AARHUS UNIVERSITET

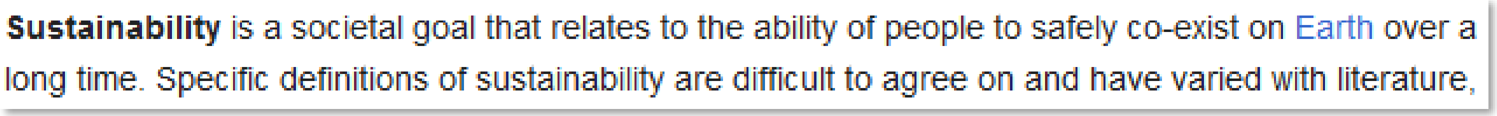
Software Engineering and Architecture

Energy Efficiency

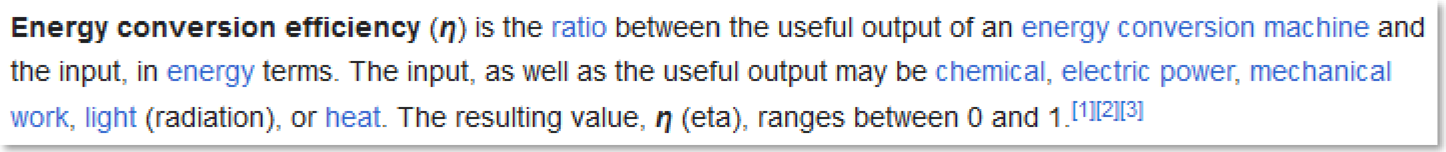
An important Quality Attribute

Motivation

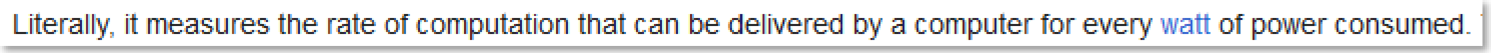
- Well… Large and important topic !


Sustainability is a societal goal that relates to the ability of people to safely co-exist on Earth over a long time. Specific definitions of sustainability are difficult to agree on and have varied with literature,

- I will delimit myself to *energy-efficiency*


Energy conversion efficiency (*η*) is the ratio between the useful output of an energy conversion machine and the input, in energy terms. The input, as well as the useful output may be chemical, electric power, mechanical work, light (radiation), or heat. The resulting value, *η* (eta), ranges between 0 and 1.[1][2][3]

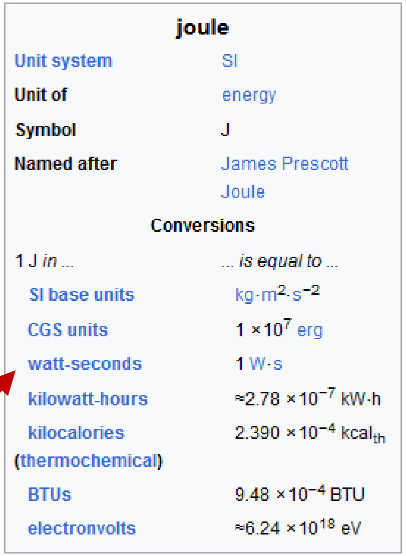
- *… or*


Literally, it measures the rate of computation that can be delivered by a computer for every watt of power consumed.

- Ala: *Patient Inger's blood-pressure is uploaded to server*
  - Architecture A spends **3.1mJ**; Architecture B spends **6.7mJ**
  - *We prefer architecture A, right?*

Energy and Power

- We are basically interested in *energy*
  - Energy = Amount of work
- **Energy** is measured in **Joule** (SI unit)
  - 1J work is done when a force of 1 newton displaces a mass 1 meter
    - Newton = force accelerating 1kg by 1m/s^2
- **Power** is measured in **Watt**
  - Power = energy / second; 1 W = 1 J/s
    - Or…
  - 1 Joule is 1 W in 1 second = 1 Ws
  - **1 KWh = 3.6 MJ**



joule	
Unit system	SI
Unit of	energy
Symbol	J
Named after	James Prescott Joule
Conversions	
1 J *in ...*	... *is equal to ...*
SI base units	kg·m^{2}·s^{-2}
CGS units	1 × 10^{7} erg
watt-seconds	1 W·s
kilowatt-hours	≈2.78 × 10^{-7} kW·h
kilocalories (thermochemical)	2.390 × 10^{-4} kcal$_{th}$
BTUs	9.48 × 10^{-4} BTU
electronvolts	≈6.24 × 10^{18} eV

100g Hellmann's Mayonnaise contains 2,965,000 J.
About 35 min sweaty bicycling…

AARHUS UNIVERSITET

Motivating Example

- Gangnam Style
  - Was shown 1.7 x 10^9 times the first year
  - Energy to stream once is 0.19kWh
  - **Total: 312 GWh**

Journal of Systems and Software
Volume 117, July 2016, Pages 185-198

Empirical evaluation of two best practices for energy-efficient software development

Giuseppe Procaccianti, Héctor Fernández, Patricia Lago

PSY - GANGNAM STYLE [Original Video]

  - Danish average house (“parcelhus”) yearly electricity consumption
    - 4.4 – 5.0 MWh
  - **~ 70.000 Danish houses**

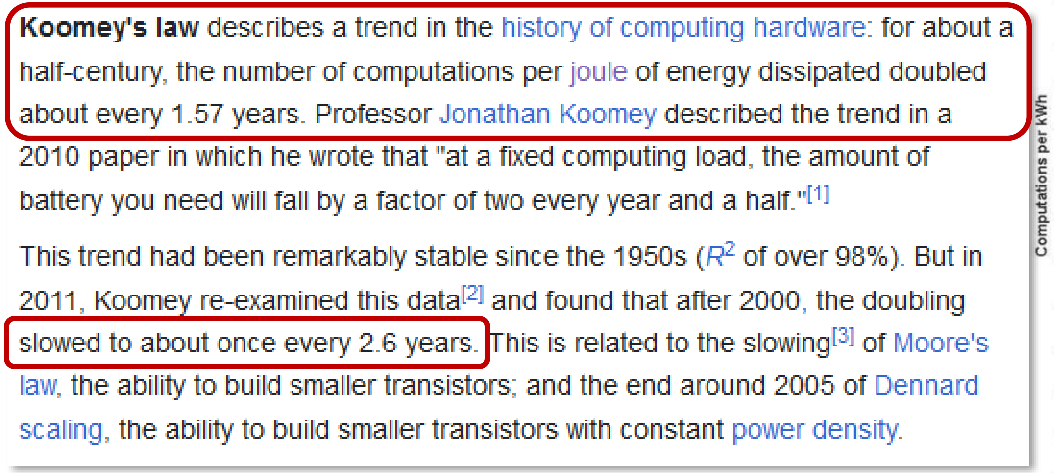
Morale: None…
But it is a bit thought provoking…

Energy = Work Done

Hardware spends energy, because our *Software* wants work to be done.

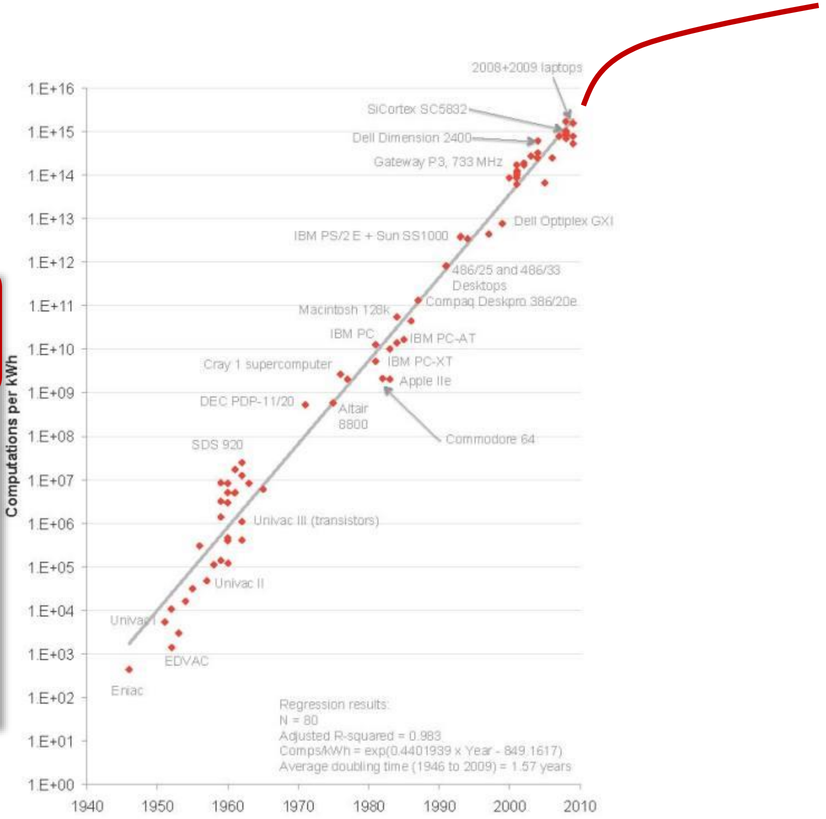
Koomey’s Law

- Hardware consumes energy
  - But is improving all the time!
    - They are the good guys!

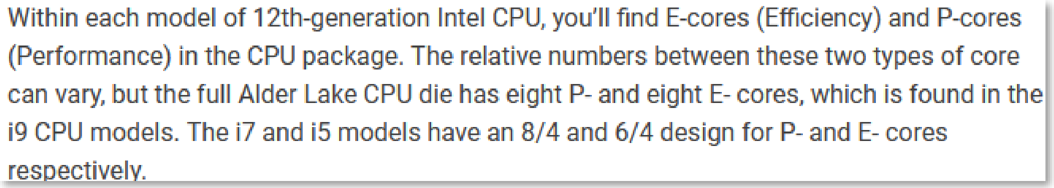


Koomey's law describes a trend in the history of computing hardware: for about a half-century, the number of computations per joule of energy dissipated doubled about every 1.57 years. Professor Jonathan Koomey described the trend in a 2010 paper in which he wrote that "at a fixed computing load, the amount of battery you need will fall by a factor of two every year and a half."[1]

This trend had been remarkably stable since the 1950s (R^2 of over 98%). But in 2011, Koomey re-examined this data[2] and found that after 2000, the doubling slowed to about once every 2.6 years. This is related to the slowing[3] of Moore's law, the ability to build smaller transistors; and the end around 2005 of Dennard scaling, the ability to build smaller transistors with constant power density.



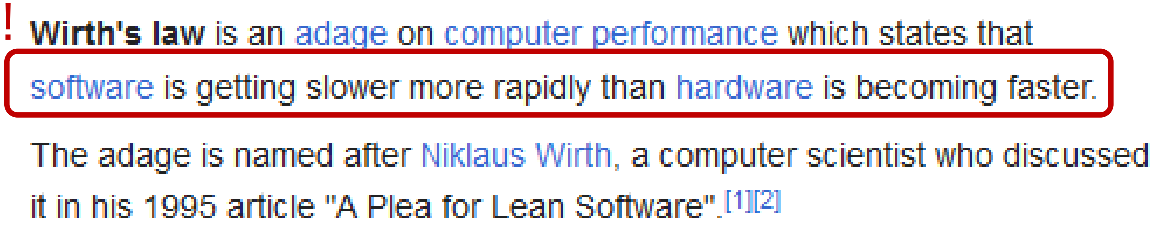
- Intel 12th Gen CPU



Within each model of 12th-generation Intel CPU, you’ll find E-cores (Efficiency) and P-cores (Performance) in the CPU package. The relative numbers between these two types of core can vary, but the full Alder Lake CPU die has eight P- and eight E- cores, which is found in the i9 CPU models. The i7 and i5 models have an 8/4 and 6/4 design for P- and E- cores respectively.

