Nouvelles fonctions dans les interfaces de communication pour l'augmentation des performance réseau des machines multi-processeur

Éric Lemoine

Soutenance de thèse

8 juillet 2004

Contexte général

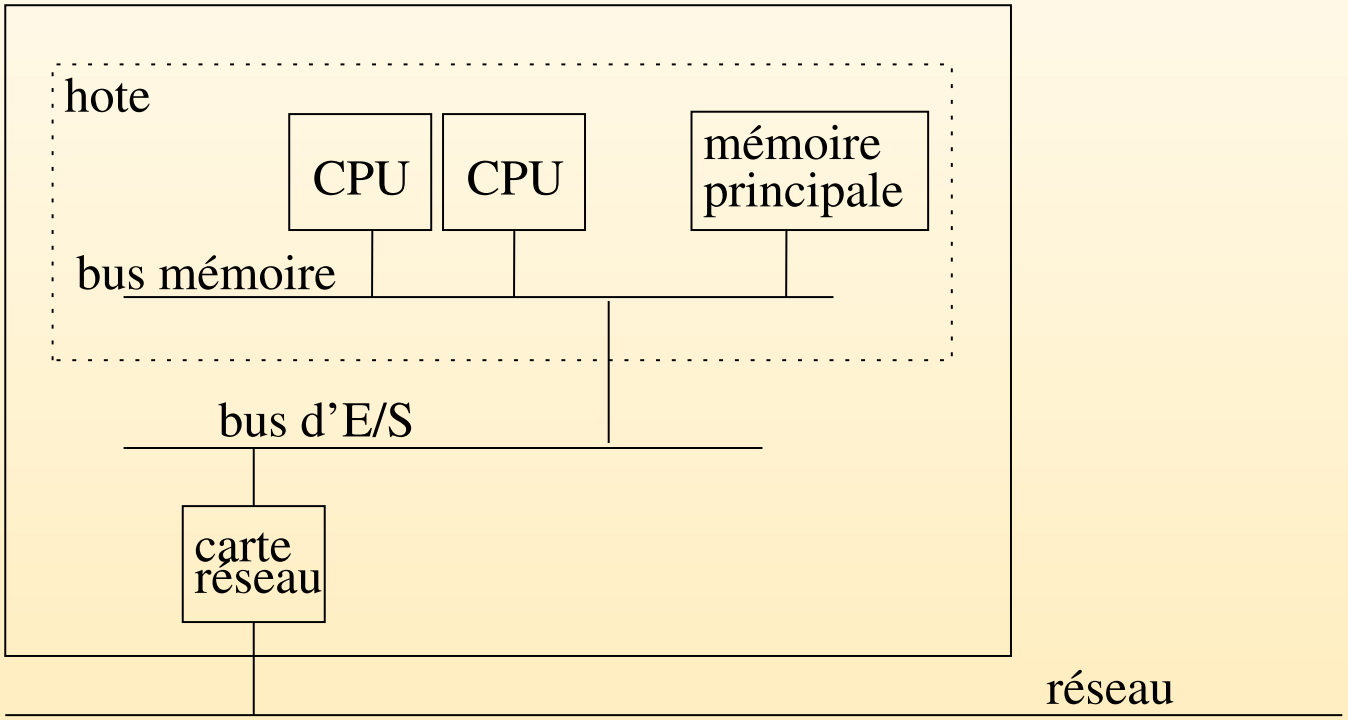
- Les débits des réseaux ne cessent d'augmenter (Ethernet 10Gbps)
- Les besoins en haut débit augmentent (multimédia, stockage)

- Les technologies mono-processeur et mémoire des machines d'extrémité ont du mal à suivre

▷ Problème : comment faire face à cet écart de performance entre la machine d'extrémité et le réseau ?

Contexte d'étude

- Architecture machine considérée :



- Réseaux IP
- Machines serveur

Performance réseau de la machine d'extrémité

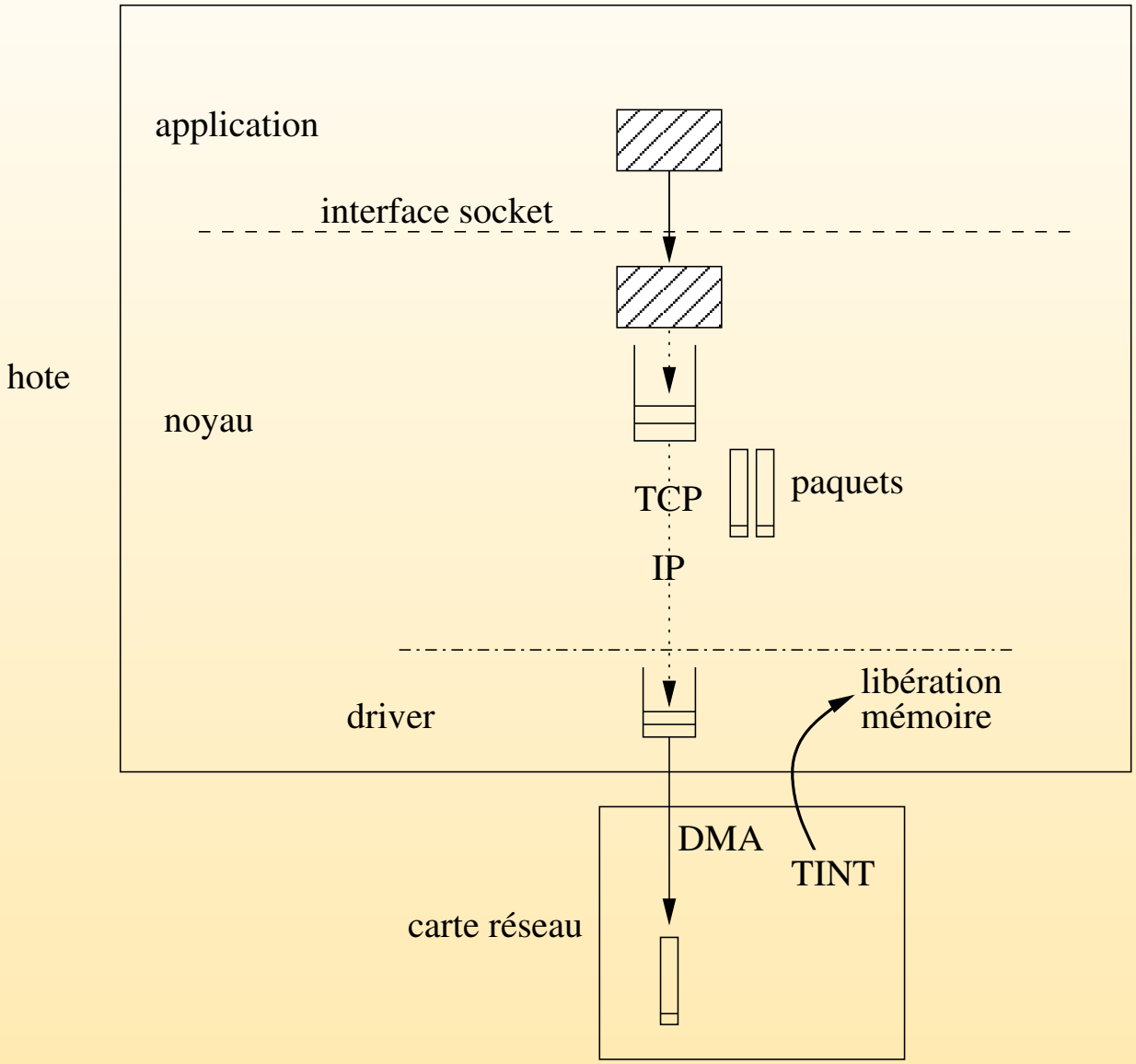
Performance de l'exécution des traitements relatifs au réseau (''traitements réseau'') par le(s) CPU de la machine d'extrémité

