Optimisations Système et Noyau

Aurélien Cedeyn

École Nationale Supérieure d'Informatique pour l'Industrie et l'Entreprise

2017-2018

Sommaire

Définitions

Définition

To make as perfect, effective, or functional as possible.

- Plusieurs pistes
  - Matérielle : déléguer au matériel des fonctions logicielles.
  - Logicielle : modifier le code d'une application pour qu'elle soit plus performante.
  - Système : configurer ou modifier le système d'exploitation pour qu'il réponde mieux aux applications pour lesquelles il est dédié.
- Comment procéder ?
  - Connaitre le matériel
  - Analyser : utiliser des outils adaptés pour comprendre une situation et connaitre les points faibles d'une application ou d'un système.

Définitions

Définition

To make as perfect, effective, or functional as possible.

- Plusieurs pistes
  - Matérielle : déléguer au matériel des fonctions logicielles.
  - Logicielle : modifier le code d'une application pour qu'elle soit plus performante.
  - Système : configurer ou modifier le système d'exploitation pour qu'il réponde mieux aux applications pour lesquelles il est dédié.
- Comment procéder ?
  - Connaitre le matériel
  - Analyser : utiliser des outils adaptés pour comprendre une situation et connaitre les points faibles d'une application ou d'un système.

L'optimisation peut être différente en fonction du but qui est recherché.

L'optimisation peut être différente en fonction du but qui est recherché.

- Développeur

L'optimisation peut être différente en fonction du but qui est recherché.

- Développeur
  - Rendre le code le plus maintenable possible.

L'optimisation peut être différente en fonction du but qui est recherché.

- Développeur
  - Rendre le code le plus maintenable possible.
  - Factorisation du code.

L'optimisation peut être différente en fonction du but qui est recherché.

- Développeur
  - Rendre le code le plus maintenable possible.
  - Factorisation du code.
  - Travail sur l'algorithme.

L'optimisation peut être différente en fonction du but qui est recherché.

- Développeur
  - Rendre le code le plus maintenable possible.
  - Factorisation du code.
  - Travail sur l'algorithme.
- Matériel

L'optimisation peut être différente en fonction du but qui est recherché.

- Développeur
  - Rendre le code le plus maintenable possible.
  - Factorisation du code.
  - Travail sur l'algorithme.
- Matériel
  - Contraintes physiques.

L'optimisation peut être différente en fonction du but qui est recherché.

- Développeur
  - Rendre le code le plus maintenable possible.
  - Factorisation du code.
  - Travail sur l'algorithme.
- Matériel
  - Contraintes physiques.
  - Cherche à diminuer la taille mémoire des éléments.

L'optimisation peut être différente en fonction du but qui est recherché.

- Développeur
  - Rendre le code le plus maintenable possible.
  - Factorisation du code.
  - Travail sur l'algorithme.
- Matériel
  - Contraintes physiques.
  - Cherche à diminuer la taille mémoire des éléments.
  - N'utiliser que les fonctions nécessaire.

L'optimisation peut être différente en fonction du but qui est recherché.

- Développeur
  - Rendre le code le plus maintenable possible.
  - Factorisation du code.
  - Travail sur l'algorithme.
- Matériel
  - Contraintes physiques.
  - Cherche à diminuer la taille mémoire des éléments.
  - N'utiliser que les fonctions nécessaire.
  - Fournir le service de la façon la plus sûre.

L'optimisation peut être différente en fonction du but qui est recherché.

- Développeur
  - Rendre le code le plus maintenable possible.
  - Factorisation du code.
  - Travail sur l'algorithme.
- Matériel
  - Contraintes physiques.
  - Cherche à diminuer la taille mémoire des éléments.
  - N'utiliser que les fonctions nécessaire.
  - Fournir le service de la façon la plus sûre.
- Système

L'optimisation peut être différente en fonction du but qui est recherché.

- Développeur
  - Rendre le code le plus maintenable possible.
  - Factorisation du code.
  - Travail sur l'algorithme.
- Matériel
  - Contraintes physiques.
  - Cherche à diminuer la taille mémoire des éléments.
  - N'utiliser que les fonctions nécessaire.
  - Fournir le service de la façon la plus sûre.
- Système
  - Avoir le plus de fonctionnalités possibles.

L'optimisation peut être différente en fonction du but qui est recherché.

- Développeur
  - Rendre le code le plus maintenable possible.
  - Factorisation du code.
  - Travail sur l'algorithme.
- Matériel
  - Contraintes physiques.
  - Cherche à diminuer la taille mémoire des éléments.
  - N'utiliser que les fonctions nécessaire.
  - Fournir le service de la façon la plus sûre.
- Système
  - Avoir le plus de fonctionnalités possibles.
  - Prendre en compte les contraintes matérielles.

L'optimisation peut être différente en fonction du but qui est recherché.

- Développeur
  - Rendre le code le plus maintenable possible.
  - Factorisation du code.
  - Travail sur l'algorithme.
- Matériel
  - Contraintes physiques.
  - Cherche à diminuer la taille mémoire des éléments.
  - N'utiliser que les fonctions nécessaire.
  - Fournir le service de la façon la plus sûre.
- Système
  - Avoir le plus de fonctionnalités possibles.
  - Prendre en compte les contraintes matérielles.
  - Faire en sorte que les codes de calcul soient le plus rapide possible.

L'optimisation peut être différente en fonction du but qui est recherché.

- Développeur
  - Rendre le code le plus maintenable possible.
  - Factorisation du code.
  - Travail sur l'algorithme.
- Matériel
  - Contraintes physiques.
  - Cherche à diminuer la taille mémoire des éléments.
  - N'utiliser que les fonctions nécessaire.
  - Fournir le service de la façon la plus sûre.
- Système
  - Avoir le plus de fonctionnalités possibles.
  - Prendre en compte les contraintes matérielles.
  - Faire en sorte que les codes de calcul soient le plus rapide possible.

Ces différents buts doivent converger.

Sommaire

Tous les codes ont un historique

Un code développé à un instant t doit inévitablement évoluer.

- Un code est conçu initialement sur un système d'exploitation avec une configuration donnée.
- Les processeurs évoluent et offrent de nouvelles instructions que les codes peuvent utiliser pour améliorer leurs performances.
- L'historique d'une application joue un rôle très important vis-à-vis des performances qu'elle peut obtenir.
- Comprendre le comportement d'un code devient primordial.

Le cas classique

Prenons un exemple :

- Ouverture d'un fichier de données.

Le cas classique

Prenons un exemple :

- Ouverture d'un fichier de données.
- Lecture du contenu.

Le cas classique

Prenons un exemple :

- Ouverture d'un fichier de données.
- Lecture du contenu.
- Fermeture du fichier.

Le cas classique

Prenons un exemple :

- Ouverture d'un fichier de données.
- Lecture du contenu.
- Fermeture du fichier.
- Traitement des données.

Le cas classique

Prenons un exemple :

- Ouverture d'un fichier de données.
- Lecture du contenu.
- Fermeture du fichier.
- Traitement des données.

Cet exemple simple fonctionne parfaitement.

Le cas classique

Prenons un exemple :

- Ouverture d'un fichier de données.
- Lecture du contenu.
- Fermeture du fichier.
- Traitement des données.

Cet exemple simple fonctionne parfaitement. Ajoutons quelques contraintes :

Le cas classique

Prenons un exemple :

- Ouverture d'un fichier de données.
- Lecture du contenu.
- Fermeture du fichier.
- Traitement des données.

Cet exemple simple fonctionne parfaitement. Ajoutons quelques contraintes :

- Le fichier est sur un système de fichier partagé.

Le cas classique

Prenons un exemple :

- Ouverture d'un fichier de données.
- Lecture du contenu.
- Fermeture du fichier.
- Traitement des données.
- Partage du résultat avec ses camarades.

Cet exemple simple fonctionne parfaitement. Ajoutons quelques contraintes :

- Le fichier est sur un système de fichier partagé.
- 1000 machines vont lancer ce code.

Le cas classique

Prenons un exemple :

- Ouverture d'un fichier de données.
- Lecture du contenu.
- Fermeture du fichier.
- Traitement des données.
- Partage du résultat avec ses camarades.

Cet exemple simple fonctionne parfaitement. Ajoutons quelques contraintes :

- Le fichier est sur un système de fichier partagé.
- 1000 machines vont lancer ce code.

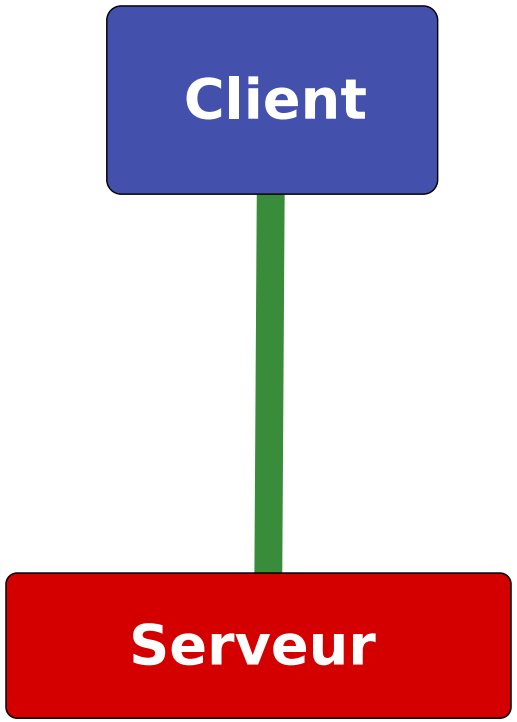
Tout s'écroule... Quelles "optimisations" peut-on faire ?

Les optimisations

- Avoir des serveurs très robustes pour absorber la charge induite par ces machines.

