



Xen Performance Monitoring

Team: Rob Gardner (HP Labs)
Lucy Cherkasova (HP Labs)
Diwaker Gupta (UCSD)



Motivation

- Open question: which CPU scheduling and resource allocation policies provide best service in a given circumstance?
- We need a performance tool capable of enabling a thorough evaluation of different CPU scheduling algorithms

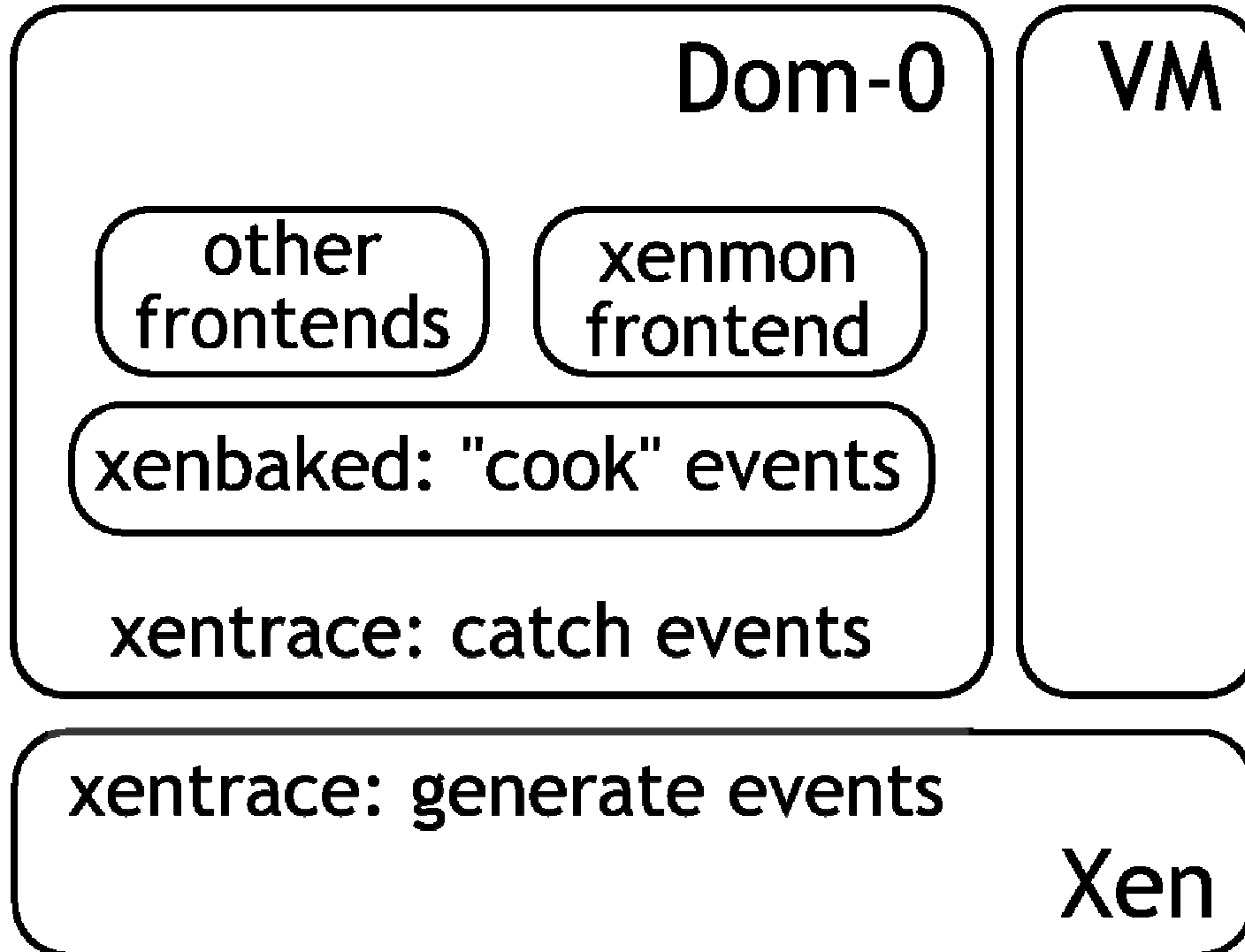
- A *light-weight, non-intrusive monitoring system* that accounts for:
 - *CPU usage* by different guest VMs;
 - *Waiting time*: amount of time a domain spends waiting run on the CPU;
 - *Blocked time*: the amount of time a domain spends blocked (or sleeping);
 - *I/O count*: the number of memory page exchanges between Dom0 (driver domain) and guest domains. This metric helps to understand the I/O communication patterns;
 - *Execution count*: the number of times a domain was scheduled to run.
- These metrics are reported at *three time scales*: execution periods, 1 sec, 10 sec.

XenMon in detail



- The XenMon infrastructure consists of various components:
 - *Xen Trace Facility* in the hypervisor;
 - *Additional QoS Trace calls* in the hypervisor;
 - *xenbaked*: a user process that takes raw event data and accumulates it into a more usable form;
 - *xenmon*: Python-based program that processes accumulated data into periodic metrics and displays them or logs them;
 - *Potential future interfaces*: Network access (via RPC) to processed data, etc.

XenMon Architecture



Xen Trace Facility

The TRACE procedure/macro is called at any time to create a new trace record.

...

```
xen_do_something_cool(stuff);  
TRACE(COOL_EVENT, dom_id);
```

...

Anatomy of a Trace Record

Each trace record contains:

- Event Identifier
- CPU number where code is executing
- Time stamp
- Optional parameters

```
TRACE(TRC_SCHED_SWITCH_INFNEXT, domid, wait_time, alloc_time)
```

Means that *domid* is about to start running; It had to wait on the run queue for *wait_time* and is scheduled to run for *alloc_time*.

