Multicore workflow characterisation methodology for payloads running on the ALICE Grid

Marta Bertran Ferrer
PhD Student - CERN

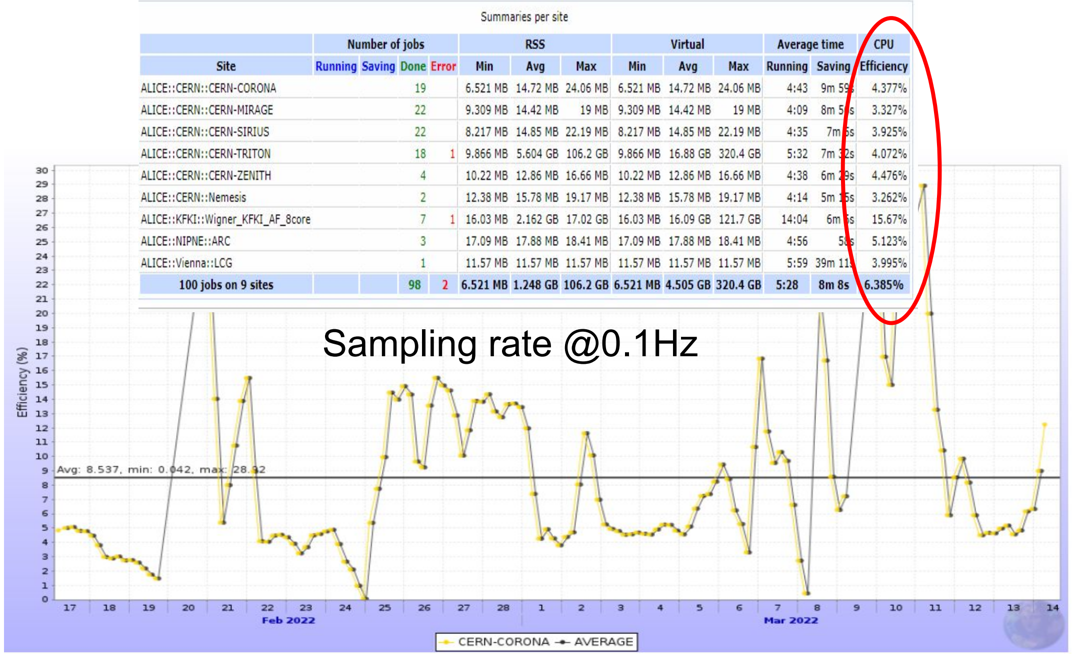
On behalf of the ALICE Collaboration

The updated Run 3 ALICE Software Stack

- After upgrade - 10x larger data volume with higher internal complexity
  - Grid single-core jobs with 2GB/core memory limits no longer feasible
  - **Multicore jobs** spawning multiple parallel processes using shared memory

- Run 3 multicore jobs invoke **order of ~100s concurrent short-lived processes**
  - Grid monitoring framework needs an **update to properly account for the resource usage**
  - Previous methodology monitored single long-lived process
    - Periodic sampling of Linux *ps* output

