Virtual Machine Migration Plan Generation Through A* Search

Eli M. Dow <emdow@us.ibm.com>
IBM Research

Jeanna N. Matthews <jnm@clarkson.edu>
Clarkson University

Abstract

Modern virtual machine (VM) management software includes components that enable consolidation of VMs for power savings or load-balancing for performance. While the existing literature provides various methods for computing a better load-balanced, or consolidated goal state, it fails to adequately suggest the best path from the system's current state to the desired goal allocation. This paper discusses an approach to efficient path finding in VM placement plan generation for cloud computing environments. Initial results indicate our prototype solution is viable for managing hundreds of VMs through the application of A* search.

1. Motivation

In this paper, we present algorithms for automatic VM placement and relocation as achieved through non-disruptive live guest relocation. There are several existing contributions to the literature in these areas, but an often-overlooked issue is just how does one get to an ideal goal assignment of VMs to hosts from some other starting state? This is our focus.

Researchers have frequently suggested novel techniques for determining how a collection of VMs should be matched to a set of physical hosts to satisfy a variety of sometimes competing goals including load-balancing or consolidation for power savings. Since our approach for migration plan generation does not depend on any particular implementation of placement determination logic, the technique should be adaptable to a variety of possible placement engines.

Regardless of the goal state chosen, systems practitioners have to actually execute a migration plan that will take the system from the current state to the prescribed ideal state. They need to execute the series of individual VM live migrations in a specific order and in a timely manner before they can derive any real value from research that suggests a goal state. Here we use the term convergence to mean the time required to go from some initial VM to host allocation to some other idealized allocation (consolidated, load-balanced, *etc*...). Convergence must be reached quickly while resource consumption during both plan generation and migration needs to be efficient enough to support frequent plan generation. Care must be taken that migration plan execution remains computationally practical even for moderate scale clouds.

Commercial efforts described in academic literature by HP and VMWare are prime examples of VM management software leveraging live guest migration [1][2]. Recent commercial products like IBM® Distributed Resource Scheduler®[3] show continued commercial interest in creating viable products from this line of research. However, commercial solutions described thus far have some shortcomings with respect to large scale optimization of VM placement. For instance, the paper by Heyser *et al* [1] use a clever algorithm tailored to minimal perturbation to the present solution in order to avoid massive reshuffling of VMs [1]. Their approach avoids the complexity of optimal path building at the expense of a potentially better global solution. It is worth noting that some research into similar problems focus on the initial admission control decision of whether to allow a new VM to be deployed for the first time into a cloud resource pool [4]. As in [1], their goal is improvement, not achievement of globally optimal placement which makes the problem more tractable.

Ad-hoc approaches to consolidation and live migration planning run into resource bottlenecks whereby some VMs are not allowed to co-reside on the same physical hardware for reasons including but not limited to: high-availability concerns, diverse networking requirements, specific isolation requirements, legal mandates governing user data co-location (which extend to the VMs hosting that data), and/or various instances where security risks are deemed too high for co-location to be allowed. Furthermore, many academics mistakenly believe that enterprise cloud data centers avoid live migration due to complexities and simply prefer to shutdown/restart VMs that need to be moved, deferring the issue of rebalancing to a degenerate *admission control* problem.

2. Reconciling An Optimal VM Placement Strategy With Current VM Allocations

Regardless of which research technique is used to obtain an idealized mapping of VMs to hosts, and regardless of which placement policies were being enforced (consolidation, load-balancing) little has been

published about techniques for generating migration plans to move from a given state to an optimized state.

A naïve migration plan generation solution considers only the present mapping of VMs to hosts along with the idealized mapping of VMs to hosts. One can easily conceive of a loop which identifies the original host for each VM and the idealized destination host for that same VM. Sometimes, however, it is simply not possible to begin executing migrations according to the naive scheme. Constraints such as hypervisor low memory conditions, VM:VM anti-colocation constraints, and VM:Host anti-colocation constraints quickly foil such schemes in anything beyond a trivial cloud computing environment.[1]

For instance, at no point in a migration plan should a host be over constrained for resources such as memory, CPU, disk I/O or networking I/O. System administrators might be willing to overcommit each of those resources with different tolerances for resource over-constraint depending on the particular resource in question. Furthermore, the disk and networking I/O categories can be partitioned into read and write / read and transmit activities independently. From this split, we control the level of over constraint on each separately while also maintaining a cumulative I/O over-constraint. As you can imagine, there is a variety of means to inadvertently over-constrain a host when moving VMs around dynamically.

What is needed is a means for finding an optimal path from the current (start) state to the optimized (goal) state, according to a set of valid transition rules (these rules encode the various user defined over-constraint, and anti-colocation policies discussed earlier) that must be enforced at each interim step.

This scenario lends itself to the application of the well-studied A* search algorithm [5] which is itself an extension of the well known Dijkstra's Algorithm with the addition of heuristics. A* has been applied in a wide selection of problem domains from video game non-player character movement over a map [6] to autonomous robot path finding in the real world [7].

The A* algorithm effectively keeps a list of potential states to be considered (the OPEN list), along with states that have been known to be encountered previously along with a measure of their fitness. The OPEN list is a data structure used in the A* algorithm that describes viable partial solutions to the overall problem. At each stage of the A* algorithm the OPEN list represents which partial paths need to be expanded one level deeper, and thereby lists the partial solutions which get to live on until the next iteration of the algorithm. The fitness values are composed of the optimally computed cost of entering that state in addition to the heuristic estimation of distance from that state to the goal state. Since each iteration of the A* algorithm expands existing valid states into subsequent "next" states, there is a branch out effect that is governed by the number of valid states that can be generated from any current given state. For example, if A* is applied to simple 2D chessboard style environments in which a valid move is one position in any direction on the board for any interior, non-edge, position yielding a maximum of 8 possible immediately reachable transition states (North, South, East, West and the four diagonal positions assuming diagonal moves are valid state transitions). In this case the count of the number of valid state transitions yields a branching factor of 8. The nominal A* approach has been shown to be effective in path finding through environments of this kind, and has found success in a variety of computer games [6].

