





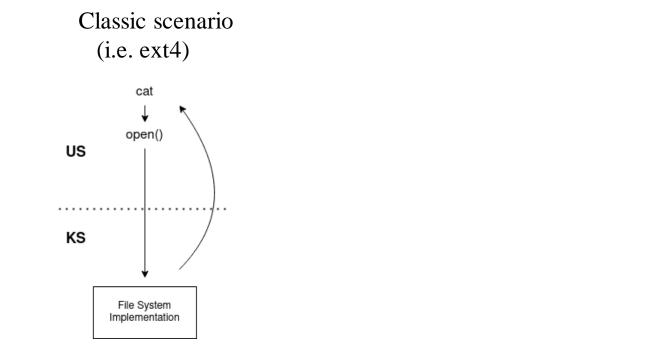
CernVM-FS Profiling

Razvan-Nicolae Virtan Supervisors: Radu Popescu, Jakob Blomer

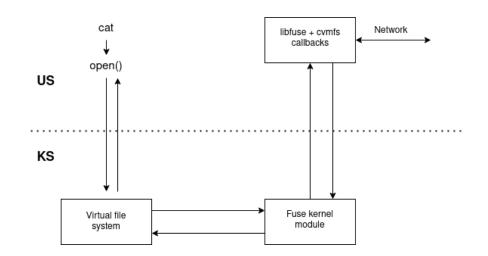
About CernVM-FS (CVMFS)

- created and optimized to deliver scientific software stacks to a distributed compute infrastructure
- offers a file system interface for software repositories

[razvan@~]\$ cat/cvmfs/atlas.cern.ch/repo/test # new cvmfs process created on local system







Why profiling?

Benchmark scenario:

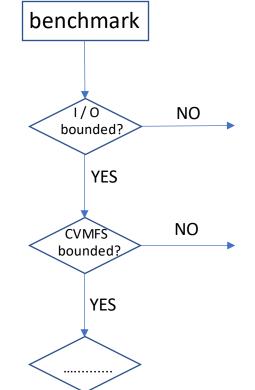
- Multi core system
- Cold Cache
- Multiple processes, attempting to access different data from the same repo (e. g. different jobs using different software versions)

Performance on a synthetic benchmark:

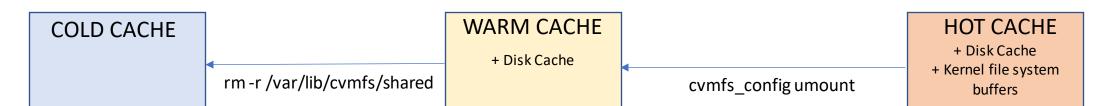
CVMFS	Local filesystem
1.38 GB for 1 min	3.97 GB for 1 min

CVMFS Profiling Goals

- 1. Develop a general set of tools & procedures for analyzing CVMFS performance
- 2. Apply these tools on some known benchmarks and spot possible bottlenecks



I / O bounded? New tool: avg_cache_time.sh



\$./profiling_tools/avg_cache_time.sh --rounds 2 ./tensorflow_benchmark.sh

Cache_Type	Real Avg (s)	User Avg(s)	Sys Avg(s)
cold	25.560	10.639	1.282
warm	12.924	10.359	1.183
hot	11.690	10.051	0.959
Cache_Type	CPU	BLOCKED	
cold	0.399	0.601	
warm	0.893	0.107	
hot	0.942	0.058	
Compare Case	Real Time	User Time	Sys Time
cold / hot	2.192	1.058	1.337
cold / warm	1.992	1.027	1.082
warm / hot	1.106	1.031	1.233

CVMFS bounded? cvmfs_talk

- How much of the blocked time is actually spent in CVMFS?
- Newly added counter to the set of cvmfs internal profiling counters
- Measure time spent in cvmfs callbacks and calculate the total

 $T_{Callbacks}$ / $T_{Blocked}$

\$ cvmfs_talk -i unpacked.cern.ch internal affairs

••••

Total Time In Callbacks = 34945ms

••••

Where is the bottleneck? New tool: generate_flamegraphs.sh

• Method 1: exhaustive ON / OFF CPU analysis using flamegraphs

\$./profiling_tools/generate_flamegraphs.sh --oncpu --dwarf --benchmark lhcb_benchmark.sh --cache hot

