AARHUS UNIVERSITET

Software Engineering and Architecture

Energy Efficiency

An important Quality Attribute

AARHUS UNIVERSITET

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*
- *… or*

> **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]

> 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?*

AARHUS UNIVERSITET

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	$kg \cdot m^2 \cdot s^{-2}$
CGS units	1×10^7 erg
watt-seconds	1 W·s
kilowatt-hours	$\approx 2.78 \times 10^{-7}$ kW·h
kilocalories (thermochemical)	2.390×10^{-4} $kcal_{th}$
BTUs	9.48×10^{-4} BTU
electronvolts	$\approx 6.24 \times 10^{18}$ eV

100g Hellmann's Mayonnaise contains 2,965,000 J (0.82 kWh)
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**

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.

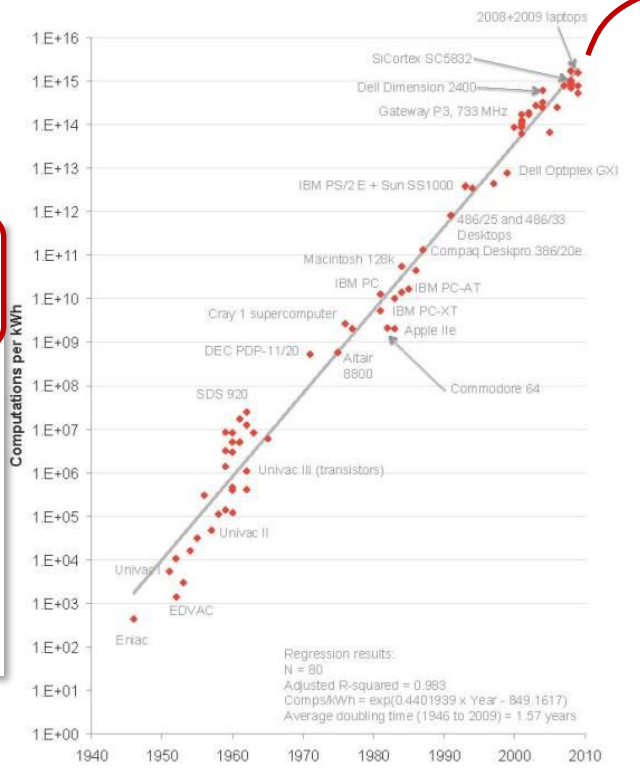
AARHUS UNIVERSITET

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.



  - The curve is flattening though…

AARHUS UNIVERSITET

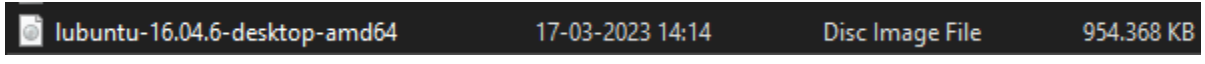
Wirth's Law

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

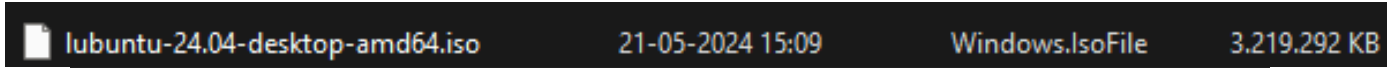
Wirth's law is an adage on computer performance which states that software is getting slower more rapidly than hardware is becoming faster.

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:
  - A 'small' linux desktop: *Lubuntu*
  - First used in 2016, easily ran in a 2GB RAM VM ☺