Wirth’s Law

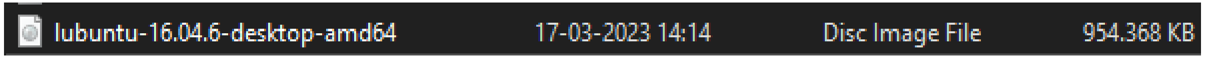
- Unfortunately, we as developers and architects are terrible at writing software or writing too much ☹
  - We are the bad guys!



Wirth's law is an adage on computer performance which states that software is getting slower more rapidly than hardware is becoming faster.

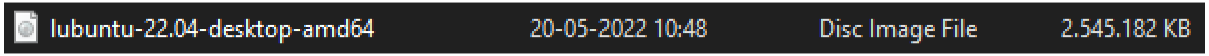
The adage is named after Niklaus Wirth, a computer scientist who discussed it in his 1995 article "A Plea for Lean Software".[1][2]

- Example:
  - For my students I like an easy, but small, linux desktop: *Lubuntu*
  - First used in 2016, easily ran in a 2GB RAM VM ☺



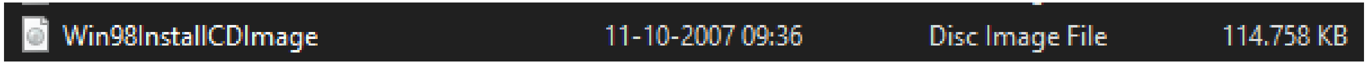
lubuntu-16.04.6-desktop-amd64	17-03-2023 14:14	Disc Image File	954.368 KB

  - Last 22.04 version, has issues running in a 4GB RAM VM ☹



lubuntu-22.04-desktop-amd64	20-05-2022 10:48	Disc Image File	2.545.182 KB

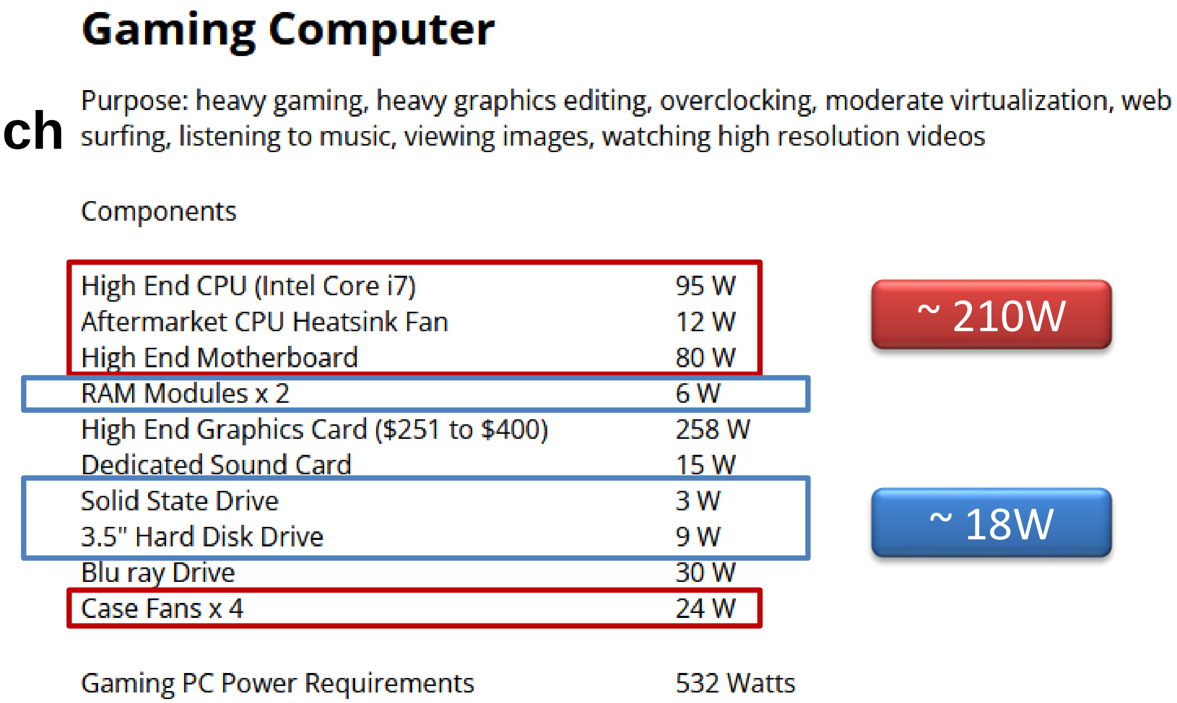
  - And – *In the old days*



Win98InstallCDImage	11-10-2007 09:36	Disc Image File	114.758 KB

What is using Power?

- Note
  - **CPU drives much else**
    - Heat/fan/ cooling
  - Note
    - SSD+DRAM is ‘cheap’ power wise…



Gaming Computer

Purpose: heavy gaming, heavy graphics editing, overclocking, moderate virtualization, web surfing, listening to music, viewing images, watching high resolution videos

Components

High End CPU (Intel Core i7)	95 W
Aftermarket CPU Heatsink Fan	12 W
High End Motherboard	80 W
RAM Modules x 2	6 W
High End Graphics Card ($251 to $400)	258 W
Dedicated Sound Card	15 W
Solid State Drive	3 W
3.5" Hard Disk Drive	9 W
Blu ray Drive	30 W
Case Fans x 4	24 W

Gaming PC Power Requirements 532 Watts

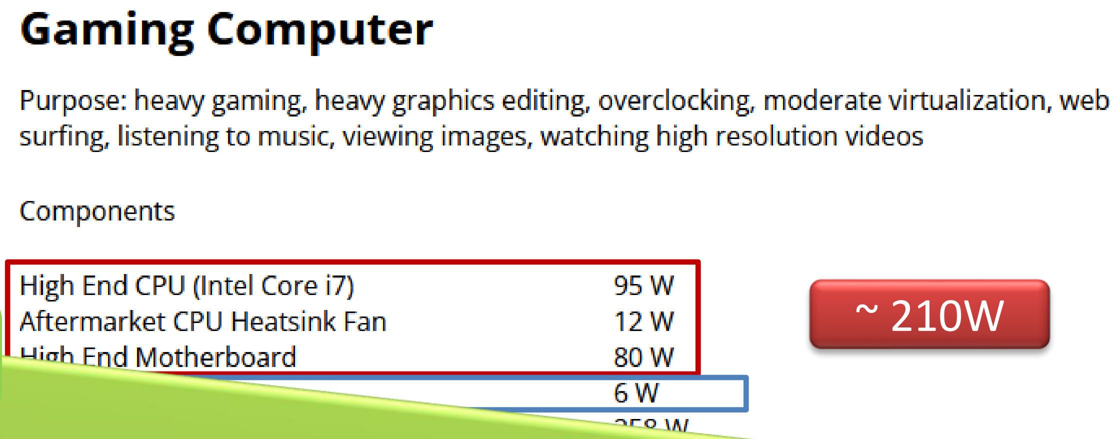
~ 210W

~ 18W

Out-of-box: Network
Devices: Screen, GPS, sensors…

What is using Power?

- Note
  - **CPU drives much else**
    - Heat/fan/ **cooling**
  - Note
    - SSD+DRAM is ‘cheap’ power wise



Gaming Computer

Purpose: heavy gaming, heavy graphics editing, overclocking, moderate virtualization, web surfing, listening to music, viewing images, watching high resolution videos

Components

High End CPU (Intel Core i7)	95 W
Aftermarket CPU Heatsink Fan	12 W
High End Motherboard	80 W
	6 W
	258 W

~ 210W

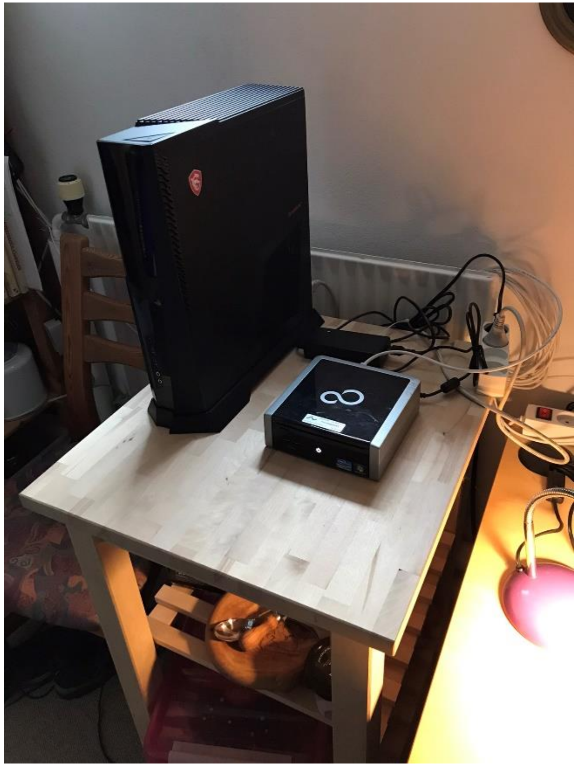
The Apollo 13 Team returned safely to earth, but was heavily *water rationed* during their perilous return flight – because they needed the water to **cool** the guidance computer!

Devices: Screen, GPS, sensors…

AARHUS UNIVERSITET

Examples: My Humble Lab

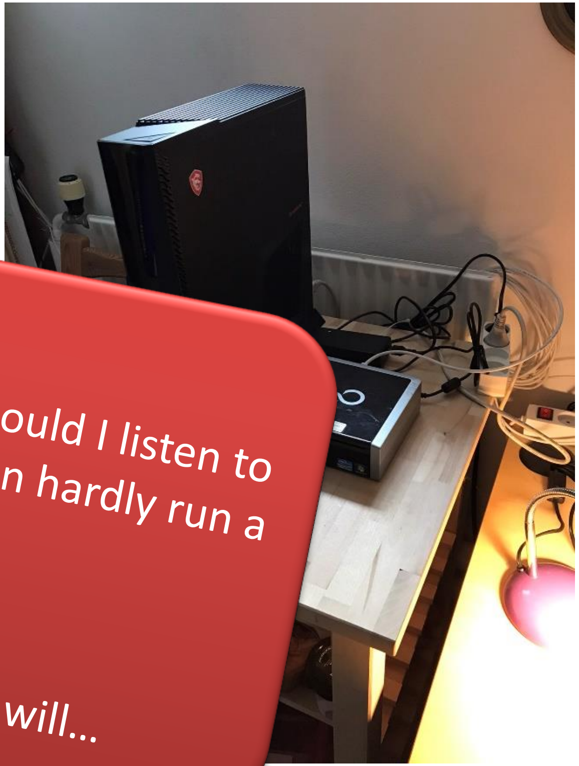
- The Lab
  - Fujitsu Esprimo Q900 (2012)
  - MSI Trident (2020)



- Installed with Ubuntu 22.04 LTS
  - Headless
    - No use for the GeForce RTX™ 2080 Ti ☺
- **Idle** Power Consumption
  - Esprimo: ~ 11 W (plug) / 2.8 W (CPU)
  - MSI: ~ 40 W (plug) / 7.4 W (CPU)
    - At ~95% CPU load@Plug: Esprimo 43W and MSI 160W

