Friendly Virtual Machines

Leveraging a Feedback-Control Model for Application Adaptation

Azer Bestavros

Joint work with

Yuting Zhang, Mina Guirguis, Ibrahim Matta, and Richard West

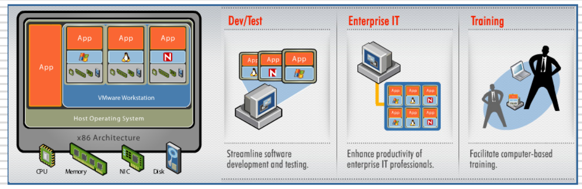
http://www.cs.bu.edu/groups/wing

First ACM/USENIX Conference on Virtual Execution Environments (VEE'05)

June 11th, 2005

Motivation: Trend #1

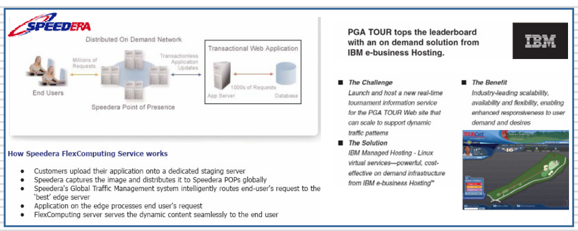
- Emergence/acceptance of VM abstraction
  - OTS VMware, UML, IBM Virtual Hosting solutions (circa '05)
  - Used mostly in closed, managed environments



[Graphics from http://www.vmware.com/]

Motivation: Trend #2

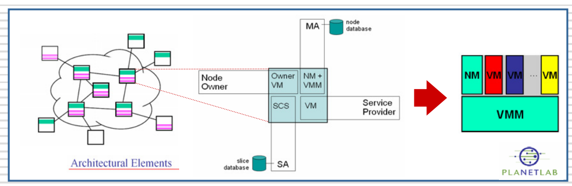
- Apps running on shared 3rd party hosts
  - PlanetLab and Emulab experimental testbeds
  - IBM, Akamai, Speedera edge-computing & hosting services



[Graphics from http://www.speedera.com & http://www.ibm.com]

Motivation: Trend #3

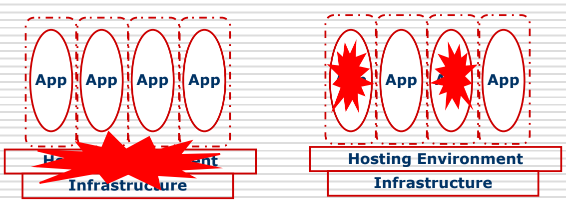
- Need to isolate independent constituents
  - Virtual web hosting; e.g., Mozilla Application VM (circa '01)
  - Shared infrastructures; e.g., Grids, Sensoria, overlays
  - PlanetLab's use of VMs for various services on a single host



[Graphics from http://www.planet-lab.org/]

Shared Host Resources: Issues

- Under-provisioned Host → Overload
  - Inefficient use of host resources
  - Unpredictability due to OS thrash mitigation measures
  - Unfair/uninformed allocation of resources to applications



Resource Allocation: How?

Three Schools of Thought

- *OS or VMM micromanages access to resources*
  - Adds complexity to common infrastructure
  - Agnostic to application adaptation
  - Special APIs not suitable for open systems
- *Reservation based allocation*
  - Inefficient, especially with highly dynamic applications
  - Incompatible with inherently "best-effort" resources
  - Hosting infrastructure must police applications
- *Best-effort allocation with overload protection*
  - Simple common infrastructure
  - Applications must adapt to resource allocation
  - No notion of fairness among disparate apps

Resource Allocation: Wish List

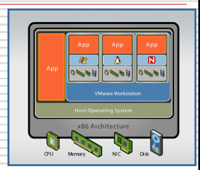
- Simple hosting infrastructure
  - Macro, not micro-management; OK to monitor, police, …
- Application autonomy
  - No explicit coordination between applications or with host
- Performance isolation
  - Applications with different bottlenecks need not tussle
- Convergence to fairness
  - System should converge to a fair allocation of resources
- Efficient resource utilization
  - No overload; no underutilization

Our Solution: E2E-style

- Minimal Host Functionality
  - Best-effort, round-robin-style resource allocation
  - Provide "congestion" feedback signal to apps
  - Implement policing of non-compliant apps
- Adaptive Resource Consumption by Apps
  - Probe available resources and react to congestion
  - Adaptation mechanisms may vary to suit apps
  - Compliance, or friendliness is well defined

An Instance of Host Sharing

- VMs as applications
  - VMs used to provide isolation, namely safety and security
  - Hosts are powerful enough to support many VMs
  - VMs compete for host resources and may exhibit radically different resource needs (e.g., memory-bound, CPU-bound, I/O-bound, net-bound, …)



