Improving Utilization through Dynamic VM Resource Allocation in Hybrid Cloud Environment

Yuda Wang, Renyu Yang, Tianyu Wo, Wenbo Jiang, Chunming Hu

School of Computing
Beihang University
yangry@act.buaa.edu.cn

In IEEE ICPADS, Hsinchu, Taiwan, 2014
Big Data & Cloud Computing

• **Big Data**
  – Data Volume
    • 2.5 Exabyte of data is generated everyday and the speed of data generation doubles every 40 months [-- Harvard Business Review 2012]
  – Data Analytics
    • Parallel tasks are executed in distributed system (Hadoop, Mesos, Yarn, Fuxi etc.)

• **Cloud Computing**
  – Pay-as-you-go service
  – Infrastructure as a service (IaaS)
    • Rent VMs leveraging virtualization techniques
    • Capsulate the application/services in each VM
  – Benefits:
    • resource sharing among users/workloads
    • Cut down the maintenance cost of the infrastructure
Background

• Hybrid Cloud Environment
  – Combinations of heterogeneous workloads
    • Batch processing job/Compute intensive job
      – E.g., MapReduce, Spark, Storm job etc
    • Multi-tenant and VM based services
      – Transactional or long-running services such as web service, virtual desktop, open storage services etc.
  • Offline and Online services
    – Application performance v.s. cluster resource utilization
      • low utilization is a very normal phenomenon
      • Evidence
Challenges:
How to consolidate the heterogeneous tasks?
How to improve utilization with as little negative impact as possible?
States of the Art and Issues

• Separate frameworks on the shared cluster
  – lack of overall cluster-level resource viewpoints
  – result in substantial overhead in data exchanges and component communications

• Current frameworks (Yarn, Mesos, Fuxi) do not support VM in their resource management system.
  – VM is unlike the typical task:
    • Infrequently created, terminated, restarted.
    • Huge cost for VM migration or reboot.
    • High availability is required.

• Especially, VM in the cloud:
  – Resource over-provision with pre-defined size/configuration
  – Guarantee the SLA
  – Impact the whole utilization
Proposed Scenario

• An “interlayer” between applications and compute nodes

A protocol/consolidation mechanism to support both online VM service and typical offline compute-intensive tasks

Elastic resource allocation mechanism to timely adjust the over-provisioned resource amount.
Basic Concept and Workflow

• Inherit concepts from Yarn

• Core philosophy:
  – De-coupling JobTracker into:
    • Resource Management + Scheduling / Monitoring
  – Following a request-based and active approach

• Roles:
  – **Resource Manager:**
    • central resource negotiator
    • responses to application’s resource requests
  – **App Master**
    • Controls the execution logic and generates the corresponding resource request.
  – **Node manager**
    • special system worker daemon to launch, monitor or kill the compute task within an execution container.
Basic Concept and Workflow

- User submits application request
- Resource Manager allocates a container for Application Master
- AM asks the RM for resource with specific request (e.g., for VM, MapReduce respectively)
- RM would try to satisfy the resource request as possible and assign the resource allocation according to some scheduling policies.
- Tasks execute on the requested nodes
Architecture

- Handle waiting resource requests
- Match between requests and available resources

Service-mode app master to manage the VM’s life-cycle: (VM placement decision), resource request, VM starting and migration control.

- Isolate the resources for the task
- execute dynamic resource re-allocation
VM Service App Master

- VM App Master is service-mode, and will be continuously running since established.
- VM placement Protocol
  - FullInfoRequest
    - asks RM for global resource amount (total resource and available resource in each node)
  - VM Placement
    - Based on the resource view, VM placement decision is conducted. (currently load balance algorithm)
    - generate a locality preferences.
  - For Virtual Cluster?
    - A bundle of customized VMs in the vcluster
    - Optimizations in a batch mode
      - Cost in the object locking, node communication etc.
- Resource Request
  - Ask RM for executable container resources.
  - High priority in the queue in RM.
- VM(s) Launching
  - According to the assigned resources, make NM to launch the containers.

```
Full Info {
  Machine List = \{M_1, M_2 \ldots M_m\}
  Available Resources = \{\ldots\}
  Used Resources = \{\ldots\}
}

VM Placement Algorithm

Resource Request:
{
  Vm_1: \{2G mem, 1core CPU, M_1\}
  Vm_2: \{2G mem, 1core CPU, M_5\}
  \ldots
  Vm_n: \{2G mem, 2cores CPU, M_{12}\}
  \ldots
}
```
Resource Dynamic Resizing

• **Objectives:**
  – resize (usually shrink) the amount of binding resource (usually over-estimated) of a running container according to the resource fluctuation.