Efficiency of O2 simulation jobs on eight-core queue



Summaries per site

	Number of jobs				RSS			Virtual			Average time		CPU
Site	Running	Saving	Done	Error	Min	Avg	Max	Min	Avg	Max	Running	Saving	Efficiency
ALICE::CERN::CERN-CORONA			19		6.521 MB	14.72 MB	24.06 MB	6.521 MB	14.72 MB	24.06 MB	4:43	9m 59s	4.377%
ALICE::CERN::CERN-MIRAGE			22		9.309 MB	14.42 MB	19 MB	9.309 MB	14.42 MB	19 MB	4:09	8m 5[illegible]s	3.327%
ALICE::CERN::CERN-SIRIUS			22		8.217 MB	14.85 MB	22.19 MB	8.217 MB	14.85 MB	22.19 MB	4:35	7m 6s	3.925%
ALICE::CERN::CERN-TRITON			18	1	9.866 MB	5.604 GB	106.2 GB	9.866 MB	16.88 GB	320.4 GB	5:32	7m 32s	4.072%
ALICE::CERN::CERN-ZENITH			4		10.22 MB	12.86 MB	16.66 MB	10.22 MB	12.86 MB	16.66 MB	4:38	6m 29s	4.476%
ALICE::CERN::Nemesis			2		12.38 MB	15.78 MB	19.17 MB	12.38 MB	15.78 MB	19.17 MB	4:14	5m 15s	3.262%
ALICE::KFKI::Wigner_KFKI_AF_8core			7	1	16.03 MB	2.162 GB	17.02 GB	16.03 MB	16.09 GB	121.7 GB	14:04	6m 5s	15.67%
ALICE::NIPNE::ARC			3		17.09 MB	17.88 MB	18.41 MB	17.09 MB	17.88 MB	18.41 MB	4:56	58s	5.123%
ALICE::Vienna::LCG			1		11.57 MB	11.57 MB	11.57 MB	11.57 MB	11.57 MB	11.57 MB	5:59	39m 11s	3.995%
100 jobs on 9 sites			**98**	**2**	**6.521 MB**	**1.248 GB**	**106.2 GB**	**6.521 MB**	**4.505 GB**	**320.4 GB**	**5:28**	**8m 8s**	**6.385%**

Sampling rate @0.1Hz

- Sampling active processes not sufficient to correctly monitor parameters

- Wrapping job execution with the `time` command - still incomplete picture
  - Child processes detached during execution

- In both cases CPU efficiency takes *user* and *system* components into account

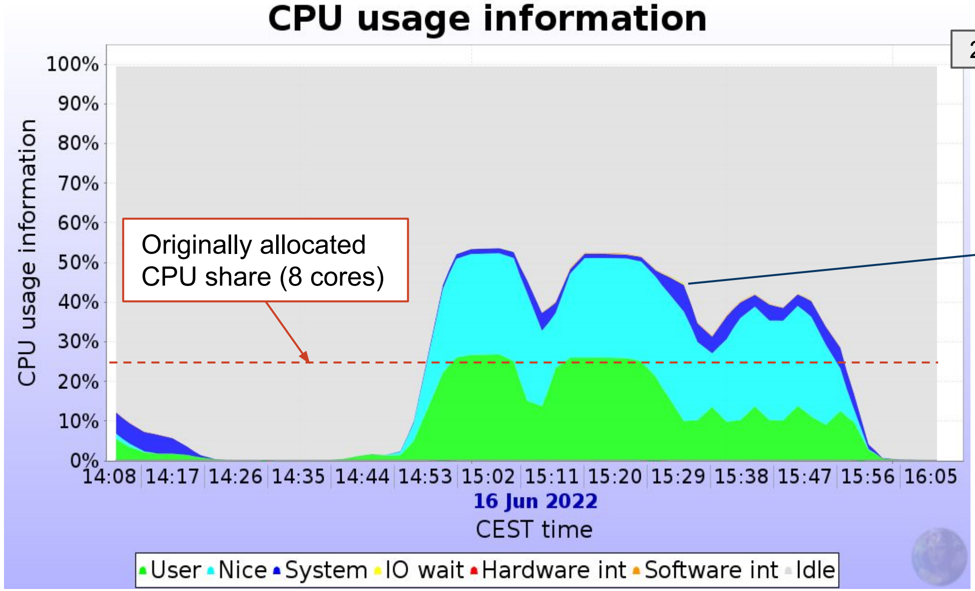
$$\textit{CPU efficiency} = \frac{\textit{User time + System time}}{\textit{Wall time}}$$

New efficiency computation and reporting

- For every job monitor iteration → listing of all the running processes and their children
- For every child - Inspection of the `/proc/PID/stat` file
  - Parsing and summation of increases on field 13 (`utime`) and 14 (`stime`)

$$\textit{CPU efficiency} = \frac{\textit{Sum (User time + System time)}}{\textit{Elapsed time * numCPUs}}$$

