Martin QUINSON
École Normale Supérieure de Lyon,
Laboratoire de l'Informatique du Parallélisme

Connaître la grille

pour mieux l'utiliser

Séminaire du 27 mars 2003

Plan

- Introduction
- Quelle connaissance pour quel ordonnancement
- NWS: Network Weather Service
- FAST: Fast's Agent System Timer
- ALNeM: Application-Level Network Mapper
- Conclusions

Introduction : la grille pourquoi et comment

- Accroissement de la demande
  - en terme de puissance de calcul :
    - simulation et modélisation (climatologie et météorologie, physique, . . .)
    - traitement de signaux ou d'images
  - en terme de capacité de stockage :
    - génomique, fouille de données, serveurs multimédia, caches web

Introduction : la grille pourquoi et comment

- Accroissement de la demande

  en terme de puissance de calcul et de capacité de stockage

- Du parallélisme à tous les niveaux :
  - unités de calcul : proc. superscalaires, unités spécialisées, pipeline
  - machines : multi-processeurs
  - système d'exploitation : multi-programmation temps partagé
  - threads : multi-programmation à grain fin temps partagé
  - plateformes : architectures distribuées

Introduction : la grille pourquoi et comment

- Accroissement de la demande

  en terme de puissance de calcul et de capacité de stockage

- Du parallélisme à tous les niveaux :

  matériel et logiciel

- Hétérogénéité croissante :
  - Plates-formes à architecture hiérarchique.
  - Hétérogénéité matérielle, logicielle, administrative, ...

Introduction : la grille pourquoi et comment

- Accroissement de la demande

  en terme de puissance de calcul et de capacité de stockage

- Du parallélisme à tous les niveaux :

  matériel et logiciel

- Hétérogénéité croissante :
  - Plates-formes à architecture hiérarchique.
  - Hétérogénéité matérielle, logicielle, administrative, ...

$\Rightarrow$ La grile est la plate-forme résultant de la mise en commun de ressources locales par des des unités administratives distictes.

Cela donne souvent une constellation WAN de réseaux LAN.

La grille aujourd'hui et demain

But : louer la puissance de calcul et la capacité mémoire à travers l'Internet

☺ Très grand potentiel :

- Besoin en de puissance de calcul et en de capacité mémoire toujours croissant
- Certains codes (ou données) sensibles doivent rester sur place
- Utilisation de serveurs de calculs accessibles à travers une interface simple

$\Rightarrow$ PSE (Problem Solving Environment), ASP (Application Service Provider)

☹ Toujours difficiles à utiliser pour les non-spécialistes :

- Pratiquement pas de transparence
- Les problèmes de sécurité et d'accounting ne sont généralement pas traités
- Souvent des PSE dépendants d'une application
- Pas de standard unique (CORBA, JAVA/JINI, sockets, . . .)

RPC et grid-computing : GridRPC

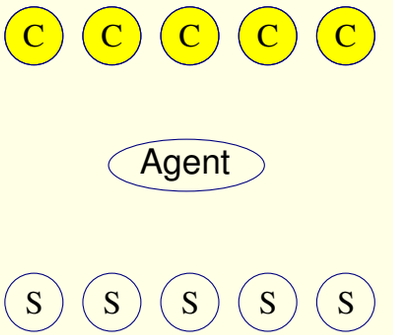
Une idée simple : Implémenter le modèle de programmation RPC sur la grille

- Fonctionnalités nécessaires :
  - Ordonnancement (localisation de ressources, éval. de perf.)
  - Distribution et migration de données
  - Sécurité, Tolérance aux pannes
  - Interopérabilité avec d'autres systèmes, . . .

RPC et grid-computing : GridRPC

Une idée simple : Implémenter le modèle de programmation RPC sur la grille

– Fonctionnalités nécessaires :
  – Ordonnancement (localisation de ressources, éval. de perf.)
  – Distribution et migration de données
  – Sécurité, Tolérance aux pannes
  – Interopérabilité avec d'autres systèmes, . . .



– Cinq composants fondamentaux :

Client Fournit plusieurs interfaces utilisateur et soumet les requêtes aux serveurs

Serveur

Agent

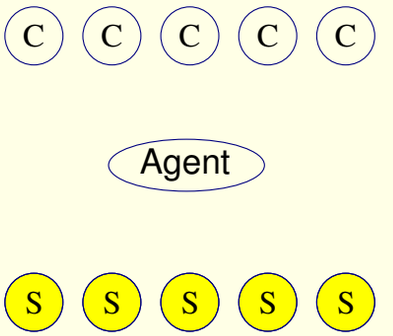
Moniteur

Base de données

RPC et grid-computing : GridRPC

Une idée simple : Implémenter le modèle de programmation RPC sur la grille

– Fonctionnalités nécessaires :
  – Ordonnancement (localisation de ressources, éval. de perf.)
  – Distribution et migration de données
  – Sécurité, Tolérance aux pannes
  – Interopérabilité avec d'autres systèmes, . . .



– Cinq composants fondamentaux :

Client Fournit plusieurs interfaces utilisateur et soumet les requêtes aux serveurs

Serveur Reçoit les requêtes des clients et exécute les modules logiciels pour eux

Agent

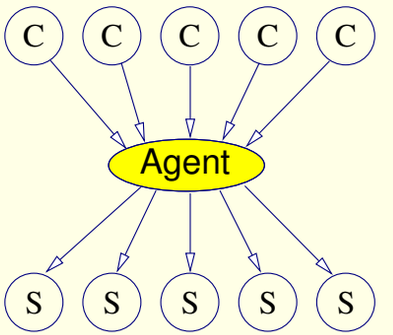
Moniteur

Base de données

RPC et grid-computing : GridRPC

Une idée simple : Implémenter le modèle de programmation RPC sur la grille

– Fonctionnalités nécessaires :
  – Ordonnancement (localisation de ressources, éval. de perf.)
  – Distribution et migration de données
  – Sécurité, Tolérance aux pannes
  – Interopérabilité avec d'autres systèmes, . . .



– Cinq composants fondamentaux :

Client Fournit plusieurs interfaces utilisateur et soumet les requêtes aux serveurs

Serveur Reçoit les requêtes des clients et exécute les modules logiciels pour eux

Agent Intercepte les requêtes des clients et les ordonnance sur les serveurs

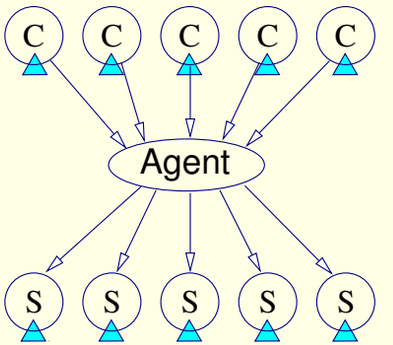
Moniteur

Base de données

RPC et grid-computing : GridRPC

Une idée simple : Implémenter le modèle de programmation RPC sur la grille

– Fonctionnalités nécessaires :
  – Ordonnancement (localisation de ressources, éval. de perf.)
  – Distribution et migration de données
  – Sécurité, Tolérance aux pannes
  – Interopérabilité avec d'autres systèmes, . . .



– Cinq composants fondamentaux :

Client Fournit plusieurs interfaces utilisateur et soumet les requêtes aux serveurs

Serveur Reçoit les requêtes des clients et exécute les modules logiciels pour eux

Agent Intercepte les requêtes des clients et les ordonnance sur les serveurs

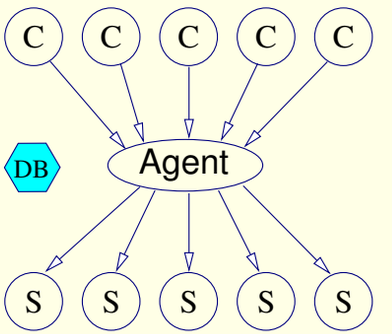
Moniteur Observe dynamiquement l'état des ressources de calcul

Base de données

RPC et grid-computing : GridRPC

Une idée simple : Implémenter le modèle de programmation RPC sur la grille

– Fonctionnalités nécessaires :
  – Ordonnancement (localisation de ressources, éval. de perf.)
  – Distribution et migration de données
  – Sécurité, Tolérance aux pannes
  – Interopérabilité avec d'autres systèmes, . . .



– Cinq composants fondamentaux :

Client Fournit plusieurs interfaces utilisateur et soumet les requêtes aux serveurs

Serveur Reçoit les requêtes des clients et exécute les modules logiciels pour eux

Agent Intercepte les requêtes des clients et les ordonnance sur les serveurs

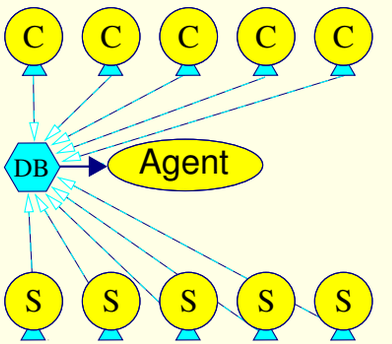
Moniteur Observe dynamiquement l'état des ressources de calcul

Base de données Connaissances statiques et dynamiques sur les ressources

RPC et grid-computing : GridRPC

Une idée simple : Implémenter le modèle de programmation RPC sur la grille

– Fonctionnalités nécessaires :
  – Ordonnancement (localisation de ressources, éval. de perf.)
  – Distribution et migration de données
  – Sécurité, Tolérance aux pannes
  – Interopérabilité avec d'autres systèmes, . . .



– Cinq composants fondamentaux :

Client Fournit plusieurs interfaces utilisateur et soumet les requêtes aux serveurs

Serveur Reçoit les requêtes des clients et exécute les modules logiciels pour eux

Agent Intercepte les requêtes des clients et les ordonnance sur les serveurs

Moniteur Observe dynamiquement l'état des ressources de calcul

Base de données Connaissances statiques et dynamiques sur les ressources

La connaissance de la plate-forme est vitale pour l'agent

Plan

- Introduction
- Quelle connaissance pour quel ordonnancement
- NWS: Network Weather Service
- FAST: Fast's Agent System Timer
- ALNeM: Application-Level Network Mapper
- Conclusions

Quelle connaissance pour quel ordonnancement

Ordonnancement aléatoire :

– Liste des tâches ; Liste des machines possibles

Ordonnancement simple :

– À propos des tâches : complexité algorithmique (comme $O(n)$)
– À propos des machines : puissance de crête ou sur un benchmark donné
– À propos des liens : matrice des capacités maximales

Ordonnancement Grid actuel :

– À propos des machines : existence, charge processeur et mémoire
– À propos des liens : matrice des capacités actuelles, (topologie)
– À propos des tâches : besoins (temps de calcul, mémoire, communication)

Quelle connaissance pour quel ordonnancement

Ordonnancement aléatoire :

– Liste des tâches ; Liste des machines possibles

Ordonnancement simple :

– À propos des tâches : complexité algorithmique (comme $O(n)$)
– À propos des machines : puissance de crête ou sur un benchmark donné
– À propos des liens : matrice des capacités maximales

Ordonnancement Grid actuel :

