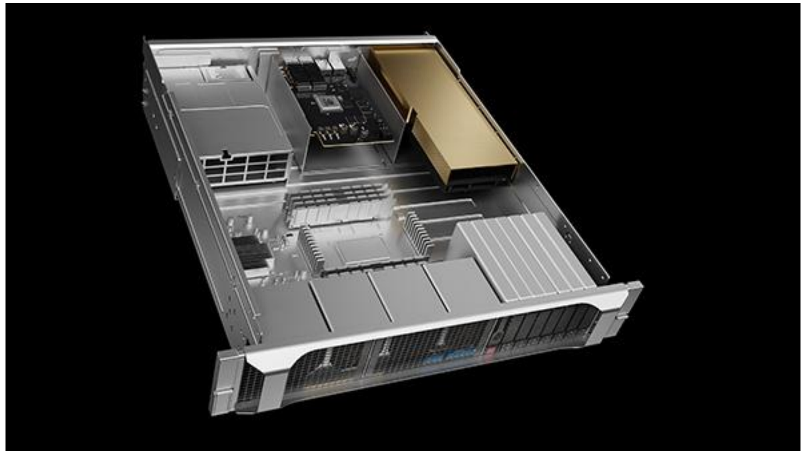
Serverless Edge Computing: challenges and opportunities stemming from heterogenous hardware

Tomasz Szydlo[1,2]

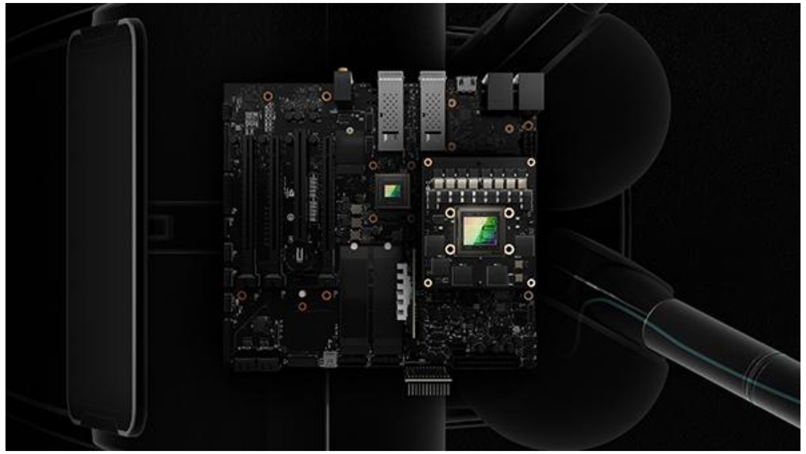
tomasz.szydlo@{agh.edu.pl|newcastle.ac.uk}

[1]*AGH University of Cracow, Cracow, PL*
[2]*School of Computing, Newcastle University, Newcastle upon Tyne, UK*

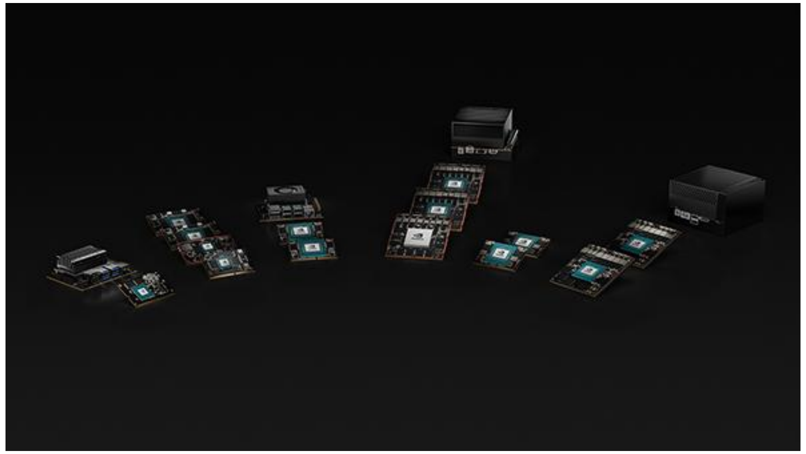
Edge Computing



Enterprise



Industrial



Embedded

Kubernetes Distributions for Edge

- Resource-constrained hardware
  - IoT devices, Edge Gateways
- Several lightweight Kubernetes distributions
  - various capabilities
  - Security
  - Maintability
- Limited hardware management and context monitoring

