New approach to optimization of block-level data processing algorithm in cloud-environment

Central & Eastern European
Software Engineering Conference in Russia, 2011

Aleksandr Povaliaev

Senior Software Engineer, EMC

EMC² where information lives®

Cloud for end-users benefits

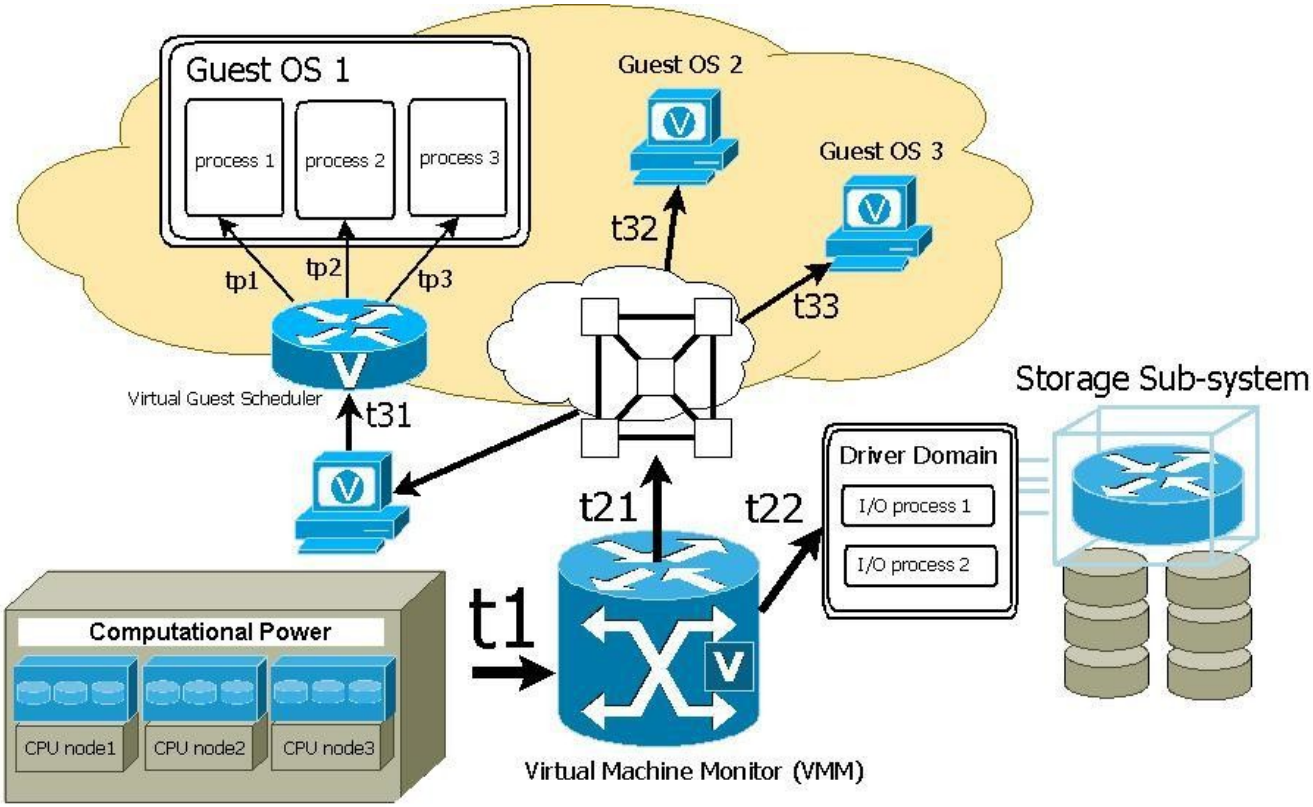
- Low-cost of maintenance
- Easy to deploy
- Scalability
- Resources-on-demand
- Hi-availability
- Security
- Wide range of services

