DRS: Advanced Concepts, Best Practices and Future Directions

Aashish Parikh, VMware, Inc.
Ajay Gulati, VMware, Inc.
Disclaimer

- This session may contain product features that are currently under development.
- This session/overview of the new technology represents no commitment from VMware to deliver these features in any generally available product.
- Features are subject to change, and must not be included in contracts, purchase orders, or sales agreements of any kind.
- Technical feasibility and market demand will affect final delivery.
- Pricing and packaging for any new technologies or features discussed or presented have not been determined.
DRS — Overview

- Ease of Management
- Initial Placement
- Runtime CPU/Memory Load-balancing
- VM-to-VM and VM-to-host Affinity and Anti-Affinity
- Host Maintenance Mode
- Add Host

Always respect resource controls!
Talk Outline

- Resource settings and Pools
  - Specifying behavior under contention

- VM Happiness
  - The key to application performance

- Load-balancing and handling multiple metrics
  - The other side of making VMs happy

- Future Directions
  - A sneak peek into DRS labs!
Resource Settings and Pools

Specifying behavior under contention
Key Concepts

- Each VM has
  - Configured capacity: 10 GHz, 24 GB RAM
  - Demand for resource: 2 GHz, 4 GB RAM

- Each host has
  - CPU & Memory capacity: 40 GHz, 96 GB RAM

Q: How many VMs can fit on the host?
A: A lot, if you don’t care about performance

Too safe: 4 (based on configured size)
Ideal: 20 (based on CPU demand)
32 (based on memory demand)
Why do we need Resource controls?

- You don’t know the demand values
- You may want to strictly isolate some VMs from others
- Sometimes: Sum of VM Demand > Cluster capacity
- Resource controls:
  - Determine who gets what under contention
  - Enable high consolidation and allow safe over-commitment!

Q: Should you worry if Sum of VM Demand > host capacity?

Answer: No, DRS handles that case!
You only need to make sure that cluster has enough capacity
Resource Controls

- Reservation: Guaranteed allocation
- Limit: Guaranteed upper bound
- Shares: Allocation in between
- Resource pools: allocation and isolation of groups of VMs

Actual allocation depends on the reservation, limit, shares and demand
Resource Pools

- A powerful abstraction for aggregation and isolation of resources
- Customer creates a resource pool and sets business requirements
- Sharing within a pool, isolation across pools
**Controls: Reservation (MHz or MB)**

- Minimum MHz or MB guaranteed
- Parent reservation $\geq$ sum of children reservations
- DRS will distribute the R value hierarchically using *current demand*

---

**Resource Pool with CPU Reservations**

![Diagram](image)
Controls: Limit (MHz or MB)

- Maximum MHz or MB allowed
- Parent limit $\leq$ sum of child limits
- Child limits set based on parent limit and current VM demand

Resource Pool with CPU Limits

- $L_{\text{Root}} = 5000$ MHz
- $L_{\text{Sales}} = U$
- $L_{\text{Marketing}} = 1000$ MHz
- 1500 MHz
- U
- U
- U
- U
- U
- U

U: Unlimited
Controls: Shares (No units)

- Relative priority **between siblings** in the tree
- Divide parent share in proportion of children’s shares

Resource Pool with Share settings
Virtual View of the World

- DRS resource pool as specified by user
- In practice, life isn’t so simple!
- VMs get mapped to different hosts and move between them

In practice, life isn’t so simple! VMs get mapped to different hosts and move between them.