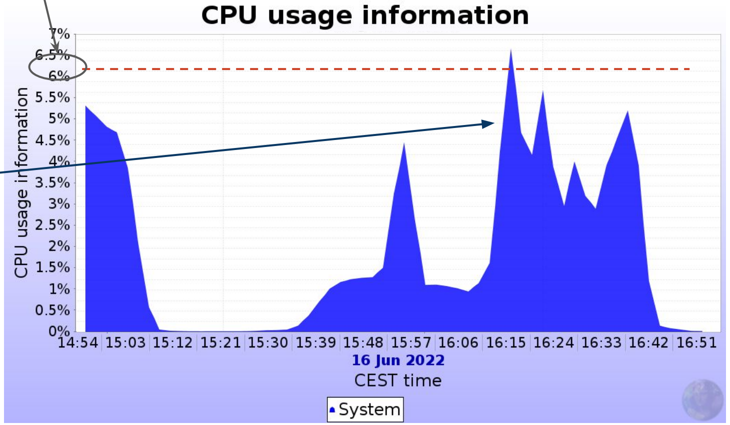
Resource usage of the payloads in a 32-core idle machine



Originally allocated CPU share (8 cores)

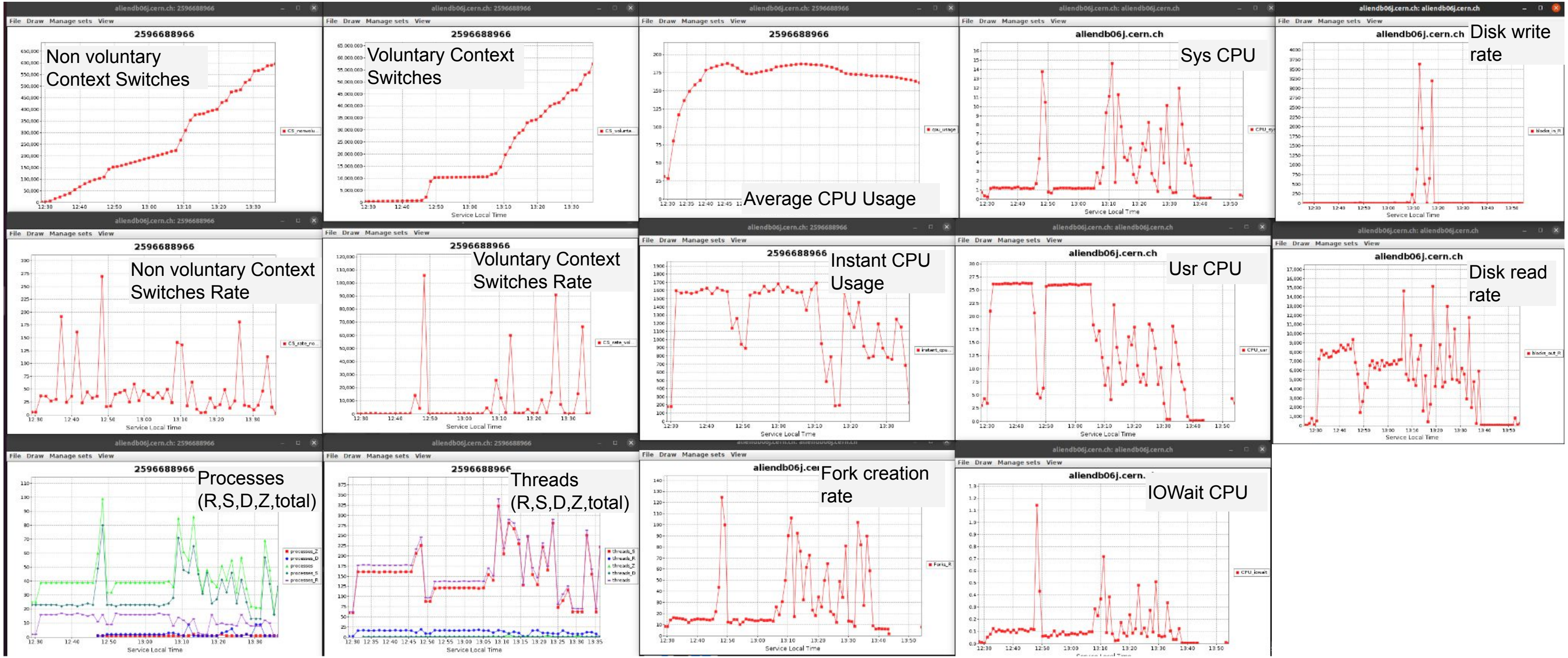
2 CPU cores

Elevated *System* CPU component



CPU efficiency →**160.25%**

Analysed parameters of the system



Non voluntary Context Switches

Voluntary Context Switches

Average CPU Usage

Sys CPU

