MEAD : un intergiciel temps-réel et tolérant aux pannes pour CORBA

Michel Chilowicz

Master 2 Informatique Recherche – Université de Marne-la-Vallée

Vendredi 3 mars 2006

Plan

MEAD : Middleware for Embedded Adaptative Dependability

A propos de MEAD...

- Intergiciel pour CORBA associant temps-réel avec tolérance aux pannes.
- Développé à l'Université Carnegie Mellon de Pittsburgh.

Problématique abordée par MEAD

Temps-réel et tolérance aux pannes : propriétés antagonistes.
Comment concilier ces deux objectifs ?
$\rightarrow$ approche par interception sous ORB et prédiction de fautes.

Objectifs de MEAD

- Transparence pour adaptabilité aisée à des applications existantes.
- Gestion pro-active de fautes :
  - Défaillances du matériel.
  - Défaillances des processus.
  - Fautes de communication : altération des messages, partitionnement du réseau.
  - Limitation des fautes temporelles.
- Non prise en charge des pannes byzantines.

CORBA : Common Object Request Broker Architecture

- Standard de l'OMG créé en 1992.
- Architecture de distribution d'objets sur un réseau : communication avec ORB.
- Indépendance du langage de programmation (existence de compilateurs IDL pour de nombreux langages) et du protocole réseau.
- Pas de tolérance aux pannes, ni de temps-réel en standard.

Fault Tolerant-CORBA

Extension CORBA pour garantir la tolérance aux pannes.

- Réplication des objets CORBA sur différents hôtes (groupes d'objets transparents) : IOR $\rightarrow$ IOGR (Interoperable Object Group Reference)
- Services de surveillance et de notification de panne.

Fault Tolerant-CORBA (2) : modes de réplication supportés

- Non-réplication (équilibrage de charge) :
  - Pour les objets sans états (les arguments d'une invocation suffisent).
- Réplication passive :
  - Réplicat primaire répondant aux requêtes.
  - Réplicats secondaires de secours.
  - Maintien d'un journal de modifications.
  - Mises à jours régulières des réplicats secondaires (points de reprise) à partir du primaire.
- Réplication active :
  - Égalité de tous les réplicats : traitement de toutes les requêtes.
  - État identique des réplicats non-défaillants.
  - Mécanisme d'élimination de réponses multiples : vote sur les sorties (résistance aux fautes byzantines).

Real Time-CORBA

- Travaux menés par RTESS PTF (Real Time, Embedded and Specialized Systems Platform Task Force).
- Support d'un ordonnancement à priorité fixe (*Rate Monotonic*).
- Objectif : éviter les inversions de priorité, respecter les échéances.
- Nécessite une prédictabilité de la couche réseau.

Real Time-CORBA (2)

Gestion des priorités

- Association à chaque invocation d'une priorité CORBA (convertible en priorité système) définie par le client ou le serveur.
- Isolation du trafic réseau par bandes de priorités.
- Utilisation de réserves de threads pour les servants.
- Utilisation de mutex CORBA (avec protocole d'héritage de priorité).

Antagonisme du temps réel et de la tolérance aux pannes

Non-déterminisme des pannes : fautes temporelles

- Pannes : évènements asynchrones non-prédictibles.
- Procédures de récupération après panne (réplication passive) non prévues.

Multi-threading sur un système tolérant aux pannes

Maintien de la cohérence des réplicats $\rightarrow$ pas de multithreading, ni d'utilisation de *timers* locaux.

Prédiction de pannes

- Exploitation d'indices annonçant une panne future.
- Identification des scénarios annonciateurs de pannes ?
- Utilisation de techniques statistiques et heuristiques pour la prédiction avec degré de confiance.

Mise en place d'un prédicateur local de fautes relié à un prédicateur global.

Récupération pro-active

Principe

- Utilisation d'un prédicateur de pannes → consommation de ressources.
- Si une panne est prévue avec un haut degré de confiance dans un futur proche :
  - Utilisation du temps restant avant panne pour la migration des processus en danger par le prédicateur global avec récupération pro-active.
  - *Rajeunissement* de processus par redémarrage.
  - Création de nouveaux réplicats.

