Proposition de sujet de stage de Master 2

Étude des stratégies de programmation et de performances du commutateur réseau Intel Tofino 2

Équipes d'accueil du laboratoire ICube : équipes Réseaux et ICPS
Encadrants : Jean-Romain Luttringer, Pascal Mérindol, Cristel Pelsser et Philippe Clauss

1 Introduction

Lors du routage de paquet dans un réseau, les fonctionnalités sous-jacentes sont classées dans deux *plans*. Le plan de contrôle contient les fonctionnalités complexes (p. ex., élaboration de la topologie et prises de décision) et le plan de données transfère les paquets de données reçus vers la bonne interface, telle que calculée par le plan de contrôle.

Le plan de contrôle effectuant des opérations complexes, ce-dernier est principalement logiciel et utilise le CPU. A l'inverse, le plan de données devant assurer un traitement des paquets à *line-rate* (respectant le débit maximum annoncé), ce dernier reposait traditionnellement sur du matériel dédié optimisé aux performances prévisibles (p. ex. des ASICs). En particulier, un type de mémoire particulier, la TCAM, est utilisé pour assurer un *longest prefix match* extrêmement rapide (quelques cycle d'horloge [1]).

Cependant, depuis peu, le plan de données des équipements n'est plus aussi immuable qu'auparavant. Les puces Tofino [2] permettent de personnaliser le plan de donnée d'un commutateur tout en garantissant un débit extrêmement élevé (400 Gpbs par interface) : certains commutateurs embarquant la puce Tofino 2 peuvent garantir jusqu'à 12,8 Tbit/s grâce à la technologie 7nm et prend en en charge une vitesse de port allant jusqu'à 400 GbE pour les environnements à très grande échelle. L'équipe Réseaux du laboratoire ICube dispose de plusieurs de ces équipements.

Sur ces équipements, le plan de donnée (c-à-d, en principe et presque exclusivement, la manière dont le commutateur décide de l'interface de sortie adéquate pour chaque destination) peut être programmé via le langage P4, offrant la possibilité de branchements conditionnels et de calculs simples. Le plan de données peut donc être intelligent, par exemple en réagissant directement aux pannes (sans dépendre d'une mise à jour du plan de contrôle) et aux changements topologiques en général, et/ou encore en adoptant des stratégies de routage limitant la consommation mémoire et énergétique du commutateur pour améliorer ses performances en général. Notamment, P4 permet un contrôle relativement fin des ressources utilisées, permettant par exemple de limiter l'utilisation de la TCAM, une ressource particulièrement énergivore.

1.1 Les mémoires TCAM

Les cellules de ces mémoires TCAM [3] ne sont pas accessibles à travers leurs adresses, comme pour les mémoires classiques, mais à travers leurs contenus. Pour une mémoire CAM binaire, l'application utilisatrice fournit un mot de donnée et la mémoire CAM recherche dans toute la mémoire pour

voir si ce mot y est stocké. Si le mot est trouvé, la CAM renvoie une liste d'une ou plusieurs adresses où le mot a été trouvé. Ainsi, une CAM est l'équivalent matériel de ce que l'on appelle un tableau associatif en logiciel. La CAM ternaire (TCAM) permet un troisième état de correspondance appelé «X» ou «quelconque» pour un ou plusieurs bits dans le mot de donnée stocké, permettant l'ajout de flexibilité dans la recherche. Par exemple, une CAM ternaire pourrait avoir un mot stocké de «10XX0» qui correspondra aux recherches des mots «10000», «10010», «10100», ou «10110». La flexibilité de recherche additionnelle vient avec un coût additionnel par rapport aux CAM binaires parce que la cellule de mémoire interne doit à présent encoder les trois possibilités d'état au lieu des deux d'une CAM binaire. Cet état additionnel est typiquement implémenté en ajoutant un bit de masque à chaque cellule mémoire.

Au contraire de la mémoire RAM classique, qui a des cellules de stockage simples, chaque bit de mémoire individuel dans une CAM complètement parallèle doit avoir son propre circuit de comparaison pour détecter une correspondance entre le bit stocké et le bit d'entrée. En plus, les sorties de correspondances de chaque cellule du mot de donnée doivent être combinées pour aboutir à un signal correspondant au mot entier. Le circuit additionnel augmente la taille physique de la puce CAM ce qui augmente le coût de fabrication, et **augmente également la puissance de dissipation** [4] puisque chaque circuit de comparaison est actif à chaque cycle d'horloge. En conséquence, une CAM n'est utilisée que dans les applications spécialisées où la vitesse de recherche ne peut pas être atteinte en utilisant une méthode moins coûteuse.

