HAL
open science

# Evaluation of 10 GbE links in Grid'5000

Romaric Guillier, Ludovic Hablot, Pascale Primet, Sébastien Soudan

## To cite this version:

Romaric Guillier, Ludovic Hablot, Pascale Primet, Sébastien Soudan. Evaluation of 10 GbE links in Grid'5000. [Rapport de recherche] LIP RR-2006-42, Laboratoire de l'informatique du parallélisme. 2006, 2+12p. hal-02102492

**HAL Id: hal-02102492**

**https://hal-lara.archives-ouvertes.fr/hal-02102492v1**

Submitted on 17 Apr 2019

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

*Laboratoire de l'Informatique du Parallélisme*

École Normale Supérieure de Lyon
Unité Mixte de Recherche CNRS-INRIA-ENS LYON-UCBL n° 5668

# *Evaluation of 10 GbE links in Grid'5000*

Romaric Guillier,
Ludovic Hablot,
Pascale Primet,
Sébastien Soudan

November 2006

Research Report N° RR2006-42

**École Normale Supérieure de Lyon**
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : +33(0)4.72.72.80.37
Télécopieur : +33(0)4.72.72.80.80
Adresse électronique : `lip@ens-lyon.fr`

# Evaluation of 10 GbE links in Grid'5000

Romaric Guillier, Ludovic Hablot, Pascale Primet, Sébastien Soudan

November 2006

**Abstract**

This report outline an evaluation of Ethernet links in Grid'5000, a French research grid built with 3000 processors on 9 sites. These sites are connected with dedicated 1 GbE or 10 GbE links. Efficient TCP parameters on this kind of links are different from Internet links are different. We evaluate TCP problems in Grid'5000 links and highlight TCP tuning to obtain good performance in this context.

**Keywords:** TCP, 10 GbE, Grid'5000, TCP tuning

**Résumé**

L'instrument Grid5000 est destiné à l'étude des problématiques, des solutions et des logiciels de grille pour le calcul et le stockage distribué à large échelle.
En 2006, Grid5000 s'est doté d'un réseau privé virtuel composé de liens d'accès à 1 ou 10Gb/s et de longueurs d'onde à 10Gb/s dédiées dans l'infrastructure DWDM de RENATER 4.
Ce rapport présente une étude de l'apport potentiel de cette infrastructure pour les applications distribuées via une évaluation des performances de TCP, protocole prépondérant dans ces applications.
Cette étude met d'abord en lumière l'incidence très importante du paramétrage du protocole dans un tel contexte et explique le faible débit observé tant par l'opérateur que par les utilisateurs. Les résultats obtenus via un calibrage adéquat ou l'utilisation de flux parallèles sont ensuite présentés. Enfin, plusieurs anomalies de configuration et de comportement de l'infrastructure sont exposées.

**Mots-clés:** TCP, 10 GbE, Grid'5000, paramètrage de TCP

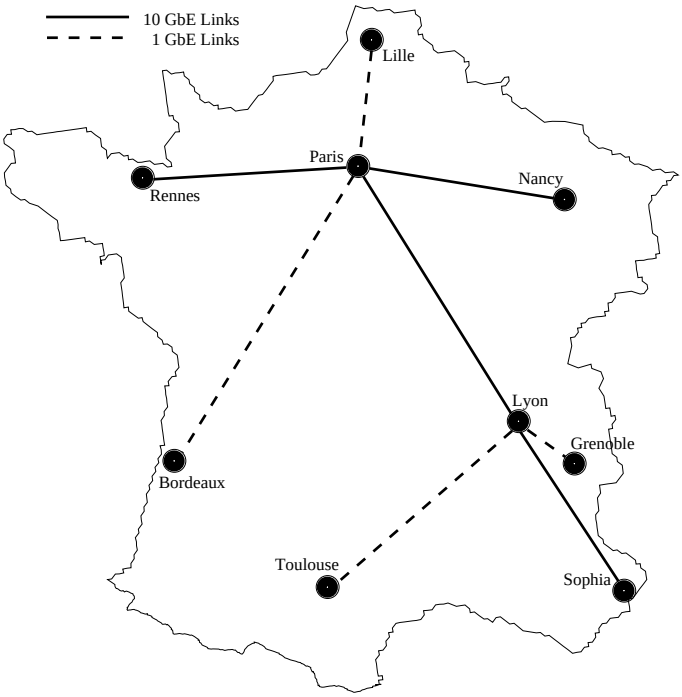



Figure 1: Architecture de Grid'5000

# 1 Introduction

Dans ce rapport, nous présentons les résultats des expériences que nous avons menées depuis Juillet 2006 sur le réseau de Grid'5000 [2]. Cette grille possède un réseau avec des liens dédiés qui devrait permettre aux utilisateurs d'obtenir de bonnes performances pour leurs applications. Nous avons donc mené une campagne de tests pour évaluer les performances de TCP sur les liens de la plateforme.

Comme le montre la Figure 1[1], les différents sites sont reliés par des liens 10 Gb/s[2] ou des liens 1 Gb/s. Le principal but des expériences était donc de vérifier que le débit théorique de ceux-ci était effectivement atteignable. Nous avons donc cherché à obtenir les débits maximaux sur ces liens. De plus, l'apparition du 10 Gb/s sur Ethernet est encore très récente. Cette technologie nouvelle pose des problèmes à TCP qui n'est pas prévu pour des réseaux rapides. Il était donc nécessaire de vérifier le comportement de TCP sur les liens 10 Gb/s de Grid'5000.

