I-JVM: une machine virtuelle Java pour l'isolation de composants dans OSGi

Nicolas Geoffray[1], Gaël Thomas[1], Gilles Muller[1], Pierre Parrend[2], Stéphane Frénot[3], Bertil Folliot[1]

nicolas.geoffray@lip6.fr

[1] Université Pierre et Marie Curie, INRIA/Regal
[2] FZI Karlsruhe
[3] INSA Lyon, INRIA/Amazones

Les composants en Java

- Qu'est-ce qu'un composant?
  - Ensemble de code composable
  - Browsers web (applets), Serveurs web (servlets), IDEs (plugins), OSGi (bundles)
- Utilisation des chargeurs de classe
  - Chargement/déchargement dynamique
  - Isolation de nom
  - Politiques de sécurité
  - Couplage dynamique: communications avec appels de méthode directs

Vulnérabilités des chargeurs de classe Java

- Pas d’isolation mémoire
  - Partage d’objets (static, String, Class)
- Pas de comptage de ressources
  - Utilisation mémoire
  - Utilisation CPU
  - Utilisation réseau
- Pas de terminaison non volontaire
  - Terminé lorsque non référencé

Solutions existantes

- Plusieurs JVMs
  - Un composant par JVM
  - Isolation système
  - Communications par appels RMI
- MVM (OOPSLA01, OOPSLA02)
  - Plusieurs applications dans une JVM
  - Isolation VM
  - Communications par Incommunicado

Solutions existantes

- Plusieurs JVMs
  - Un composant par JVM
  - Isolation système
  - Communications par appels RMI
- MVM (OOPSLA01, OOPSLA02)
  - Plusieurs applications dans une JVM
  - Isolation VM
  - Communications par Incommunicado

Incompatible avec les applications existantes

Solutions existantes

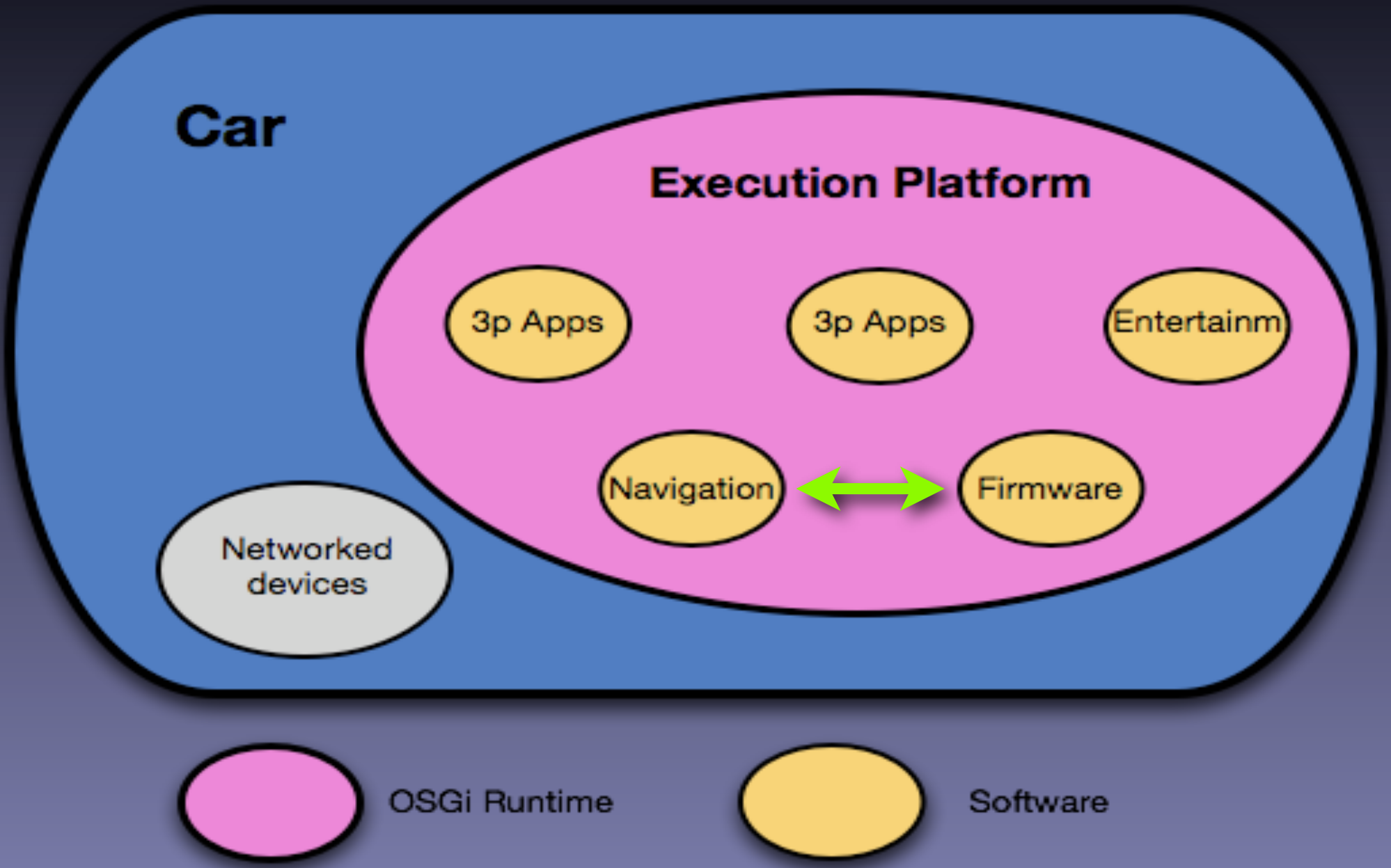
- Plusieurs JVMs
  - Un composant par JVM
  - Isolation système
  - Communications par appels RMI **(90ms)**
- MVM (OOPSLA01, OOPSLA02)
  - Plusieurs applications dans une JVM
  - Isolation VM
  - Communications par Incommunicado **(9ms)**

