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?

Google Cloud Platform

Rackspace

Microsoft Azure

Amazon Web Services

IBM Cloud

Competitive market driving datacenter efficiency
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

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

Lower is better

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

- **Observed Throughput (requests per second)**
- **99th Percentile Latency (ms)**

*Lower is better*

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

Credit

Tableau

RTDS

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

- Requesting random 100K-sized files, with I/O background workload

Credit: Tableau, RTDS, Credit2
Case Study: VM Scheduling in Xen

Credit has increasing latency tails.

Credit provides much higher throughput.

Requested 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

- Credit
- Tableau
- RTDS

99th Percentile Latency (ms)

Observed Throughput (requests per second)

Requesting random 100K-sized files, with I/O background workload

This paper
Contributions

Tableau
An unorthodox scheduling approach
tailed 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.

Low overheads
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

The Tableau Approach

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

**Requirement 2**
Guarantee utilization and scheduling latency
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

![Diagram](attachment:image.png)

- Table-Driven Dispatcher
- Semi-Offline Table Planner

Apply scheduling theory from **hard real-time systems**.
The Tableau Approach

Requirement 1
As invisible as possible.
Fast, Low overhead

Table-Driven Dispatcher
Mechanism

Requirement 2
Guarantee utilization and scheduling latency

Semi-Offline Table Planner
Policy
The Tableau Approach

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

**Requirement 2**
Guarantee utilization and scheduling latency

---

**Table-Driven Dispatcher**

**Mechanism**

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

- **Semi-Offline Table Planner**
  - Policy
  - 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
  - Each configured with a utilization and max. scheduling latency.
  - Model each VM as a periodic task\(^1\).

---

Generating Tables Quickly

Set of VMs

Each configured with a \textit{utilization} and \textit{max. scheduling latency}.

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

Partitioning

Apply recent scheduling theory from hard real-time systems.

Scheduling Table

Generating Tables Quickly

Set of VMs

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)**
Worst-case scheduling latency

Budget (C)

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

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

VM (vCPU)

Periodic Task

Budget (C)
Period (P)

Worst-case scheduling latency

Modeling VMs as Periodic Tasks
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)**

The formula for worst-case scheduling latency is:

$$2 \times (P - C)$$
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

Time (in seconds)

Number of VMs

0
20
40
60
80
100
120
140
160
180

0
0.5
1
1.5
2

All V0V 1mV
All V0V 5mV
All V0V 30mV
All V0V 100mV
Table-Generation Times

Lower is better
Table-Generation Times

Number of VMs

Time (in seconds)

0.0 0.5 1.0 1.5 2.0

- All VMs 1ms
- All VMs 5ms
- All VMs 30ms
- All VMs 100ms

Number of VMs
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)
Peak Throughput

Credit
Tableau
RTDS

99th Percentile Latency (ms)

Observed Throughput (requests per second)

VMs Capped at 25%, 100K files, I/O background workload
Peak Throughput

- 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**.

**Observed Throughput (requests per second):**

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

VMs Capped at 25%, 100K files, I/O background workload
RTDS provides controlled latencies but sacrifices throughput.

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

Credit
Tableau
RTDS

99th Percentile Latency (ms)

Observed Throughput (requests per second)

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

VMs Capped at 25%, 100K files, I/O background workload
Tableau Results in Higher Mean Latencies

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

Credit
Tableau
RTDS

Observed Throughput (requests per second)

Mean Latency (ms)

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

Tableau incurs higher mean latencies due to rigid table-based scheduling.

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

Rigidity becomes advantageous at higher request rates.

Capped VMs, 1K files, I/O background workload
Summary of Results

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/
Scheduling Overheads on 48-Core Server

<table>
<thead>
<tr>
<th></th>
<th>Credit</th>
<th>RTDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedule</td>
<td>16.40</td>
<td>4.39</td>
</tr>
<tr>
<td>Wakeup</td>
<td>7.07</td>
<td>19.16</td>
</tr>
<tr>
<td>Migrate</td>
<td>0.42</td>
<td>168.62</td>
</tr>
</tbody>
</table>

Overheads (in µs) of key scheduler operations on a 48-core server.
Model each VM as a periodic task.