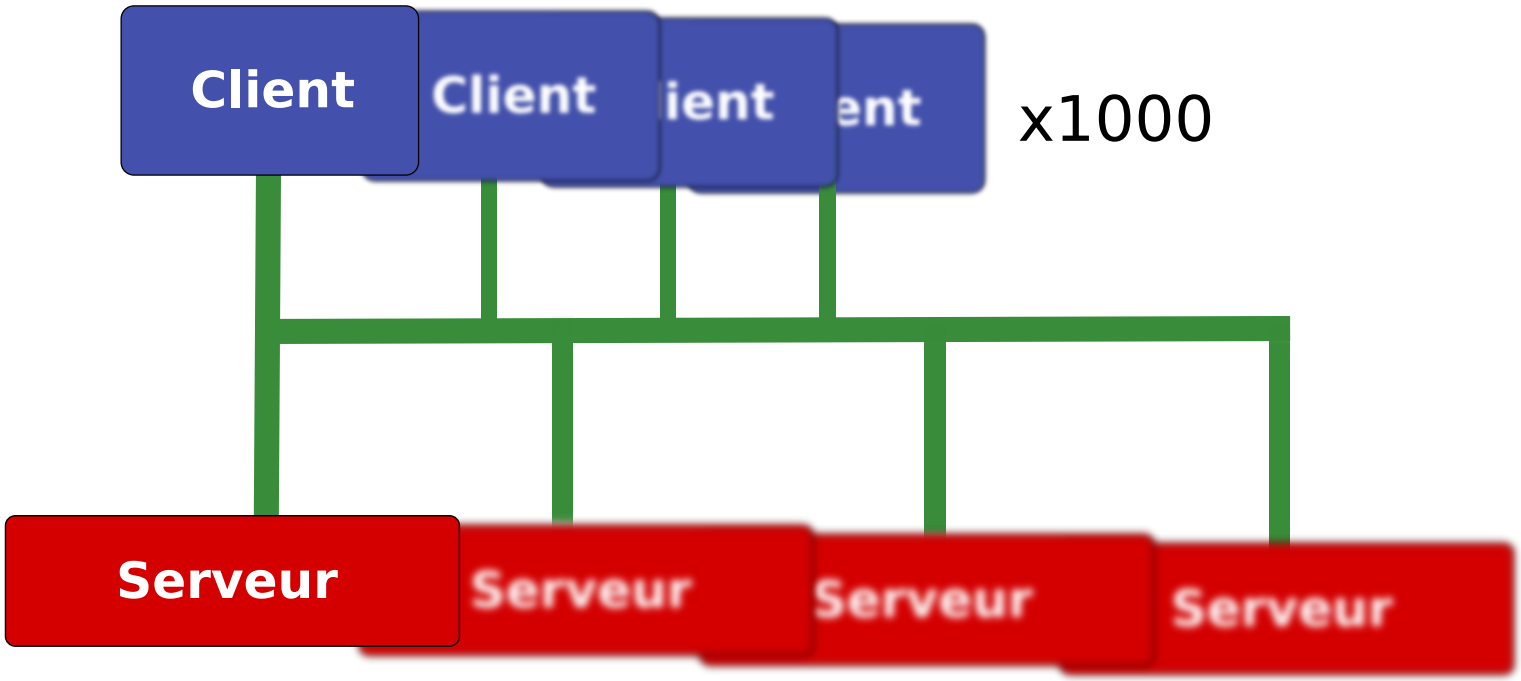
Les optimisations

- Avoir des serveurs très robustes pour absorber la charge induite par ces machines.



Les optimisations

- Avoir des serveurs très robustes pour absorber la charge induite par ces machines.



Les optimisations

- Avoir des serveurs très robustes pour absorber la charge induite par ces machines.
- Modifier le code pour que les données ne soient lues qu'une seule fois et partagées entre les différentes machines.

Le code de la première machine

Le code des autres machines

Les optimisations

- Avoir des serveurs très robustes pour absorber la charge induite par ces machines.
- Modifier le code pour que les données ne soient lues qu'une seule fois et partagées entre les différentes machines.

Le code de la première machine

- Ouverture d'un fichier de données.
- Lecture du contenu.
- Fermeture du fichier.
- Partage des données avec ses camarades.
- Traitement des données.
- Partage du résultat.

Le code des autres machines

Les optimisations

- Avoir des serveurs très robustes pour absorber la charge induite par ces machines.
- Modifier le code pour que les données ne soient lues qu'une seule fois et partagées entre les différentes machines.

Le code de la première machine

- Ouverture d'un fichier de données.
- Lecture du contenu.
- Fermeture du fichier.
- Partage des données avec ses camarades.
- Traitement des données.
- Partage du résultat.

Le code des autres machines

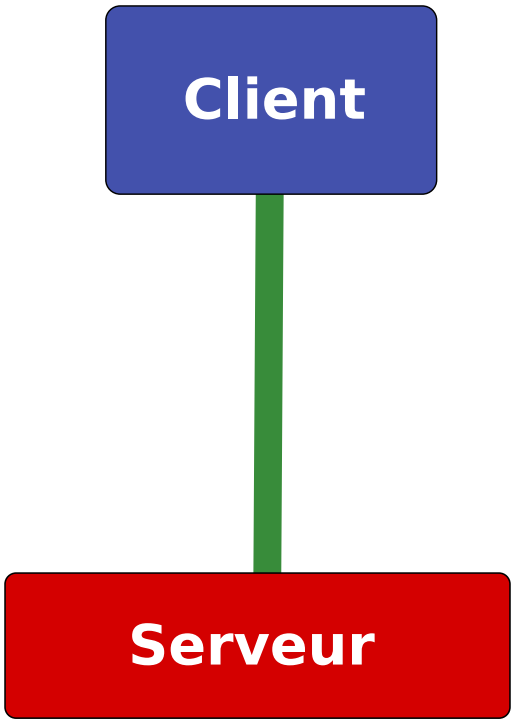
- Attente des données.
- Traitement des données.
- Partage du résultat.

Les optimisations

- Avoir des serveurs très robustes pour absorber la charge induite par ces machines.
- Modifier le code pour que les données ne soient lues qu'une seule fois et partagées entre les différentes machines.
- Modifier le système pour faire en sorte que le code n'ai pas besoin d'être modifié.

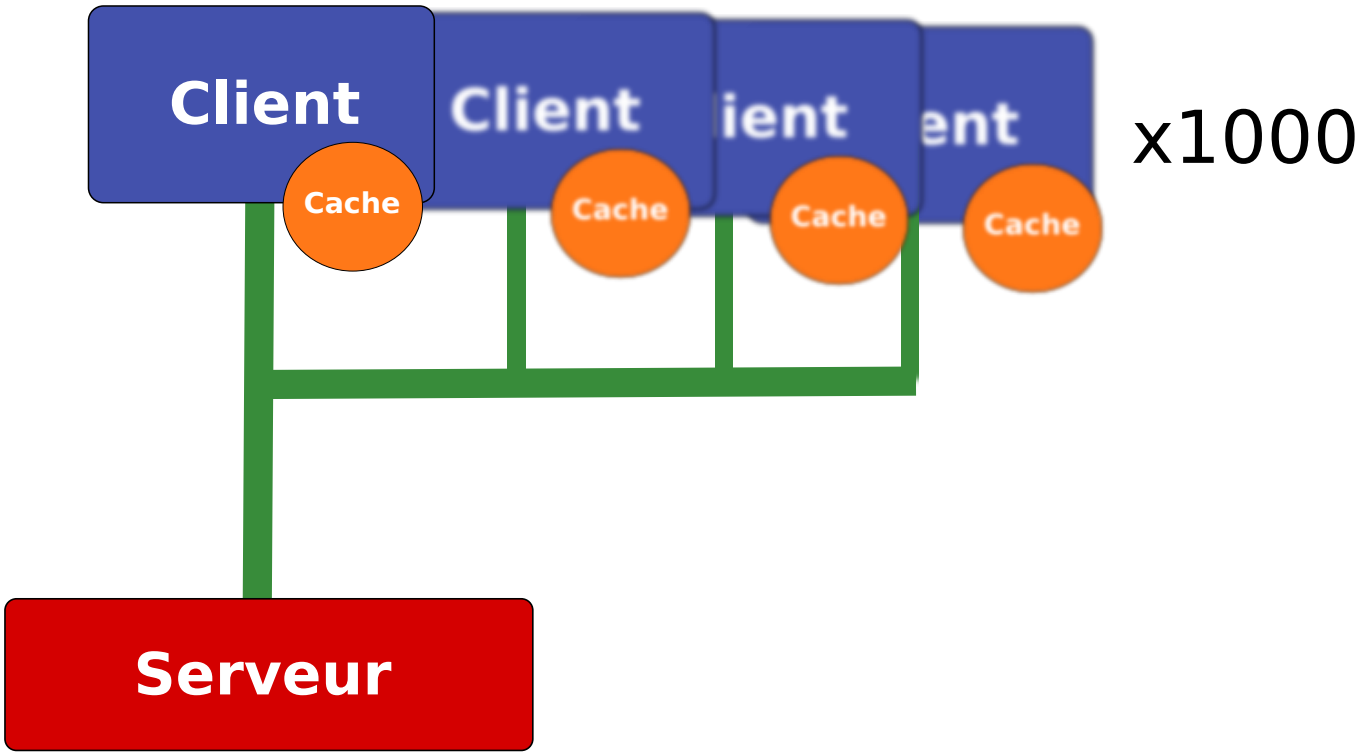
Les optimisations

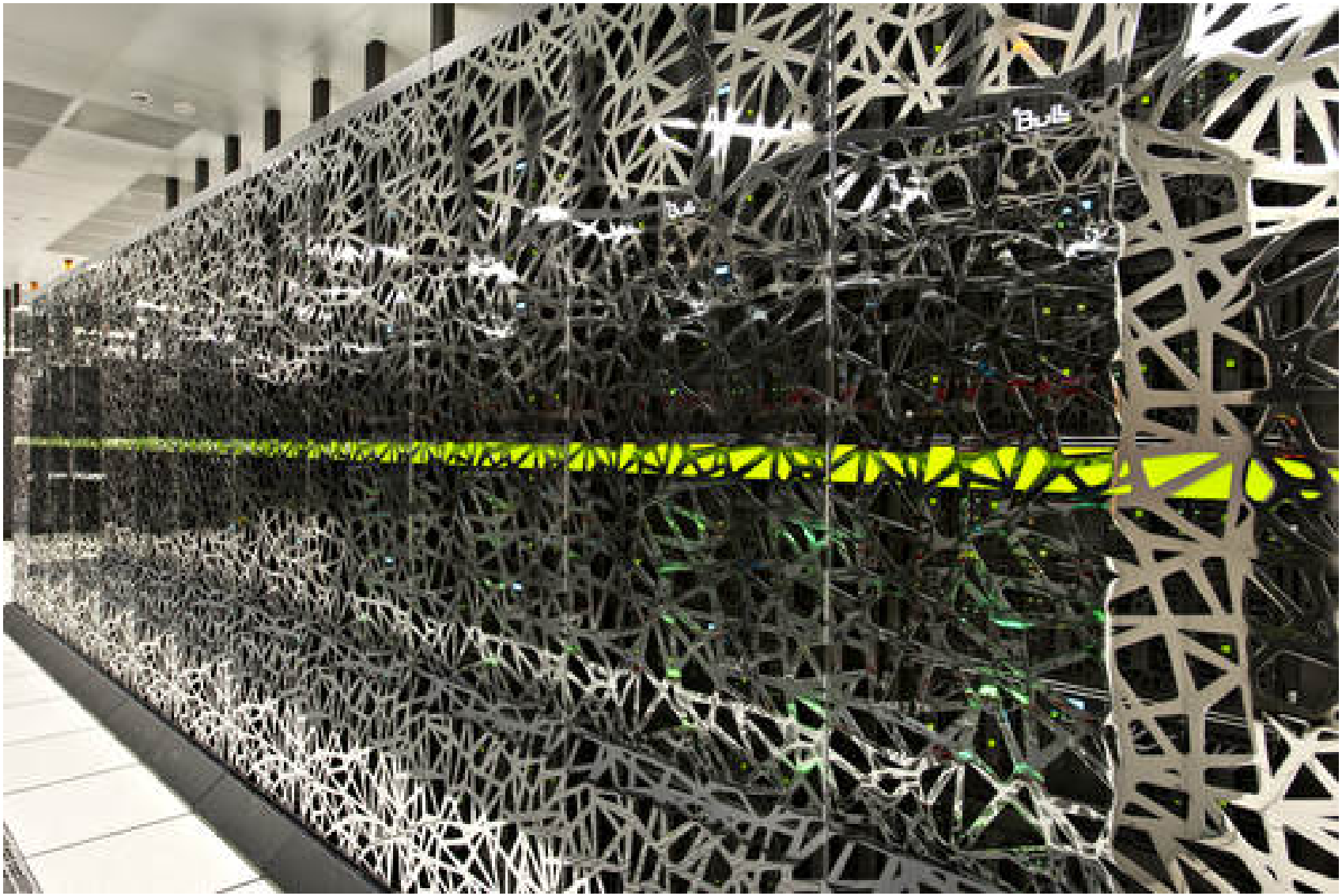
- Avoir des serveurs très robustes pour absorber la charge induite par ces machines.
- Modifier le code pour que les données ne soient lues qu'une seule fois et partagées entre les différentes machines.
- Modifier le système pour faire en sorte que le code n'ai pas besoin d'être modifié.



