Université de Sherbrooke

Migration d'un processus

Rapport de projet réalisé pour Gabriel Girard
Dans le cadre du cours IFT 630 Processus concurrents et parallélisme
Département informatique
Faculté des sciences
Université de Sherbrooke
2013/04/26

Document rédigé par :
Jonathan Massé
Francis Ouellet

Résumé

Ce document va traiter la possibilité de migrer un processus sur plusieurs machines. Cette fonctionnalité peut être intéressante pour qu'un système d'exploitation gère mieux ses ressources ou en cas de pannes. Pour qu'un autre processus puisse continuer à être exécuté sur un processus, nous avons besoin de considérer plusieurs aspects. Ces derniers s'avèrent la possibilité de suspendre le processus, de transférer son contexte d'exécution et que la reprise se fait sur le nouveau processeur. Malgré toutes les problématiques envisagées et rencontrées, nous avons trouvé une solution simple qui utilise DMTCP qui fonctionne sur Linux. En fait, c'est une application dit « Application checkpointing» [1] qui permet de faire faire des sauvegardes et des restaurations d'un processus. Pour ce qui concerne la migration, elle sera transférée sous la forme d'un fichier simplement à l'aide d'un service de partage de fichiers.

[1] C' est une technique pour ajouter une tolérance aux pannes dans les systèmes informatiques. Il se compose essentiellement de stocker l'état de l'application en cours, et plus tard, de l'utiliser pour le redémarrage de l'exécution, en cas d'échec. Référence : http://en.wikipedia.org/wiki/Application_checkpointing

Université de Sherbrooke

Table des matières

Introduction

Motivations

Il existe beaucoup de raisons pertinentes pour utiliser ce type de technologie. Voici quelques raisons.

1. *Tolérance aux pannes* : en cas d'anormalité, ou en termes de prévention, continuer un processus sur une autre machine.
2. *Partage de ressources* : une machine trop occupée peut se libérer, et lancer le processus à une autre machine moins occupée.

Rappels

Un *Processus* est un concept important. En fait, il est la représentation d'une exécution d'une séquence d'instructions. Et, son état évolue à son exécution.

Un *état ou l'image d'un processus* contient toutes les informations d'un processus et nécessaires à son exécution. Par exemple, le contenu du contexte d'exécution, etc.

Un *contexte d'exécution* est l'ensemble des données qu'un processus utilise. Ces données sont contenues dans les registres du processeur, et la zone mémoire utilisée par le processus.

Une *migration de processus* engendre le transfert d'un processus sur un processeur source vers un processeur destinataire. On peut définir cette tâche en plusieurs étapes :

1. Arrêt ou gel d'un processus. C'est-à-dire arrêter l'exécution du processus.
2. Transférer l'état du processus sur la machine destinataire.
3. Restaurer l'exécution du processus sur la nouvelle machine.

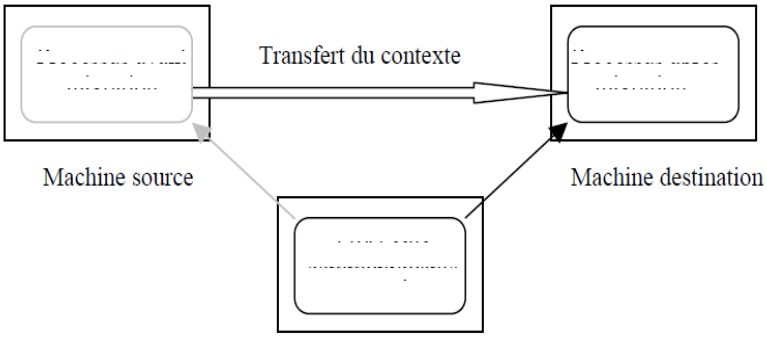

Figure 1 [2]

Problèmes

Hétérogénéité

Chaque machine peut comporter une architecture différente, comporter des registres différents, et comporter une version de système d'exploitation différent. Par exemple, il faut s'assurer qu'une traduction se fasse de la machine source et la machine destinataire, et de façon transparente le plus possible.

