An Adaptive On-line CPU-GPU Governor for Games on Mobile Devices

Po-Kai Chuang, Ya-Shu Chen and Po-Hao Huang

Speaker: Po-Hao Huang

Department of Electrical Engineering,
National Taiwan University of Science and Technology,
No. 43, Keelung Rd. Sec. 4, Taipei 10607, Taiwan (R.O.C.)

OUTLINE

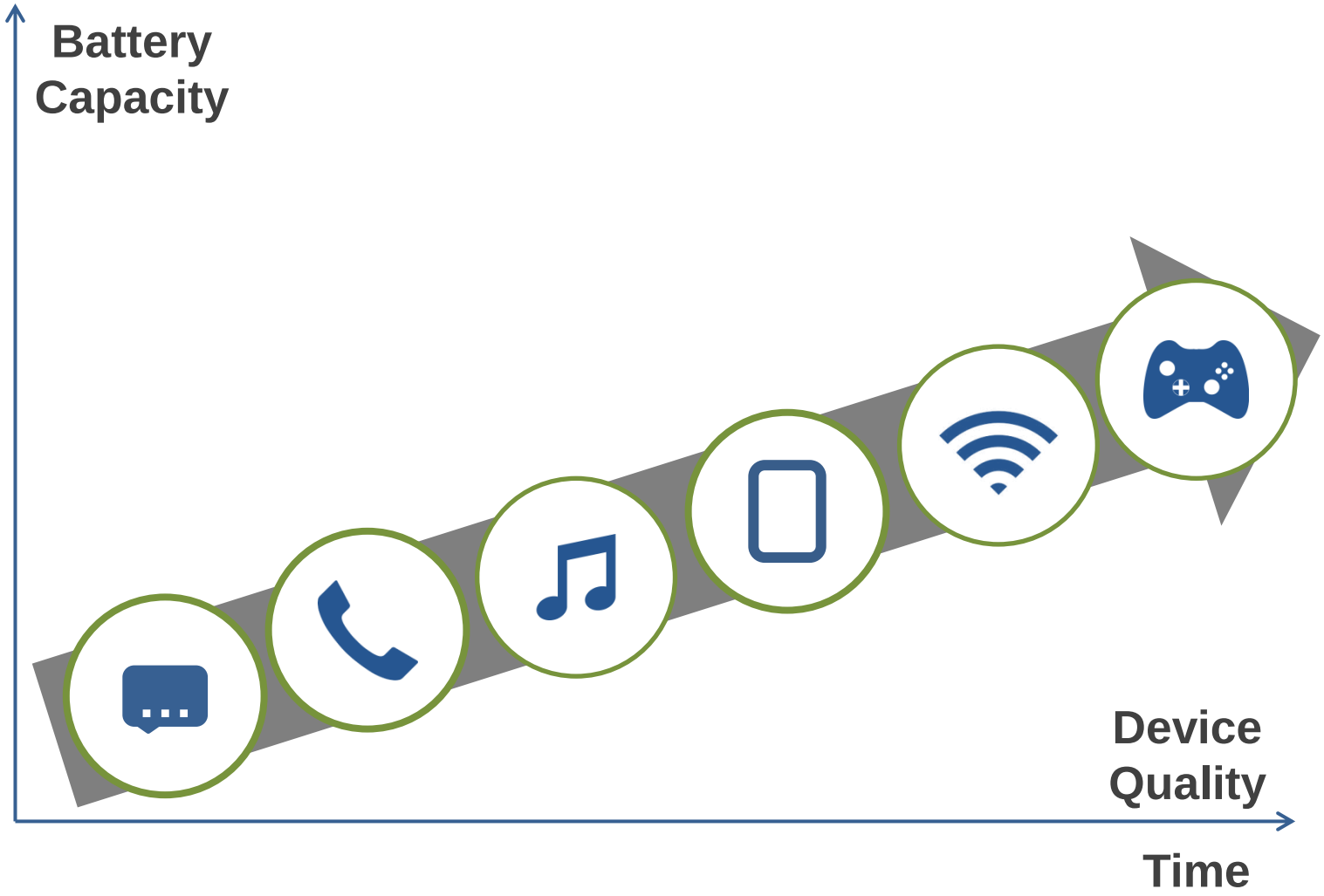
What is the problem

Which should be concerned

How to make it

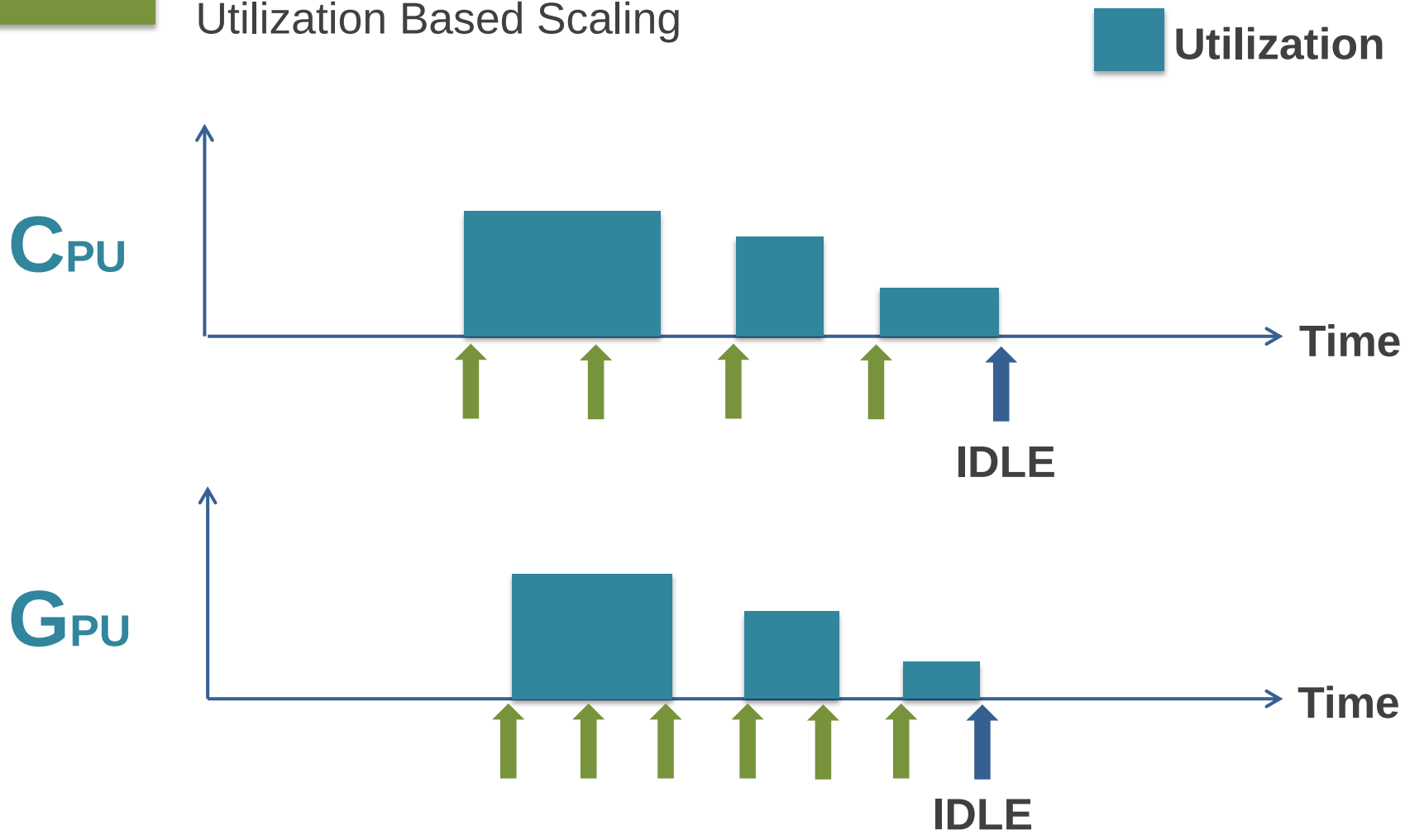
INTRODUCTION

Phone Using History



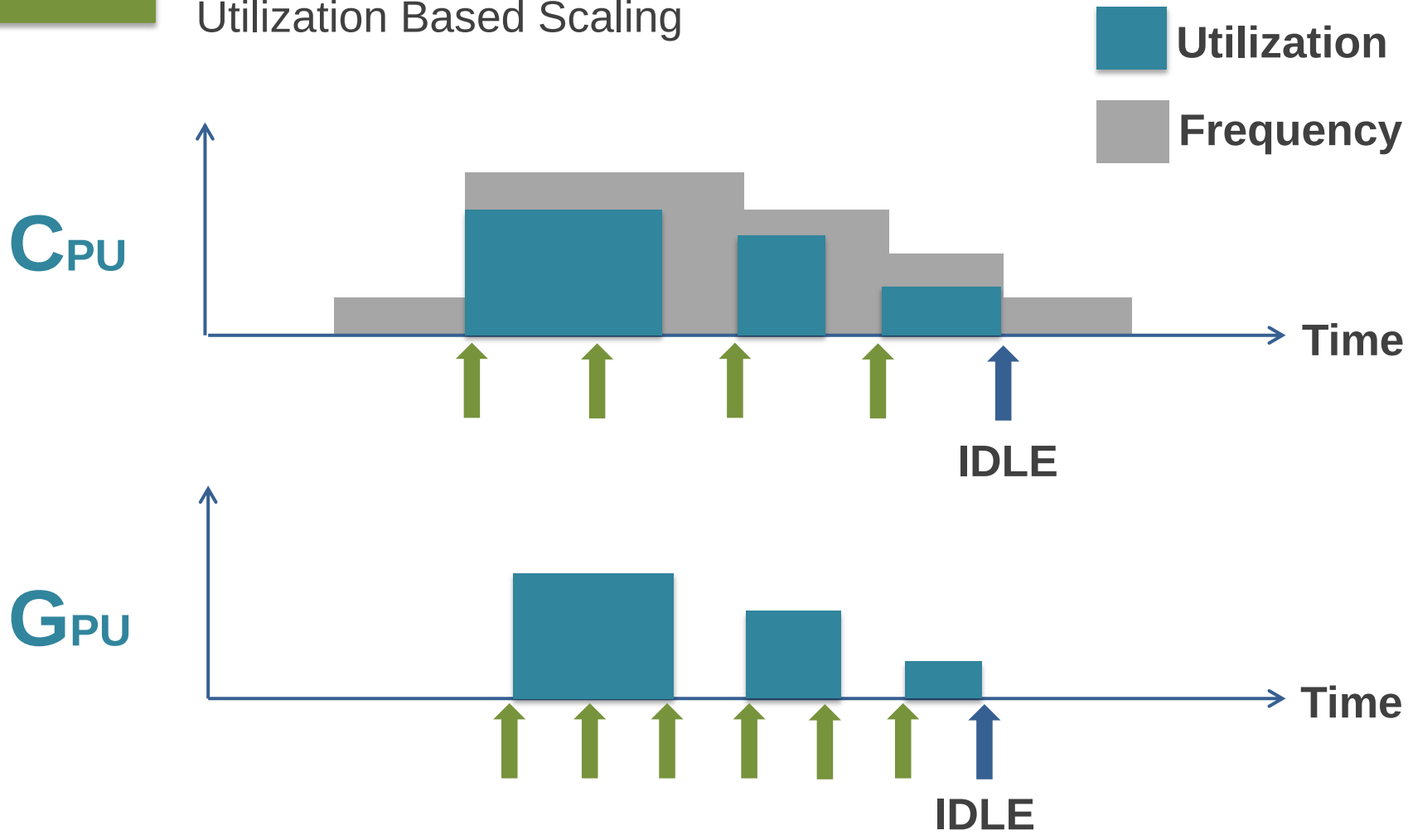
ANDROID SCALING

Utilization Based Scaling



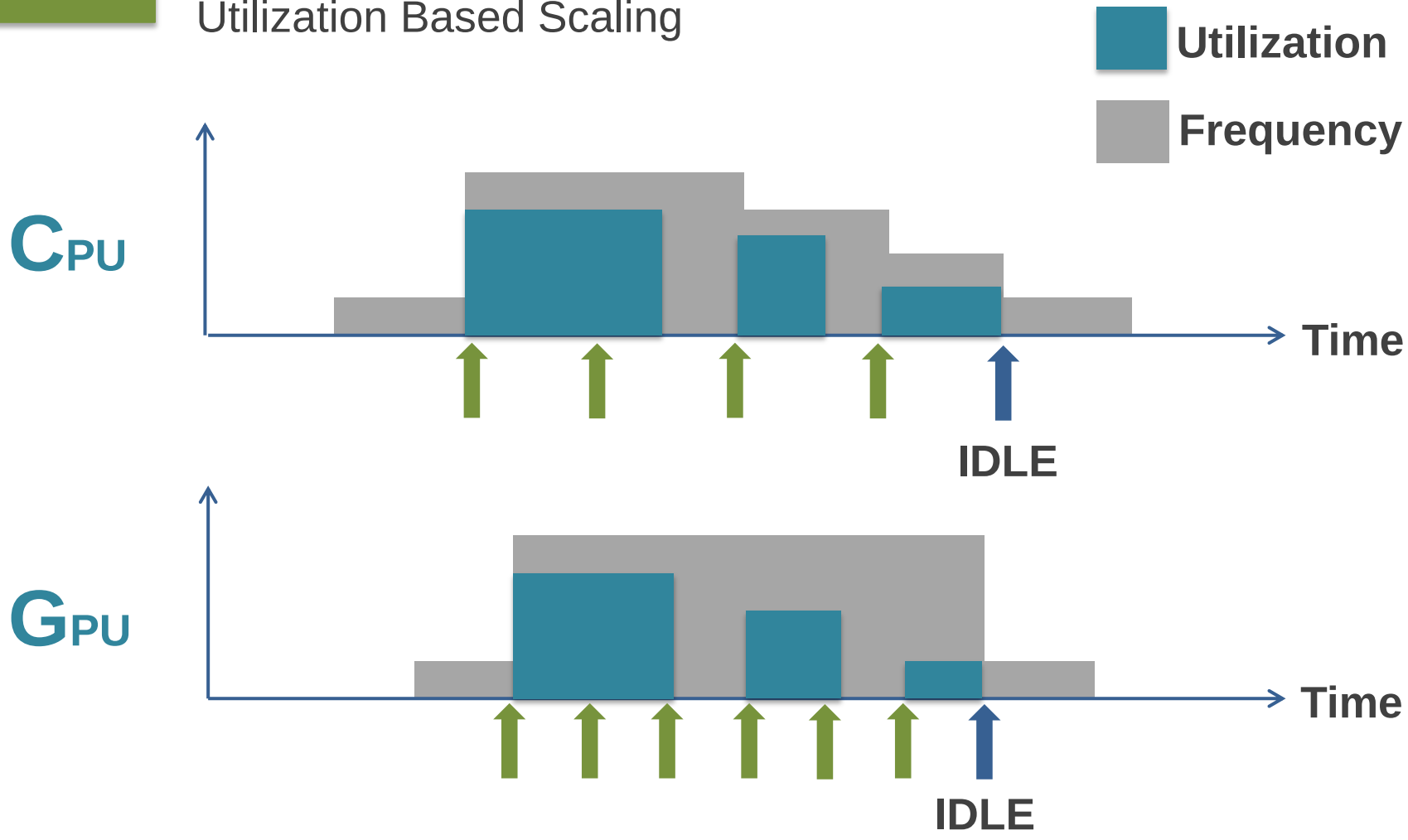
ANDROID SCALING

Utilization Based Scaling

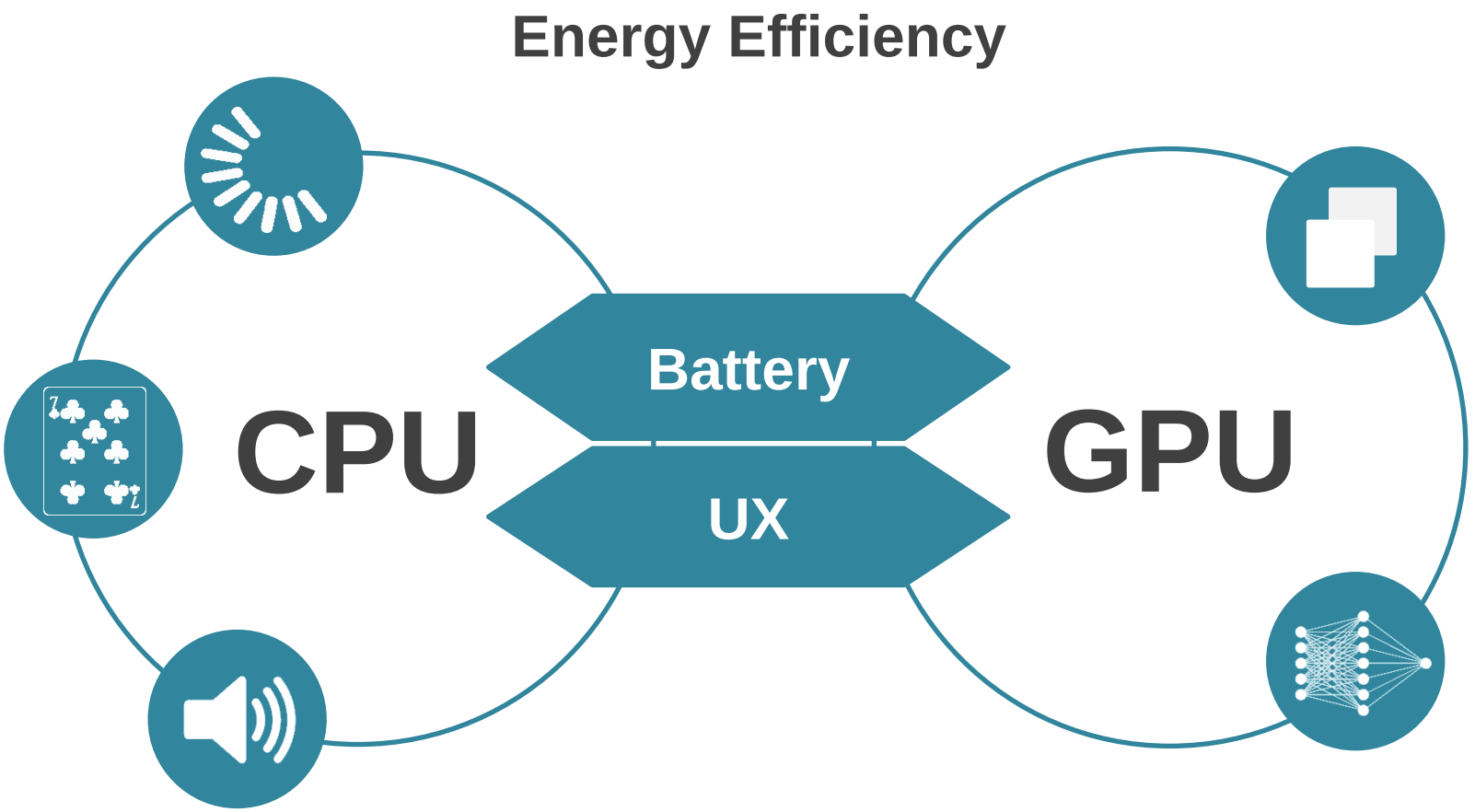


ANDROID SCALING

Utilization Based Scaling

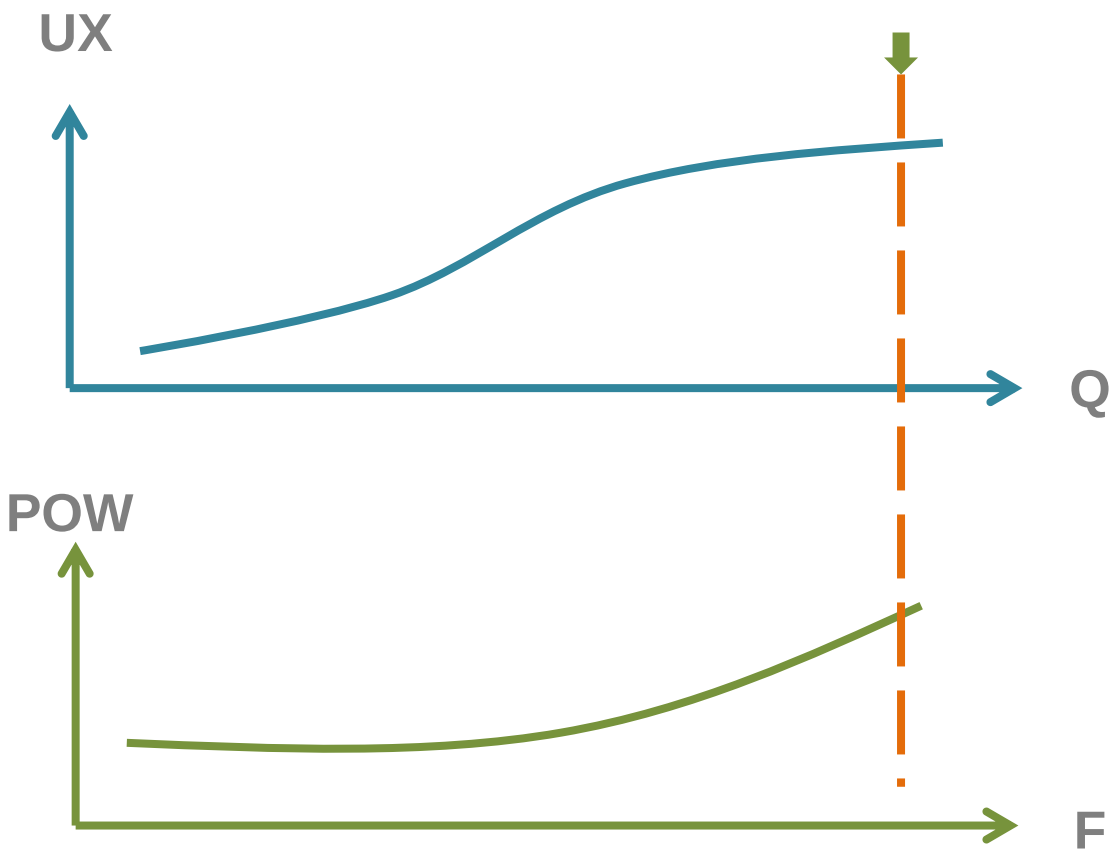


PROBLEM DEFINITION



PROBLEM DEFINITION

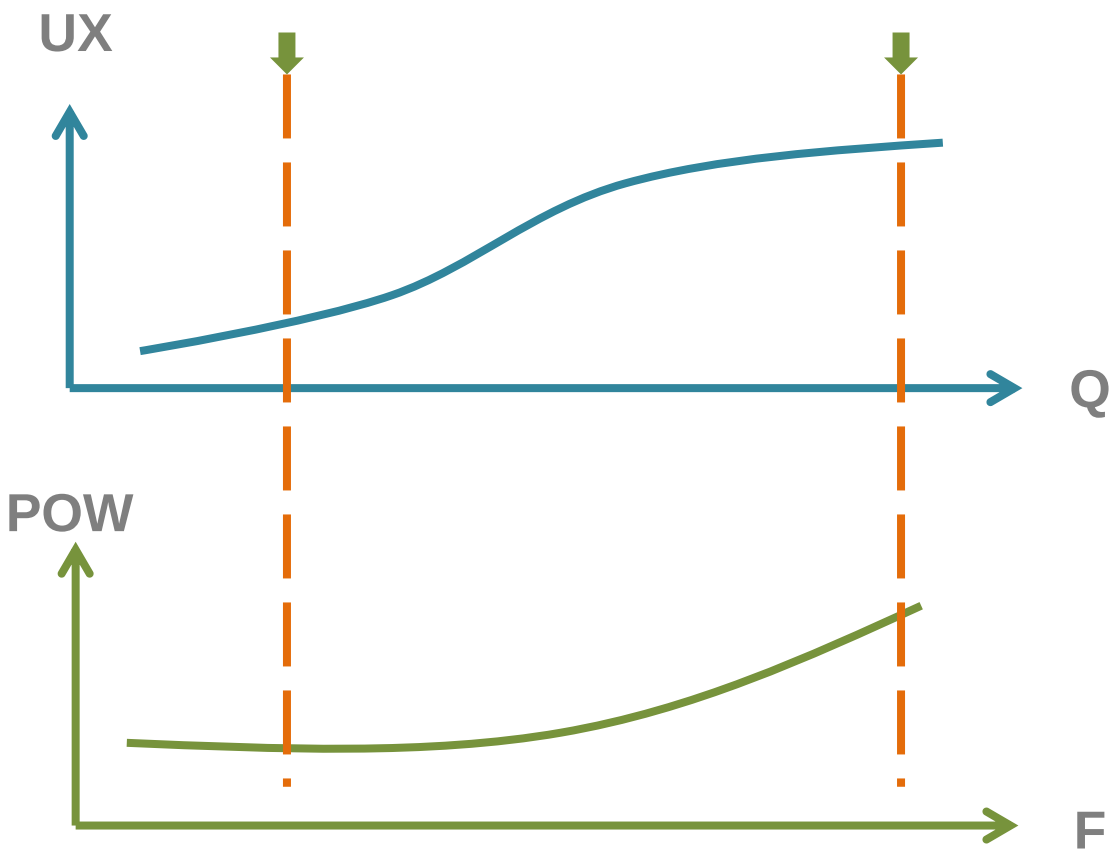
QoE Idea



Lee, Yeng-Ting, et al. "Are all games equally cloud-gaming-friendly? an electromyographic approach." Network and Systems Support for Games (NetGames), 2012 11th Annual Workshop on. IEEE, 2012.

PROBLEM DEFINITION

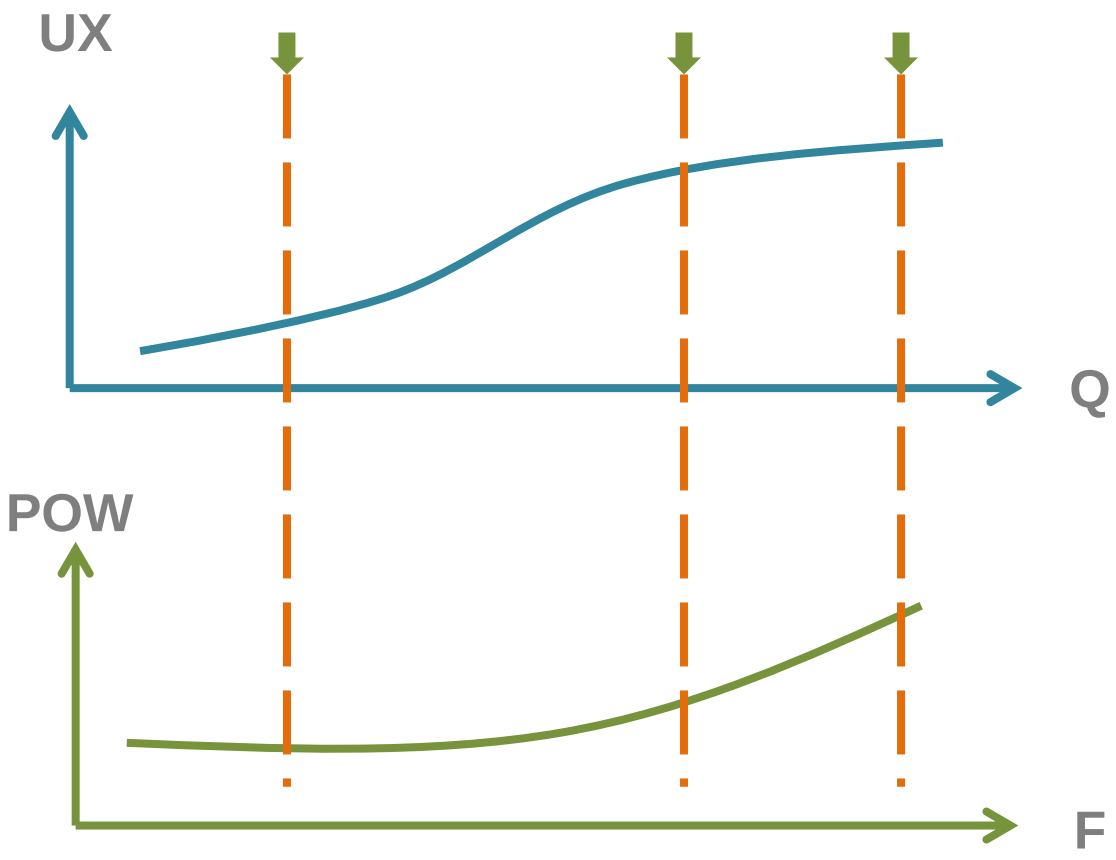
QoE Idea



Lee, Yeng-Ting, et al. "Are all games equally cloud-gaming-friendly? an electromyographic approach." Network and Systems Support for Games (NetGames), 2012 11th Annual Workshop on. IEEE, 2012.

PROBLEM DEFINITION

QoE Idea



Lee, Yeng-Ting, et al. "Are all games equally cloud-gaming-friendly? an electromyographic approach." Network and Systems Support for Games (NetGames), 2012 11th Annual Workshop on. IEEE, 2012.