Les optimisations

- Avoir des serveurs très robustes pour absorber la charge induite par ces machines.
- Modifier le code pour que les données ne soient lues qu'une seule fois et partagées entre les différentes machines.
- Modifier le système pour faire en sorte que le code n'ai pas besoin d'être modifié.




Bull

Le matériel est de plus en plus performant

- GPU

Le matériel est de plus en plus performant

- GPU
  - Utilisation de la puissance du processeur graphique pour effectuer du calcul.

Le matériel est de plus en plus performant

- GPU
  - Utilisation de la puissance du processeur graphique pour effectuer du calcul.
  - Adapté au calcul matriciel.

Le matériel est de plus en plus performant

- GPU
  - Utilisation de la puissance du processeur graphique pour effectuer du calcul.
  - Adapté au calcul matriciel.
  - Consommation énergétique réduite.

Le matériel est de plus en plus performant

- GPU
  - Utilisation de la puissance du processeur graphique pour effectuer du calcul.
  - Adapté au calcul matriciel.
  - Consommation énergétique réduite.
- Réseau

Le matériel est de plus en plus performant

- GPU
  - Utilisation de la puissance du processeur graphique pour effectuer du calcul.
  - Adapté au calcul matriciel.
  - Consommation énergétique réduite.
- Réseau
  - RDMA : Remote Direct Memory Access

Le matériel est de plus en plus performant

- GPU
  - Utilisation de la puissance du processeur graphique pour effectuer du calcul.
  - Adapté au calcul matriciel.
  - Consommation énergétique réduite.
- Réseau
  - RDMA : Remote Direct Memory Access
  - Fonctionnalités logicielles intégrées au matériel.

Le matériel est de plus en plus performant

- GPU
  - Utilisation de la puissance du processeur graphique pour effectuer du calcul.
  - Adapté au calcul matriciel.
  - Consommation énergétique réduite.
- Réseau
  - RDMA : Remote Direct Memory Access
  - Fonctionnalités logicielles intégrées au matériel.
- CPU

Le matériel est de plus en plus performant

- GPU
  - Utilisation de la puissance du processeur graphique pour effectuer du calcul.
  - Adapté au calcul matriciel.
  - Consommation énergétique réduite.
- Réseau
  - RDMA : Remote Direct Memory Access
  - Fonctionnalités logicielles intégrées au matériel.
- CPU
  - Nouvelles instruction (vectorisation...)

Le matériel est de plus en plus performant

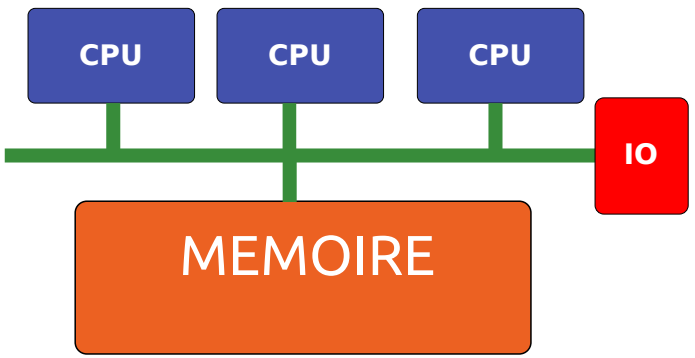
- GPU
  - Utilisation de la puissance du processeur graphique pour effectuer du calcul.
  - Adapté au calcul matriciel.
  - Consommation énergétique réduite.
- Réseau
  - RDMA : Remote Direct Memory Access
  - Fonctionnalités logicielles intégrées au matériel.
- CPU
  - Nouvelles instruction (vectorisation...)
  - Augmentation du nombre de core

Les architectures évoluent faisant apparaître de nouveaux problèmes

Uniform Memory Access

Au commencement des architectures multi-processeurs, l'ensemble des unités de calcul avaient un accès uniforme à la mémoire et au matériel

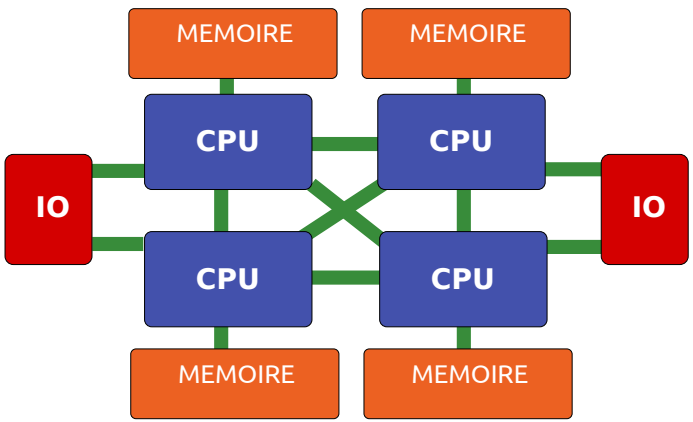
- SMP (Symetric Multi-Processor)
- Problème de passage à l'échelle de cette architecture
- Plus le nombre de processeur augmente, plus le bus est saturé et ne peut plus répondre dans des temps raisonnables.



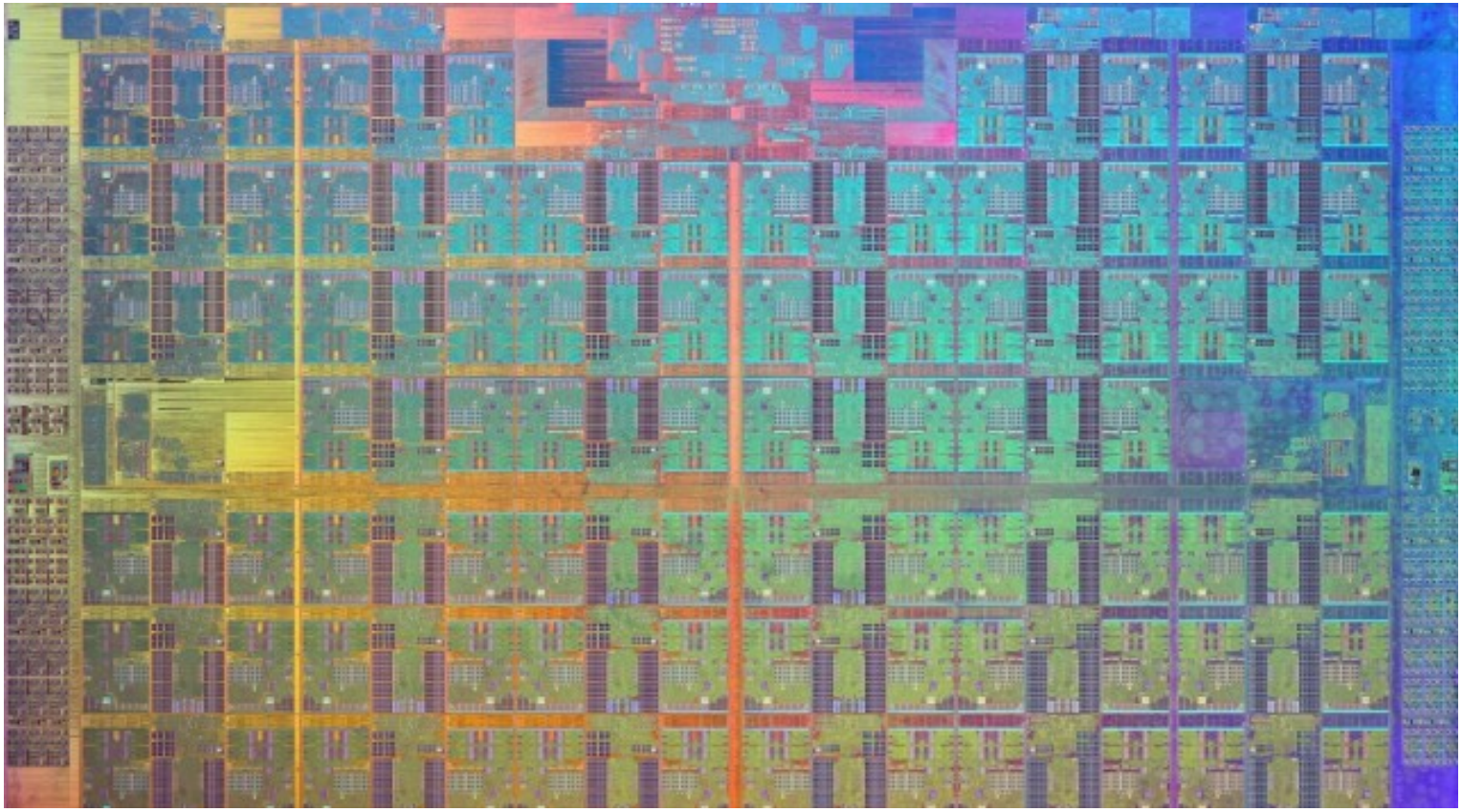
Non Uniform Memory Access

Pour répondre à ce problème, l'architecture NUMA a été créée.

- Chaque processeur accède à sa propre mémoire.
- Pour accéder à d'autre zones mémoires, il sollicite son voisin.
- La contrainte de cette architecture : l'accès à la mémoire et au matériel n'est plus uniforme.
- Nécessité de gérer finement la localisation de chaque élément du système
  - Problème des interruptions.
  - Problème d'optimisation des accès à la mémoire.
  - Problème d'optimisation des IOs.