Traitements réseau :

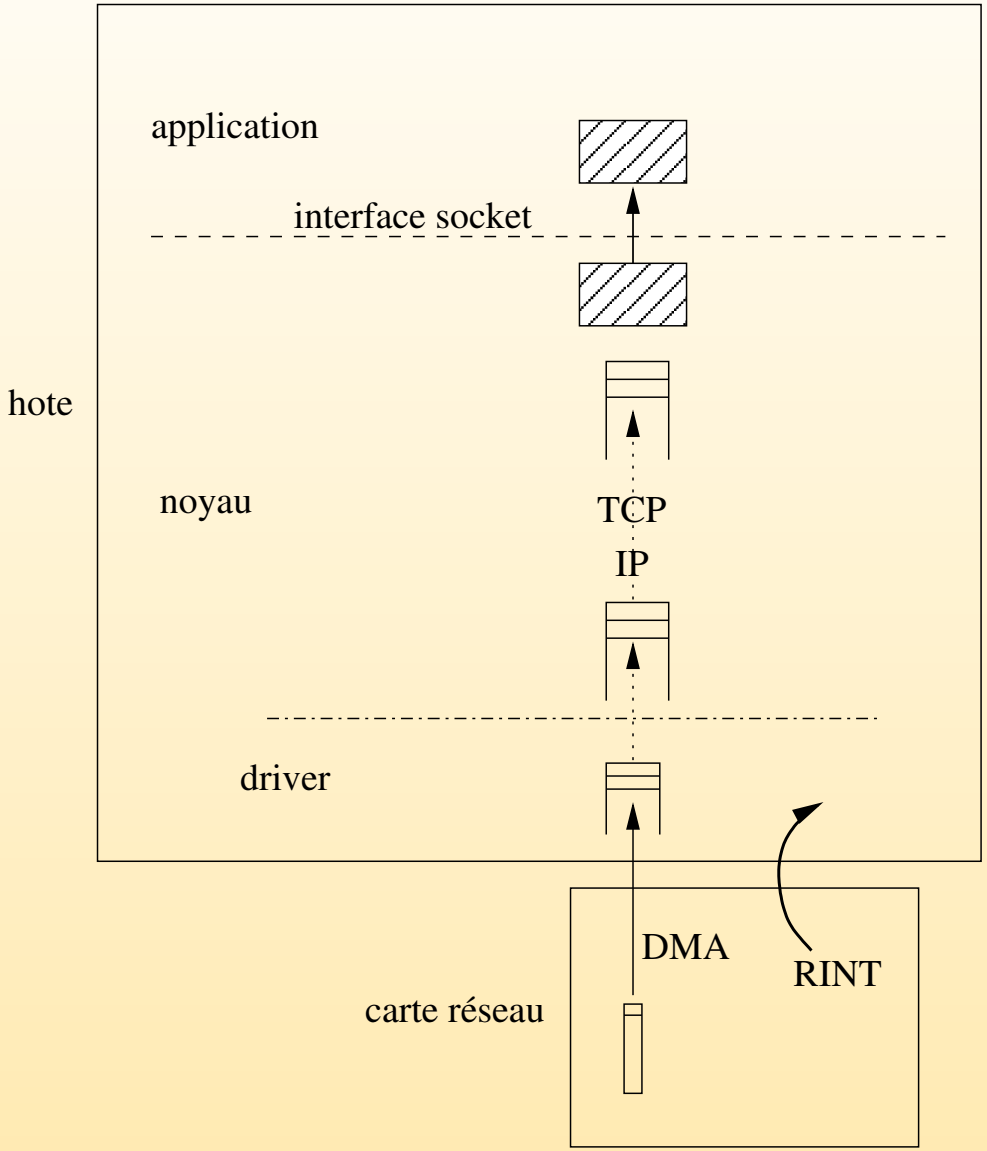
- Interface entre le noyau du système d'exploitation et la carte réseau (*driver*)
- Pile de protocoles (TCP/IP)
- Interface entre le noyau et l'application (API : Application Programming Interface)

L'ensemble des logiciels du noyau chargé des traitements réseau est appelé ''sous-système réseau'' (du noyau)

Émission de données vers le réseau



Réception de données depuis le réseau



Problématique

- Un certain coût est associé à chaque opération de la chaîne des traitements réseau (interruptions, copies mémoire, exécution des protocoles, etc.)

▷ Comment accélérer toute la chaîne des traitements réseau ?

Plan de l'exposé

- État de l'art
- Sous-système réseau NAPI : description et étude
- Proposition d'une nouvelle architecture réseau : KNET
- Étude expérimentale de KNET
- Conclusions et perspectives

Plan de l'exposé

- État de l'art
- Sous-système réseau NAPI : description et étude
- Proposition d'une nouvelle architecture réseau : KNET
- Étude expérimentale de KNET
- Conclusions et perspectives

Plan de l'état de l'art

Techniques pour l'accélération des traitements réseau dans les machines d'extrémité

Classifications des techniques :

- Réduction des coûts logiciels
  - *Réduction des coûts par octet*
  - *Réduction des coûts par paquet*
- Emploi d'unités matérielles additionnelles

Réduction des coûts par octet

Élimination des copies mémoire

- Manip. tables de pages [Keng,USENIX'96] [Chase,IEEEComm'01]
- API avec de nouvelles sémantiques [Druschel,SOSP'93] [Brustoloni,OSDI'96]
- Transfert de fichiers depuis le cache de syst. de fichiers (`sendfile()`) ✔
- Protocole d'accès à la mémoire distante (RDMA [RDMA consortium])

Élimination des calculs des sommes de contrôle (*checksum*)

- Intégration calcul du *checksum*/copie mémoire ✔
- Calcul du *checksum* lors des transferts sur le bus E/S (*checksum offload*) ✔

(✔mis en œuvre et utilisé)

Réduction des coûts par paquet

Augmentation de la taille des paquets (réellement : *jumbo frames*, virtuellement : TCP Segment Offload) ✔

Réduction de la fréquence des interruptions (RINT et TINT) par *coalescence*

- Avec aide de la carte réseau (*interrupt coalescing*) ✔
- Par modification du noyau [Mogul,TOCS'97] [Salim,USENIX'01] ✔

(✔mis en œuvre et utilisé)