RELATED WORKS

CPU + GPU +UX

[1] W.-M. Chen et al

Dynamic GPU-CPU scaling governor selecting for interactive game

[2] A. Pathania et al

An ideal of GPU-CPU scaling governor for 3D games

[3] A. Pathania et al

A GPU-CPU scaling governor for mobile games

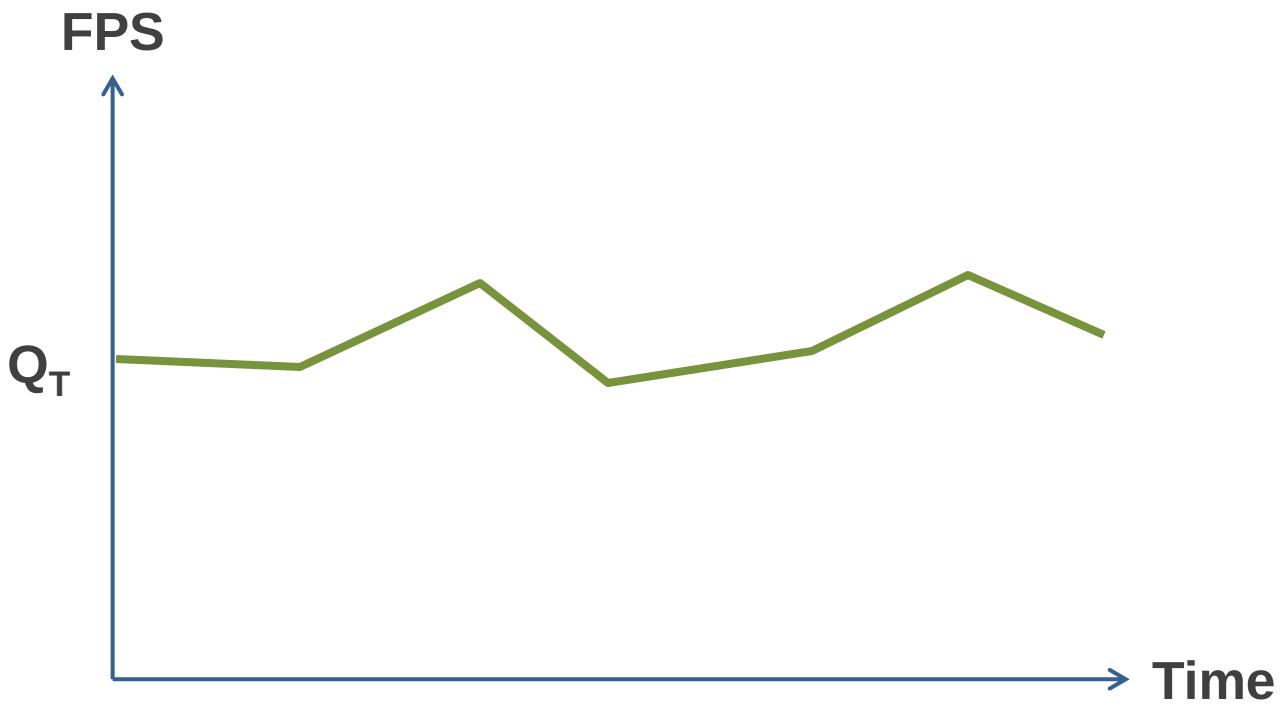
[1] W.-M. Chen, S.-W. Cheng, P.-C. Hsiu, T.-W. Kuo, A user-centric cpu-gpu governing framework for 3d games on mobile devices, in: Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, IEEE Press, 2015, pp. 224–231.

[2] A. Pathania, Q. Jiao, A. Prakash, T. Mitra, Integrated cpu-gpu power management for 3d mobile games, in: Design Automation Conference (DAC), 2014 51st ACM/EDAC/IEEE, IEEE, 2014, pp. 1–6.

[3] Pathania, Anuj, et al. "Power-performance modelling of mobile gaming workloads on heterogeneous MPSoCs." *Proceedings of the 52nd Annual Design Automation Conference*. ACM, 2015.

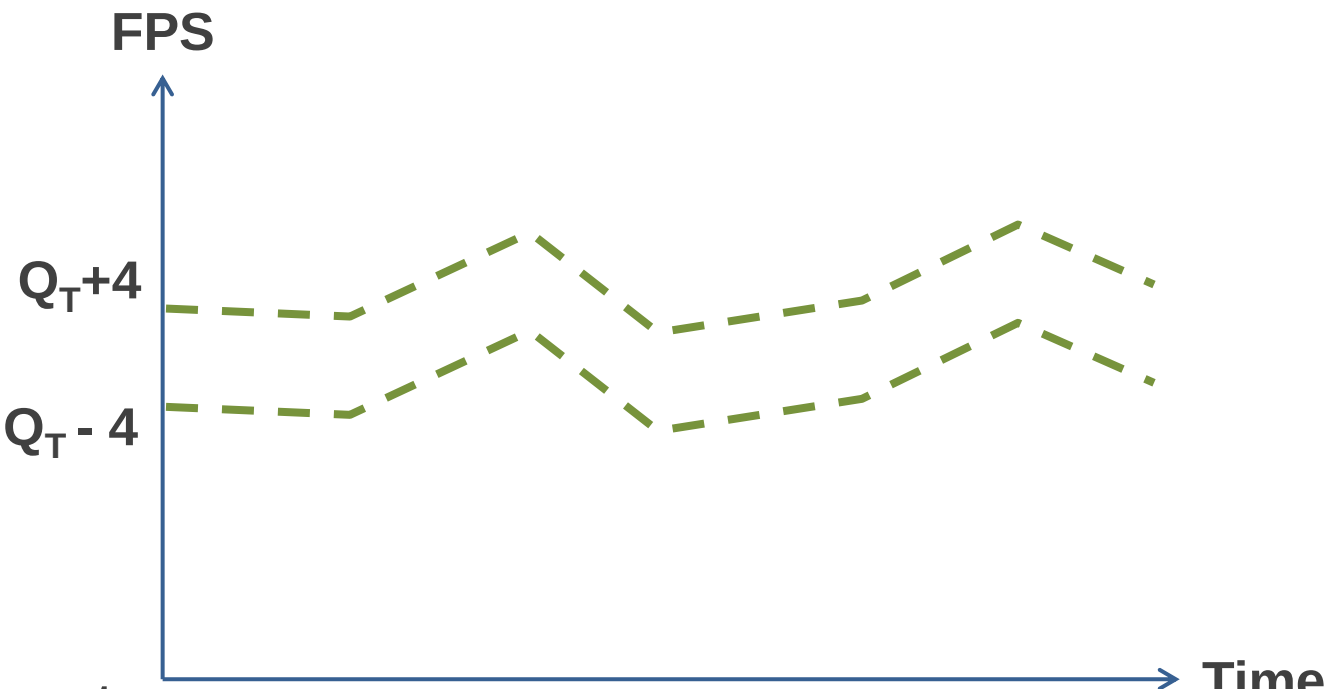
ONLINE SCALING ALGORITHM

Concept : Quality achievement



ONLINE SCALING ALGORITHM

Concept : Quality achievement



Quality Target

The current FPS meets UX if the difference between the current FPS and the target is less than the tolerance (e.g. 4).

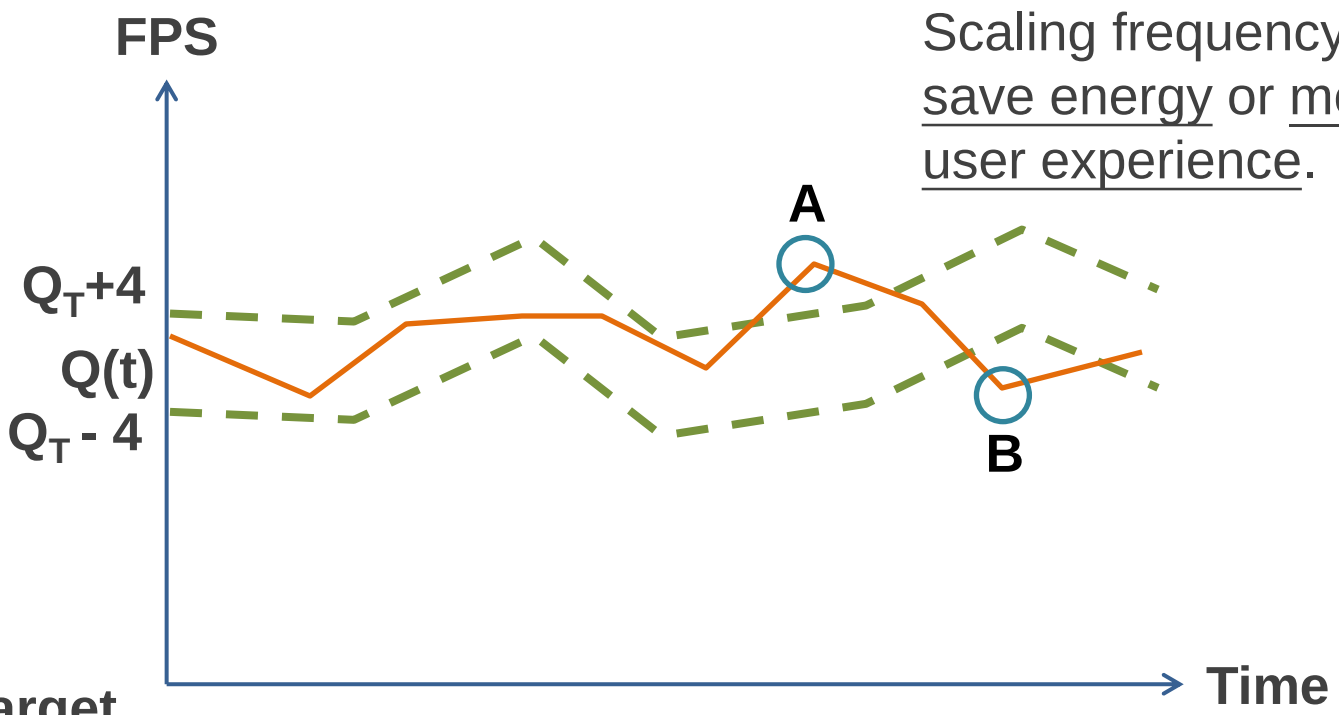
Setton, Eric, Jeonghun Noh, and Bernd Girod. "Rate-distortion optimized video peer-to-peer multicast streaming." Proceedings of the ACM workshop on Advances in peer-to-peer multimedia streaming. ACM, 2005.