TCP a été conçu pour s'exécuter sur des réseaux dans lesquels le multiplexage de flux est très important et le ratio débit d'accès/ débit de coeur est faible (cf Internet). Il est connu que son comportement sur des réseaux haut débit tels que ceux de Grid'5000 peut être assez fluctuant. Or les performances de l'exécution d'applications sur une grille sont étroitement liées aux performances des communications sous jacentes. Il est donc nécessaire de se préoccuper du comportement de TCP pour obtenir de meilleures performances. Les résultats que nous donnons ici devraient permettre aux utilisateurs de comprendre le comportement (au niveau communications) de leurs applications s'exécutant sur Grid'5000 et de paramètrer TCP en conséquence.

La suite de ce rapport est organisée de la façon suivante. Dans un premier temps, nous décrirons le protocole expérimental que nous avons suivi, puis nous présenterons les résultats obtenus. Enfin, nous présenterons les problèmes rencontrés lors de la réalisation des tests et proposerons quelques optimisations.

[1] Site Renater : http://www.renater.fr/
[2] Gigabit/s Ethernet

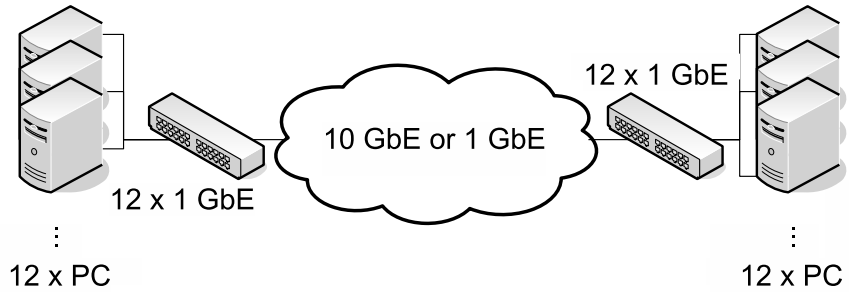



Figure 2: Architecture des expériences

# 2 Evaluation des performances de TCP sur le réseau de Grid'5000

Le principe de base est de créer des flux TCP, ayant comme source une machine d'un site vers une machine d'un autre site. La métrique que nous considérons est le débit utile. Nous utilisons l'outil iperf [3] pour générer les flux. Iperf permet non seulement de créer un flux TCP mais également de réaliser des mesures du débit utile au cours de l'expérience. Les relevés de mesure sont effectués toutes les 0.5 s.

Le schéma expérimental de la plateforme utilisée dans nos expérinces est celui de la Figure 2. Le nuage représente un chemin de Grid'5000 séparant les sites concernés. Selon le cas, il peut s'agir d'un chemin 10 GbE ou d'un chemin 1 GbE (ou d'une concaténation de liens 1 GbE et 10 GbE).

Nous avons mené les expériences suivantes entre deux sites :

- un flux par machine, une machine par site
- un flux par machine, plusieurs machines par site
- plusieurs flux par machine, plusieurs machines par site

## 2.1 Un flux par machine, une machine par site

Dans cette première expérience, nous avons utilisé les images par défaut de chaque site, sans effectuer de réglagles particulier de TCP[3]. Pour chaque paire de sites, nous avons créé un flux TCP entre une machine du premier site et une machine du second, ce sur une durée de 300 secondes. Les résultats de cette expérience nous a permis de créer la matrice des débits que l'on peut obtenir sans paramètrage. Cette matrice, représentée par la Table 1, sert de base de comparaison pour les expériences suivantes.

Nous pouvons remarquer que la plupart des sites n'atteignent pas plus de 150 Mb/s[4] sur des chemins dont le débit utile théorique[5] est de 941 Mb/s en TCP. Seuls certains flux (Nancy-Orsay, Lyon-Grenoble, Orsay-Lille) réussissent à obtenir un débit correct du fait de la proximité des sites et donc de la faible latence.

TCP utilise une fenêtre de congestion limitant le nombre de paquets "en vol", c'est à dire envoyés mais non acquittés afin d'éviter de congestionner les liens et donc d'avoir des pertes. La taille de cette fenêtre est liée à la durée d'aller-retour (*Round Trip Time*) d'un paquet et de son acquittement. Puisque l'émetteur n'a pas la garantie que le paquet ait eté délivré avant de recevoir

[3] BIC + Sack dans les images par défaut
[4] Megabits/s
[5] Les machines sont reliées par des liens 1 GbE aux switchs des sites.

| | | Source | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Bordeaux | Grenoble | Lille | Lyon | Nancy | Orsay | Rennes | Sophia | Toulouse |
| Destination | Bordeaux | | 58.1 | 61.8 | 55.9 | 81.2 | 111 | 76.3 | 68.9 | 181 |
| | Grenoble | 32.3 | | 34.0 | 151 | 39.8 | 33.7 | 34.3 | 52.6 | 48.4 |
| | Lille | 53.3 | 70.0 | | 53.6 | 112 | 199 | 55.0 | 44.3 | 33.9 |
| | Lyon | 61.5 | 230 | 71.2 | | 97.6 | 106 | 49.8 | 100 | 72.0 |
| | Nancy | 48.0 | 162 | 78.5 | 52.4 | | 777 | 54.7 | 43.3 | 32.0 |
| | Orsay | 67.8 | 54.1 | 150 | 58.8 | 936 | | 68.7 | 36.2 | 50.8 |
| | Rennes | 64.2 | 33.6 | 46.6 | 41.4 | 45.5 | 56.5 | | 27.4 | 26.3 |
| | Sophia | 47.0 | 46.1 | 29.5 | 67.4 | 28.9 | 22.3 | 25.1 | | 34.0 |
| | Toulouse | 166 | 47.6 | 29.8 | 65.7 | 29.7 | 44.3 | 26.3 | 36.1 | |

