Diagnostiquer des problèmes techniques

(dans le contexte de l'administration système et réseaux sous Linux)

Lucas Nussbaum

lucas.nussbaum@univ-lorraine.fr

Licence professionnelle ASRALL

Administration de systèmes, réseaux et applications à base de logiciels libres

Plan

1. Introduction
2. Questions à se poser
3. Méthodes pour l'investigation
4. Outils pour l'investigation
5. Après avoir identifié la cause

Introduction

- Une des principales tâches de l'administrateur système : Diagnostiquer (et résoudre) des problèmes
  - Trouver la cause (l'origine) d'un dysfonctionnement
  - Accessoirement, trouver la meilleure solution
    $\neq$ *Workaround* : contourner le problème

1. Quelques histoires : `http://beza1e1.tuxen.de/lore/`

Introduction

- Une des principales tâches de l'administrateur système : Diagnostiquer (et résoudre) des problèmes
  - Trouver la cause (l'origine) d'un dysfonctionnement
  - Accessoirement, trouver la meilleure solution
    $\neq$ *Workaround* : contourner le problème
- Beaucoup de problèmes simples
  - Faciles à comprendre et à résoudre
  - Pas vraiment besoin de méthode particulière

1. Quelques histoires : `http://beza1e1.tuxen.de/lore/`

Introduction

- Une des principales tâches de l'administrateur système : Diagnostiquer (et résoudre) des problèmes
  - Trouver la cause (l'origine) d'un dysfonctionnement
  - Accessoirement, trouver la meilleure solution
    $\neq$ *Workaround* : contourner le problème
- Beaucoup de problèmes simples
  - Faciles à comprendre et à résoudre
  - Pas vraiment besoin de méthode particulière
- Parfois, des problèmes très difficiles [1]

1. Quelques histoires : `http://beza1e1.tuxen.de/lore/`

Introduction

- Une des principales tâches de l'administrateur système : Diagnostiquer (et résoudre) des problèmes
  - Trouver la cause (l'origine) d'un dysfonctionnement
  - Accessoirement, trouver la meilleure solution
    $\neq$ *Workaround* : contourner le problème
- Beaucoup de problèmes simples
  - Faciles à comprendre et à résoudre
  - Pas vraiment besoin de méthode particulière
- Parfois, des problèmes très difficiles [1]
- Outils :
  - Connaissances techniques
  - Expérience de l'administrateur système
  - Méthodologie (encore plus importante quand on n'a pas d'expérience)

1. Quelques histoires : `http://beza1e1.tuxen.de/lore/`

Exemple

Problème :

Performance réseau trop faible lors de transferts réseaux à partir d'une machine virtuelle Xen

Connaissances techniques utiles :

Fonctionnement des réseaux, de la virtualisation, et du réseau dans le contexte de la virtualisation

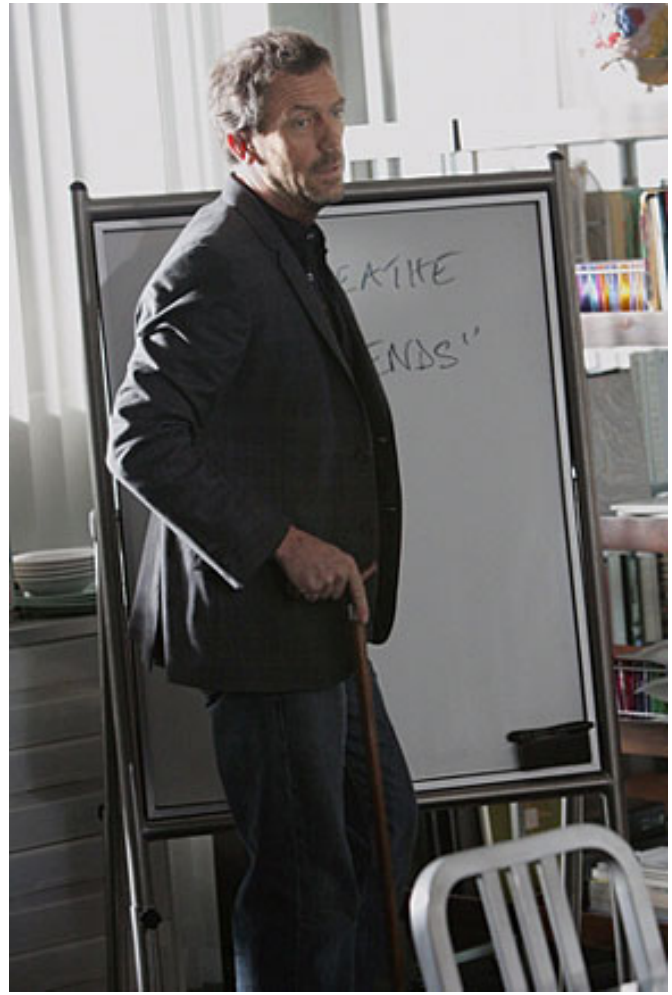
Expérience utile :

Savoir par expérience qu'il peut y avoir des problèmes de performance liés au *TCP segmentation offloading* avec les VM Xen sous Debian etch

Méthodologie :

À l'aide d'un raisonnement logique, être capable d'isoler la source du problème

Diagnostiquer : pas spécifique à l'informatique



(série *House*)

`http://en.wikipedia.org/wiki/Medical_diagnosis`

En Informatique

- Méthodologie beaucoup moins formalisée : *Recettes de cuisine*
- Ce cours :
  - Vous permettre de systématiser le diagnostic : *feuille de route*
  - Éviter des erreurs courantes
  - Vous donner quelques pistes classiques

Imaginez que vous êtes face à un problème difficile . . .