ONLINE SCALING ALGORITHM

Concept : Quality achievement

Q_T Scaling

Scaling frequency to save energy or meet user experience.



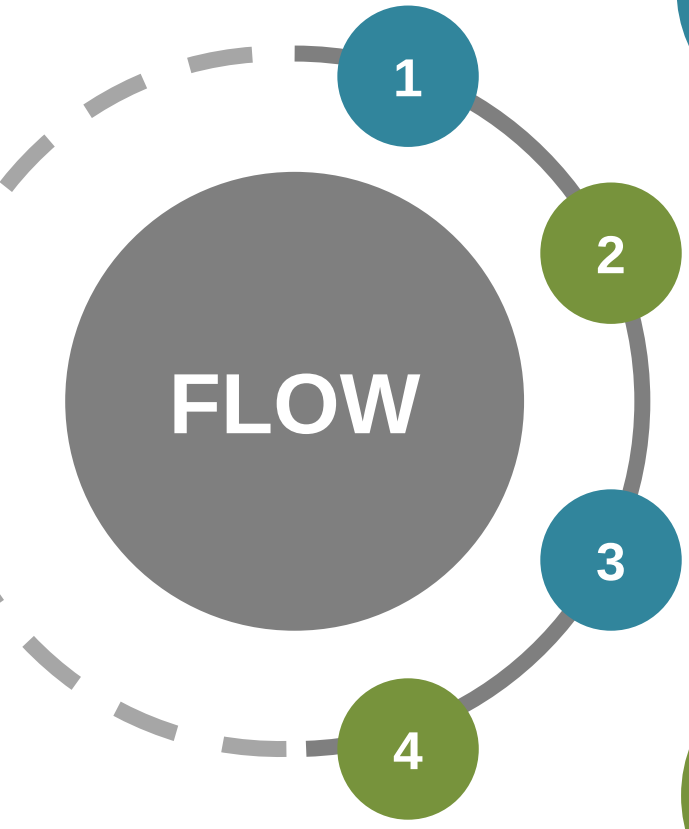
Quality Target

The current FPS meets UX if the difference between the current FPS and the target is less than the tolerance (e.g. 4).

Setton, Eric, Jeonghun Noh, and Bernd Girod. "Rate-distortion optimized video peer-to-peer multicast streaming." Proceedings of the ACM workshop on Advances in peer-to-peer multimedia streaming. ACM, 2005.

ONLINE SCALING ALGORITHM

Flow



Step1

SAMPLING

To get system **status** and the value of **targets**

Step2

DECIDING

To decide whether to **scale** or not

Step3

LEARNING

To modify the **coefficients** by a learning equation

Step4

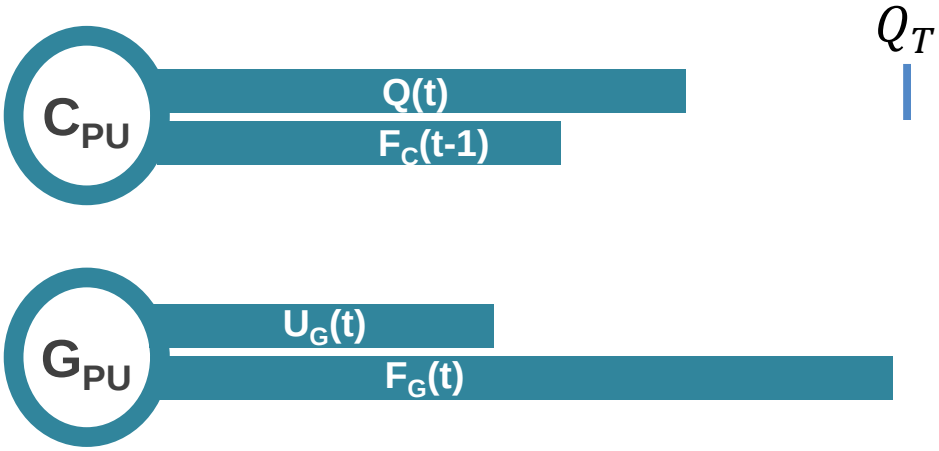
COMPUTING

To calculate the proper processor **frequency**.

COMPUTING

Q_T Scaling

Quality achievement



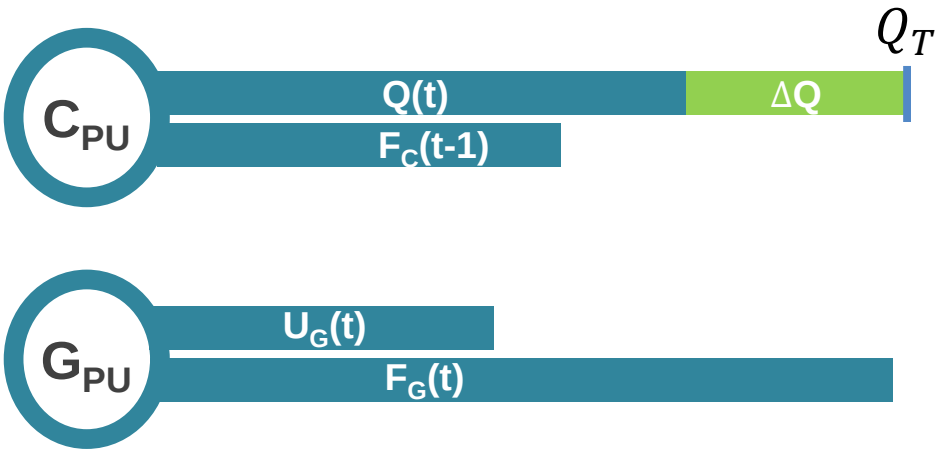
It is CPU oriented in this example.

COMPUTING

Q_T Scaling

Quality achievement

$$\Delta Q = Q_T(t) - Q(t)$$



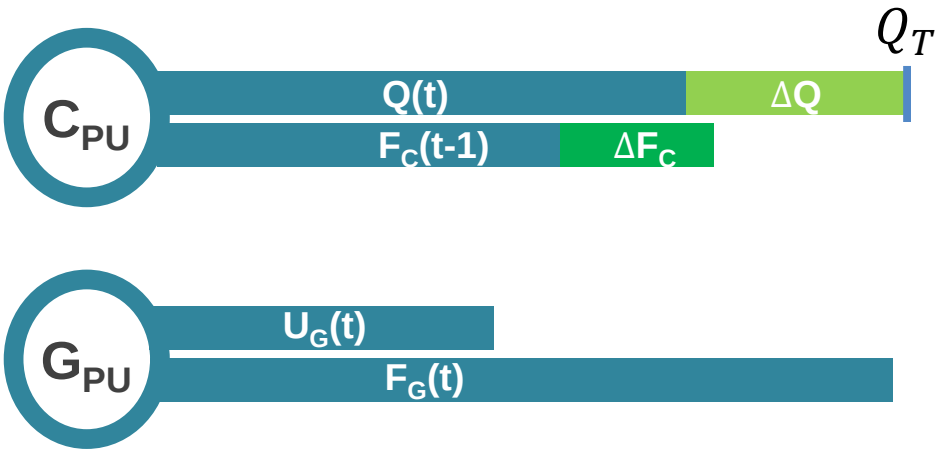
It is CPU oriented in this example.

COMPUTING

Q_T Scaling

Quality achievement

$$\Delta Q = Q_T(t) - Q(t) \qquad \Delta F_C = \Delta Q \times Q2F_C$$



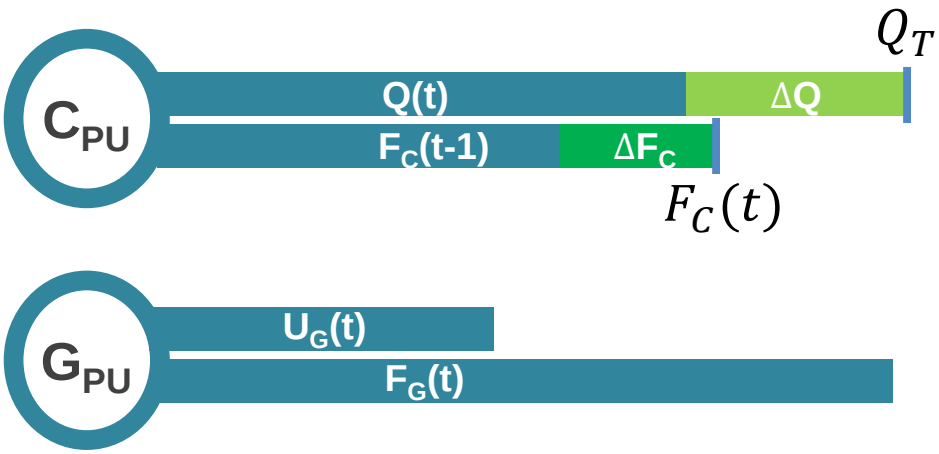
It is CPU oriented in this example.

COMPUTING

Q_T Scaling

Quality achievement

$$\Delta Q = Q_T(t) - Q(t) \qquad \Delta F_C = \Delta Q \times Q2F_C \qquad F_C(t) = F_C(t-1) + \Delta F_C$$



It is CPU oriented in this example.

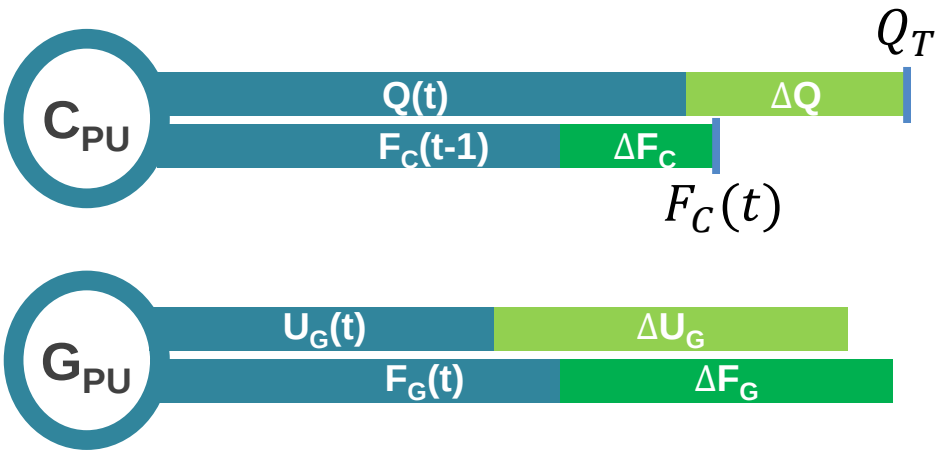
COMPUTING

Q_T Scaling

Quality achievement

$$\Delta Q = Q_T(t) - Q(t) \qquad \Delta F_C = \Delta Q \times Q2F_C \qquad F_C(t) = F_C(t-1) + \Delta F_C$$

$$\Delta U_G = U_{G_T}(t) - U_G(t) \qquad \Delta F_G = \Delta U_G \times U_G2F_G \qquad F_G(t) = F_G(t-1) + \Delta F_G$$



It is CPU oriented in this example.

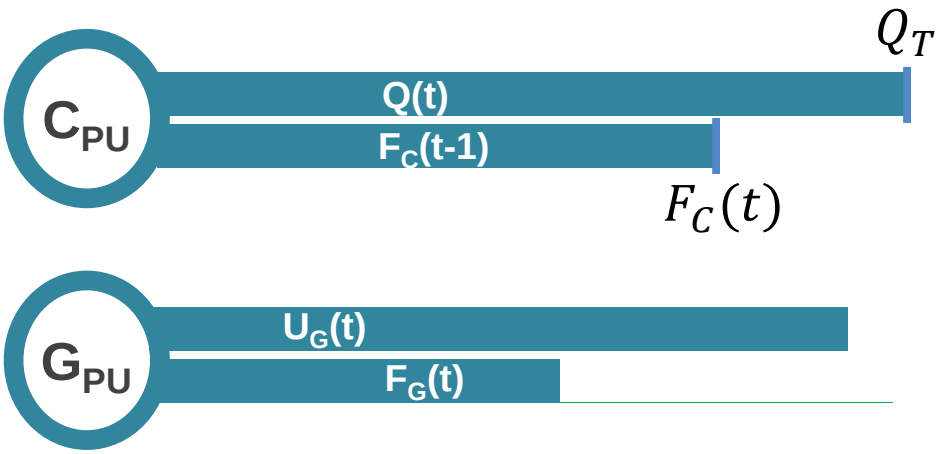
COMPUTING

Q_T Scaling

Quality achievement

$$\Delta Q = Q_T(t) - Q(t) \qquad \Delta F_C = \Delta Q \times Q2F_C \qquad F_C(t) = F_C(t-1) + \Delta F_C$$

$$\Delta U_G = U_{G_T}(t) - U_G(t) \qquad \Delta F_G = \Delta U_G \times U_G2F_G \qquad F_G(t) = F_G(t-1) + \Delta F_G$$



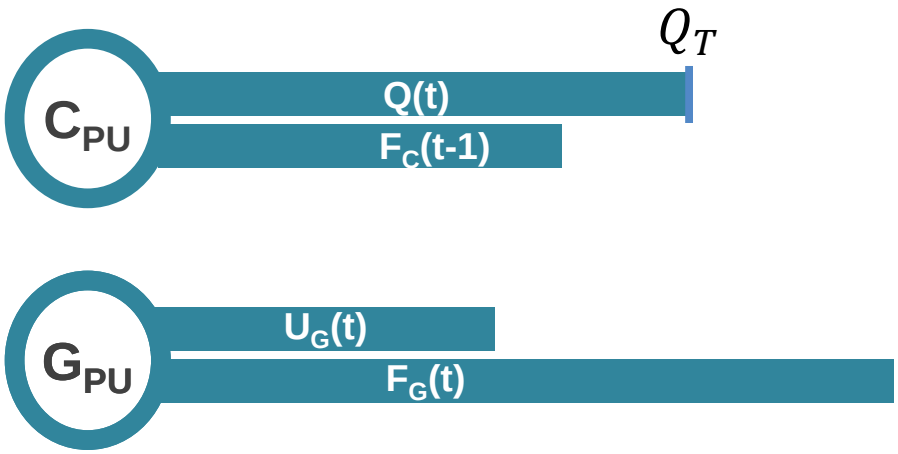
It is CPU oriented in this example.

COMPUTING

U_T Scaling

Power Saving

$$\Delta U_G = U_{G_T}(t) - U_G(t) \quad \Delta F_G = \Delta U_G \times U_G 2F_G \quad F_G(t) = F_G(t-1) + \Delta F_G$$



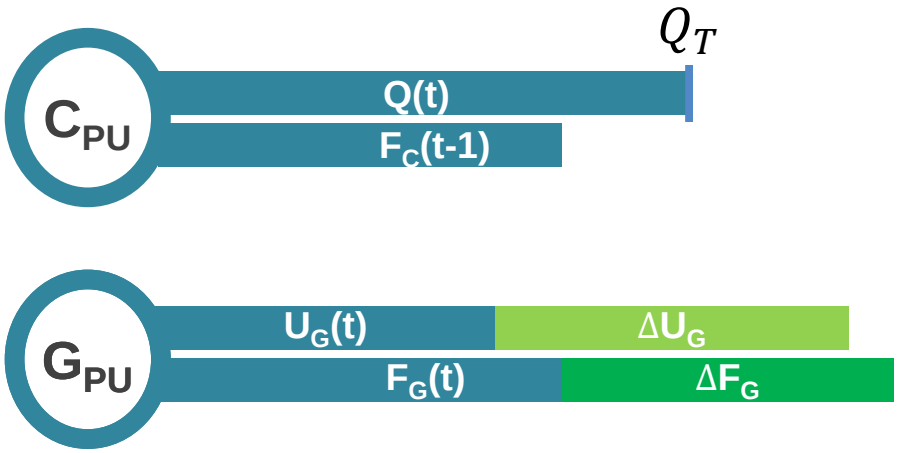
It is CPU oriented in this example.

COMPUTING

U_T Scaling

Power Saving

$$\Delta U_G = U_{G_T}(t) - U_G(t) \quad \Delta F_G = \Delta U_G \times U_G 2F_G \quad F_G(t) = F_G(t-1) + \Delta F_G$$



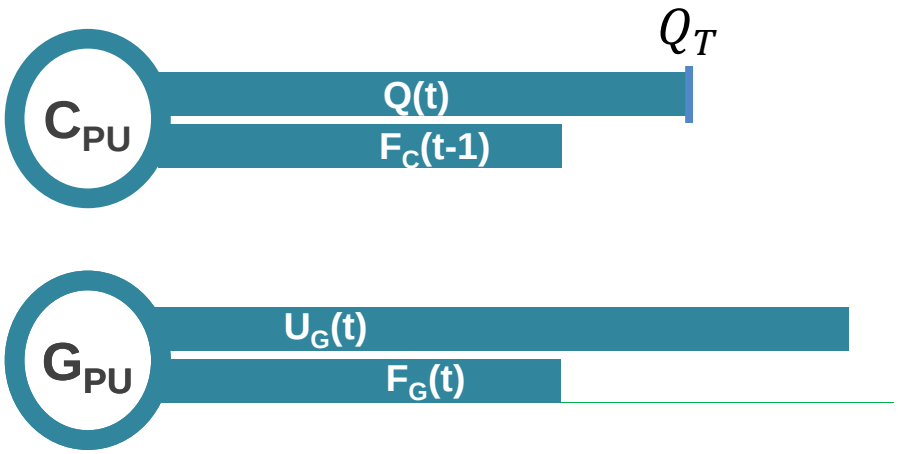
It is CPU oriented in this example.

