max planck institut
informatik

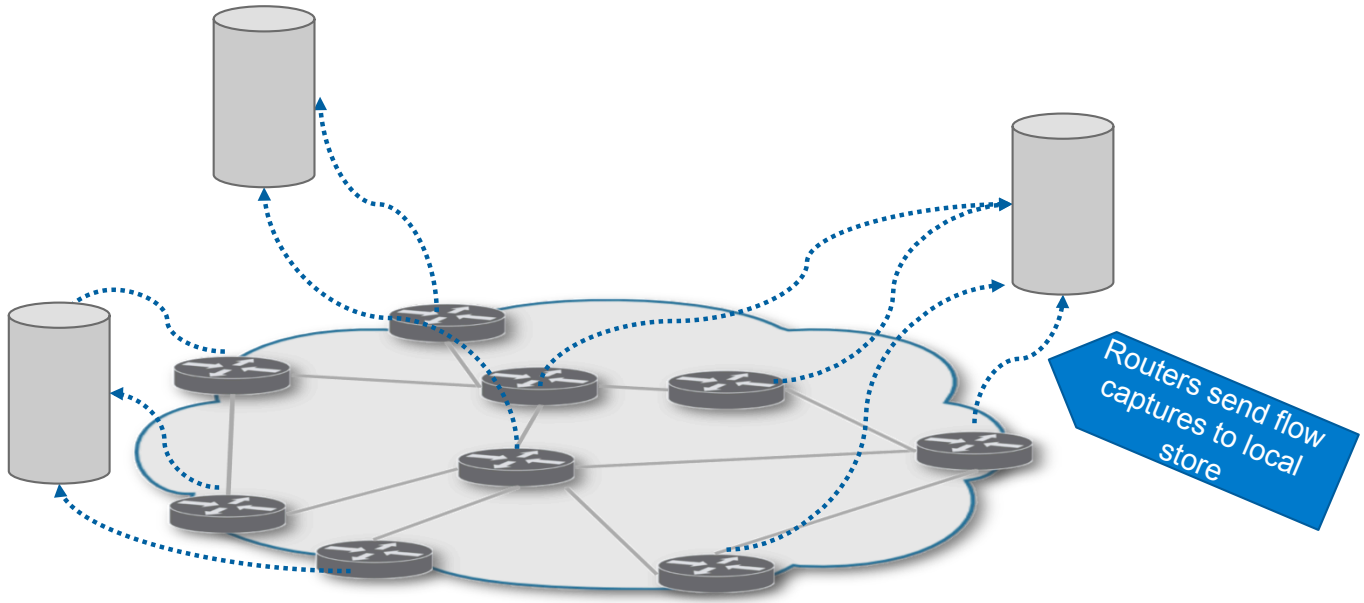
Distributed Mega-Datasets: The Need for Novel Computing Primitives

Anja Feldmann (MPI for Informatics, Saarbrücken)

Niklas Semmler, Georgios Smaragdakis (TU Berlin)