	MicroK8s	k3s	k0s	MicroShift
Key Developer	Canonical	Rancher/SuSE	Mirantis	Red Hat
License	Apache 2.0	Apache 2.0	Apache 2.0	Apache 2.0
Enterprise Support	Yes	Yes	Yes	Yes
GitHub repo	https://github.com/canonical/microk8s	https://github.com/k3s-io/k3s	https://github.com/k0sproject/k0s	https://github.com/openshift/microshift
GitHub stars	6800	21200	105	406
Contributors	146	1796	65	46
First commit	May 2018	January 2019	July 2020	April 2021
Programming Language	Python, Shell	Go	Go	Go
CNCF certified	Yes	Yes	Yes	No
Vanilla Kubernetes	Yes	Yes	Yes	Yes
Single-node cluster	Yes	Yes	Yes	Yes
Multi-node cluster	Yes	Yes	Yes	n/a
Airgap cluster	Yes	Yes	Yes	Yes
High availability	Yes	Yes	Yes	n/a
GPU acceleration	Yes	Yes	Yes	Yes
Operating System	Ubuntu (default), Linux, Windows, MacOS	Linux	Linux, Windows Server 2019 (experimental)	RHEL, CentOS Stream, Fedora, (Windows, MacOS)
CPU Architecture	x86, ARM64, s390x, Power9	x86, ARM64, ARMhf	x86-64, ARM64, ARMv7	x86_64, ARM64, RISCV64
Deployment	Snap Package	Single Binary	Single Binary	RPM Package

Heiko Koziolek and Nafise Eskandani. 2023. Lightweight Kubernetes Distributions: A Performance Comparison of MicroK8s, k3s, k0s, and Microshift. In Proceedings of the 2023 ACM/SPEC International Conference on Performance Engineering (ICPE '23)

Edge Cluster Use Cases

- Open questions
  - How to manage/prevent compute speed degradation due to CPU thermal throttling?
  - Which mini Kubernetes distribution would be the most suitable for hardware-aware extensions?
  - How to incorporate predictive maintenance techniques into Kubernetes scheduling?
- Autoscaling
  - Scaling **below** zero
    - powering off nodes when the Knative functions are not used for a long time

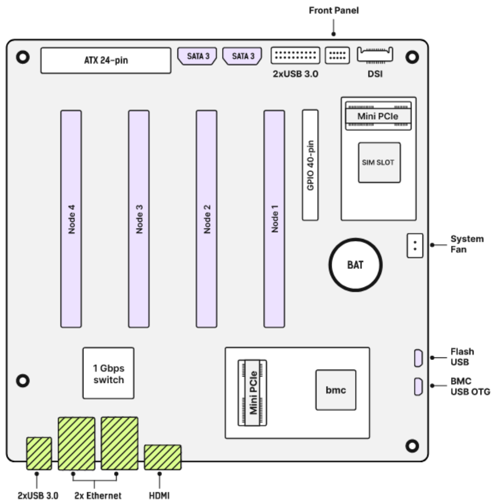
- Heterogeneous compute modules
  - Power-aware function instance selection
    - Low traffic -> slower modules
    - Heavy traffic -> faster modules
    - ML trafic -> GPU enabled modules
  - Variable ML processing quality
    - few models of various quality deployed on different compute modules
      - More precise -> GPU enabled CM
      - Less precise -> general purpose CM
    - Switching between various models based on traffic/energy

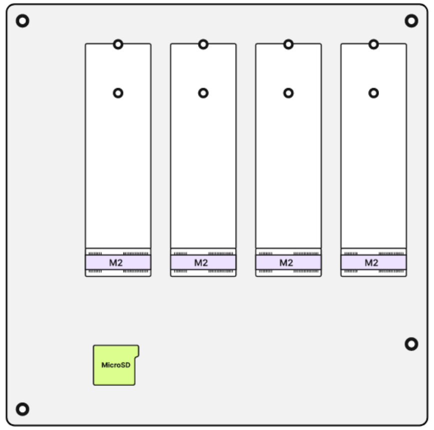
Heterogenous Edge Cluster

- Mainboard
  - Turing Pi v2 (https://turingpi.com/)
- Compute modules
  - 2 x Jetson Nano B01
  - Raspberry Pi CM4
- SSD 1TB