(Appel direct: 20µs)

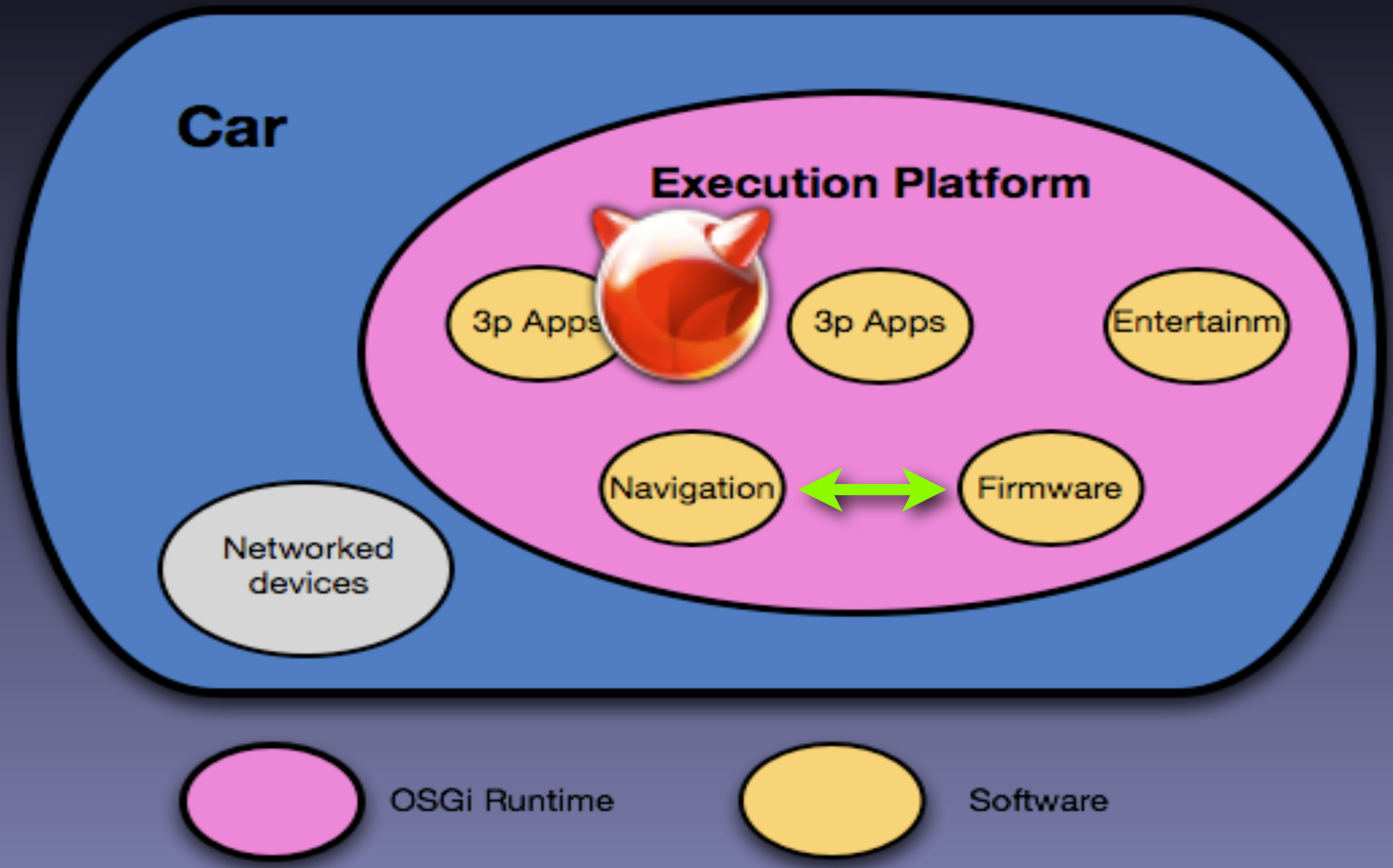
OSGi

- Plateforme d'exécution de bundles Java
  - Gestion du cycle de vie (installation, mise-à-jour, désinstallation)
  - Communications avec appels de méthode directs
- Domaines d'application
  - Automobiles, boxes internet, téléphones mobiles, IDE (Eclipse), Serveurs web (Jonas)

Un exemple OSGi



Problèmes de sécurité OSGi



Notre contribution

Un nouveau modèle d'isolation pour composants où la communication reste par appels directs

- ✓ Mémoire, ressources, terminaison
- ✓ Compatible avec l'existant
- ✓ Overhead léger lors des communications

Plan

1. Motivation
2. I-JVM
3. Résultats
4. Conclusions

I-JVM

- Un Isolate par Bundle avec droits limités
  - Isolation mémoire
  - Comptage de ressources
- Communication par appels directs
  - Threads “migrent” de contexte
  - Migration détectée à l’exécution
- Isolate0 pour le runtime OSGi
  - Peut créer/démarrer/arrêter des Isolates
  - Peut “tuer” un isolate

Isolate par bundle

- Isolation Mémoire (MVM [OOPSLA01])
  - Langage sûr avec Java
  - Variables globales dupliquées (static, string, Class)
- Comptage de ressources
  - Mémoire: algorithme GC (Price [OAKLAND03])
  - CPU: Test sur variable locale au thread
  - E/S: Modifcation du JRE (JRes [OOPSLA98])

Migration de Threads

- Appels Inter-Isolate
  - Appels directs
  - Même pile, changement de contexte
  - Objets passés par référence
- Chaque thread référence son isolate courant
  - I-JVM insère des tests à l'entrée de chaque méthode
  - L'isolate courant est mis-à-jour lors des appels inter-isolate

Terminaison d'Isolate

- Le runtime OSGi arrête les bundles *de manière volontaire*
  - Plus aucune dépendance
  - Plus de threads exécutant du code du bundle

Comment terminer des bundles non coopératifs?

Terminaison d'Isolate

1. I-JVM interrompt tous les threads
2. Les threads inspectent leur pile
    1. L'isolate interrompu est en fin de pile
    2. L'isolate interrompu est dans la pile
3. L'exécution reprend
    1. Retour vers l'isolate interrompu -> StoppedIsolateException
    2. Appel vers l'isolate interrompu -> StoppedIsolateException

