Émulation d'applications distribuées sur SimGrid via Simterpose

Louisa Bessad

Université Pierre et Marie Curie

louisa.bessad@gmail.com

7 Septembre 2015

Tester des applications distribuées

3 méthodes :

- **Exécution sur plateforme réelle** (PlanetLab, Grid'5000)
  - Étude du comportement en conditions réelles
  - Lourd et difficilement reproductible
- **Simulation** (SimGrid)
  Exécution d'un modèle de l'application
  - Mise en oeuvre simple et facilement reproductible
  - Validation d'un modèle
- **Émulation**
  Substitution de l'environnement par un logiciel
  - Exécution réelle dans un environnement virtuel
  - Une version de code

  Emulation standard ou légère ?

Tester des applications distribuées

3 méthodes :

- **Exécution sur plateforme réelle** (PlanetLab, Grid'5000)
  - Étude du comportement en conditions réelles
  - Lourd et difficilement reproductible
- **Simulation** (SimGrid)
  Exécution d'un modèle de l'application
  - Mise en œuvre simple et facilement reproductible
  - Validation d'un modèle
- Émulation
  Substitution de l'environnement par un logiciel
  - Exécution réelle dans un environnement virtuel
  - Une version de code

  Emulation standard ou légère ?

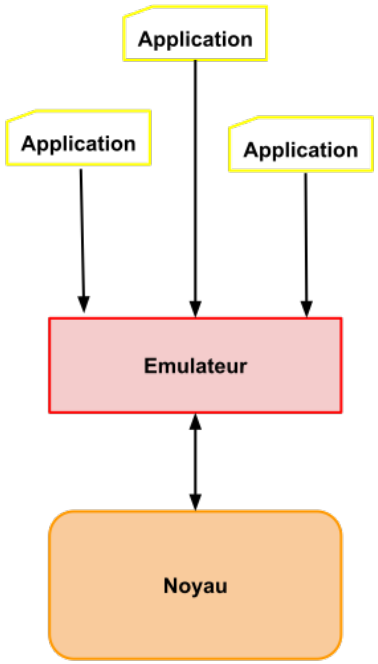
Tester des applications distribuées

3 méthodes :

- **Exécution sur plateforme réelle** (PlanetLab, Grid'5000)
  - Étude du comportement en conditions réelles
  - Lourd et difficilement reproductible
- **Simulation** (SimGrid)
  Exécution d'un modèle de l'application
  - Mise en œuvre simple et facilement reproductible
  - Validation d'un modèle
- **Émulation**
  Substitution de l'environnement par un logiciel
  - Exécution réelle dans un environnement virtuel
  - Une version de code

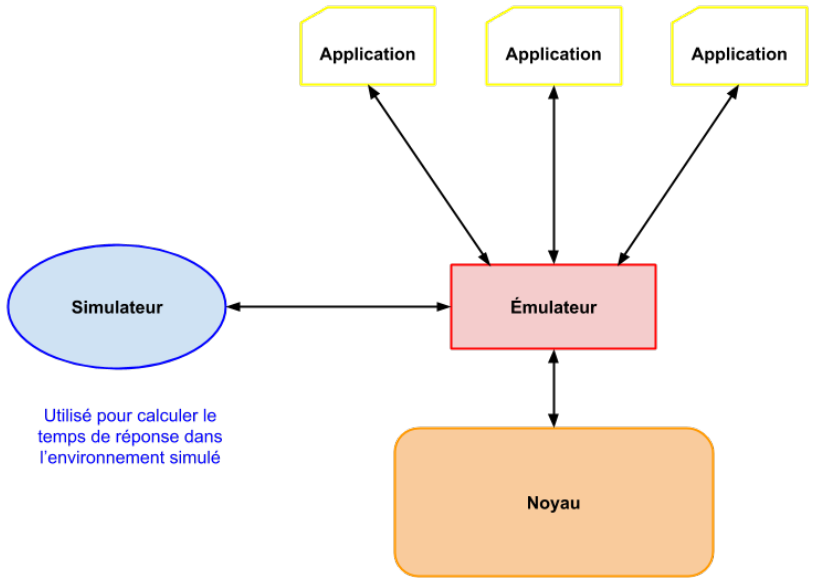
  Emulation standard ou légère ?

Émulation légère par dégradation



Simple à mettre en œuvre
Dépend de la puissance de l'hôte

Émulation légère par interception



- Simulateur : l'environnement virtuel
- Émulateur : Interception

Objectif

Propriété de l'émulateur

- Simple d'utilisation + facilement déployable
- Conditions expérimentales variées
- Expériences reproductibles
- Résistance aux pannes et fautes
- Pas d'accès au fichier source