COMPUTING

U_T Scaling

Power Saving

$$\Delta U_G = U_{G_T}(t) - U_G(t) \quad \Delta F_G = \Delta U_G \times U_G 2F_G \quad F_G(t) = F_G(t-1) + \Delta F_G$$



It is CPU oriented in this example.

DECIDING

ORIENTATION

CPU Oriented

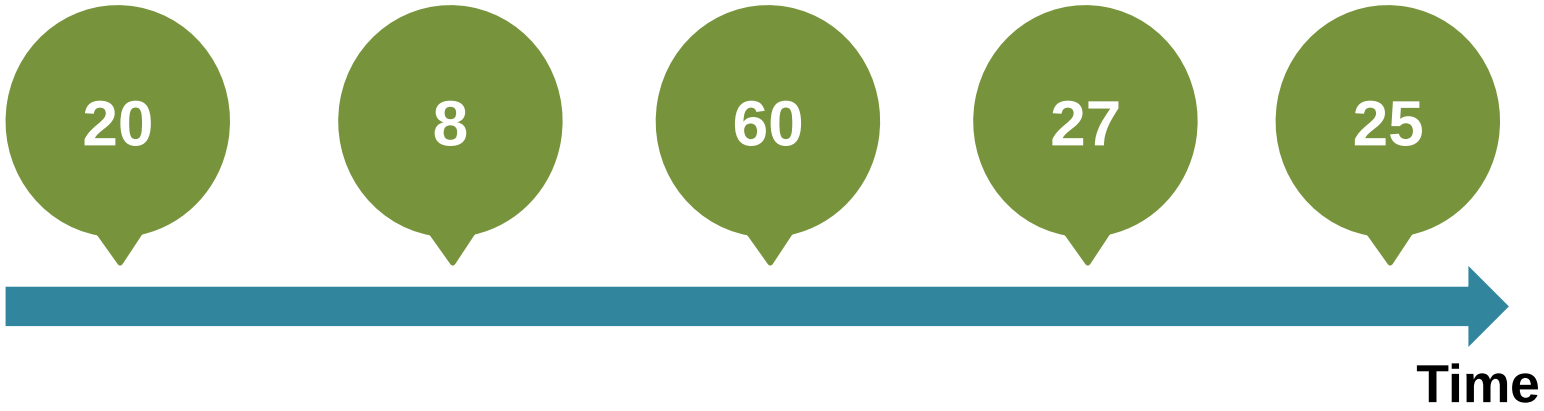
According to Linux standard, when the utilization of GPU is lower than 80%, it is CPU oriented.

SAMPLING

TARGET

Quality Target Q_T (FPS)

$$\Delta Q = Q_T(t) - Q(t) \qquad \Delta F_C = \Delta Q \times Q2F_C \qquad F_C(t) = F_C(t-1) + \Delta F_C$$

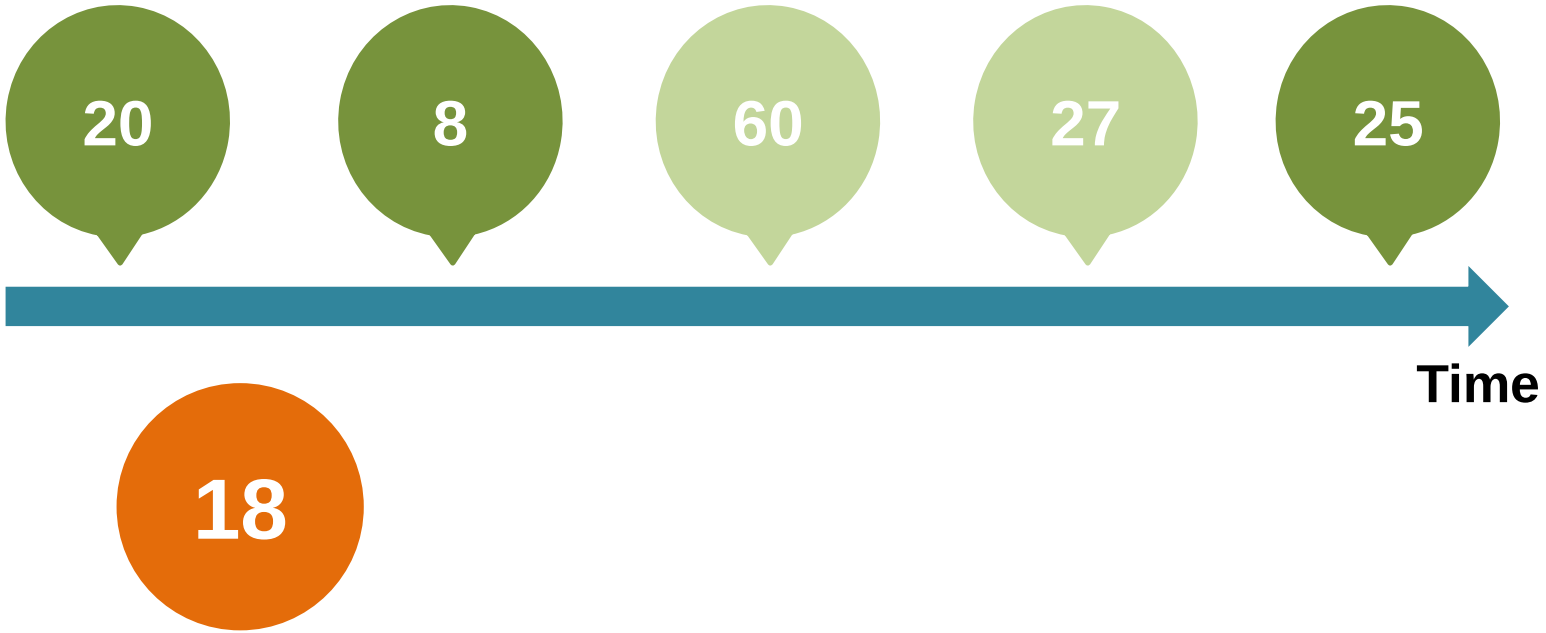


SAMPLING

TARGET

Quality Target Q_T (FPS)

$$\Delta Q = Q_T(t) - Q(t) \qquad \Delta F_C = \Delta Q \times Q2F_C \qquad F_C(t) = F_C(t-1) + \Delta F_C$$

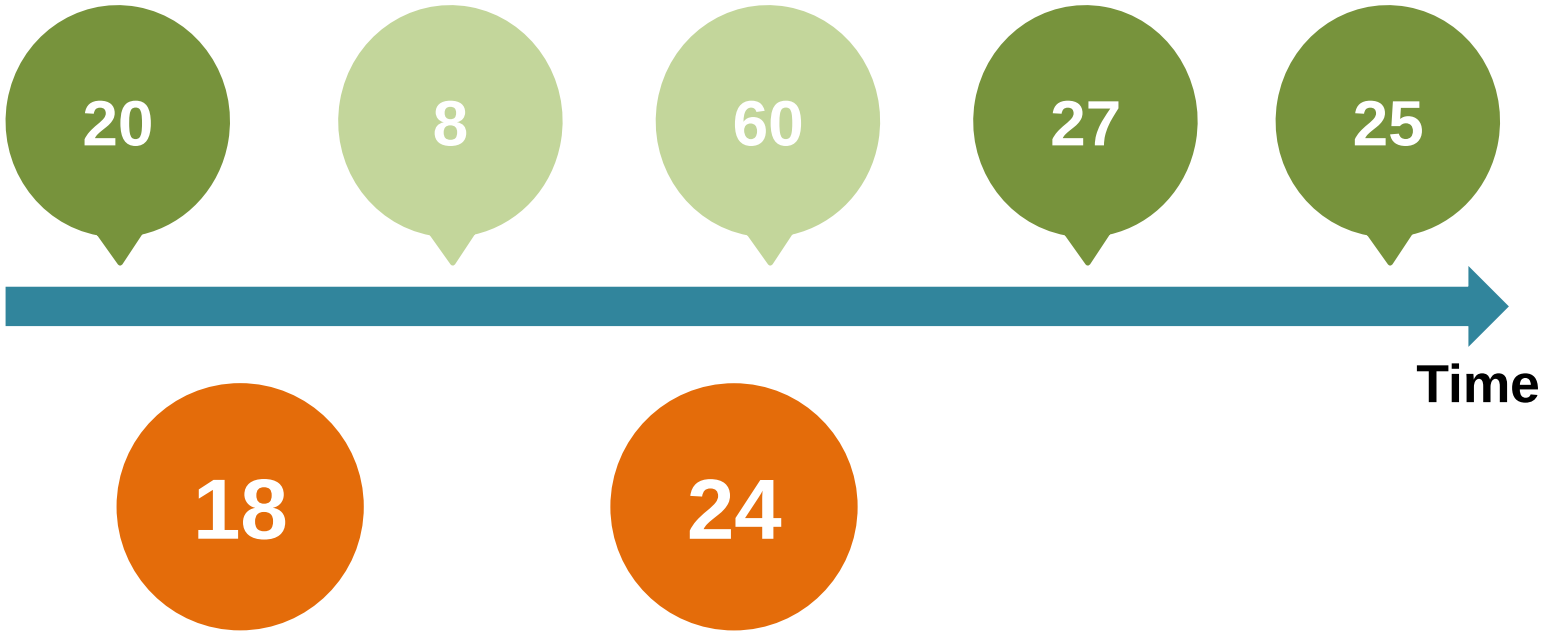


SAMPLING

TARGET

Quality Target Q_T (FPS)

$$\Delta Q = Q_T(t) - Q(t) \qquad \Delta F_C = \Delta Q \times Q2F_C \qquad F_C(t) = F_C(t-1) + \Delta F_C$$

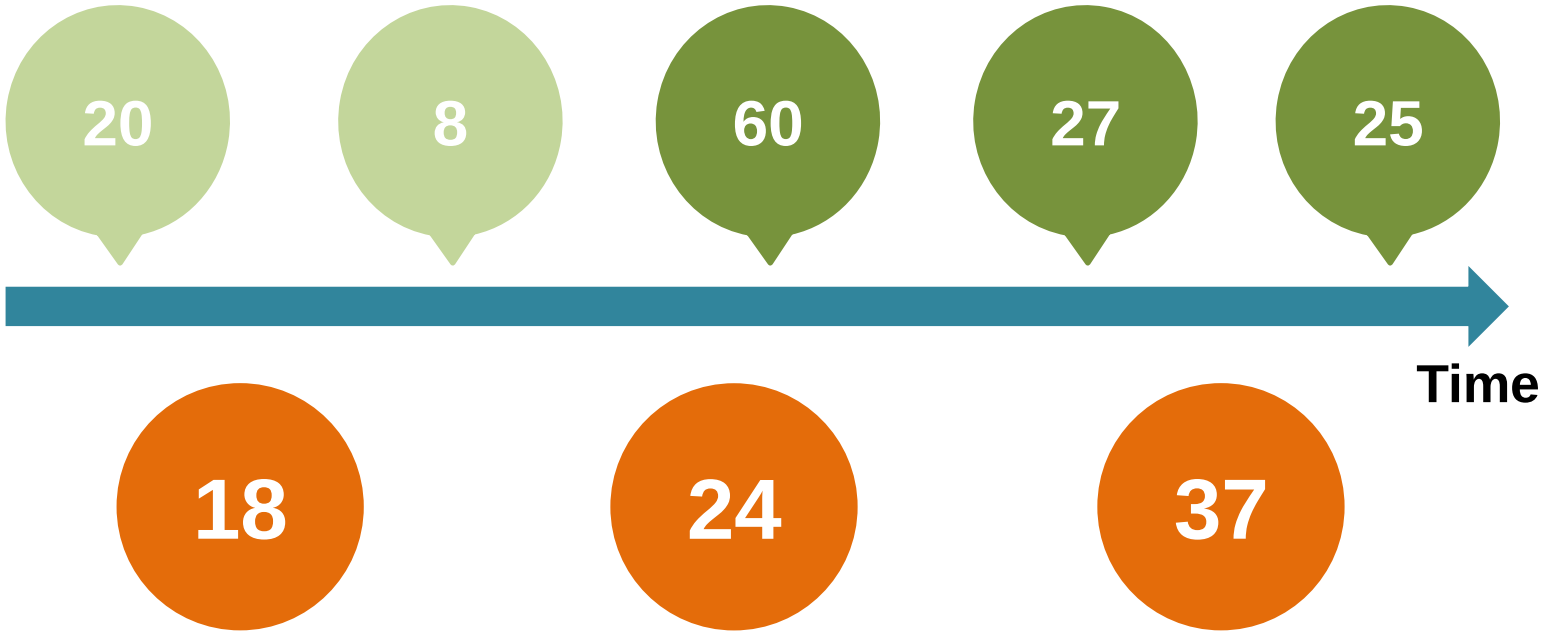


SAMPLING

TARGET

Quality Target Q_T (FPS)

$$\Delta Q = Q_T(t) - Q(t) \qquad \Delta F_C = \Delta Q \times Q2F_C \qquad F_C(t) = F_C(t-1) + \Delta F_C$$

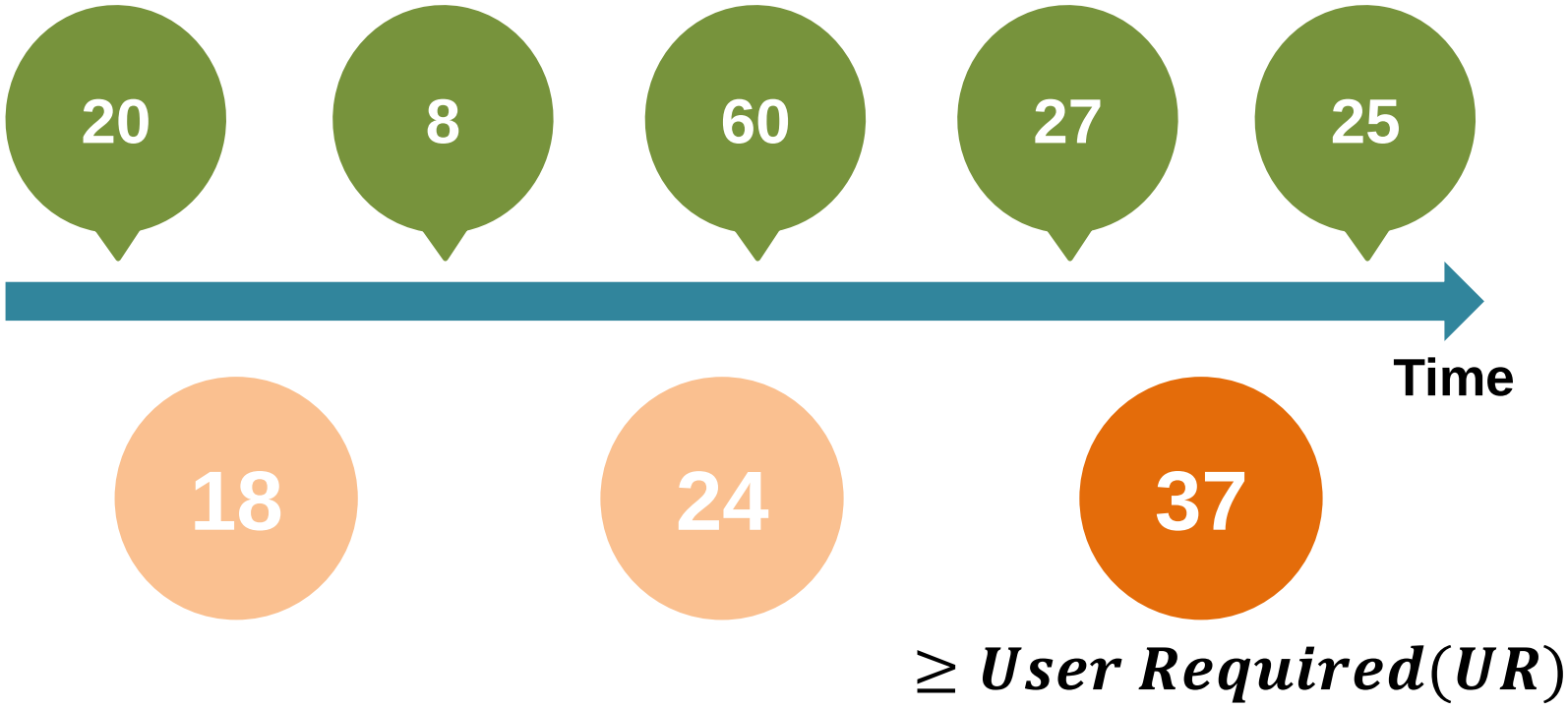


SAMPLING

TARGET

Quality Target Q_T (FPS)

$$\Delta Q = Q_T(t) - Q(t) \qquad \Delta F_C = \Delta Q \times Q2F_C \qquad F_C(t) = F_C(t-1) + \Delta F_C$$



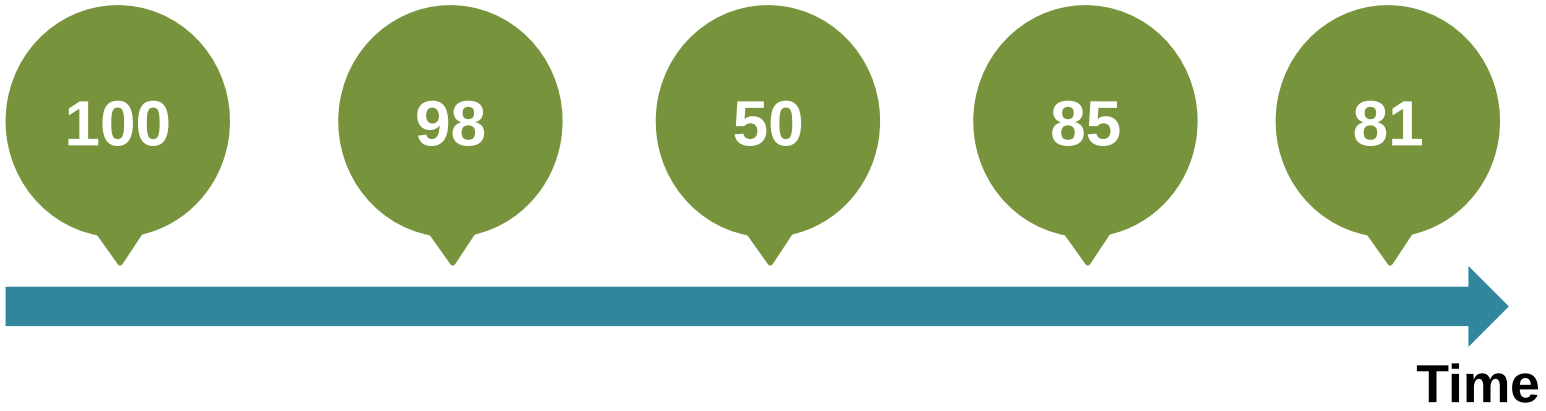
Kim, Dongwon, Nohyun Jung, and Hojung Cha. "Content-centric Display Energy Management for Mobile Devices." Proceedings of the 51st Annual Design Automation Conference. ACM, 2014.

