

BEKK

VEGTRAFIKK OG STORE DATAMENGDER

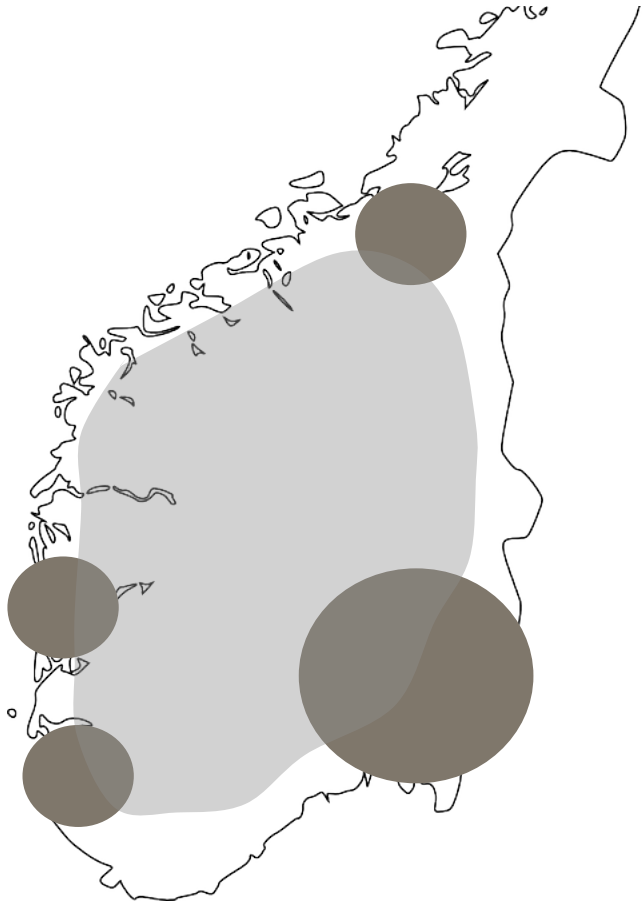
Datainn-prosjektet hos Statens vegvesen

NOKIOS 2016

Kristoffer Dyrkorn, BEKK
@kristofd

Foto: Peter Fiskerstrand. Public domain.





SLITASJE

Mekanisk

Brøyting

Piggdekk

Telehiv

Kjemisk

Salting

Sjøsprøyt

Fugler

Trafikk

Antall kjøretøy

Vekt på kjøretøy

Ansvarlig for riksveger og fylkesveier
Planlegging, konstruksjon, vedlikehold
7500 ansatte
Årlig budsjett, vegformål: 30 mrd NOK



