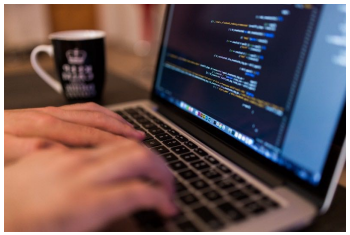
Energy Efficient Computing Systems Exploiting Online Tuning and Output Quality Management

Gianluca Palermo

Extra-Functional Properties

Functional Description

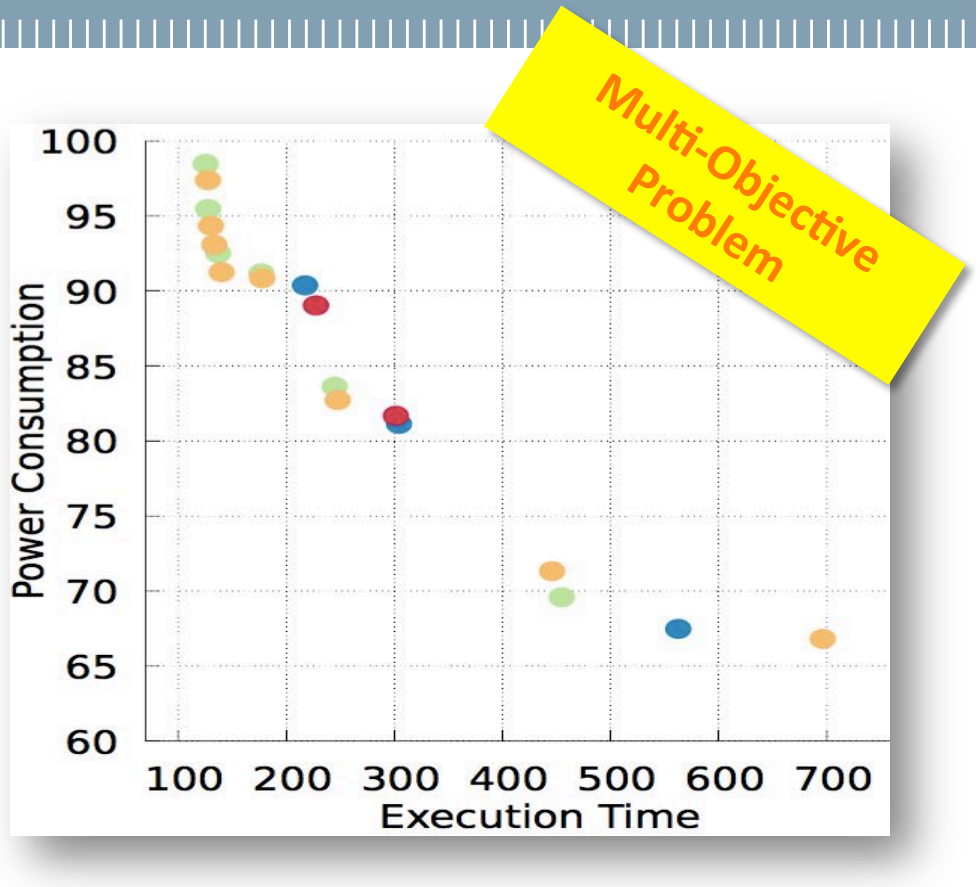


What to do…

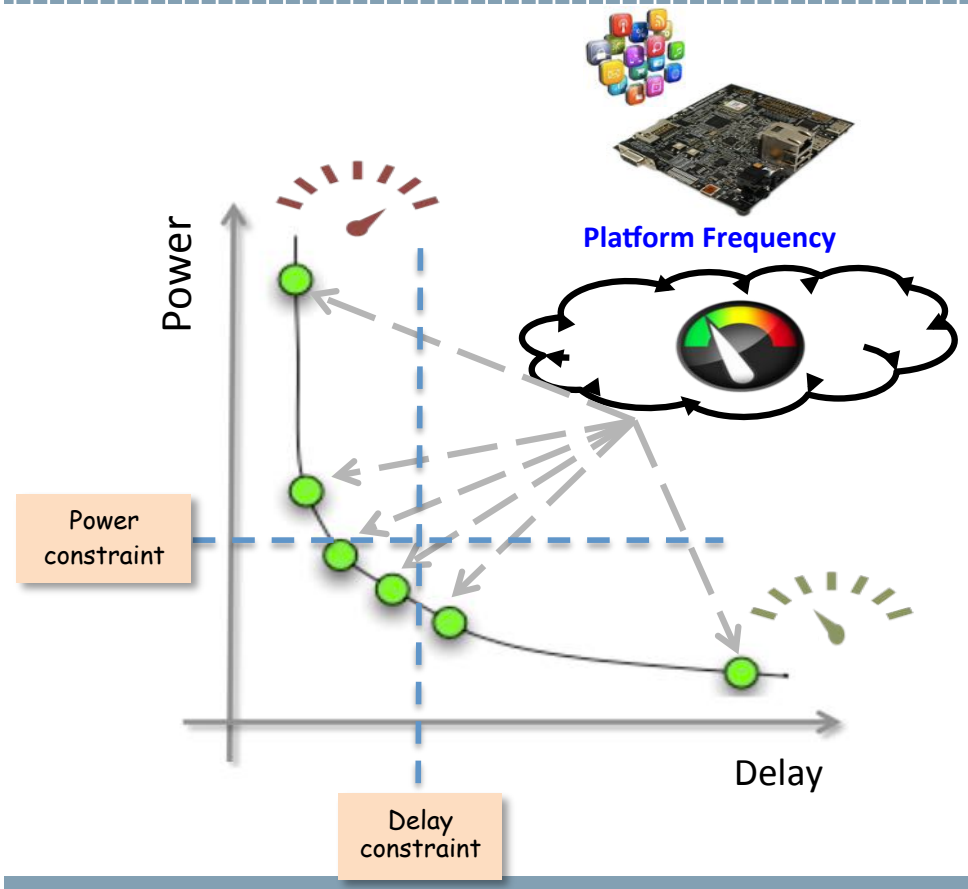
Extra-Functional Properties

How It is done…

Extra-Functional Properties

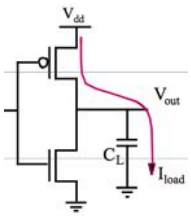


Simple Power-Performance trade-off



Dynamic Power

$$Power = f * C * V^2 + P_{static}$$



Energy constraint ???

Energy = Power * Delay

Energy Vs Power

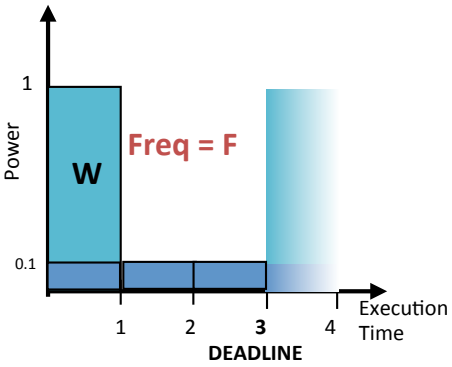
- *Power* poses constraints
  - E.g. power delivery or cooling solution

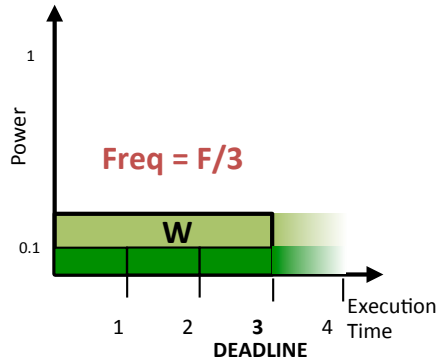
- *Energy* is in most of the cases the ultimate metric
  - It measures the *cost* of performing a fixed task

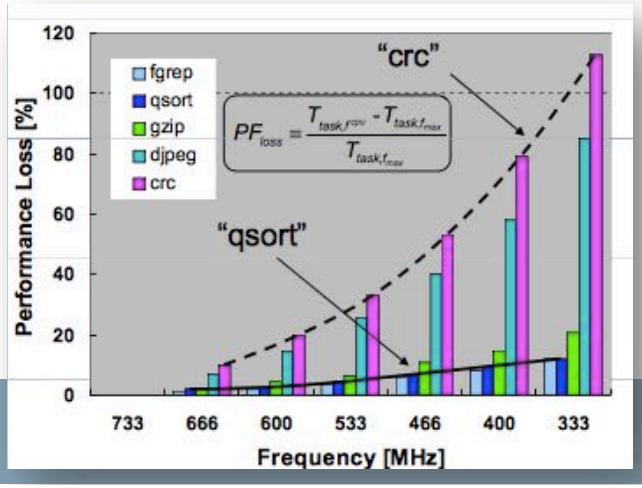
- ***How can I reduce Energy?***
  - Voltage and Frequency Scaling
  - Dynamic Power Switching
  - […]

