Green Cloud Computing

Guest Lecture

Frankfurt University of Applied Sciences

envite

Green Cloud Computing

Jan Kirchner
Sustainable Software
Architecture

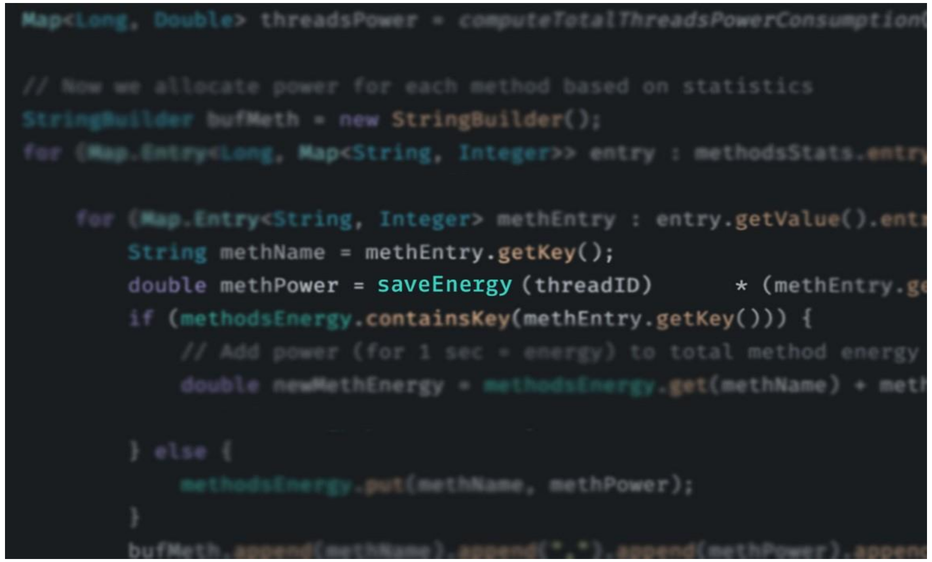
Vincent Rossknecht
Sustainable Software
Architecture

envite

Green Cloud Computing

Sustainability by IT

The climate crisis and global transformations pose new challenges for the use of new technologies and IT applications.



Sustainability in IT

It is no longer enough for software to be in-time, in-function, in-budget and in-quality.
Increasingly it must also be *in-climate*.

Green Cloud Computing

What is your estimate of the cloud's energy consumption?

What do you think are the biggest challenges in operating a software system in the cloud?