Disk write rate

Non voluntary Context Switches Rate

Voluntary Context Switches Rate

Instant CPU Usage

Usr CPU

Disk read rate

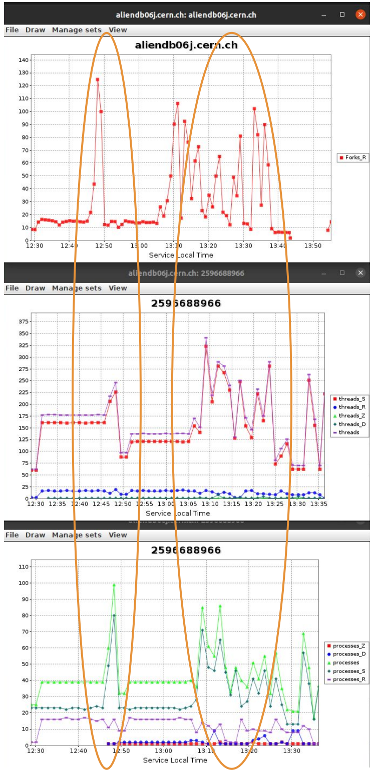
Processes (R,S,D,Z,total)

Threads (R,S,D,Z,total)

Fork creation rate

IOWait CPU

Deeper look into fork deployment rate



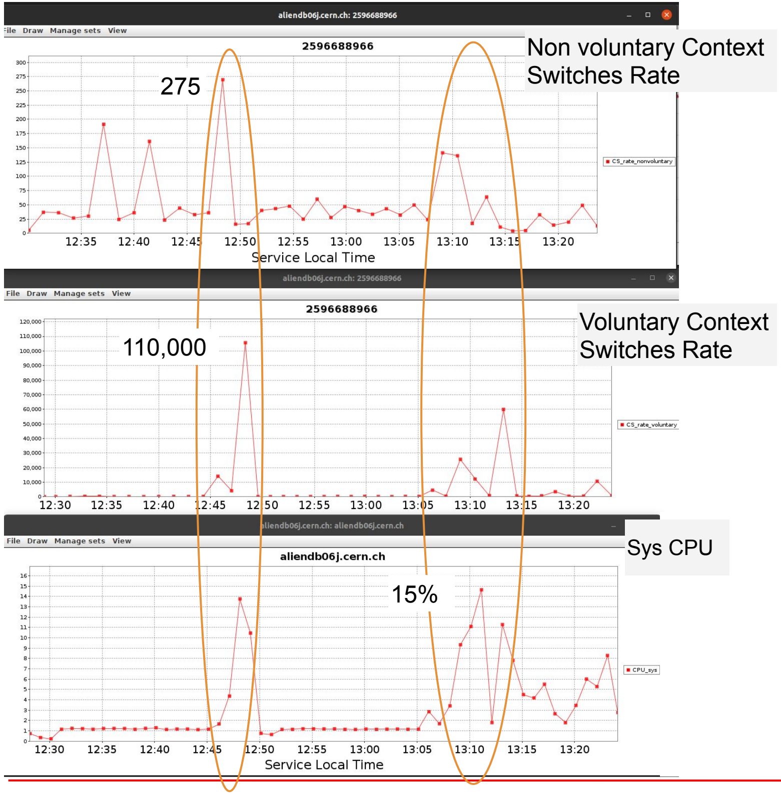
Fork creation rate

Threads (R,S,D,total)