(Dynamic) Voltage and Frequency Scaling (DVFS)

- Small reductions in voltage can be very significant
  - Power is proportional to frequency and square of voltage
- Effective way to reduce energy by providing *just-enough* computing power
- F and V are not independent variables $f \propto (V_{dd} - V_{th})^{\alpha} / V_{dd}$
- Choosing the right VF operating point is not straightforward



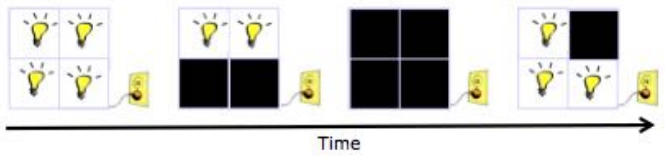




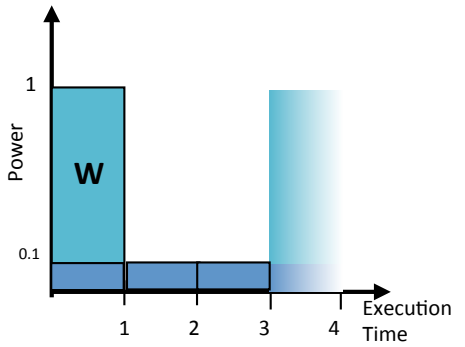
K. Choi, R. Soma, M. Pedram "Dynamic voltage and frequency scaling based on workload decomposition" ISLPED 2004

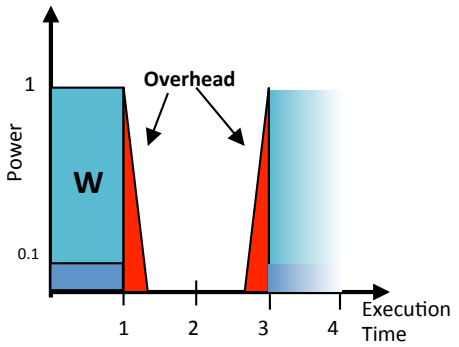
Dynamic Power Switching (DPS)

- Keeping devices powered-on consumes power
  - Cut or reduce power on idle device portions



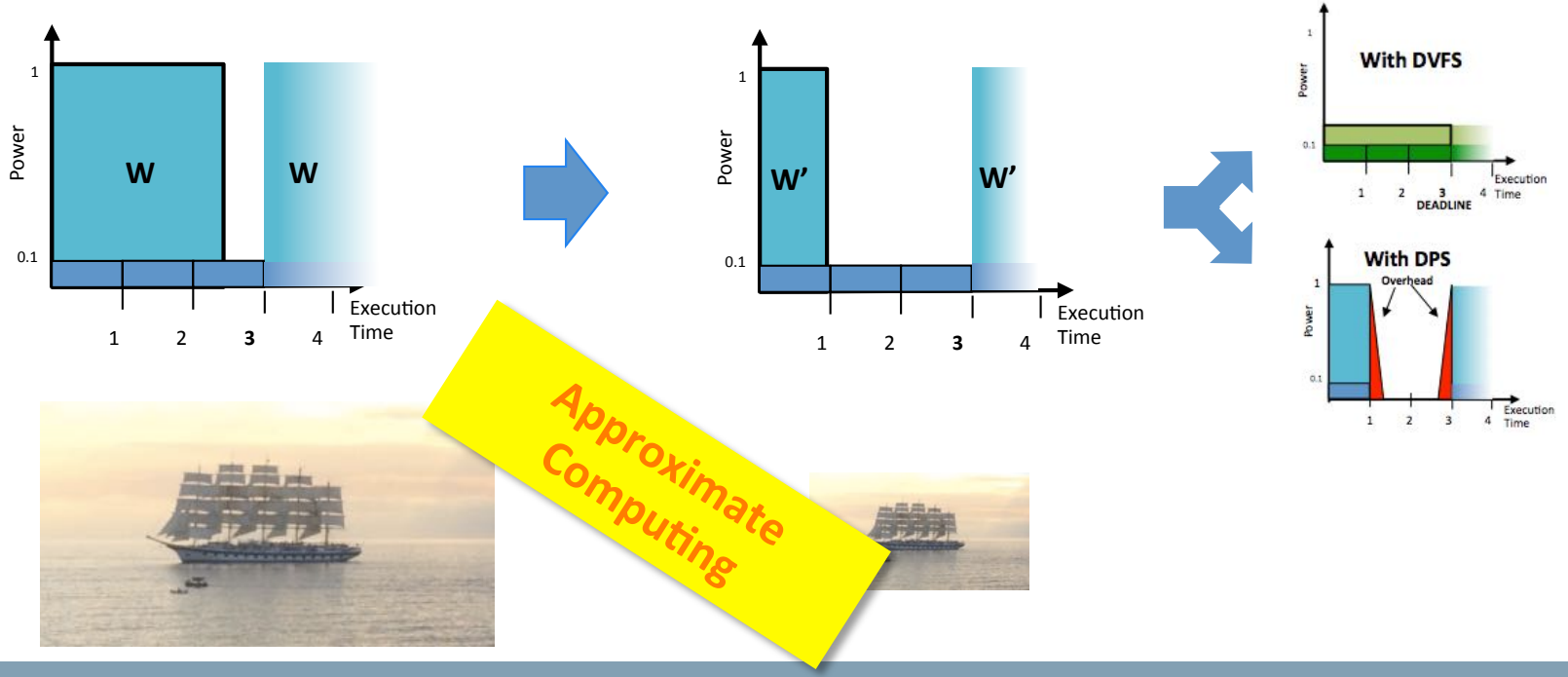
- Significant overheads for switching to deep low-power states
  - Different low-power states
    - Some of them need the context save/restore
    - E.g. Retention states –> off states
  - Wakeup latency reduces the responsiveness
- In some contexts also called *Race-to-Halt*



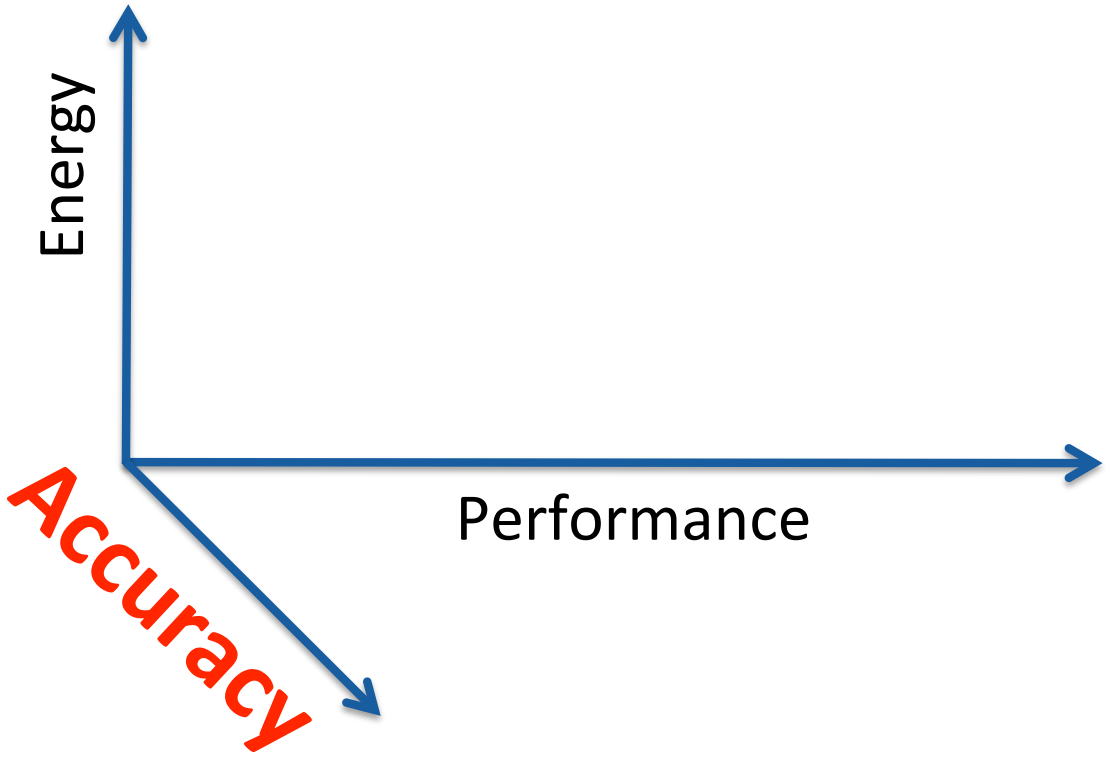


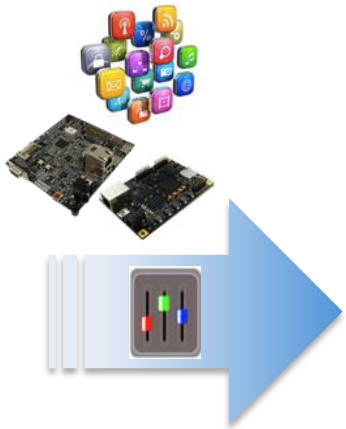
… Longer *Week-Ends* are my Dream

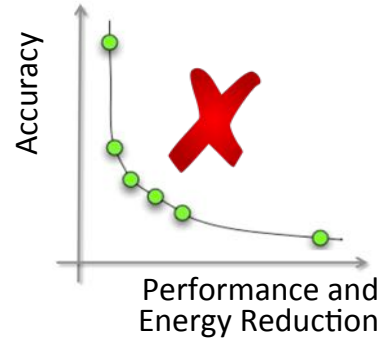
Is there any way to reduce the amount of computation to be performed to consume less?