Le gel de processus

On veut être capable de sauvegarder l'image d'un processus, et de le restaurer sur un processeur. Or, il y a des moments qu'il est impossible de faire cette sauvegarde. Par exemple, lors d'un appel système, ou le processus est dans une zone non interruptible.

Préalables pour poursuivre l'exécution sur un autre processeur

Il faut s'assurer de déterminer toutes les entités nécessaires dans un état cohérent, et de pouvoir les rendre sur le processeur destinataires. Par exemple, l'état du processus, l'horloge des systèmes, les bibliothèques système, un serveur de fichiers, etc. Ces dernières ne sont pas toujours transférables. Il faut remédier à une solution dans ce cas.

---

[2] Voir Mécanisme pour la migration de processus, page 6

Échange des messages

S'assurer que la migration a été correctement migrée. Que faisons-nous pour le maintien des messages entre les processus ? Que faisons-nous si nous migrons un processus, et que recevons des messages ?

Aspects théoriques importants

Image ou état d'un processus

La décomposition se fait sous les composantes suivantes :

1. L'espace mémoire adressable
    a. Code exécutable
    b. Piles
    c. Les données

2. Informations de contrôle et exécution du programme
    a. Valeurs des registres du processeur
        i. Compteur ordinal
        ii. Les registres généraux
        iii. Les registres d'état du processus

    b. Informations pour gérer le processus
        i. Identifiant du processus (PID)
        ii. Priorité
        iii. Etc.

    c. Informations pour gérer l'accès aux ressources et les communications
        i. Descripteurs des fichiers
        ii. Tables de pagination
        iii. Descripteurs des objets de communication
        iv. Etc.

Il ne faut pas oublier qu'on doit traiter aussi les fils d'exécution qui partagent la mémoire virtuelle du processus, mais possèdent par contre leur propre pile d'appels.

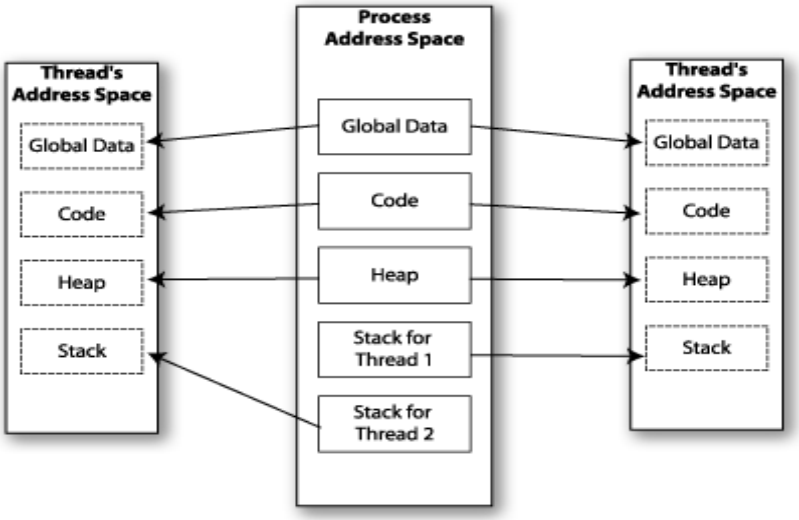

Figure 2[3]

Préalables d'un processus

On peut définir les préalables comme les représentations nécessaires à l'exécution du processus. Voici divisé en deux catégories :

1. Appels systèmes (ressources nécessaires)
    a. Horloge système
    b. Bibliothèques systèmes
2. Propres au processeur d'exécution
    a. Serveurs
    b. Processus d'application
    c. Segments de mémoire
3. Etc.

[3] Référence : http://cocoadevcentral.com/articles/000061.php

Il est primordial de s'assurer que ces préalables sont disponibles pour l'exécution du processus. On peut dire que l'état du processus est une vue d'un système de ce processus. Mais, il est dur de pouvoir prévoir tous les préalables. Certains de ces derniers doivent être généralement transférés sur le processeur destinataire. Par exemple, les structures de contrôles d'un processus. Par contre, certains ne doivent pas être transférés. Par exemple, l'horloge du processeur, et les serveurs de communication[4].