Emploi d'unités matérielles additionnelles

Unité dédiée (et spécialisée) [Canadia,CAP'88] [Cooper,CAP'90]

- *TCP Offload Engine* (TOE) : TCP/IP sur la carte réseau

Utilisation simultanée des CPU d'un multi-CPU (SMP) pour les traitements relatifs au réseau [Salehi,1996]

- Parallélisme par connexion [Nahum,OSDI'94] [Yates,Phd 97] : les connexions (flux) et non les paquets sont distribuées parmi les CPU

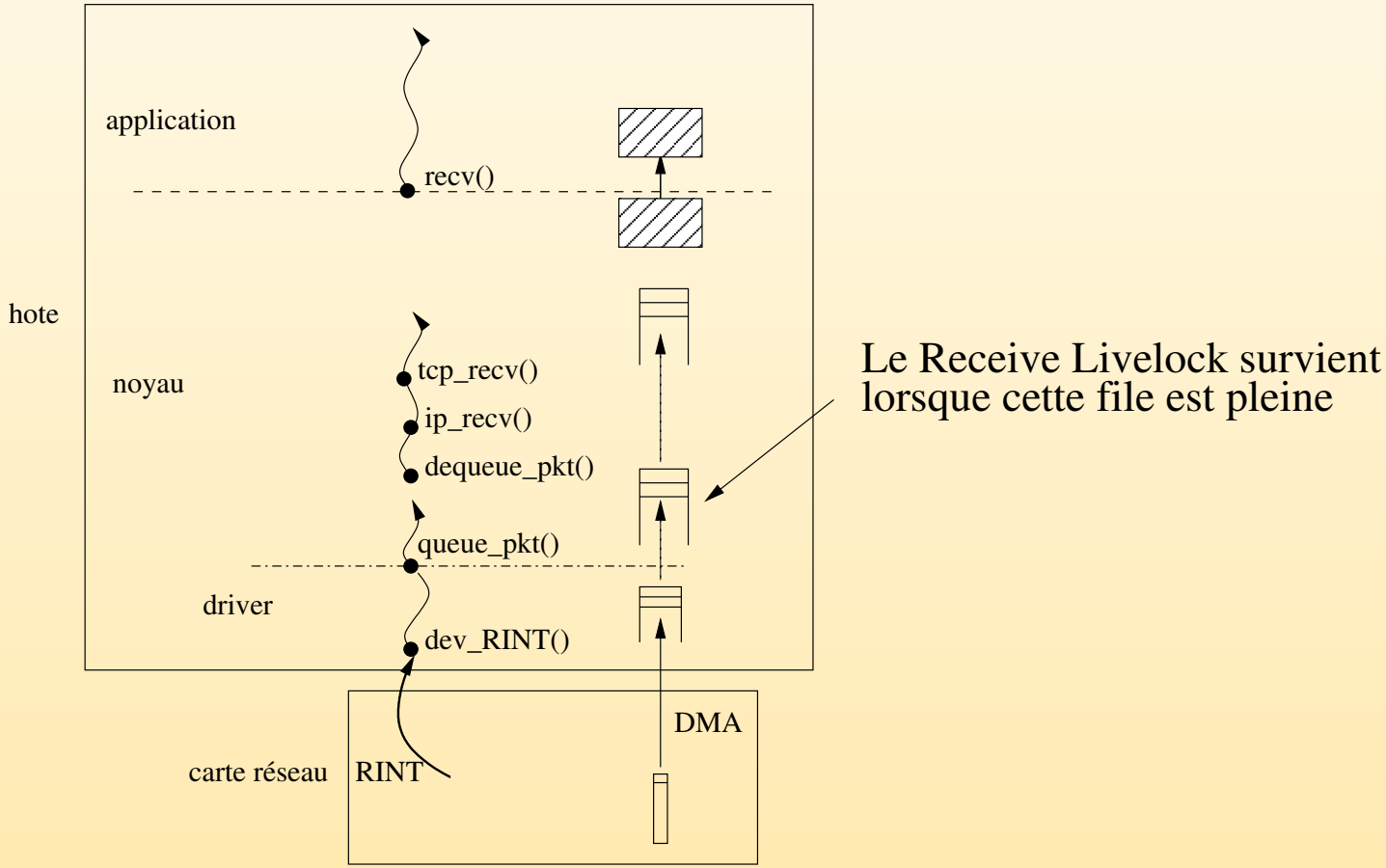
Plan de l'exposé

- État de l'art
- Sous-système réseau NAPI : description et étude
- Proposition d'une nouvelle architecture réseau : KNET
- Étude expérimentale de KNET
- Conclusions et perspectives

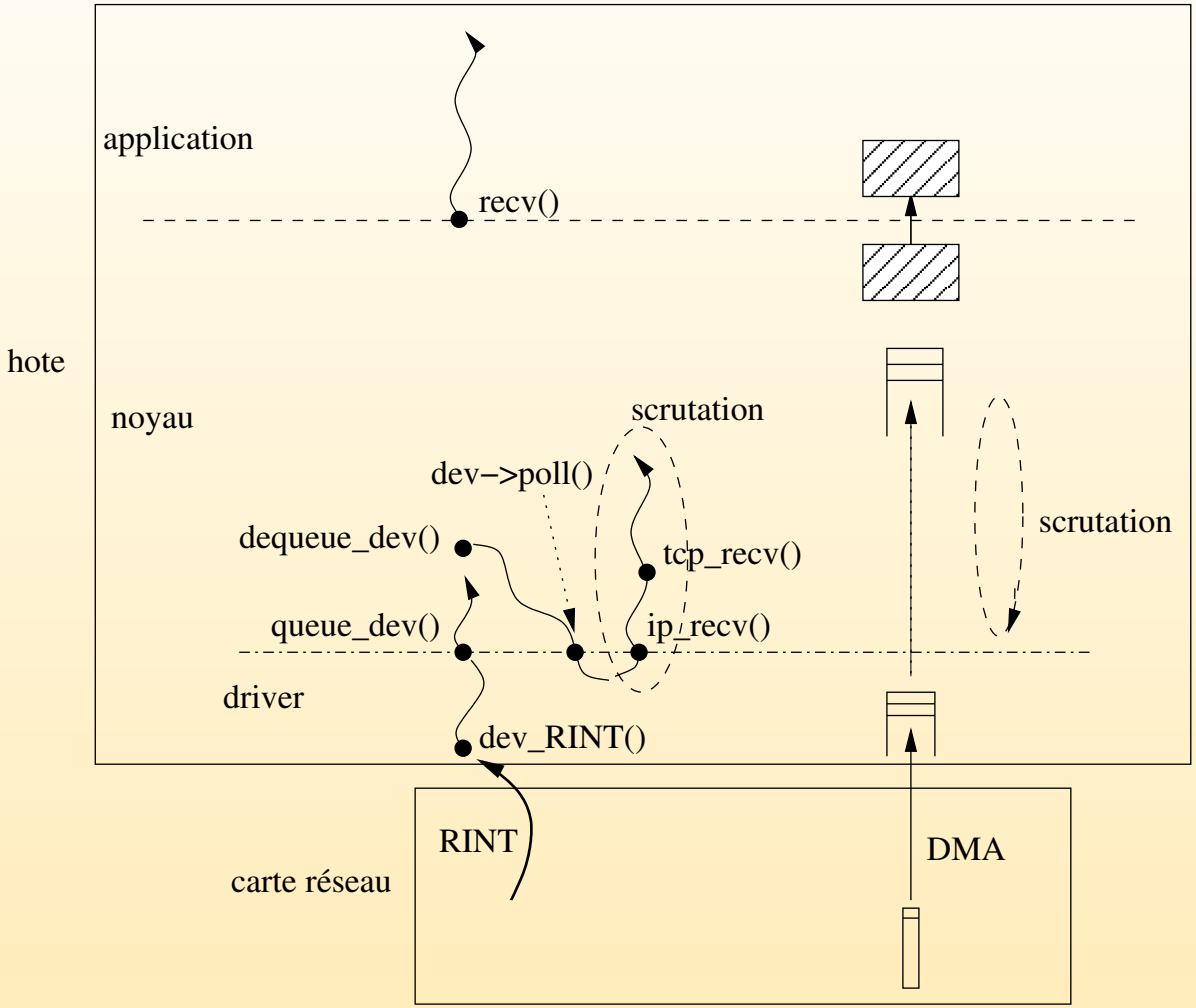
Objectifs de NAPI [Salim,USENIX'01]

