Distributed Data Management SS 2015
Prof. Dr.-Ing. Sebastian Michel
MSc. Evica Milchevski, MSc. Kiril Panev
TU Kaiserslautern, FB Informatik – Lehrgebiet Informationssysteme
Sheet 6: Handout 25.06.2015, Presentation 07.07.2015

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN
http://dbis.informatik.uni-kl.de

Assignment 1: Causal Consistency (1 P.)

Given the following processes reading from and writing to a data store.

Process 1:	$R(x)x_0$		$W(x)x_1$			$W(z)z_2$					
Process 2:					$R(x)x_1$			$W(y)y_0$			
Process 3:					$W(y)y_1$		$R(x)x_1$		$W(x)x_2$		
Process 4:		$W(z)z_0$		$W(z)z_1$						$R(x)x_2$	$W(z)z_3$

For clarification, consider the following example of a causal dependency graph. We denote a write to key k with unique version i as k_i, so the graph contains two writes: a write x_1 to key x and a second write y_1 to key y such that x_1 happens-before y_1:

$$x_1 \rightarrow y_1$$

According to the definition of causal consistency, if a process reads y_1, then its subsequent reads from x must return x_1 or another write x_i such that x_i is concurrent to x_1 or $x_1 \rightarrow x_i$. It is not causally consistent if a process reads y_1 then reads an older value x_0 with $x_0 \rightarrow x_1$ because happens-before is not respected.

(a) Draw the causal dependency graph between the individual data values x_i, y_i and z_i and write next to each value the corresponding vector clock.

(b) Consider the following clients that read from the data store. For each of them, determine if, given the values read, the data store provides causal consistency or not. Explain your answers.

Client 1:	$R(x)x_2$ $R(y)y_1$
Client 2:	$R(x)x_1$ $R(y)y_0$ $R(x)x_2$
Client 3:	$R(y)y_1$ $R(x)x_2$
Client 4:	$R(z)z_3$ $R(y)y_0$
Client 5:	$R(z)z_3$ $R(y)x_0$

Assignment 2: Consistent Hashing (1 P.)

Implement a simulation of consistent hashing to study the load balancing characteristics under different parameters. Create 100000 random numbers (or strings) as keys. Create 100 node identifiers. Use MD5 hashing and map the nodes to the id space of $0 \ldots 2^{32}$ by interpreting the MD5 hash of the node id as an integer (long), modulo 2^{32}. Assign the 100000 keys to the nodes according to the consistent hashing principle (smallest node hash that is larger than the key's hash; taking also care of the case around zero). Count for each node the number of distinct keys is obtained to store, i.e., its load (do not forget the zero counts).

(a) Compute the median, average, minimum and maximum load as well as the 25% and 75% percentiles of the load distribution (i.e., the load value that is larger than x% of the other load values).

(b) Create a plot (e.g., with Excel or Gnuplot) of the Lorenz curve of the load distribution. To do so, sort the load values in ascending order (on the x-axis). For each value on the x-axis, put the cumulative load on the y-axis. See, `http://en.wikipedia.org/wiki/Lorenz_curve`, for more details and explanations.

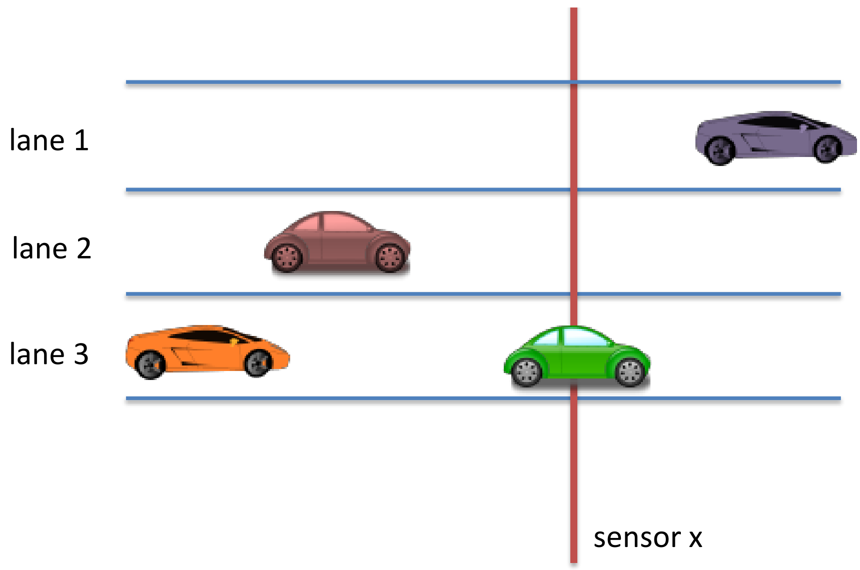
(c) Compute the Gini coefficient as an indicator for load balancing. (See `http://en.wikipedia.org/wiki/Gini_coefficient`). The Gini coefficient can be thought of as the ratio of the area that lies between the line of equality (perfect balancing, i.e., x% of the nodes have x% of the keys) and the Lorenz curve (of part b)).

(c) Extend the implementation such that each node is having T virtual nodes, by placing the node T times in the ring. Compute for $T = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10$ the Gini coefficient as in part c).

Assignment 3: CQL (1 P.)

Considering a stream-processing engine with sliding window semantics as introduced in the lecture. We have access to the following streams:

S1 (timestamp, sensorId, temperature, humidity, co2, xlocation, ylocation)
S2 (timestamp, sensorId, carId, laneId)



(a) Specify a CQL query that computes a stream of the sliding average of temperature×(1+humidity) over a 10 minutes window, every 10 seconds, for all sensors in 1000 meters around (xlocation $= 40.67$, ylocation $= -73.94$). Use a hypothetical user defined function distanceInMeter$(x1, y1, x2, y2)$ to compute the distance based on the geographic coordinates.

(b) Specify a CQL query that computes the minimal time distance of two cars on the same lane within a window of size of 60 minutes.