In-Kernel PSI Auto Monitor Feature

Pintu Kumar Agarwal

Senior Staff Engineer, Qualcomm India Pvt. Ltd.

Agenda

- About Me
- Introduction – PSI Overview
- Problem Statement
- PSI Existing Mechanism
- PSI Auto Monitor – Design, Interface & Block Diagram
- Results & Trace Output
- Gen-AI Integration & Report
- Benefits, Roadmap & Summary

About Me

- Linux Kernel engineer with over two decades of experience designing and deploying Linux based products.
- Major expertise in Linux Kernel bring-up, memory management, etc.
- Contributed several patches in upstream Linux Kernel since 2012.
- Presented talks in Linux conference worldwide. This is the 8th talk.
- Currently working in Qualcomm, India, Bangalore.
- Previously worked in Sony, Samsung and Sasken.
- Great passion and interest in Linux Kernel research and open source.
- Contact me here: [20]

Introduction

- PSI (Pressure Stall Information) reflects CPU, memory, and I/O load in the system.
- Shows load percentage in the system in the last 10, 60, 300 seconds interval.
- Available since Linux kernel 4.20 onwards under: kernel/sched/psi.
- We have developed another interface called auto monitor: kernel/sched/psi_auto_monitor.
- This is an extension on my previous ideas presented at LPC 2024 (Austria):
  - https://lpc.events/event/18/contributions/1884/attachments/1439/3069/LPC2024_PIntu_PSI.pdf
- In this talk we will cover more details about auto monitor.

```
/ # cat /proc/pressure/cpu
some avg10=10.08 avg60=0.02 avg300=0.00 total=39360
full     avg10=0.00 avg60=0.00 avg300=0.00 total=0
```

```
/ # cat /proc/pressure/memory
some avg10=10.01 avg60=0.02 avg300=0.00 total=52645
full     avg10=0.00 avg60=0.00 avg300=0.00 total=10155
```

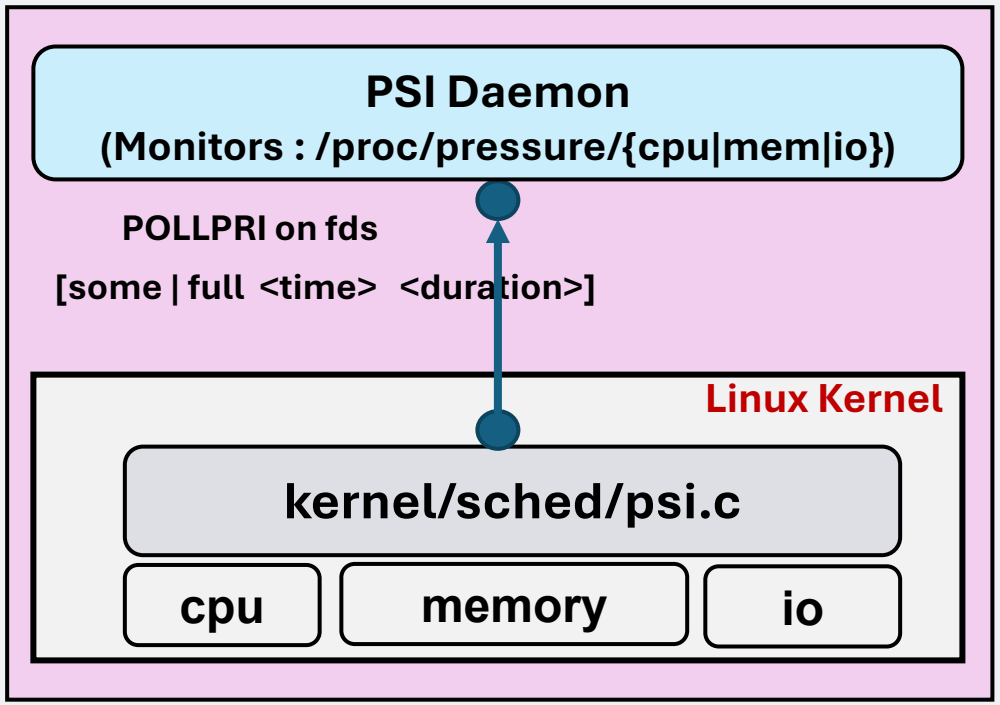
```
/ # cat /proc/pressure/io
some avg10=10.00 avg60=0.02 avg300=0.00 total=29723
full     avg10=0.00 avg60=0.00 avg300=0.00 total=0
```

```
/ # cat /proc/pressure/irq
full avg10=1.38 avg60=1.71 avg300=1.87 total=70239797
```

Problem

- What will we normally do when we are in these situation ?
  - System behaves sluggish or unresponsive at times.
  - Scheduling, latency issue, high CPU, sudden spikes, OOM or performance issues.
- First, we try to reproduce the issue with the same scenario.
- Then we may run several tools in the background to collect data.
  - top, /proc/meminfo, /proc/vmstat, iostat, smem, /proc/<pid>/smaps_rollup, perf tool, etc.
  - Enables tracing, bpf, etc. to monitor specific subsystem.
  - Many other good tools which I am not aware.
- Dump logs such as: dmesg or /var/log/messages to monitor signs of failure.
- Even develop a daemon/service in user space that monitor workloads and pressure.
- With PSI auto monitor we may not need these, while still getting useful insights.
- Systems don’t fail suddenly. They degrade under pressure. PSI shows pressure. PSI auto monitor acts on it.

