



Ein Konzept zur Lastverwaltung in verteilten objektorientierten Systemen

Markus Lindermeier

Lehrstuhl für Rechnerarchitektur und Rechnerorganisation
Institut für Informatik
Technische Universität München

linderme@in.tum.de

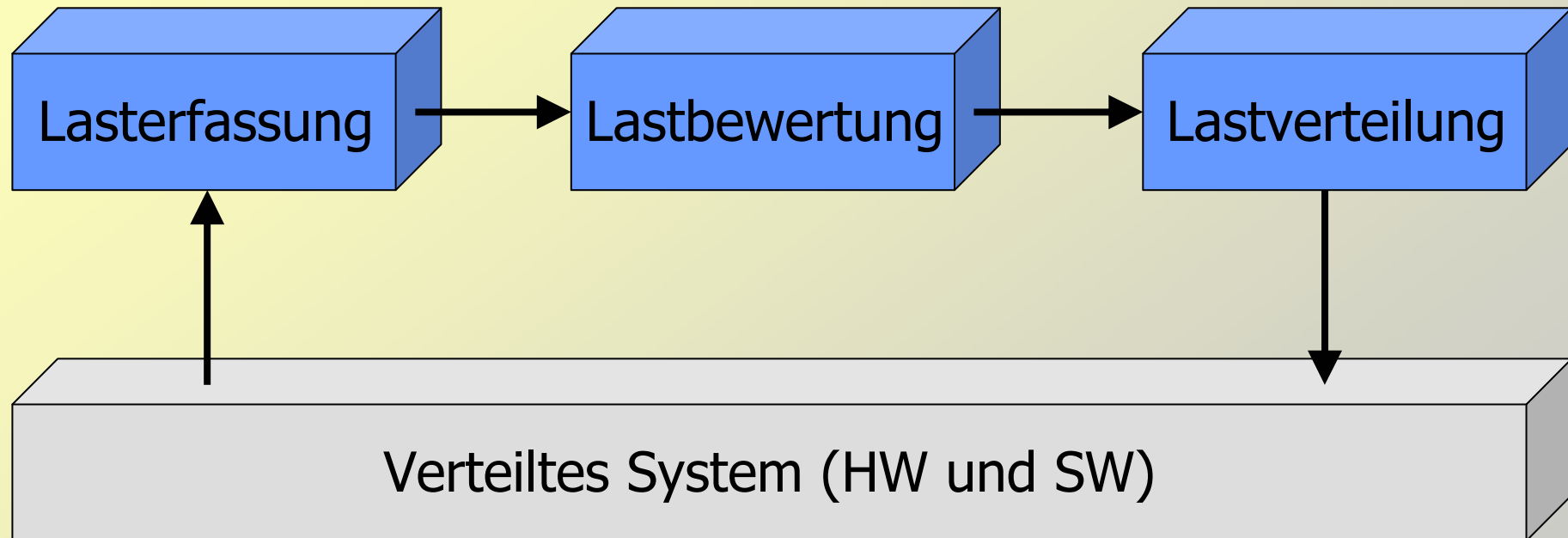
Motivation

- Lastungleichheit - Ein Problem in verteilten Systemen
 - Verteiltes System besteht aus Einzelkomponenten
 - Ungünstige Verteilung der Arbeitslast
 - Negative Auswirkung auf Laufzeitverhalten
- Lastverwaltungssysteme
 - Unterschiedliche Konzepte und Verfahren
 - Starke Abhängigkeit vom Programmiermodell
- Verteilte objektorientierte Systeme
 - Programmiermodelle (CORBA, DCOM, Java RMI)
 - Besondere Anforderungen und Möglichkeiten
 - Spezielles Lastverwaltungskonzept

Gliederung

- Architektur von Lastverwaltungssystemen
- Merkmale verteilter objektorientierter Systeme
- Kernpunkte des neu entwickelten Lastverwaltungskonzepts
- Das Lastmodell und der Lastbewertungsalgorithmus
- Das Lastverwaltungssystem LMC
- Ein Anwendungsbeispiel
- Zusammenfassung und Ausblick