1.2 Le langage de programmation P4

Considéré comme une évolution du *Software Defined Networking (SDN)*, le langage P4 permet de programmer la façon dont le flux est traité par l'acheminement de paquets sur du matériel de transmission de paquets réseaux tels que des routeurs, les commutateurs ou les pare-feux, qu'ils soient matériels ou logiciels. Comme son nom «Programmation de processeurs indépendants des protocoles» l'indique, le langage ne tient pas compte du format d'un paquet. En effet, les développeurs déclarent le traitement d'un paquet dans un programme écrit en langage P4, et le compilateur le met par la suite au format souhaité selon le matériel cible. La programmation en langage P4 est notamment utilisée pour mettre en œuvre les fonctions de transfert de niveau 3 et les fonctions INT. En plus des instructions du langage lui-même, des pragmas spécifiques au compilateur utilisé peuvent être ajoutés, afin de contrôler explicitement la localisation de certaines données (TCAM ou SRAM).

2 Objectifs du stage

Il existe différents compilateurs P4, par exemple BMv2 et Tofino. Alors que le premier est ouvert, le second est propriétaire. Lors de ce stage, l'objectif est d'explorer les choix laissés au programmeur P4 en fonction de l'environnement utilisé. Il s'agira dans un premier temps d'implémenter des stratégies et des solutions de re-routage efficace afin de réduire les temps de coupure en cas de changement topologique : comment implémenter efficacement ces fonctionnalités dans le plan de données ? Quelles sont les performances observées en fonction des choix d'implémentation (mémoire, nombre de stages consommés, energie, etc) ? La seconde étape consistera en l'étude du séquencage des services [5] : comment partager le pipeline lorsque plusieurs fonctionnalités doivent être implémentés en chaîne (par exemple filtrage, équilibrage de charge et re-routage) ? Les ressources étant limitées, comment organiser efficacement les différents codes en espace et en temps ? Enfin, l'impact énergétique des équipement réseaux étant non négligeable [6], l'étudiant se demandera quels sont les leviers disponibles pour réduire

la consommation d'énergie. Dans l'environnement Tofino, cela revient à déterminer si l'on peut influer sur la consommation en utilisant des mémoires différentes et en jouant sur la profondeur de la chaîne de traitement des paquets. En BMv2, il est possible d'étudier le compilateur afin de proposer des transformations de code et ainsi d'utiliser des éléments moins consommateurs d'énergie.

Les trois principales missions sont résumées ici :

— Étude des performances (au sens large) selon diverses stratégies de (re-)routage dans le plan de données ;

— Analyse du chaînage de fonctionnalités ou comment organiser le partage optimal de ressources ;

— Étude de la consommation énergétique selon diverses stratégies de programmation (selon pragmas, resubmit, nombres d'opérations, voire méthode de séparation des opérations dans le data-plane/control-plane).

Références

[1] A. Rasmussen, A. Kragelund, M. Berger, H. Wessing, and S. Ruepp, "TCAM-based high speed longest prefix matching with fast incremental table updates," in *2013 IEEE 14th International Conference on High Performance Switching and Routing (HPSR)*, 2013, pp. 43–48.

[2] Intel, "Intel Tofino Intelligent Fabric Processor," https://www.intel.com/content/www/us/en/products/network-io/programmable-ethernet-switch.html.

[3] P. K. Sisira, N. Aswathy, B. Prameela, and A. George, "Ternary content addressable memory," *International Journal of Engineering and Advanced Technology (IJEAT)*, vol. 9, no. 4S, pp. 2249–8958, May 2020.

[4] V. Ravikumar and R. Mahapatra, "TCAM architecture for IP lookup using prefix properties," *IEEE Micro*, vol. 24, no. 2, pp. 60–69, 2004.

[5] H. Soni, M. Rifai, P. Kumar, R. Doenges, and N. Foster, "Composing dataplane programs with μP4," ser. SIGCOMM '20. New York, NY, USA : Association for Computing Machinery, 2020, p. 329–343. [Online]. Available : https://doi.org/10.1145/3387514.3405872

[6] S. Tabaeiaghdaei, S. Scherrer, and A. Perrig, "Carbon Footprints on Inter-Domain Paths : Uncovering CO2 Tracks on Global Networks," 2022. [Online]. Available : https://arxiv.org/abs/2211.00347