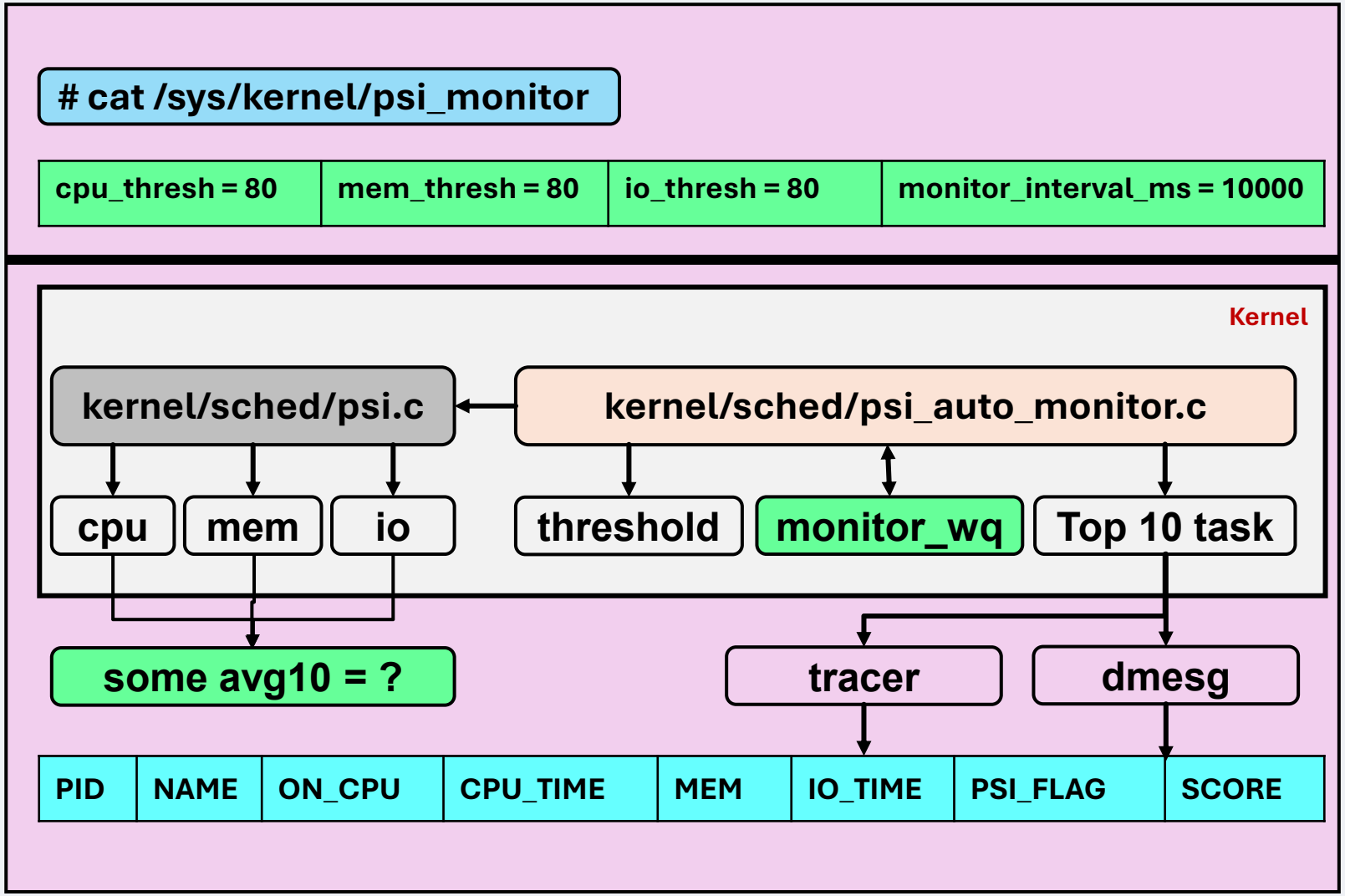
PSI Existing Mechanism



- Requires user space daemon to monitor nodes.
- Define a specific threshold window for each trigger event.
- Continuously poll for an event to occur on that fd.
- User needs to figure out which tasks causes the spike.
- Reference:
- https://docs.kernel.org/accounting/psi.html
- https://unixism.net/2019/08/linux-pressure-stall-information-psi-by-example/

"PSI triggers convert accumulated stall time within a sliding window into real-time event notifications for proactive system control."
But we still need to figure out which task causes the stall during this window.

PSI Auto Monitor Block Diagram



- Automatic PSI monitor in Kernel
- Minimal user interaction needed
- One time threshold/interval settings
- Composite weighted task score based on cpu, mem, io usage
- Sorting of tasks based on score
- If threshold breach, dump top 10 thread/process info in kernel logs
- Choose tasks on RQ and not idle
- Standard tracing interface
- Feature can be controlled using CONFIG_PSI_AUTO_MONITOR

PSI Auto Monitor Interface

❑/sys/kernel/psi_monitor/

- cpu_thresh = 80
- mem_thresh = 80
- io_thresh = 80
- monitor_interval_ms = 10000
- cpu_weight = 5
- rss_weight = 2
- io_weight = 1

These together will create a cumulative score for sorting

- These are default values set at build and boot time.
- You can change these values at runtime before running your use cases.

Results-1

- **Board: Beagle Play ; ARM64 ; Cortex-A53 ; CPU: 4 ;**
- **RAM: 2GB ; Storage: EMMC (ext4) ; Linux Kernel Version: 7.0.0-rc4+**
- **Scenario: User thread, Kernel thread, Work queues, ffmpeg, stress test**

PSI_FLAGS

ON_CPU	MEMSTALL_ RUNNING	RUNNING	MEMSTALL	IOWAIT	DECIMAL
bit 3	bit 3	bit 2	bit 1	bitt 0	
1	0	1	0	0	20
0	0	1	0	0	4
0	1	1	0	0	12
1	0	1	1	0	22