Réduire les coûts par paquet associés aux interruptions et éliminer toute possibilité de *Receive Livelock* dans Linux

Le Receive Livelock [Mogul,TOCS'97] :



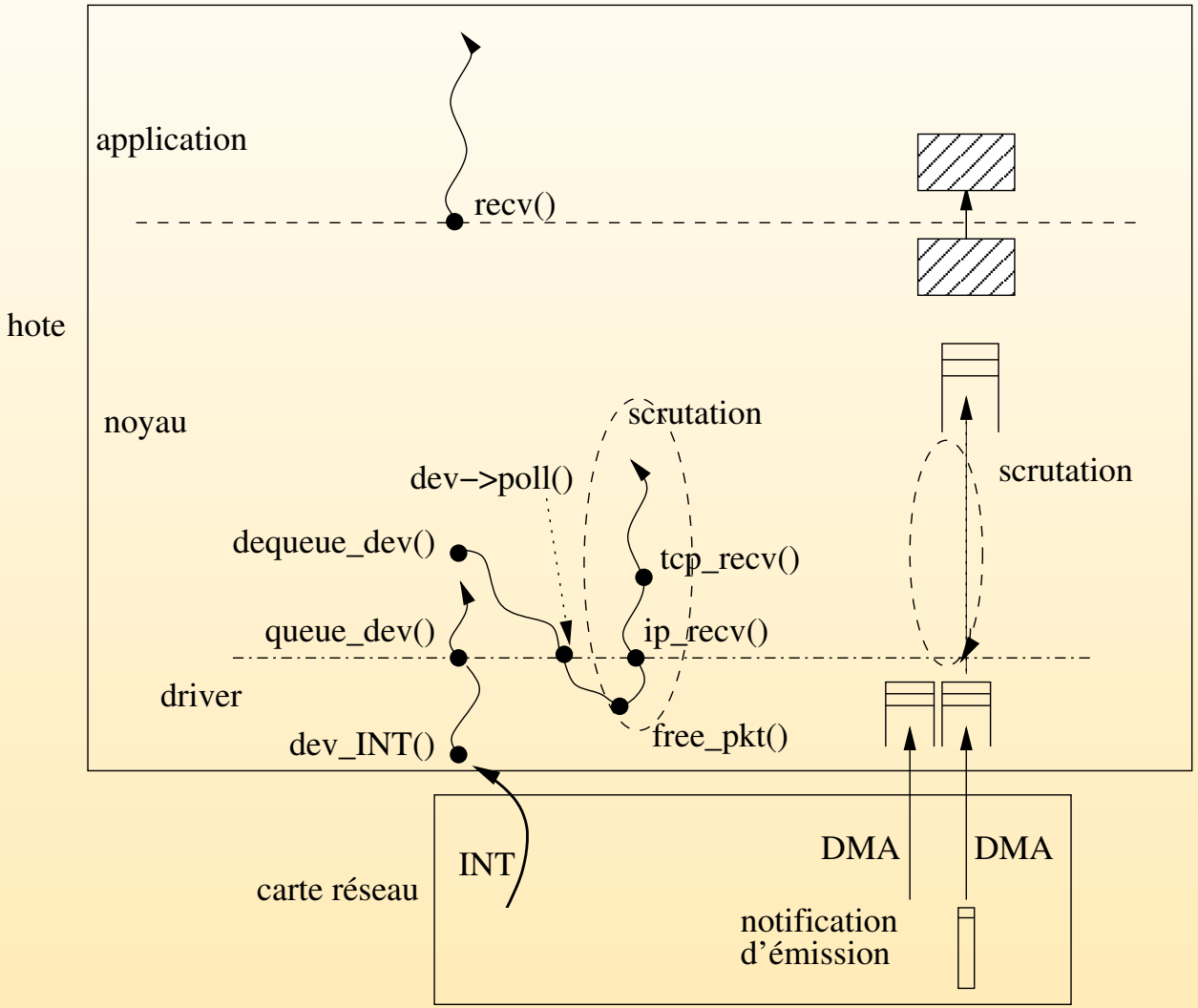
Principe de fonctionnement de NAPI [Salim,USENIX'01]



- Principe : on ne donne pas au système plus de travail qu'il ne peut en fournir
- Tant qu'il y a des paquets dans la file de réception, les RINT sont désactivées et la ''scrutation'' (*polling*) est utilisée
- Quand la file de réception est pleine, la carte jette les paquets

▷ *Fréquence des RINT réduite*

NAPI pour les notifications d'émission



- La libération mémoire des structures de données associées aux paquets transmis et le traitement des paquets reçus sont intégrés

▷ *Fréquence des RINT ET des TINT réduite*

Étude expérimentale de NAPI

Objectifs :

- Obtenir des résultats de performance de NAPI dans le cas d'un réseau à haut débit (Myrinet : 2Gbps)
- Quantifier les bénéfices des *jumbo frames* et de `sendfile()`

Développements logiciels :

- Adaptation de la suite logicielle GM-1.5 (driver + code embarqué) à NAPI
- Mise en œuvre *checksum offload* à l'émission

Cadre expérimental :

- Flux unique entre 2 machines avec le logiciel de mesure de débit Iperf (modifié pour utiliser `sendfile()`)
- 2 scénarios : machine PC 550MHz (L2 : 1Mo, RAM : 512Mo)
  machine PC 2,8GHz (L2 : 512Ko, RAM : 1Go)

Résultats de NAPI - réception UDP

Débit (Mbps), utilisation CPU (%), efficacité (Mbps/MHz)