Our E2E Solution: Friendly VMs

- VMs *adapt* their resource *consumption rate* based on congestion *feedback signal*
- Benefits:
  - Minimal resource management in host OS/VMM
  - Friendly (efficient and fair) sharing of system resources among VMs
  - Transparent to the application on top as well as the OS hosting the VMs below

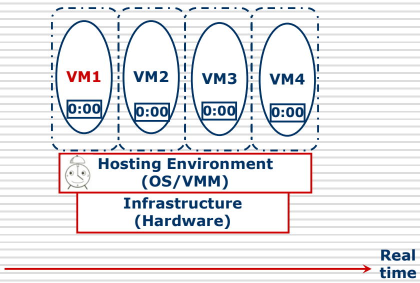
Our E2E Solution: Friendly VMs

- VMs *adapt* their resource *consumption rate* based on congestion *feedback signal*
- Elements of the solution:
  - What constitutes the feedback signal?
  - How to control consumption rate?
  - What adaptation strategy is appropriate?

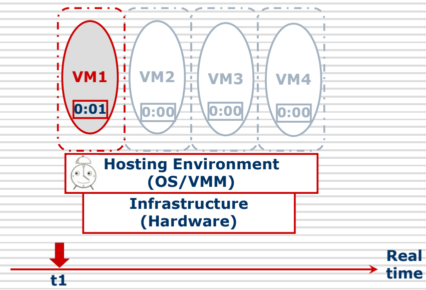
FVM: Feedback Signal

- Virtual Clock Time (VCT)
  - VCT is the time interval between two consecutive virtual clock ticks (of the VM)
  - VCT is the VM response time; it is analogous to the RTT of a network flow
  - Use VCT (or derivative thereof) to generate feedback signal

