Sched-wakeup/Sched-switch Analysis Module Trace Compass Benchmarking

Abdellah Rahmani with Pr. Michel Dagenais

January 11th, 2021

Polytechnique Montreal

DORSAL Laboratory

Agenda

Part 1 : Sched-wakeup/Sched-switch Analysis Module

- What ? Why ?
- How it works ?
- The main views

Part 2 : Trace Compass Benchmarcking

- Why benchmarcking Trace Compass
- Results
- Ongoing and future improvements of Trace Compass

Analysis Module - What ? Why ?

- Perform a latency analysis of the system
- Detect real time problems

Analysis Module - How it works ?

- Reads the events of the trace
- Filters the Sched-wakeup and Sched-Switch events
- Accesses the fields of the filtered events and extracts thread information : name, TID, timeStamps, priority
- Fills the segment-stores with the information
- Populates the views

Analysis Module - The main views : Statistics View

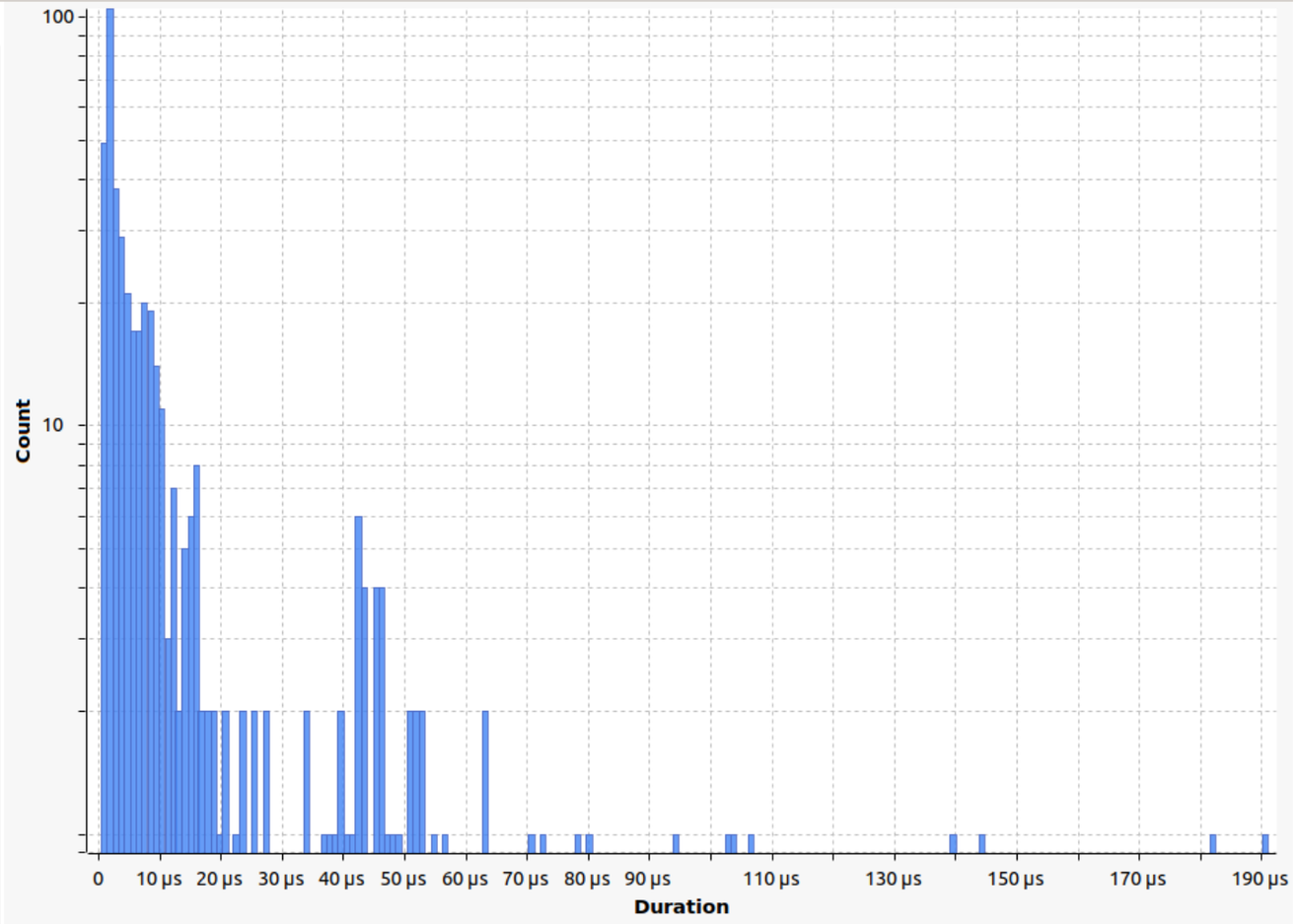
Sched_Wakeup/Switch Latency Statistics

Level	Minimum	Maximum	Average	Standard Deviation	Count	Total
kernel						
Total	477 ns	12.006 ms	29.807 µs	113.588 µs	136979	4.083 s
khungtaskd	25.164 µs	25.164 µs	25.164 µs	---	1	25.164 µs
gvfsd	1.56 µs	3.643 µs	2.442 µs	619 ns	8	19.54 µs
kworker/2:3	2.658 µs	40.241 µs	12.639 µs	18.412 µs	4	50.554 µs
