Perzentile mit Hadoop ermitteln

Ausgangspunkt

Ziel dieses Projektes war, einen Hadoop Job zu entwickeln, der mit Hilfe gegebener Parameter Simulationen durchführt und aus den Ergebnissen die Perzentile ermittelt.

Gegeben war ein SQL-Dump, aus welchem die Werte **DIRECTION**, **NODE_REF**, **DISTANCE** und **HAS_VEGETATION** in jeder Zeile zu extrahieren waren. **DISTANCE** und **HAS_VEGETATION** dienen als Parameter um für jede Zeile eine bestimmte Anzahl von **n** Simulationen durchzuführen. **DIRECTION** und **NODE_REF** sind beschreibende Daten. Im letzten Schritt soll aus allen Simulationsergebnissen bzw. für einen bestimmten **DIRECTION** Wert die Perzentile berechnet werden.

Vorgehen

Die beiden Schritte Simulation und Analyse wurden in zwei einzelne Hadoop Jobs getrennt. Die Analyse baut allerdings auf dem Datenformat, welches die Simulation ausgibt, auf.

Simulation

Bei der Erzeugung erhält der Mapper die Konfigurationsdaten, mit welcher Klasse simuliert werden soll, die Anzahl **n** an Simulationen und wie viele Stellen nach dem Komma ausgegeben werden sollen. Übergeben bekommt er eine Zeile des SQL-Dumps, aus dem er die vier in der Aufgabenstellung genannten Werte extrahiert. Anschließend wird die Simulation **n**-mal ausgeführt und die Ergebnisse werden dem Reducer übergeben. Das Simulationsergebnis selber fungiert als Schlüssel für die Ausgabe. Hadoop garantiert, dass die Ergebnisse in der Shuffle- und Sort-Phase anhand des Schlüssels sortiert werden. Dies erleichtert bei der Analyse das Ermitteln der Perzentile erheblich. Als letztes übergibt der Mapper noch für jede Zeile einmalig die **DIRECTION** als Ergebnis. Dieser Wert wird anschließend im Reducer gebraucht, um die Anzahl der Simulationen für eine spezifische **DIRECTION** zu bestimmen. Die Anzahl der Simulationen gesamt und für eine **DIRECTION** wird in der Analyse gebraucht, um die Positionen der Perzentile zu berechnen.

Der Reducer zählt die Anzahl der Simulationen insgesamt und für jede **DIRECTION** einzeln. Die Ergebnisse werden in der Datei **total_simulation.txt** gespeichert. Für jedes einzelne Simulationsergebnis werden die Zeilennummer insgesamt und die Zeilennummer für die **DIRECTION** angehangen. Ein Beispiel könnte so aussehen:

Simulationsergebnis (Schlüssel)	NODE_REF	DIRECTION	Zeilennummer DIRECTION	Zeilennummer insgesamt
4.54863	29249	0	3	59
Richtige Ausgabe	4.54863\t29249_0;3_59			

Die Zeilennummern werden angehangen, da es mit Hadoop nicht möglich ist, im Mapper oder Reducer zu erfahren, in welcher Zeile man sich befindet. Hadoop arbeitet mit **Datenblöcken**. Wenn alle Zeilen die gleiche Bytegröße haben, kann man die Zeilennummer berechnen. Das Anhängen der Zeilennummer wird jedoch als einfacher und weniger fehleranfällig angesehen und spart mehr Speicherplatz gegenüber dem Auffüllen aller Zeilen auf eine feste Länge. Die Zeilennummer für die **DIRECTION** und insgesamt wird im Reducer beginnend bei eins hochgezählt.

Die Ergebnisse des Reducer werden für jede **DIRECTION** in separierte Dateien geschrieben.

Analyse

Der Mapper rechnet sich anhand der Anzahl der Simulationen, welche aus der Datei **total_simulation.txt** bezieht, aus, welche Zeilennummer dem x-ten Perzentil entspricht. Dies macht er für alle Simulationen und alle **DIRECTION** Werte separat. Im Mapping Schritt extrahiert er die Zeilennummern, die in der Simulation angehangen wurden, und vergleicht diese nur noch mit den berechneten Nummern für die Perzentile. Stimmen die Nummern überein, wird die komplette Zeile an den Reducer übergeben. Die restlichen werden ignoriert.

