Tableau: A High-Throughput and Predictable VM Scheduler for High-Density Workloads

Manohar Vanga, Arpan Gujarati, Björn Brandenburg

MPI-SWS, Kaiserslautern, Germany
How to support high-density VM workloads

Many small VMs packed onto few cores
Why High Density?

Competitive market driving datacenter efficiency

- Google Cloud Platform
- Rackspace
- Microsoft Azure
- Amazon Web Services
- IBM Cloud
Why High Density?

High-Density VM Packing
Consolidating small, cheap VMs to use fewer resources.

Competitive market driving datacenter efficiency
Why High Density?

**High-Density VM Packing**

Consolidating small, cheap VMs to use fewer resources.

**Challenge**

Must continue to provide **consistent throughput** and **predictable latency tails**.
VM Scheduling Crucial for High-Density
VM Scheduling Crucial for High-Density

- Many VMs per core
- Many runtime decisions for allocating CPU time
- VM scheduler performance can have significant impact
Case Study: VM Scheduling in Xen
Case Study: VM Scheduling in Xen

- **Four** VMs per core, 16-core server
- Intel(R) Xeon(R) CPU E5-2667 v4 @ 3.20GHz.
- Measure **HTTPs performance** of one VM
- All other VMs running **I/O-bound stress workload**.
Case Study: VM Scheduling in Xen

99th Percentile Latency (ms) vs. Observed Throughput (requests per second)

Lower is better

More to the right is better
Case Study: VM Scheduling in Xen

- **Lower is better**
- **More to the right is better**

- Observed Throughput (requests per second)
- 99th Percentile Latency (ms)
Case Study: VM Scheduling in Xen

Credit: Tableau

Observed Throughput (requests per second)

99th Percentile Latency (ms)
Case Study: VM Scheduling in Xen

Default fair-share scheduler used in production.

Real-time scheduler (based on RT-Xen) for latency-sensitive workloads.
Case Study: VM Scheduling in Xen

Credit

Requesting random 100K-sized files, with I/O background workload
Case Study: VM Scheduling in Xen

Credit has increasing latency tails.

Credit provides much higher throughput.

Requesting random 100K-sized files, with I/O background workload
Case Study: VM Scheduling in Xen

RTDS provides limited throughput.

RTDS has consistent tail latencies across entire throughput range.

Requesting random 100K-sized files, with I/O background workload.
The Tableau VM Scheduler

Observing 99th percentile latency (ms) and observed throughput (requests per second) for different schedulers:

- **Credit**
- **Tableau**
- **RTDS**

Requested random 100K-sized files, with I/O background workload.
Contributions

Tableau

An unorthodox scheduling approach tailored for high-density public clouds.
Contributions

Tableau

An unorthodox scheduling approach tailored for high-density public clouds.

Efficient
Incurs low overheads

Predictable
Accurate control over scheduling latency.

High-throughput
Provides high SLA-aware throughput.
This Talk

- Tableau
- Evaluation
- Conclusion
This Talk

- Tableau
- Evaluation
- Conclusion
What Do We Want From a VM Scheduler?

• **Requirement 1**: Be as "invisible" as possible.

• **Requirement 2**: Guarantee utilization and ensure predictable scheduling latency for every VM.
What Do We Want From a VM Scheduler?

• **Requirement 1**: Be as "invisible" as possible.

• **Requirement 2**: Guarantee utilization and ensure predictable scheduling latency for every VM.
What Do We Want From a VM Scheduler?

• **Requirement 1**: Be as "invisible" as possible.

• **Requirement 2**: Guarantee utilization and ensure predictable scheduling latency for every VM.

Requirement 2 is a **non-trivial** problem!
What Do We Want From a VM Scheduler?

- **Requirement 1**: Be as "invisible" as possible.
- **Requirement 2**: Guarantee utilization and ensure predictable scheduling latency for every VM.

Requirement 2 is a **non-trivial** problem!

Attempting to enforce requirement 2 at **runtime** conflicts with requirement 1.
What Do We Want From a VM Scheduler?

- **Requirement 1**: Be as "invisible" as possible.
- **Requirement 2**: Guarantee utilization and ensure predictable scheduling latency for every VM.

Requirement 2 is a **non-trivial** problem!

Attempting to enforce requirement 2 **at runtime** conflicts with requirement 1.

How do we overcome these conflicting requirements?
The Tableau Approach

Exploit one key property of VM environments

VM churn on a single server is low \(^1\)

# The Tableau Approach

<table>
<thead>
<tr>
<th>Requirement 1</th>
<th>Requirement 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>As invisible as possible. Fast, Low overhead</td>
<td>Guarantee utilization and scheduling latency</td>
</tr>
</tbody>
</table>
The Tableau Approach

**Requirement 1**
As invisible as possible.
Fast, Low overhead

**Requirement 2**
Guarantee utilization and scheduling latency

Table-Driven Dispatcher
The Tableau Approach

Requirement 1
As invisible as possible. Fast, Low overhead

Requirement 2
Guarantee utilization and scheduling latency

Apply scheduling theory from hard real-time systems.