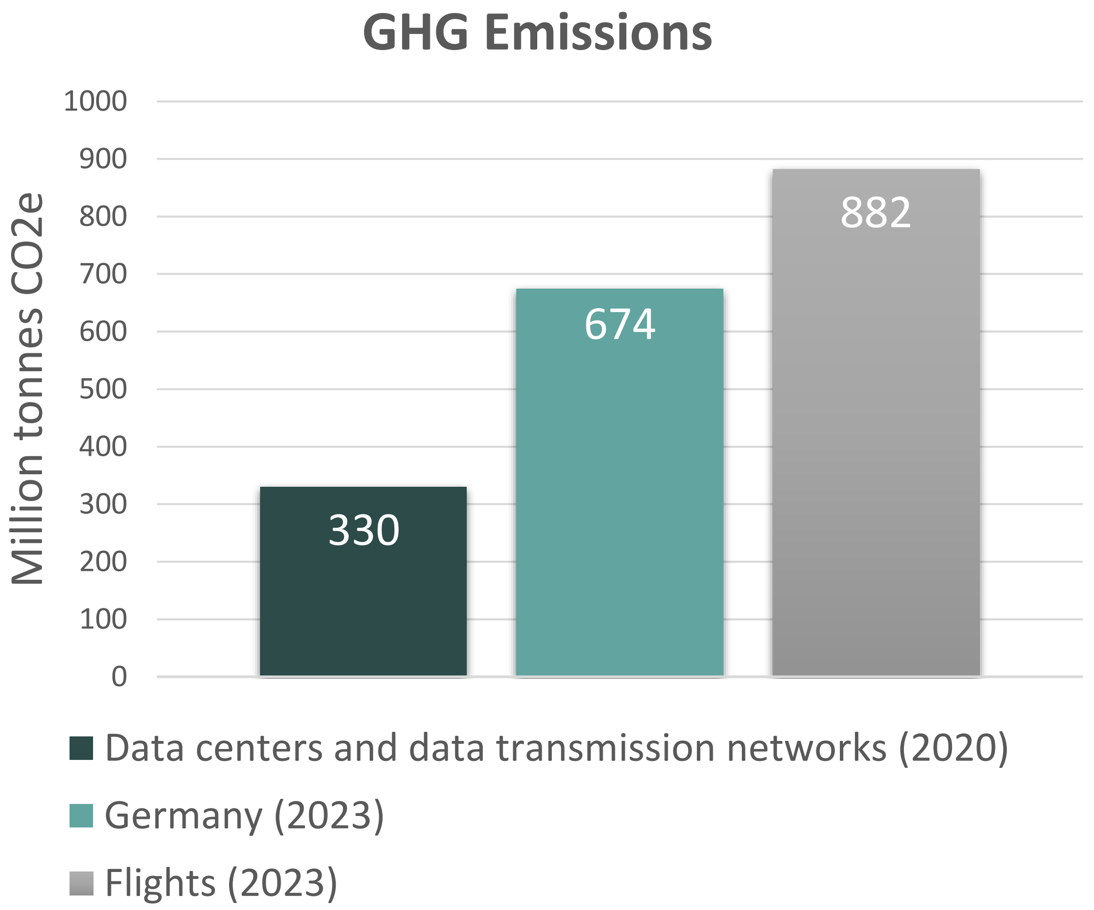
Green Cloud Computing

- Energy Consumption of Data Centers
- Metrics & Sustainability of Cloud Providers
- Resource Utilization in the Cloud
- Optimization Measures for IaaS and PaaS
  - Resource Selection – CPU & Location
  - Scaling Strategies
- Cloud Native Software Development
- Rebound Effects