psi_monitor: pressure high: **cpu=65% mem=0% io=19%** (thresh cpu=50 mem=50 io=50)
psi_monitor: logging **top 14 tasks** under pressure:
psi_monitor: pid=3544 **comm=cpu_hog_0** psi_flag=4 oncpu=2 **cputime(ms)=2283347** rss(kB)=399748 io(kB)=0 score=11616609
psi_monitor: pid=3545 comm=cpu_hog_1 psi_flag=4 oncpu=2 cputime(ms)=2282680 rss(kB)=399748 io(kB)=0 score=11613274
psi_monitor: pid=3546 **comm=mem_hog_0** psi_flag=4 oncpu=2 cputime(ms)=1024974 **rss(kB)=399748** io(kB)=0 score=5324744
psi_monitor: pid=3549 **comm=io_hog_1** psi_flag=0 oncpu=0 cputime(ms)=142733 rss(kB)=399748 **io(kB)=4296132** score=5209671
psi_monitor: pid=3548 comm=io_hog_0 psi_flag=0 oncpu=2 cputime(ms)=141822 rss(kB)=399748 io(kB)=4290384 score=5199368
psi_monitor: pid=239 comm=jbd2/mmcblk0p2- psi_flag=1 oncpu=1 cputime(ms)=113758 rss(kB)=0 io(kB)=224692 score=793482
psi_monitor: pid=7179 **comm=psi_cpu_hog** **psi_flag=20** oncpu=3 cputime(ms)=142904 **rss(kB)=0** io(kB)=0 score=714520
psi_monitor: pid=7290 comm=stress-ng psi_flag=4 oncpu=2 cputime(ms)=28095 rss(kB)=4088 io(kB)=0 score=142519
psi_monitor: pid=7289 comm=stress-ng psi_flag=4 oncpu=2 cputime(ms)=27226 rss(kB)=3972 io(kB)=0 score=138116
psi_monitor: pid=7423 comm=node psi_flag=4 oncpu=2 cputime(ms)=2553 rss(kB)=58984 io(kB)=0 score=42257
psi_monitor: pid=5632 comm=kworker/u16:2+**psi_hog_wq** psi_flag=20 oncpu=0 cputime(ms)=**3000** rss(kB)=0 io(kB)=0 score=15000
psi_monitor: pid=7292 comm=stress-ng psi_flag=0 oncpu=3 cputime(ms)=1411 rss(kB)=1688 io(kB)=4 score=7903
psi_monitor: pid=7291 comm=stress-ng psi_flag=4 oncpu=3 cputime(ms)=1396 rss(kB)=1680 io(kB)=0 score=7820
psi_monitor: pid=7080 comm=kworker/1:0+events **psi_flag=20** oncpu=1 cputime(ms)=721 rss(kB)=0 io(kB)=0 score=3605

👉 **PSI Auto Monitor accurately identifies top resource consumers during peak CPU pressure.**

Results-2

- **Machine: ARM64 ; Cortex-A55; CPU: 2 ; RAM: 1GB ; Storage: NAND ; Linux Kernel: 5.15.xx**
- **Scenario: Run CPU/memory/io stress test**

```
[366111.201912][ T5194] psi_monitor: pressure high: cpu=87% mem=64% io=17% (thresh cpu=50 mem=60 io=40)
[366111.214340][ T5194] psi_monitor: logging top 10 tasks under pressure:
[366111.225943][ T5194] psi_monitor: pid=15 comm=rcu_preempt psi_flag=12 oncpu=0 cputime(ms)=614042 rss(kB)=0 io(kB)=0 score=3070210
[366111.241945][ T5194] psi_monitor: pid=48 comm=kcompactd0 psi_flag=4 oncpu=0 cputime(ms)=96329 rss(kB)=0 io(kB)=0 score=481645
[366111.261965][ T5194] psi_monitor: pid=5318 comm=stress-ng psi_flag=22 oncpu=0 cputime(ms)=6715 rss(kB)=107180 io(kB)=32 score=87197
[366111.277932][ T5194] psi_monitor: pid=5319 comm=stress-ng psi_flag=4 oncpu=0 cputime(ms)=6852 rss(kB)=99700 io(kB)=32 score=84142
[366111.297945][ T5194] psi_monitor: pid=5311 comm=stress-ng psi_flag=4 oncpu=1 cputime(ms)=6597 rss(kB)=1484 io(kB)=5260 score=38987
[366111.313923][ T5194] psi_monitor: pid=5313 comm=stress-ng psi_flag=4 oncpu=1 cputime(ms)=6282 rss(kB)=2460 io(kB)=3816 score=36456
[366111.329921][ T5194] psi_monitor: pid=5317 comm=stress-ng psi_flag=4 oncpu=0 cputime(ms)=6815 rss(kB)=40 io(kB)=0 score=34095
[366111.358091][ T5194] psi_monitor: pid=5312 comm=stress-ng psi_flag=0 oncpu=1 cputime(ms)=6071 rss(kB)=1648 io(kB)=2832 score=34011
[366111.374022][ T5194] psi_monitor: pid=5316 comm=stress-ng psi_flag=4 oncpu=0 cputime(ms)=6723 rss(kB)=40 io(kB)=16 score=33651
[366111.390036][ T5194] psi_monitor: pid=5194 comm=kworker/1:0+events psi_flag=12 oncpu=1 cputime(ms)=4429 rss(kB)=0 io(kB)=0 score=22145
[366121.450111][ T5331] psi_monitor: pressure high: cpu=86% mem=57% io=12% (thresh cpu=50 mem=60 io=40)
[366121.478345][ T5331] psi_monitor: logging top 10 tasks under pressure:
[366121.498066][ T5331] psi_monitor: pid=67 comm=kswapd0 psi_flag=22 oncpu=1 cputime(ms)=34356 rss(kB)=0 io(kB)=0 score=171780
[366121.529939][ T5331] psi_monitor: pid=5320 comm=mem_burst psi_flag=22 oncpu=0 cputime(ms)=4514 rss(kB)=152024 io(kB)=0 score=98582
[366121.557935][ T5331] psi_monitor: pid=5319 comm=stress-ng psi_flag=12 oncpu=0 cputime(ms)=8339 rss(kB)=74528 io(kB)=96 score=79055
[366121.581913][ T5331] psi_monitor: pid=5318 comm=stress-ng psi_flag=22 oncpu=0 cputime(ms)=7923 rss(kB)=76256 io(kB)=32 score=77775
[366121.608157][ T5331] psi_monitor: pid=5311 comm=stress-ng psi_flag=4 oncpu=0 cputime(ms)=7650 rss(kB)=1776 io(kB)=6048 score=45186
[366121.638048][ T5331] psi_monitor: pid=5313 comm=stress-ng psi_flag=4 oncpu=0 cputime(ms)=7577 rss(kB)=1628 io(kB)=5092 score=43791
[366121.657977][ T5331] psi_monitor: pid=5317 comm=stress-ng psi_flag=4 oncpu=0 cputime(ms)=8044 rss(kB)=36 io(kB)=0 score=40238
[366121.681894][ T5331] psi_monitor: pid=5316 comm=stress-ng psi_flag=4 oncpu=0 cputime(ms)=7994 rss(kB)=36 io(kB)=16 score=40004
[366121.705951][ T5331] psi_monitor: pid=5312 comm=stress-ng psi_flag=12 oncpu=0 cputime(ms)=7023 rss(kB)=1952 io(kB)=2912 score=39003
[366121.737934][ T5331] psi_monitor: pid=5331 comm=kworker/1:6+events psi_flag=12 oncpu=1 cputime(ms)=1311 rss(kB)=0 io(kB)=0 score=6555
```

```
# free -m
               total        used        free      shared  buff/cache   available
Mem:             620         564          14           0          43          37
Swap:            465         246         219
```

```
# cat /proc/pressure/cpu
some avg10=85.53 avg60=44.20 avg300=13.02 total=7399971680
full avg10=0.00 avg60=0.00 avg300=0.00 total=0
# cat /proc/pressure/memory
some avg10=59.88 avg60=26.60 avg300=6.76 total=49623699
full avg10=24.38 avg60=10.90 avg300=2.77 total=26007723
# cat /proc/pressure/io
some avg10=19.60 avg60=8.06 avg300=2.03 total=13068522
full avg10=0.00 avg60=0.00 avg300=0.00 total=379648
```

```
# top -n 1
Mem: 619636K used, 15420K free, 64K shrd, 0K buff, 10128K cached
CPU: 20.6% usr 79.3% sys  0.0% nic  0.0% idle  0.0% io  0.0% irq  0.0% sirq
Load average: 6.33 2.65 1.97 10/1013 5345
  PID  PPID USER     STAT   VSZ %VSZ CPU %CPU COMMAND
   67     2 root     RW       0  0.0   0 33.3 [kswapd0]
 5345  1554 root     R     4524  0.7   1  9.9 {top} /bin/busybox.nosuid /usr/bin
 5320     1 root     R<    200m 32.3   1  6.6 ./mem_burst
 5317  5307 root     R<   18504  2.9   1  6.6 stress-ng --cpu 3 --vm 2 --io 2 --
 5316  5307 root     R<   18504  2.9   1  6.6 stress-ng --cpu 3 --vm 2 --io 2 --
 5311  5307 root     R<   18504  2.9   1  6.6 stress-ng --cpu 3 --vm 2 --io 2 --
 5313  5307 root     R<   18504  2.9   1  6.6 stress-ng --cpu 3 --vm 2 --io 2 --
 5312  5307 root     R<   18504  2.9   1  6.6 stress-ng --cpu 3 --vm 2 --io 2 --
 5319  5315 root     R<    146m 23.5   0  3.3 stress-ng --cpu 3 --vm 2 --io 2 --
 5318  5314 root     R<    146m 23.5   1  3.3 stress-ng --cpu 3 --vm 2 --io 2 --
  354     1 root     S     9124  1.4   1  3.3 /lib/systemd/systemd-udevd
   15     2 root     IW       0  0.0   1  3.3 [rcu_preempt]
   14     2 root     SW       0  0.0   0  3.3 [ksoftirqd/0]
 2428     1 system   S    2343m377.1   1  0.0 /usr/bin/qwesd
  336     1 radio    S    2287m368.2   1  0.0 /usr/bin/netmgrd
```

