HPI Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

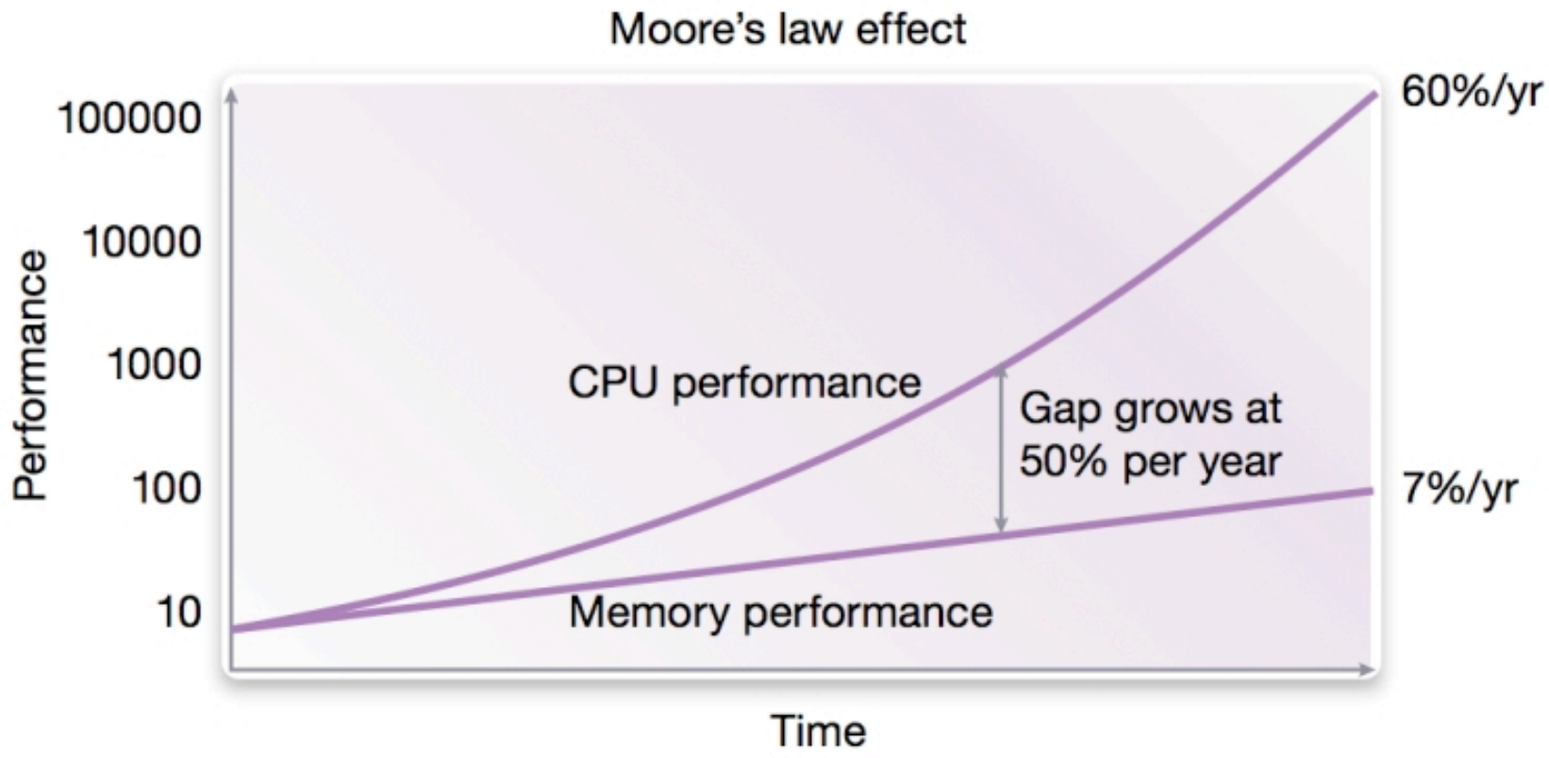
WEEK 3

BYOD

AGENDA

- Dictionary Encoding
- Organization
- Sprint 3

DICTIONARY ENCODING – MOTIVATION



- Memory access is the new bottleneck
- Decrease number of bits used for data representation