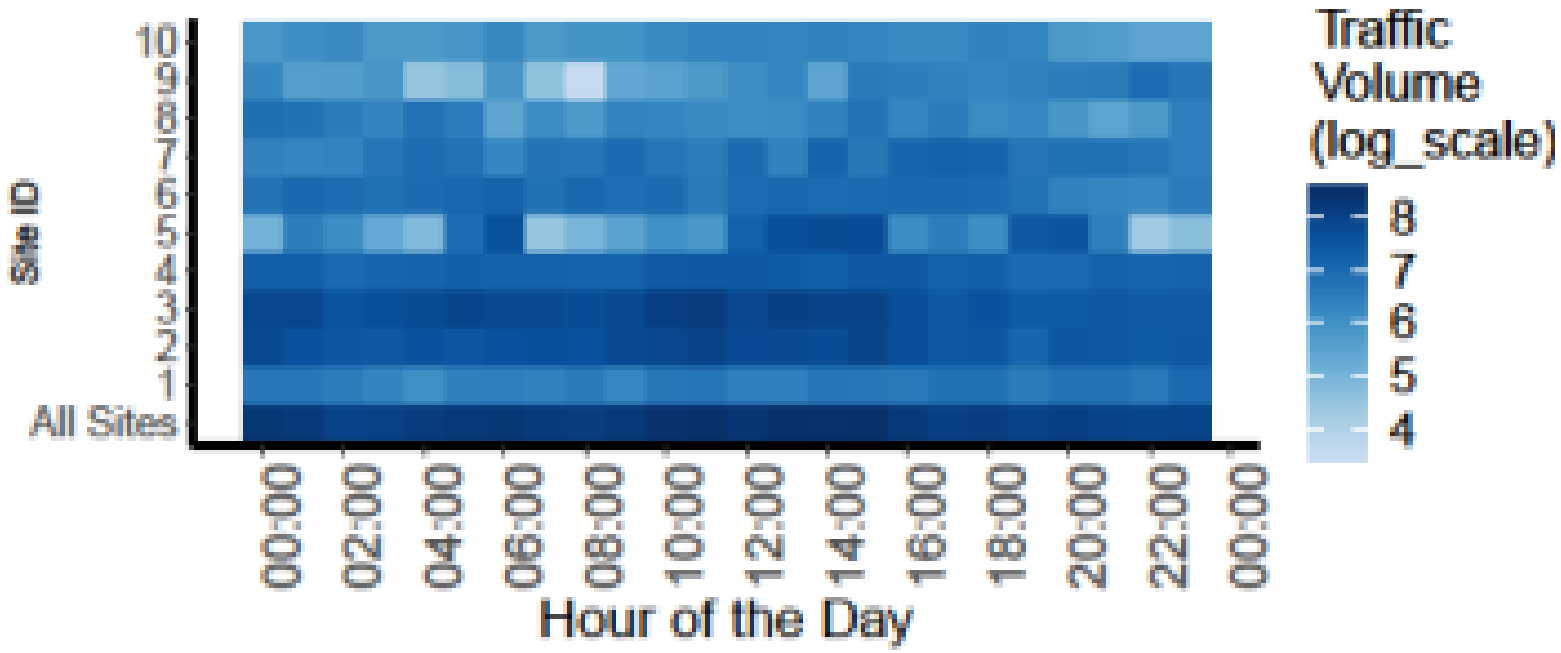
My motivation – Internet measurements

- Focus: Sampled Flow summaries collected per interface
- Collected at regional compute/storage nodes
- Analysis typically done using custom task specific scripts



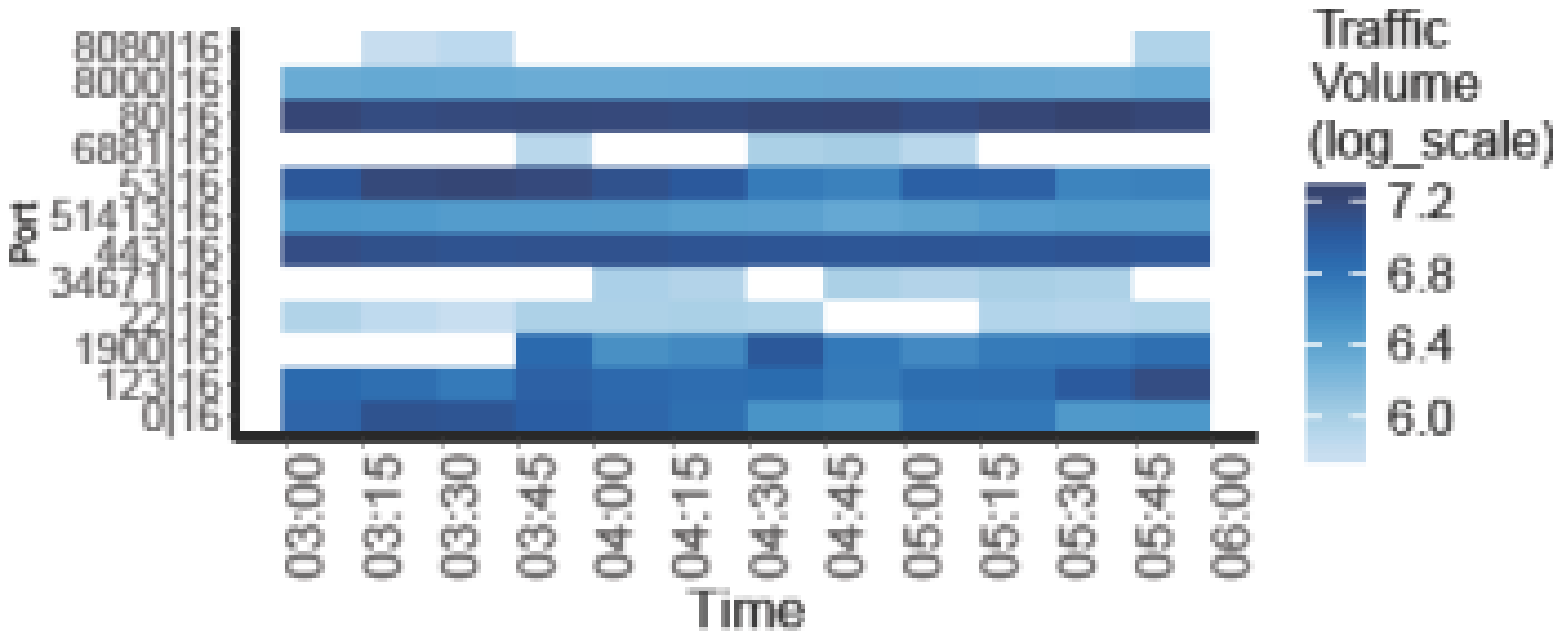
My motivation – Example analysis using IXP data

