TECHNISCHE UNIVERSITÄT CHEMNITZ

Management-Elemente für mehrdimensional heterogene Cluster

Karsten Petersen

kapet@informatik.tu-chemnitz.de

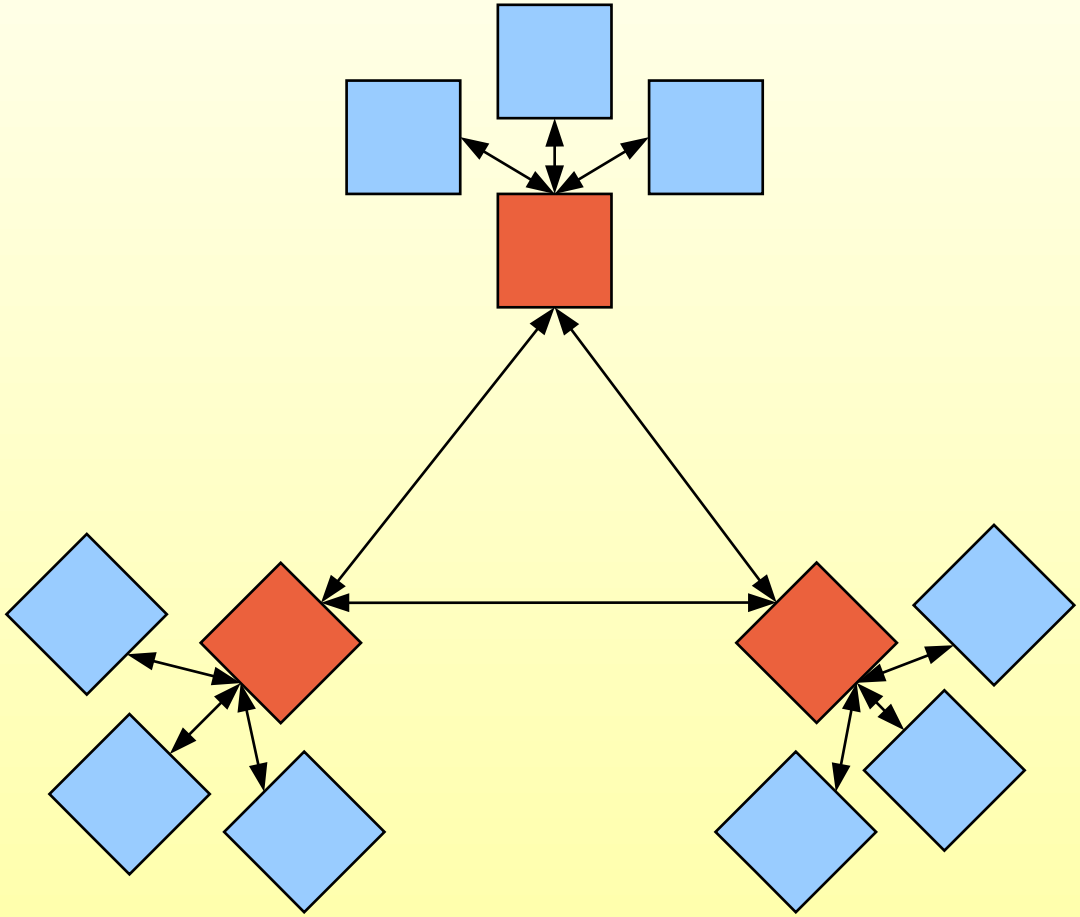
Motivation

- Verfügbar:
  - diverse Cluster
  - viele schnelle Arbeitsplatzrechner
  - Nutzer mit unterschiedlichen Anforderungen
- Nicht verfügbar:
  - Geld für kaum genutzte Ressourcen
  - Platz für billige Gefäßsysteme

Voraussetzungen

- Hard- und Software:
  - IA32 dominiert
  - Linux und Microsoft Windows
- Platzierung und Vernetzung:
  - über gesamte Einrichtung verteilt
  - hierarchische Struktur
    ⇒ geringe Bisektionsbandbreite,
    Beachtung der Topologie notwendig