EMC² where information lives

Challenges to system engineers

- Virtualization (multi-layer software stack)
- Varying load
- Guarantee of Level Of Service
- Performance Isolation
- Inter-process resource contention
- Resource allocation errors
- Efficient resources utilization

Optimization through scheduling



EMC² where information lives

High-loaded I/O-bound environment

Workload	CPU (%)	Event (events/sec)	Switch (switches/sec)	Pages Exchange (pages/sec)	I/O Execution (pages/exe)	Driver Domain (Dom0)				Guest Domain (VM1)			
						CPU (%)	Waiting	Block	Execution (exe/sec)	CPU (%)	Waiting (%)	Block (%)	Execution (exe/sec)
4 KB	97.46	242216	10525	14900	6.38	46.82	4.77	10.68	2335	50.65	37.72	5.65	2509
50 KB	49.62	342584	26981	11000	3.54	34.57	1.22	19.33	3108	15.05	1.14	36.76	3915
100 KB	44.40	332951	27720	10000	3.17	32.19	0.77	21.04	3150	12.21	1.01	39.08	3962

*

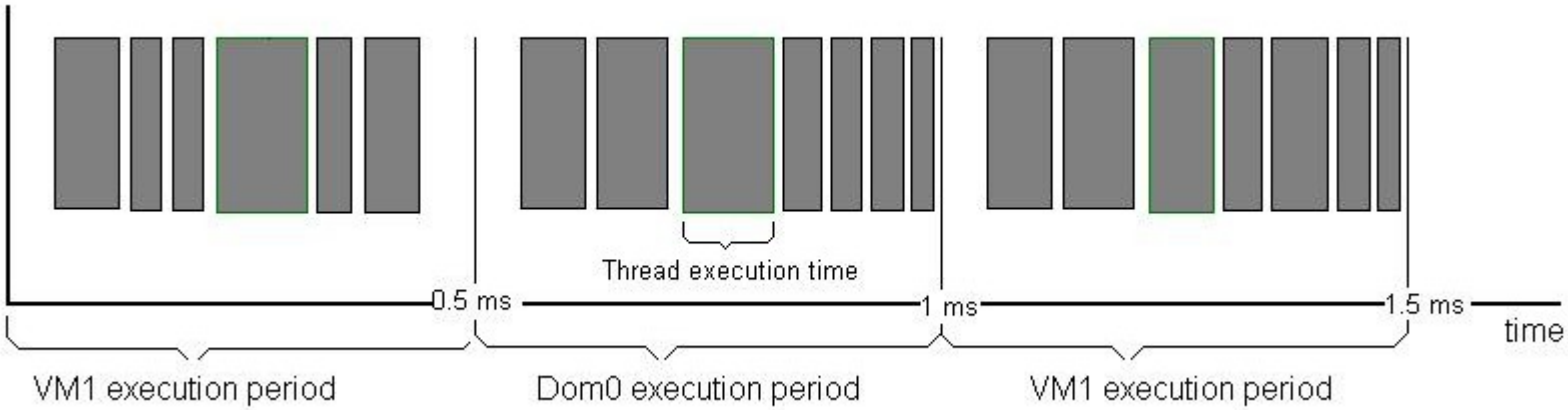
-

VMM time granularity 0. 5 ms (0.35 ms for 100 KB)

* The data has been obtained on IBM ThinkCentre A52 Workstation with two 3.2GHz Intel Pentium 4 CPUs (16KB L1 caches, 2MB L2 caches), 2 GB 400 MHz DDR memory, Hyper-threading, no VT-x / Xing Pu, Ling Liu, Yiduo Mei, Sankaran Sivathanu, Younggyun Koh, Calton Pu, "Understanding Performance Interference of I/O Workload in Virtualized Cloud Environments," Cloud Computing, IEEE International Conference on, pp. 51-58, 2010 IEEE 3rd International Conference on Cloud Computing, 2010

