« Understanding Embedded Linux Benchmarking Using Kernel Trace Analysis »

Alexis Martin

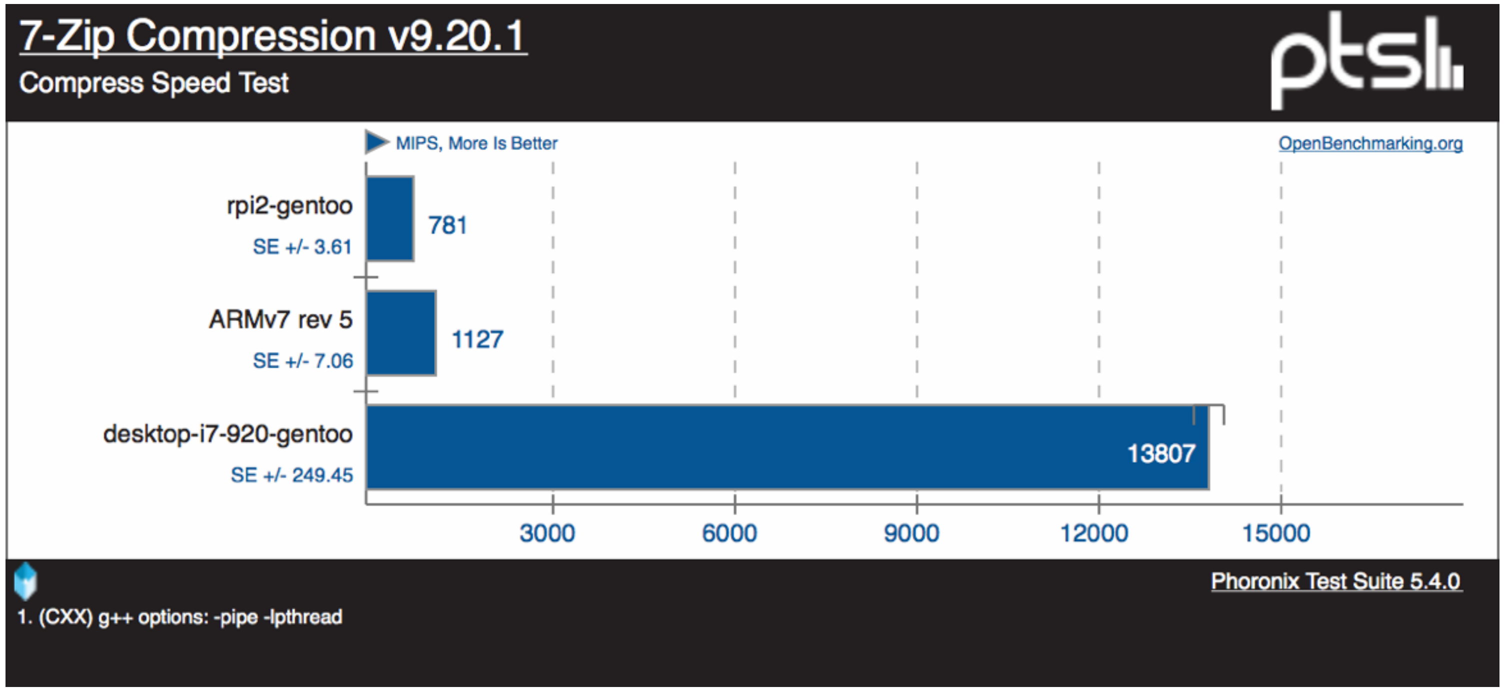
Inria / LIG / Univ. Grenoble, France

We do Need Benchmarking !

- **Benchmark** : a **standard** or point of **reference** against which things may be **compared** or assessed.
  (new Oxford American Dictionary)

- Benchmarking **computer systems**:
  - **Asses** performance in different execution settings
  - **Compare** computer systems

- **Performance** criteria:
  - speed, latency, bandwidth, power consumption, memory used, …



- **Critical** step in system design

Benchmarking is Challenging

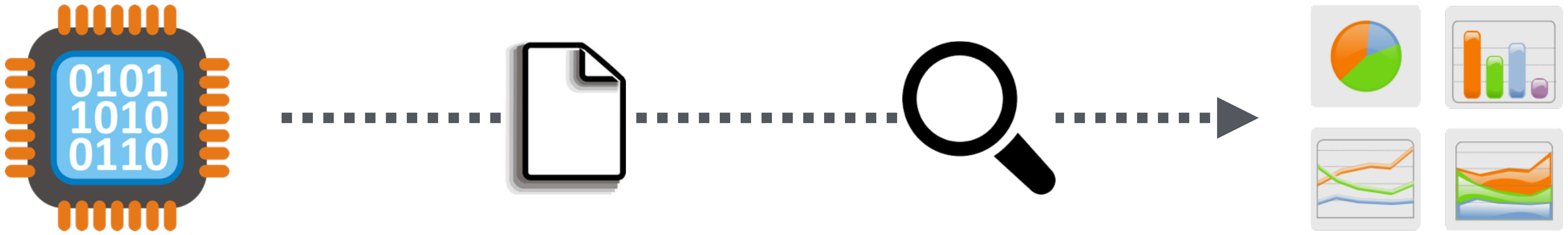
- Benchmarking construction is **difficult**
- There are **many different** benchmarks available
  - 3D rendering, DBMS test, NAS…
- In some cases benchmark is **nonexistent**
- Major motivation for using a benchmark is **popularity**
- The behavior of tests is **not necessarily known**

Understand What We Benchmark

- **Identify** what is measured and how
- **Interpret results**
- Draw a **profile**
- **Compare** different benchmarks

➜ **Help** to **chose** the right benchmark

Work Summary



1. **Execute** benchmark application (UDOO+Phoronix)
2. **Record** a trace from this execution (LTTng)
3. **Analyze** the traces (Framesoc + TraceCompass)
4. Draw a **profile** and **compare** benchmarks

Phoronix Test Suite for Benchmarking

- **Phoronix Test Suite** (PTS) is an **open-source** platform (openbenchmarking.org)
  - It contains **various** tests (over **170**)
  - PTS is **cross-platform** (i686, x86_64, ARM, PowerPC)
  - It includes every **mechanism** for **automated** tests
  - Result **sharing** for statistics and platform **comparisons**
- Tests are classified into **families**:

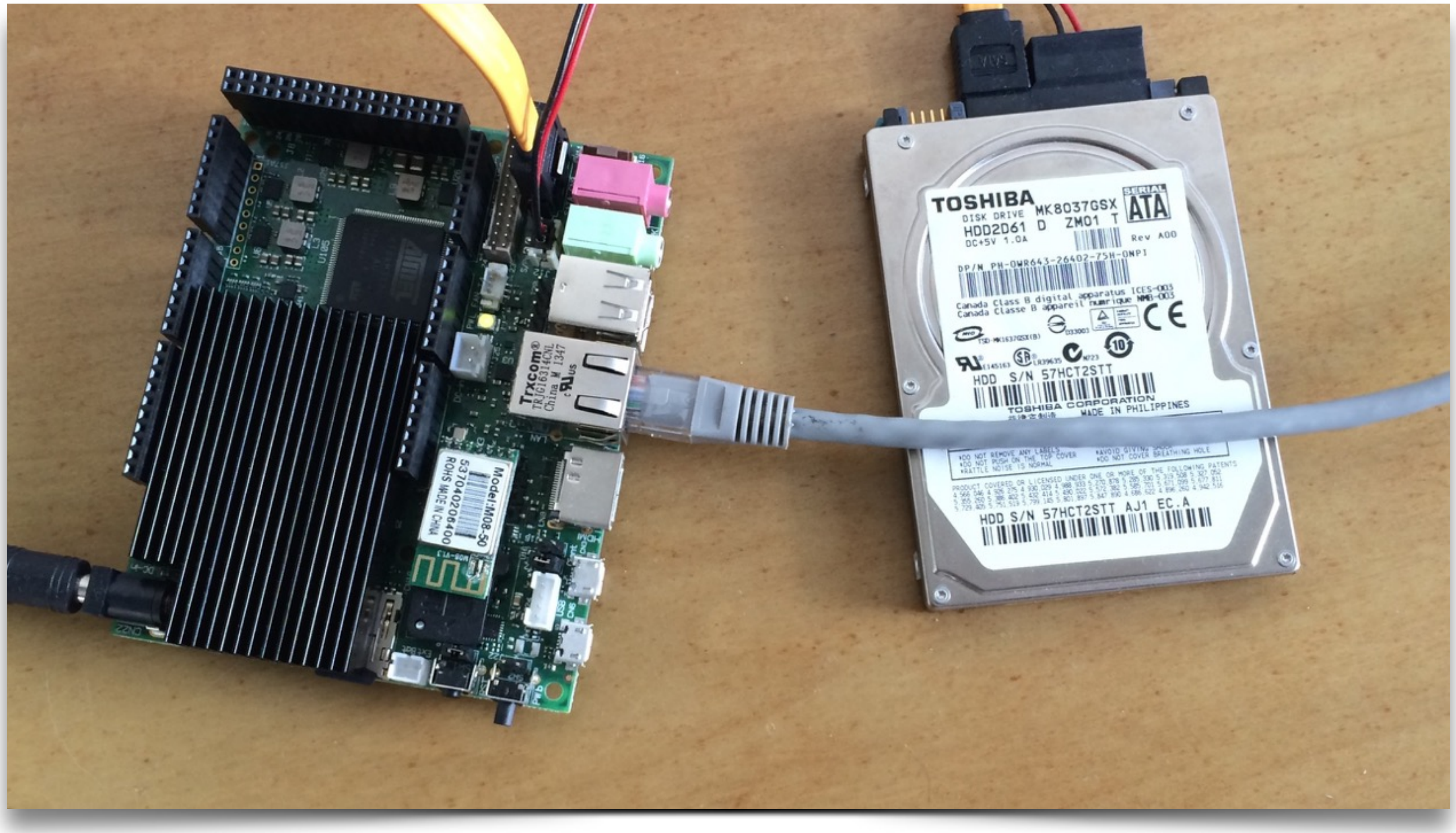
	System	Processor	Network	Memory	Graphics	Disk
# tests	6	79	1	2	53	12

Benchmark Selection

- Select 10 tests from 5 **different** families
- Use « **recommended** » tests from PTS
    - Calculated from **most used** tests

system	: **idle**, **pybench**, **phpbench**
processor	: **scimark2**, **ffmpeg**, **compress-gzip**
network	: **network-loopback**
memory	: **stream**, **ramspeed**
disk	: **dbench**

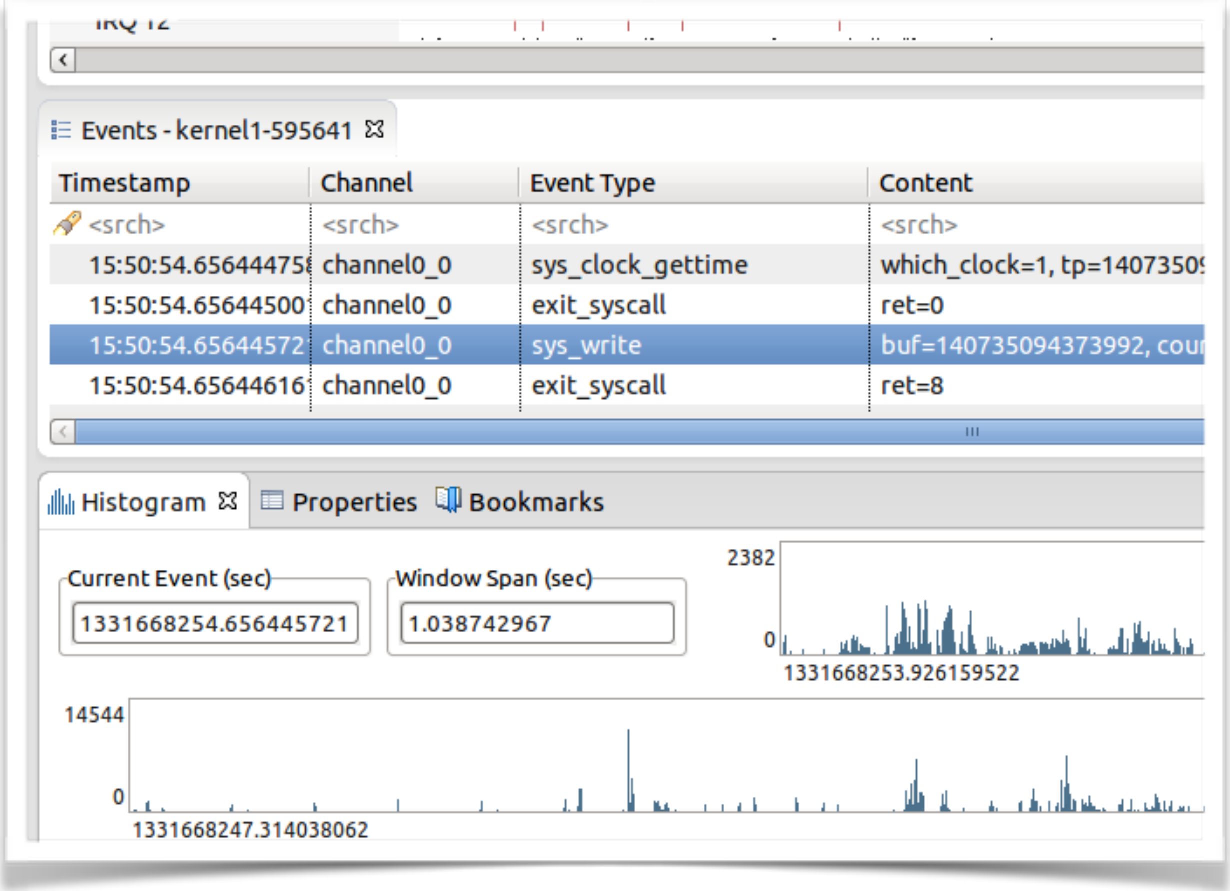
The Test Platform



- **UDOO** development board (udoo.org)
- **i.MX 6 Quad** ARM CPU (A9) @1GHz + 1 coprocessor (Cortex-M3)
- 1GB RAM, WiFi, Gigabit ethernet, HDMI, microSD, SATA
- Touchscreen, camera, GPIO
- **Debian** ARM kernel (**armmp** 3.16)

Tracing With LTTng

- **LTTng** (lttng.org) **open-source** tracing framework:
    - Trace **engine**:
        - **kernel-space**: **kprobes** & kernel **tracepoints**
        - **user-space**: **user implemented** tracepoints
    - **Viewing** and **analyzing**: Trace compass (eclipse)