Partitioning

Simulate EDF for each core.
Repeating scheduling table for each core.

Postprocessing
Coalescing small slots. Generating indices for fast lookup. Extensible design in Python.

Scheduling Table
Table Sizes

<table>
<thead>
<tr>
<th>Table Size (in MiB)</th>
<th>Number of VMs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.50</td>
<td>1</td>
</tr>
<tr>
<td>1.00</td>
<td>10</td>
</tr>
<tr>
<td>1.50</td>
<td>1.5</td>
</tr>
<tr>
<td>2.00</td>
<td>2</td>
</tr>
</tbody>
</table>

- **All VMs 1ms**
- **All VMs 5ms**
- **All VMs 30ms**
- **All VMs 100ms**
SLA-Aware Throughput (Uncapped Scenario)

Uncapped VMs, 100K files, I/O background workload
Uncapped VMs, 100K files, I/O background workload

Tableau achieves almost 1.6x peak throughput compared to Credit2
SLA-Aware Throughput (Uncapped Scenario)

Uncapped VMs, 100K files, I/O background workload

100ms SLA-aware throughput under Tableau is significantly higher.
SLA-Aware Throughput (Uncapped Scenario)

Credit is unable to maintain a 100ms latency SLA at 100 req/sec.

Uncapped VMs, 100K files, I/O background workload
Partitioning & Semi-Partitioning

**Partitioning**
- Assign VMs to **individual cores** using bin-packing heuristic.

**Semi-Partitioning**
- Split any VMs that couldn't be assigned to multiple cores.

**Optimal Scheduling**
- Guaranteed to find a **schedule**. Results in many preemptions.

*Included for completeness, but unnecessary in practice.*
Modelling VMs as Periodic Tasks

VM (vCPU)

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

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

Periodic Task

**Budget (C)**

**Period (T)**

\[
2 \times (T - C) = 2 \times (1 - U) \times T
\]

Pick any T such that \(2 \times (1 - U) \times T \leq L\)

Schedule repeats after hyperperiod (common multiple of all task periods)

Choosing T indiscriminately can result in exponential hyperperiod.
Modelling VMs as Periodic Tasks

VM (vCPU)

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

Max. Latency \((L)\)
An upper bound on scheduling delay.

Periodic Task

Budget \((C)\)

Period \((T)\)

\[2 \times (T - C) = 2 \times (1 - U) \times T\]

Pick largest \(T \in F\)

\[2 \times (1 - U) \times T \leq L\]

(F is the set of all integer divisors of 102,702,600)

Pick periods from a set of candidate periods with a known hyperperiod (102,702,600 ns = \(~102\)ms) to ensure bounded table length.
Summary of Results

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**.
## Scheduler Overheads

<table>
<thead>
<tr>
<th></th>
<th>Credit</th>
<th>RTDS</th>
<th>Tableau</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Schedule</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Wakeup</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Migrate</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Overheads (in µs) of key scheduler operations on 16-core server.*
Scheduler Overheads

Overheads (in µs) of key scheduler operations on 16-core server.

- Picking the next VM to schedule
- Unblocking a VM
- Migrating de-scheduled VM to idle core.
## Scheduler Overheads

<table>
<thead>
<tr>
<th></th>
<th>Credit</th>
<th>RTDS</th>
<th>Tableau</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedule</td>
<td>8.08</td>
<td>2.86</td>
<td>1.43</td>
</tr>
<tr>
<td>Wakeup</td>
<td>2.12</td>
<td>3.90</td>
<td>1.06</td>
</tr>
<tr>
<td>Migrate</td>
<td>0.32</td>
<td>9.42</td>
<td>0.43</td>
</tr>
</tbody>
</table>

Overheads (in μs) of key scheduler operations on 16-core server.
## Scheduler Overheads

**Heuristics and decision-making.**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Credit</th>
<th>RTDS</th>
<th>Tableau</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedule</td>
<td>8.08</td>
<td>3.51</td>
<td>2.86</td>
</tr>
<tr>
<td>Wakeup</td>
<td>2.12</td>
<td>5.19</td>
<td>3.90</td>
</tr>
<tr>
<td>Migrate</td>
<td>0.32</td>
<td>5.55</td>
<td>9.42</td>
</tr>
</tbody>
</table>

*Overheads (in µs) of key scheduler operations on 16-core server.*

*Poor scalability*
Heuristics and decision-making.