kworker/2:2	26.45 µs	149.444 µs	87.947 µs	---	2	175.894 µs
at-spi2-registr	1.313 µs	2.19 ms	31.966 µs	123.555 µs	326	10.421 ms
systemd-timesyn	11.551 µs	11.551 µs	11.551 µs	---	1	11.551 µs
kworker/2:1	1.002 µs	12.006 ms	66.791 µs	741.845 µs	262	17.499 ms
kworker/2:0	1.488 µs	2.1 ms	29.831 µs	113.292 µs	599	17.869 ms
upowerd	4.039 µs	85.469 µs	25.558 µs	29.822 µs	10	255.582 µs
gnome-terminal-	1.115 µs	2.699 ms	23.869 µs	68.233 µs	3008	71.798 ms
jbd2/sda3-8	905 ns	265.452 µs	36.425 µs	52.378 µs	43	1.566 ms
xpad	3.223 µs	1.411 ms	29.624 µs	137.305 µs	108	3.199 ms
(ostnamed)	856 ns	1.657 µs	1.245 µs	351 ns	6	7.468 µs
pool	1.026 µs	44.517 µs	5.379 µs	10.776 µs	16	86.061 µs
pool-tracker-ex	1.564 µs	445.721 µs	76.319 µs	105.502 µs	34	2.595 ms
kworker/2:1H	1.174 µs	212.453 µs	7.275 µs	14.099 µs	472	3.434 ms
evolution-alarm	20.734 µs	20.734 µs	20.734 µs	---	1	20.734 µs
rtkit-daemon	1.935 µs	46.033 µs	17.773 µs	11.043 µs	18	319.918 µs
pool-tracker-mi	13.112 µs	458.231 µs	88.728 µs	163.433 µs	7	621.094 µs
gvfs-goa-volume	1.741 µs	103.292 µs	16.727 µs	38.174 µs	7	117.089 µs
NetworkManager	2.16 µs	107.158 µs	25.908 µs	23.634 µs	48	1.244 ms
kworker/u8:27	477 ns	1.435 ms	17.641 µs	62.096 µs	1629	28.737 ms