Suspension ou restauration d'un processus

L'opération de suspension doit arrêter l'exécution du processus dans un état figé pour capturer tous les éléments nécessaires à la migration du processus. En ce qui concerne la durée, elle est très variable. Mais, il n'est pas toujours possible d'extraire un état d'un processus. Souvent, quand le processus est engagé dans un appel de procédure de système, ou il est dans une section sans interruption. Certains moments, il est possible que certains registres ne soient pas accessibles en lecture ou écriture. En ce qui concerne les moments définis pour la suspension, on peut définir des points de reprise ou faire simplement un retour en arrière où qu'on puisse extraire le contexte complet du processus.

[4] Un serveur de communication est essentiellement utilisé pour contrôler les connexions et des échanges avec l'extérieur. Il comprend un serveur classique et généralement de nombreuses interfaces de communication comme des cartes modem. Référence: http://www.marche-public.fr/Terminologie/Entrees/Serveur-de-communication.htm

Transfert de l'image du processus

Copie de l'espace d'adressage

Cette technique consiste à transférer directement l'espace d'adressage[5] entier sur le nouveau processeur. Généralement, on fait la copie de l'espace d'adresse avant de suspendre le processus. La copie induit dans ce cas un volume important de transfert de données.

Transfert à la demande

Cette technique consiste à ne pas transférer directement l'espace d'adressage sur le nouveau processeur. On découpe l'espace mémoire en pages, et cette dernière est l'unité de transfert. Généralement, on effectue une copie partielle de l'espace d'adressage. Ensuite, quand le processus est exécuté sur le nouveau processeur, et lorsqu'il y a un défaut de page, on ramène la page référencée. Dans une variante, on peut aussi précharger d'autres pages pour prévenir des défauts de pages. Cette technique permet de réduire l'espace mémoire qu'on doit copier. Elle est surtout avantageuse pour un processus qui travaille sur une faible partie de la mémoire. Cependant, il existe de nombreux processus qui travaillent beaucoup avec son espace d'adressage. Donc, cette technique risque d'être plus que coûteuse.

Stockage partagé

L'autre technique pour sauvegarder l'espace d'adressage est de sauvegarder les pages modifiées sur un système afin de stocker des données partagées. Il est important de noter que la mémoire de stockage n'est pas nécessaire un disque. On peut utiliser la

[5] L'adressage mémoire est une technique électronique et informatique permettant à certains composants d'accéder à la mémoire. Une adresse mémoire est un identifiant qui désigne une zone particulière de la mémoire physique où des données peuvent être lues et stockées, temporairement (mémoire vive) ou de façon durable (mémoire non volatile).: Référence: http://fr.wikipedia.org/wiki/Adressage_m%C3%A9moire

cache répartie sur un réseau par exemple. Celle-ci permettra de sauvegarder le contenu sur un espace mémoire (disque ou cache).

Transferts parallèles

Cette technique est utilisée généralement dans un réseau maillé. Donc, il existe différents chemins entre les deux processeurs. On envisage dans ce cas d'utiliser tous ces chemins pour répartir la charge. Il reste que c'est difficile de bien répartir les charges, et de gérer une synchronisation.

Solution appliquée dans notre projet

La technique utilisée est celle de la copie de l'espace d'adressage, vu que le temps n'est pas primordial dans une « Application checkpointing ». Le transfert aussi se fait directement sur disque à l'aide d'un répertoire partagé sur internet. Donc, le temps de traitement va dépendre du transfert réseau, le temps de lecture et écriture de disques, et le temps que l'utilisateur prend pour exécuter la commande.

Solutions possibles

1. Écrire la programmation sur un programme
    a. Qui sauvegardent l'état d'un processus, et le restaure à cet état. (Setjmp et LongJmp)
    b. Faire une migration fonctionnelle (très complexe)
2. Les outils qui traitent la problématique « Application checkpointing »
    a. Ils sont tous sur Linux
3. Les machines virtuelles qui permettent de sauvegarder et de restaurer un état de machine très précis.

