Advanced Computer Architectures
(High Performance Processors and Systems)

Computing Systems
… and other interesting things
day 2 ;)

Politecnico di Milano

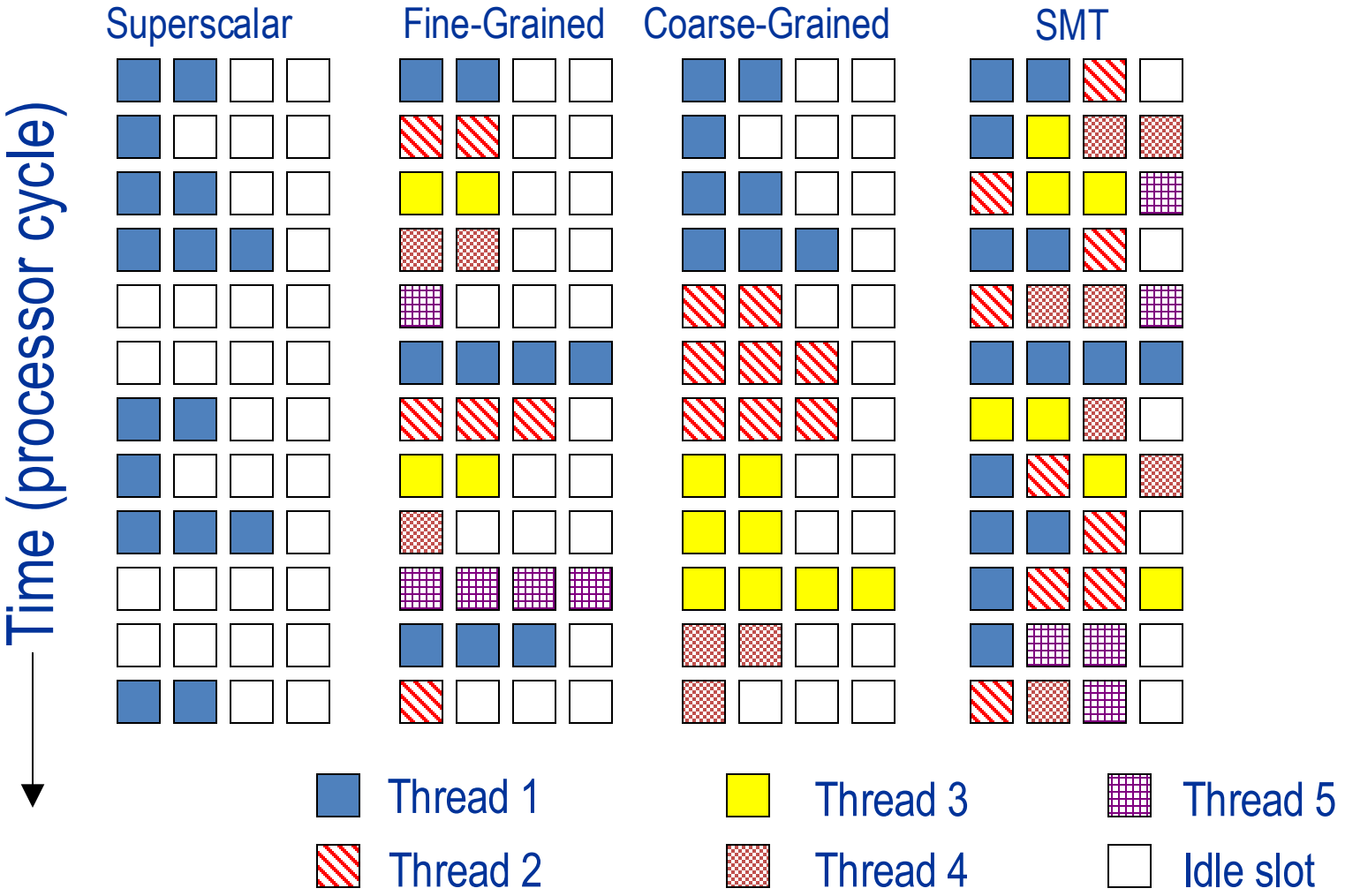
v1

Alessandro Verosimile <Alessandro.verosimile@polimi.it>
Marco D. Santambrogio <marco.santambrogio@polimi.it>

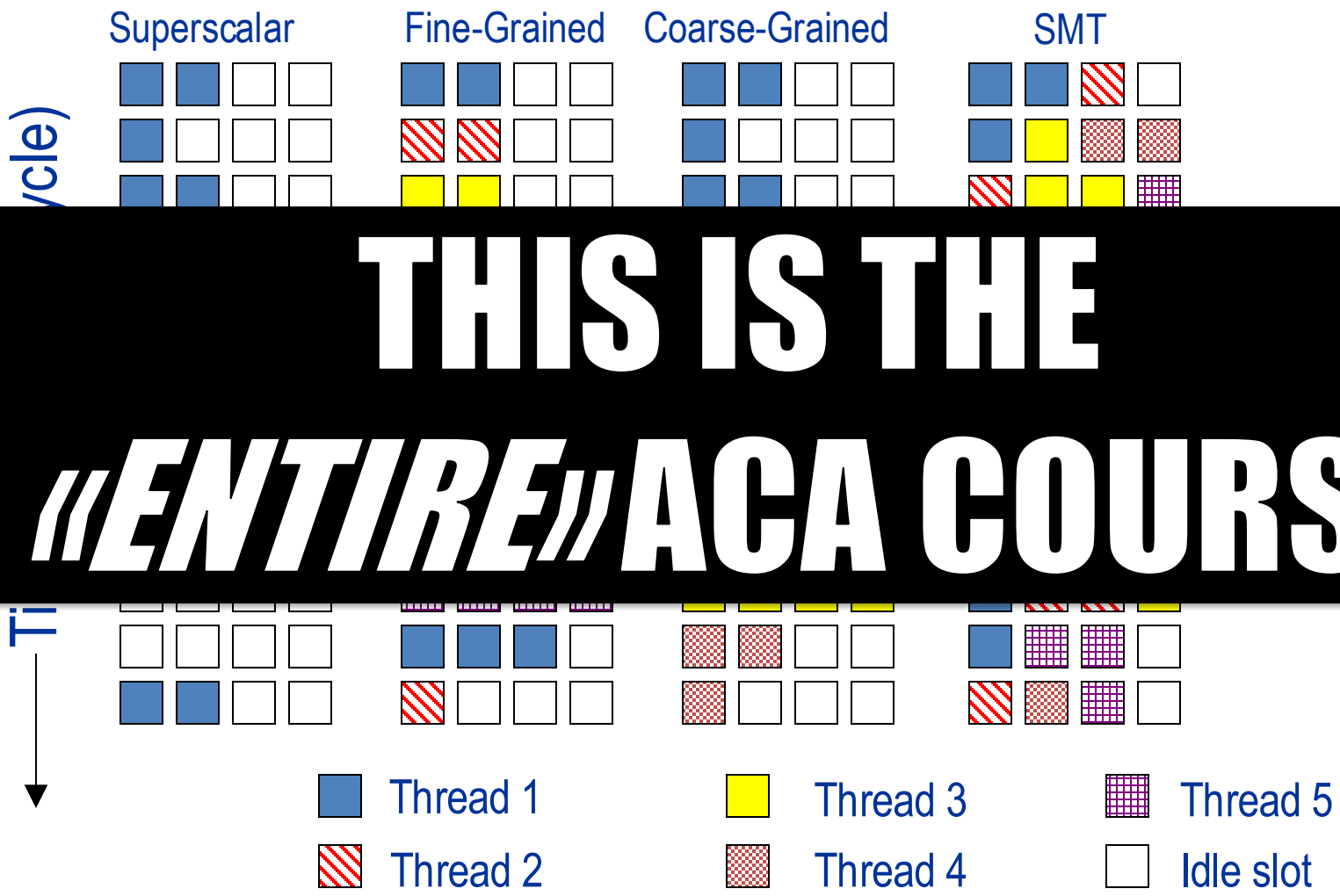
@19.3...

https://flic.kr/p/8puKvo

Multithreaded Categories



Multithreaded Categories



THIS IS THE

«ENTIRE» ACA COURSE

What now?

- Difficult to increase performance and clock frequency of the single core
- Deep pipeline:
  - Heat dissipation problems
  - Speed light transmission problems in wires
  - Difficulties in design and verification
  - Requirement of very large design groups

- Many new applications are multi-threaded

Parallel programming

- Explicit parallelism implies structuring the applications into concurrent and communicating tasks
- Operating systems offer support for different types of tasks. The most important and frequent are:
  - processes
  - threads
- The operating systems implement multitasking differently based on the characteristics of the processor:
  - single core
  - single core with multithreading support
  - multicore

Flynn Taxonomy (1966)

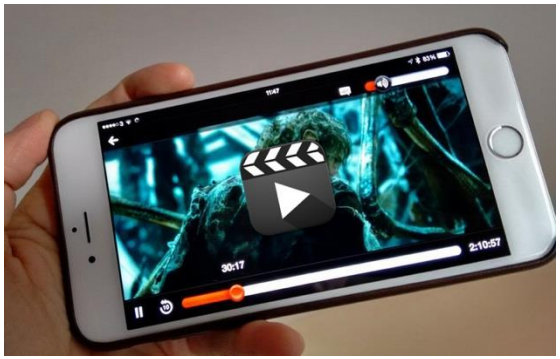
- **SISD** - Single Instruction Single Data
  - Uniprocessor systems
- **MISD** - Multiple Instruction Single Data
  - No practical configuration and no commercial systems
- **SIMD** - Single Instruction Multiple Data
  - Simple programming model, low overhead, flexibility, custom integrated circuits
- **MIMD** - Multiple Instruction Multiple Data
  - Scalable, fault tolerant, *off-the-shelf* micros

Workloads are dynamic

- On a mobile phone:
  - Phone calls
  - Short message service
  - Web browsing
  - Audio/video playing
  - Gaming

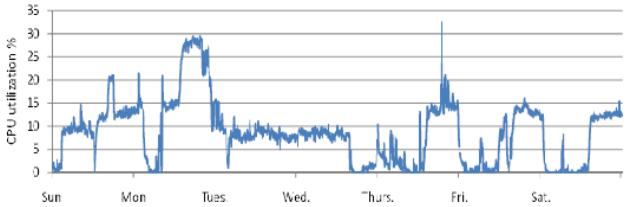
Generally short execution times, low amount of processed data, actually no QoS requirements or not-challenging ones

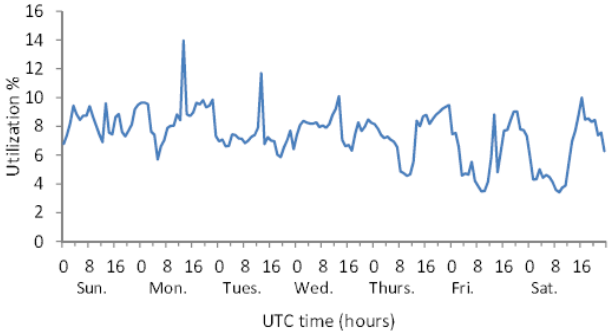
Generally considerable amount of data to be processed with specific throughputs to be fulfilled, high demanding elaborations

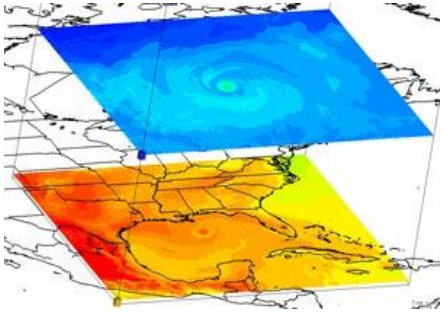


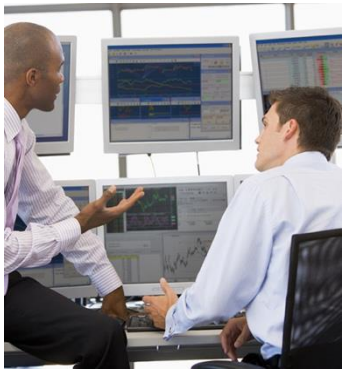
Workloads are dynamic

- On a HPC server:
  - Financial modeling and analysis
  - Fluid dynamic simulations
  - Weather and climatic modeling
  - …