	PC à 550Mhz	PC à 2,8GHz
GM MTU=1,5Ko	9 Mbps, 100% CPU (0,02 Mbps/MHz)	19 Mbps, 100% CPU (0,01 Mbps/MHz)
GM+NAPI MTU=1,5Ko	340 Mbps, 100% CPU (0,62 Mbps/MHz)	1000 Mbps, 90% CPU (0,40 Mbps/MHz)
GM MTU=9,0Ko	665 Mbps, 100% CPU (1,21 Mbps/MHz)	1982 Mbps, 80% CPU (0,89 Mbps/Mhz)
GM+NAPI MTU=9,0Ko	696 Mbps, 100% CPU (1,27 Mbps/MHz)	1979 Mbps, 60% CPU (1,17 Mbps/MHz)

- NAPI élimine le Receive Livelock
- L'utilisation de paquets de 9Ko offre un gain important ($\times 2$ à $\times 3$ en efficacité)

Résultats de NAPI - émission TCP (MTU=1,5Ko)

Efficacité (Mbps/MHz)

	PC à 550Mhz	PC à 2,8GHz
GM	0,82 Mbps/MHz	0,32 Mbps/MHz
GM+NAPI	0,84 Mbps/MHz	0,35 Mbps/MHz
GM+NAPI+CSUMOFF	1,02 Mbps/MHz	0,37 Mbps/MHz
GM+NAPI+CSUMOFF+TXPOLL	1,20 Mbps/MHz	0,72 Mbps/MHz

- NAPI en réception : de 2 à 9% de gain (GM+NAPI)
- `sendfile()` : de 6 à 21% de gain (GM+NAPI+CSUMOFF)
- NAPI en réception et émission : de 18 à 94% de gain (GM+NAPI+CSUMOFF+TXPOLL)

Conclusion sur NAPI

Bénéfices :

- Élimination du Receive Livelock
- Réduction de la fréquence des RINT et TINT (sans latence additionnelle)
  - ▷ *Gains en performance significatifs*

Mais : par construction, aucun parallélisme dans le traitement des paquets entrants dans NAPI sur SMP

- ▷ Problème ? Oui : 1,20 Mbps par MHz $\Rightarrow$ 8,4GHz pour 10Gbps !

Plan de l'exposé

- État de l'art
- Sous-système réseau NAPI : description et étude
- Proposition d'une nouvelle architecture réseau : KNET
- Étude expérimentale de KNET
- Conclusions et perspectives

Traitements réseau en parallèle

Motivations actuelles :

- Démocratisation des machines multi-CPU à mémoire partagée
- Émergence des machines à CPU multi-flot (*multi-threaded processors*)

Principal contexte applicatif : serveurs de données WWW

Parallélisme par connexion [Nahum,OSDI'94] [Yates,Phd 97]

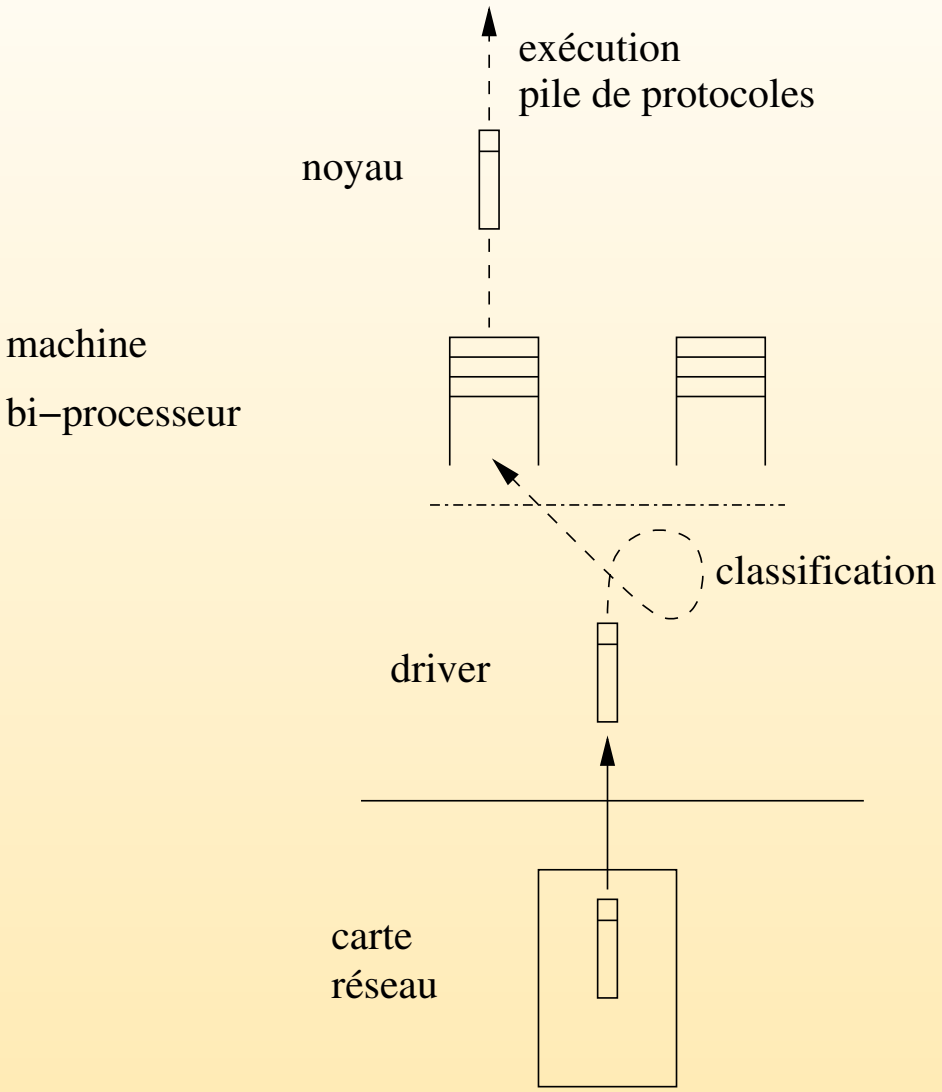
Principe : distribuer les connexions TCP parmi les processeurs (tous les paquets d'une connexion sont traités par le même CPU)

- Permet d'éviter des défauts de cache, des contentions sur des locks, et que les paquets d'une même connexion arrivent dans le désordre
- Accélération montrée par émulation [Yates,Phd 97]

Cas des serveurs WWW (application cible du fait du grand nombre de connexions manipulées) :

- Objectif : émissions simultanées
- Du fait de l'*ACK-clocking* TCP [Jacobson,SIGCOMM'88], émissions simultanées $\Rightarrow$ réceptions simultanées
  - $\Rightarrow$ Tri (''classification'') des paquets entrants avant TCP

Une possibilité : classification dans le noyau



- Contentions sur *locks*
- Défauts de cache
- Faible rapport paquets/RINT OU latence due à la coalescence
- *Receive Livelock*

