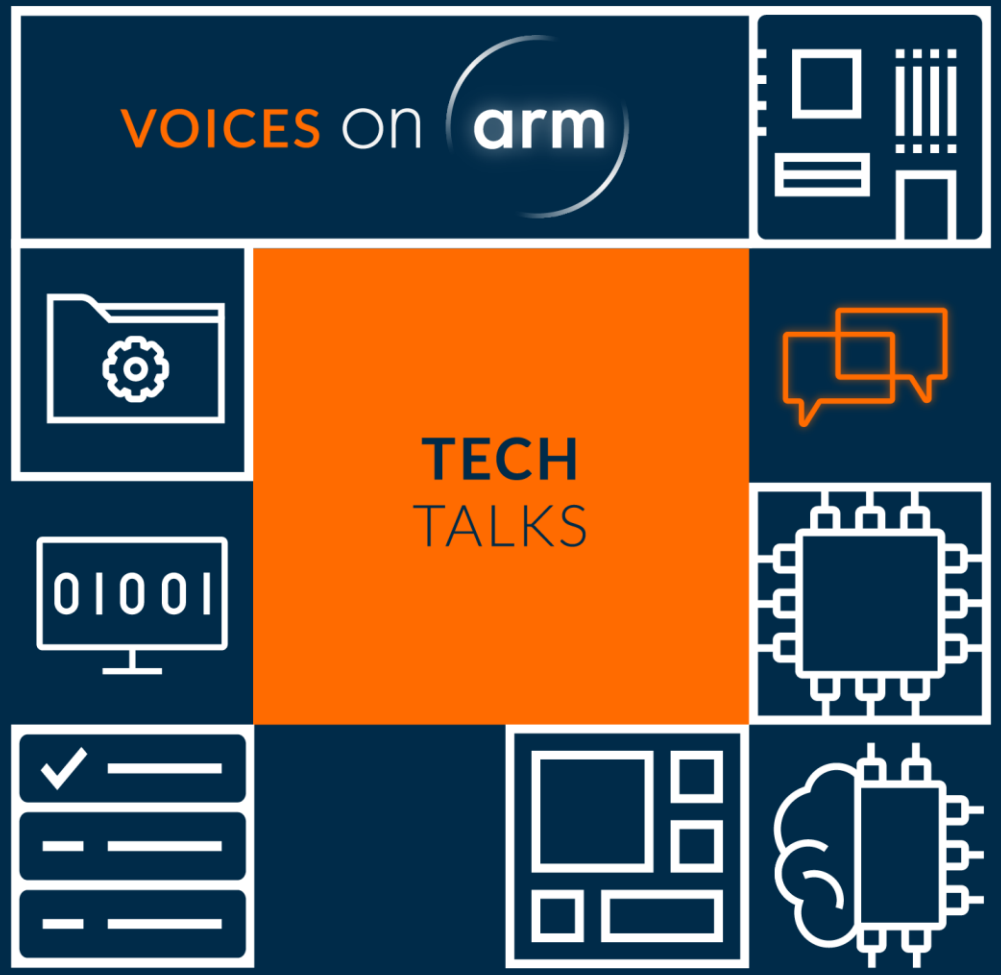
arm

Bringing Streaming Analytics to Arm-based Edge Devices



Johan Risch Lead Developer
January 31st 2023

Welcome!

Tweet us: #ArmTechTalks

View tech talks on-demand:
www.youtube.com/arm

Sign up for upcoming tech talks:
www.arm.com/techtalks

Our Upcoming Arm Tech Talks

Date	Title	Host
January 31st	Bringing Streaming Analytics to Arm-based Edge Devices	Stream Analyze
February 7th	Build Home Automation Services on a Matter Compliant Smart Home Hub Using Python	Arm & Canonical
February 14th	Shifting IoT Software Development to the Cloud with Arm Virtual Hardware enabled GitHub Actions	GitHub
February 21st	Securing IoT with Cloud Native Tooling, PARSEC and AWS Greengrass	56k Cloud
February 28th	How to reduce Friction at the Edge and Bootstrap Your IoT Projects	Eurotech
March 7th	Fast development of noise detection ML models: Qeexo AutoML and Arm Virtual Hardware	Qeexo