HOW TO HANDLE THIS DYNAMICITY

SYSTEM HAS TO BE HETEROGENEOUS

HOW TO HANDLE THIS DYNAMICITY

SYSTEM HAS TO BE HETEROGENEOUS

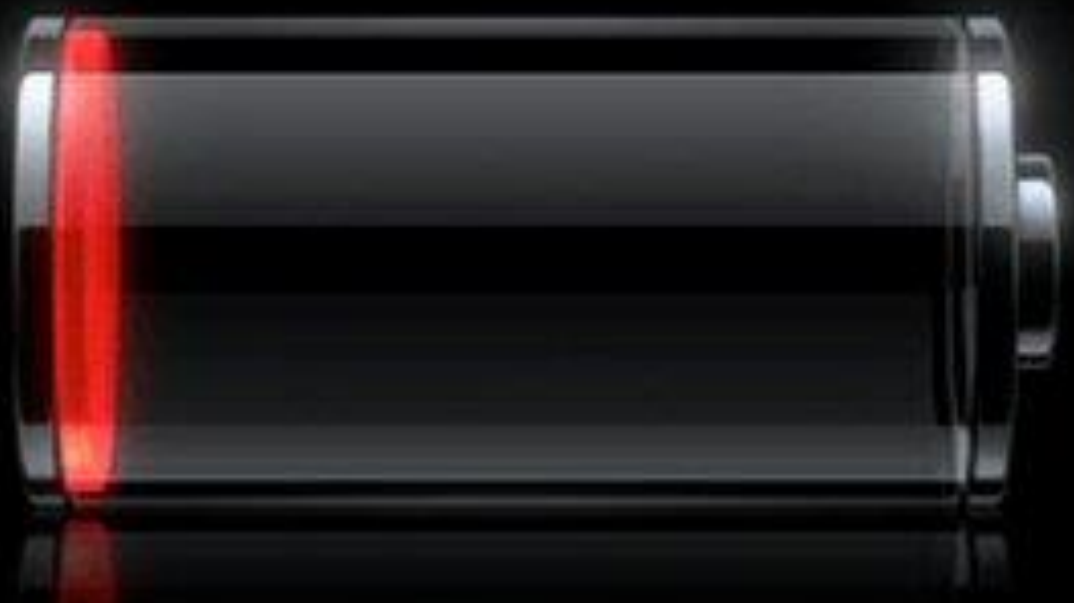
HOW TO HANDLE THIS COMPLEXITY

SYSTEM HAS TO BE ADAPTIVE

HOW TO HANDLE THIS COMPLEXITY


SYSTEM HAS TO BE ADAPTIVE
TO RECOVER
FROM DAMAGES

SYSTEM HAS TO BE ADAPTIVE

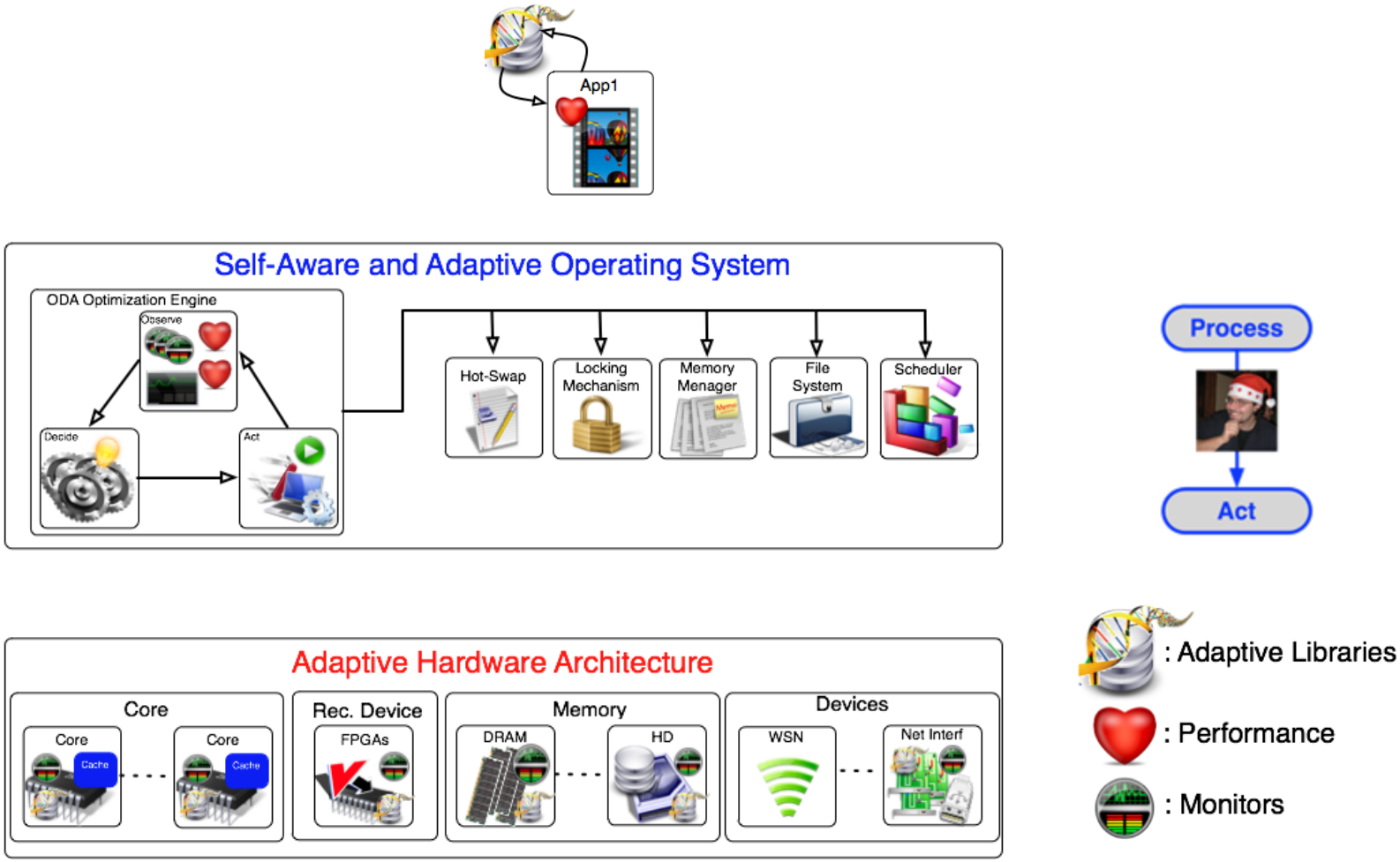


TO GUARANTEE SERVICES OVER POWER CAP AND ENERGY SAVINGS

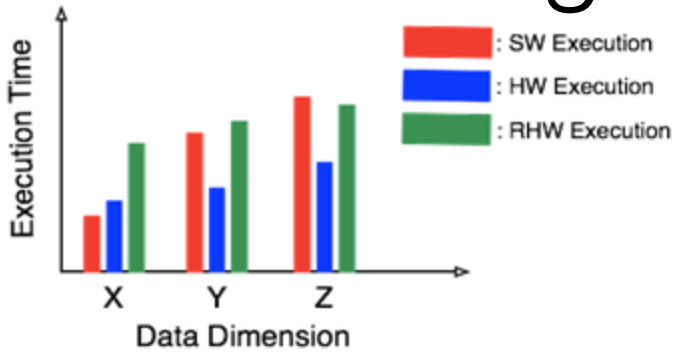

SYSTEM HAS TO BE ADAPTIVE
TO ADAPT TO

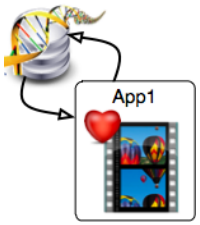

SYSTEM HAS TO BE ADAPTIVE
TO ADAPT TO
UKNOWN CONDITIONS

Reconfiguration: Online Static

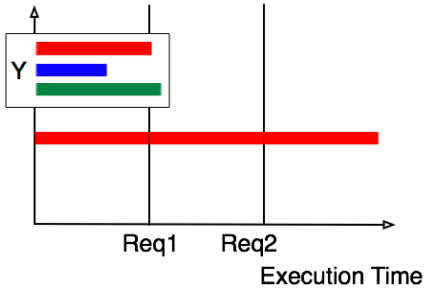


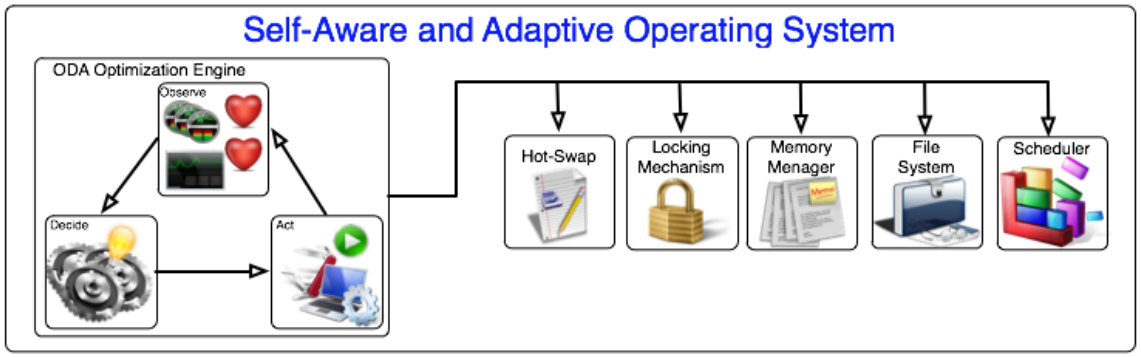
Reconfiguration: Online Static

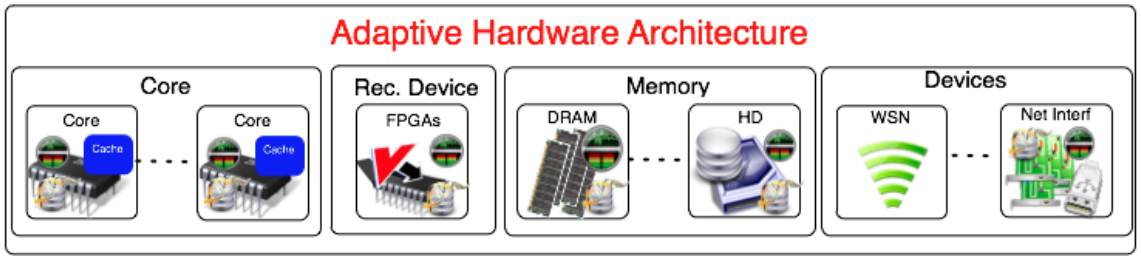


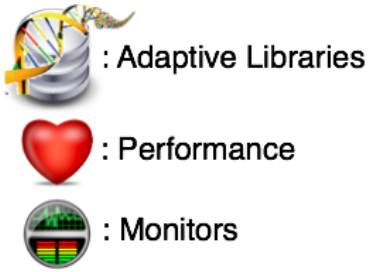


3 requests on Y data (NO HW configured)

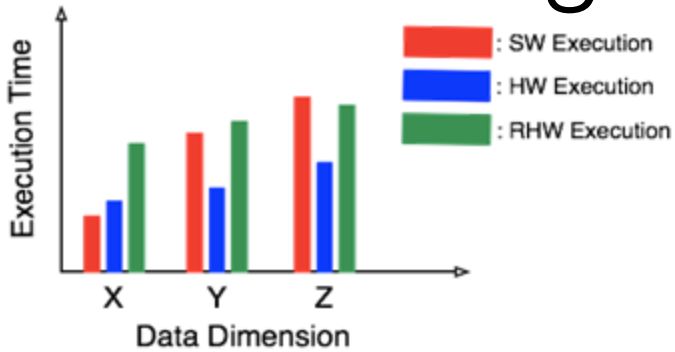


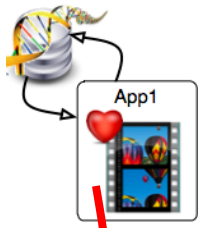




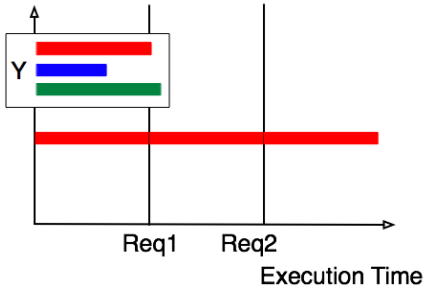


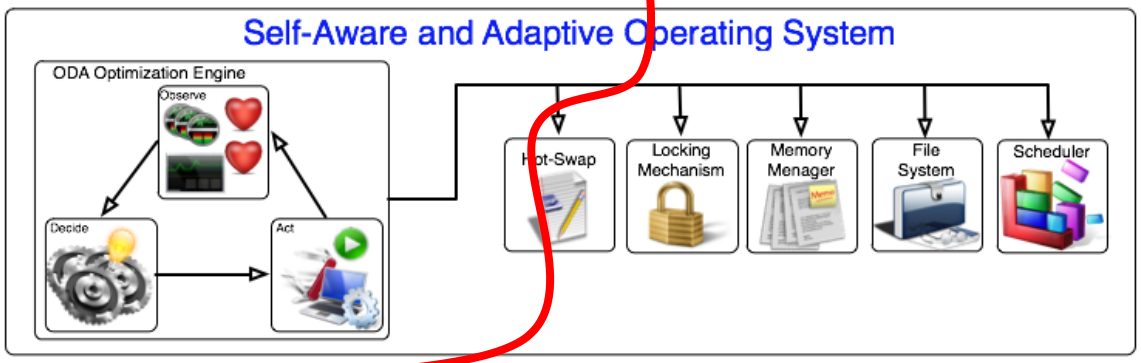
Reconfiguration: Online Static

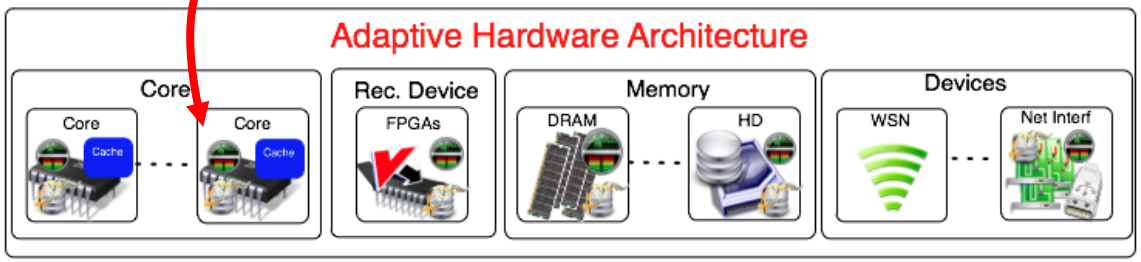


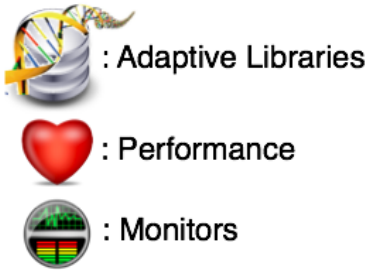


3 requests on Y data (NO HW configured)

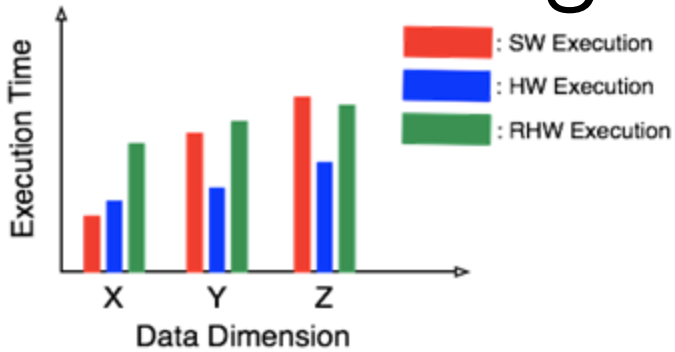


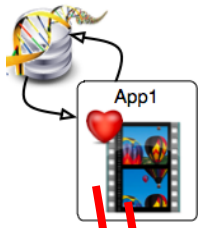




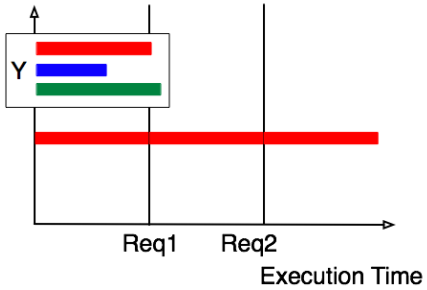


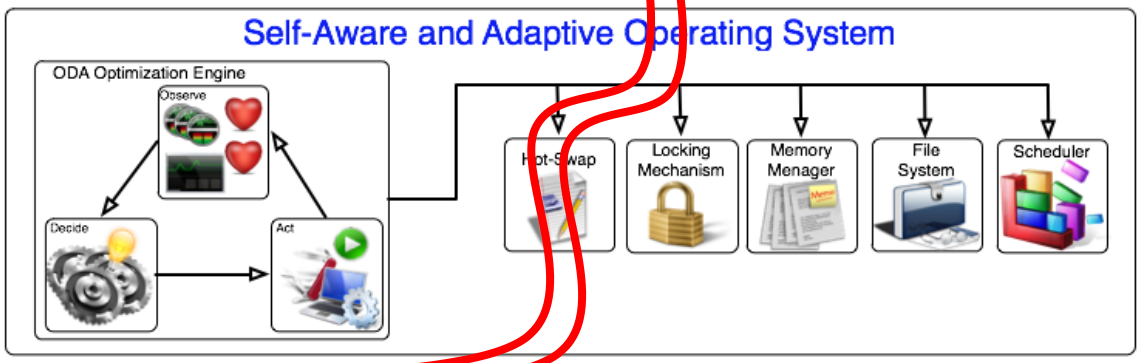
Reconfiguration: Online Static

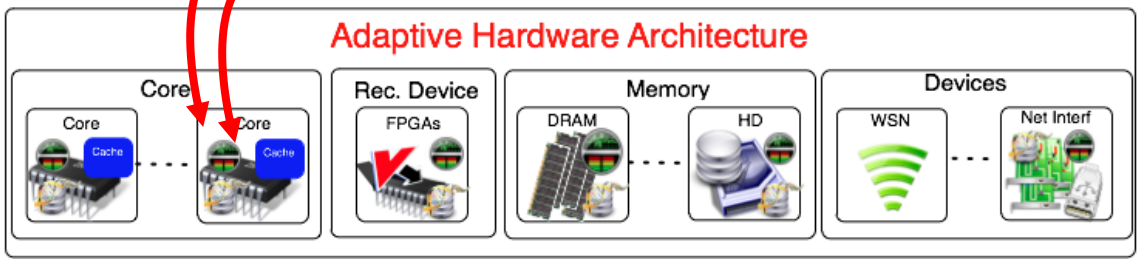


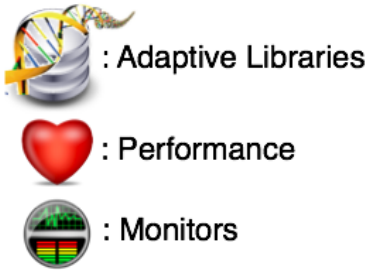


3 requests on Y data (NO HW configured)