Examples: My Humble Lab

- The Lab
  - Fujits o Q900 (2012)
  - MSI
- Insta
  - H
- **Idle** Power
  - Esprimo: ~ 11 W (plug) / 2.
  - MSI: ~ 40 W (plug) / 7.4 W (CPU)
    - At ~95% CPU load@Plug: Esprimo 43W and MSI 160W



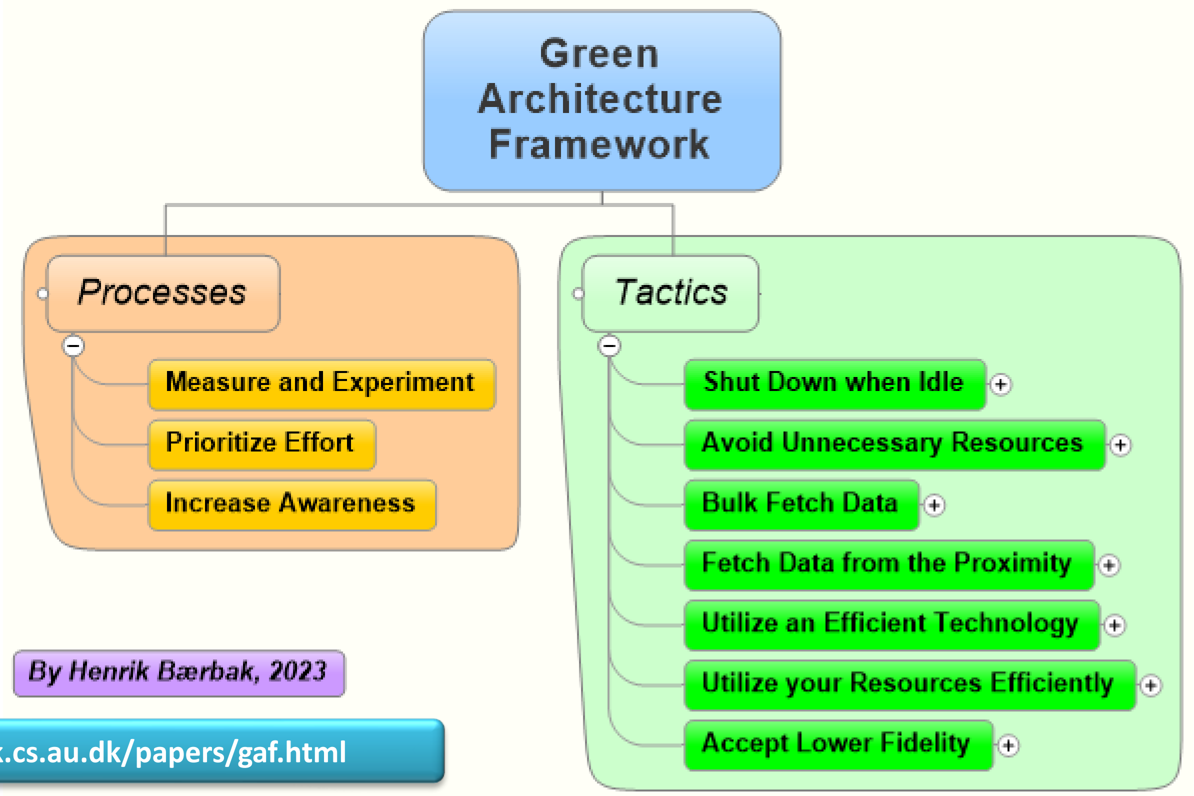
Come on, Henrik???

I run 100.000 instances in the cloud, why should I listen to your experiments on a decade old PC that can hardly run a Windows 10 OS ???

Well – **A computer is a computer.**
The data may not transfer, but the trend will…

The Framework

- *The Green Architecture framework ☺*



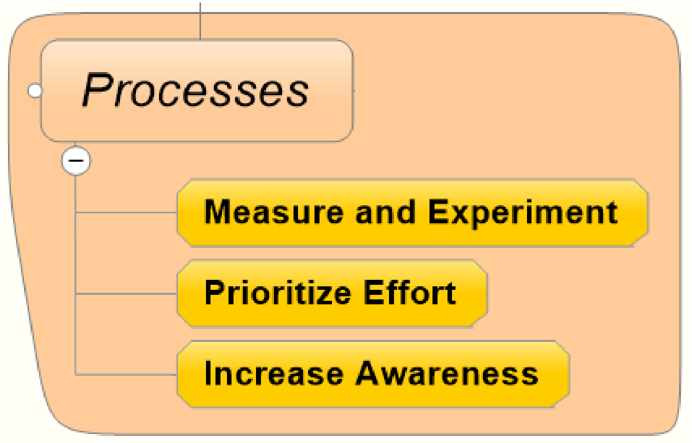
Green Architecture Framework

- *Processes*
  - Measure and Experiment
  - Prioritize Effort
  - Increase Awareness
- *Tactics*
  - Shut Down when Idle
  - Avoid Unnecessary Resources
  - Bulk Fetch Data
  - Fetch Data from the Proximity
  - Utilize an Efficient Technology
  - Utilize your Resources Efficiently
  - Accept Lower Fidelity

By Henrik Bærbak, 2023

https://baerbak.cs.au.dk/papers/gaf.html

AARHUS UNIVERSITET

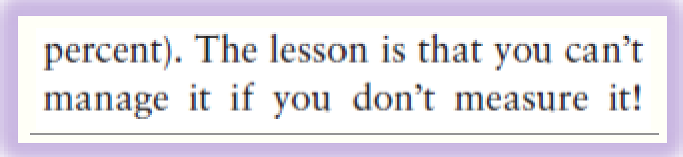


Processes

How we design Green Architectures?

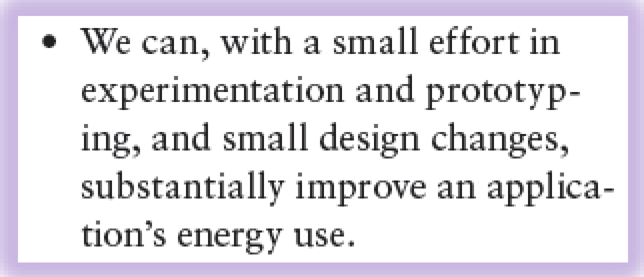
Measure and Experiment

- *You need to measure!*

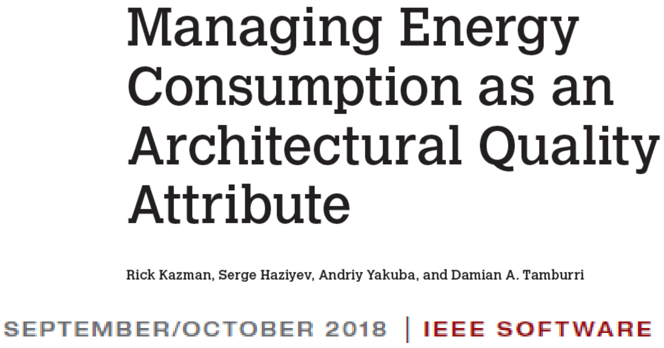


percent). The lesson is that you can't manage it if you don't measure it!

- *You need to experiment!*



- We can, with a small effort in experimentation and prototyping, and small design changes, substantially improve an application's energy use.



Managing Energy Consumption as an Architectural Quality Attribute

Rick Kazman, Serge Haziyev, Andriy Yakuba, and Damian A. Tamburri

SEPTEMBER/OCTOBER 2018 | IEEE SOFTWARE

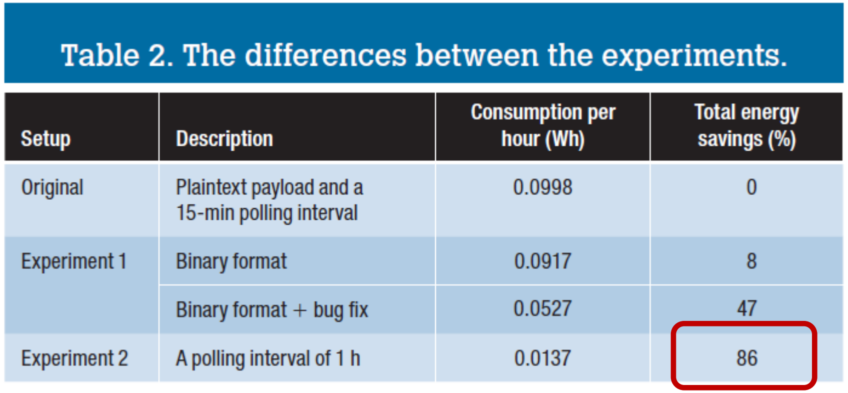


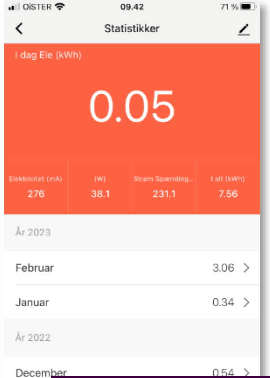
Table 2. The differences between the experiments.

Setup	Description	Consumption per hour (Wh)	Total energy savings (%)
Original	Plaintext payload and a 15-min polling interval	0.0998	0
Experiment 1	Binary format	0.0917	8
	Binary format + bug fix	0.0527	47
Experiment 2	A polling interval of 1 h	0.0137	86

Tactics: Bulk Fetch Data
+ Low Foot-print Data Formats

Measurements

- Measure *the wall power*
  - “Absolute truth”
    - I use a ‘Nedis smart plug’
      - Manual read out ☹
- Measure the ‘on-chip’ power
  - **RAPL**: *Running Average Power Limit*
    - *Only CPU (and DRAM) is measured*
- Virtual Machines?
  - No luck!
    - Cost correlate ☺



OiSTER 09.42 71 %

Statistikker

I dag Ele (kWh)

0.05

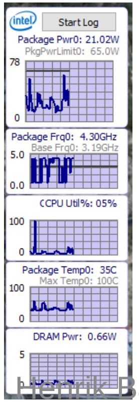
Elektricitet (mA)	(W)	Strøm Spænding...	I alt (kWh)
276	38.1	231.1	7.56

År 2023

Februar	3.06
Januar	0.34

År 2022

December	0.54
Novem	
Oktobe	



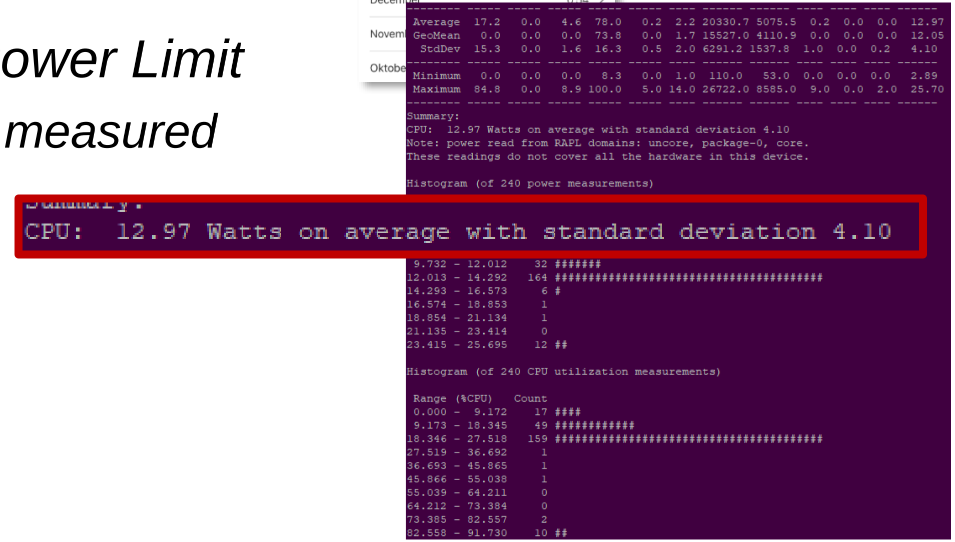
Package Pwr0: 21.02W
PkgPwrLimit0: 65.0W

