NOP - Safe & Efficient Intermittent Computing for Battery-less IoT

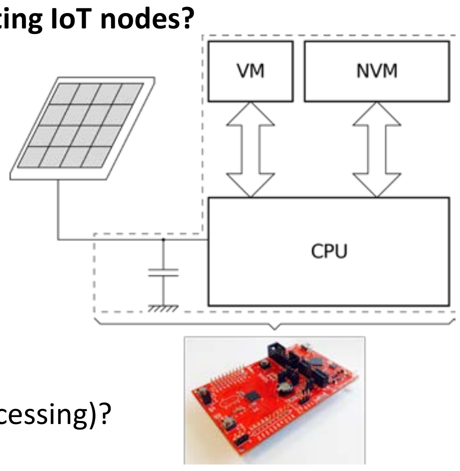
ANR, CominLabs, Investissements d'avenir, IETR, Inria, IRISA, LS2N

Battery-less Energy Harvesting IoT nodes

Eliminating electro-chemical batteries from Energy Harvesting IoT nodes?
- Reduce environmental footprint
- Extend lifespan while decreasing needs for maintenance

Building blocks already available
- Efficient and fast non-volatile memories (NVM)
- Ultra-low-power microcontrollers
- Supercapacitors

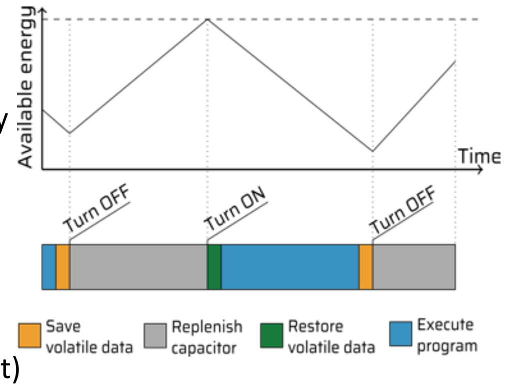
What about energy availability?
- In limited quantity at a given time
- Too limited to perform complete functions (AI, signal processing)?



No! The solution is to weave together computation steps and idle periods to spread the execution of complex functions over several charges ➞ intermittent computing

Safe & Efficient Intermittent Computing

Intermittent system
- Compute when possible
- Deal with intermittency (checkpoint / restore)
- Idle when energy is low

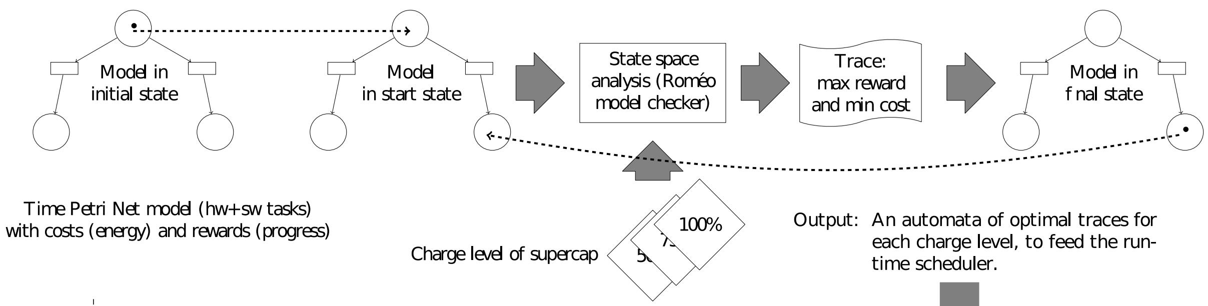


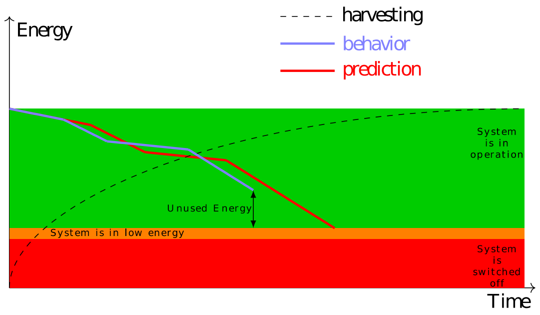
Efficiency
- No useless computations
- Minimize overhead (intermittency management)
- Consider energy harvesting

Safe intermittency
- No uncontrolled power failure
- Static guarantees (*e.g.*, atomicity or forward progress)

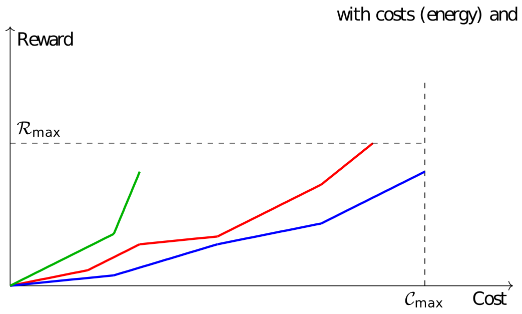
Overview of Progress #1: Optimal synthesis of intermittent execution traces