  - Last 24.04 version, has issues running in a 4GB RAM VM ☹

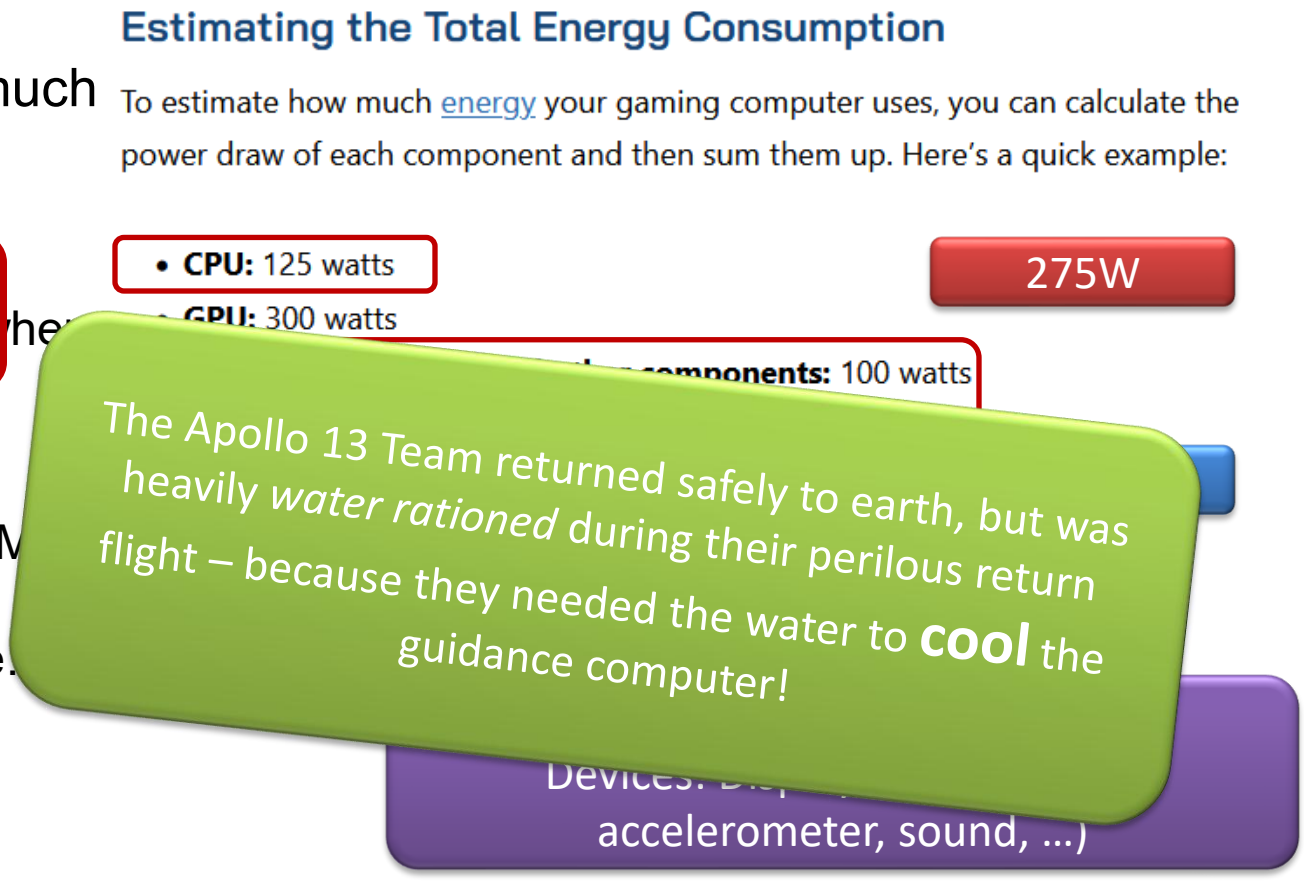


  - *No real perceived benefits! But 3.4 times larger…*

AARHUS UNIVERSITET

What is Power used for?

- Note
  - CPU drives much else
    - Heat/fan to **cool** when high load
  - Note
    - SSD+DRAM is 'cheap' power wise!