Statens vegvesen

MULIGE BRUKSOMRÅDER

Rapporter

Vedlikeholdsplaner
Verifisere endringer
Påvirkning fra vær

Sanntid

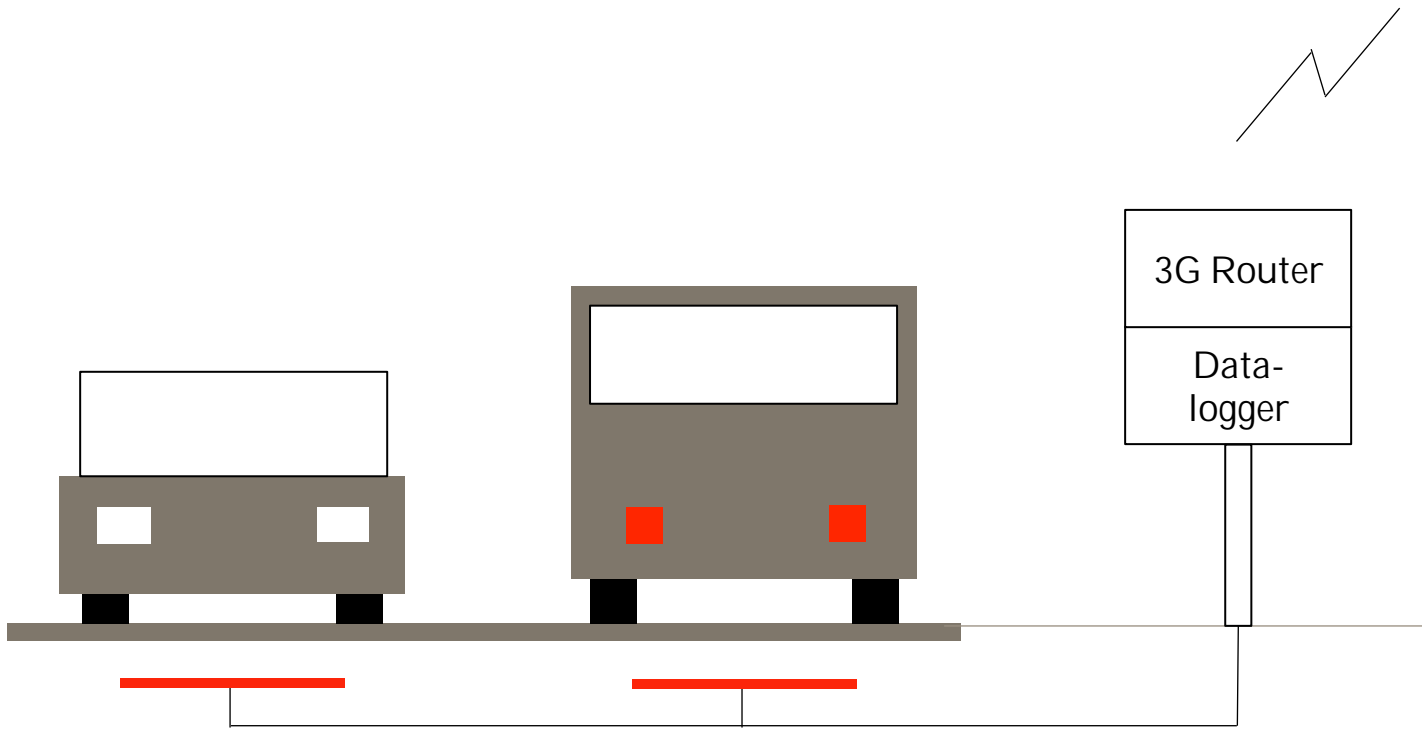
Forventet reisetid
Påvirke trafikkflyt
Utrykningskjøretøy