Processes (R,S,D,Z,total)

- Fork creation rate correlated with peak in number of concurrently running processes and threads
- This behaviour is associated with the deployment of **short-lived processes**
    - Most of the time in sleeping state
- Detected large **overhead** in process creation
    - Deeper analysis on underlying causes - detailed monitoring to search for areas to be optimized

Deeper look into context switching



Non voluntary Context Switches Rate

275

Voluntary Context Switches Rate

110,000

Sys CPU

15%

- The *system* CPU is greatly impacted by the context switching rate
- Specifically, peaks on ***voluntary*** context switching are clearly correlated to peaks on *system* CPU

Process deployment analysis

Observed high overhead / **large *System* CPU usage**

Deep analysis of **job internal behaviour**

- Origin of system calls might not be observable at first glance

Job execution wrapped with **`strace`** command:

```
strace -e trace=process -ttt -f -s 10000 -o jobId-execution.strace
```

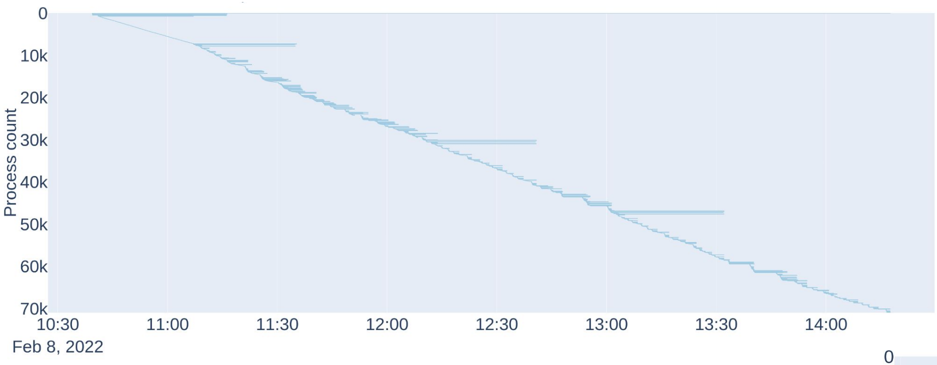
- Detailed study of the deployed processes and threads, their amount, execution frequency, time distribution and resource usage

Process deployment analysis

Observations from the reported metrics revealed some **potential areas for improvement**

- High cost of system calls execve in the *alienv* context
  - Improved by correctly loading the dependent libraries
- System calls accounting and callee identification
  - O2 process merging and decrease processes initialisations

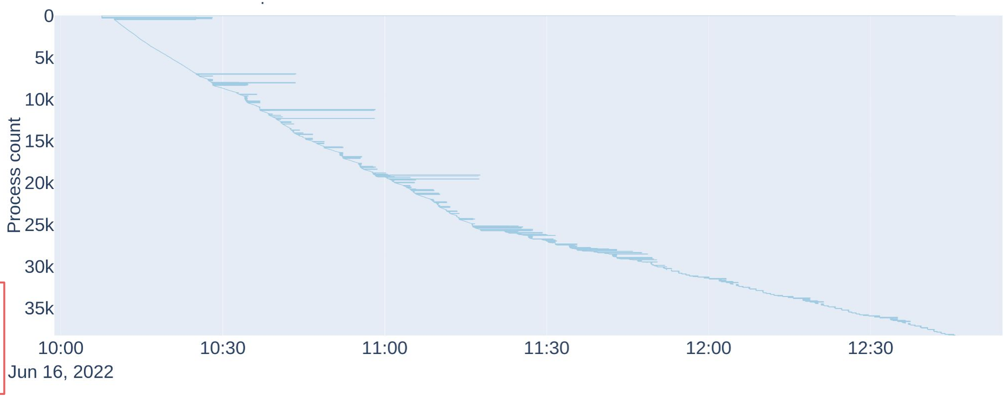
Profiled execution analysis and improvements



