# Pandora : une plate-forme efficace pour la construction d'applications autonomes

Simon Patarin et Mesaac Makpangou

DECOR'04

28 octobre 2004

# Problématique

- **Émergence de l'informatique autonome**
  - systèmes distribués complexes
  - environnement hétérogène
    - paramétrage difficile et éphémère
    - ↳ applications auto-(re)configurables

- **Support pour les applications autonomes**
  - flexibilité et auto-surveillance
    - ↳ besoin de réflexivité

# Solution proposée

- **Pandora**
  - modèle d'architecture original
    - composants minimaux et indépendants
    - communication par événements
  - facilité d'utilisation
    - support pour le déploiement, l'exécution et le contrôle des applications

- **Objectifs**
  - flexibilité **et** performance

# Plan

- **Travaux comparables**
  - ingénierie logicielle
- Modèle d'architecture
- Déployer, exécuter, contrôler
- Mise en œuvre
- Performances

# Ingénierie logicielle

- **Systèmes à composants**
  - systèmes patrimoniaux (EJB, CCM, .Net)
    - ✗ gros grain, reconfiguration dynamique très limitée
  - systèmes à grain fin (p. ex. OpenCOM, Gravity)
    - ✓ flexibilité accrue, meilleures performances
    - ✗ reconfigurations « simples » difficiles, surveillance limitée
- **Programmation par aspects**
  - tissage de propriétés non-fonctionnelles
    - ✓ à l'exécution (JAC) : reconfiguration dynamique possible
    - ✗ difficultés pour reconfigurer la logique métier

# Plan

- Travaux comparables
- **Modèle d'architecture**
  - modèle de composants
  - communication par événements
  - degrés de flexibilité
- Déployer, exécuter, contrôler
- Mise en œuvre
- Performances

# Modèle de composants

- **Composants indépendents à grain fin**
  - séparation logiques métier/non-fonctionnelles
  - réutilisation de composants génériques
  - surveillance facilitée
- **Trois niveaux d'abstraction**
  - composant : brique de base
    (code métier et non-fonctionnel)
  - pile : ensemble de composants intimement liés
    ≈ module
  - tâche : ensemble de piles communiquantes
    ≈ application

# Communication par événements

- **Pas d'interface ni d'invocation de méthode**
  - trop rigide compte-tenu du modèle précédent
  - simplification extrème des composants
  - favorise l'inter-opérabilité et la réutilisation
- **Deux types de communication**
  - inter-composants : synchrone, anonyme
  - inter-piles : asynchrone, nommé
  - ↳ laisse au concepteur d'application le choix du meilleur compromis

# Degrés de flexibilité

- **Flexibilité de paramétrage**
  - chaque composant peut être paramétré à l'exécution
    - ajustement simple du comportement de l'application
- **Flexibilité de composition**
  - choix du composant qui met en œuvre une fonctionnalité donnée
  - choix de la nature et de l'assemblage des composants qui forment un module
    - adjonction de propriétés non-fonctionnelles, extension de la plate-forme, modification d'une application en cours d'exécution

# Plan

- Travaux comparables
- Modèle d'architecture
- **Déployer, exécuter, contrôler**
  - gestion des ressources
  - exécution des piles
  - reconfiguration dynamique
- Mise en œuvre
- Performances

# Gestion des ressources

- **Nombreux fichiers de configuration**
  - définition des piles
  - localisation des bibliothèques de composants
- **Utilisation des ressources**
  - « meta » fichiers de configuration qui listent un ensemble de fichiers de configuration
  - utilisation de noms de fichier ou d'URL
  - possibilité de référencer d'autres ressources
  - ↳ **déploiement et maintenance d'une application à partir d'une seule URL**

# Exécution des piles

- **Entités de base pour l'exécution**
  - vérification d'intégrité statique
  - affectés à un processus léger
- **Micro-noyau**
  - ordonnancement des piles
  - gestion des composants
    - instanciation à la demande
    - liaisons dynamiques
    - terminaison (explicite ou temporisée)
    - ramassage des instances inaccessibles

# Reconfiguration dynamique

- **Interface entièrement réflexive**
  - accès à la configuration et à l'état du système
    - ressources, piles en cours d'exécution,…
  - possibilité de le reconfigurer dynamiquement
    - en particulier, ajout ou suppression de composants à une pile en cours d'exécution
- **Protocole à meta-objet**
  - exposition aux applications externes
  - interfaces disponibles en C, C++, Perl, Guile
- **Capteurs et moniteurs**
  - objets dédiés pour l'auto-surveillance