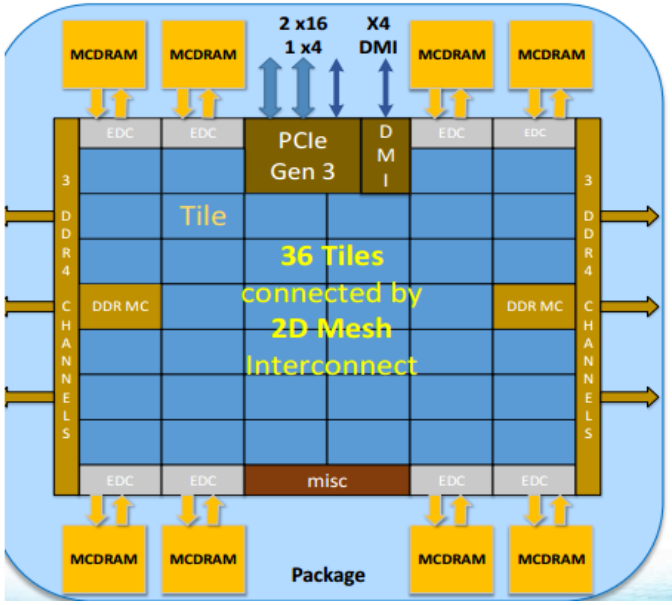


L'architecture KNight Landing



L'architecture KNight Landing

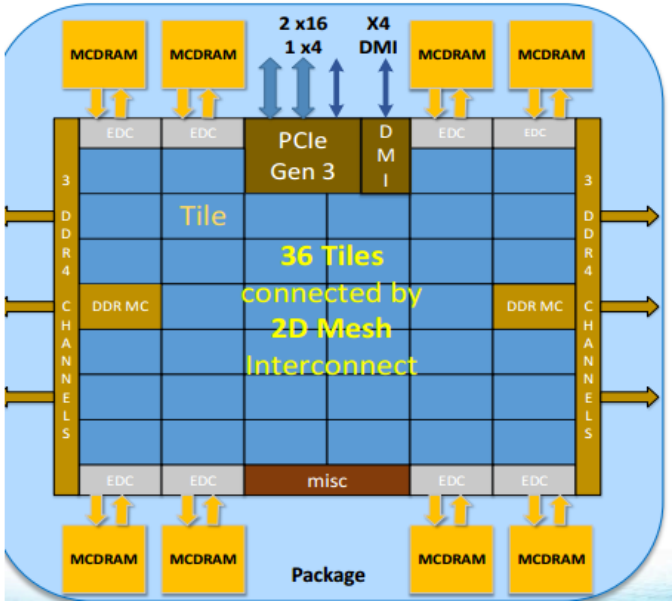
Architecture CPU particulière décuplant la problématique du nombre de CPU :



L'architecture KNight Landing

Architecture CPU particulière décuplant la problématique du nombre de CPU :

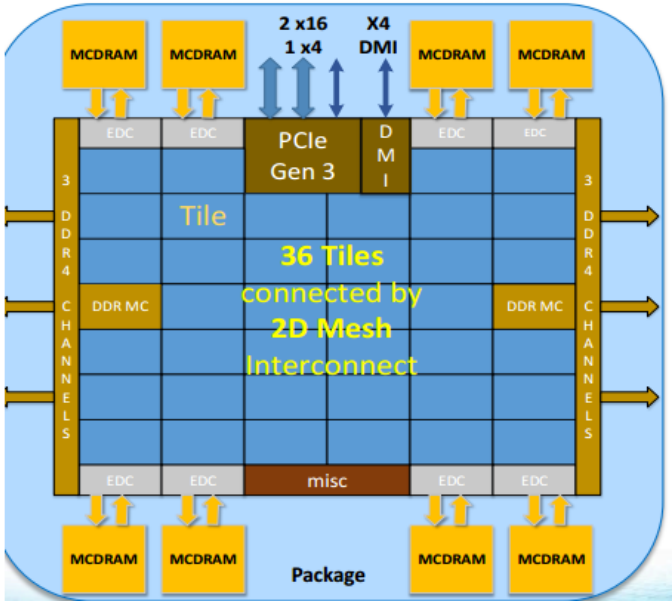
- Nouvelles instructions de vectorisation.



L'architecture KNight Landing

Architecture CPU particulière décuplant la problématique du nombre de CPU :

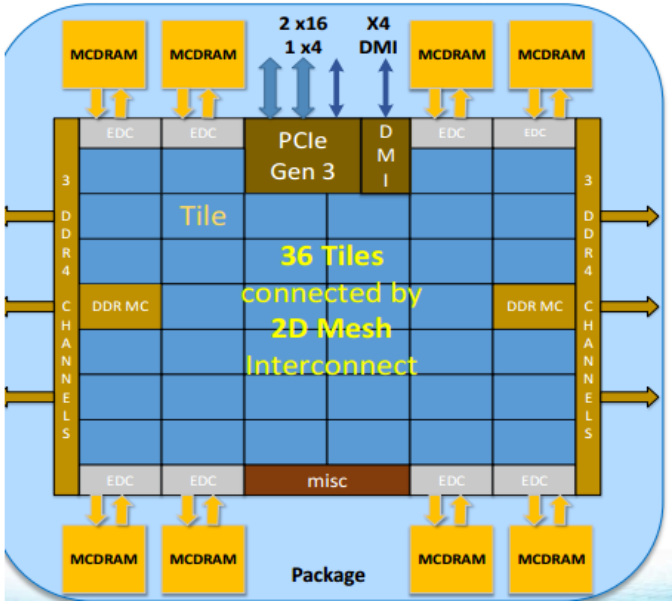
- Nouvelles instructions de vectorisation.
- Compatibilité avec l'architecture x86_64.



L'architecture KNight Landing

Architecture CPU particulière décuplant la problématique du nombre de CPU :

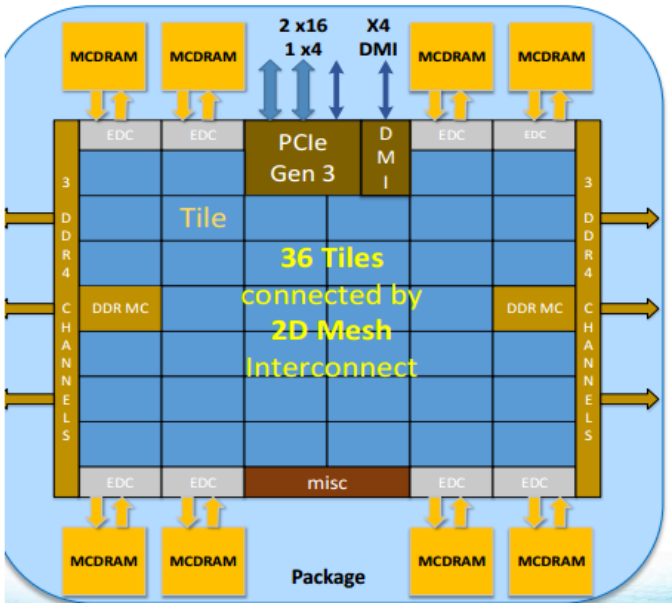
- Nouvelles instructions de vectorisation.
- Compatibilité avec l'architecture x86_64.
- Introduction d'un nouveau type de mémoire MCDRAM en plus de la DDR 4.



L'architecture KNight Landing

Architecture CPU particulière décuplant la problématique du nombre de CPU :

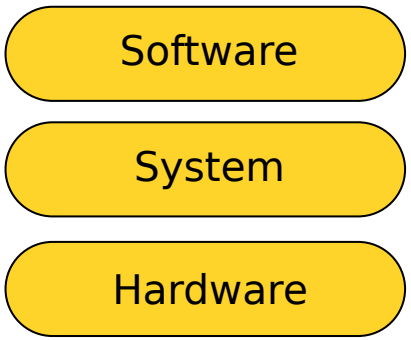
- Nouvelles instructions de vectorisation.
- Compatibilité avec l'architecture x86_64.
- Introduction d'un nouveau type de mémoire MCDRAM en plus de la DDR 4.
- 36 x 2 x 2 (144 CPUs) ! !



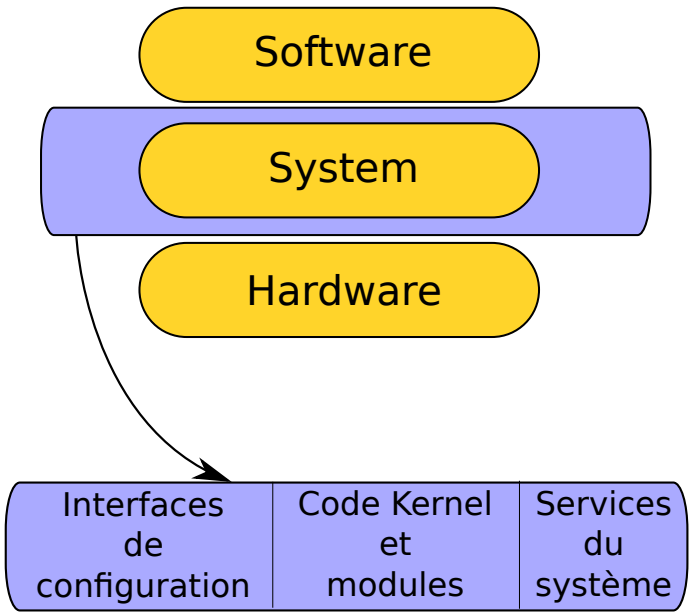
Sommaire

- Le système d'exploitation se trouve au milieu de la couche logiciel et du matériel.

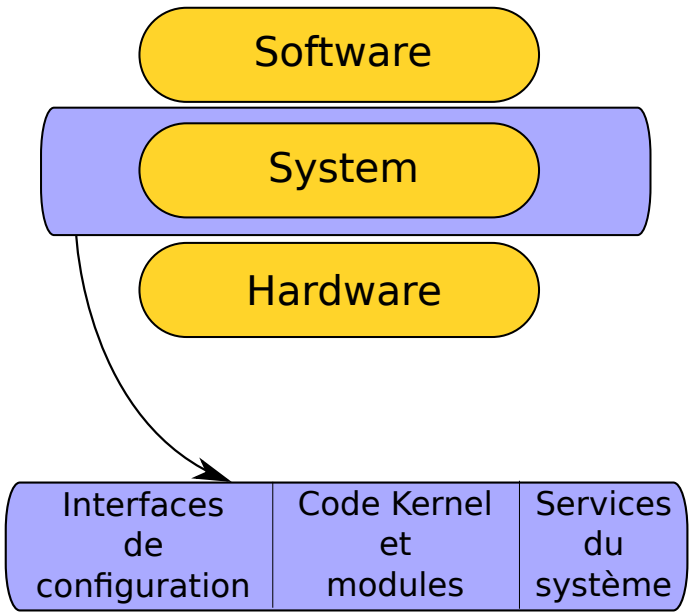
- Le système d'exploitation se trouve au milieu de la couche logiciel et du matériel.



