In-transit analytics on distributed Clouds: applications and architecture

Omer F. Rana
School of Computer Science & Informatics & Data Innovation Institute
Cardiff University, UK
ranaof@cardiff.ac.uk Twitter:@omerfrana

In collaboration with:
Manish Parashar, Javier Diaz-Montes, Mengong Zhou (Rutgers University, USA),
Ioan Petri, Tom Beach, Yacine Rezgui (Cardiff University),
Rafael Tolosana-Calasanz, Jose Banares (Univ. of Zaragoza, Spain)
Congduc Pham (University of Pau, France)
Cloud/Data Eco-System

- Increasing sensing capability closer to phenomenon being measured & increase volumes of “dynamic, distributed” data (IEEE P2413)
  - Capability to also undertake some processing on these devices
  - Increasing availability of programming support – “software defined environments”
Integrating Cloud Computing with Internet-of-Things

• “Cloud of Things” (CoT) and “Fog Computing”
  – Extending computing to the edges of the network
  – Overcoming latency constraints

• Real world/pervasive systems benefiting from Cloud infrastructure
  – Mobile & task off-loading (balancing energy usage with computation capability)
  – Internet-supported service convergence

• Significant heterogeneity in architectures and protocols for IoT
  – Device types and standards can vary significantly (e.g. iBeacons) -- development of “virtual sensors” (data reduction/fusion)
  – Often a data translation/mapping problem

• Projects:
  – Open Source IoTCloud (Sensors-as-a-Service): http://sites.google.com/site/opensourceiotcloud/
  – Open IoT (Middleware-oriented) – EU: http://www.openiot.eu/

• Commercial (mostly API based using HTTP/REST calls):
**The Lure of Clouds**

- An attractive platform for dynamic, real time service provisioning
  - Both for business & academia
- Cloud paradigm:
  - “Rent” resources as cloud services on-demand and pay for what you use
  - Potential for scaling-up, scaling-down and scaling-out, as well as for IT outsourcing and automation
  - Increasing support for dynamic deployment & configuration management
- Landscape of heterogeneous cloud services spans private & public clouds, data centers, etc.
  - Heterogeneous offering with different QoS, pricing models, availability, capabilities, and capacities
  - Variants: hybrid Clouds (“cloud bursting” & “cloud bridging”, Mobile off-loading, etc)
- Novel dynamic market-places where users can take advantage of different types of resources, quality of service (QoS), geographical locations, and pricing models
  - Various market models (on-demand, reservation, spot pricing, auctions, “Groupon”, etc)
- Cloud federations extend as-a-service models to virtualized data-centers federations
  - Bring Your Own Cloud (cf. Bring Your Own Device)

Based on slides from Manish Parashar (Rutgers Univ.)
Opening the “Network” Layer

L4/L7 capability in-network, not just L2/L3 (as currently done) – i.e. application-layer admission control, security (DPI), routing, etc

OpenFlow Switches + MiddleBox
Network Appliances (programmability)

Merge the Cloud and the network through in-network dynamic,

Abstract “in-network” en services
- Multiple data access and processing layers
- **Deciding what to do where – creation of a “decision function”**
- Different objectives: L3: power, range; L2: stream aggregation; L1: throughput
- **No need to migrate “raw” data to Cloud systems**
Data Analytics on Multi-Layered Clouds

• In-network capability:
  – Application driven, multi-node/capability driven

• Analytics:
  – In-situ (aggregation or capture site) – most common (e.g. Apache Spark, Hadoop, other in-memory, etc)
  – Data-drop (on-demand, “elastic”) – e.g. use of shared folders
  – In-transit (distributed, partial)

• New class of analysis algorithms
  – Resource-aware analytics (capacity, capability, availability)
    • Constraints influence types of analysis undertaken
  – Influenced by resource constraints (I/O, power, cost, historic performance)

• Workflow/Pipelines across layers
  – Dynamically adapt over time
  – Scale (in/out) with resource availability
  – Operation types vary in complexity & data size

AWS Lambda -- compute nodes charged by 100ms -- not the hour. First 1M node.js exec/month for free -- a monitoring challenge (http://aws.amazon.com/lambda/)
• Application scenarios for Federated Clouds
  – Analysis pipelines
• Modelling (abstractions) for federated Clouds
  – Use of Reference nets (a type of Petri net)
  – Model is directly executable
• Concluding scenario
  – Cloud-based building data analytics
Types of applications