Plan

1. Introduction
2. Questions à se poser
3. Méthodes pour l'investigation
4. Outils pour l'investigation
5. Après avoir identifié la cause

Questions à se poser : caractérisation

- **Caractérisation du problème :**
  - Quels sont les symptômes ? Qu'est-ce qui marche ou pas ?
  - Quel est le comportement attendu ?
  - Y a-t-il plusieurs symptômes en même temps, peut-être liés ?
  - Quelles sont les conséquences du problème ? (gravité ?)

Questions à se poser : caractérisation

- **Caractérisation du problème :**
  - Quels sont les symptômes ? Qu'est-ce qui marche ou pas ?
  - Quel est le comportement attendu ?
  - Y a-t-il plusieurs symptômes en même temps, peut-être liés ?
  - Quelles sont les conséquences du problème ? (gravité ?)
- **Fréquence du problème :**
  - Est-ce la première fois que ce problème apparaît ?
  - Est-il possible qu'il se soit déjà produit sans qu'il soit détecté ?
  - Depuis quand le problème est-il présent ?
  - Y a-t-il un moyen de rechercher des occurences précédentes ?

Questions à se poser : environnement

- Environnement logiciel : versions / branches
  - du logiciel affecté ?
  - des logiciels utilisés par ce logiciel affecté ? (bibliothèques, noyau, distribution, . . .)
- Matériel spécifique ?
- Perturbations (logicielles, matérielles ou réseau) ?
- Y a-t-il quelque chose de particulier dans l'environnement ?

Questions à se poser : reproductibilité

- Le problème est-il reproductible ?
- Peut-on simplifier la procédure permettant de reproduire le problème (*test case*) ?
- Se reproduit-il systématiquement, ou de manière aléatoire ?
- Attention aux *Heisenbugs* : bug qui ne se manifeste pas lorsque des outils de détection sont utilisés pour le rechercher.

Plan

1. Introduction
2. Questions à se poser
3. Méthodes pour l'investigation
4. Outils pour l'investigation
5. Après avoir identifié la cause

Méthodes

- Si le problème est présent en permanence : ***localiser* le problème**
- Si le problème est reproductible : **réduire le *test case***

Méthodes

- Si le problème est présent en permanence : ***localiser* le problème**
- Si le problème est reproductible : **réduire le *test case***

Dans les deux cas :

- Objectif : faciliter l'analyse et réduire le nombre de suspects
- Méthode (en général) : **recherche par dichotomie**
  - Sur la pile logicielle (noyau, bibliothèques, . . .)
  - Sur la pile réseau (Ethernet, IP, TCP/UDP, . . .)
  - Sur l'historique des modifications d'un logiciel (`git bisect`)

  Deux moyens d'avancer :
  - Remplacer tour-à-tour les différents éléments
    Exemple : changer de noyau
  - Tester les différentes couches séparément
    Exemple : problème avec connexions TCP, est-ce que *ping* marche ?

Méthodes (2)

- Si vous ne pouvez pas reproduire le problème :
  - Analyse pendant que le problème se produit
  - Analyse post-mortem

  $\rightarrow$ Beaucoup plus difficile !

Bonnes pratiques

- **Utilisez au maximum les informations acquises**
  Souvent, l'information est déjà là, il faut juste la trouver !
  En cas de message d'erreur peu clair, vous pouvez regarder le code source pour savoir pourquoi ce message est affiché

Bonnes pratiques

- **Utilisez au maximum les informations acquises**
  Souvent, l'information est déjà là, il faut juste la trouver !
  En cas de message d'erreur peu clair, vous pouvez regarder le code source pour savoir pourquoi ce message est affiché
- **Choisissez bien vos hypothèses et pistes d'investigations**
  - Ne cherchez pas que là où c'est facile (*streetlight effect*)
  - Quelles sont leurs probabilités ?
    (Les bugs graves dans les parties critiques du noyau sont rares ☺)
    Exemple de TCP HyStart : thread1 thread2 thread3 thread4
  - Quelles sont leurs coûts ?
    en temps d'investigation, en conséquences sur les utilisateurs, . . .
  - Quelles informations pourront être acquises :
    - ★ en cas de succès ?
    - ★ en cas d'échec ?
  - Comment pourrez-vous vérifier votre hypothèse une fois l'expérience mise en place ?