Der Reducer fügt der Zeile nur noch das Perzentil hinzu und speichert das Ergebnis im **HDFS**. Für die Ergebnisse insgesamt und für jede **DIRECTION** wird eine eigene Datei angelegt. Eine Zeile der Ausgabe sieht wie folgt aus:

Perzentil	Simulations-ergebnis	NODE_REF	DIRECTION	Zeilennummer DIRECTION	Zeilennummer insgesamt
Q_0.28	-0.55038	1971	0	1708000	13662217
Richtige Ausgabe	Q_0.28\t-0.55038\t1971_0;1708000_13662217				

Entgegen der Aufgabenstellung ein bestimmtes Perzentil zu ermitteln, werden alle **900** herausgeschrieben. Ein Hadoop Job liest immer alle übergebenen Dateien ein. Es ist daher wesentlich aufwendiger zwei Jobs zu starten, um zwei Perzentile zu ermitteln. Der Flaschenhals der Analyse besteht in der Map Phase und der Einlesegeschwindigkeit. Dieser Punkt wird in der Auswertung genauer diskutiert.

Ausführen der Hadoop Jobs

Die angehangene **Percentile.jar** Datei kann mit folgendem Kommando ausgeführt werden:

$ hadoop jar Percentile.jar percentile.config

Übergeben werden muss eine Konfigurationsdatei. Bei dem Format handelt es sich um eine typische Java-Properties-Datei. Die Einstellungen werden im Folgenden erklärt.

Simulation

Einstellung	Bedeutung
simulate	**true** oder **false**. Gibt an, ob eine Simulation durchgeführt werden soll.
input_simulate	Pfad zu den Input Dateien der Simulation.
output_simulate	Pfad, wo der Output Ordner angelegt wird.
simulationClass	Klasse, die für die Simulation verwendet werden soll. Die Klasse muss definierte Simulate Interface implementieren.
numbersAfterComma	Die Anzahl der Stellen nach dem Komma der Simulationsergebnisse.
numberOfSimulations	Die Anzahl der Simulationen.
clearOutputFolder_simulate	**true** oder **false**. Gibt an, ob der Output Ordner, falls er bereits vor dem Ausführen des Jobs existiert, gelöscht werden soll.

