

Aufgabenblatt 6

(Zu bearbeiten bis 04.12.)

Vorlesung Verteilte Systeme Wintersemester 2025/26

Aufgabe 1: Programmierung: JNDI

In der Vorlesung wurden zwei Beispiele zu JNDI gegeben, zunächst der Zugriff auf die RMI-Registry und der Zugriff auf das lokale Dateisystem.

- a) Was genau ist unter JNDI zu verstehen und wozu nutzt man dieses Interface?
- b) Machen Sie sich mit der grundsätzlichen Nutzung von JNDI vertraut. Ändern Sie Ihr HelloWorld-Programm aus den letzten Übungen so ab, dass Sie mit Hilfe von JNDI auf die RMI-Registry zugreifen können.
- c) Implementieren Sie auch das zweite Beispiel („Zugriff auf lokales Dateisystem“) aus Kap. 4.2 der Vorlesung und führen Sie es aus. Nehmen Sie dazu die im Archiv [u06Files.zip](#)¹ auf der Vorlesungswebseite zum Download bereitgestellten Jar-Files sowie [dieses JNDI Tutorial](#) zu Hilfe.

Aufgabe 2: Lastverteilungsstrategien

Lastverteilung in verteilten Systemen hat das Ziel, die auftretende Last möglichst gleichmäßig zu verteilen, um Überlast auf einzelnen Knoten zu verhindern. Ein ehrgeizigeres Ziel wäre es, die Last so zu verteilen, dass der Gesamtdurchsatz maximiert wird.

- a) Wie unterscheiden sich statische, dynamische und präemptiv-dynamische Lastverteilung voneinander? Welche Vor- und Nachteile haben die Verfahren und welche Voraussetzungen müssen jeweils gegeben sein, um eins der Verfahren anwenden zu können?
- b) Alle Lastverteilungsstrategien gründen ihre Entscheidungen auf irgendeine *Lastmetrik*. Die einfachste, oft betrachtete Lastmetrik bei theoretischer Analyse der Algorithmen ist die Anzahl der (rechenbereiten) Prozesse pro Knoten. Warum ist diese Metrik in der Praxis teilweise nicht ausreichend? Was wäre eine bessere Metrik und welche neuen Probleme treten dann auf?

Aufgabe 3: Graphpartitionierung und List-Scheduling

In der Vorlesung (Kap. 5.1.1) wurden zwei Arten von verteilten Scheduling-Algorithmen vorgestellt, die auf Graphen basieren:

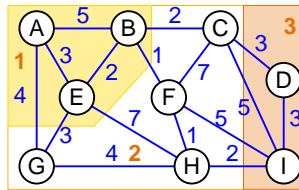
- Lastverteilung durch Graphpartitionierung und
- List-Scheduling.

Welche Gemeinsamkeiten und Unterschiede können Sie zwischen diesen beiden Klassen von Algorithmen feststellen? Unter welchen Voraussetzungen sind sie anwendbar?

¹<http://www.bs.informatik.uni-siegen.de/web/wismueller/vl/vs/u06Files.zip>

Aufgabe 4: Lastverteilung durch Graphpartitionierung

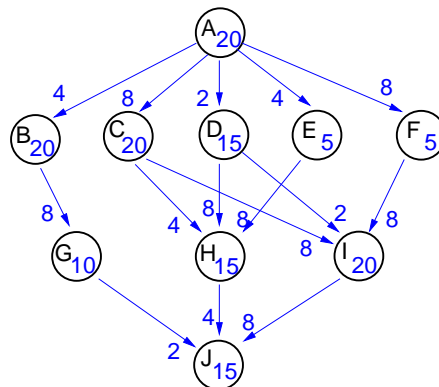
Erinnern Sie sich an die Lastverteilung durch Graphpartitionierung (s. Vorlesung Kap. 5.1.1). Nehmen Sie an, daß in der unten stehenden Abbildung der Prozess H von Knoten 2 auf Knoten 3 wechselt. Wie groß ist danach der (gesamte) Netzwerkverkehr zwischen den Knoten?



Pflichtaufgabe 5: List-Scheduling Abgabe über moodle!

In der Vorlesung wurden verschiedene Ansätze zum verteilten Scheduling vorgestellt (s. Kap. 5.1.1). Zwei häufig verwendete List-Scheduling Algorithmen sind: *High Level First with Estimated Time* (HLFET) und *Earliest Task First* (ETF). Der erste war schon in der Vorlesung dargestellt (s. Animation in Kap. 5.1.1: List-Scheduling mit HLFET). Für den zweiten Algorithmus finden Sie eine genaue Beschreibung im dem Artikel von Hagras und Janacek (über die Vorlesungsseite oder <https://ojs.cvut.cz/ojs/index.php/ap/article/download/490/322>) auf Seite 19.

Gegeben sei ein System mit 3 Prozessoren und der unten gezeigte DAG, der ein Programm, bestehend aus mehreren abhängigen Tasks, modelliert. Die Knoten beinhalten Tasks mit Ausführungszeiten und die Kanten zeigen die notwendige Kommunikation mit Übertragungsdauer.



Wie werden die Tasks auf die drei Prozessoren gescheduled (mit HLFET / mit ETF)? Zeichnen Sie jeweils ein Gantt-Diagramm. Nehmen Sie an, dass eine lokale Kommunikation keine Zeit kostet.

Pflichtaufgabe 6: Programmierung: Vektor-Uhr Abgabe über moodle!

In dieser Aufgabe sollen Sie mit Hilfe des Verteilte-Systeme-Simulators eine Vektor-Uhr (siehe Abschnitt 6.3 der Vorlesungsfolien) programmieren.

Bearbeiten Sie dazu die Aufgabe „[Vector Clocks](#)“² im Wiki (Zugriff nur aus dem Uni-VPN!).

²<https://git.bs.informatik.uni-siegen.de/dsbox/exercises/wiki/3-vector-clocks>