• Variety of applications in multimedia streaming
  – Computational Science with sensor coupling – e.g. emergency response, security, environment, etc
  – Processing requirements vary – over different timeframes

• Not just true for physical sciences
  – increasingly social scientists also face similar challenges (e.g. tension indicators in communities)

• Increasing availability of data over the Web and from government departments
  – Data from Facebook, Twitter, Flickr (text, audio, video, etc)
    • People as sensors
  – Data from government agencies – Police API, Demographic data (ONS), etc
Application 1:

Data Streaming and Complex Event Processing

Bañares, José Ángel, Rana, Omer, Tolosana-Calasanz, Rafael and Pham, Congduc. "Revenue creation for rate adaptive stream management in multi-tenancy environments". Lecture Notes in Computer Science 8193, pp. 122-137. Springer.


Virtual Power Plants & Electric Vehicles

http://www.eandfes.co.uk/

A small sized Smart Grid

---

CEV: Cluster of EV
CP: Charging Point
DG: Distributed Generation
GENCO: Generation Company
HEMS: Home Energy Management System
SM: Smart Meter

---

Innovate UK
KAMFutures
Case study scenario

- Skelton building annual average electricity use per day

- 10 passenger cars (24 kWh battery), 5 maintenance vans (55 kWh battery)
- 20% battery SoC available for V2G
- Recharging for 20 miles per day in car EV and 40 miles per day in van
- Solar scenarios 0 kWp, 30 kWp, 60 kWp, 90 kWp

http://www.eandfes.co.uk/
Case study scenario

- Skelton building annual average electricity use per day

- 0 kWp Solar installation

http://www.eandfes.co.uk/
Case study scenario

- Skelton building annual average electricity use per day

- 30 kWp Solar installation

http://www.eandfes.co.uk/
Case study scenario

- Skelton building annual average electricity use per day

![Skelton Energy Use - Annual Average with V2G](http://www.eandfes.co.uk/)

- 60 kWp Solar installation
- Solar PV tracking
Case study scenario

- Skelton building annual average electricity use per day

- 90 kWp Solar installation
- Solar PV tracking

http://www.eandfes.co.uk/
Cloud-based Regional Brokerage (Virtual Power Plants)

Technology Partners

Demonstration Sites; Funded by;

Advisory Board;

- ecotricity
- electricity north west
- e.on
- CISCO
- MITSUBISHI MOTORS
- Sir Robert McAlpine
- Western Power Distribution

- CARDIFF UNIVERSITY
- PRIFYSGOL CERDYDD
- Msp manchester science parks
- Potenza Technology
- WMG Innovative Solutions
- cenex
- energy saving trust
- KAM Futures
- moixa TECHNOLOGY
Application 2:
Analysing social media data

Conejero, Javier, Rana, Omer, Burnap, Peter, Morgan, Jeffrey, Carrion, Carmen and Caminero, Blanca, “Characterising the power consumption of Hadoop Clouds: A social media analysis case study”. CLOSER 2013: 3rd International Conference on Cloud Computing and Services Science, Aachen, Germany, 8-10 May 2013.


Social Media Analysis

• Significant quantities of data generated from social media (… but “ethical” usage important)
  – Twitter: “firehose” (100%), “gardenhose” (10%), “spritzer” (1%)
  – Facebook status updates

• Integrating this data with other sources
  – ONS (in the UK) + other curated data
  – Maps related: (various options: Open Street Maps, Google Maps, Yahoo! Placefinder etc)

• Raw data not significant
  – Looking for particular types of “events” of interest

• Common analysis types
  – Sentiment and Opinion analysis
  – Connectivity between content generators

• Collaborative On-line Social Media Observatory (COSMOS)
  – “Tension” indicators in terrestrial and on-line communities
  – Integrating data with other (conventional) indicators

http://www.cosmosproject.net/
Usable – developed with social scientists for social scientists

Reproducible/Citable Research - export/share workflow

Integrated
Open (“plug and play”)
Scalable (MongoDB data stores/Hadoop Back End)

Observing Events (Boston)

http://www.cosmosproject.net/
Observing Events

http://www.cosmosproject.net/
COSMOS Infrastructure

**COSMOS Desktop**
- Small local datasets
- Users’ API credentials
- Local analysis
- Sept ‘14 launch
  (>100 dl’s in 17 countries)