VM level \(<r,l,s>\) settings

Host A
5000 MHz

Host B
4000 MHz
How to set these controls: 4 simple rules

1. Think VM, act RPs
2. Use shares for prioritization
3. Use a tiered reservations model (e.g. more for critical apps)
4. Use limits to control usage (e.g. pay per use model)
Use Case 1: Similar Kind of Applications

- Apply rule 1: Think VM act RP
- Equal shares, Limits (if needed)

R = 10 GHz
L = 30 GHz

R (GHz): 2 2 2 2 2 2
L (GHz): 6 6 6 6 6 6
Configured: 8 8 8 8 8 8

No Distribution Possible 😞
Use Case 1: Similar Kind of Applications

- Apply rule 1: Think VM act RP
- Equal shares, Limits (if needed)

\[
\begin{align*}
R &= 10 \text{ GHz} \\
L &= 30 \text{ GHz}
\end{align*}
\]

\[
\begin{array}{cccccc}
R (GHz): & 0 & 0 & 0 & 0 & 0 \\
L (GHz): & 8 & 8 & 8 & 8 & 8 \\
Configured: & 8 & 8 & 8 & 8 & 8
\end{array}
\]

DRS can Re-Distribute based on demand

Sum of VM reservations will be 10 GHz

Sum of VM limits will be 30 GHz

- For memory, RP reservation includes VM overhead memory
Use Case 1: Similar Kind of Applications

- Apply rule 1: Think VM act RP
- Equal shares, Limits (if needed)

R = 10 GHz
L = 30 GHz

R (GHz): 2 2 2 2 2 2
L (GHz): 6 6 6 6 6 6
Configured: 8 8 8 8 8 8

No Distribution Possible 😞
Use Case 2: Multiple Application Tiers

- **Goal:** Top tiers should suffer less than bottom tiers

- **Think VM first:**
  - Tier 1 apps: VM reservation = 25% (rule 3), shares = High (rule 2)
  - Tier 2 apps: VM reservation = 10% (rule 3), shares = Low (rule 2)
Use Case 2: Multiple Application Tiers

- **Apply rule 1:**
  - Tier 1 RP: \( R = \sum \text{VM reservations}, \text{Shares} = \sum \text{VM shares} \)
  - Tier 2 RP: \( R = \sum \text{VM reservations}, \text{Shares} = \sum \text{VM shares} \)
  - Set VM level reservations = 0, shares = Normal for all VMs

\[
\begin{align*}
R &= 6 \text{ GHz} \\
\text{Shares} &= 2000 \times 3 \\
&= 6000
\end{align*}
\]

\[
\begin{align*}
R &= 1.6 \text{ GHz} \\
\text{Shares} &= 500 \times 2 \\
&= 1000
\end{align*}
\]

- **Values:**
  - \( R \) (GHz):
    - 0
    - 0
    - 0
    - 0
    - 0
    - 0
  - \( L \) (GHz):
    - U
    - U
    - U
    - U
    - U
    - U
  - Shares:
    - 1000
    - 1000
    - 1000
    - 1000
    - 1000
    - 1000
  - Configured:
    - 8
    - 8
    - 8
    - 8
    - 8
    - 8
Use Case 3: Strict priority between Tiers

- **Goal:** Top tiers should not suffer at all due to bottom tiers.

- **Think VM first:**
  - VM reservation = 50, 10 and 5% for three tiers (rule 3)
  - VM shares = High, High/10, High/100 (rule 2)

- **Apply rule 1: Resource pool settings = \(\sum\) VM settings**

<table>
<thead>
<tr>
<th>R (GHz)</th>
<th>L (GHz)</th>
<th>Shares</th>
<th>Configured</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>U</td>
<td>1000</td>
<td>8</td>
</tr>
<tr>
<td>0</td>
<td>U</td>
<td>1000</td>
<td>8</td>
</tr>
<tr>
<td>0</td>
<td>U</td>
<td>1000</td>
<td>8</td>
</tr>
<tr>
<td>0</td>
<td>U</td>
<td>1000</td>
<td>8</td>
</tr>
<tr>
<td>0</td>
<td>U</td>
<td>1000</td>
<td>8</td>
</tr>
</tbody>
</table>