– À propos des machines : existence, charge processeur et mémoire
– À propos des liens : matrice des capacités actuelles, (topologie)
– À propos des tâches : besoins (temps de calcul, mémoire, communication)

Ajouts possibles :

– Topologie d'interconnexion
– Multiples implémentations, Tâches malléables
– Parallélisme de données
– Ordonnancement à fenêtre, Data mining dans l'historique

Quelle connaissance pour quel ordonnancement

Ordonnancement aléatoire :

– Liste des tâches ; Liste des machines possibles

Ordonnancement simple :

– À propos des tâches : complexité algorithmique (comme $O(n)$)
– À propos des machines : puissance de crête ou sur un benchmark donné
– À propos des liens : matrice des capacités maximales

Ordonnancement Grid actuel :

– À propos des machines : existence, charge processeur et mémoire
– À propos des liens : matrice des capacités actuelles, (topologie)
– À propos des tâches : besoins (temps de calcul, mémoire, communication)

Trois outils pour trois besoins :

– **NWS** : Disponibilités dynamiques de la plate-forme
– **FAST** : Besoins des routines $\oplus$ Adéquation routine / machine
– **ALNeM** : Découverte de la topologie

Plan

- Introduction
- Quelle connaissance pour quel ordonnancement
- NWS: Network Weather Service
- FAST: Fast's Agent System Timer
- ALNeM: Application-Level Network Mapper
- Conclusions

The Network Weather Service : Présentation

But : connaître et prévoir les disponibilités du système

Motivation : ordonnancement

Type d'informations :

- « Quelle est la bande passante entre **pc132** et **pc124** ? »
- « Combien y a-t-il de mémoire disponible sur **pc43** ? »
- « Quelle est la charge processeur de **pc22** ? »
- « Et dans 5 minutes ? »

Auteurs : Prof. Rich Wolski (UCSB) et Al.

Utilisateurs : AppLeS, Globus, NetSolve, Ninf, DIET, . . .

NWS : Fonctionnement

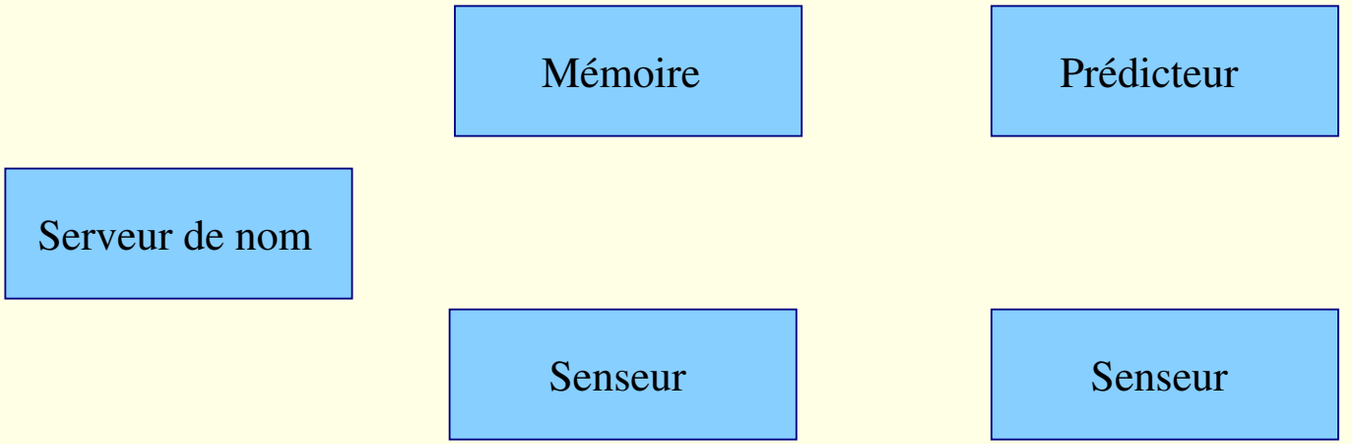
Ensemble d'applications distribué :

Senseur : Réalise les mesures

Mémoire : Stocke les résultats

Prédicateur : Prédit les variations par traitement statistique

Serveur de nom : Serveur de nom : annuaire du système, à la LDAP



NWS : Fonctionnement

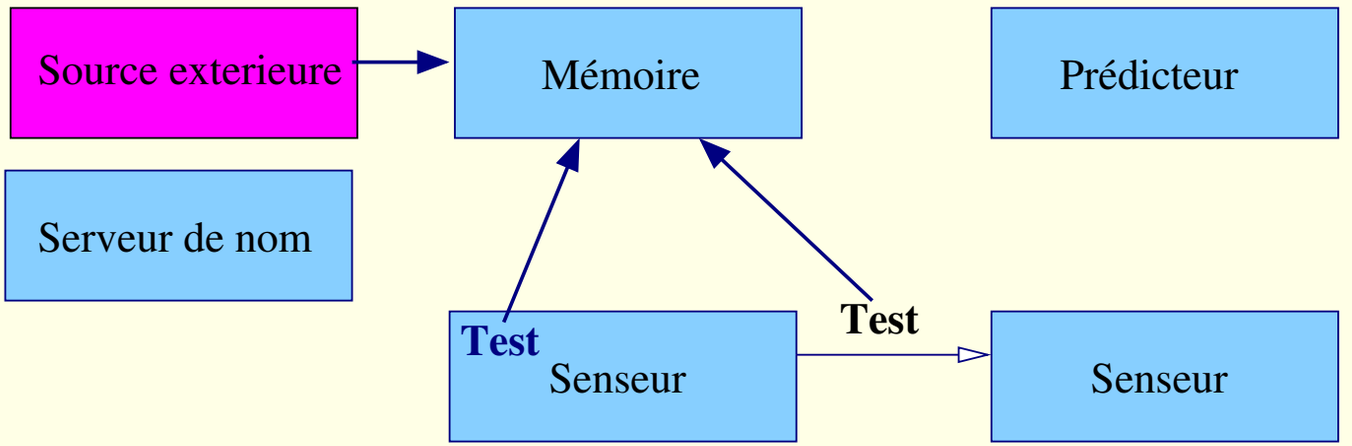
Ensemble d'applications distribué :

Senseur : Réalise les mesures

Mémoire : Stocke les résultats

Prédicateur : Prédit les variations par traitement statistique

Serveur de nom : Serveur de nom : annuaire du système, à la LDAP



Régime permanent : tests réguliers

NWS : Fonctionnement

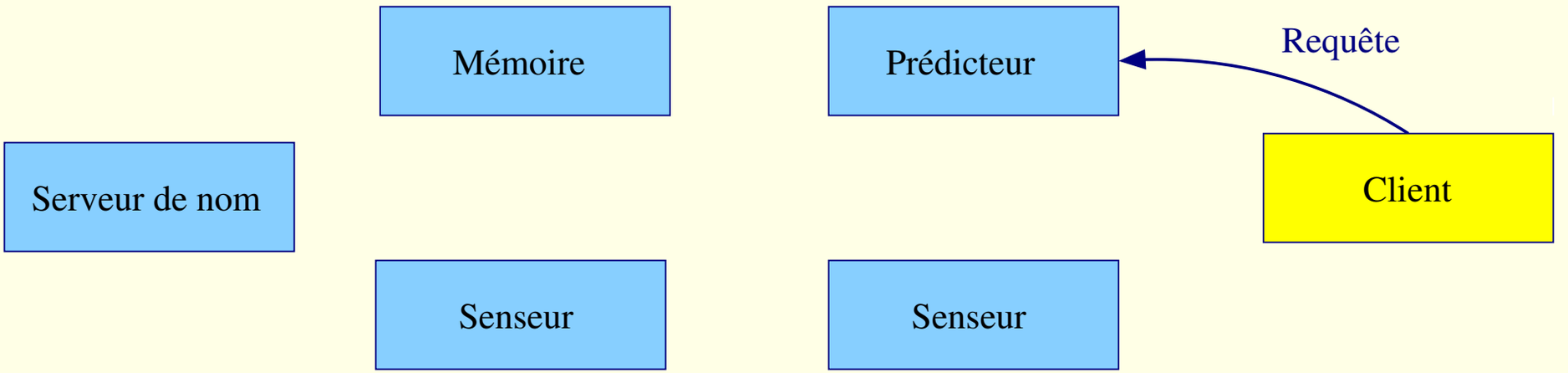
Ensemble d'applications distribué :

Senseur : Réalise les mesures

Mémoire : Stocke les résultats

Prédicateur : Prédit les variations par traitement statistique

Serveur de nom : Serveur de nom : annuaire du système, à la LDAP



Traitement d'une requête

NWS : Fonctionnement

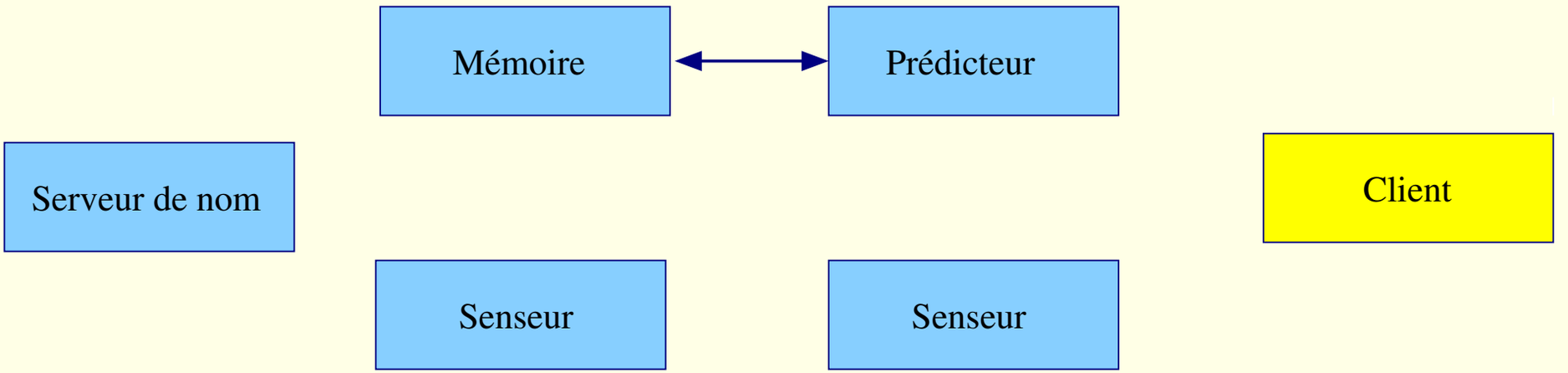
Ensemble d'applications distribué :

