Inciting Cloud Virtual Machine Reallocation With Supervised Machine Learning and Time Series Forecasts

Eli M. Dow
IBM Research, Yorktown NY

What is this talk about?

- This is a 30 minute technical talk about building autonomous cloud management systems that determines if Virtual Machines are unhappy where they reside.

- My intention is to give enough insight that anyone inclined could build a system based on these recommendations.

Motivation

The state of the art in VM dynamic placement is not sufficient to meet the needs of certain types of cloud computing data centers. Most research focuses on the "initial admittance problem" or requires application level instrumentation.

Cloud computing needs to be inexpensive … Low consumer cost leads to a need for automation and lights-out data centers. The provider goal is to use hardware resources effectively…

What do we mean by effectively? Depends on your state of mind. Consolidation vs. Load-balancing. Opposite ends of the spectrum.

People managing VM placement is a poor idea. Explicit constraints? Hidden constraints?

Automated scheduling is NP hard for long lived processes, and you can think of VMs as a long lived process especially in the KVM context where the mapping is literally 1VM to 1 process

The goal of the broader research objective that subsumes the work in this talk is to build a data center that migrates VM's using live guest migration to move cloud servers around as rapidly as we move around local processes on a single multi-core server. Note: Migration times are sub second for most VMs using RDMA style networks etc, so we can almost achieve this if only we can automate the migration decision making processes.

Automating VM Placement

- We want to automate VM placement according to some policy (load-balanced? consolidated? both?)
- Have flexibility to adapt to runtime changes in workload/compute rescue availability with agility
- Seems easy enough[1] so lets try to automate this!

[1] But seriously this is hard, so lets look at what people have done before…

Prior Work

- Tons of work on load balancing and consolidation strategies. Each their own discipline entirely[2]. The paper associated with this presentation has a good survey of the work from the literature over the last several decades.

- Open problems - “The VM Colocation Problem” says that not all VM’s may co-reside peacefully. Examples:

  - Privacy mandates

  - High-Availabilty (HA) peers

[2] I have my own take on high performance consolidation for VMs in a paper under review entitled “”

To do the management part, we first need a monitoring part!

- To do the monitoring required to load balance or consolidate there are 2 choices really:

1. In Band (Agent-based) — Put an agent into each of the VMs to collect data! (not alway ideal because it can be resource wasteful and requires custom configuration in each VM. Non-starter for IaaS…)

2. Out of Band (“Agent-less”) — Put one agent per hypervisor instance (Also a terrible idea… but only because it is harder to do!)

Is vigilance is the key to success?

- IBM Research paper titled Vigilant [3] looked at detecting faults inside virtual machines from a completely black box perspective.
- They effectively did some monitoring from the hypervisor perspective by instrumenting QEMU and other things to do some machine learning about the state of the VMs without altering them directly.
- What did they measure?
  - The number of times a guest VM was scheduled.
  - Utilization of CPU by the guest VM.
  - Time spent in the “runnable” state.
  - Time spent waiting on events.
- What were their results? They received good results, finding faults with high skill (accuracy), but the results were not very generic as they used a custom Matlab machine learning implementation to learn failure states vs nominal states.
- Can we use their general approach to learn more about the VMs being managed autonomously?

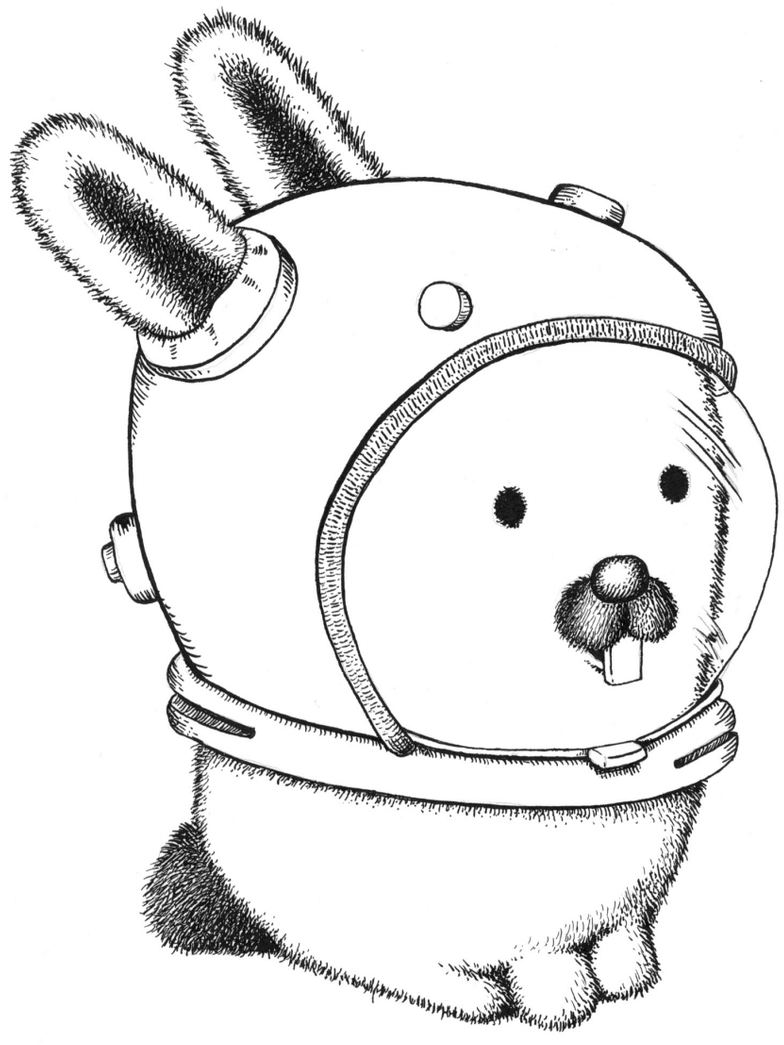
[3] Vigilant - Dan Pelleg, Muli Ben-Yehuda, Rick Harper, Lisa Spainhower, and Tokunbo Adeshiyan. 2008. Vigilant: out-of-band detection of failures in virtual machines. SIGOPS Oper. Syst. Rev. 42, 1 (January 2008), 26-31.