Prediksjon

Kommende trafikkmonstre
Anbefale veivalg
Påvirke reisevaner

Datainnsamling i vegkanten





Klokkeslett: 2016-10-26 23:58:44

Felt: 1

Hastighet: 80.9 km/t

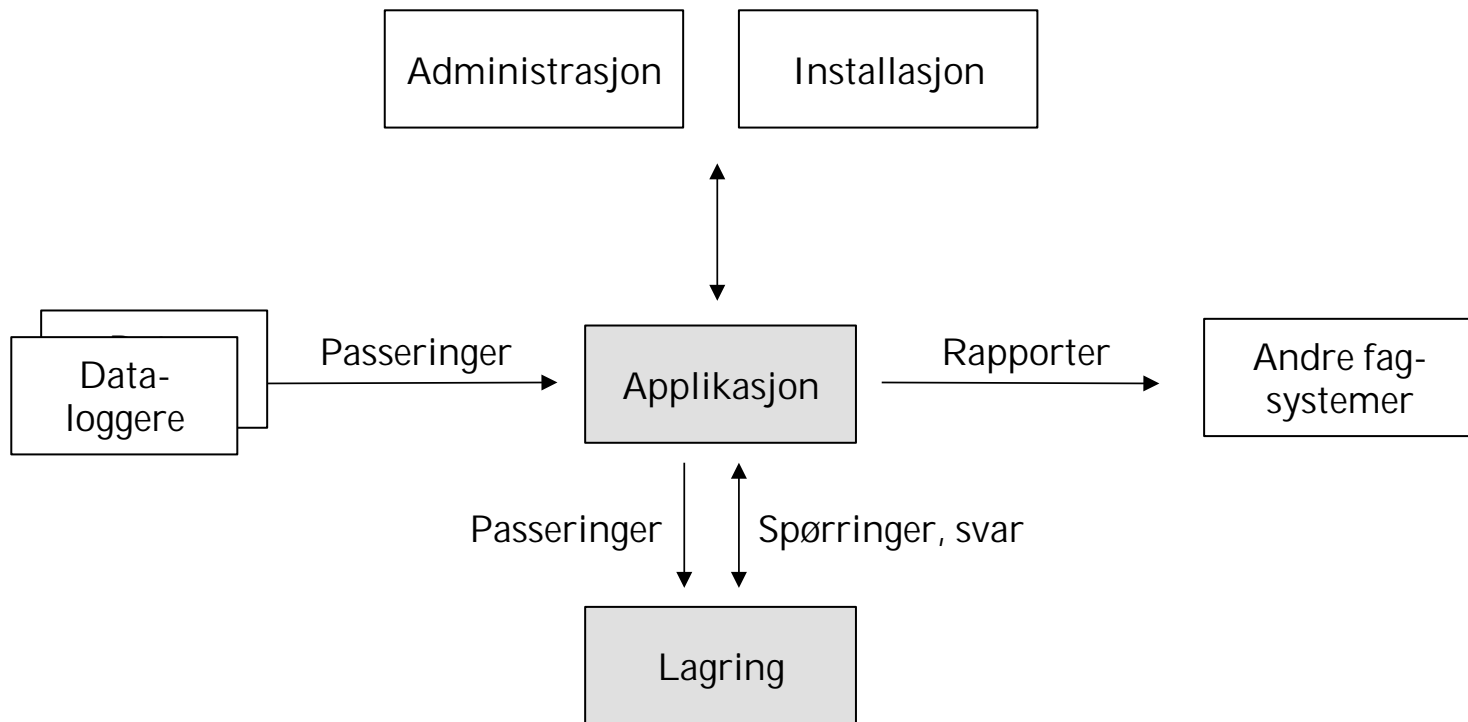
Lengde: 16.46 m

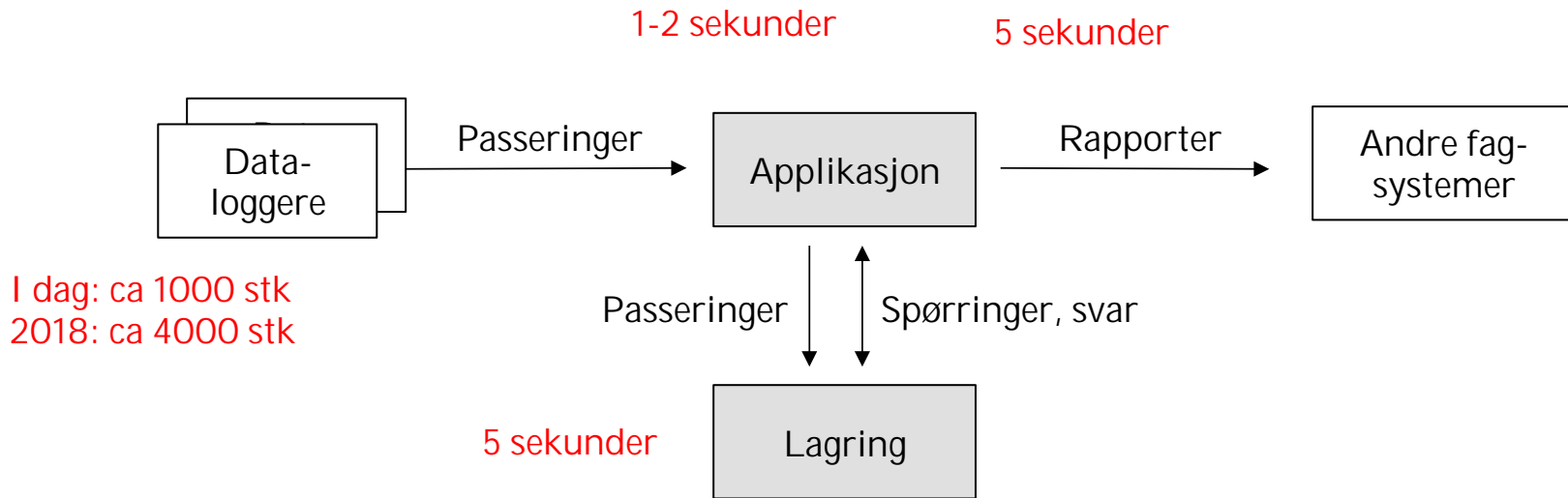
Kjøretøytype: Buss





Om løsningen





200200 - KLØFTA SØR

Trafikk på målestasjon

Medretning: OSLO



Motretning: HAMAR

KVANTITET OG KVALITET

Montering

Sanntidsvisning

Drift

Alltid tilkoblet

Rapporter

Merking, filtrering

HVA MENES MED DATA?