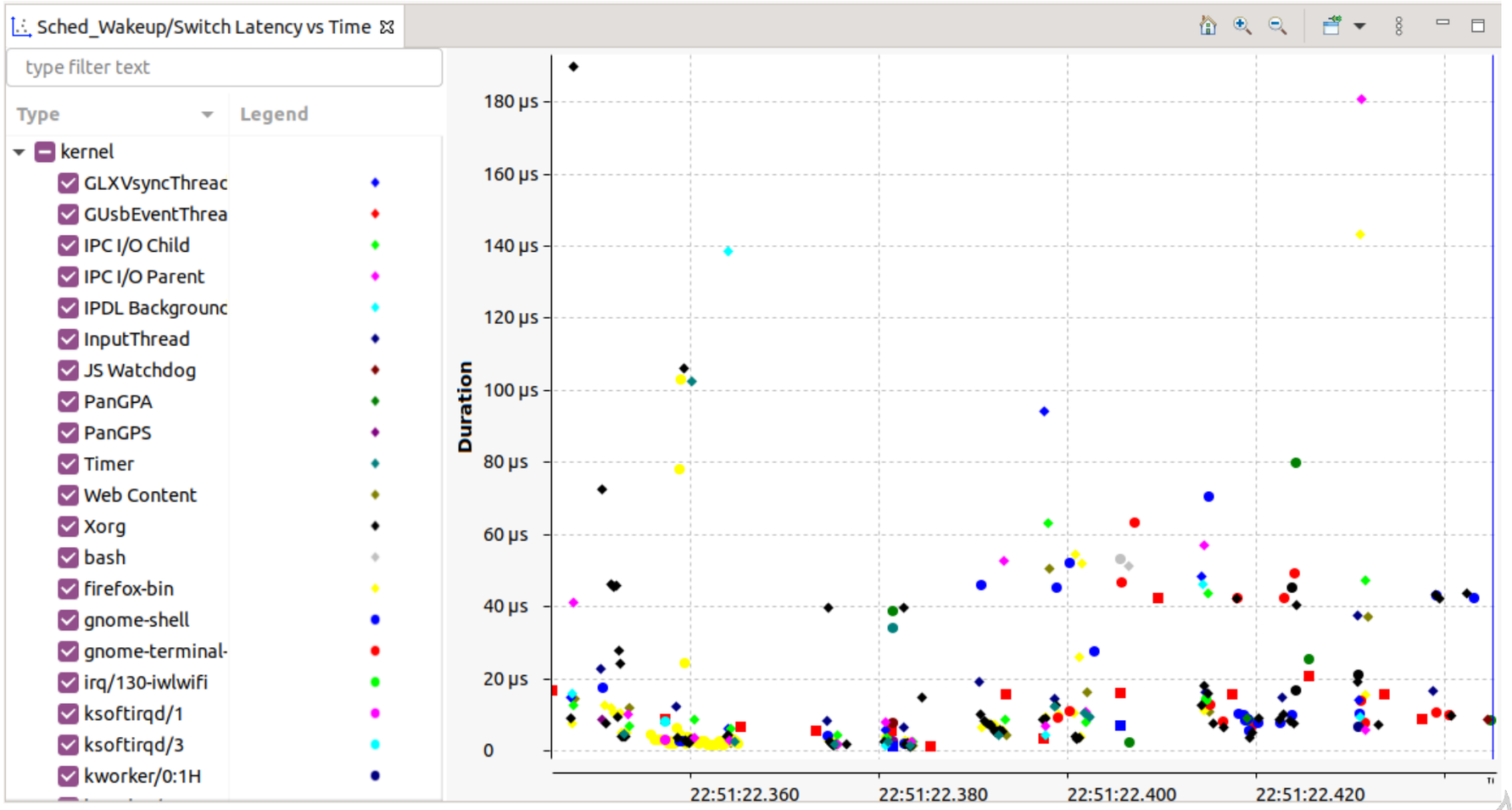
Analysis Module - The main views : Density view