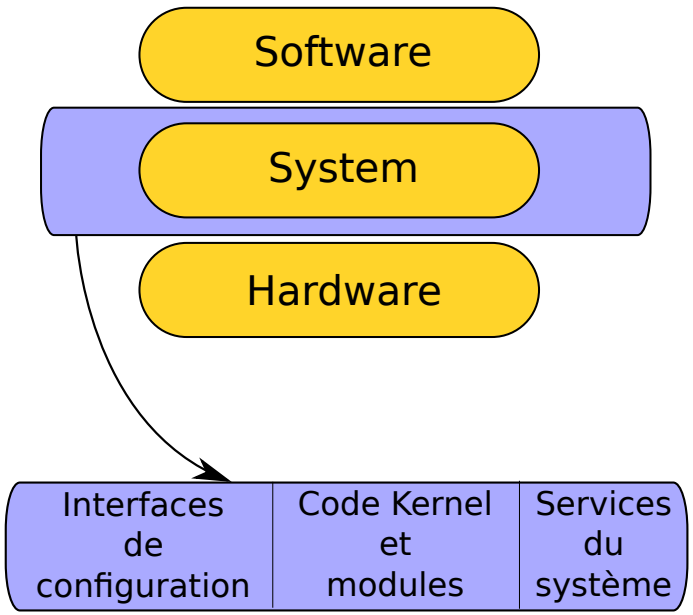
- Le système d'exploitation se trouve au milieu de la couche logiciel et du matériel.
- À ce niveau, plusieurs types d'optimisations sont possibles



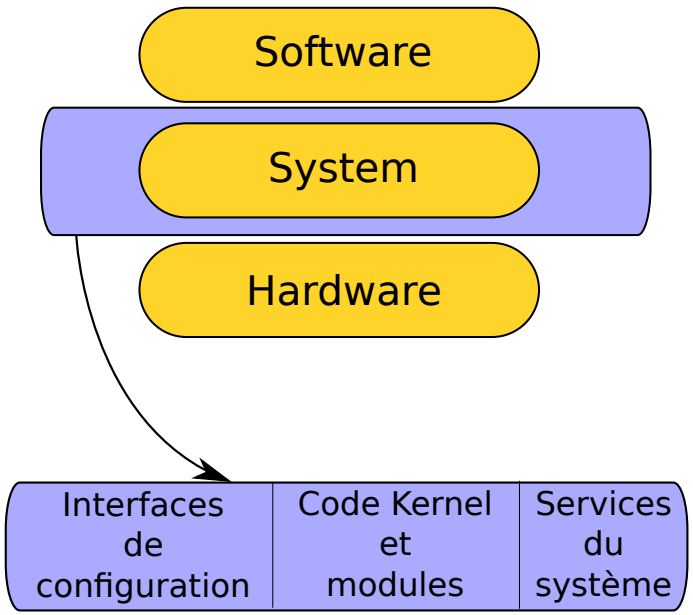
- Le système d'exploitation se trouve au milieu de la couche logiciel et du matériel.
- À ce niveau, plusieurs types d'optimisations sont possibles
  - Jouer sur la configuration du système (via les interfaces de configuration).



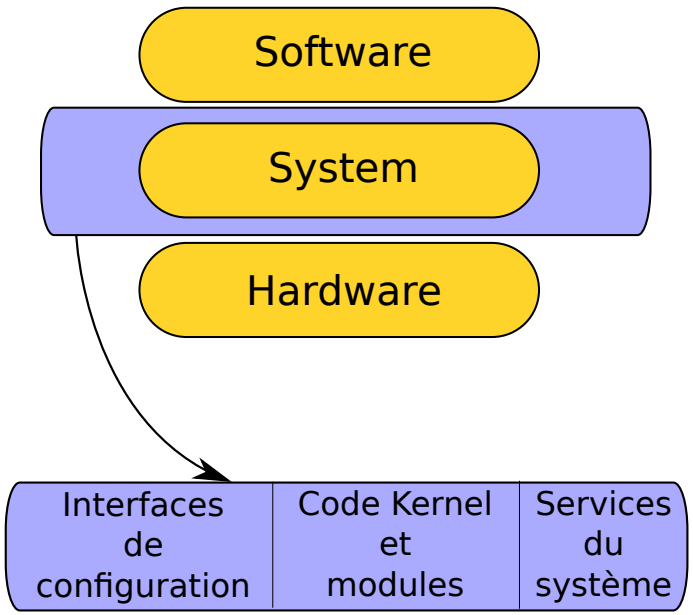
- Le système d'exploitation se trouve au milieu de la couche logiciel et du matériel.
- À ce niveau, plusieurs types d'optimisations sont possibles
  - Jouer sur la configuration du système (via les interfaces de configuration).
  - Utiliser des services ou des outils adaptés permettant de améliorer le comportement des codes.



- Le système d'exploitation se trouve au milieu de la couche logiciel et du matériel.
- À ce niveau, plusieurs types d'optimisations sont possibles
  - Jouer sur la configuration du système (via les interfaces de configuration).
  - Utiliser des services ou des outils adaptés permettant de améliorer le comportement des codes.
  - Agir sur le code du noyau.



- Le système d'exploitation se trouve au milieu de la couche logiciel et du matériel.
- À ce niveau, plusieurs types d'optimisations sont possibles
  - Jouer sur la configuration du système (via les interfaces de configuration).
  - Utiliser des services ou des outils adaptés permettant de améliorer le comportement des codes.
  - Agir sur le code du noyau.
- Au niveau système, il est possible d'analyser le comportement des différents composants et d'en déduire des modifications de configuration.



Les différentes interfaces

Les configurations par défaut d'un système correspondent à l'utilisation la plus commune qui en est faite.
Chaque distribution choisit, de façon plus ou moins arbitraire, les paramètres par défaut qui cibleront le plus de cas d'utilisation.
Chaque logiciel installé arrive avec une configuration suggérée par les packageurs de la distribution mais ne correspondent pas forcément au cas optimal.

Les différentes interfaces

Les configurations par défaut d'un système correspondent à l'utilisation la plus commune qui en est faite.
Chaque distribution choisit, de façon plus ou moins arbitraire, les paramètres par défaut qui cibleront le plus de cas d'utilisation.
Chaque logiciel installé arrive avec une configuration suggérée par les packageurs de la distribution mais ne correspondent pas forcément au cas optimal.
Au niveau purement système, deux interfaces de configuration sont particulièrement intéressantes :

- Les limites de protection du système : *ulimit*
- Les configurations par défaut des différents sous-systèmes du noyau : *sysctl*

ulimit

Dans les distributions, une configuration des limites pour chaque processus est définie.
Ces limites sont consultables via la commande *ulimit* ou directement via */proc/<PID>/limits*.

ulimit

```
$ ulimit -a
core file size          (blocks, -c) unlimited
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
file size               (blocks, -f) unlimited
pending signals                 (-i) 31440
max locked memory       (kbytes, -l) unlimited
max memory size         (kbytes, -m) unlimited
open files                      (-n) 1024
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
real-time priority              (-r) 99
stack size              (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
max user processes              (-u) 31440
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
```

ulimit

La configuration est stockée dans le fichier */etc/scurity/limits.conf* et est activée via les *Plugabble Autentication Modules (PAM)*.

limits.conf

```
# /etc/security/limits.conf
#
#Each line describes a limit for a user in the form:
#
#<domain>        <type>  <item>  <value>
#
#Where:
#<domain> can be:
#        - a user name
#        - a group name, with @group syntax
#        - the wildcard *, for default entry
#        - the wildcard %, can be also used with %group syntax,
#                 for maxlogin limit
#
#<type> can have the two values:
#        - "soft" for enforcing the soft limits
#        - "hard" for enforcing hard limits

#*               soft    core            0
#*               hard    rss             10000
#@student        hard    nproc           20
#@faculty        soft    nproc           20
#@faculty        hard    nproc           50
#ftp             hard    nproc           0
#@student        -       maxlogins       4
```

/proc/sys

Ce répertoire, dans le pseudo système de fichiers *procfs*, contient la configuration de nombreux composants du noyau.

/proc/sys

```
$ ls /proc/sys
abi  debug  dev  fs  kernel  net  user        vm
```

- La VFS (fs)
- Le réseau (net)
- Le noyau (kernel)
- Le système (vm)
- Les périphériques (device)
- Les espaces de nom, ou namespaces (user)
- La compatibilité (abi)
- Les fonctions de debugging (debug)

sysctl

- L'outil *sysctl* permet de gérer l'ensemble des paramètres de */proc/sys*

sysctl

- L'outil *sysctl* permet de gérer l'ensemble des paramètres de */proc/sys*
- *sysctl -a* liste les paramètres disponibles

sysctl -a

```
vm.admin_reserve_kbytes = 8192
vm.block_dump = 0
vm.compact_unevictable_allowed = 1
vm.dirty_background_bytes = 0
vm.dirty_background_ratio = 10
vm.dirty_bytes = 0
vm.dirty_expire_centisecs = 3000
vm.dirty_ratio = 20
vm.dirty_writeback_centisecs = 500
vm.dirtytime_expire_seconds = 43200
vm.drop_caches = 0
vm.extfrag_threshold = 500
vm.hugepages_treat_as_movable = 0
vm.hugetlb_shm_group = 0
vm.laptop_mode = 0
vm.legacy_va_layout = 0
vm.lowmem_reserve_ratio = 256   256   32   1
vm.max_map_count = 65530
vm.memory_failure_early_kill = 0
vm.memory_failure_recovery = 1
vm.min_free_kbytes = 67584
vm.min_slab_ratio = 5
```

sysctl

- L'outil *sysctl* permet de gérer l'ensemble des paramètres de */proc/sys*
- *sysctl -a* liste les paramètres disponibles
- *sysctl mon.param = valeur* change la valeur de */proc/sys/mon/param*

sysctl -a

```
vm.admin_reserve_kbytes = 8192
vm.block_dump = 0
vm.compact_unevictable_allowed = 1
vm.dirty_background_bytes = 0
vm.dirty_background_ratio = 10
vm.dirty_bytes = 0
vm.dirty_expire_centisecs = 3000
vm.dirty_ratio = 20
vm.dirty_writeback_centisecs = 500
vm.dirtytime_expire_seconds = 43200
vm.drop_caches = 0
vm.extfrag_threshold = 500
vm.hugepages_treat_as_movable = 0
vm.hugetlb_shm_group = 0
vm.laptop_mode = 0
vm.legacy_va_layout = 0
vm.lowmem_reserve_ratio = 256   256   32   1
vm.max_map_count = 65530
vm.memory_failure_early_kill = 0
vm.memory_failure_recovery = 1
vm.min_free_kbytes = 67584
vm.min_slab_ratio = 5
```

sysctl

- L'outil *sysctl* permet de gérer l'ensemble des paramètres de */proc/sys*
- *sysctl -a* liste les paramètres disponibles
- *sysctl mon.param = valeur* change la valeur de */proc/sys/mon/param*
- Configuration permanente dans */etc/sysctl.d*

sysctl -a

```
vm.admin_reserve_kbytes = 8192
vm.block_dump = 0
vm.compact_unevictable_allowed = 1
vm.dirty_background_bytes = 0
vm.dirty_background_ratio = 10
vm.dirty_bytes = 0
vm.dirty_expire_centisecs = 3000
vm.dirty_ratio = 20
vm.dirty_writeback_centisecs = 500
vm.dirtytime_expire_seconds = 43200
vm.drop_caches = 0
vm.extfrag_threshold = 500
vm.hugepages_treat_as_movable = 0
vm.hugetlb_shm_group = 0
vm.laptop_mode = 0
vm.legacy_va_layout = 0
vm.lowmem_reserve_ratio = 256   256   32   1
vm.max_map_count = 65530
vm.memory_failure_early_kill = 0
vm.memory_failure_recovery = 1
vm.min_free_kbytes = 67584
vm.min_slab_ratio = 5
```

Qu'est-ce qu'il vient faire là *debug* ?

