eSAAM 2023

on Cloud to Edge Continuum

On the Containerization and Orchestration of RISC-V architectures for Edge-Cloud computing

Francesco Lumpp (University of Verona, Italy)
francesco.lumpp@univr.it

Oct. 17, 2023

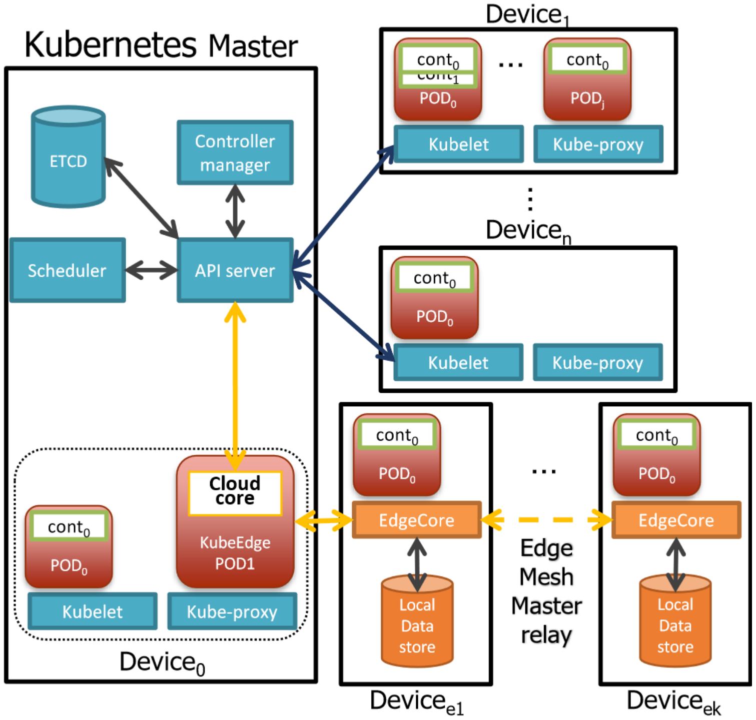
Ludwigsburg, Germany

Kubernetes on RISC-V

- There is considerable interest in open hardware architectures, covering the whole computing continuum spectrum from cloud to edge
- RISC-V architectures play an important role in this context for both academic and industrial product designs
- The potential of these architectures in the context of the computing continuum is untapped due to lack of software support
- We propose a port of the containerization and orchestration software stack on RISC-V, as well as in-depth analysis of the overhead characteristic of such a stack when compared to RISC-V

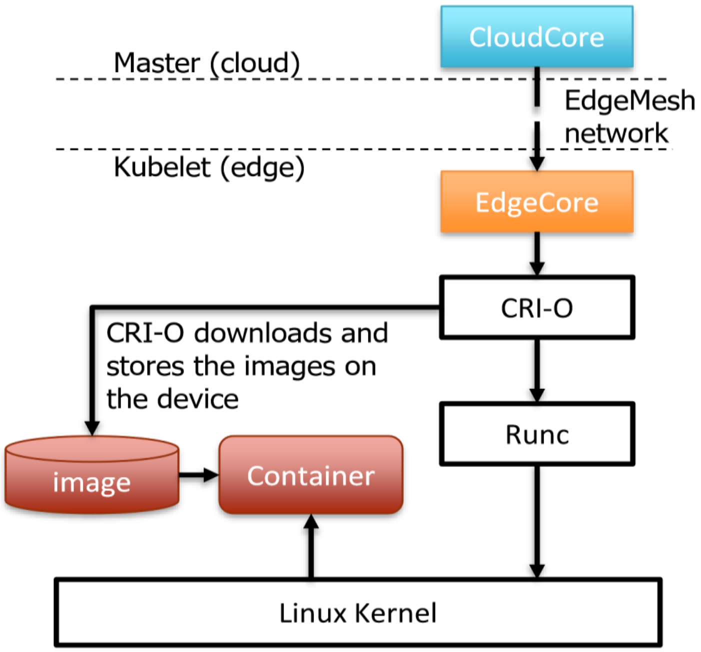
KubeEdge and RISC-V

- For orchestration, we used KubeEdge instead of Kubernetes
- KubeEdge focuses towards IoT and Edge-Cloud computing continuum (with RISC-V on the edge) thanks to more efficient design
- KubeEdge interfaces with Kubernetes, but Kubernetes must still be used on the Cloud side
- Ideal for harsh environments



