

Solving the System Challenges of Intermittent Energy-harvesting Devices

Brandon Lucia | Assistant Professor, CMU ECE

ABSTRACT Research Group - <https://intermittent.systems>

with PhD Students: Alexei Colin, Kiwan Maeng, Emily Ruppel, Vignesh Balaji
Graham Gobieski, Milijana Surbatovich

MS Students: Preeti Murthy, Amanda Marano, Neil Ryan, Devon White, Chris Meredith

BS Students: Graham Harvey, Mark McElwaine, Hannah Tomio

Industry Collaborators: Zac Manchester (Harvard/KickSat)

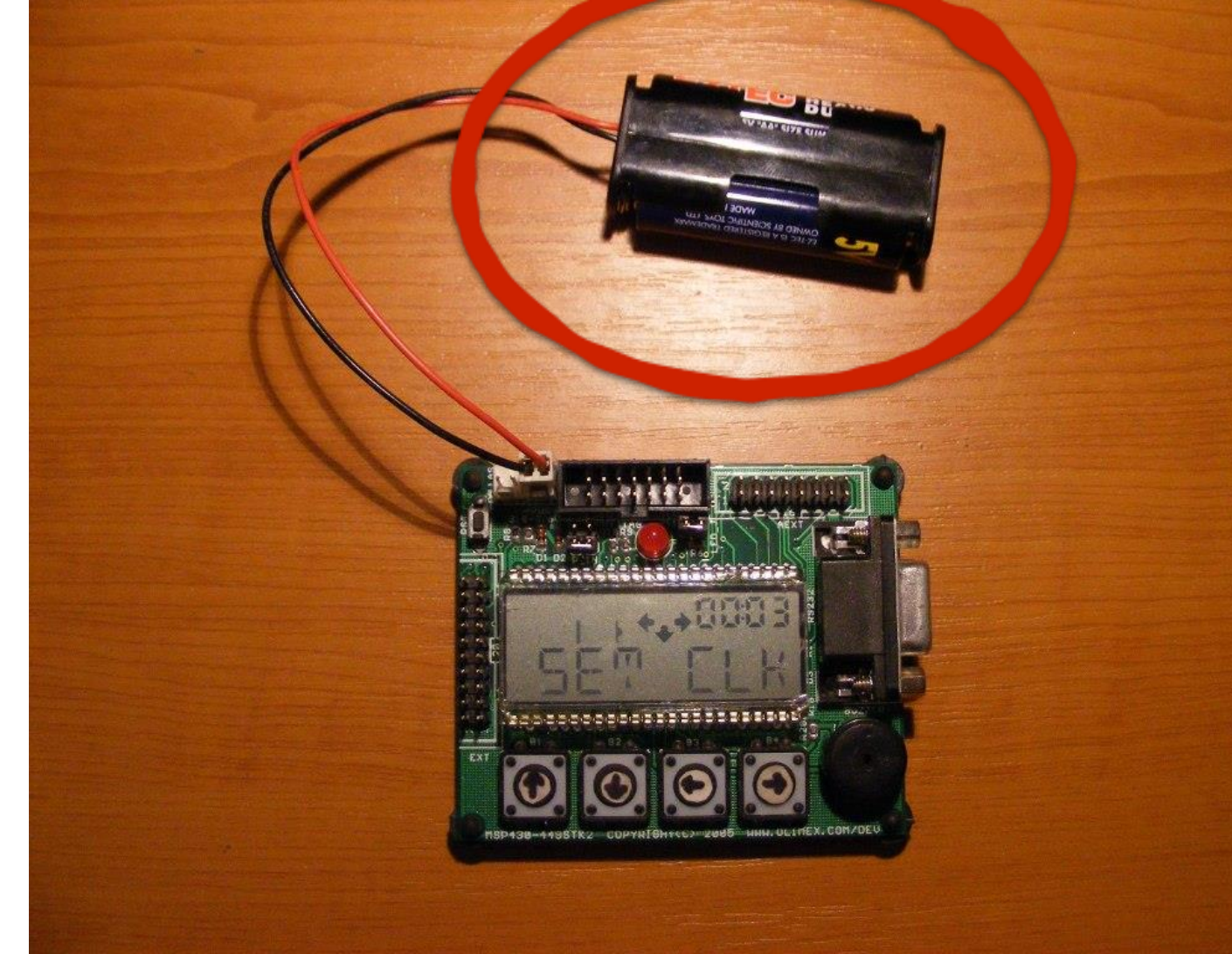
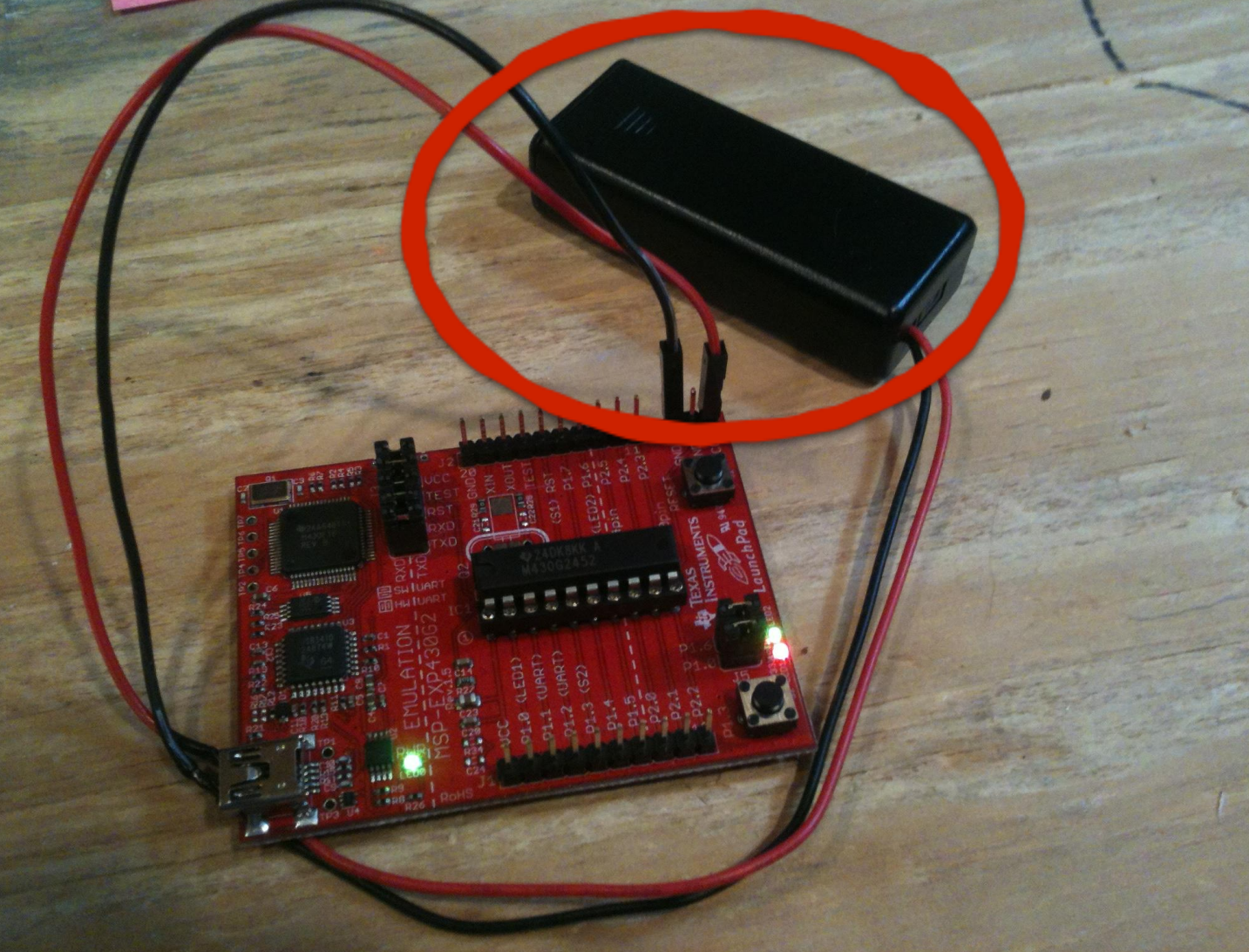
Alanson Sample (Disney Research Pittsburgh)



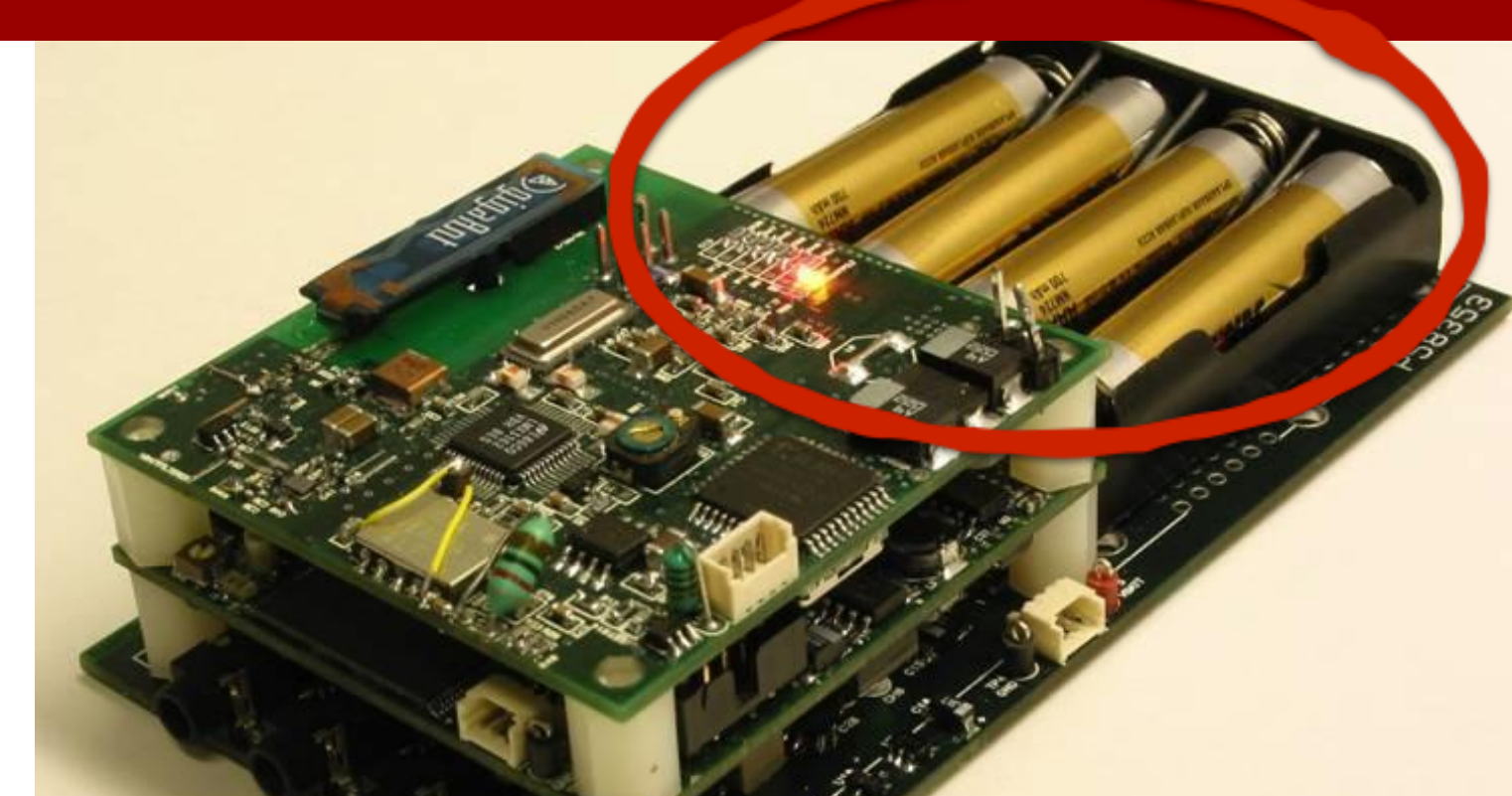
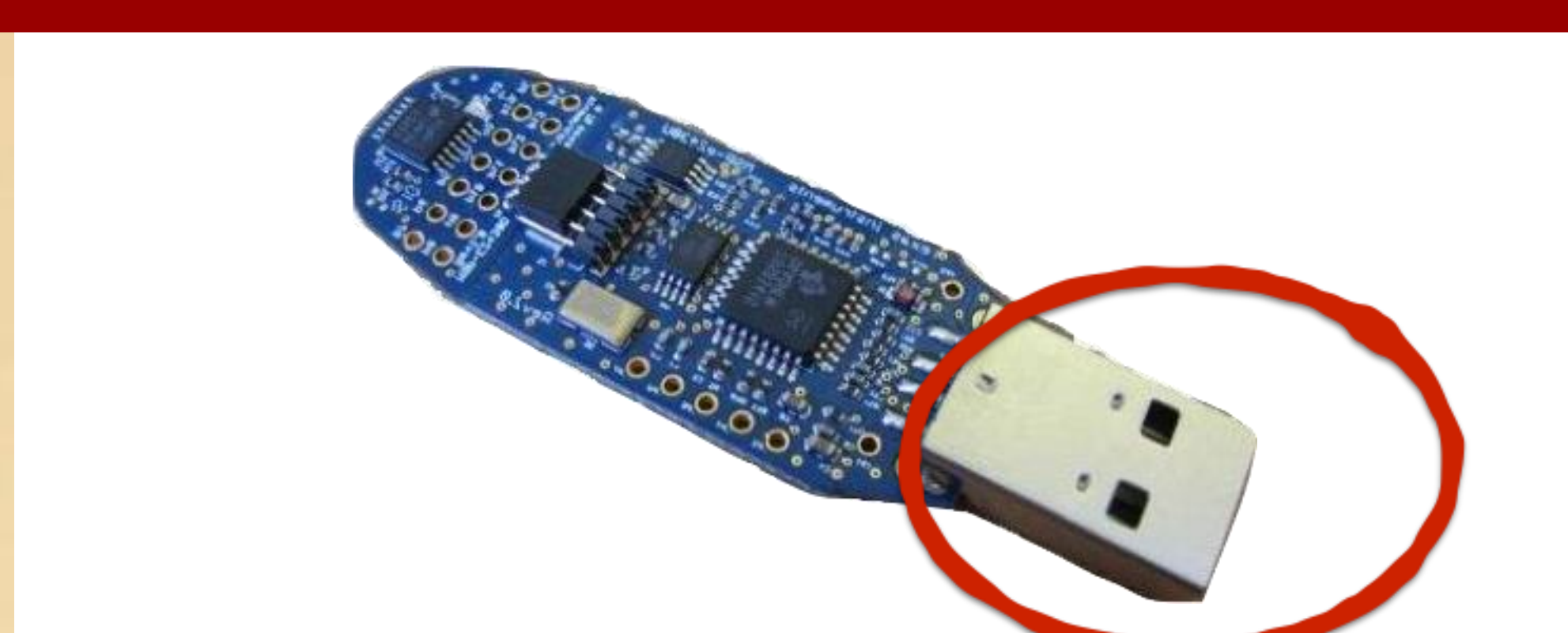
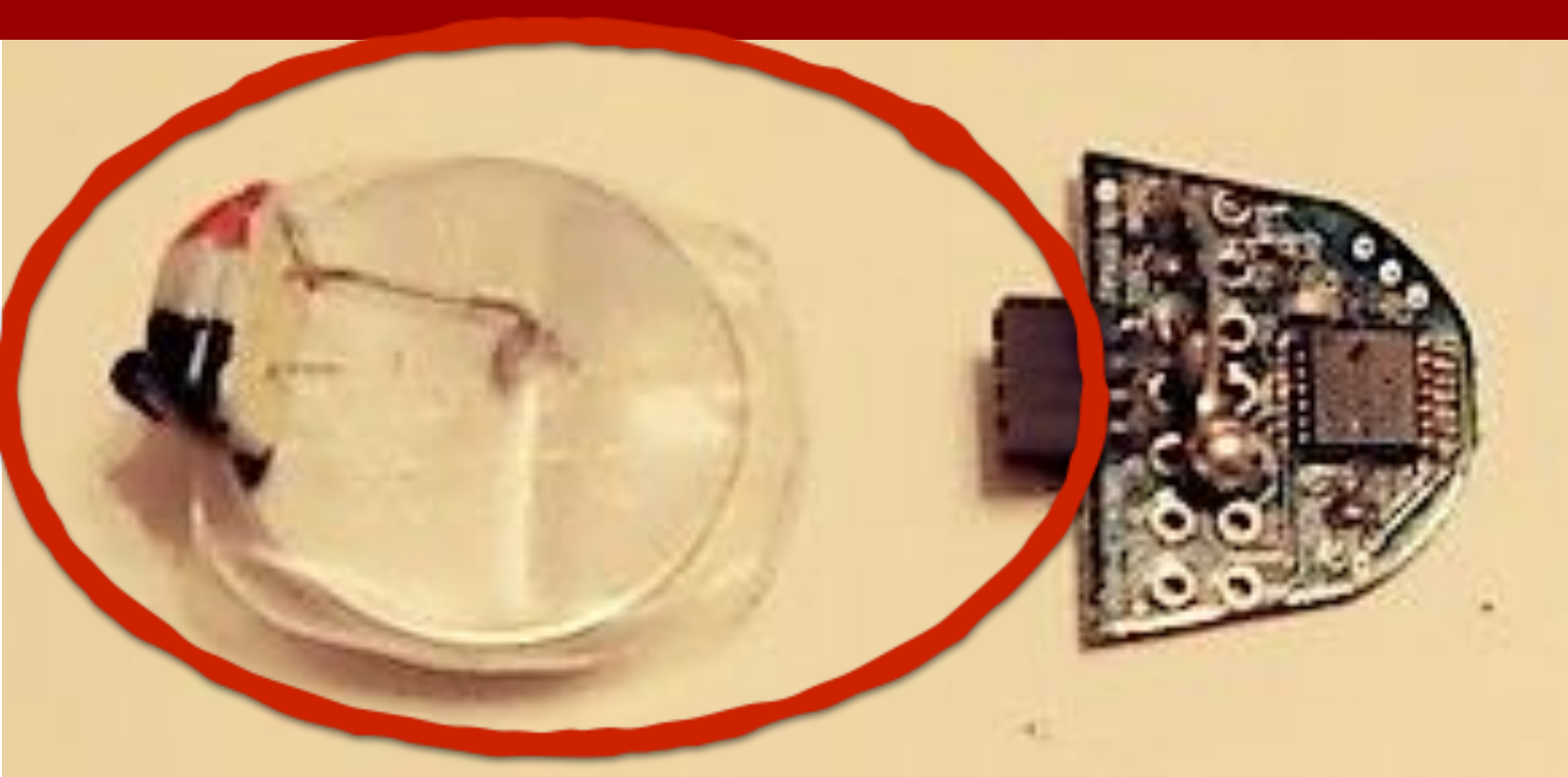


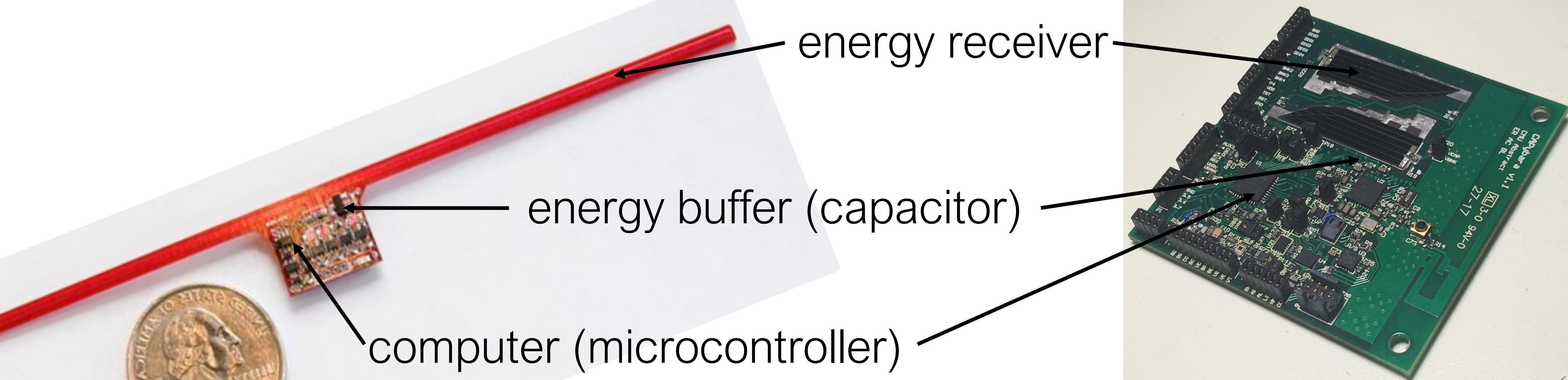
Emerging Devices Everywhere

Wearables, sensors, “Internet of Things”, tiny satellites, medical implants, environmental monitoring, security, computational art, interactive clothing, smart furniture, smart carpets, smart toilet paper