Avantage

Diminution de la latence sur récupération → Risques de fautes temporelles diminuées par rapport à une politique réactive.

Décision du déclenchement de récupération pro-active

Problématique

Quand déclencher la récupération ?

- Un déclenchement trop fréquent surcharge le système.
- Un déclenchement non-réalisé ou trop tardif nécessite une politique réactive $\longrightarrow$ menace de fautes temporelles.

Paramètres à considérer

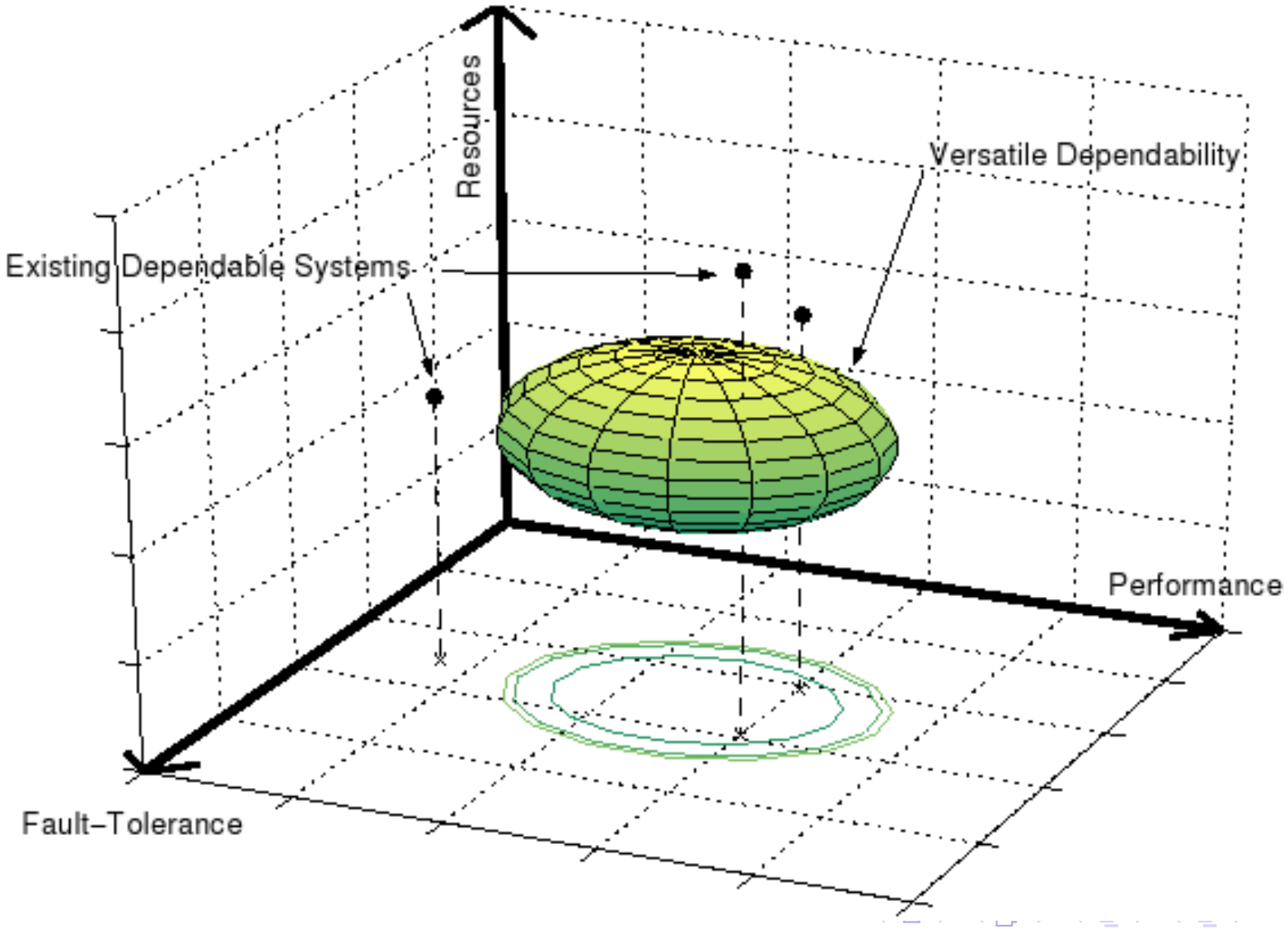
- Fréquence d'invocation des méthodes des objets.
- Fiabilité de la prédictabilité des pannes.
- Latence de récupération sur pannes.
- Délai des communications réseau.

