SmartOS: Towards Automated Learning and User-Adaptive Resource Allocation in Operating Systems

Sepideh Goodarzy[1] , Maziyar Nazari[1] , Richard Han[2] , Eric Keller[1] , Eric Rozner[1]

[1] University of colorado Boulder, [2] Macquarie University

Operating systems

At 2018, 92% of households in US -> at least one computer [1]

Even more now!

Frustrated users

The time lost due to the frustrating experiences ranged from 30.5% to 45.9% of time spent on the computer. [2]

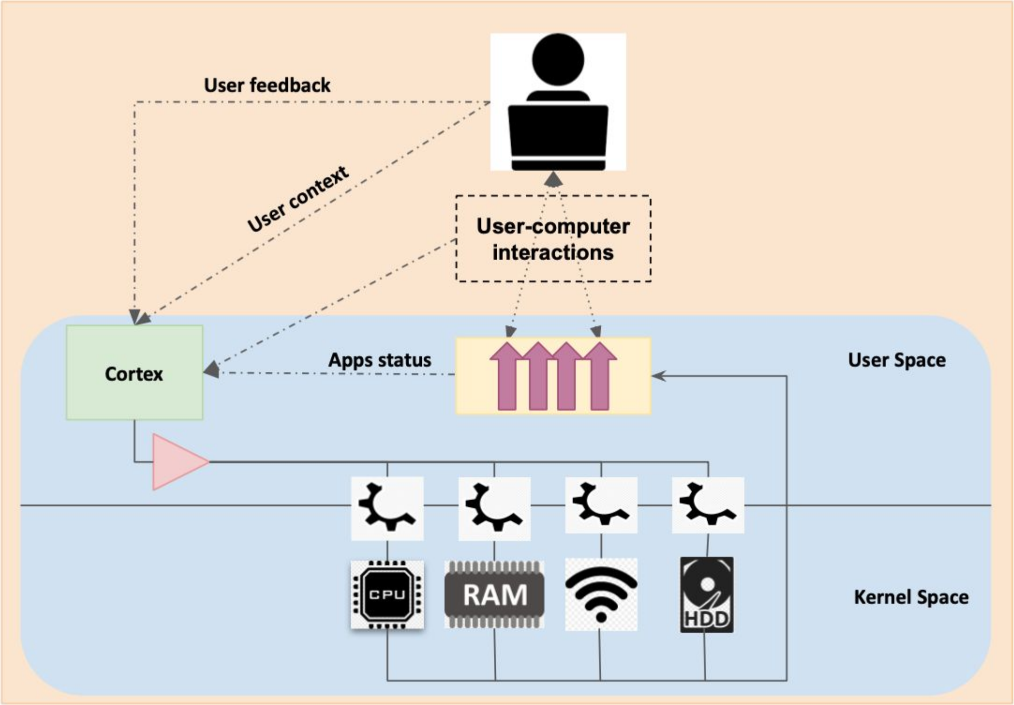
Frustration causes [2]

- Error messages
- Dropped network connections
- Long download times
- Hard-to-find features

What can be done ?

- Give more resources (CPU, memory, network bandwidth, Disk I/O, etc.) to the applications that matter **MORE**!
    - Manually By user: Not all users are computer experts! It’s time consuming and a frustrating job!
    - Automatically By OS
- What applications matter more ? Is it always the foreground ?
    - Editing a doc in Microsoft Office Word, while listening to music on youtube in Google Chrome, upgrading some software in the background and monitoring the stock widget on top of your desktop
- ***Based on the context, different users care more about some applications more than the others!***
- How we can find the important applications ?