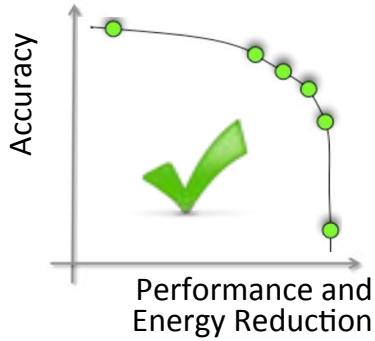


Approximate Computing









Approximate Computing Applications

- Approximation adds ***complexity***
  - Not all codes can handle it

- … but many expensive applications or kernels are naturally error-tolerant
  - Make use of *analog inputs*
    - E.g. operating noisy real-world data from noisy sensor
  - Provide *analog output*
    - E.g. targeting human perception
  - Provide multiple *good-enough* results or *no unique answer*
    - E.g. web search
  - Compute *iteratively towards convergence*
    - E.g. convergent applications over the number of iterations.

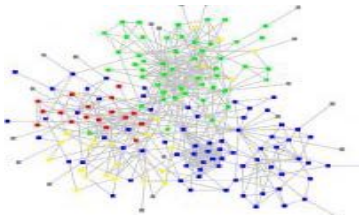
V. Chippa et al."Analysis and characterization of inherent application resilience for approximate computing " DAC 2013.

Possible Application Domains…

Image Processing

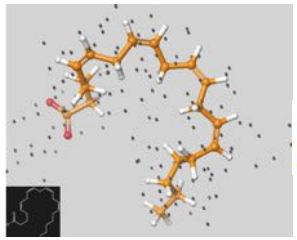
Robotics

Big Data Analytics



Graph Analytics

Multimedia



Drug Discovery

Traffic Prediction

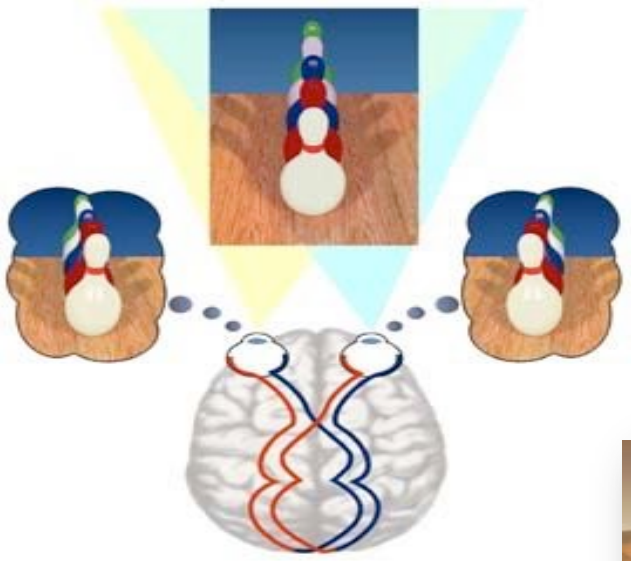
… where 100% of accuracy not always required

HOWEVER …

… we have to pay attention

No Traffic

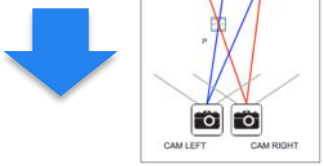
Let's have an example…

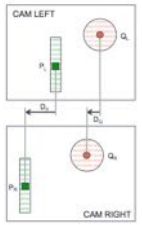


2 eyes = 3 dimensions

Left camera

Right camera

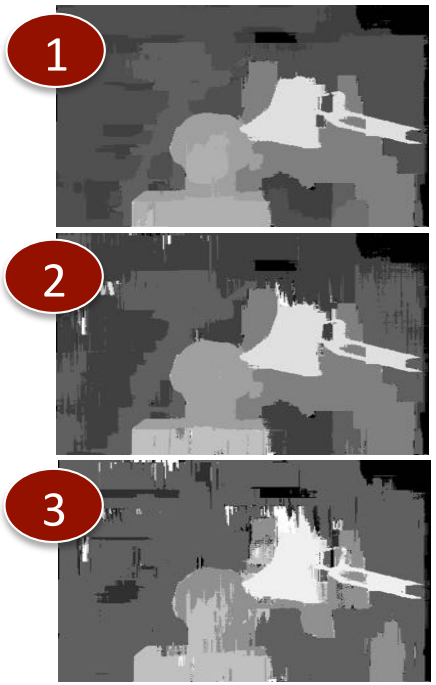




Reference disparity

K. Zhang et al. "Cross-Based Local Stereo Matching Using Orthogonal Integral Images". IEEE Transactions On Circuits and Systems For Video Technology 2009

Tunable Stereo Matching



QoR Disparity Error

5 Application Knobs*

Left camera

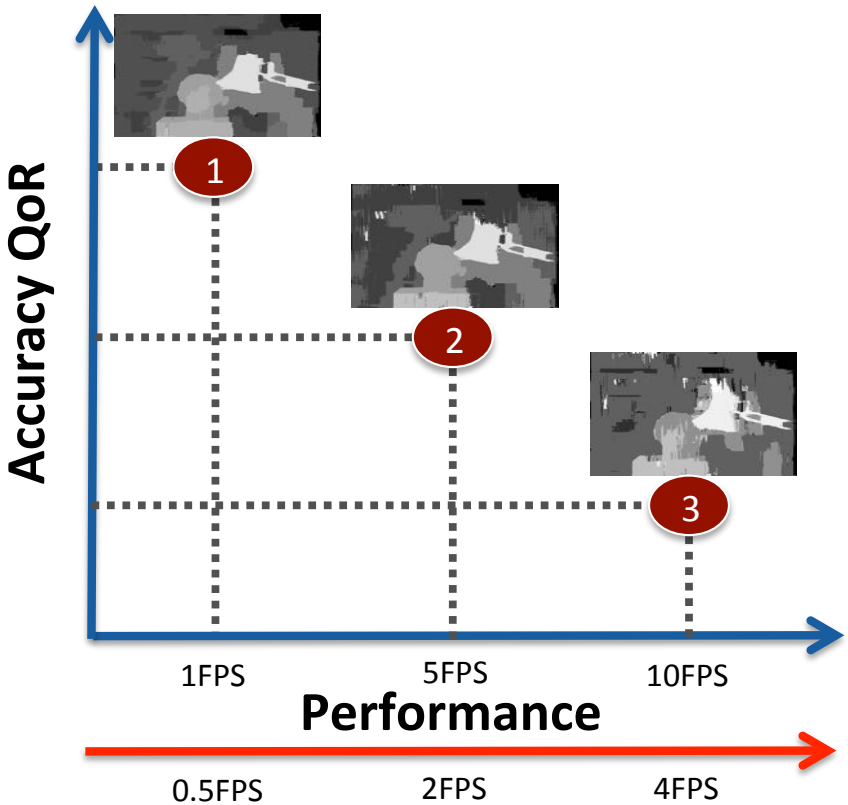
Right camera

Reference disparity

* Paone et al. "An Exploration Methodology for a Customizable OpenCL Stereo-Matching Application Targeted to an Industrial Multi-Cluster Architecture " In CODES+ISSS 2012

Trading-off Accuracy

D.Gadioli et al. "Application Autotuning to support runtime adaptivity in multicore architectures" SAMOS 2015

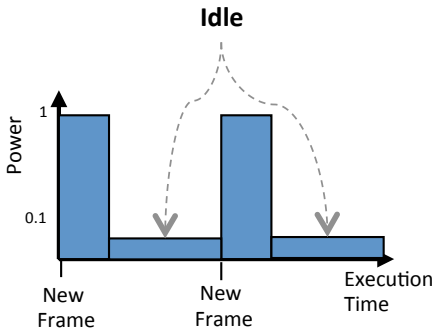


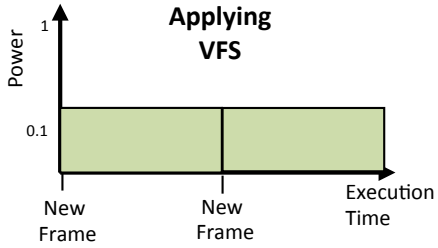
Extra-functional requirements:

What if …

1. Performance = 4FPS 2
2. Performance = F(Speed) 1 2 3
3. Min Energy; QOR>50%; Perf>=1

2

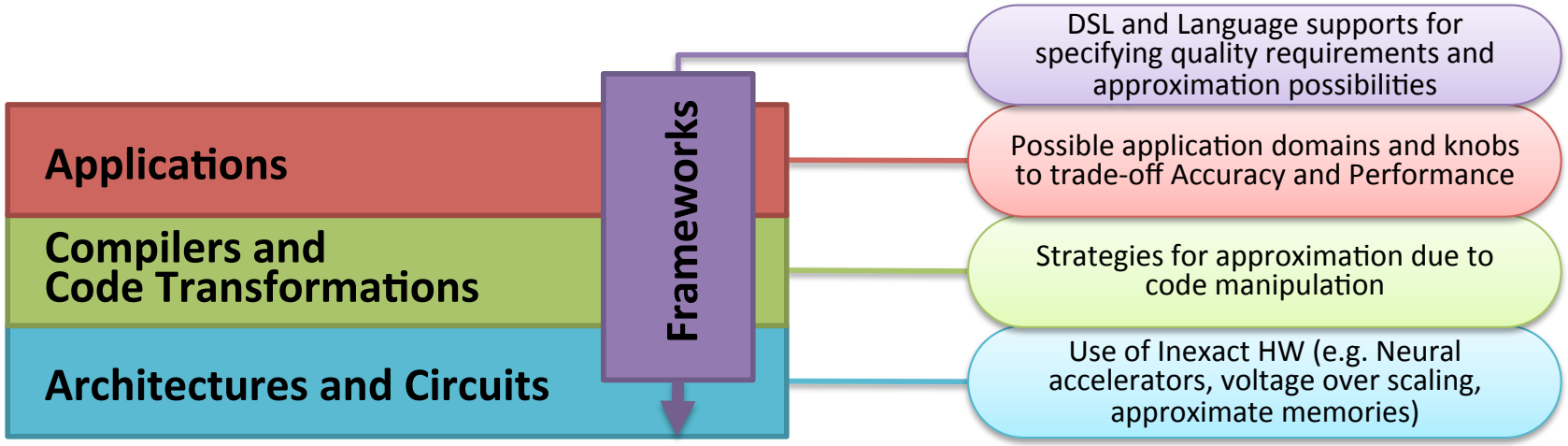




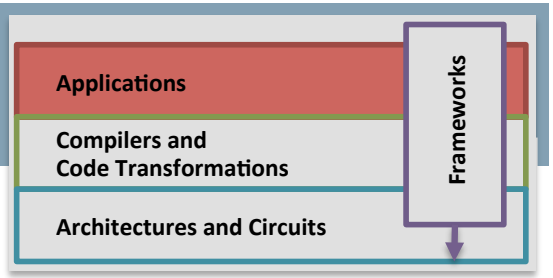
State of the Art…

In literature AC has been faced from different perspectives and at different levels

S. Mittal, "A survey of techniques for approximate computing," ACM Computing Surveys, pp. 1–34, 2016.

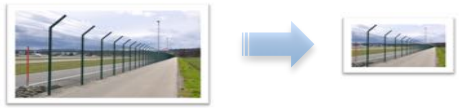


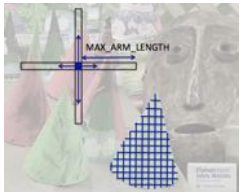
Application-level Approximations



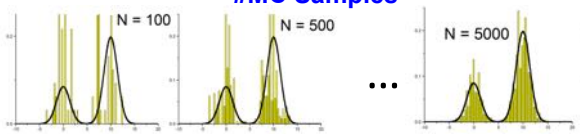
- *Application parameters*

Video Resolution





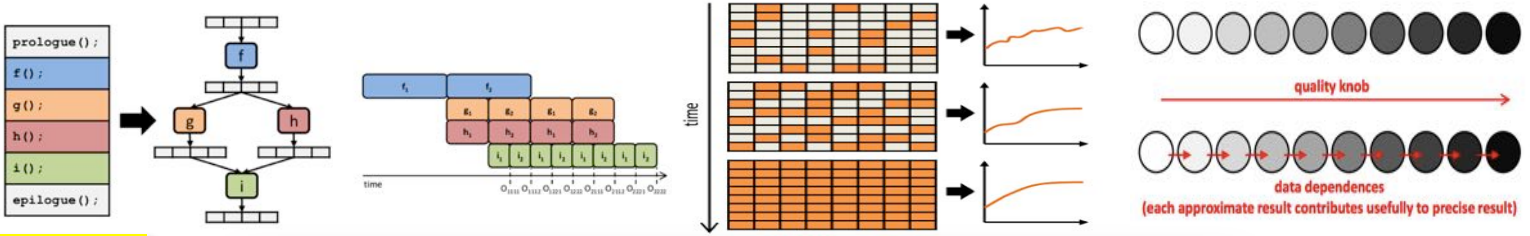
#MC Samples



- *Task skipping*

Computation model that represents an approximate application as a pipeline

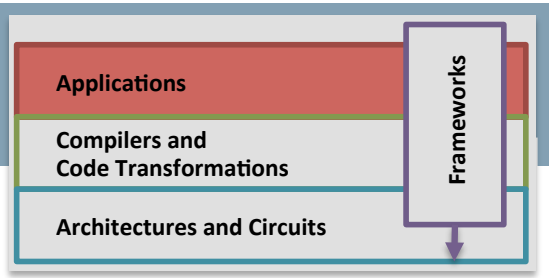
J. SanMiguel et al "The anytime automaton." ISCA 2016.



V. Vassilliadis et al " Exploiting Significance of Computations for Energy-Constrained Approximate Computing" IJPP 2016.

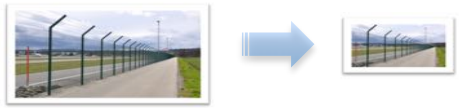
```
#pragma omp task [significant(...)] [label(...)]
    [in(...)] [out(...)] [approxfun(function())]
```

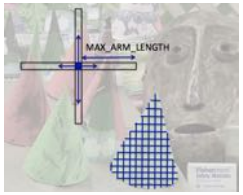
Application-level Approximations



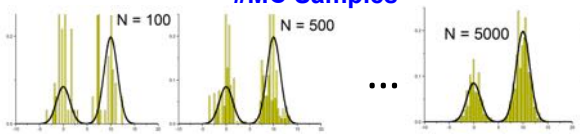
- *Application parameters*

Video Resolution





#MC Samples



- *Task skipping*

- *Multiversioning*
  - Considering different performance/accuracy trade-offs

```
switch (get_quality_level(/*...*/)) {
    case 0: foo(...) break;
    case 1: foo_approx_Q1(...) break;
    case 2: foo_approx_Q2(...) break;
        /* ... */
```

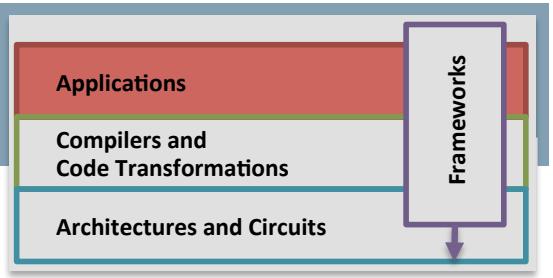
DSL or annotation based approaches

J Ansel et al. "PetaBricks: a language and compiler for algorithmic choice" PLDI 2009

B. Woongki et al "Green: a framework for supporting energy-conscious programming using controlled approximation." PLDI 2010

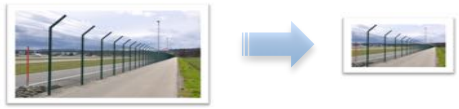
```
1  transform Sort
2  from A[n]
3  to B[n]
4  {
5    from(A a) to(B b) {
6      tunable WAYS;
7      /* Mergesort */
8    } or {
9      /* Insertionsort */
10   } or {
11     /* Radixsort */
12   } or {
13     /* Quicksort */
14   }
15 }
```

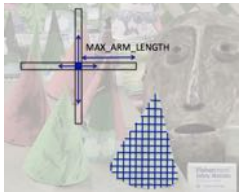
Application-level Approximations



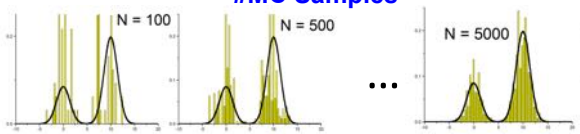
- *Application parameters*

Video Resolution





#MC Samples



- *Task skipping*

- *Multiversioning*
  - Considering different performance/accuracy trade-offs
- *Approximate Memoization*
  - User partial key for the lookup