Choix

- Émulation par interception
- Simulateur : SimGrid
- Émulateur : Simterpose

Objectif

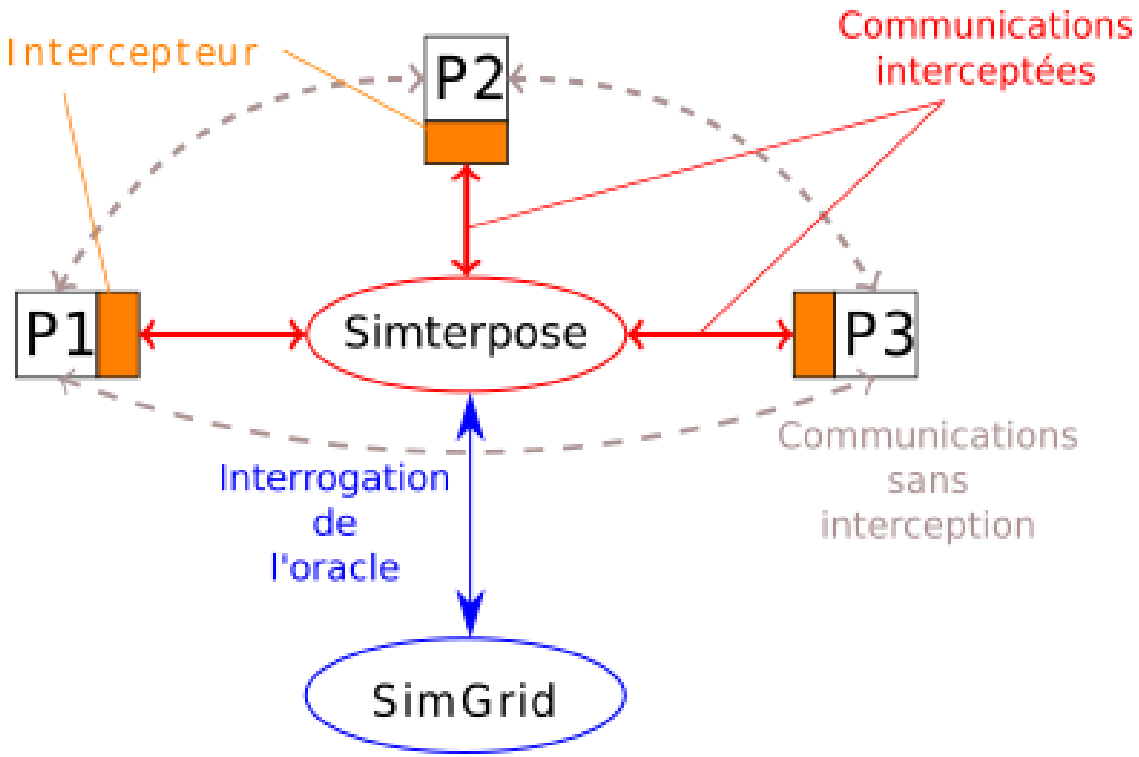
Propriété de l'émulateur

- Simple d'utilisation + facilement déployable
- Conditions expérimentales variées
- Expériences reproductibles
- Résistance aux pannes et fautes
- Pas d'accès au fichier source

Choix

- Émulation par interception
- Simulateur : SimGrid
- Émulateur : Simterpose

SimGrid et Simterpose

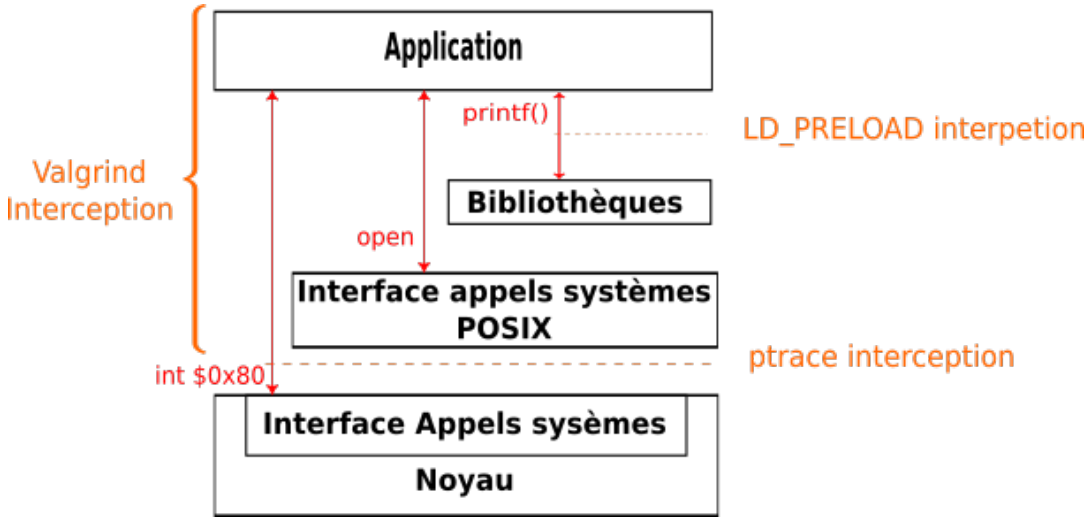


Actions à intercepter

- Communications réseau
- Temps
- Threads
- DNS

Actions à intercepter

- Communications réseau
- Temps
- Threads
- DNS



Haut niveau

Binaire : Valgrind

- Ré-écriture à la volée de fonctions à intercepter
- Temps d'exécution x 7.5

Bibliothèques : `LD_PRELOAD`

- Variable d'environnement
- Permet de charger des bibliothèques dynamiques avant les autres

Haut niveau

Binaire : Valgrind

- Ré-écriture à la volée de fonctions à intercepter
- Temps d'exécution x 7.5

Bibliothèques : `LD_PRELOAD`

- Variable d'environnement
- Permet de charger des bibliothèques dynamiques avant les autres

Bas niveau : Appels Systèmes

ptrace

- Appel système qui permet de contrôler un processus
- Choix des actions de contrôle
- Modification des appels systèmes (handler et registres)
- Nombreux changements de contexte + Non POSIX

Bas niveau : Appels Systèmes

ptrace

- Appel système qui permet de contrôler un processus
- Choix des actions de contrôle
- Modification des appels systèmes (handler et registres)
- Nombreux changements de contexte + Non POSIX