SAMPLING

TARGET

Utilization Target U_T

$$\Delta U_G = U_{G_T}(t) - U_G(t) \quad \Delta F_G = \Delta U_G \times U_G 2 F_G \quad F_G(t) = F_G(t-1) + \Delta F_G$$

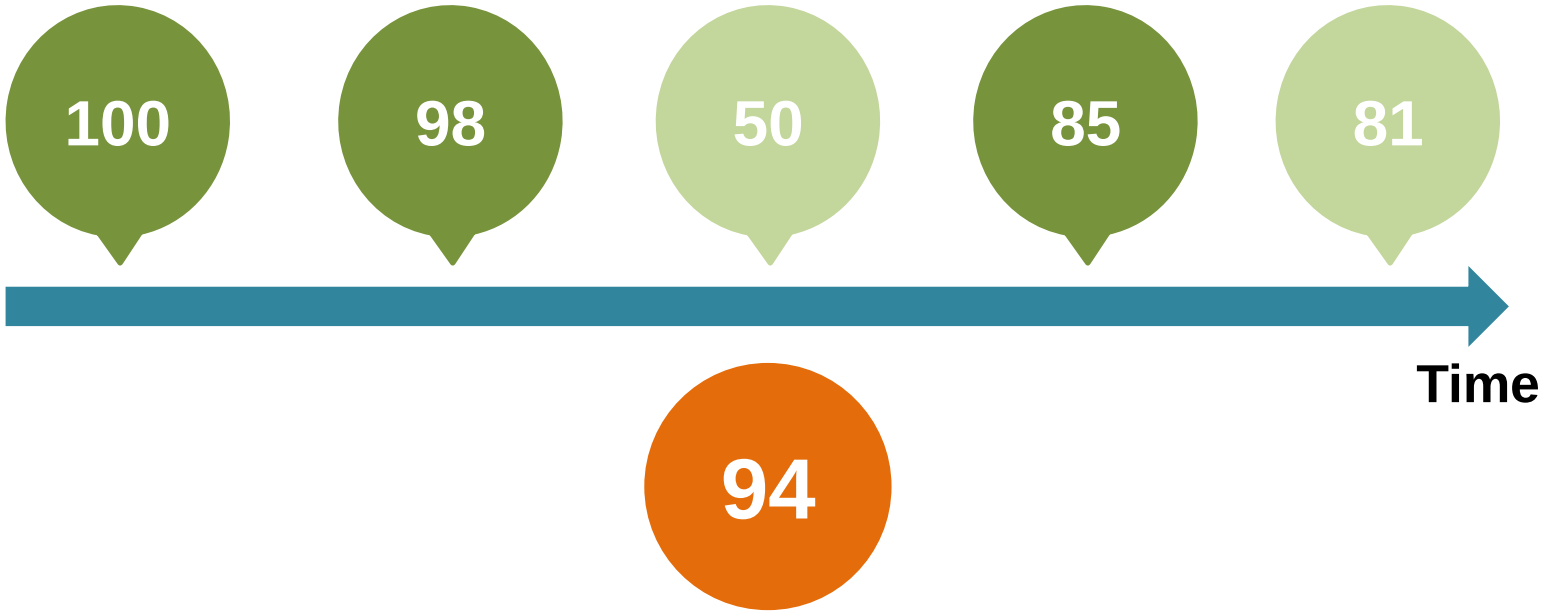


SAMPLING

TARGET

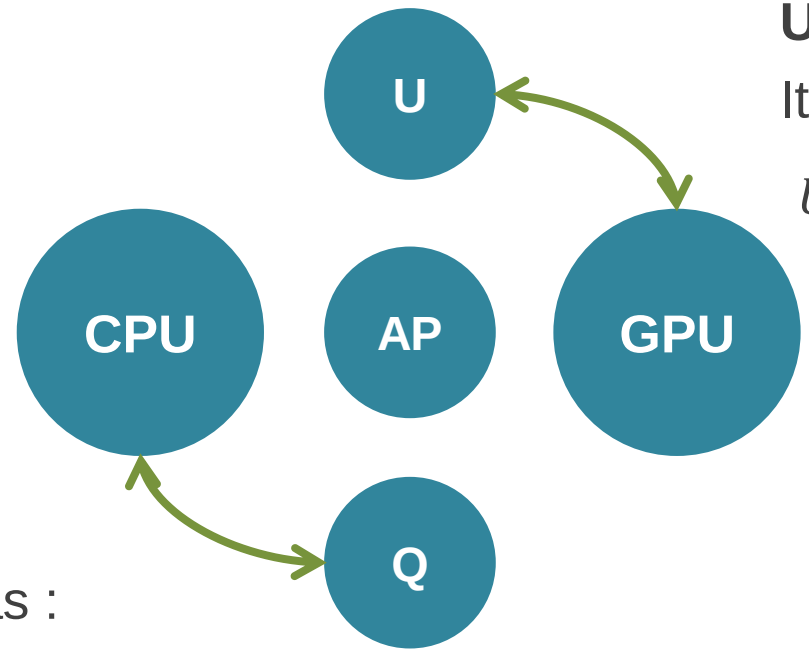
Utilization Target U_T

$$\Delta U_G = U_{G_T}(t) - U_G(t) \quad \Delta F_G = \Delta U_G \times U_G 2F_G \quad F_G(t) = F_G(t-1) + \Delta F_G$$



LEARNING COEFFICIENT

$$\Delta Q = Q_T(t) - Q(t) \qquad \Delta F_C = \Delta Q \times Q2F_C \qquad F_C(t) = F_C(t-1) + \Delta F_C$$



U_G2F_G

It is defined as :

$$U_G2F_G = \frac{\Delta F_G}{\Delta U_G}$$

$Q2F_C$

It is defined as :

$$Q2F_C = \frac{\Delta F_C}{\Delta Q}$$

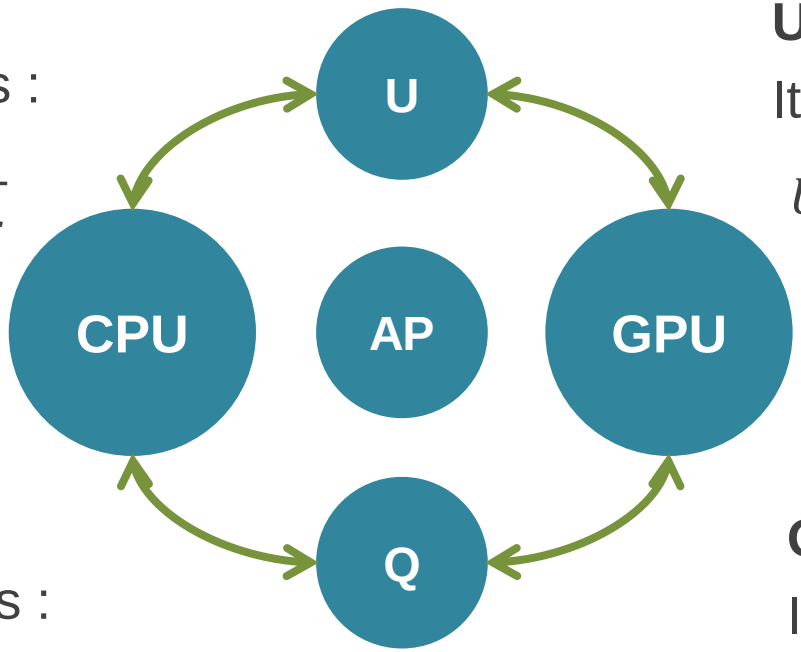
LEARNING COEFFICIENT

$$\Delta Q = Q_T(t) - Q(t) \qquad \Delta F_C = \Delta Q \times Q2F_C \qquad F_C(t) = F_C(t-1) + \Delta F_C$$

U_C2F_C

It is defined as :

$$U_C2F_C = \frac{\Delta F_C}{\Delta U_C}$$



U_G2F_G

It is defined as :

$$U_G2F_G = \frac{\Delta F_G}{\Delta U_G}$$

$Q2F_C$

It is defined as :

$$Q2F_C = \frac{\Delta F_C}{\Delta Q}$$

$Q2F_G$

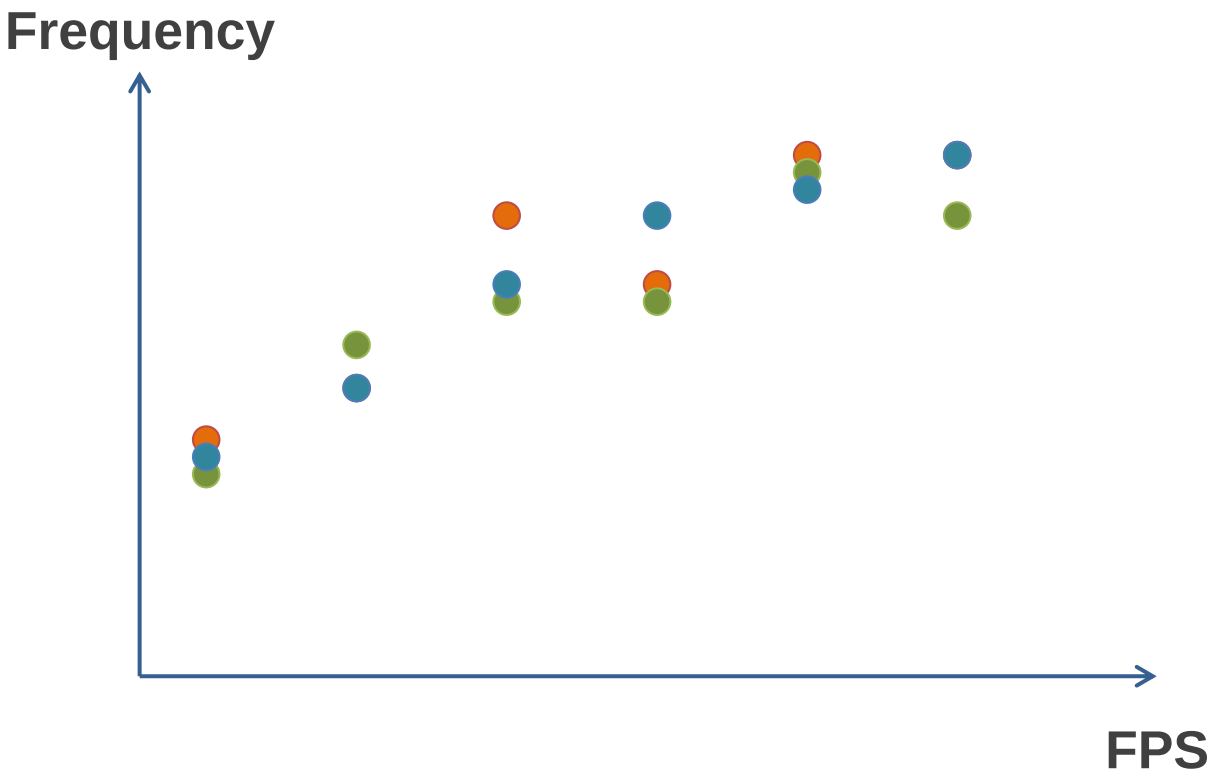
It is defined as :

$$Q2F_G = \frac{\Delta F_G}{\Delta Q}$$

LEARNING

COEFFICIENT

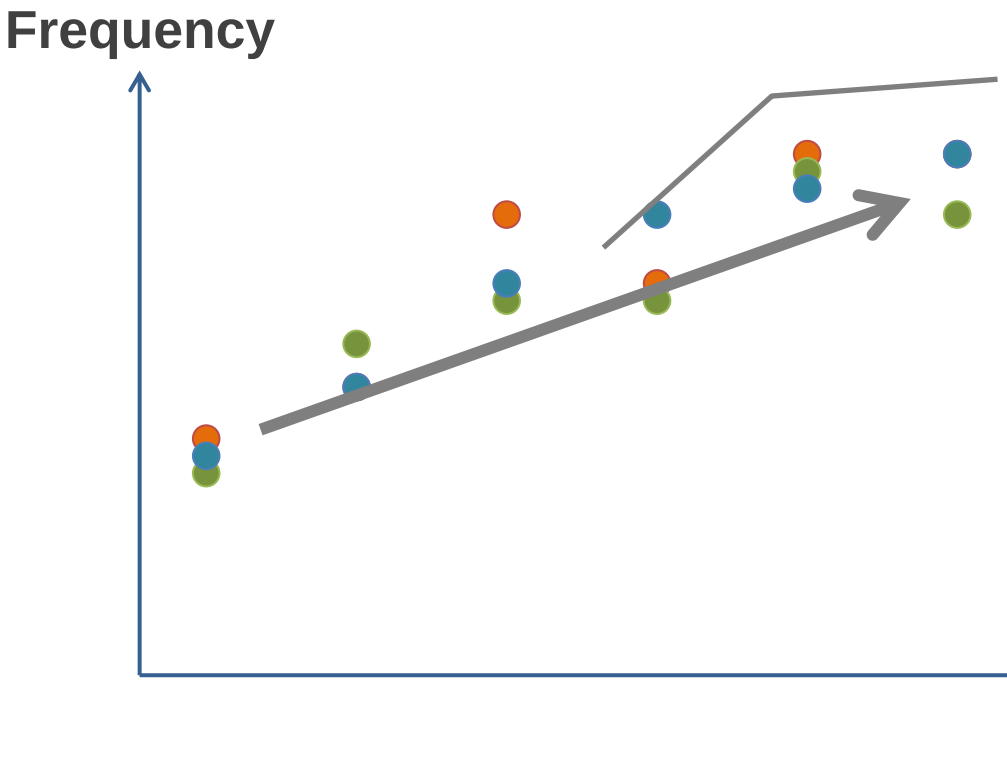
Learning v.s. Regression



LEARNING

COEFFICIENT

Learning v.s. Regression



Q2F

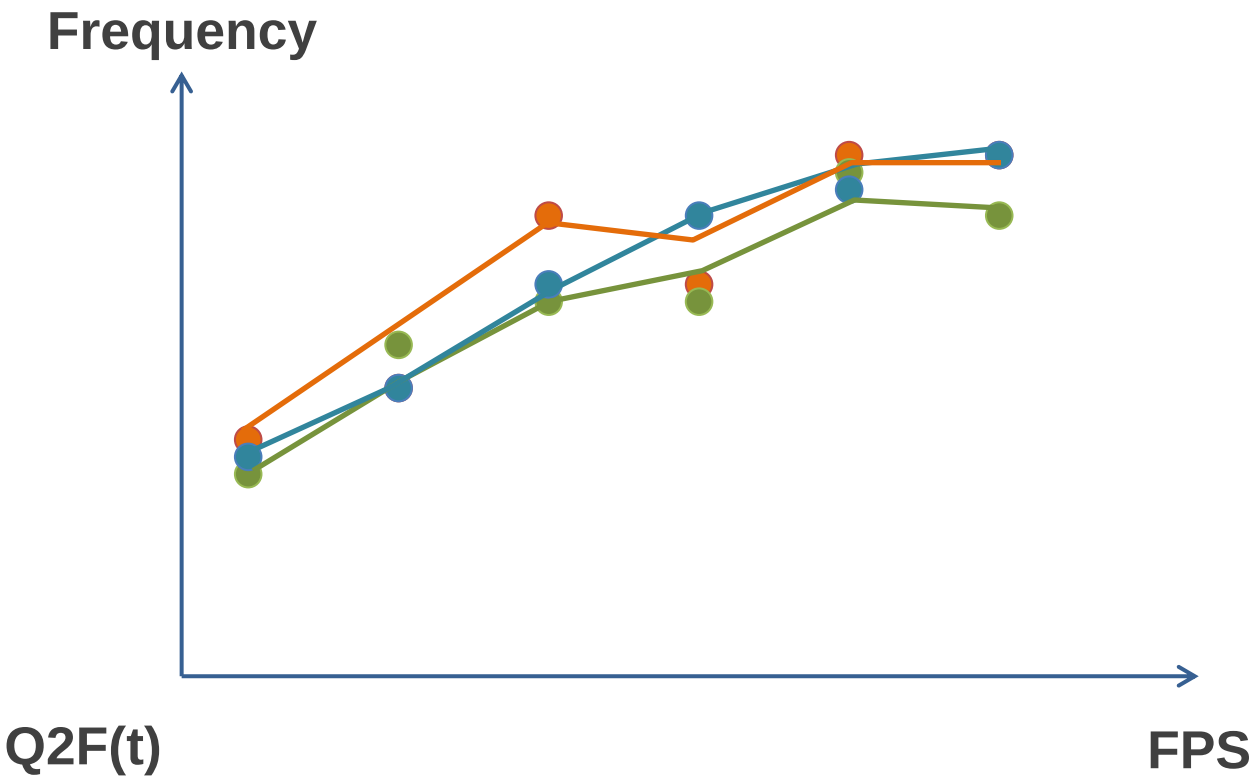
The slope of the regression line :