Objectives
- Hw & Sw tasks (true parallelism), reactive and periodic functions
- Worst-case based analysis
- Optimal schedule for each computation step
- Weave optimal computation steps and idle periods online

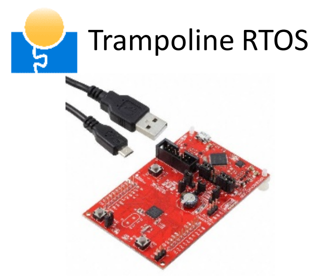


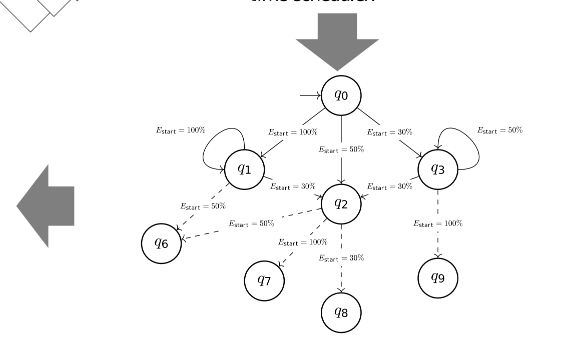


Charge level after a computation step?
➞ Measure + quantization



Optimal schedule is the red one. How to find it? ➞ formal models (see above)





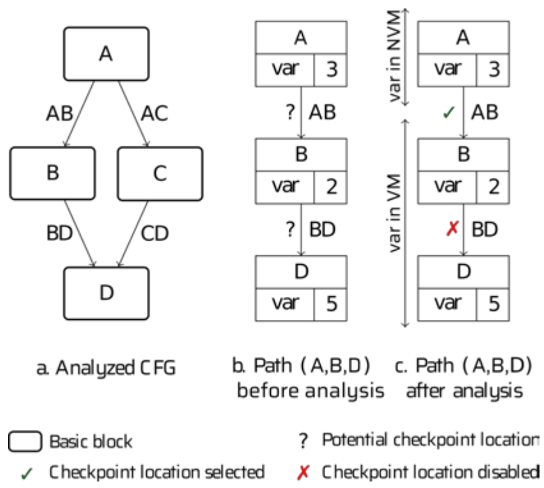
Overview of Progress #2: Compiler-level co-optimization of memory mapping and checkpoint placement

Objectives
- Split functions that take more than one charge to execute
- Minimize overhead of intermittency management

```
int sum = 0;          Save checkpoint here ?
for(int i = 0; i< SIZE; i++)
  sum += array[i];    Save checkpoint here ?
/* ..... */           Save checkpoint here ?
int a = f(sum);
                      Store in VM or NVM ?
Store in VM or NVM ?
```

ELOISE: Joint checkpoint placement and memory allocation
- Energy efficient memory mapping between checkpoints
- Insert checkpoints based on worst-case energy budget
- Target the most frequent paths first
- Account for limited size of volatile memory
- Ensure forward progress across load cycles

Evaluation: ELOISE vs. SotA (early results)
- Very promising on our benchmarks
- On average: 1.68x less energy consumed
- No useless computations
- Small number of checkpoints



Overview of Progress #3: PoC (bird recognition sensor)

System design
- Functional design ✓
- Platform design ✓
- Formal models ✓

Software payload
- Signal acquisition ✓
- Event detection ✓
- Classification (CNN) ✓
- Storage and/or communication of results ✓

Software stack
- LLVM+ELOISE ✓
- NOP fork of Trampoline RTOS ✓

Hardware platform
- TI MSP-EXP430FR5994 Launchpad ✓
- Solar panel for EH ✓
- Microphone, LPWAN, external NVM ✓

✓ Done ✓ In progress ✓ Todo

List of contributors

Jean-Luc Béchennec (LS2N-STR), Olivier Berder (IRISA-Granit), Antoine Bernabeu (LS2N-STR), Sébastien Faucou (LS2N-STR), Mikaël Briday (LS2N-STR), Matthieu Gautier (IRISA-Granit), Robin Gerzaguet (IRISA-Granit), María Méndez Real (IETR-ASIC), Sébastien Pillement (IETR-ASIC), Isabelle Puaut (Inria-PACAP), Hugo Reymond (Inria-PACAP), Erven Rohou (Inria-PACAP)

Université de Rennes 1, Inria, IMT Mines-Télécom, UBO, Université Rennes 2, UBS, Université de Nantes, ENS Rennes, INSA Rennes, CNRS, CentraleSupélec, Inserm, SciencesPo Rennes, ENIB, ENSTA Bretagne, Centrale Nantes