Senseur : Réalise les mesures

Mémoire : Stocke les résultats

Prédicateur : Prédit les variations par traitement statistique

Serveur de nom : Serveur de nom : annuaire du système, à la LDAP



Traitement d'une requête

NWS : Fonctionnement

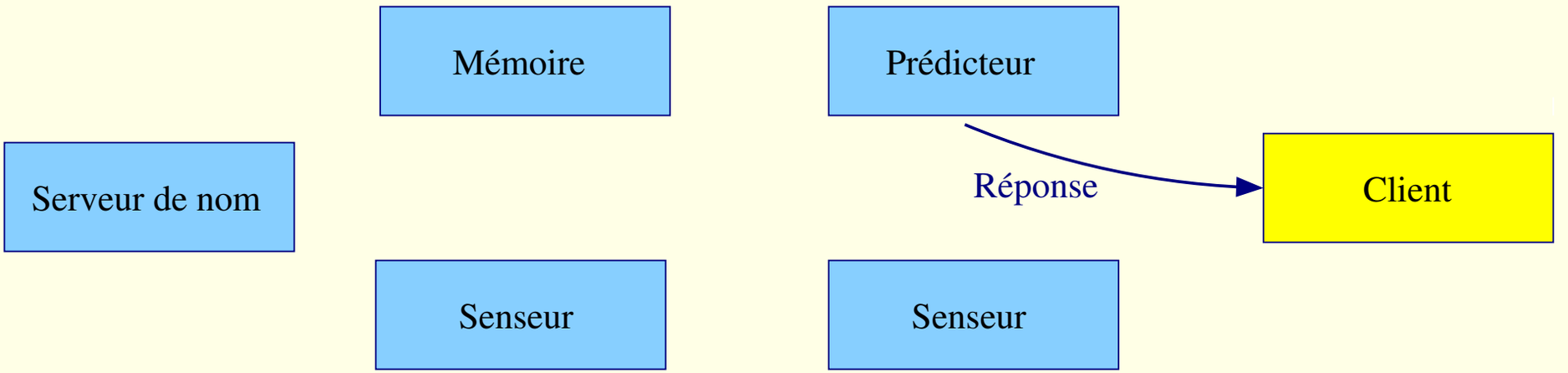
Ensemble d'applications distribué :

Senseur : Réalise les mesures

Mémoire : Stocke les résultats

Prédicateur : Prédit les variations par traitement statistique

Serveur de nom : Serveur de nom : annuaire du système, à la LDAP



Traitement d'une requête

Les mesures NWS

Les metriques offertes :

availableCpu (pour un nouvel arrivant), currentCpu (pour chaque présent), bandwidthTcp, latencyTcp (Par défaut : 64Ko en messages de 16Ko, buffer=32Ko), connectTimeTcp, freeDisk, freeMemory, (extensibilité possible mais difficile).

Les mesures NWS

Les metriques offertes :

availableCpu (pour un nouvel arrivant), currentCpu (pour chaque présent), bandwidthTcp, latencyTcp (Par défaut : 64Ko en messages de 16Ko, buffer=32Ko), connectTimeTcp, freeDisk, freeMemory, (extensibilité possible mais difficile).

Quelques problèmes rencontrés :

- CPU : Disponibilité pour un nouvel arrivant, nice, multiprocesseur.
- Mémoire : Ressources partagées, tampons, threads, swap.
- Réseaux : Intrusivité, très gros réseaux.

Les mesures NWS

Les metriques offertes :

availableCpu (pour un nouvel arrivant), currentCpu (pour chaque présent), bandwidthTcp, latencyTcp (Par défaut : 64Ko en messages de 16Ko, buffer=32Ko), connectTimeTcp, freeDisk, freeMemory, (extensibilité possible mais difficile).

Quelques problèmes rencontrés :

- CPU : Disponibilité pour un nouvel arrivant, nice, multiprocesseur.
- Mémoire : Ressources partagées, tampons, threads, swap.
- Réseaux : Intrusivité, très gros réseaux.

Solutions possibles :

- Balance Intrusivité / qualité de la mesure $\Rightarrow$ 1 test actif pour n passifs
  - CPU : lance une tâche de durée connue.
  - Mémoire : alloue en chronométrant jusqu'à swapper.
  - Réseau : à faire, en variant la taille des paquets.
- Collision entre tests réseaux $\Rightarrow$ passage de jetons

Les mesures NWS

Les metriques offertes :

availableCpu (pour un nouvel arrivant), currentCpu (pour chaque présent), bandwidthTcp, latencyTcp (Par défaut : 64Ko en messages de 16Ko, buffer=32Ko), connectTimeTcp, freeDisk, freeMemory, (extensibilité possible mais difficile).

Quelques problèmes rencontrés :

- CPU : Disponibilité pour un nouvel arrivant, nice, multiprocesseur.
- Mémoire : Ressources partagées, tampons, threads, swap.
- Réseaux : Intrusivité, très gros réseaux.

Intrusivité :

- CPU : moins de 3%
- Mémoire : 4x800K de code, autant de ressources
- Disque : 10 à 20 Mo pour le code et les bases de données
- Réseau : par défaut, un test toutes les 2mn sur la clique, de 64Ko (+NFS)

Le prédicteur NWS

- Le prédicteur utilise diverses méthodes statistiques simples
  moyenne : courante, à fenêtre fixe ou à fenêtre adaptative.
  médiane (élément du milieu de la liste triée) : mêmes variantes.
  gradian stochastique : $\mathrm{GRAD}(t, g) = (1-g) \times \mathrm{GRAD}(t-1, g) + g \times value(t)$
  dernière valeur
- Les méthodes sont en concurrence
  Données = série : D1, D2, . . ., Dn-1, Dn. On veut Dn+1.
  Application des méthodes sur D1, D2, . . ., Dn-1. Chacune prédit Dn.
  Sélection de la meilleure méthode sur Dn pour prédire Dn+1.
- Exécution en continue / à la demande
- Extensibilité simple, mais recompilation

NWS : Projets analogues et conclusion

Projets analogues

NetPerf : projet HP pour classer les composants, pas interactif

GloPerf : Globus passe à NWS

PingER : Pings à intervalles réguliers entre 600 hôtes dans 72 pays.

Iperf : Cherche la bande passante par saturation du lien

Performance Co-Pilot (SGI) :

- Même type d'architecture
- Données bas niveau (/proc) $\Rightarrow$ pas pratique pour l'ordonnanceur
- Pas de prédiction

Conclusion sur NWS

☺ Environnement complet

☺ Prévu pour l'ordonnancement

☺ Prédictions statistiques

☺ Largement utilisé

☹ Pas vraiment extensible

☹ Parfois difficile à mettre en place

☹ Seulement TCP

Plan

- Introduction
- Quelle connaissance pour quel ordonnancement
- NWS: Network Weather Service
- FAST: Fast's Agent System Timer
- ALNeM: Application-Level Network Mapper
- Conclusions

Fast Agent's System Timer : Présentation

But : Offrir à l'ordonnanceur les données dont il a besoin.

Ie, les perf. des routines sur une machine donnée à un instant donné.

Leitmotiv : interactivité, simplicité d'usage

Type d'informations :

« Combien de temps dure `dgemm(n,n,512,512,512)` sur `pc32` ? »

« Et combien de mémoire ? »

« Et en tenant compte de la charge actuelle ? »

\+ Les mêmes informations qu'NWS (wrapper)

Auteur : Martin Quinson.

Utilisateurs : DIET, un patch existe pour NetSolve.

FAST : Modélisation des besoins des routines

Différentes méthodes selon la routine :

– Routines simples (BLAS)
  Étalonnage à l'installation

– Routines complexes (ScaLAPACK)
  Décomposition par étude de la structure du source

– Routines compliquées (Matrices creuses)
  Pas de prédiction $\Rightarrow$ machine la plus puissante
  Décomposition pour extraire des parties simples

FAST : Modélisation des besoins des routines

Différentes méthodes selon la routine :

– Routines simples (BLAS)
  Étalonnage à l'installation **Ok**

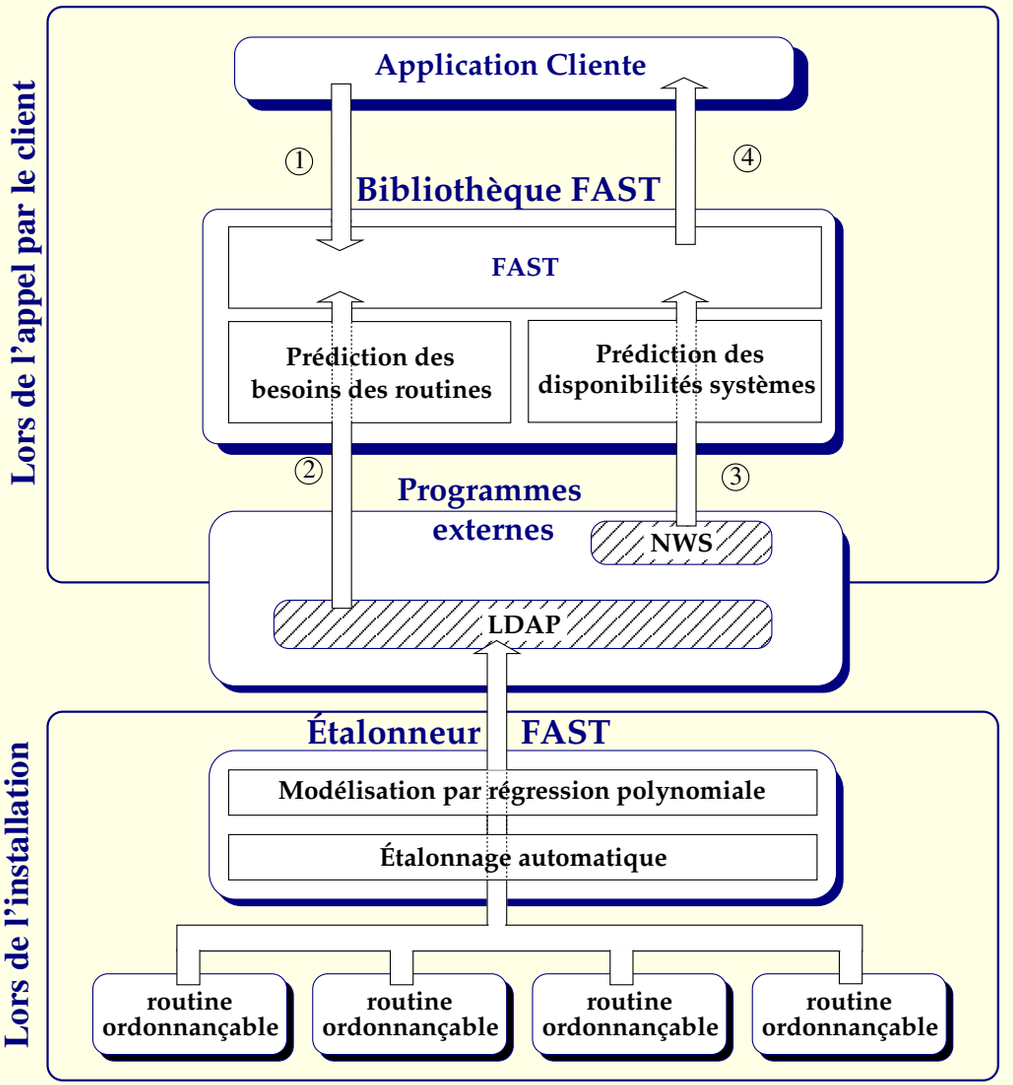
– Routines complexes (ScaLAPACK)
  Décomposition par étude de la structure du source **Freddy, en cours**