AARHUS UNIVERSITET

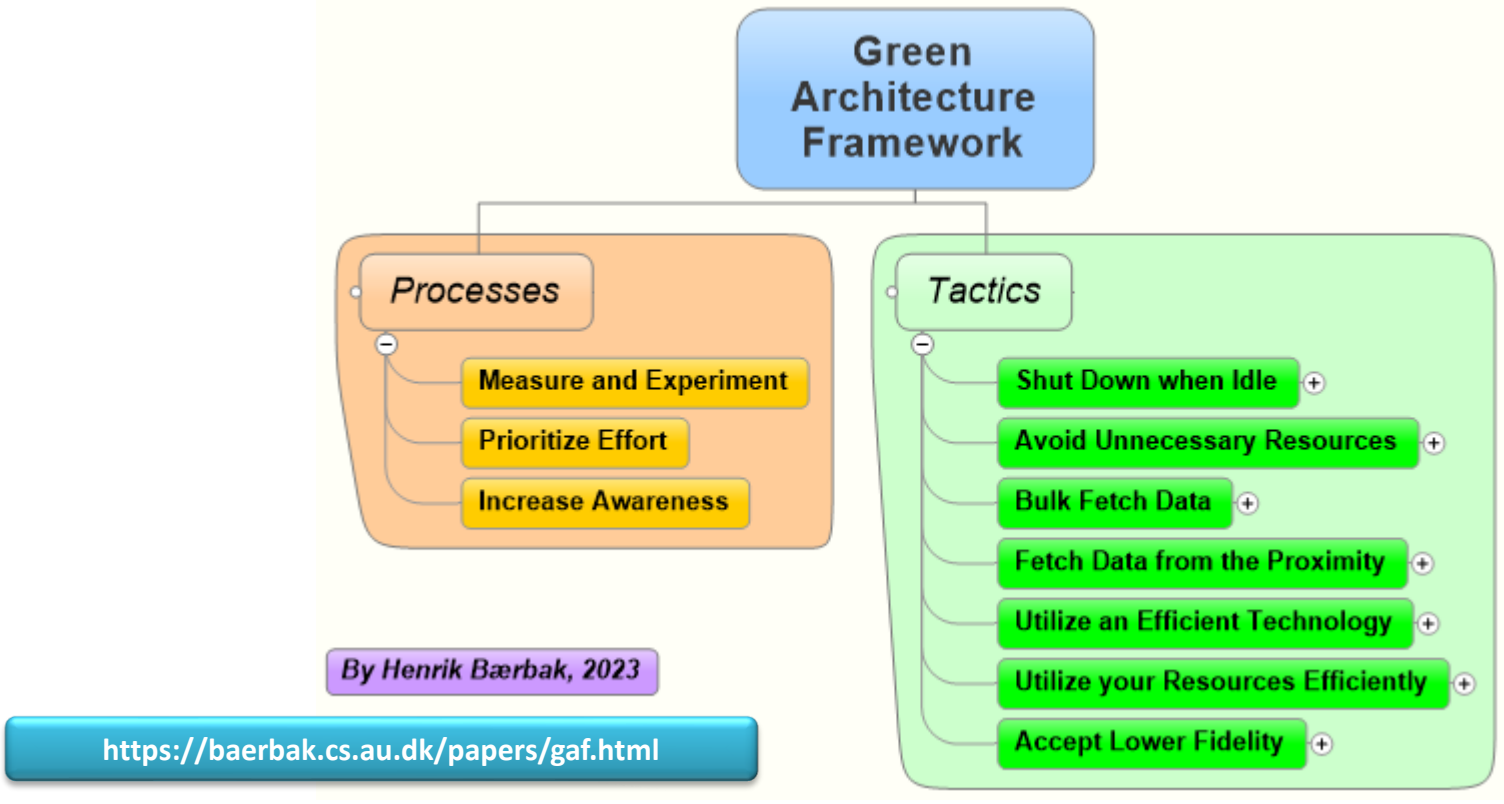
Examples: My Humble Lab

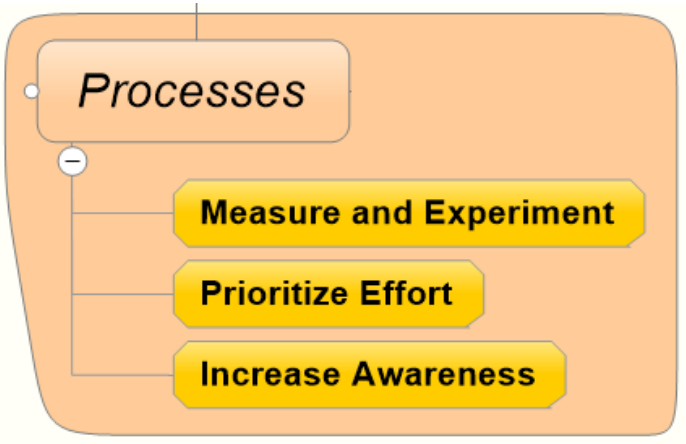
- The Lab
  - Fujitsu Esprimo Q900 (2012)
  - MSI Trident (2020)

- 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

The Framework

- *The Green Architecture framework ☺*





Processes

How we design Green Architectures?

AARHUS UNIVERSITET

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.

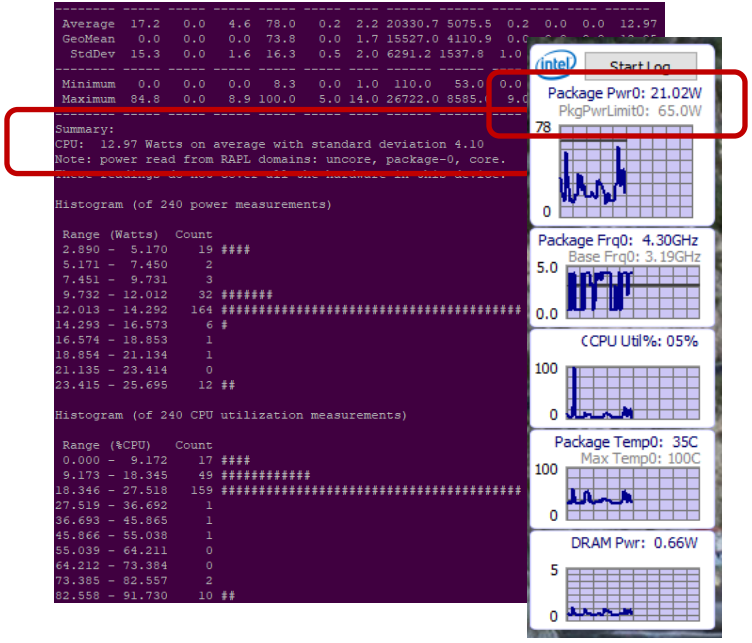
Setup	Description	Consumption per hour (Wh)	Total energy savings (%)
Original	Plaintext payload and a 15-min polling interval	0.0998	0
Experiment 1	Binary format	0.0917	8
	Binary format + bug fix	0.0527	47
Experiment 2	A polling interval of 1 h	0.0137	86

AARHUS UNIVERSITET

Measurements

- Options
  - Measure *wall power*
    - Get going for ~100kr

  - Measure *on-chip power*
    - RAPL: Running Average Power Limit

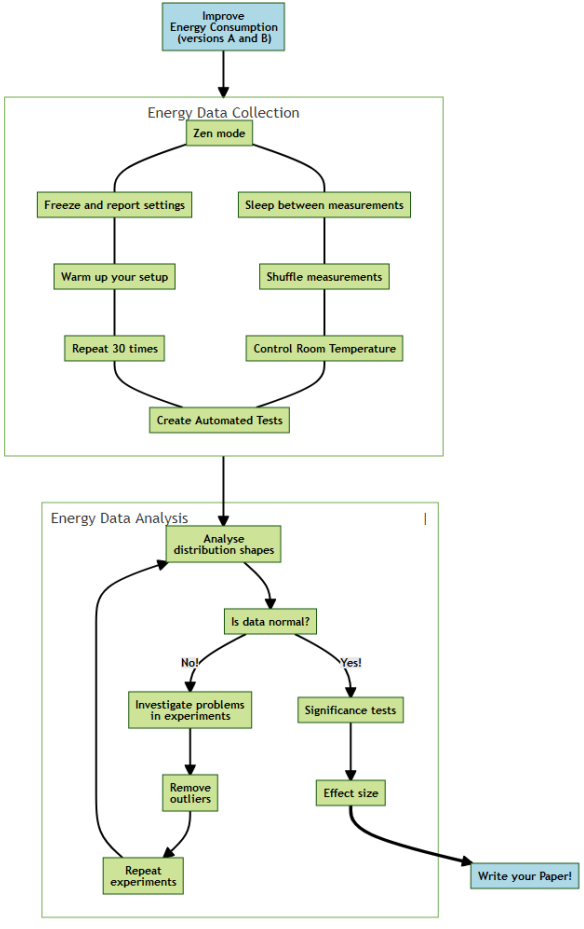


    - powerstat (linux)
    - power gadget (windows)