EMC² where information lives

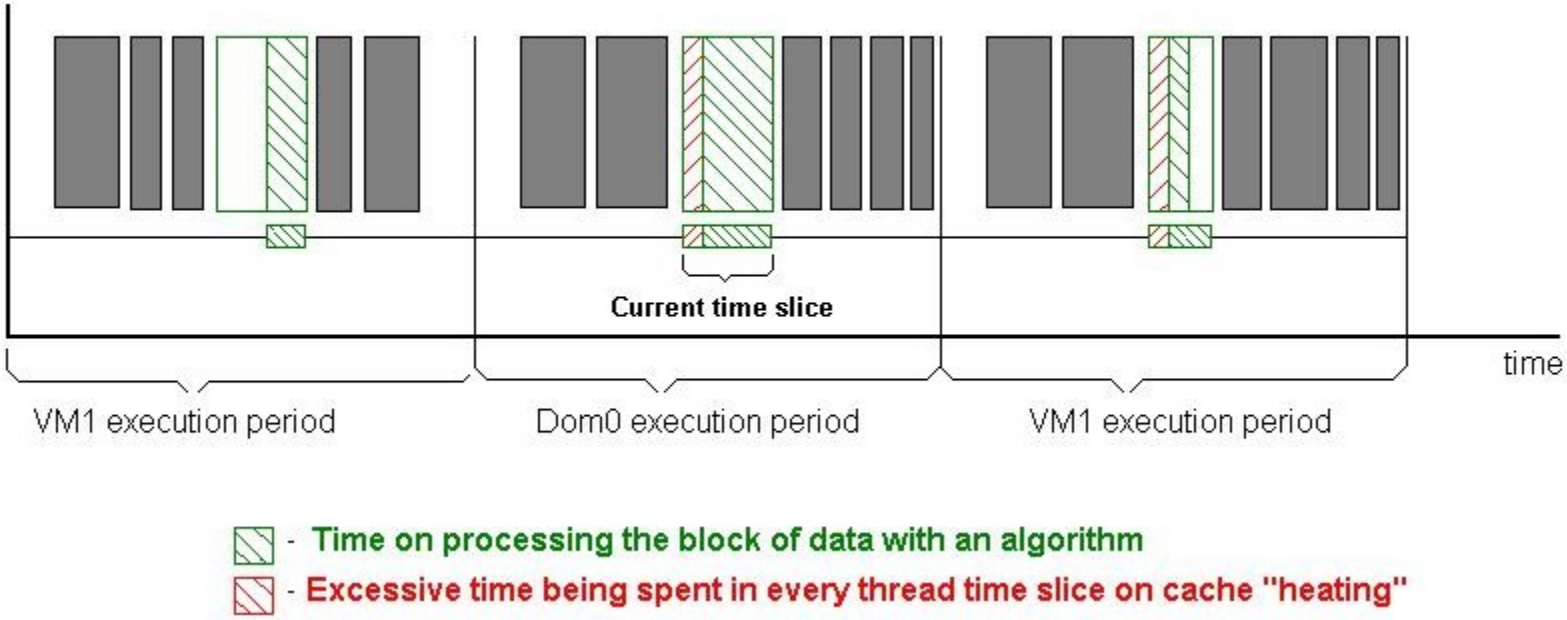
How scheduler sees a thread



… knows the state of the thread and its time slice

EMC² where information lives

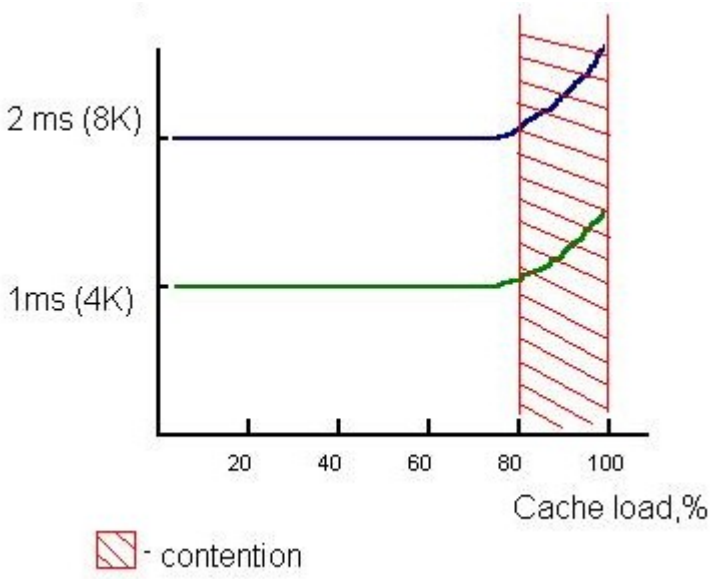
How a software engineer sees the thread



… he knows what algorithm is currently working and the size of the data block being processed

EMC² where information lives

Algorithm in high-loaded environment



- A set of instructions
- API isn't used
- Data access pattern is known
- Experience cache "heating" time
- Time of processing a single block (vs. when executing in a single-user mode) is high

EMC² where information lives

Use-case 1 (4K block encryption)

	Key length 128 bits		Key length 256 bits		*
	AVG	SD	AVG	SD	
XTS	0.0008	0.00012	0.0009	0.00012	

- Data access pattern – random
- Cache “heating” time might be high
- Operating time is almost independent on the input data
- Can be split into stages (state-machine)
- It’s known when the processing is almost completed (95%)

* The data is based on the following paper:
Mohamed Abo El - Fotouh and Klaus Diepold. 2008. A New Narrow Block Mode of Operations for Disk Encryption. In *Proceedings of the 2008 The Fourth International Conference on Information Assurance and Security* (IAS '08). IEEE Computer Society, Washington, DC, USA, 126-131.

EMC² where information lives

Use-case 2 (4K block compression)

	Compression	Decompression
lzo	0.000046	0.000014
zlib	0.000150	0.000060

*

- Data access pattern – random
- Not in-place data processing
- Cache “heating” time might be high
- Operating time is less predictable
- Can be split into stages (state-machine)
- Progress can be estimated (compression)

•The data is based on the following paper:
Using Transparent Compression to Improve SSD-based I/O Caches. Thanos Makatos, Yiannis Klonatos, Michail Flouris, Manolis Marazakis, and Angelos Bilas. Eurosys
2010 Conf.. Paris, France. April 13-16, 2010.