Where:
- **R** = 12 GHz, **Shares** = 2000*3 = 6000
- **R** = 4 GHz, **Shares** = 200*2 = 400
- **R** = 0.8 GHz, **Shares** = 20*2 = 40
Key Takeaway

- Resource controls allow high consolidation and safe over-commit

- Follow four simple rules to determine resource controls
  1. Think VM, act RPs
  2. Use shares for prioritization
  3. Use a tiered reservations model (e.g. more for critical apps)
  4. Use limits to control usage (e.g. pay per use model)

- If you need some other behavior, let us know

- References for more technical details:
  VMWare Distributed Resource Management: Design, Implementation, and Lessons Learned (VMware Technical Journal)
VM Happiness
The key to application performance
The Giant Host Abstraction

- Treat the cluster as one giant host
  - Capacity of this giant “host” = capacity of the cluster

6 hosts
CPU = 10 GHz
Memory = 64 GB

1 “giant host”
CPU = 60 GHz
Memory = 384 GB
Why is that Hard?

Main issue: **fragmentation of resource across hosts**

Primary goal: Keep VMs happy by meeting their resource demands!

1 “giant host”

- CPU = 60 GHz
- Memory = 384 GB

VM demand < 60 GHz

All VMs running happily
Why Meet VM Demand as the Primary Goal?

- VM demand satisfied <-> VM or application happiness
- Why is this not met by default?
  - Host level overload: VM demands may not be met on a host
- Need to migrate VMs to hosts with free capacity
- Note: demand ⇔ current utilization
How are Demands computed?

- Demand indicates what a VM could consume given more resources
- Demand can be higher than utilization
  - Think ready time
- CPU demand is a function of many CPU stats

\[
CPU_{demand} = CPU_{used} + CPU_{ready} \left( \frac{CPU_{run}}{CPU_{run} + CPU_{sleep}} \right)
\]

- Memory demand is computed by tracking pages in guest address space and the percentage touched in a given interval
Why not Load-balance the DRS Cluster as the Primary Goal?

- **Load-balancing is not free**
  - In case of low VM demand, load-balancing migrations have cost but no benefit

- **Load-balancing is a criteria used to meet VM demands**

Consider these 4 hosts with almost-*idle* VMs

**Q:** Should we balance VMs across hosts?

**Note:** all VMs are getting what they need
Important Metrics and UI Controls

- **CPU and memory entitlement**
  - VM deserved value based on demand, resource settings and capacity

Load is not balanced but all VMs are happy!
Key Takeaway

- **Goals of DRS are:**
  - To provide the giant host abstraction
  - To meet VM demand and keep applications at high performance
  - To use load balancing as a secondary metric while making VMs happy

- **If you care about load balancing as a key metric**
  - It can be achieved
  - Stay awake!
Load-balancing and handling multiple metrics

The other side of making VMs happy
The balls and bins problem

- **Problem: assign** \( n \) **balls to** \( m \) **bins**
  - Balls could have different sizes
  - Bins could have different sizes

- **Key Challenges**
  - Dynamic numbers and sizes of balls/bins
  - Constraints on co-location, placement and others

- **Now, what is a fair distribution of balls among bins?**

- **Welcome to DRS load-balancing**
  - VM resource entitlements are the ‘balls’
  - Host resource capacities are the ‘bins’
  - Dynamic load, dynamic capacity
  - Various constraints: DRS-Disabled-On-VM, VM-to-Host affinity, …
The balls and bins problem

- **Problem:** assign $n$ balls to $m$ bins
  - Balls could have different sizes
  - Bins could have different sizes

- **Key Challenges**
  - Dynamic numbers and sizes of balls/bins
  - Constraints on co-location, placement and others

- **Now, what is a fair distribution of balls among bins?**

- **Welcome to DRS load-balancing**
  - VM resource entitlements are the ‘balls’
  - Host resource capacities are the ‘bins’
  - Dynamic load, dynamic capacity
  - Various constraints: DRS-Disabled-On-VM, VM-to-Host affinity, …
Goals of DRS Load-balancing

- Fairly distribute VM demand among hosts in a DRS cluster
- Enforce constraints
  - Recommend mandatory moves
- Recommend moves that significantly improve imbalance
- Recommend moves with long-term benefit
How is Imbalance Measured?