$$\mathrm{Q2}F = \frac{\Delta F}{\Delta Q}$$

LEARNING

COEFFICIENT

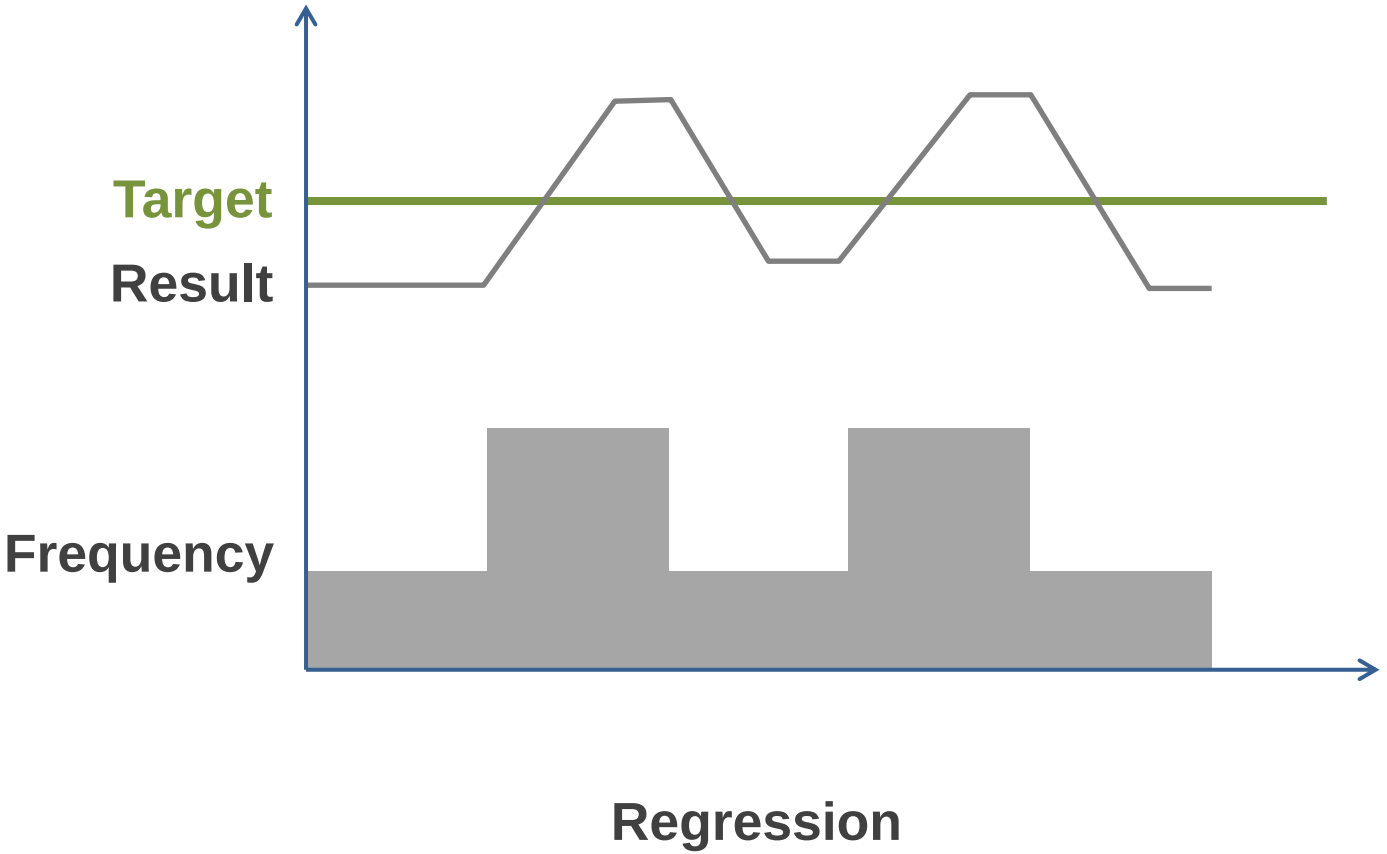
Learning v.s. Regression



A dynamical slope by learning

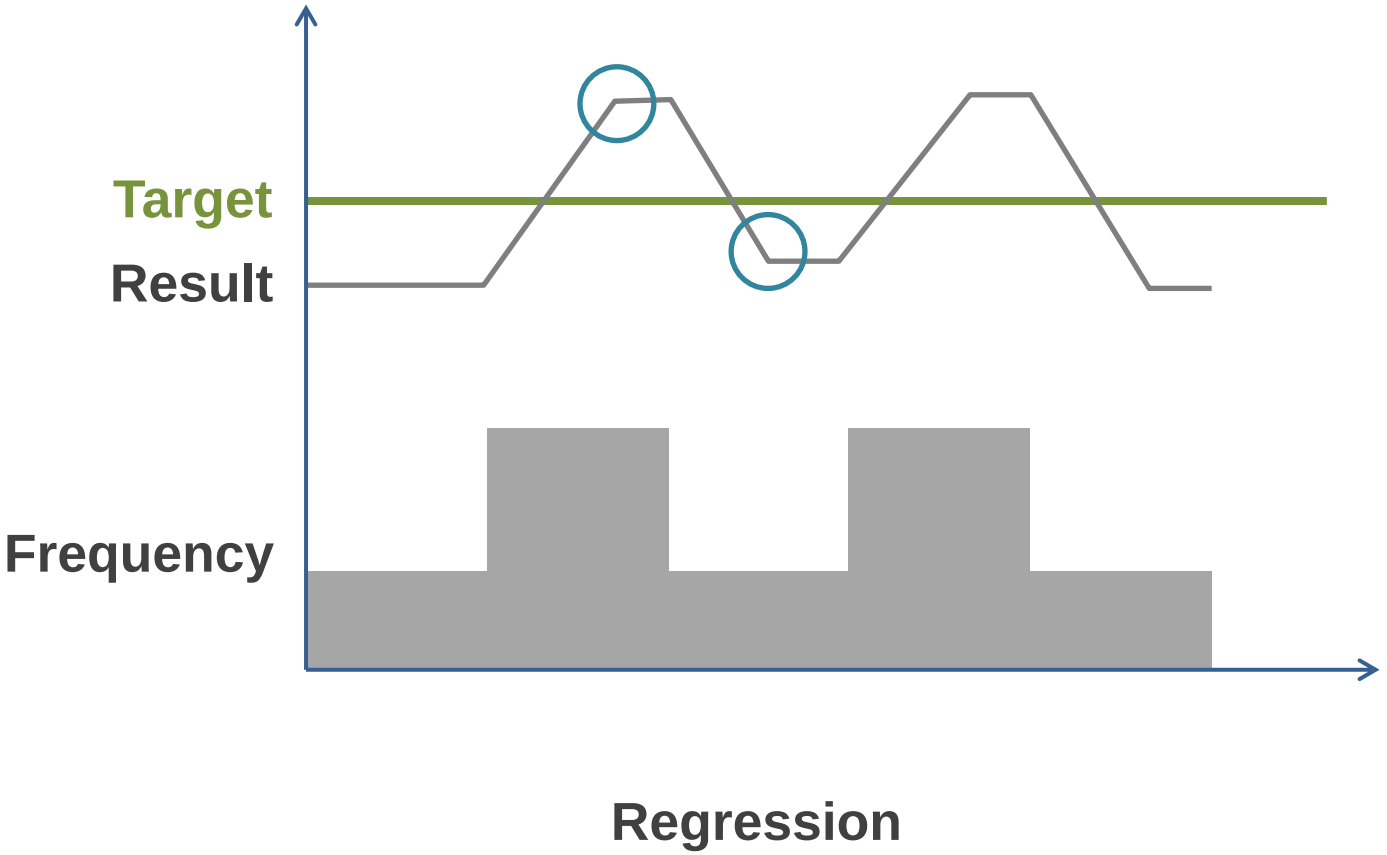
ONLINE SCALING ALGORITHM

Learning v.s. Regression



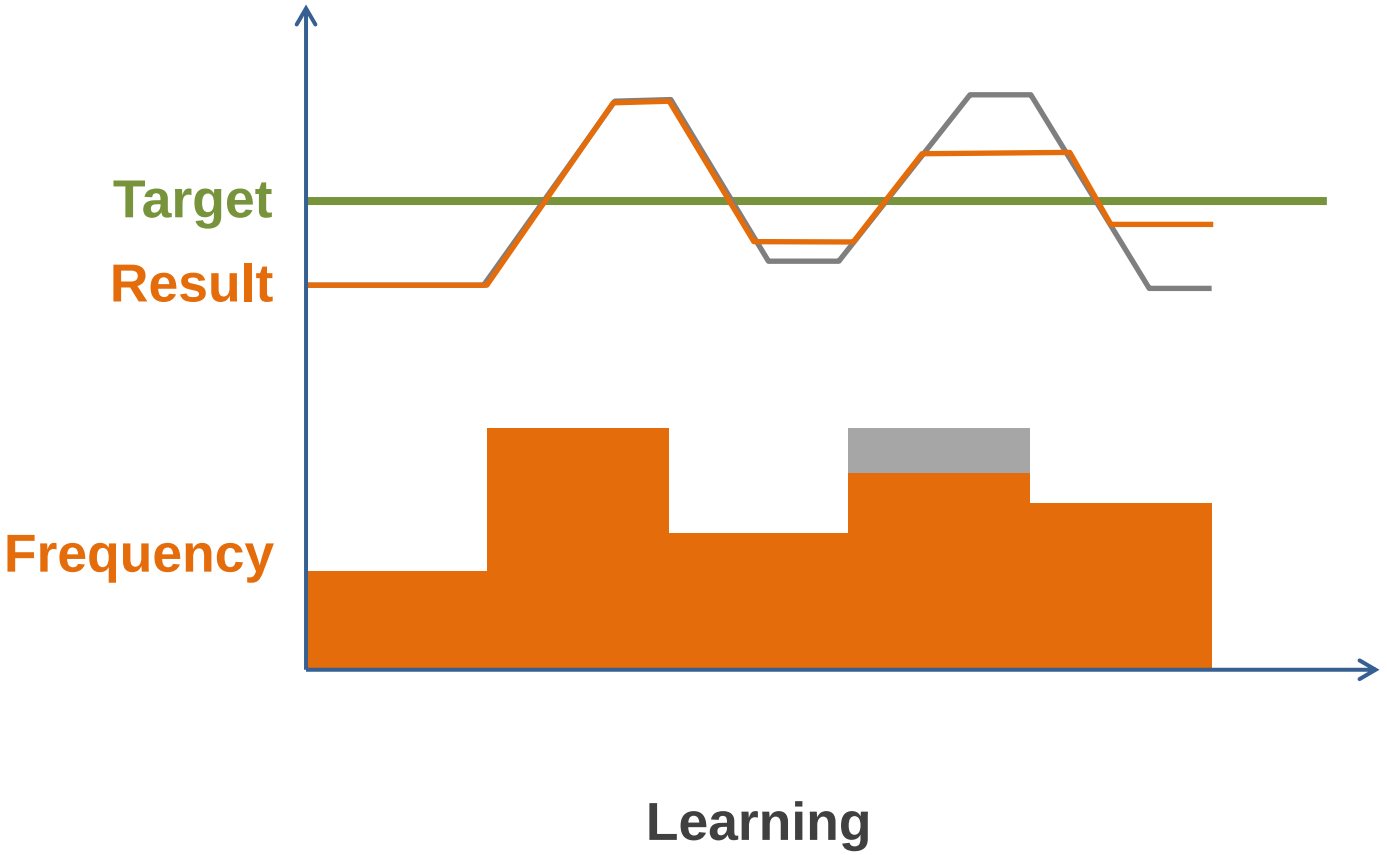
ONLINE SCALING ALGORITHM

Learning v.s. Regression



ONLINE SCALING ALGORITHM

Learning v.s. Regression



LEARNING

EQUATION

Idea of learning

$$Coef(t) = Coef(t-1) + \Delta Coef$$

LEARNING

EQUATION

Idea of learning

$$Coef(t) = Coef(t-1) + \Delta Coef$$

$$\Delta Coef = \left(Coef_{exp} - Coef(t-1)\right) \times R(t)$$

LEARNING

EQUATION

Idea of learning

$$Coef(t) = Coef(t-1) + \Delta Coef$$

$$\Delta Coef = \left(Coef_{exp} - Coef(t-1)\right) \times R(t)$$

$\mathbf{Coef_{exp}}$

The expected coefficient from the definition.

LEARNING

EQUATION

Idea of learning

$$Coef(t) = Coef(t-1) + \Delta Coef$$

$$\Delta Coef = \left(Coef_{exp} - Coef(t-1)\right) \times R(t)$$

$\mathbf{R(t)}$

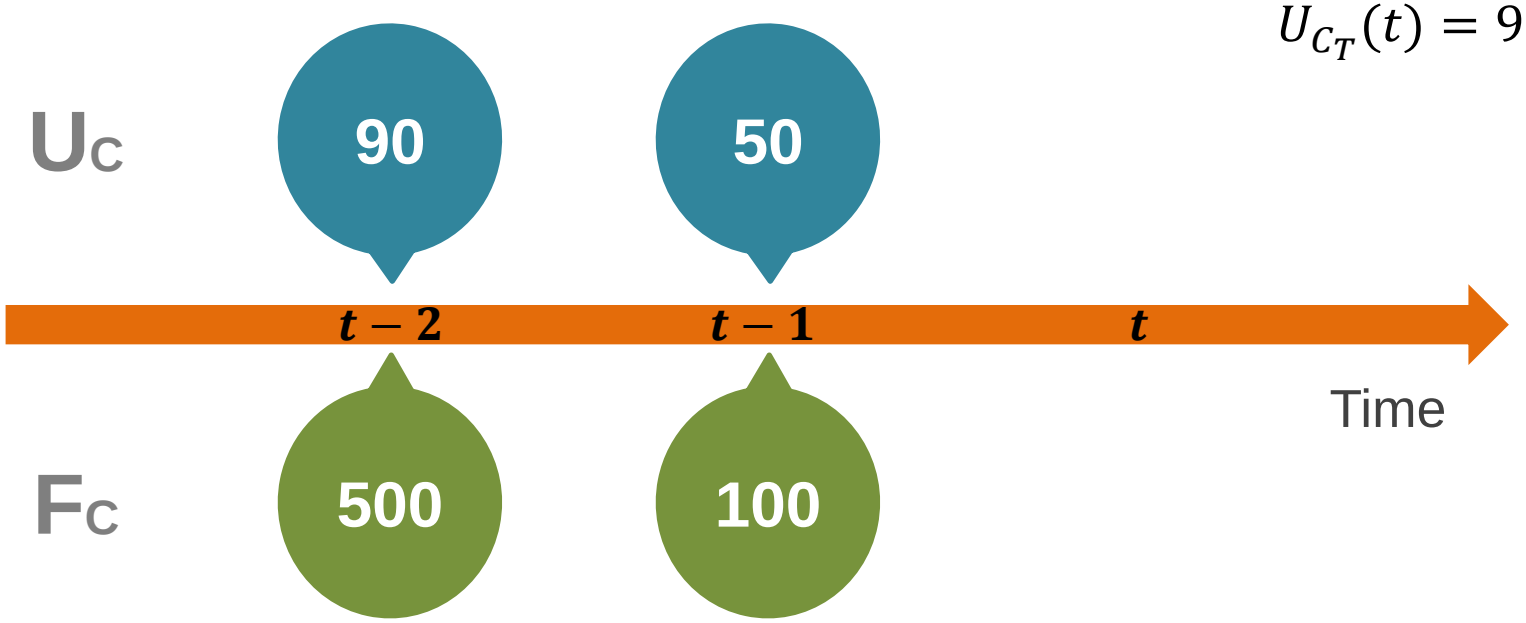
Learning rate.

LEARNING

EQUATION

Example of learning

$$U_{C_T}(t) = 90$$

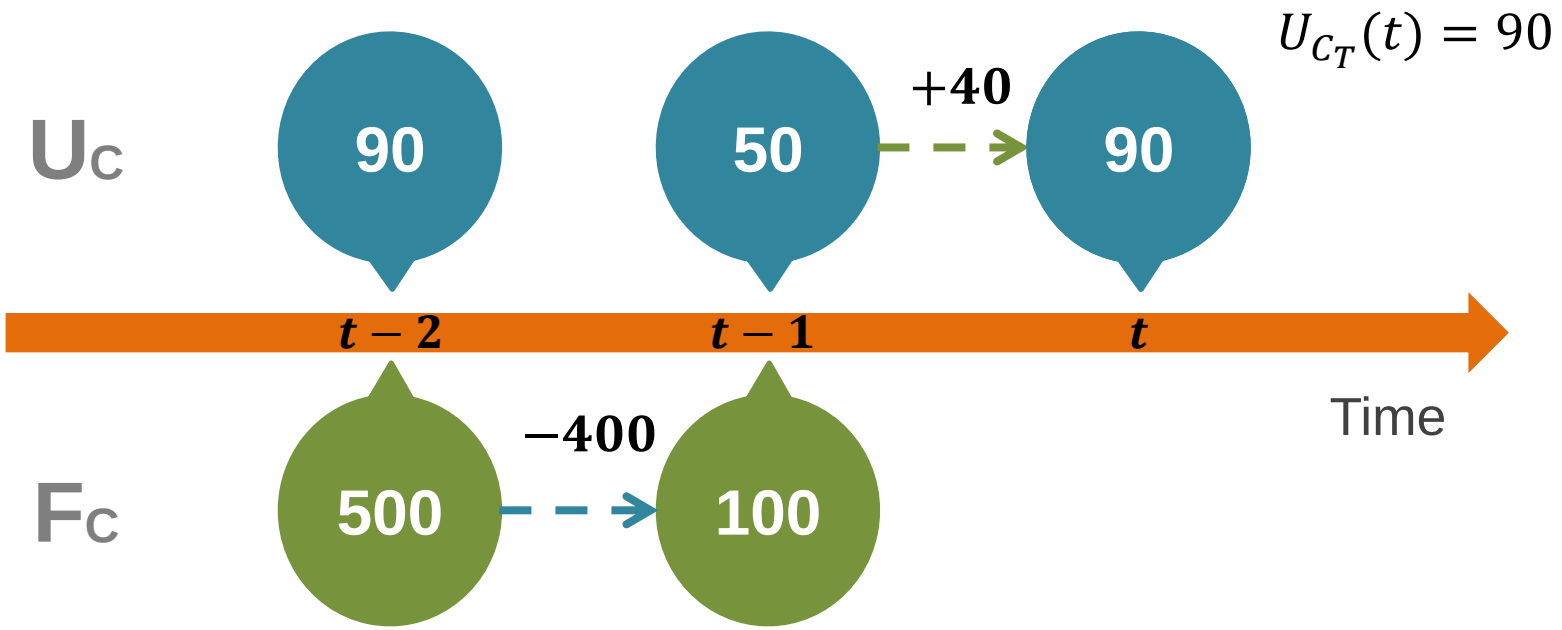


R(t) is 0.7 in this example.

LEARNING

EQUATION

Example of learning



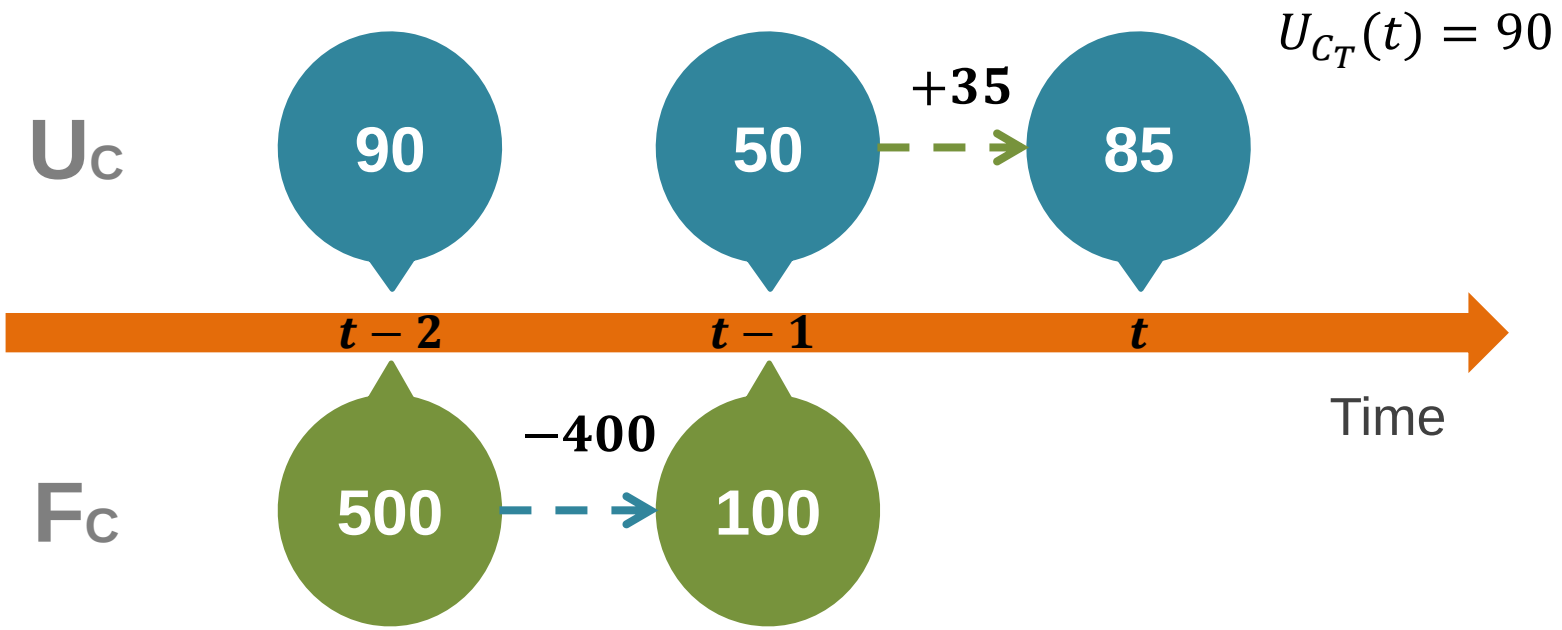
$$U_C 2F_C = -10$$

R(t) is 0.7 in this example.