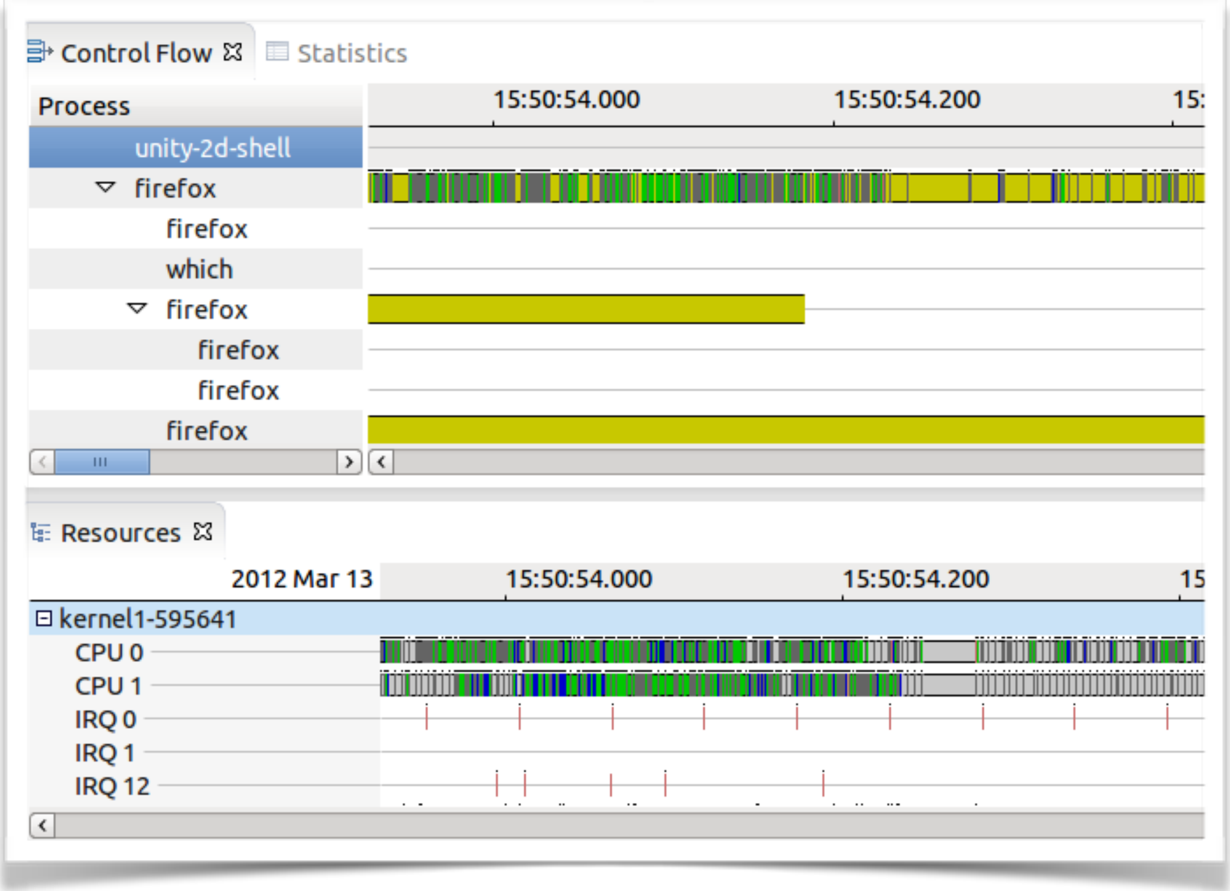


Events - kernel1-595641

Timestamp	Channel	Event Type	Content
<srch>	<srch>	<srch>	<srch>
15:50:54.65644475	channel0_0	sys_clock_gettime	which_clock=1, tp=1407350
15:50:54.65644500	channel0_0	exit_syscall	ret=0
15:50:54.65644572	channel0_0	sys_write	buf=140735094373992, cou
15:50:54.65644616	channel0_0	exit_syscall	ret=8

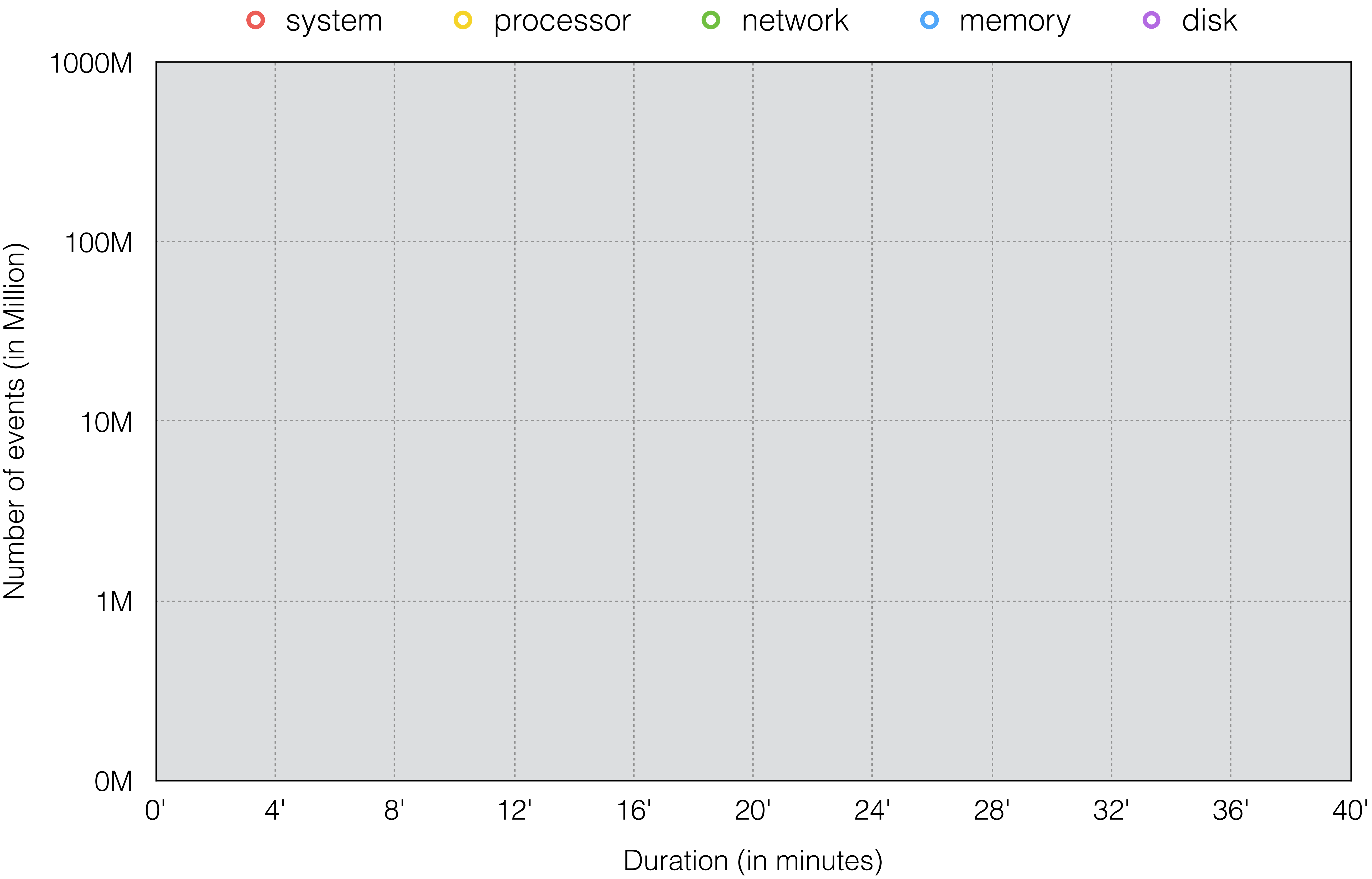
Current Event (sec): 1331668254.656445721

Window Span (sec): 1.038742967

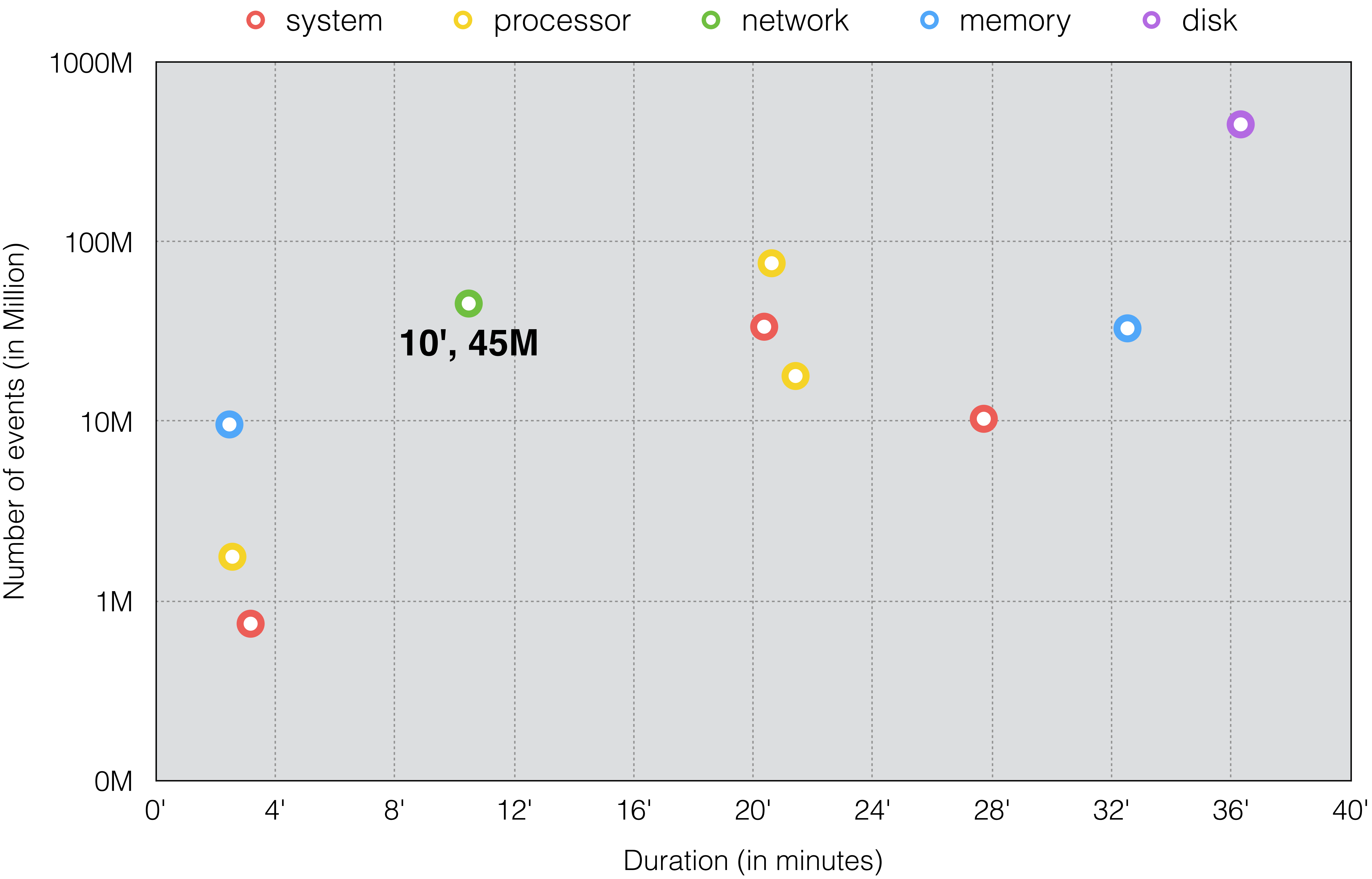


- Trace only the **kernel** to **avoid** benchmark code **modifications**

Trace Properties



Trace Properties

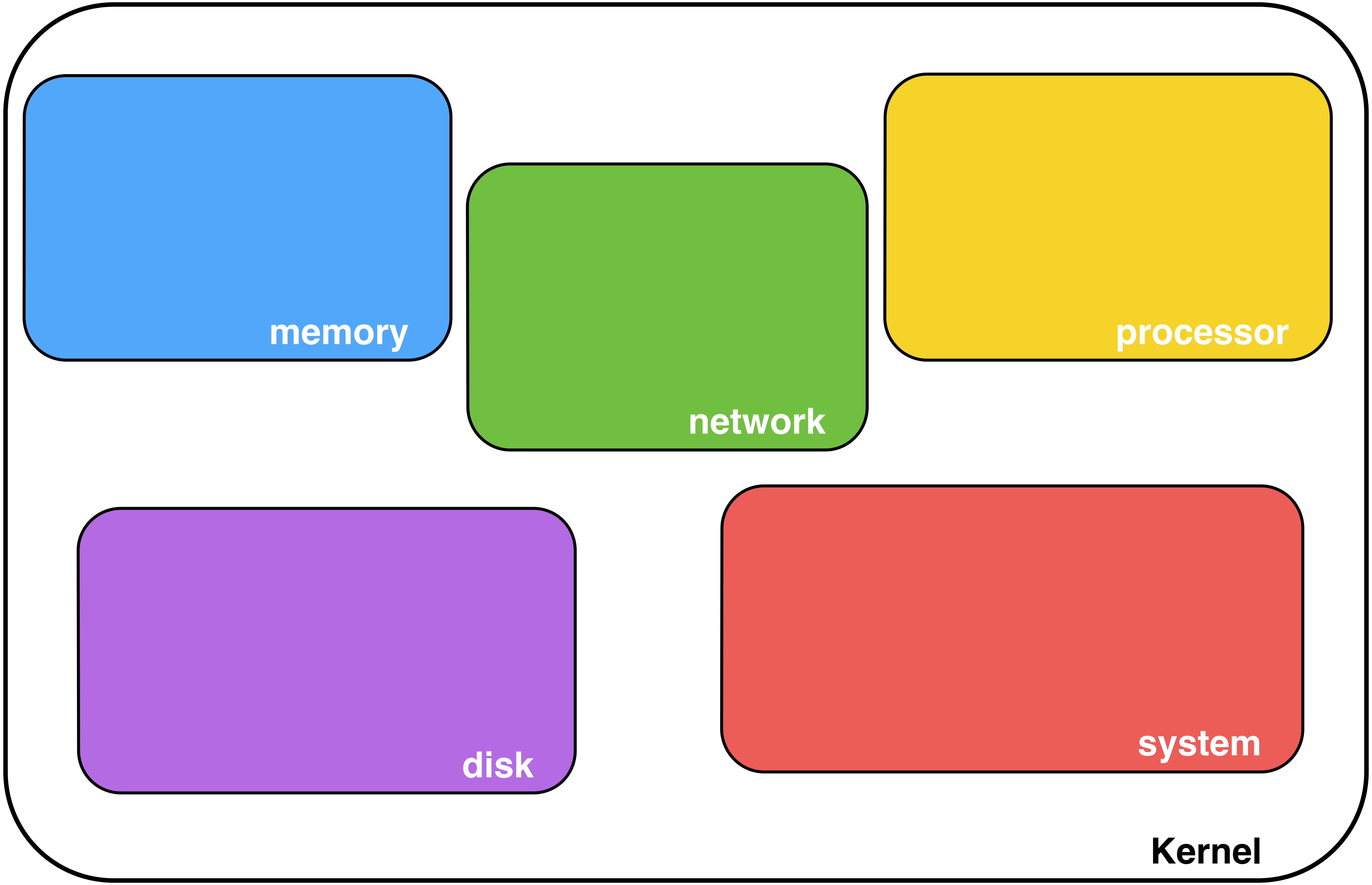


What does the Given Family Mean ?

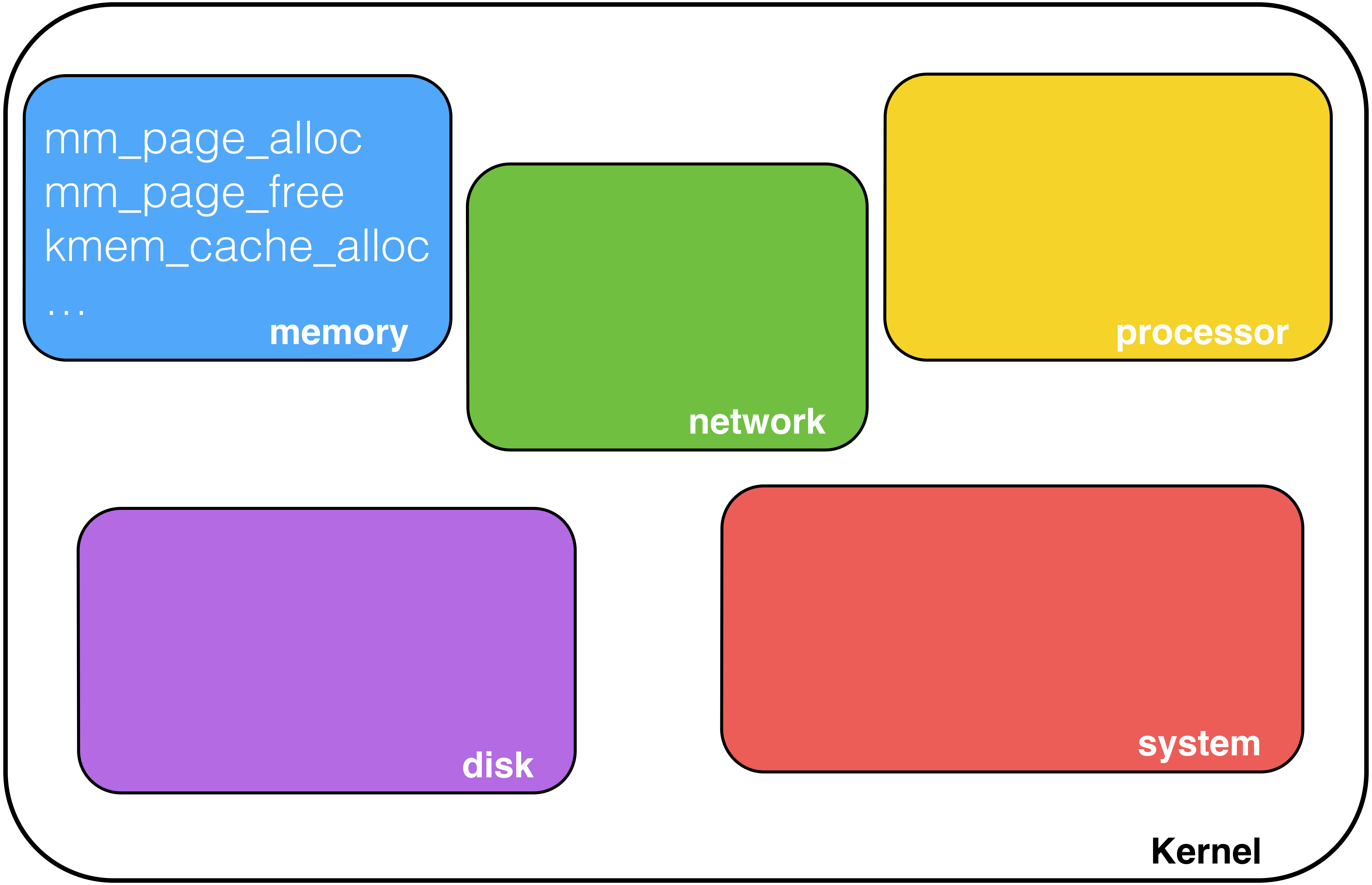
- Phoronix gives us a family **without** explanations
- Families are related to **kernel functionalities**
- **Compute** family:
  - **Biggest number** of events ?