**COSMOS Cloud**
- Scalable storage
  - Massive datasets
    (MongoDB)
- Scalable compute
  - On-demand nodes
  - Fast search & retrieve
  - Fast analysis
  - Indexing challenge
- Workflow management
- Collaboration support
- 2015 launch

---

**Data Collection**
- Persistent connection to
  Twitter 1% Stream (~4 billion)
- ONS/Police API
- Drag and drop RSS
- Import CSV/JSON

**Data Transformation**
- Word Frequency
- Point data frequency over time
- Social Network Analysis
- Geospatial Clustering
- Sentiment Analysis
- …API to plug new modules
  and benchmark tools
COSMOS: Architecture

Layer 1: Data generation (twitter feed) – can reduce captured data

Layer 2: COSMOS filters (gender or sentiment analysis)

Layer 3: Data analysis and integration with other sources (Police API, Demographic data (ONS), etc)

- COSMOS integrates a variety of different services:
  Gender analysis, sentiment analysis, Open Street maps
- Can be integrated with user supplied services
COSMOS: Architecture

Layer 1: Data generation (twitter feed) – can reduce captured data
Layer 2: COSMOS filters (gender or sentiment analysis)
Layer 3: Data analysis and integration with other sources (Police API, Demographic data (ONS), etc)

- COSMOS integrates a variety of different services:
  Gender analysis, sentiment analysis, Open Street maps
- Can be integrated with user supplied services


MODELLING COORDINATION IN MULTI-DIMENSIONAL PIPELINES
Common Theme: Pipelines

- Existence of “pipelines” – across multiple layers
  - Stream/In-Memory analysis
- Pipeline stages have different emphasis
  - Pre-Collect and store, data reduction, partial analysis, etc
- Data-driven pipeline execution
  - Inclusion of “sensing” into the pipeline
- Multiple, co-existing, concurrent pipelines
  - Superscalar pipelines
- Pipeline capability differs depending on Layer 1, 2 or 3
  - Resource availability & constraints

Abstractions: Data flow “process networks” (actors and firing rules); Coordination: Pub-Sub + Events, Tuple Space models; Implementations: Yahoo Pipes!, Storm (bolts and spouts, stream groups and topology), Pachube/Xively Cloud (Rate limited); Functional approaches: SCALA;Streamflow and Xbaya; Databases: EVE, Dequob, Calder; SummingBird (used with Storm and Scalding). Commercial: Amazon Kinesis/Lambda; Samza, Cascading, S4, Spark Cluster/Streaming, Google DataFlow/Millwheel; In Memory: Druid, VoltDB, MemSQL, NuoDB
Based on the notion of a “pipeline collision”

Pipelined Execution Semantics

1) Best Effort
   - Finished → Ready
   - Ready → Busy
   - Busy → Ready

2) Blocking
   - Finished → Blocked
   - Blocked → Busy
   - Busy → Ready

3) Buffered
   - Finished → Busy
   - Busy → Ready
   - Ready → Ready

4) Superscalar
   - Ready → Finished
   - Finished → Busy
   - Busy → Ready

5) Streaming
   - Busy → Busy → Busy

“Parallel Computing Patterns for Grid Workflows”
**Autonomic Streaming Pipeline**

- **Streaming pipeline**
  - No “blocking” semantics
  - Continuous data transmission as a stream
  - Data processing order: arrival order (implicit) or time stamp (explicit)
  - After processing – result elements form the stream

- **Autonomic streaming**
  - Data stream “reacts” to changes in (operating) environment and producer/consumer data generation/consumption rate mismatch
  - Network congestion $\rightarrow$ alter transmission data rate
  - Alternative modes of analysis: in-transit, at-source, at-sink, etc
In-transit Analysis

- Data processing while data is in movement from source to destination
- Question: what to process where and when
- Use of “slack” in network to support partial processing
- Application types:
  - Streaming & Data Fusion requirement

Flow diagram:

- S1 → T1 → T2 → D1
- S2 → T3 → T4 → T5 → D2

Delay (QoS parameter)
Workflow level Representation

PU₁

Proc. Unit: \( t_{11}, t_{12} \)

ADSS: \( t_{21}, t_{22} \)

PU₂

Proc. Unit: \( t_{21}, t_{22}, t_{23} \)

Resource \( r_1: t_{11}, t_{12} \)

Resource \( r_2: t_{21}, t_{22}, t_{23} \)

ADSS: (with in-transit processing)