Qu'est-ce qu'il vient faire là *debug* ?

- Le poids de l'histoire : *procfs*

Qu'est-ce qu'il vient faire là *debug* ?

- Le poids de l'histoire : *procfs*
- Les petits nouveaux *sysfs* et *debugfs*

La configuration du kernel

Chaque noyau a une configuration bien définie. Elle va définir les pilotes et les différentes fonctionnalités activées.

- Issue du *make [menu]config*.
- Consultable dans */proc/config.gz* (si le noyau a été compilé avec l'option *CONFIG_IKCONFIG_PROC*).
- Consultable avec le script *extract-ikconfig* (si noyau a été compilé avec l'option *CONFIG_IKCONFIG*).
- Les distributions fournissent généralement la configuration du noyau dans le package correspondant à son installation. Elle est consultable dans */boot/config-<kernel-version>*.

La configuration du kernel

configuration du noyau

```
#
# Automatically generated file; DO NOT EDIT.
# Linux/x86 4.13.12-1 Kernel Configuration
#
CONFIG_64BIT=y
CONFIG_X86_64=y
CONFIG_X86=y
CONFIG_INSTRUCTION_DECODER=y
CONFIG_OUTPUT_FORMAT="elf64-x86-64"
CONFIG_ARCH_DEFCONFIG="arch/x86/configs/x86_64_defconfig"
CONFIG_LOCKDEP_SUPPORT=y
CONFIG_STACKTRACE_SUPPORT=y
CONFIG_MMU=y
CONFIG_ARCH_MMAP_RND_BITS_MIN=28
CONFIG_ARCH_MMAP_RND_BITS_MAX=32
CONFIG_ARCH_MMAP_RND_COMPAT_BITS_MIN=8
CONFIG_ARCH_MMAP_RND_COMPAT_BITS_MAX=16
CONFIG_NEED_DMA_MAP_STATE=y
CONFIG_NEED_SG_DMA_LENGTH=y
CONFIG_GENERIC_ISA_DMA=y
CONFIG_GENERIC_BUG=y
CONFIG_GENERIC_BUG_RELATIVE_POINTERS=y
CONFIG_GENERIC_HWEIGHT=y
CONFIG_ARCH_MAY_HAVE_PC_FDC=y
CONFIG_RWSEM_XCHGADD_ALGORITHM=y
CONFIG_GENERIC_CALIBRATE_DELAY=y
CONFIG_ARCH_HAS_CPU_RELAX=y
CONFIG_ARCH_HAS_CACHE_LINE_SIZE=y
CONFIG_HAVE_SETUP_PER_CPU_AREA=y
CONFIG_NEED_PER_CPU_EMBED_FIRST_CHUNK=y
CONFIG_NEED_PER_CPU_PAGE_FIRST_CHUNK=y
CONFIG_ARCH_HIBERNATION_POSSIBLE=y
CONFIG_ARCH_SUSPEND_POSSIBLE=y
CONFIG_ARCH_WANT_HUGE_PMD_SHARE=y
```

La ligne de commande du kernel

Certaines fonctions du noyau peuvent être configurées sur sa ligne de commande.

- Contient l'option *root=* qui définit le disque sur lequel booter.
- les paramètres commençant par *rd.* sont dédiés à la phase d'initialisation dans l'*initrd*.
- La documentation est disponible dans les sources du noyau : *Documentation/kernel-parameters.txt*

La ligne de commande du kernel

Certaines fonctions du noyau peuvent être configurées sur sa ligne de commande.

- Contient l'option *root=* qui définit le disque sur lequel booter.
- les paramètres commençant par *rd.* sont dédiés à la phase d'initialisation dans l'*initrd*.
- La documentation est disponible dans les sources du noyau : *Documentation/kernel-parameters.txt*

Ligne de commande du noyau

```
$ cat /proc/cmdline
BOOT_IMAGE=/vmlinuz-linux root=UUID=799dde59-2b33-4200-8260-1e7eedf6fa4e rw crashkernel=auto
```

Les modules kernel

- Code supplémentaire du noyau
- Intégrable via la commande *modprobe*
  - Basée du insmod <fichier.ko>
  - Gère automatiquement les dépendances
  - Gestion des paramètres de chaque module
    - /etc/modprobe.d/*.conf
    - Format :
      options nom_du_module paramètres
- Pour connaître les paramètres d'un module noyau :
  - modinfo <module>

modinfo

```
$ modinfo kvm
filename:       /lib/modules/4.13.12-1-ARCH/kernel/arch/x86/kvm/kvm.ko.gz
license:        GPL
author:         Qumranet
depends:        irqbypass
intree:         Y
name:           kvm
vermagic:       4.13.12-1-ARCH SMP preempt mod_unload modversions
parm:           ignore_msrs:bool
parm:           min_timer_period_us:uint
parm:           kvmclock_periodic_sync:bool
parm:           tsc_tolerance_ppm:uint
parm:           lapic_timer_advance_ns:uint
parm:           vector_hashing:bool
parm:           halt_poll_ns:uint
parm:           halt_poll_ns_grow:uint
parm:           halt_poll_ns_shrink:uint
```

Les modules kernel

- Une fois qu'un module est chargé, le kernel crée une arborescence dans */sys/module/<module>*.

/sys/module/kvm

```
$ find /sys/module/kvm -type d
/sys/module/kvm/
/sys/module/kvm/notes
/sys/module/kvm/sections
/sys/module/kvm/parameters
/sys/module/kvm/holders
```

Les modules kernel

- Une fois qu'un module est chargé, le kernel crée une arborescence dans */sys/module/<module>*.
- Le répertoire *parameters* contient l'ensemble des paramètres du module.

/sys/module/kvm

```
$ find /sys/module/kvm -type d
/sys/module/kvm/
/sys/module/kvm/notes
/sys/module/kvm/sections
/sys/module/kvm/parameters
/sys/module/kvm/holders
```

Les modules kernel

- Une fois qu'un module est chargé, le kernel crée une arborescence dans */sys/module/<module>*.
- Le répertoire *parameters* contient l'ensemble des paramètres du module.
- Certains de ces paramètres sont modifiables à chaud, d'autre ne le sont qu'au chargement.

/sys/module/kvm

```
$ find /sys/module/kvm -type d
/sys/module/kvm/
/sys/module/kvm/notes
/sys/module/kvm/sections
/sys/module/kvm/parameters
/sys/module/kvm/holders
```

Les modules kernel

- Une fois qu'un module est chargé, le kernel crée une arborescence dans */sys/module/<module>*.
- Le répertoire *parameters* contient l'ensemble des paramètres du module.
- Certains de ces paramètres sont modifiables à chaud, d'autre ne le sont qu'au chargement.
- Pour connaître les modules chargés :

lsmod

```
$ lsmod
Module                  Size  Used by
ppp_deflate            16384  0
bsd_comp               16384  0
ppp_async              20480  1
ppp_generic            32768  7 ppp_async,bsd_comp,ppp_deflate
slhc                   20480  1 ppp_generic
ccm                    20480  3
fuse                   94208  3
arc4                   16384  2
joydev                 20480  0
mousedev               20480  0
ath9k_htc              65536  0
uvcvideo               86016  0
ath9k_common           32768  1 ath9k_htc
ath9k_hw              438272  2 ath9k_htc,ath9k_common
ath                    28672  3 ath9k_htc,ath9k_hw,ath9k_common
mac80211              688128  1 ath9k_htc
```


www.longpaws.co.uk/tickless-pet
TICKLESS
ULTRASONIC TICK AND FLEA REPELLER
FROM DAY 1
UP TO 12 MONTH PROTECTION
CHEMICAL FREE
PREVENTIVE SOLUTION
40KHZ
TICKLESS

Tickless

Historiquement, le noyau réveillait tous les processeurs régulièrement pour leur faire effectuer des tâches de nettoyage.
Par la suite, seul les processeurs, qui n'étaient pas idle, se voyaient recevoir cette notification (*tick*).

Depuis le kernel 3.10, une nouvelle fonctionnalité du noyau permet de ne par réveiller intempestivement les CPUs, même ceux qui ne sont pas dans l'état idle.

Tickless, pourquoi faire ?

- Les CPUs *idle*, qui étaient réveillés, ne pouvaient pas se mettre en économie d'énergie.

Tickless, pourquoi faire ?

- Les CPUs *idle*, qui étaient réveillés, ne pouvaient pas se mettre en économie d'énergie.
- Recevoir un *tick* sollicite les CPUs.

Tickless, pourquoi faire ?

- Les CPUs *idle*, qui étaient réveillés, ne pouvaient pas se mettre en économie d'énergie.
- Recevoir un *tick* sollicite les CPUs.
- Le Tickless permet de répondre à ces problèmes.

Dans le cadre du HPC, cette configuration est très intéressante, surtout avec un nombre de processeurs important.