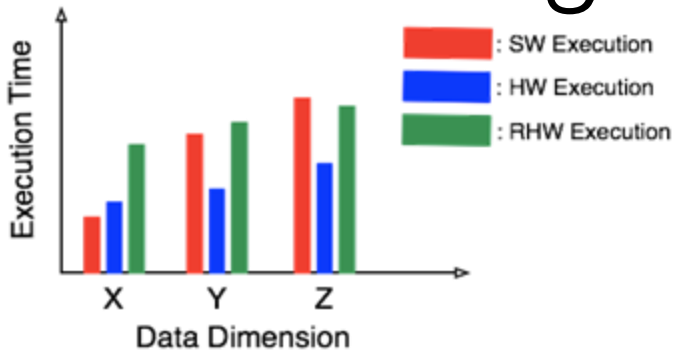


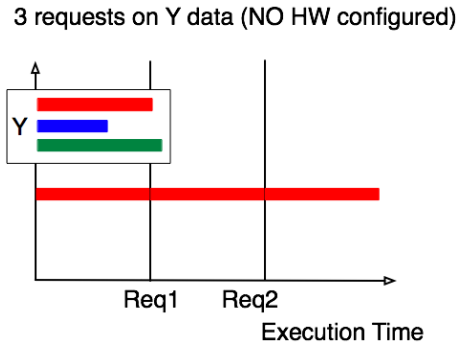


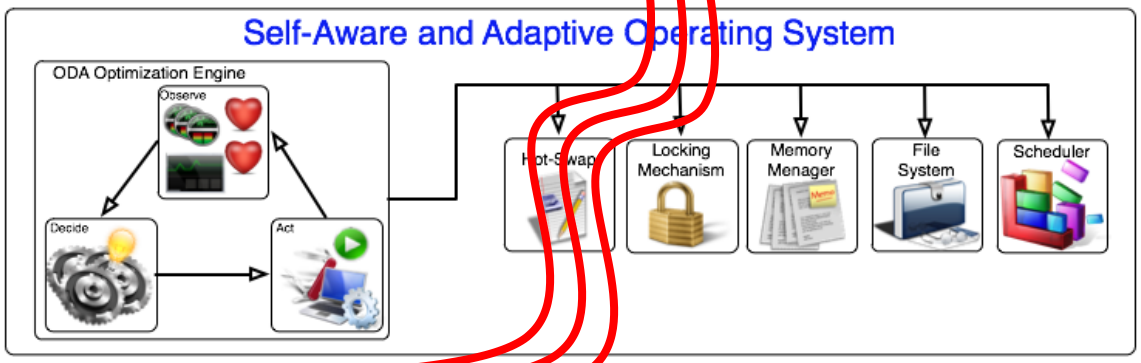


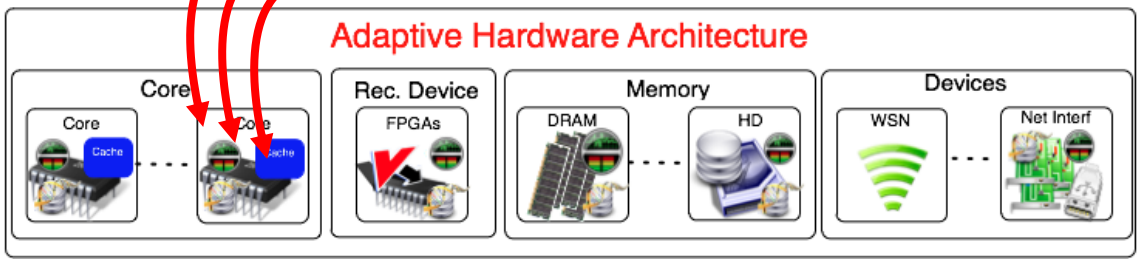


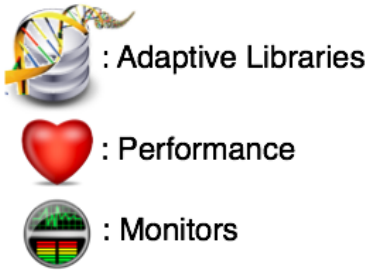
Reconfiguration: Online Static



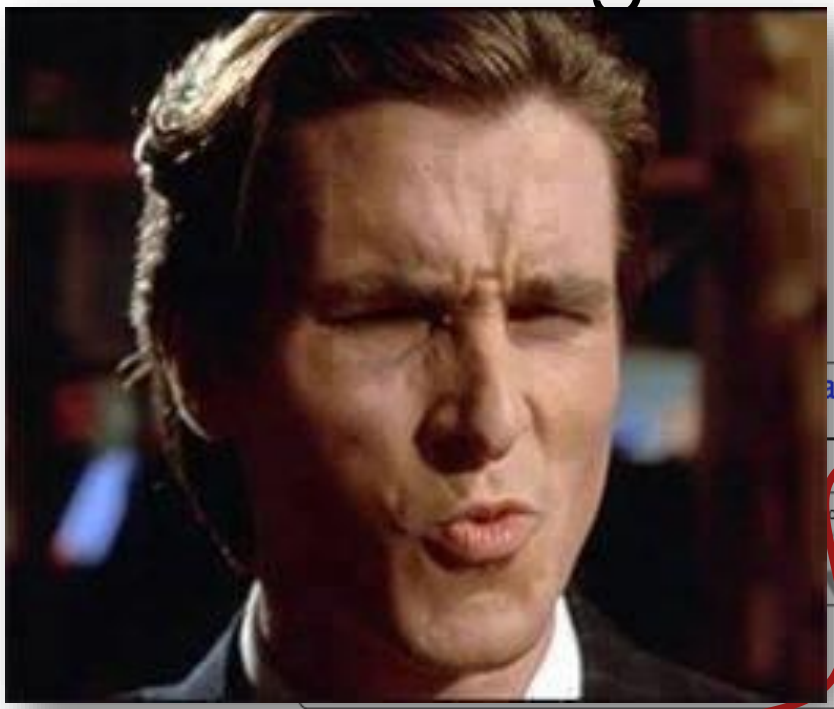


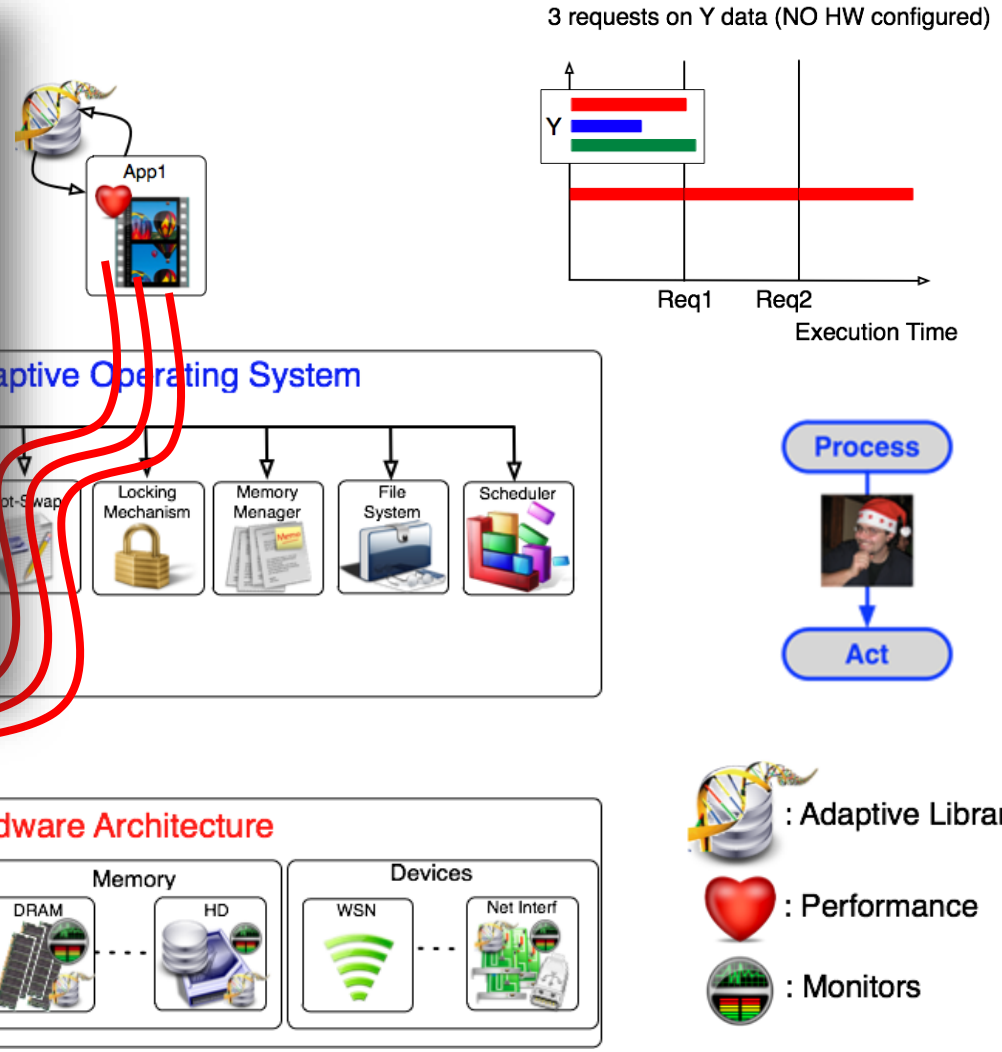






Reconfiguration: Online Static






Where to go NECST?

Trying to raise the bar

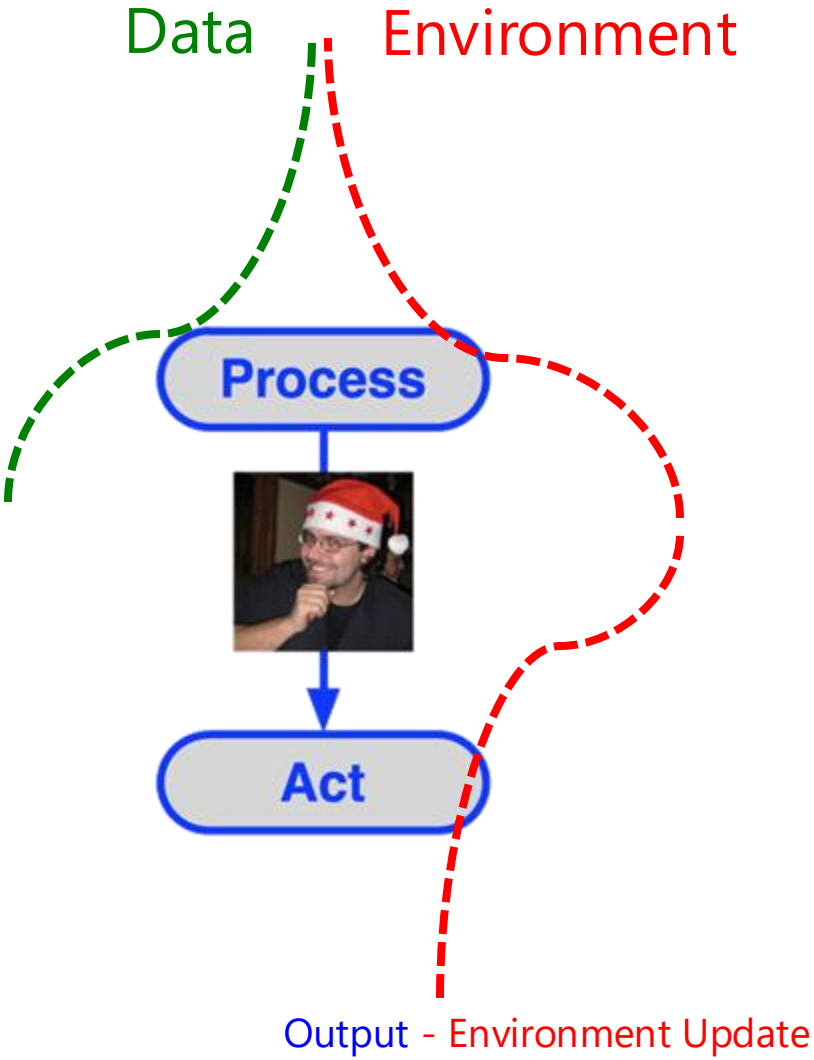
- Towards the design and implementaion of Self-adaptive and autonomic systems

Trying to raise the bar

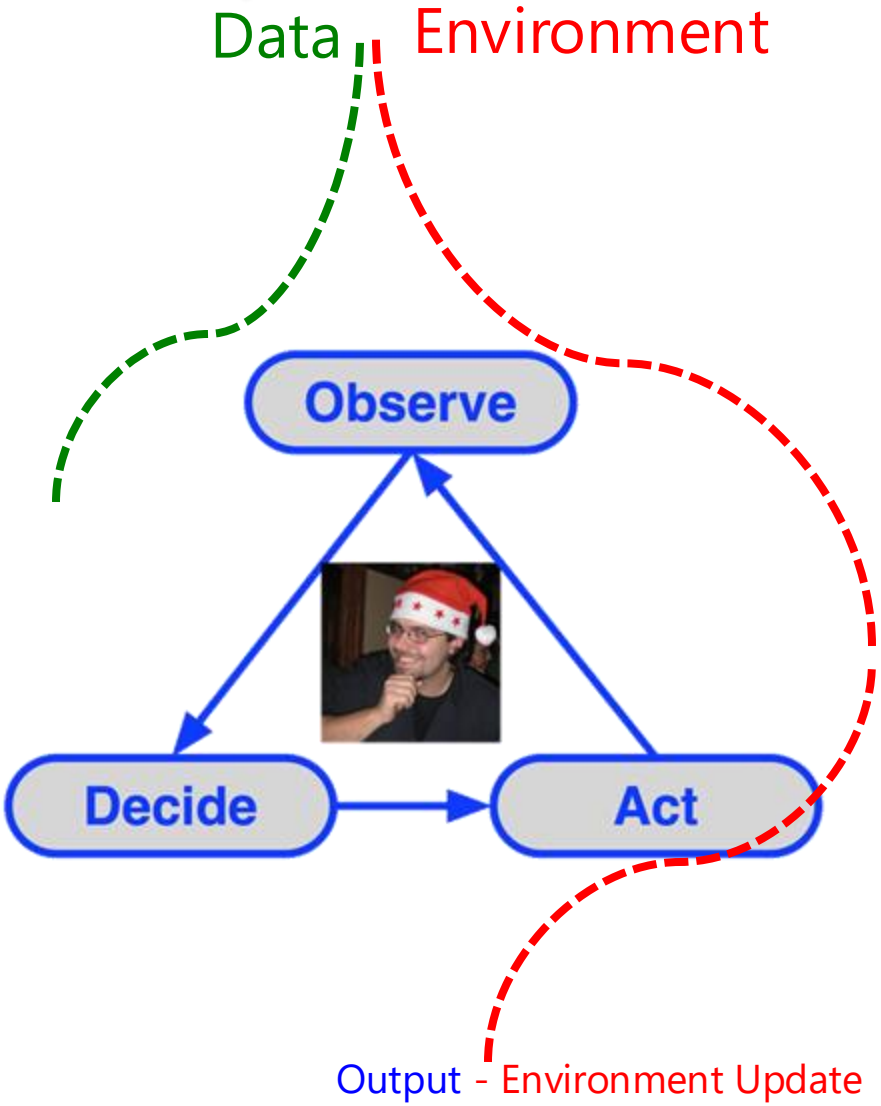
- Towards the design and implementaion of Self-adaptive and autonomic systems

- We need to include 2 more features
  - Goal-
    - Te
    - Sy
  - Appr
    - Do sary to meet goals