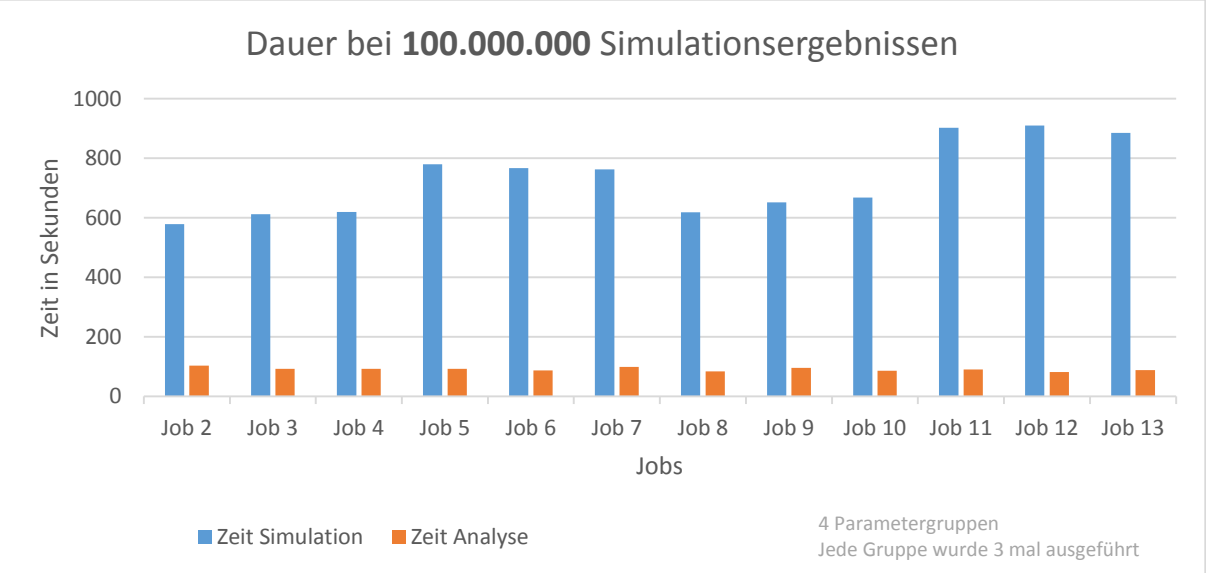
Analyse

Einstellung	Bedeutung
analyze	**true** oder **false**. Gibt an, ob eine Analyse ausgeführt werden soll.
input_analyze	Pfad zu den Input Dateien der Analyse.
output_analyze	Pfad, wo der Output Ordner angelegt wird.
simulationCount	Pfad zu der Datei, wo die Anzahl der Simulationen hinterlegt sind.
clearOutputFolder_analyze	**true** oder **false**. Gibt an, ob der Output Ordner, falls er bereits vor dem Ausführen des Jobs existiert, gelöscht werden soll.

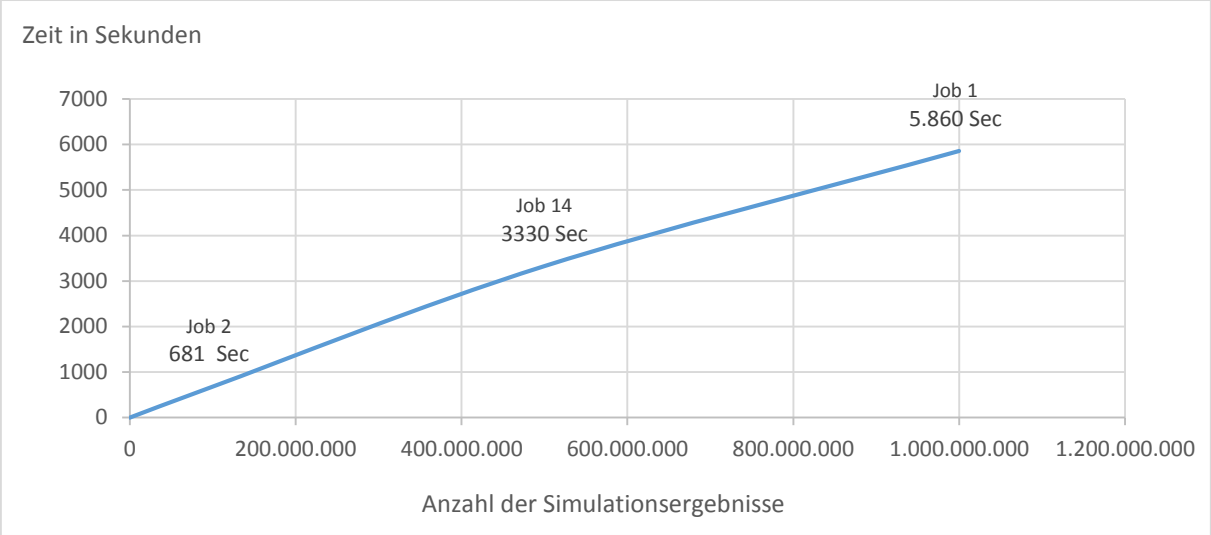
Auswertung

Die Benchmarks, die auf dem **Cluster der Friedrich-Schiller-Universität Jena** ausgeführt wurden, sind angehangen. Die Ergebnisse befinden sich in der Excel Tabelle. Der Cluster besteht aus 17 Nodes mit jeweils 8 CPU Kernen und rund 16 GB RAM (Ausnahme Node 17 mit 32 GB RAM). Jeder Knoten verwendet eine lx24-amd64 Architektur.

Auffällig ist, dass die meiste Zeit für die Simulation benötigt wird. Ein Bruchteil dessen wird für den Analysejob gebraucht. Die Gesamtdauer ist am stärksten von der Anzahl der Simulationsergebnisse abhängig. Je nachdem wie die Werte auf die Anzahl der Dateien, Simulationen und Zeilen aufgeteilt sind, variiert die Gesamtdauer leicht.

