http://www.ito.tu-darmstadt.de

Jini-Erfahrungen

Peer Hasselmeyer
Informationstechnologie Transfer Office
Technische Universität Darmstadt
peer@ito.tu-darmstadt.de

Überblick

Jini-Erfahrungen

- Projekte
- Programmierung
- Einsatz
- Geschwindigkeit
- Zusammenfassung

Projekte

Jini-Erfahrungen

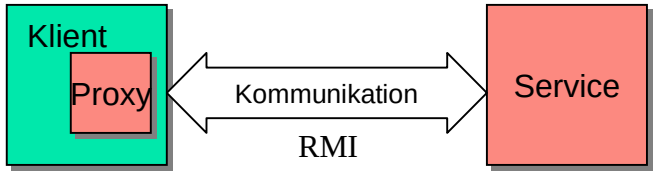
- Bosch Telecom: LCR
- Netzmanagement: „Nannies“
- Security
- graphischer Service-Browser („Jini Desktop“)
- Telekommunikation: „Virtuelle PBX”

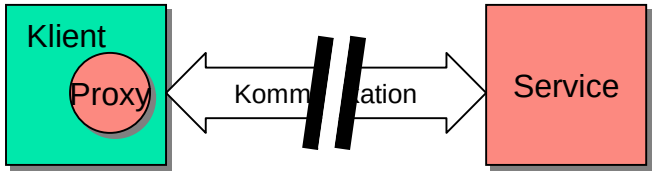
Programmierung

- Code-Transfer:
  - transferierte Klassen müssen serialisierbar sein, evtl. Factories
  - entweder „richtige“ Klasse oder generierter RMI-Stub
  - Klassen werden beim Klienten gecachet (Versionierung)
- Java2-Security:
  - RMI-Security-Manager

Realisierung

Jini-Erfahrungen





- 2 Möglichkeiten: RMI und lokal
- lokal:
  - keine Kommunikation benötigt
  - schneller
- aber:
  - größerer Code
  - Konsistenz und Synchronisation

Fehlertoleranz

- Besonderheiten:
  - Komponenten können fehlen
  - Komponenten können mehrfach vorhanden sein
  - Kommunikationsfehler können auftreten
- Intelligente Fehlerbehandlung unerläßlich
- ca. 10-30% zusätzlicher Code
- robusterer Code, mehr Funktionalität

Einsatz

Jini-Erfahrungen

- Java2-Security:
  - Policy-Files
- Code-Transfer mittels RMI:
  - große Abhängigkeit von Systemkonfiguration
  - HTTP-Server benötigt
  - Codebase-Property muß gesetzt werden
  - benötigte Klassen müssen ermittelt werden
- Classpath

Verhalten im Fehlerfall

Jini-Erfahrungen

- Abbruch von Diensten wird schnell erkannt
- Verlust der Konnektivität wird nach Timeout (TCP/IP) erkannt
- verschiedene Timeouts bei NT/Solaris/Linux

Weitere Erfahrungen

Jini-Erfahrungen

- Jini läuft sehr stabil
- Programmierung recht unkompliziert, teilweise repetitiv
- Leases nicht unter Klienten-Kontrolle: benötigen lange, bis sie auslaufen