ADSS model & simulator

Buffer

Controller

local storage

\( \lambda \) Input rate

\( \mu \) Controlled output rate

\( \omega \) Disk transfer rate

\( \delta \) Consumer’s data rate

\( B \) Bandwidth

In transit nodes: \( t_{21}, t_{22} \)

Mapping

Data transfer

\([\ldots]\)
“Nets-within-nets”
- Systems net and an object net
- Net can express creation of new net instances ("creational inscriptions") – enabling dynamic self-modification of structure
- Interaction via "synchronous channels"
- Channel can contain variables whose binding is based on unification
- Timed reference nets:
  - Time stamp attached to tokens
  - Use of timed inscriptions on arcs (control time stamps and firing delays)

Renew
- Java-based interpreter of Reference nets (an executable formalism)
- Use tuples and Java expression as the inscription language
- Objects nets can be Java objects
Reference nets

- **Petri nets**
  - directed bipartite graph
  - 2 types of nodes: places and transitions
  - arcs: place-transition, transition-place
  - tokens: move on the graph
  - static structure

- **Reference nets [1]**
  - tokens can be nets $\rightarrow$ workflow hierarchies
  - tokens can be data $\rightarrow$ data flow
  - Synchronous channels:
    - synchronise two transitions across different nets which both fire atomically at the same time
    - Both transitions must agree on the name of the channel and on a set of parameters before they can synchronise

A stage in the pipeline – demonstrating the processing unit and the ADSS

Operations to be performed at this stage

Abstract workflow

Data

Number of resources

Buffer Size

Operations that can be performed in transit

Operations that can be performed in transit

Abstract workflow

Number of resources

Buffer Size

Pu to process in stage

Linda Spaces

Pu for In transit processing
Control Strategy + Adaptation

- Reference net model executes alongside real system
- Model used to tune behaviour
- Rule-based Reasoner coupled with other machine learning strategies

Rafael Tolosana-Calasanz, José A. Bañares, Omer F. Rana: *Autonomic streaming pipeline for scientific workflows.* Concurrency and Computation: Practice and Experience 23 (16): 1868-1892 (2011)
Adapting Transfer Rates based on Network Congestion

Lamda: data generation rate; B: bandwidth; omega: hard disk transfer rate
Network congestion added: intervals 11-24; control interval: 10 secs.
Adding in-transit processing nodes

deltaU: change in processing rate (i.e. number of data items processed/time)

Rafael Tolosana-Calasanz, José Á. Bañares, Omer Rana, Congduc Pham, Erotokritos Xydas, Charalampos Marmaras, Panagiotis Papadopoulos and Liana Cipcigan,
Superscalar Pipelines and Rate Adaptation

R. Tolosana-Calasanz, J. B. Banares. C. Pham and O. Rana
“Enforcing QoS in Scientific Workflow Systems Enacted Over Cloud Infrastructures”
Journal of Computer and System Science (Elsevier), 2012
Isolating multiple concurrent pipelines

Multiple input streams with different QoS demands

- Manage input to guarantee QoS
- Manage local resources of PU to guarantee QoS
- Modify mu, omega, select routes, add in-transit proc.
Two key parameters of interest:

- **R**: Also called the **committed information rate** (CIR), it specifies how much data can be sent or forwarded per unit time on average.

- **b**: It specifies for each burst how much data can be sent within a given time without creating scheduling concerns. **Tokens in excess are normally dropped.**
Token Bucket model allows for variable rate processing with bounded traffic envelop. Each token bucket provides us tunable parameters: b, R.

Controller: monitors & modifies behaviour.

Integrating Token Bucket Into Model
## Example Rules

<table>
<thead>
<tr>
<th>Rule no.</th>
<th>Pattern</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>E</strong>: $B_i$ over threshold; <strong>C</strong>: Enable use of free resources</td>
<td>$\Delta R_i = \sum_{i=1}^{n} NumRes_i \cdot \delta_i - \sum_{i=1}^{n} R_i$</td>
</tr>
<tr>
<td>2</td>
<td><strong>E</strong>: $B_i$ over threshold; <strong>C</strong>: Enable drop of $D_i$</td>
<td>$B_i = B_i - D_i$</td>
</tr>
<tr>
<td>3</td>
<td><strong>E</strong>: $B_i$ below threshold; <strong>C</strong>: Control Stream</td>
<td>$\Delta R_i = 0$</td>
</tr>
</tbody>
</table>