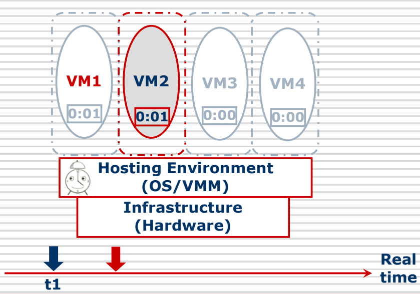
VCT: Illustration #1



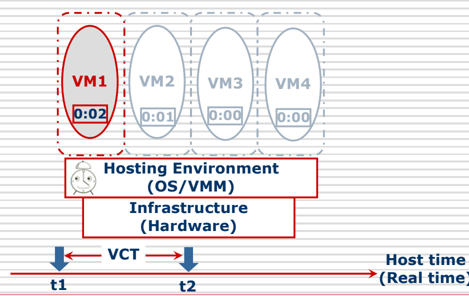
VCT: Illustration #1



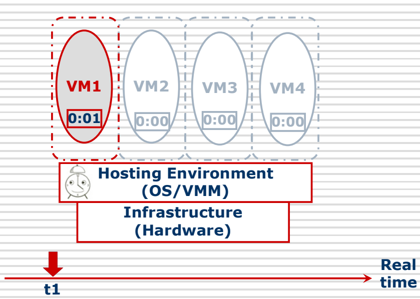
VCT: Illustration #1



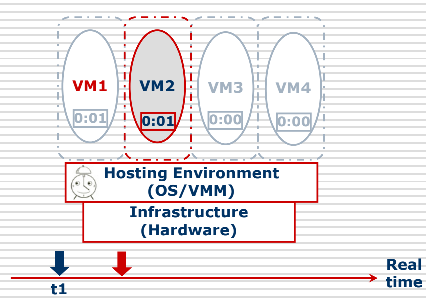
VCT: Illustration #1



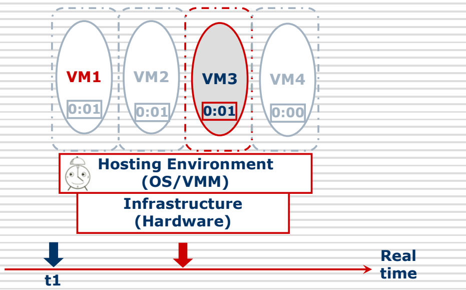
VCT: Illustration #2



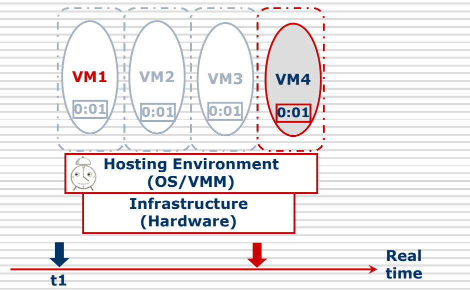
VCT: Illustration #2



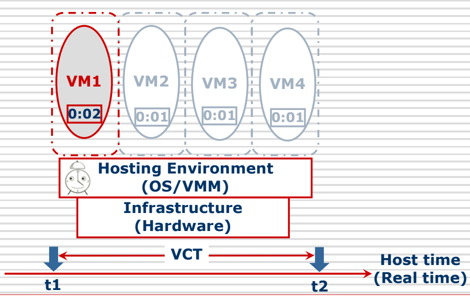
VCT: Illustration #2



VCT: Illustration #2

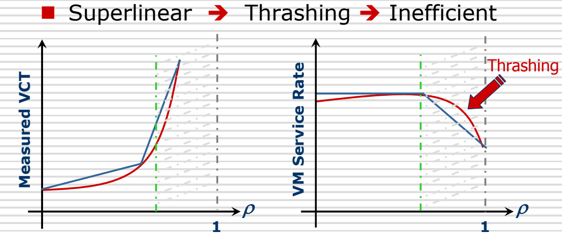


VCT: Illustration #2



VCT: A Barometer of Load

- VCT grows with load (e.g., #VMs)
  - Slow growth ~ Linear ➔ Efficient
  - Superlinear ➔ Thrashing ➔ Inefficient

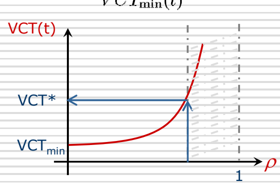


FVM: Congestion Signal

EWMA VCT: $\overline{VCT}(t) = (1-\gamma)\cdot\overline{VCT}(t-1)+\gamma\cdot VCT(t)$

Minimum VCT: $VCT_{\min}(t) = min\{\overline{VCT}(t-i) : i = 1,\ldots,w\}$

Slowdown: $R = \frac{\overline{VCT}(t)}{VCT_{\min}(t)}$



R > Threshold (H) ➔ Congestion = True
where H ~ VCT*/VCTmin

FVM: Consumption Control

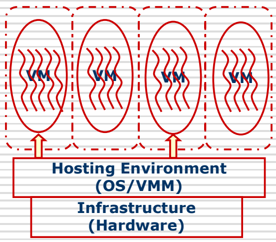
- Multi-Programming Level (MPL) Control:
  - A thread as a unit of consumption of host resources; VM is a multi-threaded application
  - # of active threads allowed for a VM constitute a cap on its resource consumption
  - Adjust # of active threads through suspension or resumption of threads within a VM

A la TCP window adaptation

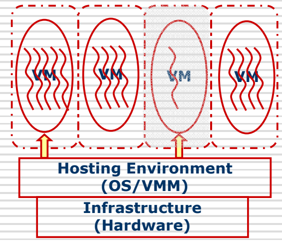
A la TCP timeout

- Rate Control:
  - Force VM to periodically sleep (or timeout)

FVM: MPL Control



FVM: Rate (timeout) Control



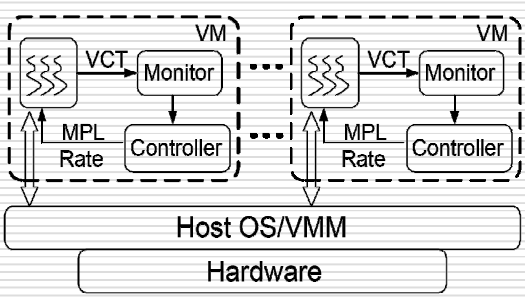
Controller: Adaptation Strategy

- AIMD (Additive-increase/Multiplicative-decrease)
  - **Adjust # of threads**
    - No Congestion → $thread_{max}$ = $thread_{max}$ + a;
    - Congestion → $thread_{max}$ = $thread_{max}$ / b;
  - **Adjust execution rate (timeout period)**
    - No Congestion → rate = rate + a;
    - Congestion → rate = rate / b;
- Different increase/decrease rules that match application requirements (e.g., smoother adaptation) could co-exist as long as they are "compatible" [JinGuoBestavrosMatta'02]

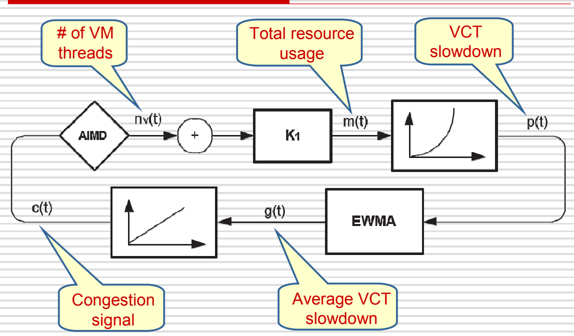
Host: Requirements

- Required:
  - Unbiased On-Demand Allocation
    - RR scheduler
- Desirable:
  - Policing Functionality (friendliness incentive)
    - Identify/penalize misbehaving applications

