CloudView: Describe and Maintain Resource Views in Cloud

Dehui Zhou, Liang Zhong, Tianyu Wo and Junbin, Kang
Beihang University
Outline

• **The Cloud and motivation**
• Related works
• CloudView: a generalized way
• Evaluation results
• Conclusions
The Cloud Computing

• **Hot and emerging technology**
  – Governments, Big companies and Researchers.
  – Lower down cost and bring more startups.
  – More researches on distributed systems and virtualization.

• **New resource management paradigm**
  – On-demand resource provision and re-organization.
  – Isolation, Resource Teaming and Shaping, Better scaling.
  – Multi-tenants with different application requirements.
Our project

- Internet based Virtual Computing Infrastructure (iVIC)
  - Provide research platform for cloud computing.
  - Mainly on resource management in different layers of the cloud software stack, and their supporting technologies.
  - Tight cooperation with the internet industry.
  - Supported by China High Tech Research Program.
User Scenario

CloudView

Describe Resource View Needed

Provide Ready to Use Resource View Data

Statistic View
Aggregation View
Index View

Resource View Data

Raw data Collected

Physical
Virtual
Application

Raw Resource Performance/Configuration Data
Motivation

• **The abstraction: resource view**
  – Application specific dataset format over the resources.
  – Mostly includes index view, aggregation view and statistics view.

• **Can we provide user specific views over the resources?**
  – The resource information data collection, processing, and retrieving in a general manner.
  – Provide user with necessary resource data.
The problem

• It is hard to describe and maintain resource views in the cloud
  – Dynamic, massive, diverse characteristics of resources
  – Users from different layers have their own views on the cloud resources

• Key challenges
  – Flexibility. Calls for a resource and view description model that include mostly used resources.
  – Scalability. Calls for an architecture for processing view maintenance jobs.
What is CloudView?

- A distributed resource view data maintenance system mitigating user specific views and raw resource information data collected.
  - Provide users with a resource view description scheme.
  - Maintain user defined views automatically.
  - Distributed view jobs executing and data storage.
  - Write-Once, read-many and versioned resource view data.
Our contributions

• Provide a resource view description scheme based on general nested key-value pairs representation of resource information.
• Abstract the view maintenance flow and design its general structures, provide the algorithm from view description to job structure.
• Design and implement a distributed resource view maintenance system, includes view compilation, job execution environment, scheduler and so on.
Outline

• The Cloud and motivation

• Related works

• CloudView: a generalized way

• Evaluation results

• Conclusions
Related works

• **Monitoring and Discovering System**
  – [Ganglia], [Globus-MDS2]

• **Management System**
  – [VMware vCenter]

• **Information Overlay**
  – [S-Club, RCT, SDIRIS]

• **Parallel Data Processing**
  – [MapReduce, Bigtable, Agrawal2009]
Our difference

• Provide a general way to define user specific view over resources and their general maintenance mechanism.

• Based on a simple assumption that resource data can be packed in nested key-value pairs.

• Achieve adaptability and scalability to satisfy different users requirements.
Outline

• The Cloud and motivation
• Related works
• **CloudView: a generalized way**
• Evaluation results
• Conclusions
Data model

• 4 basic data models to store resource information data and view data during view maintenance.
  – Abstraction of data to be processed, inspired by java utils collection model

• Includes
  – Record. General nested key-value pairs.
  – Collection. A collection of Record or Multi-map
  – Ordered Collection. Sorted by expression.
  – Multi-map. A key with multiple values.

• Wrapped primitive data types
Resource Query Language

• Not Truly “Query”, More on view specification.
• Includes
  – Dataset model and simple expressions
  – Input/output operators
    • Load and Sink
  – Dataset manipulators
    • Filter, Order by
    • Statistic Functions
    • Group by, Equal-Join
  – Based on XML notation
View Maintenance Processing Unit

- **VMP = (Operator, Inputs, Output, Arguments)**
  - Inputs may be none or more than one datasets.
  - Output may be none or at most one dataset.
  - Arguments are
    - Expressions for *group by, order by* like datasets manipulation operators.
    - Parameters for *load, sink* input/output operators.

- **The basic unit to form view maintenance jobs.**
  - In a direct acyclic graph structure.
  - Executed in topology order within a execution engine.
Execution Engine

• Each VMP is executed in a thread
  – Decoupling VMP Tasks and execution thread.
  – Better concurrency with thread pool, easier context management.

• In-time expression evaluator
  – Compile and evaluate expression of the VMP with the context of data items in dataset.
  – Data manipulation operators always include expressions to group by, filter, sort, and join data sets.
Scheduler

• **Distribute view maintenance jobs to a set of worker nodes.**
  – User defined views with periodically update time span.
  – Currently use a centralized scheduler, all scheduling information are stored in MySQL database, employs the random scheduling policy.
  – Rescheduling the job if encounter failures.
  – Will be improved in future works.
The whole process

• **Steps**
  – User describe resource view with RQL and submit to the repository.
  – The scheduler schedule view maintenance jobs to a set of execution nodes.
  – Execution node compile the view description into executable DAG structure job and execute them in topological order.
  – Set the view data ready to read.
The architecture view of CloudView system
A case study

• **Resources**
  - Virtual Machine Container aka. Physical machine
    • (uuid, clusterid, load, mem, disk_rd, disk_wr, net_tx, net_rx)
  - Virtual machine
    • (uuid, vclusterid, host, load, mem, disk_rd, disk_wr, net_tx, net_rx)
  - Applications in VM
    • (uuid, host, connections, req_rate, req_hand_rate)
A case study cont’