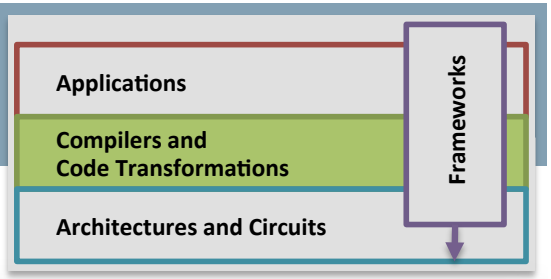
```
switch (get_quality_level(/*...*/)) {
    case 0: foo(...) break;
    case 1: foo_approx_Q1(...) break;
```

```
float foo (float p) {
    /* code without side effects */
}

float foo_wrapper(float p){
    float r;
    /* already in the table ? */
    if (lookup_table(p, &r)) return r;
    /* calling the original function */
    else r = foo(p);
    /* updating the table or not */
    update_table(p, r);
    return r;
}
```

```
(p && 0xffff0000)
```

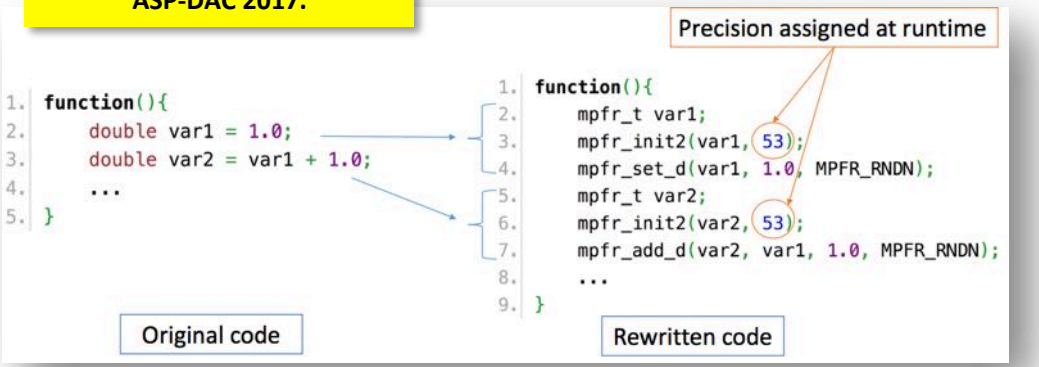
M. Alvarez et al "Fuzzy Memoization for Floating-Point Multimedia Applications" TCOM 2005.

Compiler Approaches

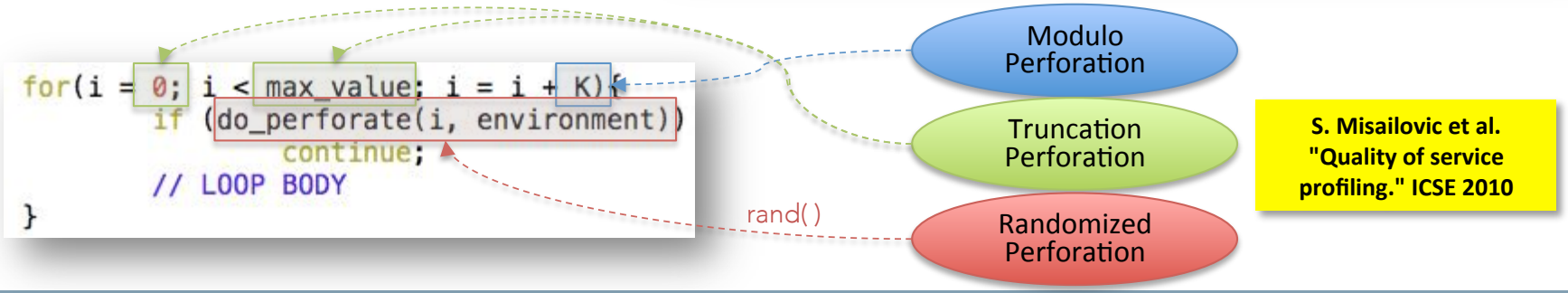


- Precision Scaling
  - FP64->FP32->FP16
  - Float2Int
  - Custom precision

N. Ho et al. "Efficient floating point precision tuning for approximate computing," ASP-DAC 2017.

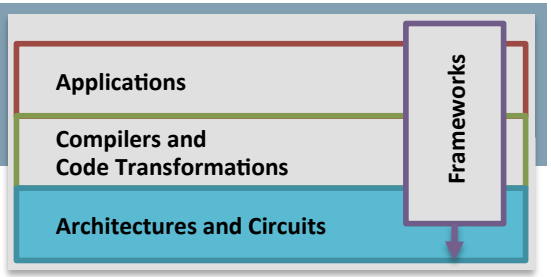


- Loop Perforation

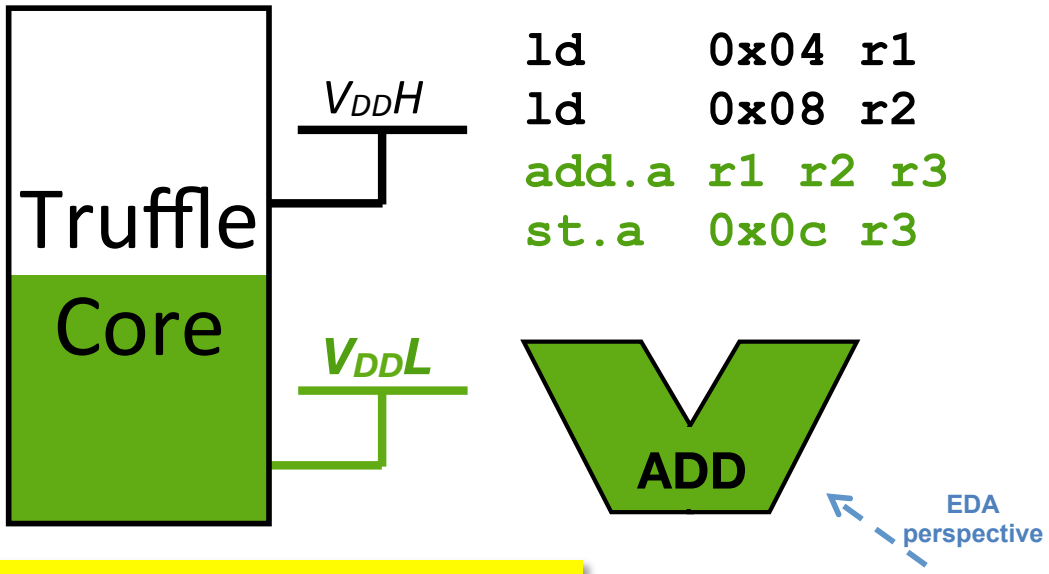


S. Misailovic et al. "Quality of service profiling." ICSE 2010

Approximate Hardware

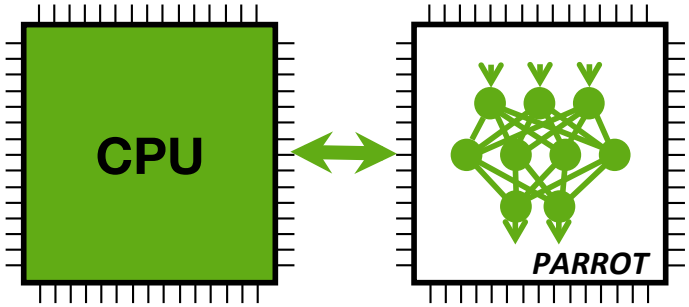


Very large literature … Dual VDD architectures, Approximate Adders/Multipliers, NPU etc…

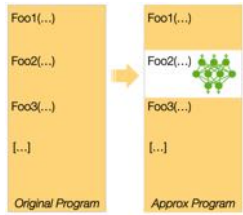


H. Esmaelizadeh et al. "Architecture support for disciplined approximate programming" ASPLOS 2012

H. Esmaelizadeh et al."Neural Acceleration for General-Purpose Approximate Programs" MICRO 2012

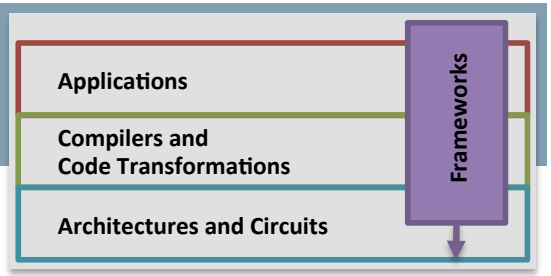


- Efficient hardware implementations
- Capable to mimic many ***APPROXIMABLE*** computations (after training)
- Fault tolerant