Sched_Wakeup/Switch Density

Start Time	End Time	Duration	Exec_Name	TID	Priority
22:51:22.345 455 216	22:51:22.345 472 007	16,791	rcu_sched	11	20
22:51:22.347 346 473	22:51:22.347 355 680	9,207	Xorg	37978	20
22:51:22.347 427 972	22:51:22.347 443 049	15,077	GLXVsyncThre	39907	20
22:51:22.347 526 189	22:51:22.347 542 188	15,999	IPDL Backgrou	39923	20
22:51:22.347 554 386	22:51:22.347 562 130	7,744	firefox-bin	39799	20
22:51:22.347 615 758	22:51:22.347 657 488	41,730	IPC I/O Parent	39885	20
22:51:22.347 710 415	22:51:22.347 723 270	12,855	IPC I/O Child	40340	20
22:51:22.347 799 091	22:51:22.347 813 671	14,580	Web Content	40336	20
22:51:22.347 633 860	22:51:22.347 825 349	191,489	Xorg	37978	20
22:51:22.350 529 560	22:51:22.350 552 530	22,970	InputThread	38004	20
22:51:22.350 751 685	22:51:22.350 760 523	8,838	PanGPS	881	20
22:51:22.350 719 538	22:51:22.350 792 925	73,387	Xorg	37978	20
22:51:22.350 883 812	22:51:22.350 901 281	17,469	gnome-shell	38149	20
22:51:22.350 943 004	22:51:22.350 955 988	12,984	firefox-bin	39799	20
22:51:22.351 156 057	22:51:22.351 163 649	7,592	Xorg	37978	20
22:51:22.351 586 826	22:51:22.351 633 571	46,745	Xorg	37978	20
22:51:22.351 711 006	22:51:22.351 723 057	12,051	firefox-bin	39799	20
22:51:22.351 855 833	22:51:22.351 901 624	45,791	Xorg	37978	20
22:51:22.351 977 041	22:51:22.351 987 620	10,579	firefox-bin	39799	20
22:51:22.352 136 623	22:51:22.352 182 818	46,195	Xorg	37978	20
22:51:22.352 257 948	22:51:22.352 268 384	10,436	firefox-bin	39799	20
22:51:22.352 343 920	22:51:22.352 353 417	9,497	Xorg	37978	20
22:51:22.352 399 026	22:51:22.352 409 308	10,282	firefox-bin	39799	20
22:51:22.352 476 521	22:51:22.352 504 615	28,094	Xorg	37978	20
22:51:22.352 551 144	22:51:22.352 561 800	10,656	firefox-bin	39799	20
22:51:22.352 629 375	22:51:22.352 653 833	24,458	Xorg	37978	20
22:51:22.352 700 028	22:51:22.352 705 178	5,150	firefox-bin	39799	20
22:51:22.352 796 777	22:51:22.352 800 930	4,153	Xorg	37978	20



Analysis Module - The main views : Latency VS Time



PART 2 :

Trace Compass Benchmarcking

The benchmarcking - Why ?

- Test How can Trace compass scale with some specific tracing use cases

- Explore some improvement opportunities

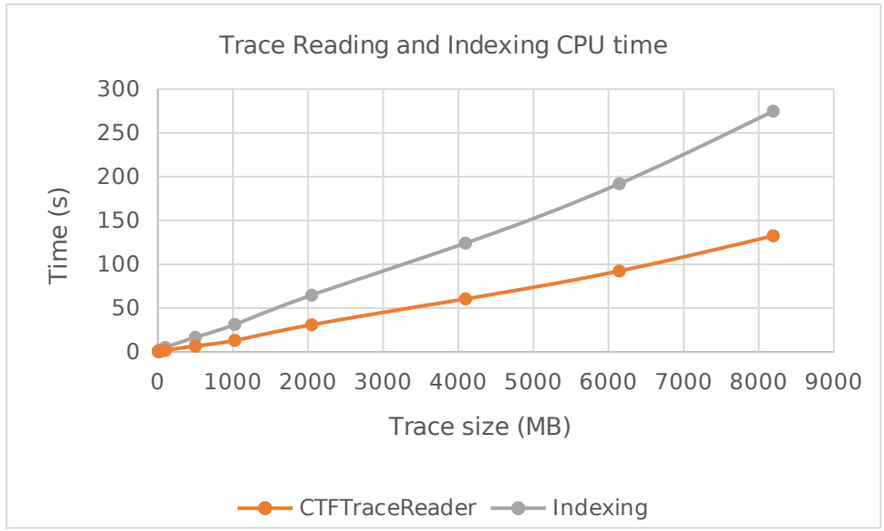
The benchmarcking- Results:metrics for one trace