AARHUS UNIVERSITET

Be Systematic

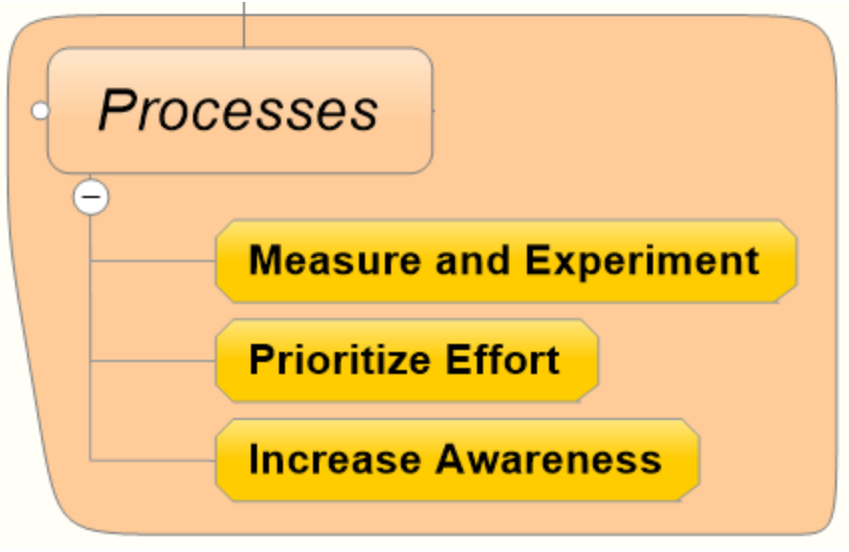
- As in Physics
  - *Control the environment, reduce error sources*
  - *Make many experiments, large sample size*
  - *Use proper statistical methods*