KubeEdge and RISC-V

- None of the software exists for RISC-V, so the complete software stack had to be adapted and recompiled
- Starting from the high-level runtimes, such as KubeEdge itself, down to the low-level software responsible for running containers, such as *runc*
- All dependencies and libraries also had to be adapted and recompiled



KubeEdge performance
RISC-V vs ARM

- The platform was successfully ported
- To verify its performance, it was tested on a RISC-V board and a similar-class ARM board, configured with the same power target and the same number of CPU cores
- For the RISC-V architecture, we used the Monte Cimone cluster, which employes the SiFive HiFive Unmatched
- For the ARM architecture, we used the Nvidia Jetson Xavier

The Monte Cimone Server Blade hosts two SiFive HiFive Unmatched boards, each with a Freedom U740 SoC. It has a form factor of 4.44 cm (1 Rack-Unit) in height

KubeEdge performance

RISC-V vs ARM (cont'd)

- A wide number of benchmarks was used to verify the container overhead, using tests to analyze the following:
  - Memory bandwidth
    - Sysbench, Stream
  - Applications performance
    - Rodinia (compute benchmark), 7-Zip, x265 Encode/Decode, POVRay (ray-tracing), OpenSSL (hash/sign/verify/encrypt)
  - System performance
    - OsBench, IPCbench, Stress-ng

Memory Bandwidth

- The system memory on RISC-V behaved better when containerized compared to ARM
- RISC-V: +0.4% --- ARM: -3.8%

Suite	Test	RISC-V			ARM64			Unit
		Native	KubeEdge	Overhead	Native	KubeEdge	Overhead	(▲▼)
Sysbench	CPU	2.47 ± 0.39%	2.46 ± 0.19%	-0.4%	6.40 ± 0.06%	6.17 ± 1.72%	-3.7%	event/s ▲
Stream	Copy	1 294 ± 0.28%	1 274 ± 2.19%	-1.5%	22 765 ± 0.51%	20 500 ± 0.15%	-10.0%	MiB/s ▲
	Scale	1 079 ± 0.50%	1 096 ± 1.01%	1.6%	23 929 ± 0.42%	22 229 ± 0.09%	-7.1%	MiB/s ▲
	Add	1 181 ± 0.20%	1 180 ± 0.89%	-0.1%	24 866 ± 0.10%	24 719 ± 1.46%	-0.6%	MiB/s ▲
	Triad	1 165 ± 0.18%	1 191 ± 1.94%	2.2%	24 499 ± 0.25%	25 044 ± 0.16%	2.2%	MiB/s ▲
			Average:	0.4%		*Average*:	-3.8%	

Applications performance

- The applications’ performance on RISC-V, once again, behaved better when containerized compared to ARM, especially in OpenSSL
- RISC-V: -1.9% --- ARM: -3.1%

Suite	Test	RISC-V			ARM64			Unit
		Native	KubeEdge	Overhead	Native	KubeEdge	Overhead	(▲▼)
Rodinia	LavaMD	12 298 ± 1.05%	13 462 ± 0.41%	-8.7%	1 731 ± 2.41%	1 742 ± 2.05%	-0.6%	s ▼
x265 3.4 [1e-3]	Bos. 1080p	150 ± 0.00%	140 ± 0.00%	-6.7%	1 240 ± 0.00%	1 230 ± 0.24%	-0.8%	fps ▲
	Bos. 4K	30 ± 0.00%	30 ± 0.00%	0.0%	340 ± 0.00%	330 ± 0.00%	-2.9%	fps ▲
7-Zip	Comp.	1 782 ± 0.80%	1 805 ± 0.77%	1.3%	7 972 ± 2.63%	6 983 ± 3.60%	-12.4%	MIPS ▲
	Decomp.	3 433 ± 0.54%	3 430 ± 0.13%	-0.1%	5 921 ± 1.36%	5 309 ± 1.10%	-10.3%	MIPS ▲
POV-Ray	Trace	2 948 ± 1.33%	2 948 ± 1.79%	-0.0%	541 ± 2.38%	541 ± 2.38%	0.0%	s ▼
OpenSSL	SHA256	66.1 ± 0.38%	63.1 ± 5.89%	-4.7%	1 589.5 ± 2.45%	1 593.1 ± 0.69%	0.2%	MB/s ▲
	SHA512	92.7 ± 1.08%	90.4 ± 0.63%	-2.5%	398.5 ± 0.38%	387.1 ± 0.40%	-2.9%	MB/s ▲
	RSA4096_s	41 ± 0.00%	42 ± 0.73%	0.5%	155 ± 0.47%	147 ± 0.48%	-5.2%	sign/s ▲
	RSA4096_v	3 139 ± 0.28%	3 124 ± 0.69%	-0.5%	11 01 ± 0.01%7	10 532 ± 0.05%	-4.4%	verify/s ▲
	AES-128	65.0 ± 0.37%	64.1 ± 0.12%	-1.5%	4 964.8 ± 0.07%	4 866.4 ± 0.09%	-2.0%	MB/s ▲
	AES-256	53.9 ± 0.55%	53.2 ± 0.62%	-1.3%	3 677.2 ± 0.01%	4 027.0 ± 0.05%	9.5%	MB/s ▲
	ChaCha20	231.1 ± 0.27%	227.3 ± 0.09%	-1.7%	2 213.2 ± 0.01%	2 080.9 ± 0.08%	-6.0%	MB/s ▲
	Poly1305	168.0 ± 0.26%	165.7 ± 0.33%	-1.4%	1 497.4 ± 0.04%	1 415.9 ± 0.03%	-5.4%	MB/s ▲
			Average:	-1.9%		*Average*:	-3.1%	