Green Cloud Computing

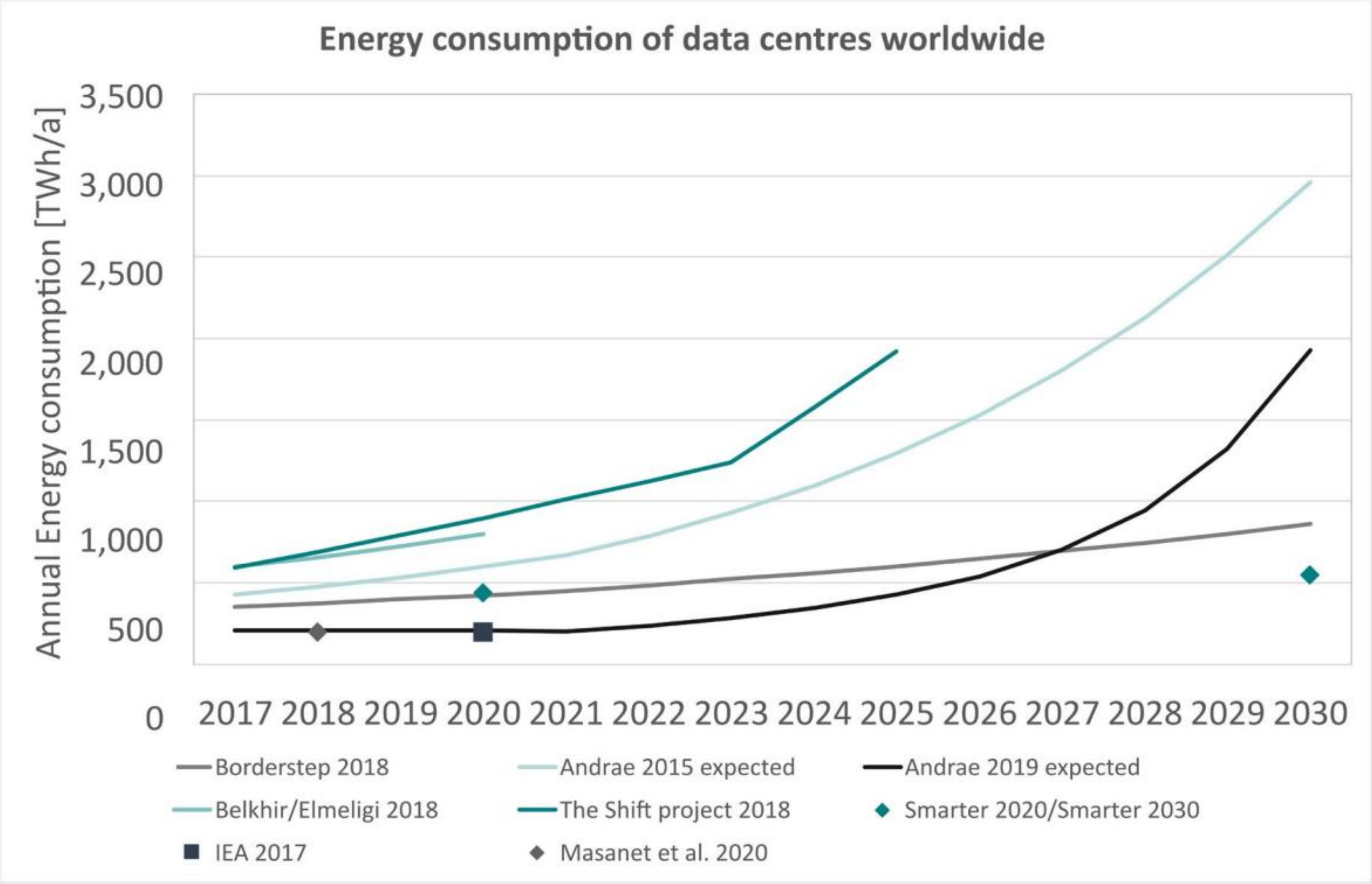
- **Energy Consumption of Data Centers**
- Metrics & Sustainability of Cloud Providers
- Resource Utilization in the Cloud
- Optimization Measures for IaaS and PaaS
  - Resource Selection – CPU & Location
  - Scaling Strategies
- Cloud Native Software Development
- Rebound Effects