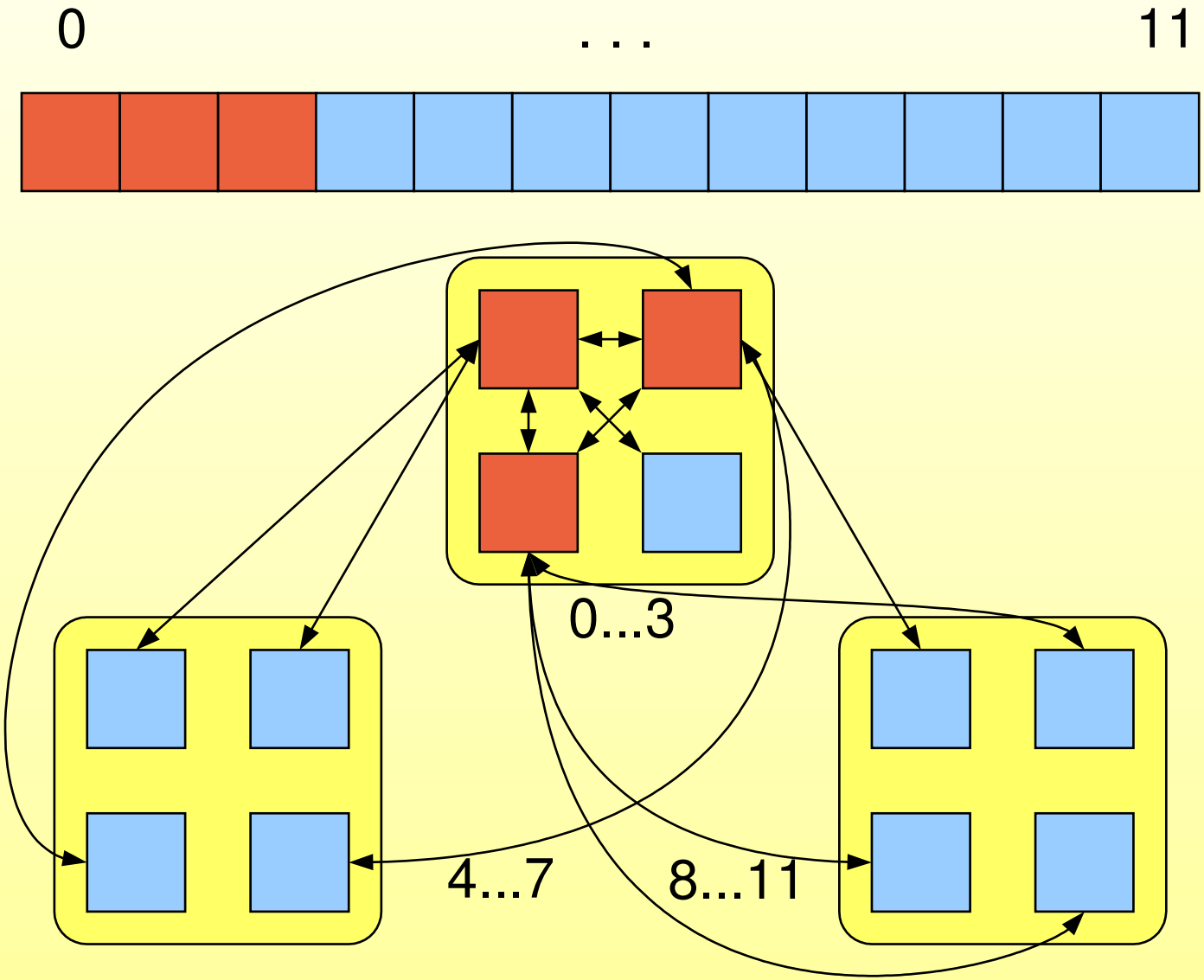
Problem: Topologie



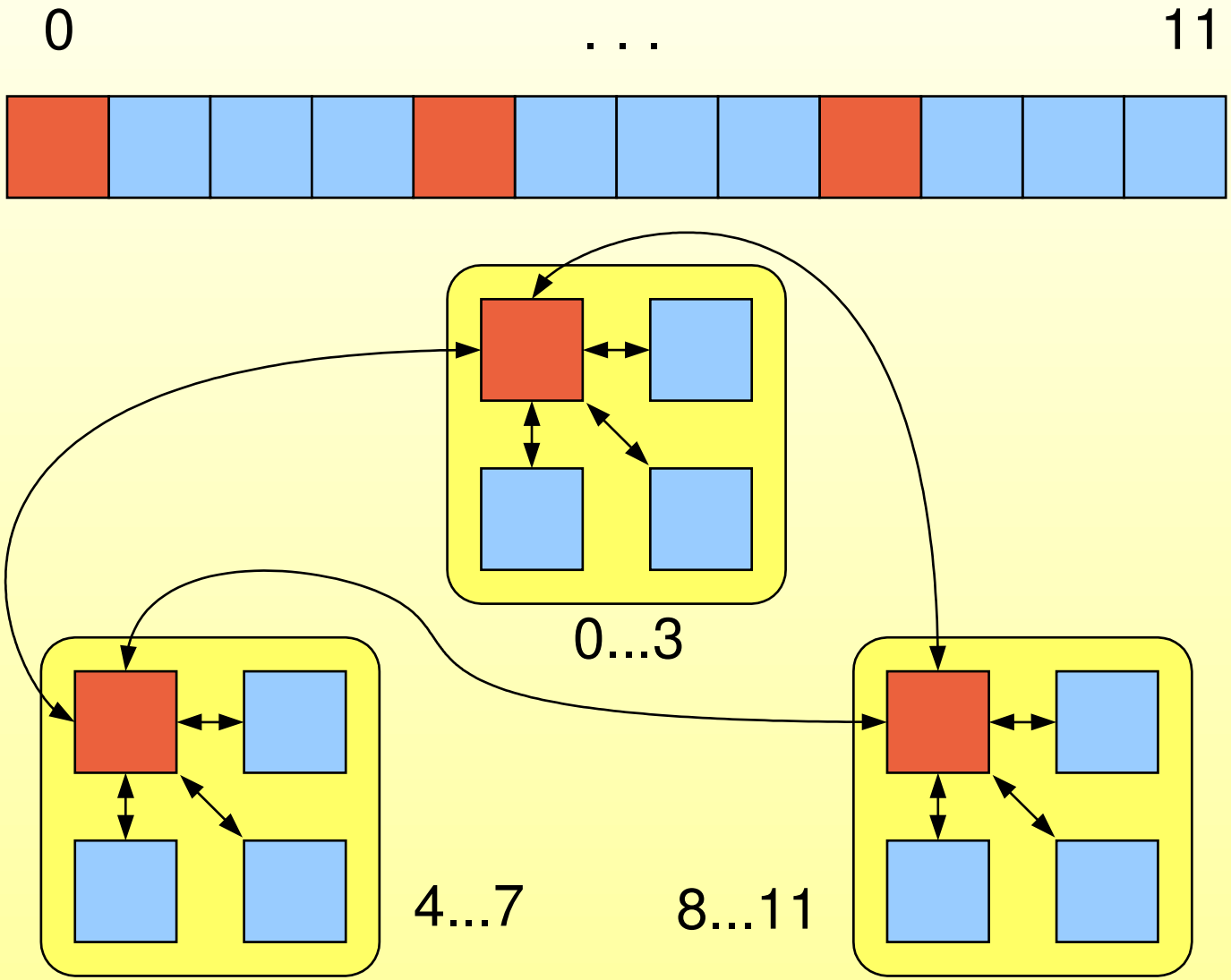
Master

Slaves

Ohne Beachtung der Topologie



Mit Beachtung der Topologie



Arbeitsplatzrechner

- Nutzung:
  - primärer Nutzer, darf nicht behindert werden!
  - regelmäßige Arbeitsphasen, sonst Leerlauf
  - potentiell sehr unzuverlässige Umgebung
- heterogen mit homogenen Teilmengen:
  - Pools
  - URZ Administrations-Dienst