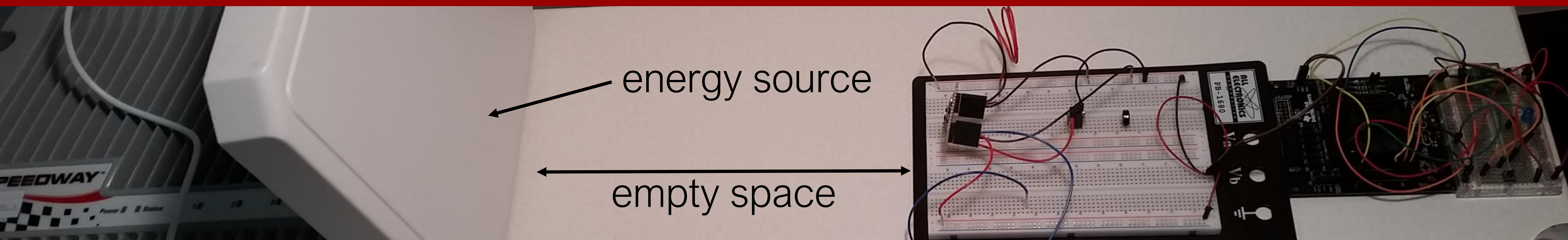


Tethered to a power source

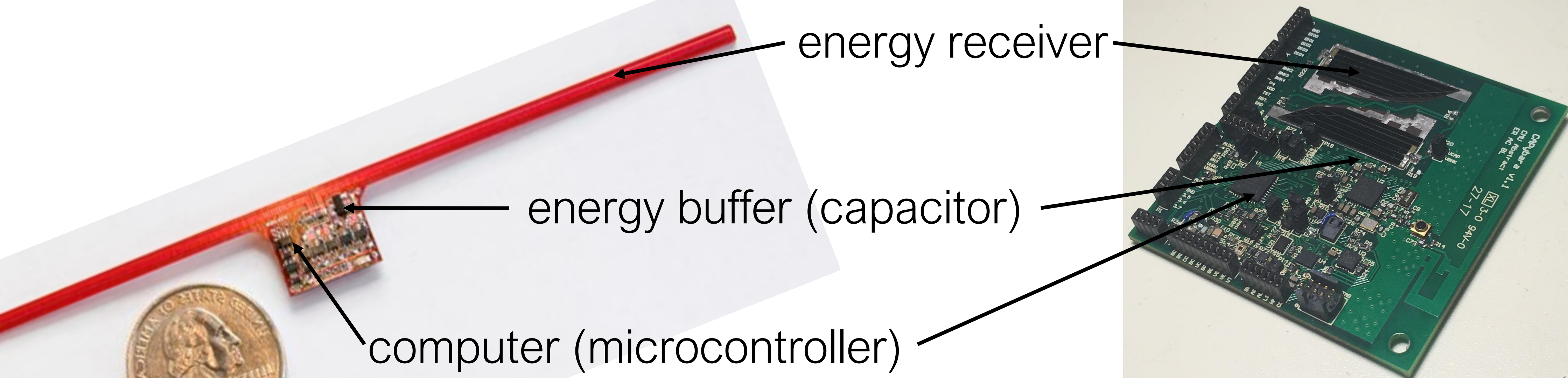




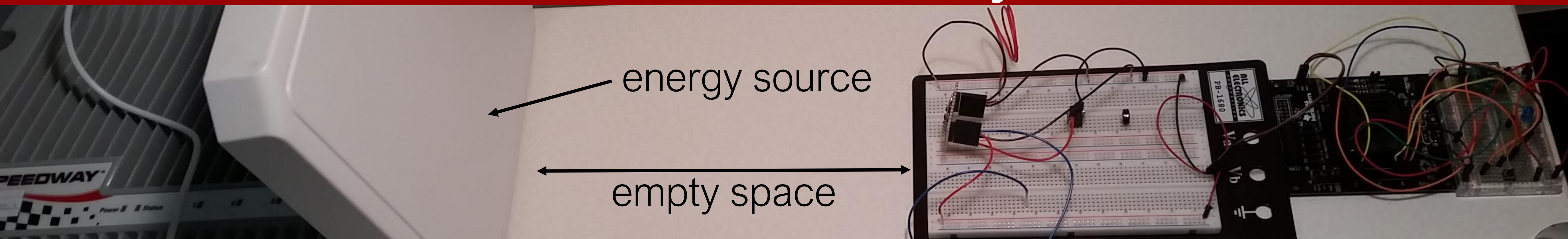
Energy harvesting untethers devices





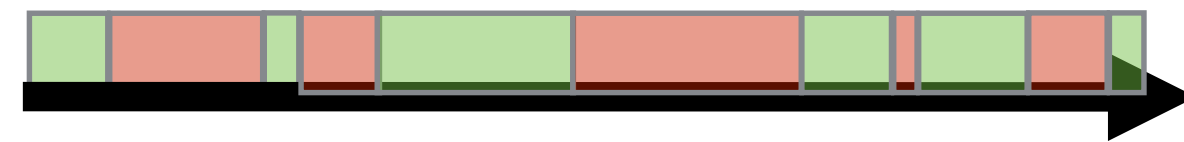


...but energy is intermittent
—and as a result—
software becomes inherently unreliable.



The Intermittent Execution Model

Reasoning About Intermittently-powered Devices

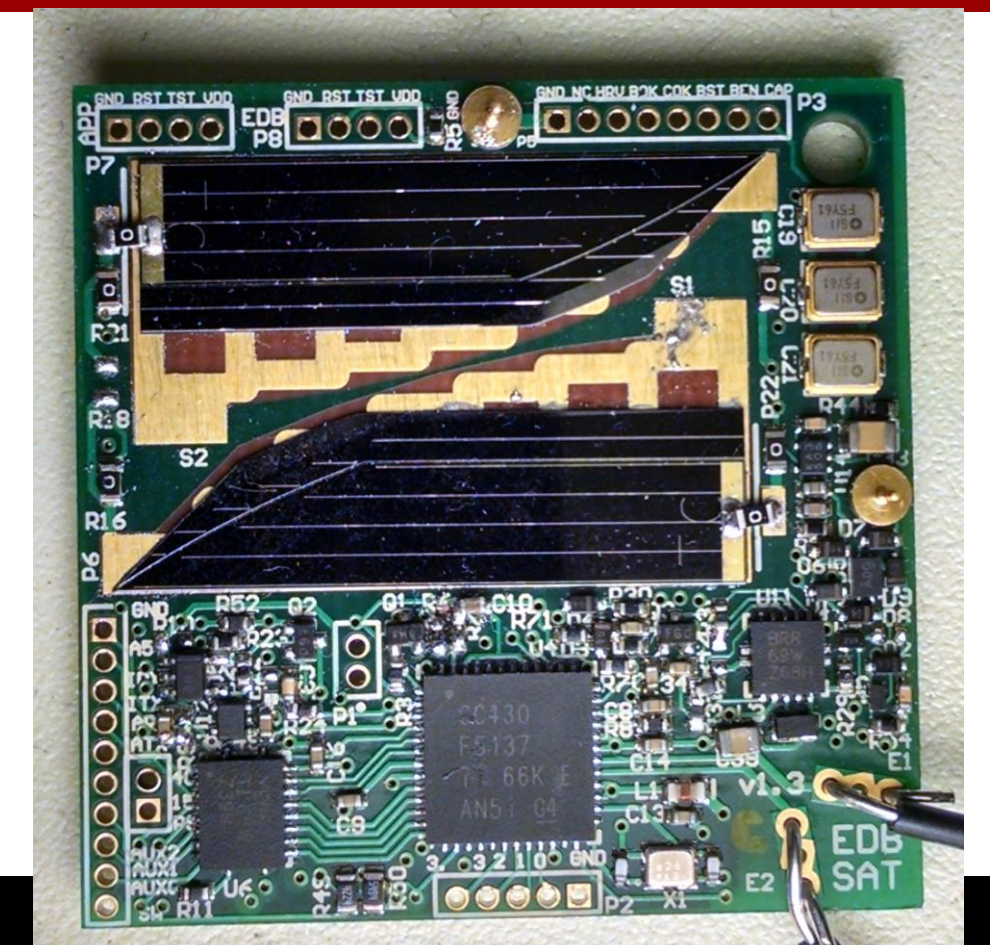


Designing for Intermittence

System Support for Reliable Intermittence

Intermittent Computing Platforms

Hardware, Programming Models, System Software



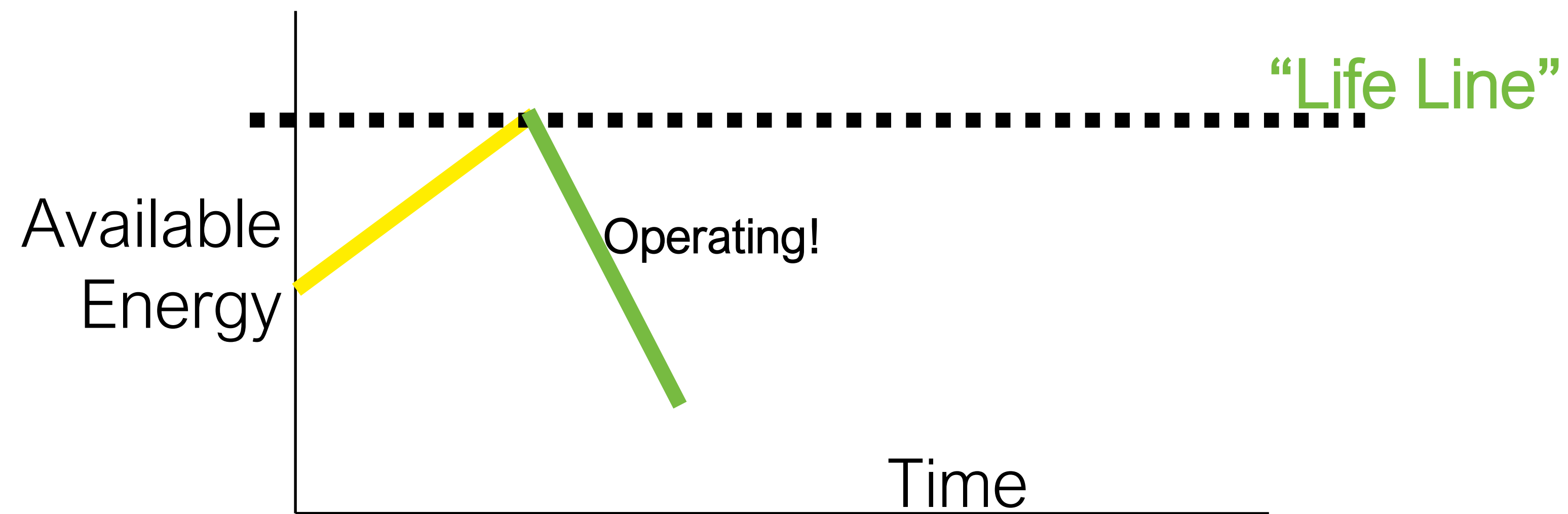
Available
Energy

Charging, dead.

Time

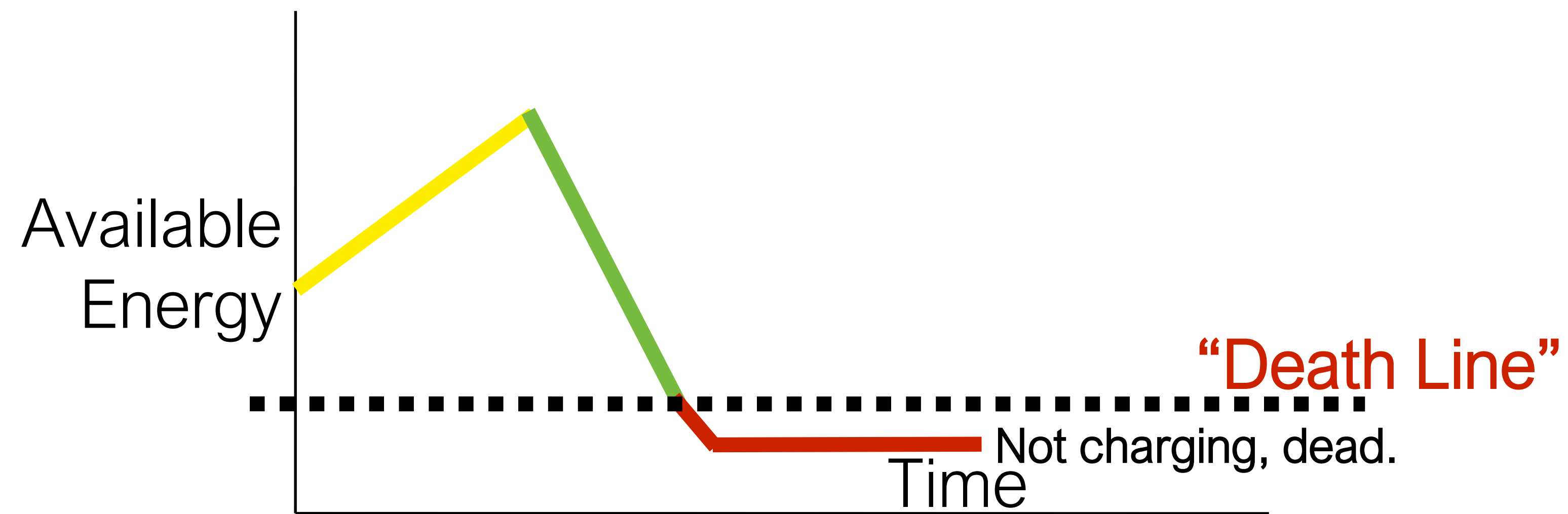
Running on intermittent energy

RF Available!