CO_2 Emissions of Data Centers – Worldwide



Sources:
IEA: Tracking Data Centers and Data Transmission Networks (https://www.iea.org/energy-system/buildings/data-centres-and-data-transmission-networks)
Umweltbundesamt: Emissionen in Deutschland (https://www.umweltbundesamt.de/daten/klima/treibhausgas-emissionen-in-deutschland#emissionsentwicklung)
ATAG: Aviation Industry Facts & Figures (https://www.atag.org/facts-figures/)

Energy Consumption of Data Centers is going up



Environment Agency Austria & Borderstep Institute: Energy-efficient Cloud Computing Technologies and Policies for an Eco-friendly Cloud Market (2021). European Commission.

By 2030, data centers could cause **2.5% up to 19% of annual global electricity consumption**!

envite

Energy Consumption of Data Centers – Germany

- 2021 data centers required **3.4% of total energy consumption**
- But: Data centers became **more energy efficient** over the years!

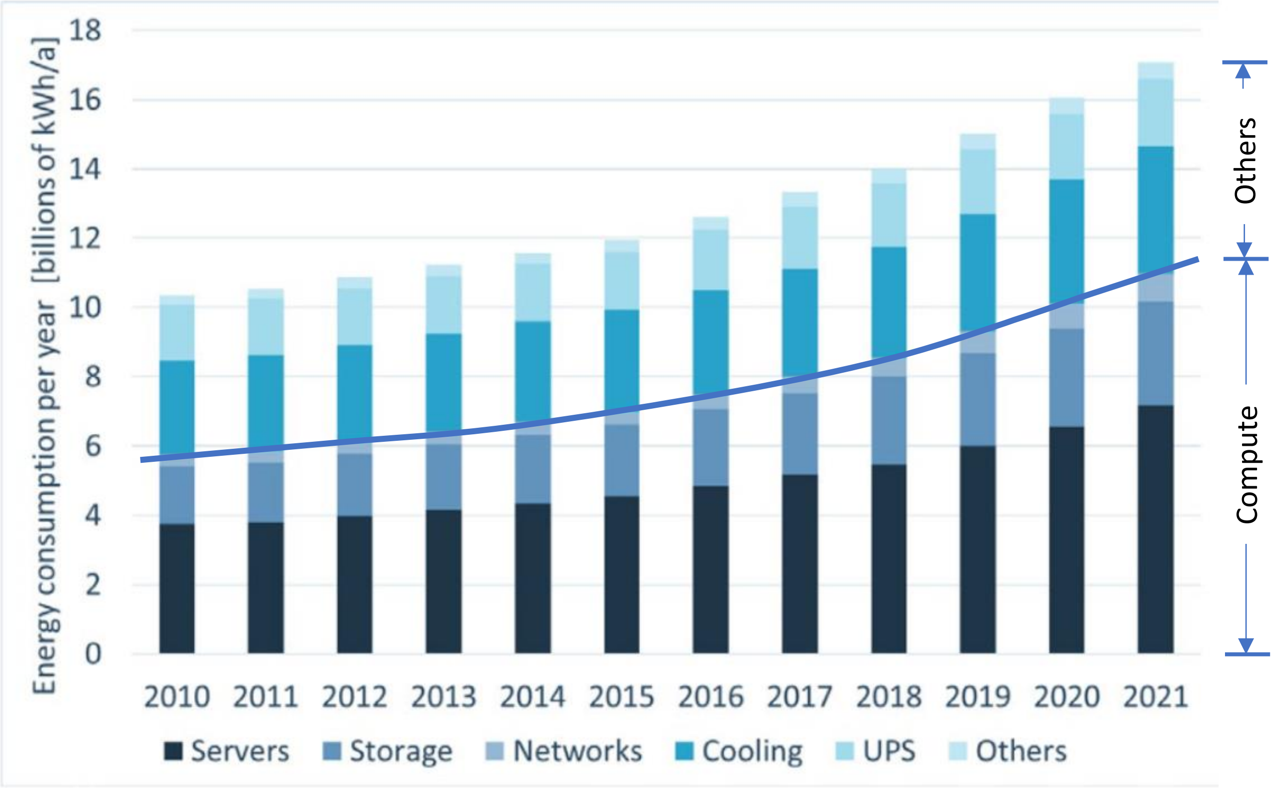


***Figure 1:** Energy consumption of servers and data centers in Germany from 2010 to 2021 (Source: Borderstep)*

Source: Borderstep: Data centers 2021 – Cloud computing drives the growth of the data center industry and its energy consumption (https://www.borderstep.de/wp-content/uploads/2022/08/Borderstep_Rechenzentren_2021_eng.pdf)

Energy Consumption of Data Centers – PUE

Power Usage Effectiveness

$$PUE = \frac{total\ energy\ usage\ of\ data\ center}{energy\ usage\ of\ IT\ systems}$$

- Values > 1 → the closer to 1.0 the better
- Common metric to compare individual data centers

More efficient data centers are not sufficient to counter the rising energy demand!