Récupération pro-active pour un système avec réplication passive

Après prédiction d'une panne du processus primaire :

1. Diffusion de l'état interne du processus primaire $\rightarrow$ synchronisation des processus secondaires.
2. Élection du nouveau processus primaire.
3. Remplacement par le nouveau processus primaire.

Compromis entre performance, tolérance aux pannes et ressources



Paramètres d'arbitrage du compromis

Paramètres ajustables

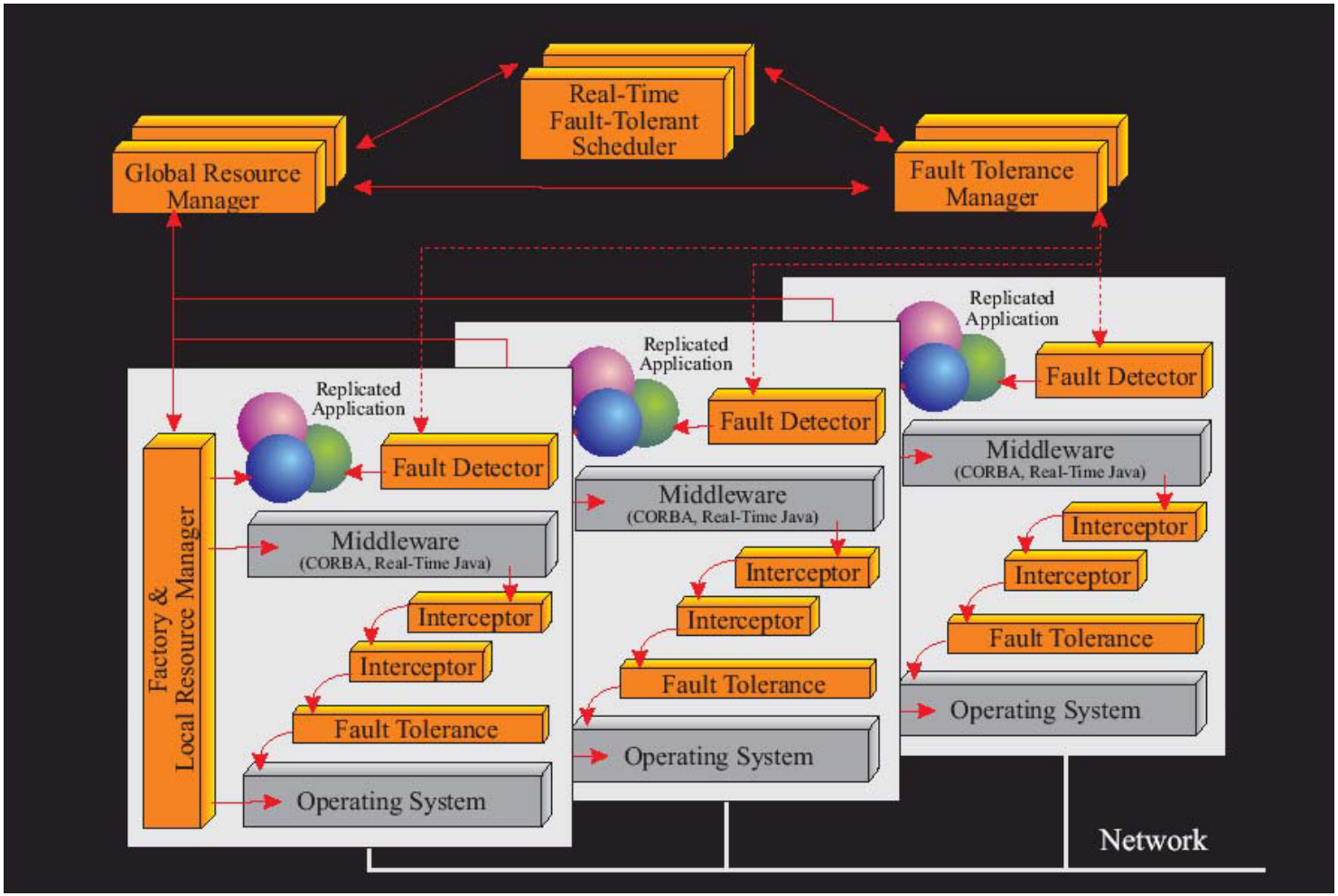
- Style de réplication (actif, passif, ...).
- Nombre de réplicats (passage à l'échelle).
- Fréquence de mise en place de points de restauration.