System performance

Suite	Test	RISC-V			ARM64			Unit
		Native	KubeEdge	Overhead	Native	KubeEdge	Overhead	(▲▼)
OSBench	Create Files	426 ± 5.48%	596 ± 4.32%	-28.4%	183 ± 2.28%	279 ± 4.71%	-34.3%	μs ▼
	↳ mount	-	425 ± 5.42%	0.2%	-	190 ± 3.64%	-3.7%	μs ▼
	C. Thread	197 ± 2.15%	212 ± 0.28%	-7.1%	136 ± 2.27%	171 ± 2.06%	-20.6%	μs ▼
	C. Processes	402 ± 2.37%	452 ± 2.47%	-10.9%	262 ± 1.59%	272 ± 1.05%	-3.7%	μs ▼
	Launch Prog.	618 ± 1.07%	673 ± 0.23%	-8.2%	924 ± 0.71%	818 ± 1.32%	13.0%	μs ▼
	Malloc	1 399 ± 0.43%	1 424 ± 1.19%	-1.8%	565 ± 1.49%	542 ± 0.23%	4.2%	μs ▼
IPC bench.	TCP Socket	117 705 ± 5.01%	112 507 ± 14.08%	-4.4%	137 263 ± 2.35%	136 135 ± 0.99%	-0.8%	msg/s ▲
	PIPE Un.	270 793 ± 0.36%	274 776 ± 1.96%	1.5%	154 689 ± 0.25%	152 937 ± 0.38%	-1.1%	msg/s ▲
	PIPE FIFO	269 931 ± 1.24%	266 265 ± 1.35%	-1.4%	155 666 ± 0.33%	157 966 ± 0.47%	1.5%	msg/s ▲
	Unix Socket	95 177 ± 2.17%	95 469 ± 0.94%	0.3%	90 560 ± 1.13%	92 266 ± 1.37%	1.9%	msg/s ▲
Stress-ng	Mutex	158 964 ± 2.16%	135 805 ± 0.52%	-14.6%	57 472 ± 4.30%	56 235 ± 2.49%	-2.2%	ops/s ▲
	Malloc	99 067 ± 0.17%	97 948 ± 0.98%	-1.1%	54 320 ± 1.16%	52 001 ± 0.22%	-4.3%	ops/s ▲
	Forking	1 851 ± 2.34%	2 020 ± 1.98%	9.1%	1 601 ± 3.07%	1 684 ± 12.59%	5.2%	ops/s ▲
	Pthread	2 690 ± 1.07%	2 340 ± 0.53%	-13.0%	3 633 ± 2.06%	3 473 ± 0.91%	-4.4%	ops/s ▲
	CPU cache	23 254 ± 4.07%	23 549 ± 8.91%	1.3%	182 042 ± 1.59%	177 345 ± 1.07%	-2.6%	ops/s ▲
	Semaphores	845 130 ± 2.05%	793 821 ± 2.37%	-6.1%	201 253 ± 9.87%	194 155 ± 5.54%	-3.5%	ops/s ▲
	Matrix Math	518 ± 0.22%	507 ± 0.17%	-2.1%	3 698 ± 0.15%	3 770 ± 0.05%	1.9%	ops/s ▲
	Vector Math	348 ± 0.33%	354 ± 0.02%	1.8%	6 484 ± 0.53%	7 084 ± 0.69%	9.3%	ops/s ▲
	Functions	2 294 ± 0.36%	2 249 ± 0.36%	-2.0%	18 766 ± 2.58%	16 004 ± 2.43%	-14.7%	ops/s ▲
	Cntx switch	84 110 ± 4.90%	66 082 ± 5.53%	-21.4%	47 990 ± 2.48%	47 865 ± 2.39%	-0.3%	ops/s ▲
			Average:	−5.4%		*Average:*	−2.9%	