arm

Stream Analyze

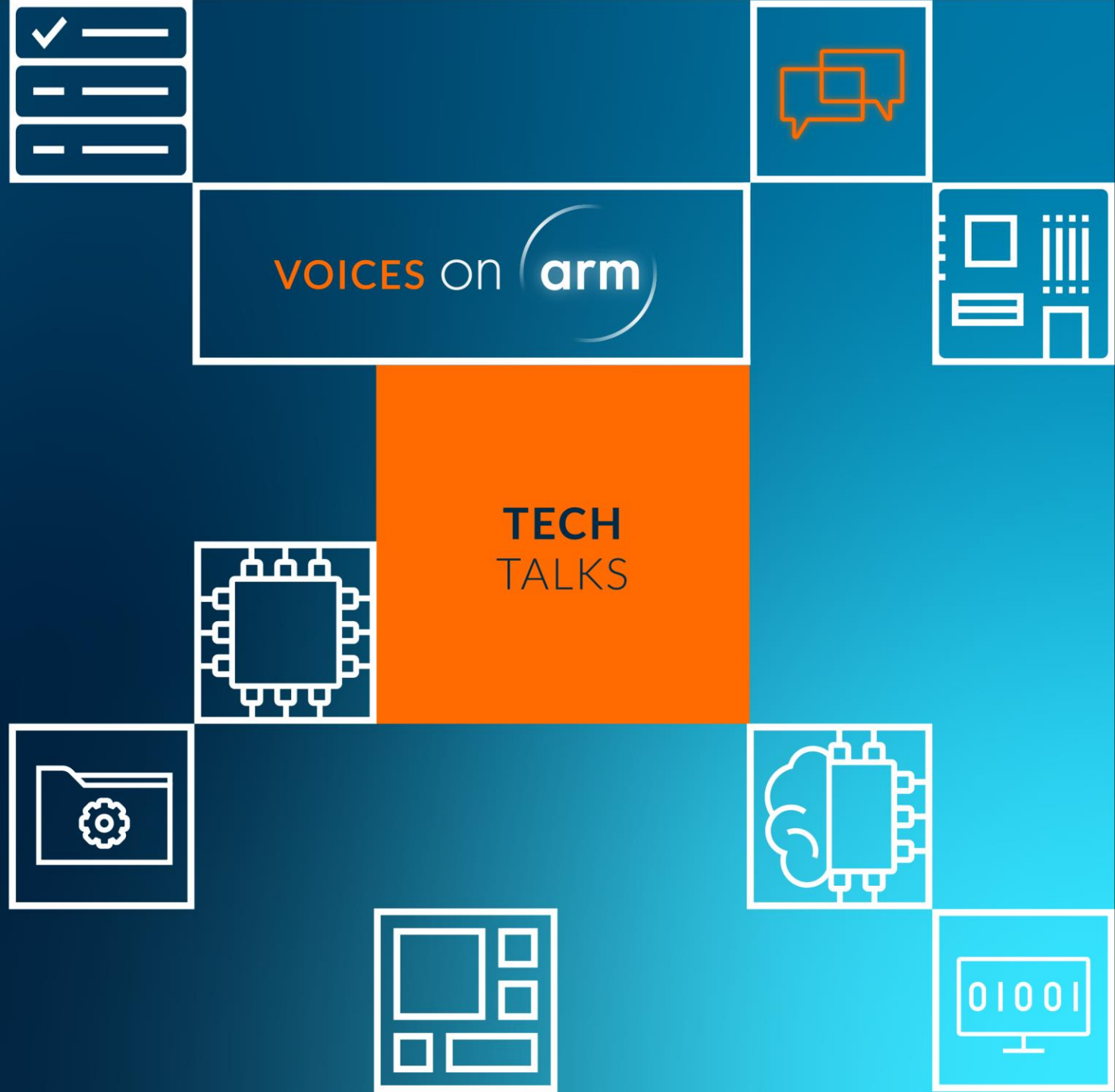
Johan has extensive experience in software development and implementation of AI solutions. Johan is a lead developer for the Stream Analyze platform and work on developing the core of SA Engine, implementation of AI models including Neural Networks, the cloud-based version of the platform and much more.

Johan Risch

Lead Developer

arm

SA Engine



SA Engine

Built in

- Main Memory Database.
- Data Stream Management System (DSMS).
- Computation engine.
- Inference engine.

Footprint

- 20kB – 6MB RAM.
- Bare metal, RTOS, OS.

SA Engine

capabilities

- SA Engine **will**
  - Allow you to query data streams in real-time on any connected device
  - Also run completely autonomous when needed
  - Use the built-in main memory database to update models and queries without changing the firmware
  - Allow you to use the best and most advanced query language there is for streaming data and running advanced analytical, ML and DL models
  - **Just-In-Time compile your queries into machine code to run on the edge device.**
  - Use any available inference runtime to run DL models (SA.NN, tflite, OpenVINO, etc.)
  - Change the way you look at, and approach, edge analytics.*
- SA Engine **will not**
  - Create a highly optimized and quantized neural network.
  - Update the device firmware using FOTA.
  - Only do (NN) inference.

SA Engine + Arm

The only viable architecture for implementing Edge Analytics at scale.

- From the embedded edges to the cloud orchestration – Arm excels at every level.