- Imbalance metric is a cluster-level metric

- For each host, we have
  \[ \text{normalized entitlement} = \frac{\sum VM \text{ entitlements}}{host \text{ capacity}} \]

- Imbalance metric is the standard deviation of these normalized entitlements

- Imbalance metric = 0 \rightarrow perfect balance

- How much imbalance can you tolerate?
The Myth of Target Balance

- UI slider tells us what star threshold is acceptable
  
  Migration Threshold
  
  Conservative                       Aggressive
  
  Apply priority 1, priority 2, priority 3, priority 4, and priority 5 recommendations. vCenter Server will apply recommendations that promise even a slight improvement to the cluster's load balance.

- Implicit target number for cluster imbalance metric
  - n-star threshold ➔ tolerance of imbalance up to n * 10% in a 2-node cluster

- Constraints can make it hard to meet target balance

- Meeting aggressive target balance may require many migrations
  - DRS will recommend them only if they also make VMs happier

If all VMs are happy, a little imbalance is not really a bad thing!
Evaluating candidates moves to generate recommendations

- Consider migrations from over-loaded to under-loaded hosts
  - $\Delta = \text{Imbalance Metric}_{\text{before move}} - \text{Imbalance Metric}_{\text{after move}}$
  - ‘$\Delta$’ is also called goodness

- Constraints and resource specifications are respected
  - VMs from Host$_2$ are anti-affine with Host$_1$

- Prerequisite/dependent migrations also considered

- List of good moves is filtered further

*Only the best of these moves are recommended*
MinGoodness Filtering: the search for significant moves

- Recall that for each candidate move, we compute
  - goodness = Imbalance Metric\text{before move} – Imbalance Metric\text{after move}
- Moves must meet threshold of goodness to be recommended
- All VMs are too small to make a significant difference

- Candidates that cannot meet this criteria are MinGoodness filtered
CostBenefit Filtering: the search for enduring quality

- **Benefit:** Higher resource availability
- **Cost:**
  - Migration cost: vMotion CPU & memory cost, VM slowdown
  - Risk cost: Benefit may not be sustained due to load variation

Candidates that cannot meet this criteria are **CostBenefit filtered**
Handling Severe Imbalance

- **Causes for severe imbalance**
  - Target too impractical, too many constraints
  - Filters too aggressive for certain inventories
  - Newly powered-on hosts or updated hosts – interplay with VUM

- **DRS automatically detects and addresses severe imbalance**
  - Filters automatically relaxed or dropped; reinstated when the situation is fixed

- **NEW** Entirely revamped in vSphere 5.1
  - New advanced config options
  - Also available in vSphere 4.1 U3 and vSphere 5.0 U2
Handling Severe Imbalance: Advanced Config Options

- **DRS relaxes filters to fix severe imbalance by default**
  - SevereImbalanceRelaxMinGoodness = 1
  - SevereImbalanceRelaxCostBenefit = 1
  - FixSevereImbalanceOnly = 1

- **You can modify default behavior, handle with care!**
  - FixSevereImbalanceOnly = 0
  - SevereImbalanceDropCostBenefit = 1
Handling Severe Imbalance: Older Releases and Updates

- Clusters severely imbalanced but unable to upgrade?