Architektur von Lastverwaltungssystemen



- Komponenten eines Lastverwaltungssystems
- Komponenten bilden einen Regelkreis



Besondere Merkmale verteilter objektorientierter Systeme

- Einheiten der Lastverteilung
 - Prozesse, Daten, Transaktionen, ...
 - Objekte und Anfragen als Lastverteilungseinheiten
- Offene Client-Server-Architektur
 - Überlast durch Hintergrundprozesse (Hintergrundlast)
 - Überlast durch zu viele Anfragen (Anfragelast)
- Transparenzanforderungen des Programmiermodells
 - Orts- und Zugriffstransparenz
 - Lastverwaltung erfordert auch transparente Lastverteilung
- Schichtung verteilter objektorientierter Systeme
 - Rechensystem und Anwendung
 - Verteiltes Ablaufsystem (Middleware)



Kernpunkte des neu entwickelten Lastverwaltungskonzepts

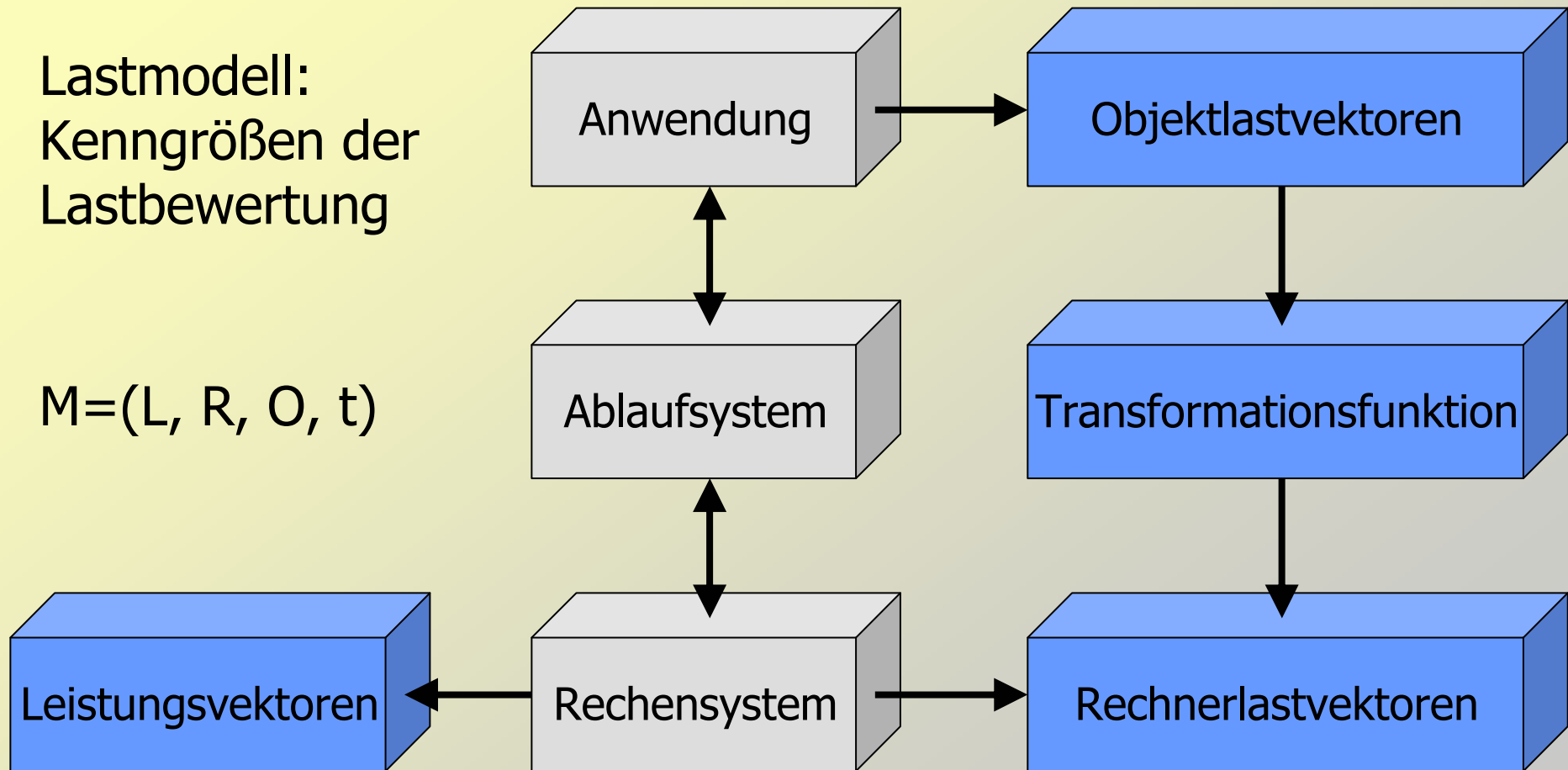
- Objekte und Anfragen als Lastverteilungseinheiten
 - Zustandslose Objekte
 - Zustandsbehaftete Objekte
- Kombination mehrerer Lastverteilungsmechanismen
 - Initialplatzierung
 - Migration zur Kompensation von Hintergrundlast
 - Replikation zur Verteilung von Anfragelast
- Transparenz der Lastverteilung
 - Migrationstransparenz
 - Replikationstransparenz
- Mehrschichtiges Lastmodell