Task

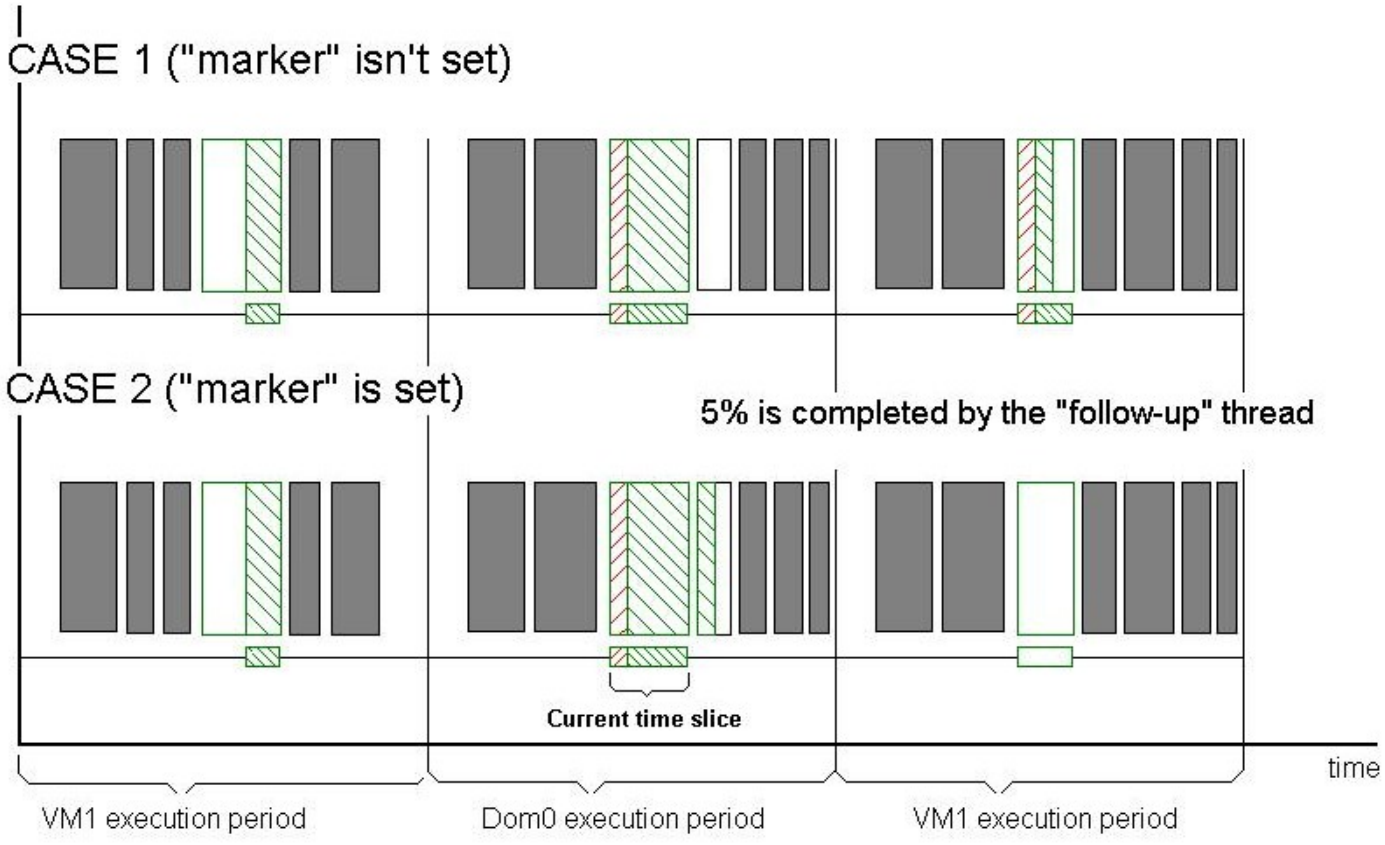
- I/O bound environment
- A thread implementing the algorithm
- The thread is processing blocks of data
- Improve performance
- To complete the last 5% of the block processing, the algorithm with memory random access pattern might require some time on cache “heating”
- Reduce cache “heating” penalties
- “Enlarge” the current time slice when 95% is hit to make the processing of the current block completed before other threads overwrite cache

Solution

- A thread is running the algorithm which processes blocks of data
- State-machine implementation of the algorithm
- Store values to know the state of the current block processing (10%, 50%, 70% completed, …)
- Introduce a “follow-up” thread
- The “follow-up” thread is to use its time slice to complete the current block processing (when 95% is hit)
- A “marker” to know when “follow-up” can use its time slice to complete the current block processing
- Lock-free implementation
- No use of API is needed

EMC² where information lives

How it works



- Time on processing the block of data with an algorithm
- Excessive time being spent in every thread time slice on cache "heating"

EMC² where information lives®

Benefits

- Cache consumption is reduced overall
- The algorithm performance is improved
- No direct influence on other threads
- Data-center performance is improved

EMC² where information lives

To Do

- “Marker” for concurrent data processing algorithms
- Most of the high-productive algorithms should still be rethought
- A general approach to be developed

EMC²
where information lives®

Thank you!

EMC²®

where information lives®