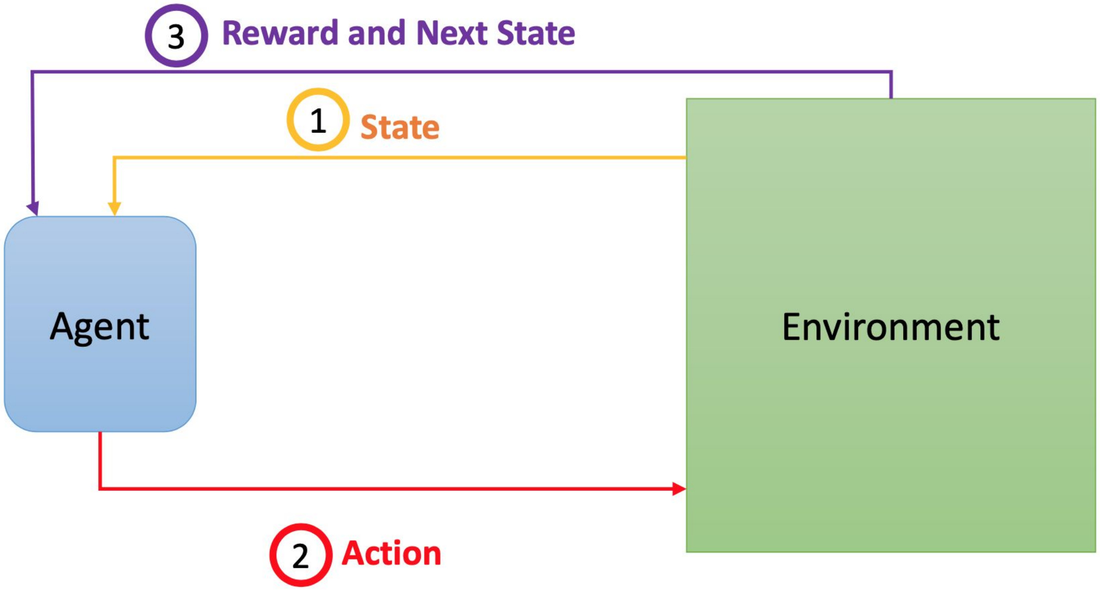
The solution



Cortex

- Heuristic based: Needs knowledge beforehand! Static! User preferences are changing constantly!
- Supervised/Unsupervised Machine Learning:
    - One time learning: Static! User preferences are changing constantly!
    - Cyclic learning/eval/deploy: What is the correct frequency to do learning? What happens if the user preferences differ from learned model faster than the learning frequency ?

Reinforcement Learning



Prototype

Reinforcement learning

State:

CPU	Memory	Network	Disk I/O	Fg	Audio/video
0/1	0/1	0/1	0/1	0/1	0/1

Action:

CPU	Memory	Network	Disk I/O
0/1	0/1	0/1	0/1

Reward:

Synthetic, +1 for the action leading to best performance and 0 otherwise

Resource Allocation

- **CPU:** *Nice value*
    - *High prio: -20*
    - *Normal Prio: 0*
- **Memory:** *OOM Adjacent score and Cgroup memory swappiness*
    - *High prio: -1000 for score and 0 for swappiness*
    - *Normal prio: 0 for score and 60 for swappiness*
- **I/O:** *I/O nice*
    - *High prio: 0 real-time class*
    - *Normal prio: 4 idle class (default)*
- **Network bandwidth:** *Cgroup network I/O prio map*
    - *High prio: 10*
    - *Normal prio: 0*

Evaluation

Heuristics

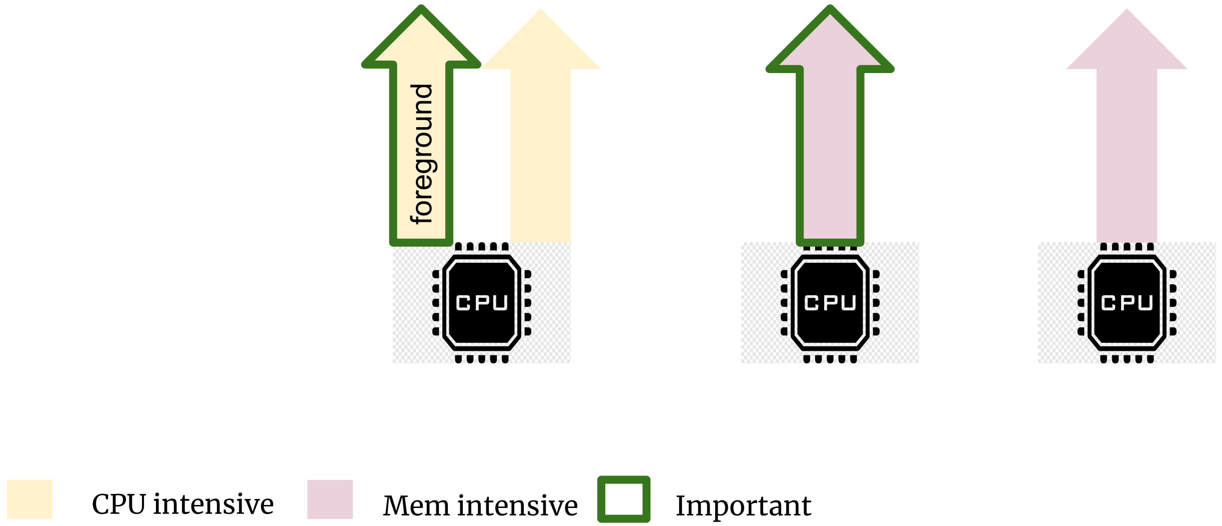
- **Linux:** *default linux CFS scheduler*
- **Fg only:** *high priority for foreground application for all resources*
- **Fg + video/audio:** *high priority for foreground and video/audio applications for all resources*
- **Fg + dependent:** *high priority for the foreground application and all other applications that foreground performance depends on.*
  - *predefined directed acyclic graph to store dependencies between applications*