### Ranges of QoS control

<table>
<thead>
<tr>
<th>Rule no.</th>
<th>Pattern</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td><strong>E</strong>: $\sum_{i=1}^{n}(\lambda_i - R_i)$ over threshold; <strong>C</strong>: Borrow $N_i$ resources</td>
<td>$\Delta NumRes = \min\left(\sum_{i=1}^{n} N_i, \sum_{i=1}^{n}(\lambda_i - R_i)/\delta_i\right)$</td>
</tr>
<tr>
<td>5</td>
<td><strong>E</strong>: $\sum_{i=1}^{n}(\lambda_i - R_i)$ over threshold; <strong>C</strong>: Pause low priority flows</td>
<td>$#Paused_{LowLevel} = \sum_{i=1}^{n}(\lambda_i - R_i)/\delta_i$</td>
</tr>
<tr>
<td>6</td>
<td><strong>E</strong>: Overthrow; <strong>C</strong>: Control Stream</td>
<td>$\Delta NumRes = 0, #Paused_{LowLevel} = 0$</td>
</tr>
</tbody>
</table>

---

R. Tolosana, J. Banares, C. Pham, O. Rana,
Resource Management Strategy

- Buy remote resources (from other Cloud provider)
- Allocate new local resources (launch new VMs)
- Redistribute pre-allocated resources from less-prioritized users
- Redistribute unused resources

José Ángel Bañares, Omer F. Rana, Rafael Tolosana-Calasanz, Congduc Pham: Revenue Creation for Rate Adaptive Stream Management in Multi-tenancy Environments. Int. Conf on Economics of Grids, Clouds, Systems & Services (GECON 2013), pp 122-137, Zaragoza, Spain, Springer
Unified resource mngt with TB

- Redistribute unused resources

- Under-utilization of resource in a flow will produce tokens in excess

- Within a service class,
  - Tokens in excess of all flows are collected and stored up to $B_{\text{max}}$ tokens
  - Token’s lifetime is limited to a few control intervals to limit inconsistency
Silver class has higher revenue and higher penalty than Bronze class for example: shortage of resource in Silver class is more costly.

Taking resources from Bronze to Silver is more revenue-efficient:
- Tokens are taken directly from a Bronze flow’s token bucket
- Can put a limit to the number of tokens the system can take

Safer than taking tokens from the Bronze unused token bucket.
ADDITION OF RESOURCES FOR GOLD CUSTOMERS
Simulation results – throughput

Second Scenario: Input of a data stream with and non-provisioning

Input/output in data/second

Time (sec)
Integration with OpenNebula

Traffic shaping achieved through the use of a Token Bucket manager. A token bucket for each data stream.

Monitoring number of accumulating packets in an intermediate buffer – triggers creation of new VM instances.

OpenNebula 4.4 (32 physical nodes), 32GB/node, 8 cores/node. 4VMs (8VMs) send packets to 20VMs (40VMs) with 5 processes/VM, at 400 packets/s.

<table>
<thead>
<tr>
<th>Packet size</th>
<th>20 VMs receiving packets</th>
<th>40 VMs receiving packets</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MEAN (msecs)</td>
<td>STDEV</td>
</tr>
<tr>
<td>8Kb</td>
<td>2.00</td>
<td>0.42</td>
</tr>
<tr>
<td>16Kb</td>
<td>2.00</td>
<td>3.09</td>
</tr>
<tr>
<td>33Kb</td>
<td>3.00</td>
<td>12.98</td>
</tr>
<tr>
<td>65Kb</td>
<td>3.00</td>
<td>21.45</td>
</tr>
</tbody>
</table>

VM cross talk and dynamic VM creation.
Managing VMs
Real Execution with OpenNebula VMs
Aggregate input & output with 4 baseline VMs and elastic provisioning

Real Execution with OpenNebula VMs: input & output of a sample stream with 4 baseline VMs and elastic provisioning
Triggering VM launch with data buffer occupancy

Number of VMs & PU Buffer Occupancy

- bPUds1
- bPUds2
- bPUds3
- bPUds4
- #VM
Comparative Simulation/Real VMs

Simulated: Aggregated Traffic

OpenNebula: Aggregated Traffic
Building Energy Simulation – Concluding Scenario

MULTI-LAYERED FEDERATED CLOUDS
CometCloud-based Multi-Layered Federated Clouds

Multiple data access and processing layers
Deciding what to do where – creation of a “decision function”
Different objectives: L3: power, range; L2: stream aggregation; L1: throughput
(use of “Software Define Networks” at L2)
No need to migrate “raw” data to Cloud systems
On-Demand Federation using CometCloud