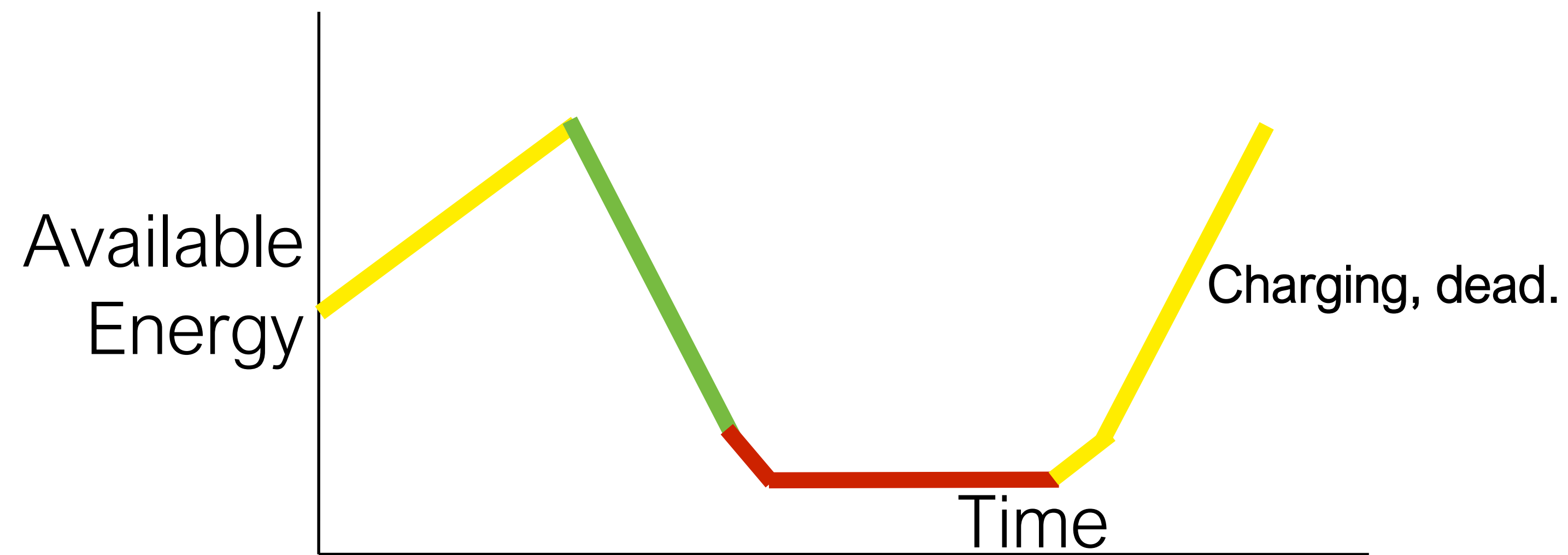
Running on intermittent energy





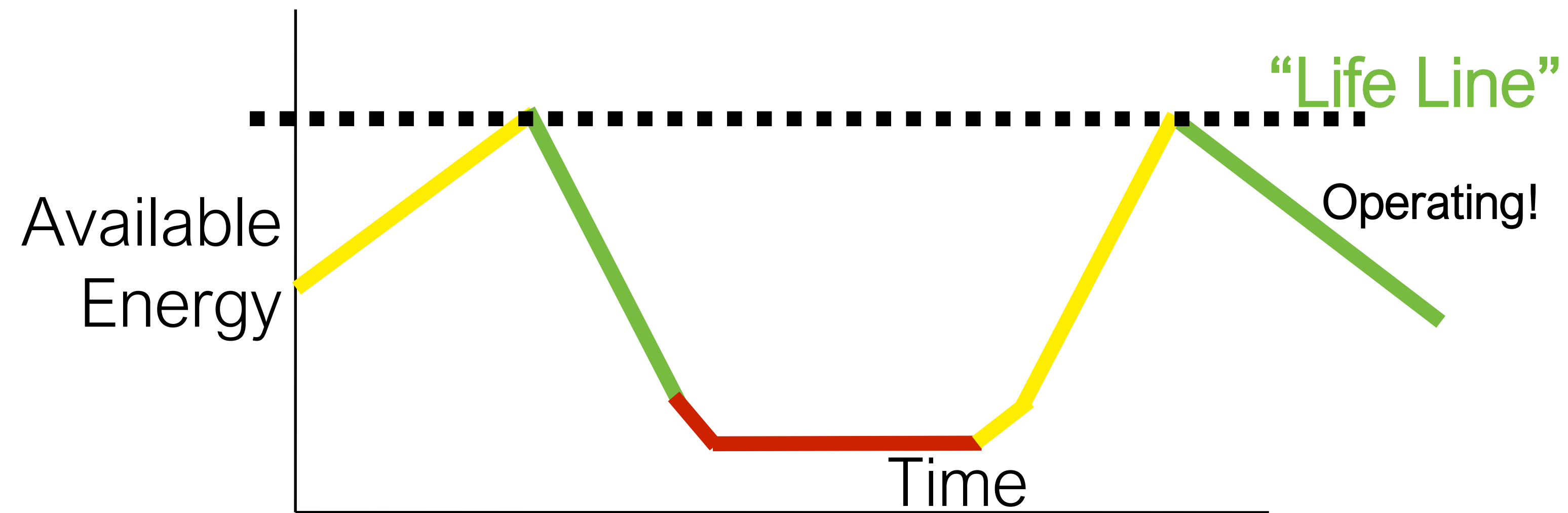
Running on intermittent energy





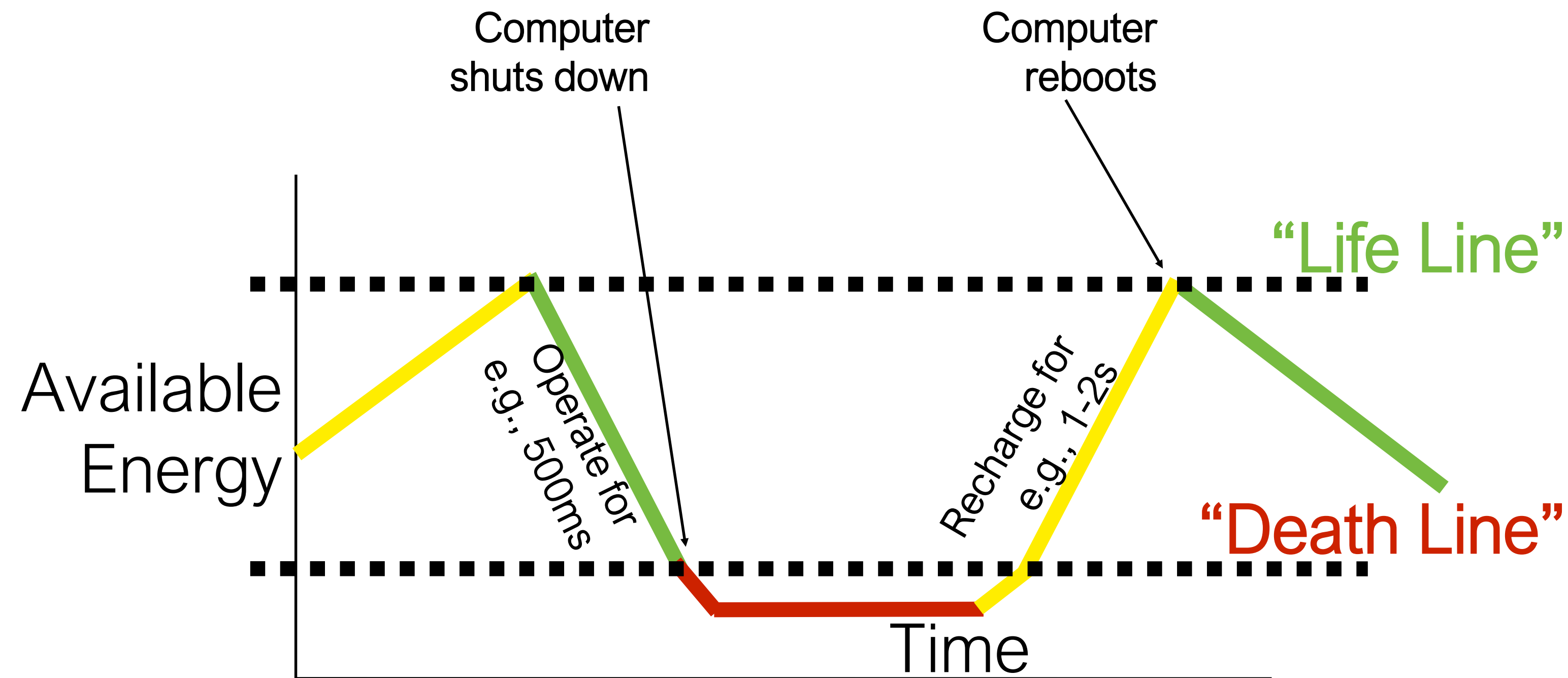
Running on intermittent energy



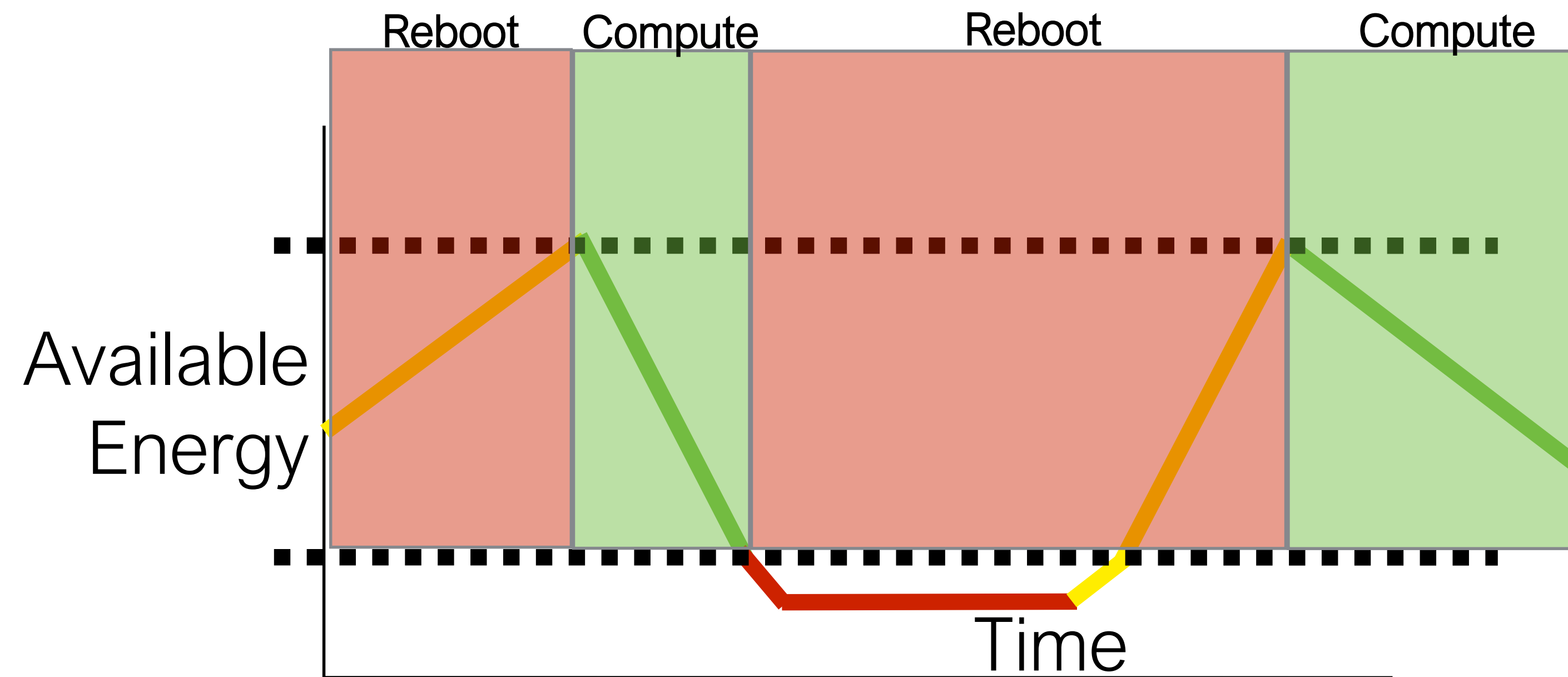


Running on intermittent energy



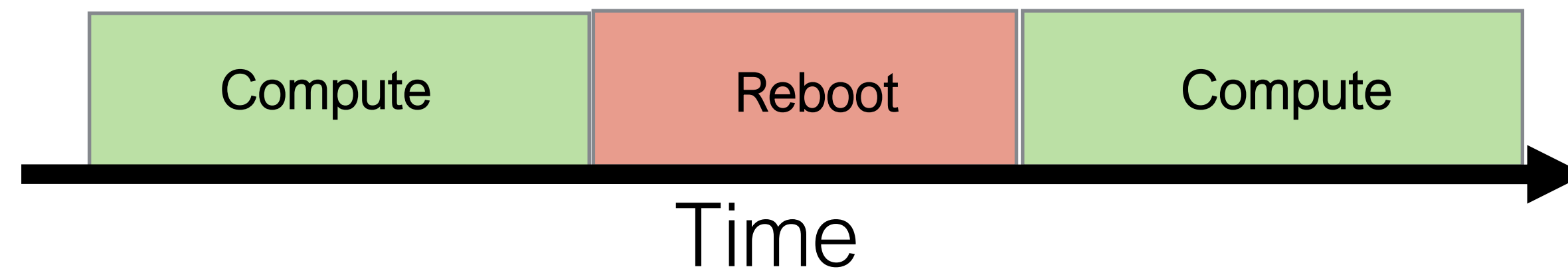


Intermittent Operation on Harvested Energy



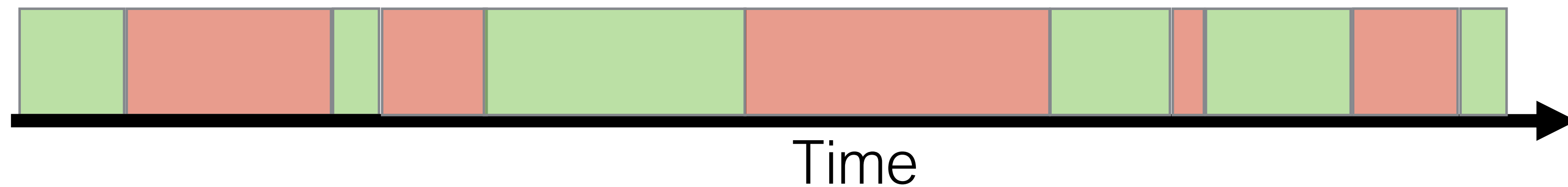
The Intermittent Execution Model

[PLDI '15]



The Intermittent Execution Model

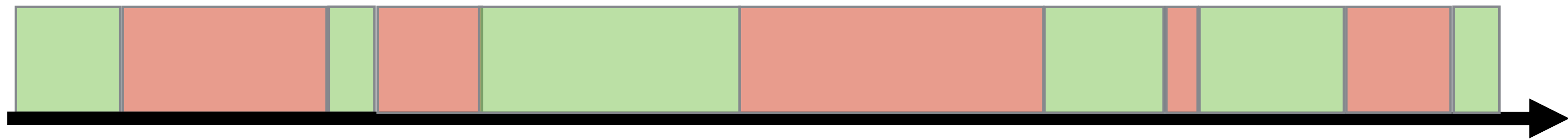
[PLDI '15]



Goal: Run programs that take longer than one green box

```
void main(void){  
    for( i = 1 .. 10)      /*Store 10 letters 'a' in the buffer*/  
        append()  
}
```

```
void append(){  
    sz++                  /*Store a letter 'a' in buf[sz]  
    buf[sz] = 'a'         and increase sz by one*/  
}
```



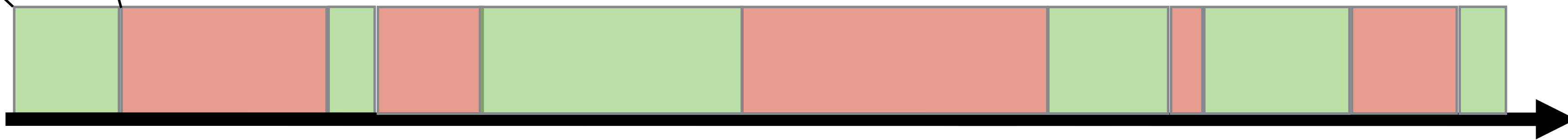
```
for( i = 1 )  
append()  
  SZ++  
  buf[sz] = 'a'
```

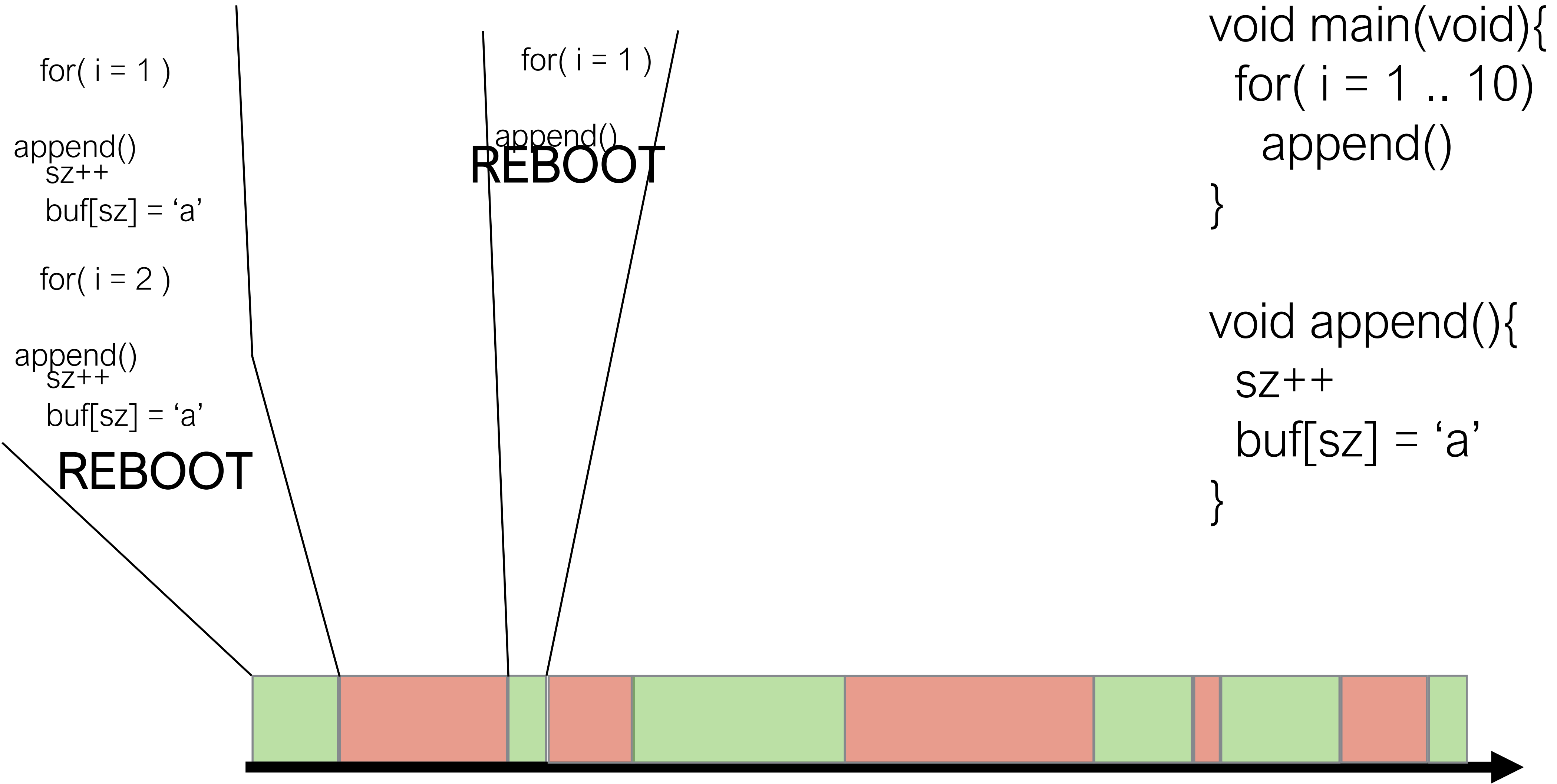
```
for( i = 2 )  
append()  
  SZ++  
  buf[sz] = 'a'
```

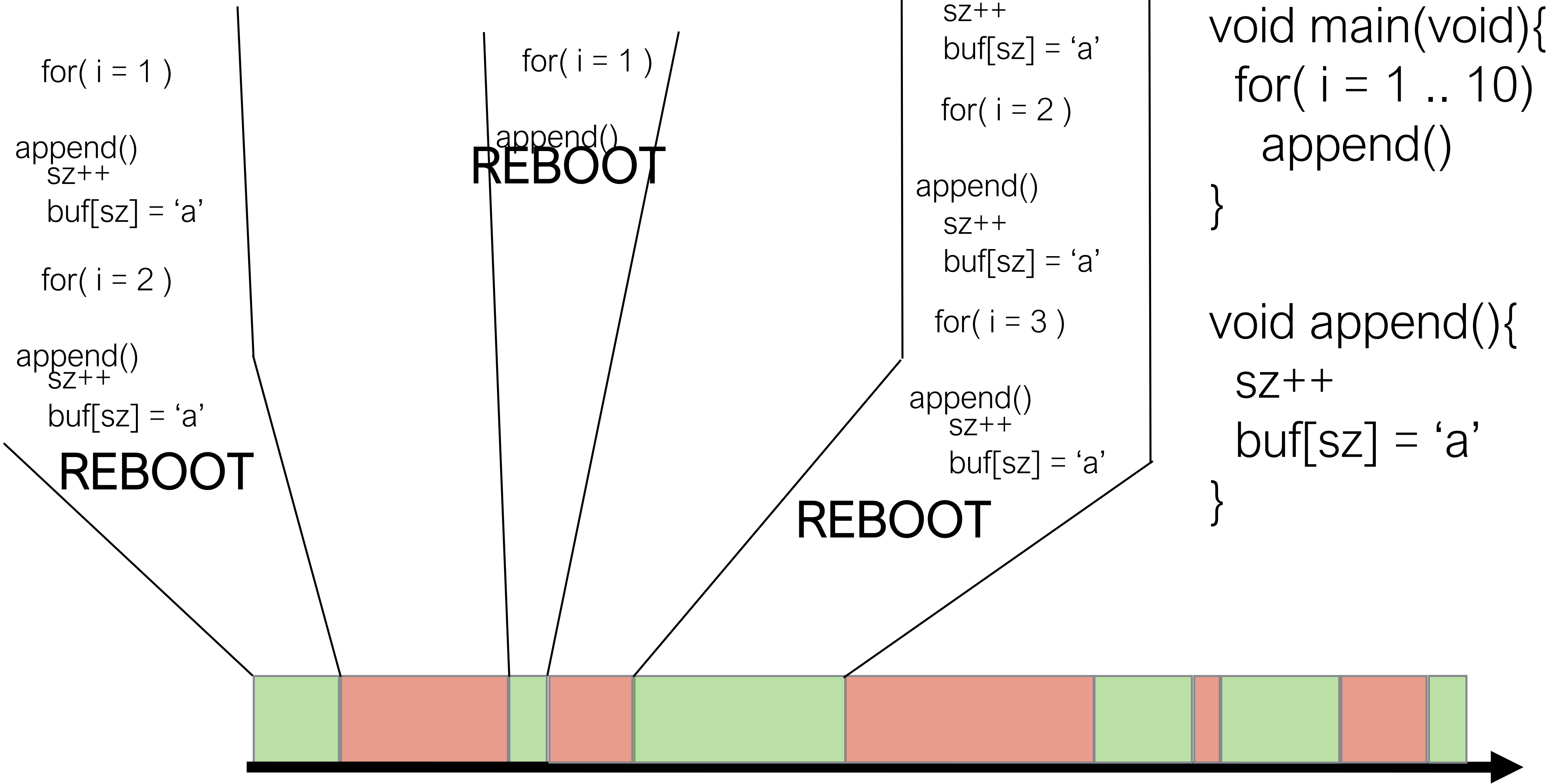
REBOOT

```
void main(void){  
  for( i = 1 .. 10)  
    append()  
}
```

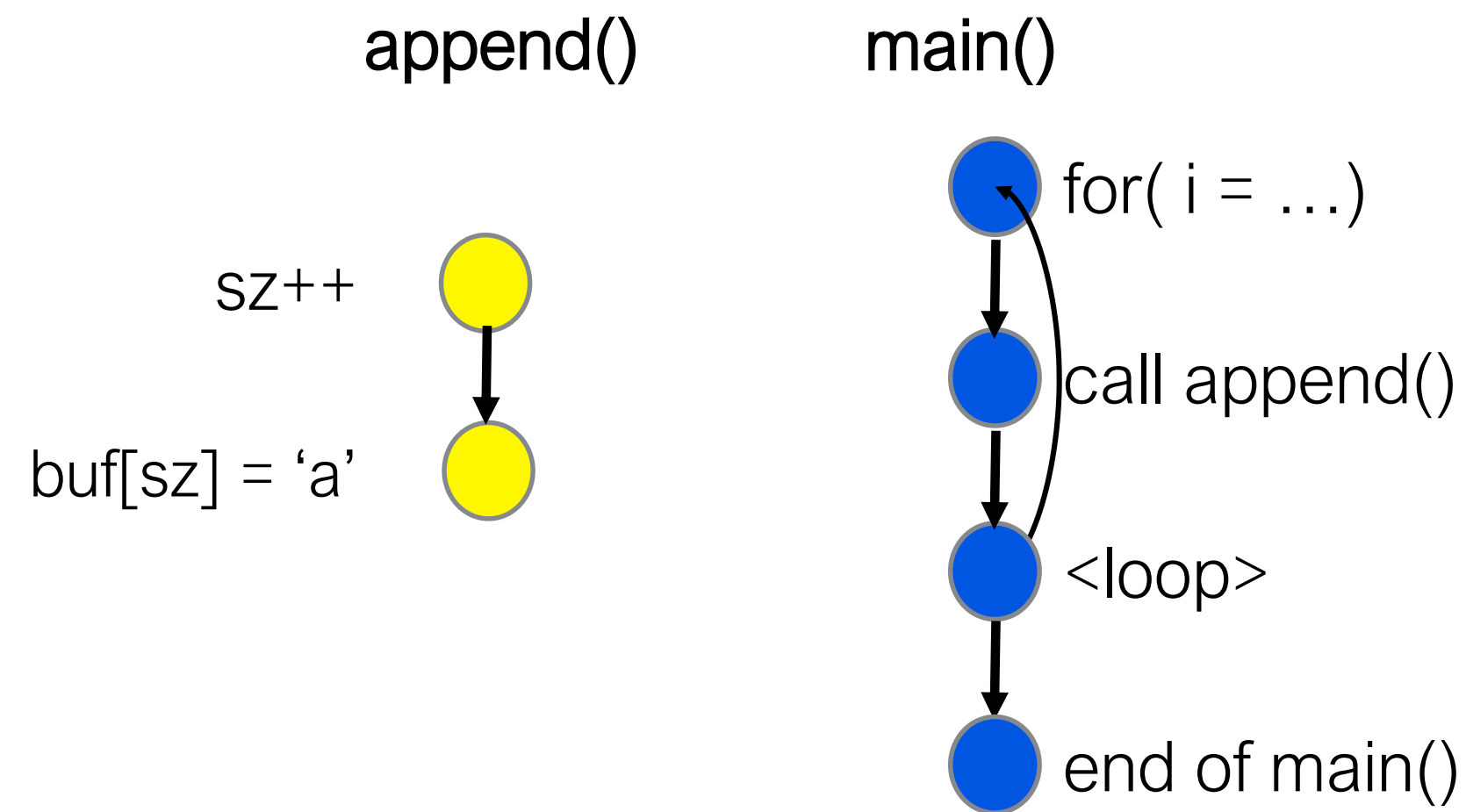
```
void append(){  
  SZ++  
  buf[sz] = 'a'  
}
```