Results-3

- **Machine: ARM64 ; Cortex-A55 ; CPU: 4 ; RAM: 1GB ; Storage: NAND ; Linux Kernel: 5.15.xx**
- **Scenario: During boot-up (without any tool)**

```
~ # dmesg | grep psi
psi_monitor: in-kernel PSI auto monitor (weighted + tracepoints) loaded
psi_monitor: pressure high: cpu=50% mem=1% io=11% (thresh cpu=50 mem=60 io=40)
psi_monitor: logging top 8 tasks under pressure:
psi_monitor: pid=2637 comm=pcid-loc-api psi_flag=12 oncpu=0 cputime(ms)=443 rss(kB)=21976 io(kB)=0 score=13203
psi_monitor: pid=1974 comm=syslogd psi_flag=0 oncpu=3 cputime(ms)=643 rss(kB)=2156 io(kB)=272 score=4565
psi_monitor: pid=2597 comm=qwesd psi_flag=4 oncpu=3 cputime(ms)=104 rss(kB)=7964 io(kB)=0 score=4502
psi_monitor: pid=1493 comm=dlt-system psi_flag=0 oncpu=3 cputime(ms)=370 rss(kB)=1984 io(kB)=136 score=2978
psi_monitor: pid=15 comm=rcu_preempt psi_flag=0 oncpu=2 cputime(ms)=443 rss(kB)=0 io(kB)=0 score=2215
psi_monitor: pid=420 comm=kworker/2:9+events psi_flag=12 oncpu=2 cputime(ms)=351 rss(kB)=0 io(kB)=0 score=1755
psi_monitor: pid=2713 comm=sdir psi_flag=0 oncpu=1 cputime(ms)=124 rss(kB)=0 io(kB)=20 score=640
psi_monitor: pid=41 comm=kworker/u8:1-memlat_wq psi_flag=0 oncpu=3 cputime(ms)=52 rss(kB)=0 io(kB)=0 score=260
```