- Cross-layer federation management using user and provider policies
- Federation is coordinated using Comet spaces at two levels

- Management space
  - Orchestrate resources in the federation
  - Interchange operational messages

- Shared execution spaces
  - Created on demand by agents
  - Provision local resources and connect to public clouds or external HPC systems

Overview of the CometCloud Space

• Virtual shared space abstraction
  – Based on application properties
  – Mapped onto a set of peer nodes

• The space is accessible by all system nodes.
  – Access is independent of the physical locations of data tuples or hosts

• Coordination/interaction through the shared spaces
  – Runtime management, push/pull scheduling and load-balancing

• Dynamically constructed transient spaces enable application to exploit context locality
Implementation

• Requirements for a site to join the federation:
  – Java support
  – Valid credentials (authorized SSH keys)
  – Configure some parameters (i.e. address, ports, number of workers)

• Resources

<table>
<thead>
<tr>
<th>Resources</th>
<th>Cardiff</th>
<th>Rutgers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machines</td>
<td>12</td>
<td>32</td>
</tr>
<tr>
<td>Core per Machine</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>Memory</td>
<td>12 GB</td>
<td>6 GB</td>
</tr>
<tr>
<td>Network</td>
<td>1 GbE</td>
<td>Infiniband</td>
</tr>
</tbody>
</table>

• Indiana site
  – Uses FutureGrid (OpenStack, Infiniband interconnect, 2 cores/machine with 4GB memory) – also supports Cloudmesh Teefaa and Rain
• Real time optimisation of building energy use
  – sensors provide readings within an interval of 15-30 minutes,
  – Optimisation run over this interval
• The efficiency of the optimisation process depends of the capacity of the computing infrastructure
  – deploying multiple EnergyPlus simulations
• Closed loop optimisation
  – Set control set points
  – Monitor/acquire sensor data + perform analysis with EnergyPlus
  – Update HVAC and actuators in physical infrastructure

Pool (indoor) – size: 25m x 16m, depth: 1,60m to 2,10m, Capacity: 760 m³
Learning Pool (indoor) – size: 16m x 4 m, depth: 1m, Capacity: 64 m³
1 Gym (indoor) provided of electric equipment (electric bicycles, etc…) 
1 Fitness room (indoor) size: 18m x 9m x 3m, Volume: 486m³
1 Volleyball court (indoor) – size: 40m x 28m x 8m, Volume: 8960 m³
2 Tennis/Five-a-side courts (outdoor, with changing rooms) – size: 30m x 20m
Federated Clouds in Building Optimisation

## FIDIA Scenario 1

### Input

- **Time:** 13:31:02
- **Date:** 2014-02-04
- **Occupancy:** 251
- **Indoor Relative Humidity:** 88.2%
- **Current Room Temperature:** 24.05°C
- **Pool Water Temperature:** 29.39°C
- **Supply Air Flow Rate:** 6.69 m³/s

### Table

<table>
<thead>
<tr>
<th>Objective</th>
<th>Variables</th>
<th>Sensors/Meters</th>
<th>Units</th>
<th>Type</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input for Optimisation</td>
<td>Occupancy</td>
<td>Occupancy Sensor</td>
<td>-</td>
<td>TPS210: People counter</td>
<td>Modbus IP</td>
</tr>
<tr>
<td></td>
<td>Indoor Temperature</td>
<td>Temperature sensor (Battery powered)</td>
<td>deg. C</td>
<td>iPoint-T: Air T &amp; RH sensor</td>
<td>Modbus IP</td>
</tr>
<tr>
<td></td>
<td>Water Temperature</td>
<td>Temperature sensor</td>
<td>deg. C</td>
<td>STP100-100: Water T sensor</td>
<td>I/O to AS</td>
</tr>
<tr>
<td></td>
<td>Indoor Humidity</td>
<td>Humidity sensor (Battery powered)</td>
<td>deg. C</td>
<td>SHO100-T: Air RH sensor</td>
<td>I/O to AS</td>
</tr>
<tr>
<td></td>
<td>Air Temperature Inlet</td>
<td>Temperature sensor (Battery powered)</td>
<td>deg. C</td>
<td>iPoint-T: Air T &amp; RH sensor</td>
<td>Modbus IP</td>
</tr>
<tr>
<td></td>
<td>Supplied Air Flow Rate</td>
<td>Velocity sensor</td>
<td>kg/s</td>
<td>TI-SAD-65: Air velocity Sensor</td>
<td>I/O to AS</td>
</tr>
<tr>
<td>Output of Optimisation</td>
<td>PMV (comfort level)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Electrical Energy</td>
<td>Electricity Meter (220-240 HVAC)</td>
<td>Kwh</td>
<td>iMeter: Electric meter</td>
<td>Modbus RS485 %</td>
</tr>
<tr>
<td></td>
<td>Additional Parameters</td>
<td>Carbon Concentration (CO2)</td>
<td>ppm</td>
<td>CO2 duct sensor</td>
<td>I/O to AS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chlorine in Air</td>
<td>ppm</td>
<td>Murco MGS: Air CI2 sensor</td>
<td>Modbus RS485</td>
</tr>
</tbody>
</table>
EnergyPlus and Building Optimisation