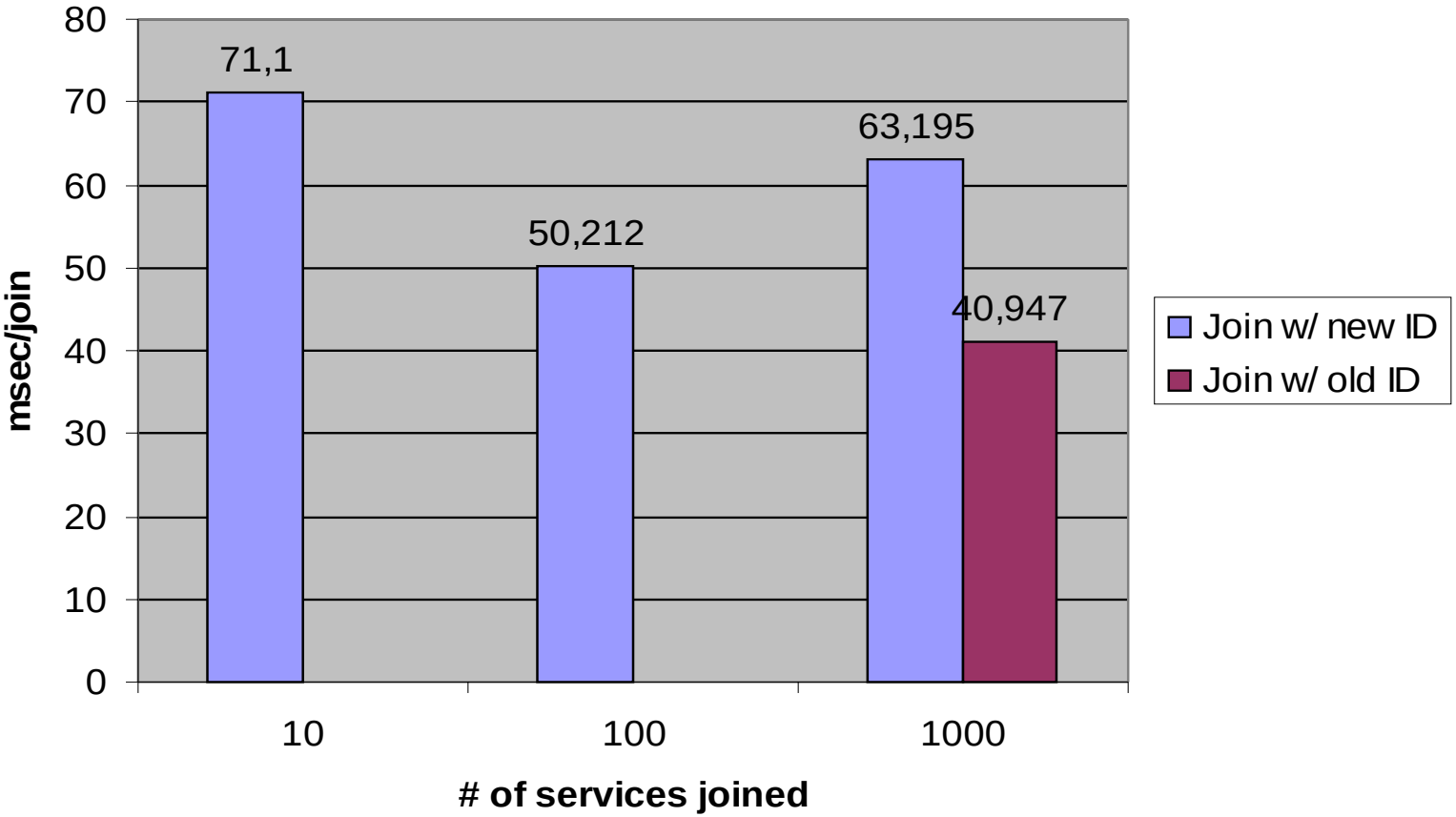
Geschwindigkeit I

- Einflüsse bei verteilter Benutzung:
  - langsamer bei verteilter Benutzung
  - aber: RMI auch lokal langsamer als “lokaler Proxy”
- Garbage Collector
- Finden des LUSes: Ø 1,3 sec