– Routines compliquées (Matrices creuses)
  Pas de prédiction $\Rightarrow$ machine la plus puissante
  Décomposition pour extraire des parties simples **À faire (pour Mumps)**

Freddy : Frédéric Suter et Eddy Caron

Mumps : Jean-Yves L'Excellent

FAST : Vue d'ensemble



FAST : Les problèmes de l'étalonnage

- Obtention des chronométrages
  - Charge extérieure
  - $\Rightarrow$ Temps processeur (utime + stime)
- Obtention de l'espace mémoire
  - Variation au cours de la vie du processus
  - $\Rightarrow$ Exécution en mode pas à pas (comme gdb)

FAST : Les problèmes de l'étalonnage

- Obtention des chronométrages
  - Charge extérieure
  - $\Rightarrow$ Temps processeur (utime + stime)
- Obtention de l'espace mémoire
  - Variation au cours de la vie du processus
  - $\Rightarrow$ Exécution en mode pas à pas (comme gdb)

Le processus est long, mais

- Uniquement à l'installation
- Réutilisation des résultats pour un parc de machines
- Étalonnage pour des machines types puis facteur correctif **En cours (?)**

FAST : L'API offerte

- FAST masque la complexité à l'ordonnanceur

  $\Rightarrow$ seulement 3 fonctions de haut niveau

  `comm_time(source, destination, data_desc)`

  `comp_time(hôte, problème, data_desc)`

  `get_time(hôte, problème, data_desc, localisation)`

FAST : L'API offerte

- FAST masque la complexité à l'ordonnanceur

  $\Rightarrow$ seulement 3 fonctions de haut niveau

  `comm_time(source, destination, data_desc)`

  `comp_time(hôte, problème, data_desc)`

  `get_time(hôte, problème, data_desc, localisation)`

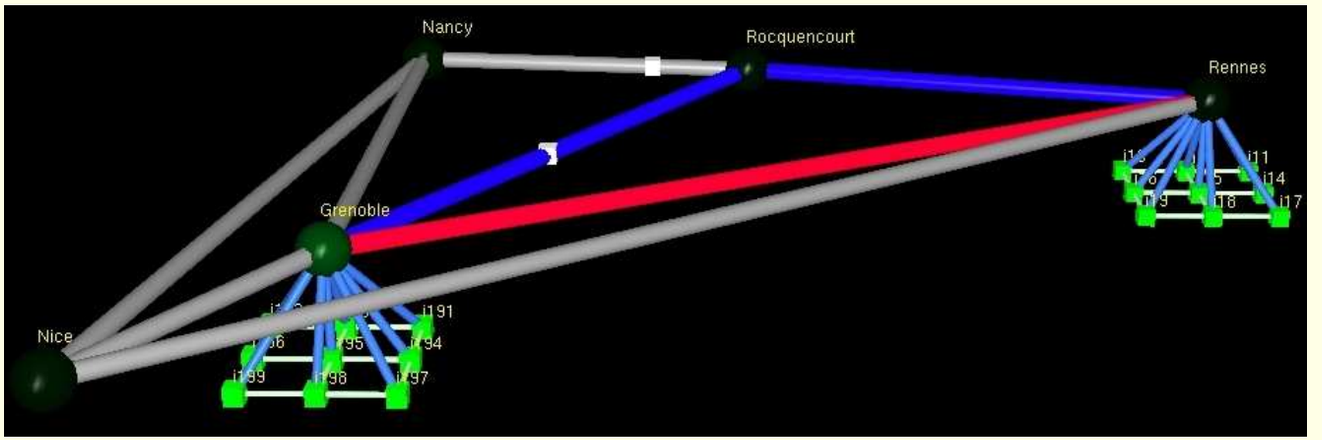
- Pour un outil de visualisation ou autre, API de bas niveau :

  `need(ressource, hôte, fonction, data_desc)`

  (ressource = temps, mémoire ou communication)

  `avail(ressource, hôte, hôte)`

  (ressource = CPU, mémoire, espace disque, bande passante, latence)



FAST : Qualité de la modélisation

Modélisation temporelle

	dgeadd		dgemm		dtrsm	
	iclust.	para.	iclust.	para.	iclust.	para.
Max.	0.02s	0.02s	0.21s	5.8s	0.13s	0.31s
error	6%	35%	0.3%	4%	10%	16%
Avg.	0.006s	0.007s	0.025s	0.03s	0.02s	0.08s
error	4%	6.5%	0.1%	0.1%	5%	7%

FAST : Qualité de la modélisation

Modélisation temporelle

	dgeadd		dgemm		dtrsm	
	iclust.	para.	iclust.	para.	iclust.	para.
Max.	0.02s	0.02s	0.21s	5.8s	0.13s	0.31s
error	6%	35%	0.3%	4%	10%	16%
Avg.	0.006s	0.007s	0.025s	0.03s	0.02s	0.08s
error	4%	6.5%	0.1%	0.1%	5%	7%

Modélisation spatiale

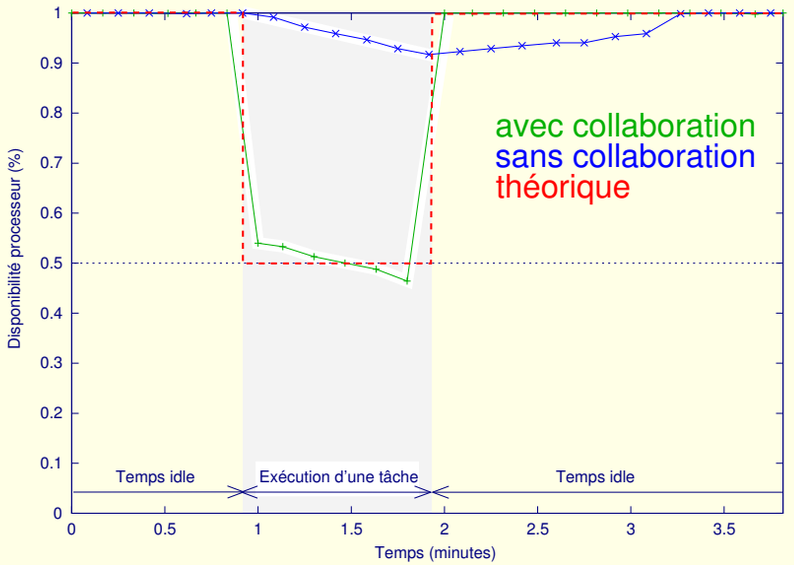
Presque parfait : Erreur maximale < 1% ; Erreur moyenne $\approx$ 0,1%

Taille du code + Taille des matrices

(constante) (polynomiale)

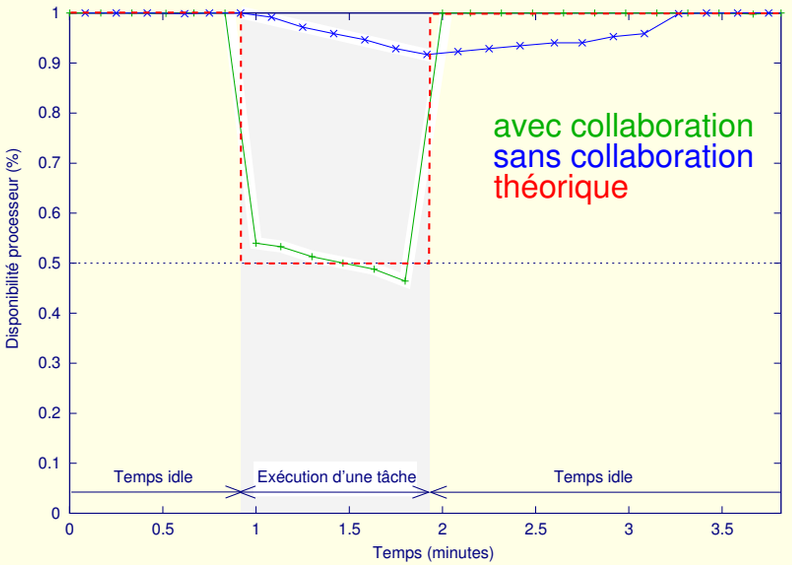
FAST : améliorations à NWS

Collaboration NWS / Sched.

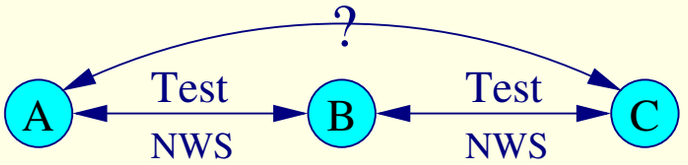


FAST : améliorations à NWS

Collaboration NWS / Sched.

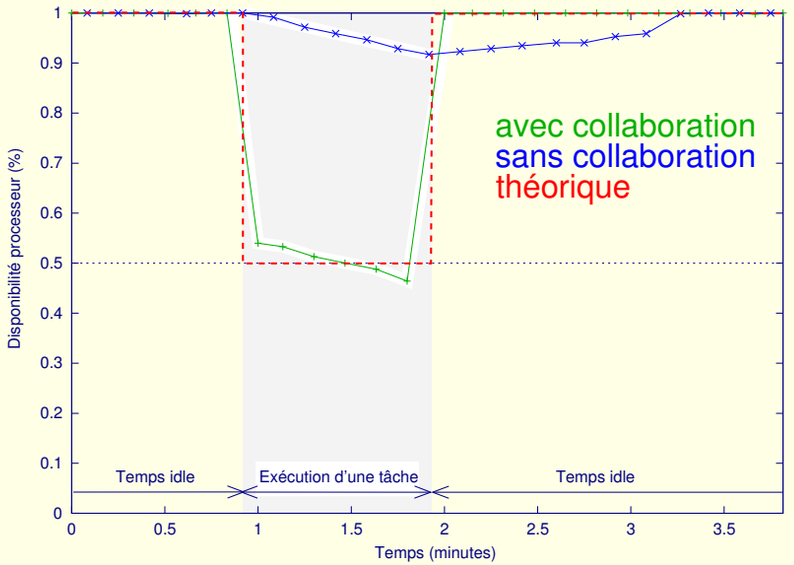


Sensor in the middle

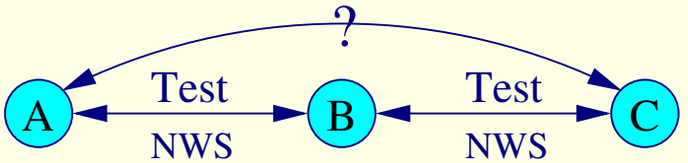


FAST : améliorations à NWS

Collaboration NWS / Sched.