- Customer: Opens ticket for increased NTP attack traffic
- IXP: Investigate NTP traffic per site



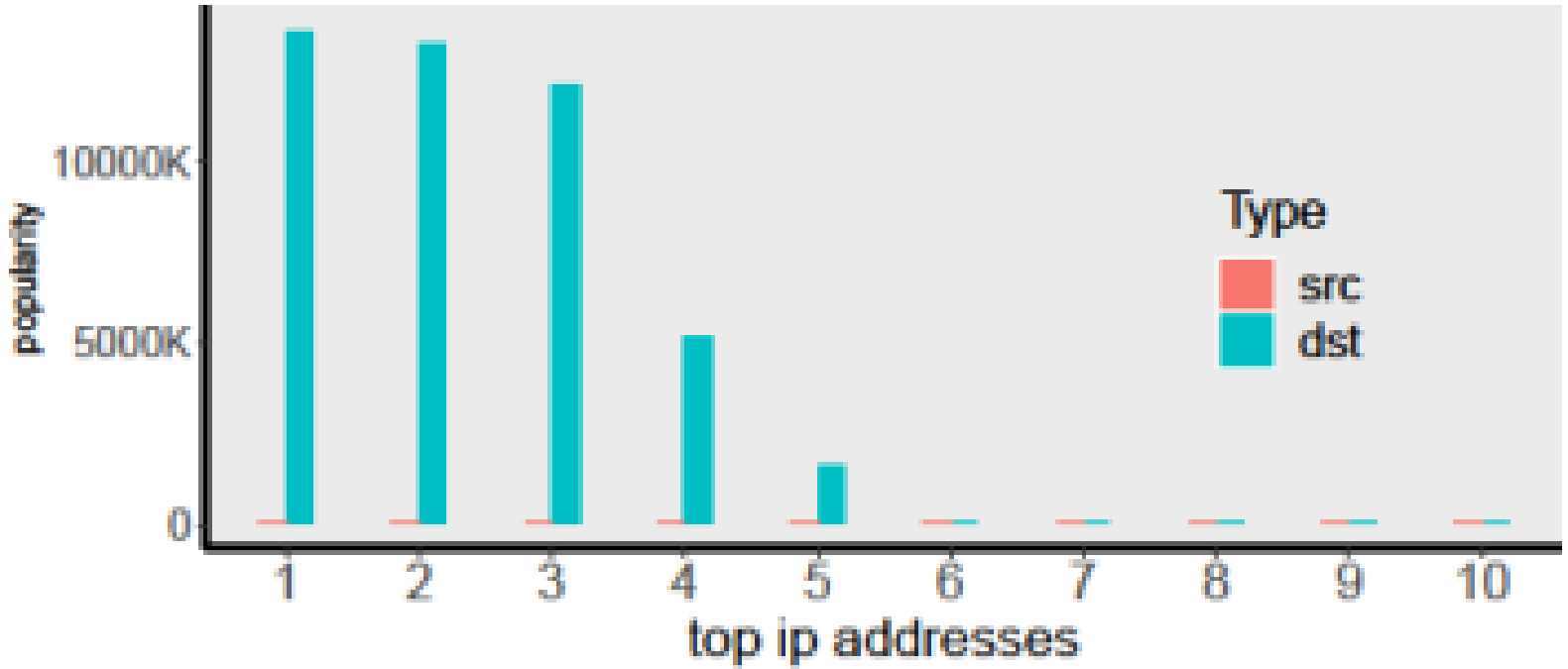
My motivation – Example analysis using IXP data