# Plan

- Travaux comparables
- Modèle d'architecture
- Déployer, exécuter, contrôler
- **Mise en œuvre**
  - protype fonctionnel
  - applications
- Performances

# Prototype fonctionnel

- **Mise en œuvre**
  - noyau et ~100 composants
  - 50 000 lignes de code en C++
  - porté sur de nombreuses architectures
    - Linux, FreeBSD, Solaris, Tru64 (Digital Unix)
    - Win32 pour le noyau et un sous-ensemble des composants
- **Disponibilité**
  - licence *open-source* INRIA, libre pour une utilisation non-commerciale

**http://www-sor.inria.fr/projects/relais/pandora/**

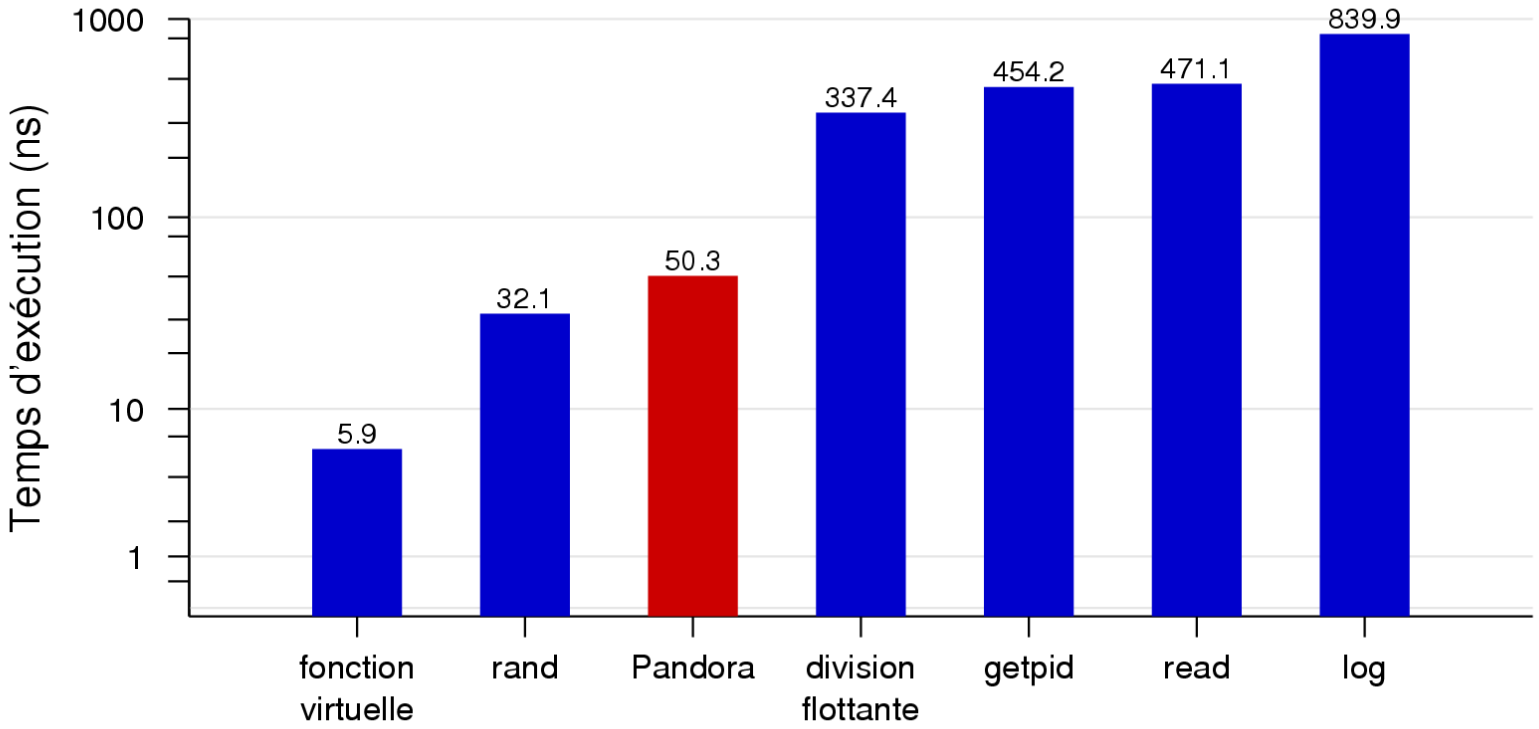
# Applications

- **Applications diverses construites avec Pandora**
  - surveillance réseau et système
  - métrologie des réseaux IP
  - analyse rétrograde de protocole
- **C/SPAN : cache Web adaptatif**
  - collaboration avec un cache Web flexible
  - adaptation du système entier à la charge et au trafic observé