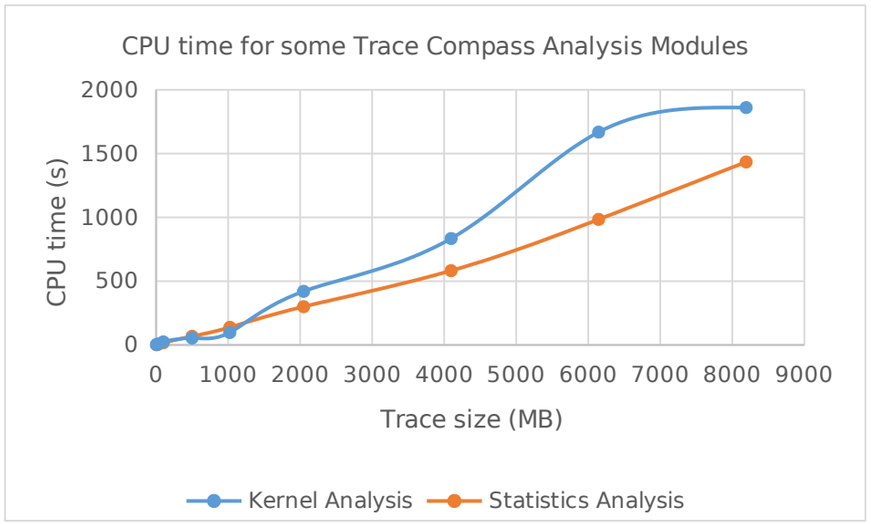
Trace information:

Type: Lttng Kernel Trace Size: ~1 GB Number of events: 35 489 216

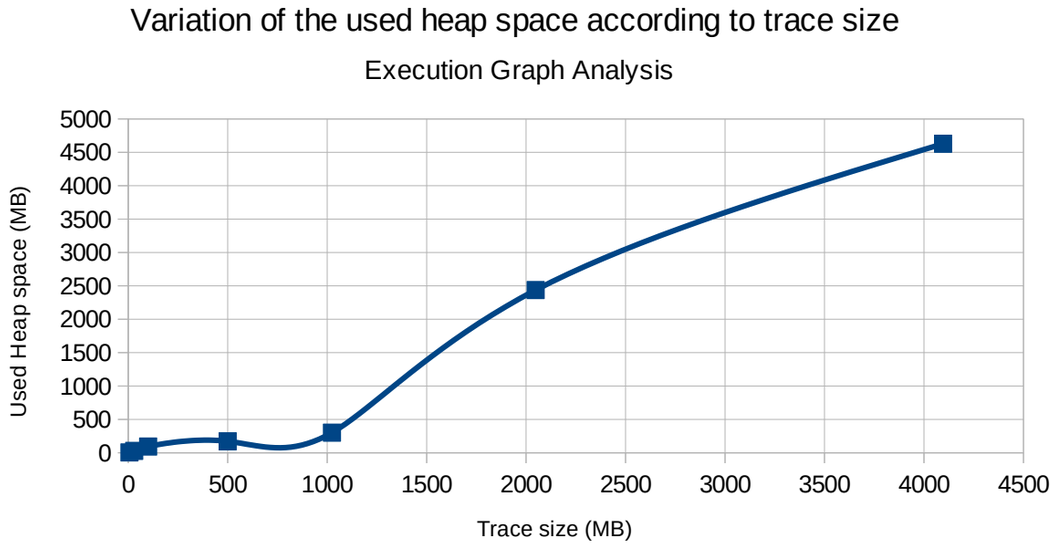
Operation/ Analysis	Trace Reading	Indexing	Kernel Analysis	Statistics Analysis	Execution Graph
CPU time (s)	11.51	4.50	88.80	133.80	129.00
Wall time (s)	10.65	4.54	35.06	40.72	48.94
Used heap space (MB)	-	-	-	-	300.23