- Customer: Opens ticket for increased NTP attack traffic
- IXP: Investigate traffic per port usage for site 5 in detail



My motivation – Example analysis using IXP data

- Customer: Opens ticket for increased NTP attack traffic
- IXP: Locates targeted IPs of NTP traffic

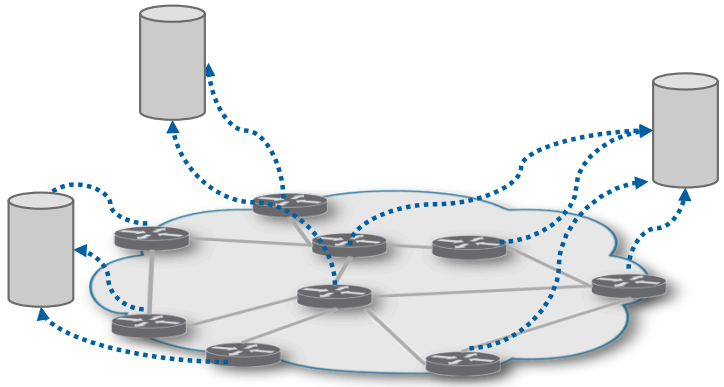


Motivation – Network operator questions: Examples

- Detect and investigate DDoS incidents, i.e.:
  - Which network part is affected
  - What are the involved sources
- Determine network trends, i.e.:
  - Popular network applications
  - Popular traffic sources/destinations
  - Aggregate flow statistics across time and sites
- Determine traffic matrix
- Traffic analysis
  - Top-K flows in terms of dst/src ports or IPs
  - Top hierarchical heavy hitters of …
  - Super-spreader detection
  - Heavy changer detection

Distributed Mega-Datasets

- Dataset
  - Single or multiple sources – limited in data volume and scope
- Mega-dataset
  - Dataset that can no longer be fully stored and/or processed within a single computer system
  - Can be handled local, e.g., a cluster of computer nodes
- Distributed mega-dataset
  - Physically distributed mega-dataset



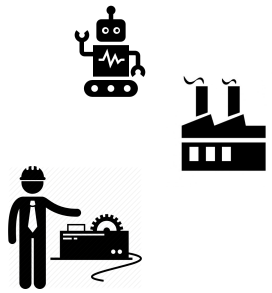
Use case 1: Internet monitoring

- Data
  - Massive streams of flow data
  - Different data sources, logs, flows, packets ➔ Distributed mega-datasets
- System
  - Spread out over a hierarchy of processes and resources
  - Subject to resource restrictions
- Analytics
  - Network state
  - Traffic engineering / Provisioning
  - Attack mitigation vs. load balancing

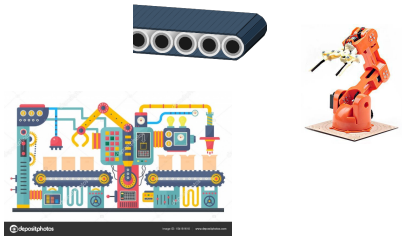
Queries: A-priory unknown

Use case 2: Smart factory

- Traditional Factory:
  - Single-purpose machine with rigid sequence of instructions
  - Interactions with other machines: e.g., by moving goods via conveyor bell
  - Frequent human interventions

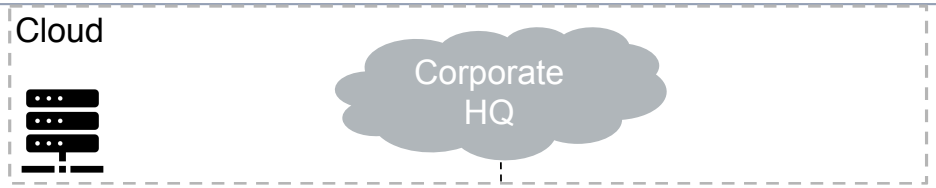


- Changes
  - Robotics, autonomous forklifts, automation, …
  - Multitude of both high- and low--resolution sensors
  - Process automation