SA Engine has a long history of running on Arm processors.

- First port was back in 2015 on an industrial Android device.
- Today we have run SA Engine on edge-devices with Arm processors from: Cortex-M3 to Cortex-A715.
- Our scaling tests utilize the 2x Ampere Altra Q80 to simulate ~25 000 edges.

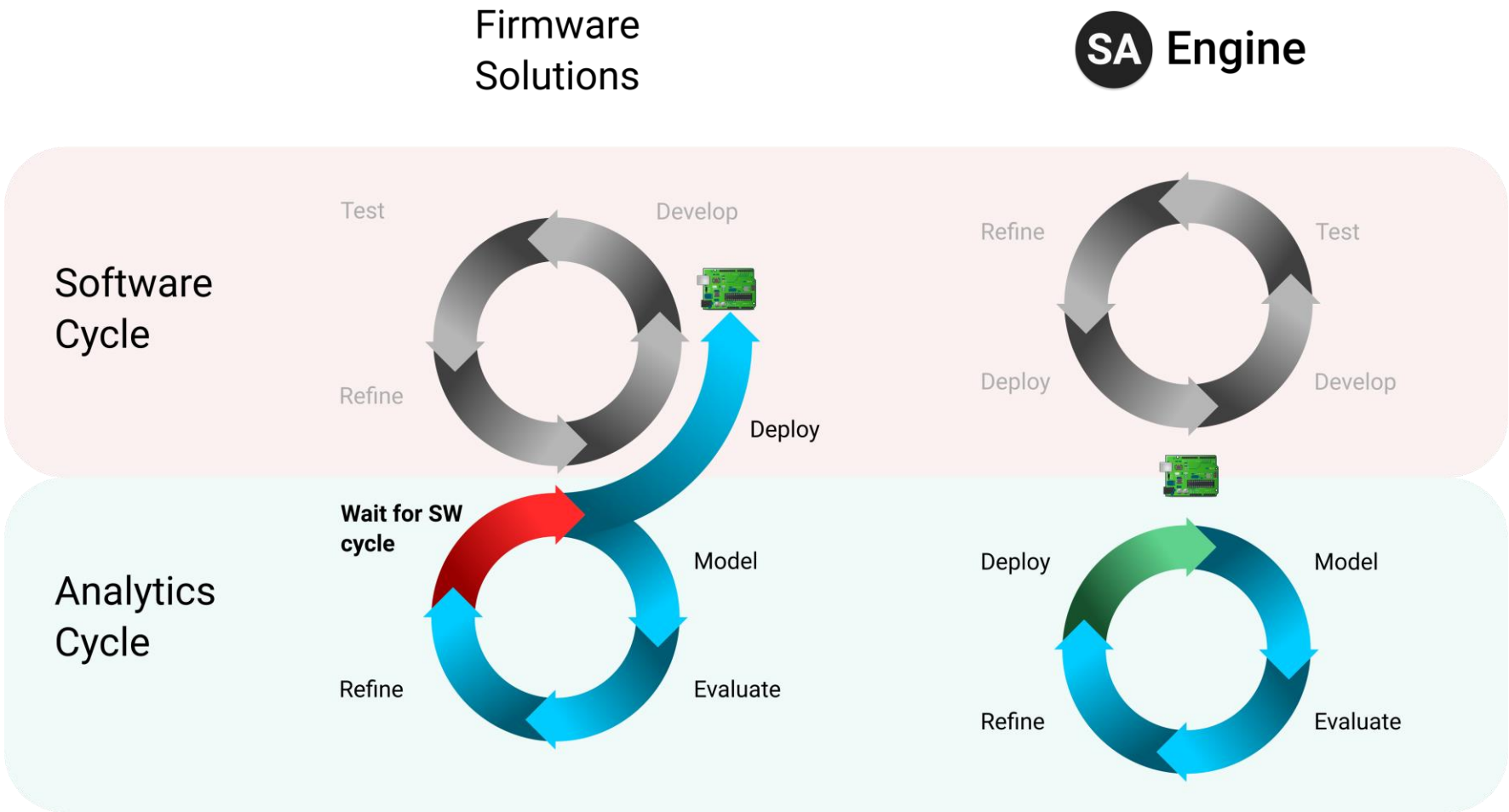
Arm is the obvious choice for us.

Lower the bar for entry into Edge Analytics

1. Complex to implement analytical models (programming)
2. Resource constrained device.
3. Time consuming or risky to update target.

1. High level query language which is declarative. → allows for a much larger user-base than regular programming languages.
2. Optimize query and finally JIT compile it to machine code → Can even beat C implementations.
3. Deploy analytics directly onto running system → No firmware updates needed during analytical process.

