Prüfungsfach: Verteilte Systeme (5 ECTS)
Datum: 14.09.2015
Prüfer: Jürgen Kleinöder
Beisitzer: Klaus Stengel
Ergebnis: sehr fair; 1.0 trotz manchen Sachen, auf die ich nicht sofort gekommen bin

Allgemein: Zeit verging sehr schnell! Am besten man redet einfach zu einem Thema, soweit es geht. Wenn der Prüfer was genauer wissen will, wird man unterbrochen. Ich bin nicht mal so weit gekommen, die erste Frage komplett zu beantworten, weil ich bei den ersten 3 Punkten unterbrochen wurde und die ganze Prüfung ging dann nur darum, die ersten drei Punkte genauer zu erklären. Kommentar am Ende: “Es war nicht zäh, wir haben uns gut unterhalten.” Also Hauptsache einfach weiterreden.

Frage 1: Was kann bei verteilten Systemen schiefgehen?

Im entfernten Fall mehr Fehler wie lokal, Unterstützung von dynamischer Bindung

Frage 2: Wie kann man das mit der dynamischen Bindung unterstützen?

Naming Service -> Remote Referenz auf Remote-Objekte -> Client holt sich die Referenz und generiert dynamisch zur Laufzeit einen Proxy (Stellvertreter) und ruft dadurch Methoden auf dem Server auf

Frage 3: Wo macht man das in der Wirklichkeit, dass man dynamisch was aufruft, von dem man die Schnittstelle nicht kennt?

Da habe ich nicht recht weiter gewusst, hab Thin Clients gesagt, war dann auch richtig. Er wollte eigentlich auf Browser aus.

Frage 4: Kann man das auch statisch hinterlegen, wie die Schnittstelle aussieht?

Ja, mit IDL, IDL erklärt

Frage 5: Was ist sonst so in verteilten Systemen anders?

Replikation statt einzelne Instanzen. Einerseits dann Zuverlässigkeit und Verfügbarkeit des Systems, aber auch Mechanismen zur Konsistenzwahrung nötig.

Frage 6: Wie macht man das?

Aktive Replikation (Funktionsweise erklärt) + hier nötig: totale Ordnung über alle Anfragen, und Replikate müssen deterministische Zustandsmaschinen realisieren.
Passive Replikation: cold passive und warm passive erklärt

Frage 7: Gibt es dabei Probleme ?

Bei aktiver: nicht deterministische Systemaufrufe (zB Zufallszahlen -> Replikate müssen sich auf einen gemeinsamen Wert einigen), externalisierende Ereignisse (zB Infos vom anderen Server holen, Client-Rückrufe -> muss einer (dabei zB Anführerknoten wählen) machen, das Ergebnis den anderen mitteilen)

Frage 8: Was kann man sonst bei Fehlern machen?

RPC-Semantiken; Maybe erklärt (wichtig: im Fehlerfall hat der Client keinerlei Info darüber, ob die Funktion ganz, teilweise oder gar nicht ausgeführt wurde)
At-Least-Once (wichtig: Idempotenz) erklärt; At-Most-Once erklärt.

Frage 9: Was hat At-Least-Once für einen bedeutenden Vorteil gegenüber von At-Most-Once?

Hab gesagt, dass man hier keinen Speicher braucht für Ressourcen, und dass man hier die Anfragen nicht identifizieren muss

Frage 10: Nicht unbedingt. (At Most Once mit Client Server Datenbank aufgemalt)
Was passiert wenn eine wiederholte Anfrage erst sehr spät beim Server ankommt - der Server hat dabei schon die Anfrage einmal ausgeführt und die gespeicherte Antwort bereits gelöscht?

Ohne weitere Maßnahmen wird hier eine nicht-idempotente Operation zweimal ausgeführt! -> Ist ein Problem. Man kann das lösen, indem die Anfrage einen Zeitstempel trägt, wann sie gesendet worden ist. Dann braucht der Server auch die Infos vom Client, wieviele Wiederholungen er macht und welche Timeouts er hat -> dann kann der Server berechnen, dass die Anfrage “sehr” alt ist und verwirft sie.

Frage 11: Was passiert, wenn bei At-Most-Once der Server abstürzt, nachdem er die Operation ausgeführt, aber das Ergebnis noch nicht gespeichert hat?

Das habe ich nicht genau gewusst, habe gemeint, dass irgendwie der Zustand bereits verändert worden ist, aber man nach Wiederanlauf des Servers nicht weiß, zu welcher Anfrage diese Veränderungen gehören… Die Lösung war dann, am Server zu protokollieren, bevor er die Anfrage ausführt, zu welcher Anfrage er gerade was bearbeitet, damit man das nach dem Wiederanlauf wieder weiß. (Diese letzte Frage hat aber nicht mehr zu der Bewertung gezählt, weil außerhalb der Zeit).