Table 1: Matrices des flux pour une paire de machines avec un unique flux TCP sans paramètrage (Mb/s).

| | | Source | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | bordeaux | grenoble | lille | lyon | nancy | orsay | rennes | sophia | toulouse |
| Destination | bordeaux | | 16.1 | 11.59 | 9.96 | 12.73 | 8.86 | 8.06 | 10.6 | 3.95 |
| | grenoble | 16.04 | | 12.83 | 3.3 | 13.24 | 15.15 | 15.22 | 9.87 | 11.36 |
| | lille | 11.61 | 12.84 | | 10.16 | 9.19 | 4.54 | 11.23 | 16.78 | 18.51 |
| | lyon | 9.99 | 3.28 | 10.17 | | 10.57 | 9.27 | 12.57 | 7.22 | 8.70 |
| | nancy | 12.72 | 13.26 | 9.19 | 10.57 | | 5.72 | 11.63 | 17.19 | 18.62 |
| | orsay | 8.85 | 15.16 | 4.53 | 9.26 | 5.72 | | 9.03 | 20.38 | 12.4 |
| | rennes | 8.04 | 15.22 | 11.23 | 12.56 | 11.63 | 9.03 | | 19.18 | 20.65 |
| | sophia | 10.60 | 9.96 | 16.78 | 7.22 | 17.19 | 20.38 | 19.17 | | 15.25 |
| | toulouse | 3.80 | 11.61 | 18.24 | 8.64 | 18.66 | 12.4 | 20.66 | 15.86 | |

Table 2: Matrices des RTT site à site (en ms)

l'acquittement, il se doit de conserver les données pour pouvoir éventuellement les retransmettre. De ce fait, les tampons associés à la socket TCP doivent permettre de stocker une fenêtre de congestion complète afin de ne pas limiter artificiellement la quantité de données "en vol" et donc le débit. La taille de ces tampons doit ainsi être égale au minimum au produit débit-délai de la connexion. Dans le cadre de Grid'5000, les délais (RTT) rencontrés varient de 3,3 ms à 20,7 ms, ainsi qu'on peut le voir sur la Table 2. Ces résultats ont été obtenus grâce à un ping de 20 échanges entre chaque paire de sites.

Cependant, le noyau linux adapte automatiquement la taille des tampons de sockets. La taille minimale est d'environ 2,625 Mo pour pouvoir exploiter les liens. Or la valeur maximum par défaut des tailles de tampons est de l'ordre de 170 ko sur les image par défaut, il n'est alors pas possible de dépasser des latences de 1,4 ms sur un lien 1 Gb/s pour obtenir un débit optimale sur TCP. Pour Orsay et Nancy où la valeur est de 4 Mo, la taille est adaptée, ce qui explique les valeurs de 777 et 936 Mb/s.

Pour pouvoir modifier les paramètres de TCP, il est nécessaire d'avoir les droits "root" sur la machine et donc de déployer son propre environnement. Deux variables doivent être modifiées pour augmenter la taille maximale du tampon TCP : */proc/sys/net/core/rmem_max* et *wmem_max* (cf [6]). Elles peuvent être modifiées dans le fichier */etc/sysctl.conf* qui doit ressembler à ceci :

```
net/core/rmem_max=4194304
net/core/wmem_max=4194304
```

Toute modification doit être suivie d'un *sysctl -p* pour être prise en compte.

L'application doit pouvoir utiliser cette taille maximale. Pour cela, nous pouvons nous appuyer sur l'auto-configuration présente dans linux qui adapte la taille des tampons lors de l'utilisation

| | | Source | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Bordeaux | Grenoble | Lille | Lyon | Nancy | Orsay | Rennes | Sophia | Toulouse |
| Destination | Bordeaux | | 771 | 725 | 862 | 911 | 884 | 852 | 875 | 685 |
| | Grenoble | 900 | | 701 | 925 | 812 | 893 | 787 | 911 | 647 |
| | Lille | 738 | 838 | | 120 | 922 | 848 | 916 | 598 | 579 |
| | Lyon | 425 | 912 | 786 | | 904 | 740 | 864 | 926 | 730 |
| | Nancy | 725 | 851 | 742 | 865 | | 854 | 938 | 931 | 622 |
| | Orsay | 799 | 866 | 777 | 869 | 936 | | 849 | 878 | 523 |
| | Rennes | 912 | 831 | 787 | 859 | 914 | 912 | | 839 | 651 |
| | Sophia | 901 | 839 | 653 | 543 | 611 | 900 | 321 | | 694 |
| | Toulouse | 928 | 859 | 784 | 882 | 933 | 923 | 939 | 909 | |

Table 3: Matrices des flux pour une paire de machines avec un unique flux TCP tuné (Mb/s).

selon les besoins ou préciser dans le code lors de la création des sockets quelle taille de tampons utiliser.

Les valeurs de l'auto-configuration se modifient selon le même protocole que précédemment. Il faut modifier le fichier *sysctl.conf* en ajoutant les lignes suivantes :

```
net/ipv4/tcp_rmem=4096 87380 4194304
net/ipv4/tcp_wmem=4096 87380 4194304
```

Ces valeurs représentent respectivement la valeur minimale, par défaut et maximale du tampon de sockets.