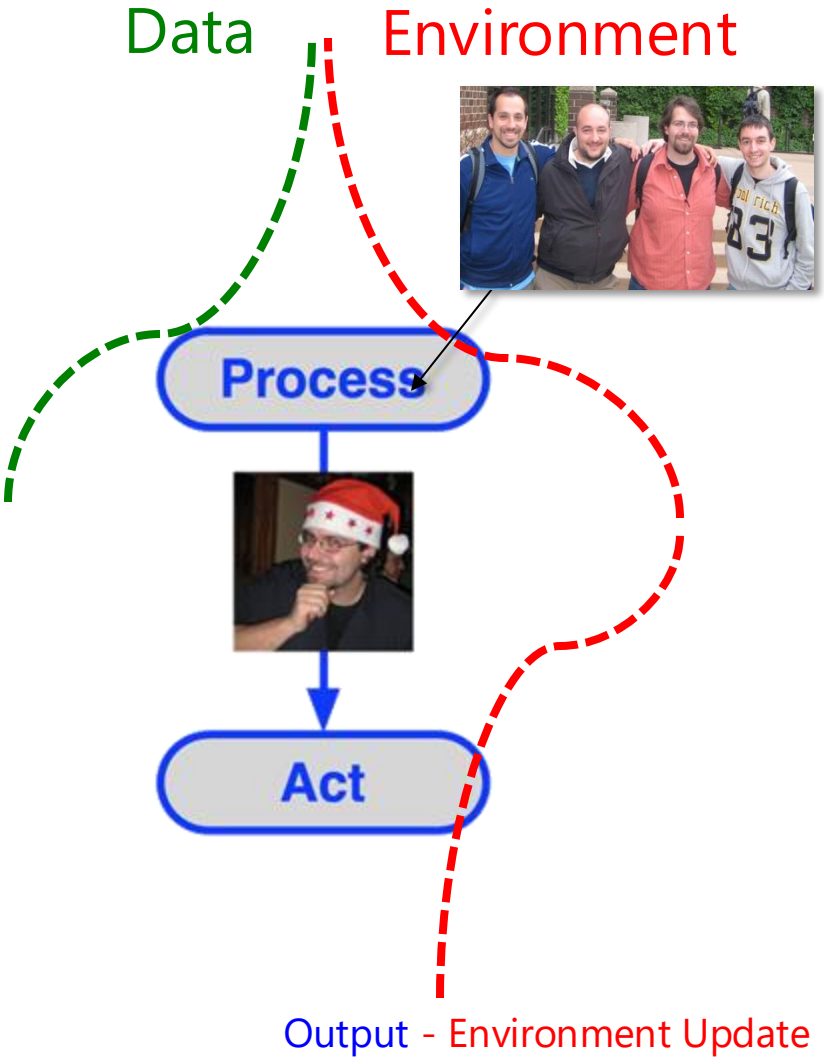
Online Static Solution



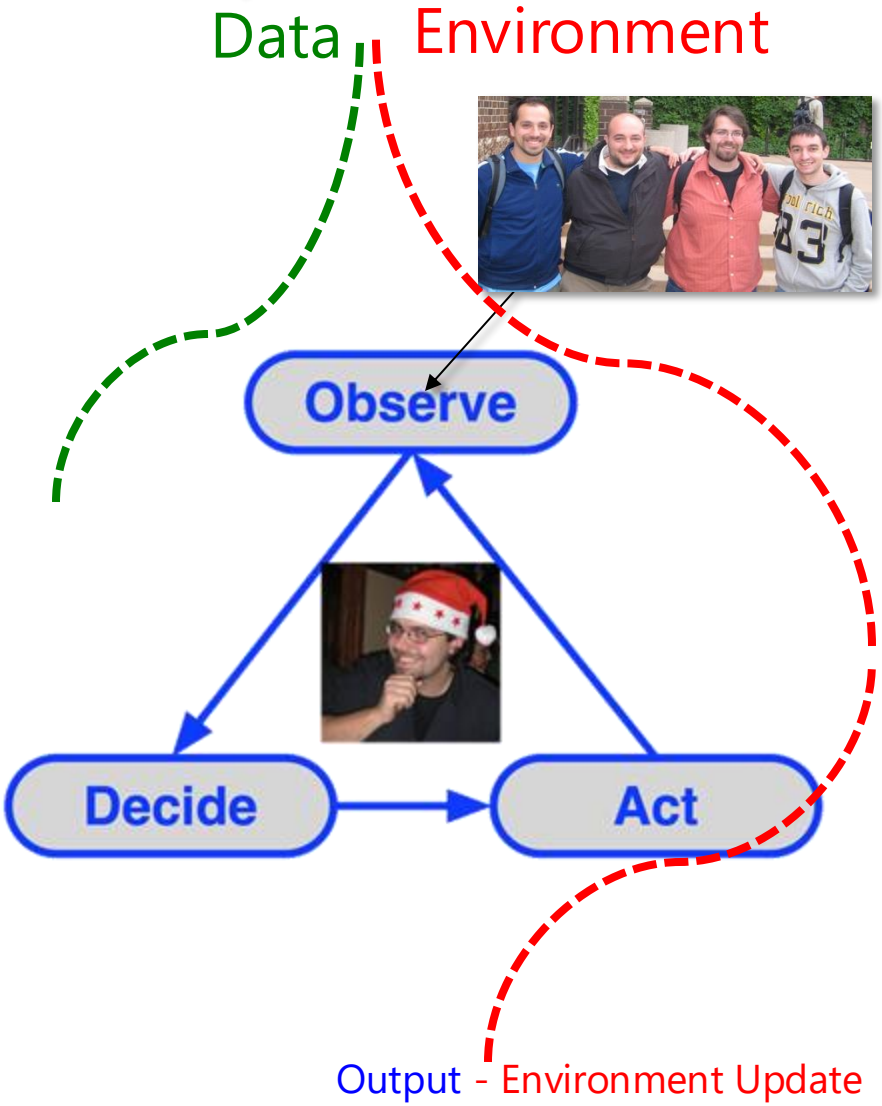
Adaptive Solution



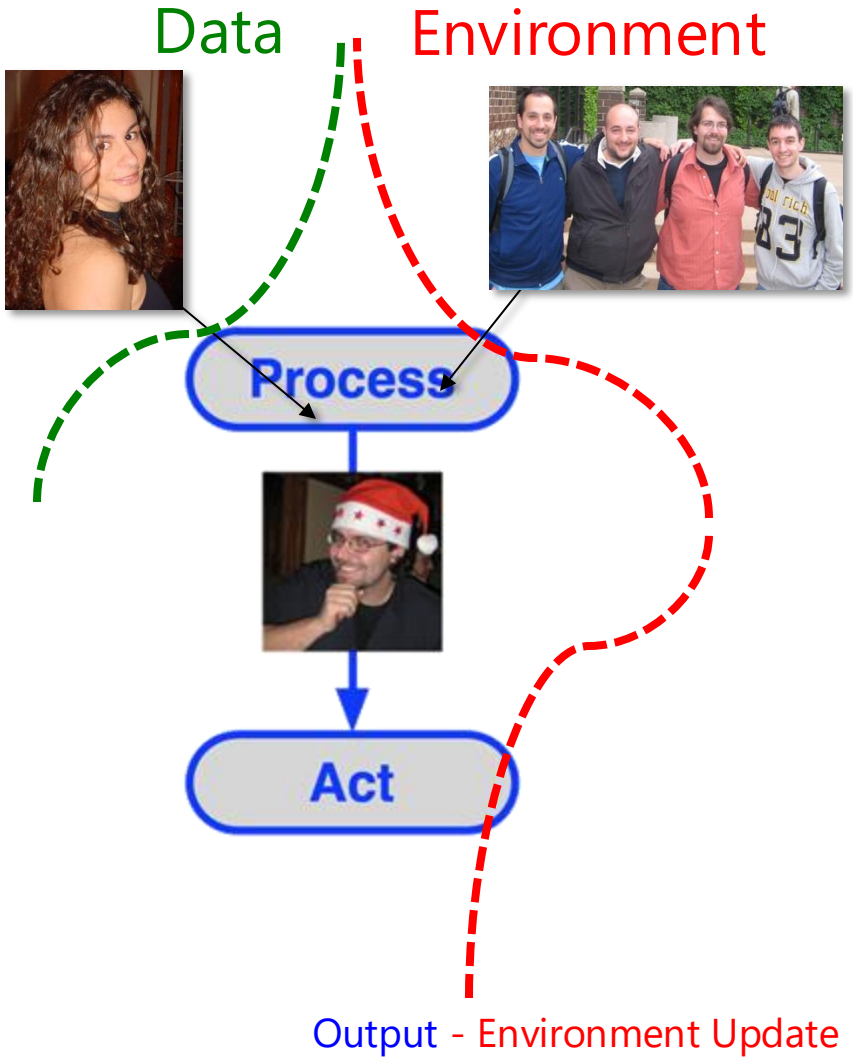
Online Static Solution



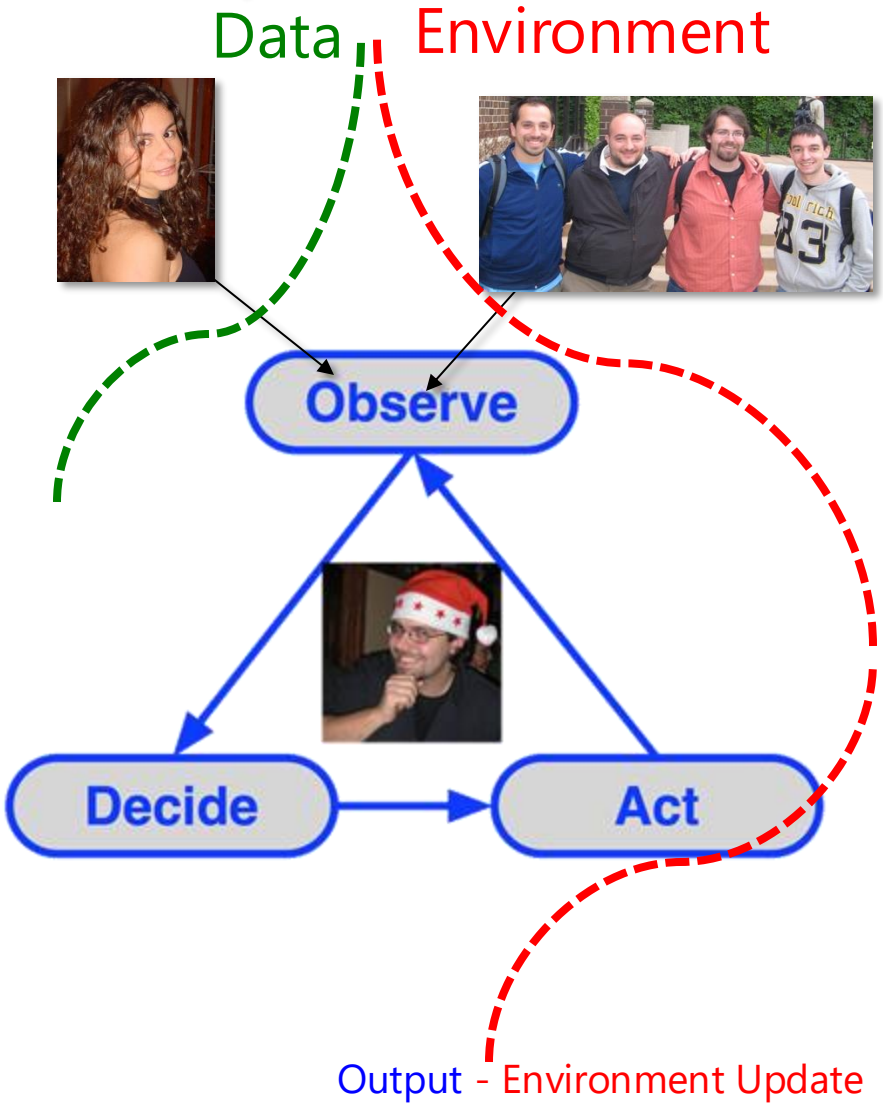
Adaptive Solution



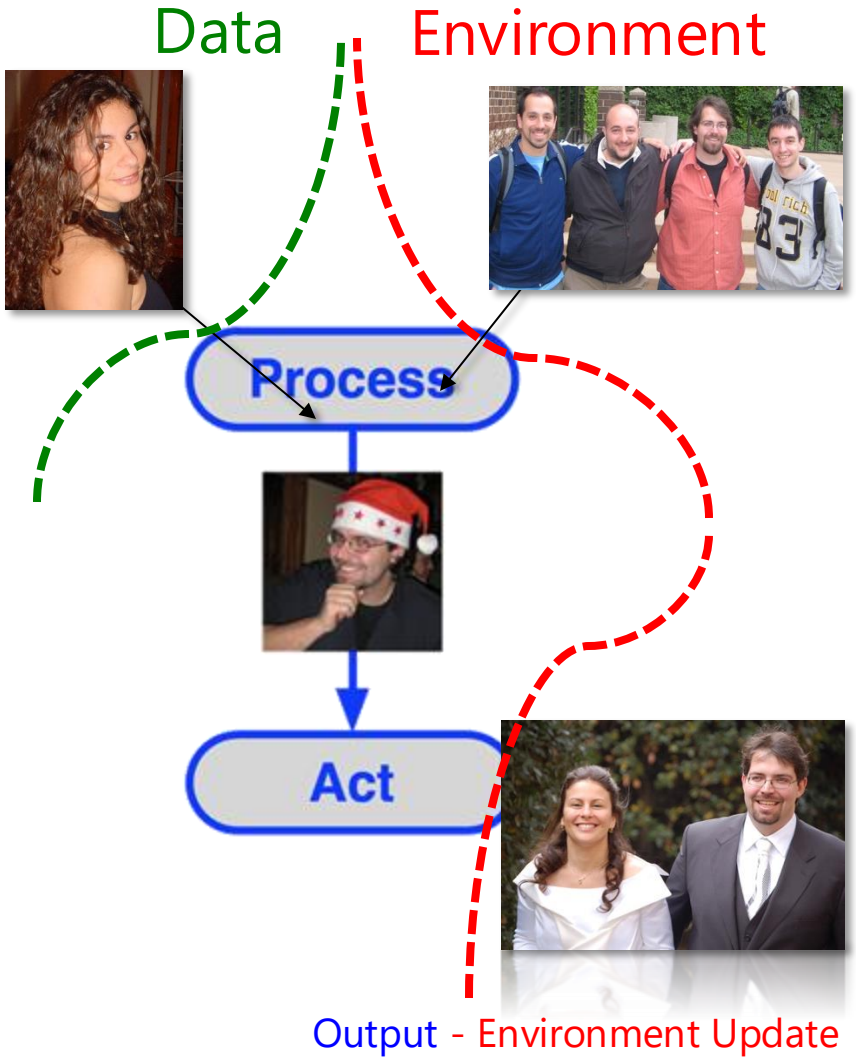
Online Static Solution



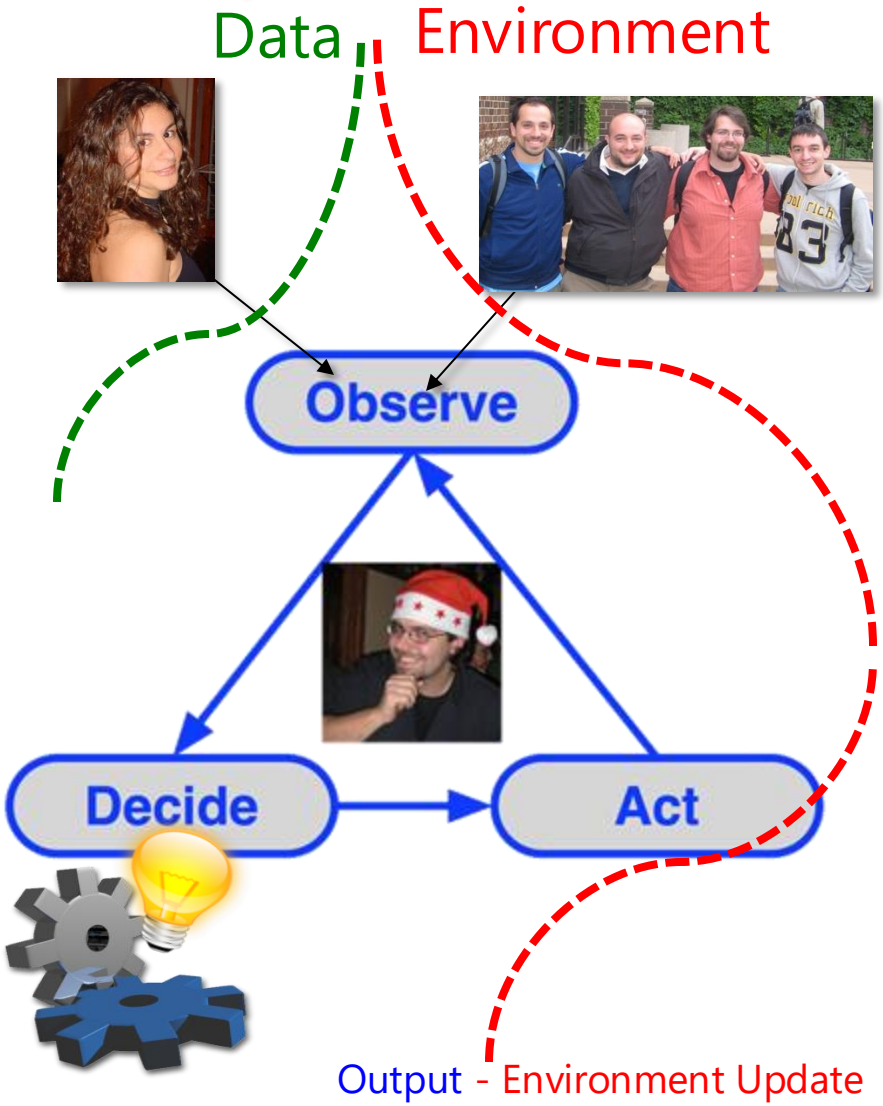
Adaptive Solution



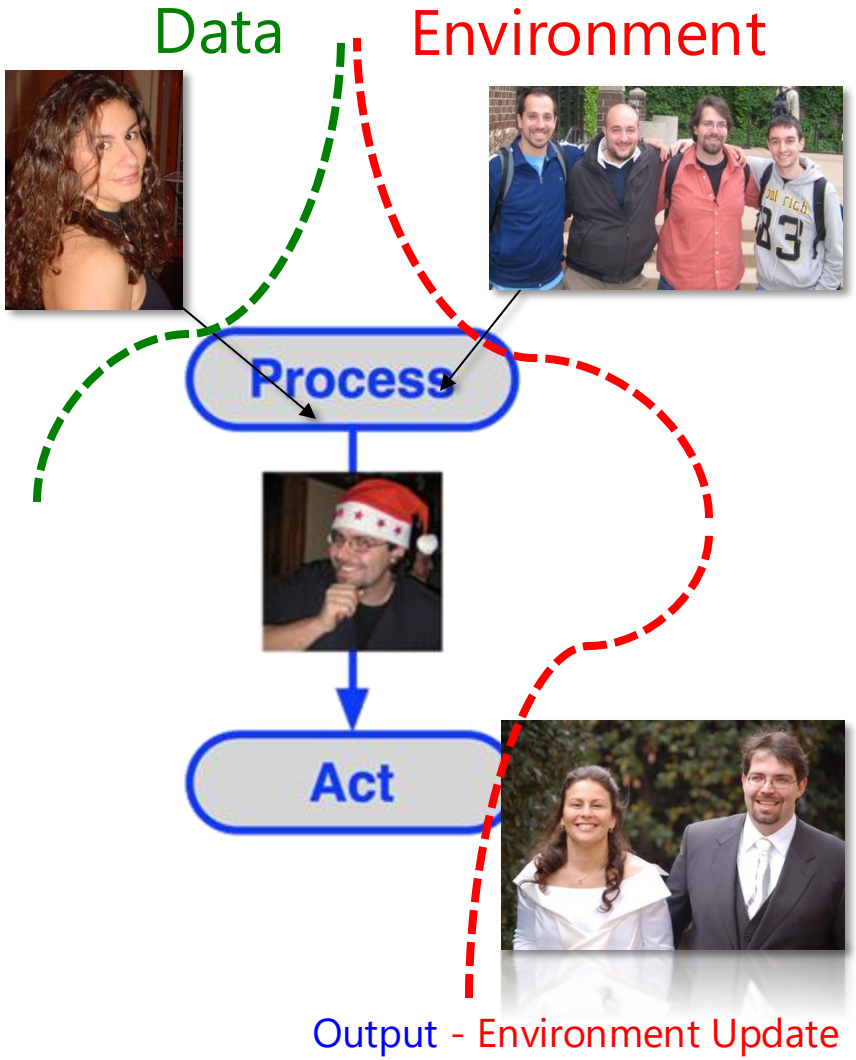
Online Static Solution



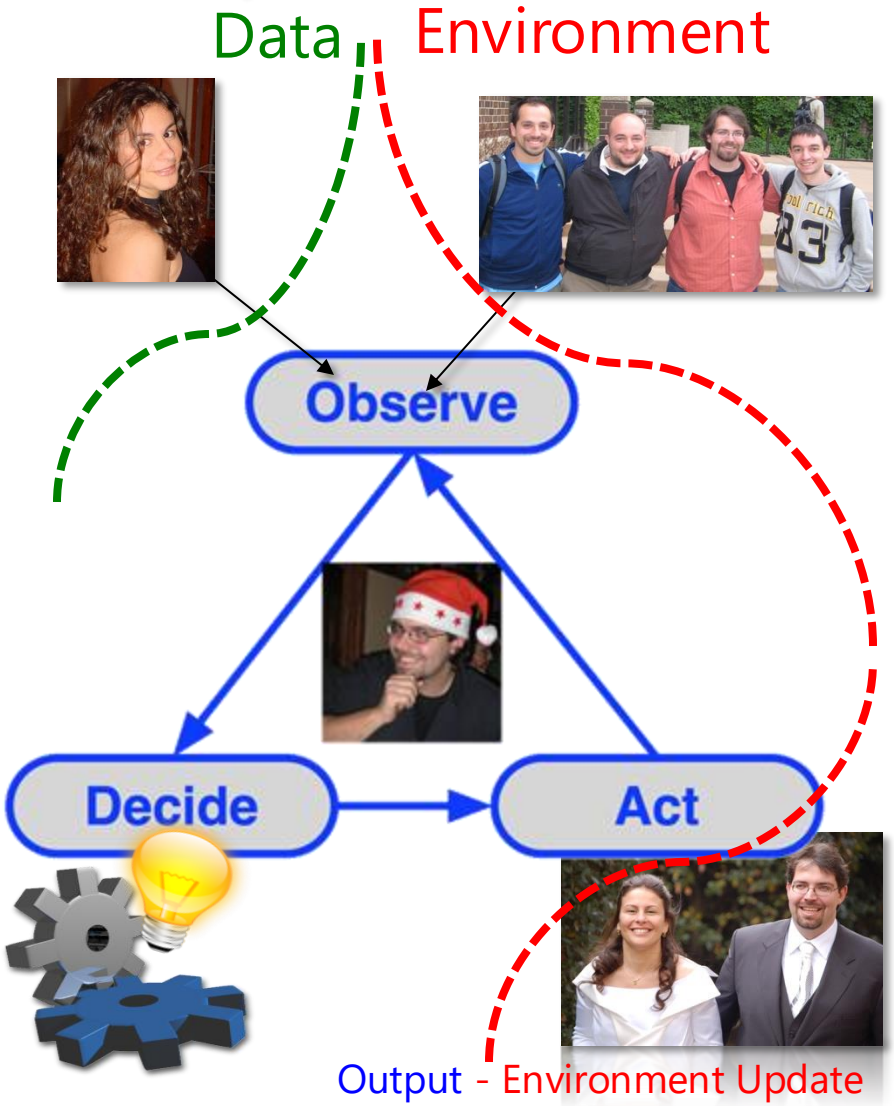
Adaptive Solution



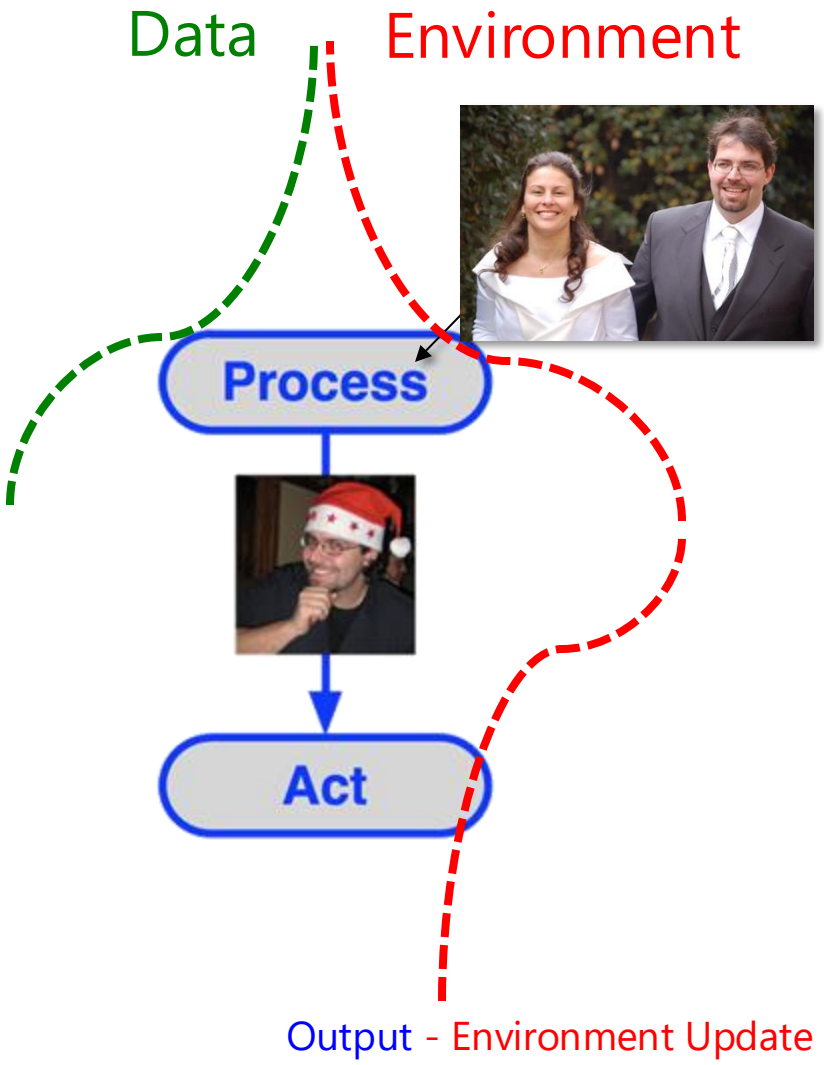
Online Static Solution



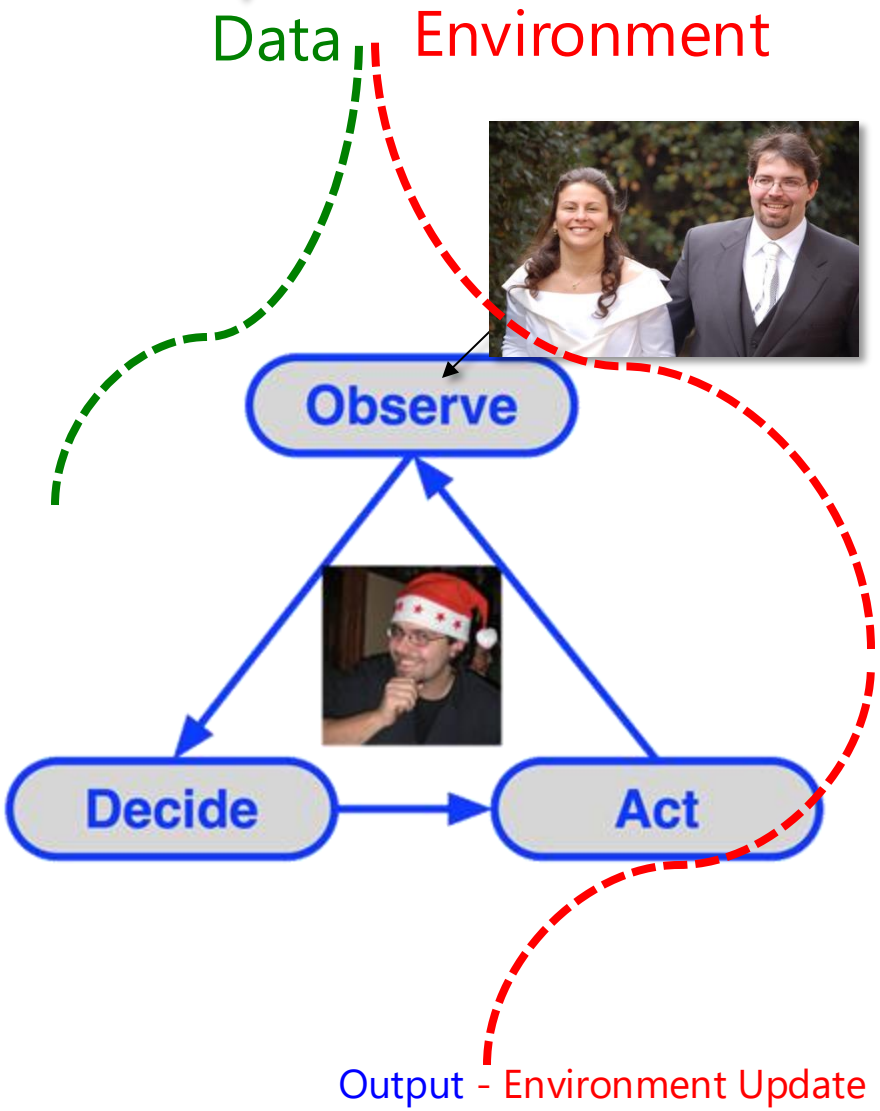
Adaptive Solution