Geschwindigkeit II

Jini-Erfahrungen

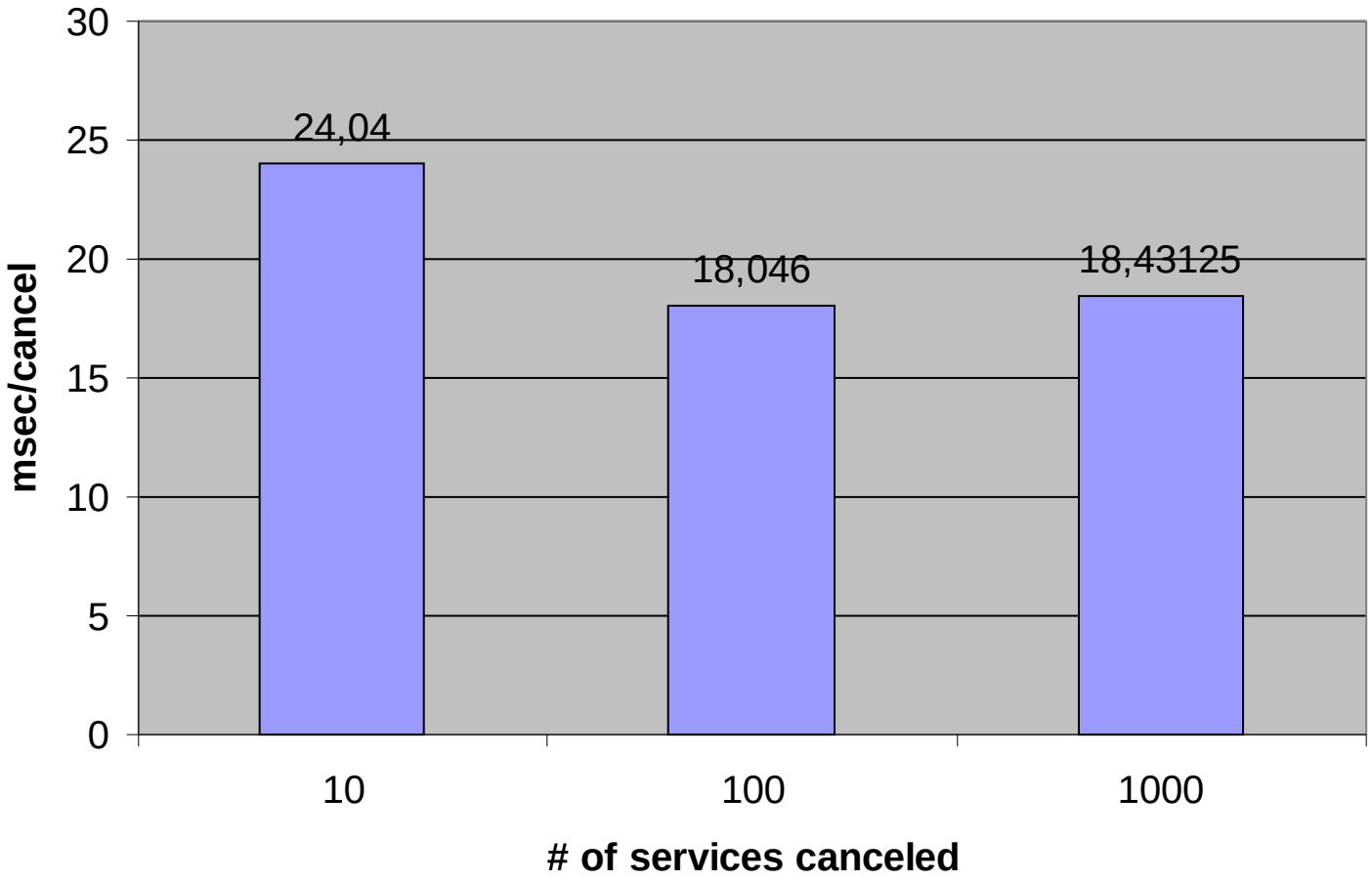
- Join-Geschwindigkeit



Geschwindigkeit III

Jini-Erfahrungen

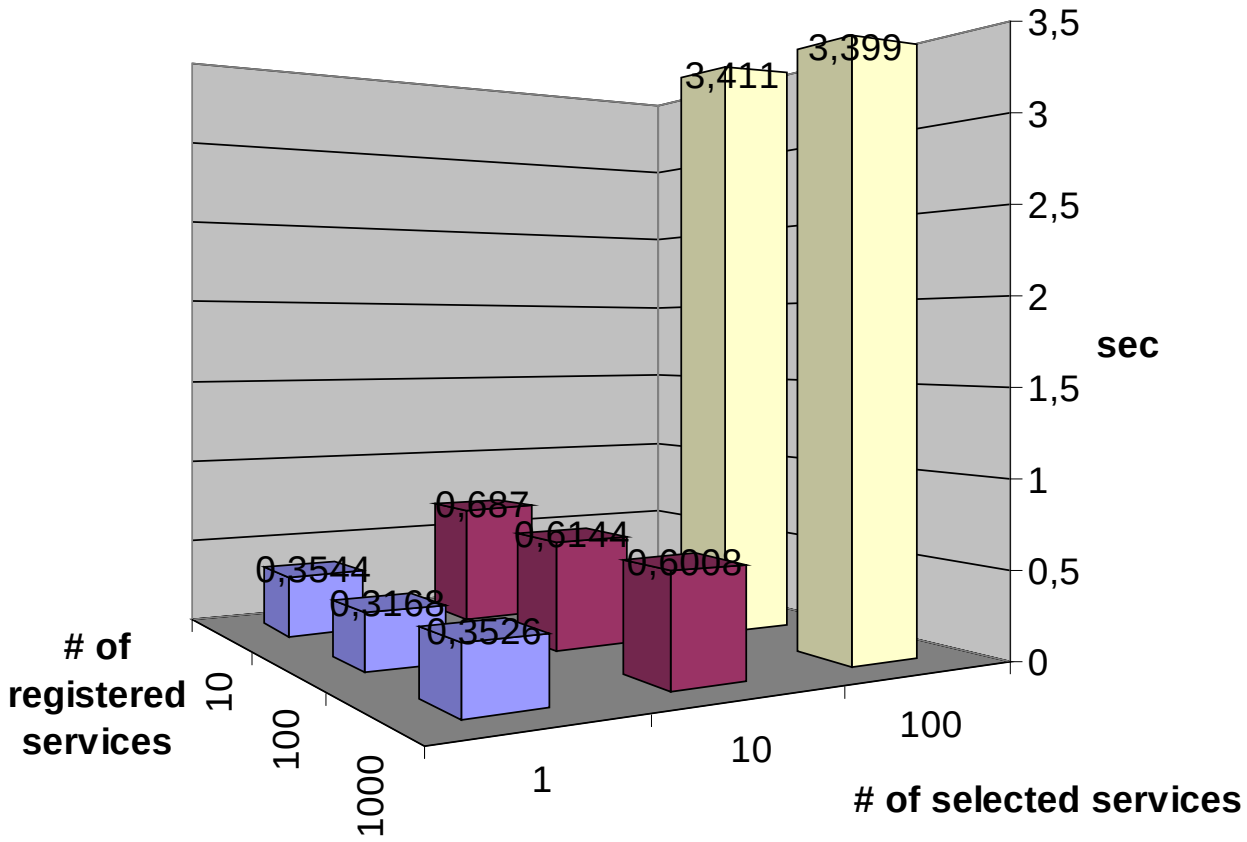
- Cancel-Geschwindigkeit



Geschwindigkeit IV

Jini-Erfahrungen

- Lookup-Geschwindigkeit



Speicherverbrauch

Jini-Erfahrungen

- Dienste/Klienten benötigen typischerweise eine JVM (>5MB)
- rmid benötigt viel Speicher (~15MB)
- Lookup-Service (~10MB)

Zusammenfassung

- Jini vereinfacht Programmierung verteilter Systeme
- Besonderheiten müssen beachtet werden: Codebase, HTTP-Server, etc.
- sinnvolle Fehlerbehandlung erfordert etwas Aufwand
- Probleme:
  - Einsatz in unterschiedlichen Umgebungen
  - großer Speicherverbrauch