SEEP: Scalable and Elastic Event Processing

Matteo Migliavacca, Ioannis Papagiannis, Peter Pietzuch
migliava@doc.ic.ac.uk ip108@doc.ic.ac.uk prp@doc.ic.ac.uk
Imperial College London

David Eyers, Jean Bacon
David.Eyers@cl.cam.ac.uk Jean.Bacon@cl.cam.ac.uk
UNIVERSITY OF CAMBRIDGE

Brian Shand
Brian.Shand@cbcu.nhs.uk
NHS

Background

Motivation

- Continuous streams of event data occur in many applications
  - Healthcare, e-Government, e-Business, fraud detection, and logistics
- Existing analysis systems have generally focused on off-line analysis
  - Event based systems analyse data in real-time
- Centralised event-processing systems are reaching their limits

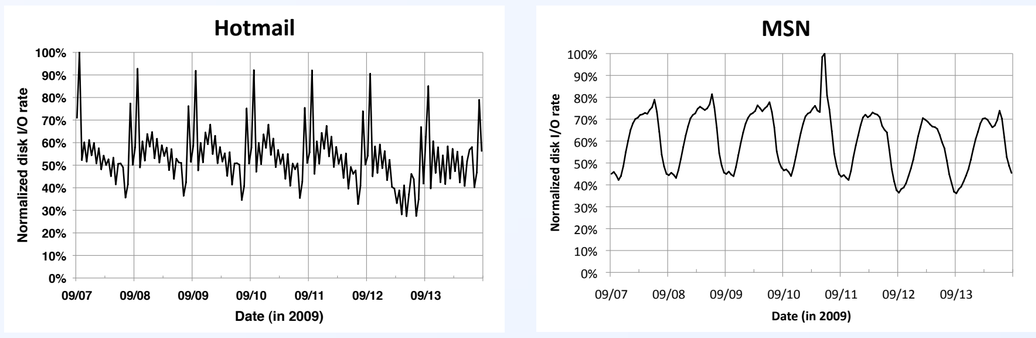
Open challenge: facilitate scaling up to thousands of machines in a cloud computing setting

Goals

1. **Scalability**: Develop architecture for event processing at cloud scale
2. **Elasticity**: Ensure that the deployed scale can change dynamically
3. **Adaptability**: Dynamic tuning of processing quality and speed or cost