						do cpu cpu		
		cvmfs2-52886/52902	cvmfs2-52886/52903	vmfs2-52886/52987	cvmfs2-52886/52992	cpu		£
		clone3				asm		
		start thread				CO	++	- E
		fuse exit				ir.		ā -
		fuse_lowlevel_notify_st	ore			sof		
		fuse_reply_open				net		
	I	loader::stub_open				na		
	swapper	cvmfs::cvmfs_open				e10		
swapper-0/0	secondar	cvmfs::Fetcher::Fetch				napi	Ш	
secondary_startup_64_no_verify	start_ker	download::DownloadM	anager::Fetch			netif		
cpu_startup_entry		WritePipe				ne		:49
do_idle		GIwrite				ne		ası
cpuidle_idle_call		entry_SYSCALL_64_aft	er_hwirame			ip_loc	L.	sys
cpuidle_enter cpuidle_enter_state		do_syscall_64				ip_pro		
sm sysvec apic timer interrupt		ksys_write vfs_write				tcp_v4 tcp_v.	+ +	hrt
swec apic timer interrupt		new sync write				tcp_v	+ +	hrt
sysvec_apic_timer_interrupt		pipe_write				sock d		try
rtimer interrupt		wake up common lo	ick			JOCK_U.	i.,	ttv
hrtimer run queues		wake up common					i	en
rtimer wakeup		pollwake					i.,	en
y to wake up							xfs.	. en
wu do activate							n	1
nqueue task							vf	
nqueue_task_fair							ks	
nqueue_entity							do	5
update_stats_enqueue_sleeper								. scl
update_stats_enqueue_sleeper								scl
schedule								. do
schedule								. CO
chedule								do
chedule_hrtimeout_range_clock								>
p_poll.constprop.0								do
p_sys_poll x64 sys poll								. en
o syscall 64								<u>Sa</u>
ntry SYSCALL 64 after hwframe								Ja do
libc poll								. do
wnload::DownloadManager::MainDow	mload						our.	. 40
art thread	- Invuu							
clone3								