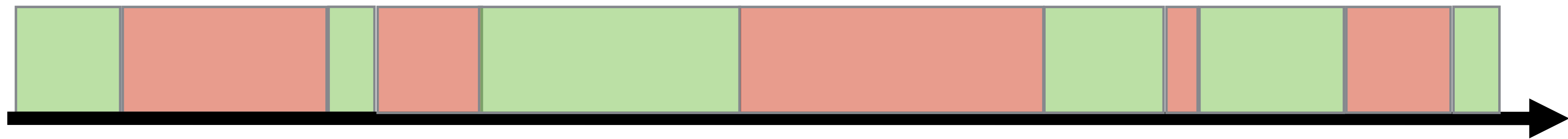


Control-flow Graph



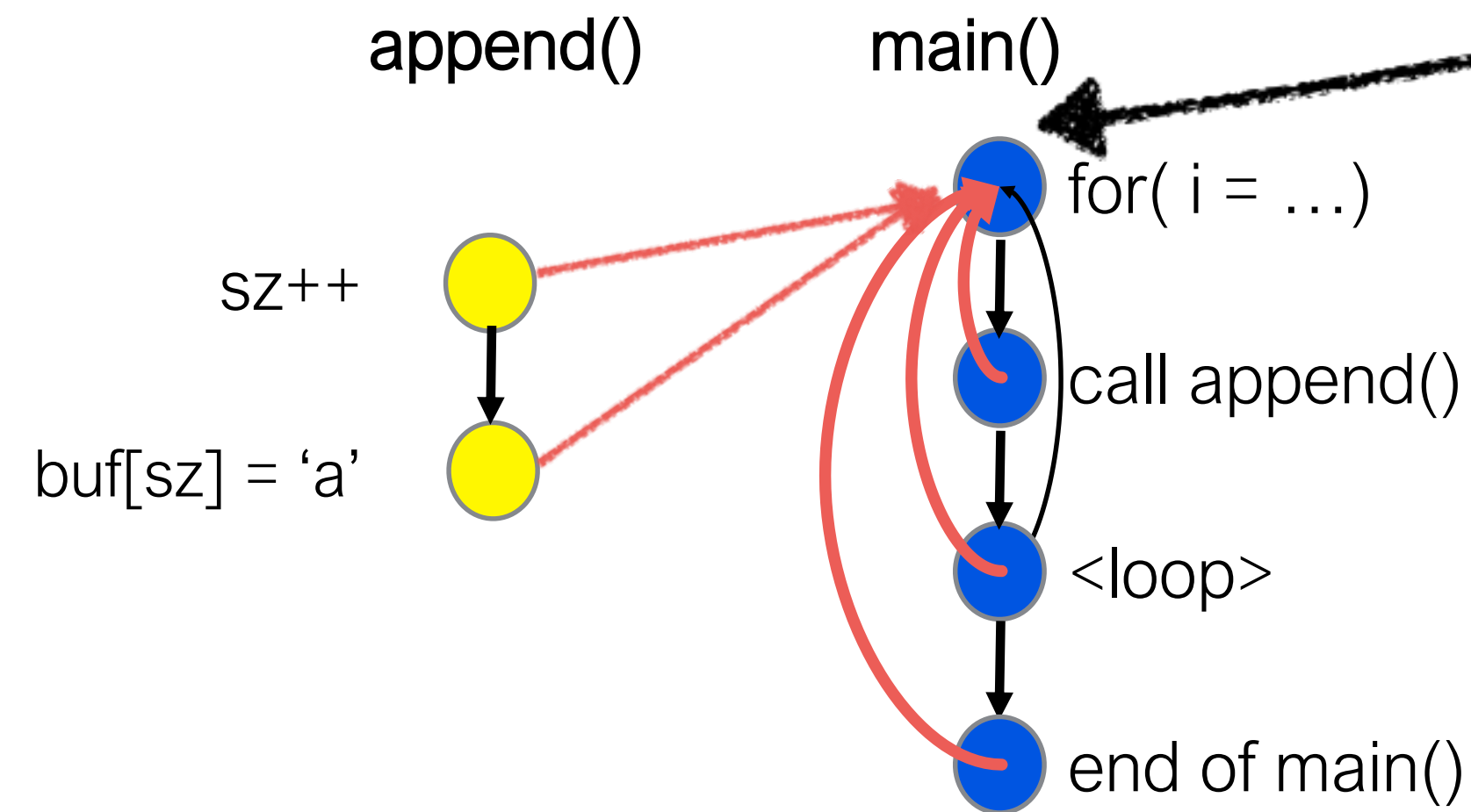
```
void main(void){  
    for( i = 1 .. 10)  
        append()  
}  
  
void append(){  
    SZ++  
    buf[sz] = 'a'  
}
```

We can model intermittence as a control-flow problem



Back in time!

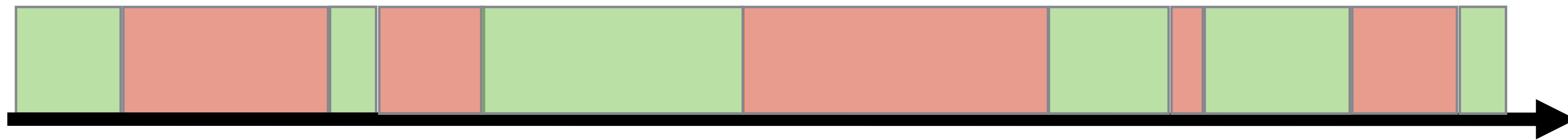
Control-flow Graph




Intermittent Execution Challenge:

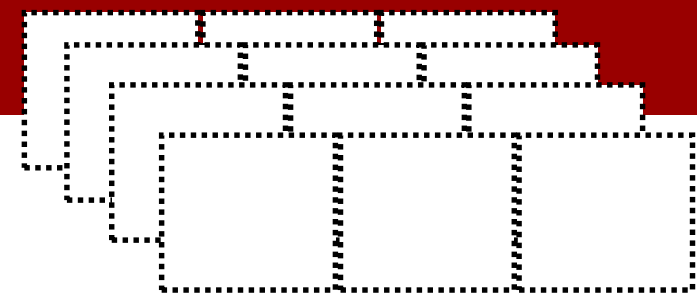
Implicit control-flow
“back in time” on reboot.

Failure Induces
Implicit Control-flow



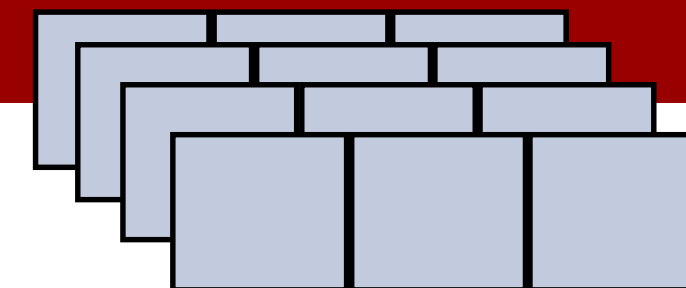


Mixture of Volatile & Non-volatile State



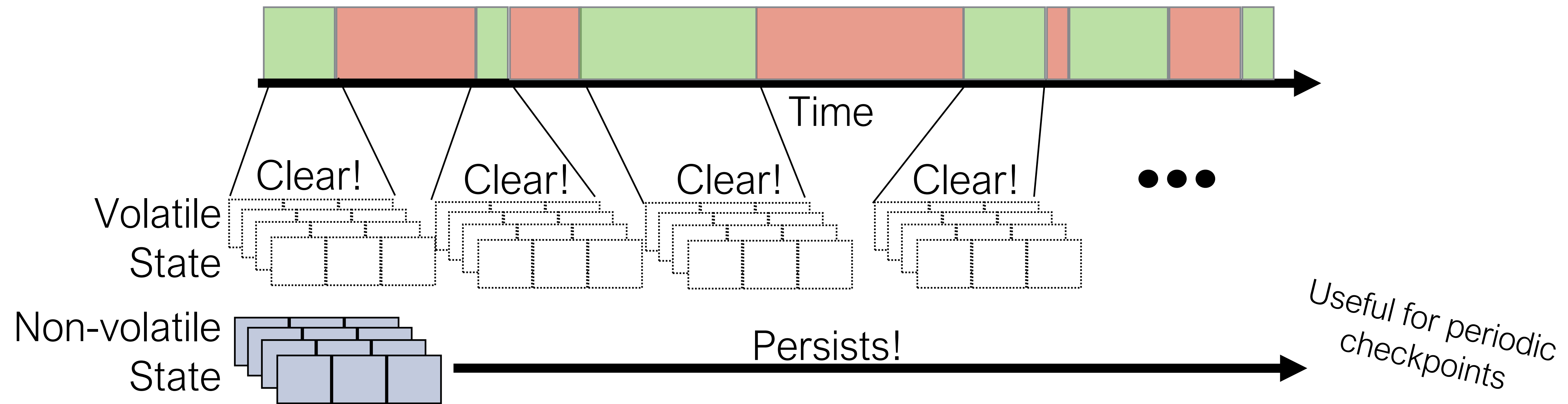
volatile memory (e.g.,
DRAM, SRAM registers)

Access latencies growing
similar w/ new technology

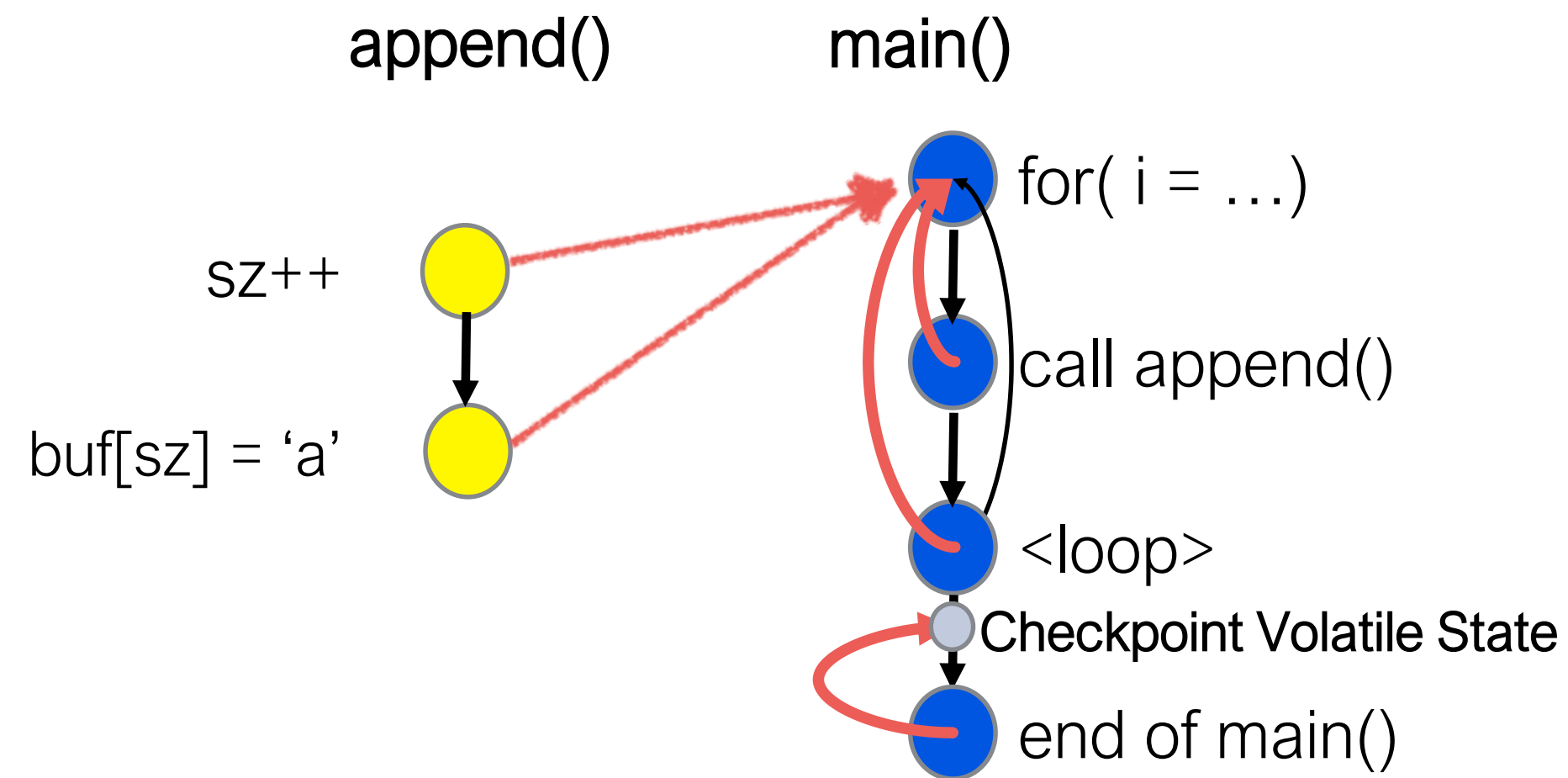


non-volatile memory
(e.g., Flash, FRAM)

Reboots clear volatile state and preserve non-volatile state



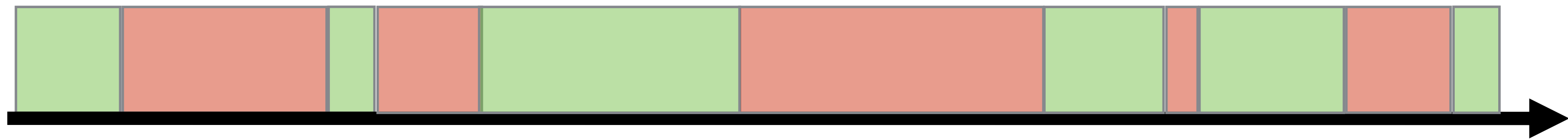
Control-flow Graph



Intermittent Execution Challenge:

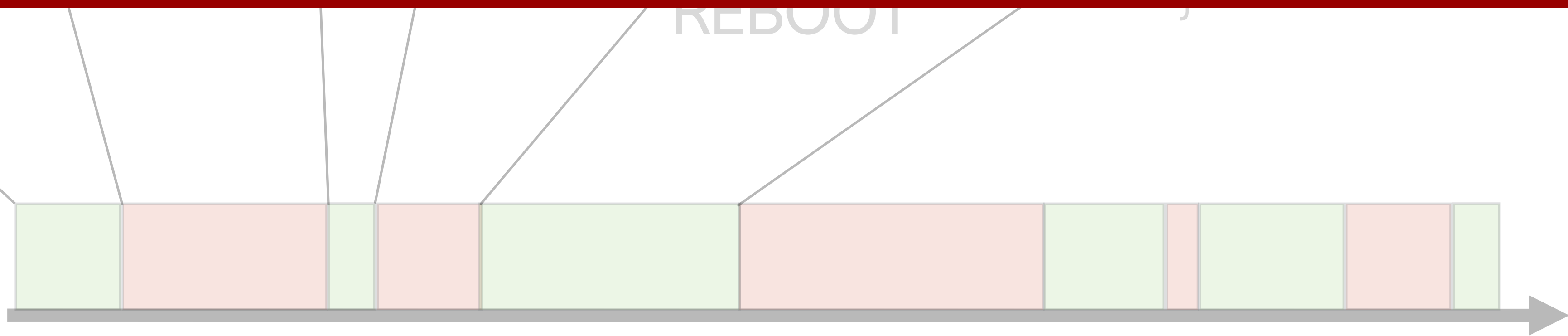
Implicit control-flow
“back in time” on reboot.

Checkpoint introduce **more complex** implicit control-flow



The Big Idea

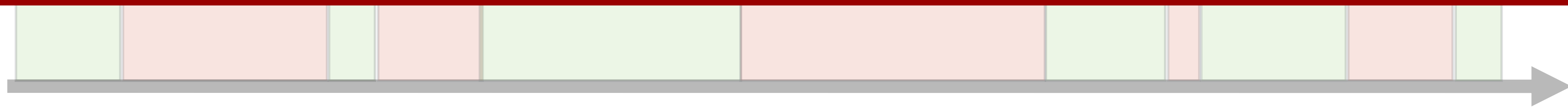
The way we think about **software** does work with the intermittent execution model



Challenge: *Intermittence Bugs*

Software that is **correct** with continuous power can be **incorrect** due to intermittent execution

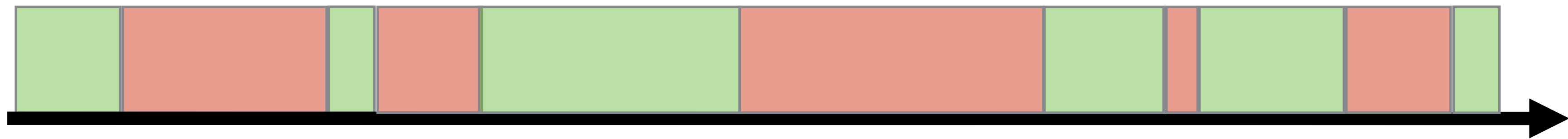
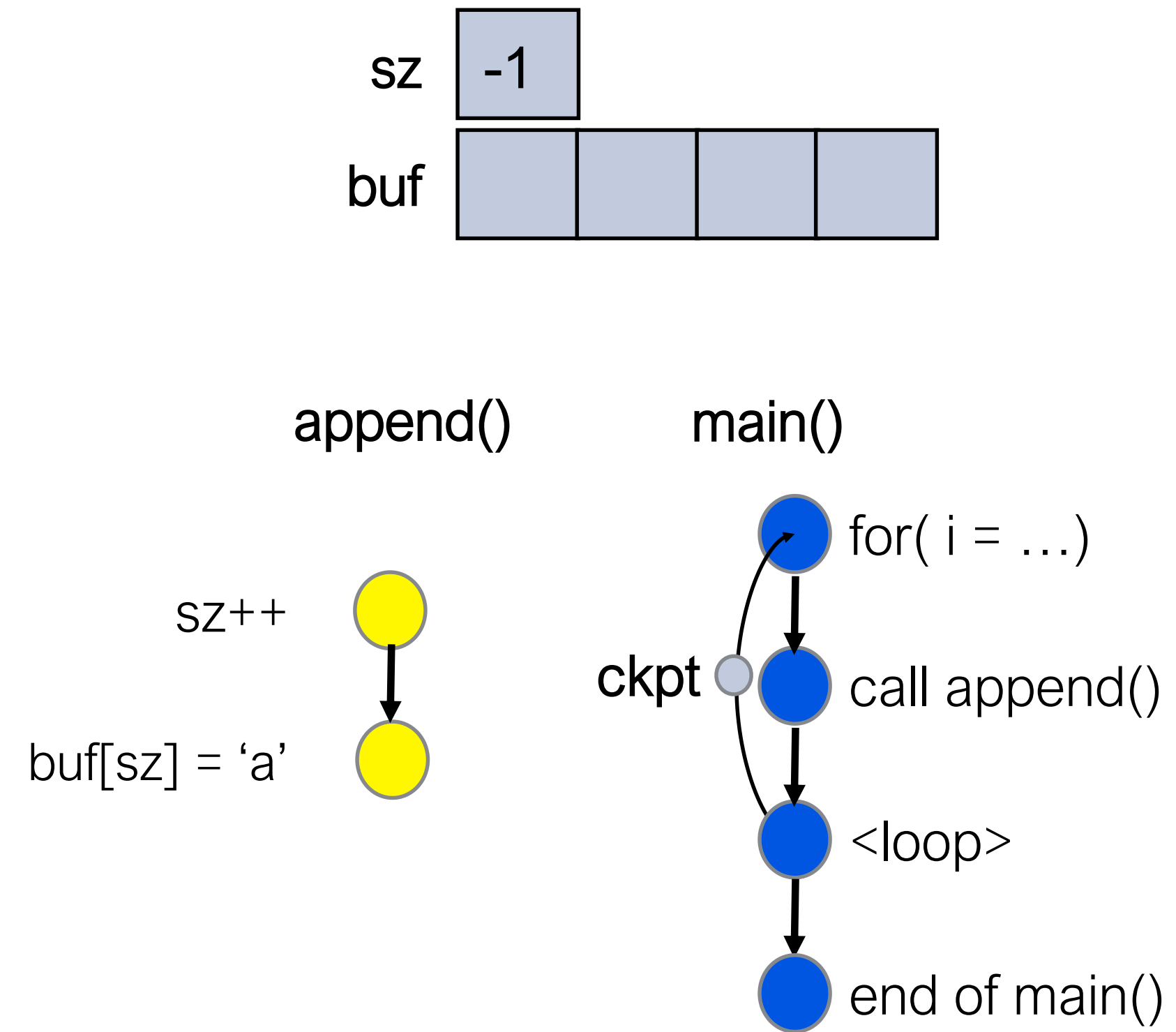
especially when we combine non-volatile and volatile memory!