Cluster

- Definition:
  - Menge eigenständiger Rechner („Knoten“)
  - nur durch Kommunikationsnetzwerk verknüpft
  - Nutzung als Einheit
- günstig: Beowulf-Cluster
  - Technik vom Massenmarkt
  - Linux als Betriebssystem

Cluster: Anwendungen

- Kommunikation mittels Nachrichten
  ⇒ „Message-Passing-Programmierung“
  - MPI – Message Passing Interface
  - PVM – Parallel Virtual Machine
- Programmierparadigmen:
  - Task-Farming (Master/Slave)
  - Single Program Multiple Data (SPMD)
  - ...

Idee: Integration

- Zusammenschalten mehrerer Cluster
  - dadurch entsteht sehr großer Cluster
  - gut für kurzzeitige große Jobs
- Einbeziehen der Arbeitsplatzrechner
  - nur kurzfristig und -zeitig nutzbar
  - gut für häufige kleine Jobs

Grid-Computing

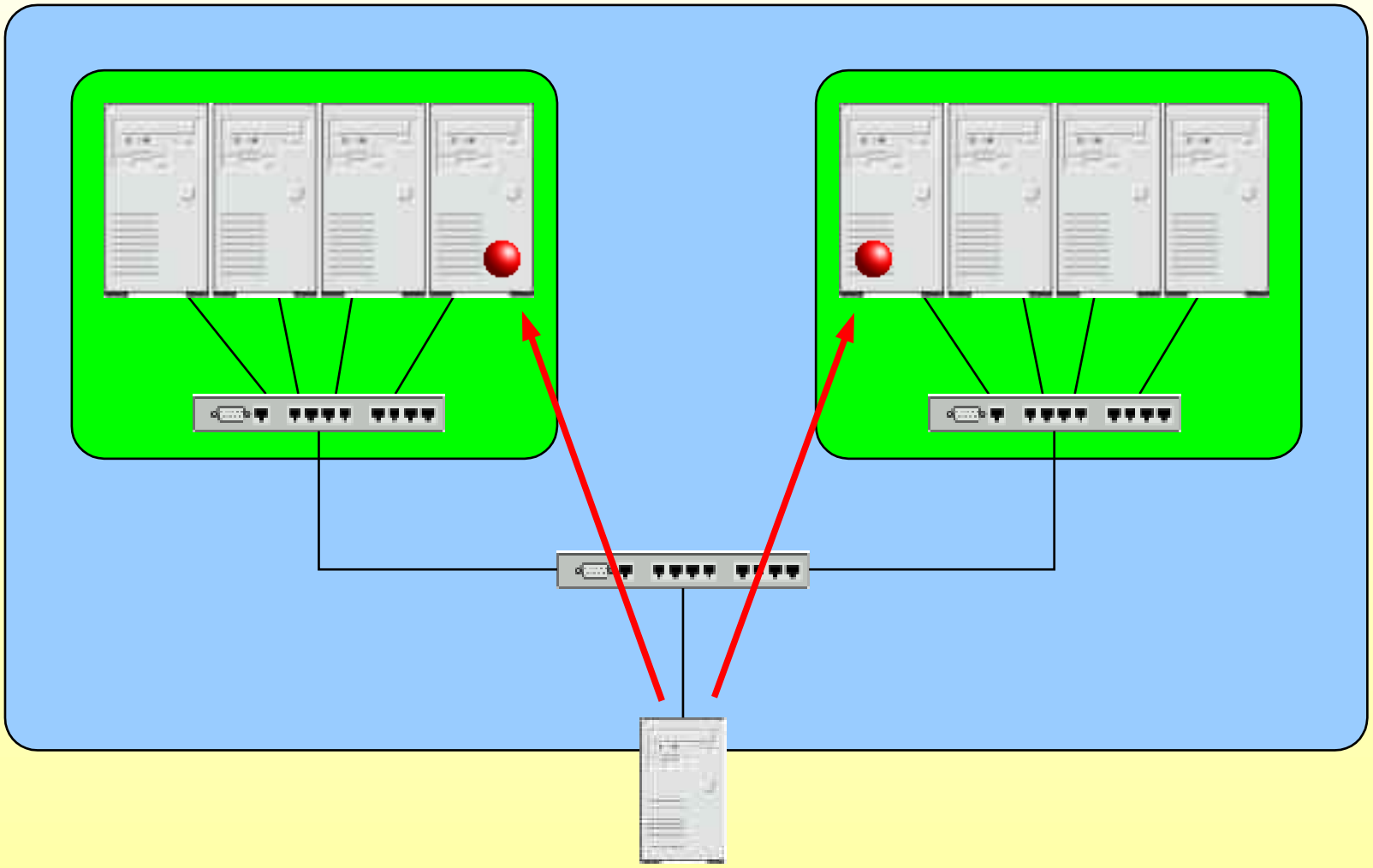
- Ziel: überall verfügbar, einheitlicher Zugriff
- Anwendungsaspekte: [nach Ian Foster]
  - On-Demand Computing
  - Distributed Supercomputing
  - High-Throughput Computing
  - Data-Intensive Computing
  - Collaborative Computing