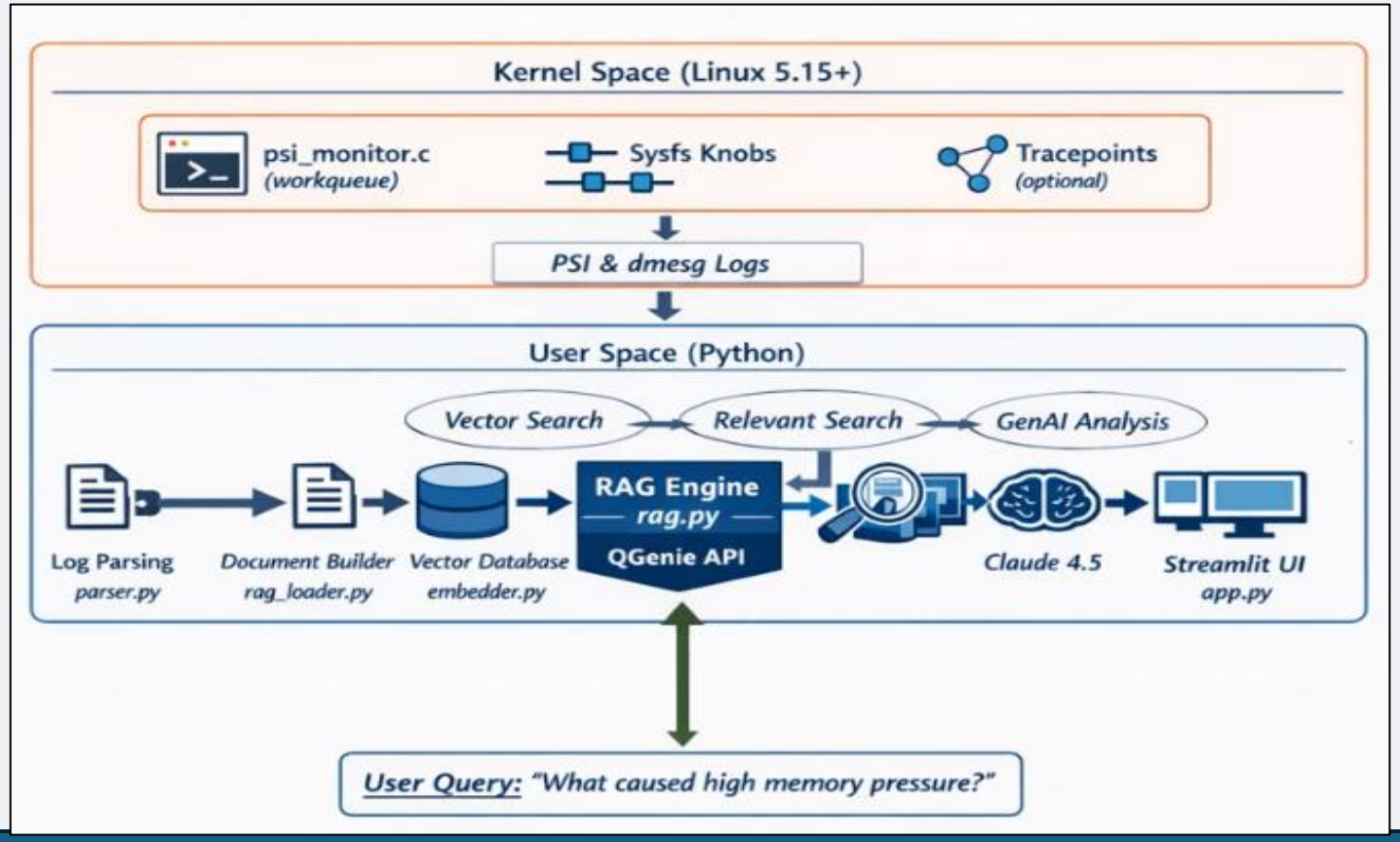
👉 **Even during boot, PSI Auto Monitor detects transient hotspots without external tools.**