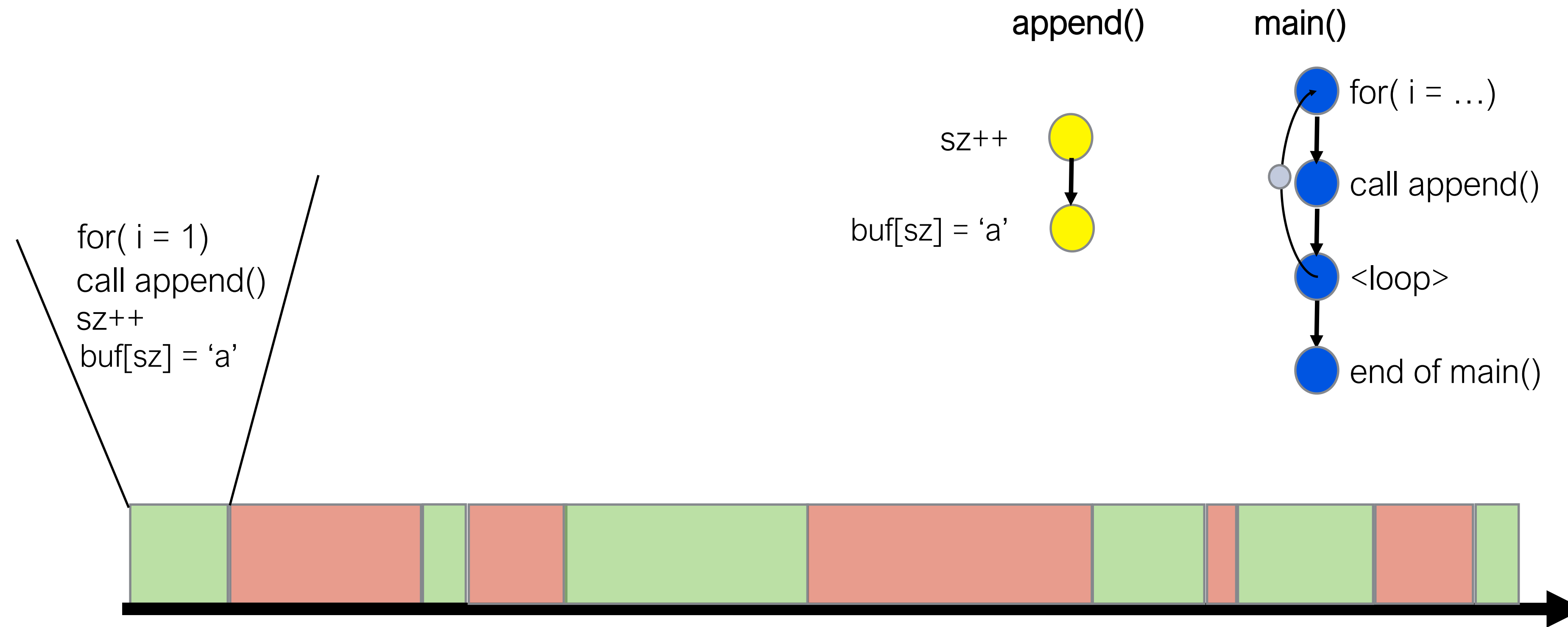
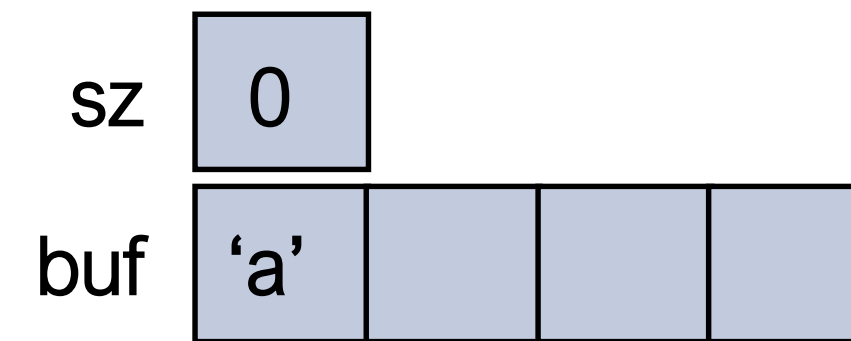
Intermittence Bug: Out-of-thin-air Values

[MSPC '14; PLDI '15]



Intermittence Bug: Out-of-thin-air Values

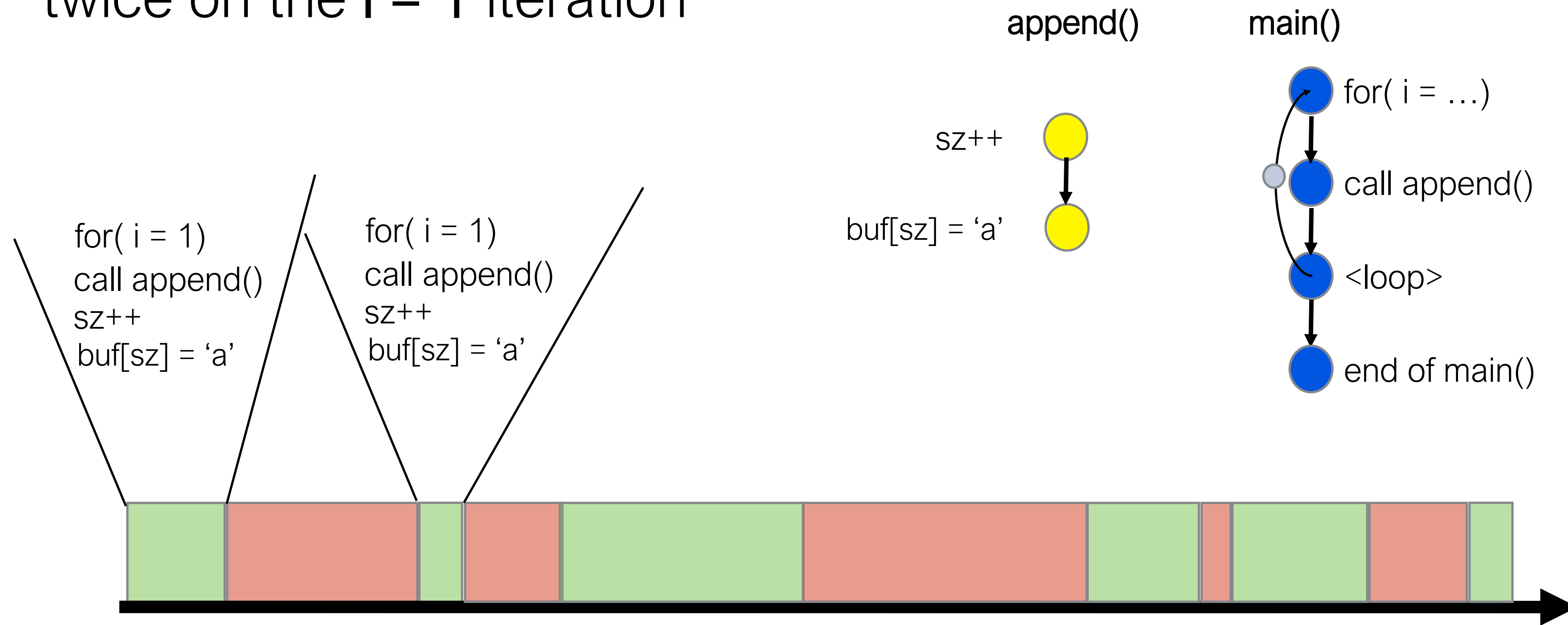
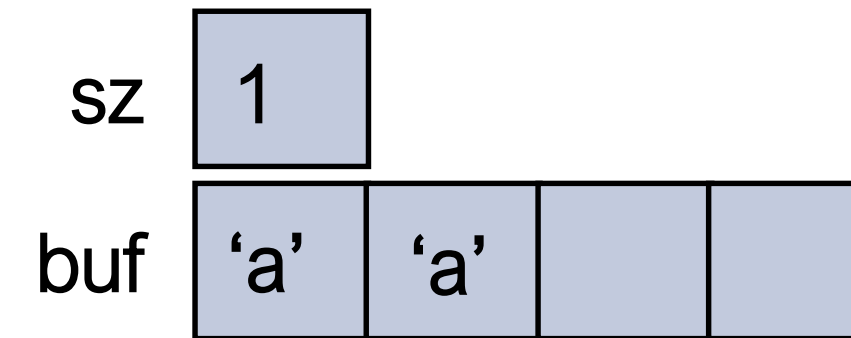
[MSPC '14; PLDI '15]



Intermittence Bug: Out-of-thin-air Values

[MSPC '14; PLDI '15]

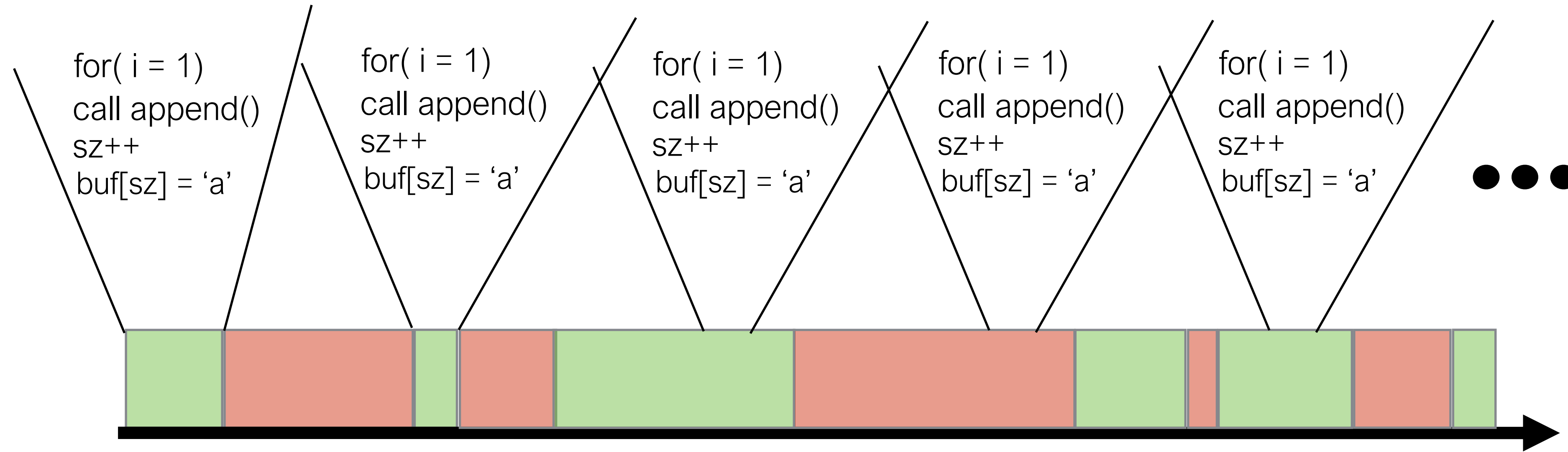
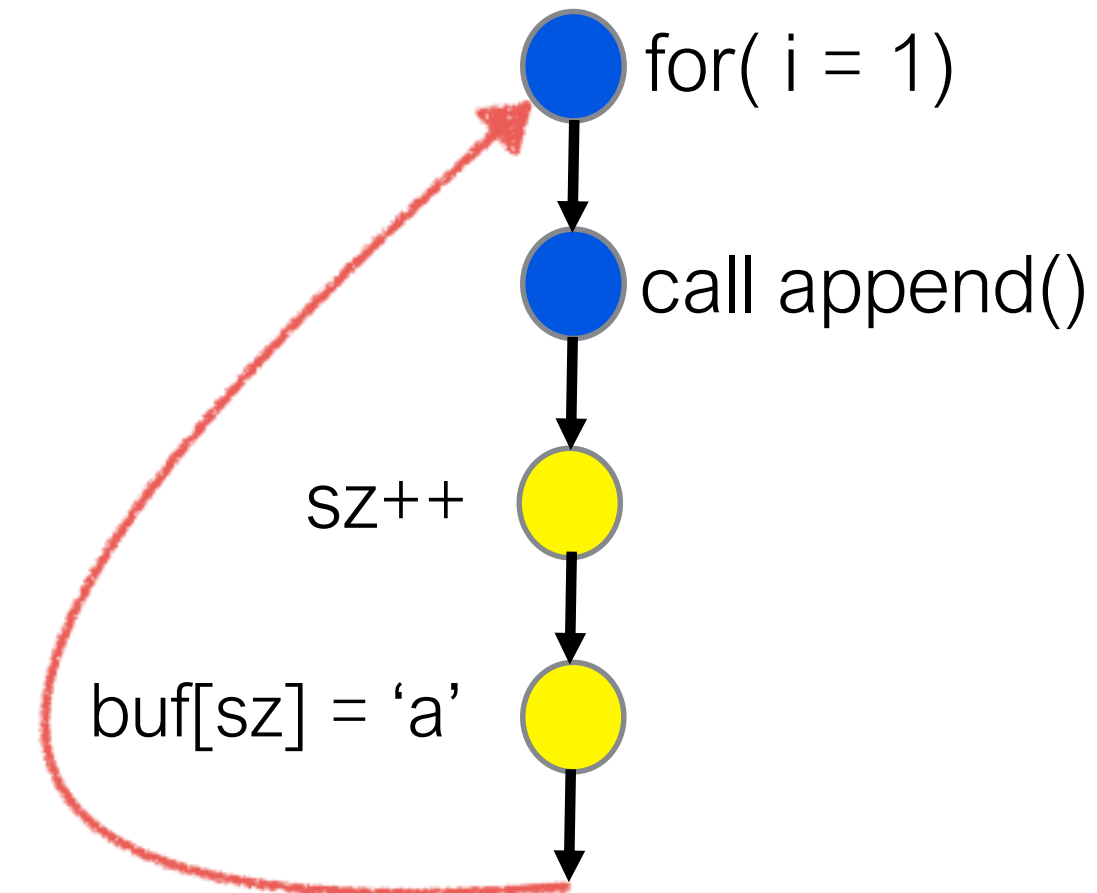
Error: 'a' is appended to buf twice on the i = 1 iteration



Intermittence Bug: Out-of-thin-air Values

[MSPC '14; PLDI '15]

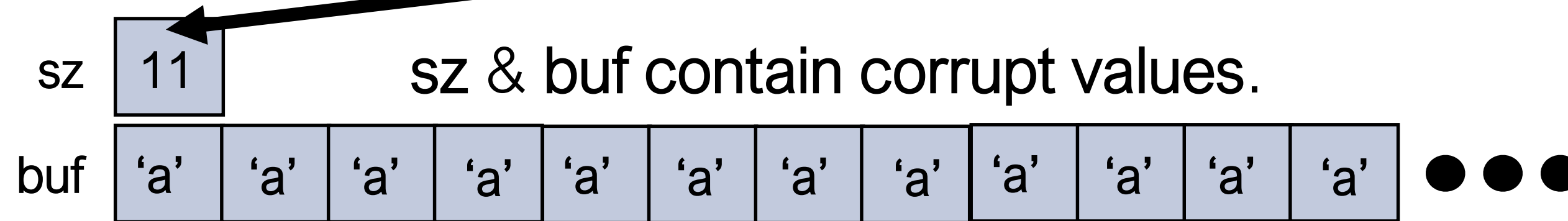
Stuck in an implicit
loop for $i = 1$



Intermittence Bug: Out-of-thin-air Values

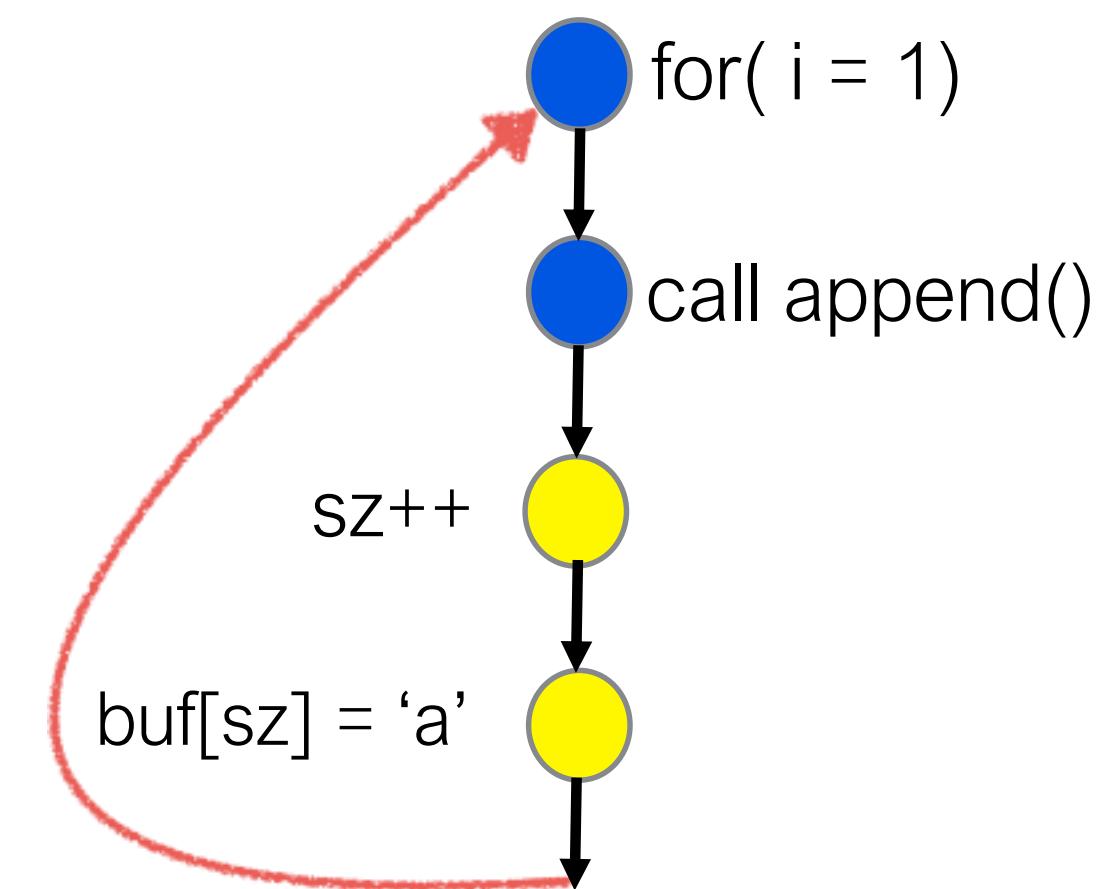
[MSPC '14; PLDI '15]

“Out of thin air” value not permitted by any continuous execution!



Memory contents depend on the *availability of harvestable energy and capacitor size!*

Stuck in an implicit loop for `i = 1`

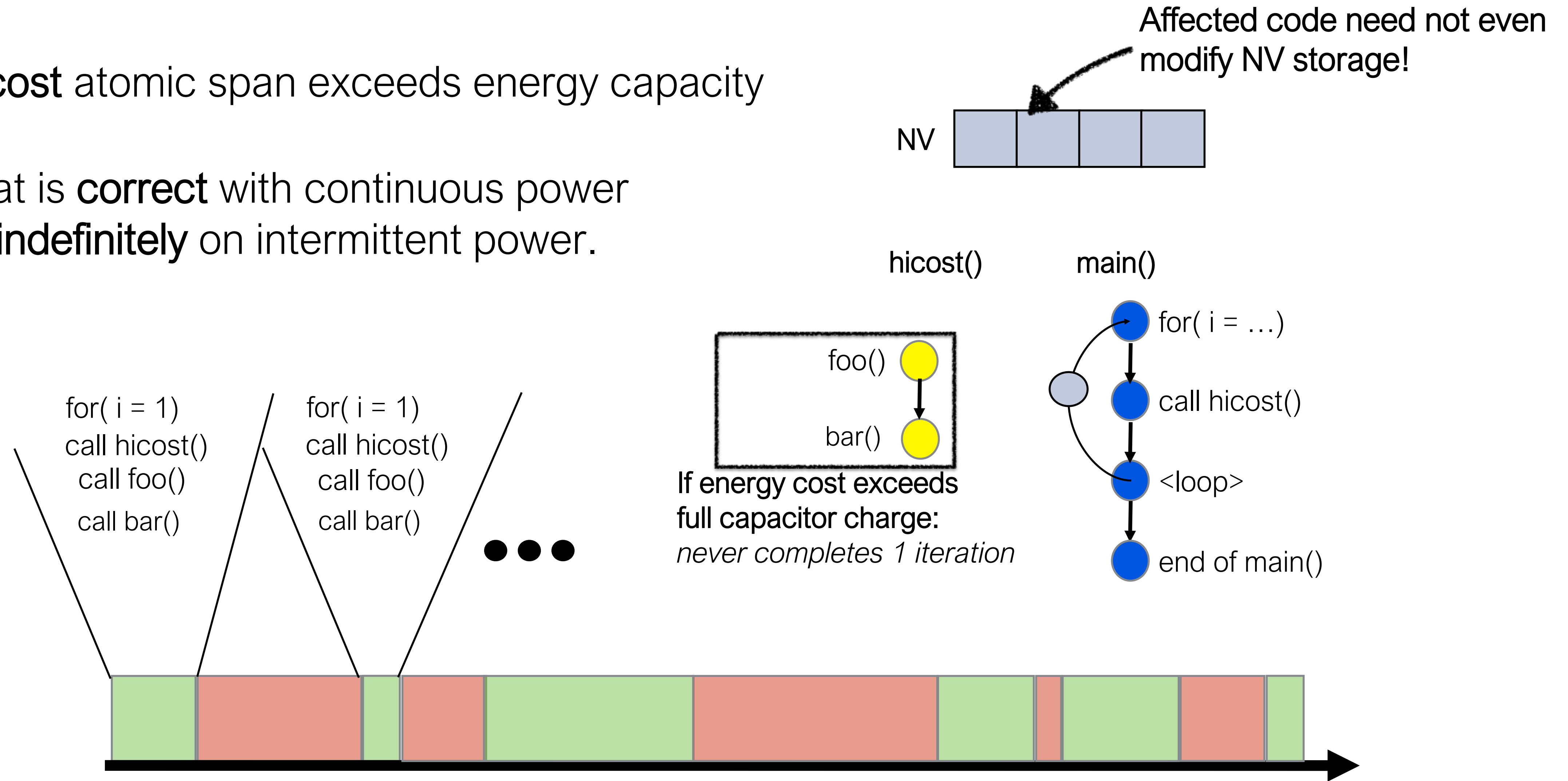


Intermittence Bug: Livelock & Non-termination

[Colin, et al. CASES '15, ASPLOS '16]

Energy cost atomic span exceeds energy capacity

Code that is **correct** with continuous power **reboots indefinitely** on intermittent power.