Les applications qui utilisent des sockets prennent par défaut la taille du buffer de socket. Si l'auto-configuration ci-dessus n'est pas correcte, il est nécessaire de passer des options lors de la création de la socket, pour qu'une application puisse utiliser la taille maximale du tampon. Ceci peut être réalisé par les lignes de code suivantes à ajouter juste après l'appel à la fonction `listen()` :

```
int snd_buf_size=4194304;
int rcv_buf_size=4194304;

setsockopt(socket, SOL_SOCKET, SO_SNDBUF, (const void *)&snd_buf_size,
(socklen_t) sizeof(snd_buf_size)));
setsockopt(socket, SOL_SOCKET, SO_RCVBUF, (const void *)&rcv_buf_size,
(socklen_t ) sizeof(rcv_buf_size)));
```

Après avoir réalisé le paramètrage de TCP avec une taille maximale de buffer à 4 Mo (suffisante pour tous les échanges dans la grille), nous avons effectué la même expérience que précédement. Les résultats sont présentés dans la Table 3. Les flux atteignent des débits de l'ordre de 800 Mb/s. On remarque que les débits maximaux ont rarement été atteints. Plusieurs raisons sont possibles : trafic concurrent, protocole, limites matérielles...

Seuls les flux provenant de Sophia, n'atteignent pas ce débit. Nous supposons que celà est dû à l'architecture réseau de Sophia qui utilise une pile de switchs reliés par un bus mais plus d'expérimentations sont nécessaires pourle confirmer. Les tests suivants montrent en effet que l'architecture du réseau de Sophia ne permet pas une équité entre tous les flux. (cf Figure 5 et section 3)

## 2.2 Plusieurs flux par site

Les tests précédents n'ont pas permis de tester les liens 10 GbE puisque que les noeuds n'ont que des cartes 1 GbE. Pour vérifier la capacité de ces liens, nous avons créé plusieurs flux parallèles afin de les remplir.

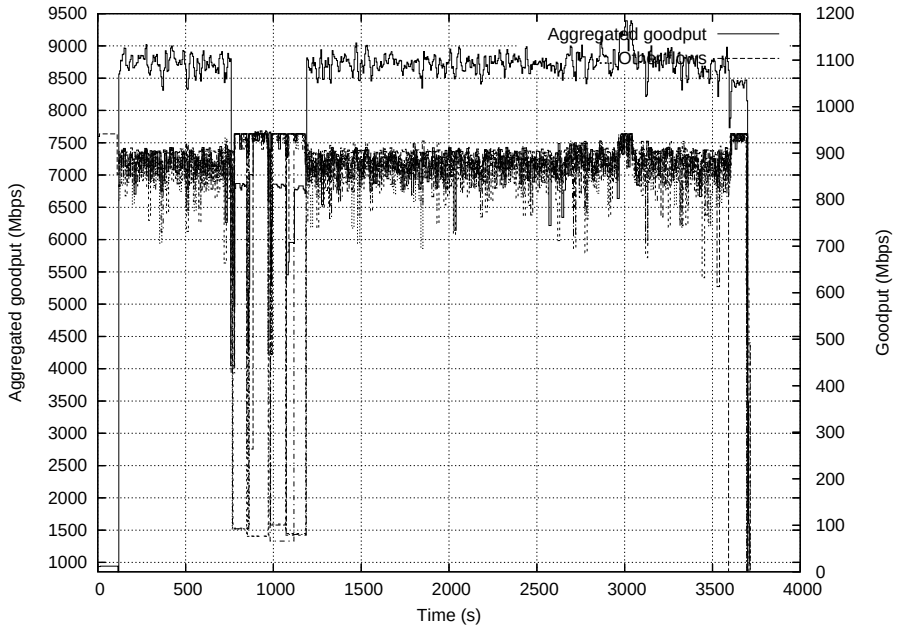



Figure 3: Bande passante aggrègée obtenue de Rennes vers Nancy sur 10 flux.

Cette expérience est semblable à la précédente mais nous avons utilisé 10 paires de noeuds au lieu d'une, ce qui nous permettait d'envoyer suffisament de trafic aggrégé. Les flux sont lancés à 1 seconde d'intervalle pour éviter la congestion lors du slow start et ainsi permettre aux flux d'obtenir le maximum de bande passante. Les machines utilisent un paramètrage adéquat de TCP.

La Figure 3 montre les résutats d'un test entre Rennes et Nancy avec 10 flux. La courbe supérieure dont l'échelle se situe à gauche montre la somme de tous les autres flux. Nous avons ainsi pu obtenir une bande passante aggrégée de 8,8 Gb/s avec un pic à 9,4 Gb/s (Débit maximal atteignable sur TCP). Les chutes observées sont sans doute dues à l'utilisation du réseau par d'autres flux concurrents.

La Figure 4 présente les résultats de l'expérience inverse (de Nancy vers Rennes). Ceux-ci sont moins probants et nous obtenons seulement un débit moyen de 2,8 Gb/s. Les pertes de performances sont dues dans cette expérience à une limitation en sortie de Nancy. Ce problème n'apparait plus sur les résultats ostérieurs, après que la limitation ait été enlevée.

Quant aux tests impliquant Sophia, ils sont très en deça de telles performances comme le montre la Figure 5. Ces tests on été menés le 23 Novembre entre 12h et 19h. Les résultats montrent clairement que l'équité entre les flux n'est pas respectée. Certains bénéficient d'une bande passante correcte (800 Mb/s) alors que d'autres plafonnent à 300 Mb/s voire à 150 Mb/s pour l'un d'entre eux. De ce fait, la bande passante aggrègée obtenue n'est pas aussi conséquente que ce qu'elle pourrait être.

## 2.3 Plusieurs flux TCP parallèles par machine

Cette expérience avait pour but de tester le comportement des machines et des liens lorsque l'on crée beaucoup de flux TCP. En effet, il est connu ([1] [8] [7]) que pour obtenir une meilleure équité de TCP entre des machines communiquant sur un lien, il est possible de créer plusieurs flux en parallèle. Des outils tels que GridFTP[5] et Psockets implémentent cette technique.