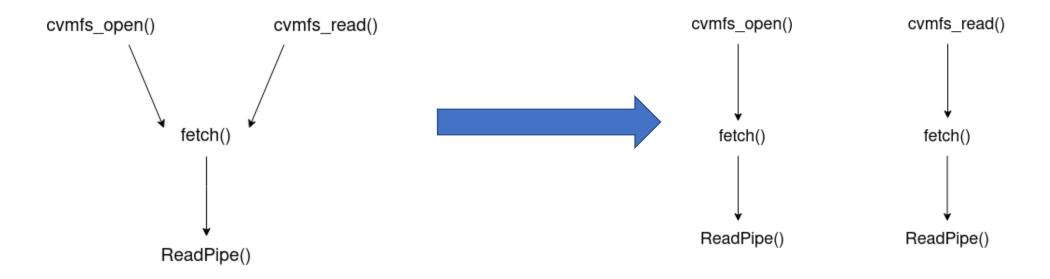
Bp- 0.0 0.0 Bp- 0.0 0	eset Zoom	Flame Graph	
ip jp jp <td< th=""><th></th><th></th><th></th></td<>			
ip jp jp <td< td=""><td></td><td></td><td></td></td<>			
ip jp jp <td< td=""><td></td><td></td><td></td></td<>			
ip jp jp <td< td=""><td></td><td></td><td></td></td<>			
ip jp jp <td< td=""><td></td><td></td><td></td></td<>			
ip jp jp <td< td=""><td></td><td></td><td></td></td<>			
ip jp jp <td< td=""><td></td><td></td><td></td></td<>			
ip jp jp <td< td=""><td></td><td></td><td></td></td<>			
ip jp jp <td< td=""><td></td><td></td><td></td></td<>			
ip jp jp <td< td=""><td></td><td></td><td></td></td<>			
ip iom iom iom ie			
ip. i			
iom.			
iom.			
ip. iom.			
iom.			
iom.			
iom.		Dag.	ie.
ionap apply ionap apply ionap file but xf. ionap file but xf. ionap file but xf. ic. ionap file but xf. ionap file but xf. ionap file but xf. ionap file but xf. ic. ic. ic. ic. ic. ip. ic. ic. ic. ic. ip. ip. ic. ic. ic. ic. ic. ip. ic. ic. ic. ic. ic. ic. ic. ic. ic. ic. ic. ic. ic. ic. ic. ic.			
top top and the but std top additional top base top additional top base top base <td></td> <td></td> <td></td>			
top			
i icon icon <t< td=""><td>- 1 I I I I I I I I I I I I I I I I I I</td><td></td><td></td></t<>	- 1 I I I I I I I I I I I I I I I I I I		
icp s icp s icp s icp revmsg sock s icp sys icp revmsg icp revmsg sock s icp revmsg icp revmsg icp revmsg sock s icp revmsg icp revmsg icp revmsg sock s icp revmsg icp revmsg icp revmsg sched icp revmsg icp revmsg icp revmsg sched icp exit icp revmsg icp revmsg icp revmsg sched icp exit icp revmsg icp revmsg icp revmsg sched icp exit icp revmsg icp revmsg icp revmsg sched icp exit icp revmsg icp revmsg icp revmsg sched icp exit icp revmsg icp revmsg icp revmsg icp onloce icp exit icp revmsg icp revmsg icp revmsg icp onloce icp exit icp revmsg icp revmsg icp revmsg icp onloce icp exit icp revmsg icp revmsg icp revmsg icp onloce icp onloce icp revmsg icp revmsg icp onloce icm revmsg icp revmsg icp revmsg <td>i iii</td> <td>tc</td> <td>ip_q</td>	i iii	tc	ip_q
sock s sock s do syscall 64 inct precynag sock s sys entry SYSCALL 64 al. inct precynag f x64 GL write sys inct precynag f x64 GL write x64 sys.rect inct precynag sched entry SYSCALE 64 al. sched inct precynag sched entry fubre PosixCacheManager: Write alls inlate fast do sys.edl 64 sched fubre PosixCacheManager: Write alls inlate fast fubre PosixCacheManager: Write alls inlate fast fubre PosixCacheManager: Write alls inlate fast do poll.con fubre SSHA cynfs: TransactionSink: finflate fubre PosixCacheManager: Write alls inlate fast fubre PosixCacheManager: Write alls inlate fast do poll.con fubre SSHA cynfs: TransactionSink: finflate fast fubre PosixCacheManager: Write alls inlate fast fubre PosixCacheManager: Write alls inlate fast do poll.con fubre SSHA cynfs: TransactionSink: finflate fast fubre fubre fubre fubre do poll.con fubre Curl curl wr fubre fubre fubre fubre fubre			
sys entry SYSCALL 64 af. inet recoving x64 CI write sys recvinon scha inet recving sys recvinon inet recving scha entry Index sol infate fast inet recving scha entry Infate fast infate fast infate fast infate fast sched infate infate fast infate fast infate fast infate fast infate fast infate fast infate fast infate fast infate fast infate fast infate fast infate fast infate fast infate fast infate fast infate fast infate fast infate fast infate fast infate fast infate fast infate fast infate fast infate fast infate fast infate fast infate fast infate fast infate fast infate fast infate fast infate fast infate fast infate fast infate infate fast infate fast infate fast infate fast infate fast infate fast infate infate infate infate infate infate fast infate fast <t< td=""><td></td><td></td><td></td></t<>			
x64 GL write x64 Sys reextrom sche in sched		sock s do system 04	
n. no sys.el PosixCacheManager:E. do sys.ell do sys.ell.			
isched ista cvmfs::TransactionSinke: inite entry SYSCALL ischedu ista cvmfs::TransactionSinke: inite ista: ista: ischedu ista cvmfs::TransactionSinke: inite ista: ista: <t< td=""><td>f</td><td>do sys PosixCacheManager::F.</td><td></td></t<>	f	do sys PosixCacheManager::F.	
Image: State Stat		entry [libc PosixCacheManager::Write adl inflate_fast	do_syscall_64
ido poll.con. iiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii			
do sys poll Curl bu Curl cient write Cu Curl read		Curl s., Snas., ZID::Decompress2.Stream2Sink	IDC_recv
x64 sys poll Cur. Curl http Curl readwrite do syscall 64 multi runsingle Imulti socket entry SYSCAL. multi socket Imulti socket libc poll Curl multi socket action Imulti socket download:DownloadManager::MainDownload Imulti socket Imulti socket start thread Cone3 Imulti socket Imulti socket			
entry SYSCAL. multi socket hib poll curl multi socket action download: DownloadManager::MainDownload start thread _clone3 wrfs2-7/42332	x64_sys_poll	Cur. Curl http Curl readwrite	
libc poll curl multi socket action download:DownloadManager::MainDownload start thread clone3 vmfs2-?/42332			
download::DownloadManager::MainDownload Start thread clone3 vmis2-?/42332			N
start.thread clone3 vmfs2-/H2332	download::DownloadMan	in muu socket action ager: MainDownload	
		iger "Main Sownoad	
	clone3		
	vmfs2-?/42332		

Where is the bottleneck? cvmfs_talk

- Flamegraphs are inaccurate / expensive and can fail on multi-threaded scenarios.
- Method 2:
 - add more timers in the CVMFS code, in places that can become bottlenecks
 - generate partial flamegraphs internally, for relevant parts of the CVMFS code