➜ We want to check if the **announced** family **corresponds** to the **computed** one

Assigning Family to Events

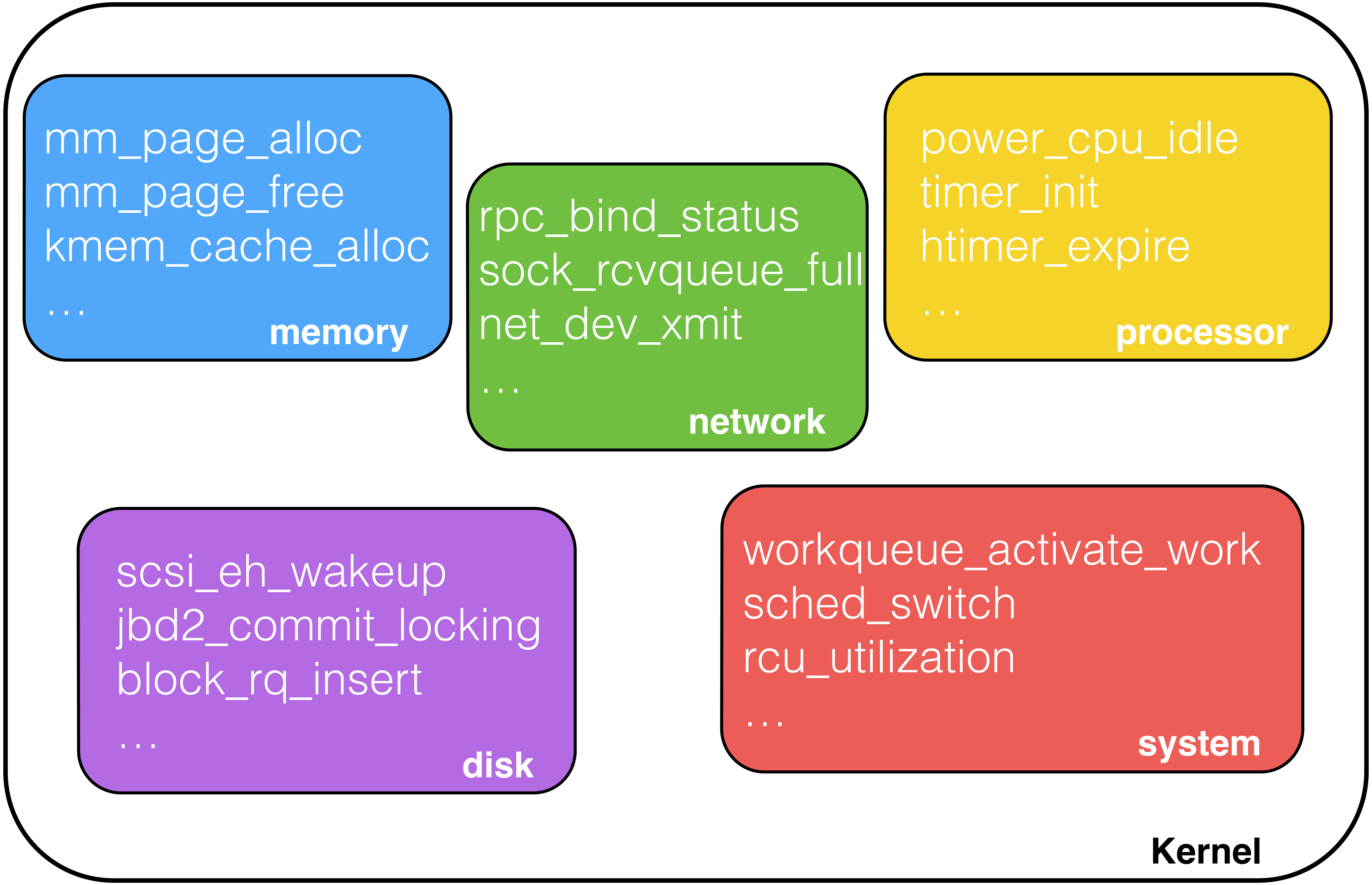


Assigning Family to Events



mm_page_alloc
mm_page_free
kmem_cache_alloc
…

Assigning Family to Events



mm_page_alloc
mm_page_free
kmem_cache_alloc
…
memory

rpc_bind_status
sock_rcvqueue_full
net_dev_xmit
…
network

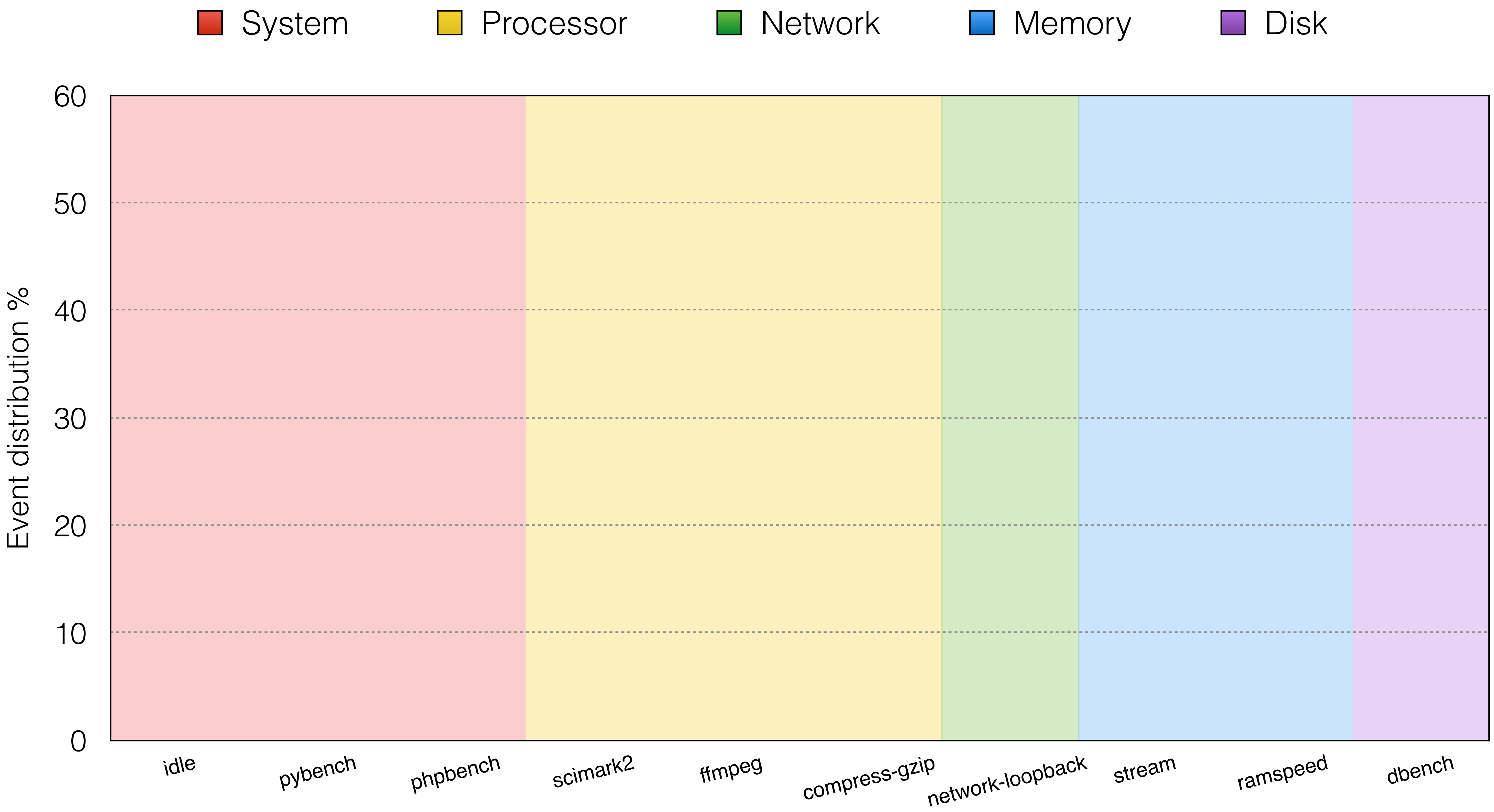
power_cpu_idle
timer_init
htimer_expire
…
processor

scsi_eh_wakeup
jbd2_commit_locking
block_rq_insert
…
disk

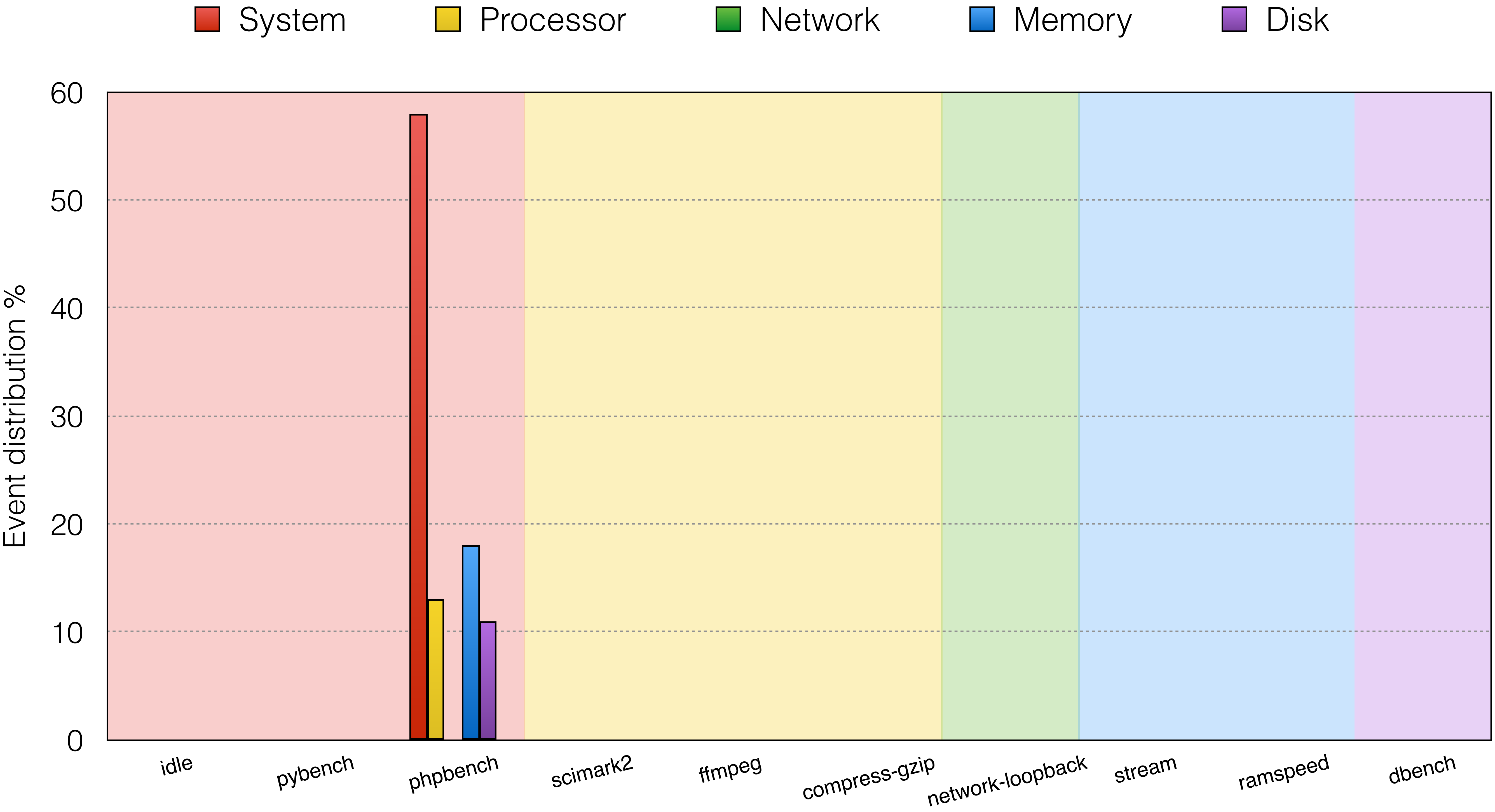
workqueue_activate_work
sched_switch
rcu_utilization
…
system