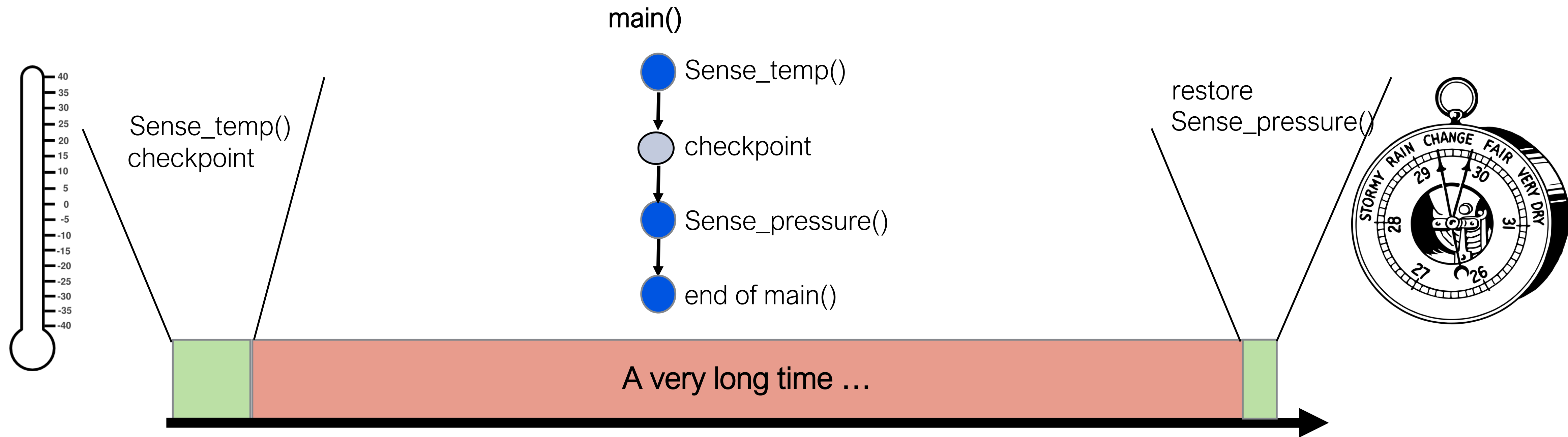


Intermittence Bug: I/O Atomicity

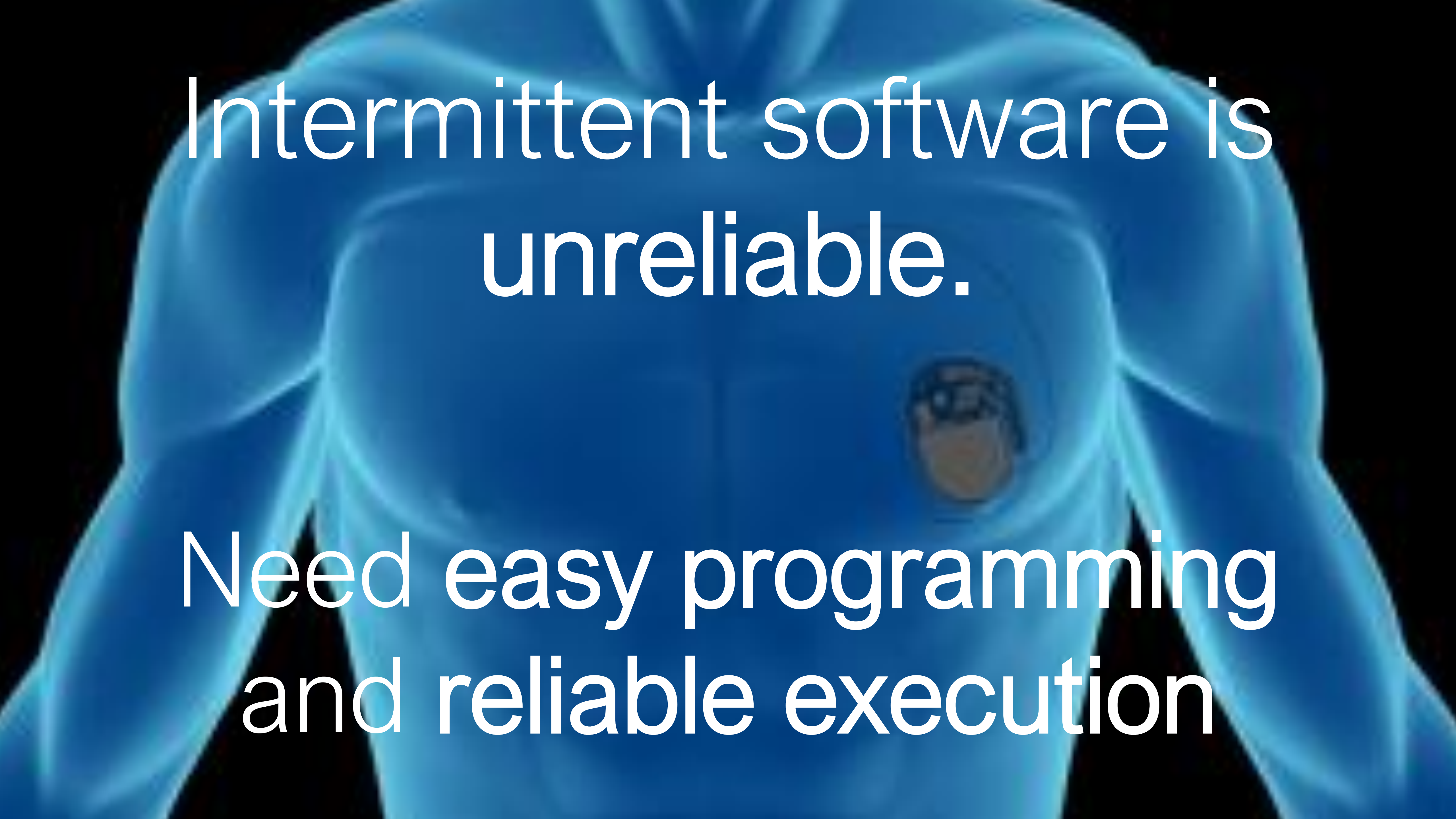
[Colin OOPSLA '16, Maeng OOPSLA '17, Hester et al SenSys '17]

Checkpoint between I/O operations that should execute together leaves I/O inconsistent

No atomicity control w/ automatic checkpoints

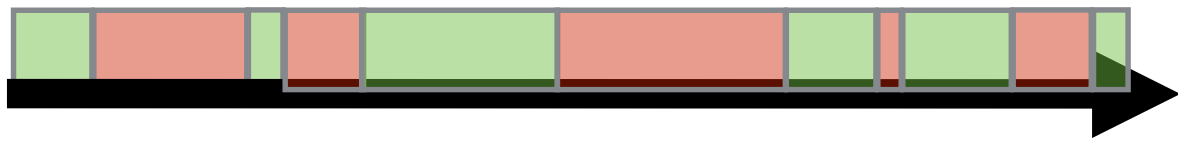


Inconsistent data read at different moments in time



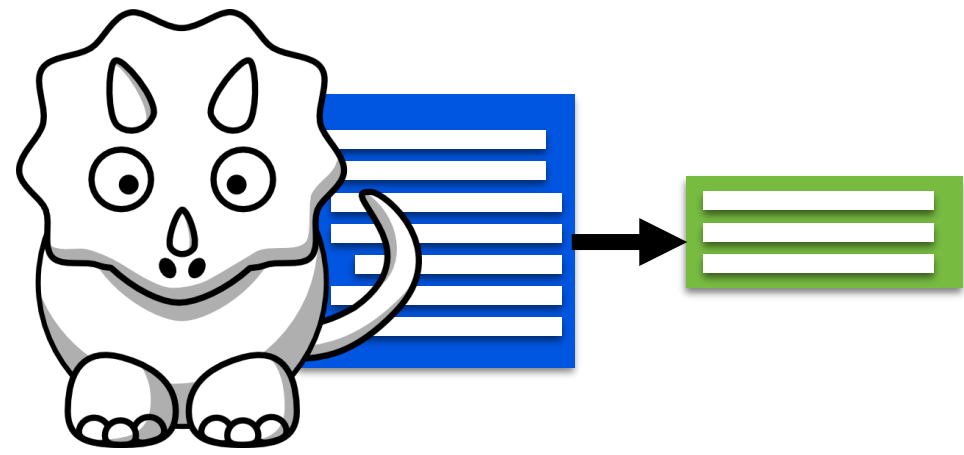
Intermittent software is
unreliable.

Need easy programming
and reliable execution



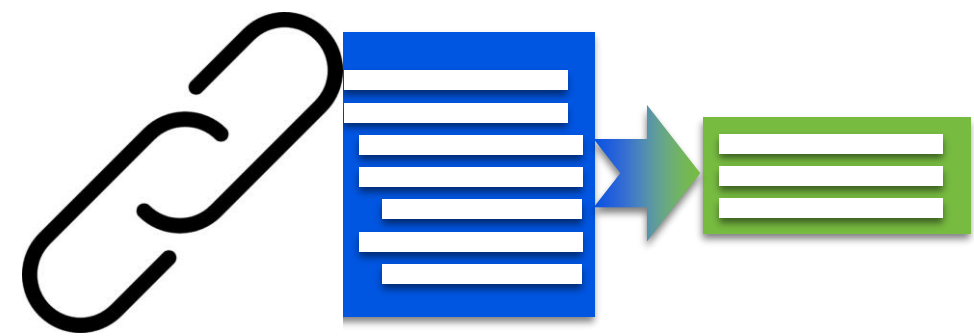
Reliable Intermittent Execution

Task-based Programming Models



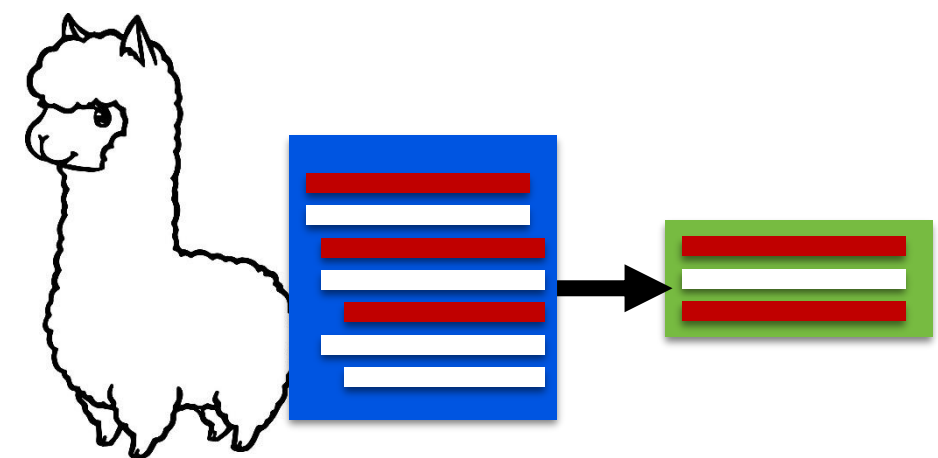
Dino

Arbitrary Task-based Intermittent Programming [PLDI'15]



Chain

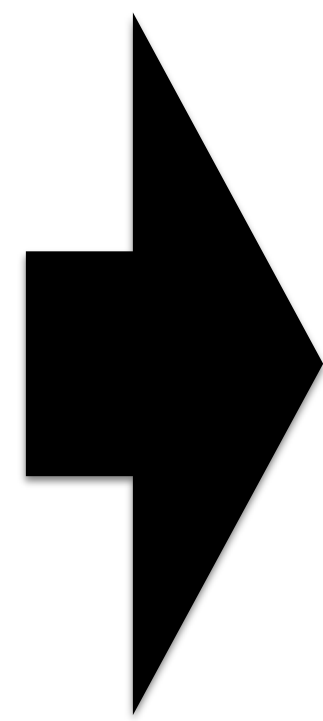
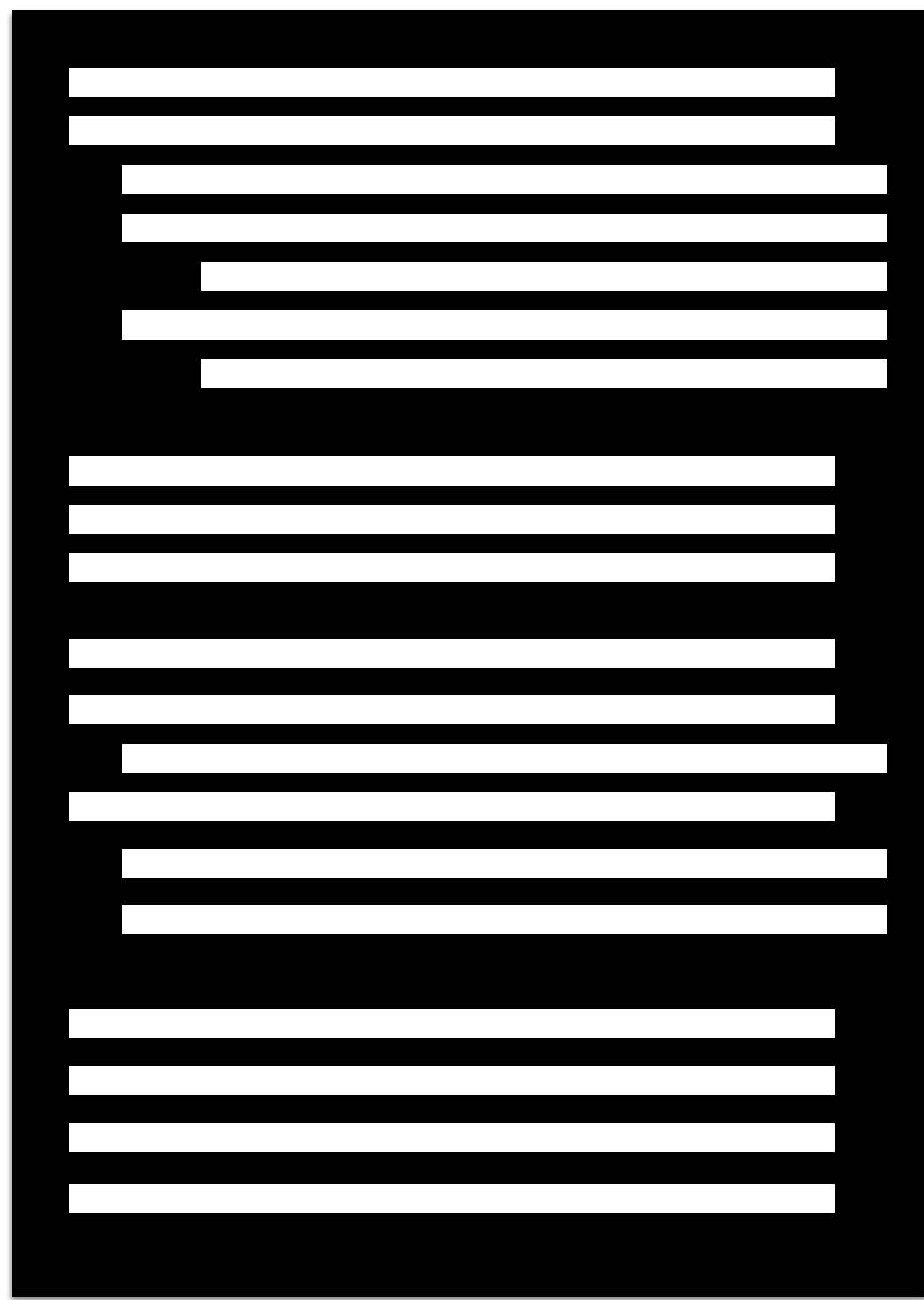
Task-based Intermittence w/ Static Data Duplication [OOPSLA'16]



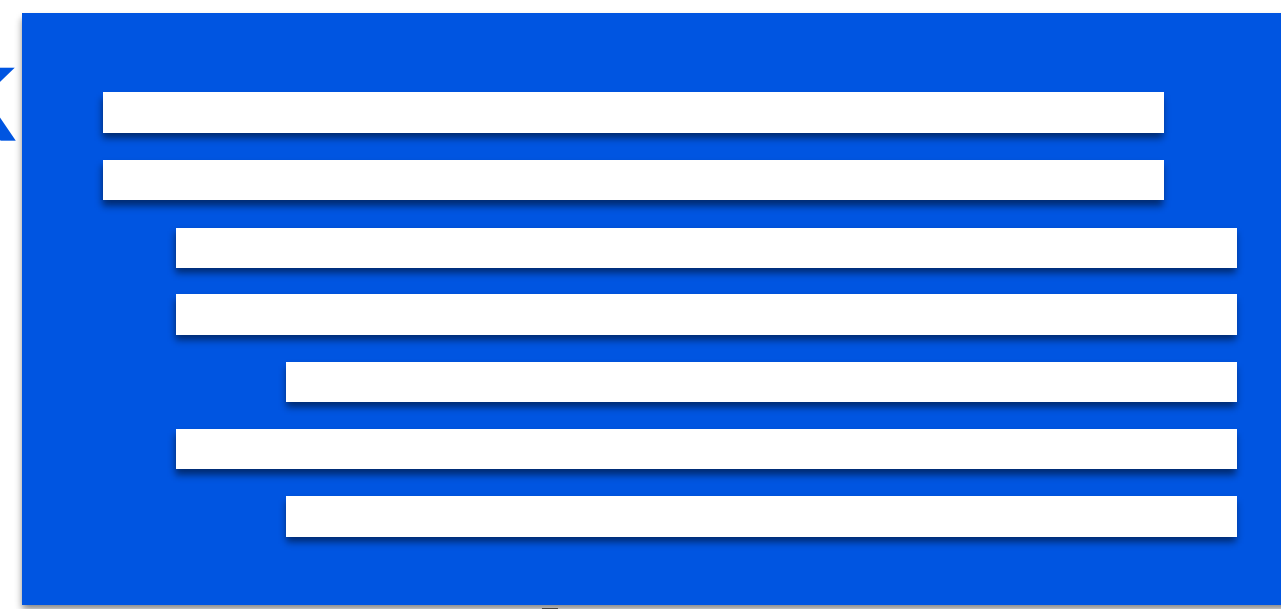
Alpaca

Task-based Intermittence w/ Dynamic Privatization [OOPSLA'17]

Original Program



Task



Tasks have *atomic* all-or-nothing semantics



Task Boundary

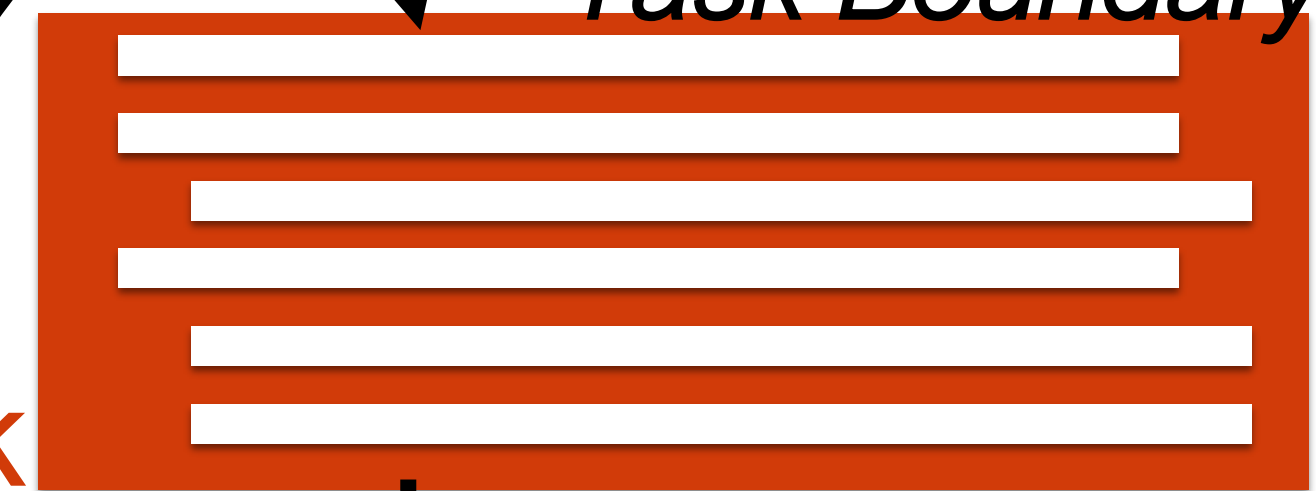
Task

Task boundaries selectively checkpoint *volatile* and *non-volatile* state

No *implicit* control-flow!