FVM: Framework



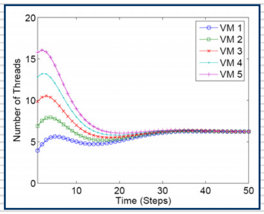
FVM: Analytic Model



FVM: Analytic Model Results

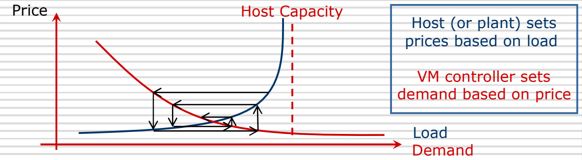
Linearized model allows us to:

- Relate convergence & transient characteristics to parameters, e.g., AIMD/EWMA constants, various delays, gain, …
- Sketch adaptation transients numerically



FVM: Convergence

- Congestion signal constitutes prices fed back to VMs as the load on the host varies



- Convergence and stability can be proved through Lyapunov function [Kelly'99]

f-UML: A FVM Prototype

- Based on User Mode Linux (UML) VM
  - UML is a VM abstraction that allows guest Linux systems to run at user-level on top of a Linux host
- Added ~ 500 lines to UML code
  - VCT measurement, congestion signal generation, and controller implemented in a single function `fvm_adapt()` which is added to the `time_handler()` for SIGALRM/SIGVALRM

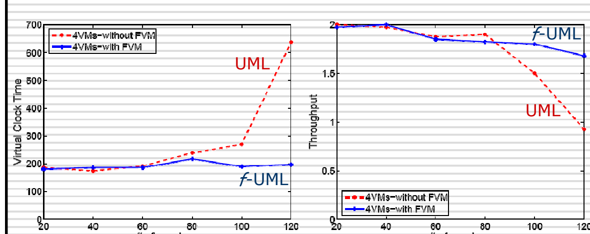
f-UML: Parameters

Parameter	Setting
Control Period	5 sec
Window of VCT_{min}	60 sec
EWMA constant for $\overline{VCT}$	0.3
Initial limit on the number of thread	10
Slowdown threshold	2.5
AIMD additive constant (MPL control)	1 thread
AIMD additive constant (rate control)	0.1 Hz (=1/Ts)
AIMD multiplicative constant	1.5

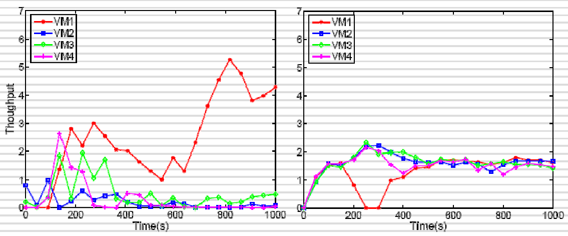
f-UML: Evaluation

- Web server experiments
  - 4 VMs on host, with Apache 2.0 running on each VM
  - Client requests invoke memory-bound CGI scripts
- VMs tested
  - Original UML
  - *f*-UML prototype (with MPL control)
- Metrics (per VM & averaged over 4 VMs)
  - VCT
  - Throughput
  - Fairness Index

f-UML vs UML: Baseline Results

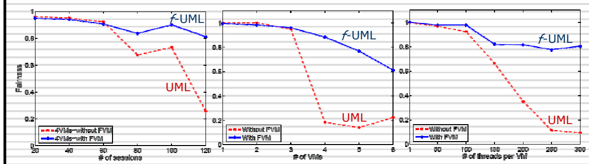


f-UML vs UML: Throughput



UML *f*-UML

f-UML vs UML: Fairness



$$\mathrm{FI} \;=\; 1-\frac{\sqrt{\sum_{i=1}^{V}(x_i-o_i)^2}}{\sqrt{\sum_{i=1}^{V}o_i^2}}$$

FVM: Food for Thought

VM resource consumption throttling:

- What if all threads are not created equal?
- Which thread should be suspended?

FVM framework extensions:

- Other feedback signals? adaptation mechanisms? …
- Extension to friendliness over host clusters, grids, …

Friendly wrappers:

- Could an application be made friendly post-mortem?
- Could friendliness be "strongly typed"?
- What is the role of compilers in casting friendliness?

Take Home Messages

VM Friendliness

- The incarnation of the E2E argument for multi-resource management in shared hosting environments
- A resource consumption etiquette that leaves the choice of mechanism used for compliance to the application

FVM Framework

- Lends itself well to emerging open VM hosting systems
- Reduces significantly the complexity of underlying host
- *f*-UML implementation establishes feasibility

Friendly Virtual Machines

Leveraging a Feedback-Control Model for Application Adaptation

More information available from WING Publications

http://www.cs.bu.edu/groups/wing

http://www.cs.bu.edu/groups/wing

First ACM/USENIX Conference on Virtual Execution Environments (VEE'05)

June 11th, 2005