Dynamic Application Autotuning for Approximate Computing

Gianluca Palermo – gianluca.palermo@polimi.it

Introduction

DYNAMIC AUTOTUNING

At Run-time according to different situation

Made Automatically

Capability to customize application knob values

It is a way to constantly improve performance/energy with low developer effort over a wide range of run-time situations

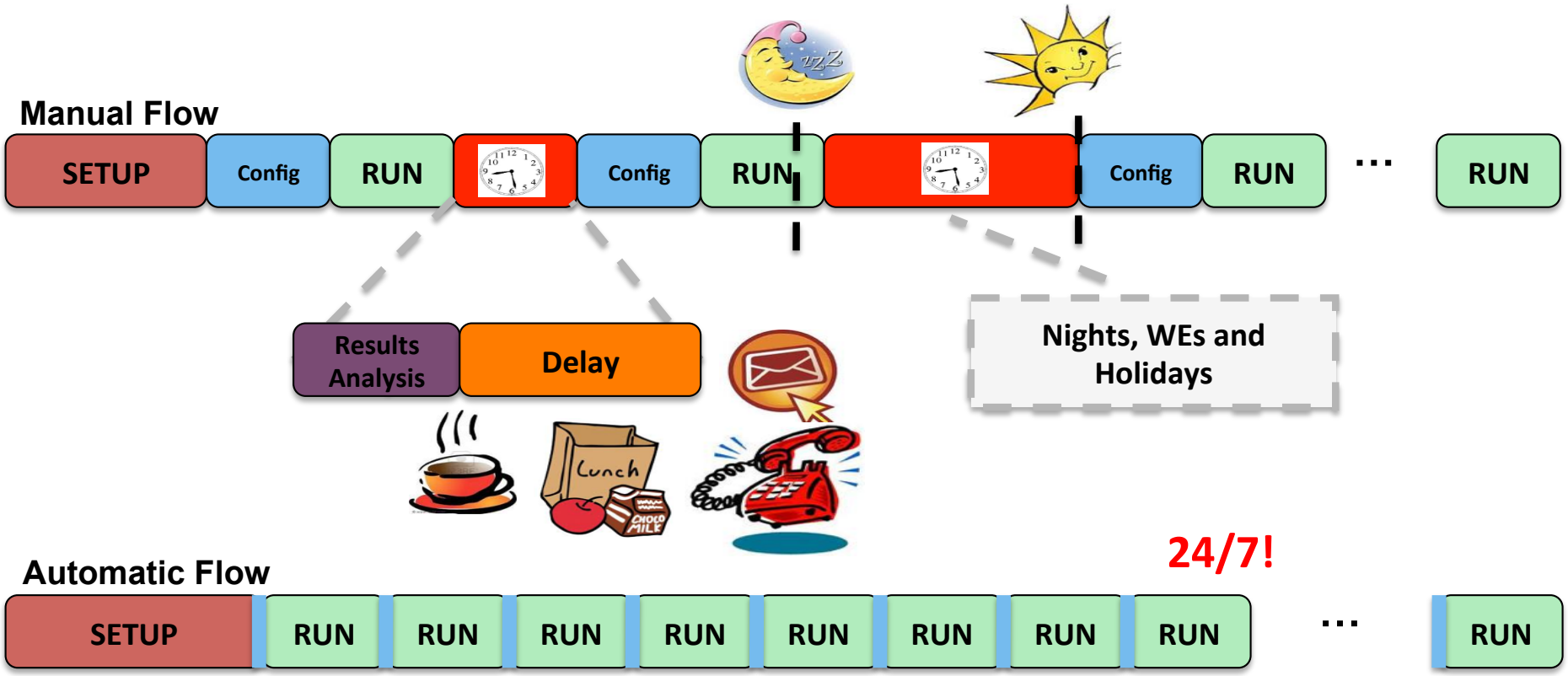
Application Tuning: Why?

- **Applications are parameterized** to facilitate the portability and improve the productivity (*best practice*)
- Large set of application ***knobs*** **with not so easy relations** with the extrafunctional properties
- Complex architectures including **frequent updates and novel evolutions**

Support for performance portability on wider range of today architectures

Possibility to easily exploit upcoming (or future) platforms

Automatic Vs Manual Tuning

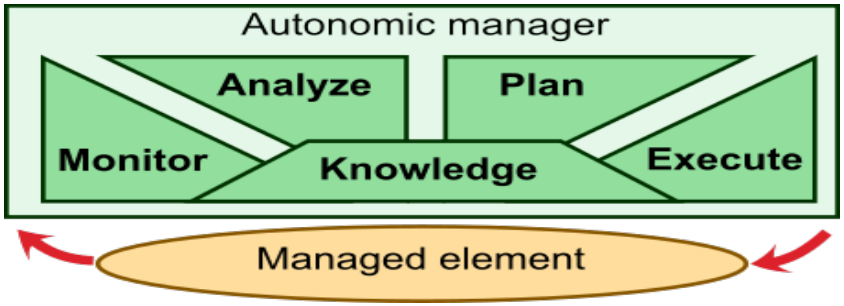


Dynamic: Why?

- The application **requirements** might **change** at runtime
  - E.g. Depending on the observed scene
- The extra-functional properties might be **input dependent**
  - Input features (e.g. input size, autocorrelation)
- The extra-functional properties might depend on the **system workload**
  - Shared computational resources
  - Core frequencies

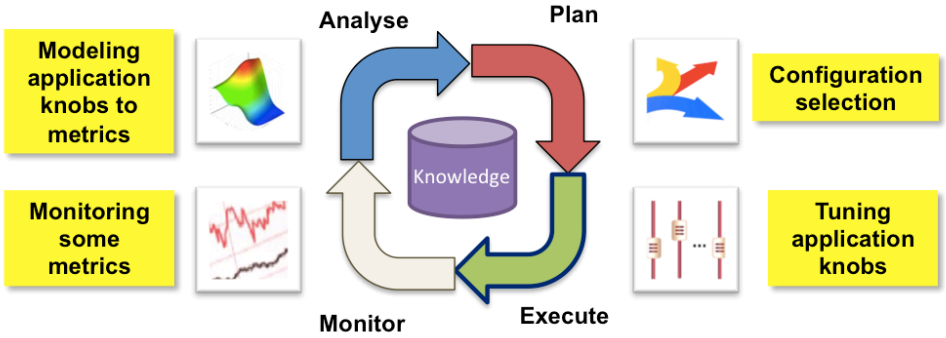
Dynamic: The autonomic computing vision

Each application is considered an **autonomic agent**, which is capable of self-management.