Sensor in the middle

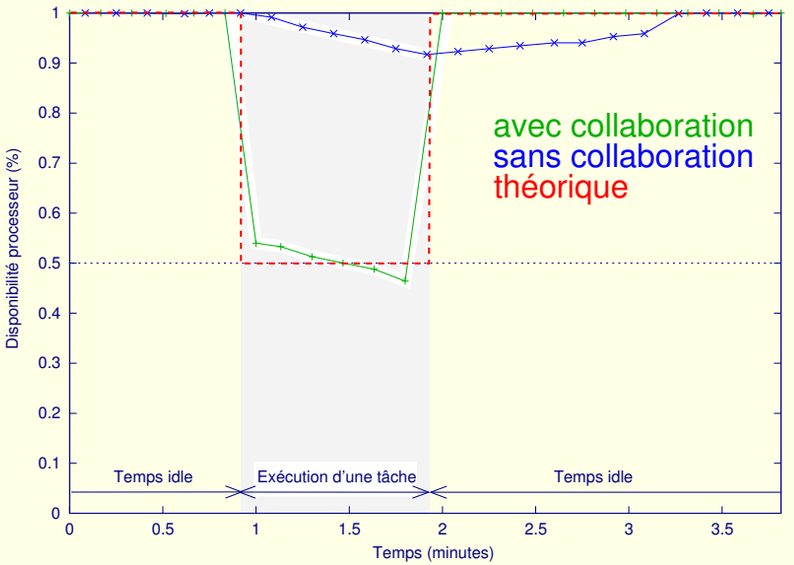


$$\mathrm{bp}(AC) = \min\left(\mathrm{bp}(AB); \mathrm{bp}(BC)\right)$$

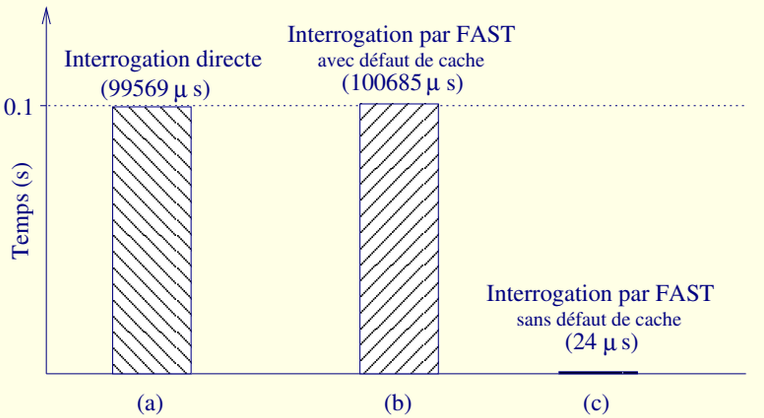
$$\mathrm{lat}(AC) = \mathrm{lat}(AB) + \mathrm{lat}(BC)$$

FAST : améliorations à NWS

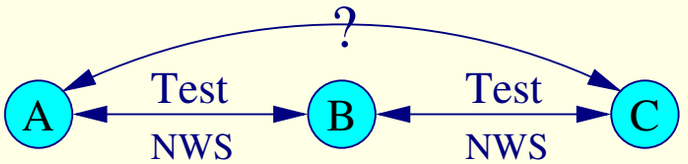
Collaboration NWS / Sched.



Temps de réponse



Sensor in the middle

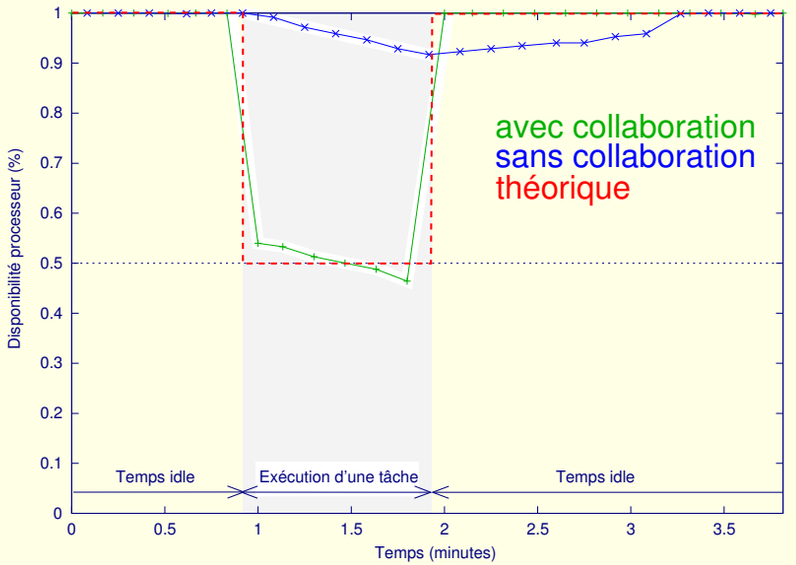


$$\mathrm{bp}(AC) = \min(\mathrm{bp}(AB); \mathrm{bp}(BC))$$

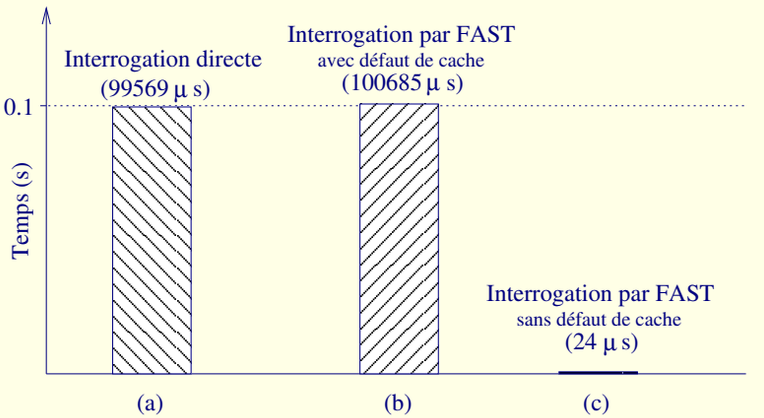
$$\mathrm{lat}(AC) = \mathrm{lat}(AB) + \mathrm{lat}(BC)$$

FAST : améliorations à NWS

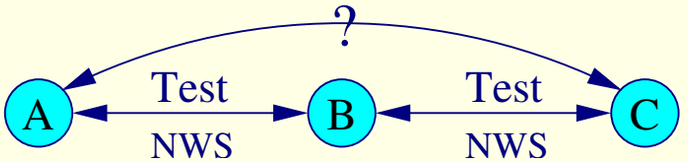
Collaboration NWS / Sched.



Temps de réponse



Sensor in the middle



$$\mathrm{bp}(AC) = \min(\mathrm{bp}(AB); \mathrm{bp}(BC))$$

$$\mathrm{lat}(AC) = \mathrm{lat}(AB) + \mathrm{lat}(BC)$$

Maintenance

- Configuration centralisée
- Déploiement automatique
- Quelques bugs en moins
- (D'autres bugs en plus)

Plan

- Introduction
- Quelle connaissance pour quel ordonnancement
- NWS: Network Weather Service
- FAST: Fast's Agent System Timer
- ALNeM: Application-Level Network Mapper
- Conclusions

Application-Level Network Mapper : Présentation

But : Découvrir la topologie du réseau

Motivation : Ordonnancement

Type d'informations :

« Quelle est la bande passante escomptée entre **A** et **B** ? »

« Et si un transfert sature le lien de **C** vers **D** ? »

Auteurs : Arnaud Legrand, Martin Quinson.

Utilisateurs : Projet en cours, pas encore d'implémentation

Usages prévus : Simulation, Placement, Prédiction de macro-comm.

ALNeM : Pourquoi un nouveau projet ?

Configuration manuelle : source d'erreur.

Traceroute : pas de bande passante, pas d'interférence, niveau 3 d'OSI.

Pathchar : Comme traceroute, et fait varier la taille des paquets

- ☺ Bande passante de tous les liens
- ☹ Pas d'interférence
- ☹ Nécessité d'être root
- ☹ Lent

SNMP, BGP :

- ☹ Pas d'interférence
- ☹ Besoin d'autorisation

ENV : tests d'interférence

- ☹ Vue maître esclave

ALNeM : Notations et formalisation

$\mathrm{bw}(\mathrm{ab})$: bande passante ab ;

$\mathrm{bw}_{/\!/\,\mathrm{cd}}(\mathrm{ab})$: bande passante ab quand cd est saturé.

ALNeM : Notations et formalisation

$bw(ab)$: bande passante ab ;

$bw_{/\!/\,cd}(ab)$: bande passante ab quand cd est saturé.

Définition 1 : $\frac{bw_{/\!/\,cd}(ab)}{bw(ab)} > 0.9 \Leftrightarrow (ab) /\!/_{rl} (cd)$ (pas d'interférence)

Définition 2 : $\frac{bw_{/\!/\,cd}(ab)}{bw(ab)} < 0.7 \Leftrightarrow (ab) \between_{rl} (cd)$ (interférence)

ALNeM : Notations et formalisation

$bw(ab)$: bande passante ab ;

$bw_{/\!/cd}(ab)$: bande passante ab quand cd est saturé.

Définition 1 : $\frac{bw_{/\!/cd}(ab)}{bw(ab)} > 0.9 \Leftrightarrow (ab) /\!/_{rl} (cd)$ (pas d'interférence)

Définition 2 : $\frac{bw_{/\!/cd}(ab)}{bw(ab)} < 0.7 \Leftrightarrow (ab) \curlywedge_{rl} (cd)$ (interférence)

Définition 3 : matrice d'interférence $I(V, \curlywedge_{rl})$

$$I(V, \curlywedge_{rl})(a, b, c, d) = \begin{cases} 1 & \text{si } (ab) \curlywedge_{rl} (cd) \\ 0 & \text{sinon} \end{cases}$$

ALNeM : Notations et formalisation

$bw(ab)$: bande passante ab ;

$bw_{/\!/\,cd}(ab)$: bande passante ab quand cd est saturé.

Définition 1 : $\frac{bw_{/\!/\,cd}(ab)}{bw(ab)} > 0.9 \Leftrightarrow (ab) \,/\!/_{rl}\, (cd)$ (pas d'interférence)

Définition 2 : $\frac{bw_{/\!/\,cd}(ab)}{bw(ab)} < 0.7 \Leftrightarrow (ab) \,\curlywedge_{rl}\, (cd)$ (interférence)

Définition 3 : matrice d'interférence $I(V, \curlywedge_{rl})$

$$I(V, \curlywedge_{rl})(a, b, c, d) = \begin{cases} 1 & \text{si } (ab) \,\curlywedge_{rl}\, (cd) \\ 0 & \text{sinon} \end{cases}$$

INTERFERENCEGRAPH : Étant donné $\mathcal{H}$ et $I(\mathcal{H}, \widetilde{G})$, trouver un graphe $G(V, E)$ et un routage dans ce graphe tels que :

$$\begin{cases} \mathcal{H} \subset V \\ I(\mathcal{H}, \curlywedge_G) = I(\mathcal{H}, \curlywedge_{\widetilde{G}}) \\ |V| \text{ est minimal.} \end{cases} .$$

ALNeM : Algorithme

- Étape 1 : mesure des interférences entre tous les couples de machines
  - Algo naïf en N^4, 30 secondes par pas $\Rightarrow$ 50 jours pour 20 noeuds.
  - Optimisations grâce à traceroute
    - Tests indépendants en parallèle
    - Valider des groupes d'informations