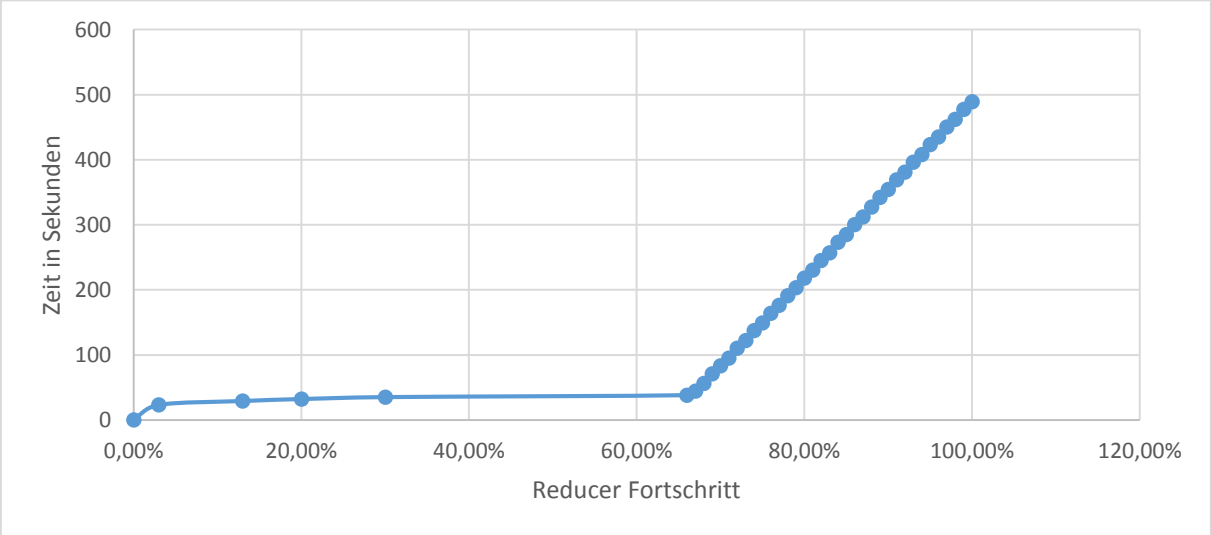


Die Gesamtdauer steigt proportional mit der Anzahl der Simulationsergebnisse.



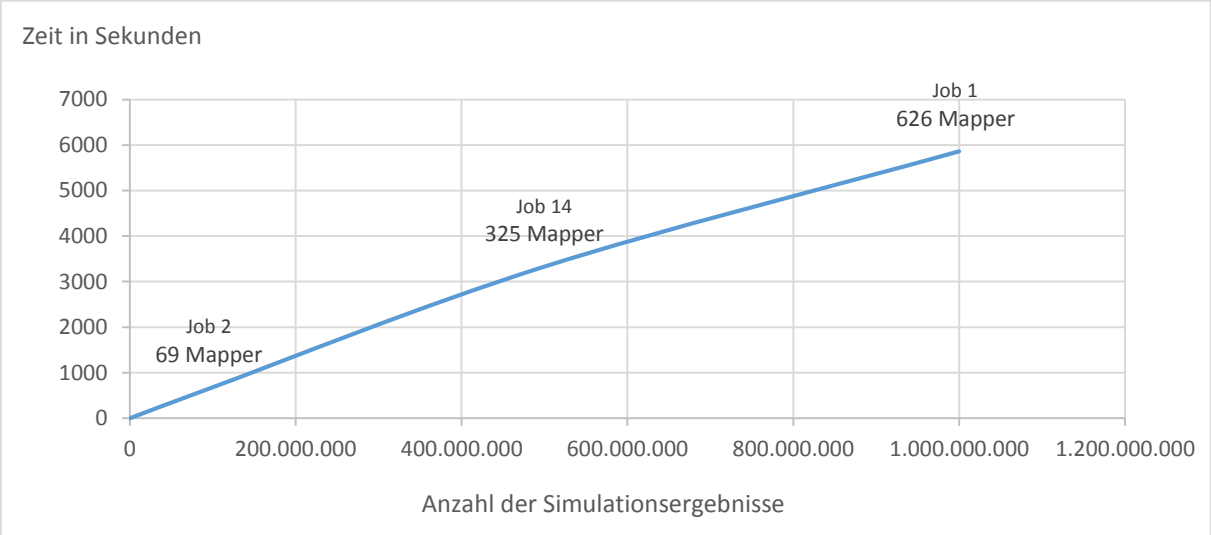
Betrachtet man die Simulation genauer, fällt auf, dass der Reducer die meiste Zeit benötigt. Je nach Aufteilung der Parameter für die Anzahl der Simulationen, Dateien und Zeilen, benötigt der Mapper mehr oder weniger Zeit. Dies hängt vor allem mit der Anzahl der Mapper-Instanzen zusammen, die für den Job erstellt werden. Je mehr Mapper vorhanden sind, desto weniger Zeit wird für das Mapping benötigt. Die Zeit für den Reducer ist relativ konstant und wird von den Eingabeparametern nicht beeinflusst. Der Grund ist simpel: mit Standardkonfiguration von Hadoop wird nur ein einziger Reducer erstellt, der die Simulationsergebnisse linear abarbeitet und ins **HDFS** schreibt. Die Zeit ist damit vor allem von der **IO-Performance** abhängig.