• **Core Idea:**
  – Resource Monitoring
    • A daemon monitors the real-time resource usage via Linux *proc*.
  – Assigned Resource Resizing
    • In NM: calculates the new assigned amount via sliding window algorithm.
    • In AM: submits a new resource request (RR) *(increase/decrease)* in an incremental way to the RM.
  – Actions Taking
    • Decrease RR or increase RR whilst the node has sufficient resource?
      – Go ahead, changing to the new amount.
    • Increase RR when no sufficient resource could be assigned by RM?
      – VM migration…

Sliding window size is predefined as the max *retryTime*. If all the values in the sliding window surpass a threshold, a new assigned amount is set.
VM Migration Protocol

• VM live-Migration offers:
  – To service provider: High Availability and Fault tolerance
  – To User: performance QoS

Proposed Protocol:
• Daemon to VM Master: sends Migration Request (step 1)
• VM Master to RM: sends Resource Request in order to launch a new container in a particular node. (step 2-3)

After the resource is assigned:
• VM Master to Daemon: move the VM to the new container (step 4-5)
• Daemon to VM Master: sends Migration Finished signal, VM Master releases the previously assigned resource. (step 6-7)
Evaluation Setup

• **Cluster:**
  
  5x dell servers with Intel(R) Core(TM) i7 860 2.80GHz CPU, 4GB memory and 320GB disk

• **Benchmarks**
  
  – TPC-C: simulate an on-line e-commerce for VM
  – Wikipedia
  – WordCount: count each word’s frequency within key-value structure text files for MapReduce

• **Metrics**
  
  – Request Response Time/ Interactive Latency (TPC-C, wiki)
  – Job Complete Time (wordcount)
Proposed Re-allocation Effects

Steps:
In this experiment, different pair-VM combinations are executed for a period of time, and each VM runs a specific benchmark.

Results:
- Pre-defined static memory assignment is roughly twice the actual used amount, indicating great resource reservation and low utilization.
- The allocated memory with our system is only 10% margin on average, and can improve the resource utilization especially in idle workloads environment.

The figure shows the normalized assigned memory amount with, without dynamic allocation mechanism and the actual memory usage.
Impact on Workload Performance

Five concurrent wordcount jobs are kept for a period of time while VMs with the benchmark under different configurations are submitted.

Considering: 1) the max number that the cluster can hold 2) Workload Performance

- (a) the number of TPC-C VM
- (b) the performance of TPC-C VM
- (c) the average JCT of wordcount jobs
- (d) the number of Wikipedia VM
- (e) the performance of Wikipedia VM
- (f) the average JCT of wordcount jobs

**the max number** in pre-defined allocation is much less than our dynamic allocation mechanism.

A slight overhead (within 1%) in comparison with pre-defined allocation strategy.

The extended JCT of wordcount workload is no more than 1%.
Performance Comparisons Among Different Approaches

Steps:
1) Manage MapReduce and VM separately in the cluster
2) Use a hybrid framework with dynamic re-allocation
3) Use a hybrid framework with pre-defined allocation

Considering: 1) the max number that the cluster can hold 2) Workload Performance

Results:
- Additional 8 VMs could be launched in dynamic allocation although the response latency and the corresponding JCT is slightly increased.
- Separate managed cluster could hold more workloads but suffers from great performance degradation.
Performance Benefits from VM migration

Steps:
- generate an overloaded scenario by load injection.
- increase the inner load (e.g., grow the user thread number) of a specific VM and measure the fluctuation of the interaction latency.

Results:
- The VM is supposed to be migrated once it cannot obtain sufficient resources.
- The latency of Wikipedia and TPC-C benchmark are reduced through VM migration, thereby significantly maintaining the user’s QoS.

<table>
<thead>
<tr>
<th>Workload</th>
<th>Before migration</th>
<th>After migration</th>
<th>reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wikipedia</td>
<td>6467078</td>
<td>4368376</td>
<td>32.45%</td>
</tr>
<tr>
<td>TPC-C</td>
<td>86094.29</td>
<td>69876.83</td>
<td>18.84%</td>
</tr>
</tbody>
</table>
Conclusion

• A unified resource management framework for both online services (like VM) and offline data processing service (like MapReduce).

• An effective resource elastic adjustment mechanism according to the fluctuation of resource request in the VM to improve the holistic cluster utilization.

• A VM migration based approach that enables the high availability and performance of VM application without violating the SLA of other tasks.
Thank you for your attention!
Appendix
Memory usage (GB) vs. time

Predefined memory configuration
Sliding Window

Sliding window size is predefined as the max `retryTime`. If all the values in the sliding window surpass a threshold -> new assigned amount.