- Setting these options temporarily can provide immediate relief
  - MinGoodness = 0; allows any migration with goodness > 0
  - CostBenefit = 0; considers expensive/ephemeral moves too

- Setting these options can cause a large number of migrations!
  - Don’t forget to set them back to 1 when balance is restored!
  - DRS Load-balancing runs better with filters
Handling Multiple Metrics

- Optimizing for multiple metrics is hard
  - DRS uses smart heuristics, tries not to *hurt* any metric

  \[
  \text{score}_{\text{move}} = w_{\text{CPU}} \times \text{score}_{\text{CPU}} + w_{\text{MEM}} \times \text{score}_{\text{MEM}}
  \]
  - Weights are computed dynamically based on resource utilization

- Customers have requested even more metrics!

- Extra metric: **CPU ready time**
  - Handling severe imbalance reduces ready time in many cases
  - Considering doing even more here, talk to us if this is important to you
Extra metric: “Eggs-in-a-basket” – Restrict #VMs per Host

- Recall this example

- This cluster is perfectly balanced w.r.t. CPU, Memory
- However, spreading VMs around is important to some of you!
- *NEW* Advanced config option in vSphere 5.1
  - LimitVMsPerESXHost = 6
- DRS will not admit or migrate more than 6 VMs to any Host
- This may impact VM happiness, load-balancing
Key Takeaway

- DRS recommends moves that **reduce cluster imbalance**

- **Constraints** can impact achievable imbalance metric

- Filters help improve the quality of moves and reduce noise
  - Moves must satisfy MinGoodness, CostBenefit analyses to be recommended

- DRS automatically detects and addresses severe imbalance (**NEW**)

- DRS handles many metrics effectively
Future directions
A sneak-peek into DRS Labs!
Turbo mode Load-balancing

- Most aggressive load-balancing option

- One-time, manual-mode load-balancing call – no holds barred

- Pros
  - Reach lowest possible imbalance metric value possible
  - Maximum exploration of solution space
  - No MinGoodness or CostBenefit filtering

- Cons
  - No MinGoodness or CostBenefit filtering!
  - May recommend a large number of migrations
Eggs-in-a-basket: AutoTune

- Recall the eggs-in-a-basket metric
  - \( \text{LimitVMsPerESXHost} = 6 \)

- If 12 new VMs are powered on, Host\(_4\) cannot accept any!
  - Need to manually change the LimitVMsPerESXHost value

- Instead, automatically adjust value = mean + buffer\% \times mean
  - Set new option for buffer\% just once: LimitVMsPerESXHostPercent = 50

- Number of VMs = 20 + 12 (new); mean = 32 \( \div \) 4 = 8; buffer = 50\%

- New value is automatically set to 8 + 50\% \times 8 = 12

- Host\(_4\) is now able to accept 6 more VMs!
What-if DRS: Building castles in a DRS sandbox

- Safely operate DRS on hypothetical inventories and configurations

- Real-world queries and DRS metrics
  - “If DPM was enabled on the cluster, what would VM happiness look like?”
  - “Which migrations will be triggered by breaking the vm₆ – vm₈ affinity rule?”
  - “If host₂ is upgraded, how will any newly unlocked features work with DRS?”

- Complex queries
  - "If I added a clone of host₄, remove host₃ and add 10 clones on vm₂₄ to my cluster, will any of my constraints be violated? What will my new inventory look like after load-balancing?"

- Use for trouble-shooting and capacity planning!
In Summary

- Use resource pool settings instead of VM settings when possible
  - Ease of management
  - Safe over-commitment

- Resource control recipes can be used to handle various use cases

- VM Happiness is the key goal of DRS

- Load balancing keeps VMs happy and resource utilization fair
  - As long as all VMs are happy, a little imbalance is not always a bad thing!

- Tons of cool DRS stuff in the pipeline, feedback welcome!
FILL OUT A SURVEY

EVERY COMPLETE SURVEY IS ENTERED INTO DRAWING FOR A $25 VMWARE COMPANY STORE GIFT CERTIFICATE
DRS: Advanced Concepts, Best Practices and Future Directions

Ajay Gulati, VMware, Inc.
Aashish Parikh, VMware, Inc.