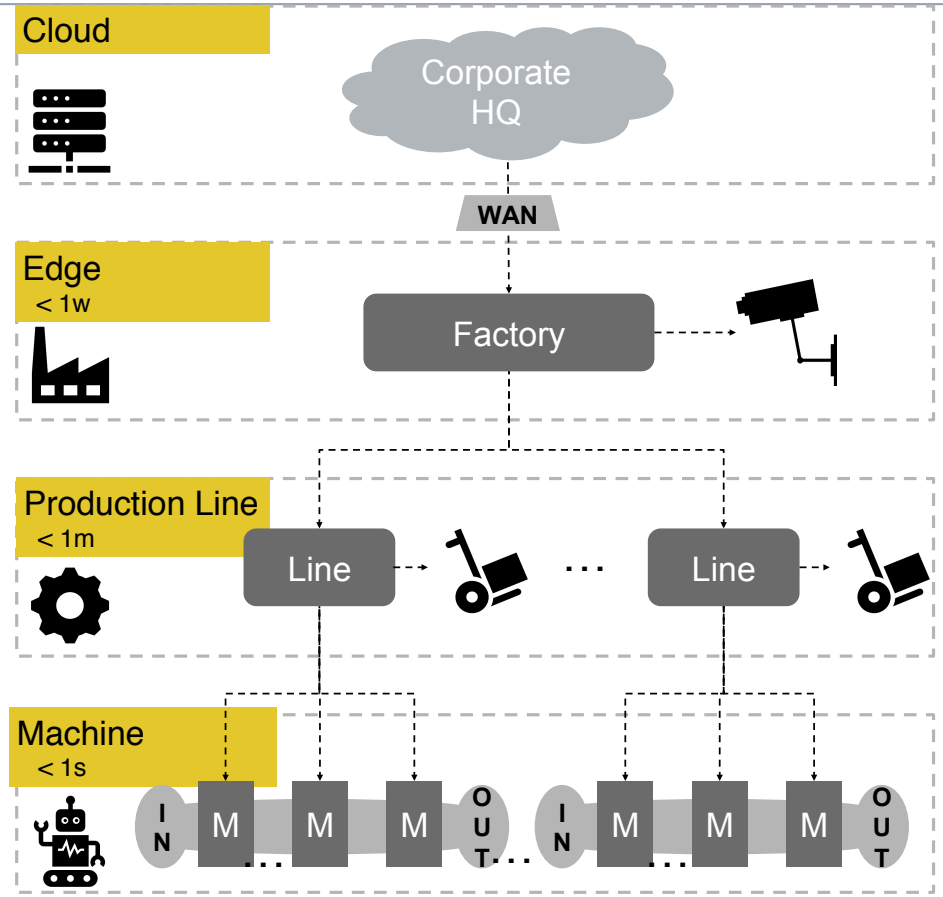
Use case 2: Smart factory – hierarchical organization

- Company
  - Corporate HQ
- Factory
  - Organizational unit
- Production line
  - Typically multiple per factory
  - Single or multiple assembly lines
- Machines
  - Typically multiple per production line
  - Complex data sensors



Use case 2: Smart factory – different timing requirements

- Company
  - Corporate HQ
- Factory
  - Organizational unit
- Production line
  - Typically multiple per factory
  - Single or multiple assembly lines
- Machines
  - Typically multiple per production line
  - Complex data sensors



Use case 2: Smart Factory

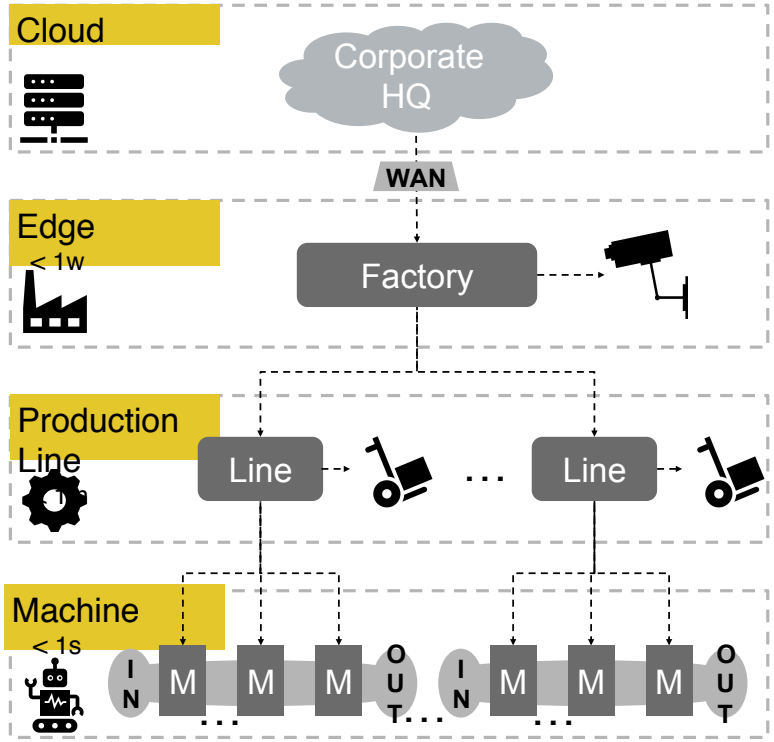
- Data
  - Massive streams of sensor data, i.e., high-resolution camera feeds
  - Many different data sources – per machine, per factory
- System
  - Machine control
  - Hierarchical structure: Machine, production line, factory
- Analytics
  - Status of production
  - Predictive maintenance
  - Maintenance vs. process optimization