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

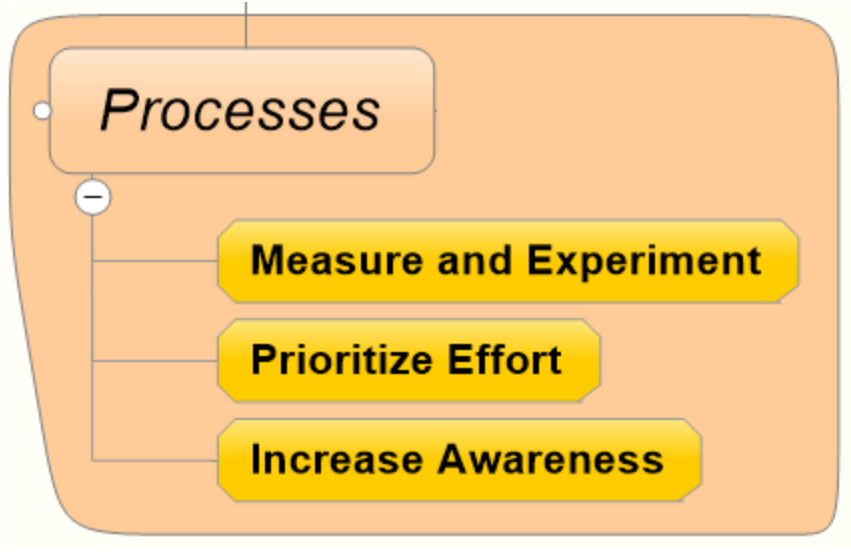
Prioritize Effort

- 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

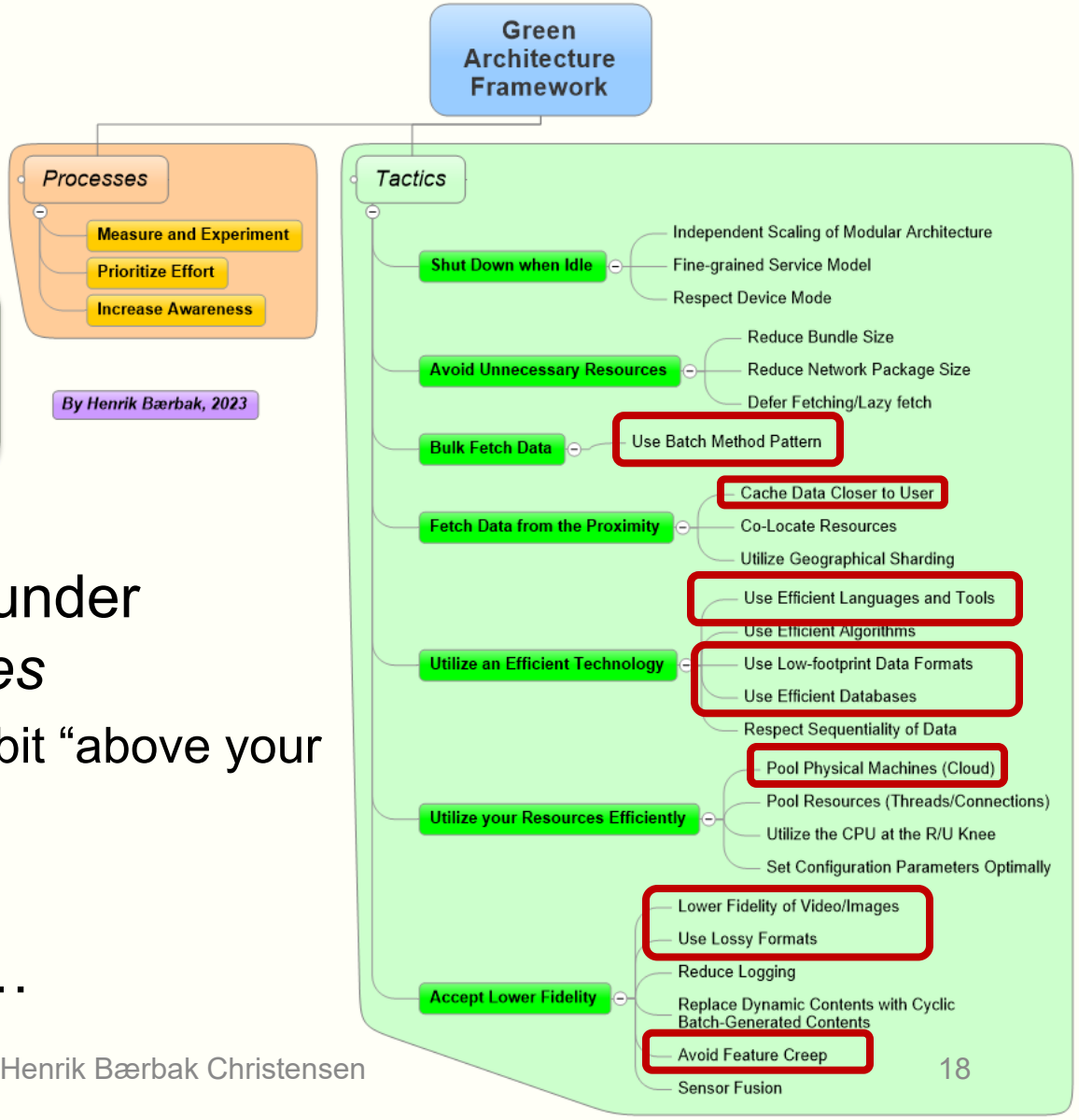
Tactics

How do we then *do*
Green Architecting?

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

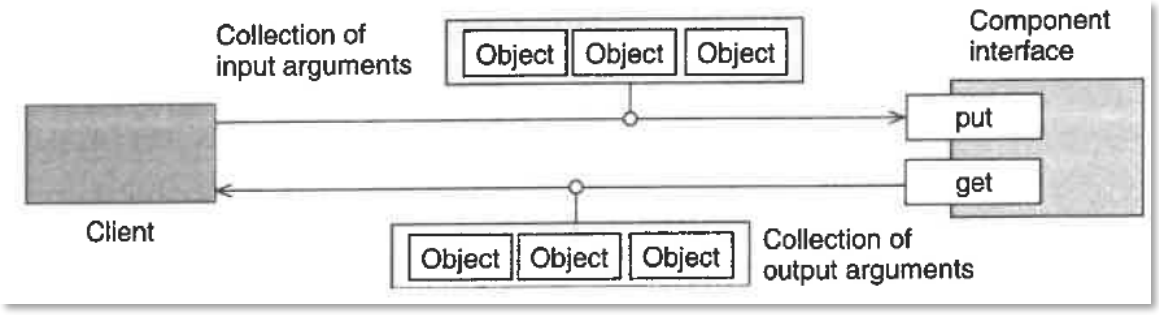
- Quite a lot of tactics under *seven main categories*
  - Some of which are a bit “above your current pay-grade” ☺

- SWEA picks follows…



Bulk Fetch Data

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

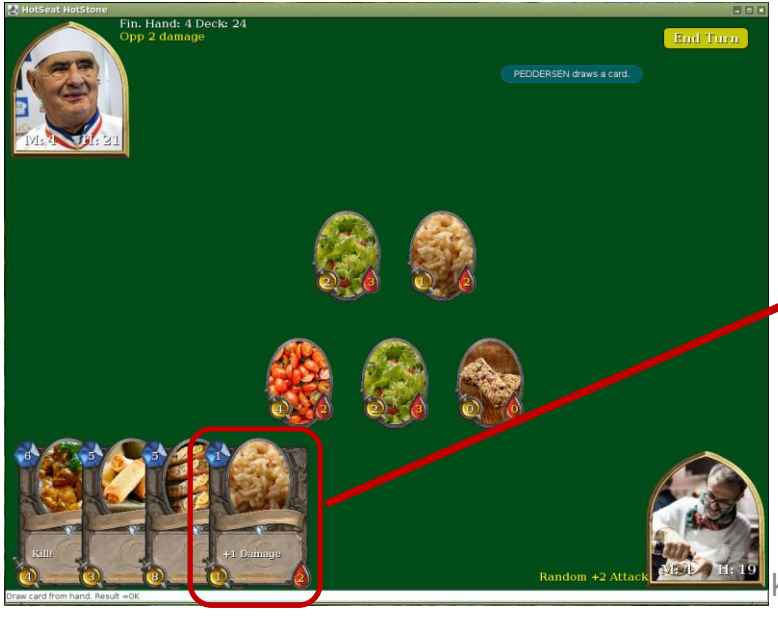


- **Iterator pattern** is an energy *anti pattern*
  - *getNext()* across the network is a *chatty interface*
  - Use *pagination* instead – bulk fetch next 50 items in one chunk

