



Performance benchmark of LHCb code on state-of-the-art x86 architectures

Daniel Hugo Campora Perez, Niko Neufled, Rainer Schwemmer

CHEP 2015 - Okinawa



Background



- The LHCb High Level Trigger (HLT) farm
 - 1800 servers
 - Approximately 40.000 x86 compatible cores
 - Reduce detector data stream from 50-60 GB/s to 1 GB/s
 - 100 m underground
 - Heterogeneous compute cluster
- Farm Upgrade
 - More computing power is necessary to handle the LHC lumi upgrade
 - Decommissioned 550 oldest machines (Nehalem Based Systems)
 - Bought 800 Haswell systems as replacement
- Constraints
 - Farm needs to be able to filter an additional 700.000 events per second (2/3 of old capacity)
 - Has to fit in a 200 kW power envelope due to cooling limitations





Benchmark



- Determine the most cost efficient compute platform
- Help our colleagues to tune the Trigger
- Create a live DVD to distribute to vendors
- DVD has only 4 GB of space
 - 1.x GB of operating system
 - 2 GB of sample data for input
 - Leaves < 1 GB of space for actual HLT software
 - Need to somehow get rid of unnecessary baggage in our software packages
- Use "strace" on running HLT instance to find the minimum number of files required for the trigger
- Copy onto live DVD
 - + One touch wrapper script

THEFT	A Tablergenerativeveryonerraner/mb					
27133	mmap(0x3ec9e00000, 2232320, PROT READIPROT EXEC, MAP PRIVATEIMAP DENYWRITE, 3, 0) = 0x3ec9e00000					
27133	mprotect(8x3ec9e1d800, 2097152, PROT NONE) = 8					
27133	MRAD (9X3eca01d000, 16384, PROT READ PROT WRITE, MAP PRIVATE IMAP FIXED IMAP DENYWRITE, 3, 8x1d000) = 8x3eca01d000					
27133	close(3) = 0					
27133	open("/11b64/11bd1 so 2, 0 RD0N(Y) = 3					
27133	100/00 "11/17/01/01/01/01/01/01/01/01/01/01/01/01/01/					
27133	fstat(3, (st modes) IEPE6(0755, st s120=22536, 1) = 0					
27133	MBAD (013055200000 2100000 PPOT READIPROT EXEC. MAD PRIVATEIMAD DENVWRITE, 3, 0) = 013055200000					
27133	22 minuplexected out 210000, File Part Nove					
27133	man(8x3ec5482868, 8192, PROT READIPROT WRITE, MAP PRIVATE HAP ETXED MAP DENYWRITE, 3, 8x2888) = 0x3ec5482868					
27133	close(2) = 0					
2 133	nnen("/lib64/libc so 60 0 R00N(Y) = 3					
27133	Teach 1 (11) (11) (11) (11) (11) (11) (11) (
27133	fstatis (st modes TEPERIA75, st size=1926888)) = a					
27133	man (NII A 696 DECT PEAD DETTE MAD DETVATE MAD ANONYMOUS -1 8) = 8x7F9655794668					
27133	MADE RY 36 44 499 49 4 4 4 4 4 4 4 4 4 4 4 4 4 4					
27133	mrotort(8v3ar4f8h900, 2003656, DPDT, MALE) = 8					
27133	MAN (AVARCS184868 26488 DOOT FEALEDRY WITE MAD DEVVALEIMAD ELVERIMAD DEVVWETTE 3 AV184868) = AV3855184888					
27133	man (Avar 518 FOR 18606 DOT PEAL DOT PEAL DOT WRITE MAD DIVATE MAD ETVENIMAD ETVENIMAD AVAN WRITE, 1 A) = Avar 518 FOR					
27133	close(3) = 0					
27133	man(NUL) AROS DOOT PEADLOOT WOTTE MAD DETVATEINAD ANONYMOUS -1 B) = 8x7F0065792088					
27132	man (NUL) 4005 DON'T WOTTE MAD DEVALUAD ANONYMOUS 1 8 = 877505570089					
27123	arch net1(APU ST (S. 8770)Short(2) = 8					
27122	mrotect(avaccaccaccaccaccaccaccaccaccaccaccaccac					
27133	mprotect(Aviaccifization, 16384, PPOT PEAD) = 0					
27133	mprotect($8x3cr4c1f808, 4096, PROT READ$) = 0					
27133	Building (ax75966)796000 80377 = 0					
27133	rt signrocmask(STG BLOCK MULL [] 8) = 6					
27133	open("/dev/ttv", 0 ROWELD NONELOCK) = 3					
27133						
27133	hrb(6) = ev128f866					
27133	brk(8x12b8998) = 0x12b8999					
27133	open("/usr/lib/locale/locale-archive", 0 RDON(Y) = 3					
27133	fstat(3, (st mode=5 [FPE6]8644, st size=99154480,)) = 0					
27133	mman(NULL 99154488, PROT READ, MAP PRIVATE 3, 0) = 0x7f9e85982908					
27133	close(3) = 0					
27133	netuid() = 18848					
27133	netaid() = 1003					
27133	octevid) = 10948					
27133	optend() = 1003					
27133	<pre>cl_stoprocmash(sts_BLOCK, NULL, [], 8) = 6</pre>					
2 333	open("/proc/meminfo", PRONLYIO CLOEXEC) = 3					
27133	$f_{statis, st mode=s}$ (FREG[0444, st size=0,)) = 0					
27133	mmap(NULL, 4096, PROT READIPROT WRITE, MAP PRIVATE MAP ANONYMOUS, -1, 0) = 0x7f9e6b7a8000					
27133	read(3, "MemTotal: 24729748 kB\nMemF", 1824) = 1024					
27133	close(3) = 0					
27133	munmap(0x7f9e6b7a8000, 4096) = 6					
27133	rt_sigaction(SIGCHLD, (SIG_DFL, [], SA_RESTORER, 0x3ec4e329a0), (SIG_DFL, [], 0), 8) = 0					
27133	rt_sigaction(SIGCHLD, [SIG_DFL, [], SA_RESTORER, 0x3ec4e329a0), (SIG_DFL, [], SA_RESTORER, 0x3ec4e329a0), 8) = 0					
27133	rt_sigaction(SIGINT, (SIG_DFL, [], SA_RESTORER, 0x3ec4e329a0), (SIG_DFL, [], 0), 8) = 0					
27133	rt_sigaction(SIGINT, (SIG_DFL, [], SA_RESTORER, 0x3ec4e329a0), (SIG_DFL, [], SA_RESTORER, 0x3ec4e329a0), 8) = 0					
27133	rt_sigaction(SIGQUIT, (SIG_DFL, [], SA_RESTORER, 0x3ec4e329a0}, (SIG_DFL, [], 0}, 8) = 6					
27133	rt_sigaction(SIGQUIT, (SIG_DFL, [], SA_RESTORER, 0x3ec4e329a0), (SIG_DFL, [], SA_RESTORER, 0x3ec4e329a0), 8) = 0					
27133	rt_sigprocmask(SIG_BLOCK, NULL, [], 8) = 0					
27133	rt_sigaction(SiGQUIT, (SiG_IGN, [], SA_RESTORER, 0x3ec4e329a0), (SIG_DFL, [], SA_RESTORER, 0x3ec4e329a0), 8) = 0					
27133	uname({sys="Linux", node="hltf0101",}) = 0					
27133	rt_sigprocmask(SIG_BLOCK, NULL, [], 8) = 0					
97499	Dop///wer/libs//aconu/aconu andulas cache" 0 PDON(X) = 2					