Task Boundary

Task



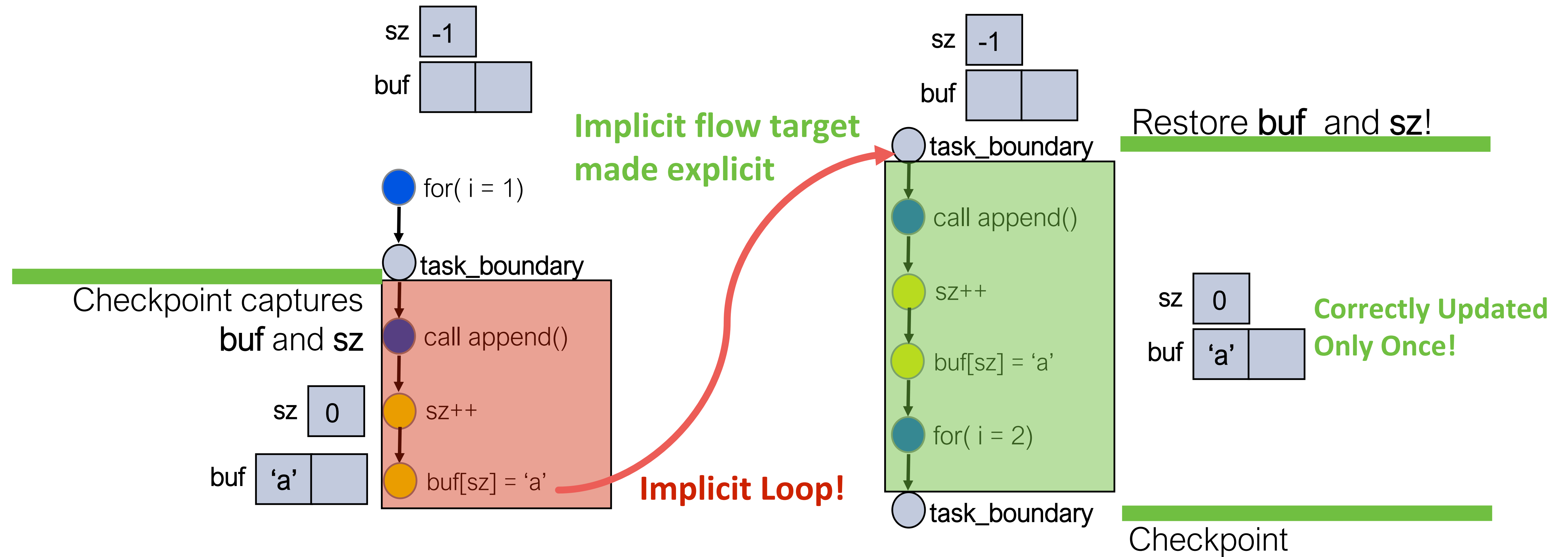
Task Boundary

Task

Task-atomic Execution in DINO

[PLDI '15]





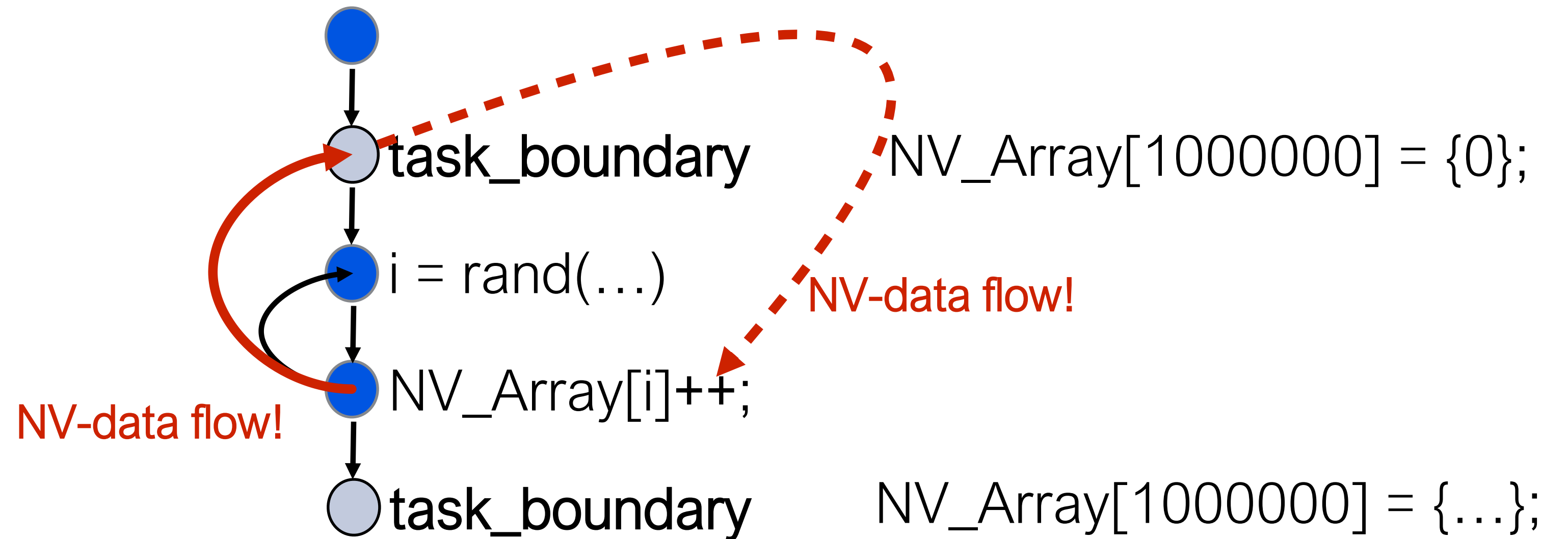
DINO Eliminates OoTA Values

[PLDI '15]

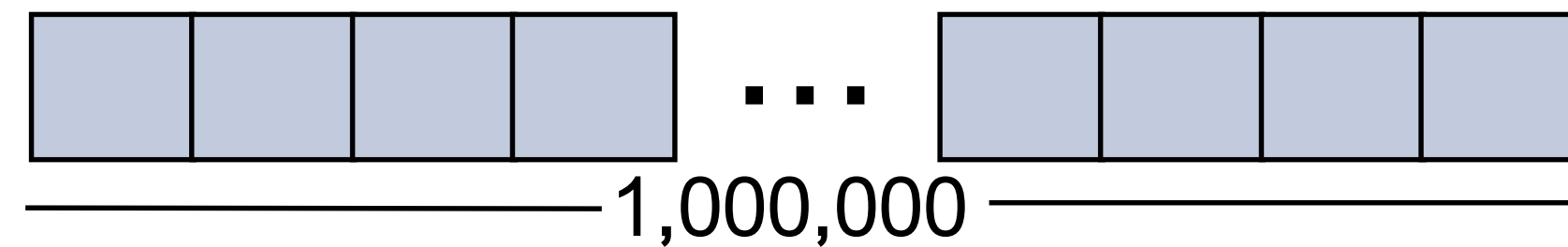
```
int NV_Array[1000000];
task_boundary
i = rand(1000000);
```

```
NV_Array[i]++;
```

```
task_boundary
```

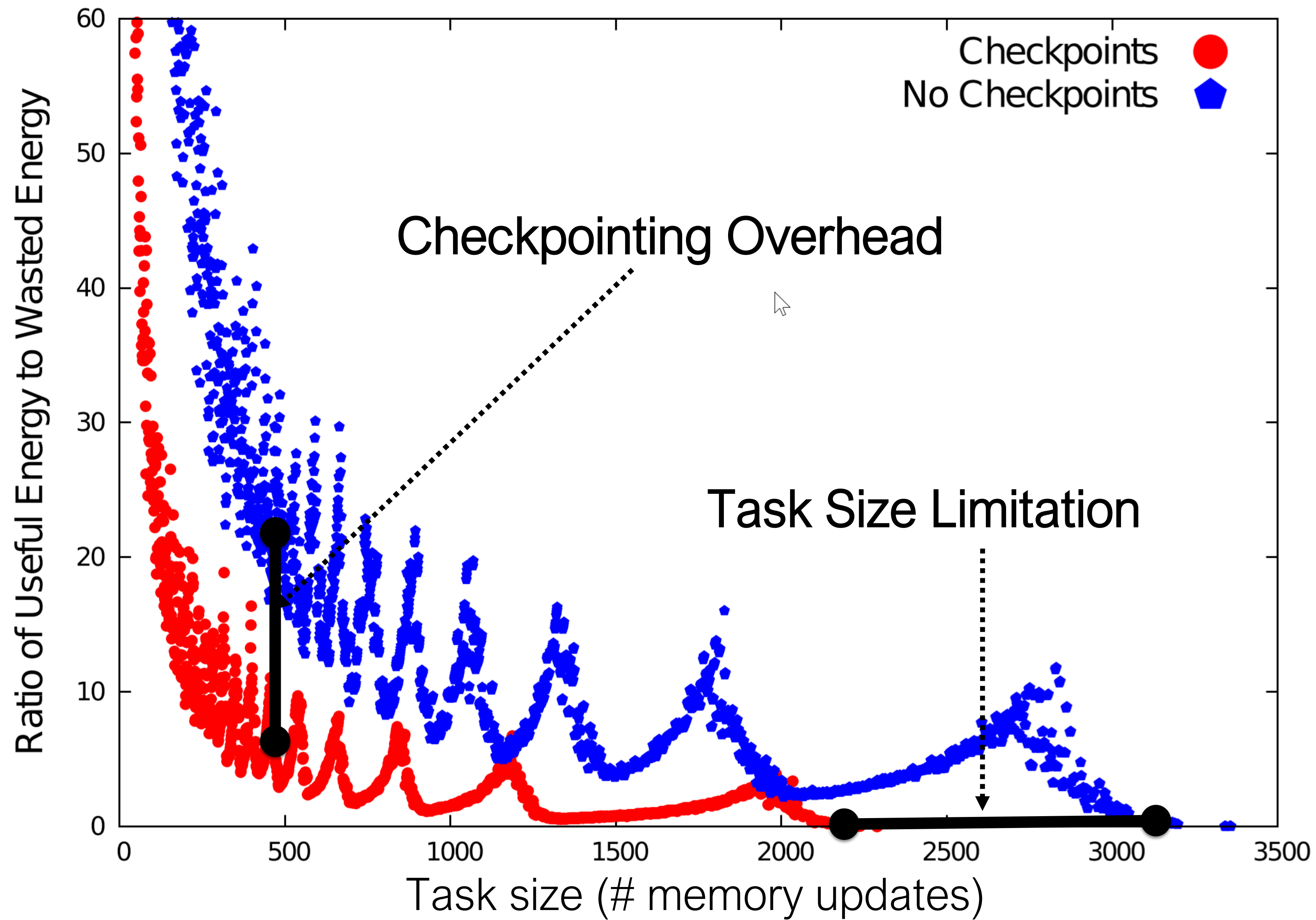


Compiler cannot know i in advance! Must version all of `NV_Array`.



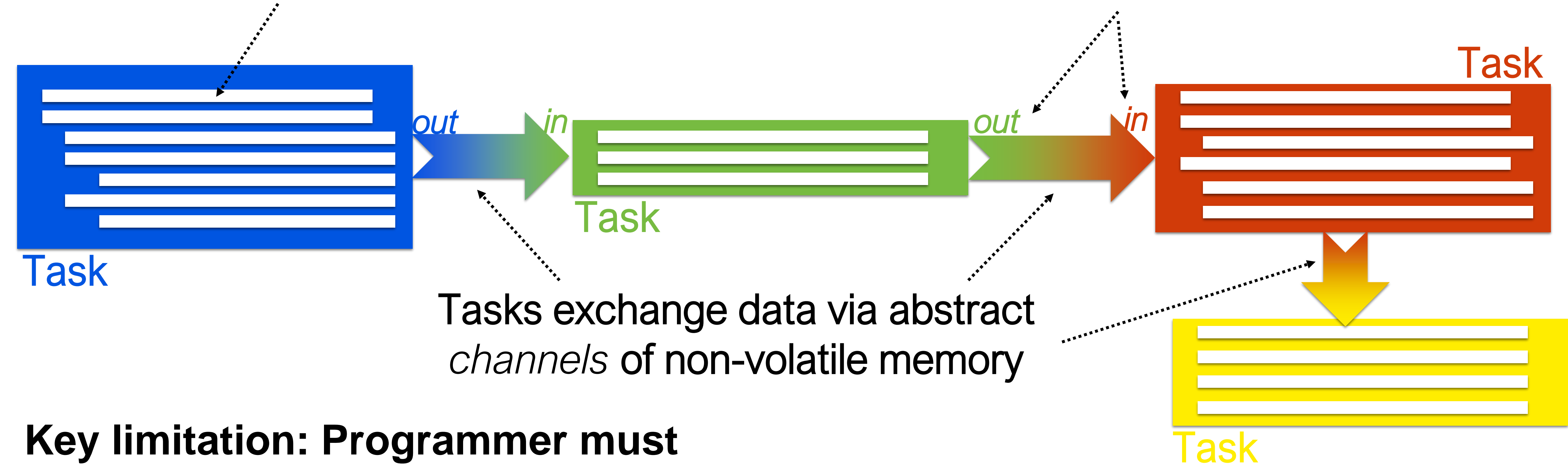
What data to include in checkpoint?

All NV data that **may** be accessed before a reboot, **plus volatile**.

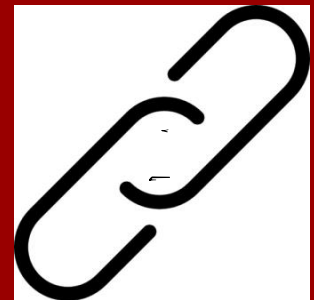


Programmer explicitly defines tasks and task-flow graph in code

Channels *statically multiversion* data separating inputs from outputs



Key limitation: Programmer must learn new programming model

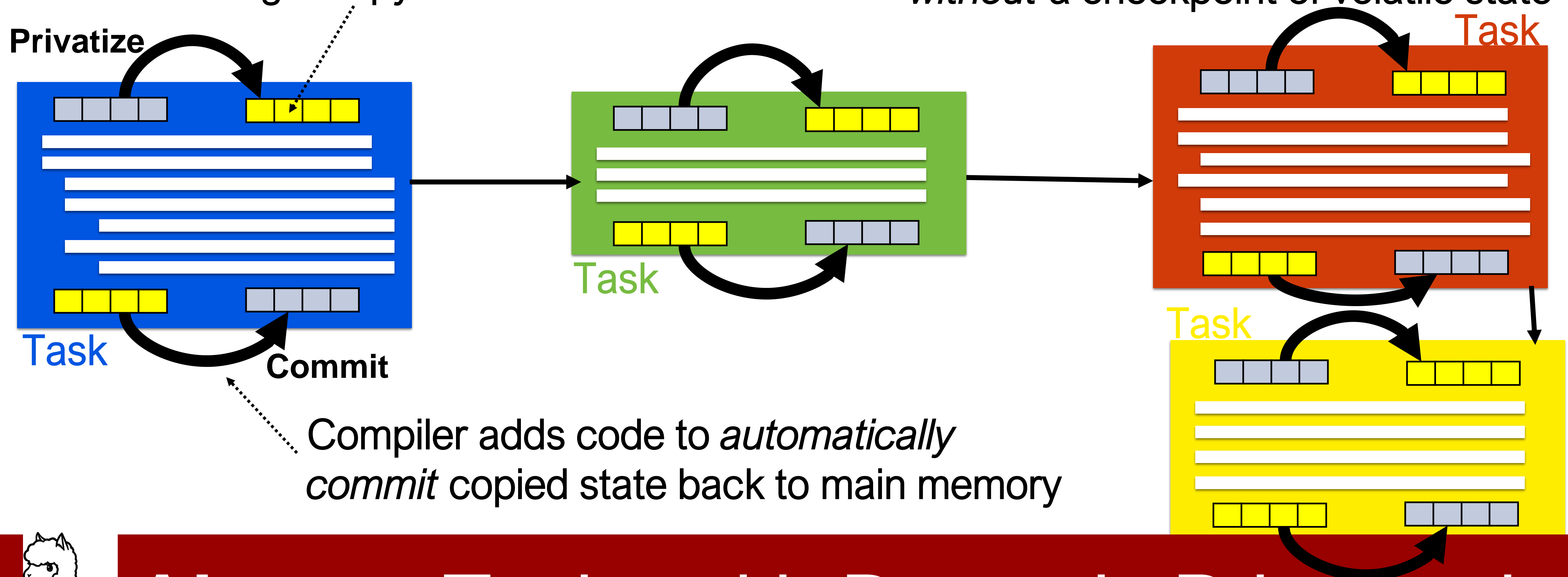


Chain: Static data multi-versioning

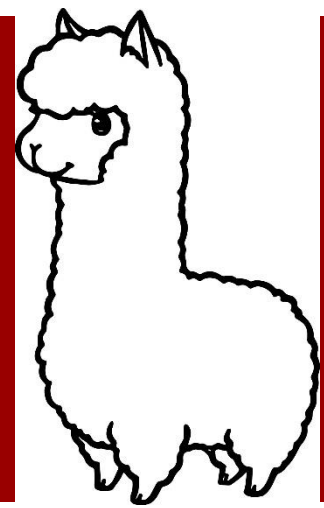
[Colin, et al OOPSLA '16]

Compiler *automatically privatizes* data creating a copy safe to use in a task.

Alpaca tasks are restartable *without* a checkpoint of volatile state



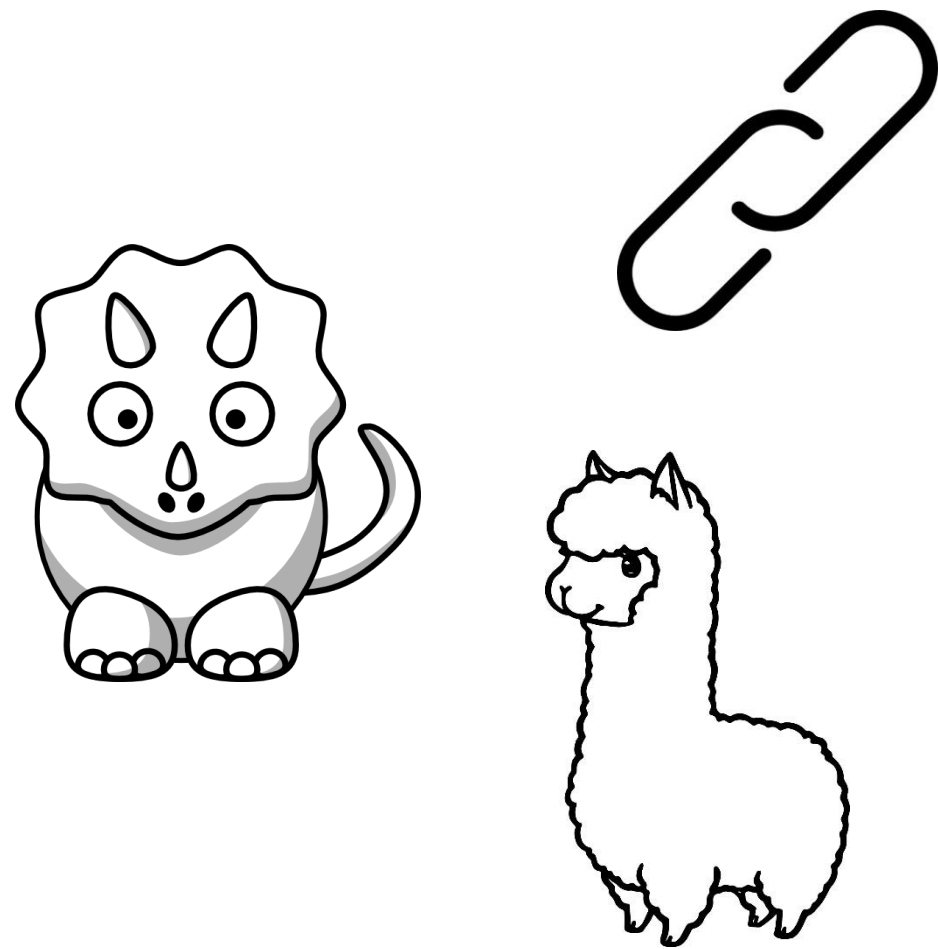
Compiler adds code to *automatically commit* copied state back to main memory



Alpaca: Tasks with Dynamic Privatization

[Maeng, et al OOPSLA '17]

DINO, Chain, Alpaca Programming Model



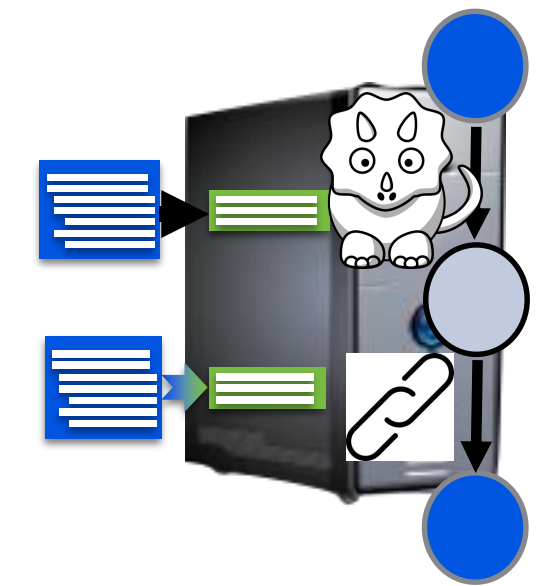
Task, Channel Definitions

Task-aware Compiler



Task Data-flow Analysis
Link to Task Runtime
Channels, Versioning,
Privatization

Runtime Library



DINO: Checkpoint & Recovery
Chain: Task Management
Alpaca: Privatization/Commit

Prototype Implementations

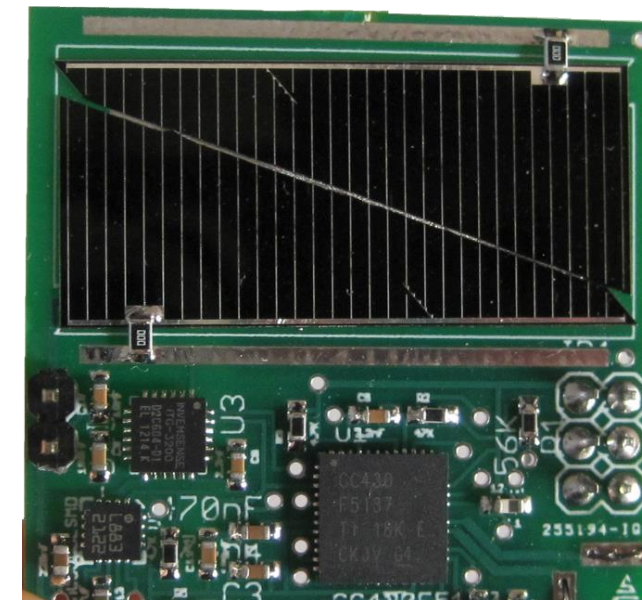
Energy-harvesting Platform Prototypes



WISP5 Energy-harvesting Device



RF Power Setup



Custom PCBs

Applications

Activity Recognition
(accelerometer+ML)