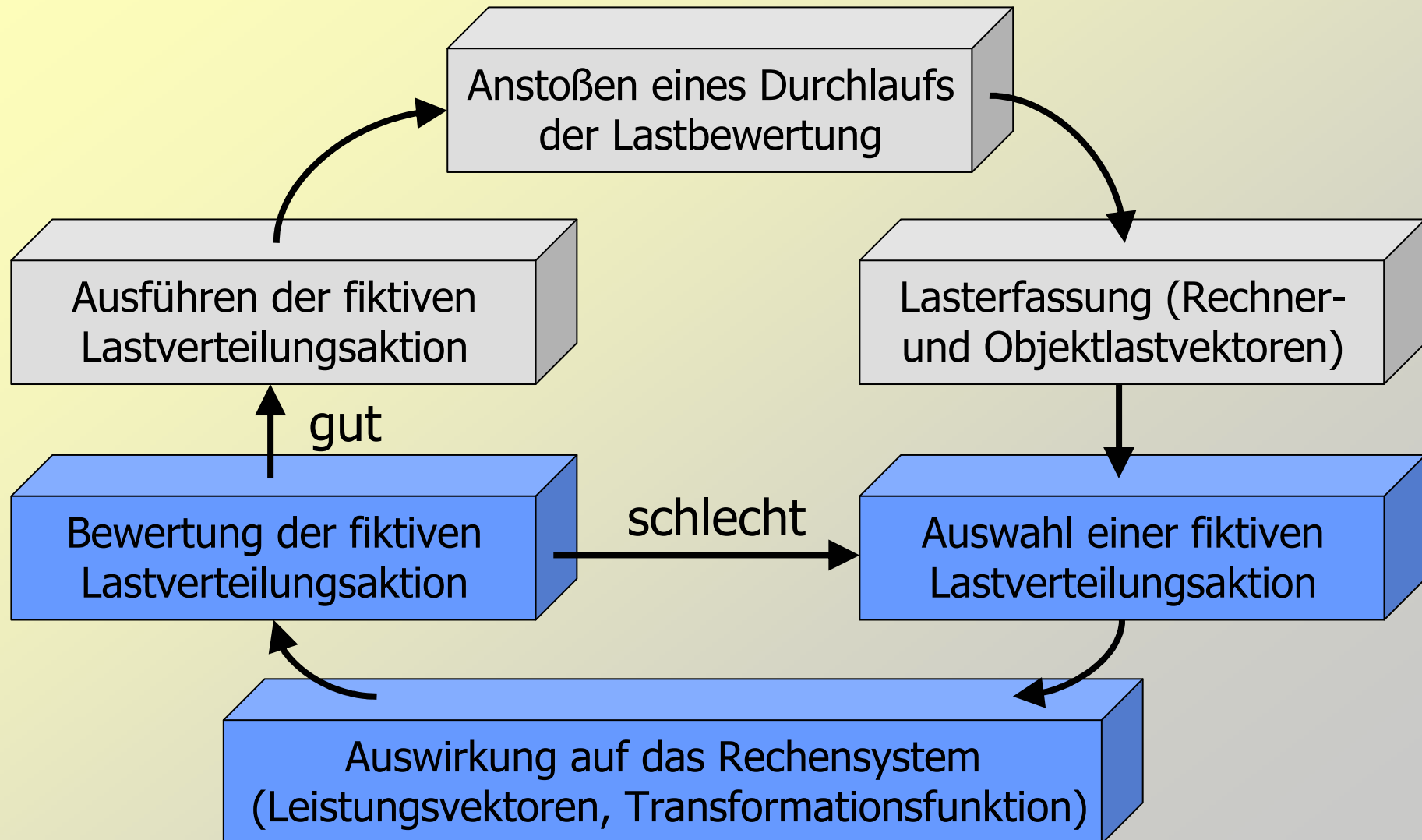
Ein Lastmodell für verteilte objektorientierte Systeme