S. Venkataramani et al, "SALSA: systematic logic synthesis of approximate circuits" DAC 2012
I. Scarabottolo et al "Circuit Carving: A Methodology for the Design of Approximate Hardware" DATE 2018

Approximate Frameworks (1)



EnerJ

- Java extension with for Approximate Computation

Construct
@Approx, @Precise, @Top
endorse(e)
@Approximable
@Context
_APPROX

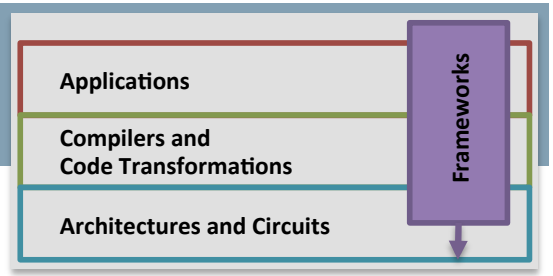
```
@approx   int a = ...;
@precise  int p = ...;
```

- Including Compiler and Runtime envisioning the usage of approximate HW

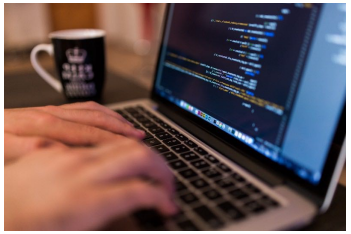
```
class FloatSet {
    @context float[] nums = ...;
    float mean() {
        calculate mean
    }
    @approx float mean_APPROX() {
        calculate mean of half
    }
}

@approx FloatSet someSet = ...;
someSet.mean();
```

A. Sampson et al. "EnerJ: Approximate Data Types for Safe and General Low-Power Computation" PLDI 2011

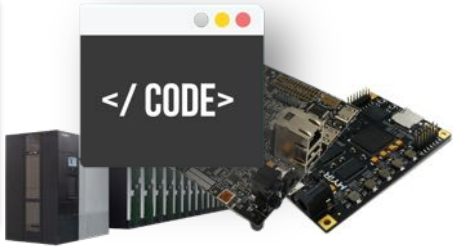
Approximate Frameworks (2)



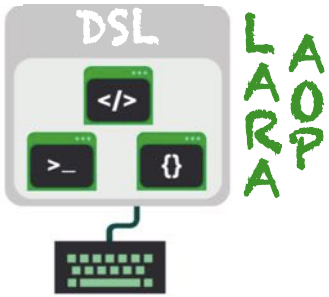
Functional Description



C/C++ w/ OpenMP,
MPI, OpenCL, Matlab



Extra-Functional Requirements and
Approximation aware-transformations
E.g. Adaptivity features, Tuning Knobs,
Code transformations,
Power/Performance constraints



Multiversioing aspect
including Modulo Perforation

```
aspectdef FunctionCloningAndModuloPerforation

input
    funct_name,
    modulo_value
end

/* The names of the new functions */
var approx_funct_name = func_name + '_approx';

/* Clone functions */
select function{func_name} end
apply
    exec Clone(approx_funct_name);
end

/* Perform modulo perforation on the outermost loop */
select function{approx_funct_name}.loop end
apply
    exec ModuloPerforation(modulo_value);
end
condition
    $loop.is_outermost
end

/* Change calls to the function */
select file.call{funct_name} end
apply
    var originalCode = $call.code + ';';
    var newCode = originalCode.split(funct_name).join(approx_funct_name);

    insert after %{
        switch (get_best_version(/*...*/)) {
            case 0: [[originalCode]] break;
            case 1: [[newCode]] break;
            default: [[originalCode]] break;
        }
    }%;
end

println('\n Modulo Perforation on the outermost loop done!');
end
```

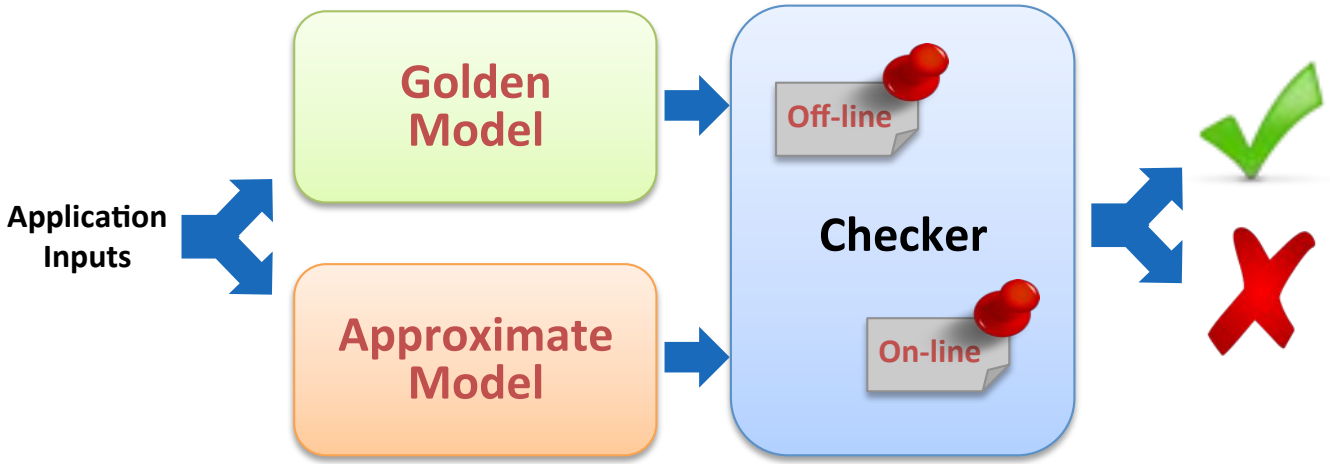
D. Gadioli et al. "SOCRATES: A seamless online compiler and system runtime autotuning framework for energy-aware applications" DATE 2017

ANTAREX

Silvano et al. "Autotuning and adaptivity in energy efficient HPC systems: the ANTAREX toolbox" Computing Frontiers 2018

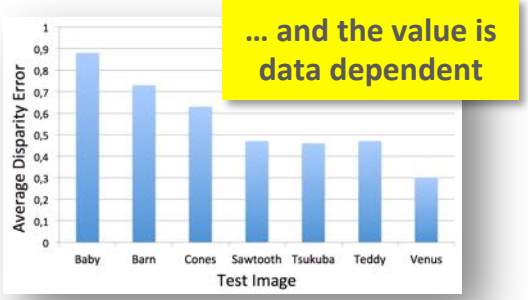
Controlling the Approximation

Need for monitoring the Quality of Result (QoR)

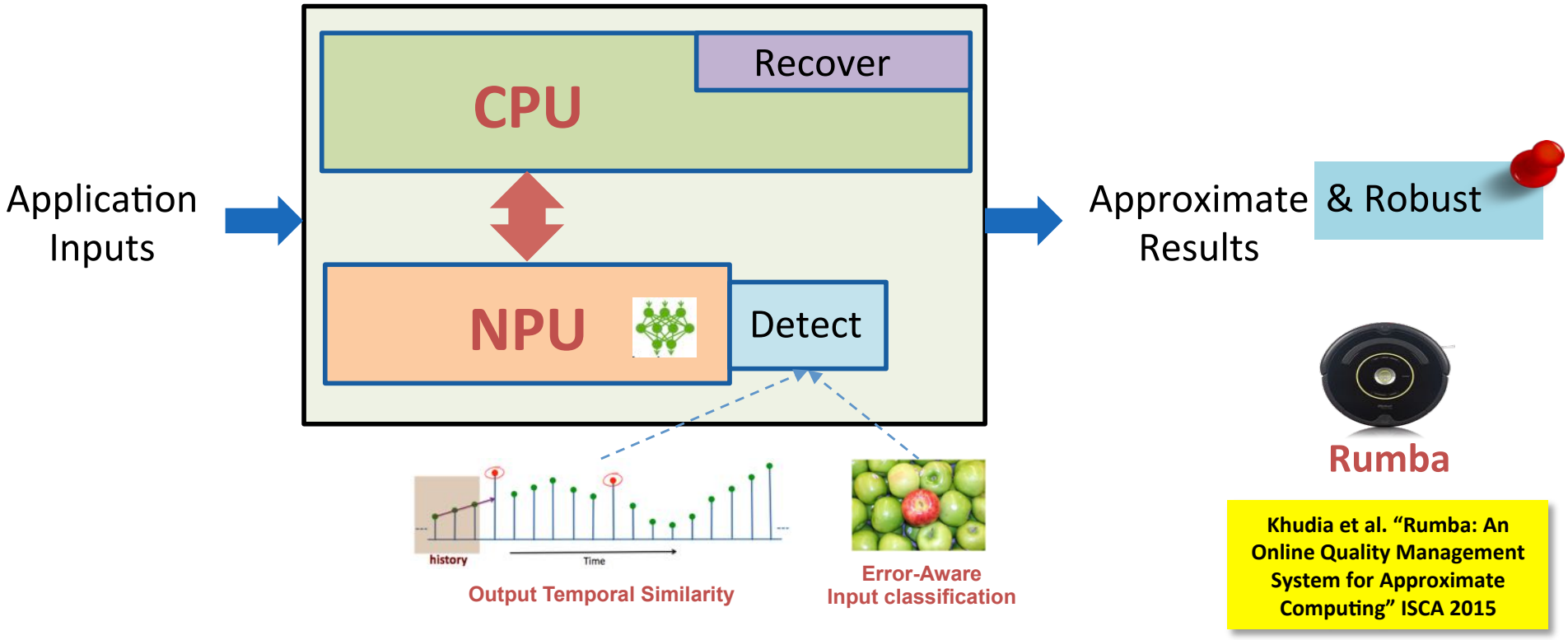


It has to be noted that establishing a suitable error measure is highly application dependent…

