4]). Le temps d'accès comprend le temps nécessaire à la localisation de la donnée et le temps d'accès en lecture/écriture dans le cache local. Il s'ajoute à cela un temps lié à l'acheminement de la donnée du cache voisin vers le cache local. Ce temps dépend principalement de la configuration du *NoC*. Dans le cas d'une lecture, la donnée est transférée sur le *NoC* et ensuite copiée localement. Lorsque le type d'accès est une écriture il n'est pas nécessaire de copier la

donnée sur le cache local, le calcul des temps d'accès pour la lecture (resp. écriture) sont alors :

$$\begin{aligned} T_{local-access-rd} = T_{meta-rd-cache-proc} + T_{ctrl-req-NoC} + T_{data-rd-NoC} \\ + T_{data-NoC} + T_{data-wr-cache-NoC} + T_{data-rd-cache-proc} \end{aligned} \quad (1)$$

$$T_{local-access-wr} = T_{meta-rd-cache-proc} + T_{ctrl-req-NoC} + \quad T_{data-wr-cache-proc} \quad (2)$$

Le modèle considère que la contention réseau est inexistante : tous les paquets à l'entrée d'un routeur sont servis en même temps (pas de notion de file d'attente). Cette hypothèse simplifie le modèle temporel, mais ne correspond pas à ce qui se passe en réalité sur le réseau. Une extension de ce modèle à été développée afin d'améliorer la précision en prenant en compte la concurrence entre les requêtes d'accès dans le réseau.

2.3. Modèle d'analyse temporelle étendu

Afin d'améliorer la précision du modèle temporel avec la prise en compte de la contention réseau, nous proposons d'utiliser un modèle TLM (*Transactional-Level Modeling*) [9] pour le réseau-sur-puce. Comme pour le modèle temporel précédent, ce modèle repose sur l'interprétation d'une trace d'accès mémoire pour obtenir l'enchaînement des messages générés par le protocole de cohérence sur une architecture donnée. Contrairement au modèle précédent, il est possible avec le TLM de jouer les messages de manière parallèle et de mettre en valeur la concurrence d'accès aux ressources physiques. Les paramètres du modèle concernent le dimensionnement du *NoC* (ex. la taille des files d'entrée et de sortie, la bande passante), le dimensionnement des paquets de communication qui y circulent (ex. la taille du paquet de contrôle et de données) ainsi que les pénalités temporelles associées à chaque communication ce qui permet de définir le nombre de cycles pour la transmission d'un *flit* (un élément atomique constituant les messages) d'un point A à un point B du réseau. D'autres éléments de configuration peuvent être modifiés tels que les algorithmes de routage et d'arbitrage. Une modélisation de type TLM a pour avantage d'offrir un modèle configurable en fonction des besoins de l'utilisateur tout en restant sur un niveau d'abstraction suffisamment élevé pour rester dans des temps de calcul raisonnables. Les informations de configuration du *NoC* telles que la taille de la puce sont déduites à partir de la trace des communications réseau. La mémoire extérieure est modélisée par un nœud situé à l'extrémité du *NoC* avec des temps d'accès représentatifs d'un nœud NUMA. Une distinction est faite entre les paquets de contrôle et les paquets de données, dont la destination est respectivement un nœud processeur et un nœud mémoire. Enfin les routeurs sont les nœuds du réseau qui gèrent les communications entre l'ensemble des éléments du système. Chaque paire (processeur, mémoire cache) est reliée à un routeur qui assure le lien avec les autres pairs. Le tableau 1 indique les temps considérés dans notre étude et sont issus d'une étude de l'état de l'art des architectures sur puce [8]. Ces temps correspondent à des estimations moyennes qui peuvent être modifiés selon l'architecture représentée.

3. Évaluation temporelle du protocoles de glissement de donnée

Cette section présente des résultats expérimentaux d'analyse temporelle des protocoles de glissement de données à l'aide du modèle temporel proposé. La première analyse porte sur la variation des performances du système en fonction des différentes configurations de plateforme. Une deuxième analyse porte sur l'étude de différentes applications synthétiques (correspondant à des charges mémoire différentes) et sur l'étude de l'impact de la variation du niveau de parallélisme sur les performances du *NoC*.