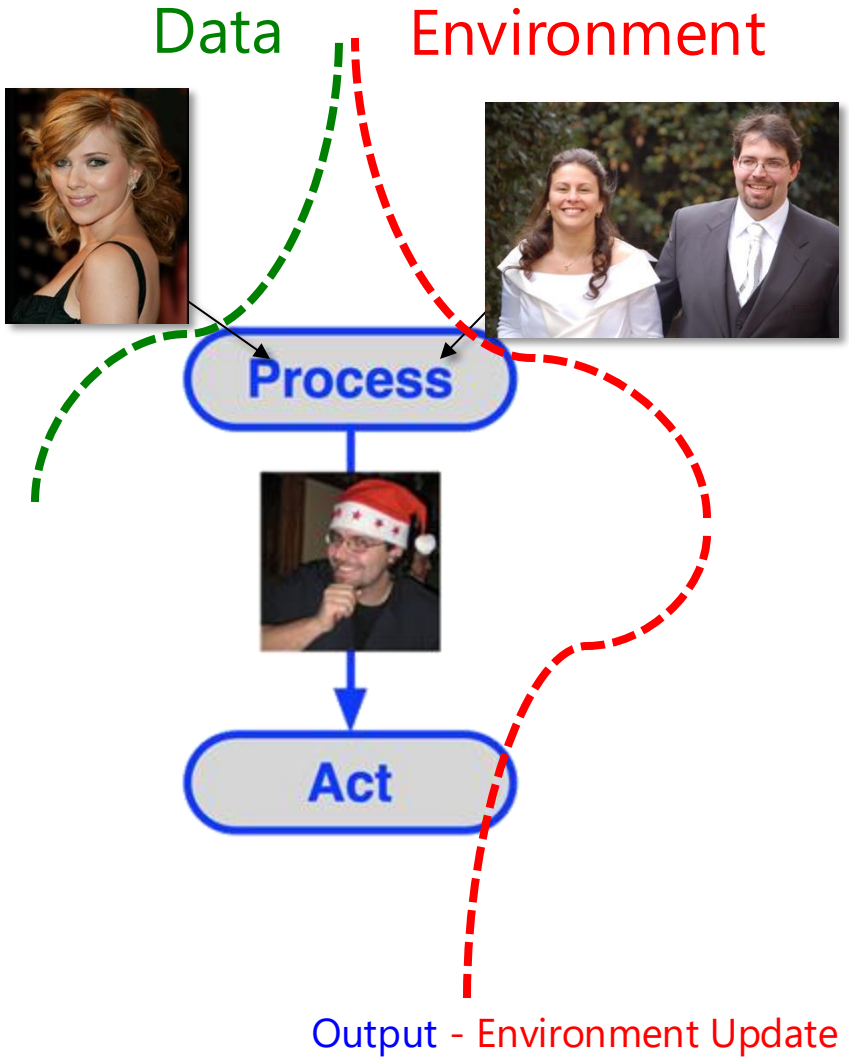
Online Static Solution



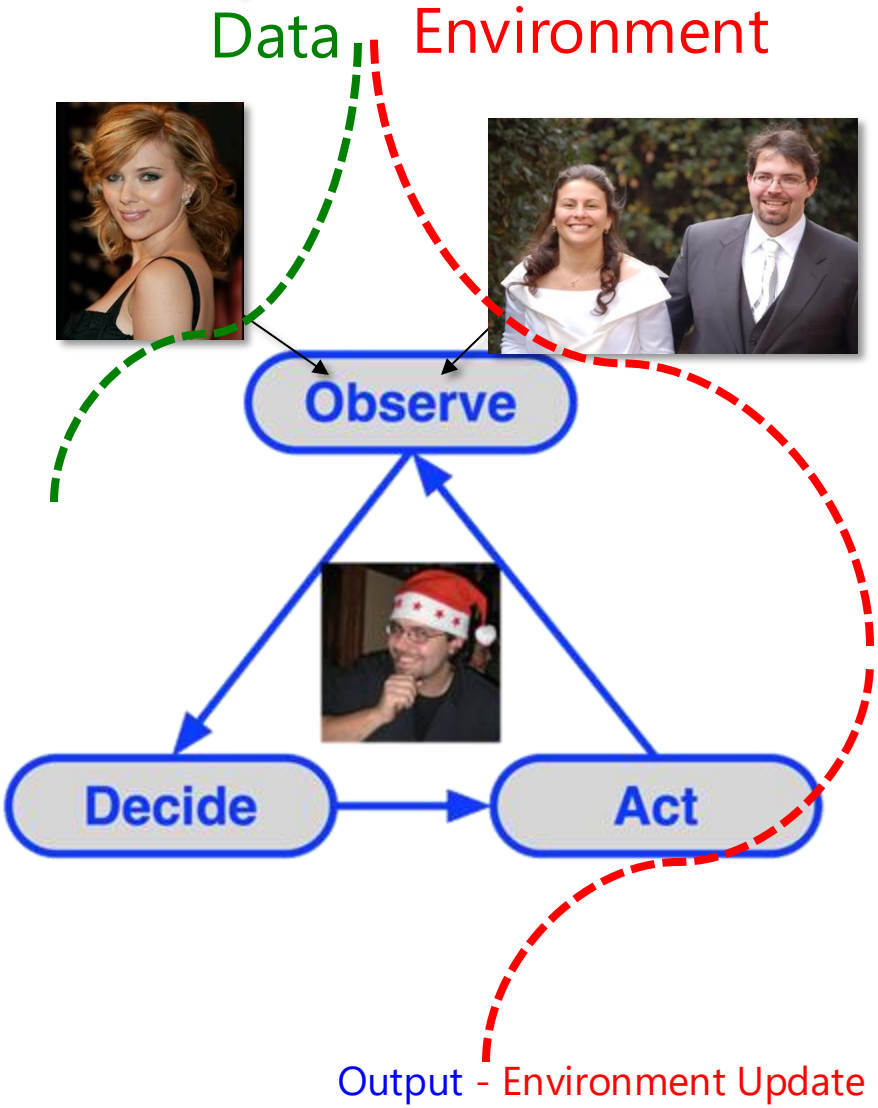
Adaptive Solution



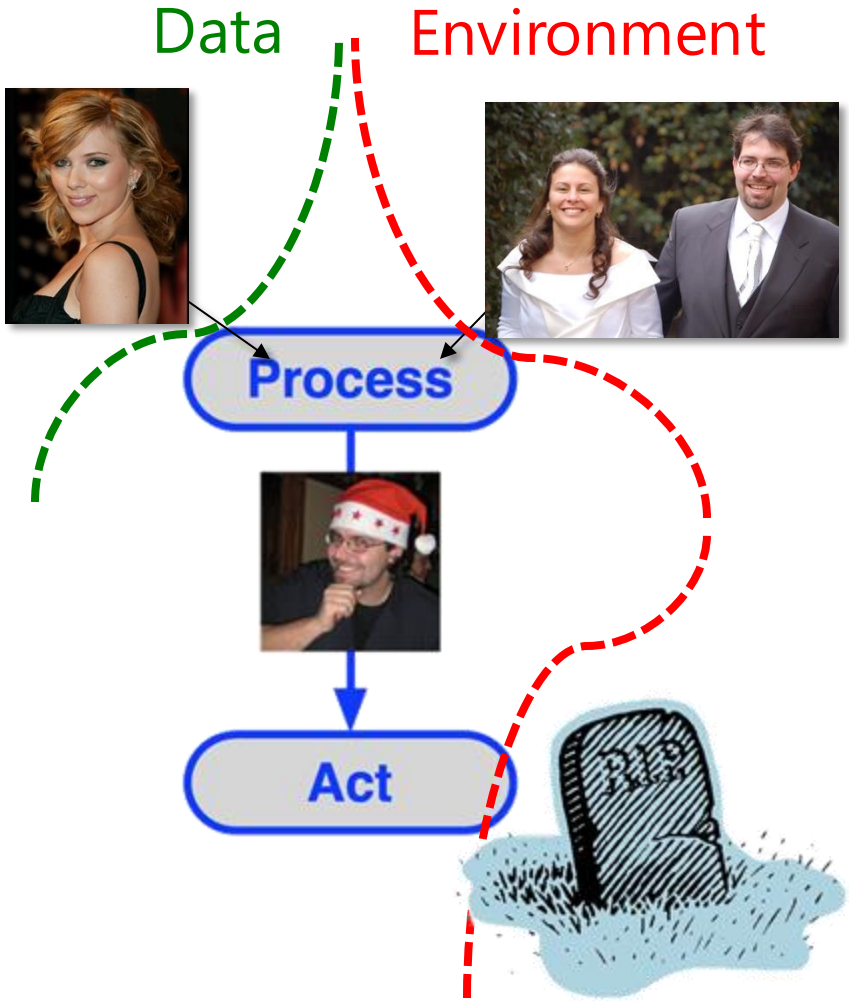
Online Static Solution



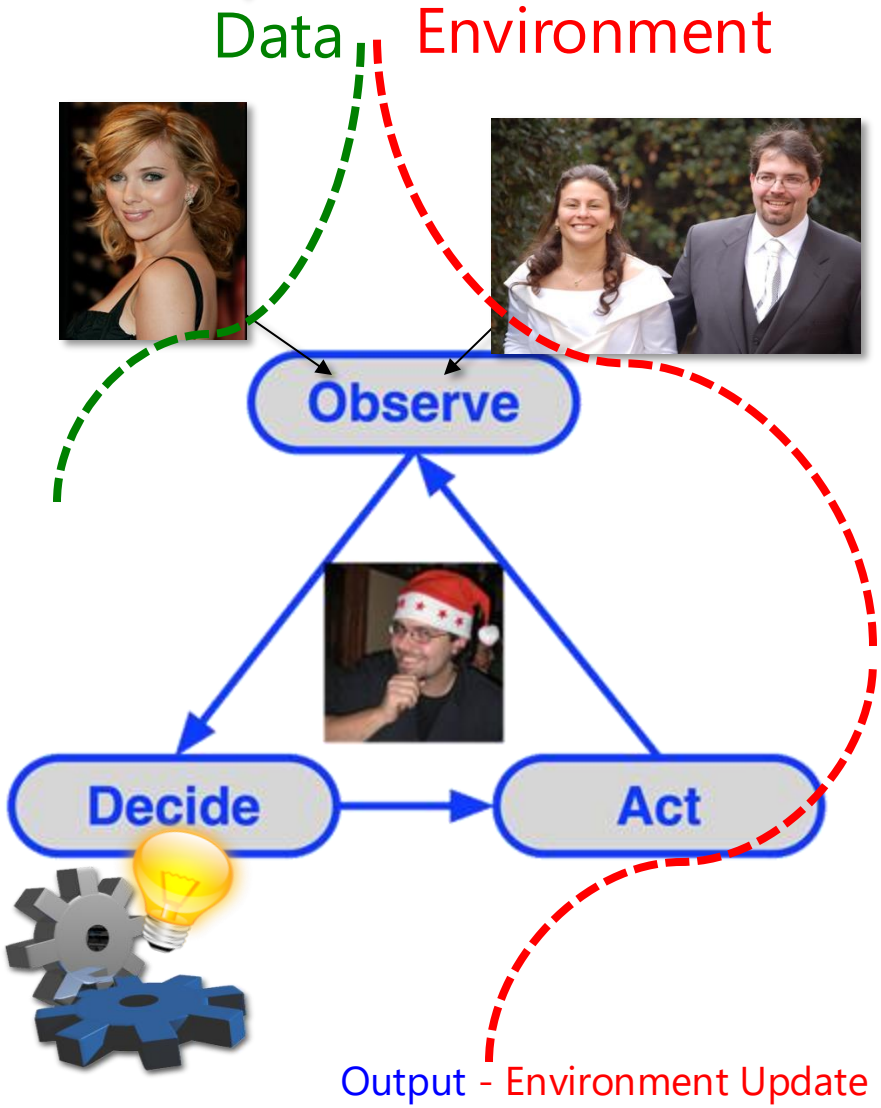
Adaptive Solution



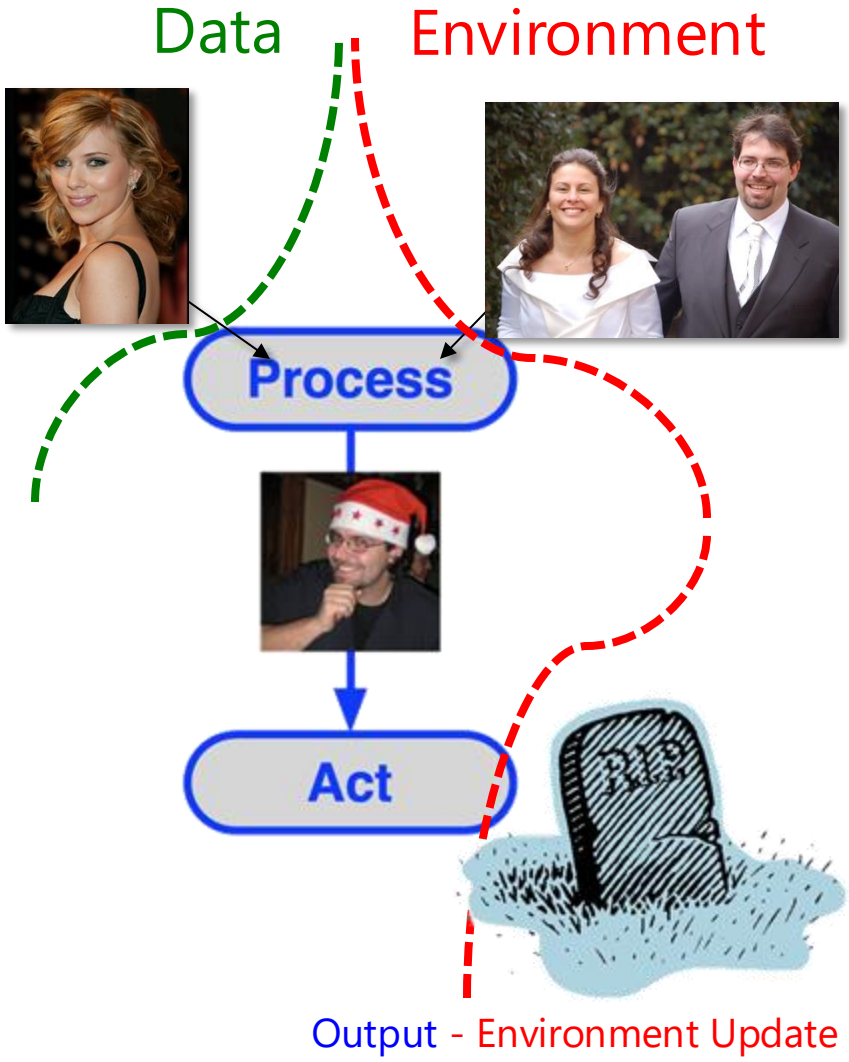
Online Static Solution



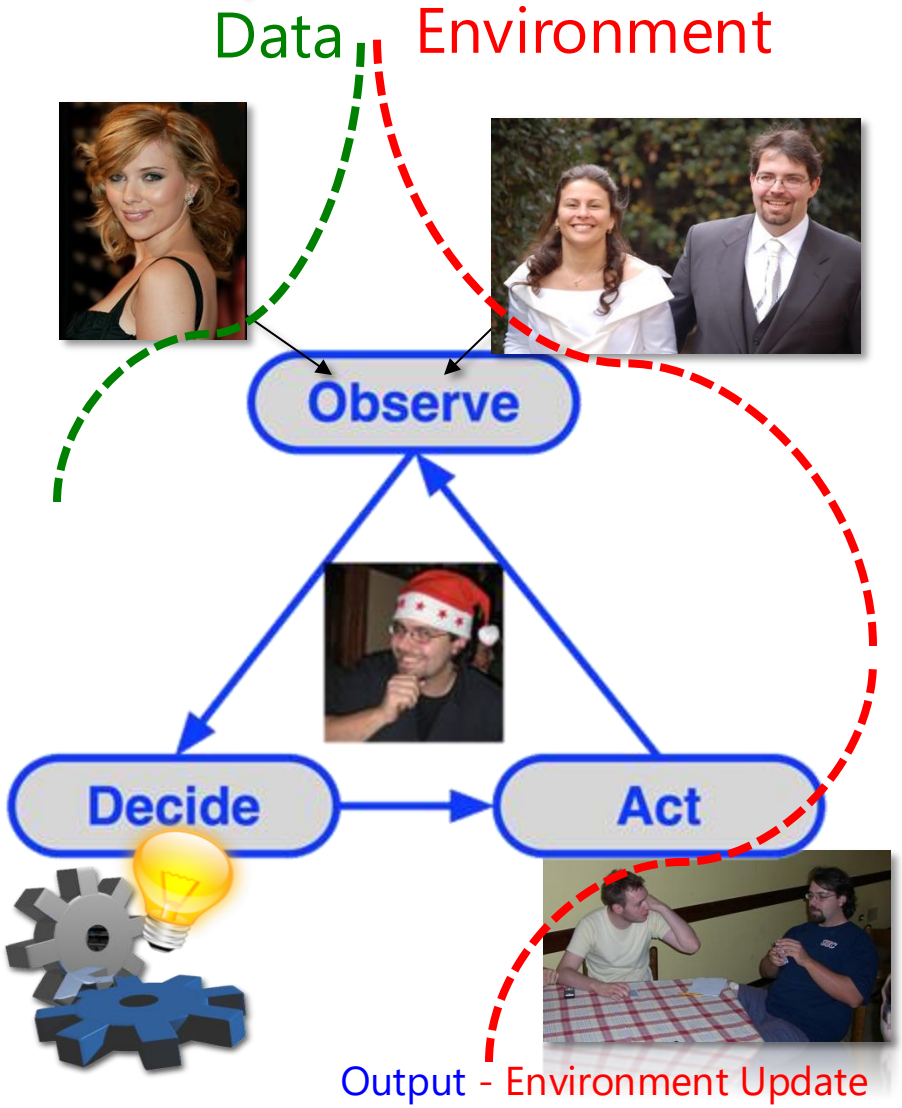
Adaptive Solution



Online Static Solution

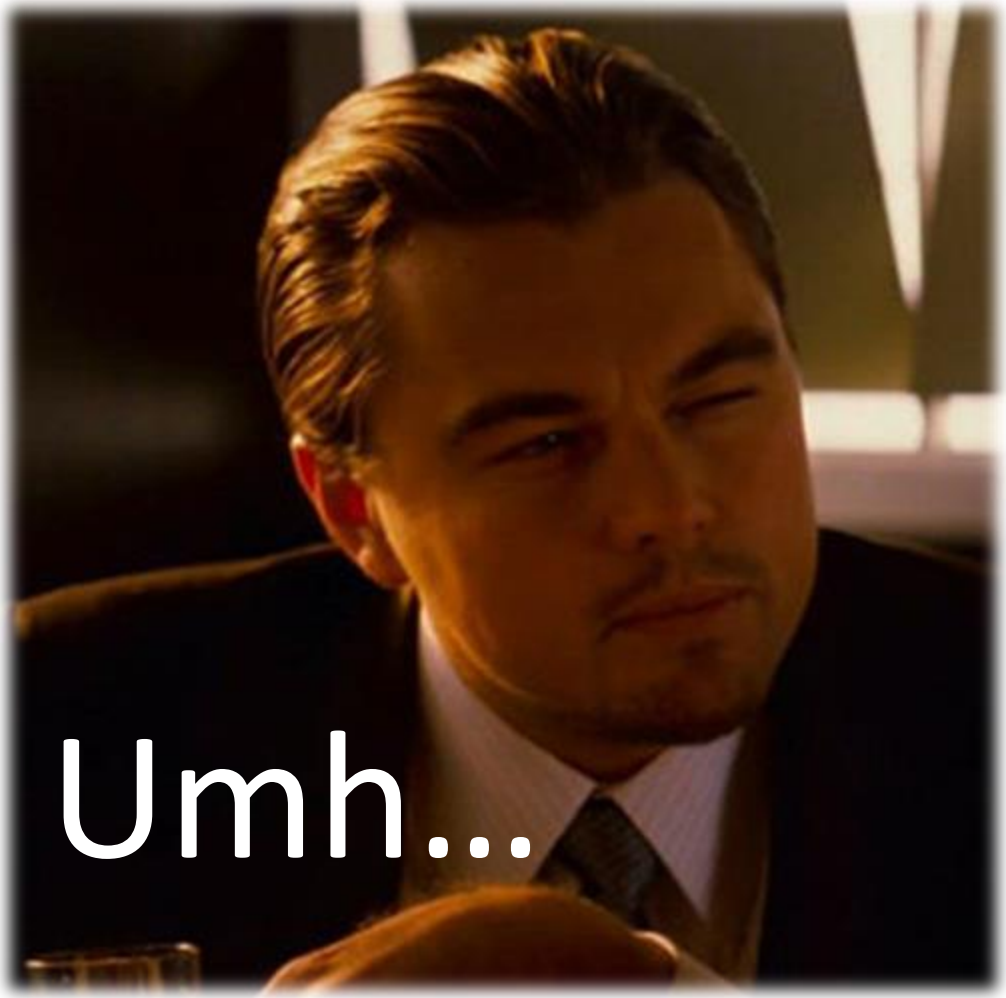


Adaptive Solution



Nice idea, but

Nice idea, but how to use it!

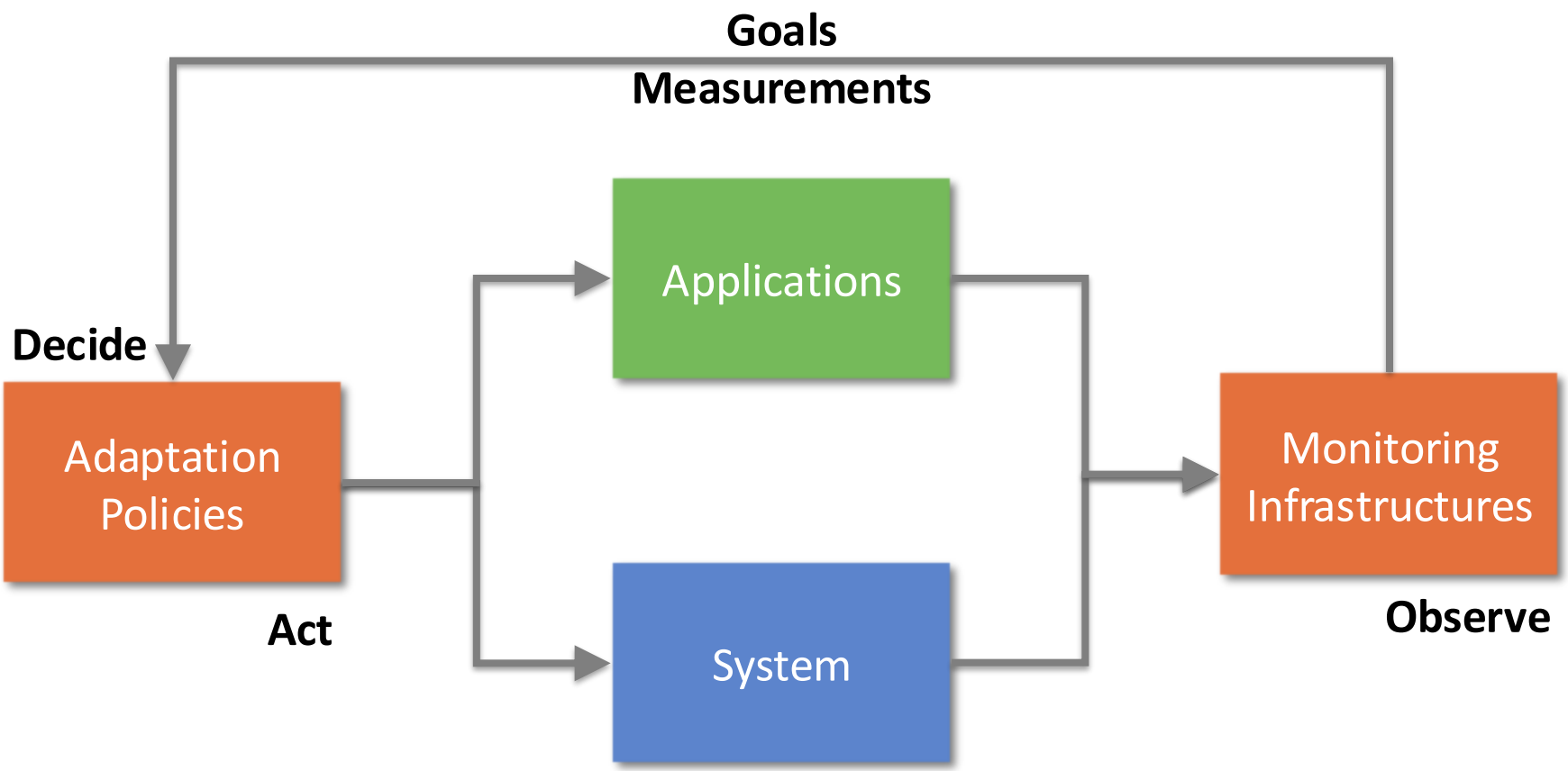


Autonomic Operating System

- The AcOS project aims at