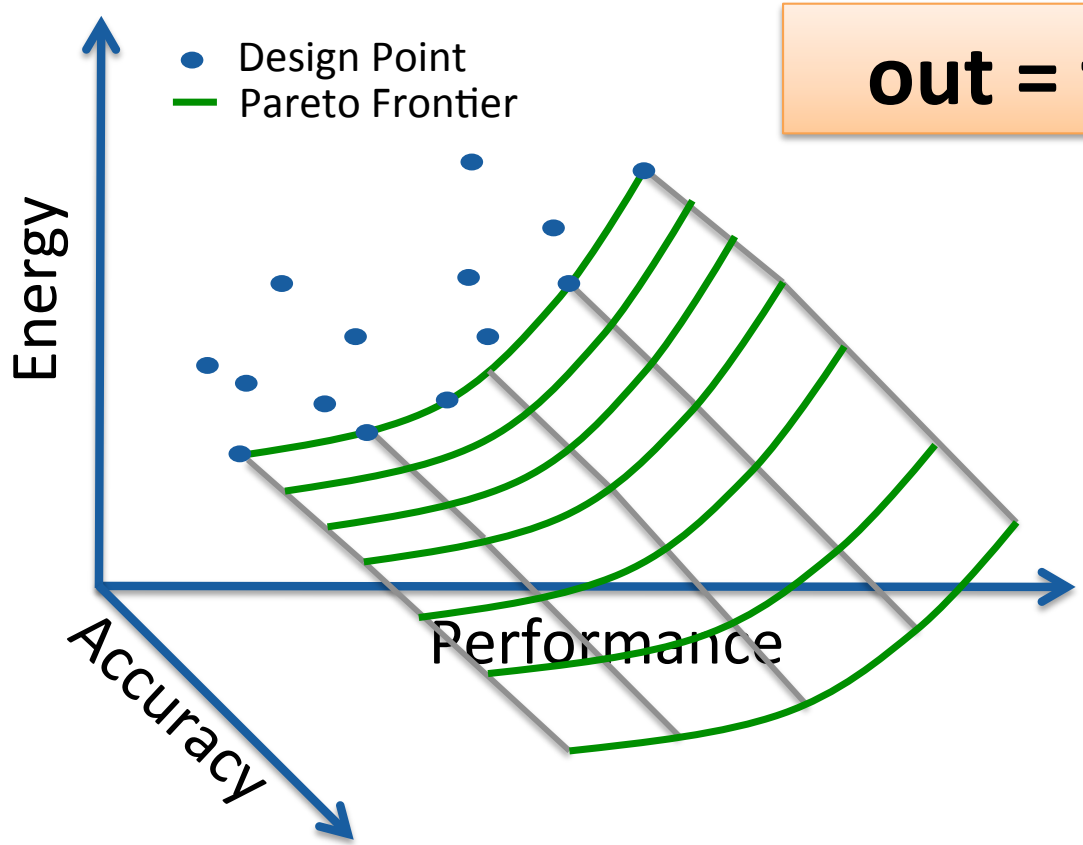


Kephart, Jeffrey O., and David M. Chess. "The vision of autonomic computing." *Computer* 36.1 (2003)

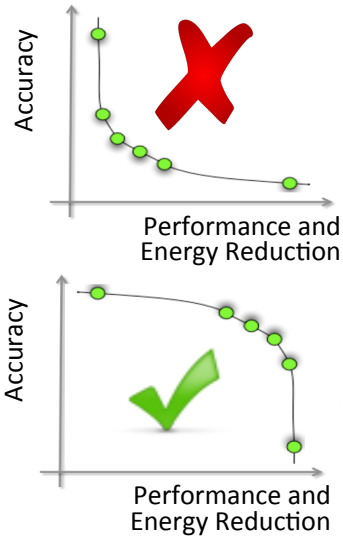
- self-configuration
- self-healing
- self-protection
- **self-optimization**



Multi-objective optimization problem



$$out = f(input, k_1, k_2, \ldots, k_n)$$



Approximate Computing Applications

- Approximation adds complexity
  - Most of a code base can't 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…

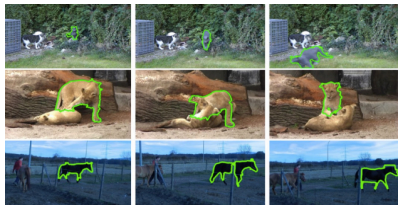
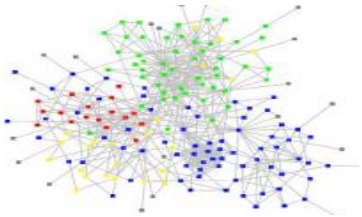


Image Processing

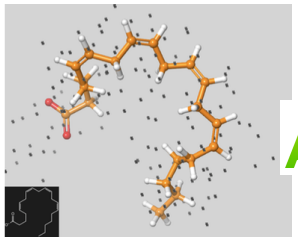
Robotics

Big Data Analytics



Graph Analytics

Multimedia



Drug Discovery

ANTAREX

Traffic Prediction

… where 100% of accuracy not always required

HOWEVER …

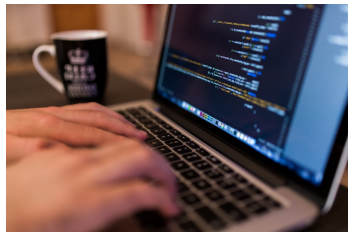
… we have to pay attention

No Traffic

No Intrusion

Extra-Functional Properties

Functional Description



What to do…

Extra-Functional Properties

How It is done…

Extra-functional properties

$$out = f(input, k_1, k_2, \ldots, k_n)$$

EFP of the output:

- Size (i.e. of the file)
- Accuracy (i.e. PSNR)
- Dimension (i.e. Resolution)

EFP of the elaboration:

- Throughput/Latency
- Power/Energy consumption
- Resource utilization

Application level Knobs

- Algorithm Selection
- Application Parameters
- Parallelism
- Code Perforation

Compiler level Knobs

- Flag selections
- Loop transformations
- Float-to-Fixed point Conversion
- Parallel Pattern Replacement

Target problem definition

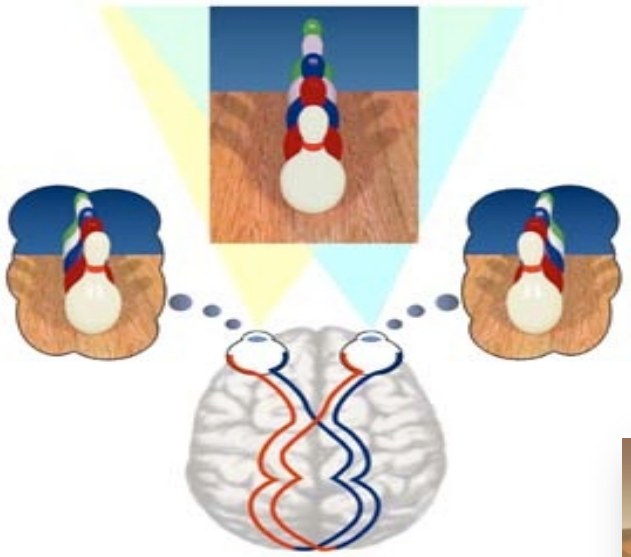
$$out = f(input, k_1, k_2, \ldots, k_n)$$

Which values should we use?

The best ones? What do you mean with "best"?
One single configuration?