- There is a quite significant difference in the system-level performance
- Especially when considering the context switch duration on RISC-V when containerized
- To understand this extreme overhead, we performed some additional analysis

System performance (cont'd)

- There is a clear difference in the context switch performance, which might be caused by two factors:
  1) The container data structures themselves inside the Linux kernel
  2) Some part of the software stack for containerization is not optimized on RISC-V
- Considering that all system calls are intercepted by the container runtime, we **manually** created a container **without** container runtime, this should differentiate between the two overhead cases
  a) Overhead -> container data structures
  b) No overhead -> container runtime
- If the "manual" containerization works, the container should behave normally

System performance (cont'd)

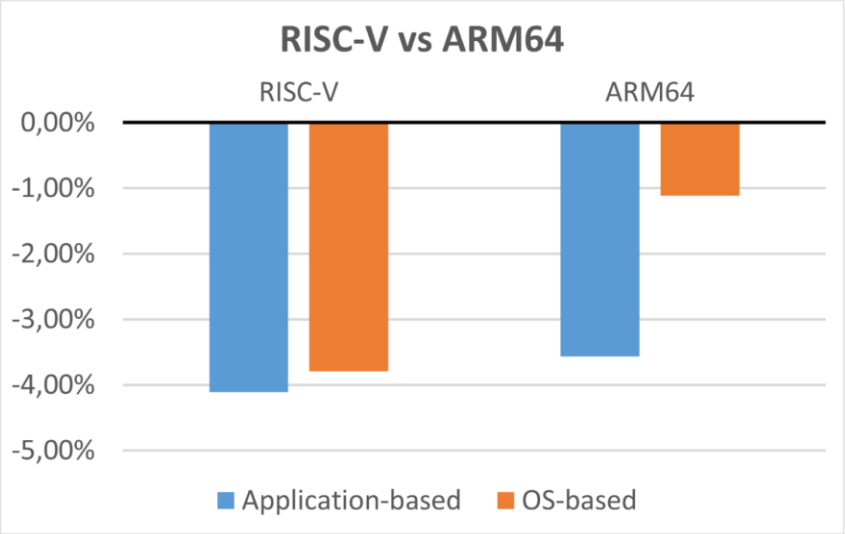
- We impose a limit of 128MB on the container and run the context switch benchmark again
- If the container is killed once it overflows 128MB, then the container is behaving normally, and we can verify the context switch time
- We also ran the benchmark natively on the system, without containers, as ground truth

	Native	Manual	KubeEdge
Syscall time	192.18 ns	191.72 ns	206.48 ns
OOM kill	No	Yes	Yes

- The manual container is working and has no overhead, thus, the container runtime is causing the additional overhead

Summary

- Performance of RISC-V containerization is very good for compute-intensive applications
- Applications that make heavy use of system calls and context switches could become significantly slower when containerized, compared to ARM
- Overall, the performance delta between ARM and RISC-V is comparable



Conclusions

- We ported the containerization and orchestration stack to RISC-V, paving the way for Edge-Cloud computing continuum on an entirely new architecture
- We verified the software readiness of such an architecture for containerization and found that some optimization is missing in the container runtime for system calls and context switching
- The software stack and installation guide is openly available at: gitlab.com/parco-lab/kubeedge-v

eSAAM 2023

on Cloud to Edge Continuum

Thank you!

Questions?

Sponsored by:

EUCloudEdgeIoT.eu CODECO NEMO nephele

Organized by:

POLITÉCNICA ECLIPSE FOUNDATION HELLENIC REPUBLIC UNIVERSITY OF MACEDONIA