Automata-based Dynamic Data Processing for Clouds

Reginald Cushing, Marian Bubak, Adam Belloum, Cees de Laat

System Network Engineering
University of Amsterdam

BigDataClouds
25th August 2014
Outline

• Motivation
  – Traditional workflow process model
• Automata data processing model
  – Transition Function
  – Transition Operations
• Implementation
  – Data Routing
  – Process Congestion Avoidance
• Example Data State Graphs/Workflows
• Results
  – State Transitions
  – Process Congestion Avoidance
• Conclusion/Future Work
Motivation - 1

- How can we model data transformation during processing?
  - Workflows, etc model tasks and their respective execution order.
  - Provenance captured data lineage after processing.
- How can we model data processing agnostic from the underlying infrastructure?
  - Data processing schema
  - Workflows often fit the resources.
- How can we have collaboration in data processing between Vms/Clouds/Groups?
Motivation – 2

- The standard black box process:
  - $A \rightarrow [p] \rightarrow B$ ($A$ and $B$ are different entities)

- What if:
  - $A \rightarrow [p] \rightarrow A'$ (both input and output are $A$ but in different states)
  - This allows us to reason about data using state graphs.
  - Data processing is described as a sequence of state transitions.
Motivation - 3
Motivation - 3
If the side effect of processing data object is changing its state then an automata can represent the possible state a data can be in.

Specifically we use an NFA which is a 5 tuple:

- \( (Q, \Sigma, \delta, q_0, F) \)
- \( Q \): finite set of states
- \( \Sigma \): input alphabet
- \( \delta \): transition function
- \( F \): set of final states

Input alphabet are functions.

The transition function is the process (black box) transforms data from state to state.
Instead of an input symbol, the transition function takes another function $\sigma$ as input. $\sigma$ takes data and state as input and produces transformed data and new state. $\sigma$ accepts multiple states and outputs multiple states thus it can perform AND/OR operations on states.
Transition Operations

- State transition logic follows boolean algebra.
- Functions can output multiple states.
- Basic state combinations are AND, OR.
• Each VM/Node runs an identical copy of the stack.
• Data processing is decentralized.
• Communication adapters allow for P2P, MessageServer, File, HTTP, etc packet communication.
Processing Functions

- Each node can host a number of functions
- Each function performs a state transitions and data processing
- Functions process data packets
- Functions can optionally implement split/run/merge functions
  - Split: splits data packets into many packets
  - Run: processes the data packets
  - Merge: merges the data packets
Data Packet Split/Merge

VM1

Function A

split()

run()

merge()

VM2

Function A

run()

VM3

Function A

run()
Data Routing

- Each node builds a state routing table
- Depending on the routes data can be partitioned amongst replica functions or replicated to different functions.
  - Data replicated to S1:func1, S1:func2 but split between endpoint list of each.

<table>
<thead>
<tr>
<th>State Tag</th>
<th>Endpoint List</th>
</tr>
</thead>
</table>
Data partitioning on heterogeneous performing resources is not straightforward.
- Fine-grain partitioning increases parallelism, increases overhead (communication, etc).
- Coarse grain partitioning decreases overhead, decreases parallelism.
- Too much data can crash a VM.

Dynamically controlling granularity.
- Successively increment data payload using PRTT (Processing Round Trip Time).
- Reduce payload when degradation is detected.
Processing Efficiency

Efficiency = \frac{t_{\text{exec}}}{(t_{\text{now}} - t_{\text{timestamp}})}
The graph describing the allowable data transitions is also a data protocol header which allows data to be self contained and routable and processable.
MRI – State Graph
Combined State Network
State Transition Pipeline

- Creating a chain of state transitions on 11 Vms
- Each VM hosted 2 transitions: 1 Forward, 1 Backward
PCA – 1 Machine Shared Memory

- Functions on same node.
- Communication through shared memory.
- Stable Efficiency ~0.85
- Little change in efficiency after ~150 Data Records.
- Minimum comm overhead.
PCA – 2 VMs P2P

- Functions on different nodes on same virtual network.
- Communication P2P through ZMQ.
- Stable Efficiency ~0.65

- Little change in efficiency after ~1550 Data Records.
Functions on different nodes.
Communication through 3rd party message queue server.
Evidence of jitter.

- Efficiency increase but never reached plateau.
- Efficiency was degrading due to VM running out of memory.
Functions on different nodes 1 on local private cloud and the other on EC2.
Communication through 3\textsuperscript{rd} party message queue server.

Evidence of heavy jitter.
Strange result that we have two efficiency curves?
Conclusion/Future Work

• Data automata modeling provides
  – Better understanding of data transformations
  – Doubles as routing table for distributed architectures
  – Facilitates computing collaboration

• Data model can be considered as a schema. Data records can have an additional dimension, state, which results in a kind of 3D database.
Questions?

• REFs
  – https://github.com/recap/pumpkin