Paramètres applicatifs

Paramètres obtenus par analyse statique ou dynamique.

- Contraintes temporelles.
- Taille des requêtes et réponses $\longrightarrow$ trafic réseau.
- Fréquence des requêtes.
- Ressources utiles (complexité algorithmique).

Vue d'ensemble de l'architecture MEAD [3]



Interception

Objectif

- Transparence du support du temps-réel et de la tolérance aux pannes.
- Adaptation simple à des applications existantes.
- Utilisation d'un ORB standard.

Chaîne d'interception

Insertion de l'architecture MEAD sous un ORB standard par chaîne d'interception à l'exécution.
Chaînage des intercepteurs en parallèle ou série.
Inspiré de la programmation orientée aspect.

Gestionnaire de réplication

- Réplication des composants selon une politique définissable par le développeur.
- Aide à la définition de politiques de réplication par un outil de conseil de fiabilité (*reliability advisor*).
- Décide du nombre de réplicats, de leur localisation et du style de réplication.
- Utilisation de points de sauvegarde différentiels (moins de trafic réseau, récupération plus rapide).

Gestionnaires hiérarchiques de fautes

Détecteurs de fautes locales

Détecte et remonte dans la hiérarchie les fautes d'objets et processus.

Passage à l'échelle

Gestionnaires hiérarchiques : remontée ou non des fautes au gestionnaire de réplication.

Gestionnaires hiérarchiques de ressources

Organisation hiérarchique

Surveillance locale de l'utilisation de ressources par interception.
Remontée des informations aux gestionnaires de ressources globaux.

Gestionnaire de ressources global racine

Vue d'ensemble de l'utilisation de ressources.
Décisions d'équilibrage de charge : migration d'objets et processus.

Ordonnanceur temps-réel tolérant aux pannes (RT-FT)

Ordonnanceur hors-ligne

Analyse de faisabilité temps-réel et prévision d'ordonnancement en l'absence de fautes.

Ordonnanceur dynamique

Prédiction du temps de récupération dans le pire des cas.
Détermination des meilleurs moments pour la récupération.

Gestionnaire de politique de tolérance aux pannes pro-active

- Détecteur de fautes locales.
- Analyseur local de fautes.
- Gestionnaire global de politique pro-active :
  - Réception des rapports des analyseurs locaux.
  - Utilisation d'algorithmes de prédiction de pannes.
  - Décisions de migration préventive de processus.

Conclusion

Innovations de MEAD

- Conciliation du temps réel et de la tolérance aux pannes par politique pro-active.
- Utilisation transparente par interception.
- Sauvegarde d'états différentiels.
- Ajustement manuel ou automatique de paramètres pour assurer un compromis entre performances, ressources et tolérance aux pannes.

Conclusion (2)

Problèmes

- Surcoût introduit par l'interception.
- Surcoût de la politique pro-active.
- Difficulté de la prédictabilité des fautes.
- Non-support des pannes byzantines (projet Starfish).

Références


Tom Bracewell and Priya Narasimhan.
A middleware for dependable distributed real-time systems.
Technical report.


Tudor Dumitras and Priya Narasimhan.
An architecture for versatile dependability.
Technical report.


Priya Narasimhan.
Middleware for embedded adaptative dependability (MEAD).


Soila M. Pertet.
Proactive fault-recovery in distributed systems.
Master's thesis, Carnegie Mellon University, 2004.