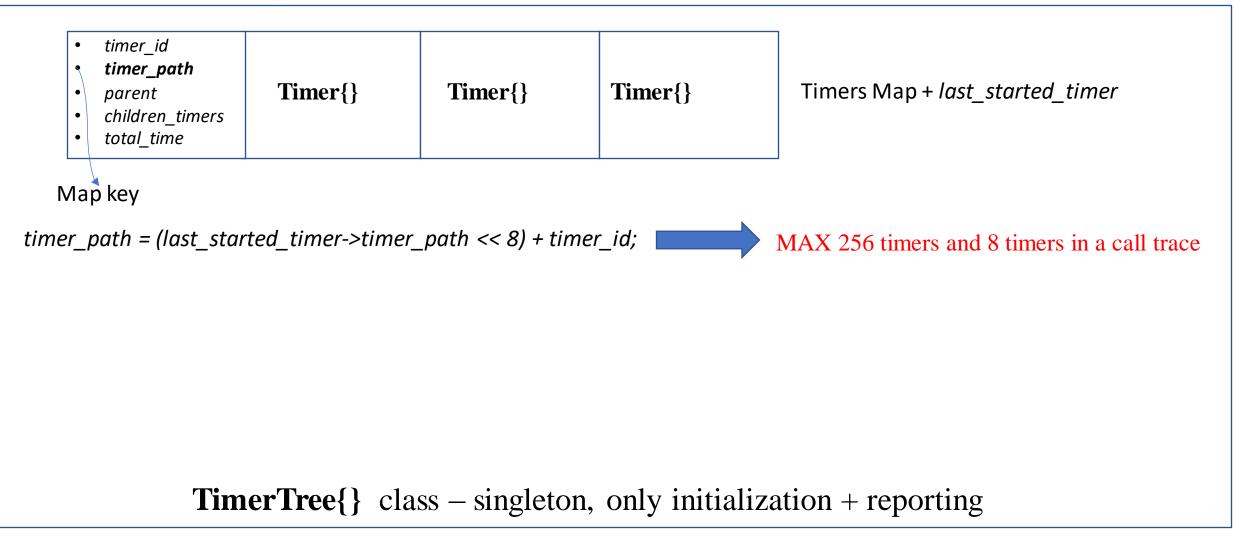
=> New timers interface in CVMFS

Timer Requirements



• The same timer, reached from two different call stacks, needs different records.

Timer implementation



RAII timer interface

• TimerGuard is a wrapper object for the Timer backend

TimerGuard()	~TimerGuard()		
 search / add timer to map record t₀ 	 compute <i>dt</i> add <i>dt</i> to the total time 		

• Some usage examples:

.....

```
cvmfs_open() {
   TimerGuard timer_guard("cvmfs_open()", CVMFS_OPEN_TIMER, ...);
```

```
.....
{
    TimerGuard timer_guard(...);
    retval = DecompressZStream2Sink(...);
    if ( retval == zlib::kStreamDataError ) {
        .....
    }
}
.....
```

Output example

fetch() 54ms

----Time in decompression 8ms

MainDownload() Running on a thread ----Time blocked on network 125985ms ----Time in decompression 37832ms

cvmfs_getattr() Oms

cvmfs_lookup() 1887ms ----fetch() 6ms ------Waiting for MainDownload 5ms

cvmfs_opendir() 14072ms ----fetch() 2389ms ------Waiting for MainDownload 2348ms

cvmfs_readdir() 153ms

cvmfs_releasedir() 80ms

cvmfs_open() 1059845ms ----fetch() 1049681ms -----Waiting for MainDownload 1015512ms

cvmfs_read() 4999ms

cvmfs_release() 581ms

cvmfs_forget_multi() 2283ms

cvmfs_forget() 227ms

Waiting for MainDownload 17ms

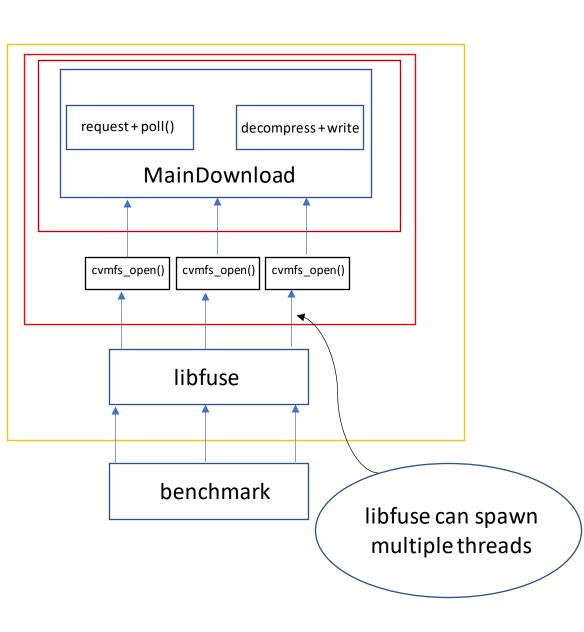
Total Time Spent in Callbacks = 1084204ms