Queries: A-priory unknown

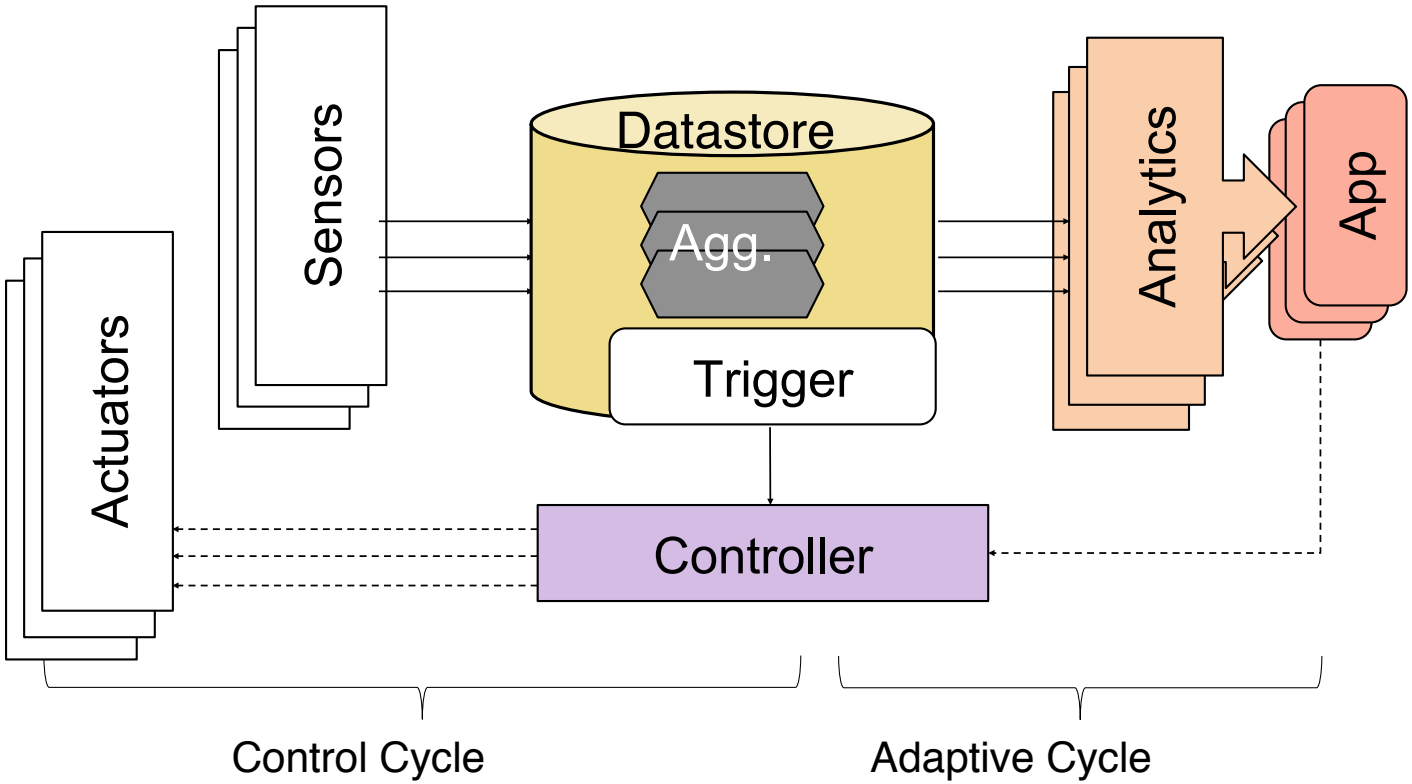
Challenges

- Large number of devices producing data streams
- Rapid local decision making
- Hierarchical structure
- A priori unknown queries
- Increasing computational requirements
- Massive combined data rates
- High data variability
- Analytics require full knowledge
- Varying requirements across applications