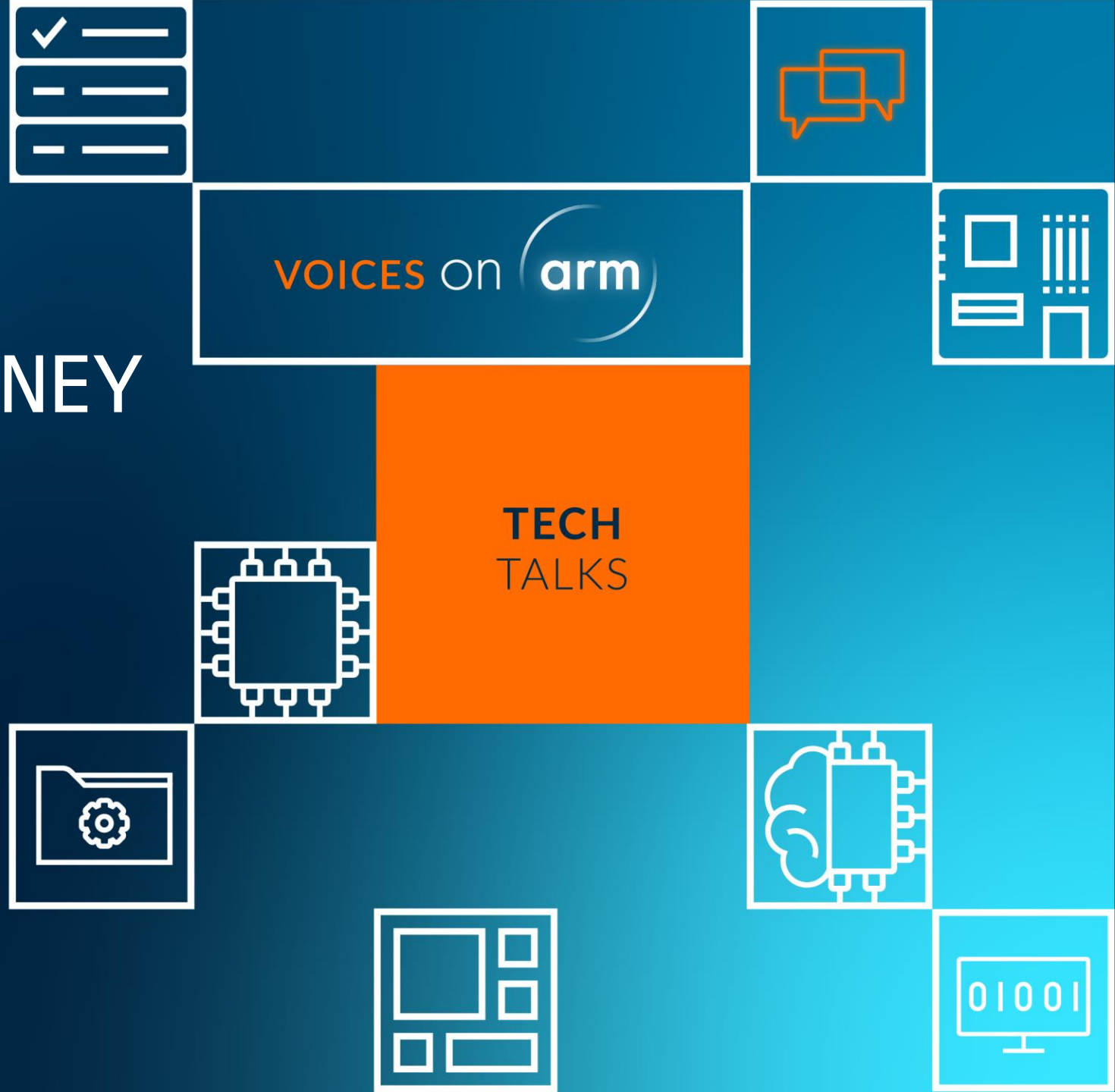
Software Cyle vs Analytics Cycle



arm

arm

SEND + MORE = MONEY



SEND+MORE=MONEY

Verbal arithmetic - Each letter is a digit. No two letters can be the same digit. The leading digit of a multi-digit number must not be zero.