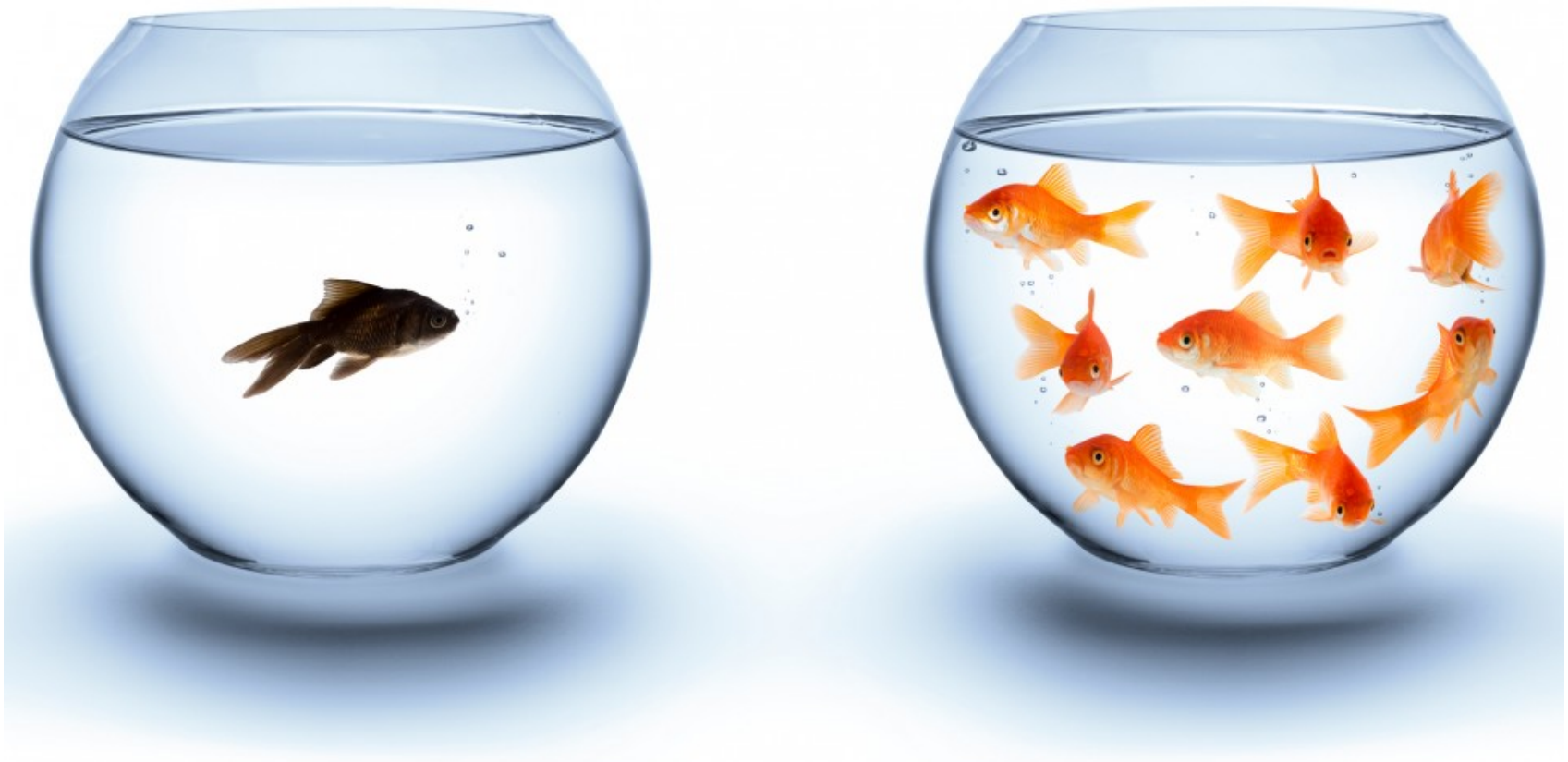
Tickless : mise en place

- Le noyau doit avoir été compilé avec l'option : *CONFIG_NO_HZ_FULL*
- Sur la ligne de boot du noyau, activer l'option : *nohz_full* = [*cpus*]
- Ou avoir compilé le noyau avec l'option : *CONFIG_NO_HZ_FULL_ALL*

Tickless : mise en place

- Le noyau doit avoir été compilé avec l'option : *CONFIG_NO_HZ_FULL*
- Sur la ligne de boot du noyau, activer l'option : *nohz_full* = [*cpus*]
- Ou avoir compilé le noyau avec l'option : *CONFIG_NO_HZ_FULL_ALL*

Note : Seul le CPU sur lequel le noyau a booté recevra les *tick*s.



IsolCPU

Dans le contexte du HPC, les codes de calcul sont très sensibles aux perturbations du système d'exploitation.
La synchronisation des communications entre les différents processus est primordiale pour garantir des performances optimales.

IsolCPU

Dans le contexte du HPC, les codes de calcul sont très sensibles aux perturbations du système d'exploitation.
La synchronisation des communications entre les différents processus est primordiale pour garantir des performances optimales.

L'idée est d'isoler tous les processus du système sur un subset de processeurs.

IsolCPU : mise en place

- Sur la ligne de boot du noyau, activer l'option : *isol_cpu* = [*cpus*]

isolcpu

```
isolcpus=       [KNL,SMP] Isolate CPUs from the general scheduler.
                Format:
                <cpu number>,...,<cpu number>
                or
                <cpu number>-<cpu number>
                (must be a positive range in ascending order)
                or a mixture
                <cpu number>,...,<cpu number>-<cpu number>

                This option can be used to specify one or more CPUs
                to isolate from the general SMP balancing and scheduling
                algorithms. You can move a process onto or off an
                "isolated" CPU via the CPU affinity syscalls or cpuset.
                <cpu number> begins at 0 and the maximum value is
                "number of CPUs in system - 1".

                This option is the preferred way to isolate CPUs. The
                alternative -- manually setting the CPU mask of all
                tasks in the system -- can cause problems and
                suboptimal load balancer performance.
```

IsolCPU : mise en place

- Sur la ligne de boot du noyau, activer l'option : *isol_cpu* = [*cpus*]
- L'ordonnanceur ne choisira plus aucun des processeurs indiqués par cette option.

isolcpu

```
isolcpus=       [KNL,SMP] Isolate CPUs from the general scheduler.
                Format:
                <cpu number>,...,<cpu number>
                or
                <cpu number>-<cpu number>
                (must be a positive range in ascending order)
                or a mixture
                <cpu number>,...,<cpu number>-<cpu number>

                This option can be used to specify one or more CPUs
                to isolate from the general SMP balancing and scheduling
                algorithms. You can move a process onto or off an
                "isolated" CPU via the CPU affinity syscalls or cpuset.
                <cpu number> begins at 0 and the maximum value is
                "number of CPUs in system - 1".

                This option is the preferred way to isolate CPUs. The
                alternative -- manually setting the CPU mask of all
                tasks in the system -- can cause problems and
                suboptimal load balancer performance.
```

IsolCPU : mise en place

- Sur la ligne de boot du noyau, activer l'option : *isol_cpu* = [*cpus*]
- L'ordonnanceur ne choisira plus aucun des processeurs indiqués par cette option.
- Nécessité d'avoir un outil permettant de choisir explicitement le CPU sur lequel doit être exécuté un processus (*numactl*, *slurmi*).

isolcpu

```
isolcpus=       [KNL,SMP] Isolate CPUs from the general scheduler.
                Format:
                <cpu number>,...,<cpu number>
                or
                <cpu number>-<cpu number>
                (must be a positive range in ascending order)
                or a mixture
                <cpu number>,...,<cpu number>-<cpu number>

                This option can be used to specify one or more CPUs
                to isolate from the general SMP balancing and scheduling
                algorithms. You can move a process onto or off an
                "isolated" CPU via the CPU affinity syscalls or cpuset.
                <cpu number> begins at 0 and the maximum value is
                "number of CPUs in system - 1".

                This option is the preferred way to isolate CPUs. The
                alternative -- manually setting the CPU mask of all
                tasks in the system -- can cause problems and
                suboptimal load balancer performance.
```

IsolCPU : mise en place

- Sur la ligne de boot du noyau, activer l'option : *isol_cpu* = [*cpus*]
- L'ordonnanceur ne choisira plus aucun des processeurs indiqués par cette option.
- Nécessité d'avoir un outil permettant de choisir explicitement le CPU sur lequel doit être exécuté un processus (*numactl*, *slurmi*).
- Problème avec les processus multi-threadés (sauf avec openMP).

isolcpu

```
isolcpus=       [KNL,SMP] Isolate CPUs from the general scheduler.
                Format:
                <cpu number>,...,<cpu number>
                or
                <cpu number>-<cpu number>
                (must be a positive range in ascending order)
                or a mixture
                <cpu number>,...,<cpu number>-<cpu number>

                This option can be used to specify one or more CPUs
                to isolate from the general SMP balancing and scheduling
                algorithms. You can move a process onto or off an
                "isolated" CPU via the CPU affinity syscalls or cpuset.
                <cpu number> begins at 0 and the maximum value is
                "number of CPUs in system - 1".

                This option is the preferred way to isolate CPUs. The
                alternative -- manually setting the CPU mask of all
                tasks in the system -- can cause problems and
                suboptimal load balancer performance.
```

numactl

Pour répondre à la problématique de placement de processus sur tel ou tel CPU, il existe la technique de punaisage (*pining*).

numactl

Pour répondre à la problématique de placement de processus sur tel ou tel CPU, il existe la technique de punaisage (*pining*).
C'est l'outil *numactl* qui va nous permettre d'effectuer ce *pining*.

numactl

Pour répondre à la problématique de placement de processus sur tel ou tel CPU, il existe la technique de punaisage (*pining*).
C'est l'outil *numactl* qui va nous permettre d'effectuer ce *pining*.

Pour effectuer ce placement, *numactl* utilise les appels systèmes suivants :

- *sched_getaffinity* : syscall de récupération de l'affinité d'un processus.
- *sched_setaffinity* : syscall de modification de l'affinité des processus.

numactl

Pour répondre à la problématique de placement de processus sur tel ou tel CPU, il existe la technique de punaisage (*pining*).
C'est l'outil *numactl* qui va nous permettre d'effectuer ce *pining*.