Bas niveau : Appels Systèmes

ptrace

- Appel système qui permet de contrôler un processus
- Choix des actions de contrôle
- Modification des appels systèmes (handler et registres)
- Nombreux changements de contexte + Non POSIX

Bas niveau : Appels Systèmes

ptrace

- Appel système qui permet de contrôler un processus
- Choix des actions de contrôle
- Modification des appels systèmes (handler et registres)
- Nombreux changements de contexte + Non POSIX

Bas niveau : Appels Systèmes

Uprobes

- Insertion de points d'arrêt (API Noyau)
- Module noyau contient le handler de l'appele
- Évite les changements de contexte
- Rapide + accès à toutes les ressources

seccomp/BPF

- Choix des appels systèmes autorisés à s'exécuter
- Fait uniquement de l'interception
- Utilise `ptrace` pour modifier l'appel système

Bas niveau : Appels Systèmes

Uprobes

- Insertion de points d'arrêt (API Noyau)
- Module noyau contient le handler de l'appele
- Évite les changements de contexte
- Rapide + accès à toutes les ressources

seccomp/BPF

- Choix des appels systèmes autorisés à s'exécuter
- Fait uniquement de l'interception
- Utilise `ptrace` pour modifier l'appel système

Quels outils pour quelle interception

ptrace

- Coût : Moyen
- Mise en œuvre : Complexe
- Utilisation :
  Thread (partiel) + Réseau

LD_PRELOAD

- Coût : Faible
- Mise en œuvre : Simple
- Utilisation :
  Temps + Thread + DNS

Implémentation

- Réseau de communications : `ptrace`
- Temps : `LD_PRELOAD`
- Thread : `ptrace` + `LD_PRELOAD`
- DNS : `LD_PRELOAD`

Quels outils pour quelle interception

`ptrace`

- Coût : Moyen
- Mise en œuvre : Complexe
- Utilisation :
  Thread (partiel) + Réseau

`LD_PRELOAD`

- Coût : Faible
- Mise en œuvre : Simple
- Utilisation :
  Temps + Thread + DNS

Implémentation

- Réseau de communications : `ptrace`
- Temps : `LD_PRELOAD`
- Thread : `ptrace` + `LD_PRELOAD`
- DNS : `LD_PRELOAD`

Quels outils pour quelle interception

ptrace

- Coût : Moyen
- Mise en œuvre : Complexe
- Utilisation :
  Thread (partiel) + Réseau

LD_PRELOAD

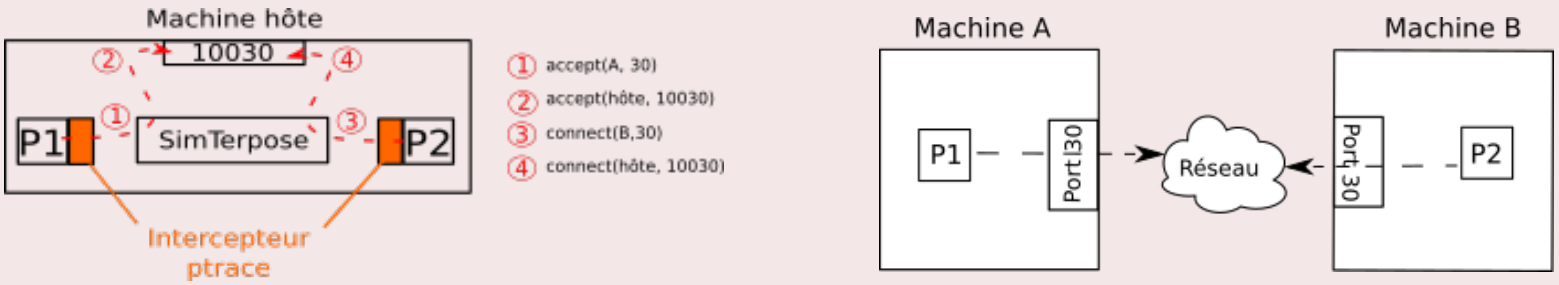
- Coût : Faible
- Mise en œuvre : Simple
- Utilisation :
  Temps + Thread + DNS

Implémentation

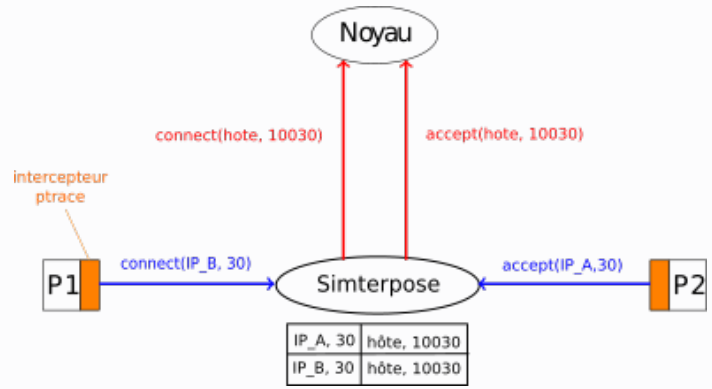
- Réseau de communications : `ptrace`
- Temps : `LD_PRELOAD`
- Thread : `ptrace` + `LD_PRELOAD`
- DNS : `LD_PRELOAD`