Architecture réseau KNET

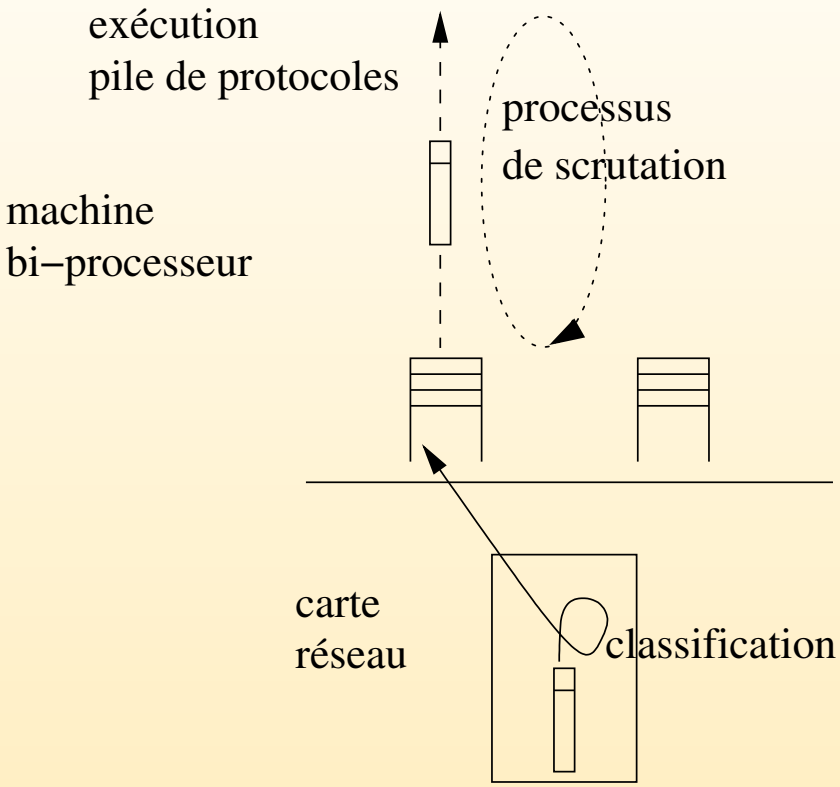
Premier objectif : mise en œuvre efficace de l'approche ''parallélisme par connexion''

- Réduction des locks, des défauts de cache, de la fréquence des RINT
- Élimination du *Receive livelock*

Approche :

- Classification des paquets entrants par la carte réseau
- Utilisation d'autant de files de récep. que de CPU entre le driver et la carte réseau
- Utilisation du mécanisme hybride RINT/scrutation de NAPI
- Activation/dé-activation des RINT d'une manière indépendance pour chaque file (CPU)

Architecture réseau KNET (suite)



- Pas de locks sur les files de réception
- Pas de défauts de cache sur les structures associées aux paquets
- Rapport paquets/RINT élevé ET aucune latence addition-nelle
- Pas de *Receive Livelock*

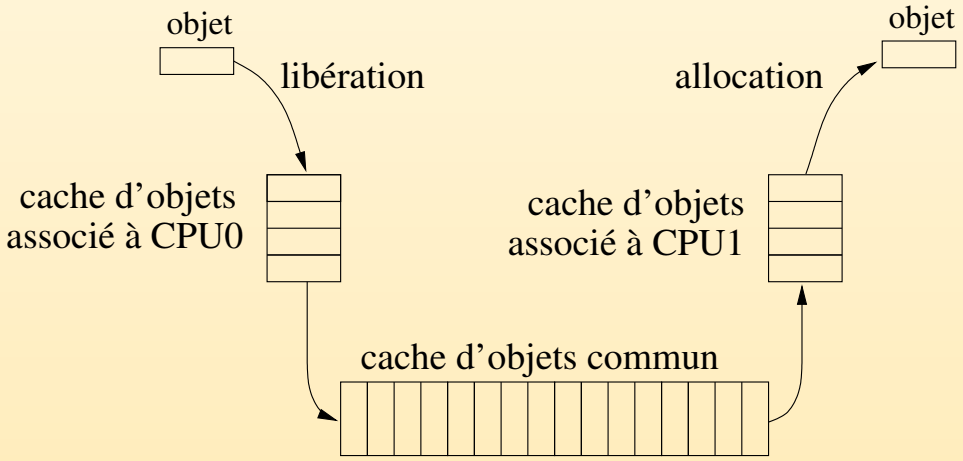
Notifications d'émission dans KNET

- Objectif : minimiser la fréquence des RINT et des TINT
- Principe (identique au cas de NAPI) : intégrer le traitement des notifications d'émission dans le processus de scrutation

Approche immédiate : un seul CPU pour le traitement des notifications d'émission

$\Rightarrow$ Sous-exploitation des caches d'objets de l'allocateur mémoire du noyau

- Contentions sur des locks
- Contentions sur les lignes de caches contenant les structures de données associées aux paquets (`skb` dans Linux)



Distribution des notifications parmi les CPU

Mécanismes :

- Utilisation d'autant de files de notifications d'émission que de CPU dans le driver
- Le driver passe l'identifiant de la file de notifications à la carte réseau au moment de la demande d'émission d'un paquet
- La carte réseau place chaque notification d'émission dans la bonne file

Proposition de 2 stratégies :

- RR (*Round-Robin*) : distribution de type *tourniquet*
- AFFIN (*Affinity*) : libération par le processeur allocateur

Plan de l'exposé

- État de l'art
- Sous-système réseau NAPI : description et étude
- Proposition d'une nouvelle architecture réseau : KNET
- Étude expérimentale de KNET
- Conclusions et perspectives

Mise en œuvre de KNET

Environnement :

- Le réseau Myrinet de Myricom : contrôleurs réseau programmables, suite logicielle disponible (GM), haut débit (2Gbps)
- Noyau Linux 2.4

Développements logiciels :

- Nouvelles fonctions dans le code embarqué de GM-1.5 : classification (utilisant IPsrc), activation/dé-activation des interruptions par CPU, checksum offload
- Nouvelles fonctions dans le driver Linux GM-1.5 : files de réception et de notifications d'émission par CPU, `poll()`
- Module du noyau Linux
  - ▷ *Nouvelle interface driver/noyau*
  - ▷ *Utilisation de threads noyau, une par CPU (attachée au CPU), pour l'exécution de* `poll()` (liée au fait que la carte réseau ne peut pas générer une interruption vers le CPU de son choix)