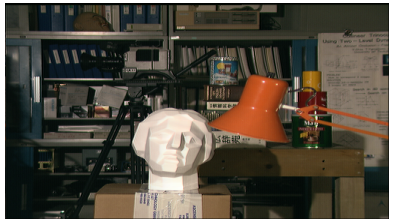


Let's have an example…

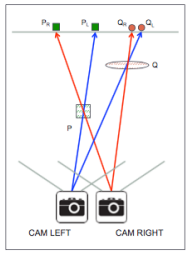


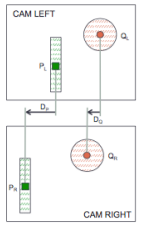
2 eyes = 3 dimensions

Left camera



Right camera





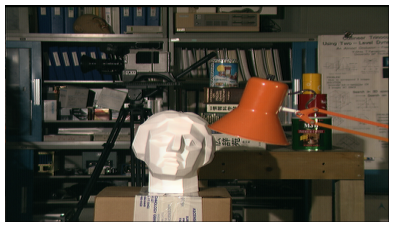
Reference disparity

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

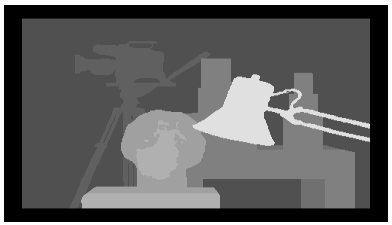
QoR Disparity Error

Left camera



Right camera

5 Application Knobs*

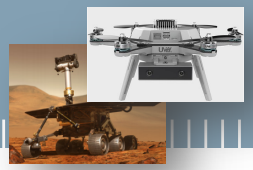


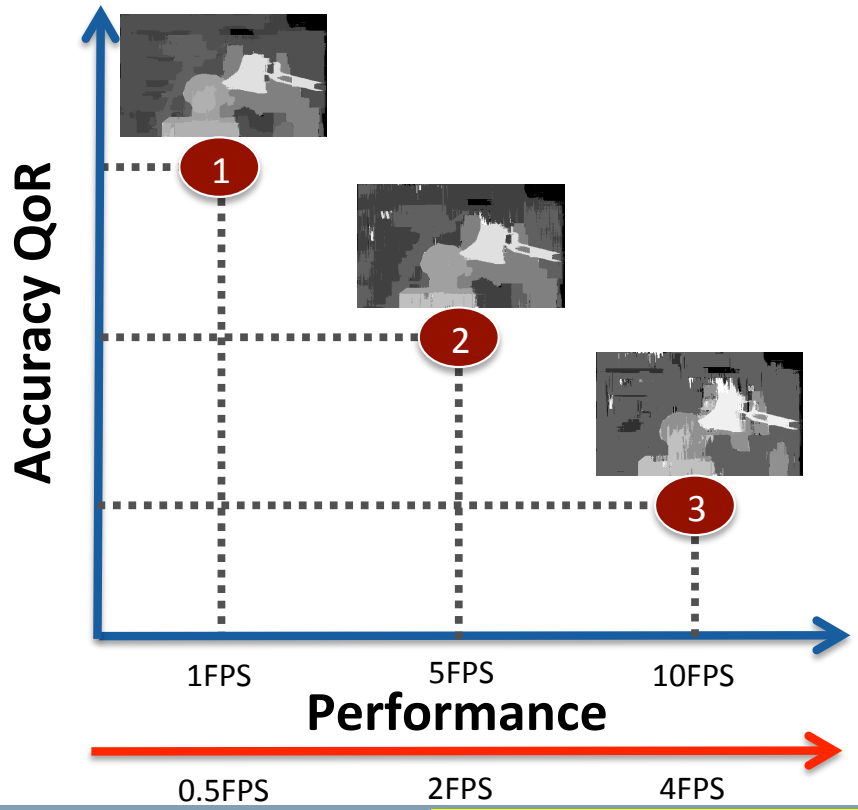
Reference disparity

* 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

D. Palossi "Self-Sustainability in Nano Unmanned Aerial Vehicles: A Blimp Case Study" CF 2017

Autotuning Problem

The definition of "best" should be expressed as a **multi-objective constrained optimization problem**

$$out = f(input, k_1, k_2, \ldots, k_n)$$

Select **K1…Kn** to optimize **EFP(out/f)** subject to **EFP(out/f) constraints**, and considering **input characteristics**

Autotuning SoTa Overview

Applications Autotuning: State of the Art

Classification of autotuning approaches:

- Type of the decision
- How they are packaged into the application
- Approach to the decision
- When to apply

P. Balaprakash, J. Dongarra,T. Gambling, M. Hall, J.K. Hollingsworth, B. Norris, R. Vuduc – Proceedings of IEEE

Approach	Definition	Examples
How Packaged		
Library	A library is tuned for different architectures and input data sets.	ATLAS, OSKI, MKL, FFTW, SPIRAL
Compiler-directed	The compiler generates multiple implementations and selects among them.	CHiLL, Orio, PetaBricks
Application-level	The programmer expresses parameters and code variants to select among.	ActiveHarmony, Orio, OpenTuner
Approach to Selection		
Model-based	A performance or ML model guides selection.	Performance model, Classifier model
Search-based	Empirical search to identify selection.	Heuristic search
Hybrid	A model prunes search space, followed by search.	Prune undesirable areas of search space
Types of Decisions		
Algorithm selection	Select among alternative algorithms that are functionally equivalent.	PETSc, Trilinos, Lighthouse
Code variant selection	Select among fundamentally different implementations of an algorithm.	Data layout/representation
Parallelization	Select among different parallelization strategies.	SIMD(y/n), OpenMP vs. CUDA, multi-level
Code transformation	Select among different code transformation sequences.	Tile (y/n), Fusion (y/n), Wavefront (y/n)
Parameter tuning	Adjust context-specific parameters.	# of ranks/threads, Tile size, Unroll factor
When to Apply		
During porting	Library/application migrates to new architecture.	Adjust data layouts and parameters
Offline as needed	Search or training phase explores mapping selection.	Code targets new context
Online, incrementally	Autotuning decisions improved over application runs.	Expand search space for context
Dynamic, runtime	Consult model to select appropriate implementation.	Based on input features
Integration into Application		
Tuned code	Tuned code is inserted into the application.	Architecture-specific code
Selectable code	Multiple implementations and a selector.	Input-dependent selection
Dynamic	Code is dynamically generated.	Too many variants
Full tuning integration	Tuning occurs as part of build.	Compact code and forward scalability
Runtime Measurement		
Instrumentation	Gather standard measurements transparently to the user.	Dyninst, TAU, HPCToolkit
Semantic annotation	Tools or user direct instrumentation.	Caliper
Performance database	Collect measurements in a database.	TAUdb

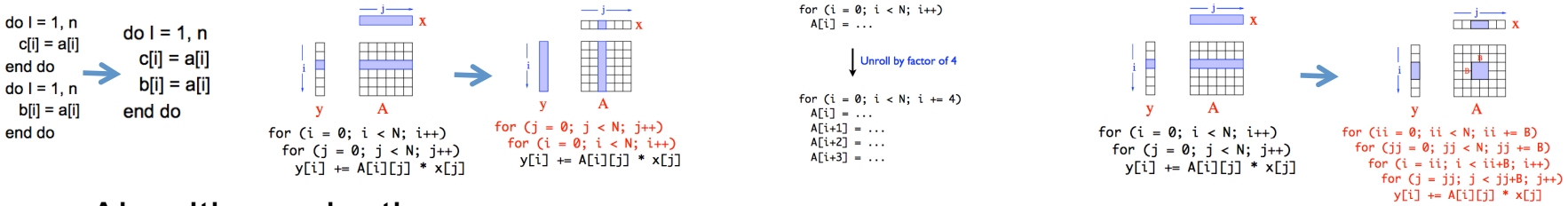
Type of decisions

- Compiler options

A. Ashouri, W. Killian, J. Cavazos, G. Palermo, and C. Silvano. A Survey on Compiler Autotuning using Machine Learning. ACM Computing Surveys (CSUR)

- Code transformations and data layout:
  - I.e. Fusion, Interchange, Unrolling, Tiling

D. Bacon, S. Graham, O. Sharp. Code Transformations for High-Performance Computing. ACM Computing Surveys (CSUR)



- Algorithm selection
  - Select among different implementations of functionally equivalent versions of the algorithm (i.e. for sorting: bubble, merge, quick, heap, radix, bucket…)

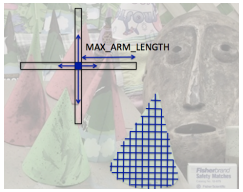
J. Ansel et al. "PETABRICKS: a language and compiler for algorithmic choice" PLDI09

- Parallelization:
  - Parallel patterns, #workers, OmpThread, multi-level parallelism parameters
- …Approximation parameters

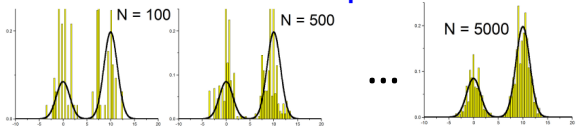
Type of decisions: Application-level approximations

- *Application parameters*

Video Resolution



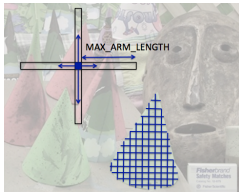
#MC Samples



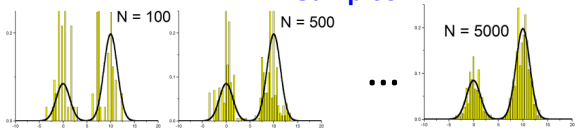
Type of decisions: Application-level approximations

- *Application parameters*

Video Resolution



#MC Samples

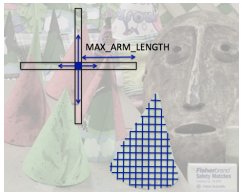


- *Task skipping*

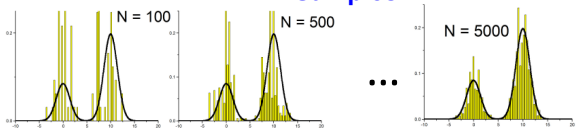
Type of decisions: Application-level approximations

- *Application parameters*

Video Resolution



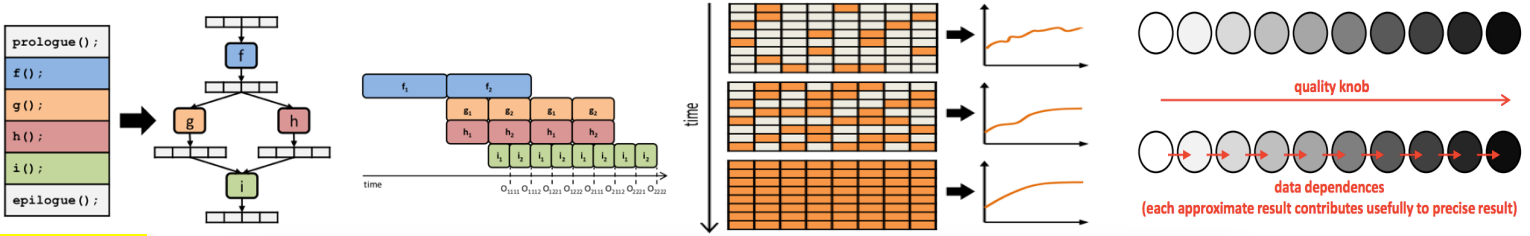
#MC Samples



- *Task skipping*

Computation model that represents an approximate application as a pipeline

M.J. San et al "The anytime automaton." ACM SIGARCH Computer Architecture News. 2016.