Table-Driven Dispatcher

Semi-Offline Table Planner
The Tableau Approach

**Requirement 1**
As invisible as possible. Fast, Low overhead

**Requirement 2**
Guarantee utilization and scheduling latency

- Table-Driven Dispatcher
- Semi-Offline Table Planner
- Mechanism
- Policy
**The Tableau Approach**

**Requirement 1**
As invisible as possible.
Fast, Low overhead

**Requirement 2**
Guarantee utilization and scheduling latency

- **Table-Driven Dispatcher**
- **Semi-Offline Table Planner**

Dispatcher is **completely unaware** of VM-specific requirements!
The Tableau Approach

Requirement 1
As invisible as possible. Fast, Low overhead

Requirement 2
Guarantee utilization and scheduling latency

Table-Driven Dispatcher
Mechanism

Semi-Offline Table Planner
Policy

Dispatcher is **completely unaware** of VM-specific requirements!

Easy to extend using **high-level** languages, tools, and libraries.
The Tableau Approach

Requirement 1
As invisible as possible.
Fast, Low overhead

- Table-Driven Dispatcher
- Mechanism

Dispatcher is completely unaware of VM-specific requirements!

Requirement 2
Guarantee utilization and scheduling latency

- Semi-Offline Table Planner
- Policy

Easy to extend using high-level languages, tools, and libraries.
Can be pre-generated or generated on a separate machine.
The Tableau Approach

- Table-Driven Dispatcher
- Mechanism

- Semi-Offline Table Planner
- Policy
Generating Tables Quickly

Set of VMs

Each configured with a utilization and max. scheduling latency.
Generating Tables Quickly

Set of VMs

Each configured with a **utilization** and **max. scheduling latency**.

No more information than existing schedulers (e.g., Credit requires a relative weight and timeslice)
Generating Tables Quickly

Set of VMs

Model each VM as a periodic task\(^1\).

Each configured with a utilization and max. scheduling latency.

Generating Tables Quickly

Set of VMs
- Each configured with a utilization and max. scheduling latency.

Model each VM as a periodic task\(^1\).

Partitioning

Apply recent scheduling theory from hard real-time systems.

Scheduling Table

---

Generating Tables Quickly

- Set of VMs
  - Each configured with a utilization and max. scheduling latency.

Model each VM as a periodic task\(^1\).

Performed **entirely in userspace** of supervisory VM.

Implemented **in Python using a mature library (SchedCAT)**.

Apply recent scheduling theory from hard real-time systems.

Scheduling Table

---

Generating Tables Quickly

Set of VMs

Each configured with a utilization and max. scheduling latency.

Model each VM as a periodic task\(^1\).

Partitioning

Apply recent scheduling theory from hard real-time systems.

Scheduling Table

---

Modeling VMs as Periodic Tasks

VM (vCPU)

*Utilization (U)*
A percentage of CPU time reserved for VM.

*Max Sched. Delay (L)*
An upper bound on scheduling delay.

Periodic Task

*Budget (C)*

*Period (T)*
Modeling VMs as Periodic Tasks

**VM (vCPU)**

**Utilization (U)**
A percentage of CPU time reserved for VM.

**Max Sched. Delay (L)**
An upper bound on scheduling delay.

**Periodic Task**

**Budget (C)**

**Period (P)**
Modeling VMs as Periodic Tasks

**VM (vCPU)**

**Utilization (U)**
A percentage of CPU time reserved for VM.

**Max Sched. Delay (L)**
An upper bound on scheduling delay.

---

Periodic Task

**Budget (C)**

**Period (P)**

---

Periodic Task with budget and period.
Modeling VMs as Periodic Tasks

**VM (vCPU)**

**Utilization (U)**
A percentage of CPU time reserved for VM.

**Max Sched. Delay (L)**
An upper bound on scheduling delay.

---

**Periodic Task**

**Budget (C)**

**Period (P)**

---

**2 x (P - C)**

---
Generating Tables Quickly

Set of VMs

Each configured with a utilization and max. scheduling latency.

Model each VM as a periodic task.

Partitioning

Apply recent scheduling theory from hard real-time systems.