ALNeM : Algorithme

- Étape 1 : mesure des interférences entre tous les couples de machines
  - Algo naïf en N^4, 30 secondes par pas $\Rightarrow$ 50 jours pour 20 noeuds.
  - Optimisations grâce à traceroute
    - Tests indépendants en parallèle
    - Valider des groupes d'informations
- Étape 2 : reconstruction d'un graphe d'interférence. Idée de l'algo :
  - Traiter les sous-arbres
  - Traiter les cycles
  - Traiter les cliques

ALNeM : Algorithme

- Étape 1 : mesure des interférences entre tous les couples de machines
  - Algo naïf en N^4, 30 secondes par pas $\Rightarrow$ 50 jours pour 20 noeuds.
  - Optimisations grâce à traceroute
    - Tests indépendants en parallèle
    - Valider des groupes d'informations
- Étape 2 : reconstruction d'un graphe d'interférence. Idée de l'algo :
  - **Traiter les sous-arbres**
    - H = ensemble de machines interférant avec toutes les autres

    $\Rightarrow$ Ces machines sont dans le même sous-arbre
    - On les remplace par un représentant unique
    - On recommence
  - Traiter les cycles
  - Traiter les cliques

ALNeM : Algorithme

- Étape 1 : mesure des interférences entre tous les couples de machines
  - Algo naïf en N^4, 30 secondes par pas $\Rightarrow$ 50 jours pour 20 noeuds.
  - Optimisations grâce à traceroute
    - Tests indépendants en parallèle
    - Valider des groupes d'informations
- Étape 2 : reconstruction d'un graphe d'interférence. Idée de l'algo :
  - Traiter les sous-arbres
  - **Traiter les cycles**
    - A, B les deux machines ayant le plus d'interférences avec l'exterieur

    $\Rightarrow$ Ces machines sont proches dans un cycle
    - On les separe
    - On recommence
    - On recolle les resultats
  - Traiter les cliques

ALNeM : Algorithme

- Étape 1 : mesure des interférences entre tous les couples de machines
  - Algo naïf en N^4, 30 secondes par pas $\Rightarrow$ 50 jours pour 20 noeuds.
  - Optimisations grâce à traceroute
    - Tests indépendants en parallèle
    - Valider des groupes d'informations
- Étape 2 : reconstruction d'un graphe d'interférence. Idée de l'algo :
  - Traiter les sous-arbres
  - Traiter les cycles
  - **Traiter les cliques**
    - H = ensemble de machines sans interférence entre elles

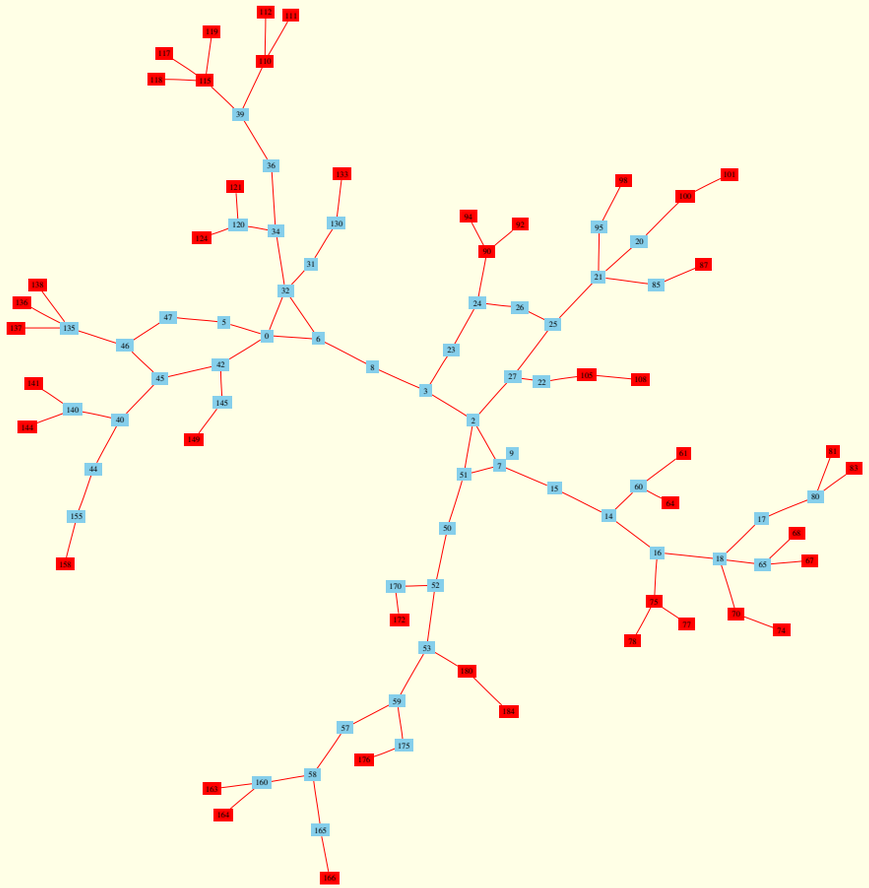
    $\Rightarrow$ Ces machines forment une clique
    - On les connecte en clique

ALNeM : Algorithme

- Étape 1 : mesure des interférences entre tous les couples de machines
  - Algo naïf en N^4, 30 secondes par pas $\Rightarrow$ 50 jours pour 20 noeuds.
  - Optimisations grâce à traceroute
    - Tests indépendants en parallèle
    - Valider des groupes d'informations
- Étape 2 : reconstruction d'un graphe d'interférence. Idée de l'algo :
  - Traiter les sous-arbres
  - Traiter les cycles
  - Traiter les cliques

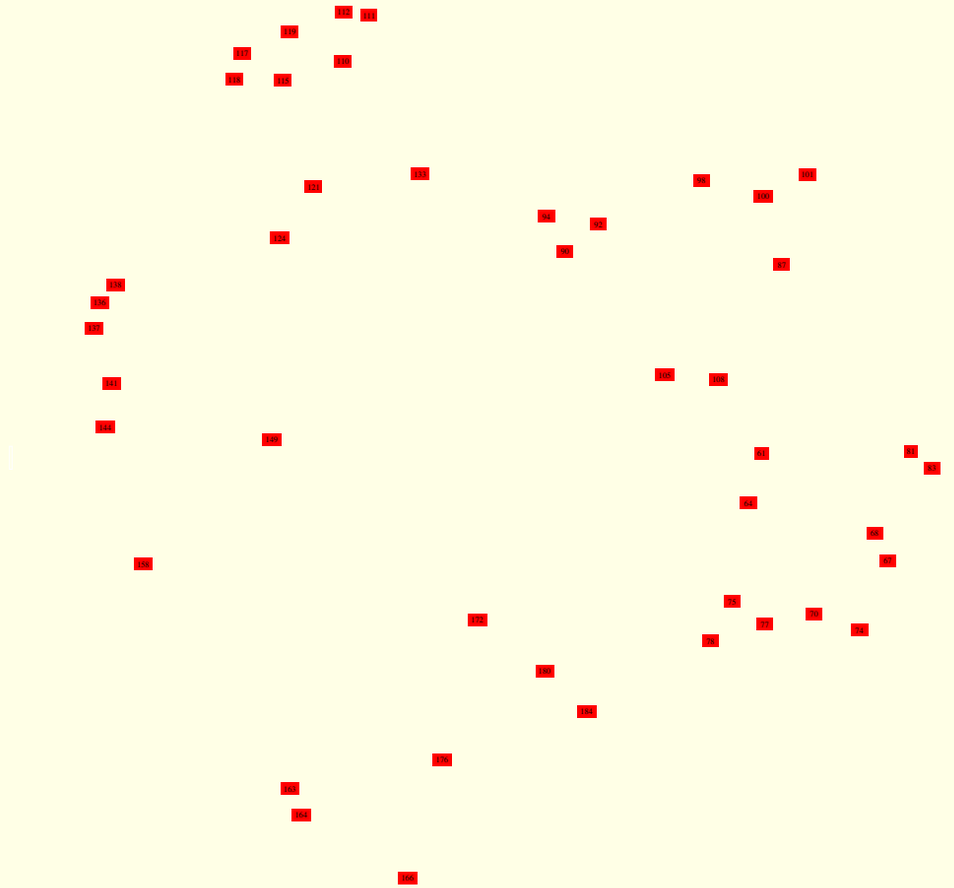
Travaux en cours

ALNeM : Exemple « à la main »



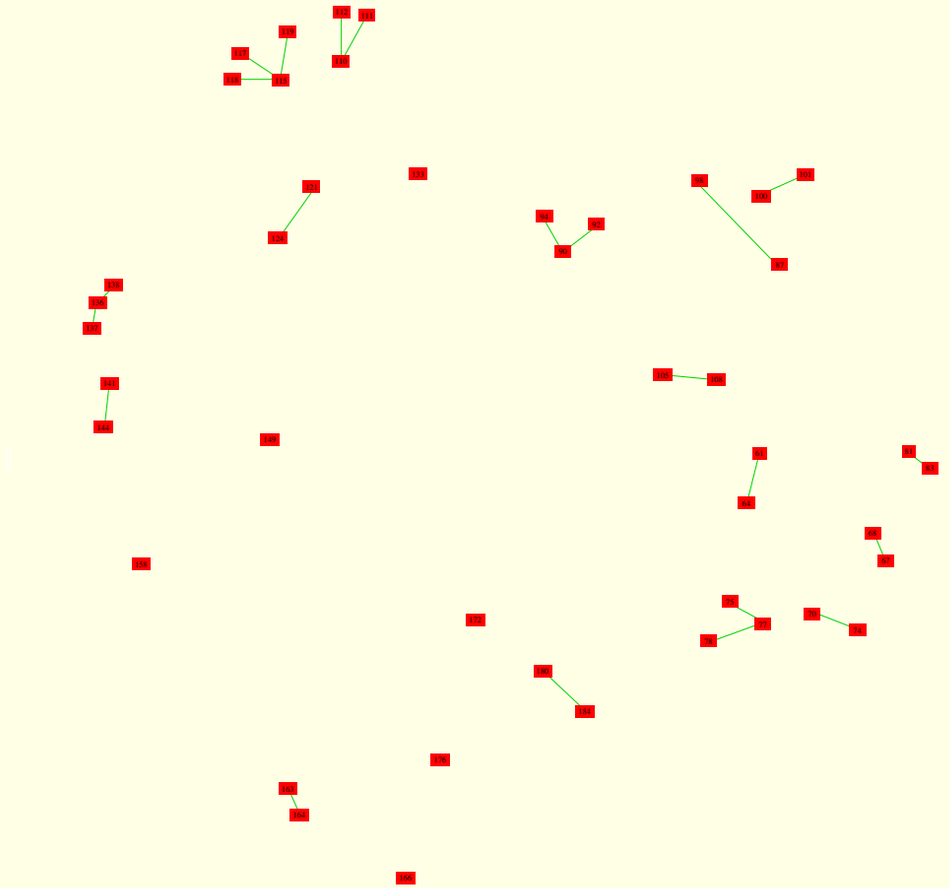
Graphe « réel »

ALNeM : Exemple « à la main »