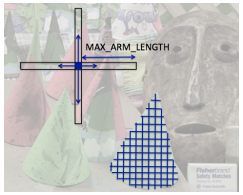


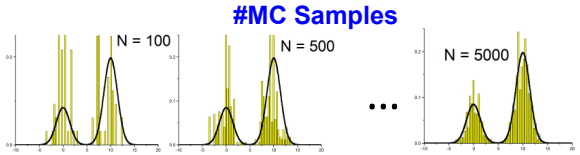
V. Vassilliadis et al " Exploiting Significance of Computations for Energy-Constrained Approximate Computing" IJPP 2016.

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

Type of decisions: Application-level approximations

- *Application parameters*





- *Task skipping*

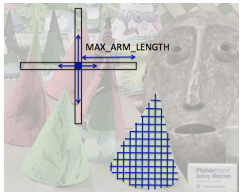
- *Multiversioning*
  - Considering different performance/accuracy trade-offs

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

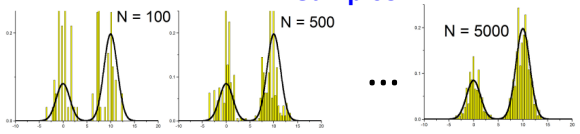
Type of decisions: Application-level approximations

- *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

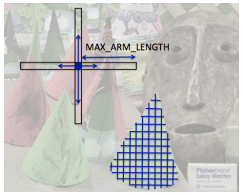
B. Woongki et al "Green: a framework for supporting energy-conscious programming using controlled approximation." PLDI 2010

```
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 }
```

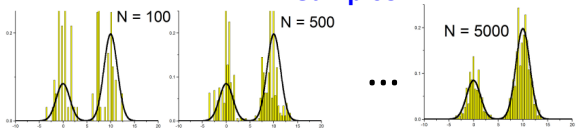
Type of decisions: Application-level approximations

- *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.

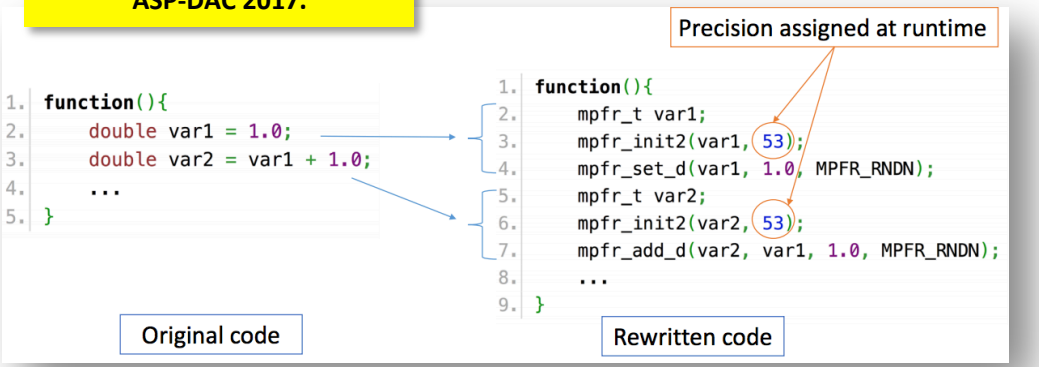
Type of decisions: Compiler approaches

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

Type of decisions: Compiler approaches

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

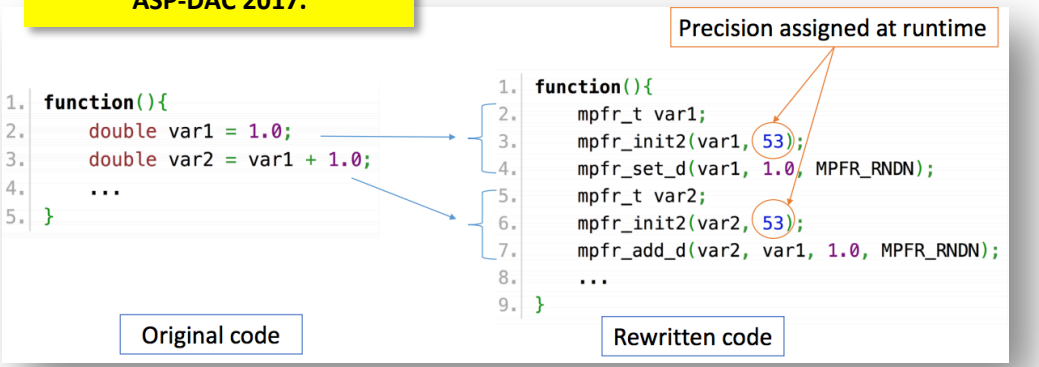
N. Ho et al. "Efficient floating point precision tuning for approximate computing," ASP-DAC 2017.



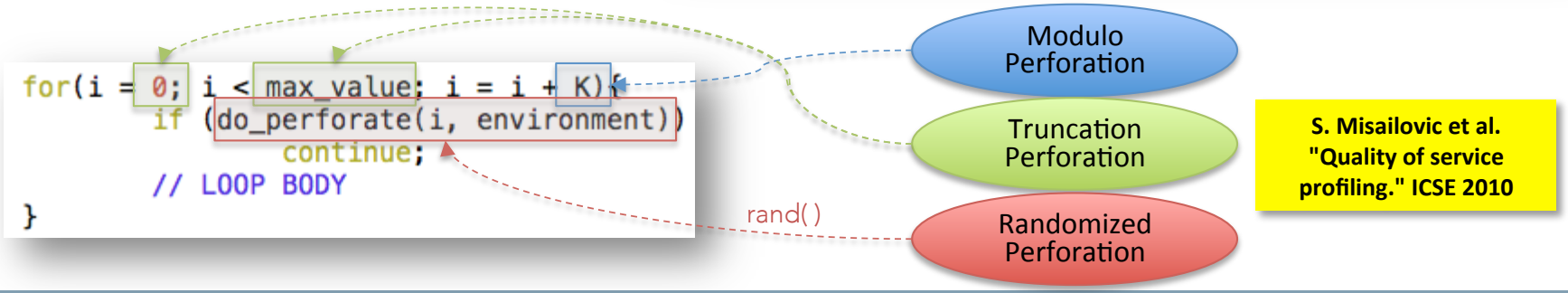
Type of decisions: Compiler approaches

- 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

How they are packaged

- Library-based
  - Mainly Performance-critical routines
  - Parameterization and tuning engine are not exposed
- Compiler-Level
  - The code transformations and variants are injected by the compiler/code generation framework
  - The tuning engine can be embedded into the compiler or parameters are exposed for tuning
- Application-Level
  - The programmer parameterizes the application
  - External tuning frameworks

R. Whaley, J. Dongarra, "ATLAS: Automatically tuned linear algebra software," SC98

M. Frigo, S. Johnson, "The design and implementation of FFTW3," Proc. of IEEE 2005

R. Vuduc, et al. "OSKI: A library of automatically tuned sparse matrix kernels," Journal. of Physics, 2005

M. Puschel, et al. "SPIRAL: A generator for platform-adapted libraries of signal processing algorithms," JHPCA04

A. Tiwari et al. "CHiLL: scalable auto-tuning framework for compiler optimization," IPDPS09

B. Norris, et al. "Annotation-based empirical performance tuning using ORIO," IPDPS09

J. Ansel et al. "PETABRICKS: a language and compiler for algorithmic choice" PLDI09

D. Gadioli, et al."SOCRATES - A seamless online compiler and system runtime autotuning framework" DATE18

J. Ansel et al."OPENTUNER: An extensible framework for program autotuning," PACT14

D. Gadioli et al. "mARGOt: a Dynamic Autotuning Framework Targeting Adaptivity and Controllable Approximation" TCOM19

Approach to the decision

- Search-Based
  - Complete enumeration (small configuration space)
  - Heuristics used to navigate the configuration space
    - Genetic algorithms, simulated annealing, particle swarm optimization
  - Mix of global and local searches