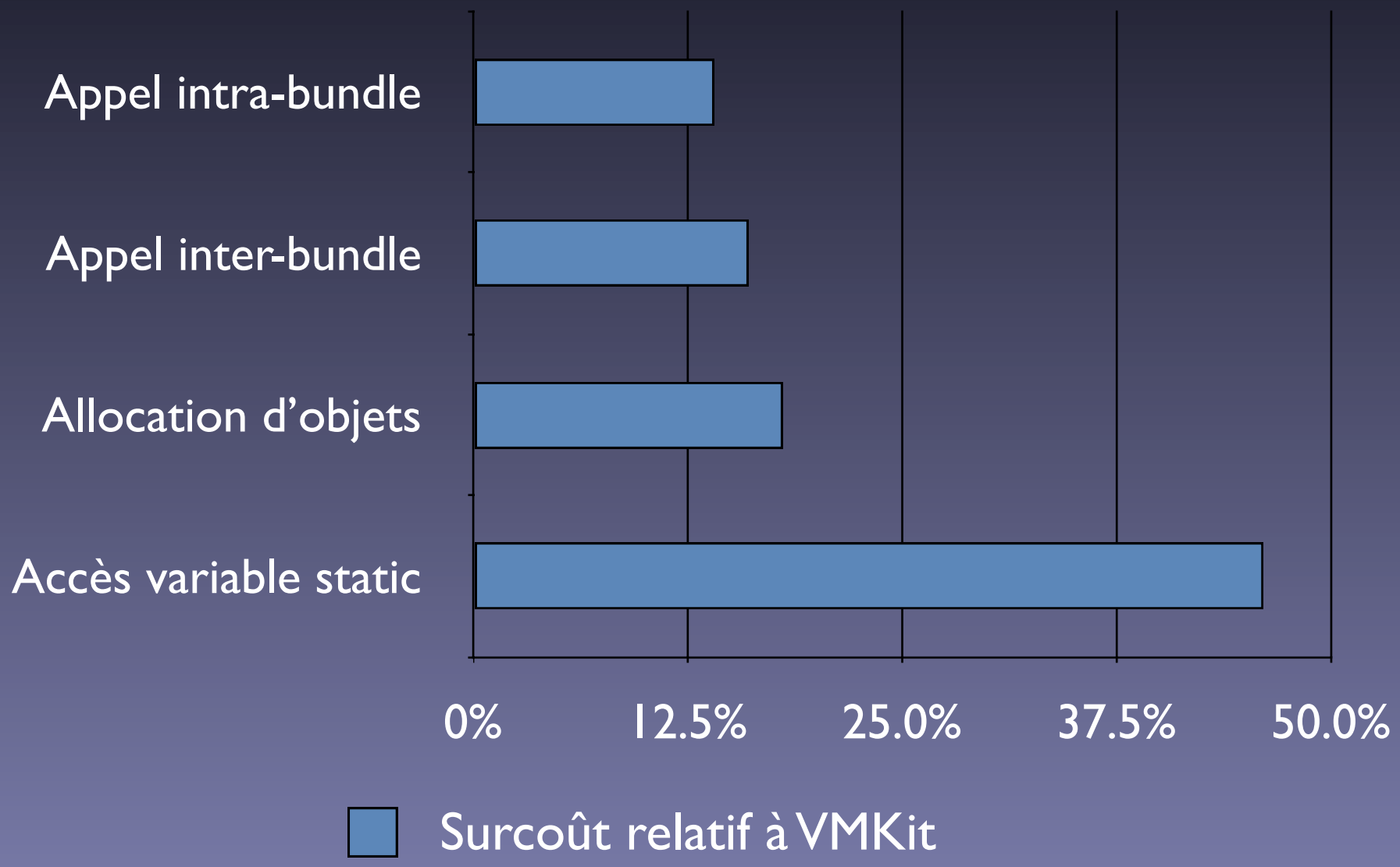
Implémentation d'I-JVM

- Codée avec VMKit (http://vmkit.llvm.org)
- Ajout de ~650 lignes de code
  - Isolation mémoire (~200 loc)
  - Tests de migration (~150 loc)
  - Comptage de ressources (~100 loc)
  - Isolate par bundle (~50 loc)
  - Terminaison d'Isolate (~150 loc)