Scheduling Table
Table-Generation Times

- Number of VMs
- Time (in seconds)

- All VMs 1ms
- All VMs 5ms
- All VMs 30ms
- All VMs 100ms
Table-Generation Times

Lower is better
Table-Generation Times

- Number of VMs: 0, 20, 40, 60, 80, 100, 120, 140, 160, 180
- Time (in seconds): 0, 0.5, 1, 1.5, 2

Lines in the graph:
- Red circles: All VMs 1ms
- Green crosses: All VMs 5ms
- Blue plus signs: All VMs 30ms
- Blue squares: All VMs 100ms
Table-generation times are reasonable compared to VM creation and teardown times.
The Tableau Approach

Table-Driven Dispatcher

Mechanism

Semi-Offline Table Planner

Policy
Implementation in Xen

- Popular open-source hypervisor (Amazon AWS)
- Supervisory VM (domain-0) created at boot time.
• Simple, table-driven dispatcher implemented within the hypervisor.
• Userspace daemon responsible for re-generating tables whenever a VM is created.

• ~1,600 lines of Python code.
For work-conserving behavior, idle time in tables (white blocks) yields to round-robin scheduler. Picks runnable core-local VMs to schedule.
This Talk

- Tableau
- Evaluation
- Conclusion
Summary of Results

Tableau incurs lower runtime overheads compared to the other evaluated Xen schedulers.
Summary of Results

Tableau incurs lower runtime overheads compared to the other evaluated Xen schedulers.

Tableau enables accurate control over scheduling latency.
Tableau incurs lower runtime overheads compared to the other evaluated Xen schedulers.

Tableau enables accurate control over scheduling latency.

Tableau achieves higher SLA-aware application throughput.
Summary of Results

Tableau incurs lower runtime overheads compared to the other evaluated Xen schedulers.

See our paper for details!

Tableau enables accurate control over scheduling latency.

Tableau achieves higher SLA-aware application throughput.
Platform

- **Server machine:**
  - 16 cores (2 sockets), 512 GiB RAM
  - Intel(R) Xeon(R) CPU E5-2667 v4 @ 3.20GHz
  - Ubuntu 16.04.3
  - Xen 4.9

- **Load generation machine:**
  - Identical machine connected via 10G ethernet.
Experimental Setup

- We simulate a multi-tenant datacenter environment.
  - 4 VMs/core (25% utilization each).
  - 1 **vantage** VM, rest **background** VMs
  - Background VMs run different workloads based on **stress-ng** tool.

- Schedulers configured based on best practices:
  - 5ms timeslice in Credit.
  - Equivalent configuration in Tableau and RTDS (max 20ms scheduling latency)
SLA-Aware Throughput

99th Percentile Latency (ms)

Observed Throughput (requests per second)
SLA-Aware Throughput

99th Percentile Latency (ms)

Observed Throughput (requests per second)

Lower is better

More to the right is better
Peak Throughput

Credit  
Tableau  
RTDS

Observed Throughput (requests per second)

99th Percentile Latency (ms)

VMs Capped at 25%, 100K files, I/O background workload
Observed Throughput (requests per second)

99th Percentile Latency (ms)

VMs Capped at 25%, 100K files, I/O background workload
Tableau achieves the highest peak throughput.

VMs Capped at 25%, 100K files, I/O background workload
Credit achieves comparable peak throughput but latencies rise earlier.
**Peak Throughput**

- **Credit**
- **Tableau**
- **RTDS**

RTDS provides controlled latencies but **sacrifices throughput**.

VMs Capped at 25%, 100K files, I/O background workload
SLA-Aware Throughput

VMs Capped at 25%, 100K files, I/O background workload
Tableau achieves higher SLA-aware (50ms) throughput than other schedulers.
Tableau Results in Higher Mean Latencies

Hard-capped VMs under Tableau incur higher mean latencies.
Tableau Results in Higher Mean Latencies

Observed Throughput (requests per second)

Capped VMs, 1K files, I/O background workload
Tableau Results in Higher Mean Latencies

Capped VMs, 1K files, I/O background workload

Tableau incurs **higher mean latencies** due to rigid table-based scheduling.
Tableau Results in Higher Mean Latencies

Rigidity becomes advantageous at higher request rates.

Observed Throughput (requests per second)

Capped VMs, 1K files, I/O background workload
Tableau incurs lower runtime overheads compared to the other evaluated Xen schedulers.

Tableau enables accurate control over scheduling latency.

Tableau achieves higher SLA-aware application throughput.

Hard capped VMs under Tableau incur higher mean latency, but entirely controllable.
This Talk

- Tableau

- Evaluation

- Conclusion
Contributions

Tableau

An unorthodox scheduling approach tailored for high-density public clouds.

**Efficient**
Incurs low overheads

**Predictable**
Accurate control over scheduling latency.

**High-throughput**
Provides high SLA-aware throughput.
Thanks!

Source-code available at:
http://tableau.mpi-sws.org/