LEARNING

EQUATION

Example of learning



$$U_C 2F_C = -10$$

R(t) is 0.7 in this example.

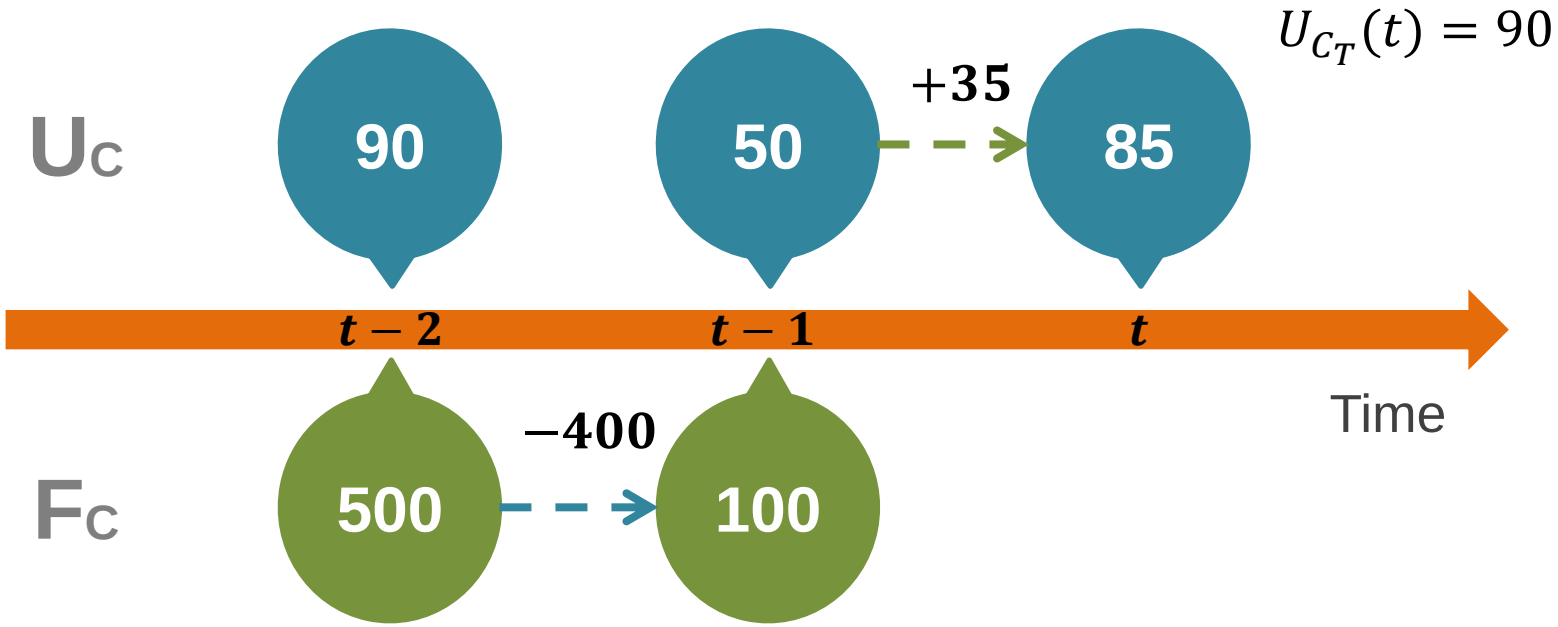
LEARNING

EQUATION

Example of learning

$$\Delta Coef = \left(Coef_{exp} - Coef(t-1)\right) \times R(t)$$

$$\Delta U_C 2F_C = \left(\qquad \right) \times 0.7 =$$



$$U_C 2F_C = -10$$

R(t) is 0.7 in this example.

LEARNING

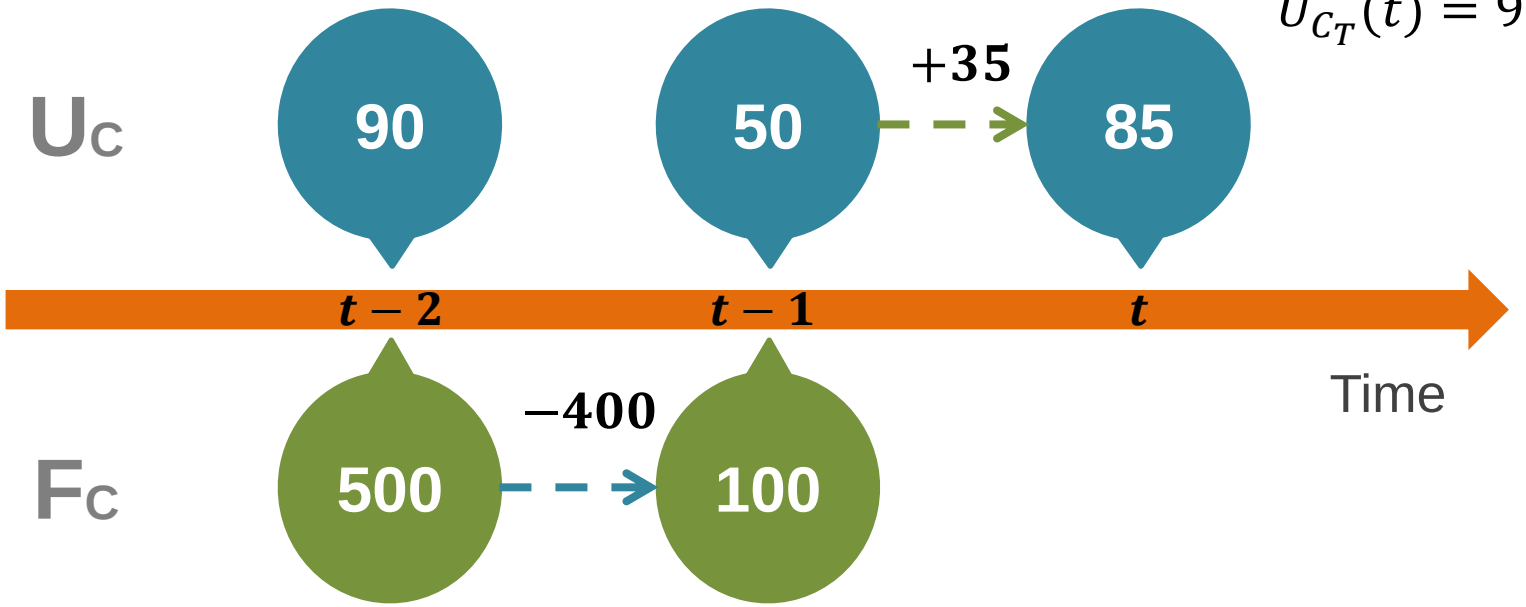
EQUATION

Example of learning

$$\Delta Coef = \left(Coef_{exp} - Coef(t-1)\right) \times R(t)$$

$$\Delta U_C 2F_C = \left(\frac{-400}{35}\right) \times 0.7 =$$

$$U_{C_T}(t) = 90$$



$$U_C 2F_C = -10$$

R(t) is 0.7 in this example.

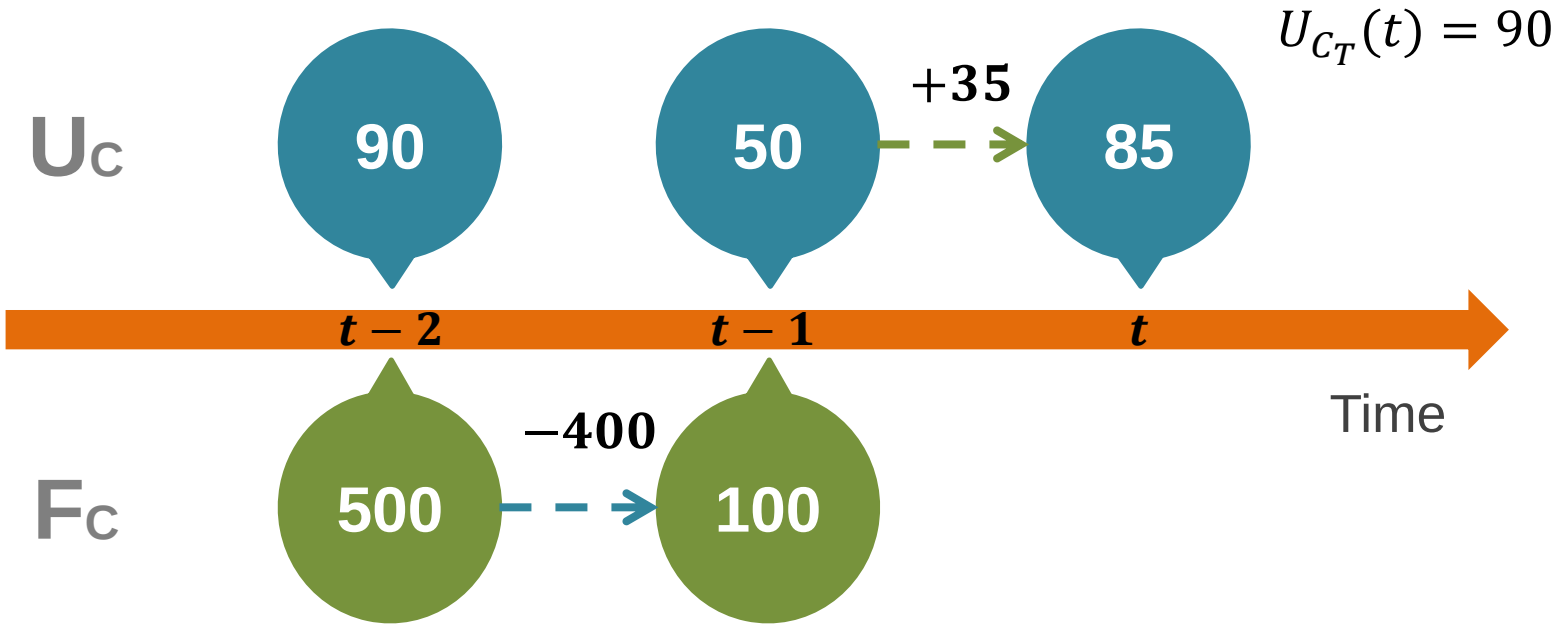
LEARNING

EQUATION

Example of learning

$$\Delta Coef = \left(Coef_{exp} - Coef(t-1)\right) \times R(t)$$

$$\Delta U_C 2F_C = \left(\frac{-400}{35} - (-10)\right) \times 0.7 =$$



$$U_C 2F_C = -10$$

R(t) is 0.7 in this example.

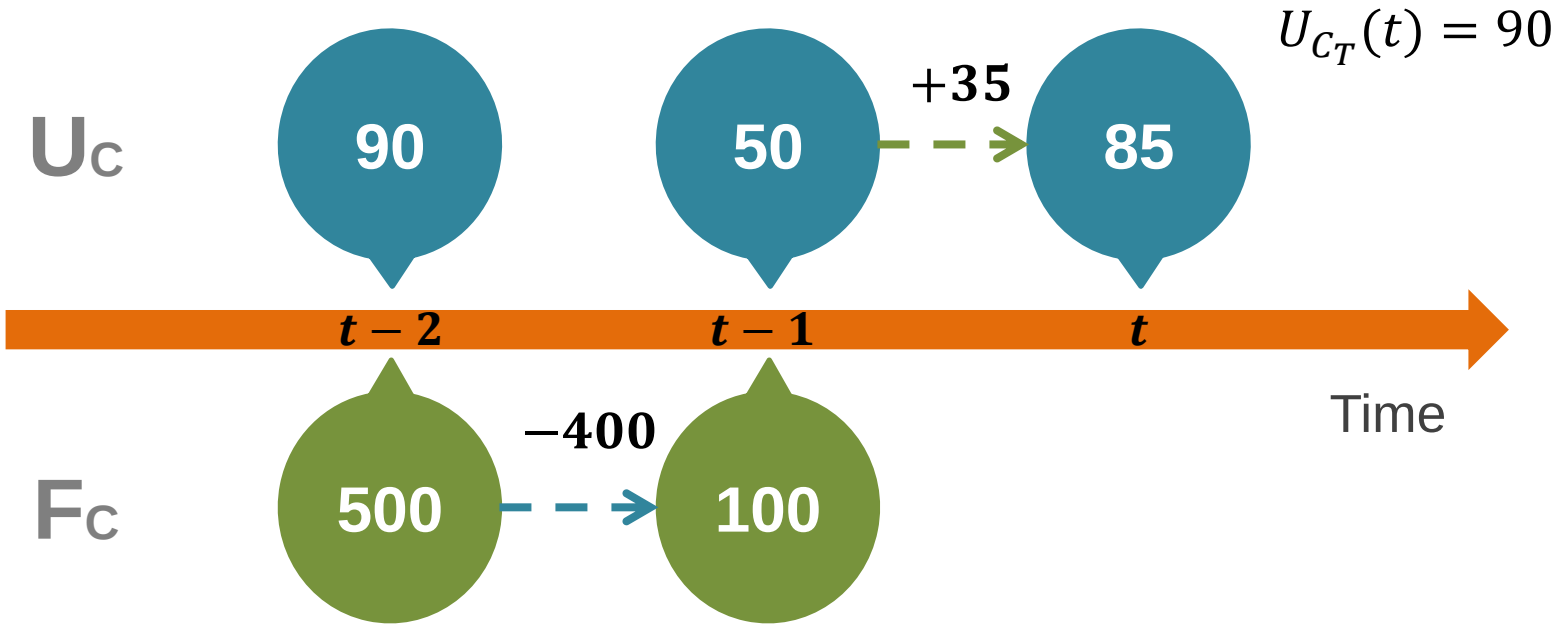
LEARNING

EQUATION

Example of learning

$$\Delta Coef = \left(Coef_{exp} - Coef(t-1)\right) \times R(t)$$

$$\Delta U_C 2F_C = \left(\frac{-400}{35} - (-10)\right) \times 0.7 = -1$$



$U_C 2F_C = -10$ $U_C 2F_C = -10 + (-1)$

R(t) is 0.7 in this example.

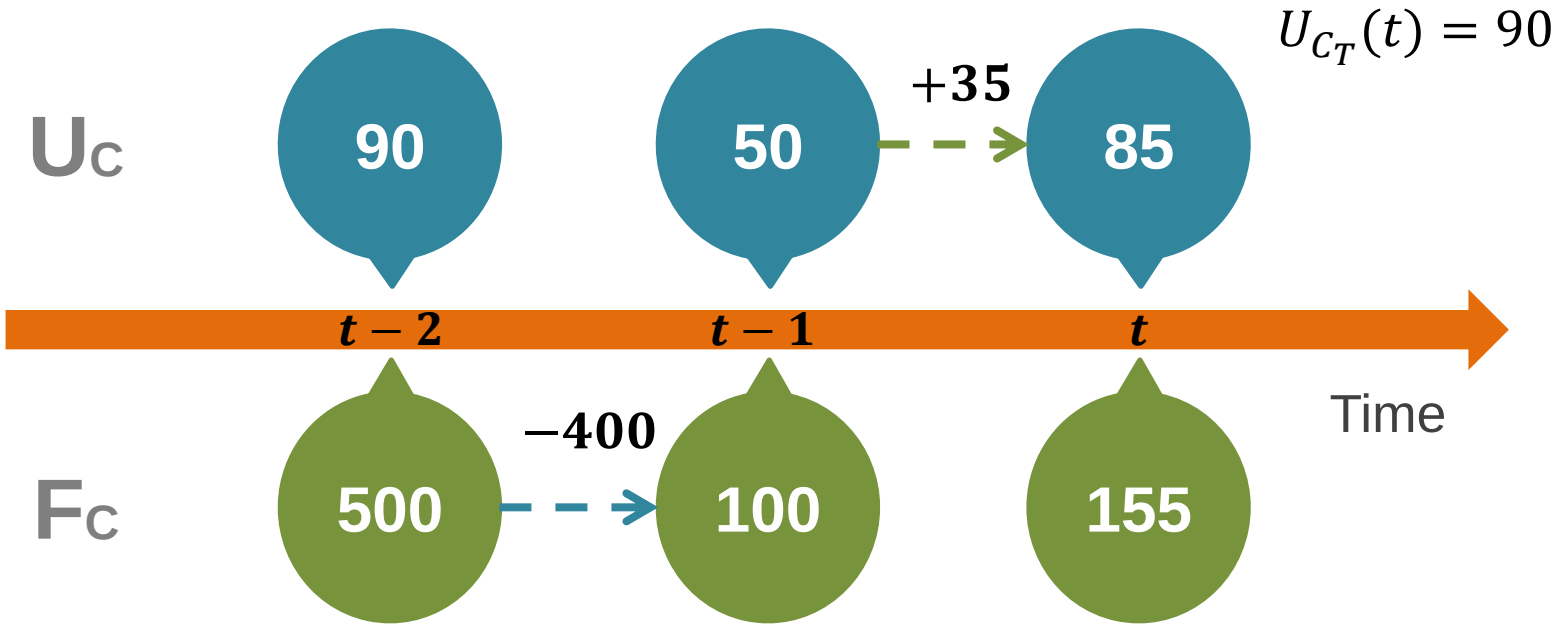
LEARNING

EQUATION

Example of learning

$$\Delta Coef = \left(Coef_{exp} - Coef(t-1)\right) \times R(t)$$

$$\Delta U_C 2F_C = \left(\frac{-400}{35} - (-10)\right) \times 0.7 = -1$$

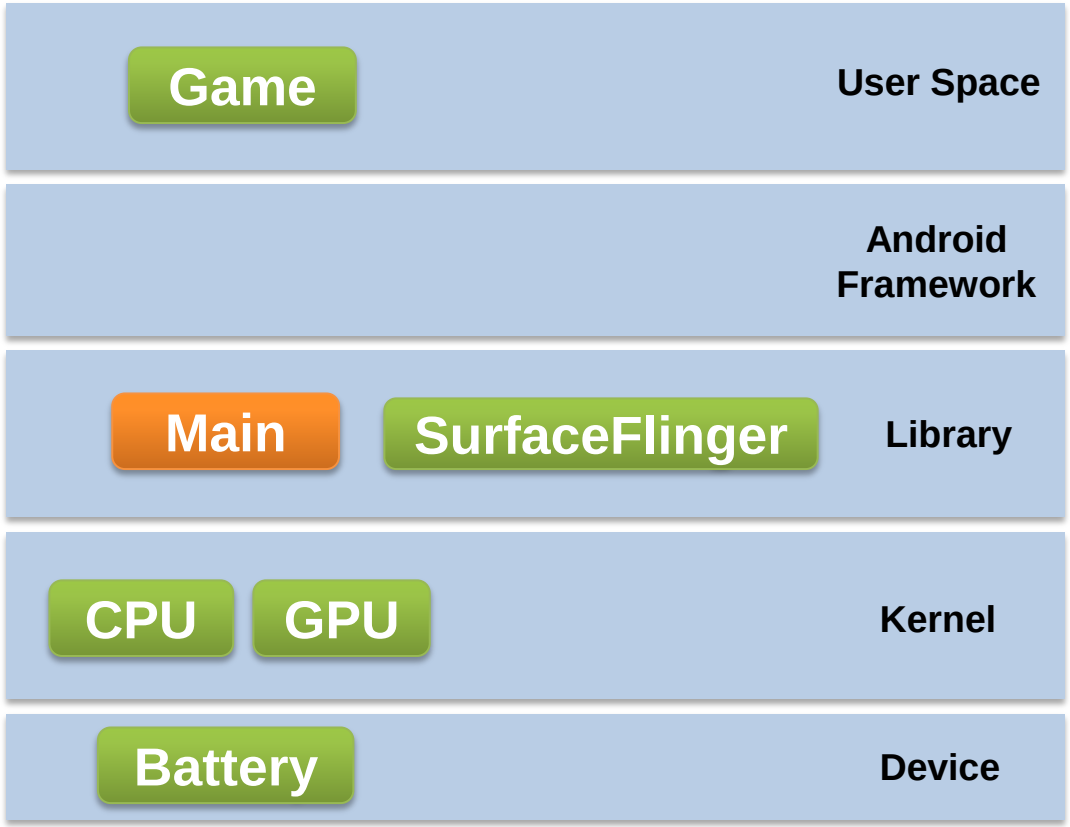


$U_C 2F_C = -10$

$U_C 2F_C = -10 + (-1)$

R(t) is 0.7 in this example.