Réseau de communications et médiations

Vision du réseau



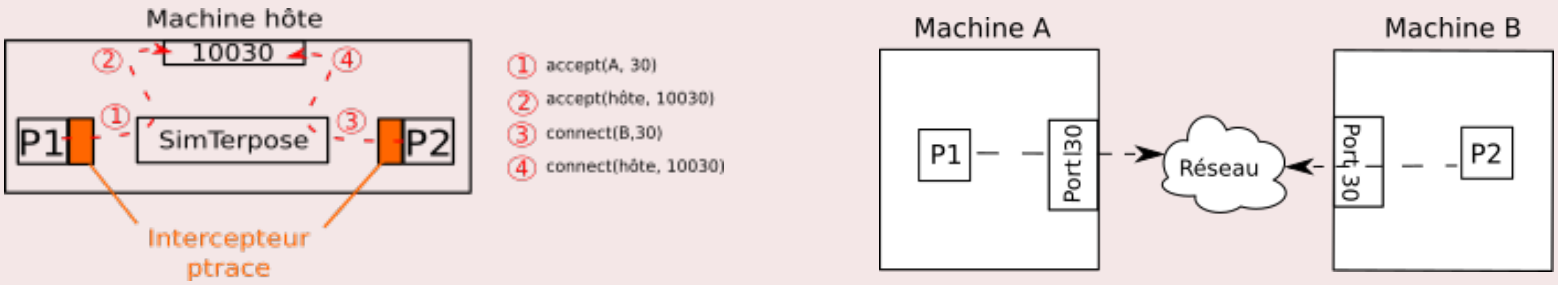
Address Translation



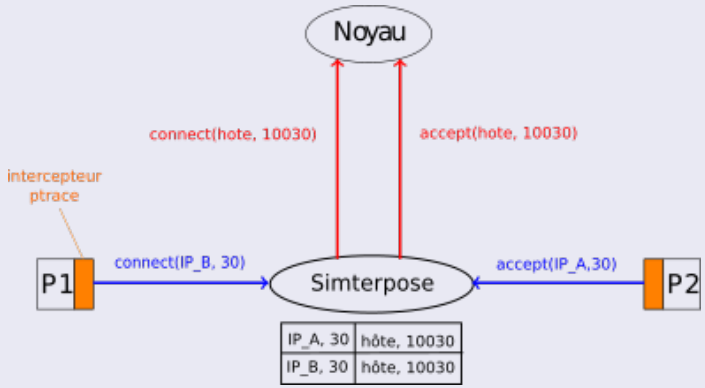
Full Mediation

Réseau de communications et médiations

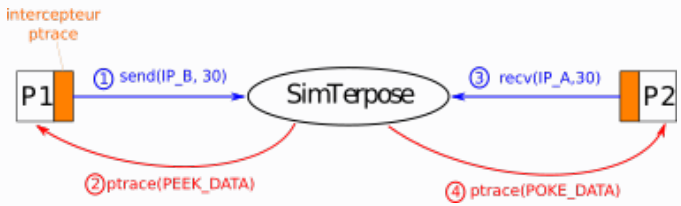
Vision du réseau



Address Translation

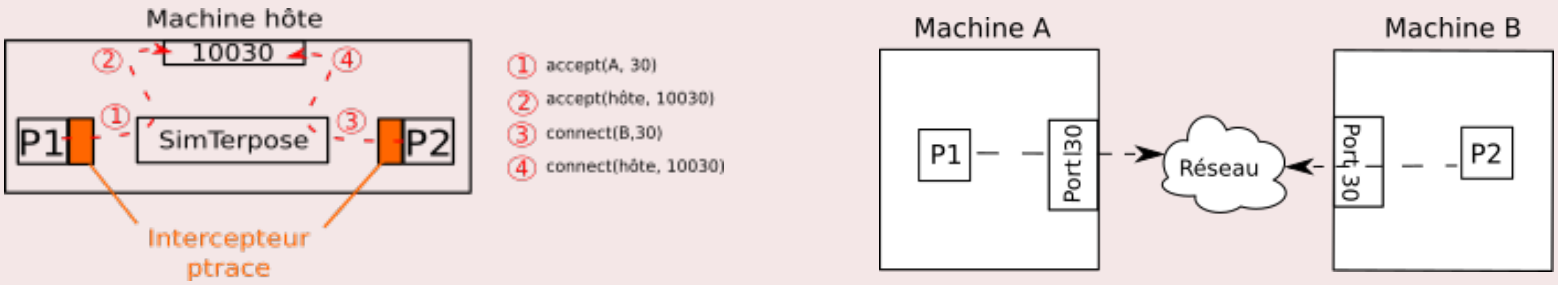


Full Mediation