Bonnes pratiques (2)

- **Documentez les investigations**
  - Liste des occurences du problème, extraits de logs, ...
  - Avec date, heure et nom de la machine (si besoin)
  - Hypothèses rejetées (et justification)
  - Permet à quelqu'un d'autre d'apporter de l'aide
  - Permet de tester des hypothèses

Bonnes pratiques (2)

- **Documentez les investigations**
  - Liste des occurences du problème, extraits de logs, ...
  - Avec date, heure et nom de la machine (si besoin)
  - Hypothèses rejetées (et justification)
  - Permet à quelqu'un d'autre d'apporter de l'aide
  - Permet de tester des hypothèses
- **Appelez à l'aide**
  - À lire : *How To Ask Questions The Smart Way* : `http://www.catb.org/esr/faqs/smart-questions.html`
  - Points les plus importants :
    - ★ Choisissez votre cible avec soin

      Vous serez ignoré si vous visez trop haut pour un problème simple
    - ★ Soyez sûr de votre analyse avant de hurler au bug
    - ★ Décrivez les symptômes le plus précisément possible

      (Une info que vous jugez inutile peut être utile pour quelqu'un d'autre)
    - ★ Ne mélangez pas symptômes (= faits) et hypothèses

Plan

1. Introduction
2. Questions à se poser
3. Méthodes pour l'investigation
4. Outils pour l'investigation
5. Après avoir identifié la cause

Documentations

Différents niveaux de qualité/confiance :

1. Code source, changelogs
2. Rapports de bugs (qualité différente selon les projets)
3. Listes de diffusion (*mailing lists*), souvent en anglais
4. Documentations officielles (`man`, site web, `/usr/share/doc/`)
5. Documentations non officielles (how-to)
6. Forums (car peu de développeurs y participent)

Soyez critiques vis-à-vis des documentations !

- Fraîcheur ? (attention aux traductions !)
- Qui en est l'auteur ? Peut-on lui faire confiance ?
- Est-ce vraiment le même problème que le mien ?
- Est-ce la même version du logiciel que la mienne ?

Outils classiques

- Logs :
  - Générés par la plupart des services dans `/var/log/`
  - Niveau de log paramétrable pour chaque service ou via `syslog`
  - Pour le noyau : `dmesg` (recopiés dans `/var/log/messages`)
    - Certains crashs de processus sont loggés par le noyau, et peuvent apporter des informations supplémentaires

Outils classiques

- Logs :
  - Générés par la plupart des services dans `/var/log/`
  - Niveau de log paramétrable pour chaque service ou via `syslog`
  - Pour le noyau : `dmesg` (recopiés dans `/var/log/messages`)
    - ★ Certains crashs de processus sont loggés par le noyau, et peuvent apporter des informations supplémentaires

- Logiciel :
  - `-verbose, -debug`
  - Parfois mode *debug* activable à la compilation

Outils classiques

- Logs :
  - Générés par la plupart des services dans `/var/log/`
  - Niveau de log paramétrable pour chaque service ou via `syslog`
  - Pour le noyau : `dmesg` (recopiés dans `/var/log/messages`)
    - ★ Certains crashs de processus sont loggés par le noyau, et peuvent apporter des informations supplémentaires

- Logiciel :
  - `-verbose, -debug`
  - Parfois mode *debug* activable à la compilation

- Système :
  - `gdb` et autres débogueurs (`valgrind`)
  - `ltrace` : trace les appels de bibliothèques
  - `strace` : trace les appels systèmes
  - Systemtap, bpf/bcc : instrumentation du noyau

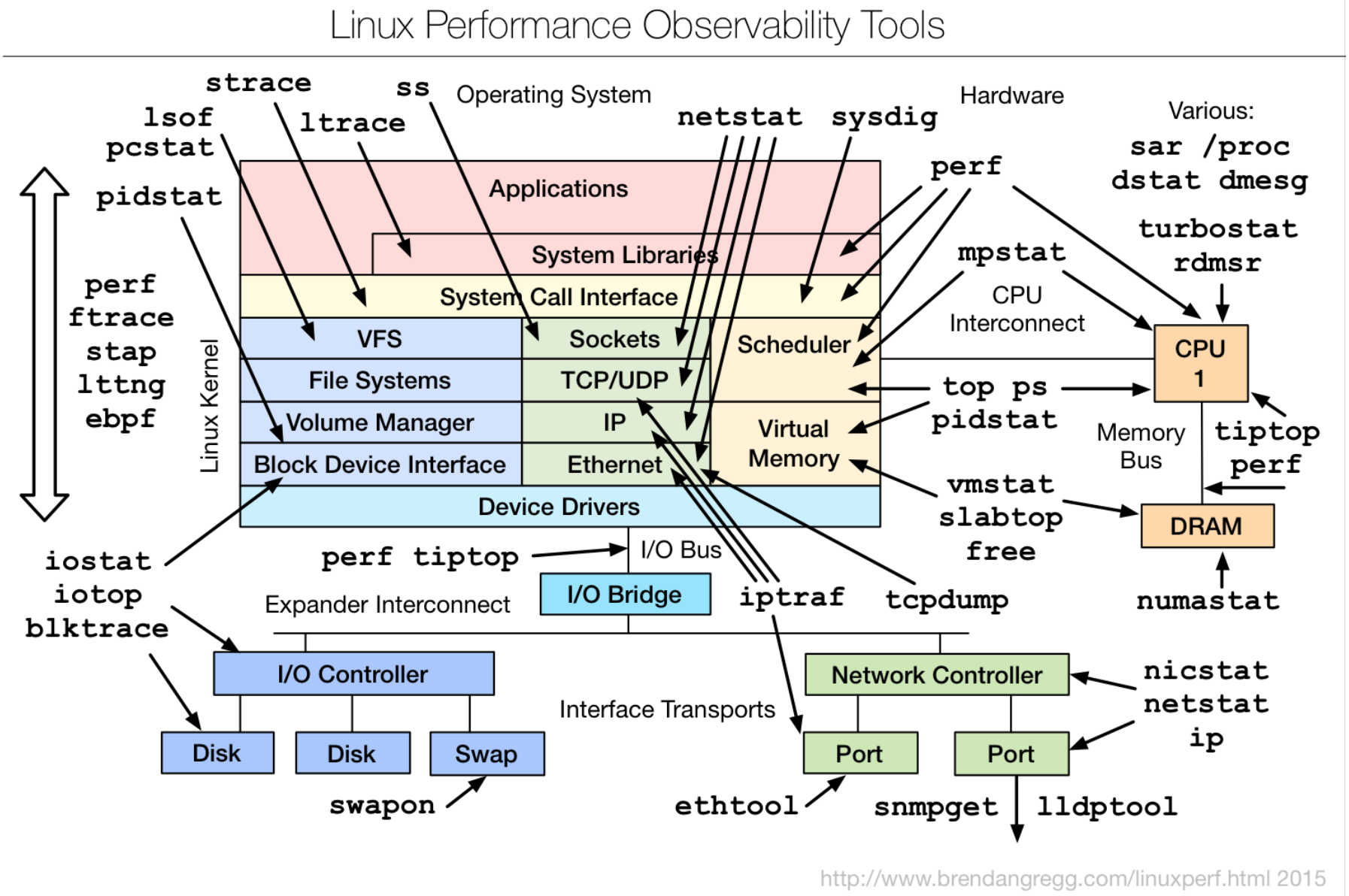
Outils classiques (2)