Kernel

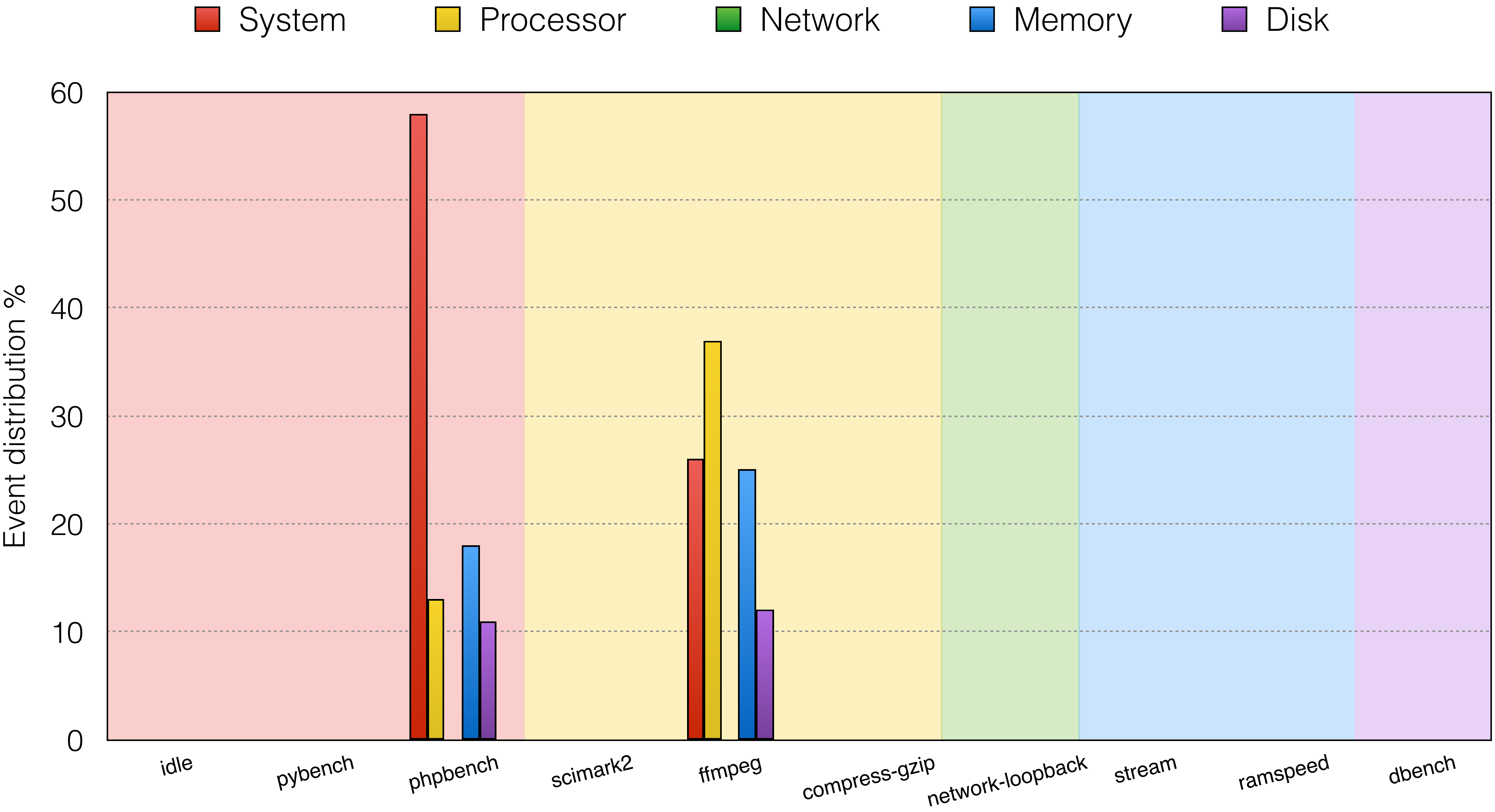
Family Distribution



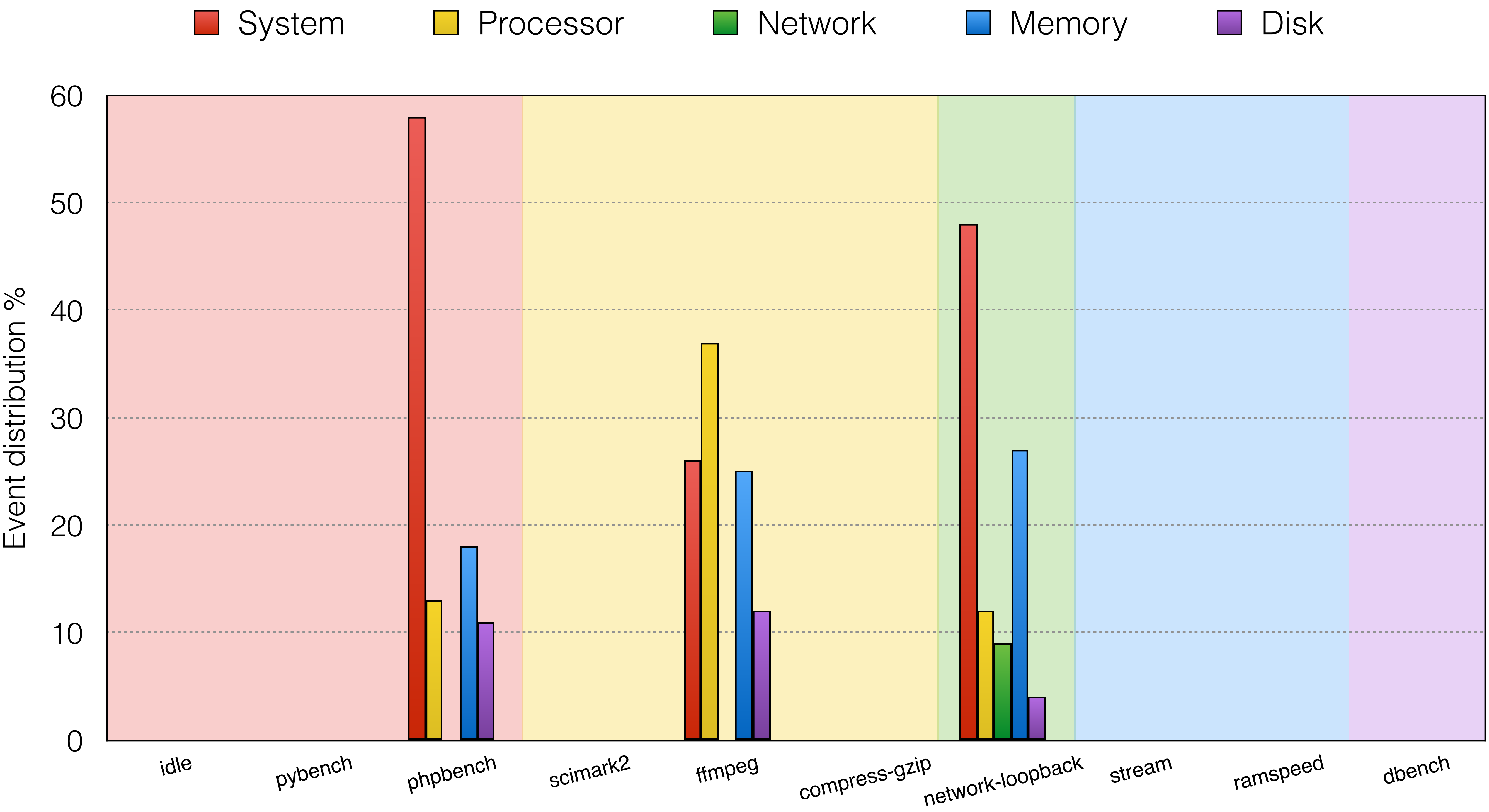
Family Distribution



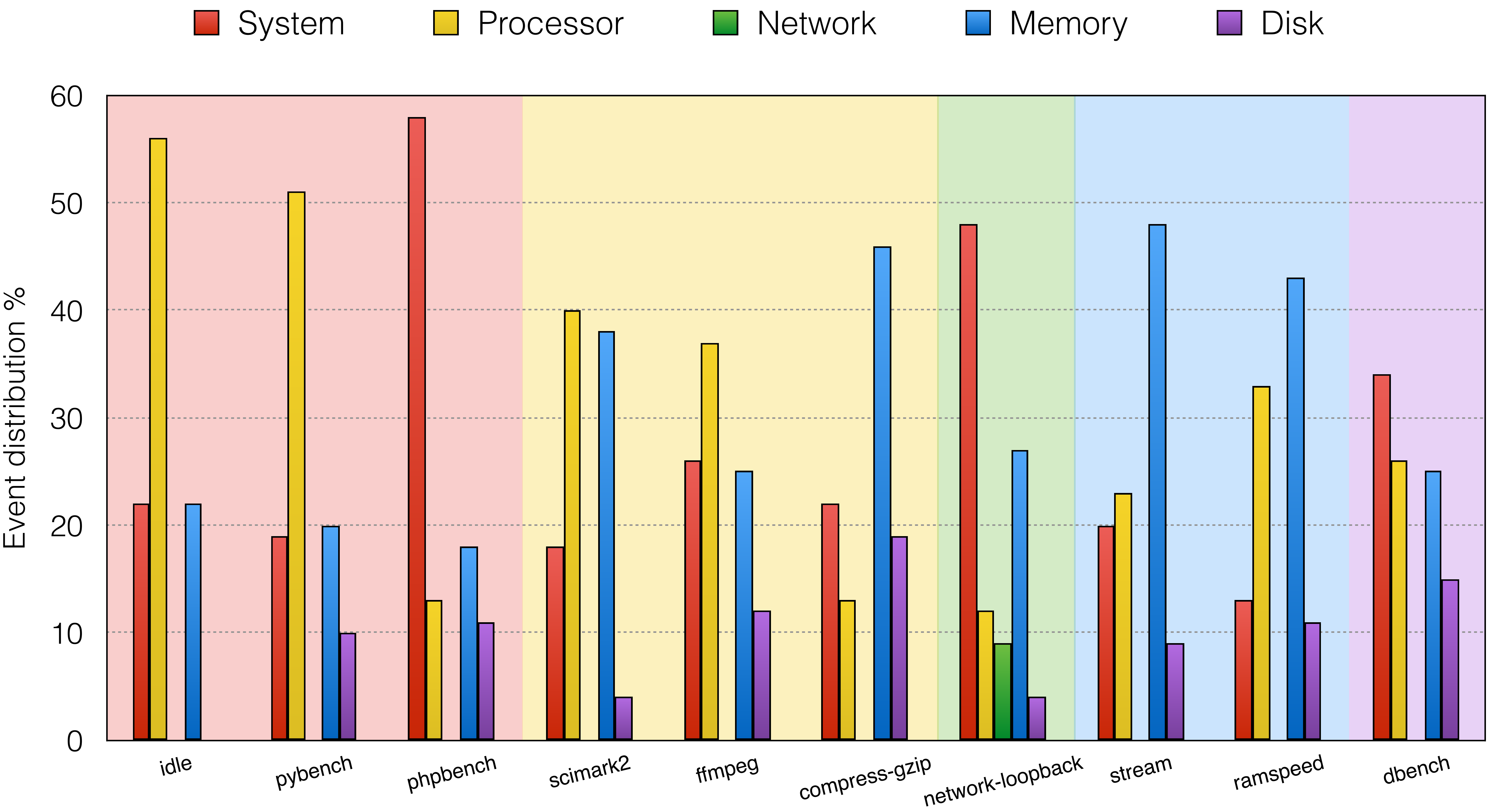
Family Distribution



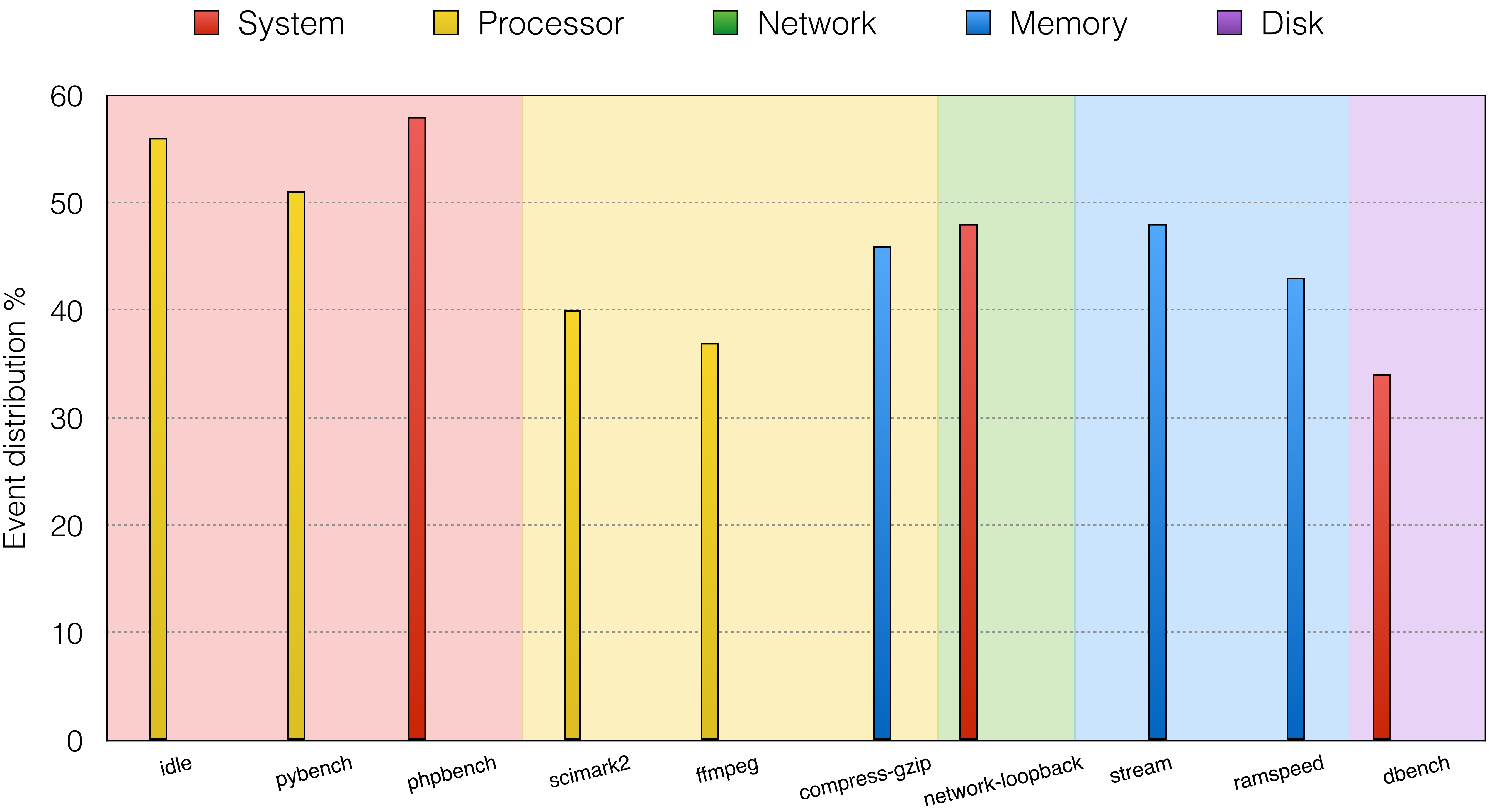
Family Distribution



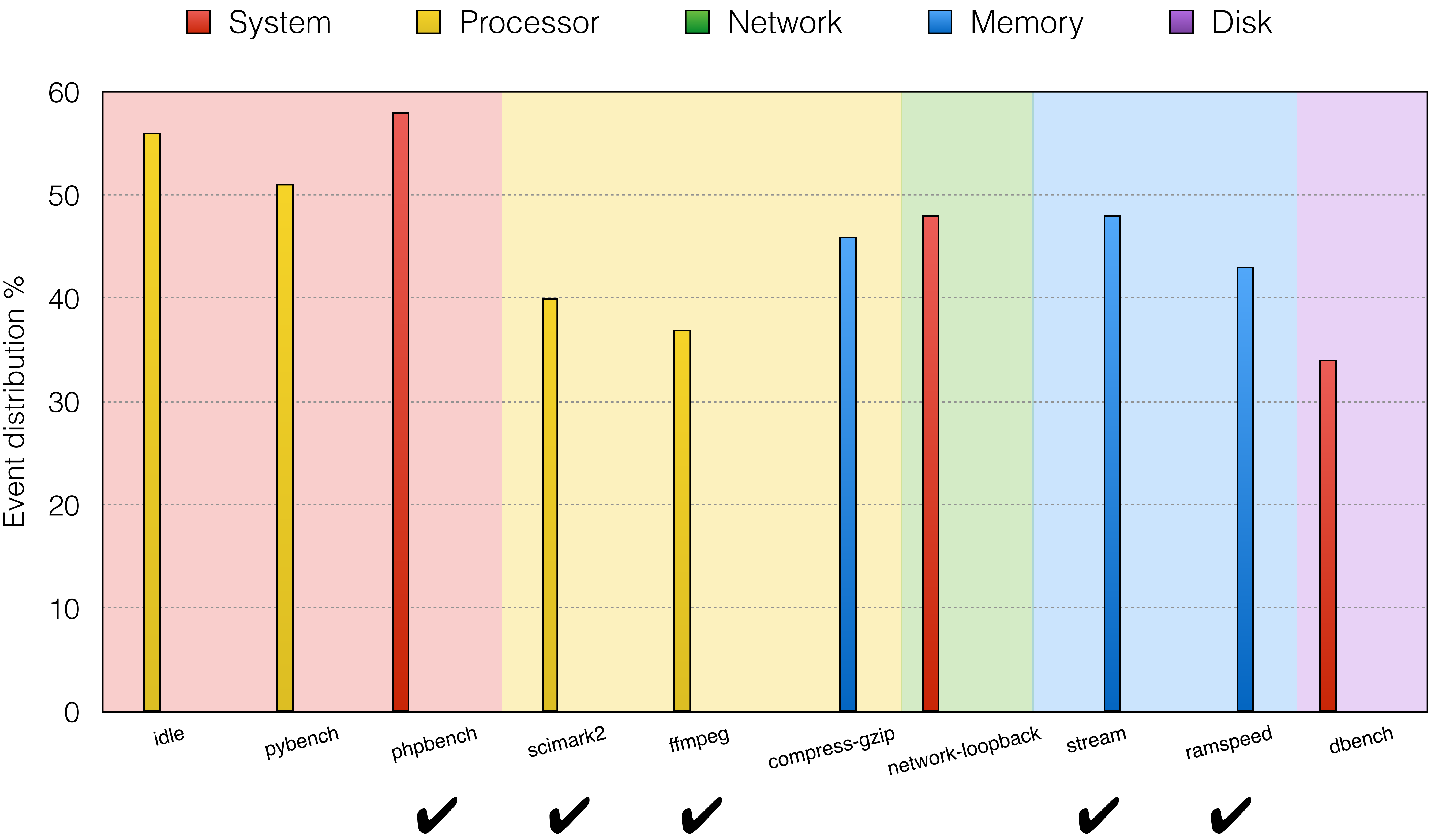
Family Distribution



Family Distribution



Family Distribution