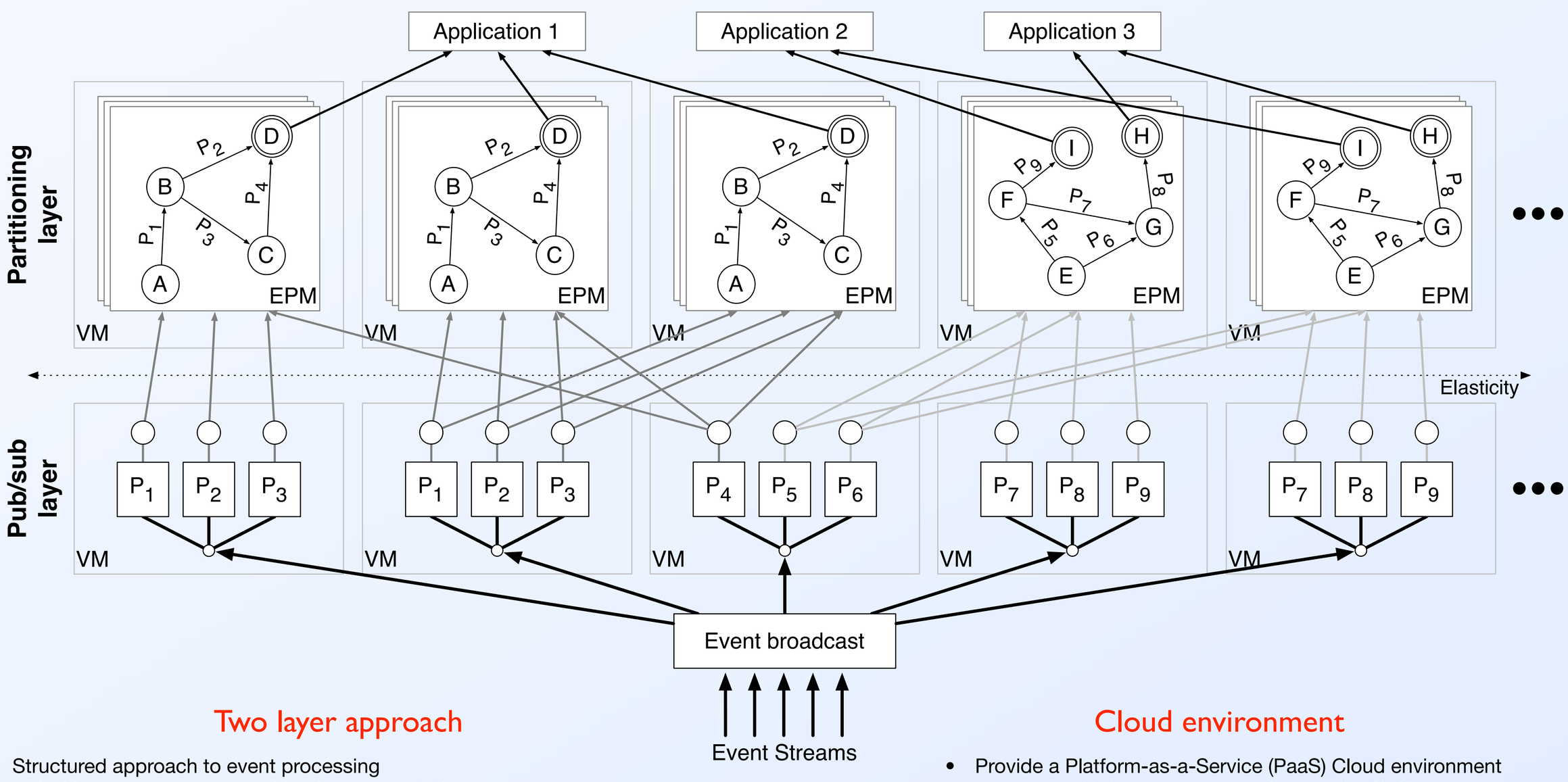
Workload Characteristics

- Existing usage of large-scale services has peaks and troughs
- Significant scope for improvement for **elasticity** and **adaptability**



Adapted from "Sierra: a power-proportional, distributed storage system", Eno Thereska, Austin Donnelly, and Dushyanth Narayanan. MSR-TR-2009-153, November 2009

SEEP Architecture



Two layer approach

- Structured approach to event processing
  - Architecture separates matching from computation
  - Reduces complexity for improved scalability
  - Two layers can scale independently

Publish/subscribe layer

- Incoming event streams broadcast to P/S layer nodes
- Large number of matching predicates ($P_1, P_2, ..., P_n$) on incoming events
- Matched events dispatched to appropriate VMs in partitioning layer
- Inverted index created over predicates to speed up event matching
  - Predicates composed from language for efficient indexing
  - Predicates indexed according to matched attributes, operator and value
  - Techniques from publish/subscribe literature can be reused
- By filtering first, the volume of events in the next layer can be reduced

Partitioning layer

- Event Processing Machines (EPMs) perform event processing
- EPMs implemented as non-deterministic FSAs
  - EPMs composed of detection / aggregation states
  - Each EPM instance contains state S derived by events matched so far
  - States linked by edge predicates (computed in P/S layer)
- When matched events dispatched to EPM
  - EPM makes transition to new state (or might be discarded)
  - Transition might generate new EPM instances (non-determinism)
  - Aggregation function incorporates the new event in S
  - When an accepting state is reached, state S is delivered to the application

Cloud environment

- Provide a Platform-as-a-Service (PaaS) Cloud environment
- Make use of Infrastructure-as-a-Service (IaaS) Cloud providers

Target features

- **Expressiveness**: provide detection of event sequences and aggregation
  - e.g. value of cumulative purchases above the average for that month
  - We do not aim to solve general purpose data analysis computations

- **Elasticity**: the system must be able to adapt to varying event rates
  - Could be through controlled decision: e.g. cost considerations
  - Might be through unforeseeable workload variations

- **Fault tolerance**: the system must be able to handle failures in VMs
  - Still must maintain throughput and/or latency bounds

Future Work

- Which applications best match the EPM model's expressiveness?
- How do extensions to the EPM model impact upon the architecture?
- How can persistence be best integrated into the SEEP architecture?
- Test large-scale, distributed deployment of the system
- Develop an open software platform for hosting SEEP applications