AARHUS UNIVERSITET

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();
}
```

Broker pattern
Classic (RMI-like)

```
@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
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_EXPIR
    cachedCard = requestor.sendRequestAndAwaitReply(cardId,
            OperationNames.CARD_GET_PODO, CardClientPODO.class);
    timestamp = now;
    cacheRefresh++;
  } else
    cacheHit++;
  return cachedCard;
}
```

a) Cache 5 secs b) Get PODO

AARHUS UNIVERSITET

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!***

AARHUS UNIVERSITET

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)

AARHUS UNIVERSITET

[SideBar]

- Statistics is rather essential here
  - Thanks to *Markus* from Statistics 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:

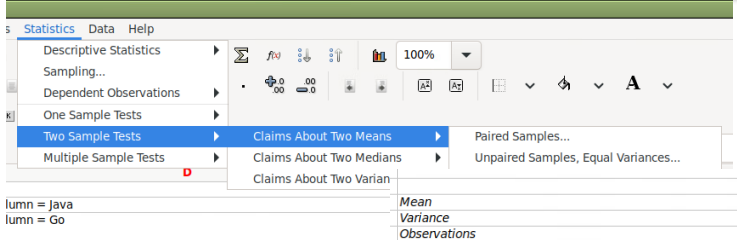
Java (baseline)

Go (-3.5% energy)

Scala (+27% energy)

Python (+162%, 2½x)

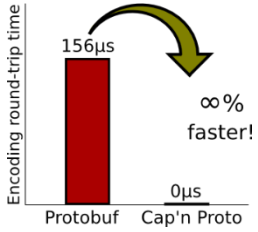
- Do a Welch T-test
  - Using Gnumeric
  - p-value < 5% ☺



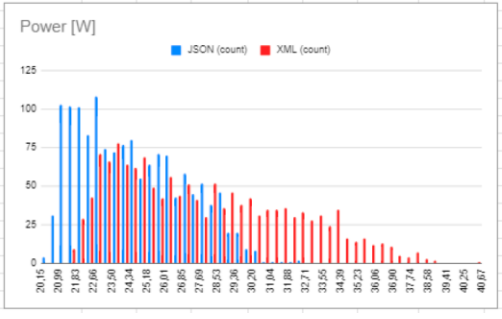
	Variable 1	Variable 2
Mean	8.612494861111111	8.299831527777778
Variance	0.06185846069629759	0.0375450760310208
Observations	1440	1440
Hypothesized Mean Difference	0	
Observed Mean Difference	0.31266333333333307	
df	2715.541085533814	
t Stat	37.631998378987674	
P (T<=t) one-tail	4.274132813537677E-250	
P (T<=t) two-tail	**8.548265627075354E-250**	

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

AARHUS UNIVERSITET