Temps d'accès	Accès local	Accès voisin	Accès distant		Accès extérieur
			Chaînage	Suivi	
$T_{meta-rd-cache-proc}$	7	7	7	7	7
$T_{ctrl-req-NoC}$	0	4	$2 \times r$	$2 \times r$	$2 \times r$
$T_{data-rd-NoC}$	0	7	7	7	100
$T_{data-NoC}$	0	$2 \times (2 + Fn)$	$r \times (2 + Fn)$	$r \times (2 + Fn)$	$r \times (2 + Fn)$
$T_{data-wr-cache-NoC}$	0	10	10	10	10
$T_{data-wr-cache-proc}$	0	10	10	10	10
$T_{data-rd-cache-proc}$	0	7	7	7	7

TABLE 1 – Modèle temporel (r correspond au nombre de routeurs traversés par un paquet, et F_n correspond au nombre de *flits* envoyés sur le *NoC*)

3.1. Analyse de la variation de la topologie *NoC*

Le protocole de glissement de données est une approche coopérative entre les mémoires des différents cœurs [5]. Les données sont ainsi autorisées à emprunter l'espace de stockage du voisinage afin de réduire la charge dans les zones les plus stressées de la puce. Le rayon de glissement des données est un paramètre principal de ce protocole. Il permet de définir la taille de l'espace coopératif. La valeur maximale de ce rayon est directement liée à la taille de la puce. Un autre axe de dépendance entre le protocole de glissement et les paramètres du réseau est la structure du voisinage définie par la topologie du réseau. Le modèle temporel étendu basé sur un simulateur *TLM* nous permet de varier plusieurs paramètres du *NoC* telles que sa taille et sa topologie. Nous considérons dans cette étude deux types de topologies 2D : topologie en maille, topologie en tore. Contrairement à la topologie en maille, les cœurs sur les bordures d'une topologie tore ont le même nombre de voisins que ceux situés au milieu de l'architecture. La métrique utilisée pour l'évaluation des performances est la latence d'accès cumulative pour l'ensemble des accès mémoire. Le tableau 2 montre une comparaison entre les latences d'accès obtenues avec une topologie en maille et celles obtenues avec une topologie en tore. Afin d'étudier différentes charges mémoire, le nombre d'accès augmente progressivement entre la première trace (première ligne du tableau) et la cinquième trace (dernière ligne du tableau). Nous constatons dans ces résultats que pour une partie des traces mémoire de meilleures performances sont obtenues grâce à une topologie Tore. Pour les protocoles de glissement de données une topologie en tore permet d'étendre la migration des données sur les extrémités de la puce. La topologie tore peut avoir un effet négatif dans d'autres cas si la coopération entre les cœurs des deux extrémités de la puce pénalise les performances de ces derniers. Le choix d'une configuration du *NoC* est un compromis entre le comportement général de l'application et la répartition de la charge sur la puce.

3.2. Analyse de la contention du *NoC*

Dans cette deuxième analyse nous étudions l'impact de différents niveaux de parallélisme d'accès et différents rayons de glissement. Les messages générés par le *CacheValidator* n'importent aucune information sur les temps de transmission de chaque message. Ce temps de transmission dépend de nombreux facteurs dont le niveau de contention générée dans le *NoC*. La figure 1 montre la variation du taux de contention en fonction des valeurs du rayon de glissement. Cette contention est contrôlée par le paramètre du niveau de parallélisme. En premier lieu le niveau de parallélisme est maintenu à 50%, seul le rayon de migration est varié entre trois valeurs ($valeur_{min} = 1$, $valeur_{moy} = 7$, $valeur_{max} = 14$). Le taux de contention augmente généralement avec l'augmentation du rayon de glissement. Cette information permet de limiter le rayon de glissement afin d'éviter des taux de contention du réseau très élevés ce qui dégraderait drastiquement les performances du *NoC*.Ensuite, le rayon de glissement

est fixé à 1 et c'est le niveau de parallélisme qui prend des valeurs entre : 0% et 100%. Notons bien qu'un taux de 0% correspond au cas où tous les accès sont traités séquentiellement. Les mesures obtenus montre que le niveau de contention ne peut pas être négligé dans tous les cas à la différence du modèle temporel initial qui considère un *NoC* parfait et sans contention. Le tableau 2 montre les taux de contention correspondants aux différents niveaux de parallélisme. Ce tableau montre que pour les traces avec une plus faible charge mémoire (première trace) une simulation complètement parallèle des accès double le taux de contention du réseau. En revanche, pour des traces plus stressées (avec un plus grand nombre d'accès mémoire) le niveau de parallélisme de la simulation a moins d'impact sur la dégradation des performances du *NoC*. Ce dernier constat est expliqué par le fait que la migration des données dans la puce génère un grand trafic d'abord pour éloigner les données et ensuite pour les récupérer. C'est un des désavantages d'utiliser une telle approche. Il n'est donc pas judicieux d'utiliser le glissement de données avec toutes les charges mémoire et avec tout type de réseau *NoC*. Les résultats correspondent aux même traces mémoire utilisé dans le tableau précédent.

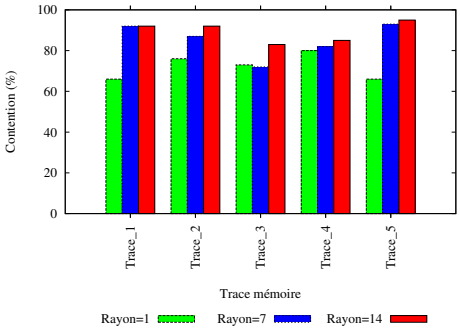

FIGURE 1 – Le taux de contention en fonction du rayon de glissement.