The benchmarcking – Results : according to trace size variation

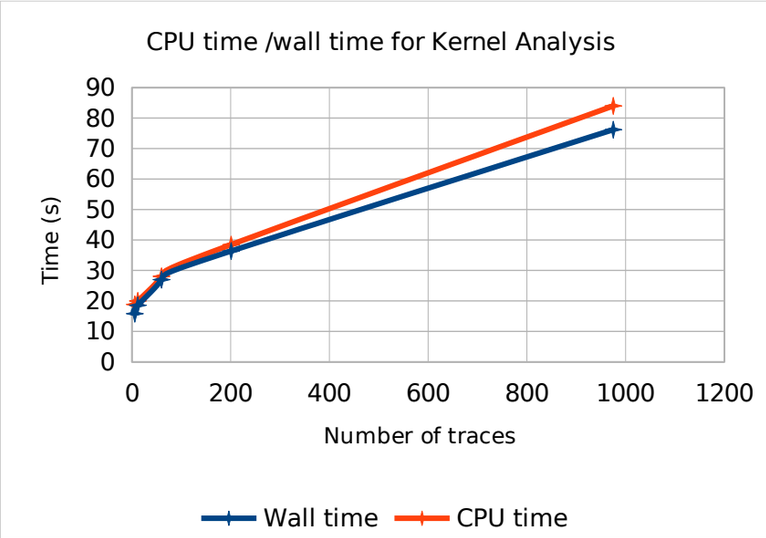


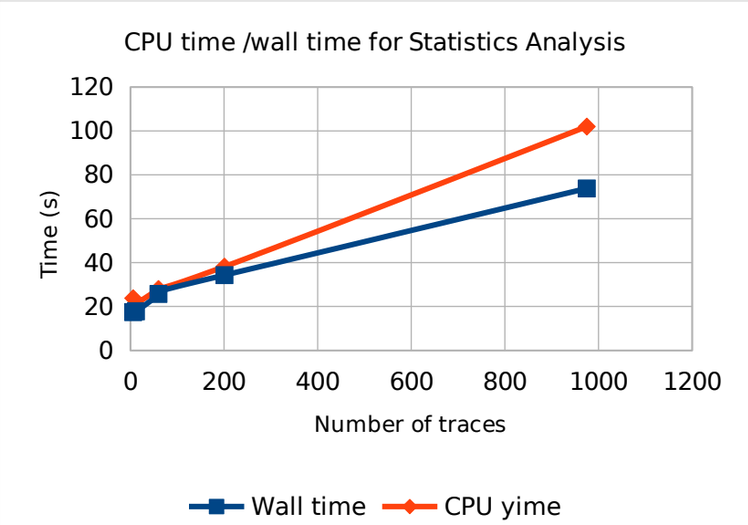


One trace is tested at a time while varying its size

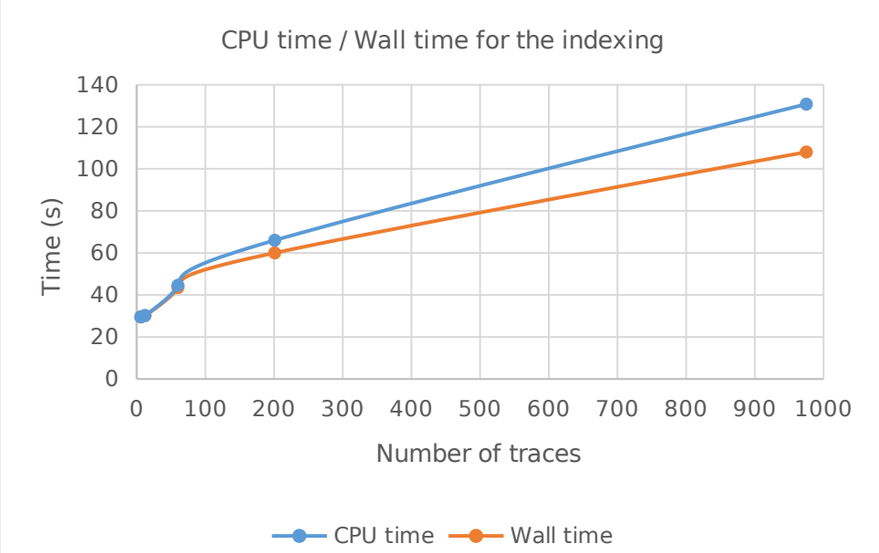


The benchmarcking- Results: according to the number of traces





In this test, we are varying the number of traces while keeping the total size of the traces (experiment) constant at 6 GB



The benchmarcking- Ongoing and future improvements

- Critical path Analysis on disk
- Partial history tree
- Preprocessing of the traces to precompute the main analyses, possibly in parallel for cluster tracing
- Distributed Architecture of trace compass

Q&A