Visualiseringer

Rapporter og statistikker

Data med metadata

Data fra representative trafikkforhold

Passeringer med høy presisjon

Passeringer uten opplagte målefeil

Leverandøruavhengige data

Datasett uten huller eller duplikater

Rådata fra dataloggere



Stigende verdi

STORDATA KREVER HØY SIKKERHET OG ROBUSTHET

Angrep

Blokkering

Lekkasje

Forfalskning

Personvern

Høyt detaljnivå?

Stort volum?

Misbruk

Fartsrekord-app

VIKTIGSTE GEVINSTER AV STORDATA

Nøyaktighet

Riktigere informasjon

Kjent kvalitet

Metadata

Oppløsning

Enkeltpasseringer

Målestasjoner

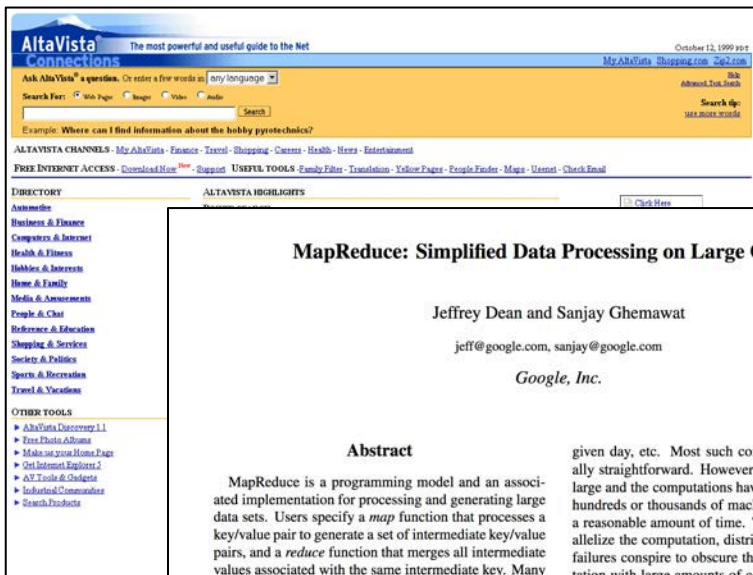
Frihet

Frie spørringer

Møte ikke avdekkede behov

Sanntid og historisk

TEKNOLOGISK UTVIKLING HAR GJORT DETTE MULIG



MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

Google, Inc.

Abstract

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a *map* function that processes a key/value pair to generate a set of intermediate key/value pairs, and a *reduce* function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model, as shown in the paper.

Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling ma-

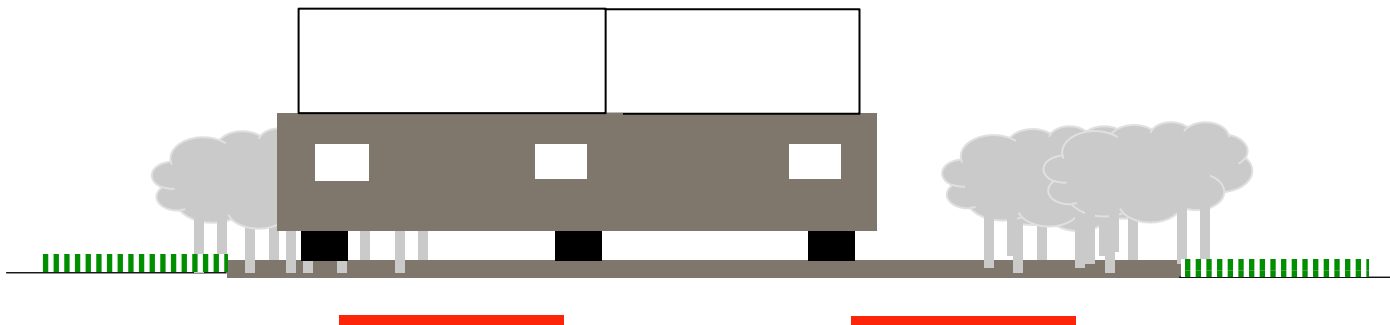
given day, etc. Most such computations are conceptually straightforward. However, the input data is usually large and the computations have to be distributed across hundreds or thousands of machines in order to finish in a reasonable amount of time. The issues of how to parallelize the computation, distribute the data, and handle failures conspire to obscure the original simple computation with large amounts of complex code to deal with these issues.

