Test's für Verteilte Systeme

Was wurde getestet:

Praktikum2:

1.) *Test : Daten von Verbraucher/Erzeuger kommen per UDP an Zentrale an*
was passiert ?
Es werden leere Pakete abgesendet, es werden schlecht formatierte Pakete gesendet und korrekt formatierte Pakete mit ungültigen Daten.
Test Ergebnis:
Die Zentrale verwirft die Pakete die versendet und angekommen sind und stürzt nicht ab.

2.)*Test: Daten kommen mittels TCP am HTTP-Server an und werden vollständig* ausgegeben
Was passiert ?
Ändern Sie die variable **BUFFER_SIZE** in unserer Datei docker-compose
128 bedeutet ComponentInfo's werden in 2 Pakete aufgeteilt
13 bedeutet ComponentInfo's werden auf ca. 200 Pakete aufgeteilt
Test Ergebnis:
Die Zentrale fasst die aufgeteilten Pakete korrekt zusammen.
Sie können unter http://localhost:1234/ausgabe sehen, dass die Zentrale immer noch ComponentInfo's erhält unter anderem die ID, Erzeuger/Verbraucher, kW und die Zeit.

3.) *Performance Test:*
Den Performance Test haben wir mit hilfe von einem Tool namens httpload durchgeführt, hier kommen unsere Resultate.
Test Ergebnis:
Vollständige Anfragen: 1000
Fehlgeschlagene Anfragen: 0
Zeitaufwand für Tests: 0,663 Sekunden
Anfragen pro Sekunde: 1609.96 [#/sec] (Mittelwert)
Zeit pro Anforderung: 6.283 [ms] (Mittelwert)
Übertragungsrate: 687,47 [Kbytes/sec] empfangen
Insgesamt übertragen: 451005 Bytes

Praktikum3:

1.) *Test: RPC*
Was passiert?
Gehen Sie in den Root Ordner von unserem Projekt und starten Sie Powershell/cmd/gitbash. Anschließend geben sie follgenden befehl ein 'docker attach Client', damit Sie sich mit unserem External Client Verbinden. Wenn Sie jetzt 'history' eingeben dann sehen Sie auf der Konsole alle Ausgaben von der Zentrale. Außerdem kann man auf der http://localhost:1234/ausgabe seite das selbe sehen.
Test Ergebnis:
Die Browser ergebnis und Konsolen ergebnise sind identisch. Die Komplette 'history' wurde per RPC vollständig übertragen.

2.) *Test: Erzeugte/Verbrauchte kW*
Test Ergebnis:
Die Ausgabe der Zentrale kann im Terminal oder unter localhost:1234/ausgabe abgerufen werden. Hier können Sie sehen, dass die Differenz zwischen verbrauchten und erzeugten kW immer kleiner

als 60kw ist.

3.) *Perfomance Test: Durchschnittlicher Aufruf mit Apache Thrift*
Was passiert?
Gehen Sie in den Root Ordner von unserem Projekt und starten Sie Powershell/cmd/gitbash. Anschließend geben sie follgenden befehl ein 'docker attach Client', damit Sie sich mit unserem External Client Verbinden. Wenn Sie jetzt 'test' eingeben, fragt der Client die erste Seite mit 100 Einträgen an, dies macht der dann 100.000 mal (numberOfCalls).

Test Ergebnis:
Die durchschnittliche Zeit für den Empfang einer Seite mit 100 Einträgen über RPC beträgt ~0,11ms.

Praktikum4:

1.) *Test: MQTT*
Was passiert?
Die Zentrale soll Daten aus den folgenden MQTT empfangen:
power/component/n/info wobei n = 1...5 seien kann.

Test Ergebnis:
Ausgabe in der Konsole, power/component/1...5/info

2.) Test: Daten über MQTT sind korrekt
Was passiert?
Gehen Sie in den Root Ordner von unserem Projekt und starten Sie Powershell/cmd/gitbash. Anschließend geben sie follgenden befehl ein 'docker attach Client', damit Sie sich mit unserem External Client Verbinden. Wenn Sie jetzt 'status' eingeben, erhält man die ausgabe für kW, wieviel verbraucht und erzeugt wurde. Ebenfalls kann man auf der http://localhost:1234/ausgabe die ausgegebenen componeninfos ansehen.

3.) *Performance Test: MQTT, UDP*
Was passiert?
Die Zentrale wartet bis eine nachricht mit den jeweiligen componentinfos angekommen ist. Einmal für MQTT und UDP, das geht solange bis beide 10.000 componeninfos erhalten haben.

Test Ergebnis:
UDP: Braucht ca. ~8,6s für 10.000 componeninfos und MQTT: braucht für die selbe menge ~16,7

Praktikum5:

1.) *Test:Komplette Daten von der Zentrale:*
Was passiert?
Gehen Sie in den Root Ordner von unserem Projekt und starten Sie Powershell/cmd/gitbash. Anschließend geben sie follgenden befehl ein 'docker attach Client', damit Sie sich mit unserem External Client Verbinden. Wenn Sie jetzt 'history 1, history 2 oder history 3' eingeben, erhält man die jeweilige Ausgabe von der zugehörigen Zentrale.

Test Ergebnis:
Daten von der jeweiligen nachgefragten Zentrale wird ausgegeben, in folgendender Form: ID, Name des Erzeugers/Verbrauchers, die gesamte kW und die Zeit.

2.) *Test: Client antwort von Zentrale*
Was passiert?
Gehen Sie in den Root Ordner von unserem Projekt und starten Sie Powershell/cmd/gitbash. Zuerst Stop Zentrale1 und Zentrale2, anschließend geben sie follgenden befehl ein 'docker attach Client', damit Sie sich mit unserem External Client Verbinden. Wenn Sie jetzt 'history 1, history 2 oder history 3' eingeben, wird nur die Zentrale 3 antworten.

Test Ergebnis:
Zentrale 1 und Zentrale 2 wurde von uns gestoppt und somit wird nur noch Zentrale 3 antworten.

3.) *Performance Test: Leistung mit mehreren Zentralen*
Was passiert?
Gehen Sie in den Root Ordner von unserem Projekt und starten Sie Powershell/cmd/gitbash. Anschließend geben sie follgenden befehl ein 'docker attach Client', damit Sie sich mit unserem External Client Verbinden. Wenn Sie jetzt 'test' eingeben, fragt der Client die erste Seite mit 100 Einträgen an, dies macht der dann 100.000 mal (numberOfCalls).

Test Ergebnis:
1 Zentrale Aktiv ~213s;
2 Zentrale Aktiv ~109s;
und bei 3 Zentralen Aktiv 68s.
Was bedeutet, wenn mehrere Zentralen gleichzeitig laufen das es einen großen Leistungsvorteil bringt.