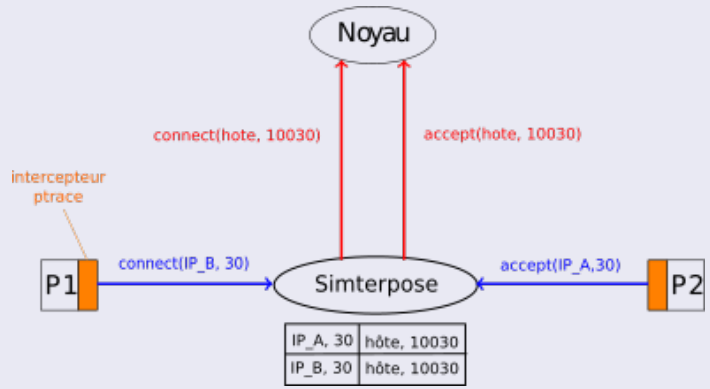


Réseau de communications et médiations

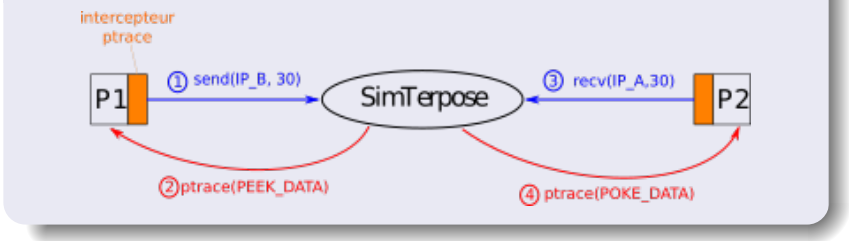
Vision du réseau



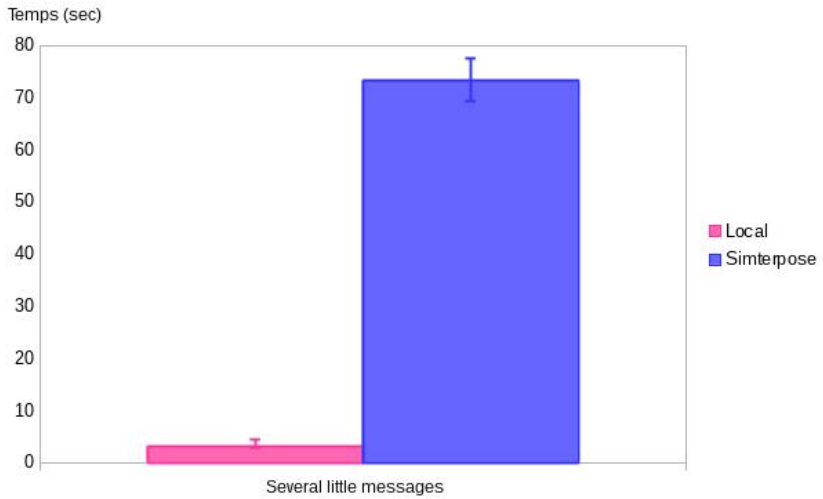
Address Translation

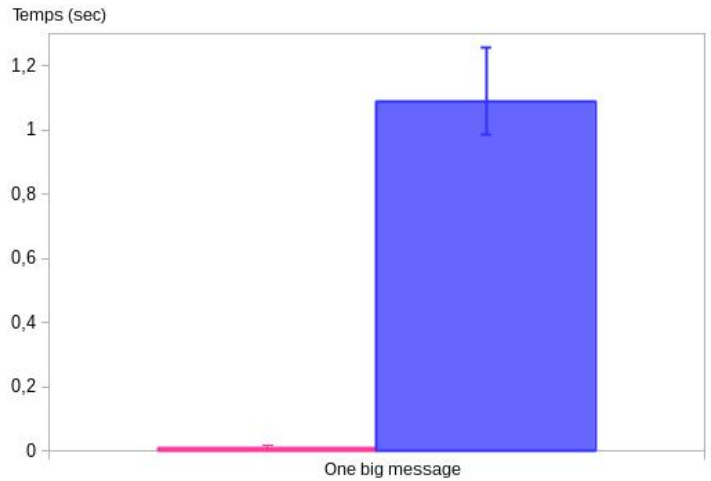


Full Mediation



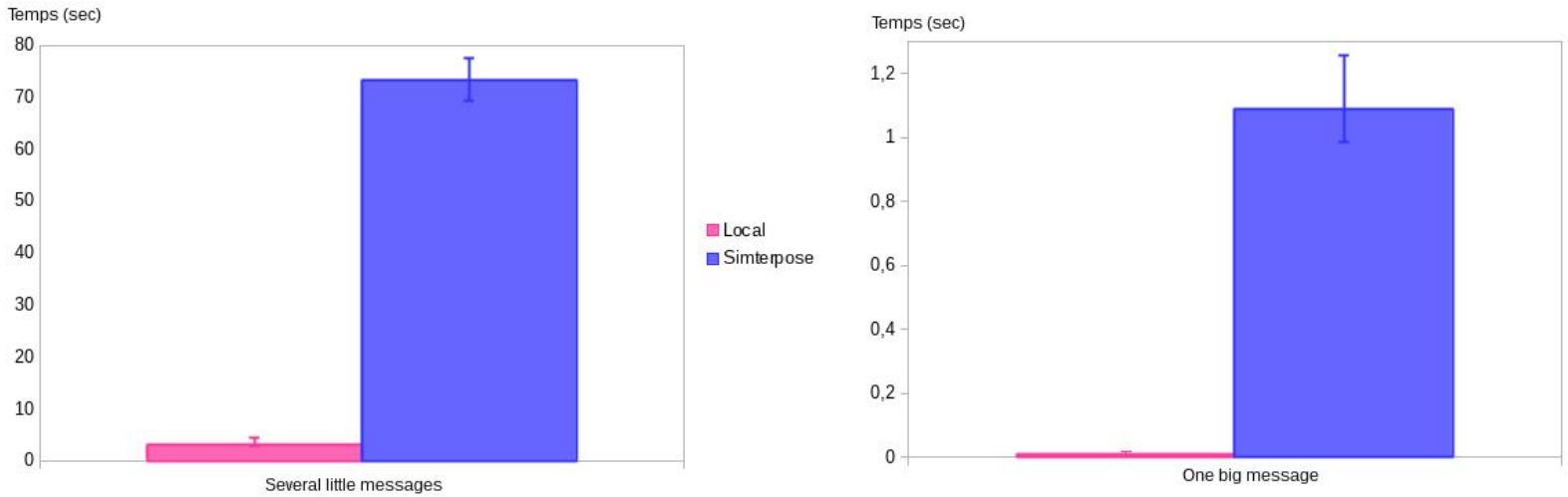
Overhead concernant le temps d'exécution