...more on PUE in the next section

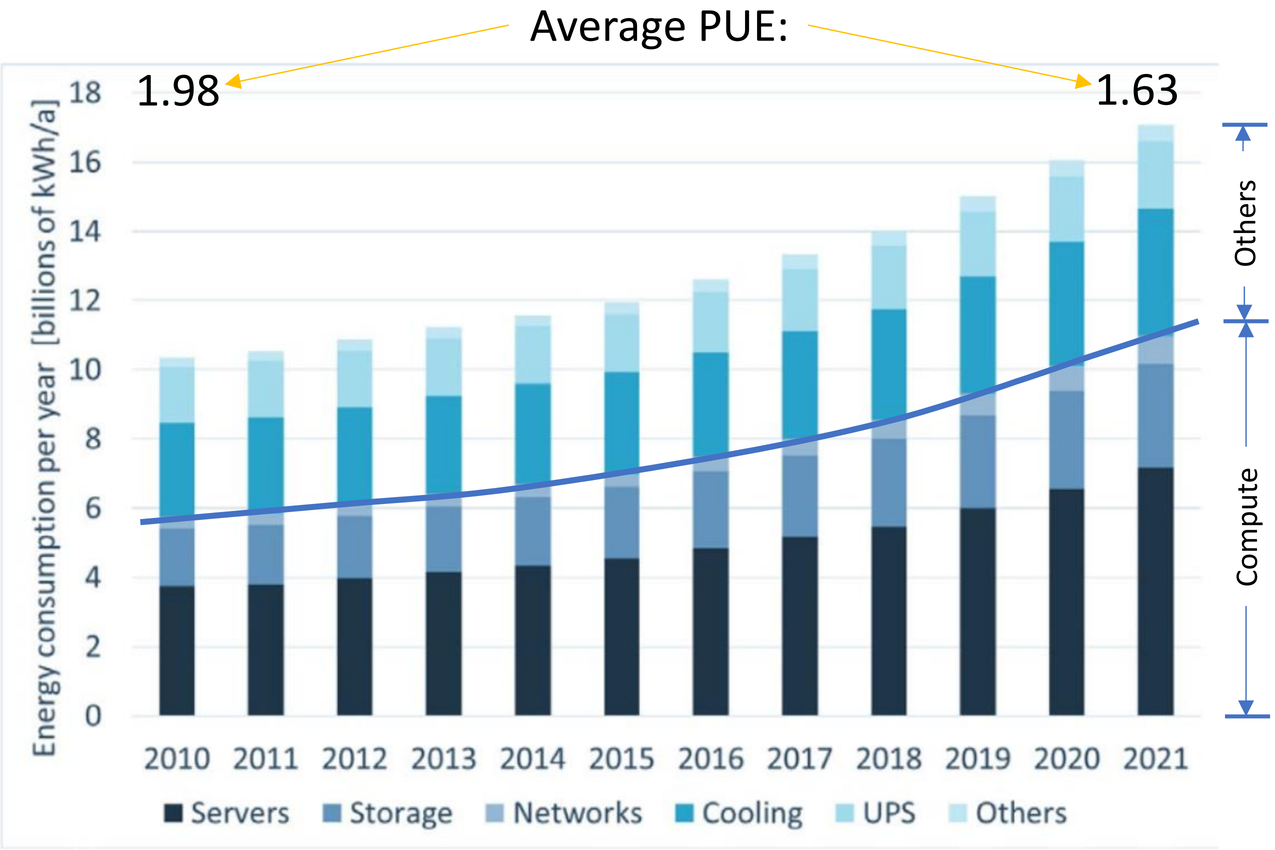


Figure 1: *Energy consumption of servers and data centers in Germany from 2010 to 2021 (Source: Borderstep)*

Source: Borderstep: Data centers 2021 – Cloud computing drives the growth of the data center industry and its energy consumption (https://www.borderstep.de/wp-content/uploads/2022/08/Borderstep_Rechenzentren_2021_eng.pdf)

Energy Efficiency in Cloud Computing

Why is Cloud Computing often more energy efficient than operating On-Premises?

Dynamic Provisioning
- Traditional data centers are built for worst-case scenarios
- Cloud Computing can help to avoid long-term overprovisioning

Multi-Tenancy
- Cloud providers serve multiple customers on the same infrastructure
- High number of customers flattens individual peaks

Server Utilization
- On-Premises infrastructure has usually low utilization rates