Capture every: open/stat/execv



Application Profile

- Processes very short lived, highly independent tasks, a.k.a Events
 - 200-300 ms per Event
- No inherent multi-threading
 - Parallelism is achieved by launching multiple instances
 - Launch N instances of program
 - Choose N for max machine throughput
- Mixture of strongly branching code and floating point operations
- Memory footprint is approximately 1.1 GB
 - 600 MB static
 - 400-500 MB dynamic
- Processes are created by fork()ing a master process
 - Reduces memory footprint
 - Accelerates startup
 - Has some issues \rightarrow TBD







Results





Benchmark Output



- Plot 1: Machine throughput vs. number of running instances
- Plot 2: Increase in throughput per additional instance
- Benchmark scans the optimum number of program instances
- Results from previous generation and Haswell Dual Socket farm nodes



Interesting little detail



Program seems to run faster on first socket than on second

- Effect is also visible on old generation nodes, but to lesser extent

Rainer Schwemmer - CHEP – Okinawa 2015

Consequences of NUMA Architecture

- Fork() master process is scheduled to a particular NUMA node and allocates most of its memory there
- Children are distributed over the 2 sockets depending on number of children
 - First to the socket with master
 - Then other socket(s)
- Off master processes are hit by additional latency in QPI/HT hop
- Spawn as many masters as sockets
 - Lock master and children to specific NUMA node with numactl
 - Up to 14% performance gain in machine throughout!
- Haswell 10+ core CPUs are internally divided into two NUMA nodes
 - 2-3% additional gain
 - Enable 'Cluster On Die' (COD) feature in BIOS

	СРU	Decisions/s No NUMA	Decisions/s NUMA	NUMA Gain
-	Intel X5650 (8 cores)	599.6	648.8	1.08
	Opteron X272	632.35	682	1.08
	E_2630 v3 (8 cores)	865	986	1.14
	E_2650 v3 (10 cores)	1129	1210	1.07







8



Other Memory Issues





Performance Overview



- Performance model based on benchmark and memory measurements
- Assumes a dual socket system



Costs / Power

- Cost of dual socket system for fixed throughput
 - 24 blades for Avoton
- Costs included:
 - Main board
 - Memory
 - PSU
 - CPU
 - Chassis
 - Double memory required for 2687W and above
 - AMD6376 and Avoton based on system quotes
- Power consumption
- Measured
 - Avoton
 - AMD 6376
 - E5-2630 v3
 - E5-2690 v3
- Other Power consumption
 estimated
 - 1.6 x TDP x N_{sockets}
 - Fits all measured systems within 5%



and and a set of the s

AND 6376 2150



45-263043



A word about AMD and Avoton



• AMD

- Pretty good value for price (not much choice)
- Power consumption is an issue
- Avoton
 - Actually Better than portrayed here
 - Suffers from two key shortcomings in our case
 - No Hyperthreading
 - Very little cache per core (512 k)
 - \rightarrow Instructions per Clock cycle: < 0.4
 - If your code is optimized for this it should be quite efficient
 - Many slow systems
 - Network overhead
 - Administration overhead
 - HDDs (if you need them)







- We have created a stand alone version of our High Level Trigger
- Significantly improved Trigger performance with NUMA awareness
- Memory access optimization has become even more critical with latest Intel CPU generation
 - Both Xeon and Avoton
- Selected E5-2630 v3 as most cost efficient platform for our new farm
- This does not mean it's also the best platform for you though





Acknowledgements

- Companies / Institutions helping with access to test platforms and power measurements
- CERN IT
- E4 Computer Engineering
- Intel Swindon Labs
- ASUS
- Macle GMBH





Questions