- Réseau :
  - `ethtool` : Ethernet / carte réseau
  - `ifconfig`, `route`, `ip`, `netstat -pauletn`,...
  - `nmap` : scanner de ports
  - `tcpdump` : analyseur de trafic
  - `wireshark` : analyseur de trafic plus évolué (graphique)
    Utilisation de wireshark à distance (*Network pipes*) :
    `wireshark -k -i <(ssh host "dumpcap -P -w - -f 'not tcp port 22'")` (Il faut se connecter en root à *host*)

Outils classiques (2)

- Réseau :
  - `ethtool` : Ethernet / carte réseau
  - `ifconfig`, `route`, `ip`, `netstat -pauletn`,...
  - `nmap` : scanner de ports
  - `tcpdump` : analyseur de trafic
  - `wireshark` : analyseur de trafic plus évolué (graphique)
    Utilisation de wireshark à distance (*Network pipes*) :
    `wireshark -k -i <(ssh host "dumpcap -P -w - -f 'not tcp port 22'")` (Il faut se connecter en root à *host*)
- Outils de diagnostic spécifiques à des matériels ou technologies :
  - `/proc/mdstat` pour le RAID logiciel
  - `tune2fs` pour les systèmes de fichier `ext[2-4]`
  - ...

Outils classiques (3)



source : `http://www.brendangregg.com/linuxperf.html`

Méthodes classiques (performance)

- USE method
  = When there's a performance problem, check for each resource : Utilisation, Saturation, Errors
- RED method
  = When there's a performance problem, check for each service : Rate (number of req/s), Errors (number of failing req/s), Duration
- *Four golden signals* (Latency, Traffic, Errors, Saturation)

Outils basiques à regarder systématiquement

- Pendant l'incident :
  - `top` : load average ? swap ? utilisation CPU ? commandes utiles : **1** (développer les CPUs), **M** (trier par utilisation mémoire), **i** (n'afficher que les processus actifs), **u** (restreindre à un utilisateur)
  - suivi général : `vmstat 1`, `dstat`
  - problèmes I/O : `iostat -xz 1`
  - problèmes réseaux : `slurm` (regarder les compteurs d'erreurs)
- Pendant l'incident ou post-mortem (après un crash, etc) :
  - logs (`/var/log/*`)
    - Avec systemd : `journalctl -e`
  - `dmesg | tail`
  - Métrologie, monitoring
  - Utilisez au maximum les infos acquises (messages d'erreur)

Plan

1. Introduction
2. Questions à se poser
3. Méthodes pour l'investigation
4. Outils pour l'investigation
5. Après avoir identifié la cause

Après avoir identifié la cause

- Est-ce la cause *racine*, ou une conséquence ?
- Comment auriez-vous pû le trouver plus vite ? Êtes-vous passé à côté d'indices ?
- Y a-t-il un bug à signaler ?
  Comment signaler efficacement un bug
  `http://www.chiark.greenend.org.uk/~sgtatham/bugs-fr.html`

  « Le premier objectif d'un rapport de bug est de permettre au programmeur de voir le bug de ses propres yeux. Si vous *ne pouvez* le faire planter devant lui, donnez lui des instructions *détaillées* afin qu'il puisse le faire planter par lui-même.
  Si le premier objectif ne peut être atteint, et donc si le programmeur ne peut voir l'erreur se produire, le second objectif d'un rapport de bug est de décrire ce qui s'est mal passé. Décrivez tout, en détails. Dites ce que vous avez vu, mais dites aussi ce que vous vous attendiez à voir. Recopiez les messages d'erreur, *surtout* s'ils contiennent des nombres. Bien entendu, si vous pensez en être capable, diagnostiquez l'erreur par vous-même, mais si vous le faites, vous devriez tout de même lui communiquer les symptômes. »