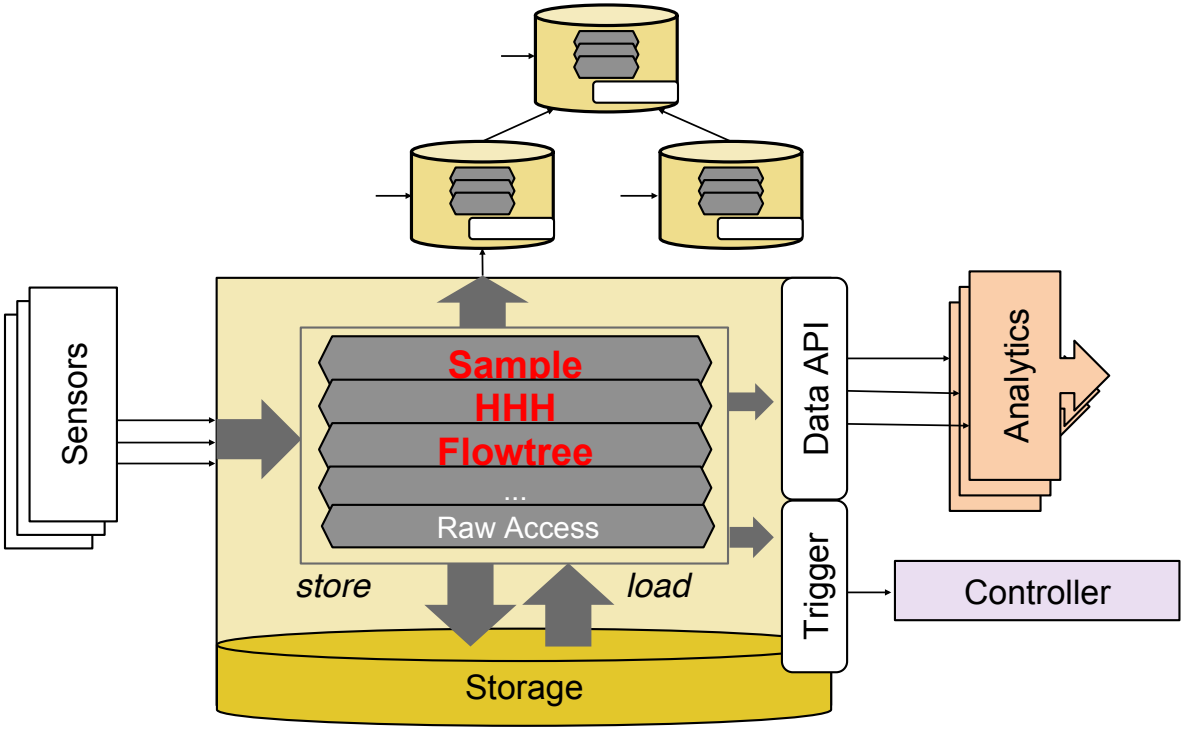
Proposed architecture



Data aggregation architecture: Single element



Datastore: A closer look



Computing primitives

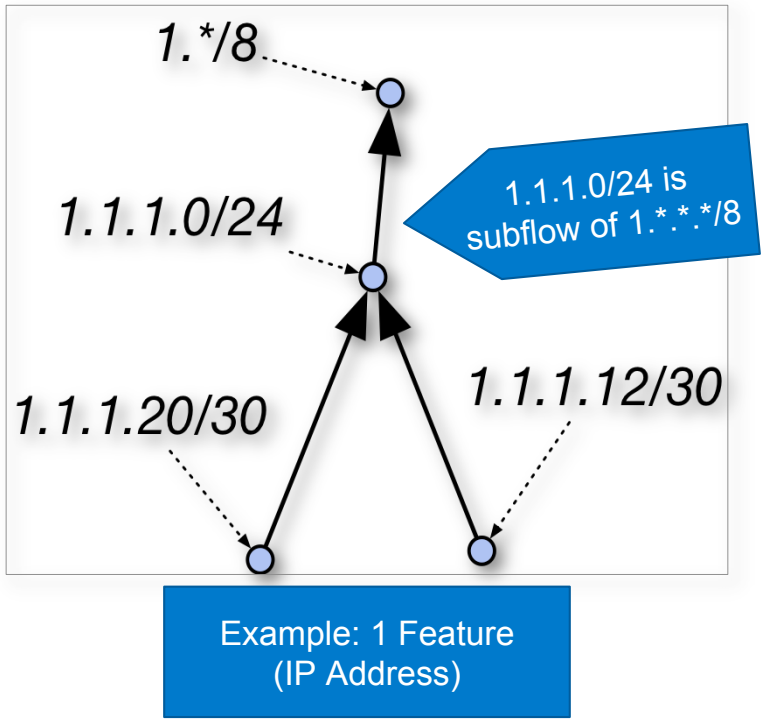
- Task
  - Efficiently summarize data hierarchically across multiple sources and locations to answer a priory unknown queries