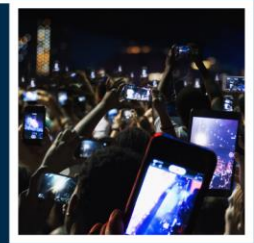
We will compare the performance of:

- SA Engine - https://gist.github.com/johanrisch/db6d4ad7a0ba931814a2dfc1468cbd38
- C - https://gist.github.com/jeremieroy/584216655d60eac06ae3
- Python - https://programmingpraxis.com/2012/07/31/send-more-money-part-1/ posted by *Catalin Cristu*
- Gecode - Built-in example
- PostgreSQL - https://gist.github.com/johanrisch/5a7b8b64d255cc89cb7e5f56ef9d9dbb

arm

DEMO TIME

VOICES on arm



TECH TALKS

SEND+MORE=MONEY results

Problem fomulation	SA Engine(s)	C(s)	Python(s)	Gecode(s)	PostgreSQL(s)
Regular	0.005	0.01	2	2e-5	5.642
M=1	5e-4	N/A	0.2	2e-5	0.863
Linear algebra	8e-5	N/A	N/A	2e-5	0.354

arm

Going from a Query to running on an Arm Cortex-M4

VOICES on arm

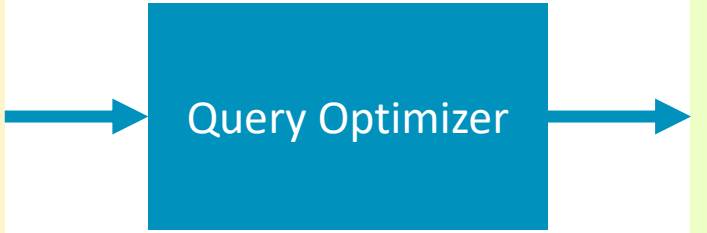
TECH TALKS

Running a query in SA Engine

SLOG: Streamed LOGic

SLOG Execution plan

```
select x
  from Real x,
       Integer i
 where i in range(10)
   and x = sin(i)
```



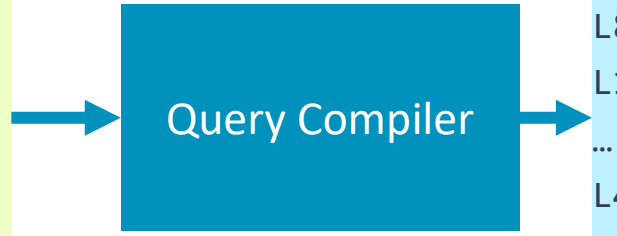

```
(CREATE-FUNCTION *SELECT*
   (-> (REAL X+))
   (LOCALS: (INTEGER I))
   DO (AND (CALL #[extpred \"IOTA--+\"]
            (IOTA 1 10 I+))
           (FUNCALL FUNCTION:SIN
              (SIN I- X+))))
```


arm

Running a query in SA Engine

SLAP: Streaming Logic Assembly Program

```
(CREATE-FUNCTION *SELECT*
   (-> (REAL X+))
   (LOCALS: (INTEGER I))
   DO (AND (CALL #[extpred \"IOTA--+\"]
           (IOTA 1 10 I+))
        (FUNCALL FUNCTION:SIN
           (SIN I- X+))))
```