Package Frq0: 4.30GHz
Base Frq0: 3.19GHz

CCPU Util%: 05%

Package Temp0: 35C
Max Temp0: 100C

DRAM Pwr: 0.66W



```
-------- ----- ----- ----- ----- ----- ---- ------ ------ ---- ---- ---- ------
 Average  17.2   0.0   4.6  78.0   0.2  2.2 20330.7 5075.5  0.2  0.0  0.0  12.97
 GeoMean   0.0   0.0   0.0  73.8   0.0  1.7 15527.0 4110.9  0.0  0.0  0.0  12.05
  StdDev  15.3   0.0   1.6  16.3   0.5  2.0 6291.2 1537.8  1.0  0.0  0.2   4.10
-------- ----- ----- ----- ----- ----- ---- ------ ------ ---- ---- ---- ------
 Minimum   0.0   0.0   0.0   8.3   0.0  1.0  110.0   53.0  0.0  0.0  0.0   2.89
 Maximum  84.8   0.0   8.9 100.0   5.0 14.0 26722.0 8585.0  9.0  0.0  2.0  25.70
-------- ----- ----- ----- ----- ----- ---- ------ ------ ---- ---- ---- ------
Summary:
CPU:  12.97 Watts on average with standard deviation 4.10
Note: power read from RAPL domains: uncore, package-0, core.
These readings do not cover all the hardware in this device.

Histogram (of 240 power measurements)
```

```
Summary:
CPU:  12.97 Watts on average with standard deviation 4.10
```

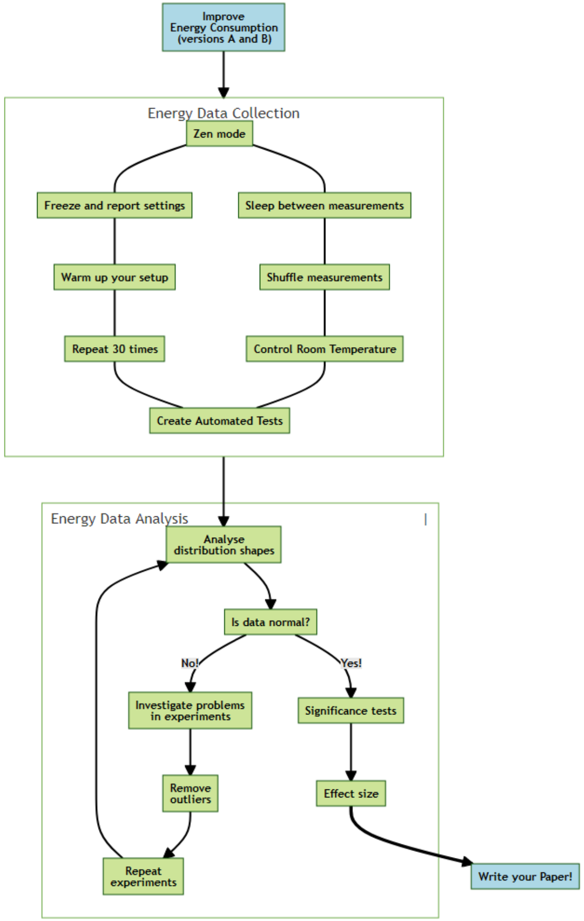
```
 9.732 - 12.012    32 #######
12.013 - 14.292   164 #########################################
14.293 - 16.573     6 #
16.574 - 18.853     1
18.854 - 21.134     1
21.135 - 23.414     0
23.415 - 25.695    12 ##

Histogram (of 240 CPU utilization measurements)

 Range (%CPU)   Count
 0.000 -  9.172    17 ####
 9.173 - 18.345    49 ############
18.346 - 27.518   159 #########################################
27.519 - 36.692     1
36.693 - 45.865     1
45.866 - 55.038     1
55.039 - 64.211     0
64.212 - 73.384     0
73.385 - 82.557     2
82.558 - 91.730    10 ##
```

Be Systematic

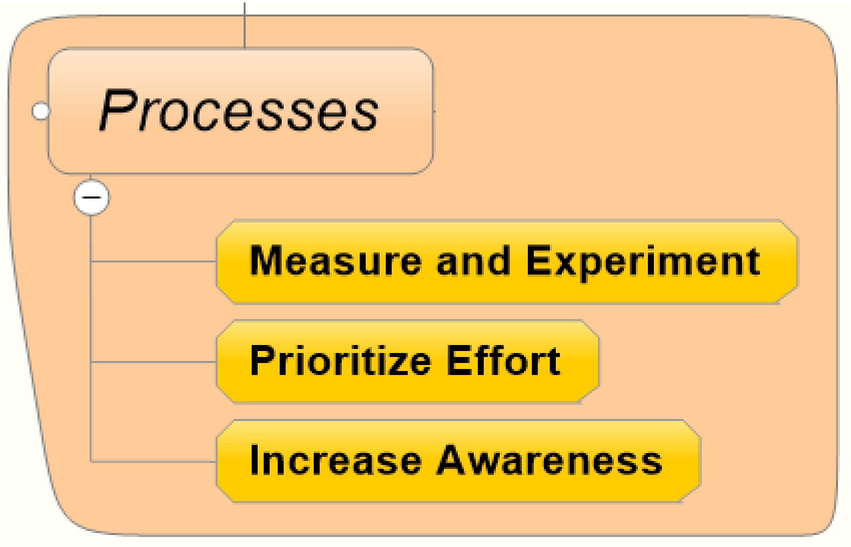
- As in Physics
  - *Control the environment, reduce error sources*
  - *Make many experiments, large sample size*
  - *Use proper statistical methods*



- Luiz Cruz (2021)
  - https://luiscruz.github.io/2021/10/10/scientific-guide.html

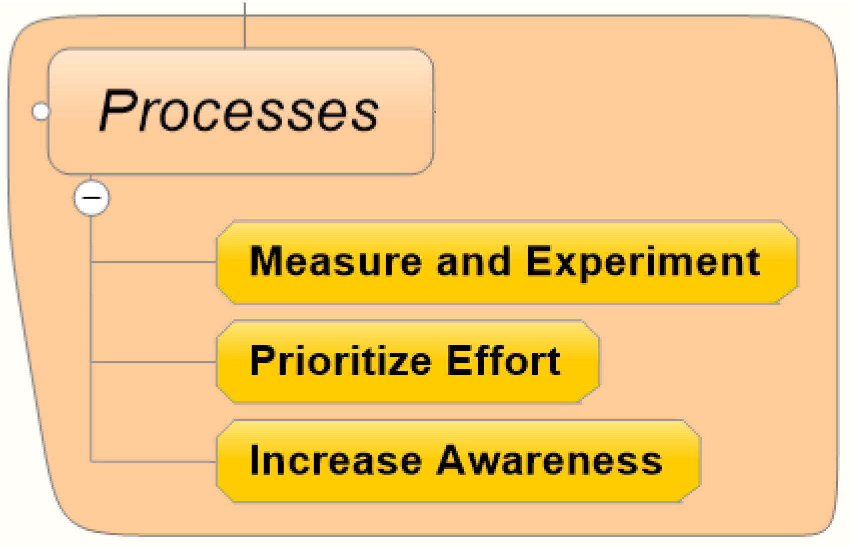
Prioritize Effort

- Prioritize Effort
  - Know the usage profile
    - (by measurements ☺)
  - And invest your effort where it counts
    - Those user stories that are executed the most and that can be optimized the most – are the ones to spend your effort on optimizing
      - Sounds reasonable, but you need to know the usage profile ☺



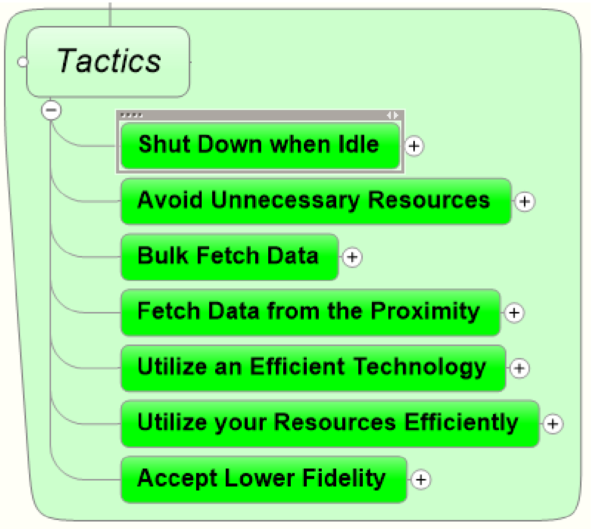
Increase Awareness

- Increase Awareness
  - All architects/developers/ stakeholders informed about how to increase energy-efficiency



- From my kitchen. Which one is 2W and which 40W?
  - You have to tell the kids which one to prefer ☺

AARHUS UNIVERSITET



Tactics

Shut Down when Idle

Avoid Unnecessary Resources

Bulk Fetch Data

Fetch Data from the Proximity

Utilize an Efficient Technology

Utilize your Resources Efficiently

Accept Lower Fidelity

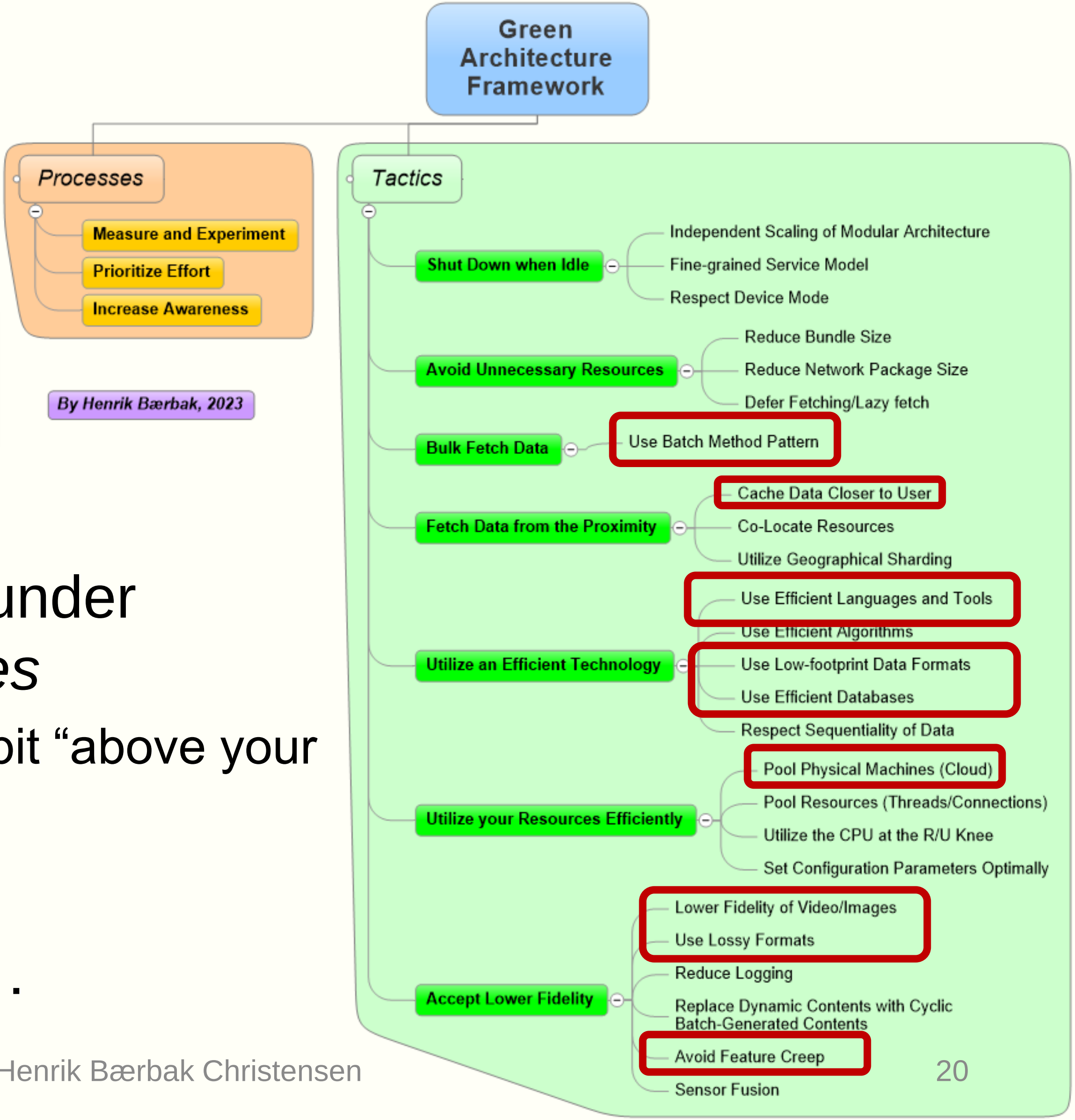
Tactics

How do we then *do*
Green Architecting?