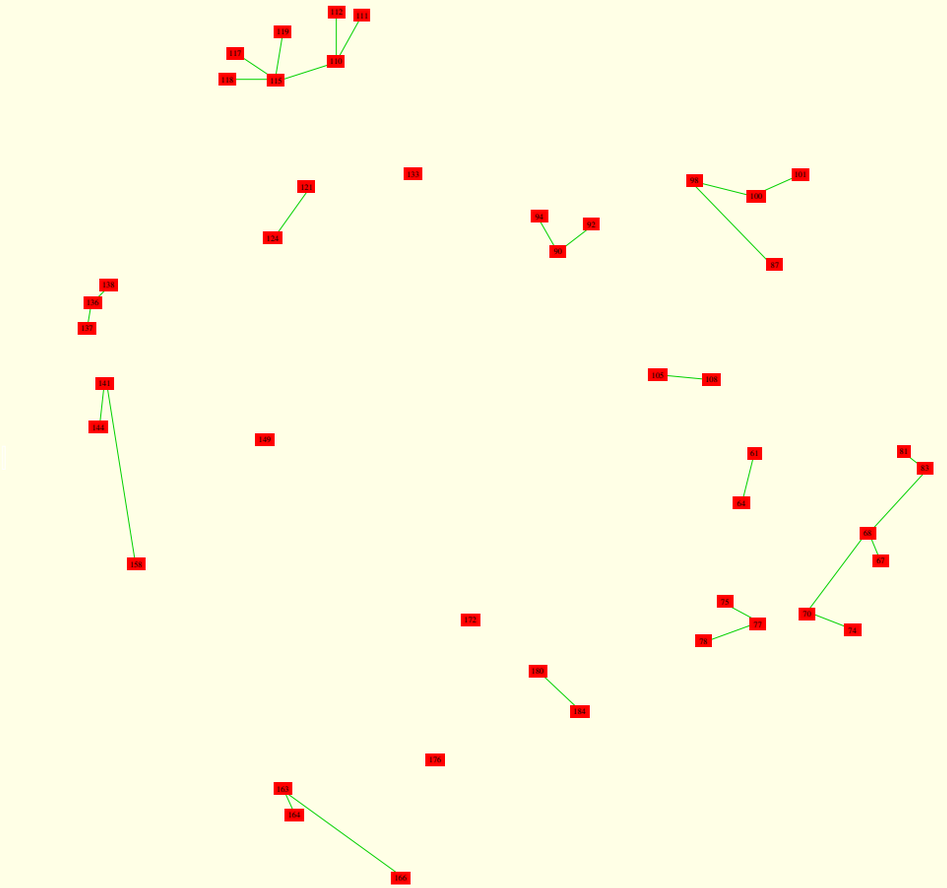
Données de l'algorithme (avec la matrice d'interference)

ALNeM : Exemple « à la main »



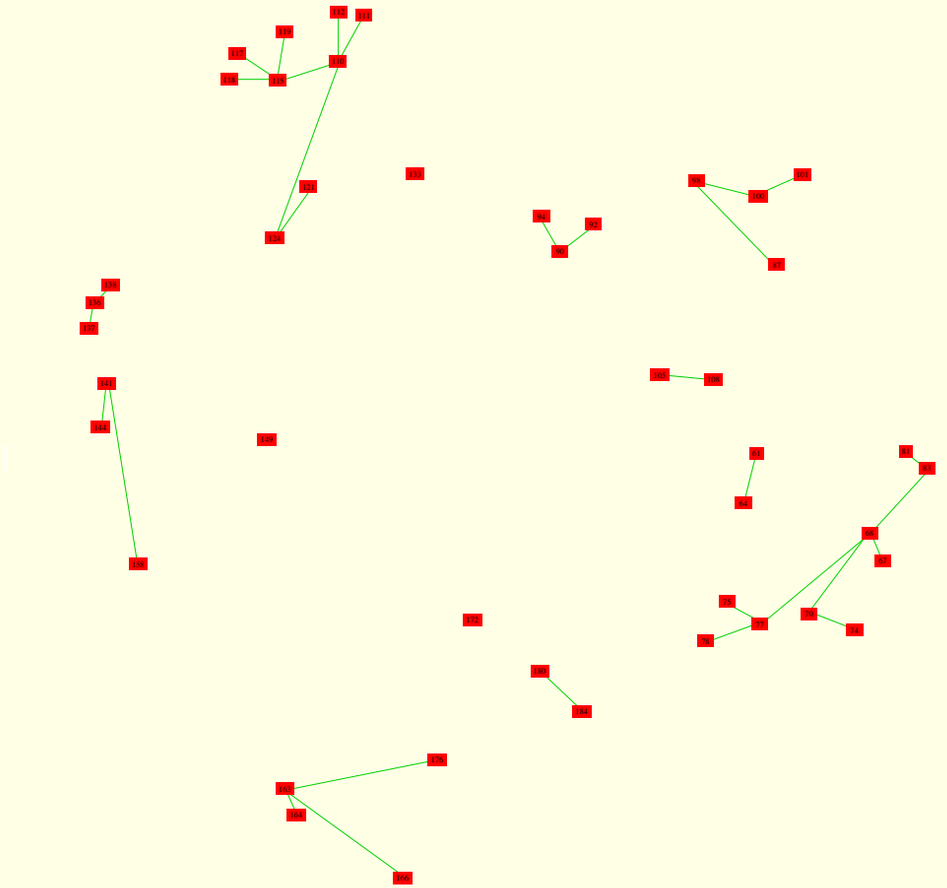
Recherche des sous-arbres : Étape 1

ALNeM : Exemple « à la main »



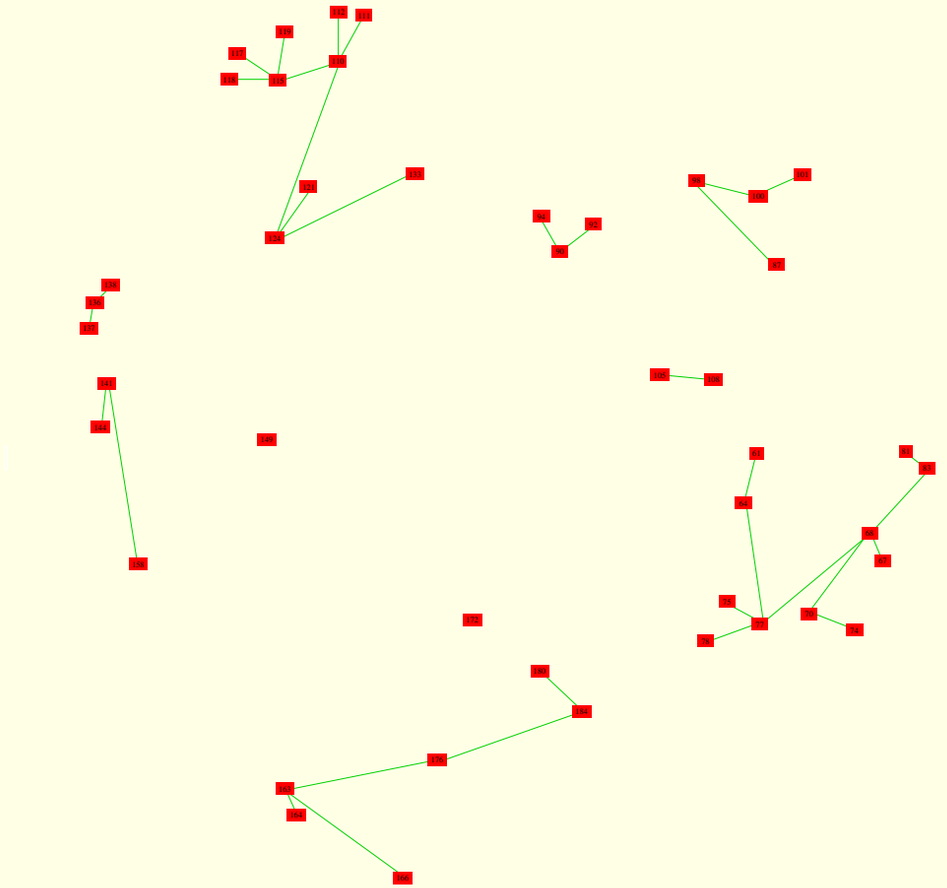
Recherche des sous-arbres : Étape 2

ALNeM : Exemple « à la main »



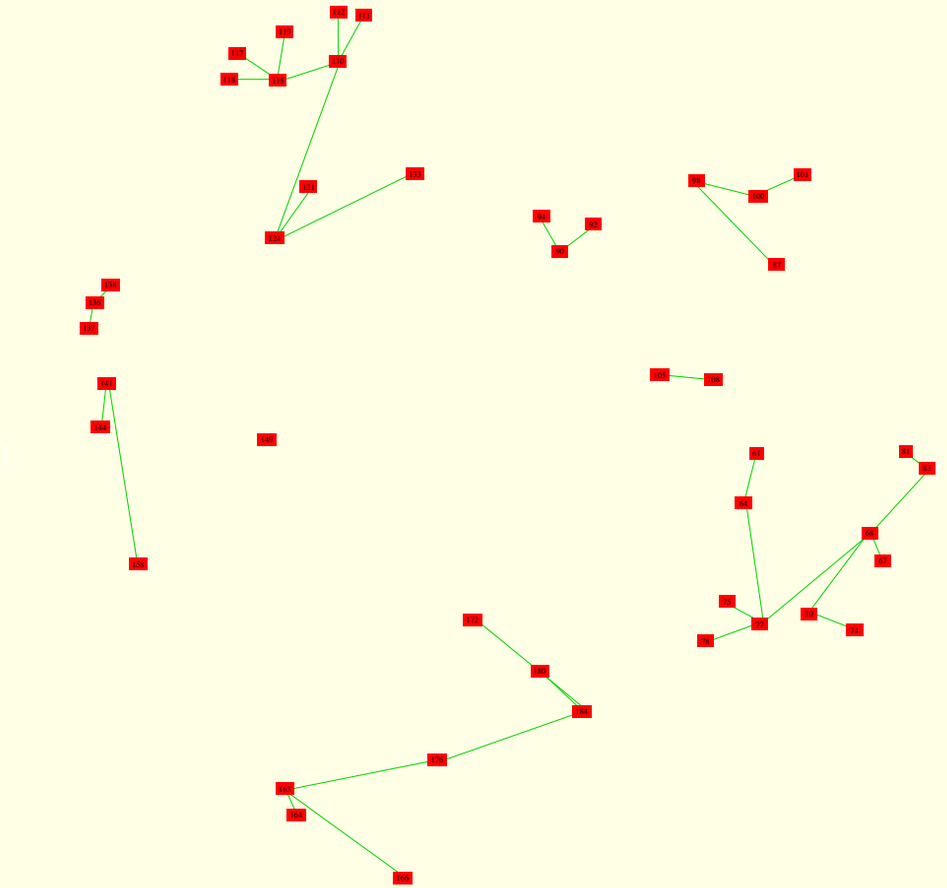
Recherche des sous-arbres : Étape 3

ALNeM : Exemple « à la main »