Lastmodell:
Kenngrößen der
Lastbewertung

$M=(L, R, O, t)$



Der Algorithmus zur Lastbewertung



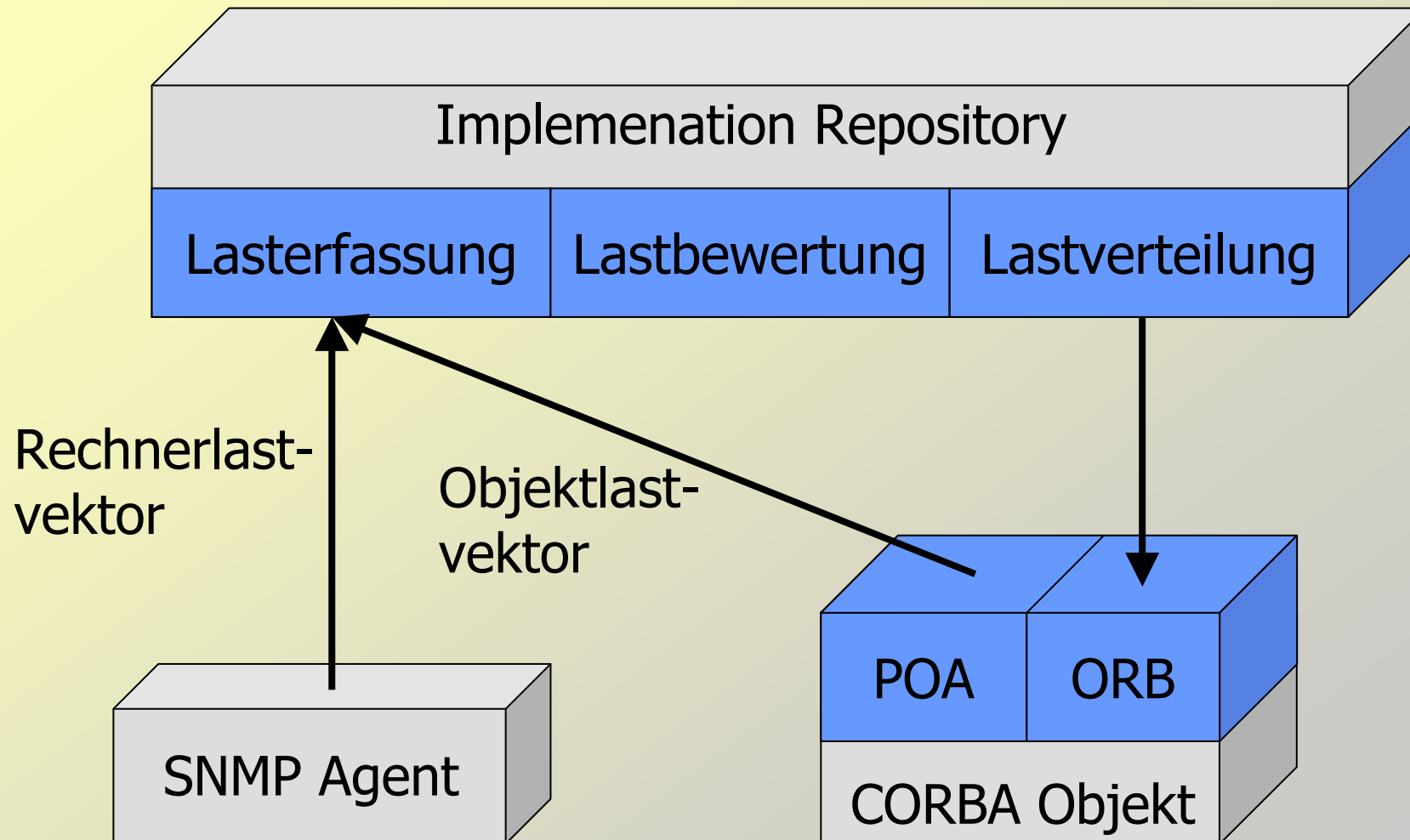


Das Lastverwaltungssystem LMC – Load Managed CORBA

- Lastverwaltungssystem für CORBA
- Integration in das CORBA-Laufzeitsystem
 - Migrationstransparenz
 - Replikationstransparenz
- Erweiterung des CORBA Standards
 - Erzeugen/Aktivieren von Objekten
 - Objektpersistenz (zustandsbehaftete Objekte)
- Nahtlose Integration in das CORBA-Programmiermodell
 - Erweiterung der ORB- und POA-Klassen
 - Abwärtskompatibilität zum CORBA Standard
 - Geringer Mehraufwand für den Entwickler