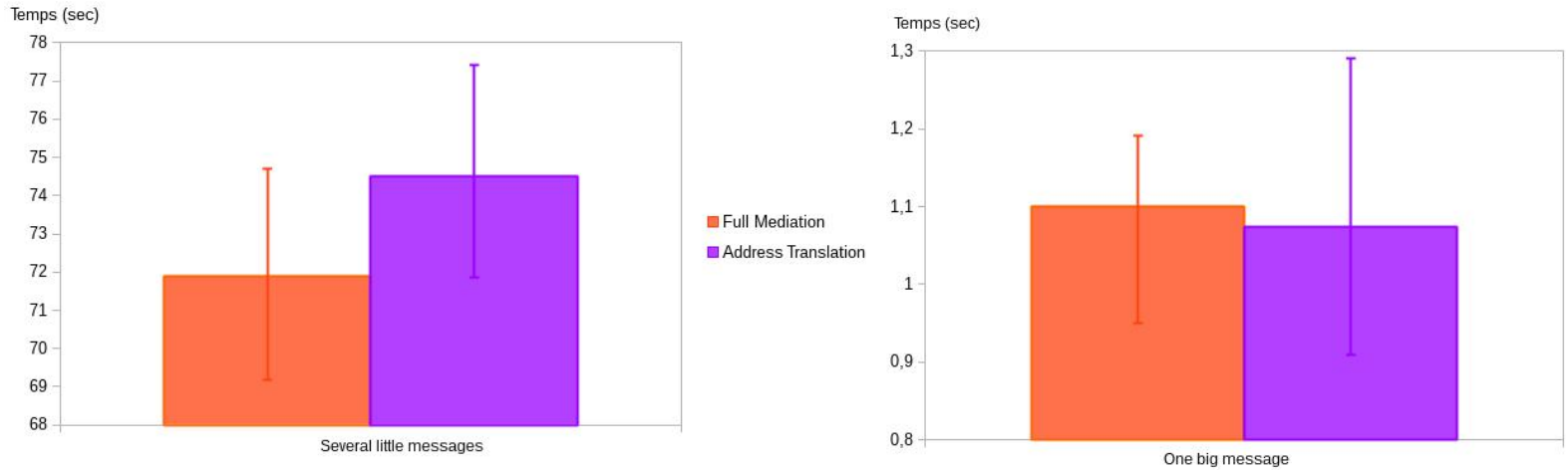
- Temps moyen d'exécution :
  - Gros message : 0.01s en local, 1,05s avec Simterpose
  - Petits messages : 3s en local, 73.5s avec Simterpose
- Analyse :
  - Nombreux appels systèmes et changements de contexte

Overhead concernant le temps d'exécution



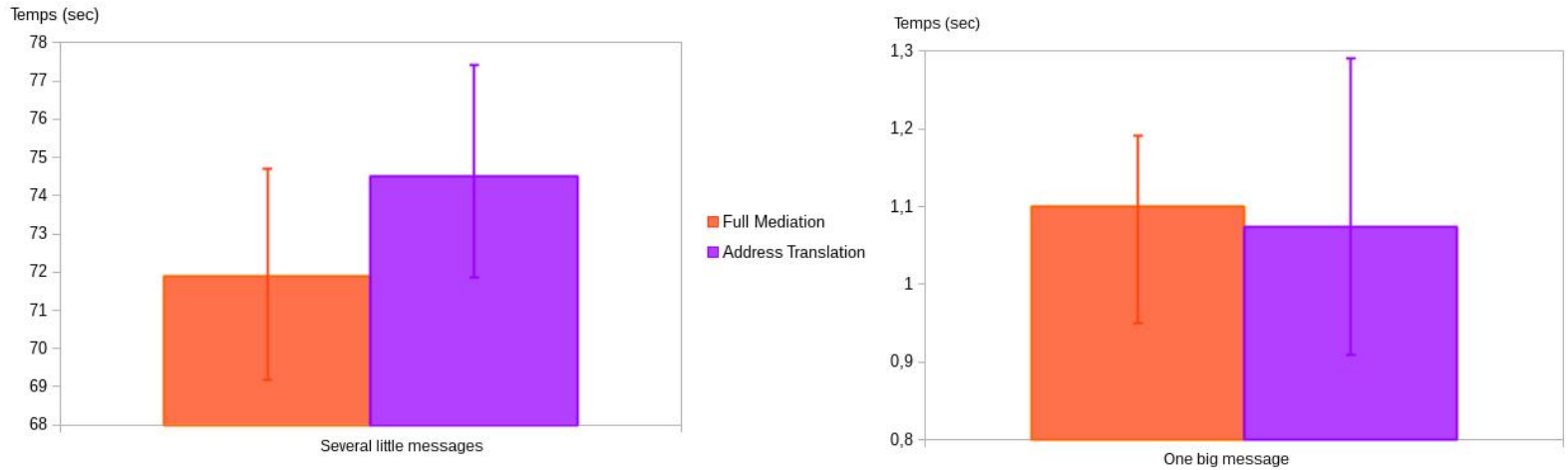
- Temps moyen d'exécution :
  - Gros message : 0.01s en local, 1,05s avec Simterpose
  - Petits messages : 3s en local, 73.5s avec Simterpose
- Analyse :
  - Nombreux appels systèmes et changements de contexte