  - designing and prototyping a patch for commodity operating systems (e.g. Linux, FreeBSD)
  - being capable to observe its own execution and optimize, in a self-aware manner, its behavior with respect to the external environment, to user needs and to applications demands

[integrated/used in different research projects]

AcOS: via an intelligent ODA loop



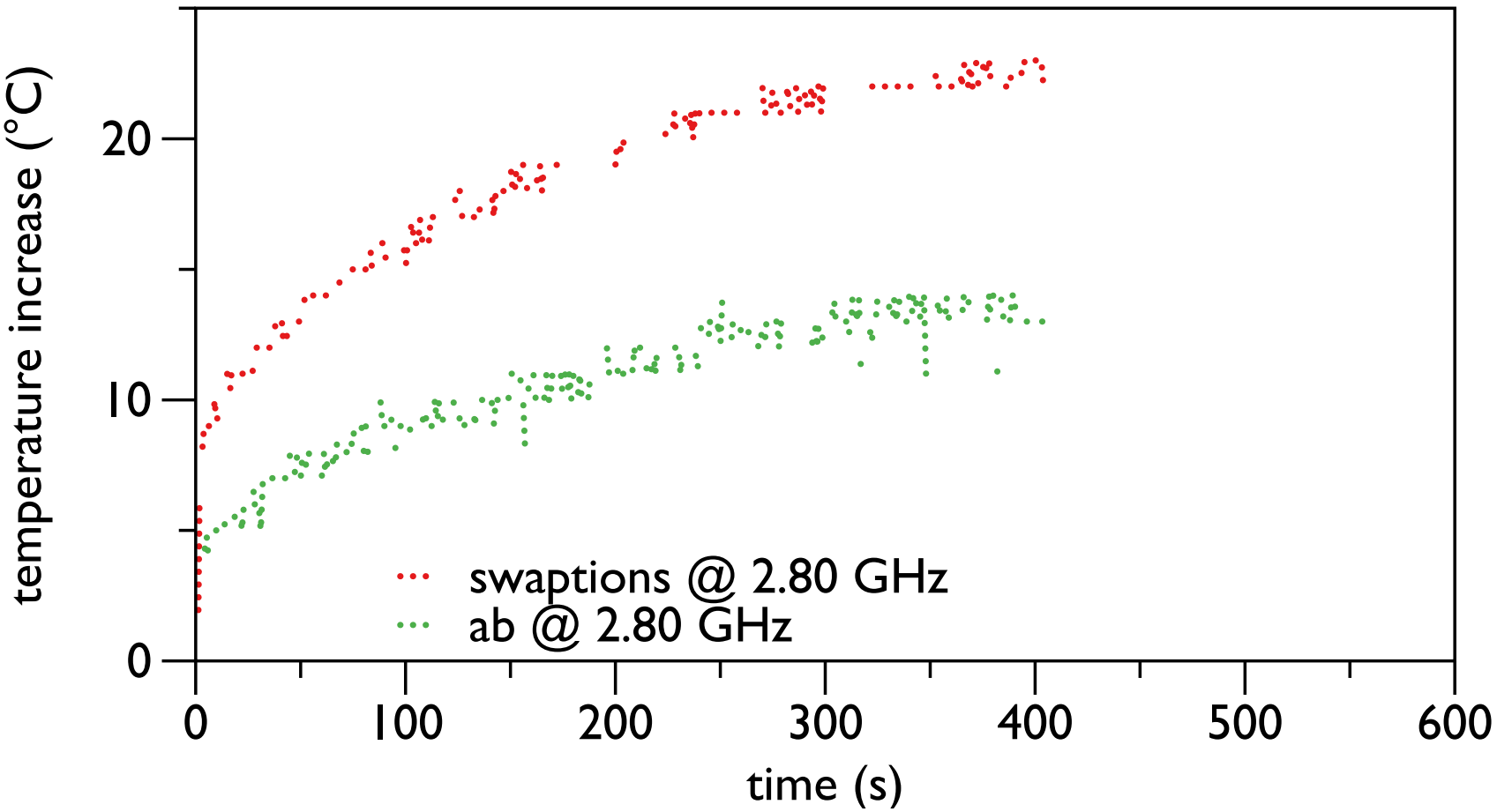
Trying to raise the bar (again)

- AcOS took into consideration performance…
- Any other *HOT* topic?

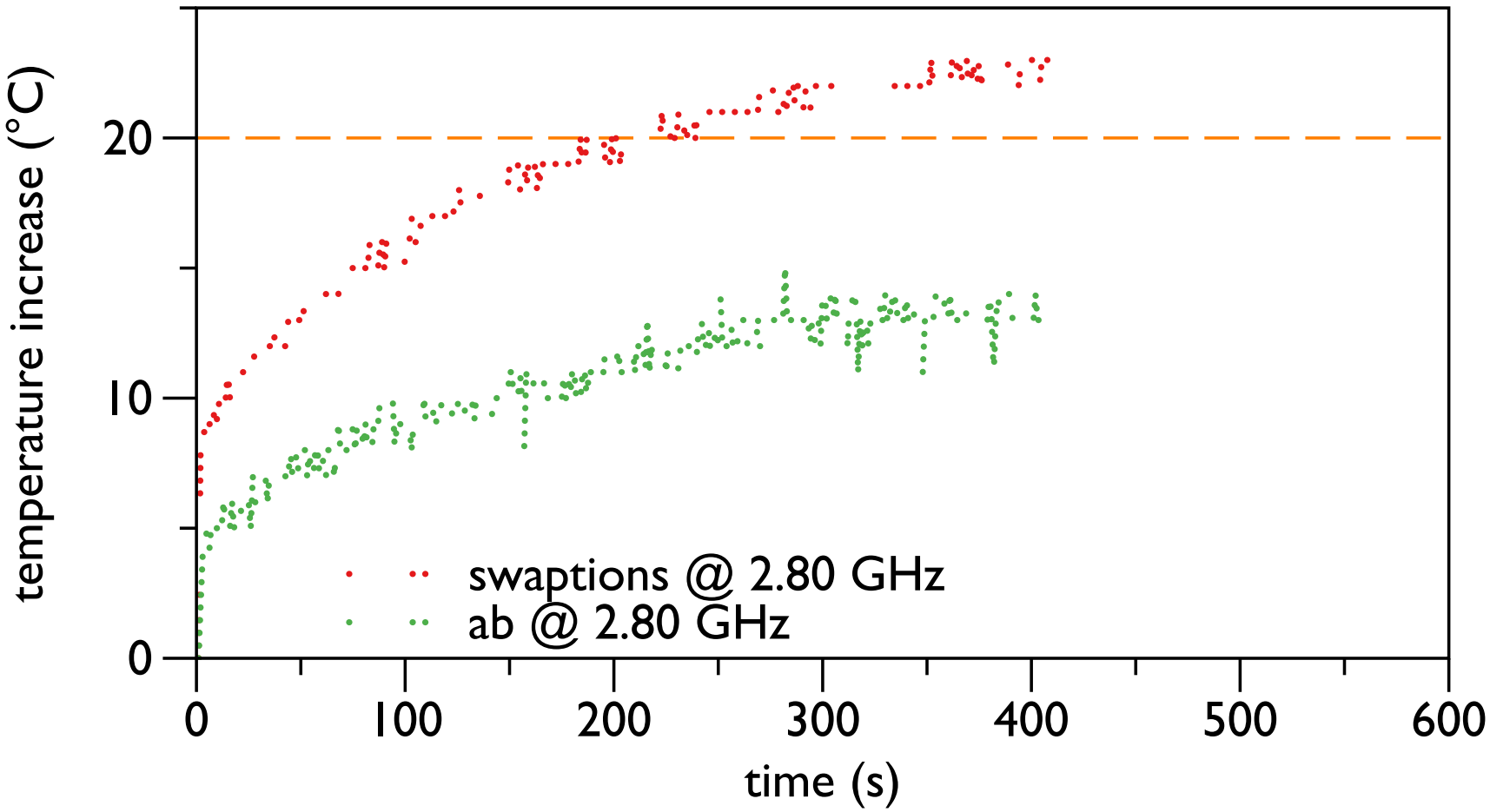
Trying to raise the bar (again)

- AcOS took into consideration performance…
- Any other *HOT* topic?
  - What about temperature Control/Management!