Problèmes rencontrés

Le système d'exploitation Windows

Nous nous sommes vites rendu compte que les langages tels que C# étaient une très mauvaise idée, car ils ont une sécurité dans .NET qui bloque l'accès aux processus et ressources qui ne leur appartiennent pas. Nous avons donc décidé de développer une application en C++ qui permet d'écrire et de lire dans un processus et des ressources qui ne leur appartiennent pas.

Nous avons découvert que Windows bloque l'accès entre chaque processus et qu'il est impossible d'avoir accès à la mémoire d'un programme sauf si le programme en exécution est en mode « debug ». Le mode « debug » permet de donner le privilège d'avoir accès à la mémoire des autres processus.

Nous avons donc programmé une fonction qui permet d'élever les privilèges du processus, nous nous sommes basés sur des exemples sur Internet, pour ensuite avoir l'accès à la mémoire des autres processus avec l'aide de leur PID de processus.

Voici un exemple de cette fonction :

```
////////////////////////////////////////
// Permet d'élevé les privileges en Windows.
// Basé sur le site http://celestialcoding.com/code-snipplets/enable-debug-
privileges/
////////////////////////////////////////
void EnableDebugPriv()
{
    HANDLE hToken;
    LUID luid;
    TOKEN_PRIVILEGES tkp;
       ////////////////////////////////////////
       // Ouverture des privileges.
       ////////////////////////////////////////
    OpenProcessToken(GetCurrentProcess(), TOKEN_ADJUST_PRIVILEGES | TOKEN_QUERY,
&hToken);
    LookupPrivilegeValue(NULL, SE_DEBUG_NAME, &luid);
```

```
    ////////////////////////////////////////
    // Changement des privilèges.
    ////////////////////////////////////////
    tkp.PrivilegeCount = 1;
    tkp.Privileges[0].Luid = luid;
    tkp.Privileges[0].Attributes = SE_PRIVILEGE_ENABLED;
    AdjustTokenPrivileges(hToken, false, &tkp, sizeof(tkp), NULL, NULL);
    CloseHandle(hToken);
}
```

Nous avons été par la suite soumis à trois problèmes :

1) Comment trouver l'adresse de base du programme?

   Pour ce problème a été résolu, nous avons codé une fonction en C++ qui fait une requête au système pour recevoir l'adresse du début du programme en mémoire.

   Voici un exemple de cette fonction :

```
////////////////////////////////////////
// Permet d'obtenir l'adresse de départ d'un programme.
// Basé sur le site http://stackoverflow.com/questions/12451976/a-remote-
process-base-address-and-snapshot
////////////////////////////////////////
DWORD getProcBaseAddress(DWORD targetPID)
{
    hModuleSnap = CreateToolhelp32Snapshot( TH32CS_SNAPMODULE, targetPID );
    MODULEENTRY32 me32;
    me32.dwSize = sizeof( MODULEENTRY32 );
    Module32First( hModuleSnap, &me32 );
    cout << "Proc Base Address = " << (DWORD)me32.modBaseAddr;
    return (DWORD) me32.modBaseAddr;
}
```

2) Comment trouver les registres (le pointeur de pile, etc.) d'un autre processus? L'utilisation de « setjump » et « longjump » sont utilisées seulement pour nous même, mais pour un processus qui n'est pas le nôtre, comment faisons-nous ? Le développement sur cet aspect serait expliqué plus tard.

3) Comment restaurer les « handle »[6]? Les « handles » en Windows sont des pointeurs qui pointent sur des objets qui ont été précédemment instanciés et qui appartiennent au système d'exploitation. Nous avons constaté que même si nous faisions une sauvegarde de la mémoire et que nous la restaurions plus tard, tous les pointeurs de type « HANDLE » devenaient invalides et qu'il est impossible de les restaurer. Nous avons aussi remarqué que tous les éléments dans une fenêtre Windows possèdent leur propre « HANDLE ».