Cadre expérimental

Serveur HTTP face à 4 clients

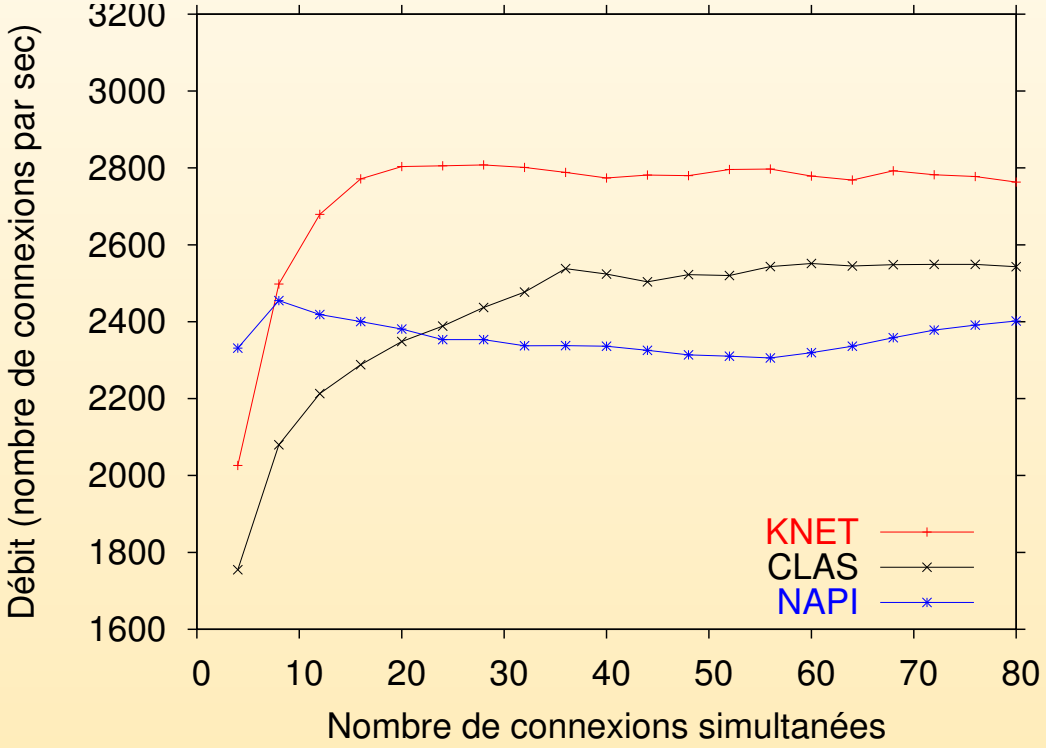
- 2 applications serveur HTTP :
  - *Webfs : serveur simple utilisant select() et sendfile()*
  - *Apache : serveur (populaire) multi-threadé utilisant sendfile()*
- 2 générateurs de trafic :
  - *Sclient (modifié) : fait des requêtes sur un unique fichier*
  - *WebStone : utilise une distribution de fichiers (de 5Ko à 5Mo)*

Machine observée : PC à 4 CPU à 550MHz (L2 : 1Mo, RAM : 512Mo)

Gains mesurés à la saturation de la machine

Effet du parallélisme et de la classification sur la carte réseau

[Webfs/Sclient - fichiers de 20Ko]

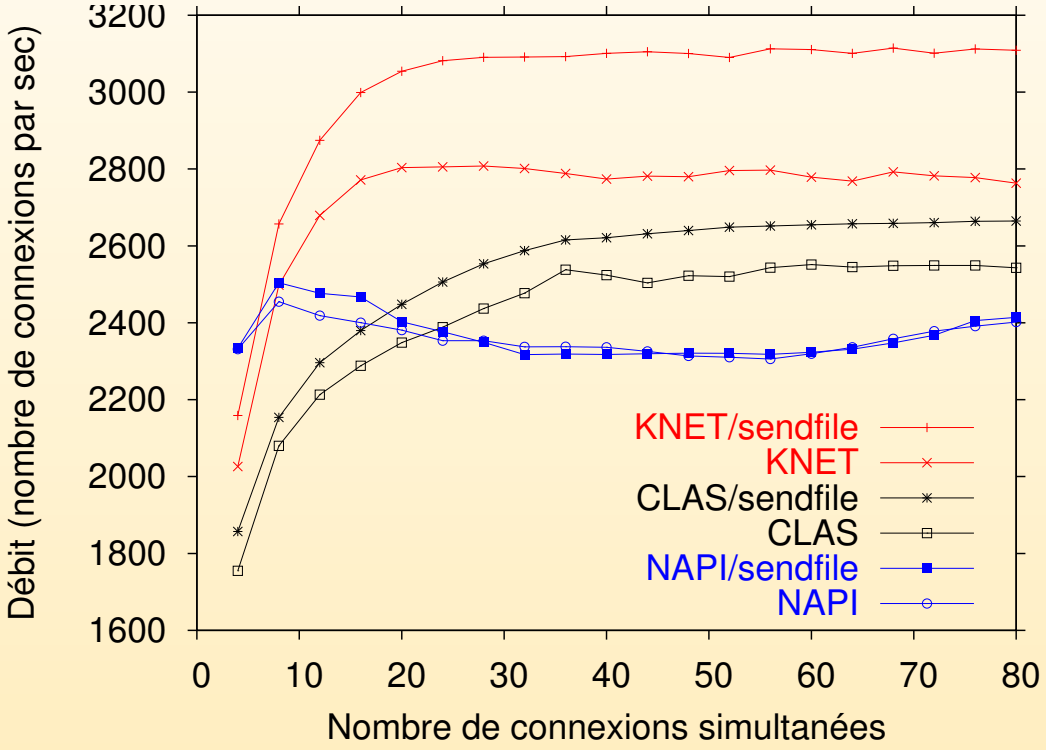


- *KNET apporte 17% de gain par rapport à NAPI - du fait du parallélisme (4 CPU)*
- *KNET apporte 12% de gain par rapport à CLAS (8× moins de* RINT et 6% CPU passé dans les locks pour CLAS)

(CLAS correspond au cas où la classification est opérée dans le noyau)

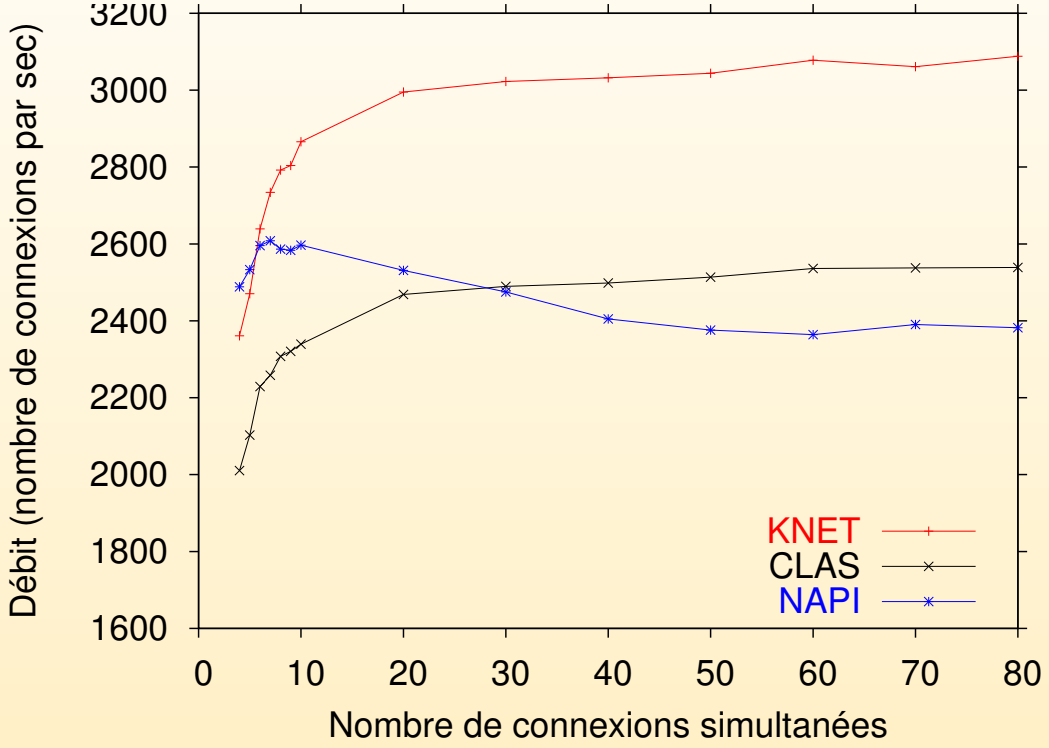
Effet de l'utilisation de `sendfile()`

[Webfs/Sclient - fichiers de 20Ko]