Beyond Vigilance?

- What can we measure out of band (hypervisor level) for KVM VMs that might let us place VMs intelligently and automatically? (A lot)
- But can we do this efficiently enough to be responsive (pseudo real time)?

- CPU consumption as a percentage of the host
- Characters read in memory
- Characters written to memory
- Bytes read to memory
- Bytes written to memory
- Bytes received over a net connection
- Packets received over a net connection
- Read packets errors encountered on a net connection
- Read packets dropped on a net connection
- Bytes sent over a net connection
- Packets sent over a net connection
- Written packets errors encountered on a net connection
- Written packets dropped on a net connection
- Page faults incurred during the interval
- Time spent blocked on the scheduler run queue

Glenda (Space Bunny) to the rescue! (*/Proc* File System[4])



[4] Glenda is the evil creation of Renee French for the purposes of total and utter world domination as part of the Plan 9 from Bell Labs.

To do the management part, we need a monitoring part

- In KVM, all virtual machines map to a single process just like your email client or web browser. Also note that the KVM kernel and the Linux kernel are the same thing.

- We can read from */proc* file system for the data values about each process running, and by extension we can read information about each virtual machine.

- Implementation in C allows us to measure values every 3 seconds with no measurable overhead on modern hardware. Note that */proc* file system reads are effectively memory reads from the Linux kernel data structures for managing processes.

We have data about VMs, now what?

- Lets take a page form the vigilant playbook and try to learn what it is like for virtual machines to be happy (nominal) or sad (constrained)…

- I have no idea from looking at the raw data what constitutes a happy or sad VM…

- So rather than guessing, we can use the machine learning approach to infer those salient characteristics from the data…

What Machine Learning Implementation Should We Use?

- We are effectively looking for binary classification here and we are seeking supervised learning techniques by running some benchmarks on the VMs to determine their nominal and constrained behavior.

- We defer to literature for supervised machine learning classification:
    - Support Vector Machines - recently caught the world on fire. They work well on lots of problems but are sort of magic (hyperplanes and such).
    - Decision Tree Classifiers - not as in vogue presently, but have the elegant property that their derived classification strategy can be explained to people in english

- What should we do? Which do we pick? I am wracked with indecisiveness…

Do Both?

- We opted to do both!
- In a previous work[5] I published a means for taking a single data stream and constructing trained SVM models with LibSVM as well constructing a Binary Decision tree classifier model in parallel.
- The work also compiled the binary decision tree output into a small shared object code embodiment that could be rapidly evaluated at runtime. These modules can be swapped out at runtime as newer models are learned/ uploaded to each hypervisor.
- We recycled this more generic work and built a VM runtime classification system.
- Our monitor obtains the runtime parameters on a per-vm basis then invokes the trained classifier periodically to make sure the VM is still in a happy state.

[5] The Journal article was entitled “Transplanting Binary Decision Trees”

Anticipation

A reactionary approach to problems as they occur is a horrible strategy in general.

So we can detect problems happening right now… but maybe we should anticipate them?

Sounds like we need some predictive analytics to forecast future VM unhappiness…

Predictive Analytics

- In this case, analytics = signal processing + math
- There have been attempts at sophisticated resource forecasting systems that are very complicated to explain and build. So why not simplify a bit and see how well we can do?
- “New Math” vs “Old Math” (I opt to go with the math from the 1960s)
- The resource measurements for each vm can be seen as a signal or time series. For instance, CPU consumption over time makes a line graph that you can think of as a signal.
- Let us apply some univariate time series forecasting techniques…

Smooth Criminal

- Harmonic means (for averaging rates)
- Linear Regression (do we need anything more than good old y=mx+b?)
- Exponential Smoothing family of algorithms (to deal with trends, periodicity, and periodic trends when lines wont do!)
- A neural network (it sounds great doesn't it?)

How To Predict The Future[6]

- We mine the VMs at runtime as described before, and in addition to classifying for happiness right now, we predict the forecast for each parameter and then classify again using the estimated sure state to ensure the VM will remain happy for some period of time.

- How far do we need to forecast? The answer is that you must forecast the duration of time that it would take to move the longest-migrating VM you have (which is a function of the paging rate and memory size assuming no persistent disk migration).

- Generally we only need to forecast some small interval (5 minutes or so).

[6] The formulas needed are included in the paper reference etc along with applicability explanations.

Recap:
Contributions to the Art

In summary, the *Incite* work presented here demonstrates the rigorous design methodology and technical material necessary for cloud developers and practitioners to create a viable replication of our first-of-a-kind system for using runtime classification of virtual machines as over-constrained for the purposes of triggering workload balancing suitable for deployment in an IaaS cloud computing environment.

The combination of exceptionally light weight data mining, optional hierarchical time series forecasting, and high performance run time classification functions allows for rapid indication of potential over-constraint situations suitable for triggering remedial action or initiating system operator alerts.

The goal is autonomous, frequent, reallocation of VM resources in the cloud data center that respect explicit and implicit constraints.

What Next?

Thank You!