Nous avons donc conclu qu'en Windows dû au fait des « HANDLES » qui deviennent invalides, il est impossible de recréer un programme par une sauvegarde de la mémoire. La seule solution serait que le programme puisse gérer son arrêt et qu'il gère sa réouverture et recharger son dernier état ou sinon de faire un « snapshot » sur le système d'exploitation comme VMware par exemple.

Nous avons aussi trouvé des éléments intéressant qui permettent de faire des transactions en C# et qui permettent de recharger un état précédent de l'application, sauf qu'encore une fois cette méthode fonctionne seulement pour le processus lui-même et ne peut fonctionner pour un processus qui n'est pas nous-même.

Setjmp /LongJmp

En C++ ou en C, l'utilisation des setjmp et longjmp est intéressante si nous voulions revenir à un état précédent d'un programme, toutefois, les instructions doivent être exécutées par le processus qui souhaite revenir dans un état antérieur.

[6] *Handle* est un utilitaire qui affiche des informations sur les descripteurs ouverts pour tous les processus du système. Référence: http://technet.microsoft.com/fr-ca/sysinternals/bb896655.aspx

La commande « setjmp » permet de sauver les registres et le pointeur de pile. Et, on peut restituer cet état à l'aide de la commande « longjmp ». Par contre, dans tous les exemples, la commande « setjmp » n'envoyait qu'une seule donnée. Or, nous voulons que toutes les données soient bien réinitialisées à l'état du programme où nous avons mis un « setjmp ».

Aussi, ces commandes ne traitent pas les descripteurs de fichiers, la pile de l'allocation dynamique, etc. Ces commandes sont généralement utilisées pour la gestion des erreurs (même si elle est déconseillée). Elle ne supporte pas le « multithreading ». Par contre, Il existe POSX.1-2001 une version plus avancée des « setjmp » et « longjmp » qui permet de soutenir le « multithreading ». Elle est sous le nom de « setcontext » et « getContext »[7].

J'ai lu sur certains forums que les variables locales devaient être volatiles[8] : « For setjmp/longjmp to work, you need to declare local variables as volatile on an IA64 architecture »[9]. Voici un exemple qui a été testé sur une machine assez récente (Intel i5-2410M, 2.3 GHz) qui sera présentée sous la prochaine page:

[7] Pour plus de détails : voici le lien http://en.wikipedia.org/wiki/Setcontext

[8] Ne jamais mettre cette variable sur un registre, et empêche l'optimisation de la variable en question.

[9] Référence: http://stackoverflow.com/questions/7997600/performance-overhead-of-using-volatile-for-setjmp-longjmp

```
#include <stdio.h>      /* printf */
#include <setjmp.h>     /* jmp_buf, setjmp, longjmp */
    ..
jmp_buf buf;
void f()
{
  int local_var = 1;
  if (setjmp( buf ) == 1)
  {
    printf("%d",local_var);
    return ;
  }

  local_var = 2;
  longjmp( buf, 1 );
}

int main(int argc, char* argv[])
{
  f();
  return 0;
}
```

Lors de l'exécution de ce programme, sans ou avec l'indicateur volatile, la variable locale ne revient jamais au bon état.

Vu à la suite de ces problématiques, et que le problème de faire fonctionner ça sur une autre machine n'était pas regardé, nous avons regardé d'autres solutions.

Machines virtuelles

Les machines virtuelles pouvaient être intéressantes, mais les sauvegardes sont généralement tout l'ensemble d'un état d'un système d'exploitation, et non un processus précis. On trouvait la solution moins intéressante et nous l'avons laissé de côté.

Problèmes de compatibilité

CryoPid

Cet outil est assez simple, il permet de sauvegarder un processus au moment qu'on lance la commande, et crée un nouvel exécutable dans l'état qu'il a été sauvegardé.

Ce logiciel, malgré sa faible complexité d'utilisation, a été très difficile à utiliser. Il n'est pu à jour depuis plusieurs années (dernière version date de 2008) et n'est plus supporté sur les nouvelles versions de Linux. Je l'ai testé sous des anciennes versions de Linux qui devait être compatible, mais, hélas, mon architecture n'était pas supportée. Bref, nous avons changé assez rapidement de solutions.