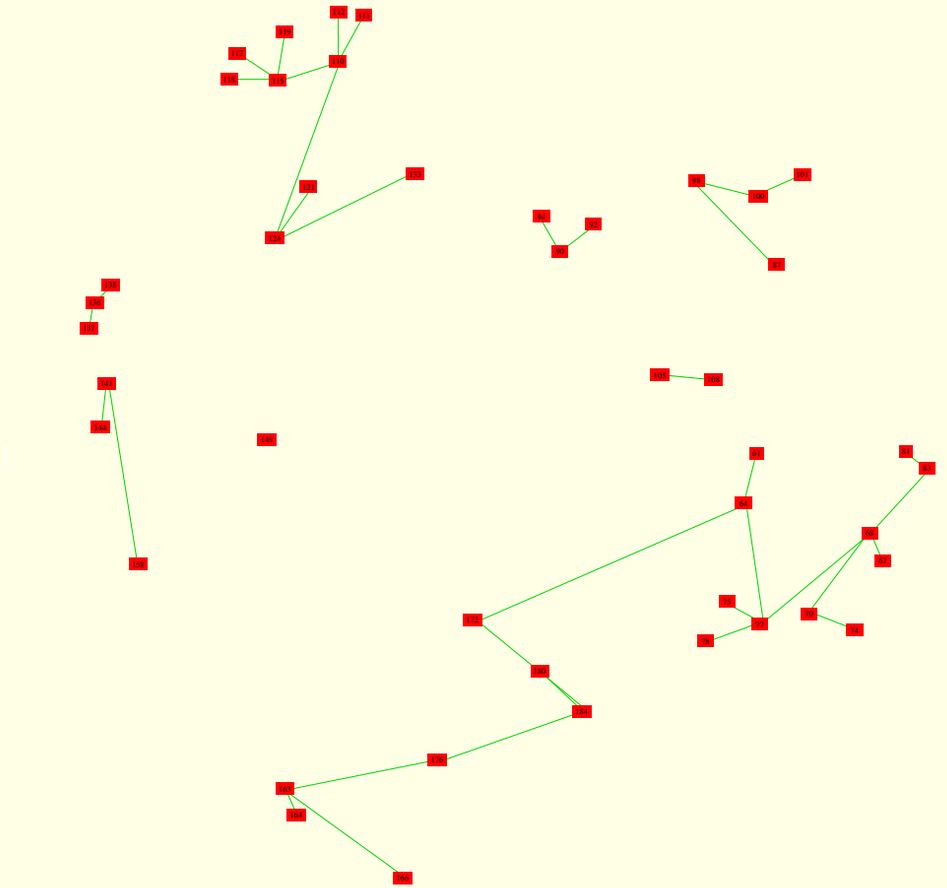
Recherche des sous-arbres : Étape 4

ALNeM : Exemple « à la main »



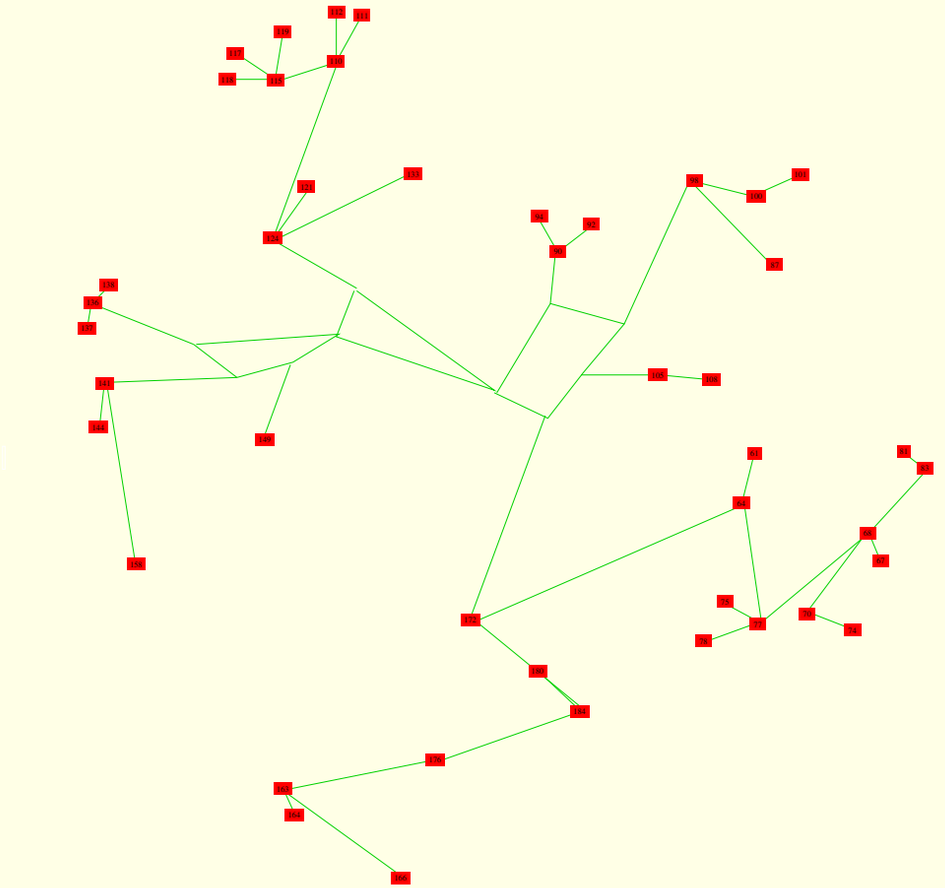
Recherche des sous-arbres : Étape 5

ALNeM : Exemple « à la main »



Résultat de l'algorithme ne cherchant que les arbres

ALNeM : Exemple « à la main »



Résultat cherché (avec les cycles)

Plan

- Introduction
- Quelle connaissance pour quel ordonnancement
- NWS: Network Weather Service
- FAST: Fast's Agent System Timer
- ALNeM: Application-Level Network Mapper
- Conclusions

Conclusions

– La grille est pleine d'avenir, mais reste difficile d'accès

Conclusions

- La grille est pleine d'avenir, mais reste difficile d'accès
  - Problèmes classiques d'ordonnancement
  - Résolution plus complexe par manque d'informations

Conclusions

- La grille est pleine d'avenir, mais reste difficile d'accès
  - Problèmes classiques d'ordonnancement
  - Résolution plus complexe par manque d'informations
- Obtenir ces informations est difficile $\Rightarrow$ outils spécialisés :

  NWS Connaissance des disponibilités de la plate-forme

  FAST Connaissance des besoins des routines ($\oplus$ interface unifiée)

  ALNeM Topologie réseau pour connaître les interactions entre flux

Conclusions

- La grille est pleine d'avenir, mais reste difficile d'accès
  - Problèmes classiques d'ordonnancement
  - Résolution plus complexe par manque d'informations
- Obtenir ces informations est difficile $\Rightarrow$ outils spécialisés :

  NWS Connaissance des disponibilités de la plate-forme

  FAST Connaissance des besoins des routines ($\oplus$ interface unifiée)

  ALNeM Topologie réseau pour connaître les interactions entre flux
- Avenir
  - Améliorer les outils **NWS**, **FAST** et **ALNeM**

Conclusions

- La grille est pleine d'avenir, mais reste difficile d'accès
  - Problèmes classiques d'ordonnancement
  - Résolution plus complexe par manque d'informations
- Obtenir ces informations est difficile $\Rightarrow$ outils spécialisés :

  NWS Connaissance des disponibilités de la plate-forme

  FAST Connaissance des besoins des routines ($\oplus$ interface unifiée)

  ALNeM Topologie réseau pour connaître les interactions entre flux
- Avenir
  - Améliorer les outils **NWS**, **FAST** et **ALNeM**
    - Nouveaux senseurs (NFS, réseau non TCP, activité de l'utilisateur)
    - Prédictions à plus long terme
    - Serveur de nom distribué
    - User-friendly

Conclusions

- La grille est pleine d'avenir, mais reste difficile d'accès
  - Problèmes classiques d'ordonnancement
  - Résolution plus complexe par manque d'informations
- Obtenir ces informations est difficile $\Rightarrow$ outils spécialisés :

  NWS Connaissance des disponibilités de la plate-forme

  FAST Connaissance des besoins des routines ($\oplus$ interface unifiée)

  ALNeM Topologie réseau pour connaître les interactions entre flux
- Avenir
  - Améliorer les outils NWS, **FAST** et **ALNeM**
    - Intégration de Freddy pour les routines parallèles
    - Expérimentations sur des routines creuses
    - D'autres métriques ?
    - D'autres sources d'informations
    - Réactivité, Portabilité, User-friendly.

Conclusions

- La grille est pleine d'avenir, mais reste difficile d'accès
  - Problèmes classiques d'ordonnancement
  - Résolution plus complexe par manque d'informations
- Obtenir ces informations est difficile $\Rightarrow$ outils spécialisés :

  NWS Connaissance des disponibilités de la plate-forme

  FAST Connaissance des besoins des routines ($\oplus$ interface unifiée)

  ALNeM Topologie réseau pour connaître les interactions entre flux
- Avenir
  - Améliorer les outils NWS, FAST et **ALNeM**
    - Généraliser à toutes les formes de graphes (non symétriques)
    - Preuve de NP-complétude
    - Implémenter (simulateur et vraie vie)
    - Optimiser encore
    - Tests dans SimGrid

Conclusions

- La grille est pleine d'avenir, mais reste difficile d'accès
  - Problèmes classiques d'ordonnancement
  - Résolution plus complexe par manque d'informations
- Obtenir ces informations est difficile $\Rightarrow$ outils spécialisés :

  NWS Connaissance des disponibilités de la plate-forme

  FAST Connaissance des besoins des routines ($\oplus$ interface unifiée)

  ALNeM Topologie réseau pour connaître les interactions entre flux
- Avenir
  - Améliorer les outils NWS, FAST et ALNeM
  - Utiliser toutes ces informations
    - Intégration de FAST à DIET
    - Intégration d'ALNeM à NWS et FAST
    - Utilisation d'ALNeM pour automatiser le placement de NWS et DIET

Conclusions

- La grille est pleine d'avenir, mais reste difficile d'accès
  - Problèmes classiques d'ordonnancement
  - Résolution plus complexe par manque d'informations
- Obtenir ces informations est difficile $\Rightarrow$ outils spécialisés :

  NWS Connaissance des disponibilités de la plate-forme

  FAST Connaissance des besoins des routines ($\oplus$ interface unifiée)

  ALNeM Topologie réseau pour connaître les interactions entre flux
- Avenir
  - Améliorer les outils NWS, FAST et ALNeM
  - Utiliser toutes ces informations
  - Implémentations multiples, tâches malléables, *etc.*

Conclusions

- La grille est pleine d'avenir, mais reste difficile d'accès
  - Problèmes classiques d'ordonnancement
  - Résolution plus complexe par manque d'informations
- Obtenir ces informations est difficile $\Rightarrow$ outils spécialisés :

  NWS Connaissance des disponibilités de la plate-forme

  FAST Connaissance des besoins des routines ($\oplus$ interface unifiée)

  ALNeM Topologie réseau pour connaître les interactions entre flux
- Avenir
  - Améliorer les outils NWS, FAST et ALNeM
  - Utiliser toutes ces informations
  - Implémentations multiples, tâches malléables, *etc.*

Any questions ?