Turing Pi v2

- Turing Pi 2 board
  - equipped with an Allwinner T113-S3
    - provides BMC (Baseboard Management Controller) functionality -> (API on the right)
      - based on Linux
      - e.g. power on/off of the nodes
  - 4 x slots for compute modules
    - can be mixed
  - Network switch
    - VLAN support
  - Mini PCI Express slots for WAN connectivity

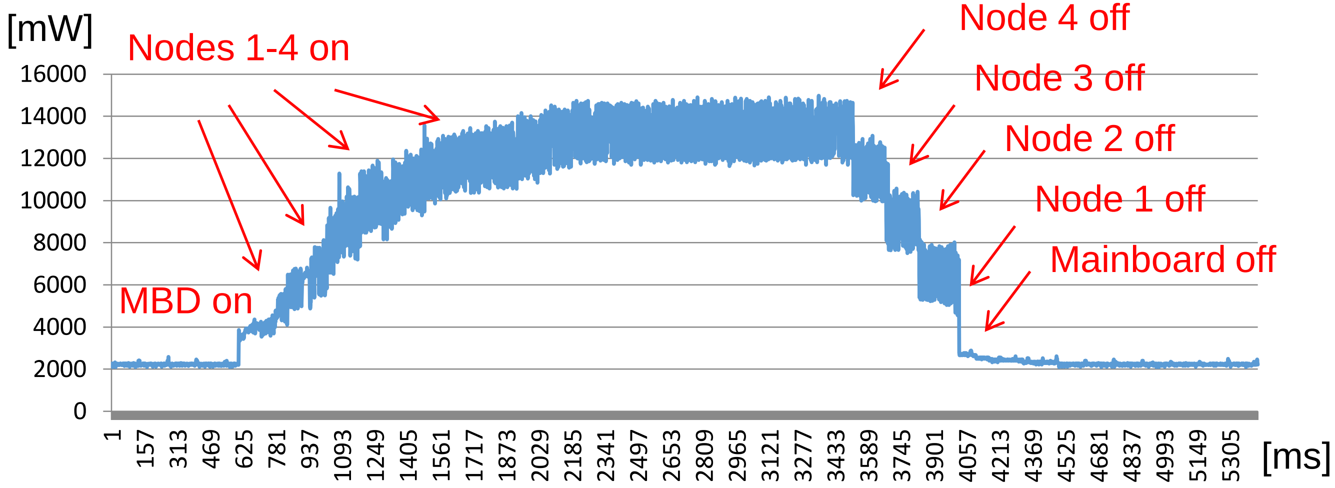





```
# tpi -h
Usage: tpi [host] <options...>
Options:
        -p, --power          (on off status) Power management
        -u, --usb            (host device status) USB mode,Must be used with the node command
        -n, --node           (1 2 3 4) USB selected node
        -r, --resetsw         reset switch
        -U, --uart           uart opt get or set
        -C, --cmd            uart set cmd
        -F, --upgrade <img>  upgrade fw
        -f, --flash <img>    flash an image to a specified node
        -l, --localfile      when flashing (-f), the specified file will be loaded locally from the device
        -m, --msd            load the node as mass storage device.
        -x, --clear_msd      pull rpiboot pin low and restart node.
        -h, --help           usage
example:
        $ tpi -p on //power on
        $ tpi -p off //power off
        $ tpi -u host -n 1 //USB uses host mode to connect to Node1
        $ tpi --uart=get -n 1 //get node1 uart info
        $ tpi --uart=set -n 1 --cmd=ls//set node1 uart cmd
        $ tpi --upgrade=/mnt/sdcard/xxxx.swu    //upgrade fw
        $ tpi -r  //reset switch
        $ tpi -n 1 -l -f /mnt/sdcard/raspios.img  // flash image file to node 1
        $ tpi -m -n 1  //(Rpi only) load the MSD driver. When executed successfully,
        // log into the BMC and use 'dmesg' to see the names of the new block devices,
        // and mount them as you wish.
        $ tpi -x -n 1  // clear msd node and restart
```

In-Lab Edge Cluster Prototype

- Power monitoring
  - Voltage and current (INA219)
- Fan vibration monitoring
- RPi Pico board for environmental context monitoring with TinyML



Summary

- Edge Computing is gaining momentum
  - EdgeAI & TinyML applications
- Heterogenous Edge Clusters
  - Fine grained workload management
  - Power control
  - Redundancy
- Cluster management and scheduling
  - Should hollistically cover also the hardware