The application of A* to the VM placement/ convergence problem is different in a number of important ways. In this problem domain, we consider a state to be some mapping of VMs to hosts. A valid state transition is the application of any valid VM migration to some other host.

Whenever A* is applied to a new problem domain, a another critical step is the selection of functions $H(x)$ and $G(x)$ that guide the implementation. Intuitively, $H(x)$ reflects the cost of a transition from one state to another and $G(x)$ reflects an upper bound on the estimated cost of reaching the goal state from some new state. Each candidate path from an origin/initial VM layout to an idealized goal state is evaluated according to a function $F(x) = H(x) + G(x)$.

2.1. Validation of A*-derived Migration Plan Generation

To validate the output of our A*-derived migration plan, we simulated the effect of the actions prescribed on the starting state sequentially, validating that no constraints are violated at each iteration. This helped us

[1] To enforce co-location and anti-colocation policy, we opted to leverage the concept of virtual machine contracts [8]. Our implementation of co-location and anti-colocation contracts were the addition of small XML stanzas to the existing virtual machine XML definition files. Upon achieving some success with constructing optimal consolidation plans, we then set out to construct an algorithm for high performance live migration plan generation.

demonstrate correctness of our implementation with respect to selecting viable heuristics used within our computation of $H(x)$ and $G(x)$. In addition, this allowed us to construct a what-if mode similar to that espoused in the VMware paper [2]. In other words, we use the simulation approach to verify our algorithms, but we believe the simulator could also be used generally to enable system administrators to be able to forecast the results of an arbitrary migration on an arbitrary VM to host mapping. Practically speaking, data center operators could use our simulator to observe the recommended migration plan in action before they execute it. This would allow them to increase their trust and confidence in the automated VM allocation/ migration strategy for some time before they trust an autonomously managed migration control system.

2.2. Test Cases

We constructed a number of test cases to illustrate the performance impact of our A* algorithm under realistic scenarios that an automated VM orchestration system might encounter. In each of our cloud modeling scenarios, VMs were initially distributed using strategies that emulate potential data center VM allocations (e.g., random, consolidated, or load-balanced placement). Tests varied in the nature of the initial condition and the resulting VM distribution. For each test, we measured runtime, memory consumption, performance, and ensured correctness while varying the number of hosts and VMs simulated.

3. Early Results and Experimentation

To support our investigations into the suitability of this approach, we systematically ran over 1,000 simulations on an IBM Blade Server® with 96GB RAM, and 24 logical CPU cores (Intel Xeon E5-2640). The simulations varied in VM counts, host counts, initial VM distribution mechanism, desired migration policy (consolidation, load balancing, etc…), all while checking each step in the solution generation process for correctness to validate the work presented here. Our largest (in terms of managed objects) experiment simulated 200 hosts and 1,760 VMs (a consolidation ratio of 8.8:1). The largest simulated consolidation ratio was 9:1 while the lowest was 3:1. The mean solution generation time across all experiments was 1 minute and 24 seconds, with a standard and deviation of 3 minutes and 12 seconds. The longest A* solution time that completed ran for just over 25 minutes, and 90% of solutions were generated in less than 4 minutes. As expected, we experimentally observed runtime correlates highly with RAM consumption.

4. Conclusion

In this paper, we introduced a novel means for creating an optimal path from an arbitrary initial VM to host mapping onto a goal-state mapping through the application of A*. This approach is complementary to placement algorithms that consider migration planning but is justifiably done separately as a means to isolate the mechanism of migration planning from VM placement policy. Simulations indicate efficient implementations of A* are viable for moderate-sized data centers. An extended version of this work including more complete implementation details and the solution parallelization is forthcoming.

5. References

[1] Hyser, C., Mckee, B., Gardner, R., Watson, B.J.: Autonomic vir-tual machine placement in the data center. HP Labs Tech Report HPL-2007-189, Feb 2007

[2] Gulati, Ajay, et al. "VMware distributed resource management: Design, implementation, and lessons learned." VMware Technical Journal 1.1 : 45-64. 2012

[3] IBM United States Software Announcement 213-590, dated December 10, 2013. Available online: http://www-01.ibm.com/common/ssi/rep_ca/0/897/ENUS213-590/ENUS213-590.PDF

[4] Tantawi, A. N. "A Scalable Algorithm for Placement of Virtual Clusters in Large Data Centers," in Proceedings of the 2012 IEEE 20th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, Washington, DC, USA, 2012, pp. 3–10.

[5] Hart, P. E.; Nilsson, N. J.; Raphael, B. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". IEEE Transactions on Systems Science and Cybernetics SSC4 4 (2): 100–107. 1968

[6] Algfoor, Z. A., Sunar, M. S., & Kolivand, H. "A comprehensive study on pathfinding techniques for robotics and video games." International Journal of Computer Games Technology, 2015.

[7] C. Popirlan, M. Dupac, "An Optimal Path Algorithm for Autonomous Searching Robots.", Annals of University of Craiova, Math. Comp. Sci. Ser. Vol. 36(1), pp. 37-48, (2009)

[8] Matthews, J., Garfinkel, Tal., Hoff, C., Wheeler, J. Virtual machine contracts for datacenter and cloud computing environments. In Proceedings: 1st workshop on Automated control for datacenters and clouds (ACDC '09). ACM, New York, NY, USA, 25-30.