- Set of *tactics*
  - *Architectural design decision to impact energy-efficiency*

- Quite a lot of tactics under *seven main categories*
  - Some of which are a bit "above your current pay-grade" ☺

- SWEA picks follows…

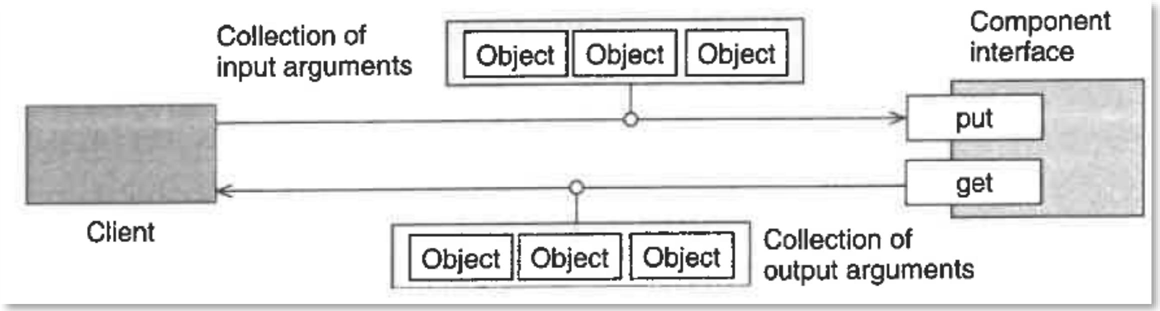


Green Architecture Framework
Processes
Measure and Experiment
Prioritize Effort
Increase Awareness
By Henrik Bærbak, 2023
Tactics
Shut Down when Idle
Independent Scaling of Modular Architecture
Fine-grained Service Model
Respect Device Mode
Avoid Unnecessary Resources
Reduce Bundle Size
Reduce Network Package Size
Defer Fetching/Lazy fetch
Bulk Fetch Data
Use Batch Method Pattern
Fetch Data from the Proximity
Cache Data Closer to User
Co-Locate Resources
Utilize Geographical Sharding
Utilize an Efficient Technology
Use Efficient Languages and Tools
Use Efficient Algorithms
Use Low-footprint Data Formats
Use Efficient Databases
Respect Sequentiality of Data
Utilize your Resources Efficiently
Pool Physical Machines (Cloud)
Pool Resources (Threads/Connections)
Utilize the CPU at the R/U Knee
Set Configuration Parameters Optimally
Accept Lower Fidelity
Lower Fidelity of Video/Images
Use Lossy Formats
Reduce Logging
Replace Dynamic Contents with Cyclic Batch-Generated Contents
Avoid Feature Creep
Sensor Fusion

AARHUS UNIVERSITET

Bulk Fetch Data

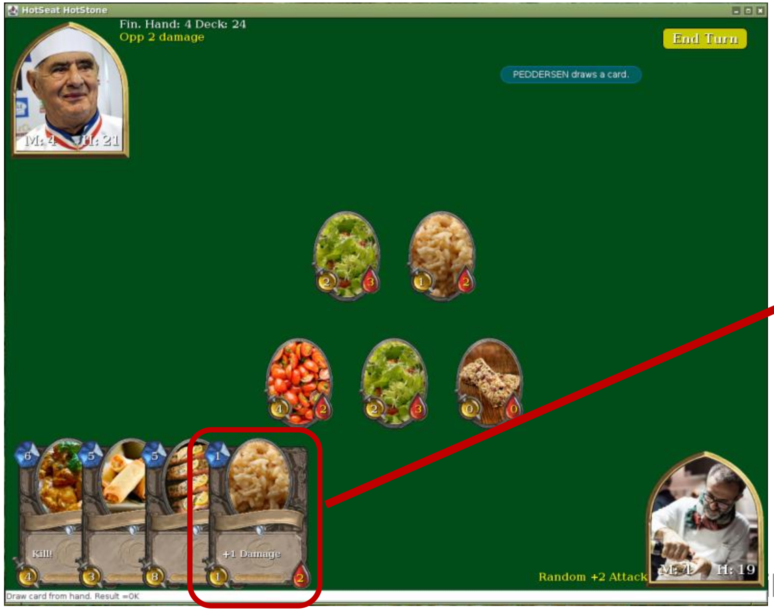
- *“Buy 50 things at the super market once, instead of making 50 trips buying a single thing”*
  - POSA4(2007): **Batch Method**

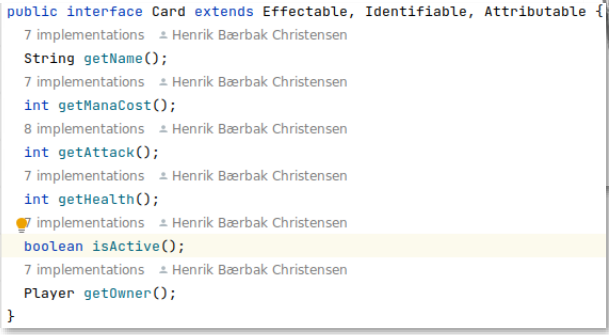


- **Iterator pattern** is an energy *anti pattern*
  - *getNext()* across the network is a *chatty interface*
  - Use *pagination* instead – bulk fetch next 50 items in one chunk

Bulk Fetch Data

- *“Buy 50 things at the super market once, instead of making 50 trips buying a single thing”*
- Example
  - Classic OO is often a very *fine-grained API*



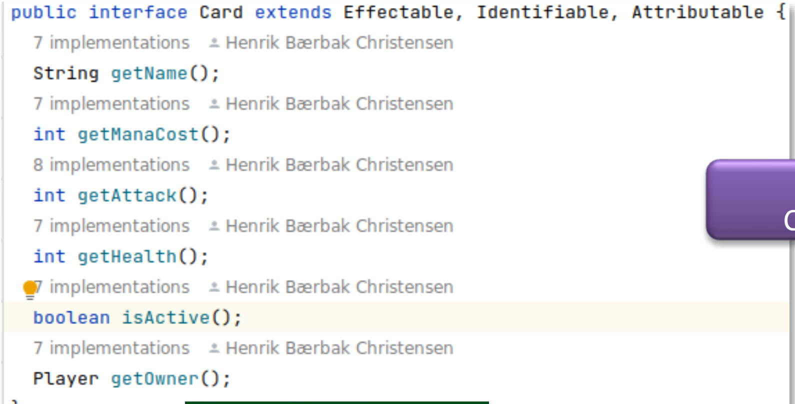


```
public interface Card extends Effectable, Identifiable, Attributable {
  7 implementations  ≗ Henrik Bærbak Christensen
  String getName();
  7 implementations  ≗ Henrik Bærbak Christensen
  int getManaCost();
  8 implementations  ≗ Henrik Bærbak Christensen
  int getAttack();
  7 implementations  ≗ Henrik Bærbak Christensen
  int getHealth();
  7 implementations  ≗ Henrik Bærbak Christensen
  boolean isActive();
  7 implementations  ≗ Henrik Bærbak Christensen
  Player getOwner();
}
```

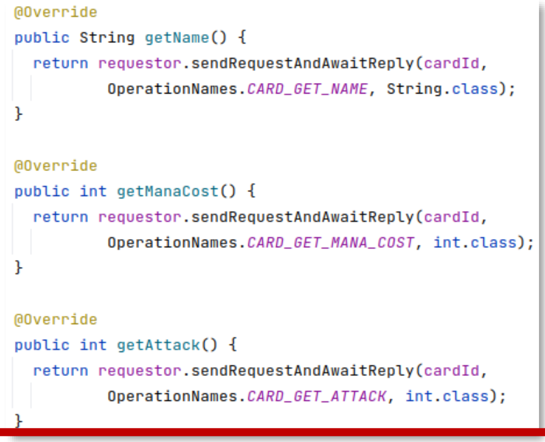
The UI needs to get all card data from the server when redrawing UI…

Example of A/B Architecture

- The Card interface
  - **Two** remote implementations



```
public interface Card extends Effectable, Identifiable, Attributable {
  7 implementations  ≗ Henrik Bærbak Christensen
  String getName();
  7 implementations  ≗ Henrik Bærbak Christensen
  int getManaCost();
  8 implementations  ≗ Henrik Bærbak Christensen
  int getAttack();
  7 implementations  ≗ Henrik Bærbak Christensen
  int getHealth();
  7 implementations  ≗ Henrik Bærbak Christensen
  boolean isActive();
  7 implementations  ≗ Henrik Bærbak Christensen
  Player getOwner();
}
```



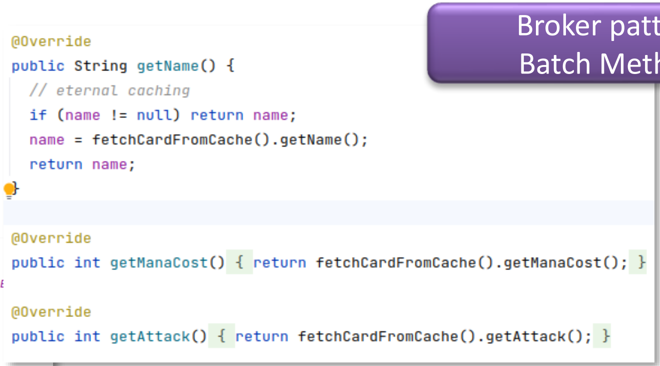
```
@Override
public String getName() {
  return requestor.sendRequestAndAwaitReply(cardId,
          OperationNames.CARD_GET_NAME, String.class);
}

@Override
public int getManaCost() {
  return requestor.sendRequestAndAwaitReply(cardId,
          OperationNames.CARD_GET_MANA_COST, int.class);
}

@Override
public int getAttack() {
  return requestor.sendRequestAndAwaitReply(cardId,
          OperationNames.CARD_GET_ATTACK, int.class);
}
```