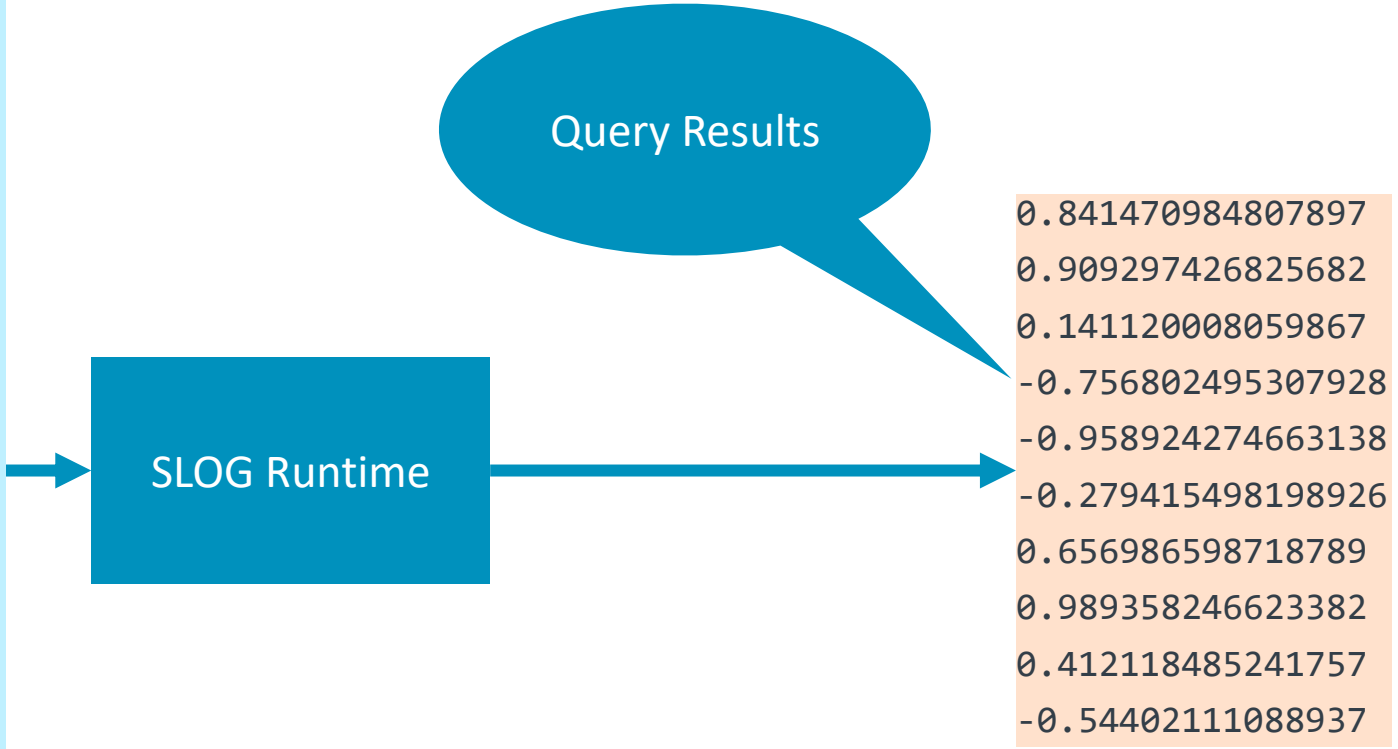


SLOG+SLAP Execution plan

```
(CREATE-FUNCTION *SELECT*
   (-> (REAL X+))
   DO (CALL-SLAP \"
         .code16 ; ARMv7 thumb-2 VFPv4
         .thumb
L0:      push    {r3, r4, r5, r6, r7, lr}
L2:      mov     r4, r0
L4:      mov     r5, r1
L6:      movs    r6, #1
L8:      movs    r7, #0
L10:     bl      L30
…
L48:     ldr     r3, [r4, #96]   ; SIN
L50:     blx     r3
L52:     vmov.f64 d2, d0
L56:     vstr    d2, [r5, #8]
L60:     mov     r0, r5
L62:     ldr     r3, [r4, #0]    ; CONTFN
…
L68:     pop     {r3, r4, r5, r6, r7, pc}
         end
X+))
```

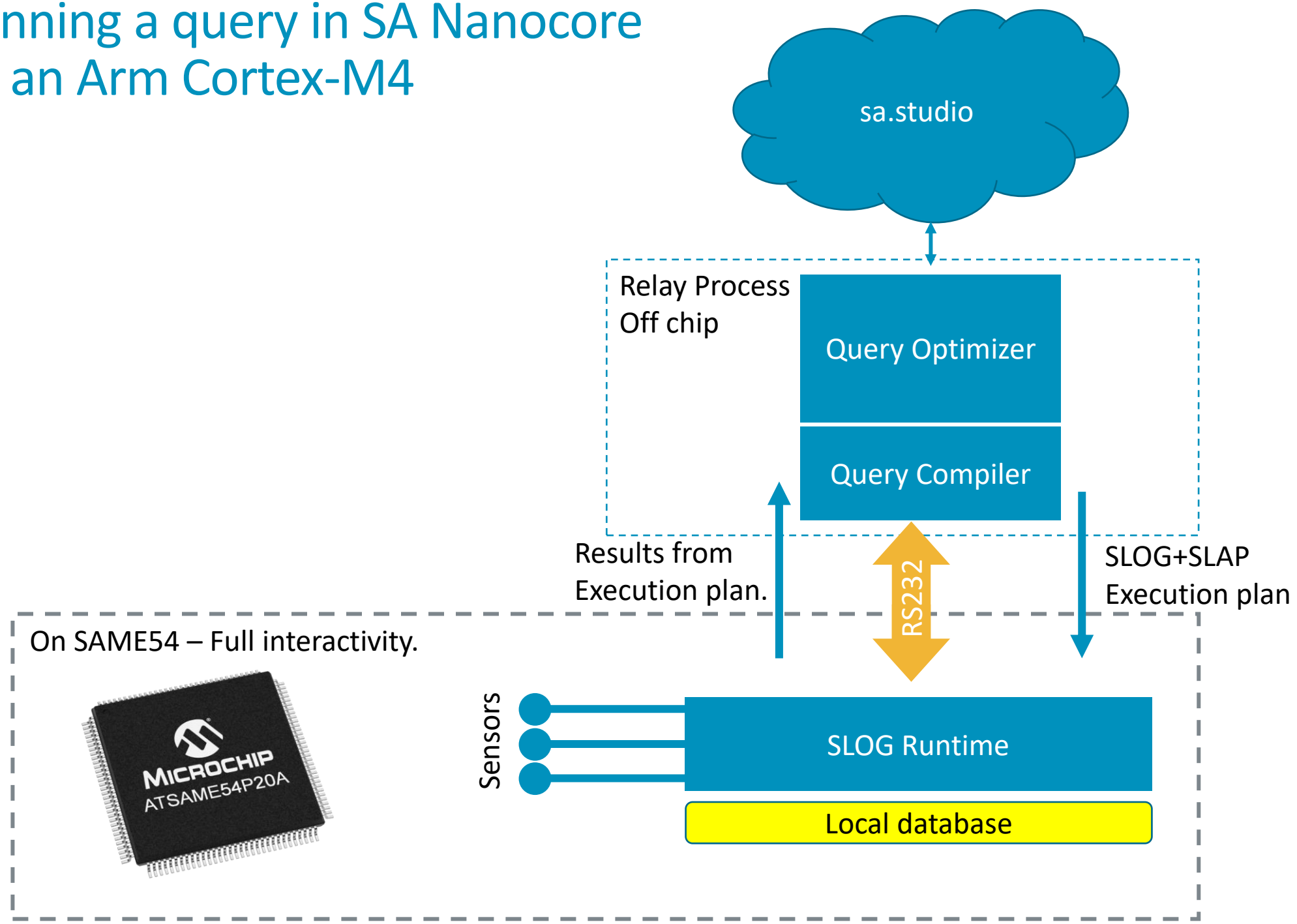
Running a query in SA Engine

```
(CREATE-FUNCTION *SELECT*
   (-> (REAL X+))
   DO (CALL-SLAP \"
        .code16 ; ARMv7 thumb-2 VFPv4
        .thumb
L0:     push    {r3, r4, r5, r6, r7, lr}
L2:     mov     r4, r0
L4:     mov     r5, r1
L6:     movs    r6, #1
L8:     movs    r7, #0
L10:    bl      L30
…
L48:    ldr     r3, [r4, #96]   ; SIN
L50:    blx     r3
L52:    vmov.f64 d2, d0
L56:    vstr    d2, [r5, #8]
L60:    mov     r0, r5
L62:    ldr     r3, [r4, #0]    ; CONTFN
…
L68:    pop     {r3, r4, r5, r6, r7, pc}
        end
X+))
```