- *KNET apporte 30% de gain par rapport à NAPI et 17% par rapport à CLAS*

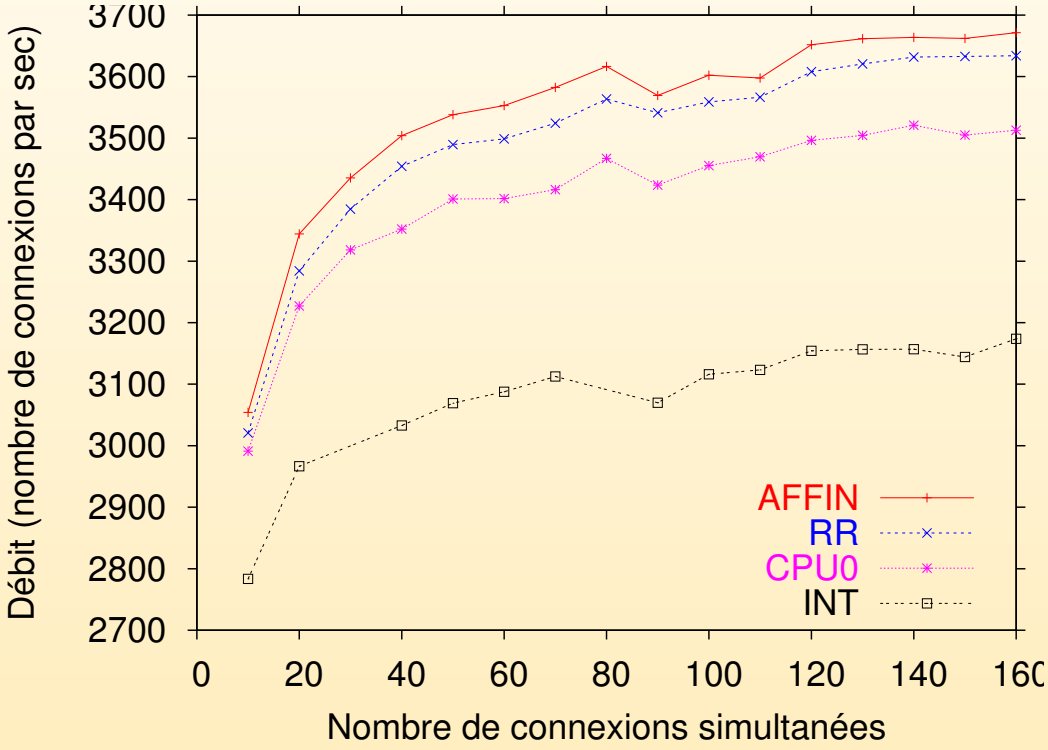
Conditions plus réalistes : Apache/Webstone



- *KNET apporte 27% de gain par rapport à NAPI et 20% par rapport à CLAS*

Notifications d'émission dans KNET - résultats

[Apache/WebStone]



- *L'intégration du traitement des notifications d'émission dans le processus de scrutation apporte 16% de gain*
- *La stratégie AFFIN apporte un gain faible de 2% (file d'émission unique dans Linux entre IP et le driver)*

Conclusion sur KNET

Classifier les paquets dans la carte réseau apporte :

- Robustesse : élimination du Receive Livelock (par construction)
- Efficacité : gain de 20% sur une machine à 4 CPU (par rapport à CLAS)

L'intégration du traitement des notifications d'émission dans le processus de scrutation conduit à un gain de 16% sur une machine à 4 CPU

Plan de l'exposé

- État de l'art
- Sous-système réseau NAPI : description et étude
- Proposition d'une nouvelle architecture réseau : KNET
- Étude expérimentale de KNET
- Conclusions et perspectives

Résumé des contributions de la thèse

- Étude expérimentale du sous-système réseau NAPI
- Proposition de l'architecture réseau KNET
- Mise en œuvre et évaluation de KNET
- Propositions de nouvelles fonctions dans les cartes réseau :
  - *Classificateur de paquets*
  - *Gestion de plusieurs files de réception*
  - *Gestion de plusieurs files de notifications d'émission*
  - *Plusieurs registres d'activation/dé-activation des interruptions*

- Publication : Éric Lemoine, CongDuc Pham et Laurent Lefèvre. Packet classification in the NIC for improved SMP-based Internet servers. Publié dans *Proceedings of IEEE 3rd Internation Conference on Networking (ICN'04)*
- Architecture KNET à l'étude pour introduction dans Solaris et les contrôleurs réseau de Sun

Perspectives

- Nouveaux développements autour de la stratégie AFFIN avec autant de files d'émission que de CPU entre IP et le driver dans le noyau de Linux
- Étude des bénéfices liés à des techniques d''affinité'' entre les processus de l'application et les threads réseau
- Expériences avec des cartes réseau prenant en charge le mécanisme PCI *Memory-Signalled Interrupt*
- Expériences sur les futures machines *multi-core/multi-flot*
- Étude de nouveaux algorithmes de distribution de charge dans le cas de machines avec des architectures complexes (SMP/*multi-core/multi-thread*)

Autres contributions

- Projet Clint : développement et évaluation d'un module SunMPI pour le réseau d'interconnexion Clint

  Membres : Nicolas Fugier, Marc Herbert, Éric Lemoine et Bernard Tourancheau

  Publication : Nicolas Fugier, Éric Lemoine, Marc Herbert et Bernard Tourancheau. MPI for the Clint Gb/s interconnect, A hardware/software design experience. Publié dans *Recent Advances in Parallel Virtual Machine and Message Passing Interface*

- Projet TOAD : étude expérimentale des technologies réseau TOE et RDMA par émulation

  Membres : Roland Westrelin, Nicolas Fugier, Éric Lemoine et Erik Nordmark

  Publication : Roland Westrelin, Nicolas Fugier, Erik Nordmark, Kai Kunze et Éric Lemoine. Studying Network Protocol Offload With Emulation: Approach And Preliminary Results. À paraître dans *Proceedings of the 12th Annual IEEE Symposium of High Performance Interconnects*