Broker pattern
Classic (RMI-like)

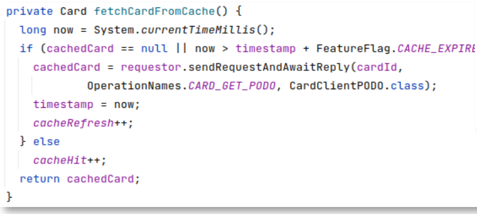
Broker pattern
Batch Method



```
@Override
public String getName() {
  // eternal caching
  if (name != null) return name;
  name = fetchCardFromCache().getName();
  return name;
}

@Override
public int getManaCost() { return fetchCardFromCache().getManaCost(); }

@Override
public int getAttack() { return fetchCardFromCache().getAttack(); }
```



```
private Card fetchCardFromCache() {
  long now = System.currentTimeMillis();
  if (cachedCard == null || now > timestamp + FeatureFlag.CACHE_EXPIRE
    cachedCard = requestor.sendRequestAndAwaitReply(cardId,
            OperationNames.CARD_GET_PODO, CardClientPODO.class);
    timestamp = now;
    cacheRefresh++;
  } else
    cacheHit++;
  return cachedCard;
}
```

a) Cache 5 secs b) Get PODO

Bulk Fetch Data

- *"Buy 50 things at the super market once, instead of making 50 trips buying a single thing"*
- Comparison
  - Classic Broker (the one *you make*)
    - 5.66W (σ 0.90W)
  - Batch Method Broker (using caching)
    - 4.12W (σ 0.79W)
    - (Reducing number of network calls to 43%)
  - Saving **27% energy**

- ***And this is on the server side only!***

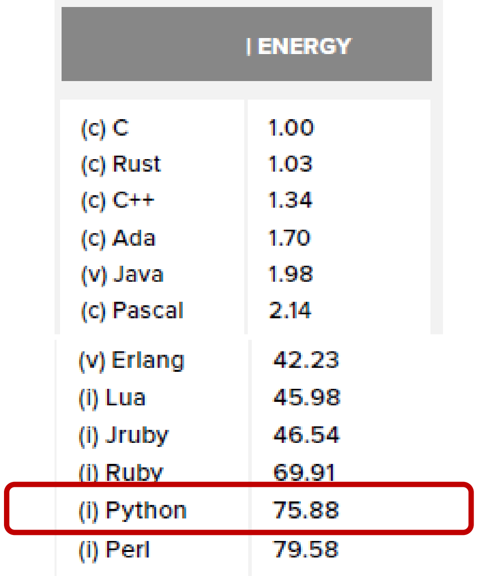
Fetch Data from the Proximity

- *“Have a stock of supplies to avoid a lot of trips to the super market”*

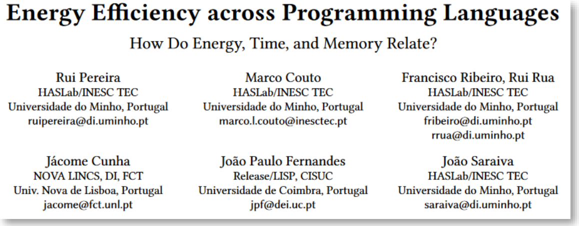
- **Cache Data Closer to User**
  - The *Batch Method Broker* is one such example
  - Content-Delivery-Networks (CDN)
    - Store web contents (caching) physically near to the users to provide faster load times by avoiding “long distance network transmission”

Utilize an Efficient Technology

- *“Switch the 20 W halogen bulb to a 4 W LED bulb”*
- **Use Efficient Languages and Tools**
  - (This 2017 study used rather unrealistic benchmark programs)
    - Mandelbrot???



	ENERGY
(c) C	1.00
(c) Rust	1.03
(c) C++	1.34
(c) Ada	1.70
(v) Java	1.98
(c) Pascal	2.14
(v) Erlang	42.23
(i) Lua	45.98
(i) Jruby	46.54
(i) Ruby	69.91
(i) Python	75.88
(i) Perl	79.58



Energy Efficiency across Programming Languages

How Do Energy, Time, and Memory Relate?

Rui Pereira
HASLab/INESC TEC
Universidade do Minho, Portugal
ruipereira@di.uminho.pt

Marco Couto
HASLab/INESC TEC
Universidade do Minho, Portugal
marco.l.couto@inesctec.pt

Francisco Ribeiro, Rui Rua
HASLab/INESC TEC
Universidade do Minho, Portugal
fribeiro@di.uminho.pt
rrua@di.uminho.pt

Jácome Cunha
NOVA LINCS, DI, FCT
Univ. Nova de Lisboa, Portugal
jacome@fct.unl.pt

João Paulo Fernandes
Release/LISP, CISUC
Universidade de Coimbra, Portugal
jpf@dei.uc.pt

João Saraiva
HASLab/INESC TEC
Universidade do Minho, Portugal
saraiva@di.uminho.pt

Own experiment of a 3 endpoint REST Service impl:

Java (baseline)

Go (-3.5% energy)

Scala (+27% energy)

Python (+162%, 2½x)

[SideBar]

- Statistics is rather essential here
  - Thanks to *Markus* and all his predecessors ☺
- Question
  - *Is Go really 3.5% more efficient or is it due to statistical uncertainty?*
- Answer

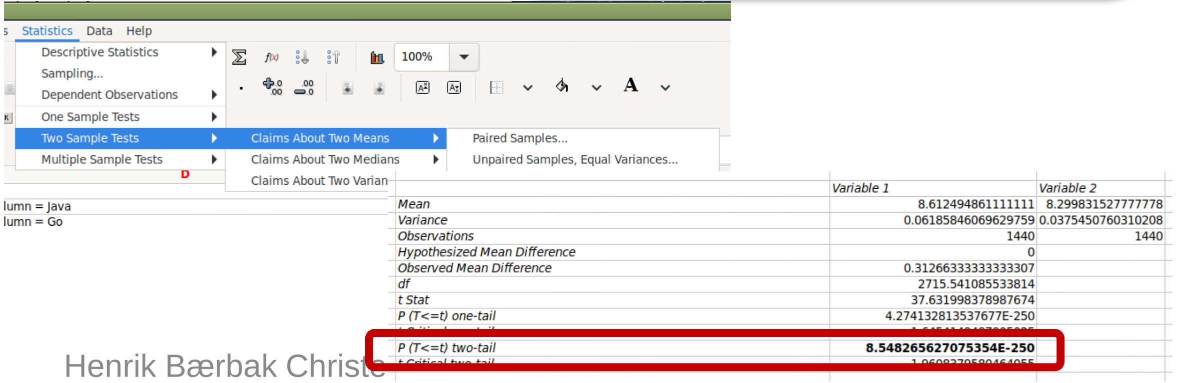
We can formulate our hypothesis test as follows:

H_0: The means of energy consumption of versions A and B are equal.

H_1: The means of energy consumption of versions A and B are different.

Own experiment of a 3 endpoint REST Service impl:
Java (baseline)
Go (-3.5% energy)
Scala (+27% energy)
Python (+162%, 2½x)

- Do a Welch T-test
  - Using Gnumeric
  - p-value < 5% ☺

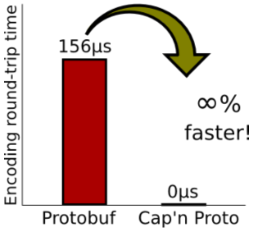


lumn = Java
lumn = Go

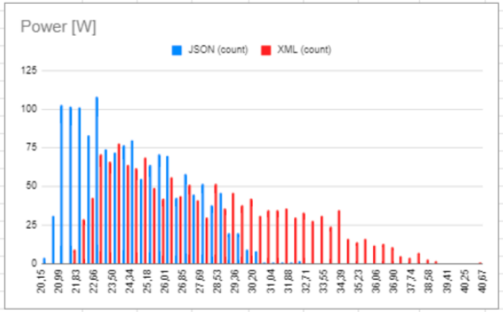
	Variable 1	Variable 2
Mean	8.6124948611111	8.29983152777778
Variance	0.06185846069629759	0.0375450760310208
Observations	1440	1440
Hypothesized Mean Difference	0	
Observed Mean Difference	0.312663333333307	
df	2715.541085533814	
t Stat	37.631998378987674	
P (T<=t) one-tail	4.274132813537677E-250	
[illegible]	[illegible]	
P (T<=t) two-tail	**8.548265627075354E-250**	
t Critical two-tail	[illegible]	

Utilize an Efficient Technology

- *“Switch the 20 W halogen bulb to a 4 W LED bulb”*
- **Use Low-footprint Data Formats**
  - *XML is much more verbose than JSON*
  - Binary formats: ProtoBuf, Cap’n Proto



– Part-time students did a XML versus JSON experiment



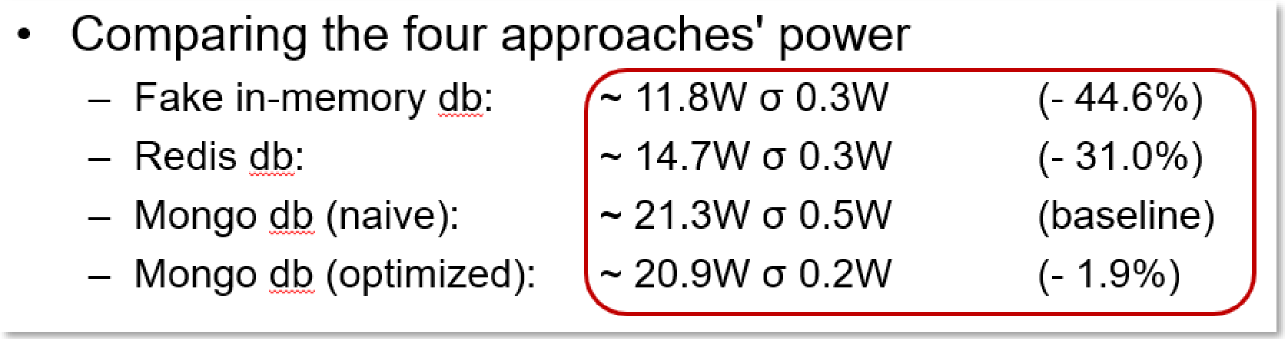
JSON: 24.5W (σ 2.5w)
XML: 27.9W (σ 4.1W)
That is 12.2% saved energy by using JSON over XML

Utilize an Efficient Technology

- *“Switch the 20 W halogen bulb to a 4 W LED bulb”*
- **Use Efficient Databases**
  - If only a ‘blob storage’ / key-value store is necessary then pick one, rather than a SQL or a MongoDB database

– Example

- REST service (three endpoints: One POST and two GET)



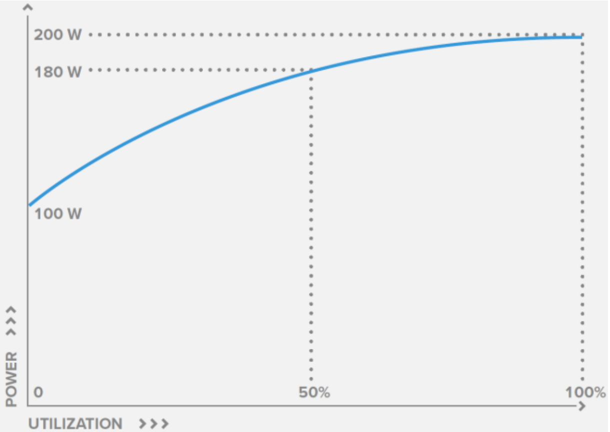
- Comparing the four approaches' power

– Fake in-memory db:	~ 11.8W σ 0.3W	(- 44.6%)
– Redis db:	~ 14.7W σ 0.3W	(- 31.0%)
– Mongo db (naive):	~ 21.3W σ 0.5W	(baseline)
– Mongo db (optimized):	~ 20.9W σ 0.2W	(- 1.9%)

Utilize your Resources Efficiently

- *“Prepare several items in the oven at the same time”*
- An *idling* computer spends between 1/4 - 2/3 power compared to a *busy* computer
  - The *non-proportionality of energy consumption*

- Which means:
  - **Per-transaction energy cost is lowering as the computer is more heavily utilized**



AARHUS UNIVERSITET

Utilize your Resources Efficiently

- *“Prepare several items in the oven at the same time”*
- **Pool Physical Machines (Cloud)**
  - Host a lot of VM on same physical machine means *when A is not using the CPU, then B have it*
    - *Cloud centers are better at that than on-premise*

- **Pool Resources (Threads/Connections)**
  - Threads and connections are expensive to create and deallocate
    - Pool them

Own experiment: Three-tier system with MariaDB storage. A) Naïve ‘connection-pr-request’ connector; B) C3P0 ‘pool’.
Pooled connection spent **about 62.5% less energy**.