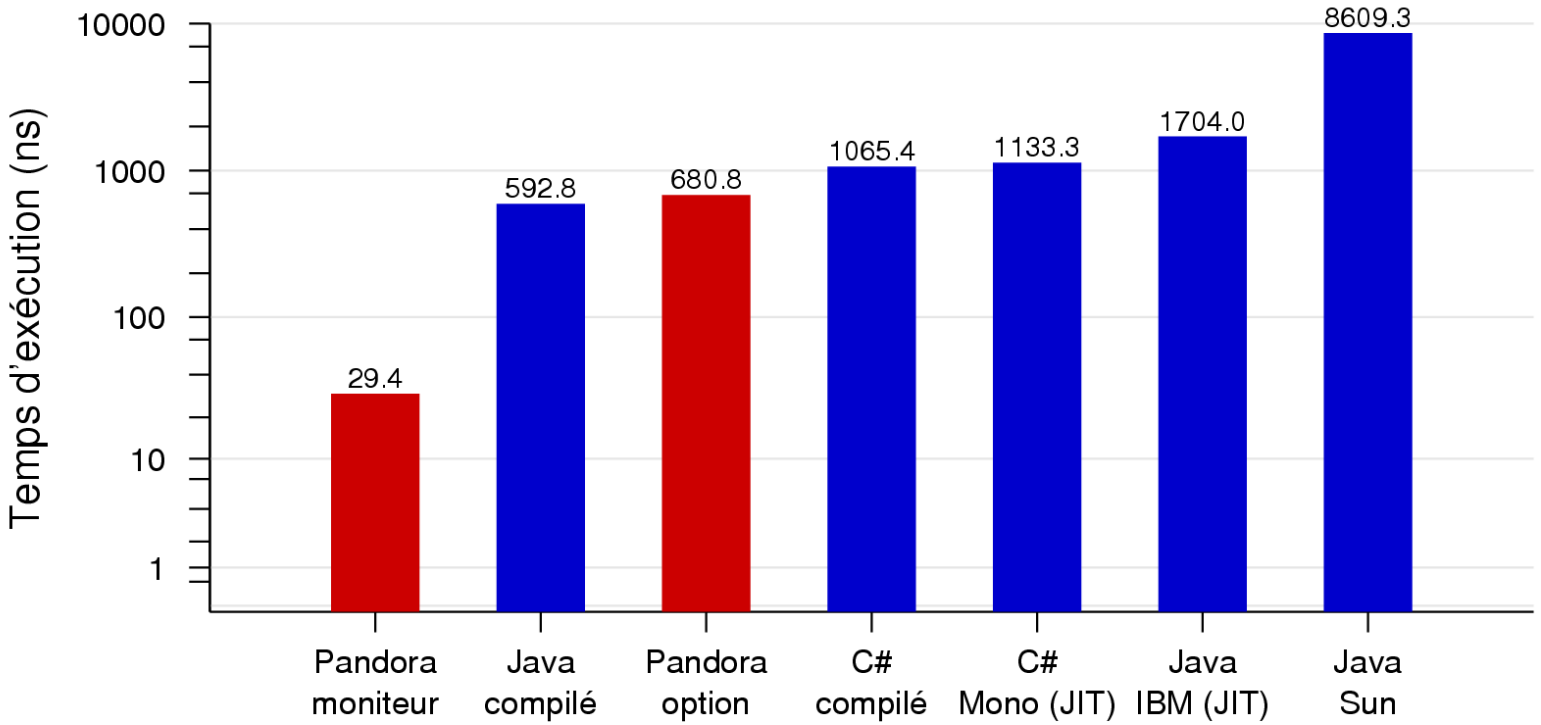
# Plan

- Travaux comparables
- Modèle d'architecture
- Déployer, exécuter, contrôler
- Mise en œuvre
- **Performances**
  - traversée de composants
  - introspection

# Traversée de composants



# Introspection



# Conclusion

## Pandora

- modèle d'architecture original
  - confluence d'approches éprouvées : systèmes à composants, langage de description d'architecture, programmation orientée aspects
- haute flexibilité et bonnes performances
- prototype disponible et déjà utilisé

## Limitations

- modèle relativement contraignant : portage d'applications existantes non trivial