MIDI Natural User Interface

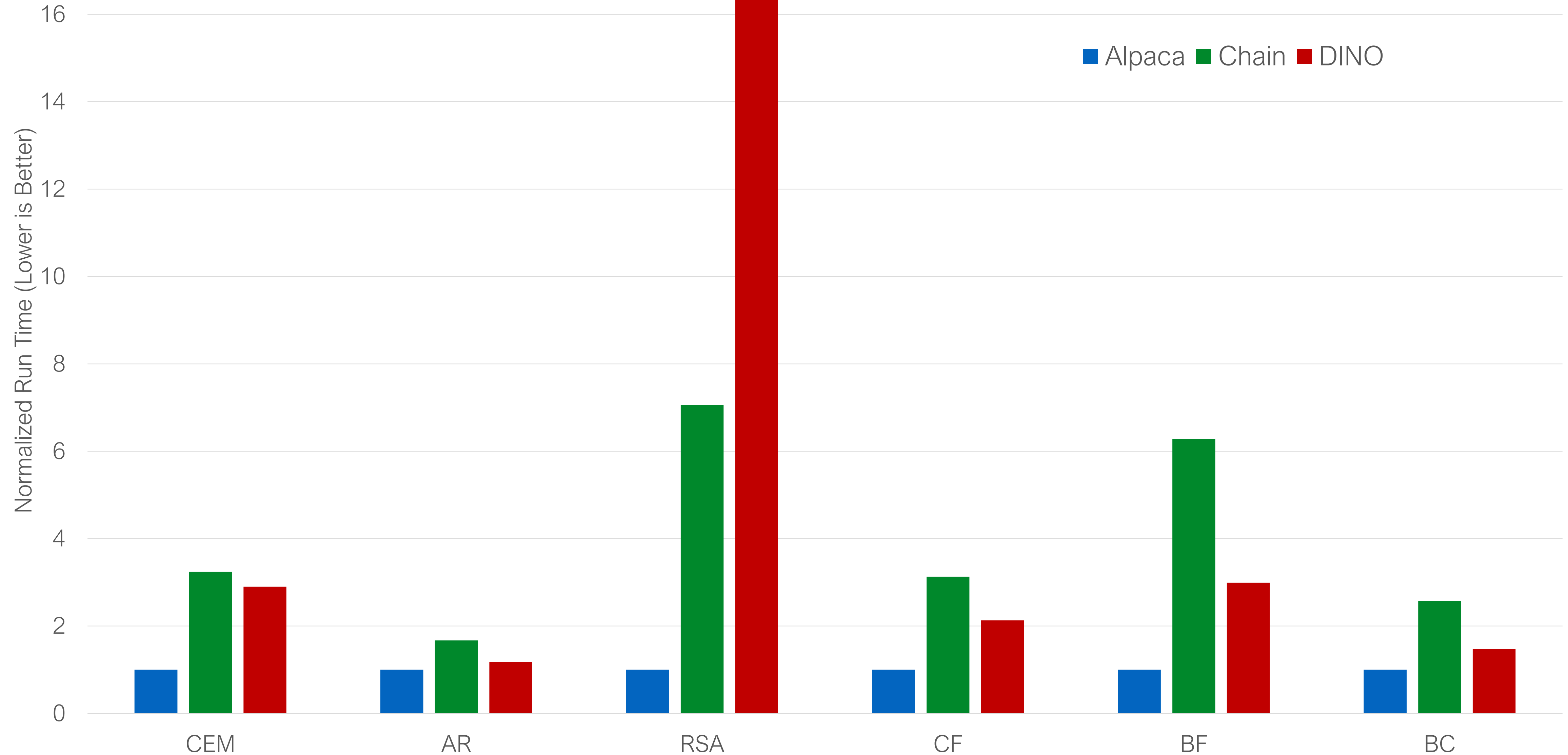
Cold-chain Equipment Monitor

Multi-granular Sensor Log

Key Result:

Applications suffer **errors & failures**
without our system support for tasks

Alpaca outperforms DINO and Chain





Use Alpaca, Chain and DINO for your research!
<http://intermittent.systems> | <http://github.com/CMUAbstract>
<https://cmuabstract.github.io/alpaca-landing-page/>



Alpaca

A programming language for writing reliable software for intermittent, energy-harvesting computer systems

Download VM (10.67GB)
VM contains Alpaca and other previous state-of-the-art systems (Chain, DINO)
UserID: reviewer PW: review.

Pull source code from Github
<https://github.com/CMUAbstract/alpaca-oopsla2017>

CMUAbstract / alpaca-oopsla2017

Unwatch 2 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Settings Insights

all-in-one repo for artifact eval & distribution

10 commits 1 branch 0 releases 1 contributor

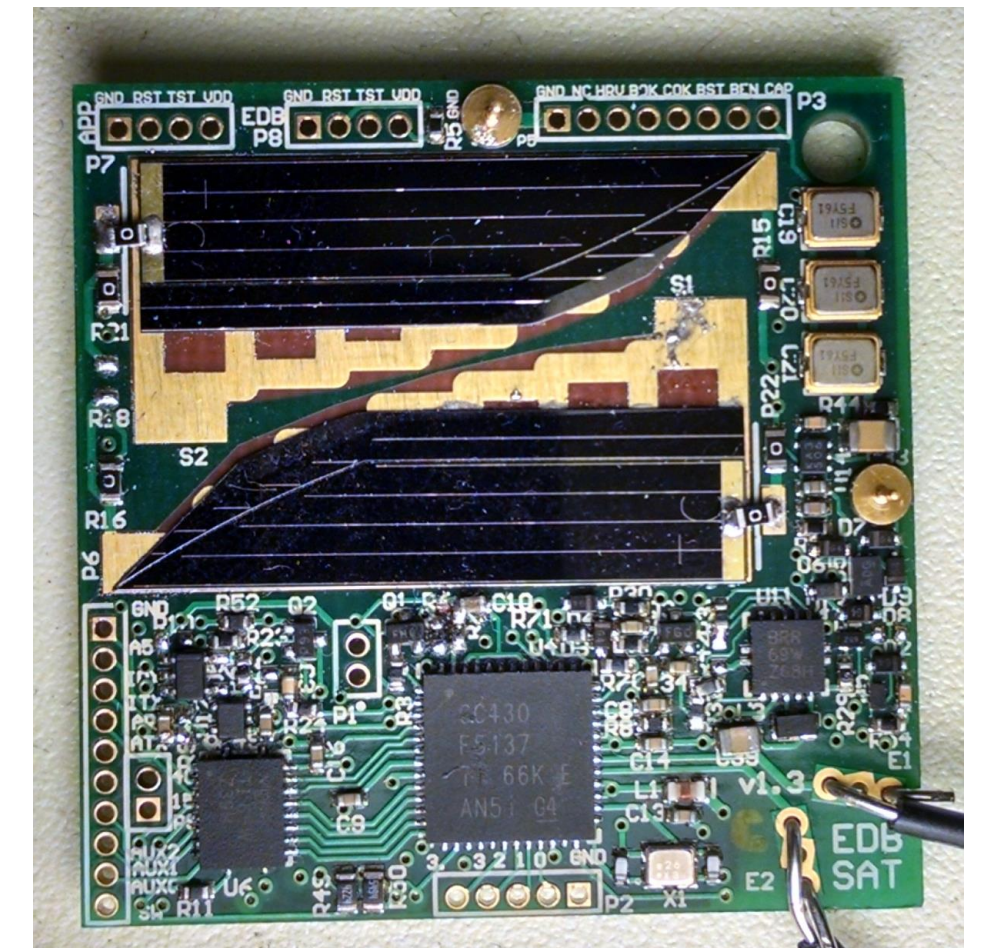
Branch: master New pull request Create new file Upload files Find file Clone or download

crapbox submodule update Latest commit 338aa9b 6 hours ago		
bld	init	23 days ago
data	init	23 days ago
ext	submodule update	6 hours ago
golden_result	init	23 days ago
src	init	23 days ago
.gitignore	modified gitignore	23 days ago
.gitmodules	changing ssh to https	23 days ago
Makefile	init	23 days ago
README.md	README changed	23 days ago

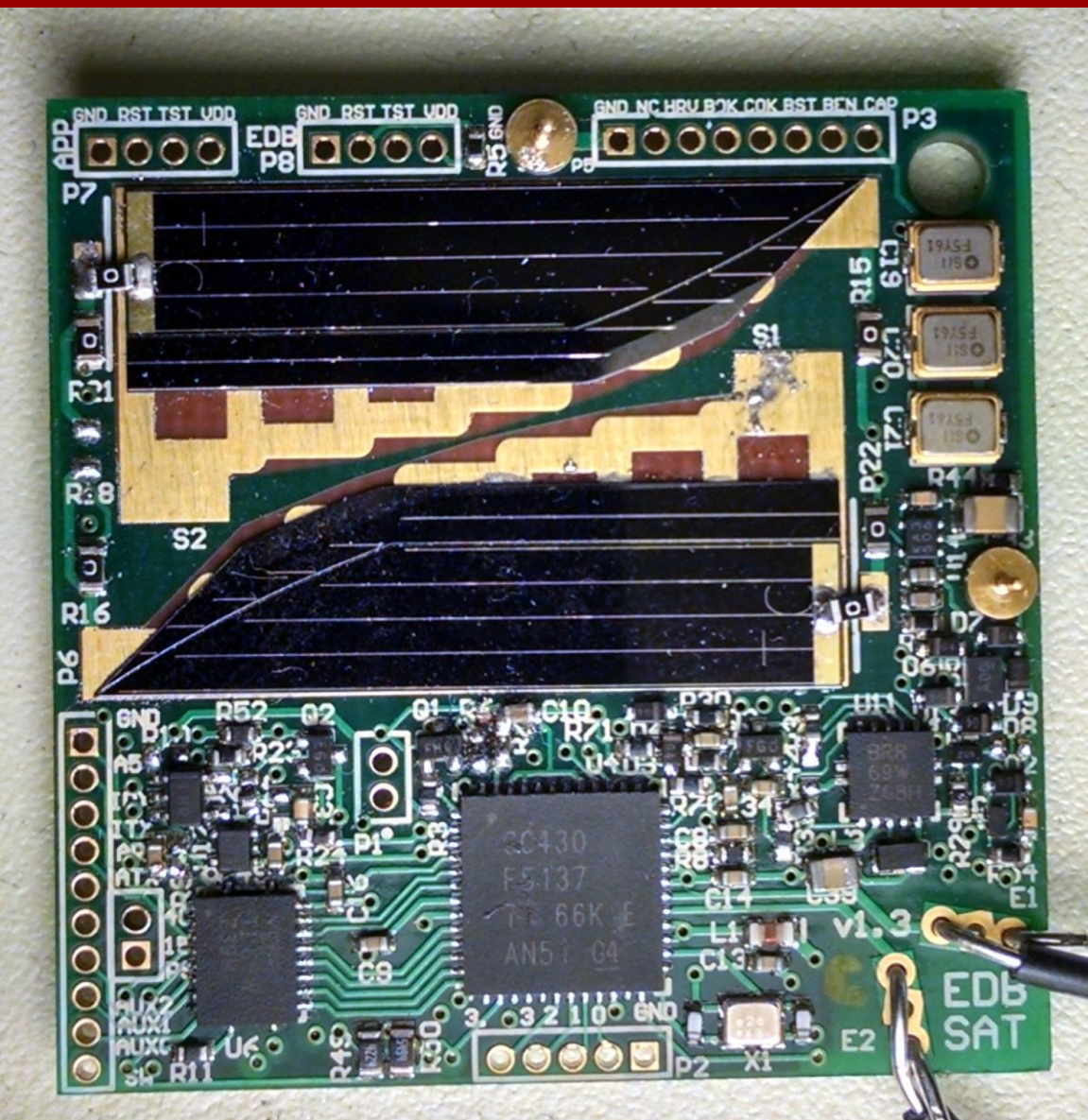
```
Terminal - reviewer@chainvm:~/src/apps/oopsla2017
File Edit View Terminal Tabs Help
mkdir -p ". /."
/opt/llvm/llvm-install/bin/clang -emit-llvm -c -DALPACA -I../src/include -I../src/include/...
postdinc++ -isysroot /none -O0 -g -std=c99 -Wall -MD -I /opt/ti/mspgcc/lib/gcc/msp430-elf/4.9.1/in...
rc -I/home/reviewer/src/apps/alpaca-oopsla2017/ext/libio/src/include -I/home/reviewer/src/apps/d...
../src/alpaca.c:18:16: warning: incompatible pointer types initializing 'uint8_t **' (aka 'uns...
[-Wincompatible-pointer-types]
nv uint8_t** data_src_base = &data_src;
^
../src/alpaca.c:22:16: warning: incompatible pointer types initializing 'uint8_t **' (aka 'uns...
[-Wincompatible-pointer-types]
nv uint8_t** data_dest_base = &data_dest;
^
../src/alpaca.c:26:16: warning: incompatible pointer types initializing 'unsigned int *' with...
nv unsigned* data_size_base = &data_size;
^
3 warnings generated.
/opt/llvm/llvm-install/bin/llvm-link -o libalpaca.a.bc alpaca.bc
make[2]: Leaving directory '/home/reviewer/src/apps/alpaca-oopsla2017/ext/alpaca/AlpacaRuntime/L...
make[1]: Leaving directory '/home/reviewer/src/apps/alpaca-oopsla2017/bld/alpaca'
make TOOLCHAIN=alpaca -e -C bld/alpaca all
make[1]: Entering directory '/home/reviewer/src/apps/alpaca-oopsla2017/bld/alpaca'
mkdir -p ". /."
/opt/llvm/llvm-install/bin/clang -emit-llvm -c -I/home/reviewer/src/apps/alpaca-oopsla2017/ext/L...
s/alpaca-oopsla2017/ext/libmspprintf/src/include -I/home/reviewer/src/apps/alpaca-oopsla2017/ext...
spbase/src/include -DVERBOSE=0 -I/src/include -DBOARD_MSP_TS430 --target=msp430 -D_MSP430FR5969...
i/mspgcc/lib/gcc/msp430-elf/4.9.1/include -I /opt/ti/mspgcc/msp430-elf/include -I /opt/ti/mspgcc...
apps/alpaca-oopsla2017/ext/libmsp/src/include -I/home/reviewer/src/apps/alpaca-oopsla2017/ext/li...
nclude -I/home/reviewer/src/apps/alpaca-oopsla2017/ext/alpaca/AlpacaRuntime/libalpaca/src/incl...
alpaca.bc
/opt/llvm/llvm-install/bin/opt -debug -stats -load /home/reviewer/src/apps/alpaca-oopsla2017/ext...
-alpaca \
-o main cem alpaca.alpaca.bc main cem alpaca.bc
Args: /opt/llvm/llvm-install/bin/opt -debug -stats -load /home/reviewer/src/apps/alpaca-oopsla20...
.bc main_cem_alpaca.bc
Features:
CPU:generic
/opt/llvm/llvm-install/bin/llvm-link -o cem.a.bc main cem alpaca.alpaca.bc /home/reviewer/src/ap...
alpaca-oopsla2017/ext/libmspmath/bld/clang/libmspmath.a.bc /home/reviewer/src/apps/alpaca-oopsla20...
/alpaca/AlpacaRuntime/libalpaca/bld/clang/libalpaca.a.bc
/opt/llvm/llvm-install/bin/llc -O0 cem.a.bc -o cem.S
```

Intermittent Computing Platforms

Hardware, Programming Models, System Software



Intermittence-safe HW/SW Systems [ASPLOS 2018]



EDBSat Chip-scale satellite

- 3cmx3cmx4mm, solar power
- Magnetometer + gyro sensors
- Low-power radio (to earth)
- Custom “burst” power system
- Integrated HW/SW diagnostics
- Programmed in intermittence-safe Chain programming lang.
- Launch partner: KickSat/NASA



Cappybara multi-sensor platform

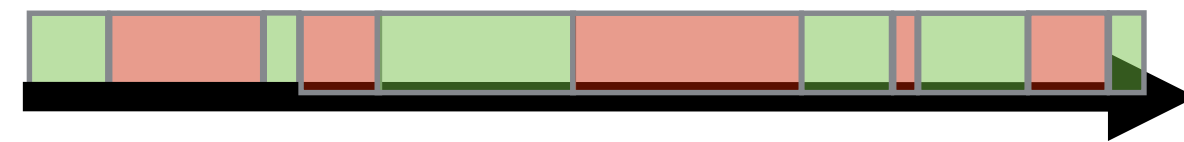
- 5cmx5cm, solar/RF power
- 10 sensor array
- Bluetooth Low-energy Radio
- Software-switchable variable-capacity “burst” power system
- Integrated HW/SW diagnostics
- Supports Alpaca & Chain intermittent programming langs.

Key Insight: Co-design PL, OS, Arch, and power system for intermittent operation



The Intermittent Execution Model

Reasoning About Intermittently-powered Devices

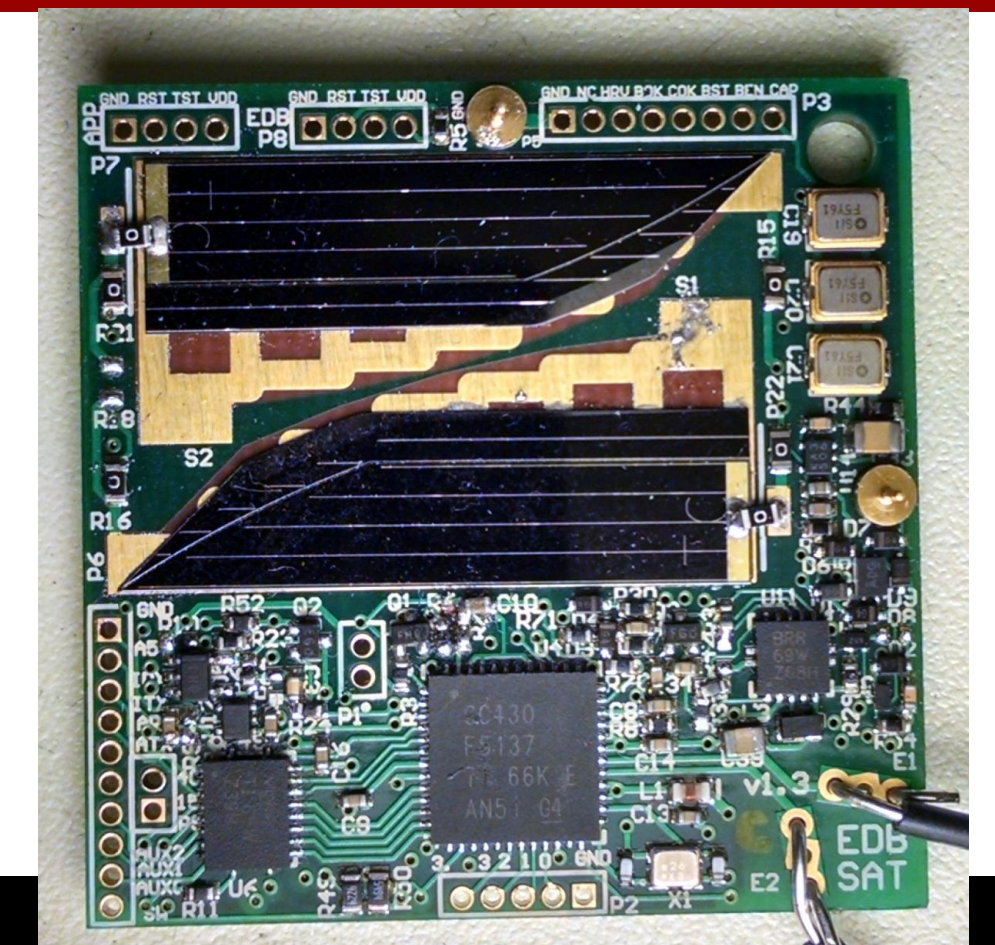


Designing for Intermittence

System Support for Reliable Intermittence

Intermittent Computing Platforms

Hardware, Programming Models, System Software



Solving the System Challenges of Intermittent Energy-harvesting Devices

Brandon Lucia | Assistant Professor, CMU ECE

ABSTRACT Research Group - <https://intermittent.systems>

with PhD Students: Alexei Colin, Kiwan Maeng, Emily Ruppel, Vignesh Balaji
Graham Gobieski, Milijana Surbatovich

MS Students: Preeti Murthy, Amanda Marano, Neil Ryan, Devon White, Chris Meredith

BS Students: Graham Harvey, Mark McElwaine, Hannah Tomio

Industry Collaborators: Zac Manchester (Harvard/KickSat)

Alanson Sample (Disney Research Pittsburgh)