CRIU et BLCR

Nous n'avons pas vraiment posé un regard sur ces solutions.

Explication de la solution choisie

Environnement

Système d'exploitation : Linux Ubuntu	Quantal Quetzal (12.10)
Kernel	3.5
DMTCP	1.2.7

DMTCP

Le fonctionnement est que DMTCP crée un coordonnateur par défaut sur la machine, et lors de la création du processus, la variable d'environnement LD_PRELOAD est utilisée pour charger la librairie dmtcphijack.so. Donc, la librairie roule sur le fils d'exécution principal, et un second fils d'exécution est lancé pour faire la sauvegarde de processus. Un socket[10], et des registres sont créés sur le coordonnateur. Aussi, un gestionnaire de signal est créé (SIGUSR2), et le contrôle retombe au processus utilisateur. Quand un point de sauvegarde est appelé, il envoie sa requête au processus utilisateur. Ce dernier communique par signal aux autres fils d'exécutions.

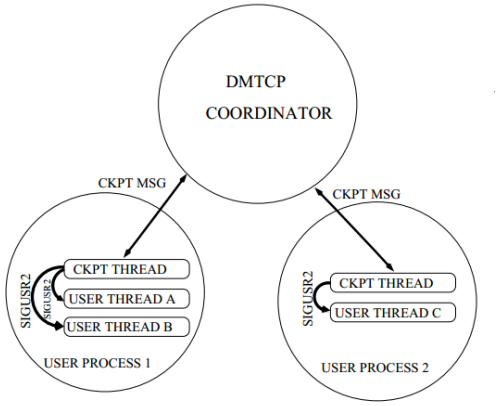

Figure 3[11]

Cette solution permet de faire trois opérations précises :

- Arrêter un processus
- L'état du processus est écrit sur un fichier
- Le processus recommence au point qui a été arrêté

---

[10] Connecteur réseau

[11] Voir la référence *DMTCP and Condor : a new checkpointing mechanism, page 4*

Cette solution comporte de nombreux avantages. Elle supporte le « multithreading », est développée depuis 5 ans, open source et fonctionne sur de nombreuses plateformes. La version du Kernel doit être égale ou supérieure à 2.6.9.

Le coordonnateur est un serveur sans état qu'on utilise pour sauvegarder des processus. Ces responsabilités sont de laisser des processus utilisateurs s'inscrire à lui, et définir de points de contrôles aux processus.

Commandes

dmtcp_checkpoint exécutable

Cette commande permet à un exécutable d'être associé au coordonnateur au moment qu'il va devenir un processus.

dmtcp_coordinator

Exécute un coordinateur qui permet de faire afficher les processus qui ont initialisé par DMTCP, faire un point de sauvegarde sur les processus, etc. L'option « c » permet de sauvegarder tous les processus sur DMTCP. Et, crée plusieurs fichiers. Le fichier .dmtcp contient tous les éléments nécessaires pour redémarrer le processus à l'état sauvegardé. Un script .sh est généré aussi pour s'assurer que le processus relancé fonctionne sur n'importe quelles autres machines.

dmtcp_restart ckpt_filename_global_id.dmtcp

Cette commande permet d'initialiser la continuation de l'état du processus, et continuer son exécution. Le global_id est un numéro unique qui permet la migration de processus. Pour plus d'information, voici un lien qui contient plusieurs publications à ce sujet :

http://dmtcp.sourceforge.net/publications.html.

Installation de l'application

L'installation est très simple sur cette application. Il suffit d'avoir une version linux Ubuntu à jour (par exemple 12.10), et d'installer l'application « DMTCP » à l'aide de la logithèque Ubuntu.

Évaluation de la solution

Langages supportés	C, C++, Java, GNU List, python, Perl, PHP, Mysql, etc.
Bibliothèques parallèles	OpenMPI, OpenMP, MPICH2, etc.
Éditeurs de texte	Vi, vim, cscope, etc.
Applications	VNC, etc.
Bibliothèques systèmes	TCP/IP sockets, fork, exec, mutex, descripteurs de fichiers, les libraries « run-time », etc.