Quelle médiation pour quel type d'application



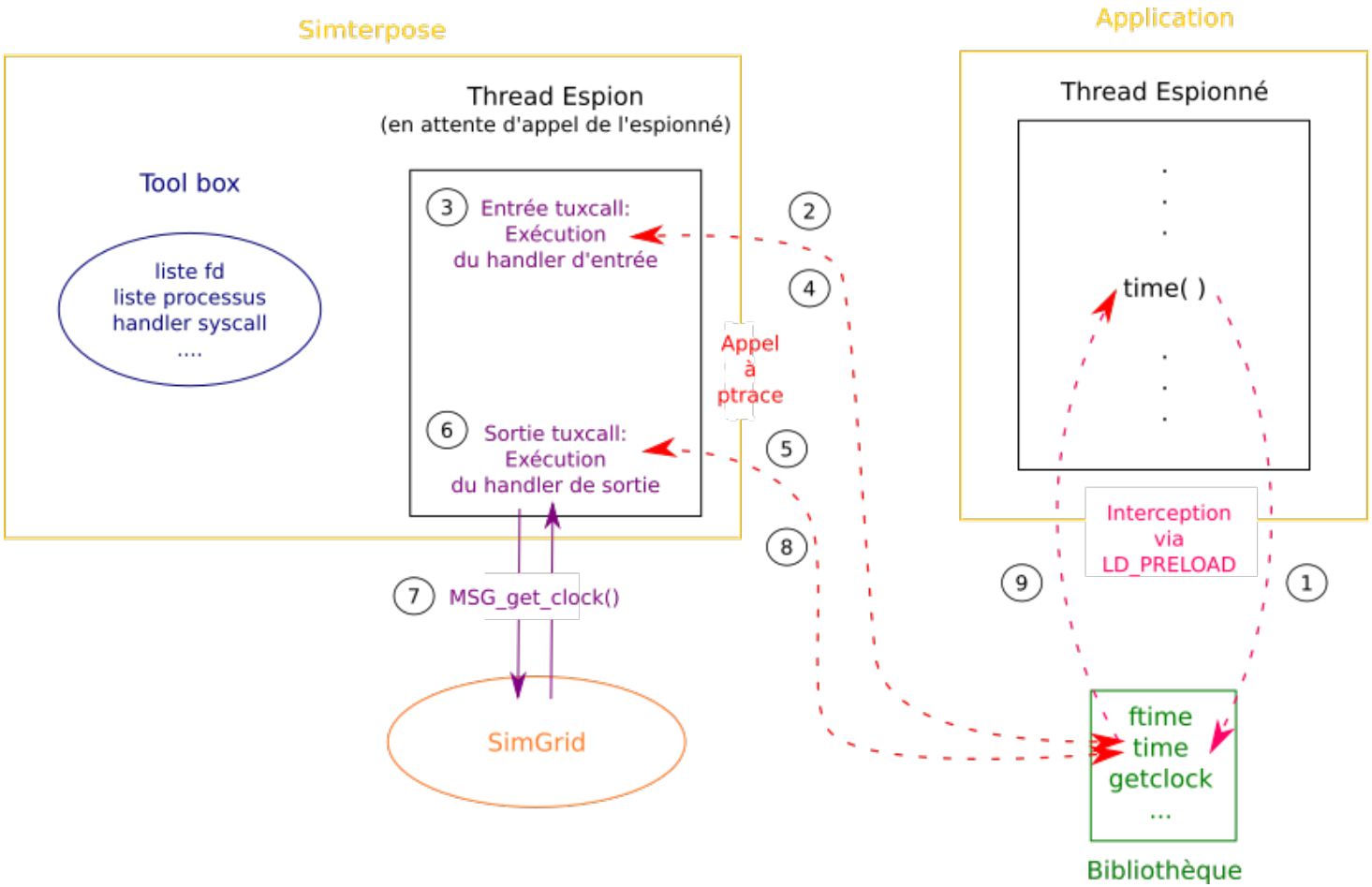
- Temps moyen d'exécution
  - Gros message : Écart de 2.5%
  - Petits messages : Écart de 3%
- Analyse
  - Petits messages : *full mediation* plus rapide car aucun appel système exécuté
  - Gros message : Gestion mémoire ralentie la *full mediation*

Quelle médiation pour quel type d'application

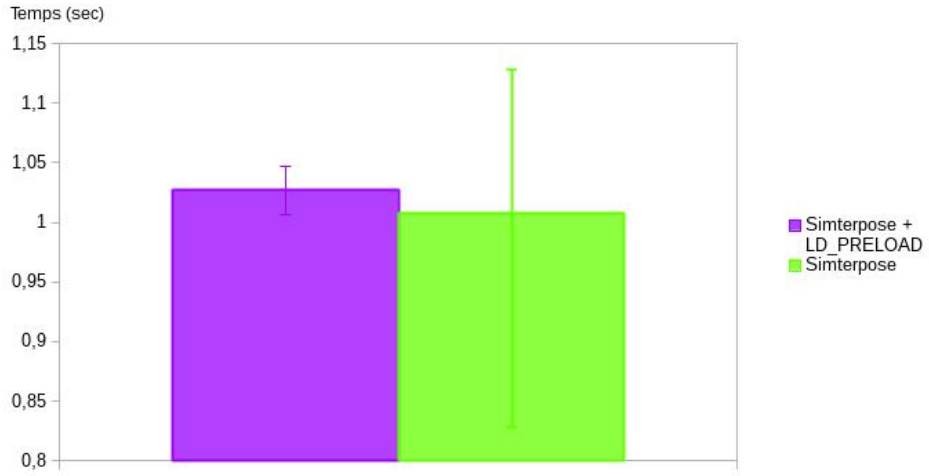


- Temps moyen d'exécution
  - Gros message : Écart de 2.5%
  - Petits messages : Écart de 3%
- Analyse
  - Petits messages : *full mediation* plus rapide car aucun appel système exécuté
  - Gros message : Gestion mémoire ralentie la *full mediation*