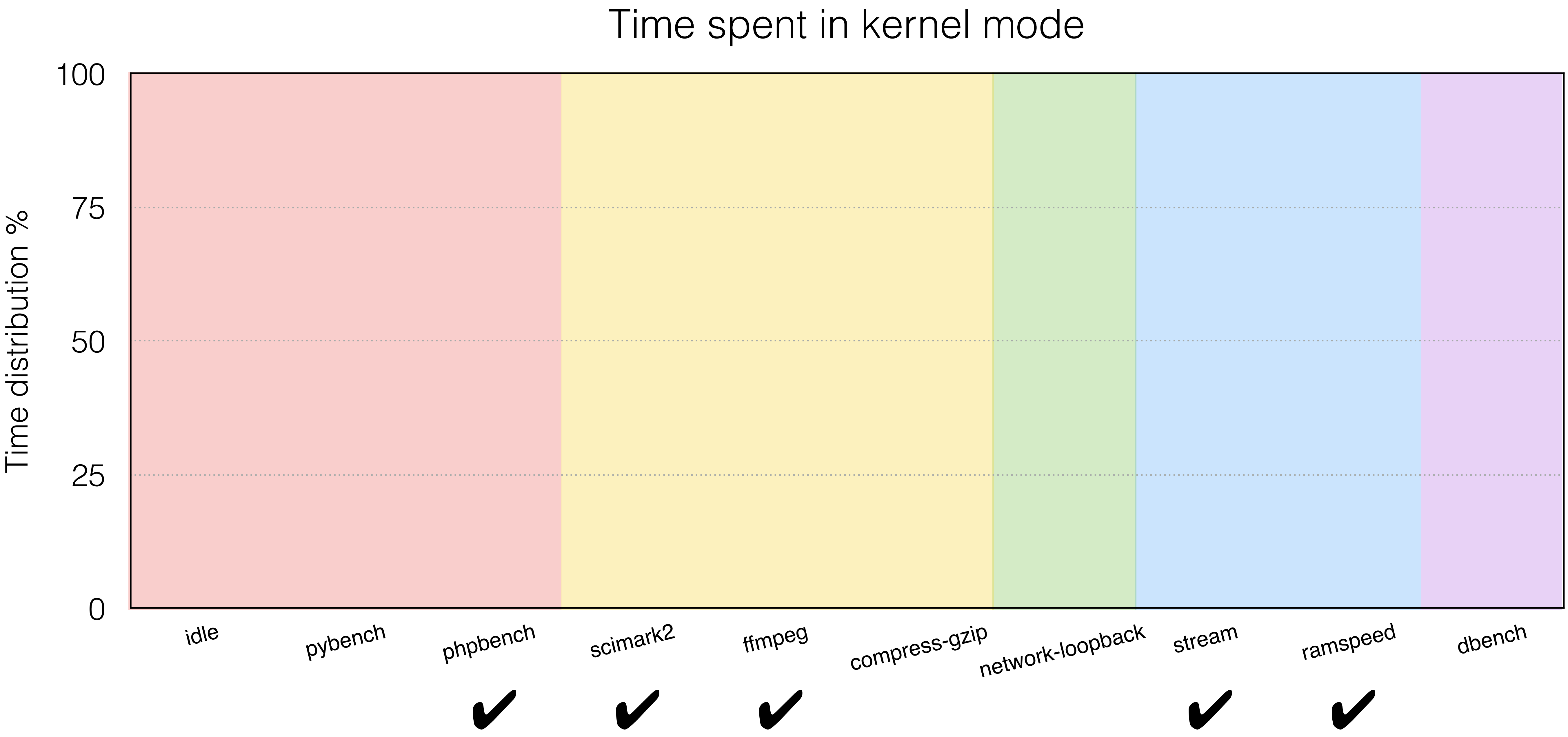


Family Distribution is not Enough

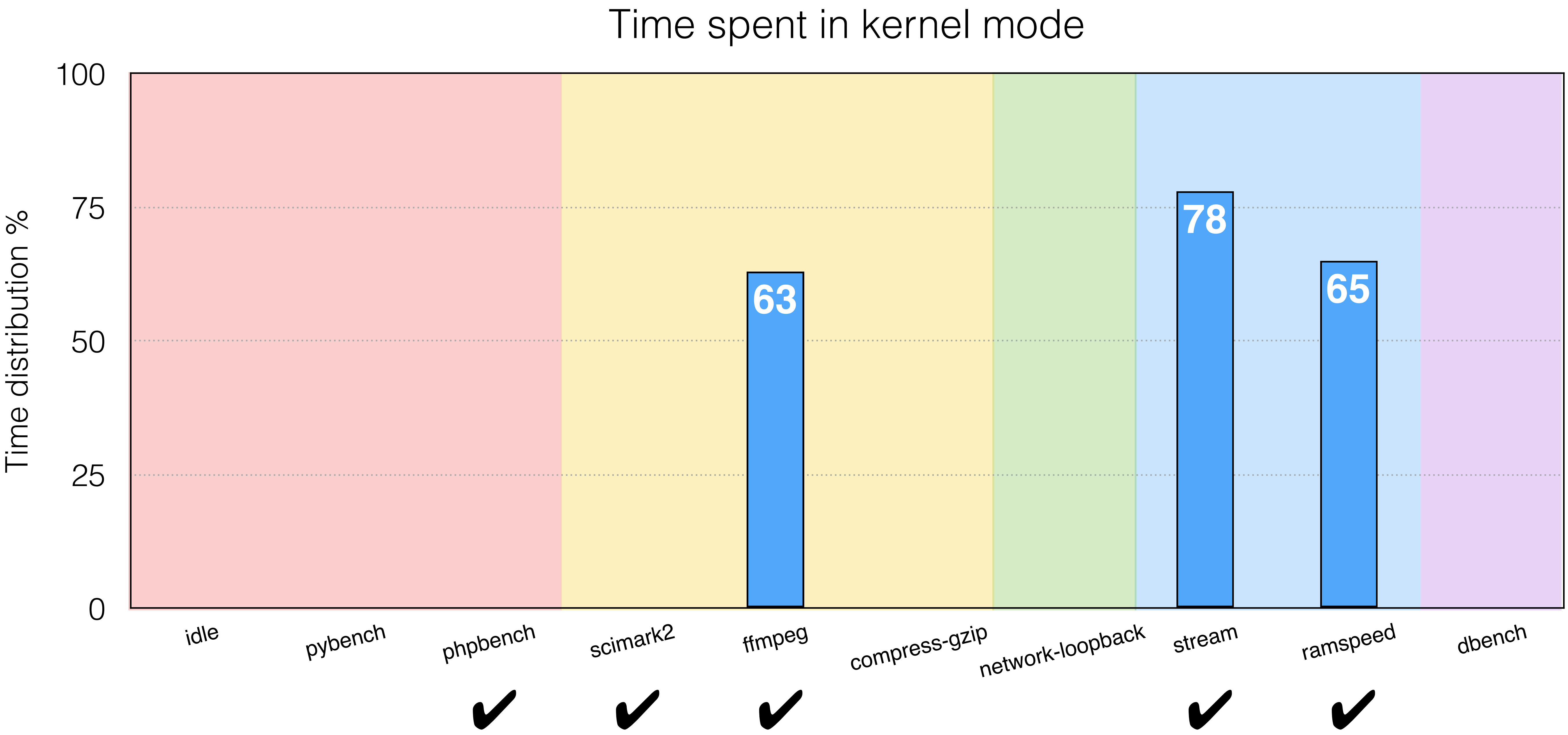
- **Computed** family = **announced** family ?
  - **5 matches** over **10**
- **Kernel function** is **different** from one to another benchmark
  - **No relation** between announced and calculated families
- We trace **only kernel** part