Process durations Gantt plot

Originally:

Total process+thread count - **72.5K**

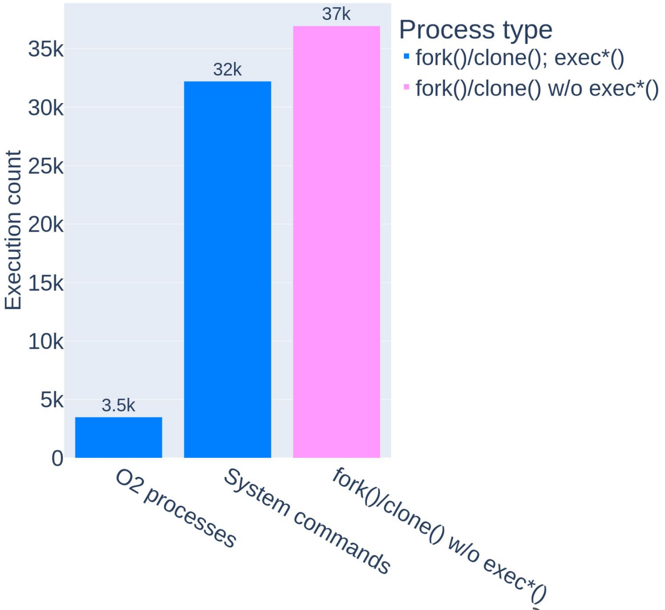


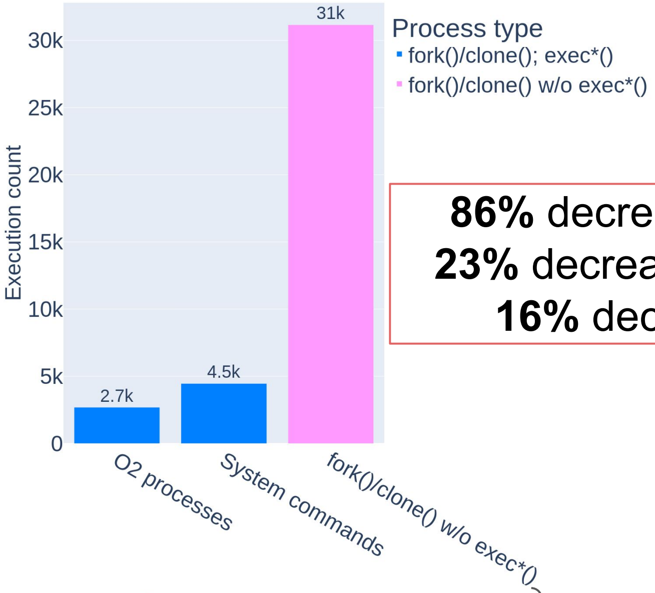
After improvements:

Total process+thread count - **38.3K**

Count decreased by **47.17%**

Profiled execution analysis and improvements



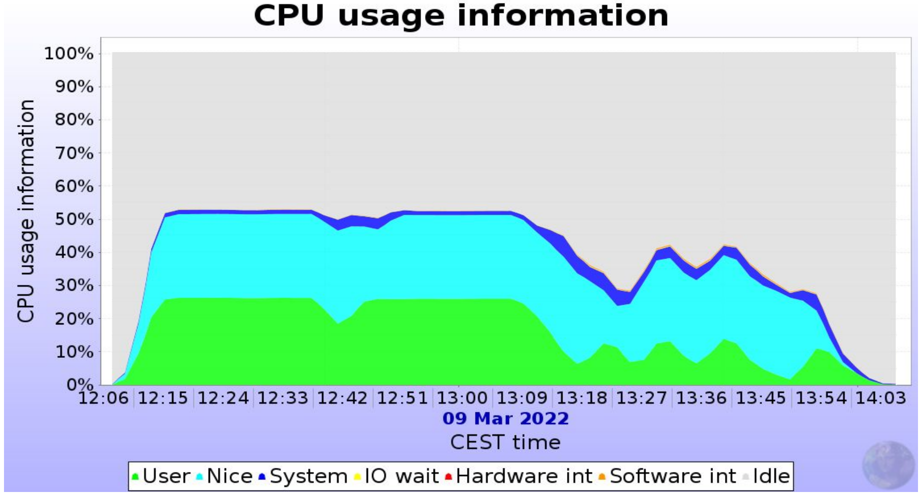


86% decrease on system calls
23% decrease on O2 processes
16% decrease on threads