Hadoop gibt den Fortschritt des Reducer aus. Die ersten 66% davon vergehen relativ schnell im Vergleich zum letzten Drittel. Vermutlich sammelt und sortiert Hadoop in dieser Zeit die Daten auf einem Node. Nach der 66% Marke beginnt der eigentliche Reducer zu arbeiten. Dies kann mit entsprechenden Ausgaben gezeigt werden. Der Code, der ab diesem Punkt ausgeführt wird, ist relativ simpel und wenig aufwendig (Vergleich Zeile 68 bis 85 **SimulationReducer** Klasse). Damit wird deutlich, dass die Simulation am meisten von der **IO-Performance** abhängig ist.



Die Analyse verläuft relativ zügig und ist am meisten von der Lesegeschwindigkeit der Daten abhängig. Wie beim Vorgehen beschrieben wurde, müssen nur die entsprechenden Zeilen für die Perzentile gesucht und abgespeichert werden. Am Ende werden stets nur **900** Zeilen geschrieben. Die Anzahl der Simulationsergebnisse ist wesentlich größer.

Die Zeit für die Analyse steigt mit einem kleineren Faktor im Vergleich zu der Anzahl der Simulationsergebnisse. Das hängt mit der Anzahl der Mapper für den Analysejob zusammen. Je mehr Ergebnisse vorhanden sind, desto mehr Mapper werden gestartet, die die Simulationsergebnisse parallel einlesen und auswerten. Der Reducer erhält immer **900** Ergebnisse, die er linear abarbeitet. Er stellt daher keinen Flaschenhals dar.



Fazit

Zusammenfassend ist es wichtig festzuhalten, dass die meiste Zeit für das Schreiben der Daten benötigt wird. Das größte Optimierungspotential bei gleicher Hardware besteht in der Anzahl der Mapper für beide Jobs.

Um Perzentile zu ermitteln, müssen ungeordnete Daten erst sortiert werden. Dies übernimmt in diesem speziellen Fall die Simulation. Um anschließend die richtige Zeile aus den Daten auszuwählen, ist Hadoop – meiner Meinung nach – nicht die richtige Wahl, da Hadoop die Inputdaten nicht anhand von Zeilen sondern Datenblöcken teilt. Um dieses Problem zu lösen, hängt die Simulation die Zeilennummern an, was funktioniert, aber nicht sehr elegant wirkt.

Ausblick

Es sind einige offene Punkte vorhanden, mit denen man die Simulation und das Ermitteln der Perzentile verbessern könnte. Wie im Benchmark ersichtlich ist, spielt die Aufteilung der Anzahl der Simulationen, Dateien und Zeilen eine entscheidende Rolle für die Mapping-Dauer. Die drei Parameter und die Chunksize für einen Mapper stehen in Wechselwirkung. Hier kann ein besseres Optimum ermittelt werden.

Mehrere parallel bzw. verteilt arbeitende Reducer könnten die Performance entscheidend beeinflussen. Beispielhaft könnten bei der Simulation acht Reducer jeweils eine der acht Ausgabedateien füllen.

Durch das Ausgabeformat der Simulation erübrigt sich der Analysejob in der Theorie, da man nur die entsprechenden Zeilennummern der Perzentile ablesen muss. Hier muss untersucht werden, ob ein einfaches Programm dies vielleicht sogar schneller erledigt, wenn es uninteressante Bereiche der Datei überspringt. Hadoop liest alle Zeilen ein. Für diese Aufgabe ist eine feste Zeilenlänge nötig.

Bisher wurden die Simulation und Analyse in zwei Jobs getrennt. Schaltet man die Jobs direkt hintereinander, wird der Reducer der Simulation möglicherweise überflüssig. Zur Erinnerung: dieser Reducer hat die meiste Zeit für die Ausgabe der Ergebnisse benötigt.