Naïve: 192tps
C3P0: 514 tps

Disclaimer: Naïve had coarse-grained locks applied…

Accept Lower Fidelity

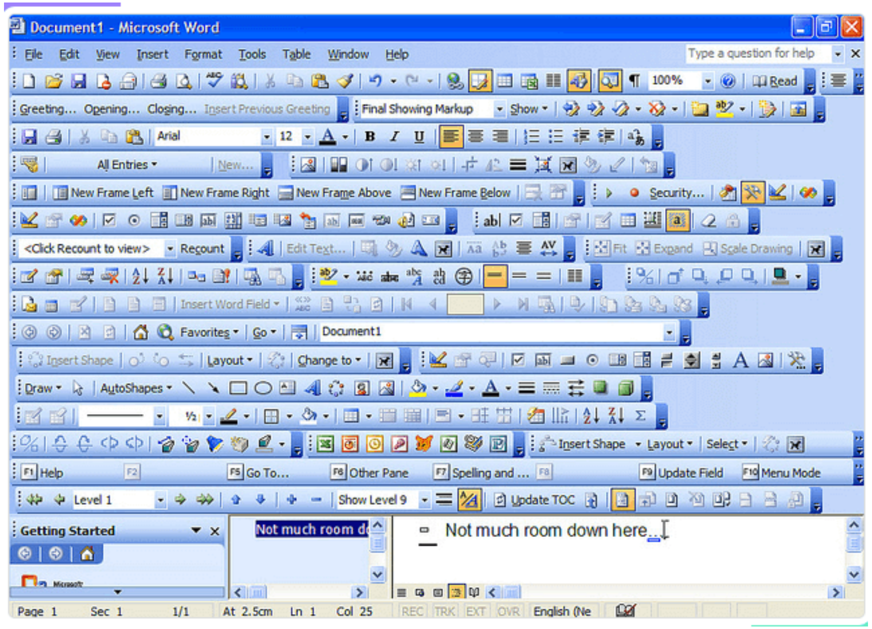
- *"Turn the room temperature down from 21° to 19°"*
- **Replace Dynamic Contents with Batch**
    - Change webpage dynamic content (expensive) with batch once-per-hour (or per-day) computation of a static webpage (cheap)

- **Lower Fidelity of Video/Images**
    - Use 720p instead of 1080p (halves the size)
    - Downscale images server side
        - Use JPEG rather than GIF/PNG

AARHUS UNIVERSITET

Accept Lower Fidelity

- *“Turn the room temperature down from 21° to 19°”*
- **Avoid Feature Creep**
  - Do we really need it all???



Not much room down here.

Maybe it is about time, we start taking *out* features, instead of just adding more!

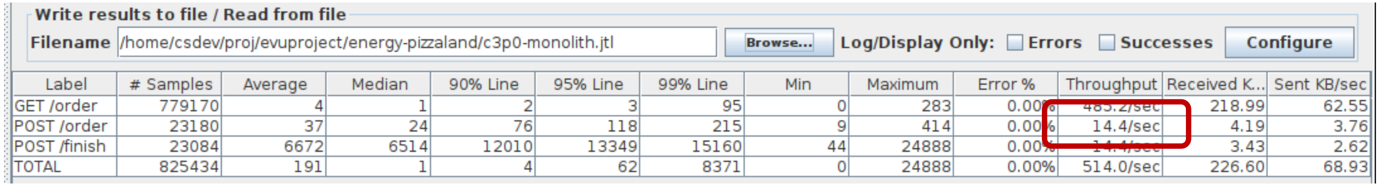
Light editions with Green Label.

Accept Lower Fidelity

- **Avoid Feature Creep**

‘PizzaLand’ Experiment:
A ‘core’ REST based pizza ordering system with ordering and inventory system in MariaDB; deployed on a 2012 i5 CPU @ 2.5GHz/4 core + 8GB DDR3 RAM

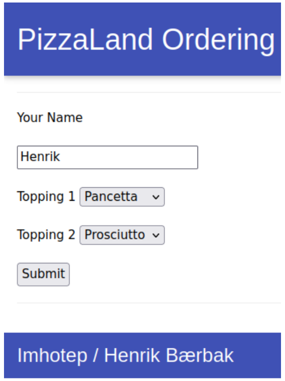
Handles 51,800 orders per hour!



Write results to file / Read from file

Filename /home/csdev/proj/evuproject/energy-pizzaland/c3p0-monolith.jtl

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received K...	Sent KB/sec
GET /order	779170	4	1	2	3	95	0	283	0.00%	485.2/sec	218.99	62.55
POST /order	23180	37	24	76	118	215	9	414	0.00%	14.4/sec	4.19	3.76
POST /finish	23084	6672	6514	12010	13349	15160	44	24888	0.00%	14.4/sec	3.43	2.62
TOTAL	825434	191	1	4	62	8371	0	24888	0.00%	514.0/sec	226.60	68.93



PizzaLand Ordering

Your Name: Henrik

Topping 1: Pancetta

Topping 2: Prosciutto

Imhotep / Henrik Bærbak

And a Category Missing…

… to be added to GAF

Do Not Buy New Stuff

- *Potential Addition* to Green Architecture Framework
  - **Keep the old machines running**
    - More of a hardware tactic but…
  - Laptop running 8h a day for 4 years

Daily computations	61.5 kg CO2eq
Production and Shipping	361 kg CO2eq

  - *Thus around 75-85% of total emissions is manufacturing!*

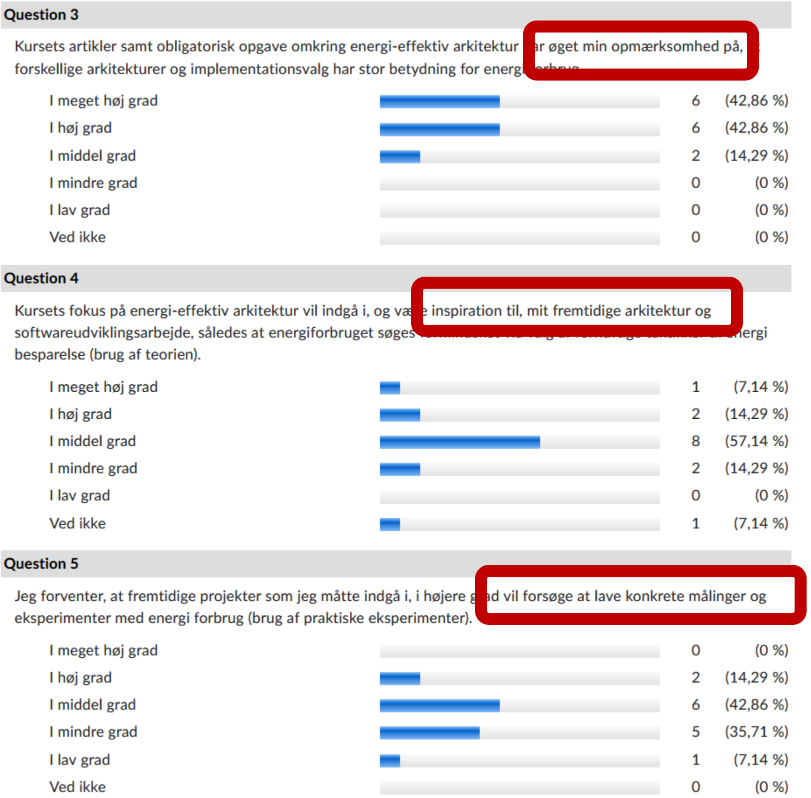
https://circularcomputing.com/news/carbon-footprint-laptop/

Summary and Discussion

AARHUS UNIVERSITET

Summary

- Questionnaire to part-time students



Question 3

Kursets artikler samt obligatorisk opgave omkring energi-effektiv arkitektur har øget min opmærksomhed på, at forskellige arkitekturer og implementationsvalg har stor betydning for energiforbrug.

I meget høj grad	6	(42,86 %)
I høj grad	6	(42,86 %)
I middel grad	2	(14,29 %)
I mindre grad	0	(0 %)
I lav grad	0	(0 %)
Ved ikke	0	(0 %)

Question 4

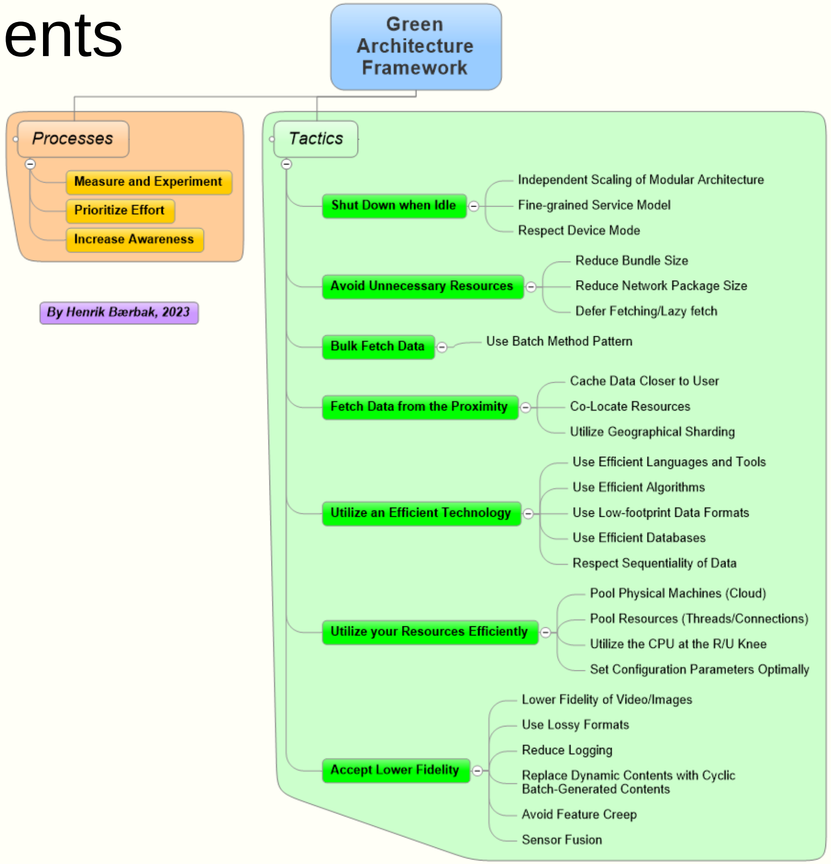
Kursets fokus på energi-effektiv arkitektur vil indgå i, og være inspiration til, mit fremtidige arkitektur og softwareudviklingsarbejde, således at energiforbruget søges [illegible] energi besparelse (brug af teorien).