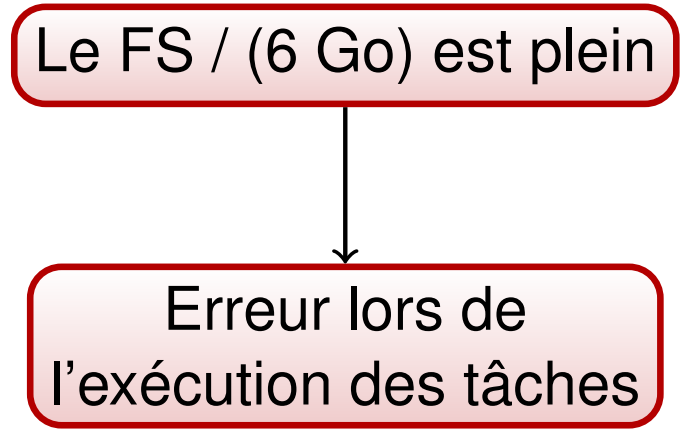
L'arbre des causes : comprendre les origines

- Objectifs :
  - Guider, structurer l'identification des causes d'un incident
  - Aider à identifier les pistes d'action possibles pour
    - Éviter que cet incident se reproduise
    - Éviter d'autres incidents partageant la même cause
- Utilisé pour les accidents du travail
- Méthodologie :
  1. Partir de l'événement non souhaité
  2. Se demander : pourquoi est-ce arrivé ?
  3. Itérer
  4. Puis, pour chaque cause identifiée, chercher des pistes d'action, qui peuvent être techniques, humaines ou organisationnelles (*brainstorming* $\rightsquigarrow$ pistes pas forcément toutes bonnes)