<table>
<thead>
<tr>
<th></th>
<th>RTDS</th>
<th>Tableau</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedule</td>
<td>4.39</td>
<td>1.43</td>
</tr>
<tr>
<td>Wakeup</td>
<td>19.16</td>
<td>1.06</td>
</tr>
<tr>
<td>Migrate</td>
<td>168.62</td>
<td>0.43</td>
</tr>
</tbody>
</table>

**RTDS Overheads on 48-core server**

Poor scalability
Scheduler Overheads

<table>
<thead>
<tr>
<th>Operation</th>
<th>Credit</th>
<th>RTDS</th>
<th>Tableau</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedule</td>
<td>8.08</td>
<td>3.51</td>
<td>2.86</td>
</tr>
<tr>
<td>Wakeup</td>
<td>2.12</td>
<td>5.19</td>
<td>3.90</td>
</tr>
<tr>
<td>Migrate</td>
<td>0.32</td>
<td>5.55</td>
<td>9.42</td>
</tr>
</tbody>
</table>

Heuristics and decision-making.

Overheads (in µs) of key scheduler operations on 16-core server.

Poor scalability
# Scheduler Overheads

Overheads (in µs) of key scheduler operations on 16-core server.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Credit</th>
<th>RTDS</th>
<th>Tableau</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedule</td>
<td>8.08</td>
<td>2.86</td>
<td>1.43</td>
</tr>
<tr>
<td>Wakeup</td>
<td>2.12</td>
<td>3.90</td>
<td>1.06</td>
</tr>
<tr>
<td>Migrate</td>
<td>0.32</td>
<td>9.42</td>
<td>0.43</td>
</tr>
</tbody>
</table>

**Significant reduction in runtime overheads**
## Scheduler Overheads

<table>
<thead>
<tr>
<th></th>
<th>Credit</th>
<th>RTDS</th>
<th>Tableau</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedule</td>
<td>8.08</td>
<td>2.86</td>
<td>1.43</td>
</tr>
<tr>
<td>Wakeup</td>
<td>2.12</td>
<td>3.90</td>
<td>1.06</td>
</tr>
<tr>
<td>Migrate</td>
<td>0.32</td>
<td>9.42</td>
<td>0.43</td>
</tr>
</tbody>
</table>

*Overheads (in µs) of key scheduler operations on 16-core server.*

- Significant reduction in runtime overheads
- Inherently scalable
Summary of Results

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**.
Scheduling Latency

Maximum observed ping latency (ms)

Lower is better

VMs Capped at 25%

- Credit
- RTDS
- Tableau
Scheduling Latency

With idle background, predictable scheduling delays.

VMs Capped at 25%
Maximum observed ping latency (ms)

With a I/O or CPU background, Credit's tail latency increases.
Scheduling Latency

With an I/O or CPU background, RTDS, and Tableau continue to have predictable scheduling delays.

*VMs Capped at 25%*
Limitations

Tableau incurs **higher mean latencies** for low throughputs with hard-capped VMs.

Table-generation **increases VM startup and teardown times**.
Dealing with Table-Generation Time

- Cache pre-generated tables
- Pre-generate fixed-utilization slots
- Generate tables on an external (faster) server
This Talk

- Tableau
- Evaluation
- Limitations
- Conclusion
Limitations

Tableau incurs **higher mean latencies** for low throughputs with hard-capped VMs.

Table-generation **increases VM startup and teardown times.**
Limitations

Tableau incurs **higher mean latencies** for low throughputs with hard-capped VMs.

Table-generation **increases VM startup and teardown times.**
Tableau Results in Higher Mean Latencies

Observed Throughput (requests per second)

Mean Latency (ms)

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

Tableau incurs **higher mean latencies** due to rigid table-based scheduling.

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

Observed Throughput (requests per second)

Mean Latency (ms)

Credit becomes advantageous at higher request rates.

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