• The index view of Physical machine for VM live migration target choosing
  – “an index view” of Physical machines based on the number of virtual machines it hosts, resource it has, and their performance.
  – Express in SQL-like language
    • “select uuid from vmcs where mem.free > 512MB and avg(load) < 0.5 and uuid in (select distinct host from vms where count(*) < 3 group by host) order by (1 - load) * mem.free”
Express in RQL

```xml
<view name="index_view" sched="600s">
  <ios>
    <load domain="ivic.org.cn" output="vmcs" args="{name:vmcs}"/>
    <load domain="ivic.org.cn" output="vms" args="{name:vms}"/>
    <sink domain="ivic.org.cn" input="ods"
      args="{name:idxs,timestamp:2010-11-10 15:00}"/>
  </ios>

  <operators>
    <operator type="filter" input="vmcs" output="vmcs_ftd"
      args="{eval:'mem.free']==512MB && avg(load)<0.5'}"/>
    <operator type="groupby" input="vms" args="{eval:'host'}"
      output="vmsg"/>
    <operator type="statfunc.count" input="vmsg" args="{eval:'*'}"
      output="vmsgc"/>
    <operator type="filter" input="vmsgc" output="vmsgcf"
      args="{eval:'count(*) <= 3'}"/>
    <operator type="equaljoin" input="vmsgcf,vmc_ftd" output="joint-ds"
      args="{eval:'uuid'}"/>
    <operator type="orderby" input="joint-ds" output="ods"
      args="{eval:'(1 - load) * mem.free'}"/>
  </operators>
</view>
```
The Complied Job Structure

• **Node**
  – The View Maintenance Processing Unit (VMP).
  – Includes: Operator, Arguments, Datasets.
  – Each VMP forms a Java Callable Task.

• **Edge**
  – Data dependency (Input dataset the pointing node) between the datasets.
Possible result data of the View

**vmcs**

<table>
<thead>
<tr>
<th>uuid</th>
<th>load</th>
<th>mem_free</th>
</tr>
</thead>
<tbody>
<tr>
<td>a10</td>
<td>0.72</td>
<td>1024MB</td>
</tr>
<tr>
<td>a11</td>
<td>0.45</td>
<td>782MB</td>
</tr>
<tr>
<td>a12</td>
<td>0.25</td>
<td>1234MB</td>
</tr>
</tbody>
</table>

**vms**

<table>
<thead>
<tr>
<th>uuid</th>
<th>host</th>
<th>load</th>
</tr>
</thead>
<tbody>
<tr>
<td>b10</td>
<td>a10</td>
<td>0.68</td>
</tr>
<tr>
<td>b11</td>
<td>a11</td>
<td>0.52</td>
</tr>
<tr>
<td>b12</td>
<td>a10</td>
<td>0.34</td>
</tr>
<tr>
<td>b13</td>
<td>a10</td>
<td>0.87</td>
</tr>
<tr>
<td>b14</td>
<td>a11</td>
<td>0.22</td>
</tr>
<tr>
<td>b15</td>
<td>a12</td>
<td>0.73</td>
</tr>
<tr>
<td>b16</td>
<td>a12</td>
<td>0.15</td>
</tr>
</tbody>
</table>

**result index view data**

<table>
<thead>
<tr>
<th>uuid</th>
<th>(1-load)*mem_free</th>
</tr>
</thead>
<tbody>
<tr>
<td>a12</td>
<td>925.5</td>
</tr>
<tr>
<td>a11</td>
<td>430.1</td>
</tr>
</tbody>
</table>
Outline

• The Cloud and motivation
• Related works
• CloudView: a generalized way
• Evaluation results
• Conclusions
Experiments Setup

• Hardware & Software
  – With 16 commodity computers, 1 as the meta-data server, 1 as the scheduler, 4 as simulator, 10 for data storage and view maintenance job execution. All connected with 1GB Ethernet.
  – Java 1.6, Cassandra 0.6.2 and Debian Linux 5.0

• Data
  – Simulate $10^5 \sim 12 \times 10^5$ resources, each generate 256 bytes resource information data in every 10 seconds. Collect about 50 GB resource raw data totally.
  – Maintain 3 type of views: Index, Aggregate, and Statistics.
View maintenance cost

- The time cost for maintain different type of views
  - resources are in different domains, each job with input data in 10~50MB
  - view definition are limited in domains
Query latency

- Time cost to query over different type of views with concurrent clients
  - View data are spread over different nodes
  - Clients randomly select worker nodes to issue queries.

![Query latency over materialized view](image)

**Y-axis (T in ms)**
- 5.5
- 6
- 6.5
- 7
- 7.5
- 8
- 8.5
- 9
- 9.5
- 10

**X-axis (Number of concurrent clients)**
- 10
- 15
- 20
- 25
- 30
- 35
- 40
- 45

Legend:
- RangeQuery
- AggreQuery
- StatsQuery
Outline

• The Cloud and motivation
• Related works
• CloudView: a generalized way
• Evaluation results

• Conclusions
Conclusion and future works

• Conclusion
  – On demand control of resources calls for on demand view over resources in cloud.
  – CloudView provide a general way to describe and maintain resource views.
  – It has benefits as well as limitations.

• Future works
  – Improve scheduling policy and determine the view dependencies while doing view updating.
  – Provide more efficient data storage methods.
  – More detailed performance evaluation model.
Q&A  Thanks very much

Please pay attention to www.ivic.org.cn