As a reaction to this complexity, we designed a new abstraction that allows us to express the simple computations we were trying to perform but hides the messy details of parallelization, fault-tolerance, data distribution and load balancing in a library. Our abstraction is inspired by the *map* and *reduce* primitives present in Lisp

→ NoSQL

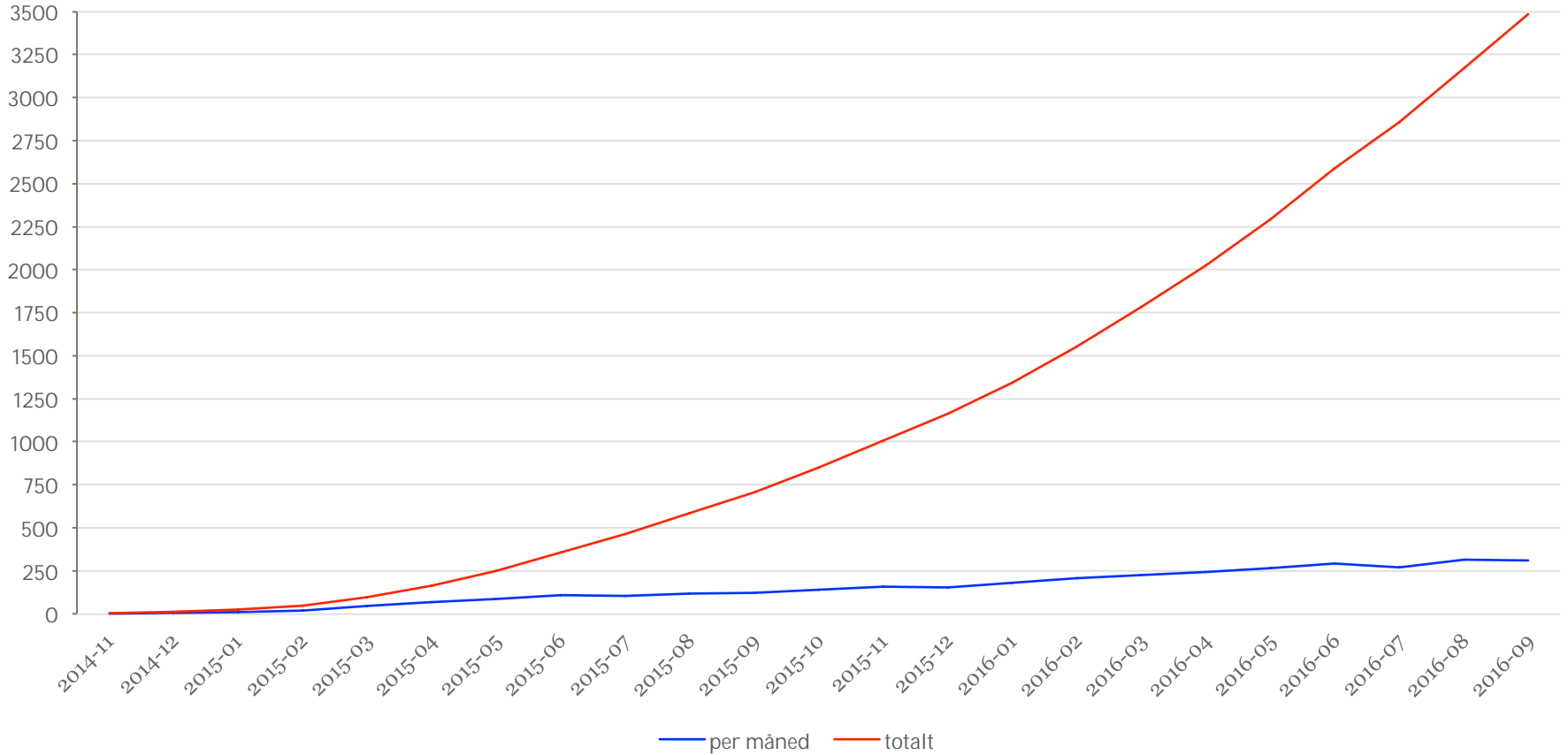
Kvalitetssikring i praksis

TRØBBEL PÅ UTVIKFJELLET



Store data? Raske data?

Kjøretøypasseringer (mill)



BEKK

SPØRSMÅL OG SVAR

Kristoffer Dyrkorn, BEKK

Kontaktperson, Statens vegvesen: Lars Meisingseth