Points à améliorer sur ce type de projet

- Faire une étude plus poussée des concepts théoriques sur ce sujet
- Comment s'assurer que la migration a été faite correctement ?
- Pousser plus loin DMTCP

Conclusion

Bref, la migration de processus n'est pas le procédé le plus trivial qui soit. Elle exige plusieurs opérations importantes : la sauvegarde d'un processus, le transfert et sa restauration. L'image du processus doit rester la même avant et après la migration. Aussi, elle se doit d'être cohérente. Elle permet d'être plus tolérante aux pannes, et d'allouer mieux les ressources (ce cas n'est pas traité dans le cadre de notre projet). Notre solution a été DMTCP qui permet de sauvegarder et restaurer un processus sur Linux.

Bibliographie

- Columbia University. *Linux-Cr : Transparent Application Checkpoint-Restart in Linux* [En ligne], http://landley.net/kdocs/ols/2010/ols2010-pages-159-172.pdf

- Iris. *DMTCP : transparent checkpointing for cluster computations and the desktop* [En ligne], http://iris.lib.neu.edu/cgi/viewcontent.cgi?article=1011&context=comp_info_sci_fac_pubs

- Institut national de recherche informatique et en automatique. *Zero overheadJava Thread Migration* [En ligne], http://hal.archives-ouvertes.fr/docs/00/06/99/13/PDF/RT-0261.pdf

- Charles Weddle. *Java process migration*[En ligne], http://www.charlesweddle.com/s-jpm/doc/jpm-design.pdf

- Université Joseph Fourier. *Mécanismes pour la migration de processus* [En ligne], http://sardes.inrialpes.fr/~bouchena/publications/DEA98/DEA.pdf
- Enseignement Polytechnique. *Go to non locaux en C* [En ligne], http://www.enseignement.polytechnique.fr/informatique/profs/Jean-Jacques.Levy/poly/majifa-syst-96/cours/cours4.html

- Wikipédia. *Setjmp.h* [En ligne], http://en.wikipedia.org/wiki/Setjmp.h

- IBM. *Longjmp* [En ligne], http://publib.boulder.ibm.com/infocenter/iadthelp/v7r0/index.jsp?topic=/com.ibm.etools.iseries.langref.doc/rzan5mst160.htm

- Embescom. *Implementing the setjmp and the longjmp* [En ligne], http://www.embecosm.com/appnotes/ean9/html/ch04s01s02.html

- Team Lib. *Setjmp and longjmp for multilevel in C* [En ligne], http://vapvarun.com/study/softE/john%20wiley%20and%20sons%20-%20programming%20with%20object-oriented%20programming/5399final/lib0095.html

- ShareSource. *What is cryopid* [En ligne], http://sharesource.org/project/cryopid/wiki/What_is_cryopid/

- Cryopid. *Crypid fo Linux* [En ligne], http://cryopid.berlios.de/

- Surf Sara. *DMTCP* [En ligne], https://www.surfsara.nl/systems/lisa/software/dmtcp

- DMTCP. *DMTCP* [En ligne], http://dmtcp.sourceforge.net/

- Northeastern University. *DMTCP and Condor : a new checkpointing mechanism* [En ligne], http://research.cs.wisc.edu/htcondor/CondorWeek2010/condor-presentations/cooperman-dmtcp.pdf

- DMTCP. *DMTCP* docs[En ligne], http://dmtcp.sourceforge.net/docs/index.html

- DMTCP. *DMTCP* FAQ[En ligne], http://dmtcp.sourceforge.net/FAQ.html

- Partie Windows Fonction « `getProcBaseAddress`» en C++ http://stackoverflow.com/questions/12451976/a-remote-process-base-address-and-snapshotl
- Partie Windows Fonction « `EnableDebugPriv`» en C++ `http://celestialcoding.com/code-snipplets/enable-debug-privileges/`