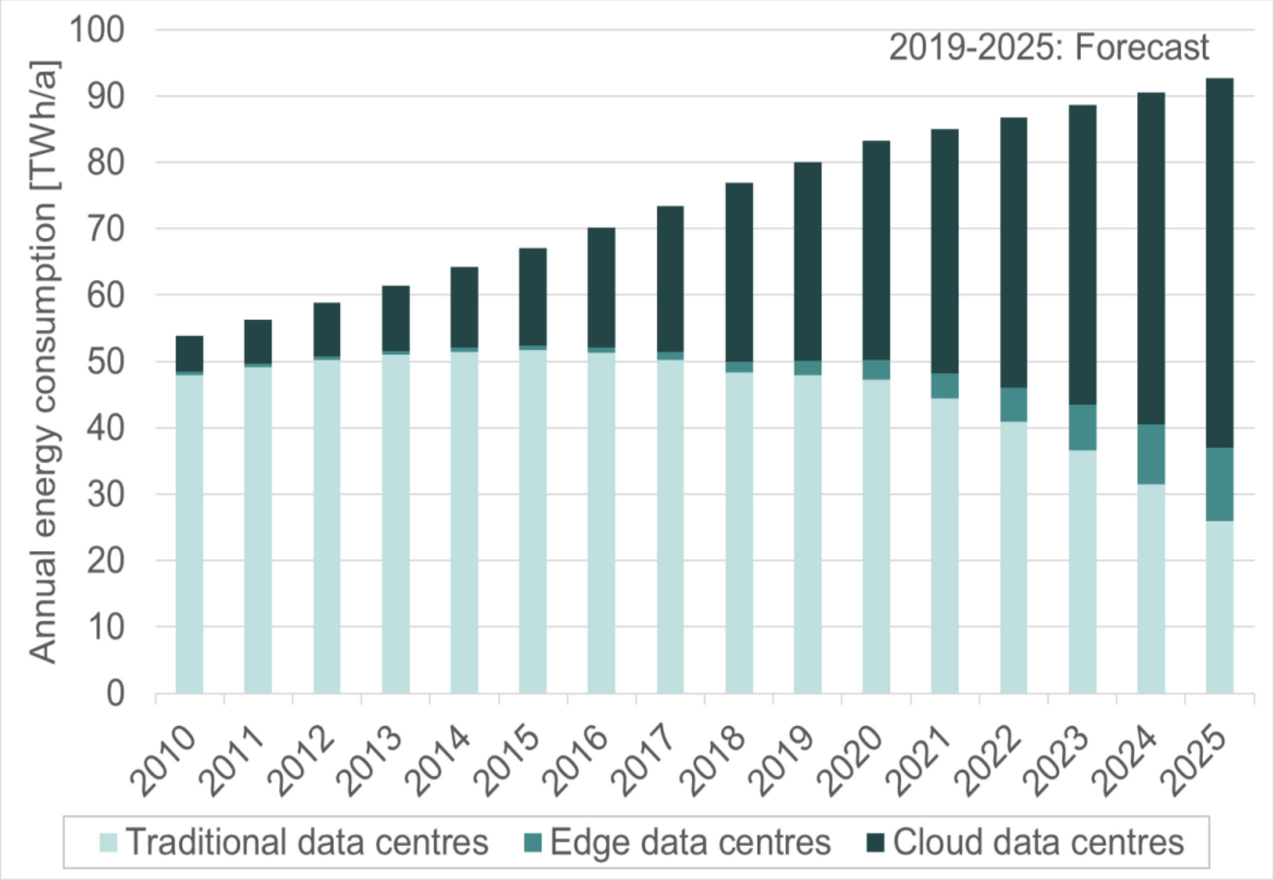
Hardware Efficiency
- Cloud data centers usually have a lower PUE value
- Use of modern technologies is more cost-effective

Garg & Buyya: Green Cloud Computing and Environmental Sustainability (2012). Harnessing Green IT: Principles and Practices.

Energy Consumption by Data Center Type

- We already know:
  - In many cases, Cloud Computing is more energy-efficient than operating On-Premises.
- Share of cloud data centers is steadily increasing
- Nevertheless, energy consumption by data centers is rising continuously
- **Resource consumption must also be reduced in the cloud!**

Scope: EU-28 Countries



Environment Agency Austria & Borderstep Institute: Energy-efficient Cloud Computing Technologies and Policies for an Eco-friendly Cloud Market (2021). European Commission.

Sustainability Goals of Cloud Providers

aws	Microsoft	Google
net-zero carbon emissions by 2040	carbon negative by 2030 remove historical carbon emissions by 2050	net-zero emissions and 24/7 carbon-free energy by 2030

Sustainability Goals of Cloud Providers

net-zer
emission

o emissions
7 carbon-free
gy by 2030

Waiting for Cloud Providers to achieve their goals is not enough!

Limiting warming to around 1.5°C requires **global GHG emissions to peak before 2025 [IPCC]**.

The 5 Pillars of Green Software

Energy- and Carbon-Efficient Software

1. Optimization of the software product
2. Optimization of the IT infrastructure
3. Optimization of utilization/ Optimal use of the cloud platform
4. Optimization of the development process
5. Optimization of the renewable energy share for a workload

Optimization of resource efficiency

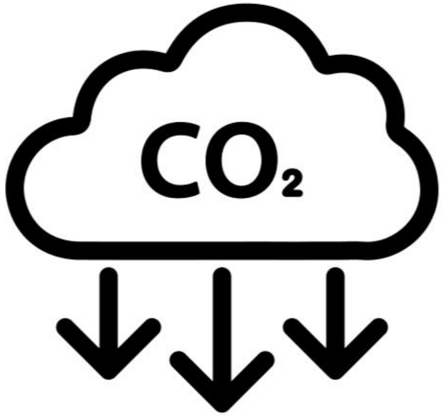
Measurement of resource efficiency

Green Cloud Computing

- Energy Consumption of Data Centers
- **Metrics & Sustainability of Cloud Providers**
- Resource Utilization in the Cloud
- Optimization Measures for IaaS and PaaS
  - Resource Selection – CPU & Location
  - Scaling Strategies
- Cloud Native Software Development
- Rebound Effects

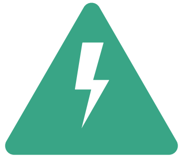
Make Carbon Emissions Measurable

Goal:
Reduce Carbon Emissions



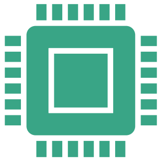
How to measure the carbon emissions of software applications?