I meget høj grad	1	(7,14 %)
I høj grad	2	(14,29 %)
I middel grad	8	(57,14 %)
I mindre grad	2	(14,29 %)
I lav grad	0	(0 %)
Ved ikke	1	(7,14 %)

Question 5

Jeg forventer, at fremtidige projekter som jeg måtte indgå i, i højere grad vil forsøge at lave konkrete målinger og eksperimenter med energi forbrug (brug af praktiske eksperimenter).

I meget høj grad	0	(0 %)
I høj grad	2	(14,29 %)
I middel grad	6	(42,86 %)
I mindre grad	5	(35,71 %)
I lav grad	1	(7,14 %)
Ved ikke	0	(0 %)



Green Architecture Framework

Processes
- Measure and Experiment
- Prioritize Effort
- Increase Awareness

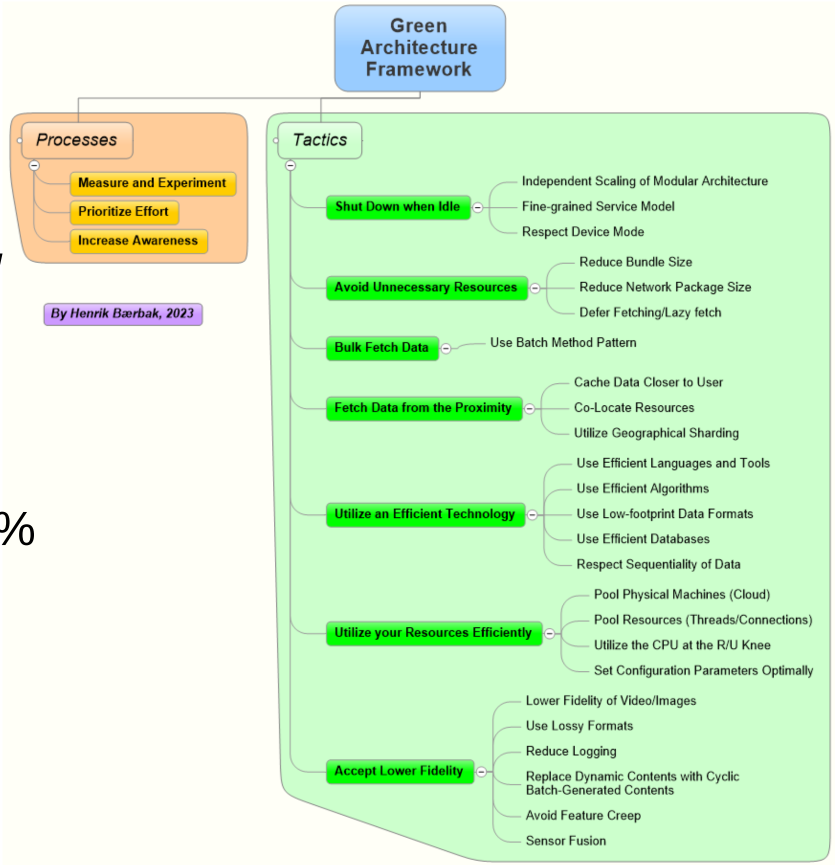
By Henrik Bærbak, 2023

Tactics
- Shut Down when Idle
  - Independent Scaling of Modular Architecture
  - Fine-grained Service Model
  - Respect Device Mode
- Avoid Unnecessary Resources
  - Reduce Bundle Size
  - Reduce Network Package Size
  - Defer Fetching/Lazy fetch
- Bulk Fetch Data
  - Use Batch Method Pattern
- Fetch Data from the Proximity
  - Cache Data Closer to User
  - Co-Locate Resources
  - Utilize Geographical Sharding
- Utilize an Efficient Technology
  - Use Efficient Languages and Tools
  - Use Efficient Algorithms
  - Use Low-footprint Data Formats
  - Use Efficient Databases
  - Respect Sequentiality of Data
- Utilize your Resources Efficiently
  - Pool Physical Machines (Cloud)
  - Pool Resources (Threads/Connections)
  - Utilize the CPU at the R/U Knee
  - Set Configuration Parameters Optimally
- Accept Lower Fidelity
  - Lower Fidelity of Video/Images
  - Use Lossy Formats
  - Reduce Logging
  - Replace Dynamic Contents with Cyclic Batch-Generated Contents
  - Avoid Feature Creep
  - Sensor Fusion

Summary

- However, they all require an **investment…**
  - – *More complex code*
  - – *Experiments are time consuming!*

- Low Hanging Fruits (?)
  - Get utilization of CPUs up to ~75%
  - Low-footprint data formats
  - Bulk fetch data + Caching
  - Keep your old machines running



Green Architecture Framework

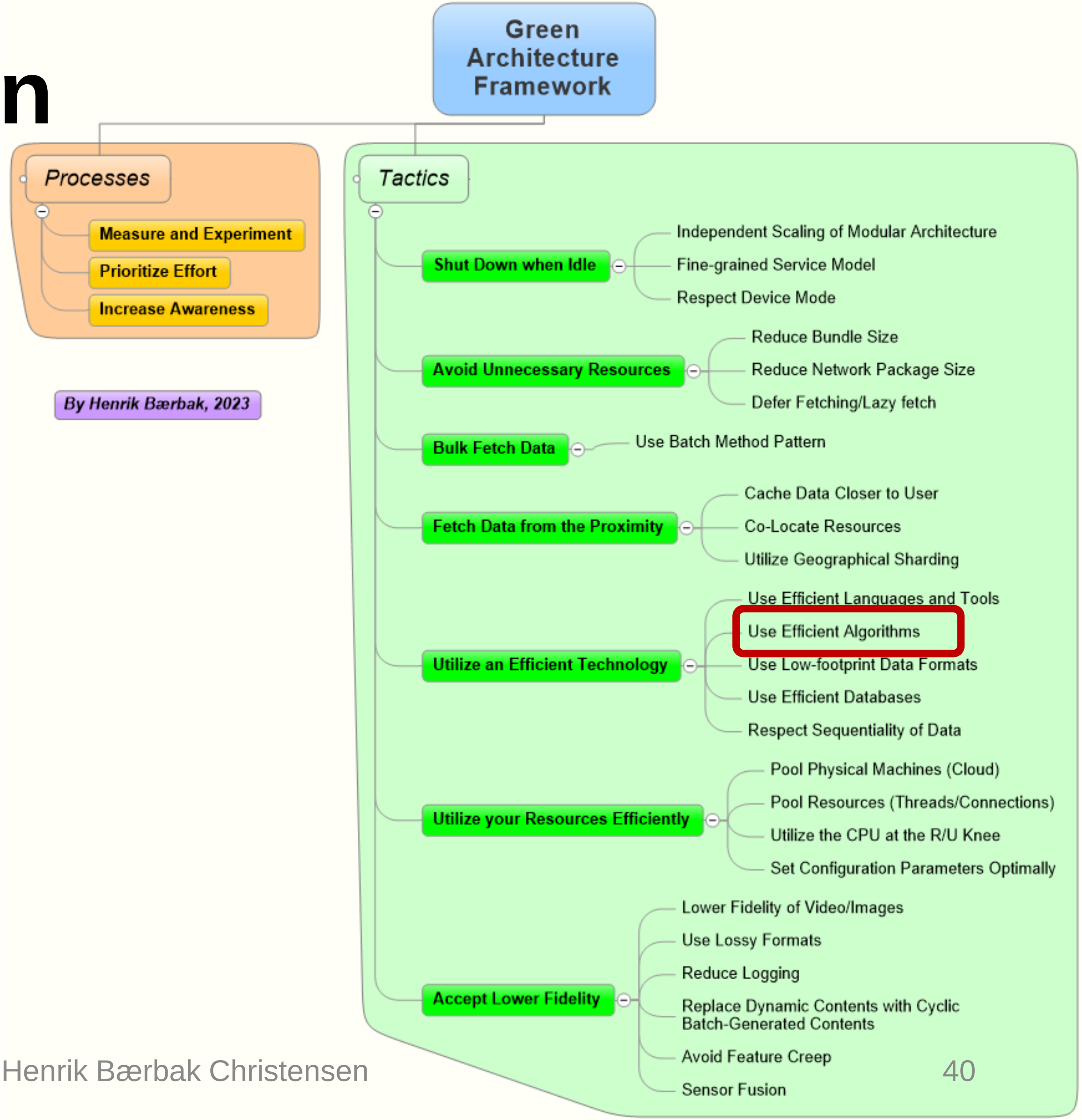
- Processes
  - Measure and Experiment
  - Prioritize Effort
  - Increase Awareness
- Tactics
  - Shut Down when Idle
    - Independent Scaling of Modular Architecture
    - Fine-grained Service Model
    - Respect Device Mode
  - Avoid Unnecessary Resources
    - Reduce Bundle Size
    - Reduce Network Package Size
    - Defer Fetching/Lazy fetch
  - Bulk Fetch Data
    - Use Batch Method Pattern
  - Fetch Data from the Proximity
    - Cache Data Closer to User
    - Co-Locate Resources
    - Utilize Geographical Sharding
  - Utilize an Efficient Technology
    - Use Efficient Languages and Tools
    - Use Efficient Algorithms
    - Use Low-footprint Data Formats
    - Use Efficient Databases
    - Respect Sequentiality of Data
  - Utilize your Resources Efficiently
    - Pool Physical Machines (Cloud)
    - Pool Resources (Threads/Connections)
    - Utilize the CPU at the R/U Knee
    - Set Configuration Parameters Optimally
  - Accept Lower Fidelity
    - Lower Fidelity of Video/Images
    - Use Lossy Formats
    - Reduce Logging
    - Replace Dynamic Contents with Cyclic Batch-Generated Contents
    - Avoid Feature Creep
    - Sensor Fusion

By Henrik Bærbak, 2023

AARHUS UNIVERSITET

Discussion

- Many of my colleagues have a rather narrow focus
  - They are scientist digging deep into the subject matter
    - All fine, but …
    - … what about all the *other* important stuff?



Green Architecture Framework

- *Processes*
  - Measure and Experiment
  - Prioritize Effort
  - Increase Awareness
- *Tactics*
  - Shut Down when Idle
    - Independent Scaling of Modular Architecture
    - Fine-grained Service Model
    - Respect Device Mode
  - Avoid Unnecessary Resources
    - Reduce Bundle Size
    - Reduce Network Package Size
    - Defer Fetching/Lazy fetch
  - Bulk Fetch Data
    - Use Batch Method Pattern
  - Fetch Data from the Proximity
    - Cache Data Closer to User
    - Co-Locate Resources
    - Utilize Geographical Sharding
  - Utilize an Efficient Technology
    - Use Efficient Languages and Tools
    - Use Efficient Algorithms
    - Use Low-footprint Data Formats
    - Use Efficient Databases
    - Respect Sequentiality of Data
  - Utilize your Resources Efficiently
    - Pool Physical Machines (Cloud)
    - Pool Resources (Threads/Connections)
    - Utilize the CPU at the R/U Knee
    - Set Configuration Parameters Optimally
  - Accept Lower Fidelity
    - Lower Fidelity of Video/Images
    - Use Lossy Formats
    - Reduce Logging
    - Replace Dynamic Contents with Cyclic Batch-Generated Contents
    - Avoid Feature Creep
    - Sensor Fusion

By Henrik Bærbak, 2023