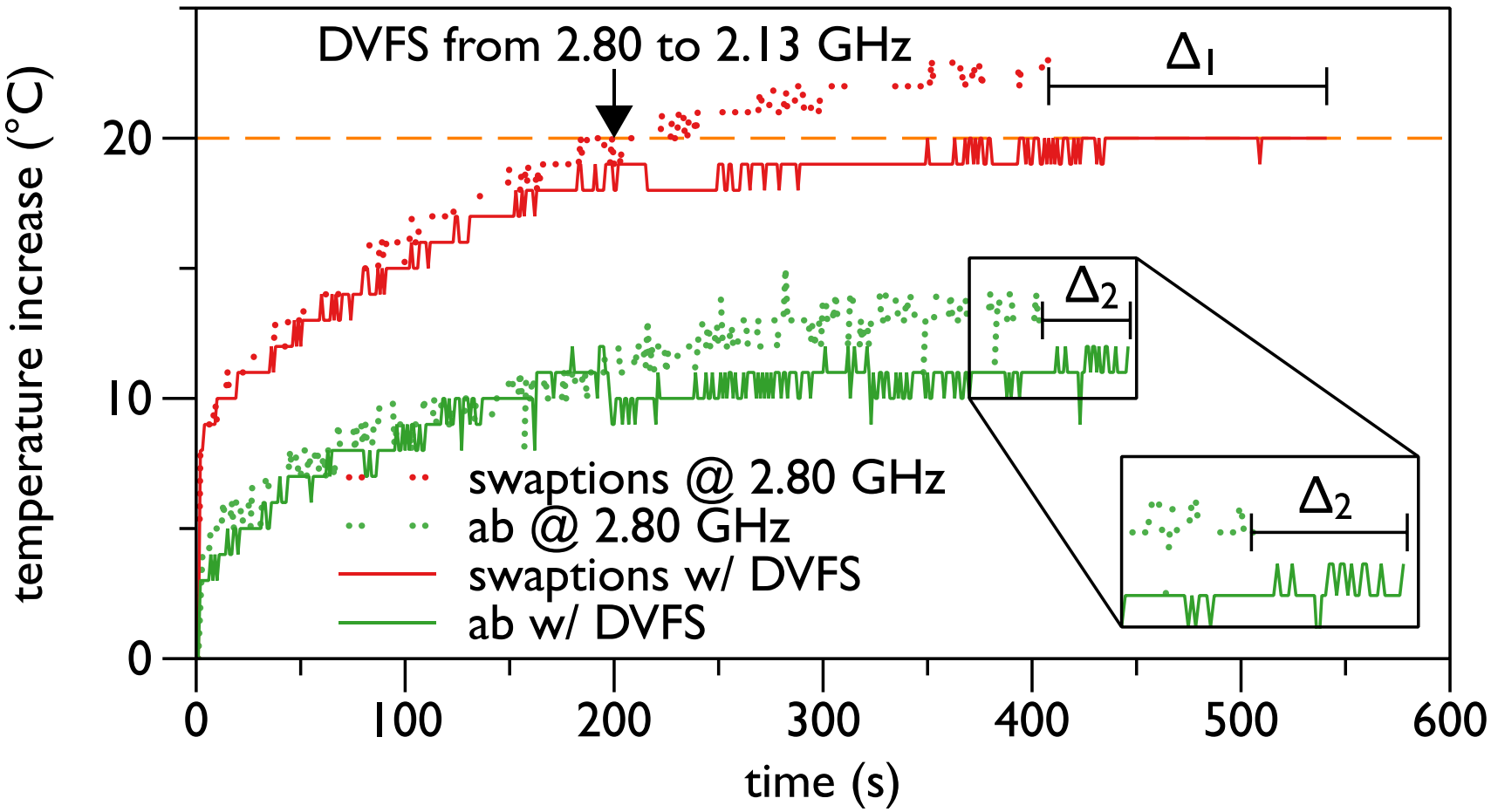
Temperature Control/Management starting scenario



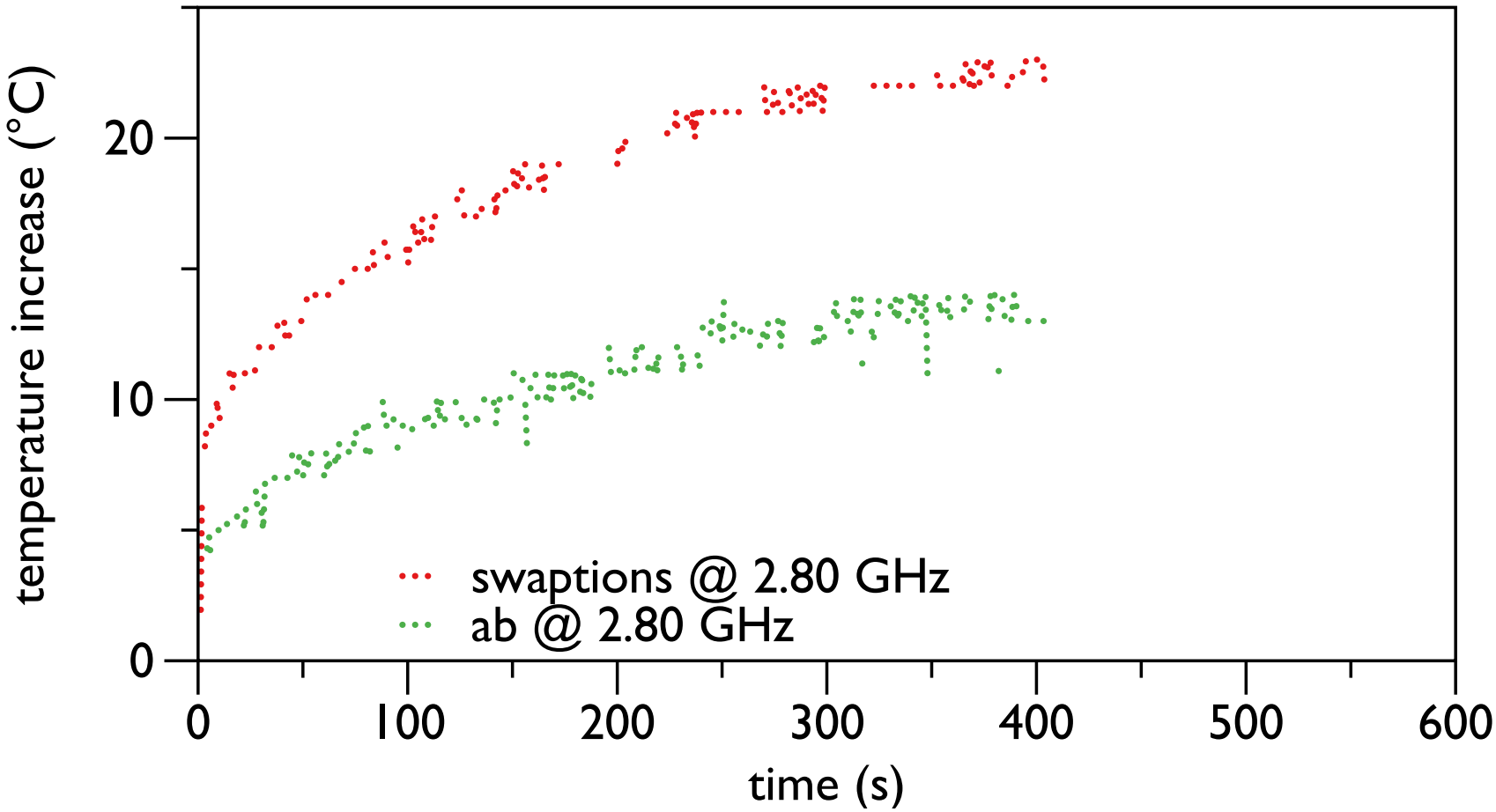
Temperature Control/Management set a temperature cap



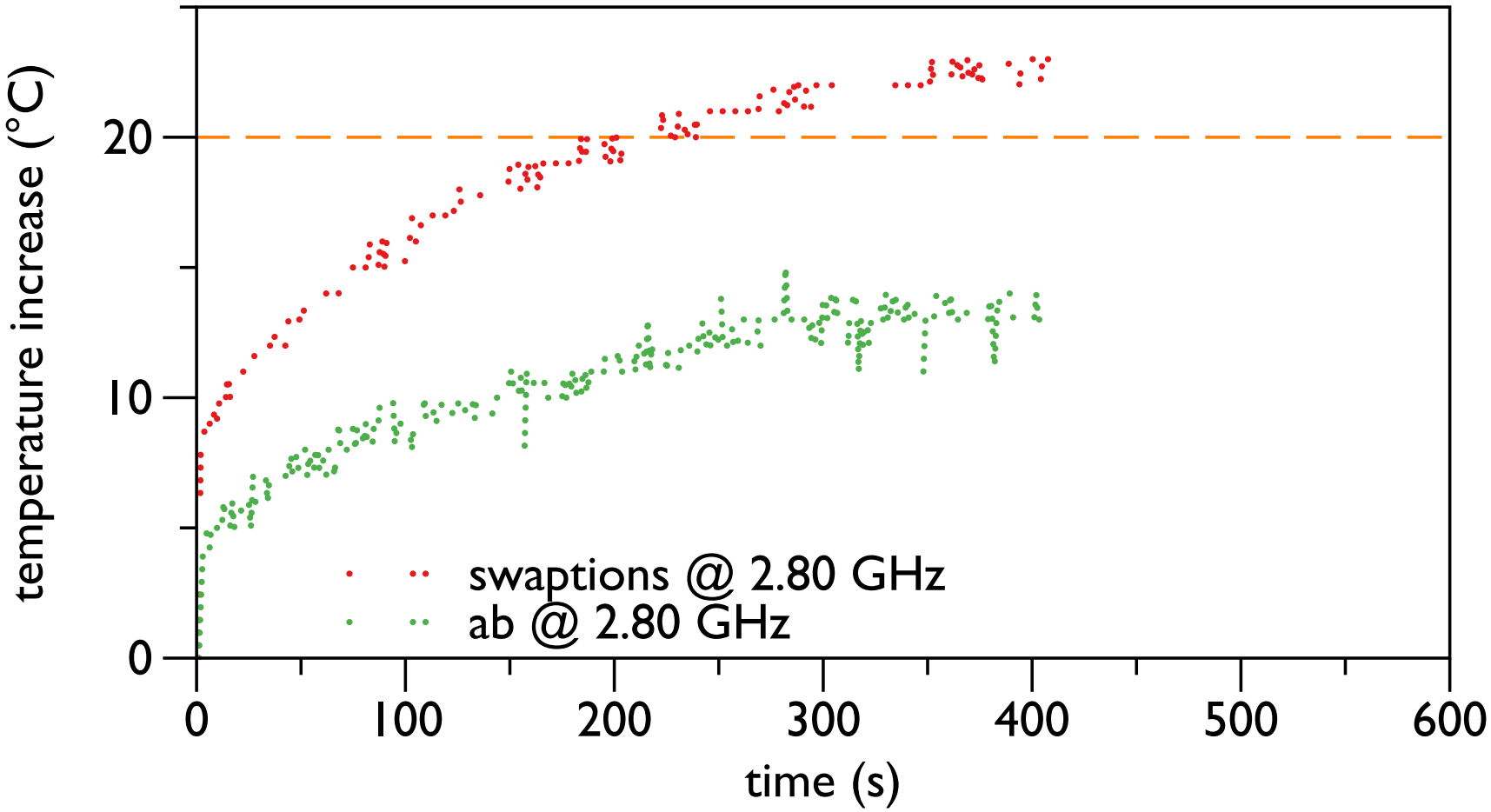
Temperature Control/Management DVFS is dangerous



Temperature Control/Management back to the starting scenario

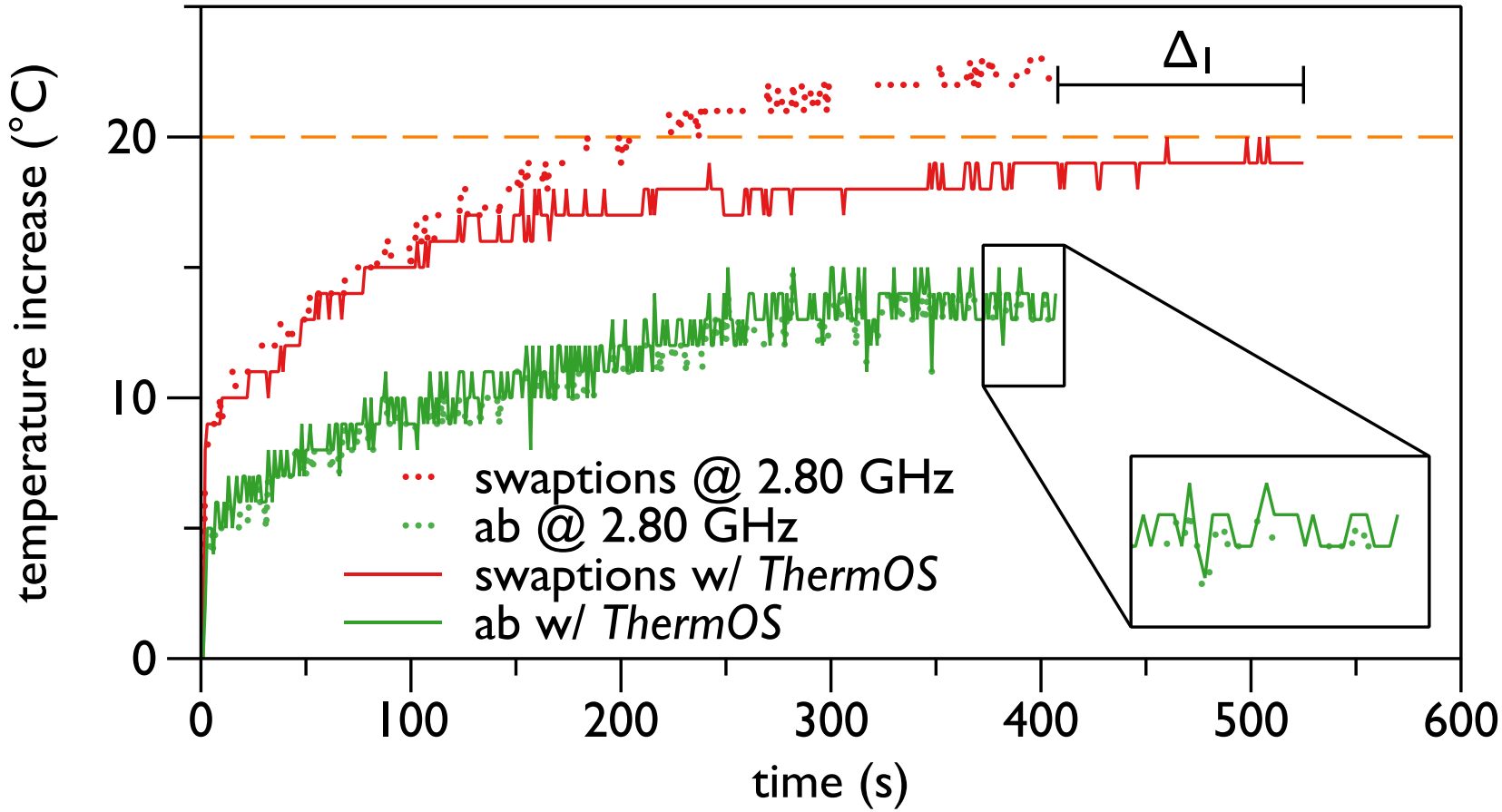


Temperature Control/Management set the same temperature cap



Temperature Control/Management ThermOS*

* F. Sironi, M.Maggio, R.Cattaneo, G.Francesco Del Nero, D. Sciuto, and M. D.Santambrogio. 2013. ThermOS: system support for dynamic thermal management of chip multi-processors. In *Proceedings of the 22nd international conference on Parallel architectures and compilation techniques* (*PACT '13*).



TODAY...

https://flic.kr/p/8puKvo

Problem

- Consider two CPUs: CPU1 e CPU2.
- CPU1 has clock cycle of 2 ns while CPU2 has an operating frequency of 700MHz.
- Given the following frequencies of occurrence of the instructions for the two CPUs

Operation type	Frequency	CPU1	CPU2
A	0.3	2	2
B	0.1	3	3
C	0.2	4	3
D	0.3	2	2
E	0.1	4	3

Questions

a. Compute the average CPI for CPU1 and CPU2
b. Which is the fastest CPU?
c. Assume that CPU1 is modified in such a way that:
   - All instructions require 7.5 clock cycles;
   - Memory stalls are not considered;
   - The miss penalty is 6 clock cycles

   Assuming a miss rate of 13% and assuming 3 memory references on average for each instruction determine the impact on performance of the cache with respect to an ideal cache (100% hit)?
d. What about the performance in the case of no cache, starting from the CPU1 of case (c)?

Solution – (a)

- Compute the average CPI for CPU1 and CPU2

Operation type	Frequency	CPU1	CPU2
A	0.3	2	2
B	0.1	3	3
C	0.2	4	3
D	0.3	2	2
E	0.1	4	3

Recall: $\text{CPI} = \dfrac{\text{Clock cycles}}{\text{Instruction}}$ $\text{CPI} = \sum_{i=1}^{n} \text{CPI}_i * F_i$ where $F_i = \dfrac{I_i}{\text{Instruction Count}}$

Solution – (a)

- Compute the average CPI for CPU1 and CPU2

Operation type	Frequency	CPU1	CPU2
A	0.3	2	2
B	0.1	3	3
C	0.2	4	3
D	0.3	2	2
E	0.1	4	3

Recall: $\text{CPI} = \frac{\text{Clock cycles}}{\text{Instruction}}$ $\text{CPI} = \sum_{i=1}^{n} \text{CPI}_i * F_i$ where $F_i = \frac{I_i}{\text{Instruction Count}}$

Solution – (a)

- Compute the average CPI for CPU1 and CPU2

Operation type	Frequency	CPU1	CPU2
A	0.3	2	2
B	0.1	3	3
C	0.2	4	3
D	0.3	2	2
E	0.1	4	3

CPI–CPU1 = 0.3*2 + 0.1*3 + 0.2*4 + 0.3*2 + 0.1*4 = 0.6+0.3+0.8+0.6+0.4 = **2.7**

Solution – (a)

- Compute the average CPI for CPU1 and CPU2

Operation type	Frequency	CPU1	CPU2
A	0.3	2	2
B	0.1	3	3
C	0.2	4	3
D	0.3	2	2
E	0.1	4	3

CPI–CPU1 = 0.3*2 + 0.1*3 + 0.2*4 + 0.3*2 + 0.1*4 = 0.6+0.3+0.8+0.6+0.4 = **2.7**

CPI–CPU2 = 0.3*2 + 0.1*3 + 0.2*3 + 0.3*2 + 0.1*3 = 0.6+0.3+0.6+0.6+0.3 = **2.4**

Solution – (b)

- Which is the fastest CPU?

Solution – (b)

- Which is the fastest CPU?

Recall: **"X is n times faster than Y" means**

$$\frac{Performance(X)}{Performance(Y)} = \frac{Exe(Y)}{Exe(X)}$$

$$\text{CPU time} = \left(\sum_{i=1}^{n} \text{IC}_i \times \text{CPI}_i \right) \times \text{Clock cycle time}$$

Solution – (b)

- Which is the fastest CPU?

- Recall… Exe-CPUx = (IC * CPIx)/Freqx
- Data:
  - CPI1= 2.7, CPI2 = 2.4
  - Freq1 = 500MHz , Freq2 = 700MHz

Solution – (b)

- Which is the fastest CPU?

- Recall… Exe-CPUx = (IC * CPIx)/Freqx
- Data:
  - CPI1= 2.7, CPI2 = 2.4
  - Freq1 = 500MHz , Freq2 = 700MHz

- Speedup = Exe-CPU1/Exe-CPU2 =

Solution – (b)

- Which is the fastest CPU?