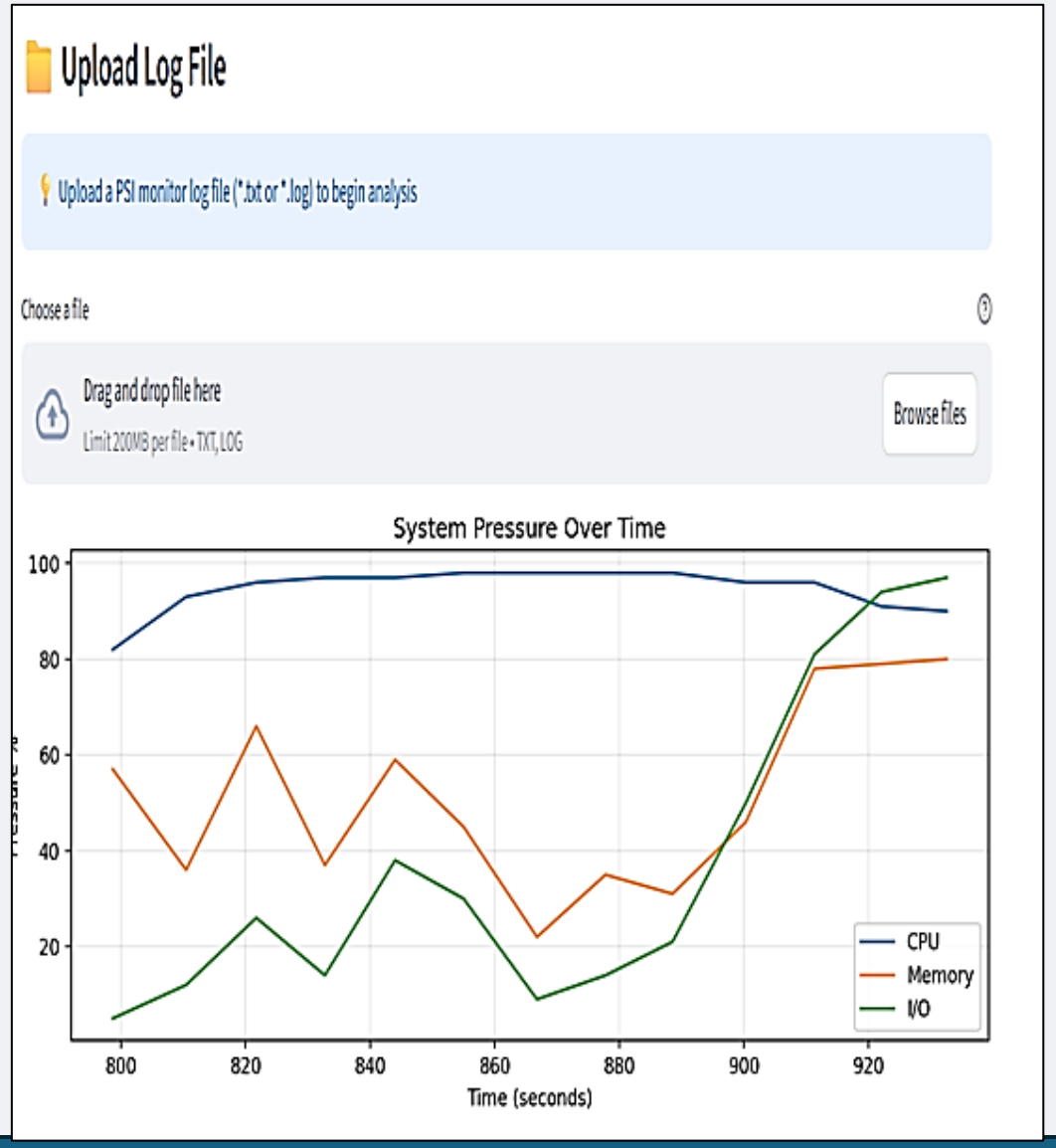
Trace Output

kworker/1:1-307455 [001] 194092.074586: psi_monitor_top_task: pid=239 comm=jbd2/mmcblk0p2- cpu_ms=31458 rss_kb=0 io_kb=510752 score=668042
kworker/1:1-307455 [001] 194092.371423: psi_monitor_top_task: pid=60 comm=kswapd0 cpu_ms=3184 rss_kb=0 io_kb=0 score=15920
kworker/1:1-307455 [001] 194092.388191: psi_monitor_top_task: pid=311145 comm=kworker/u16:2+psi_hog_wq **cpu_ms=1119** rss_kb=0 io_kb=6288 score=11883
kworker/1:1-307455 [001] 194092.400643: psi_monitor_top_task: pid=312843 comm=stress-ng cpu_ms=1547 rss_kb=1748 io_kb=4 score=8613
kworker/1:1-307455 [001] 194092.416480: psi_monitor_top_task: pid=307455 **comm=kworker/1:1+events** **cpu_ms=977** rss_kb=0 io_kb=0 score=4885
kworker/1:1-307455 [001] 194102.572312: psi_monitor_top_task: pid=239 comm=jbd2/mmcblk0p2- cpu_ms=31743 rss_kb=0 io_kb=513912 score=672627
kworker/1:1-307455 [001] 194102.602736: psi_monitor_top_task: pid=312854 comm=cpu_hog_1 cpu_ms=9557 rss_kb=331572 io_kb=0 score=213571
kworker/1:1-307455 [001] 194102.628326: psi_monitor_top_task: pid=312853 comm=cpu_hog_0 cpu_ms=7948 rss_kb=331572 io_kb=0 score=205526
kworker/1:1-307455 [001] 194102.668315: psi_monitor_top_task: pid=312855 comm=mem_hog_0 cpu_ms=5382 rss_kb=331572 io_kb=0 score=192696
kworker/1:1-307455 [001] 194102.712311: psi_monitor_top_task: pid=312857 comm=io_hog_0 cpu_ms=663 rss_kb=331572 io_kb=8720 score=177821
kworker/1:1-307455 [001] 194102.744720: psi_monitor_top_task: pid=312840 comm=stress-ng cpu_ms=10409 rss_kb=127600 io_kb=0 score=115845
kworker/1:1-307455 [001] 194102.772663: psi_monitor_top_task: pid=312841 comm=stress-ng cpu_ms=9684 rss_kb=132464 io_kb=0 score=114652
kworker/1:1-307455 [001] 194102.800355: psi_monitor_top_task: pid=312894 comm=psi_cpu_hog cpu_ms=18633 rss_kb=0 io_kb=0 score=93165
[...]
kworker/1:1-307455 [001] 194113.051632: psi_monitor_top_task: pid=312854 comm=cpu_hog_1 cpu_ms=11830 rss_kb=486784 io_kb=0 score=302542
kworker/1:1-307455 [001] 194113.068361: psi_monitor_top_task: pid=312853 comm=cpu_hog_0 cpu_ms=10735 rss_kb=486784 io_kb=0 score=297067
kworker/1:1-307455 [001] 194113.079961: psi_monitor_top_task: pid=312856 comm=mem_hog_1 cpu_ms=6681 rss_kb=486784 io_kb=0 score=276797
kworker/1:1-307455 [001] 194113.096329: psi_monitor_top_task: pid=312894 comm=psi_cpu_hog cpu_ms=26634 rss_kb=0 io_kb=0 score=133170
kworker/1:1-307455 [001] 194113.154753: psi_monitor_top_task: pid=307455 **comm=kworker/1:1+events** **cpu_ms=1404** rss_kb=0 io_kb=0 score=7020
kworker/1:1-307455 [001] 194123.291079: psi_monitor_top_task: pid=239 comm=jbd2/mmcblk0p2- cpu_ms=32324 rss_kb=0 io_kb=520040 score=681660
kworker/1:1-307455 [001] 194123.302840: psi_monitor_top_task: pid=312894 comm=psi_cpu_hog cpu_ms=36599 rss_kb=0 io_kb=0 score=182995
kworker/1:1-307455 [001] 194123.314151: psi_monitor_top_task: pid=312854 comm=cpu_hog_1 cpu_ms=15582 rss_kb=186056 io_kb=0 score=170938
kworker/1:1-307455 [001] 194123.326410: psi_monitor_top_task: pid=312853 comm=cpu_hog_0 cpu_ms=15158 rss_kb=186056 io_kb=0 score=168818
kworker/1:1-307455 [001] 194123.350721: psi_monitor_top_task: pid=312856 comm=mem_hog_1 cpu_ms=8622 rss_kb=186056 io_kb=0 score=136138
kworker/1:1-307455 [001] 194123.362811: psi_monitor_top_task: pid=312841 comm=stress-ng cpu_ms=13234 rss_kb=132456 io_kb=0 score=132398
kworker/1:1-307455 [001] 194123.375056: psi_monitor_top_task: pid=312857 comm=io_hog_0 cpu_ms=880 rss_kb=186056 io_kb=11960 score=109388
kworker/1:1-307455 [001] 194123.423022: psi_monitor_top_task: pid=307455 **comm=kworker/1:1+events** **cpu_ms=1523** rss_kb=0 io_kb=0 score=7615

GEN-AI Integration

- Kernel detects PSI threshold, top contributing tasks are logged
- Logs are parsed into structured events and fed into a vector database
- GenAI analyzes and explains pressure patterns
- Users query the system using natural language





GEN-AI Report

Resource Pressure Trends

- **CPU**: Steadily deteriorating, reaching 87% pressure—system is CPU-bound with minimal headroom
- **Memory**: Most concerning trend, more than doubling to 64% — approaching OOM conditions
- **IO**: Moderate but persistent (12-17%), suggesting storage subsystem cannot keep pace with demand

Processes Causing Maximum Pressure

1. **stress-ng instances** (PIDs 5318, 5319): Primary workload generators
2. **rcu_preempt** (PID 15): Kernel RCU mechanism overwhelmed by scheduling demands
3. **kswapd0** (PID 67): daemon working overtime

Overall System Health Assessment

- The system is experiencing **severe resource contention** across all metrics during the monitored **52-second period**.
- With 6 consecutive pressure events and no recovery intervals, the system is operating in a critically stressed state that requires immediate intervention.