DICTIONARY ENCODING – MOTIVATION

- Dictionary encoding is an “easy-to-implement” fixed-width compression and basis for several other compression techniques
- Idea: encode every distinct value of a vector (large) with a distinct fixed-length *integer* value (small)

DICTIONARY ENCODING – EXAMPLE: SAMPLE DATA

- World population: 8 billion records

recID	fname	lname	gender	city	country	birthday
…	…	…	…	…	…	…
39	John	Smith	m	Chicago	USA	12.03.1964
40	Mary	Brown	f	London	UK	12.05.1964
41	Jane	Doe	f	Palo Alto	USA	23.04.1976
42	John	Doe	m	Palo Alto	USA	17.06.1952
43	Peter	Schmidt	m	Potsdam	GER	11.11.1975
…	…	…	…		…	…

DICTIONARY ENCODING – EXAMPLE: ENCODE A COLUMN

recID	fname
…	…
39	John
40	Mary
41	Jane
42	John
43	Peter
…	…

Dictionary for "fname"

valueID	Value
…	…
23	John
24	Mary
25	Jane
26	Peter
…	…

Attribute Vector for "fname"

position	valueID
…	…
39	23
40	24
41	25
42	23
43	26
…	…

- Dictionary stores all distinct values with an implicit valueID
- Attribute vector stores valueIDs for all entries in the column (positions are stored implicitly)

DICTIONARY ENCODING – EXAMPLE: COMPRESSION RATE

- 5 million distinct values, all have a size of 50 B
  - Bits required per valueID: ceil($\log_2(5{,}000{,}000)$) b = 23
  - Dictionary size: $5 * 10^6 * 50$ B = $250 * 10^6$ B = 0.250 GB
  - Attribute vector size: $8 * 10^9 * 23$b = $184 * 10^9$ b = 23 GB
  - Uncompressed: $8 * 10^9 * 50$ B = $400 * 10^9$ B = 400 GB
  - compression rate = uncompressed size / compressed size
    = 400GB / (23 GB + 0.250 GB) ≈ 17

DICTIONARY ENCODING – QUERY DATA

recID	fname
…	…
39	John
40	Mary
41	Jane
42	John
43	Peter
…	…

Dictionary for "fname"

valueID	Value
…	…
23	John
24	Mary
25	Jane
26	Peter
…	…

Attribute Vector for "fname"

position	valueID
…	…
39	23
40	24
41	25
42	23
43	26
…	…

- Retrieve all persons (recIDs) with name "Mary"

DICTIONARY ENCODING – QUERY DATA

recID	fname
…	…
39	John
40	Mary
41	Jane
42	John
43	Peter
…	…

Dictionary for “fname”

valueID	Value
…	…
23	John
24	Mary
25	Jane
26	Peter
…	…

Attribute Vector for “fname”

position	valueID
…	…
39	23
40	24
41	25
42	23
43	26
…	…

- Retrieve all persons (recIDs) with name “Mary”
  - 1. Search valueID for “Mary” (requested value)
  - 2. Scan Attribute vector for “24” (found valueID)

DICTIONARY ENCODING – SORTED DICTIONARY: ADVANTAGES

- Dictionary entries are sorted by their value
  - Dictionary lookup complexity: O(log(n)) instead O(n)
  - Speed up range queries
  - Dictionary entries can be further compressed

DICTIONARY ENCODING – DISADVANTAGES

- Dictionary entries are sorted by their value
  - Resorting for every new value that does not belong to the end of the sorted sequence (relatively cheap)
    - Updating the attribute vector (costly)
- Dictionary adds additional indirection for materialization
- Overhead for large number of data modifying operations

DICTIONARY ENCODING – IN OPOSSUM

- Dictionary encoding is applied to full chunk
- Sorted dictionaries are used
- valueIDs are the C99 fixed-width unsigned integer types (http://en.cppreference.com/w/c/types/integer)

QUESTIONS

ORGANIZATION

- First sprint is due **today**
- Review instructions will be released tomorrow
- Document and interfaces for 2nd sprint will be released today
- Let’s take a look at the interface