  ➜ Check the **distribution** of time during which the **kernel** is **used**

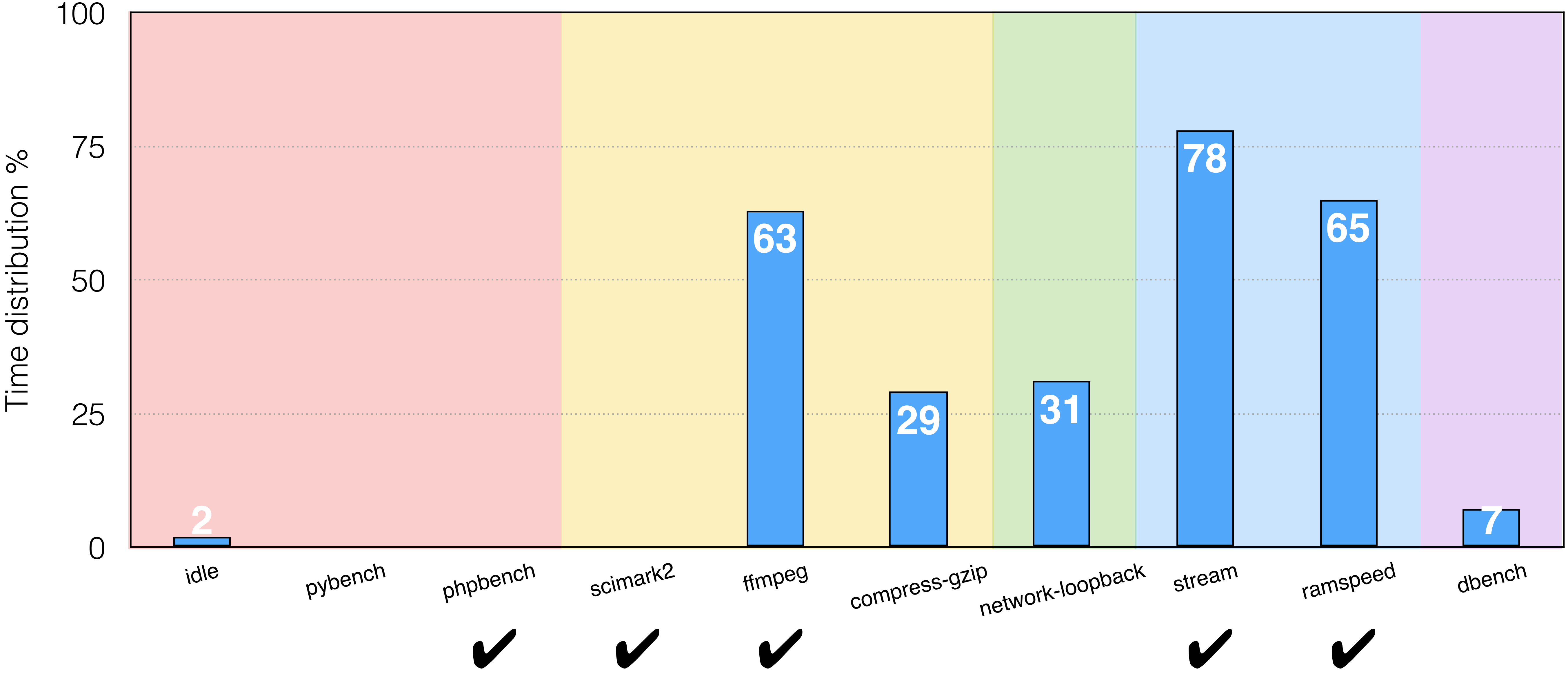
Kernel-time vs. User-time



Kernel-time vs. User-time

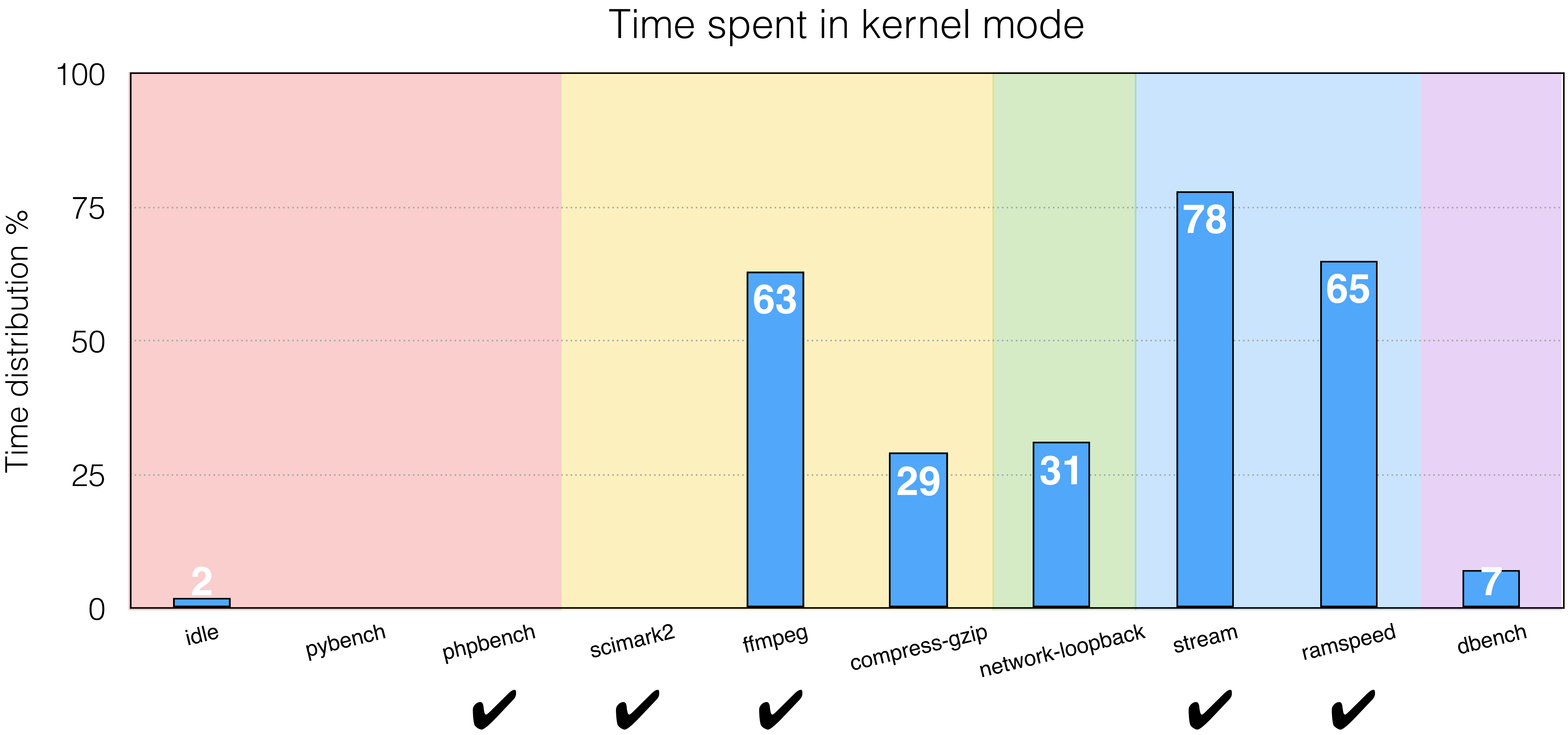


Kernel-time vs. User-time



Time spent in kernel mode

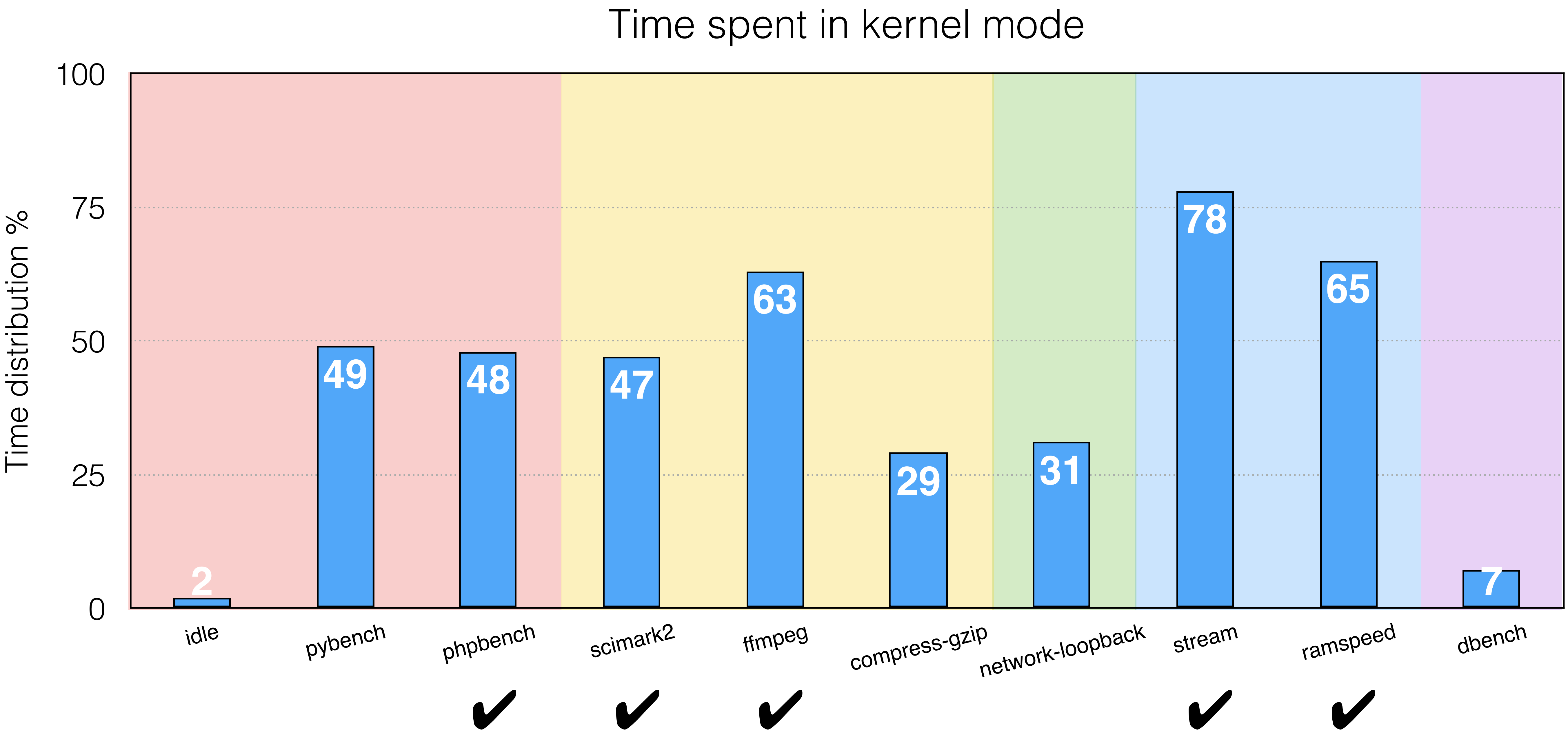
Kernel-time vs. User-time



Long time spent in kernel mode ➜ **Right** computed family

Short time spent in kernel mode ➜ **Wrong** computed family

Kernel-time vs. User-time



Long time spent in kernel mode ➜ **Right** computed family

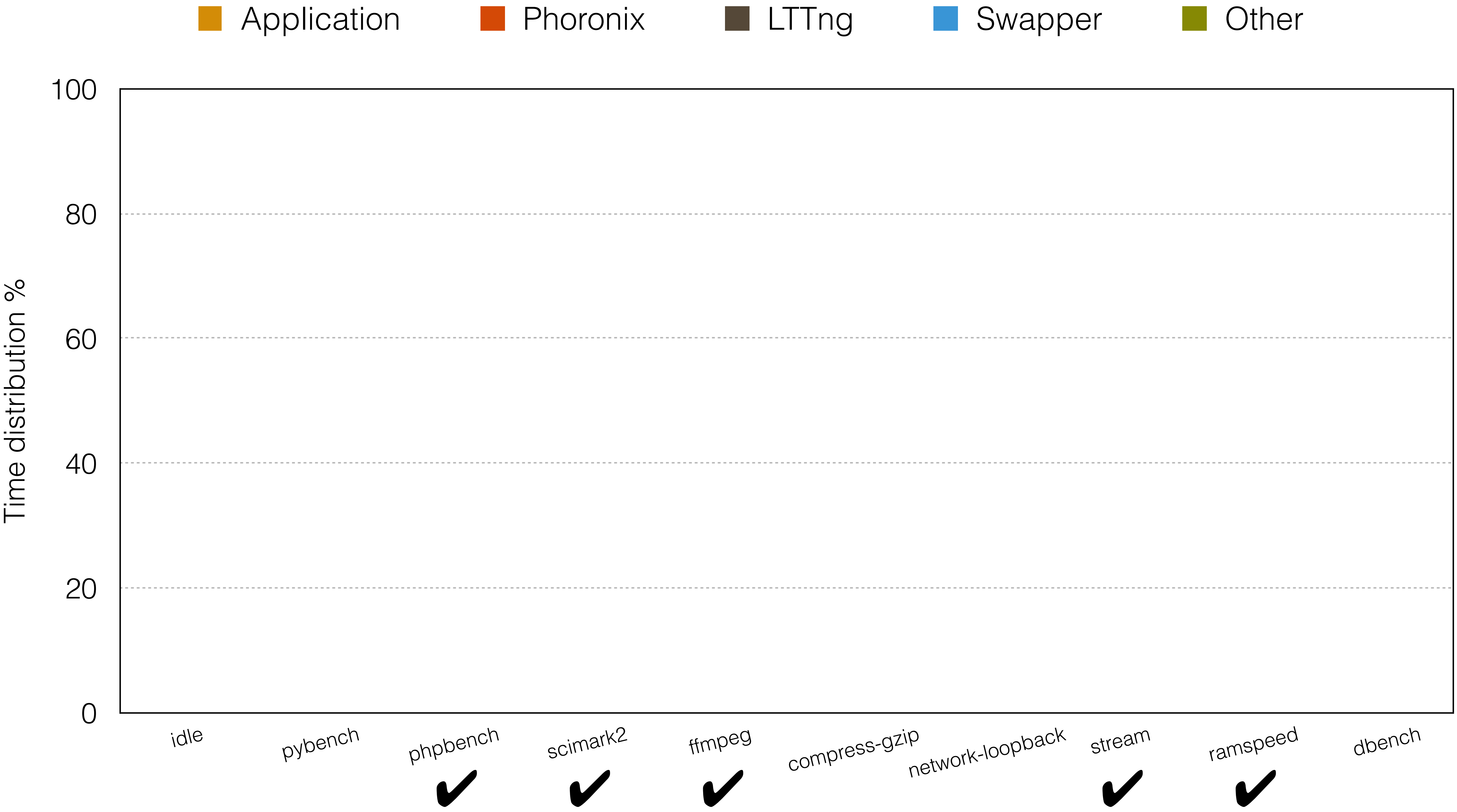
Short time spent in kernel mode ➜ **Wrong** computed family

Do We Observe More Than the Benchmark ?

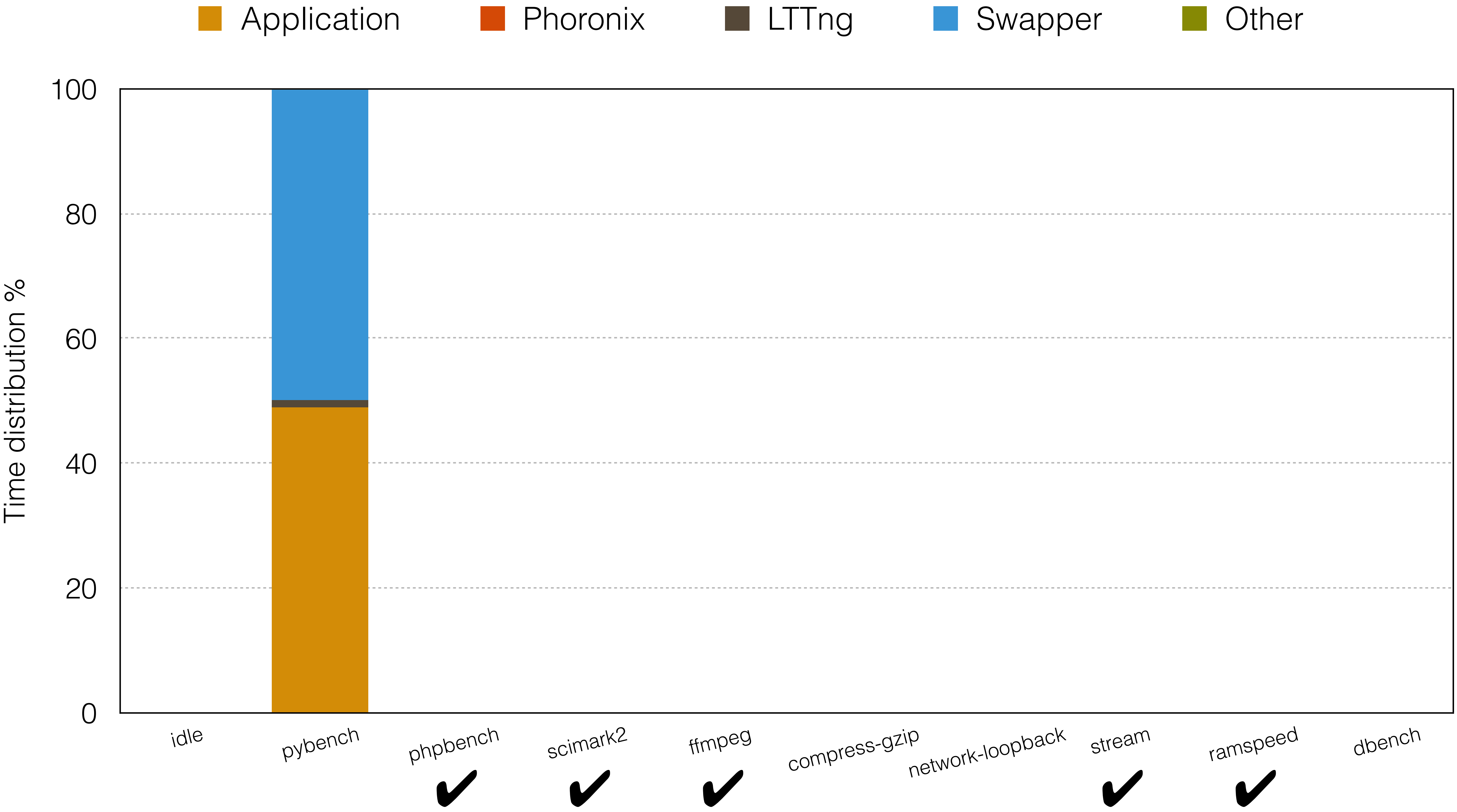
- Big **stack** of programs for **running** those benchmarks:
    - ssh
        - custom bash script
            - LTTng
            - Phoronix
                - Benchmark
- Analyze **overhead induced** by those programs

➜ Observe events by **processes**

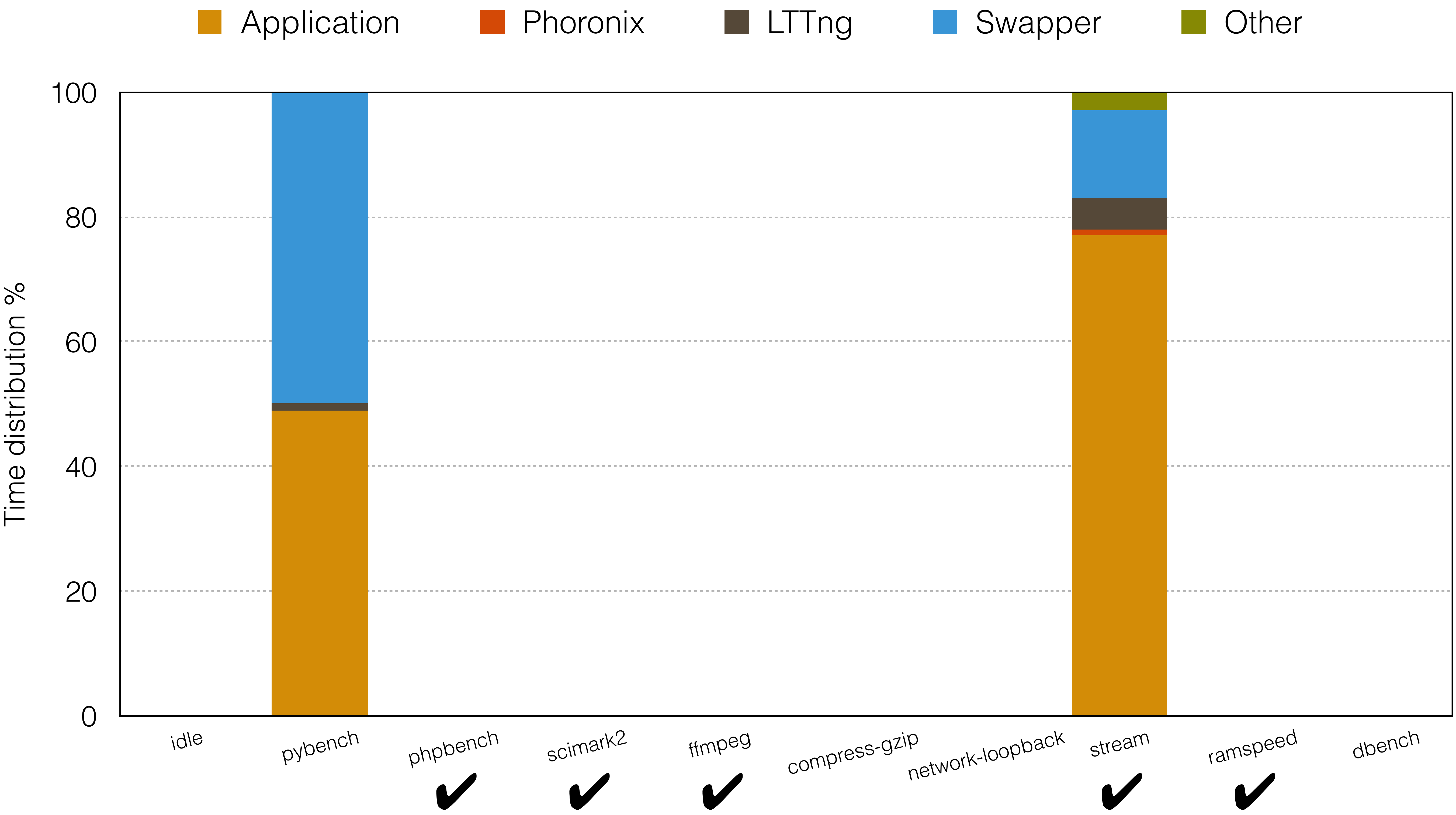
Time Spent by Processes



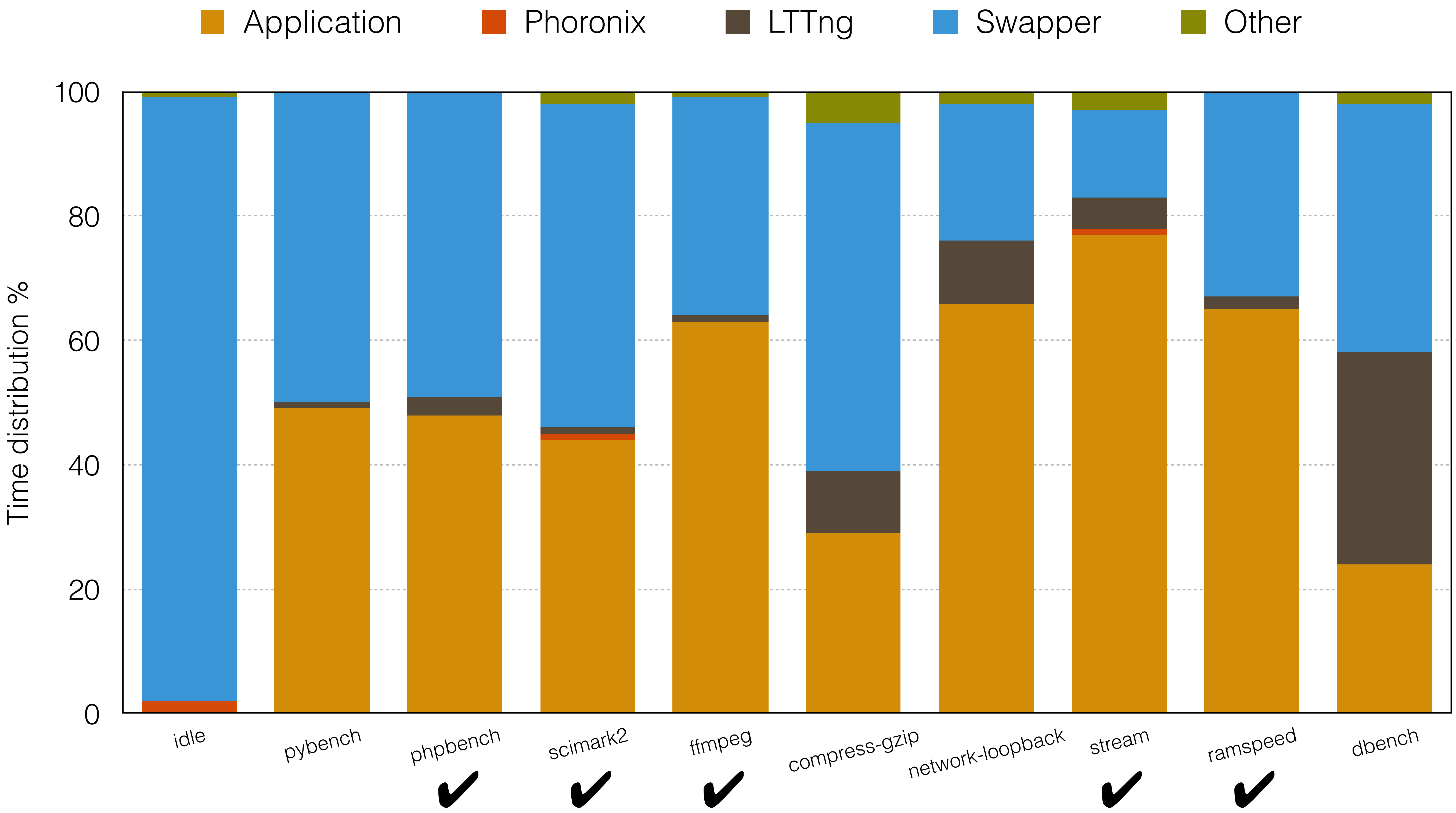
Time Spent by Processes



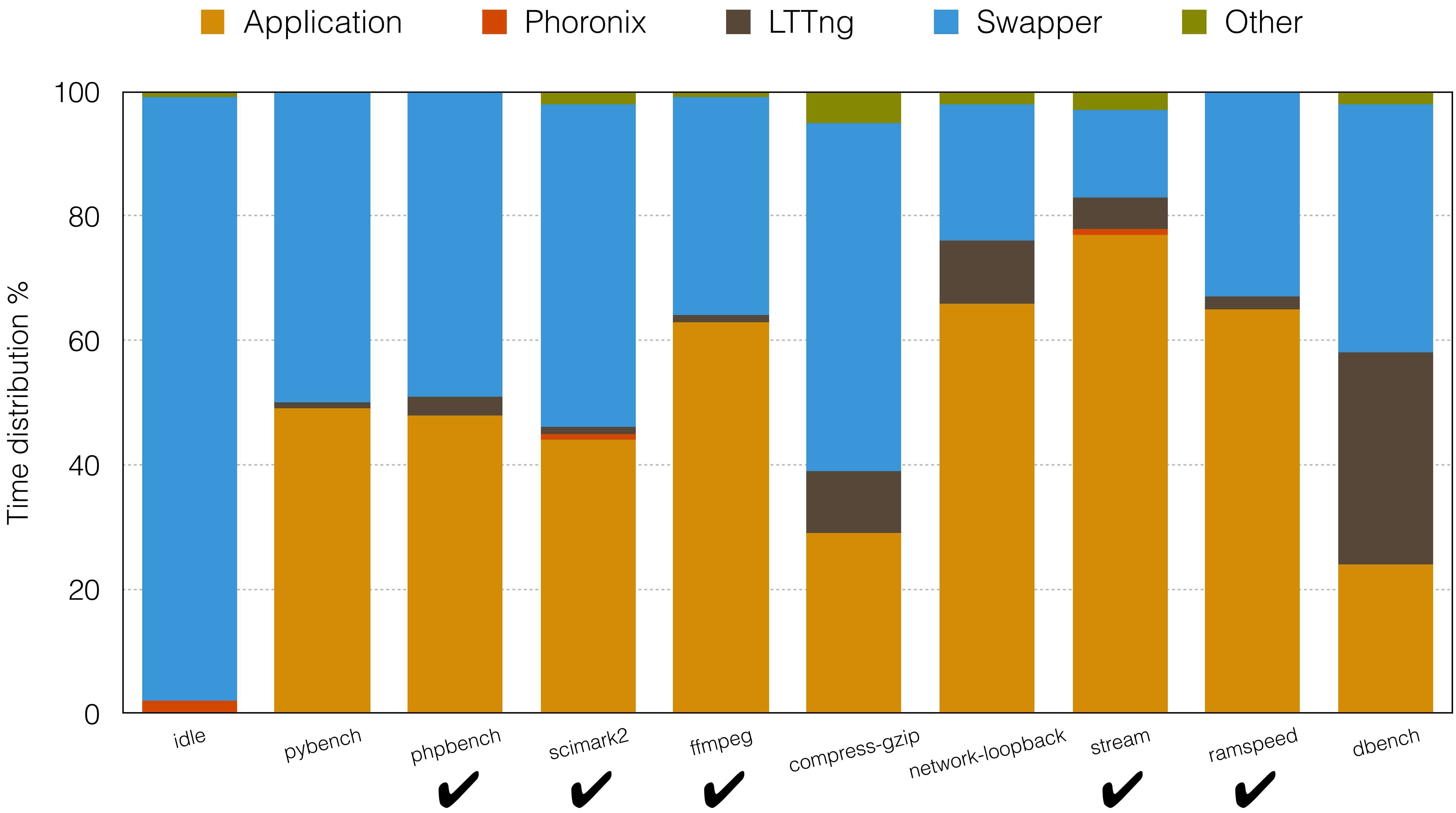
Time Spent by Processes



Time Spent by Processes



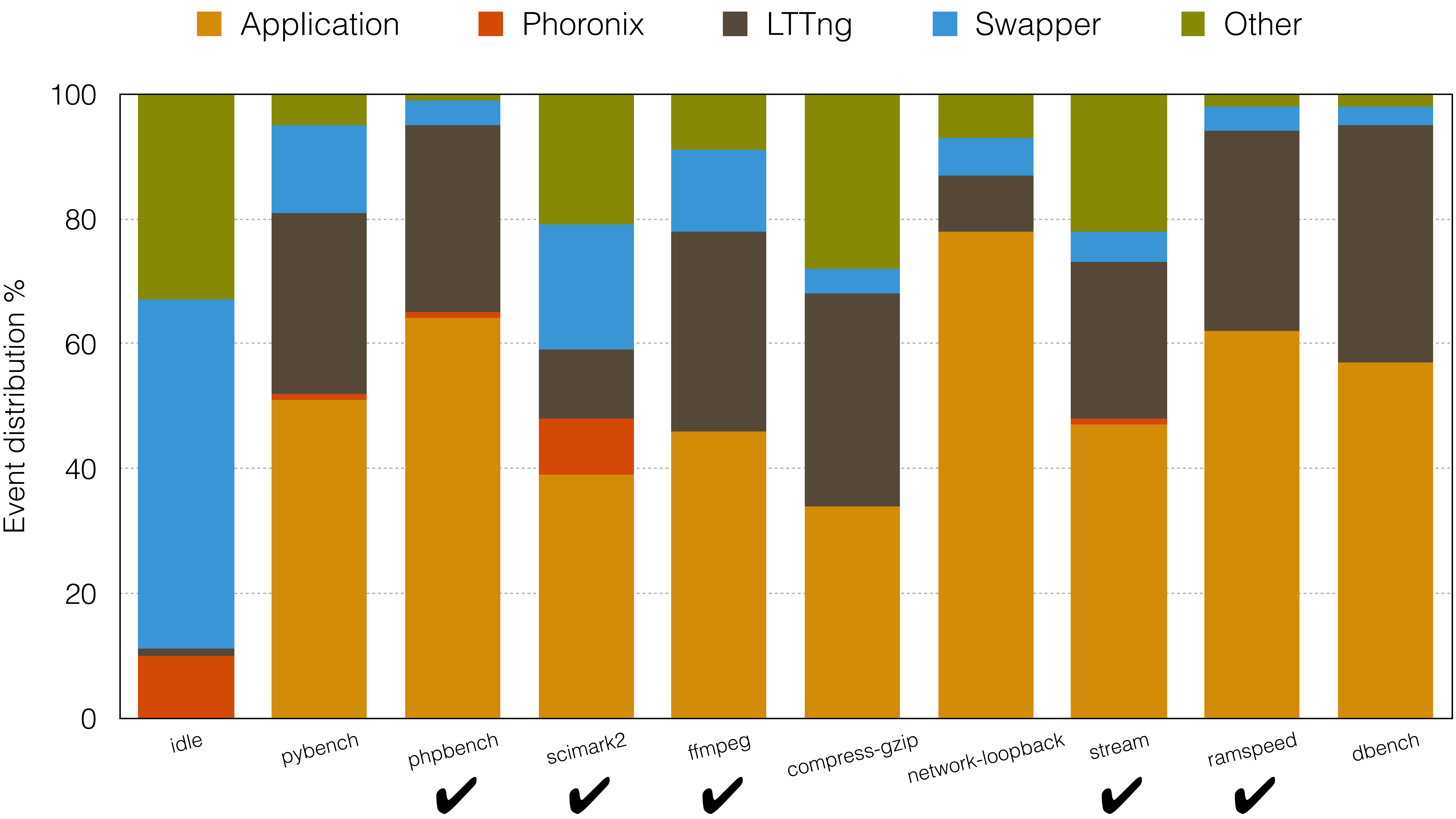
Time Spent by Processes



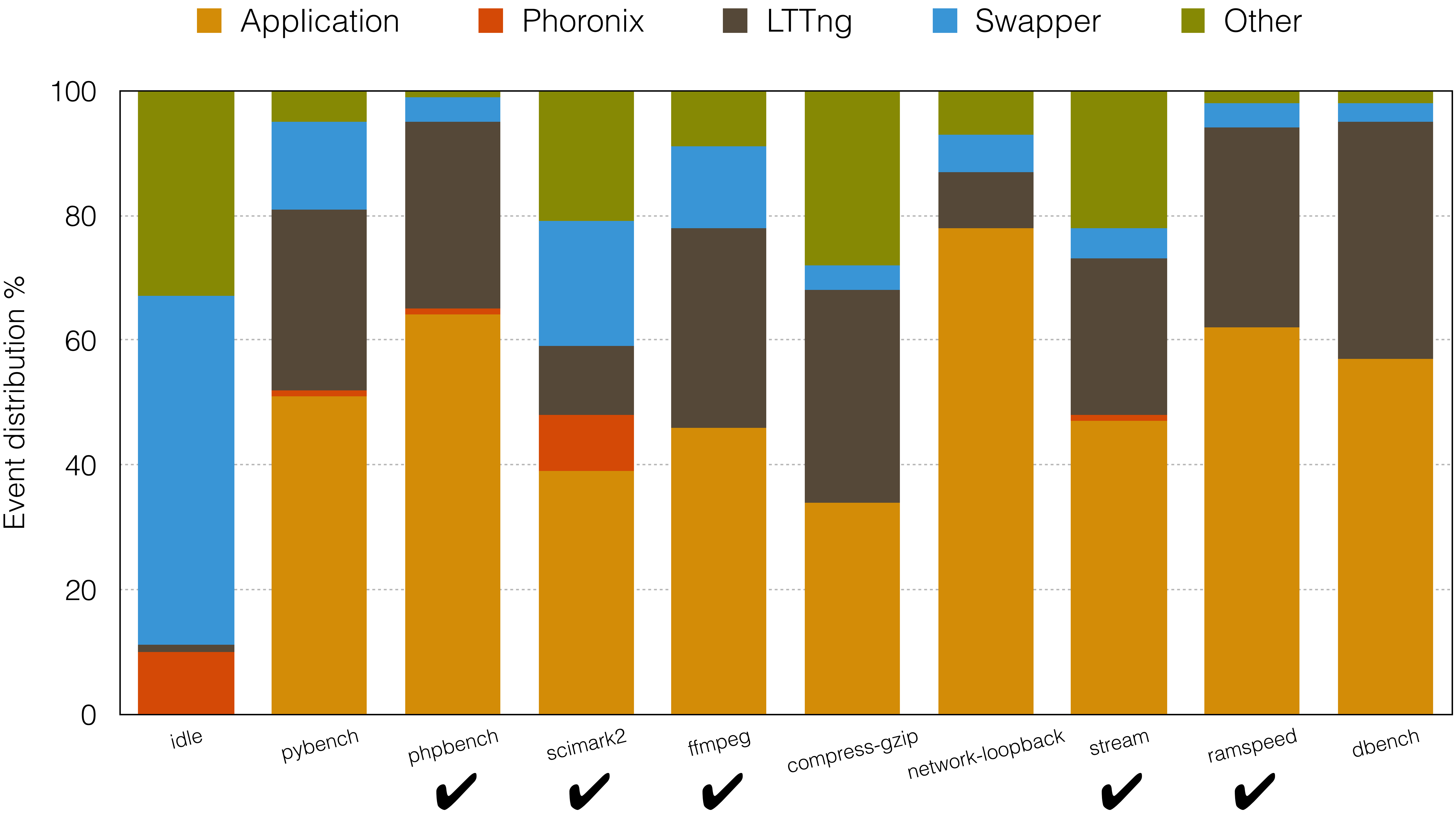
Swapper = **idle**

Phoronix: **low** intrusion

Event Distribution by Processes



Event Distribution by Processes



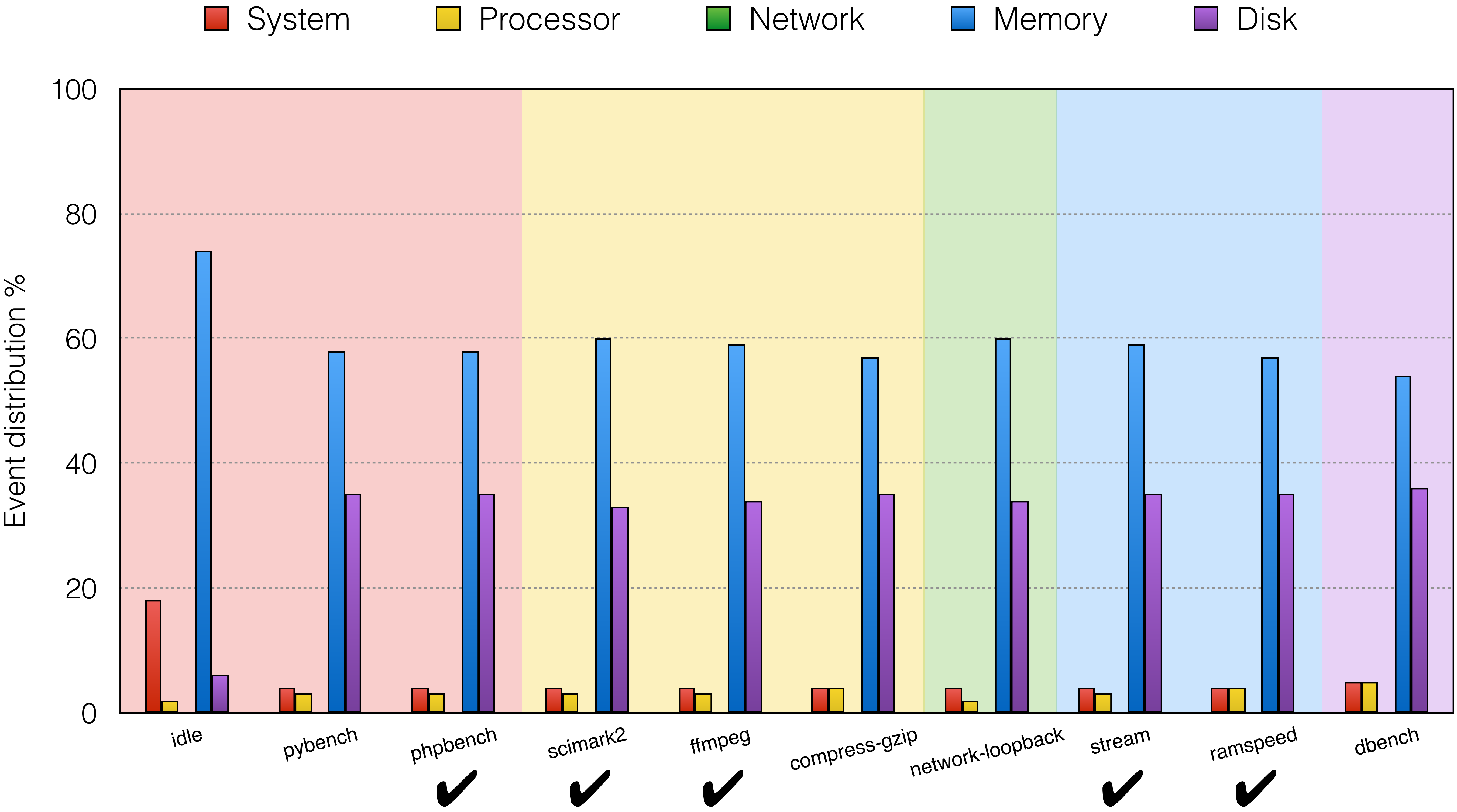
LTTng produces a **huge number** of events

Analysis of LTTng Overhead

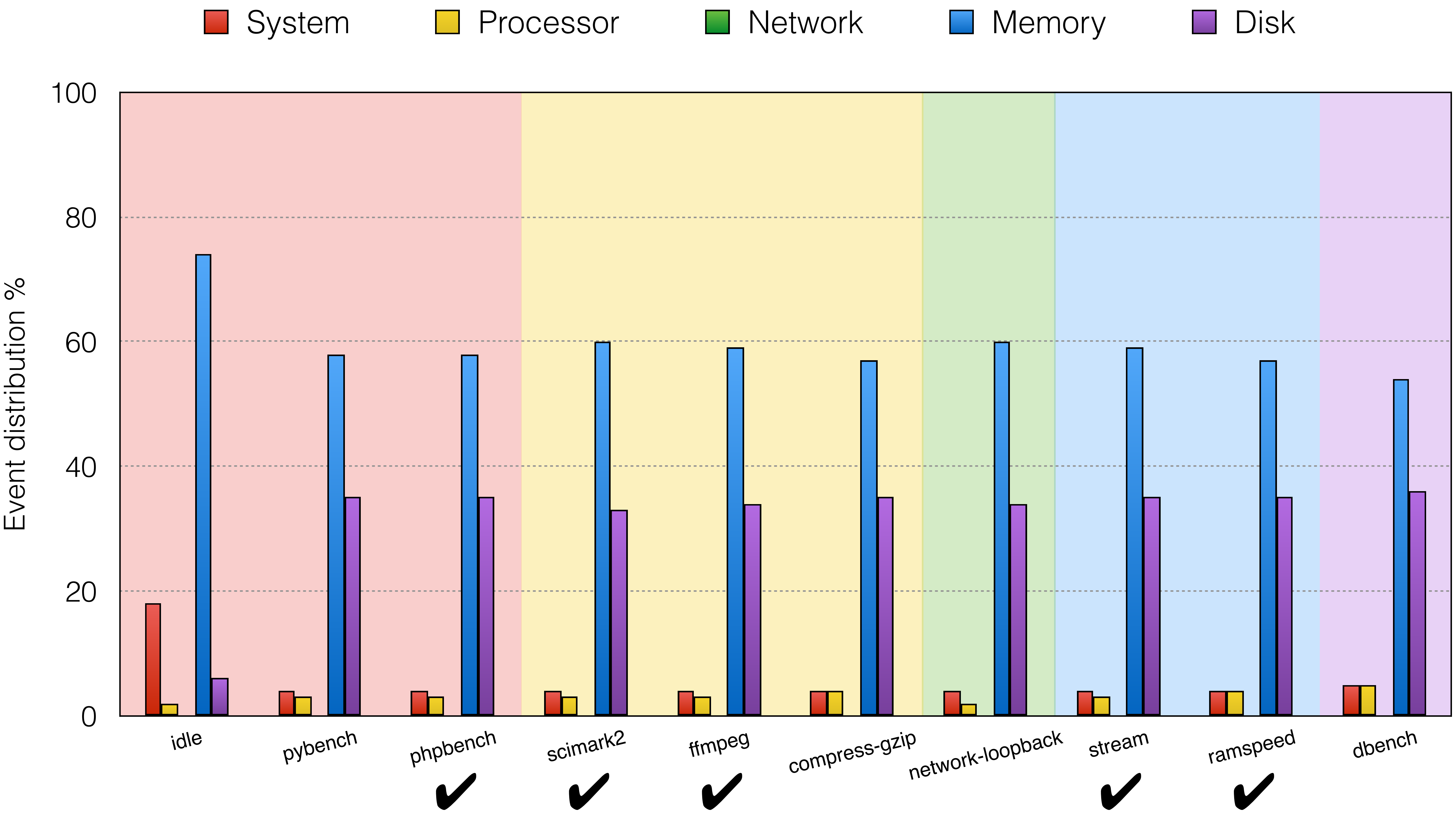
- **Not easy** to get only events from the benchmark
  - Names **depend on** benchmark
  - Some benchmarks are **not** only **a single program**
    - **several instances** of the same program
    - network-loopback = cat + dd + netcat
- **Overhead** comes mainly from **LTTng**
- LTTng overhead is **easy to remove** from trace
  - **Get** events from process **by name** and extract it

➜ Overhead **removed**, we observe only the benchmark

LTTng Overhead Profile

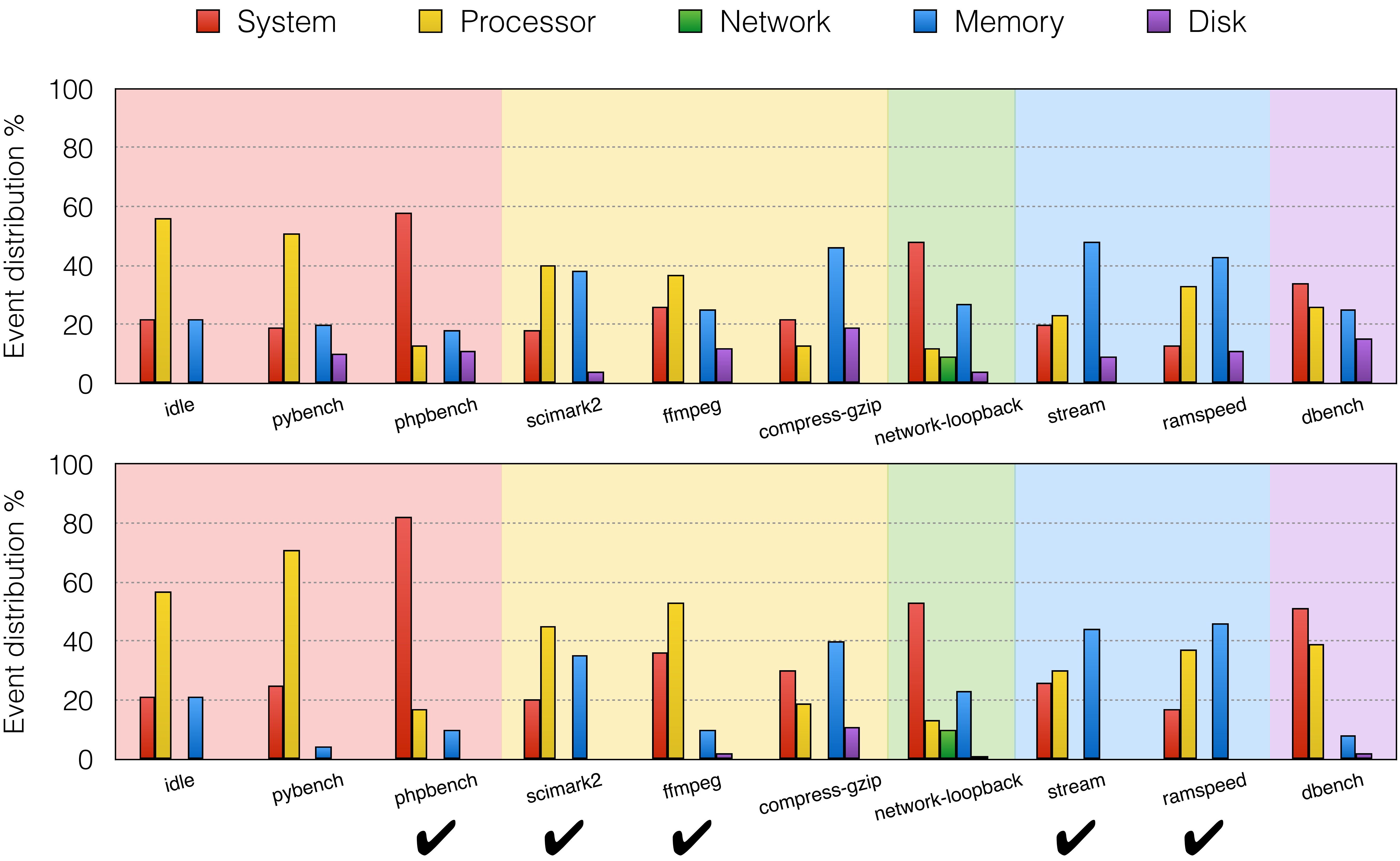


LTTng Overhead Profile



Stable using of kernel events

Real Benchmark Profile



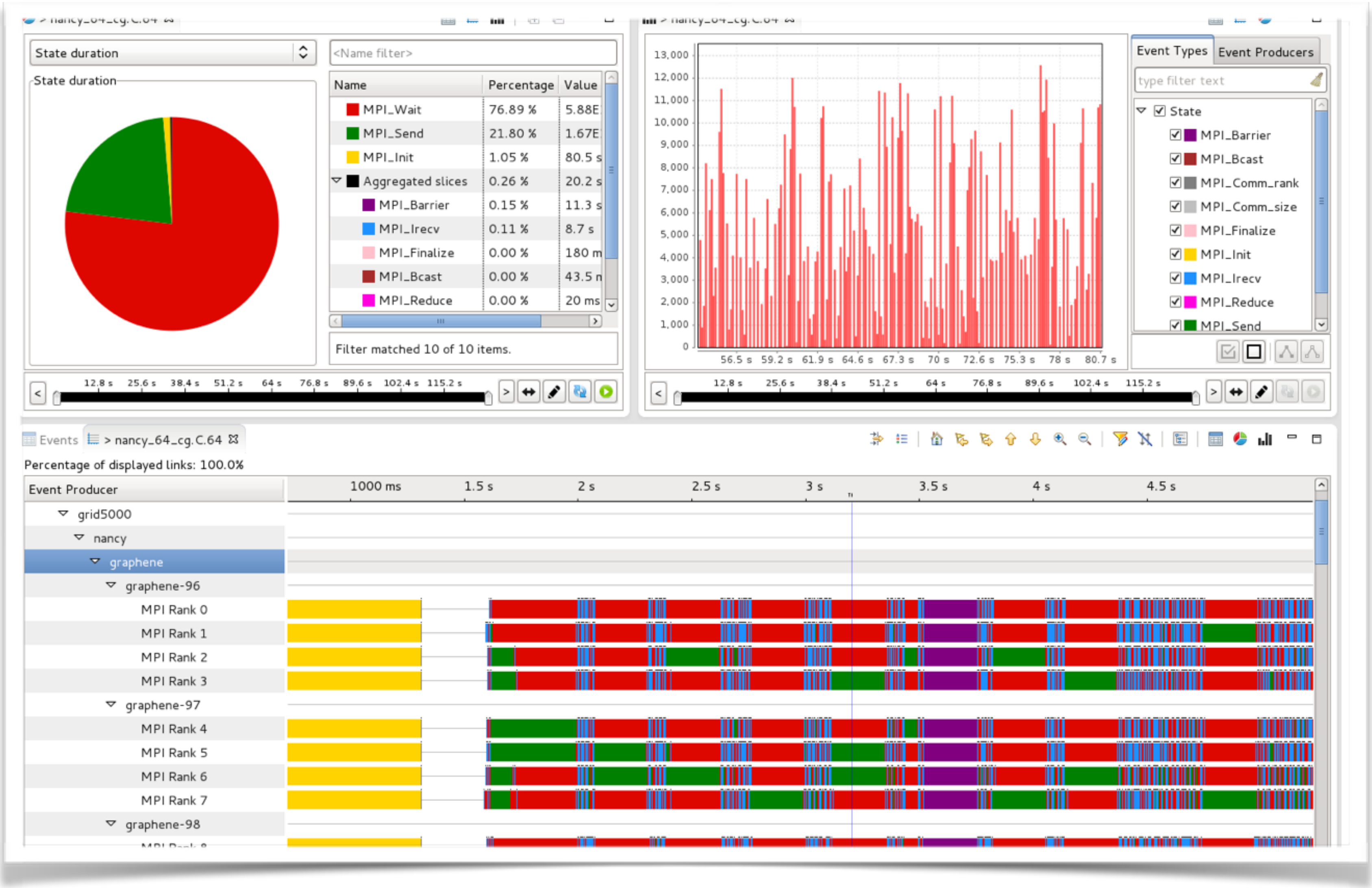
Conclusion

- Benchmark results:
  - **Better understanding** of benchmarking programs
    - **Profile** the kernel use (families, duration)
    - What can **impact** the performance
  - Most used benchmarks on phoronix are **very different**
    - **Different** profiles for **similar** tests
- **Intrusiveness** of used tools:
  - Phoronix is **not intrusive** (for long benchmarks)
  - LTTng **produces many** kernel **events**
    - **Constant** profile (memory + disk)
    - We **know** how to **remove** this overhead for the analysis

➜ **Generic** way to analyze benchmarks

Acknowledgment

- This work was done and funded within the **SoC-TRACE** project (link)
  - French ministry of industry
  - Inria, UJF, STMicroelectronics, ProbaYes
- **Framesoc** tool is an outcome of this project (soctrace-inria.github.io/framesoc/)
  - **Framework** for the **management** and **analysis** of traces



State duration

Filter matched 10 of 10 items.

Name	Percentage	Value
🟥 MPI_Wait	76.89 %	5.88E
🟩 MPI_Send	21.80 %	1.67E
🟨 MPI_Init	1.05 %	80.5 s
▽ ■ Aggregated slices	0.26 %	20.2 s
MPI_Barrier	0.15 %	11.3 s
MPI_Irecv	0.11 %	8.7 s
MPI_Finalize	0.00 %	180 m
MPI_Bcast	0.00 %	43.5 n
MPI_Reduce	0.00 %	20 ms

Percentage of displayed links: 100.0%

Thank You !