arm

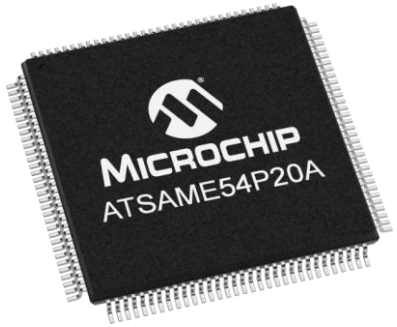
Running a query in SA Nanocore on an Arm Cortex-M4

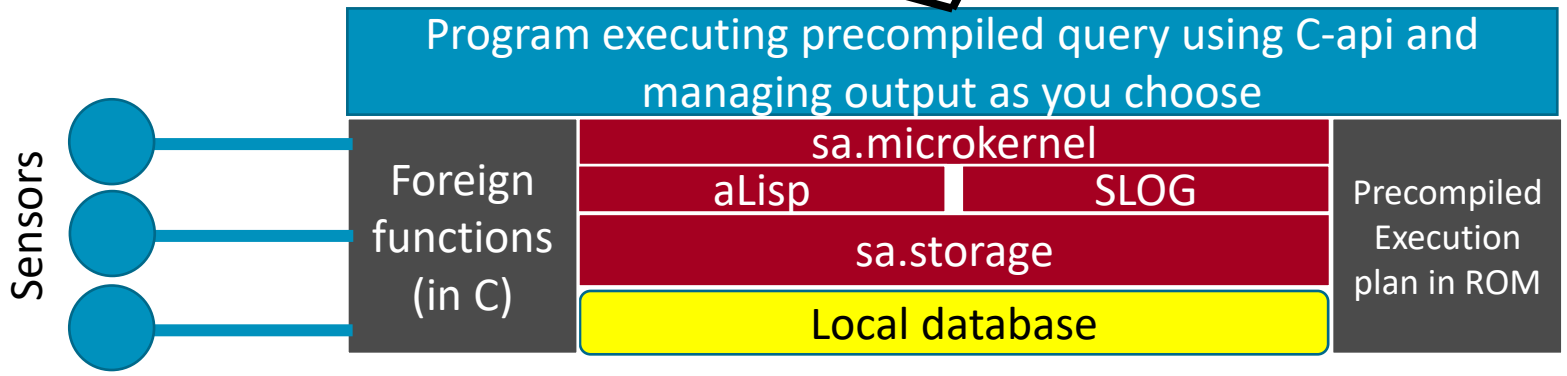


```
ohandle callback(bintype env, ohandle data) {
… User defined callback…
}

void main() {
   sa_evaluate(PRECOMPILED_QUERY_STR, &callback);
}
```