Gestion du temps

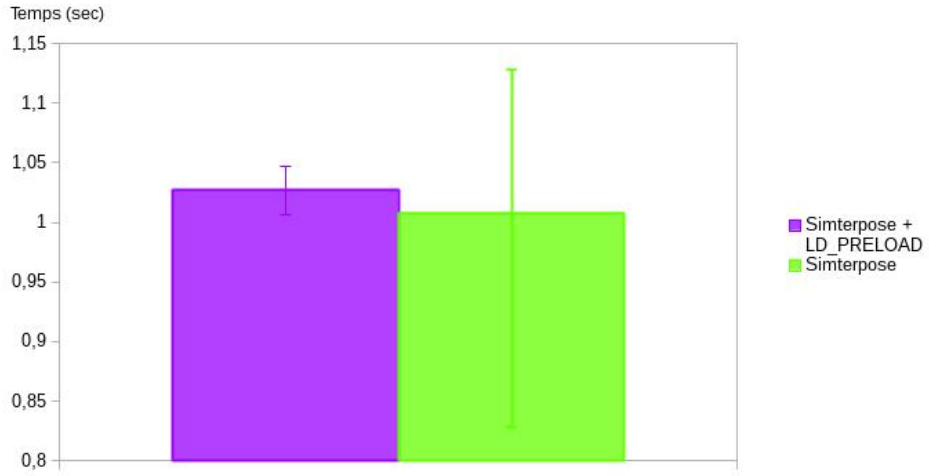


Gestion du temps



- Temps moyen d'exécution
  - Sans interception : 1s
  - Avec interception : 1.02s
- Analyse
  - Appel à la bibliothèque VDSO sans interception coûteuse en accès mémoire

Gestion du temps



- Temps moyen d'exécution
  - Sans interception : 1s
  - Avec interception : 1.02s
- Analyse
  - Appel à la bibliothèque `VDSO` sans interception coûteuse en accès mémoire

Améliorations

- Portabilité de Simterpose sur des architectures 32 bits
- Mise à niveau de la version de SimGrid utilisée
- Création d'un système de fichiers pour Simterpose
- Utilisation de Valgrind possible

Améliorations

- Portabilité de Simterpose sur des architectures 32 bits
- Mise à niveau de la version de SimGrid utilisée
- Création d'un système de fichiers pour Simterpose
- Utilisation de Valgrind possible

Améliorations

- Portabilité de Simterpose sur des architectures 32 bits
- Mise à niveau de la version de SimGrid utilisée
- Création d'un système de fichiers pour Simterpose
- Utilisation de Valgrind possible

Améliorations

- Portabilité de Simterpose sur des architectures 32 bits
- Mise à niveau de la version de SimGrid utilisée
- Création d'un système de fichiers pour Simterpose
- Utilisation de Valgrind possible

Pour finir

Conclusions

- 2 fonctionnalités implémentées
- Virtualisation possible pour les 2 fonctionnalités
- Améliorations apportées

Perspectives

- 2 fonctionnalités restantes
- Troisième type de médiation "accès direct"
- Nouvelles expériences
  - Influence du nombre de processus sur les performances et la mémoire
  - Tirage aléatoire des tailles de messages
  - Exécution avec BitTorrent

Pour finir

Conclusions

- 2 fonctionnalités implémentées
- Virtualisation possible pour les 2 fonctionnalités
- Améliorations apportées

Perspectives

- 2 fonctionnalités restantes
- Troisième type de médiation "accès direct"
- Nouvelles expériences
  - Influence du nombre de processus sur les performances et la mémoire
  - Tirage aléatoire des tailles de messages
  - Exécution avec BitTorrent

Annexe : Utilisation de Simterpose

- Pour une exécution simple (avec interception réseau uniquement)

  ```
  > sudo simterpose -s platform.xml deploiement.xml
  ```

- Pour une exécution avec en plus l'interception du temps

  ```
  > sudo LD_PRELOAD=lib.so -s platform.xml
  deploiement.xml
  ```

- Pour utiliser un débogueur

  ```
  > sudo simterpose "gdb -args" -s platform.xml
  deploiement.xml
  > sudo simterpose valgrind -s platform.xml
  deploiement.xml
  ```

Annexe : Interception temporelle

```
/* Macro to ask the clock to SimGrid */
#define get_simulation_time(clock) \
syscall(SYS_tuxcall, clock)

/* Time function */
time_t time(time_t *t){
  double* sec = (double *) malloc(sizeof(double));
  get_simulation_time(sec);
  if (t != NULL)
  t = (time_t *) sec;
  return (time_t)(*sec);
}
```

Annexe : Interception temporelle

```
void syscall_tuxcall(reg_s * reg,
    process_descriptor_t * proc){
  if (proc_entering(proc))
  proc_inside(proc); /* syscall_tuxcall_enter */
  else
  syscall_tuxcall_post(reg, proc);  /*
      syscall_tuxcall_exit */
}

void syscall_tuxcall_post(reg_s * reg,
    process_descriptor_t * proc){
  proc_outside(proc);
  XBT_DEBUG("tuxcall_post");
   /* Ask the clock to SimGrid */
  double clock = MSG_get_clock();

  /* Put the return value in argument register of
      the syscall */
  ptrace_poke(proc->pid, (void *) reg->arg[1], &
      clock, sizeof(double));
}
```

Architecture de SimGrid