Globus + MPICH-G2

(Distributed Supercomputing)

- Globus-Toolkit:
  - Sicherheits-Infrastruktur (X.509 Zertifikate)
  - einheitlicher Zugriff auf Ressourcen
  - Anbindung an verschiedene Jobmanager
  - Unterstützung für MPI-Jobs
- MPICH-G2:
  - Kommunikationsdevice für MPICH
  - nutzt Globus: Prozesse und WAN-Komm.

MPICH-G2



MPI lokal

Globus

MPICH-G2

Condor

(High-Throughput Computing)

- Pool vorhandener Rechner
  - automatische Erkennung freier Ressourcen
- zentraler Management-Rechner
  - gleicht Anforderungen mit Ressourcen ab
  - schickt Jobs an verfügbare Rechner
- rechnende Knoten sichern evtl. Checkpoints
- „PVM“ Universum: Task-Farming mit PVM

Probleme

- mehrere Interfaces für ein Problem, aber:
  - Globus-Condor-Jobmanager für Globus
  - Condor-G („Globus“ Universum für Condor)
- Bedarf für High-Throughput Computing?
- Laufzeit schneidet manchmal MTBF
  - Fehlertoleranz notwendig
  - Lösung: Checkpointing

Checkpointing

- „Zwischenstand“ eines Prozesses sichern:
  - CPU-Register, Speicher, Prozessstatus
- Implementation:
  - Systemebene, transparent
  - Anwendungsebene, transparent
  - Anwendungsebene, nicht transparent

Checkpointing verteilter Systeme

- Konsistenzsicherung, z. B.:
  - unabhängiges Checkpointing
  - blockierendes, koordiniertes Checkpointing
  - Bronevetsky-Marques-Pingali-Stodghill
- Datenlagerung:
  - lokal / zentral / verteilt
  - pessimistisch / optimistisch

Eigene Lösung

- einfach aber flexibel
- koordiniertes, blockierendes Checkpointing
- Anwendungsebene, nicht transparent
  - komplett durch Nutzer gesteuert
  - keine verbotenen Systemfunktionen
- Probleme:
  - Lagerproblem ungelöst
  - nicht für alle Programme geeignet

Vielen Dank für Ihre Aufmerksamkeit!