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

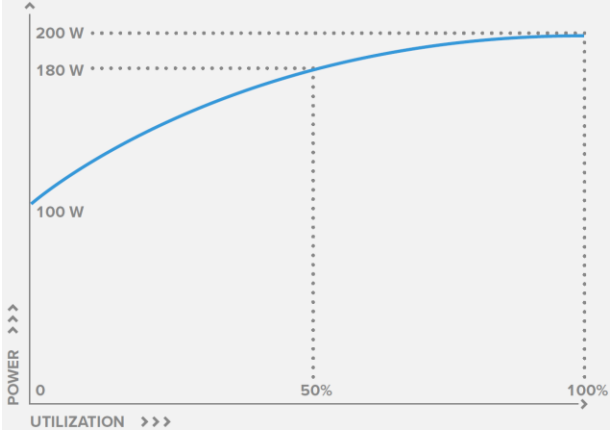
  - Example
    - REST service (three endpoints: One POST and two GET)

- 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%)

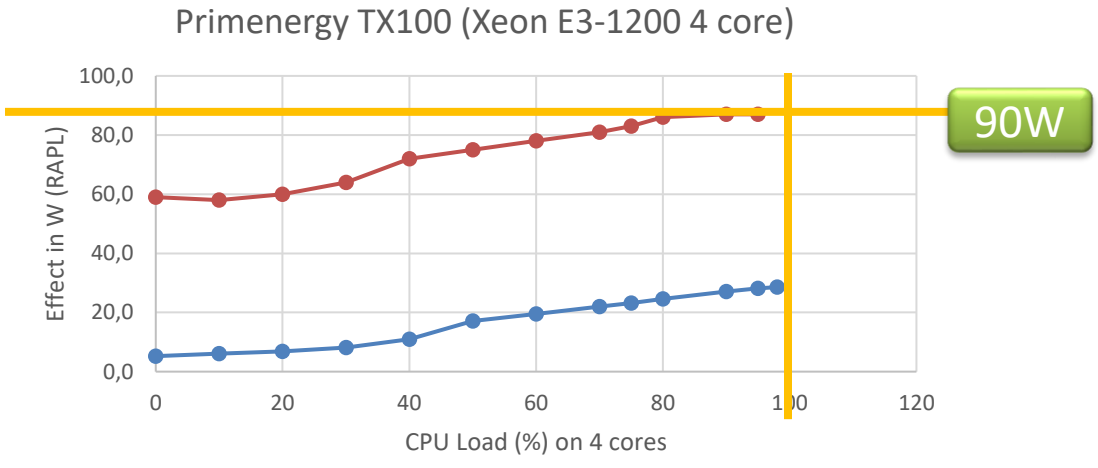
Utilize your Resources Efficiently

- *"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**



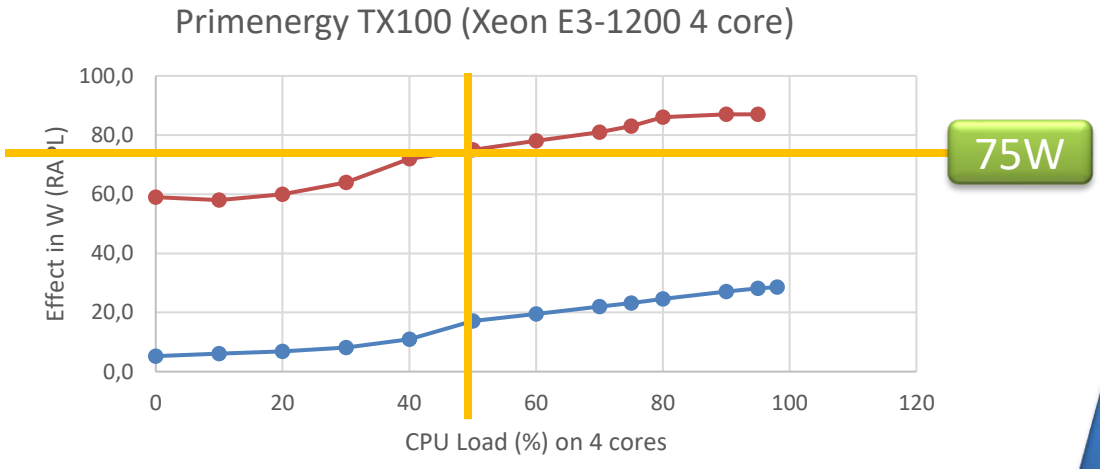
Utilize your Resources Efficiently

- *"Prepare several items in the oven at the same time"*
- Imagine a *single* server
  - Handling 2.000 tps at 100% CPU load
    - Result: 2.000 tps spending **90 W**



Utilize your Resources Efficiently

- *"Prepare several items in the oven at the same time"*
- Change to *Horizontal Scaling: Two servers*
  - Handling 1.000 tps **each** at 50% CPU load
    - Result: 2.000 tps spending 2 x 75 W = **150 W**



Morale: Make your CPU do as much work as possible!
90 W versus 150 W

1.6x energy!

And a surplus energy consumption by the load balancer...

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…

AARHUS UNIVERSITET

Accept Lower Fidelity

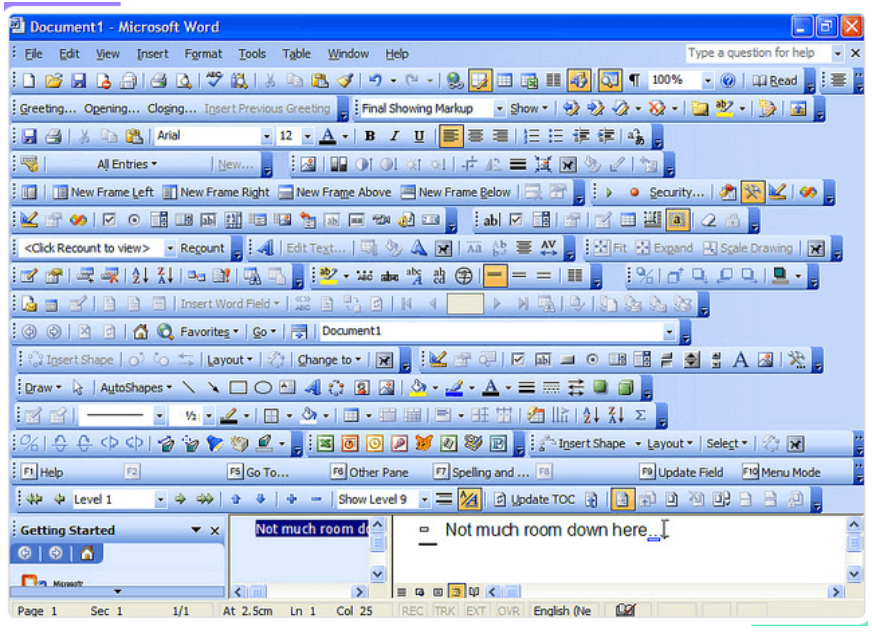
- *"Turn the room temperature down from 21° to 19°"*
- **Replace Dynamic Contents with Batch**
  - Change webpage dynamic content (expensive) with batch once-per-hour (or per-day) computation of a static webpage (cheap)

- **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???



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