Topologies	Maille	Tore	Niveaux de parallélisme	0%	25%	50%	75%	100%
Latences cumulatives	7778 cycles	9813 cycles	Taux de contention	39%	39%	28%	45%	57%
	89625 cycles	87588 cycles		83%	88%	93%	89%	95%
	828885 cycles	772475 cycles		80%	76%	87%	87%	89%
	1836805 cycles	1969467 cycles		66%	76%	73%	80%	66%
	3719665 cycles	3584895 cycles		97%	81%	71%	78%	92%

TABLE 2 – À gauche :Taux de contention pour différents niveaux de parallélisme allant de 0% (simulation séquentielle) à 100% (simulation parallèle). À droite : Les latences cumulatives obtenues pour chaque trace mémoire étudiée avec les deux topologies maille(colonne gauche) et tore.

4. Conclusion

La proposition d'un modèle temporel de haut niveau permet d'obtenir une évaluation en nombre de cycles des performances des protocoles de cohérence à un coût raisonnable ce qui répond à notre besoin. L'extension de ce modèle en utilisant un simulateur TLM de NoC permet d'augmenter la précision du modèle en prenant en compte l'effet de la contention du réseau sur les performances globales des protocoles. Une telle simulation offre plus de flexibilité à l'utilisateur dans la configuration de la plateforme cible (topologie, taille de la puce, algorithme de routage, etc). Elle permet également de configurer le niveau de stress du NoC à travers le paramètre du niveau de parallélisme qui détermine le taux de communications injectées en parallèle dans le réseau. Le temps de simulation est de l'ordre de quelques milliers de millisecondes pour des traces d'accès complexes, ce qui reste raisonnable par rapport à d'autres outils de simulation. Cet avantage est d'une grande importance dans le cadre de ces travaux car il peut s'intégrer facilement dans la processus de compilation proposé. Enfin, l'étude des performances de protocoles présentée est un exemple d'analyse pouvant orienter le choix et le paramétrage des protocoles de cohérence.

Bibliographie

1. Open systemc iniative osci, systemc documentation., 2004.
2. Open virtual platforms, 2008.
3. Aubry (P.), Beaucamps (P.-E.), Blanc (F.), Bodin (B.), Carpov (S.), Cudennec (L.), David (V.), Doré (P.), Dubrulle (P.), De Dinechin (B. D.) et al. – Extended cyclostatic dataflow program compilation and execution for an integrated manycore processor. *Procedia Computer Science*, vol. 18, 2013, pp. 1624–1633.
4. Chaker (H.), Cudennec (L.), Dahmani (S.), Gogniat (G.) et Sepúlveda (M. J.). – Cycle-based model to evaluate consistency protocols within a multi-protocol compilation tool-chain. – In *Proceedings of the 2015 International Workshop on Code Optimisation for Multi and Many Cores*, p. 8. ACM, 2015.
5. Dahmani (S.), Cudennec (L.) et Gogniat (G.). – Introducing a data sliding mechanism for cooperative caching in manycore architectures. – In *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International*, pp. 335–344. IEEE, 2013.
6. Genko (N.), Atienza (D.), De Micheli (G.), Mendias (J. M.), Hermida (R.) et Catthoor (F.). – A complete network-on-chip emulation framework. – In *Design, Automation and Test in Europe, 2005. Proceedings*, pp. 246–251. IEEE, 2005.
7. Marescaux (T.), Mignolet (J.-Y.), Bartic (A.), Moffat (W.), Verkest (D.), Vernalde (S.) et Lauwereins (R.). – Networks on chip as hardware components of an os for reconfigurable systems. *In : Field Programmable Logic and Application*, pp. 595–605. – Springer, 2003.
8. Patterson (D. A.) et Hennessy (J. L.). – *Computer organization and design : the hardware/software interface*. – Newnes, 2013.
9. Sepulveda (J.), Strum (M.) et Wang (J.). – A tlm-based network-on-chip performance evaluation framework. – In *Proc. 3rd Symposium on Circuits and Systems, Colombian Chapter*, pp. 54–60, 2007.
10. Ventroux (N.), Guerre (A.), Sassolas (T.), Moutaoukil (L.), Blanc (G.), Bechara (C.) et David (R.). – Sesam : An mpsoc simulation environment for dynamic application processing. – In *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*, pp. 1880–1886. IEEE, 2010.
11. Wolkotte (P. T.), Holzenspies (P. K.) et Smit (G. J.). – Fast, accurate and detailed noc simulations. – In *Networks-on-Chip, 2007. NOCS 2007*, pp. 323–332. IEEE, 2007.