Pour effectuer ce placement, *numactl* utilise les appels systèmes suivants :

- *sched_getaffinity* : syscall de récupération de l'affinité d'un processus.
- *sched_setaffinity* : syscall de modification de l'affinité des processus.

numactl

```
$ numactl --hardware
available: 1 nodes (0)
node 0 cpus: 0 1 2 3
node 0 size: 7878 MB
node 0 free: 3784 MB
node distances:
node   0
  0:  10
```

numactl

```
$ numactl
usage: numactl [--all | -a] [--interleave= | -i <nodes>] [--preferred= | -p <node>]
               [--physcpubind= | -C <cpus>] [--cpunodebind= | -N <nodes>]
               [--membind= | -m <nodes>] [--localalloc | -l] command args ...
       numactl [--show | -s]
       numactl [--hardware | -H]
       numactl [--length | -l <length>] [--offset | -o <offset>] [--shmmode | -M <shmmode>]
               [--strict | -t]
               [--shmid | -I <id>] --shm | -S <shmkeyfile>
               [--shmid | -I <id>] --file | -f <tmpfsfile>
               [--huge | -u] [--touch | -T]
               memory policy | --dump | -d | --dump-nodes | -D

memory policy is --interleave | -i, --preferred | -p, --membind | -m, --localalloc | -l
<nodes> is a comma delimited list of node numbers or A-B ranges or all.
Instead of a number a node can also be:
  netdev:DEV the node connected to network device DEV
  file:PATH  the node the block device of path is connected to
  ip:HOST    the node of the network device host routes through
  block:PATH the node of block device path
  pci:[seg:]bus:dev[:func] The node of a PCI device
<cpus> is a comma delimited list of cpu numbers or A-B ranges or all
all ranges can be inverted with !
all numbers and ranges can be made cpuset-relative with +
the old --cpubind argument is deprecated.
use --cpunodebind or --physcpubind instead
<length> can have g (GB), m (MB) or k (KB) suffixes

$ numactl --physcpubind=3 --membind=0 /usr/sbin/httpd
```

Les cgroups

- Les *cgroup*s permettent d'isoler finement les processus
- Des limites par utilisateur ou groupe peuvent être mises en place
  - Limiter la mémoire
  - Limiter le nombre de CPU (cpuset)
  - Limiter les IO effectuées
  - Limiter l'accès aux périphériques

/sys/fs/cgroup

```
$ ls /sys/fs/cgroup/
blkio  cpuacct            cpuset   freezer  net_cls              net_prio    pids     unified
cpu    cpu,cpuacct  devices  memory   net_cls,net_prio       perf_event  systemd
```

Les cgroups

- Les *cgroup*s permettent d'isoler finement les processus
- Des limites par utilisateur ou groupe peuvent être mises en place
  - Limiter la mémoire
  - Limiter le nombre de CPU (cpuset)
  - Limiter les IO effectuées
  - Limiter l'accès aux périphériques

/sys/fs/cgroup

```
$ ls /sys/fs/cgroup/
blkio  cpuacct            cpuset   freezer  net_cls              net_prio    pids     unified
cpu    cpu,cpuacct  devices  memory   net_cls,net_prio       perf_event  systemd
```

Documentation : Documentation/cgroup-v2.txt

Sommaire


POLICE PUBLIC CALL BOX
POLICE PUBLIC CALL BOX
POLICE TELEPHONE
FREE
FOR USE OF
PUBLIC
PULL TO OPEN

Les deux phrases les plus prononcées par les utilisateurs :

Les deux phrases les plus prononcées par les utilisateurs :

- C'est lent !

Les deux phrases les plus prononcées par les utilisateurs :

- C'est lent !
- Ça ne marche pas !

Les deux phrases les plus prononcées par les utilisateurs :

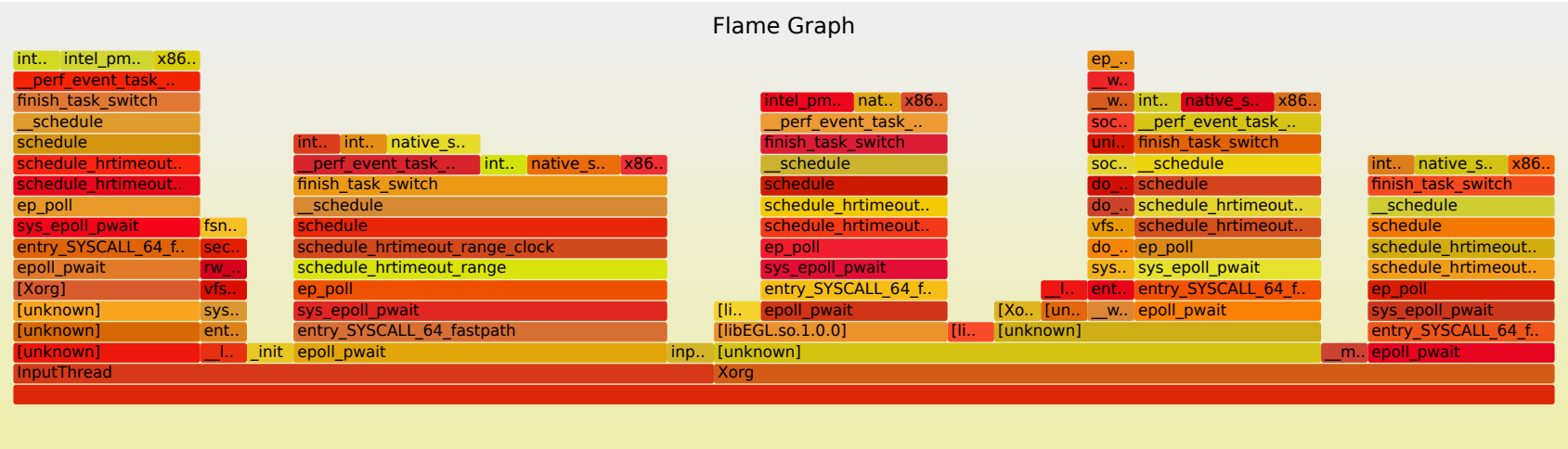
- C'est lent !
- Ça ne marche pas !
- ~~Où est le docteur ??~~

Flamegraph

- Prend un ensemble de trace provenant d'outils externes.
  - gdb
  - perf
  - systemtap

Flamegraph

- Prend un ensemble de trace provenant d'outils externes.
  - gdb
  - perf
  - systemtap
- Analyse le contenu de ces traces et génère un histogramme d'utilisation.



http://www.brendangregg.com/flamegraphs.html

perf probe

Dans l'épisode précédent, nous avons parlé de l'outil perf.
Il permet de visualiser les fonctions les plus appelées du kernel.
Mais il peut faire bien mieux...

`http://www.brendangregg.com/perf.html`

perf probe

perf probe

```
# perf probe -L vfs_open
<vfs_open@/usr/src/debug/kernel-4.13.fc26/linux-4.13.13-200.fc26.x86_64/fs/open.c:0>
      0  int vfs_open(const struct path *path, struct file *file,
                      const struct cred *cred)
      2  {
      3          struct dentry *dentry = d_real(path->dentry, NULL, file->f_flags);

      5          if (IS_ERR(dentry))
                         return PTR_ERR(dentry);

      8          file->f_path = *path;
      9          return do_dentry_open(file, d_backing_inode(dentry), NULL, cred);
     10  }

         struct file *dentry_open(const struct path *path, int flags,
                                  const struct cred *cred)

# perf probe  vfs_open:8 path
Added new events:
  probe:vfs_open       (on vfs_open:8 with path)
  probe:vfs_open_1     (on vfs_open:8 with path)

You can now use it in all perf tools, such as:

        perf record -e probe:vfs_open_1 -aR sleep 1

# perf record -e probe:vfs_open_1 -aR sleep 1
```

perf probe

perf script

```
[ perf record: Woken up 143 times to write data ]
[ perf record: Captured and wrote 35.816 MB perf.data (466530 samples) ]
# perf script | head -n 40
            perf  1278 [000]   671.237370: probe:vfs_open_1: (ffffffffa926ffea) path=0xffffab6300c63dc0
            perf  1279 [000]   671.237426: probe:vfs_open_1: (ffffffffa926ffea) path=0xffffab6300bdbd10
            perf  1279 [000]   671.237508: probe:vfs_open_1: (ffffffffa926ffea) path=0xffffab6300bdbbb0
           sleep  1279 [000]   671.237777: probe:vfs_open_1: (ffffffffa926ffea) path=0xffffab6300bdbdc0
           sleep  1279 [000]   671.237790: probe:vfs_open_1: (ffffffffa926ffea) path=0xffffab6300bdbdc0
           sleep  1279 [000]   671.237992: probe:vfs_open_1: (ffffffffa926ffea) path=0xffffab6300bdbdc0
           sleep  1279 [000]   671.238028: probe:vfs_open_1: (ffffffffa926ffea) path=0xffffab6300bdbdc0
           sleep  1279 [000]   671.238037: probe:vfs_open_1: (ffffffffa926ffea) path=0xffffab6300bdbdc0
           sleep  1279 [000]   671.238049: probe:vfs_open_1: (ffffffffa926ffea) path=0xffffab6300bdbdc0
           sleep  1279 [000]   671.238059: probe:vfs_open_1: (ffffffffa926ffea) path=0xffffab6300bdbdc0
           sleep  1279 [000]   671.238068: probe:vfs_open_1: (ffffffffa926ffea) path=0xffffab6300bdbdc0
           sleep  1279 [000]   671.238077: probe:vfs_open_1: (ffffffffa926ffea) path=0xffffab6300bdbdc0
           sleep  1279 [000]   671.238085: probe:vfs_open_1: (ffffffffa926ffea) path=0xffffab6300bdbdc0
           sleep  1279 [000]   671.238093: probe:vfs_open_1: (ffffffffa926ffea) path=0xffffab6300bdbdc0
           sleep  1279 [000]   671.238097: probe:vfs_open_1: (ffffffffa926ffea) path=0xffffab6300bdbdc0
           sleep  1279 [000]   671.238106: probe:vfs_open_1: (ffffffffa926ffea) path=0xffffab6300bdbdc0
           sleep  1279 [000]   671.238115: probe:vfs_open_1: (ffffffffa926ffea) path=0xffffab6300bdbdc0
           sleep  1279 [000]   671.238129: probe:vfs_open_1: (ffffffffa926ffea) path=0xffffab6300bdbdc0
           sleep  1279 [000]   671.238139: probe:vfs_open_1: (ffffffffa926ffea) path=0xffffab6300bdbdc0
           sleep  1279 [000]   671.238147: probe:vfs_open_1: (ffffffffa926ffea) path=0xffffab6300bdbdc0
           crazy   955 [000]   671.238210: probe:vfs_open_1: (ffffffffa926ffea) path=0xffffab6300ccbdc0
           crazy   955 [000]   671.238213: probe:vfs_open_1: (ffffffffa926ffea) path=0xffffab6300ccbdc0
           crazy   955 [000]   671.238215: probe:vfs_open_1: (ffffffffa926ffea) path=0xffffab6300ccbdc0
           crazy   955 [000]   671.238218: probe:vfs_open_1: (ffffffffa926ffea) path=0xffffab6300ccbdc0
           crazy   955 [000]   671.238220: probe:vfs_open_1: (ffffffffa926ffea) path=0xffffab6300ccbdc0
           crazy   955 [000]   671.238222: probe:vfs_open_1: (ffffffffa926ffea) path=0xffffab6300ccbdc0
           crazy   955 [000]   671.238224: probe:vfs_open_1: (ffffffffa926ffea) path=0xffffab6300ccbdc0
           crazy   955 [000]   671.238226: probe:vfs_open_1: (ffffffffa926ffea) path=0xffffab6300ccbdc0
           crazy   955 [000]   671.238228: probe:vfs_open_1: (ffffffffa926ffea) path=0xffffab6300ccbdc0
           crazy   955 [000]   671.238230: probe:vfs_open_1: (ffffffffa926ffea) path=0xffffab6300ccbdc0
```

Sommaire

Questions