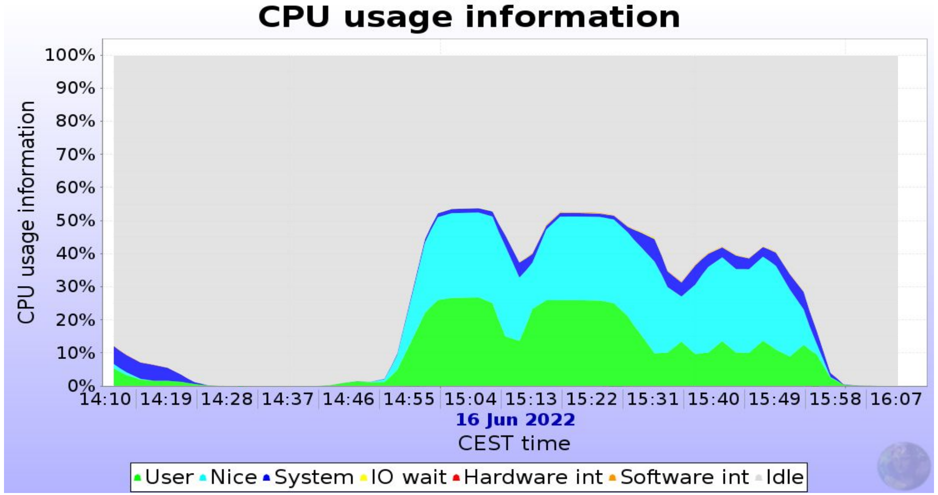
32k system calls
3.5k O2 processes
37k threads
} 72.5k total

4.5k system calls
2.7k O2 processes
31k threads
} 38.2k total

Profiled execution analysis and improvements



1:46h



1:09h

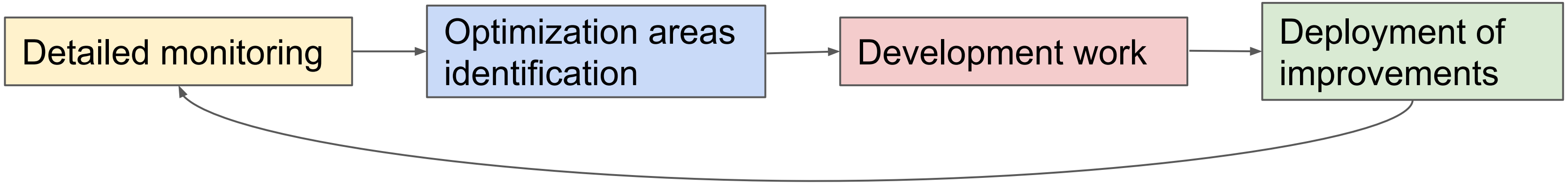
Execution time decreased by **~35%**

Outlook

- The high rate of process turnaround has been addressed with an **improved accounting of the used resources**
- The new efficiency accounting led to **real-time detailed monitoring of jobs**
    - Detailed monitoring as source for **spotting payload optimization areas**
- Execution wrapped with **`strace`** for process deployment and execution analysis
- Enhancements introduced in framework leading to improvements in several areas
    - Understanding of code behaviour and process count
    - CPU utilization
    - Code execution time

Outlook

- We aim to continue to study payloads and introduce further optimizations with the implemented **profiling methodology**



Detailed monitoring

Optimization areas identification

Development work

Deployment of improvements