- Model-based
  - Pure analytic model of the performance
    - Requires a huge knowledge of the target problem
  - Source code based model
    - Based on static code analysis
    - Requires to generate the modified code
  - Training-based
    - Subset of configurations are evaluated and a predictive model is built using Machine Learning techniques

- Hybrid
  - Sequence of Model- and Search-based techniques

A. Rasch, et al. "ATF: A Generic Auto-Tuning Framework," HPCC17

R. Miceli et al., "AUTOTUNE A plugin-driven approach to the automatic tuning of parallel applications" 2012

B. Norris, et al. "… ORIO," IPDPS09

R. Whaley, J. Dongarra, "ATLAS…" SC98

J. Ansel et al."OPENTUNER … ," PACT14

G. Mariani, et al. "Scaling Properties of Parallel Applications to Exascale." IJPP16

P. Balaprakash et al. "Can search algorithms save large-scale automatic performance tuning?" ICCS11

J. Bergstra, et al. "Machine learning for predictive auto-tuning with boosted regression trees," INPAR12

T. Martinovic, et al. "On-line Application Autotuning Exploiting Ensemble Models" arXiv19

G. Palermo, et al. "ReSPIR: A Response Surface-based Pareto Iterative Refinement for DSE". TCAD09

G. Mariani, et al. "OSCAR: An Optimization Methodology Exploiting Spatial Correlation" TCAD12

M. Zuluaga,"SMART design space sampling to predict Pareto-optimal solutions." LCTES12

When to apply

- During Porting
  - Mainly for libraries
  - The code is tuned to adjust data layout and parameters to the new architecture

R. Whaley, J. Dongarra, "ATLAS…" SC98

M. Frigo, et al. "… FFTW3," Proc. of IEEE 2005

R. Vuduc, et al. "OSKI: …" Journal. of Physics, 2005

M. Puschel, et al. "SPIRAL: …" JHPCA04

- Offline
  - The configuration space is profiled and the optimal configuration is applied

J. Ansel et al."OPENTUNER: An extensible framework for program autotuning," PACT14

B. Norris, et al. "Annotation-based empirical performance tuning using ORIO," IPDPS09

J. Ansel et al. "PETABRICKS: a language and compiler for algorithmic choice" PLDI09

- Dynamically at RunTime
  - Autotuning decision can be improved during the application runs
  - Possibility to use a larger production environment
  - Decisions are taken considering input data characteristics and execution context

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

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

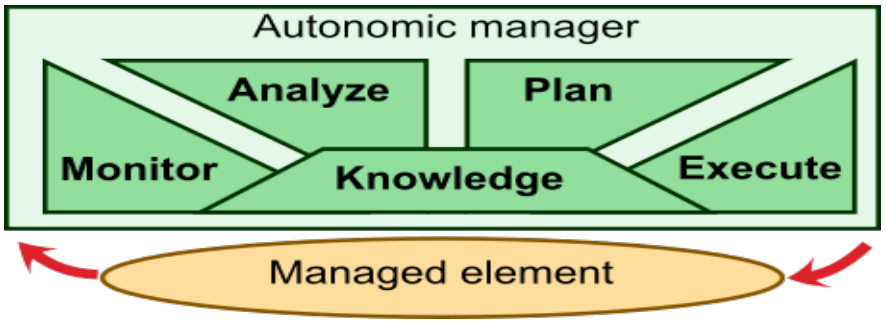
D. Gadioli et al. "mARGOt: a Dynamic Autotuning Framework …" TCOM19

Off-line Profiling and On-Line Monitoring

The Knowledge is the key

The Knowledge of the system behaviour is the key point for any decision

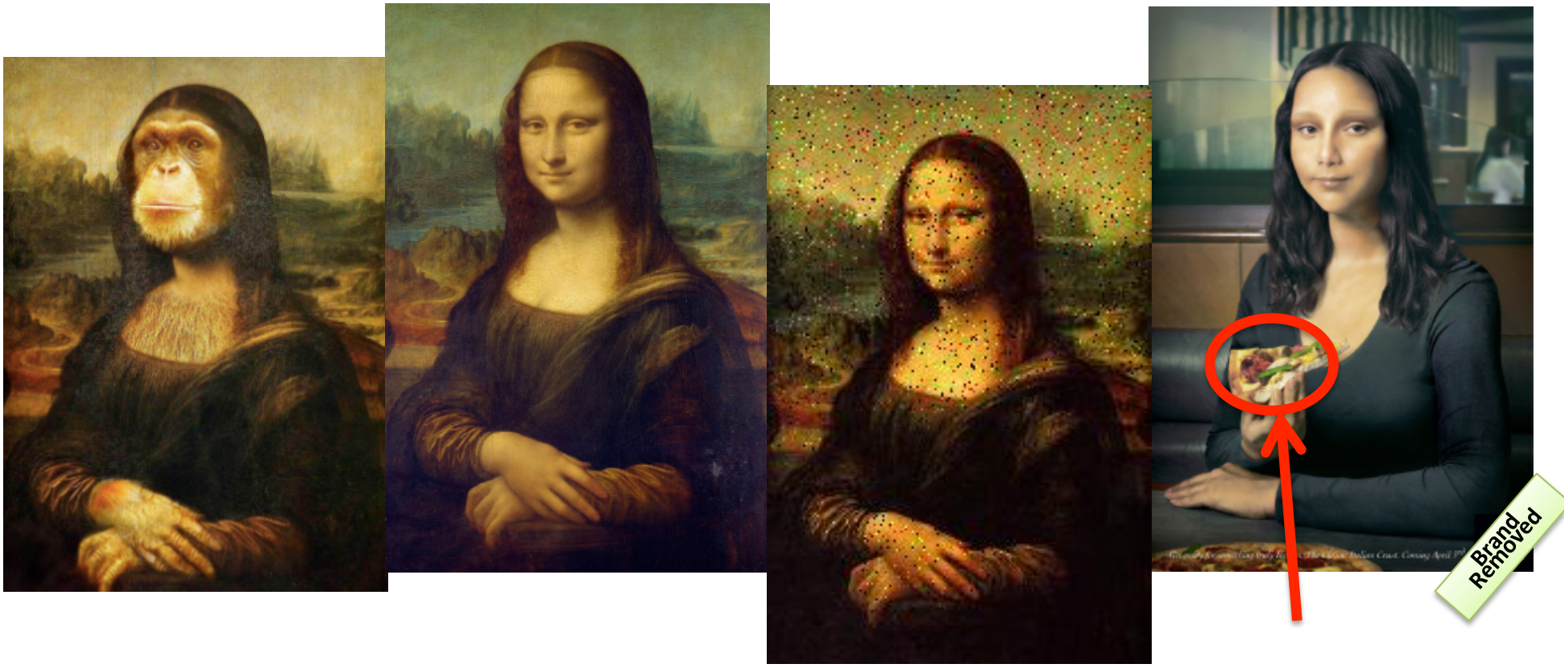
- Offline case: How to configure?
- Online case: How to re-configure?



Energy/Power/ Latency are easily measurable…

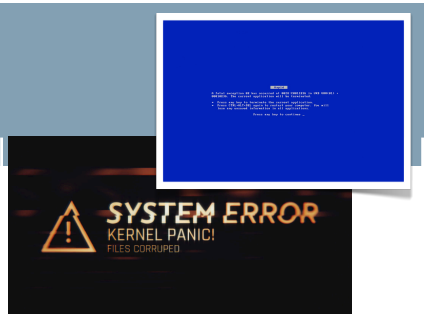
What about approximation effects?!

Why taking care of approximation effects?