IMPLEMENTATION



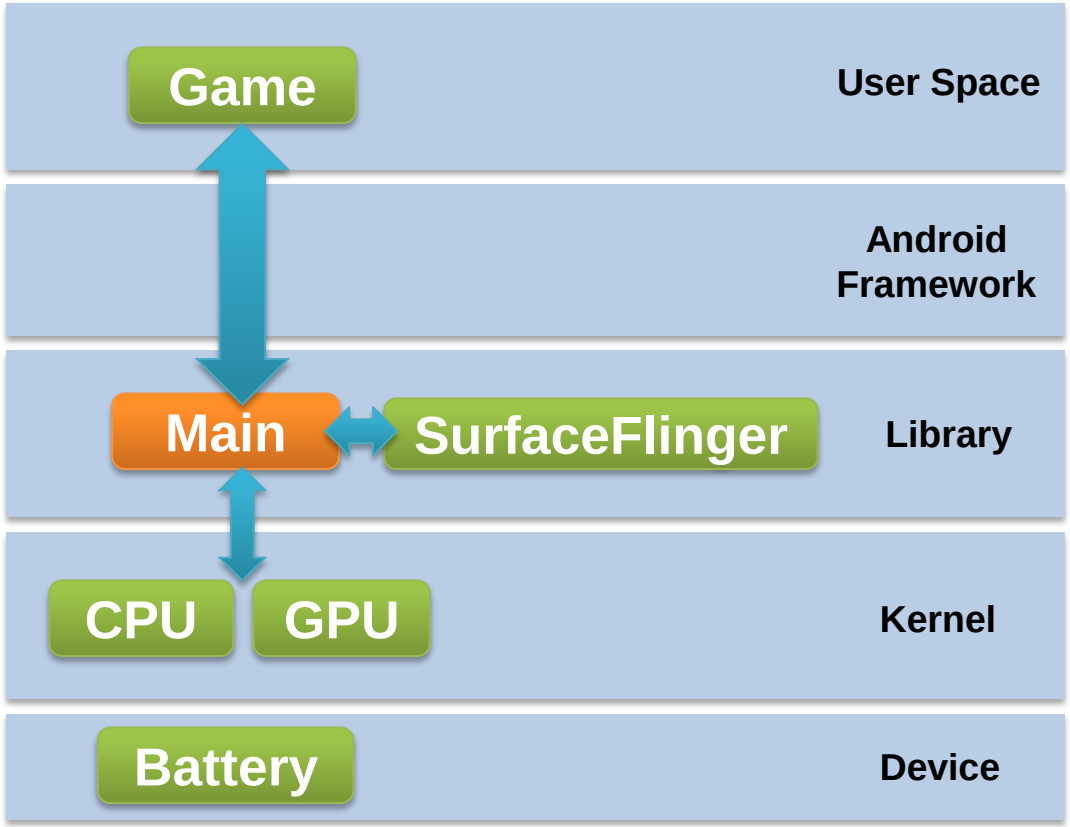
Framework

Main function

CPU/GPU Control

SurfaceFlinger service

IMPLEMENTATION



Framework

Main function

CPU/GPU Control

SurfaceFlinger service

EXPERIMENT

Platform : Google NEXUS 7

System

Android 4.2.2

Kernel

3.1.10

CPU

Quad-Core Cortex A9 1.2GHz

GPU

ULP GeForce 416MHz

EXPERIMENT

Test Items : 3 Games

Break Bricks

Breakout-style games

ASPHALT 8

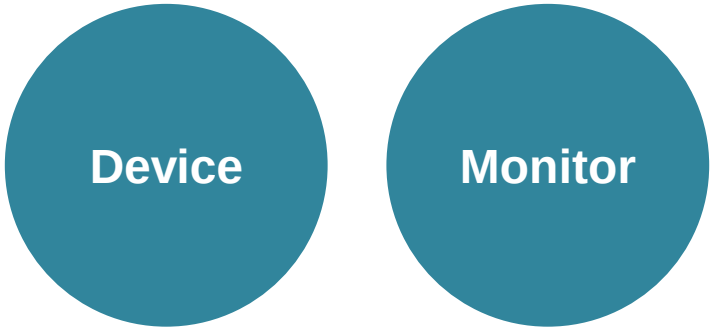
Car racing game

Brain Dots

Puzzling and drawing

EXPERIMENT

Environment : Testing Methodology



Device

Game playing
Play by HiroMacro.

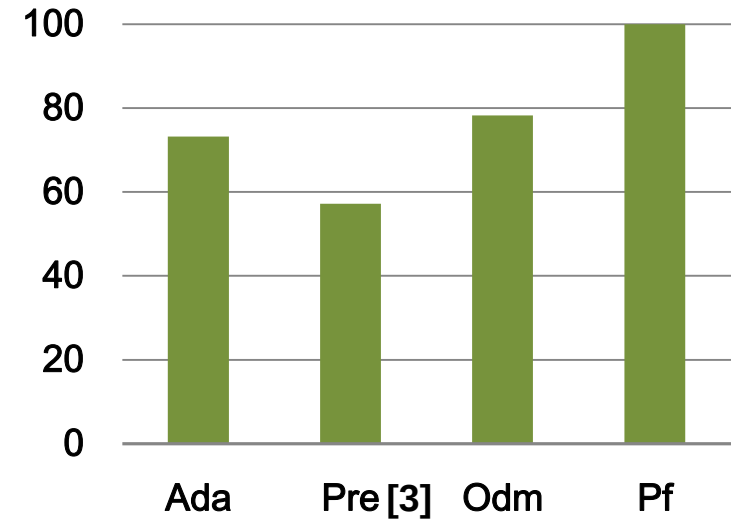
Monitor

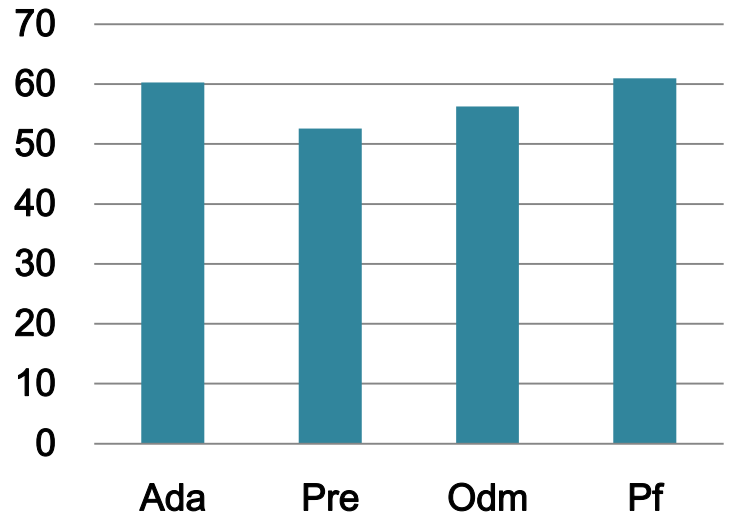
Record power
Running governor

HiroMacro :
https://play.google.com/store/apps/details?id=com.prohiro.macro&hl=zh_TW.

EXPERIMENT

Results : Break Bricks

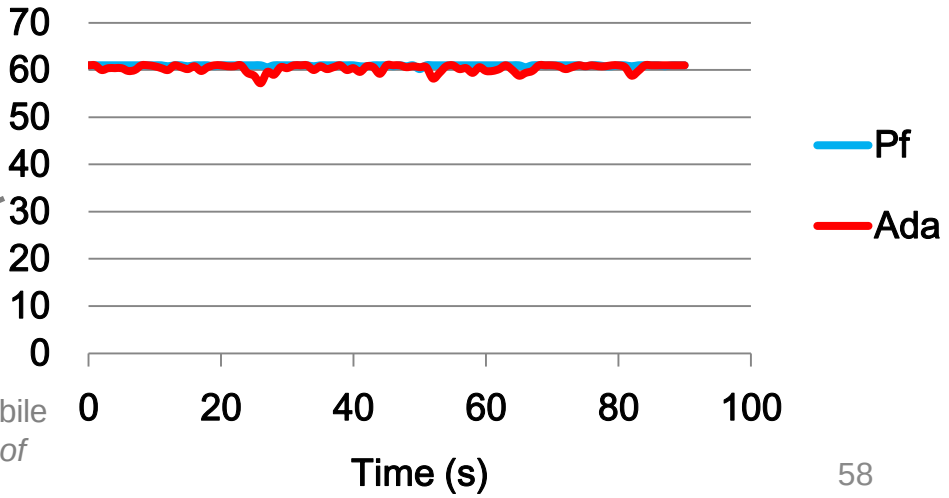




Normalized Average Power

Average FPS

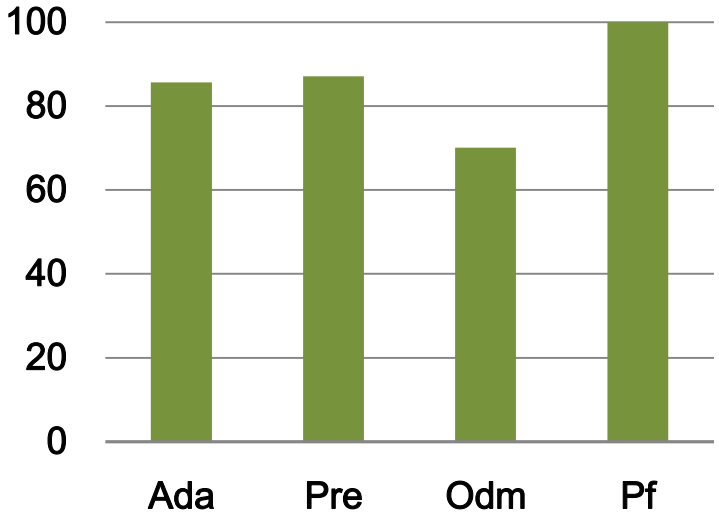
FPS Fluctuation

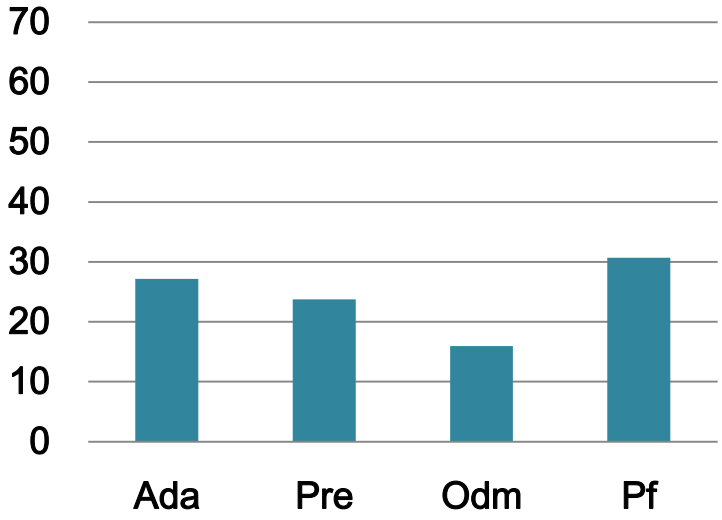


[3] Pathania, Anuj, et al. "Power-performance modelling of mobile gaming workloads on heterogeneous MPSoCs." *Proceedings of the 52nd Annual Design Automation Conference*. ACM, 2015.

EXPERIMENT

Results : ASPHALT 8

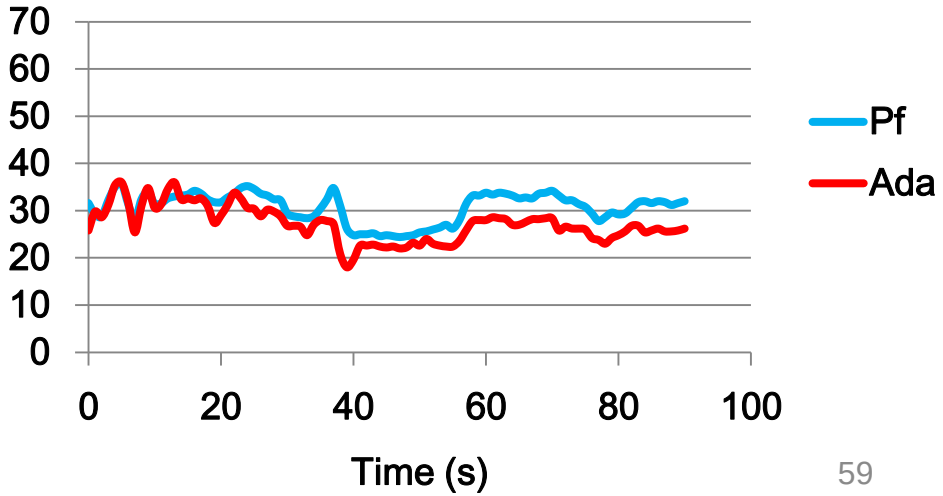




Normalized Average Power

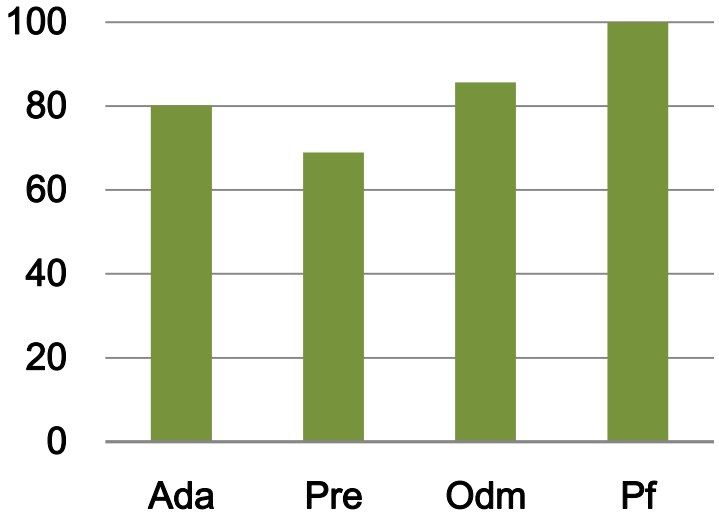
Average FPS

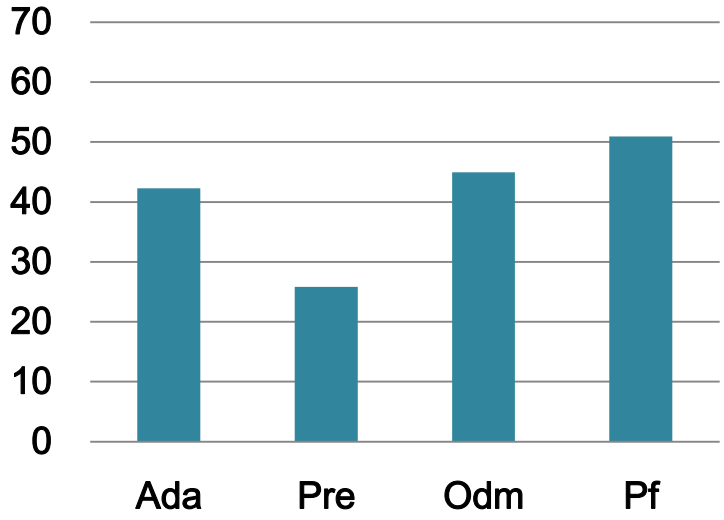
FPS Fluctuation



EXPERIMENT

Results : Brain Dots

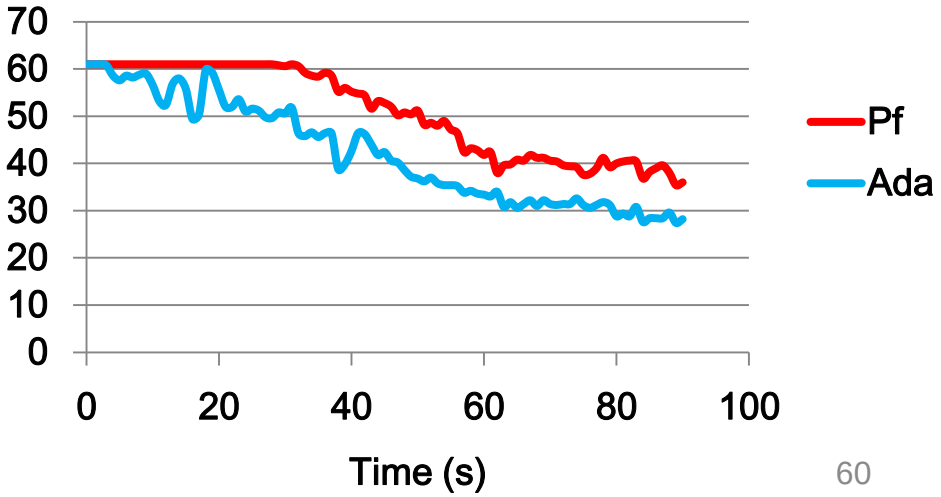




Normalized Average Power

Average FPS

FPS Fluctuation



CONCLUSION

- Power saving governor with quality of user experience
- GPU & CPU frequencies scaling for mobile games
- Adaptive online frequency governor

THANK YOU

Q & A