Federation constraints

Two metrics:
- Time to complete
- Results quality
Trading quality of results vs. overall simulation time

• Each Master decides how to compute the received job:
  – (i) where to compute the tasks: (a) Single CometCloud or (b) federated CometCloud;
  – (ii) how many combinations to run giving the deadline received from the user.

\[ \text{cost function: } f(X) : C \to R \text{ where } C \text{ is a set of constraints (cost, deadline) and } R \text{ is a set of decisions based on the existing constraints } C. \]
Evaluation

• In our experiments we use two different configurations
  – (a) _single cloud context_ where all the tasks have to be processed locally
  – (b) _federation cloud context_ where the sites have the option of outsourcing tasks to remote sites.

• We use as inputs for our calculation
  – (i) _CPU time of remote site_ as the amount of time spent by each worker to compute the tasks and
  – (ii) _storage time on remote site_ as the amount of time needed to store data remotely.

• All the costs have been calculated in £ derived from Amazon EC2 cost.
the federation site has two options: (i) run tasks on the local infrastructure (single cloud case) or (ii) outsource some tasks to a remote site (federation cloud case). A corresponding deadline of 1 hour, only 34 out of 38 can be completed. In the federation in 55 minutes by outsourcing 15 to the remote site. The process of outsourcing has an associated cost of 7.46 £.
Experiment 2: Job uncompleted:

Table V: Input Parameter: Experiment 2

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>Deadline</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>{16,17,18,19,20,21,22,23,24}</td>
<td>{0,1}</td>
<td>{0,1}</td>
<td>{0,1}</td>
<td>1 Hour</td>
</tr>
</tbody>
</table>

Table VI: Results: Experiment 2

<table>
<thead>
<tr>
<th></th>
<th>Single Cloud</th>
<th>Federated Cloud</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>Cost</td>
<td>0</td>
<td>£ 7.90</td>
</tr>
<tr>
<td>Tasks</td>
<td>72</td>
<td>72</td>
</tr>
<tr>
<td>Deadline</td>
<td>1 hour</td>
<td>1 hour</td>
</tr>
<tr>
<td>Tuples exchanged</td>
<td>-</td>
<td>15</td>
</tr>
<tr>
<td>CPU on remote site</td>
<td>-</td>
<td>5637.27 Sec</td>
</tr>
<tr>
<td>Storage on remote site</td>
<td>-</td>
<td>1869.41 Sec</td>
</tr>
<tr>
<td>Completed tasks</td>
<td>37/72</td>
<td>58/72</td>
</tr>
</tbody>
</table>

- In the context of single cloud federation (3 workers) only 37 out of 72 tasks are completed within the deadline of 1 hour.
- Exchanging 15 tuples between the two federation sites, with increased cost for execution and storage.
**Experiment 3: Job uncompleted–parameters ranges extended:**

<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>Deadline</th>
</tr>
</thead>
<tbody>
<tr>
<td>{14,15,16,17,18,19,20}</td>
<td>{0,1}</td>
<td>{0,1}</td>
<td>{0,1}</td>
<td>1h 30 min</td>
</tr>
<tr>
<td>{21,22,23,24,25,26,27}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table VIII: Results: Experiment 3**