La Table 4 présente le résultat des expériences entre Rennes (client) et Nancy (serveur) pour 11 paires de noeuds. Chaque paire de machine emet un nombre de flux parmi 1, 2, 5 et 10. De la même manière que précédemment, les flux ont été décalés d'une seconde. Ces tests ont été menés le 15 Septembre entre 15h et 18h. Comme le montre l'indice de Jain[4] qui permet de mesurer l'équité, les différentes machines obtiennent un débit équitable.

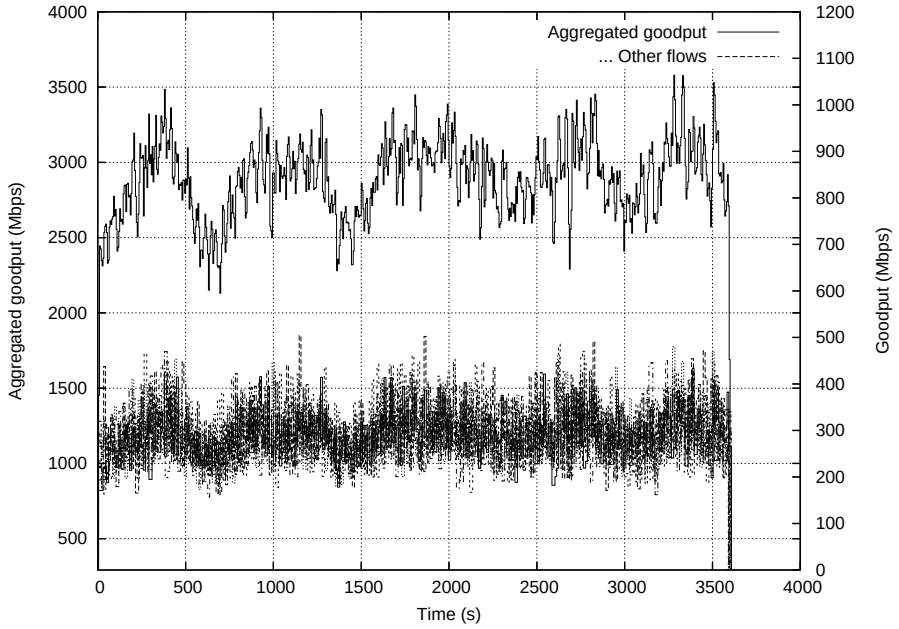



Figure 4: Bande passante aggrègée obtenue de Nancy vers Rennes sur 10 flux.

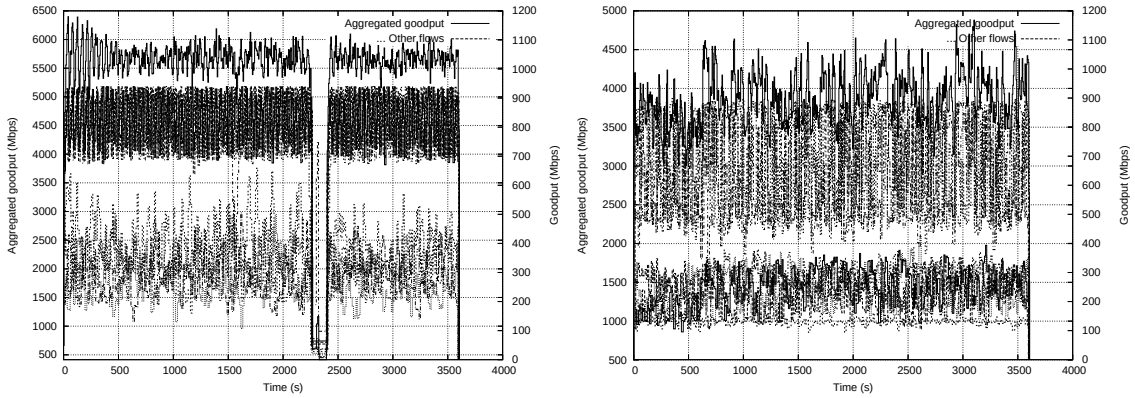



Figure 5: Bande passante aggrègée obtenue de Sophia vers Rennes (et inversement) sur 10 flux.

| Nombre de flux par noeud | 1 | 2 | 5 | 10 |
|---|---|---|---|---|
| Moyenne des bandes passante aggrègées (Mb/s) | 8353.66 | 8795.21 | 8994.21 | 9207.78 |
| Moyenne des débits des flux (Mb/s) | 761.70 | 399.83 | 163.53 | 83.71 |
| Indice de Jain par noeud | 0.9993 | 0.99943 | 0.999831 | 0.999823 |

Table 4: Flux parallèles sur chaque noeud dans Grid'5000 pour 11 paires de noeuds

Le premier constat est que l'augmentation du nombre de flux par machine permet d'améliorer la bande passante totale obtenue. Ainsi, on peut voir passer la bande passante aggrègée de 8.3 Gb/s pour 1 flux à 9.2 Gb/s pour 10 flux par machine.

La Figure 6 montre les courbes des tests ci-dessus. On peut y voir que la bande passante aggrègée est plus stable lorsqu'il y a plus de flux. Le pic de la figure (b) est dû à une erreur d'aggrégation due à la précision de l'outil iperf. La bande passante moyenne obtenue pour chaque flux est équitable mais la bande passante instantanée de chaque flux est très instable. Ceci est du au fait qu'avec 11 noeuds, il y a congestion, les flux subissent des pertes de paquets et leur fenêtre de congestion diminue de moitié (dans le cas du fast retransmit) ou jusqu'à 1 en cas de timeout.