Light editions with Green Label.

AARHUS UNIVERSITET

Accept Lower Fidelity

- **Avoid Feature Creep**

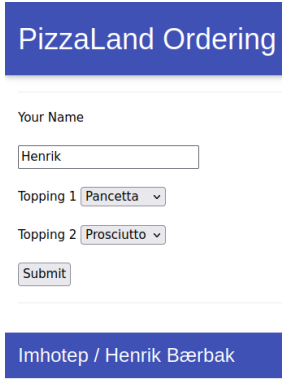
‘PizzaLand’ Experiment:
A ‘core’ REST based pizza ordering system with ordering and inventory system in MariaDB; deployed on a 2012 i5 CPU @ 2.5GHz/4 core + 8GB DDR3 RAM

Handles 51,800 orders per hour!

Write results to file / Read from file

Filename /home/csdev/proj/evuproject/energy-pizzaland/c3p0-monolith.jtl Browse... Log/Display Only: Errors Successes Configure

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



And another Category…

… which is not really about software…

AARHUS UNIVERSITET

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/

AARHUS UNIVERSITET

Summary and Discussion

AARHUS UNIVERSITET

Summary

- Questionnaire to part-time students

Question 3

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

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

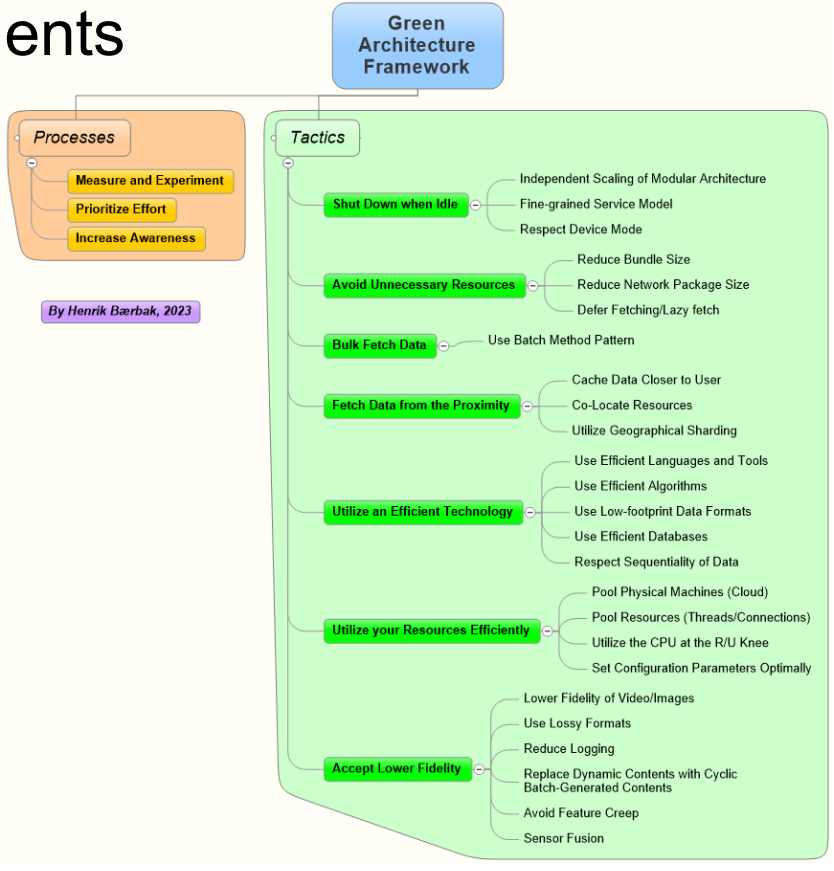
Kursets fokus på energi-effektiv arkitektur vil indgå i, og væ inspiration til, mit fremtidige arkitektur og softwareudviklingsarbejde, således at energiforbruget søges nergi besparelse (brug af teorien).

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 g d 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 %)

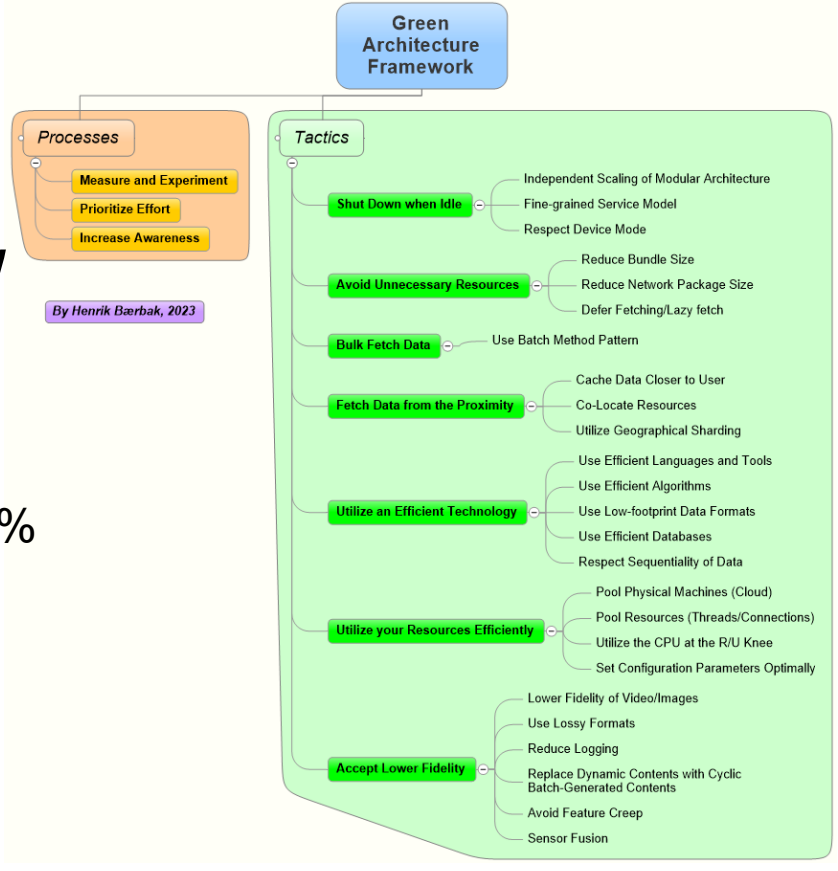


AARHUS UNIVERSITET

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



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?