- Design properties
  - Support arbitrary queries
  - Ability to combine summaries
  - Adjustable level of aggregation granularity
  - Self-adaptive
  - Domain adaptive (can take advantage of domain knowledge)

Flowtree: An example for a computing primitive

- A hierarchical self-adjusting data structure

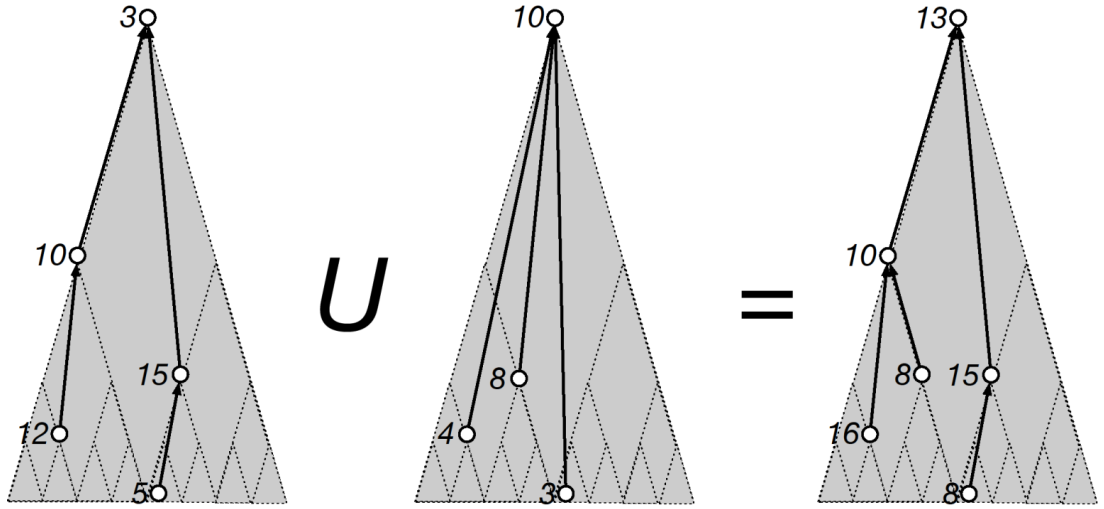
- Use natural hierarchies on network flows
  - Generalize flows (using wildcards):
    - IP address is part of an IP prefix
    - Port is part of a port range
- Can use single or multiple features
  - Src IP address
  - Src port and src IP



Flowtree: Self-adjusting data structure

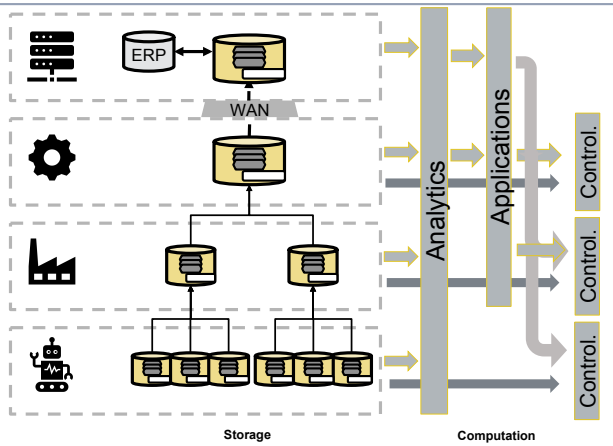
Properties

- Hierarchical heavy hitter detection (using any well-defined hierarchy)
- Supported operators:
  - Merge
  - Compress
  - Diff
  - Query
  - Drill down



Distributed Mega-Datasets: The Need for Novel Computing Primitives

- We are in the age of distributed mega-datasets
- Need an architecture for flexible processing



- Need novel computing primitives
  - Efficiently summarize data hierarchically across multiple sources and locations to answer a priory unknown queries