Risk: Without intervention, system will likely experience OOM kills within minutes to hour.

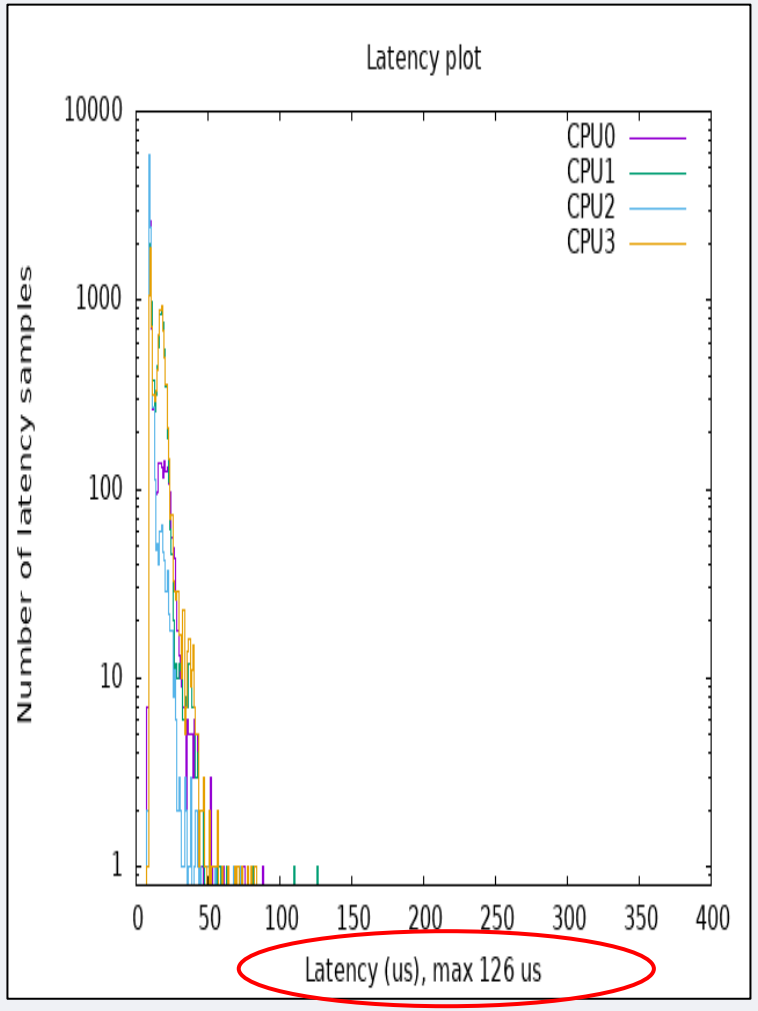
"Gen-AI is an optional post-processing layer built on top of PSI Auto Monitor logs."

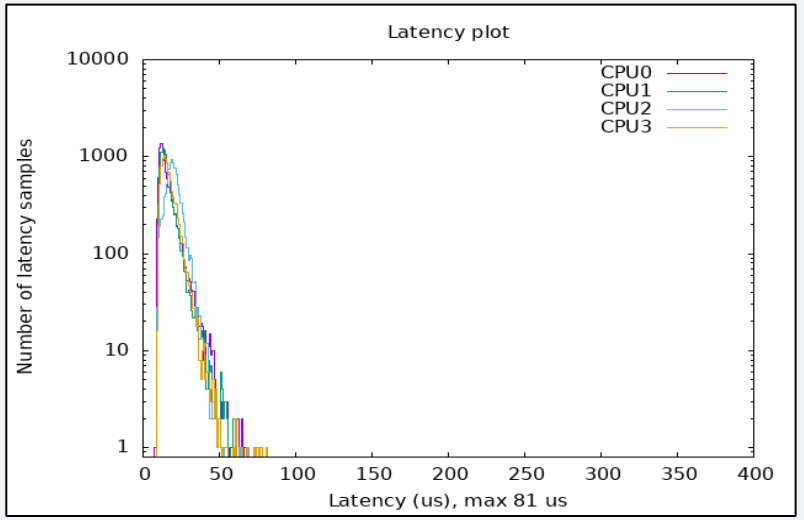
Results (Logs to Table)

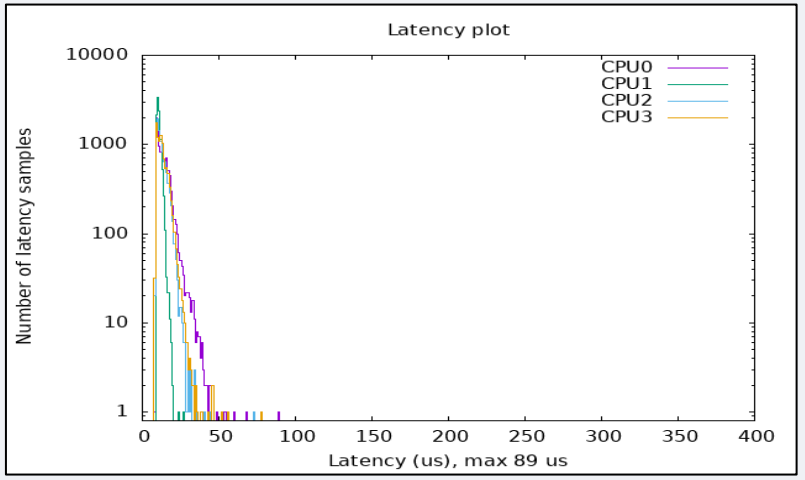
PID	comm	psi_flag	oncpu	cputime (ms)	rss (kB)	io (kB)	score
3544	**cpu_hog_0**	4	2	**2,283,347**	399,748	0	11,616,609
3545	cpu_hog_1	4	2	2,282,680	399,748	0	11,613,274
3546	**mem_hog_0**	4	2	1,024,974	**399,748**	0	5,324,744
3549	**io_hog_1**	0	0	142,733	399,748	**4,296,132**	5,209,671
3548	io_hog_0	0	2	141,822	399,748	4,290,384	5,199,368
239	jbd2/mmcblk0p2-	1	1	113,758	0	224,692	793,482
7179	**psi_cpu_hog**	**20**	3	142,904	**0**	0	714,520
7290	stress-ng	4	2	28,095	4,088	0	142,519
7289	stress-ng	4	2	27,226	3,972	0	138,116
7423	node	4	2	2,553	58,984	0	42,257
5632	kworker/u16:2+**psi_hog_wq**	20	0	**3,000**	0	0	15,000
7292	stress-ng	0	3	1,411	1,688	4	7,903
7291	stress-ng	4	3	1,396	1,680	0	7,820
7080	kworker/1:0+events	20	1	721	0	0	3,605