SAME54 – Canned models



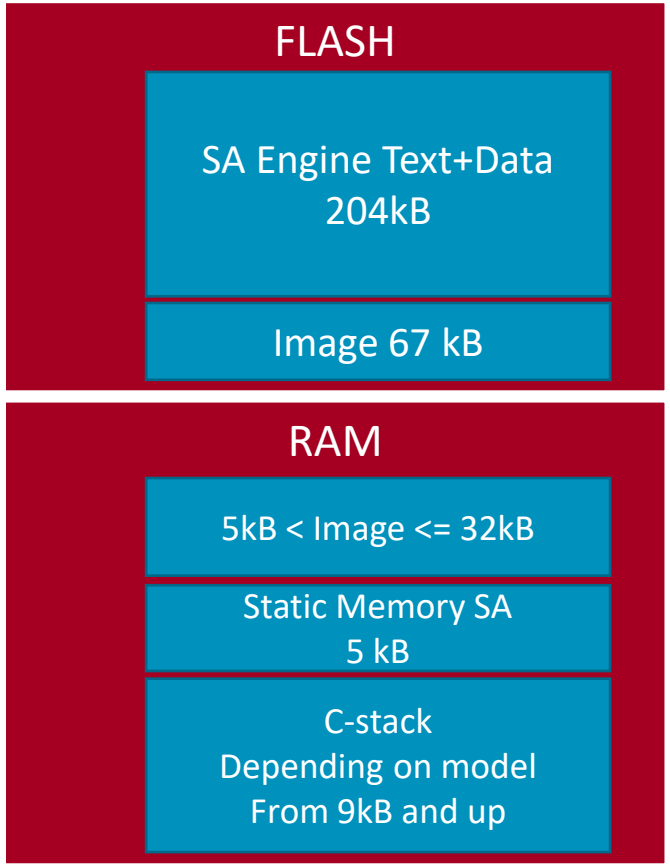


arm

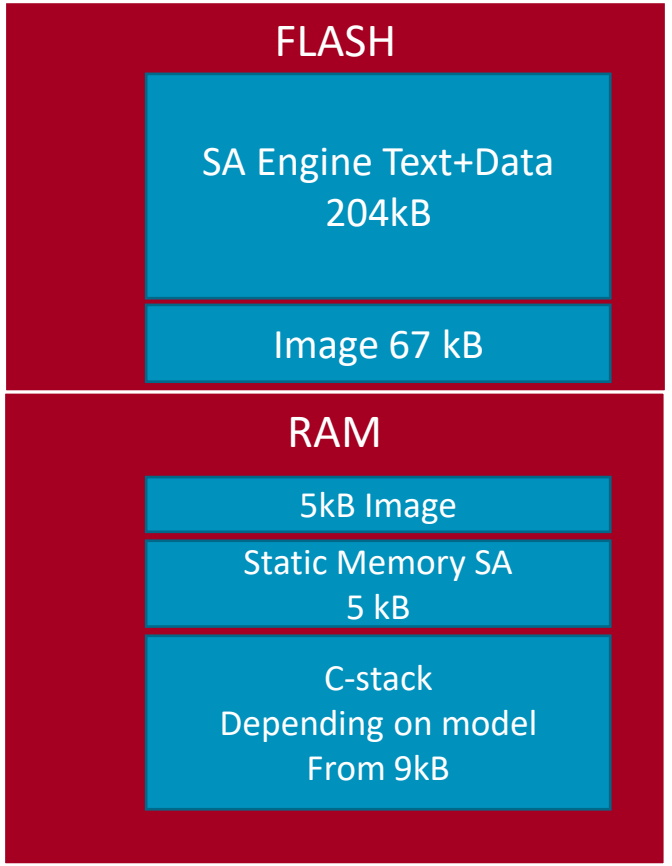
Running SA Nanocore on an MCU

Memory Requirements in detail.

General footprint of SA Nanocore



Minimal memory to boot SA Nanocore



arm

DEMO TIME

VOICES on arm

TECH TALKS

What’s next?

- We are continuously working on compiling more and more SLOG to SLAP.
  - The latest addition made it possible to define convolutions over images in OSQL almost fully compiled.
- Optimize SA Nanocore on flash size.
  - We have not yet trimmed the c-code flash size for SA Nanocore.
- Add more off the shelf H/W platforms for users to test.
  - 10 more platforms in the coming two years.
- Improve UX by making the setup easier to configure.
- Generate Canned C-programs from a model defined in OSQL.

arm

Tweet us: #ArmTechTalks

View tech talks on-demand:
www.youtube.com/arm

Sign up for upcoming tech talks:
www.arm.com/techtalks

Thank You
Danke
Gracias
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה