# KRASH: Reproducible CPU Load Generation on Many-Core Machines

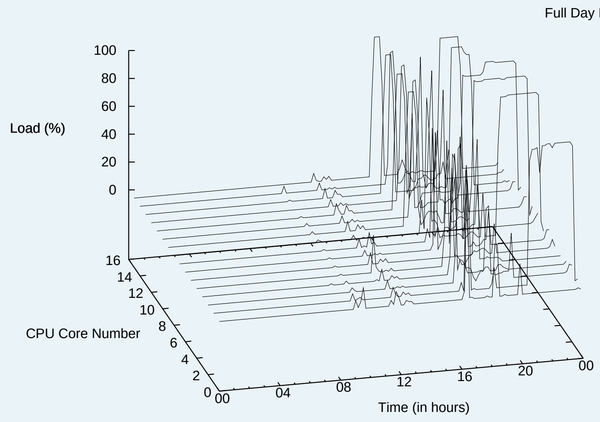
Swann Perarnau, Guillaume Huard

INRIA MOAIS Team, LIG Lab, Grenoble University

## Context

- New SMP Servers are many-cores, shared among multiple users.
  - ↪ System dynamic heterogeneity.
- New programming paradigms developed for heterogeneous environments.
  - ↪ Reproducible experiments are needed to compare them.

## Multi-user systems are dynamic heterogeneous platforms




Daily CPU load of a 16 cores server

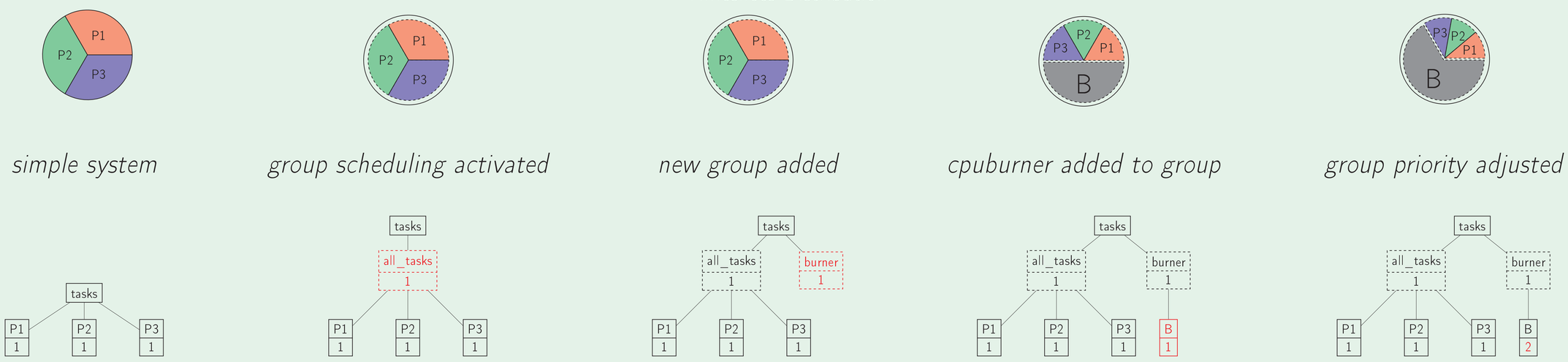
## Our Approach

- Use dedicated machines to emulate multi-user environments.
  - ↪ Easier to control.
- Use a load generator.
  - ↪ A loaded machine mimics an heterogeneous one.
  - ↪ A load profile can be used to emulate **any** heterogeneity.

## Our goals

- Reproducibility: obtain the same load load independently of the environmental conditions.
- Unobtrusiveness: do not alter qualitative behavior of the system (scheduler policy, I/O).
- Precision and reactivity: produce realistic load profiles in a dynamic environment.
- avoid intrusiveness: additional load induced by the generator process must be moderate.

## How KRASH works: 5 steps to achieve cpuburner fixed share attribution




Groups and Processus Hierarchy (with priorities)

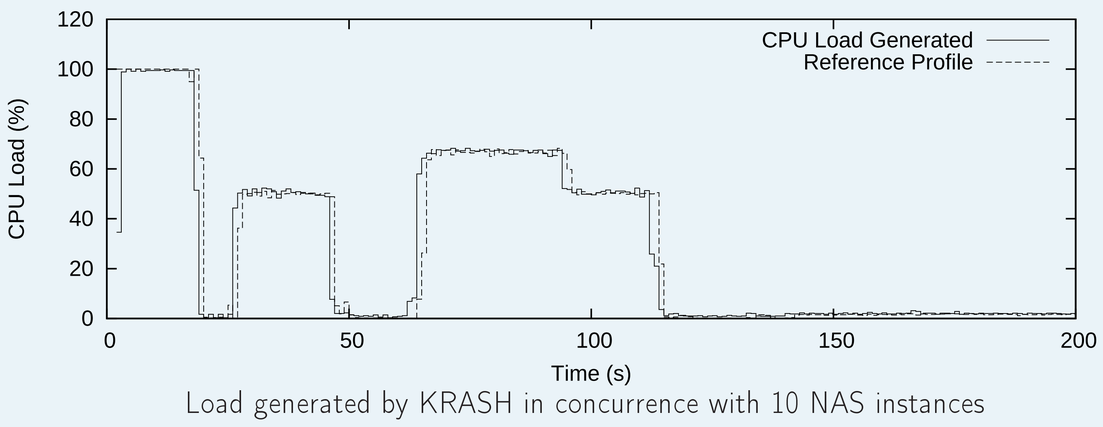
## Qualitative Comparison of Existing Solutions

| | General dynamic load profile | Side effects on scheduler | Arbitrary number of processes | Intrusiveness | Resolution |
|---|---|---|---|---|---|
| KRASH | Yes | Negligible | Yes | Negligible | Same as scheduler |
| Wrekavoc | Not implemented | Low | No | Active poll | Higher than scheduler |
| Real time priority | Not implemented | High | Yes | Periodic wakeup | Poor |
| Frequency scaling | No | Medium | Yes | Negligible | Higher than scheduler |

## Kernel for Reproduction and Analysis of System Heterogeneity

- Design:
  - Use a burner process to steal timeslices from applications.
  - Use a supervisor to ensure correctness of the load.
- Technicalities:
  - Use group scheduling to give timeslices to the burner independently of the number of applications.
  - Use notifications to cope with group updates.

## KRASH Validation




Load generated by KRASH in concurrence with 10 NAS instances

## Load Generators Obtrusiveness

| | Time to copy a file | | Slowdown |
|---|---|---|---|
| | average | standard deviation | |
| None | 10.2 | 0.8 | 1 |
| KRASH | 20.5 | 0.5 | 2 |
| Wrekavoc | 24.9 | 1.7 | 2.4 |
| Real time priority | 36.6 | 1.8 | 3.6 |
| Frequency scaling | 24.3 | 1.8 | 2.4 |

Influence of a 50% load on file copying (1GB) evaluated for several methods.

## Conclusion

- KRASH generates a reproducible CPU load and provides more features than previous solutions.
- Future works will extend KRASH to other parts of a multi-user system (memory, disk, network).