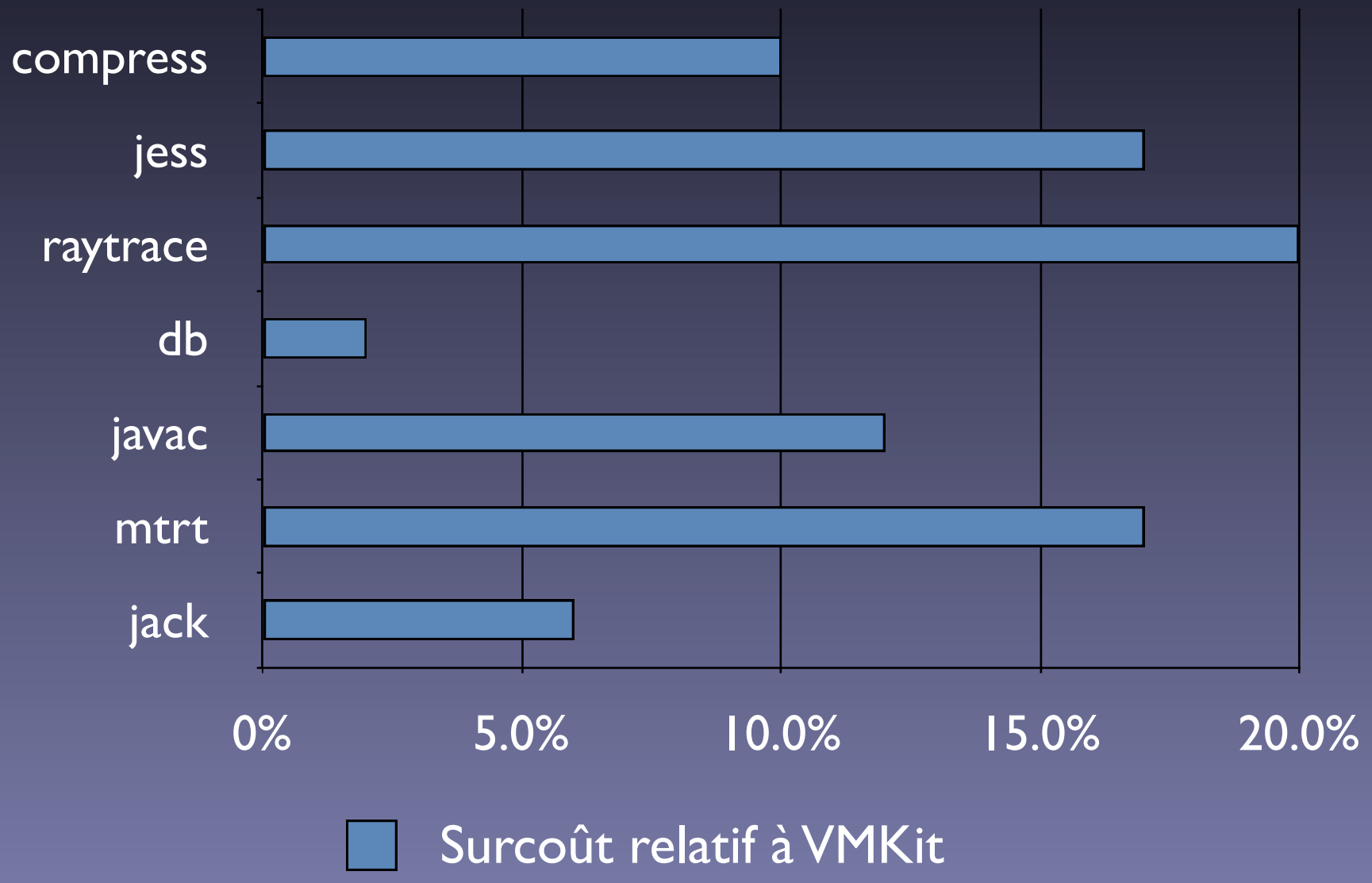
Plan

1. Motivation
2. I-JVM
3. Résultats
4. Conclusions

Surcoût à l'exécution

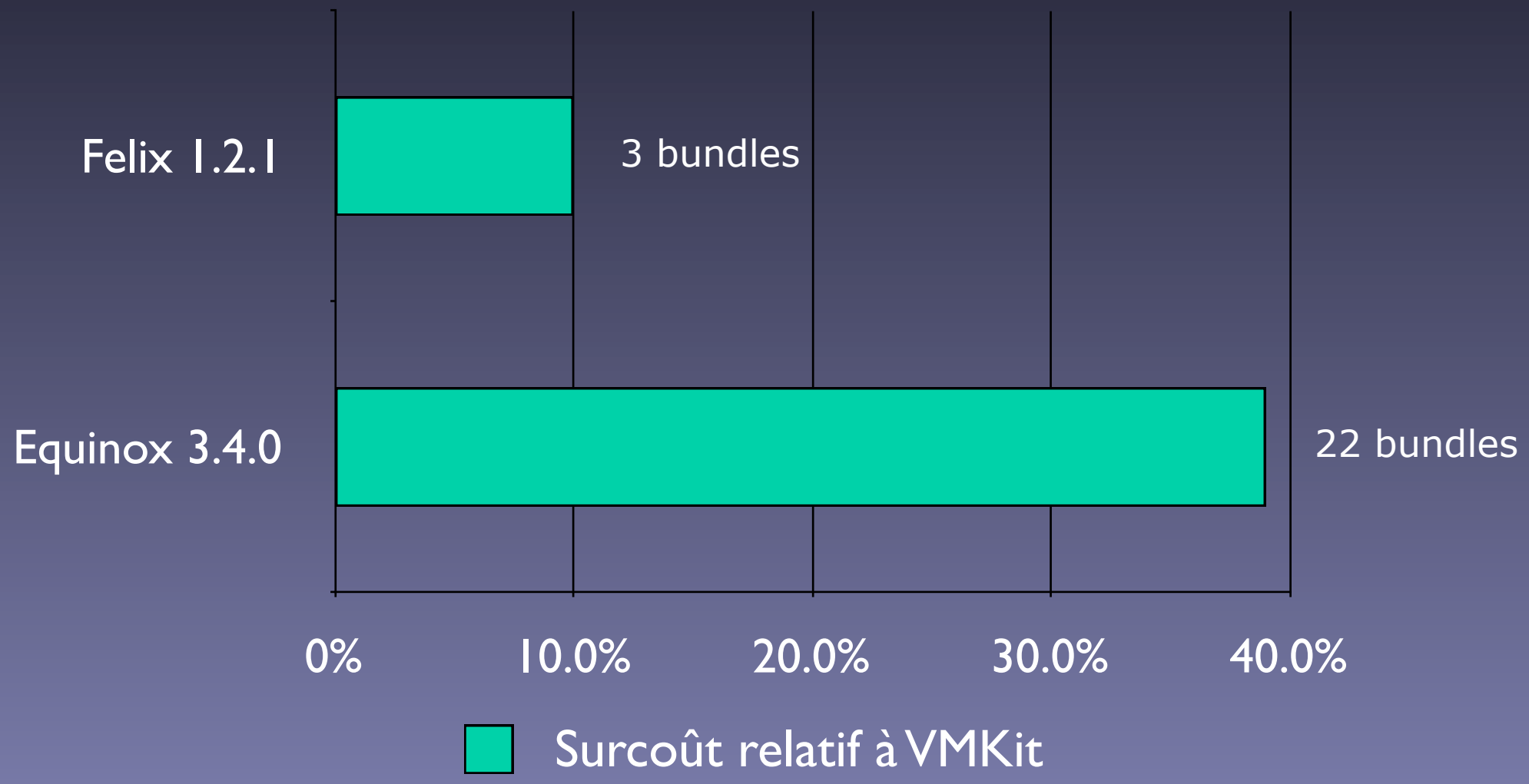


Surcoût à l'exécution



Surcoût mémoire

- Exécution de runtime OSGi avec les bundles de base (administration, shell, service de nommage, ...)



Vulnérabilités

Classification et attaques OSGi (Parrend [SPE09])

- Isolation mémoire
    - Modification d’une variable static
    - Synchronized sur une variable static
- Comptage de ressources
    - Sur-utilisation mémoire
    - Sur-utilisation CPU
    - Création d’objets excessive
    - Création de threads excessive
    - Thread bloqué

Vulnérabilités

Classification et attaques OSGi (Parrend [SPE09])

- Isolation mémoire
  - Modification d'une variable static
  - Synchronized sur une variable static
- Comptage de ressources
  - Sur-utilisation mémoire
  - Sur-utilisation CPU
  - Création d'objets excessive
  - Création de threads excessive
  - Thread bloqué

Par défaut avec I-JVM

Vulnérabilités

Classification et attaques OSGi (Parrend [SPE09])

- Isolation mémoire
  - Modification d'une variable static
  - Synchronized sur une variable static
- Comptage de ressources
  - Sur-utilisation mémoire
  - Sur-utilisation CPU
  - Création d'objets excessive
  - Création de threads excessive
  - Thread bloqué

Par défaut avec I-JVM

Remonté à l'administrateur

Conclusions

- Un nouveau modèle d'isolation
  - Pour plateformes orientées service
  - Surcoût léger
- Travaux futurs: système autonome
  - Dans ces travaux, comptage de ressources comme aide pour l'administration
  - Dans travaux futurs, écriture de politiques de ressource