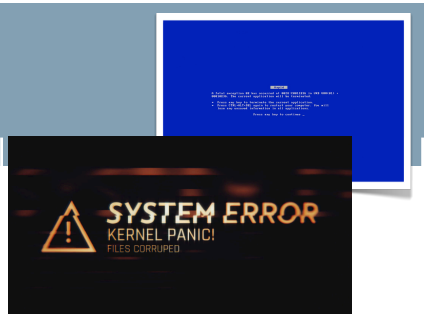


Controlling the Approximation

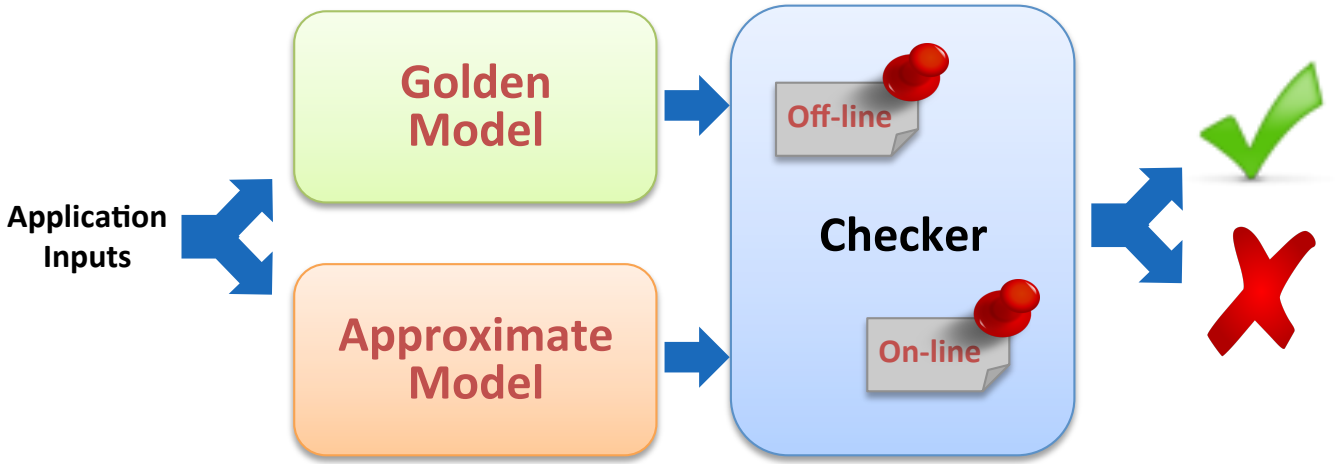
Need for monitoring/profiling the Quality of Result (QoR)



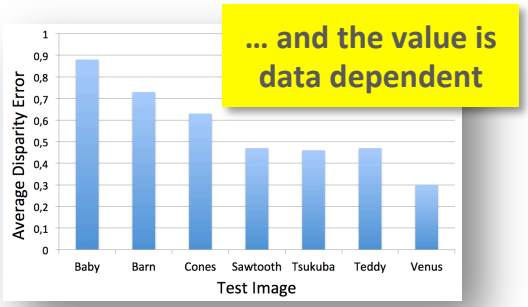
Controlling the Approximation



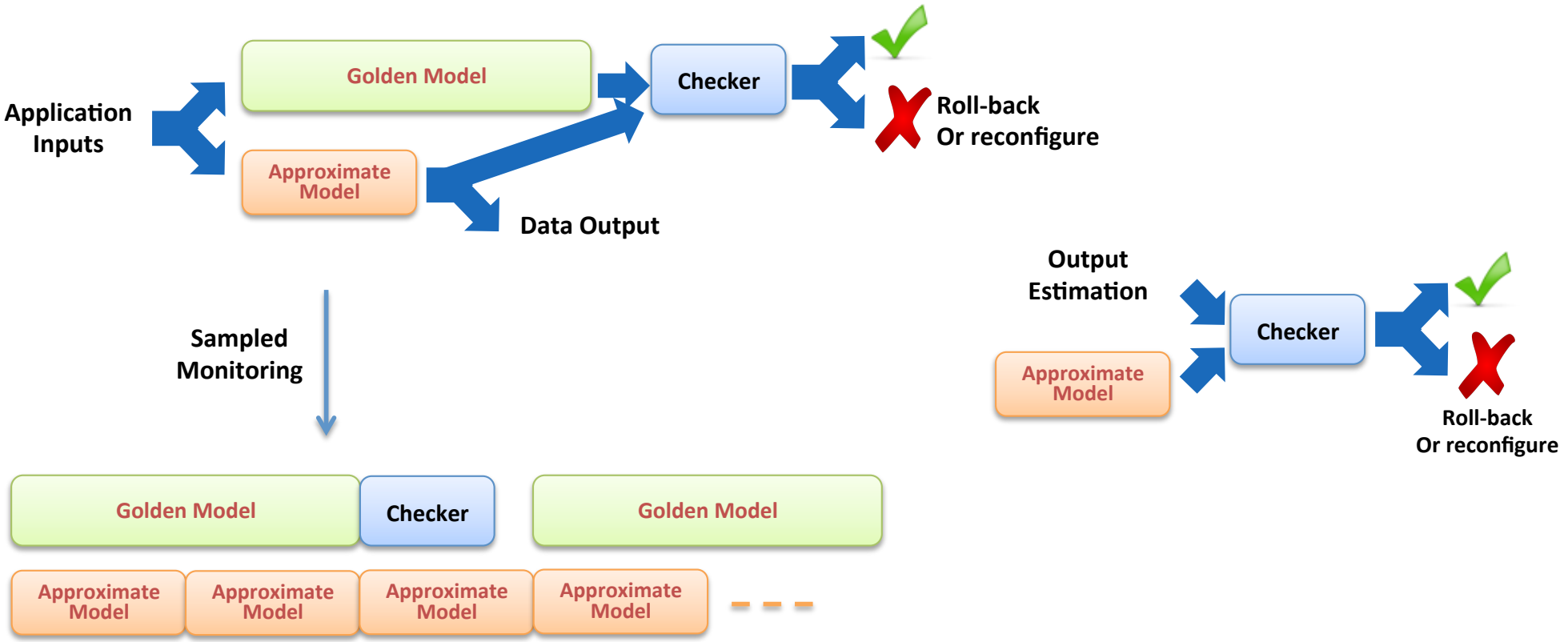
Need for monitoring/profiling the Quality of Result (QoR)



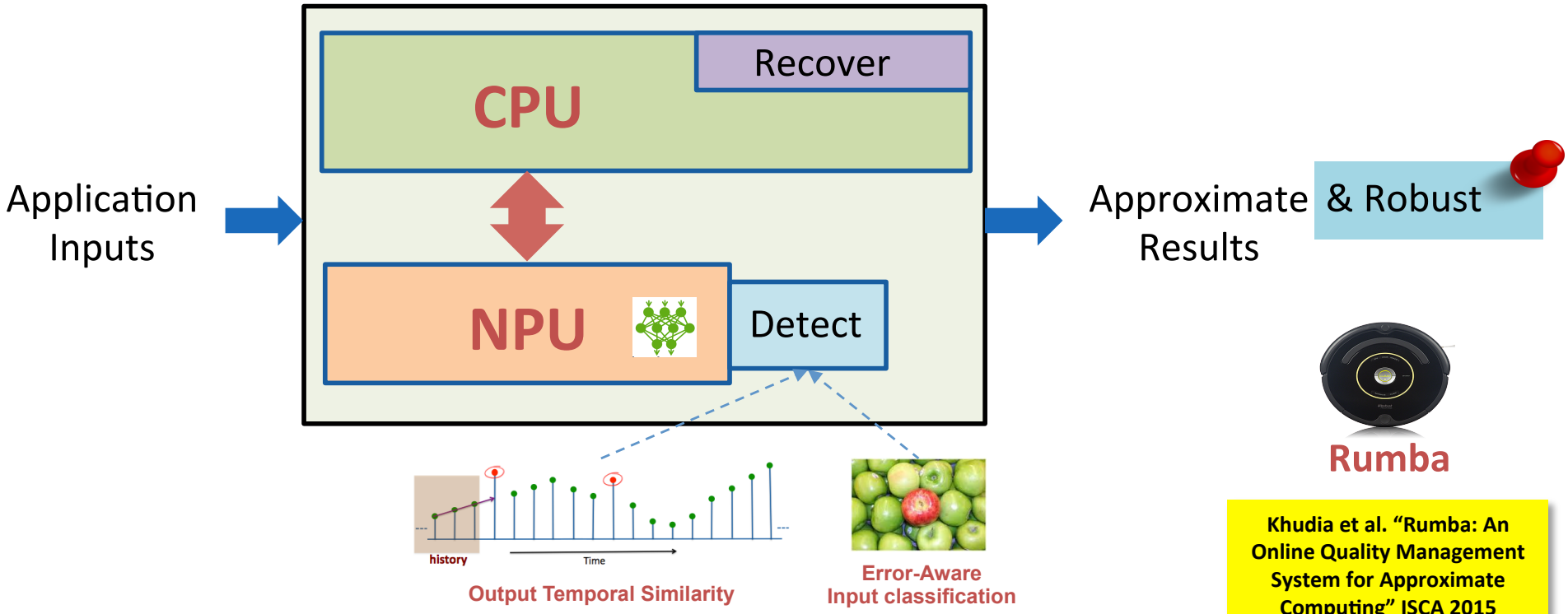
It has to be noted that establishing a suitable error measure is highly application dependent…