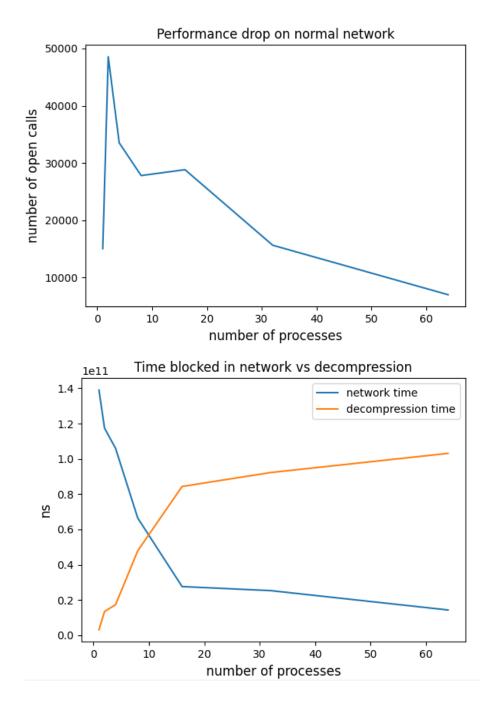
Synchronization issues

- 2 threads attempting to create the same timer
 - One lock associated for the whole map
- 2 threads attempting to modify the same timer
 - timer_id
 timer_path
 parent
 children_timers
 total time atomic
- Each thread needs it's own *last_started_timer*
 - Put *last_started_timer* in TLS

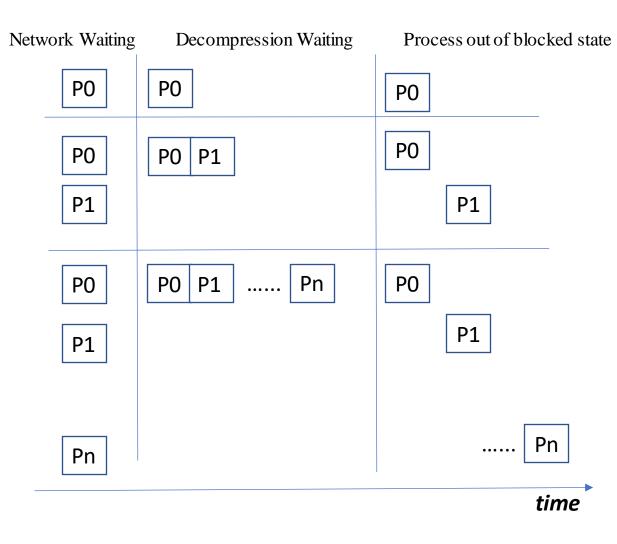
Benchmark scenario:

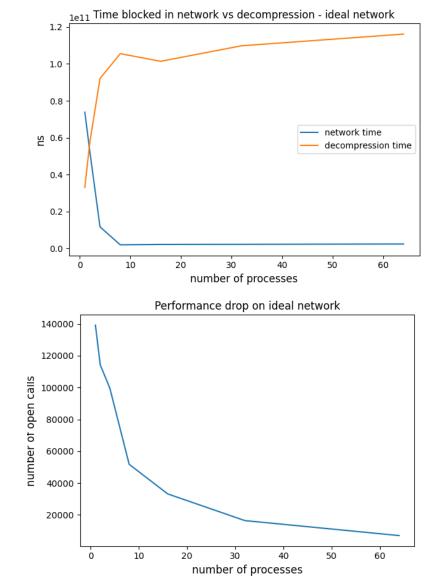
- Multi core system
- Cold Cache
- Multiple processes, attempting to access different data from the same repo (e. g. different jobs using different software versions)





The reason behind the bottleneck





Processes queuing for decompression

Conclusions

- Profiling tools are essential for further improvements in CVMFS performance.
- We developed external tools for the first steps of the analysis.
- We developed internal timers that can offer an in-depth view on the possible bottlenecks.
- We found that, by parallelizing the data decompression, we can improve performance on multiple-processes / multiple-data scenarios.
- We are currently looking into a new benchmark, provided by the Alice experiment.