… and the value is data dependent



Online Monitoring: an Example

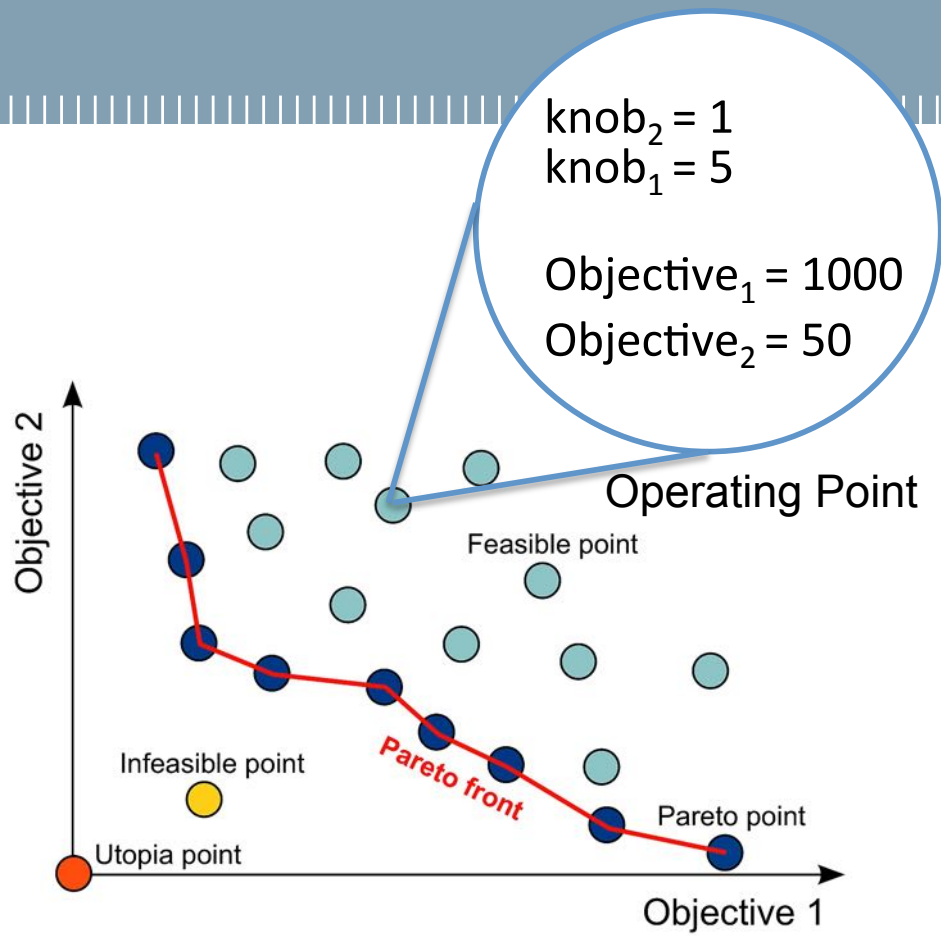


Off-line Quality Profiling

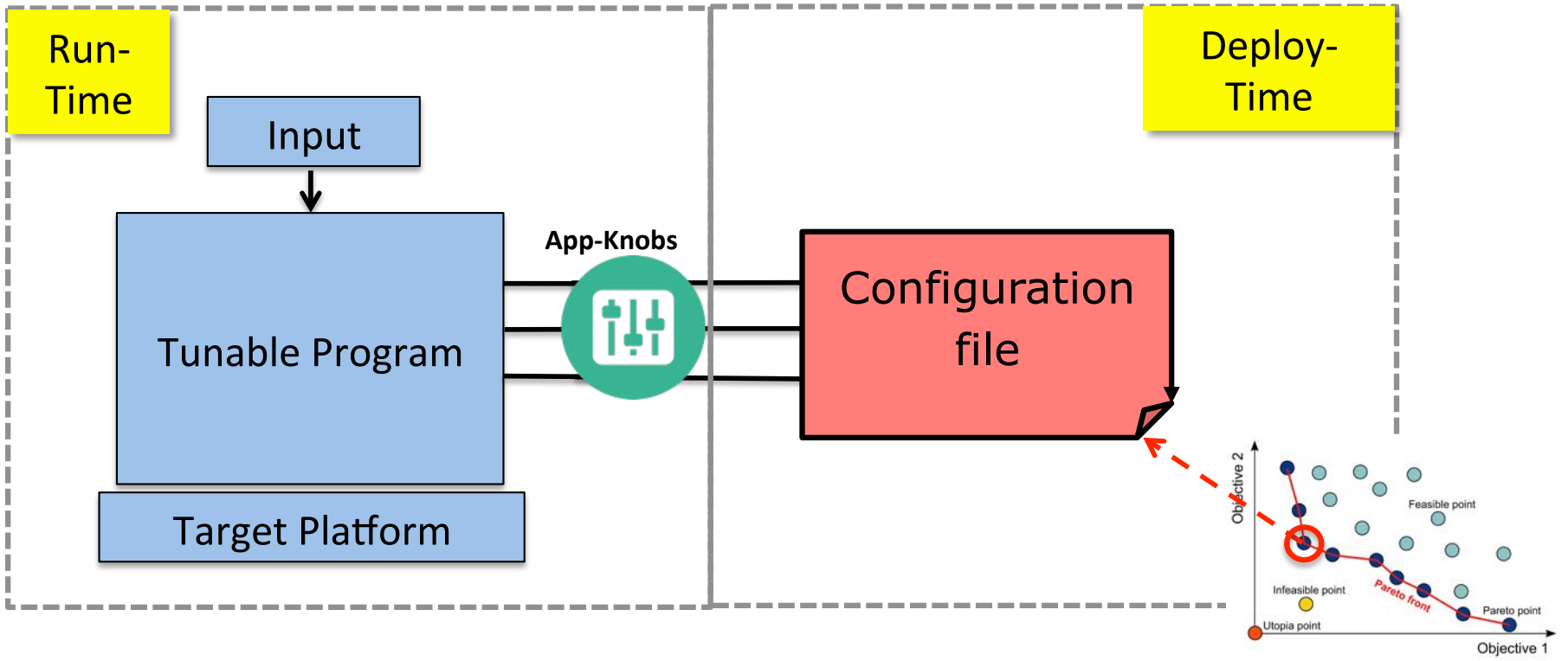
- Pure autotuning approaches
  - Error = f(x,i)

At **design time**:

1. **Instrument** the application
2. Perform a **Design Space Exploration**
3. **Store** the Pareto front

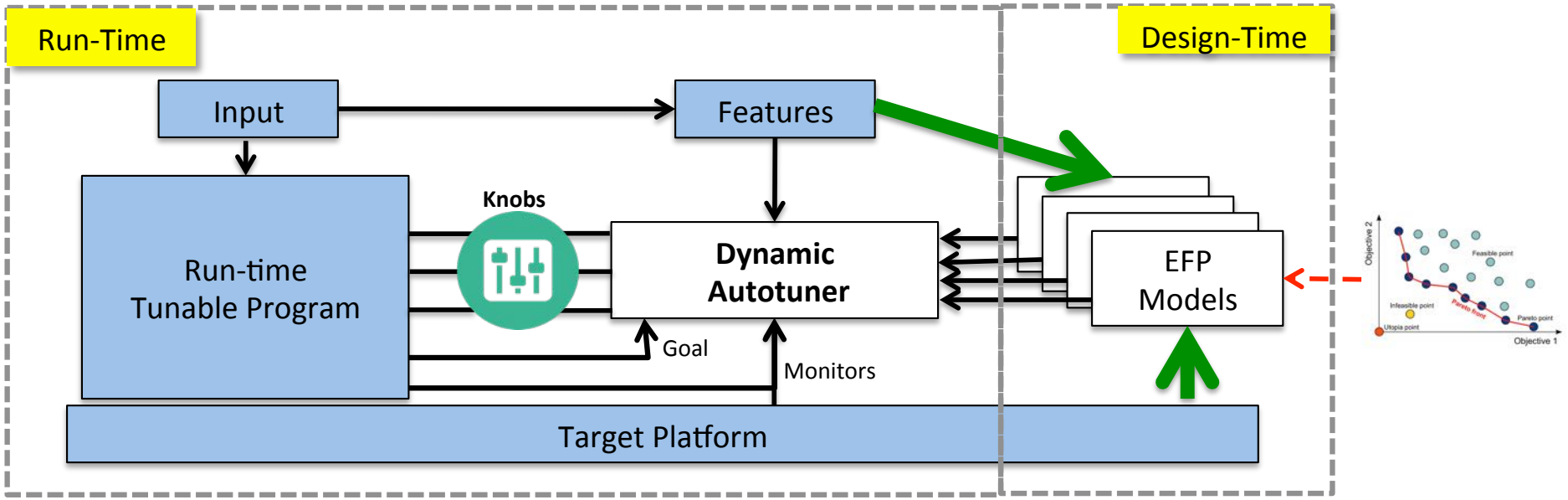


Configuring a Tunable Application



Off-line Quality Profiling (cont…)