- Recall… Exe-CPUx = (IC * CPIx)/Freqx
- Data:
  - CPI1= 2.7, CPI2 = 2.4
  - Freq1 = 500MHz , Freq2 = 700MHz

- Speedup = Exe-CPU1/Exe-CPU2 =
  = ((IC * CPI1)/Freq1)*(Freq2/(IC * CPI2)) =

Solution – (b)

- Which is the fastest CPU?

- Recall… Exe-CPUx = (IC * CPIx)/Freqx
- Data:
  - CPI1= 2.7, CPI2 = 2.4
  - Freq1 = 500MHz , Freq2 = 700MHz

- Speedup = Exe-CPU1/Exe-CPU2 =
  = ((IC * CPI1)/Freq1)*(Freq2/(IC * CPI2)) =
  = (IC * CPI1 * Freq2)/(IC * CPI2 * Freq1) =

Solution – (b)

- Which is the fastest CPU?

- Recall… Exe-CPUx = (IC * CPIx)/Freqx
- Data:
  - CPI1= 2.7, CPI2 = 2.4
  - Freq1 = 500MHz , Freq2 = 700MHz

- Speedup = Exe-CPU1/Exe-CPU2 =
  = ((IC * CPI1)/Freq1)*(Freq2/(IC * CPI2)) =
  = (IC * CPI1 * Freq2)/(IC * CPI2 * Freq1) =
  = (CPI1*Freq2)/(CPI2 * Freq1) =

Solution – (b)

- Which is the fastest CPU?

- Recall… Exe-CPUx = (IC * CPIx)/Freqx
- Data:
  - CPI1= 2.7, CPI2 = 2.4
  - Freq1 = 500MHz , Freq2 = 700MHz

- Speedup = Exe-CPU1/Exe-CPU2 =
  = ((IC * CPI1)/Freq1)*(Freq2/(IC * CPI2)) =
  = (IC * CPI1 * Freq2)/(IC * CPI2 * Freq1) =
  = (CPI1*Freq2)/(CPI2 * Freq1) =
  = (2.7 * 700MHz)/(2.4 * 500MHz) =
  = 1890/1200 = **1.575**

Solution – (c)

- Assume that CPU1 is modified in such a way that:
  - All instructions require 7.5 clock cycles;
  - Memory stalls are not considered;
  - The miss penalty is 6 clock cycles

  Assuming a miss rate of 13% and assuming 3 memory references on average for each instruction determine the impact on performance of the cache with respect to an ideal cache (100% hit)?

Solution – (c)

- Assume that CPU1 is modified in such a way that:
  - All instructions require 7.5 clock cycles;
  - Memory stalls are not considered;
  - The miss penalty is 6 clock cycles

  Assuming a miss rate of 13% and assuming 3 memory references on average for each instruction determine the impact on performance of the cache with respect to an ideal cache (100% hit)?

Solution – (c)

- The question
  - All instructions require 7.5 clock cycles;
  - The miss penalty is 6 clock cycles
  - Assuming a miss rate of 13% and assuming 3 memory references on average for each instruction

- Became
  - CPI_exe = 7.5
  - MISS Penalty = 6
  - MISS Rate = 0.13
  - Memory references = 3

Solution – (c)

- CPI_exe = 7.5
- MISS Penalty = 6
- MISS Rate = 0.13
- Memory references = 3

- CPI_cache =
  = CPI_exe+(References*MISSPenalty*MISSRate) =
  = 7.5 + (3*0.13*6) = 7.5 + 2.34 = **9.84**

Solution – (c)

- CPI_exe = 7.5
- MISS Penalty = 6
- MISS Rate = 0.13
- Memory references = 3

- CPI_cache =
  = CPI_exe+(References*MISSPenalty*MISSRate) =
  = 7.5 + (3*0.13*6) = 7.5 + 2.34 = **9.84**

- CPI_ideal_cache (MISS Rate = 0) = **7.5**

Solution – (d)

- What about the performance in the case of no cache, starting from the CPU1 of case (c)?

Solution – (d)

- What about the performance in the case of no cache, starting from the CPU1 of case (c)?

- CPI_exe = 7.5
- MISS Penalty = 6
- MISS Rate = 1
- Memory references = 3

Solution – (d)

- What about the performance in the case of no cache, starting from the CPU1 of case (c)?

- CPI_exe = 7.5
- MISS Penalty = 6
- MISS Rate = 1
- Memory references = 3

- CPI_no_cache =

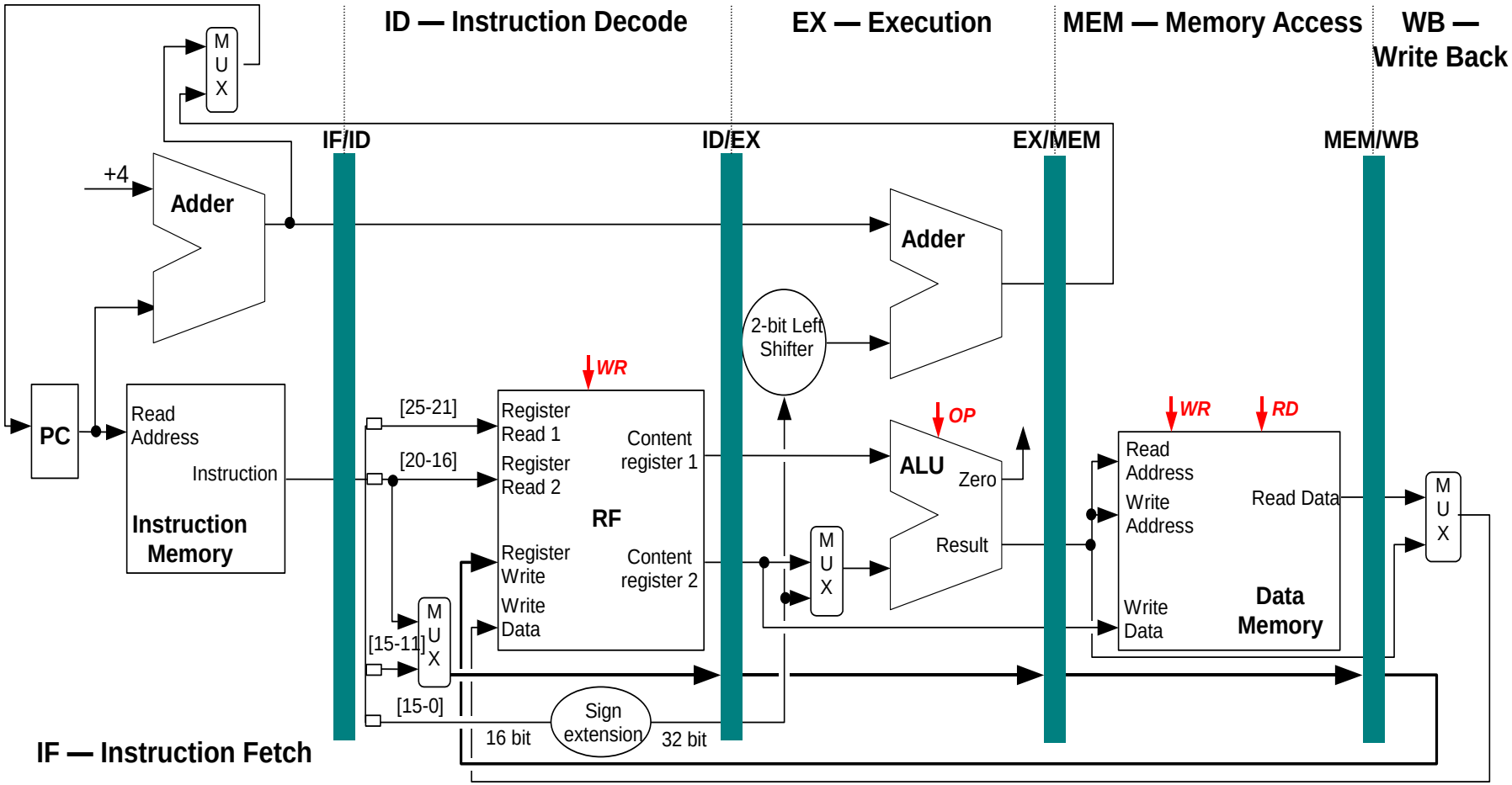
= CPI_exe+(References*MISSPenalty*MISSRate) = = 7.5 + (3*6*1) = 7.5 + 18 = **25.5**

Pipeline

- Assume that a given 5 stage pipeline MIPS architecture is working with a clock cycle equal to 2 ns. Using the MIPS code shown below,

```
I1: add $s8, $s2, 6
I2: sub $s4, $s8, $s7
I3: add $s5, $s4, $s7
I4: lw $s6, 12($s4)
I5: sub $s7, $s4, $s6
```

Recall: Implementation of MIPS pipeline



The Problem of Hazards

- A hazard is created whenever there is a dependence between instructions, and instructions are close enough that the overlap caused by pipelining would change the order of access to the operands involved in the dependence.
- Hazards prevent the next instruction in the pipeline from executing during its designated clock cycle.
- Hazards reduce the performance from the ideal speedup gained by pipelining.

Three Classes of Hazards

- Structural Hazards: Attempt to use the same resource from different instructions simultaneously
  - Example: Single memory for instructions and data
- Data Hazards: Attempt to use a result before it is ready
  - Example: Instruction depending on a result of a previous instruction still in the pipeline
- Control Hazards: Attempt to make a decision on the next instruction to execute before the condition is evaluated
  - Example: Conditional branch execution

Data Conflicts

- Draw the pipeline schema showing all the RAW data conflicts.

Istr	CK1	CK2	CK3	CK4	CK5	CK6	CK7	CK8	CK9	CK10	CK11	CK12	CK13
I1	IF	ID	EX	MEM	WB								
I2		IF	ID	EX	MEM	WB							
I3			IF	ID	EX	MEM	WB						
I4				IF	ID	EX	MEM	WB					
I5					IF	ID	EX	MEM	WB				

Type of Data Hazard

- Read After Write (RAW)
  $Instr_J$ tries to read operand before $Instr_I$ writes it

```
I: add r1,r2,r3
J: sub r4,r1,r3
```

- Caused by a "Dependence" (in compiler nomenclature). This hazard results from an actual need for communication.

Data Hazards: Possible Solutions

- Compilation Techniques:
  - Insertion of nop (no operation) instructions
  - Instructions Scheduling to avoid that correlating instructions are too close
    - The compiler tries to insert independent instructions among correlating instructions
    - When the compiler does not find independent instructions, it insert nops.
- Hardware Techniques:
  - Insertion of “bubbles” or stalls in the pipeline
  - Data Forwarding or Bypassing

Data Conflicts - Solution

I1: add $s8, $s2, 6

I2: sub $s4, $s8, $s7

I3: add $s5, $s4, $s7

I4: lw $s6, 12($s4)

I5: sub $s7, $s4, $s6

Istr	CK1	CK2	CK3	CK4	CK5	CK6	CK7	CK8	CK9	CK10	CK11	CK12	CK13
I1	IF	ID	EX	MEM	WB								
I2		IF	ID	EX	MEM	WB							
I3			IF	ID	EX	MEM	WB						
I4				IF	ID	EX	MEM	WB					
I5					IF	ID	EX	MEM	WB				

Forwarding Paths

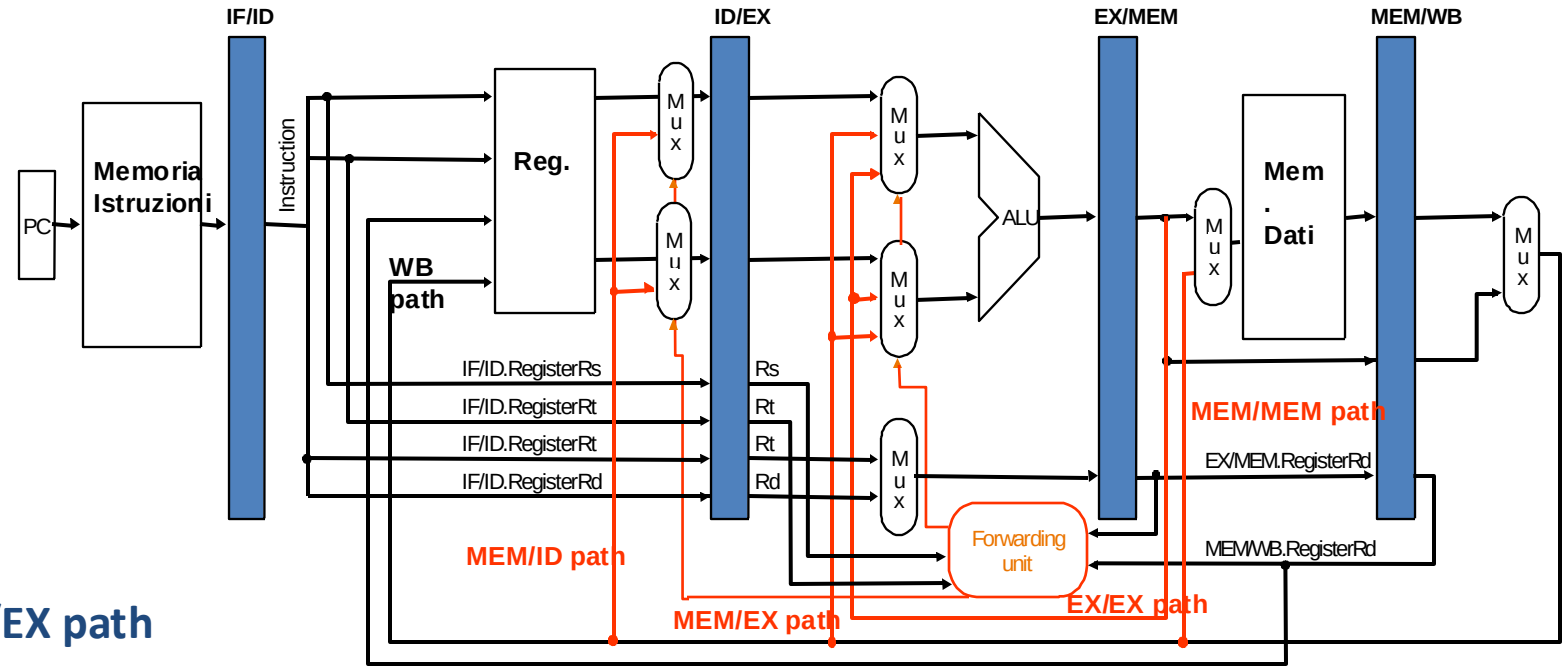
- Assume you can introduce all the possible forwarding paths.
- Draw the pipeline schema showing all the forwarding paths that are used to improve the execution time.

Recall: Forwarding

- Data forwarding uses temporary results stored in the pipeline registers instead of waiting for the write back of results in the RF.

- We need to add multiplexers at the inputs of ALU to fetch inputs from pipeline registers to avoid the insertion of stalls in the pipeline.

Recall: Implementation of MIPS with Forwarding Unit



- EX/EX path
- MEM/EX path
- MEM/ID path
- MEM/MEM path

Forwarding Paths - Solution

I1: add $s8, $s2, 6

I2: sub $s4, $s8, $s7

I3: add $s5, $s4, $s7

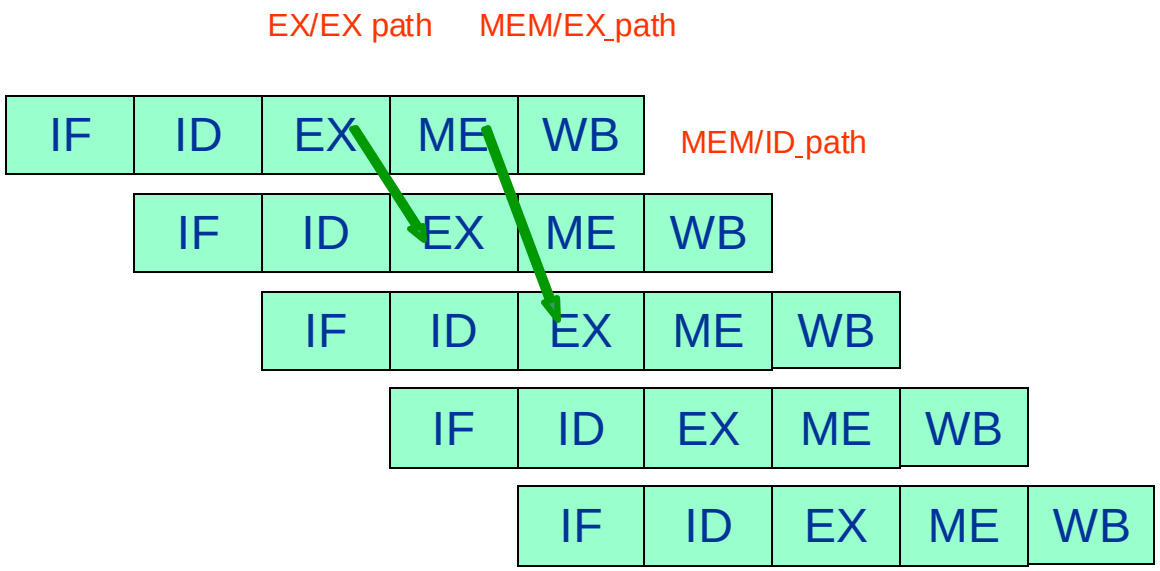
I4: lw $s6, 12($s4)

I5: sub $s7, $s4, $s6

Istr	CK1	CK2	CK3	CK4	CK5	CK6	CK7	CK8	CK9	CK10	CK11	CK12	CK13
I1													
I2													
I3													
I4													
I5													

Recall: Forwarding “an extra”

```
sub  $2, $1, $3
and $12, $2, $5
or  $13, $6, $2
add $14, $2, $2
sw  $15,100($2)
```



Reschedule

- Reschedule the instructions to reduce the stalls. Draw the pipeline schema showing all the RAW data conflicts, considering the pipeline providing all available forwarding paths

Reschedule - Solution

I1: add $s8, $s2, 6

I2: sub $s4, $s8, $s7

I3: add $s5, $s4, $s7

I4: lw $s6, 12($s4)

I5: sub $s7, $s4, $s6

Istr	CK1	CK2	CK3	CK4	CK5	CK6	CK7	CK8	CK9	CK10	CK11	CK12	CK13
I1													
I2													
I3													
I4													
I5													

QUESTIONS?

Alessandro Verosimile <Alessandro.verosimile@polimi.it>
Marco D. Santambrogio <marco.santambrogio@polimi.it>