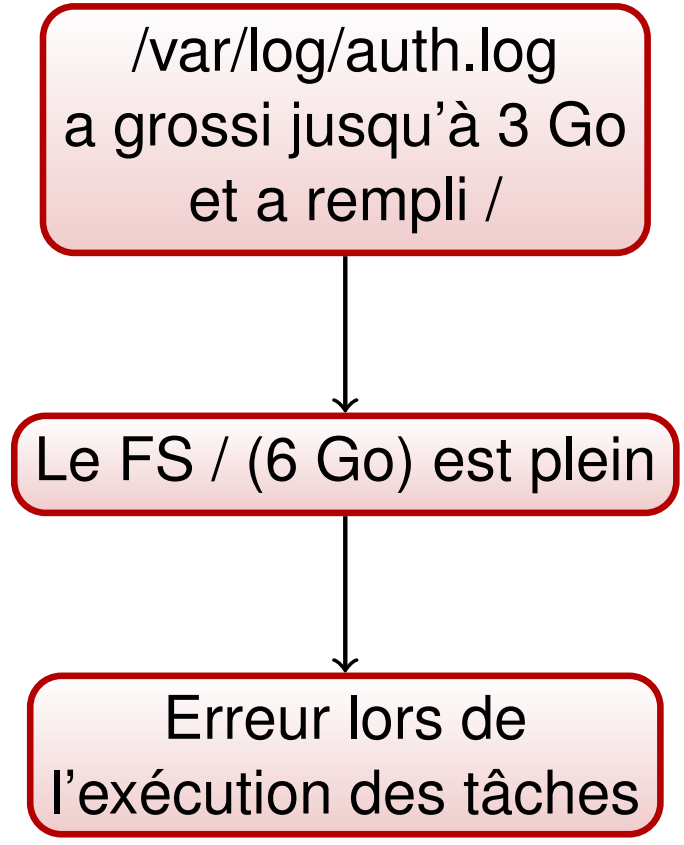
L’arbre des causes : exemple

Erreur lors de
l’exécution des tâches

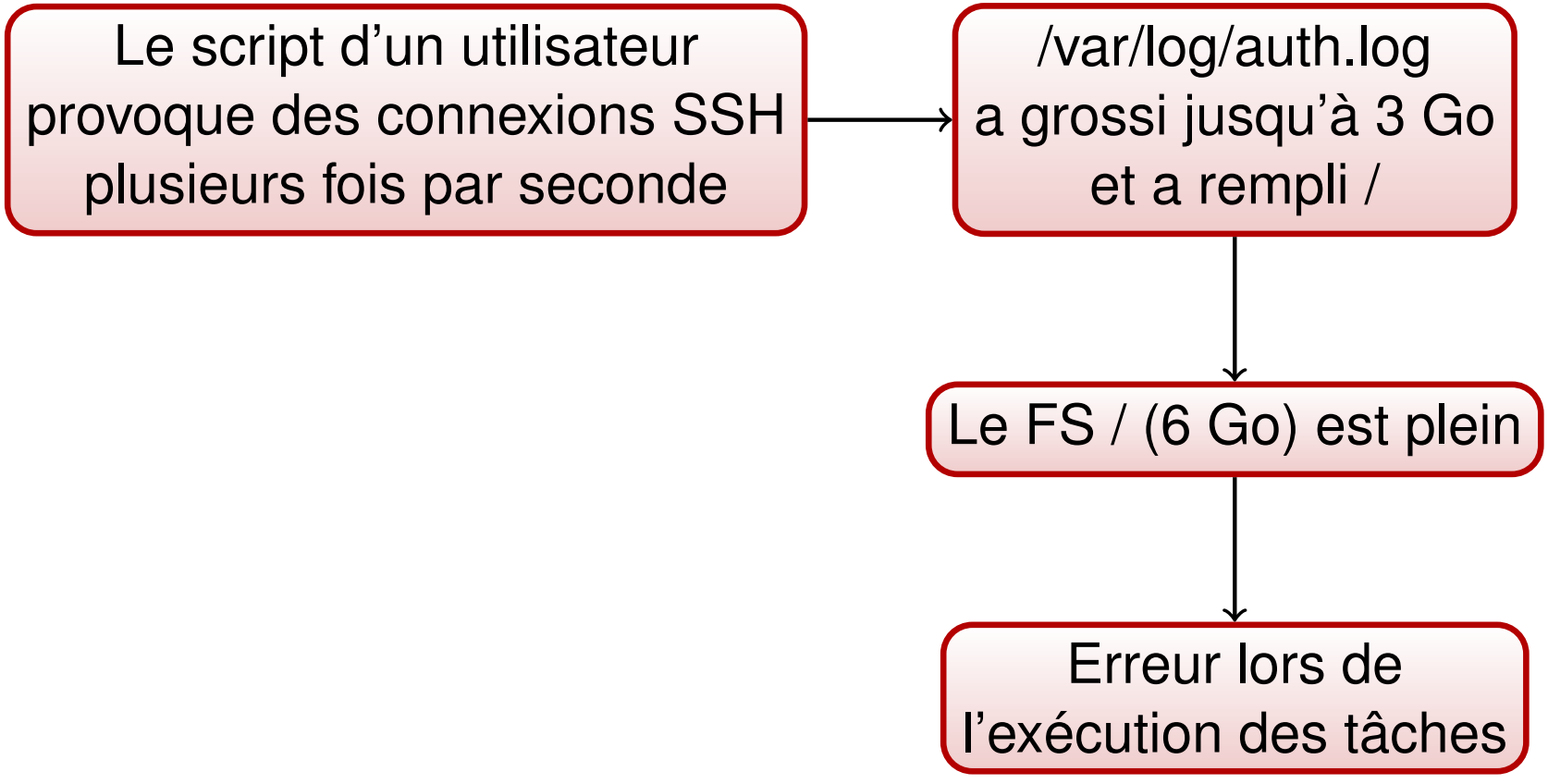
L’arbre des causes : exemple



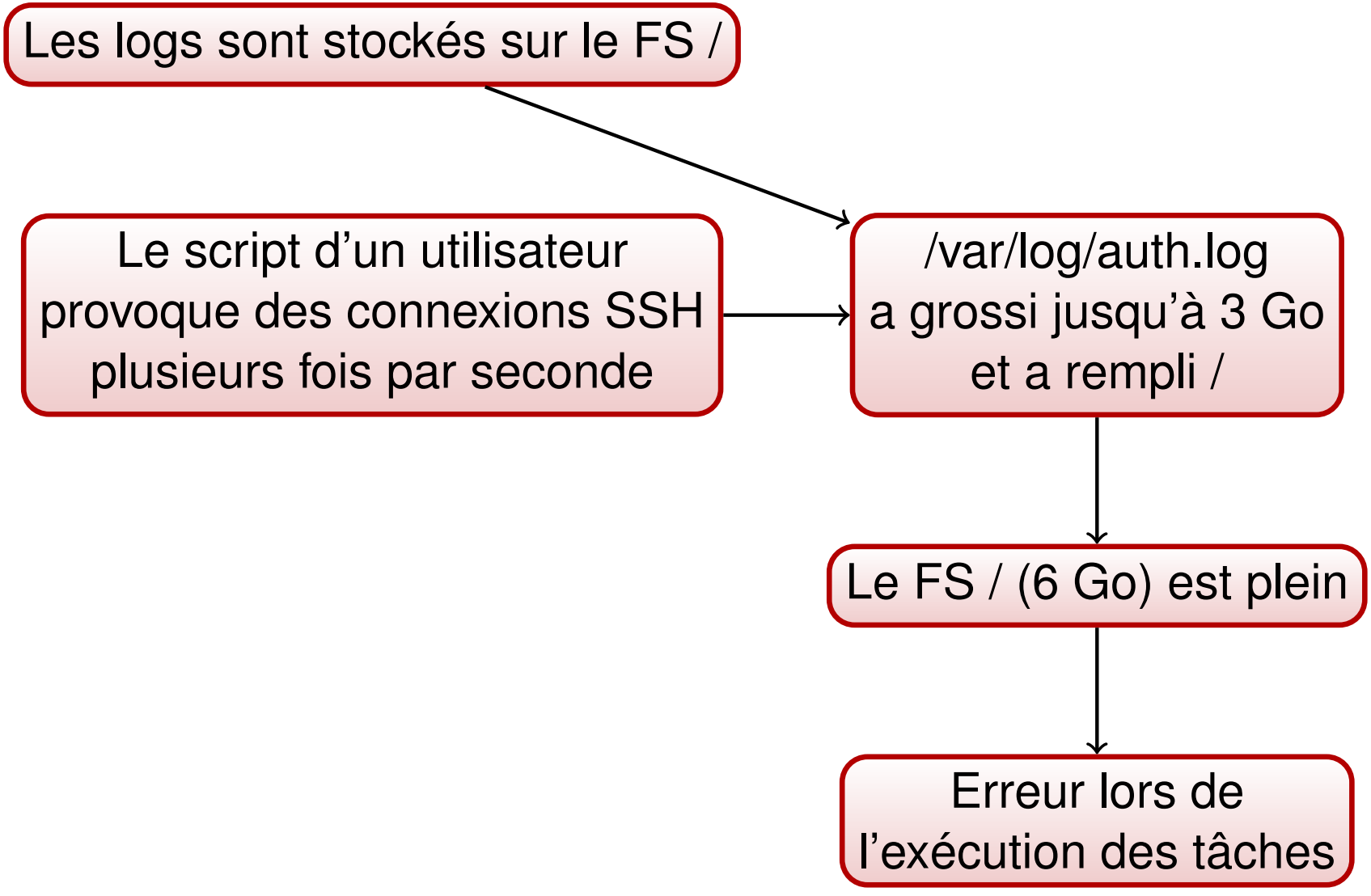
L’arbre des causes : exemple



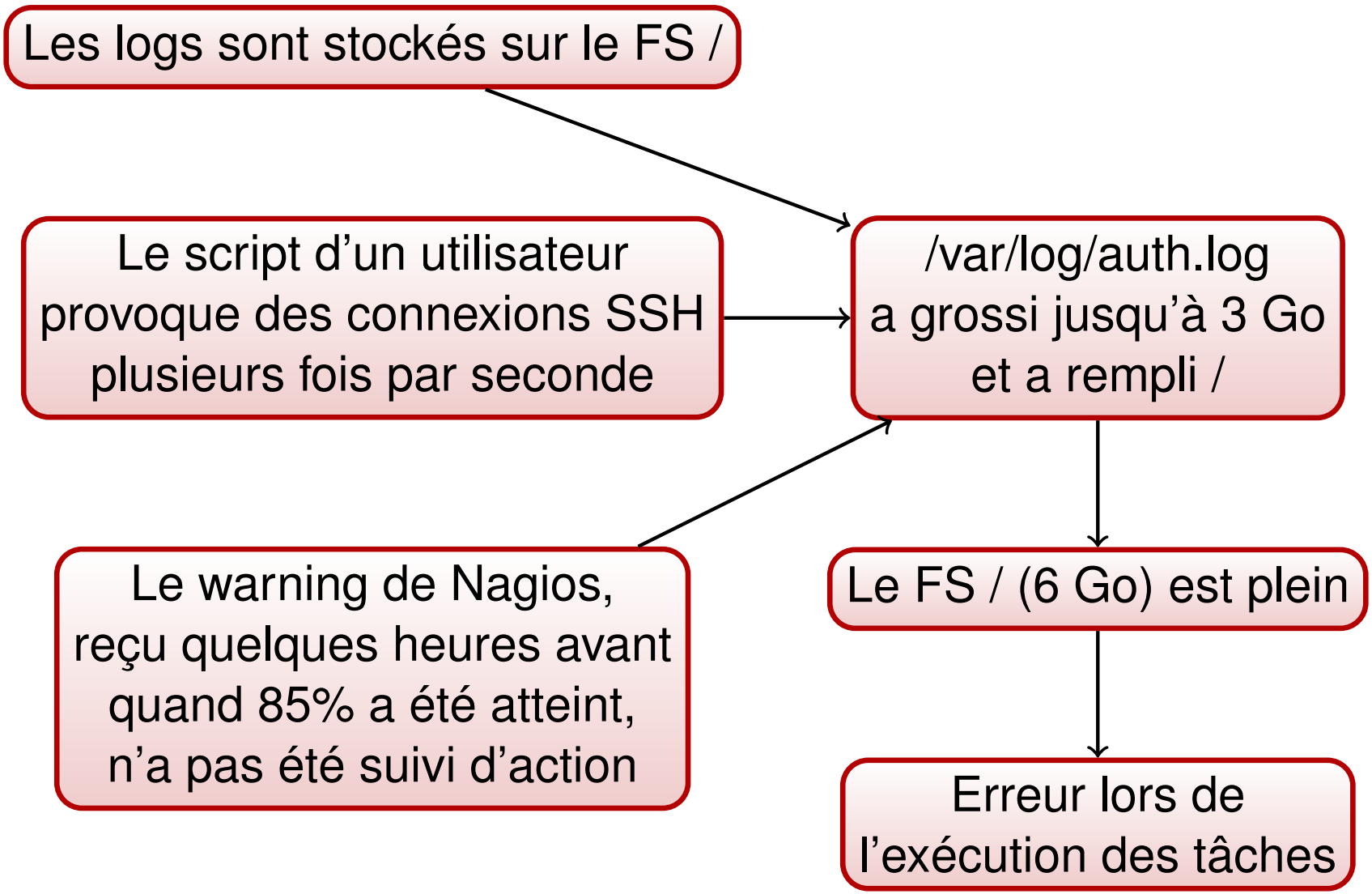
L’arbre des causes : exemple



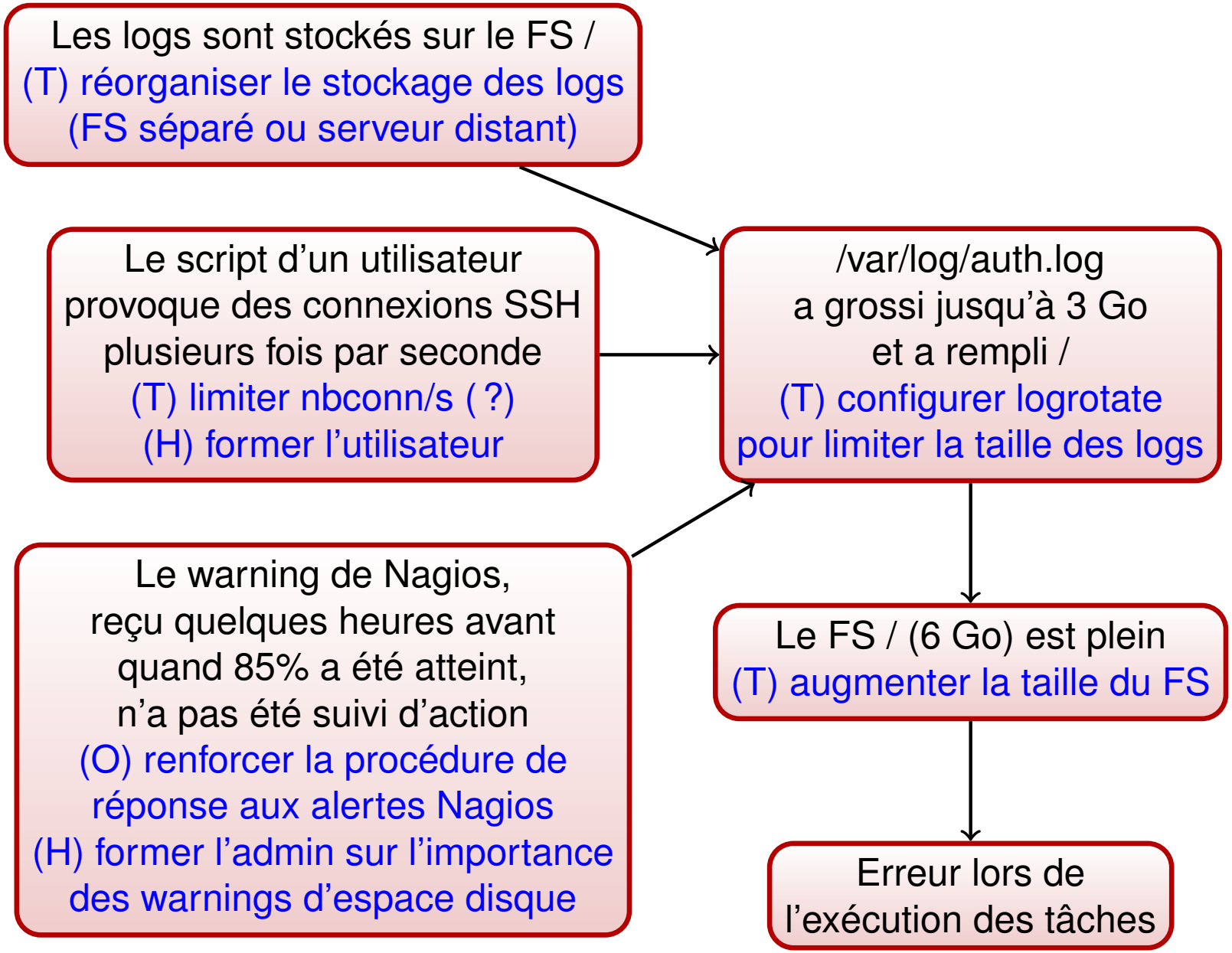
L’arbre des causes : exemple



L'arbre des causes : exemple



L'arbre des causes : pistes d'action



Exercices (1/2)

- Observez vos logs (dans `/var/log`). Trouvez un message d'erreur et essayez de comprendre sa cause.
- Avec la commande `dmesg`, observez les messages affichés par le noyau. Examinez en particulier la séquence de boot. Trouvez un message d'erreur et essayez de comprendre sa cause.
- Installez un serveur SSH sur votre machine. Activez le mode *debug*, et connectez-vous en SSH au serveur.
- Avec `ethtool`, regardez à quel débit votre carte réseau a négocié sa connexion.

Exercices (2/2)

- En utilisant `strace` sur `ls` puis `ls -l`, cherchez (et vérifiez avec la page de manuel correspondante) :
  - l'appel système utilisé pour récupérer l'ensemble des entrées d'un répertoire
  - l'appel système utilisé pour récupérer les attributs d'un fichier
- Où `ps` récupère-t-il les informations qu'il affiche ?
- Avec un autre étudiant, lancez `wireshark` à distance comme indiqué dans la présentation, et observez le trafic. Voyez-vous le trafic de l'ensemble des postes de la salle ? Pourquoi ?
- Essayez les autres outils mentionnés dans la présentation.