# 3 Problèmes rencontrés sur Grid'5000

Comme nous avons pu le voir dans la section précédente, les performances de TCP sont assez décevantes dans Grid'5000 si le paramètrage de TCP n'est pas ajusté. Nous présentons ci-dessous d'une part les problèmes constatés sur la plateforme et d'autre part certaines des optimisations nécessaire pour permettre de tirer au mieux parti des performances du réseau.

Le fait de tester de façon systématique les interconnexions réseaux de Grid'5000 a permis de mettre à jour un certain nombre de problèmes qui prouvent l'intérêt de faire régulièrement ce genre de tests.

**Défaillance d'un switch sur Lyon** Lors des premiers tests locaux réalisés sur le site de Lyon pour préparer les mesures de bande passante décrites dans les sections précédentes, nous avons pu constater qu'un certain nombre de noeuds de ce site avaient un problème de performance. En effet, nous ne parvenions pas à dépasser la centaine de Mb/s, même dans le cas de tests avec des noeuds branchés sur le même switch, même en redéployant une nouvelle image.

Après investigations, il est apparu que le problème était dû à une défaillance de certains ports d'un des switchs du site. Le problème a été résolu par le constructeur.

**Creux de débit** Nous avons pu mettre en évidence un autre problème en faisant d'autres mesures de débit en local à Lyon (entre deux noeuds sur le même switch).

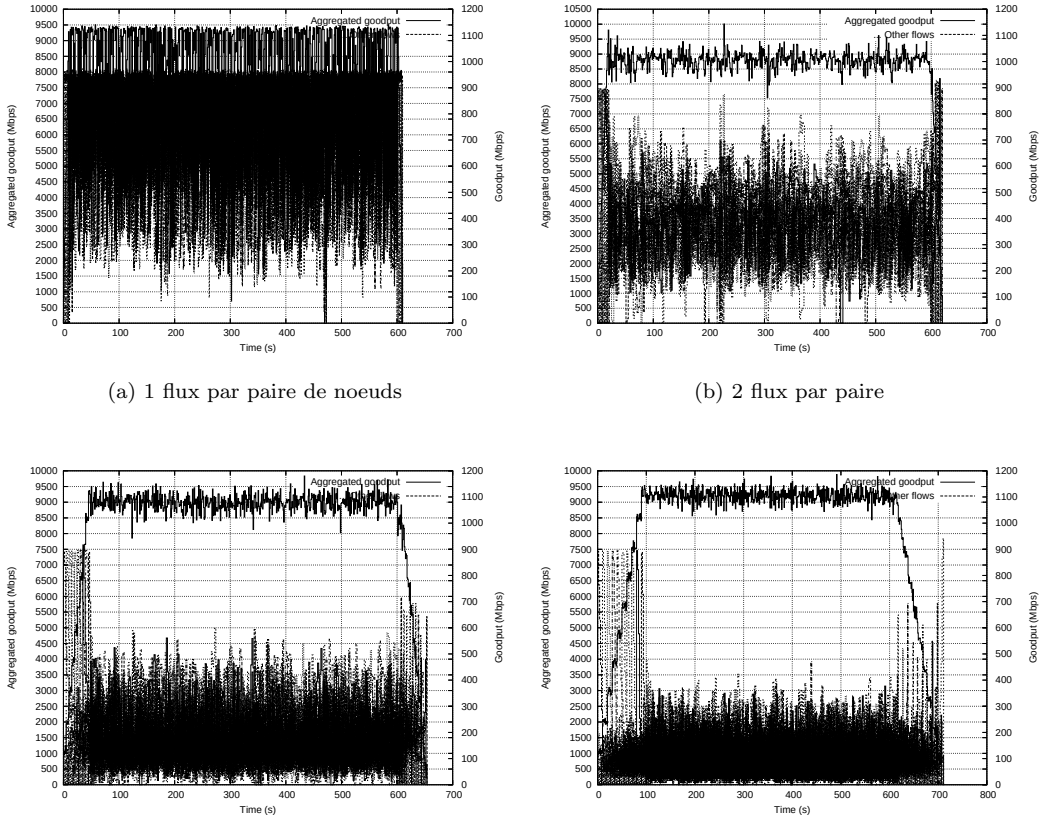
En utilisant `iperf` et en lui demandant d'afficher des moyennes intermédiaires toutes les 0.5 secondes (meilleure résolution possible), on aperçoit sur la Figure 7 des creux correspondant à des pertes d'environ 20% de débit. Ces creux interviennent de façon régulière, à intervalle d'environ 40s.

Au départ, on pourrait penser qu'il s'agit d'un artefact dû à la résolution des mesures faites avec `iperf`, mais le problème est corroboré par la mesure de la taille de la fenêtre de congestion qui présente elle aussi des creux importants avec ce même intervalle de 40s.

Comme on obtient des résultats similaires en changeant de méthodes de controle de congestion ou en utilisant des versions de noyau différente, que le problème est reproductible sur d'autres sites (Sophia et Rennes), c'est probablement un problème lié au hardware.

Le problème persiste sur tous les sites qui ont des IBM e-server 325 et il pourrait s'agir d'un problème d'interaction entre le firmware de la carte réseau et le BMC[6].