OnLine Monitoring



Online Monitoring: an example



Off-line Quality Profiling

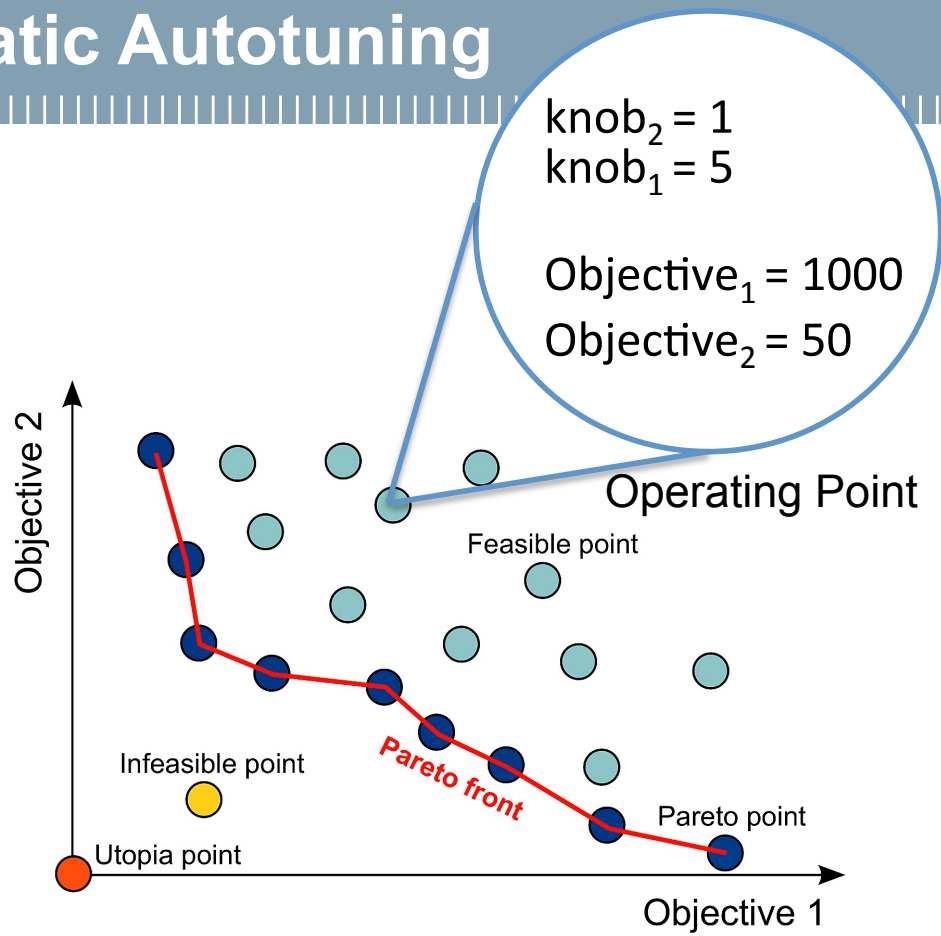
- Pure autotuning approaches
  - Error = f(x,i)

Off-line Quality Profiling – Static Autotuning

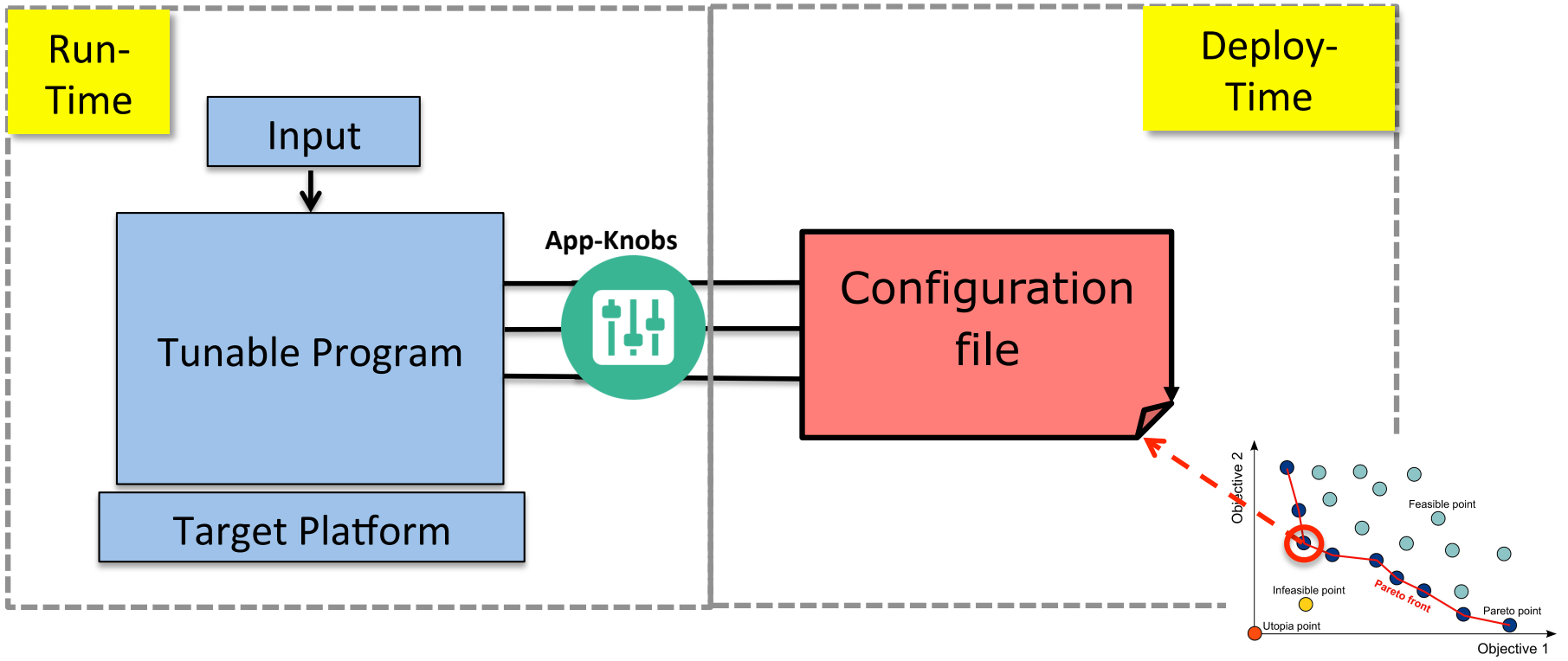
- Pure autotuning approaches
  - Error = f(x,i)

At **design time**:

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

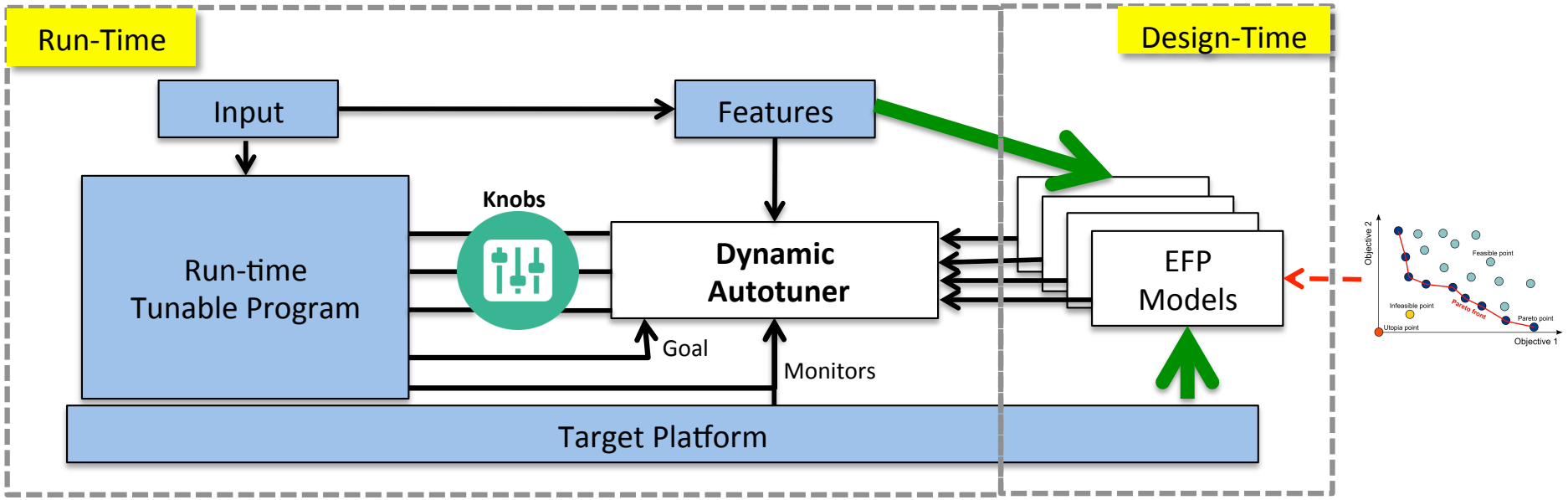


Configuring a Tunable Application



Off-line Quality Profiling – Dynamic Autotuning

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

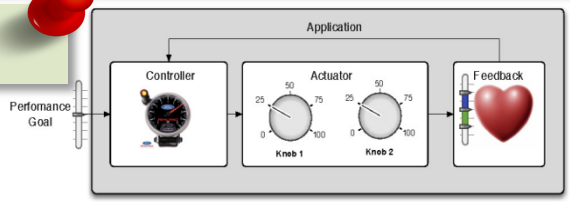


Off-line Quality Profiling – Dynamic Autotuning

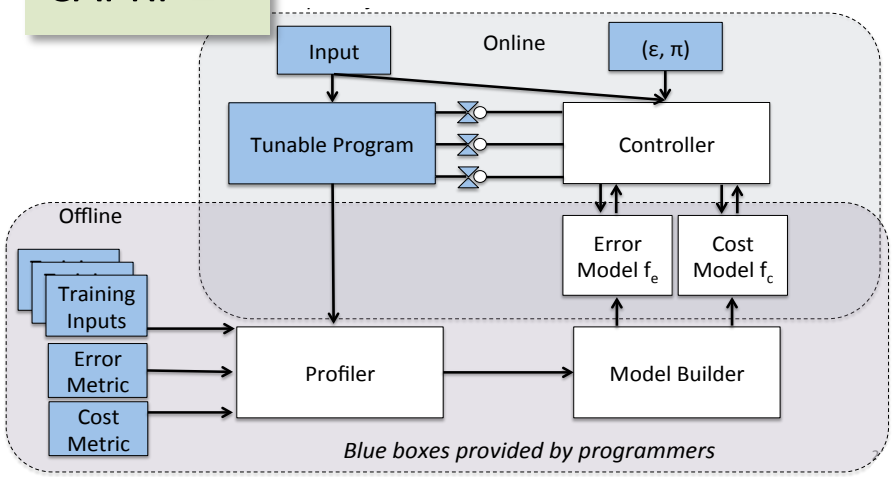
- Pure autotuning approaches
  - Error = f(x,i)
- Proactive and/or reactive approaches:
  - Error < K => x' : f(x',i) < K

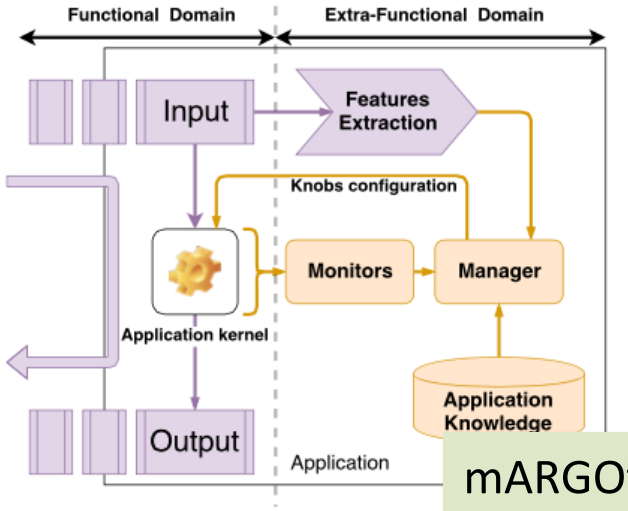
H. Hoffmann et al. "Dynamic knobs for responsive power-aware computing." ASPLOS 2012; H. Hoffmann "JouleGuard: […]" SOSP 2015

PowerDial



CAPRI





mARGOt

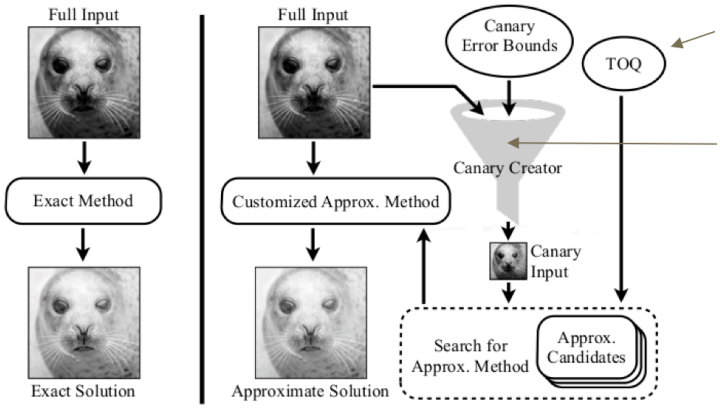
X. Sui et al. "Proactive Control of Approximate Programs" ASPLOS 2016

D. Gadioli et al. "mARGOt: a Dynamic Autotuning Framework Targeting Adaptivity and Controllable Approximation" TCOM19

Removing Offline Quality Profiling

IRA – Input Responsive Approximation

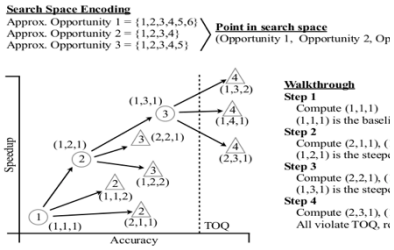
It leverages **canary inputs** to select the approximation level



Quality threshold

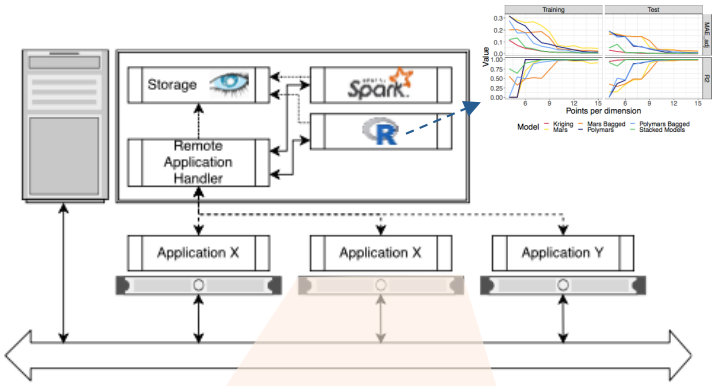
Smallest canary that has the same <property> as the full input:

- mean
- variance
- local homogeneity
- autocorrelation

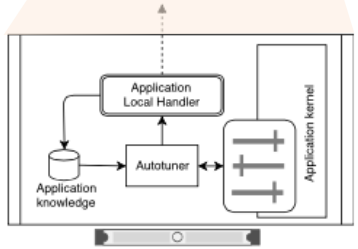


M. Laurenzano et al. "Input responsiveness: using canary inputs to dynamically steer approximation." PLDI 16.

mARGOt - AGORA



(a) Global structure of the distributed DSE framework



(b) Structure of an application instance

T. Martinovic et al. "On-line Application Autotuning Exploiting Ensemble Models" arXiv:1901.06228

Thanks also to…

ANTAREX

European Commission | Horizon 2020 European Union funding for Research & Innovation

http://www.antarex-project.eu/