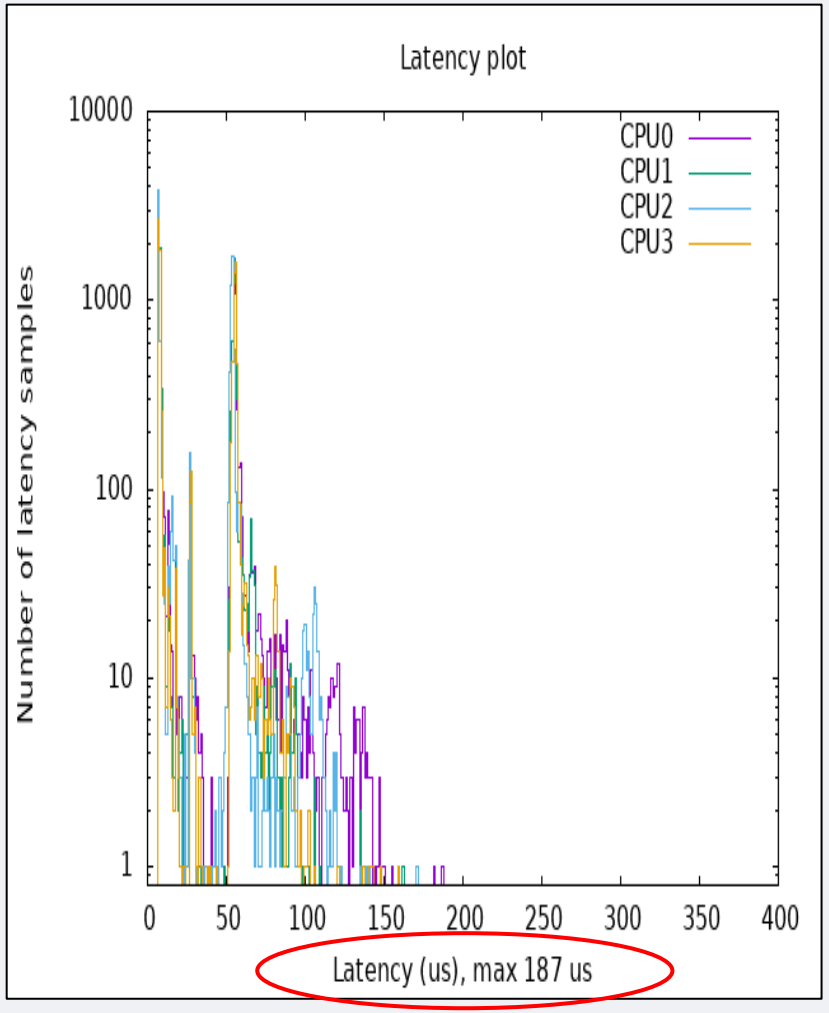
- ***cpu_hog_0*** *has the highest* ***cputime*** *(2,283,347 ms)*
- ***mem_hog_0*** *has the highest* ***rss*** *(399,748 kB)*
- ***io_hog_1*** *has the highest* ***io*** *(4,296,132 kB)*
- ***psi_cpu_hog*** *has* ***psi_flag=20*** *and* ***rss=0****, indicating a PSI stall monitor process*
- ***kworker/u16:2+psi_hog_wq*** *also has* ***psi_flag=20*** *and a notably low* ***cputime*** *(3,000 ms)*

Results – PREEMPT_RT CyclicTest









"👉 PSI Auto Monitor helps correlate latency spikes with system pressure in real-time workloads."

Benefits

- Near-zero overhead: No user intervention or tools needed to monitor load.
- All in one: Does not need multiple tools to monitor cpu/memory/io usage.
- Helps in detecting sudden spikes along with neighboring tasks.
- Early detection of culprit tasks before the issue occurs.
- Portable, tunable, lightweight and deployment across platforms.
- Helps identify bottle necks during boot-up and aid for optimization.
- Faster debugging, RCA, better observability, and reduced manual analysis effort.

PSI Auto Monitor transforms Linux pressure signals into an early-warning and automated control system for stable, resilient performance.

Roadmap and Plan

- Linux Kernel upstream contribution and preparation in progress.
- First RFC patch will be posted soon.
- A reference patch and test details are available here:
  - https://github.com/pintuk/KERNEL/tree/master/PSI_WORK
- Collects data on various boards with real workloads and use cases.
- Find ways to identify latency issues with PREEMPT_RT Kernel.
- Publish more real workload samples and results in GitHub.
- Default PSI window (10s) is too coarse for embedded/IoT — exploring finer-grained trigger integration.
- Include IRQ pressure metrics as well.
- Auto monitor within cgroups level as well.

Summary

- PSI is excellent in detecting contention issue but needs external intervention.
- PSI auto monitor converts PSI into automated, real-time monitoring.
- It also helps in early detection of CPU, Memory, and IO pressure.
- Identifies top contributing tasks without user intervention.
- Works across platforms (embedded → automotive → server).
- Can be easily applied to any kernel (5.x+) [Just 2-3 patches].
- Improves debugging efficiency and system observability.
- Provides foundation for AI-driven performance analysis.

Contact Me

- **Email: pintu.ping@gmail.com**
- **LinkedIn: www.linkedin.com/in/pintu-k-agarwal-b73a31b**
- **GitHub Profile: https://github.com/pintuk**
- **LF Profile: https://openprofile.dev/profile/pintuk**
- **You can find my past papers here: https://github.com/pintuk/CONFERENCE**
- **More details about my PSI work: https://github.com/pintuk/KERNEL/tree/master/PSI_WORK**

Thank You!

Questions & Discussion

Pintu Kumar Agarwal | pintu.ping@gmail.com