[6] Baseboard Management Controller




(a) 1 flux par paire de noeuds

(b) 2 flux par paire

(c) 5 flux par paire

(d) 10 flux par paire

Figure 6: Plusieurs flux parallèles sur 11 paires de noeuds entre Rennes et Nancy

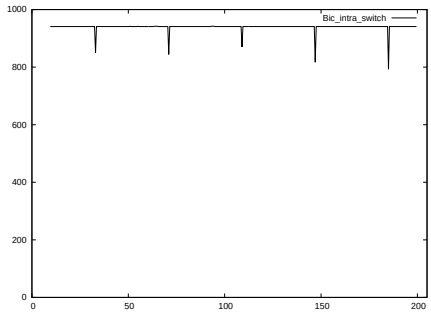



Figure 7: Creux de débit mesuré entre deux noeuds du site de Lyon

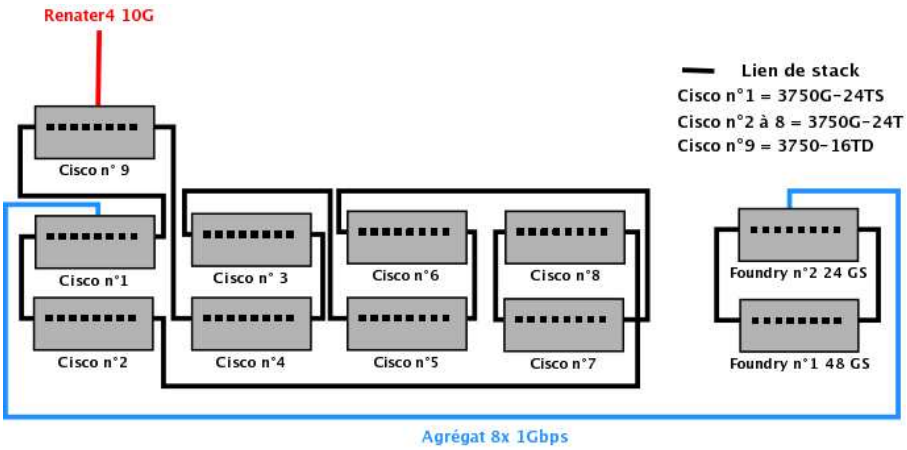



Figure 8: Topologie réseau du site de Sophia

**Le cas Sophia** Dans nos tests de stress de l'architecture Grid'5000, nous avons essayé de nous focaliser sur les backbones de 10 Gb/s de Grid'5000. L'approche la plus simple étant de se servir des sites proposant une sortie à 10 Gb/s[7] pour effectuer nos tests.

Malheureusement, nous avons rencontré un certain nombre de problèmes avec le site de Sophia qui viennent principalement de la topologie adoptée (Figure 8) avec des switchs[8] en anneau et de grosses incertitudes sur le trajet des paquets (ie. l'arbre recouvrant).

Pour exemple, les mesures de débit que nous avons effectuées dans la nuit du 27 Septembre 2006 entre Sophia et Nancy avec 12 noeuds choisis aléatoirement parmi ceux disponibles, que nous avons déployés et paramétrés. Nous sommes parvenus à obtenir au maximum 6.5 Gb/s[9] alors que la capacité nominale du lien est de 10 Gb/s. La Figure 9 montre le débit obtenu en sortie du switch de Sophia.

Des tests similaires entre Rennes et Nancy donnent des résultats de l'ordre de 9.8 Gb/s (de débit réel).

**Saturation des liens 10 Gb/s** La saturation des liens a déjà été traitée en partie en ce qui concerne le lien Nancy-Rennes. Pour le lien Paris-Lyon, il nous est actuellement impossible de le saturer dans un sens ou dans l'autre. En effet, comme nous l'avons déjà vu le site de Sophia pose problème puisqu'il ne peut fournir de flux d'un débit suffisant. De même, les flux à destination de Sophia n'atteignent pas un haut débit ce qui pose problème.

**Hétérogénéité** Il ne faut pas non plus aussi oublier qu'au delà de l'hétérogénéité hardware des sites, il existe aussi une forte hétérogénéité logicielle. Par exemple, au mois de novembre 2006, si on ne considère que les distributions/versions de noyau installées, seuls deux sites possèdent une version identique Lille et Bordeaux (cf Table 5). Les autres sites ont donc autant de versions de noyau différentes, et il y a donc 8 versions de pile TCP différentes possibles ayant chacune un comportement différent.

Les versions récentes du noyau Linux ont modifié la valeur par défaut du timer. Ceci peut avoir une influence selon les paramètrage du noyau. Le noyau Linux utilise la variable `netdev_max_backlog` pour connaître le nombre de paquets qui peuvent être traités simultanément par un processeur à chaque tic de timer. La valeur du timer influe sur le nombre de paquets qui pourront être traités.

Aussi pour qu'il n'y ait pas de limitation de la bande passante, il est nécessaire de faire attention aux deux valeurs suivantes. Par défaut, la valeur de `netdev_max_backlog` est de 300.

[7] Sophia, Rennes et Nancy, actuellement
[8] Cisco 3750
[9] et nous n'avons pas réussi à obtenir mieux au cours d'autres tentatives.