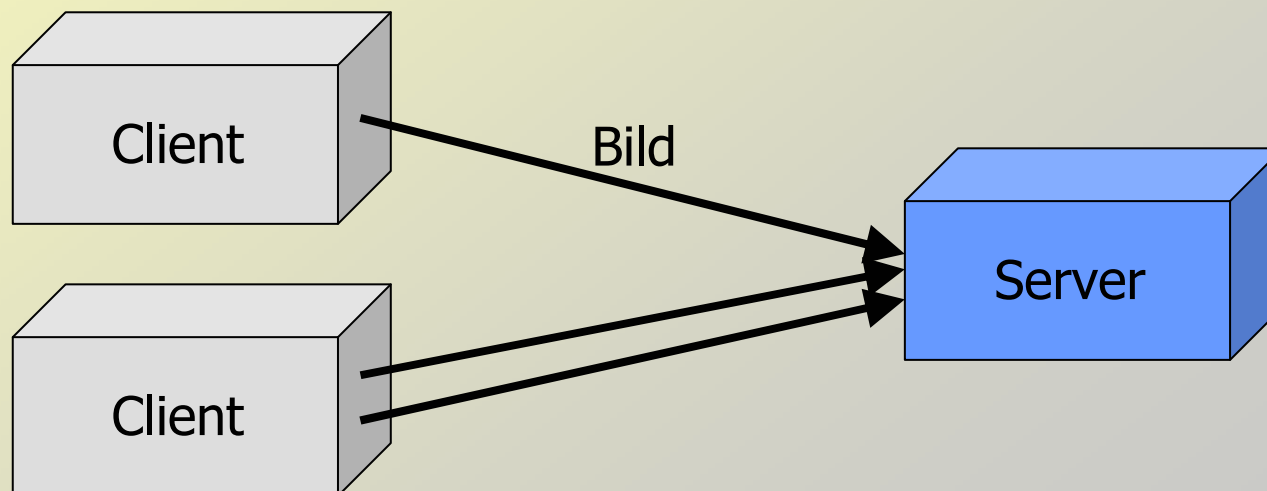


LMC – Architektur und Implementierung

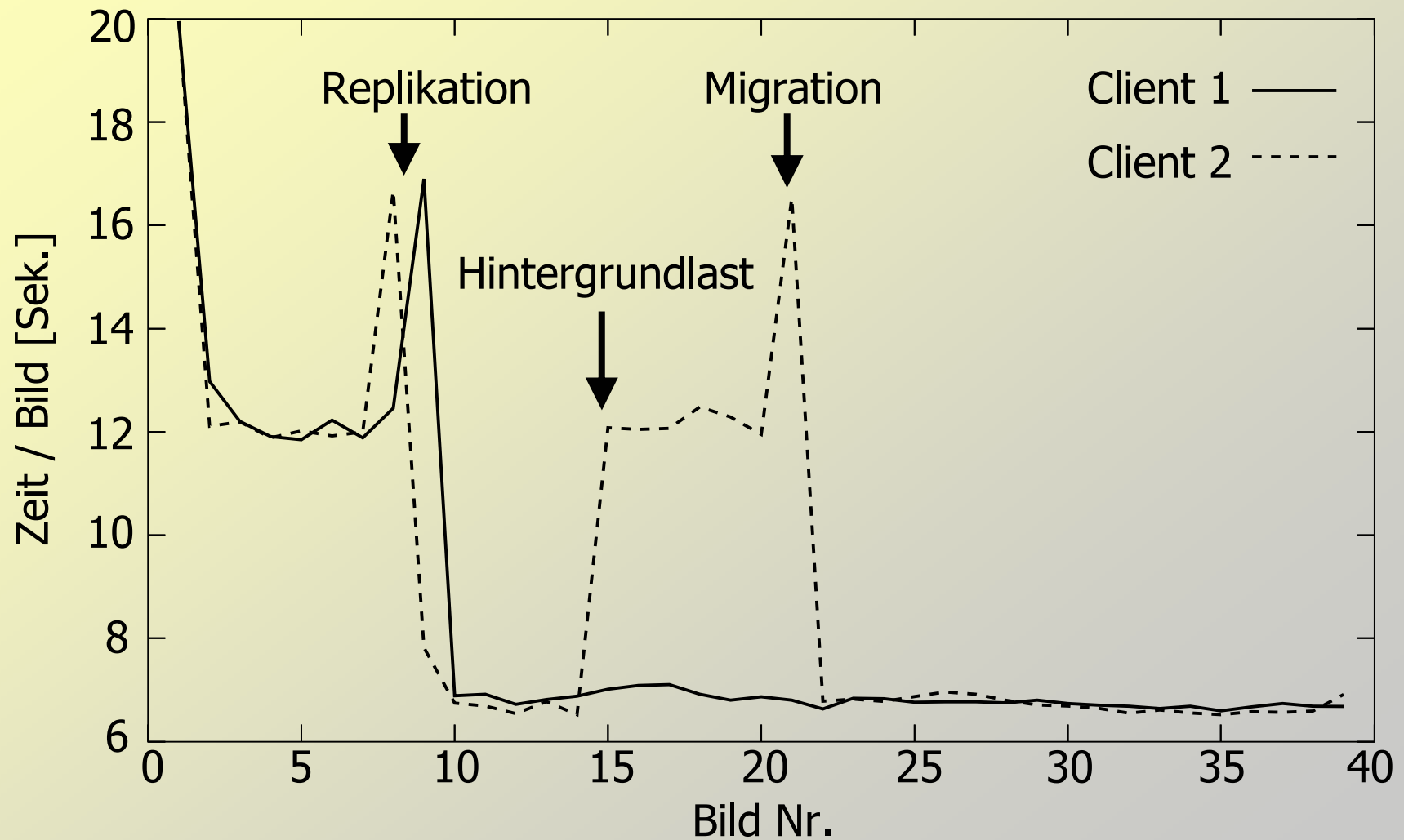


Ein Anwendungsbeispiel aus der medizinischen Bildverarbeitung

- Programmpaket SPM (Statistical Parametric Mapping)
 - Analyse der Gehirnaktivität mittels Computer-Tomographie
 - Aufbereitung von PET und MRT Bildsequenzen
- Realignment-Komponente
 - Korrektur der Bewegungen der Versuchsperson
 - Ausrichten aller Bilder einer Sequenz am 1. Bild



Die Realignment-Anwendung



Zusammenfassung und Ausblick

- Spezielles Lastverwaltungs-konzept erforderlich
 - Kombination mehrerer Lastverteilungsmechanismen
 - Transparenz der Lastverteilung
 - Mehrschichtiges Lastmodell
- Prototypische Implementierung für CORBA zeigt Anwendbarkeit und Nutzen dieses Konzepts
- Ausblick für künftige Arbeiten
 - Überschneidungen zwischen Lastverwaltung und anderen Diensten (Wegewahl, Ein-/Ausgabe, Fehlertoleranz)
 - Bisher werden diese nur getrennt betrachtet
 - Synergieeffekte durch kombinierte Betrachtung