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

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

Motivation: Trend #2
- Apps running on shared 3rd party hosts
  - PlanetLab and Emulab experimental testbeds
  - IBM, Akamai, Speedera edge-computing & hosting services

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

Shared Host Resources: Issues
- Under-provisioned Host ➔ Overload
  - Inefficient use of host resources
  - Unpredictability due to OS thrash mitigation measures
  - Unfair/uniformed 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

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 #2

Infrastructure (Hardware)
Hosting Environment (OS/VMM)

Real time

0:01

VM1
VM2
VM3
VM4

VCT: A Barometer of Load

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

\[ \rho \]

\[ \frac{1}{\text{VM Service Rate}} \]

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

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

\[ R > \text{Threshold (H)} \Rightarrow \text{Congestion = True} \]

where \( H \sim \frac{\text{VCT}^*}{\text{VCT}_{\text{min}}} \)
**Controller: Adaptation Strategy**

- **AIMD (Additive-increase/Multiplicative-decrease)**
  - Adjust # of threads
    - No Congestion: \[ \text{thread}_{\text{max}} = \text{thread}_{\text{max}} + a; \]
    - Congestion: \[ \text{thread}_{\text{max}} = \frac{\text{thread}_{\text{max}}}{b}; \]
  - Adjust execution rate (timeout period)
    - No Congestion: \[ \text{rate} = \text{rate} + a; \]
    - Congestion: \[ \text{rate} = \frac{\text{rate}}{b}; \]
- 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

![Graph showing model results]

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]

![Graph illustrating convergence]

\-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

\-UML: Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control Period</td>
<td>5 sec</td>
</tr>
<tr>
<td>Window of VCT</td>
<td>60 sec</td>
</tr>
<tr>
<td>EWMA constant for VCT</td>
<td>0.3</td>
</tr>
<tr>
<td>Initial limit on the number of thread</td>
<td>10</td>
</tr>
<tr>
<td>Slowdown threshold</td>
<td>2.5</td>
</tr>
<tr>
<td>AIMD additive constant (MPL control)</td>
<td>1 thread</td>
</tr>
<tr>
<td>AIMD additive constant (rate control)</td>
<td>0.1 Hz (=1/Ts)</td>
</tr>
<tr>
<td>AIMD multiplicative constant</td>
<td>1.5</td>
</tr>
</tbody>
</table>

\-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
  - \-UML prototype (with MPL control)
- Metrics (per VM & averaged over 4 VMs)
  - VCT
  - Throughput
  - Fairness Index

\-UML vs UML: Baseline Results

![Graph comparing \-UML and UML performance]
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