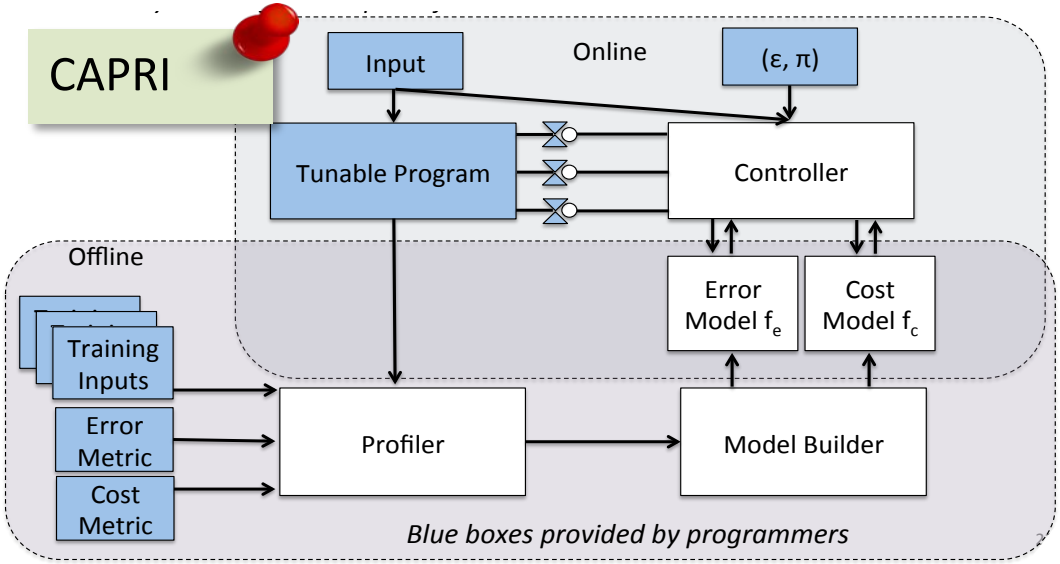
- Pure autotuning approaches
  - Error = f(x,i)
- Proactive and/or reactive approaches:
  - Error < K => x' : f(x',i) < K



Off-line Quality Profiling (cont…)

- Pure autotuning approaches
  - Error = f(x,i)
- Proactive and/or reactive approaches:
  - Error < K => x' : f(x',i) < K

CAPRI

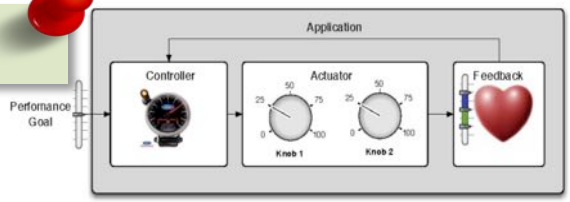


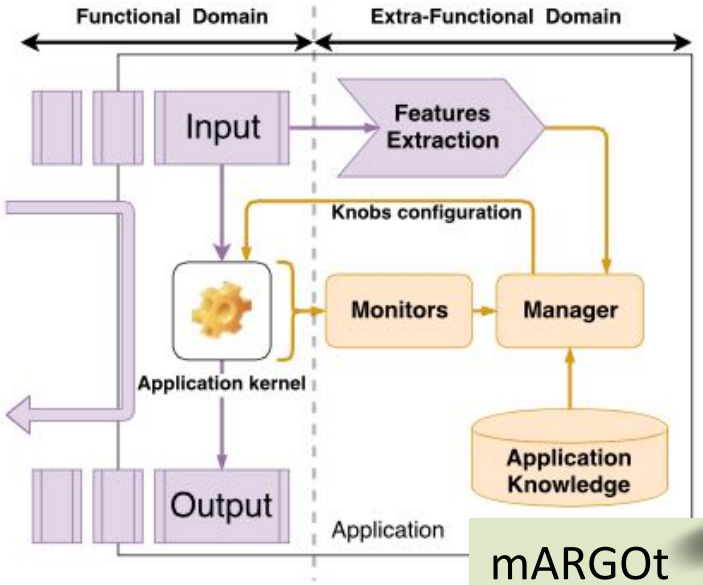
X. Sui et al. "Proactive Control of Approximate Programs" ASPLOS 2016

H. Hoffmann "JouleGuard: […]" SOSP 2015

H. Hoffmann et al. "Dynamic knobs for responsive power-aware computing." ASPLOS 2012

PowerDial





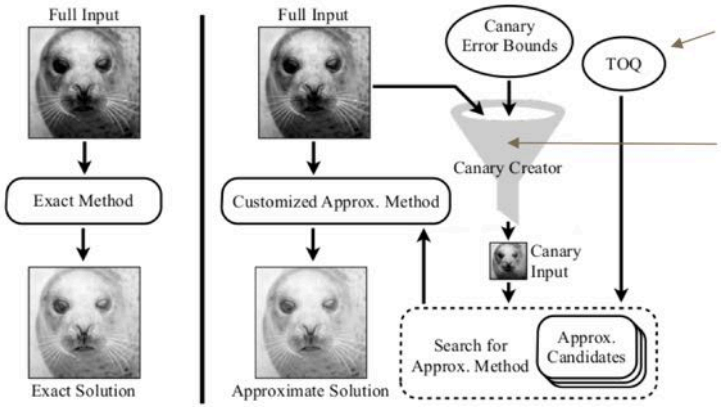
mARGOt

D. Gadioli et al. "Application autotuning to support runtime adaptivity in multicore architectures" SAMOS15
https://gitlab.com/margot_project/

Removing Offline Quality Profiling

IRA – Input Responsive Approximation

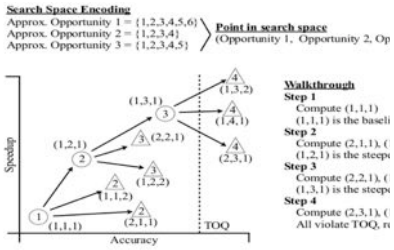
It leverages **canary inputs** to select the approximation level



Quality threshold

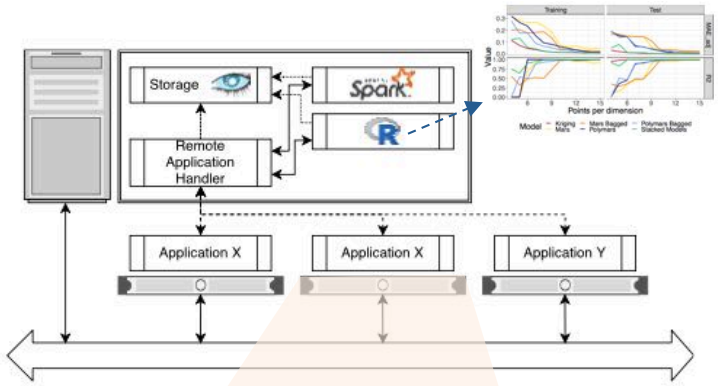
Smallest canary that has the same <property> as the full input:

- mean
- variance
- local homogeneity
- autocorrelation

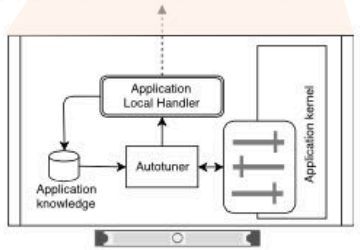


M. Laurenzano et al. "Input responsiveness: using canary inputs to dynamically steer approximation." PLDI 16.

mARGOt - AGORA



(a) Global structure of the distributed DSE framework



(b) Structure of an application instance

D. Gadioli et al. “mARGOt: a Dynamic Autotuning Framework Targeting Adaptivity and Controllable Approximation” SoftwareX

Conclusions… Why taking Care of Approximation?