<table>
<thead>
<tr>
<th></th>
<th>Single Cloud</th>
<th>Federated Cloud</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>Cost</td>
<td>0</td>
<td>£ 10.70</td>
</tr>
<tr>
<td>Tasks</td>
<td>112</td>
<td>72</td>
</tr>
<tr>
<td>Deadline</td>
<td>1 h 30 min</td>
<td>1 h 30 min</td>
</tr>
<tr>
<td>Tuples exchanged</td>
<td>-</td>
<td>22</td>
</tr>
<tr>
<td>CPU on remote site</td>
<td>-</td>
<td>7983.74 sec</td>
</tr>
<tr>
<td>Storage on remote site</td>
<td>-</td>
<td>2687.15 sec</td>
</tr>
<tr>
<td>Completed tasks</td>
<td>42/112</td>
<td>62/112</td>
</tr>
</tbody>
</table>

- we extend the deadline associated to 1 hour and 30 minutes
- when using the federation to outsource a percentage of tasks we observe that the number of tasks completed increases to 62
Summary of results
Integration with Software Defined Networks

SDN emulation using Mininet

Use of three sites (Rutgers (New Jersey, US), Cardiff (UK) and FutureGrid (Indiana, US) to simulate use of multi-hop interaction).

Different cost of execution per site + cost of network data transfer
• We consider that building data is available at each site, with data being generated at different rates.

• The amount of input data to be transferred can be 10MB, 20MB, or 30MB.

• We assume SDN capabilities are available across all sites.

• We allocate five SDN channels between each pair of sites with a guaranteed bandwidth of 1 Mbps.

• We also have a network channel without QoS guarantees, called shared channel, that has a bandwidth of up to 0.2 Mbps.

The policy used in our experiments is selecting the site that can complete the workload with the minimum Time to Completion (TTC) subject to Cost < Budget.

If a job cannot be completed at any site within the given deadline and budget constraint, this job is declined.

The TTC of a job is DataTransfer + ComputationTime.

The Cost is DataTransferCost + ComputationCost.

The shared network is free, while the cost of the SDN network varies based on utilization. The default cost of each SDN network channel is $0.05/second. This cost increases when utilization exceeds 50%.

The cost of the network is calculated as follows:

\[
Cost = BaseCost \times (1 + (\frac{ChannelsInUse}{TotalChannels} - 0.5) \times 2)
\]  
(1)
Table II: Total amount of time spent transferring shared network and SDN network.

<table>
<thead>
<tr>
<th>Transfer Time</th>
<th>Estimated</th>
<th>Real</th>
<th>Estir</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rutgers</td>
<td>560</td>
<td>4652</td>
<td>24</td>
</tr>
<tr>
<td>Cardiff</td>
<td>180</td>
<td>1588</td>
<td>4</td>
</tr>
<tr>
<td>Futuregrid</td>
<td>264</td>
<td>1489</td>
<td>6</td>
</tr>
</tbody>
</table>

Table III: Number of jobs outsourced computed locally.

<table>
<thead>
<tr>
<th>job Source</th>
<th>Outsourced</th>
<th>Local</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rutgers</td>
<td>49</td>
<td>74</td>
</tr>
<tr>
<td>Cardiff</td>
<td>96</td>
<td>40</td>
</tr>
<tr>
<td>Futuregrid</td>
<td>100</td>
<td>30</td>
</tr>
</tbody>
</table>

Figure 4: Average Execution and Network transfer time for local site and outsourcing to a remote site with and without SDN.
Figure 3: Summary of experimental results. At the top we have the price of reserving SDN over time and at the bottom we have the number of jobs outsourced using SDN over time.
Conclusion …

- Emergence of data-driven + data intensive applications
- Use of Cloud/data centres and edge nodes collectively
- Pipeline-based enactment a common theme
  - Various characteristics – buffer management and data coordination
  - Model development that can be integrated into a workflow environment
- Automating application adaptation
  - … as infrastructure changes
  - … as application characteristics change
Collaborators …

- **COSMOS**: Jeffrey Morgan, Peter Burnap, William Housley, Matthew Williams, Adam Edwards (Cardiff) + Rob Procter (Warwick)
- Ioan Petri (Cardiff, CS)
- Yacine Rezgui, Haijiang Li. Tom Beach (Cardiff ENGIN)
- Manish Parashar, Javier Diaz-Montes, Ivan Rodero, Mengsong Zou (Rutgers University, US)
- Rafael Tolosana, Ricardo Rodrigeez and Jose Banares (University of Zaragoza, Spain)
- Congduc Pham (University of Pau, France)