Our Trace Events

- TRC_SCHED_SWITCH_INFNEXT : some domain just started running
- TRC_SCHED_SWITCH_INFPREV : some domain just stopped running
- TRC_SCHED_SLEEP : some domain blocked
- TRC_SCHED_WAKE : some domain unblocked
- TRC_MEM_PAGE_FLIP : memory page exchange happened
- A few other infrequently used ones for domain creating, destruction, etc.

xenbaked

- Background user process that “cooks” the raw trace data into an easily digestible form
- For instance, domain blocked time can be computed by observing a sequence of trace events:
 - -SLEEP event for dom X
 - -arbitrary sequence of other events
 - -WAKE event for dom X

Periodic Accumulation

- Xenbaked accumulates each metric every $1/10^{\text{th}}$ second (configurable via command line option)
- So if a domain is blocked for $1/2$ second, for example, we have a sequence of blocked times:
 - ...0, 0, 0, 100ms, 100, 100, 100, 100, 0, 0, ...
 - Each number represents time that the domain was blocked during a $1/10^{\text{th}}$ second period

Periodic Accumulation

- Enough data is kept to create a 10 second history
- Periodic accumulation is done for a number of metrics:
 - CPU time gotten
 - Time blocked
 - Time waiting
 - Time allocated
 - Execution count
 - I/O count

Periodic Accumulation

- Picture a circular buffer of 100 data points for each of the metrics:
- Array Datapoints[100] of {
 - Cpu time gotten
 - Time blocked
 - Time waiting
 - Time allocated
 - Execution count
 - I/O count}

Periodic Accumulation

- Now duplicate all that for each domain:
- Array Domains[NDOMAIN] of {
 - Array Datapoints[100] of {
 - CPU time gotten
 - Time blocked
 - Time waiting
 - Time allocated
 - Execution count
 - I/O count

Periodic Accumulation

- And duplicate again for each *CPU*:
- Array CPUs[ncpu] of {
 - Array Domains[NDOMAIN] of {
 - Array Datapoints[100] of {
 - CPU time gotten
 - Time blocked
 - Time waiting
 - Time allocated
 - Execution count
 - I/O count

Data Sharing

- That huge data structure is kept in a shared memory segment
- This makes it accessible to other applications
- For instance, an app could be written to display the data...

XenMon Screen Shot



The screenshot displays the XenMon tool interface. The main window shows system statistics for the last 10 seconds and the last 1 second, categorized by CPU ID (0, 1, 2, 7). The statistics include CPU usage, execution time, blocked time, waited time, execution count, and I/O count. Below the statistics, it shows the total records lost (302 for the last 10 seconds, 62 for the last 1 second).

Last 10 seconds				Last 1 second			
0	860.22 ms	76.94%	736.33 us/ex	838.44 ms	82.33%	822.00 us/ex	Gotten
0			44.64 ms/ex			47.97 ms/ex	Allocated
0	0.00 ns	0.00%	0.00 ns/io	0.00 ns	0.00%	0.00 ns/io	Blocked
0	256.30 ms	22.93%	219.39 us/ex	181.45 ms	17.82%	177.90 us/ex	Waited
0			1168/s			1020	Execution count
0	23957/s		20/ex	20406		20.01/ex	I/O Count
<hr/>							
1	86.34 ms	7.72%	104.82 us/ex	79.24 ms	7.78%	101.99 us/ex	Gotten
1			5.00 ms/ex			5.00 ms/ex	Allocated
1	1.01 s	90.26%	57.43 us/io	922.43 ms	90.58%	55.83 us/io	Blocked
1	22.97 ms	2.05%	27.88 us/ex	17.07 ms	1.68%	21.97 us/ex	Waited
1			823/s			777	Execution count
1	17570/s		21/ex	16522		21.26/ex	I/O Count
<hr/>							
2	171.36 ms	15.33%	443.51 us/ex	100.64 ms	9.88%	396.23 us/ex	Gotten
2			5.00 ms/ex			5.00 ms/ex	Allocated
2	753.20 ms	67.37%	117.92 us/io	794.45 ms	78.01%	204.55 us/io	Blocked
2	193.62 ms	17.32%	501.12 us/ex	123.42 ms	12.12%	485.90 us/ex	Waited
2			386/s			254	Execution count
2	6387/s		16/ex	3884		15.29/ex	I/O Count
<hr/>							
7	0.00 ns	0.00%	0.00 ns/ex	0.00 ns	0.00%	0.00 ns/ex	Gotten
7			0.00 ns/ex			0.00 ns/ex	Allocated
7	0.00 ns	0.00%	0.00 ns/io	0.00 ns	0.00%	0.00 ns/io	Blocked
7	0.00 ns	0.00%	0.00 ns/ex	0.00 ns	0.00%	0.00 ns/ex	Waited
7			0/s			0	Execution count
7	0/s		0/ex	0		0.00/ex	I/O Count

Records lost: 302 (Min: 0, Max: 16) Records lost: 62 (Min: 0, Max 16)

The bottom part of the screenshot shows three shell console windows. The top one shows a user logging in and running a command: `time dd if=/dev/sda1 of=/dev/null bs=1024k`. The middle one shows a list of CPU usage statistics for various processors. The bottom one shows a user logging in via SSH and running a command: `ps -ef`.

XenMon Status

- Released to the Xen community
- Accepted into Xen 3.0
- Runs on x86, x86-64 and Itanium (IA64)
- Performance case study using XenMon is available as HP Labs Report HPL-2005-187
<http://www.hpl.hp.com/techreports/2005/HPL-2005-187.html>
- Future Enhancements
 - A more reliable trace buffer facility with flow control
 - More metrics
 - Optimized display script – current python implementation sucks up some cpu time
 - Graphical display front-end