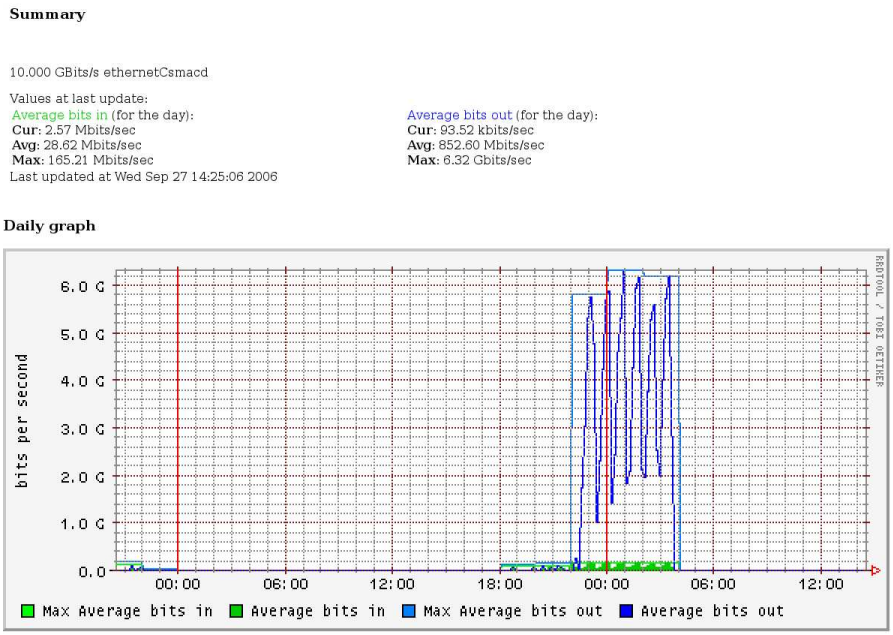



Figure 9: Mesure aggrégée entre Sophia et Nancy

| Site | Distribution | Noyau |
|---|---|---|
| Bordeaux | Fedora | 2.6.13 |
| Grenoble | Ubuntu | 2.6.15 |
| Lille | Fedora | 2.6.13 |
| Lyon | Debian | 2.6.15 |
| Nancy | Debian | 2.6.16 |
| Orsay | Debian | 2.6.8 |
| Rennes | Ubuntu | 2.6.12 |
| Sophia | RedHat | 2.4.21 |
| Toulouse | Fedora | 2.6.10 |

Table 5: Distributions et noyaux Linux utilisés sur les images par défaut de Grid'5000

Exemple, avec les valeurs par défaut :

```
netdev_max_backlog = 300
timer = 250
paquets_traités = 300 * 250 = 75000
taille moyenne d'un paquet = 1500 octets
bande passante maximale = 75000 * 1500 = 112.5 Mo/s = 900 Mb/s
```

Ceci n'est pas suffisant pour un débit à 1 Gb/s. Il faut donc augmenter la valeur de la variable dans le fichier `/etc/sysctl.conf` en fonction du débit théorique de la carte (1 GbE ou 10 GbE) :

```
net.core.netdev_max_backlog = 1000
```

# 4 Conclusion

Ce rapport présente les résultats des expériences menées depuis juillet dans le réseau Grid'5000. Celles-ci avaient pour but de tester le réseau et particulièrement le réseau 10 Gb/s. Les résultats montrent l'importance de faire attention aux couches basses pour toute exécution d'applications sur la grille. En effet, si le comportement de TCP n'est pas maitrisé, les applications ne s'exécutent pas de manière optimale et les résultats peuvent être aléatoires. Pour remédier à une partie des problèmes, il est nécessaire d'effectuer certaines optimisations sur TCP comme nous l'avons vu.

Par la suite, en continuant dans le même but, nous allons essayer de comprendre si le contrôle de congestion standard implémenté dans TCP sur la grille est utile ou s'il peut être amélioré en fonction de nos propres conditions. Nous allons également nous occuper à mettre en place un système de métrologie. Celui-ci devrait permettre à terme de fournir à tous les utilisateurs un support pour l'exécution ou le diagnostic de leurs applications. Le but est de rendre visible un certain nombre de métriques concernant le réseau qui font pour le moment défaut. Cette métrologie devra être disponible de bout en bout pour que tous les goulots d'étranglement puissent être maitrisés par l'utilisateur.

# References

[1] Eitan Altman, Dhiman Barman, Bruno Tuffin, and Milan Vojnovic. Parallel TCP Sockets: Simple Model, Throughput and Validation. In *Proceedings of the IEEE INFOCOM*, 2006.

[2] F. Cappello, F. Desprez, M. Dayde, E. Jeannot, Y. Jegou, S. Lanteri, N. Melab, R. Namyst, P. Primet, O. Richard, E. Caron, J. Leduc, and G. Mornet. Grid'5000: A large scale, reconfigurable, controlable and monitorable grid platform. In *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, Seattle, Washington, USA, November 2005. https://www.grid5000.fr/.

[3] Mark Gates, Ajay Tirumala, Jon Dugan, and Kevin Gibbs. Iperf user documentation. Iperf Web Site. http://dast.nlanr.net/Projects/Iperf/iperfdocs_1.7.0.php.

[4] Raj Jain. *The art of computer systems performance analysis*, chapter 3. John Wiley and Sons, Inc, 1991.

[5] Sang B. Lim, Geoffrey Fox, Ali Kaplan, Shrideep Pallickara, and Marlon Pierce. GridFTP and parallel TCP support in naradabrokering. *Lecture notes in Computer Science*, (0302-9743):93–102, 2005.

[6] Matt Mathis and Raghu Reddy. Enabling high performance data transfers on hosts. Technical report, Pittsburgh Supercomputer Center, 1999. http://www.psc.edu/networking/projects/tcptune/.

[7] H. Sivakumar, S. Bailey, and R. L. Grossman. PSockets: the case for application-level network striping for data intensive applications using high speed wide area networks. In *Supercomputing '00: Proceedings of the 2000 ACM/IEEE conference on Supercomputing*, page 37, Washington, DC, USA, 2000. IEEE Computer Society.

[8] Zongsheng Zhang, Go Hasegawa, and Masayuki Murata. Is parallel TCP really effective for fast long-distance networks ?