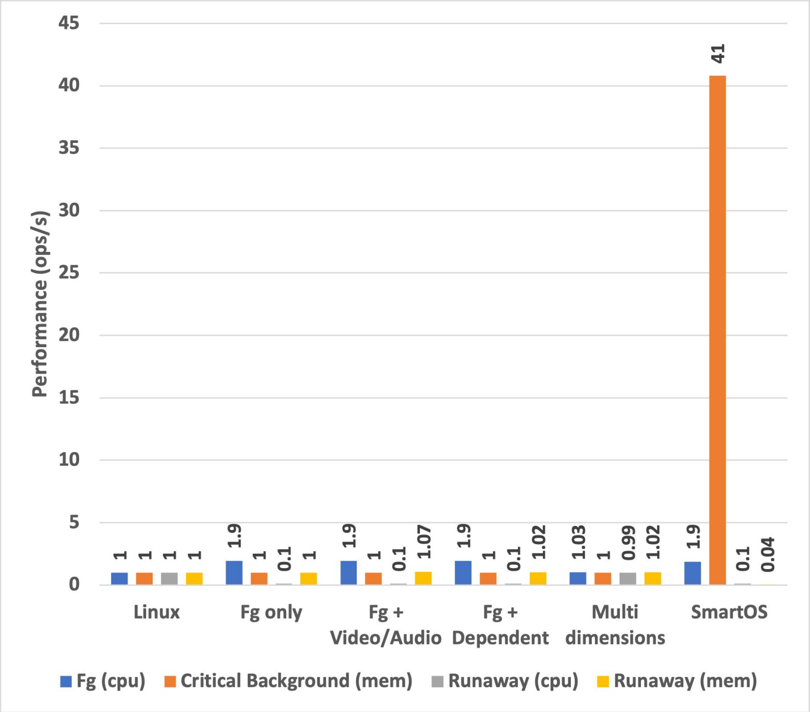
Heuristics

- **Multi-dimension:**
    - *Uses a predefined map that stores the essential resources per application's performance*
    - *High prio for the foreground application in all resources necessary to its performance*
    - *Assign the remaining resources to the important applications to the user*
    - *Important applications are stored in a hash map defined by asking the user in the beginning.*

Variation of Dimensions

- Ubuntu 20.04 virtual machine with 8 GB of memory, 4 processors, and 50 GB VDI disk drive.





For more scenarios please refer to the paper

SmartOS Dynamicity and Convergence

- 4 different scenarios
- 60 seconds on each scenario (extreme case)
- Ask for the feedback after each scenario and move to the next scenario

With less frequent change of applications, SmartOS is able to achieve convergence even sooner!

RL Algorithm	Feedbacks
DQN	52000
QLearning	28400
Sarsa	3680
Double Qlearning	2400
A2C	1600
Monte Carlo	400

SmartOS Overhead

- SmartOS adaptation to each user feedback takes:
    - 0.218 ms total execution time
    - 0.21 ms pure CPU time
    - So less compared to human adaptation time which is in order of seconds
- Required memory for cortex 21.3 MB
- Negligible overhead!

Discussion and Future Work

- Real world scenarios
  - Human study
  - Adding implicit user feedback
  - Continuous state and action vector
  - More complex user context
- SmartOS failures
  - In case of failure return back to Linux CFS
- Cross platform
  - Place the cortex in the cloud
  - Work with multiple devices
  - Learn across users

Conclusion

- Learning based operating system
- Adjusting resource allocation based on user preferences
- Works better than heuristics
- Monte carlo achieves convergence sooner than other RL algorithms
- Overhead is negligible (order of tenth ms)

Thank you!

Email: *Sepideh.goodarzy@colorado.edu*

Any Questions?

Resources

- https://www.census.gov/newsroom/press-releases/2021/computer-internet-use.html
- https://www.researchgate.net/publication/2834775_Understanding_Computer_User_Frustration_Measuring_and_Modeling_the_Disruption_from_Poor_Designs