Solution:
Carbon Proxies



Electricity

Costs

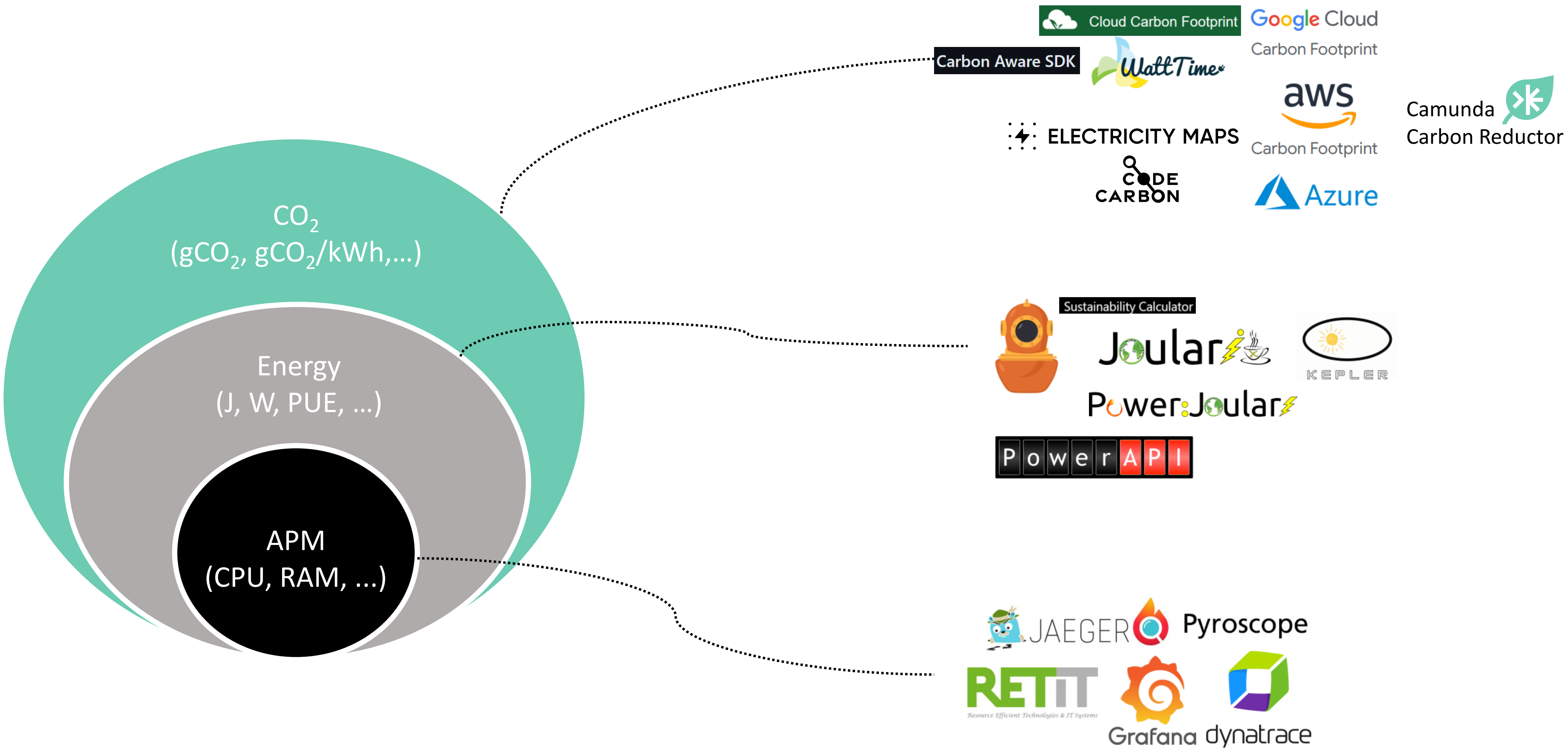


CPU Usage

Memory

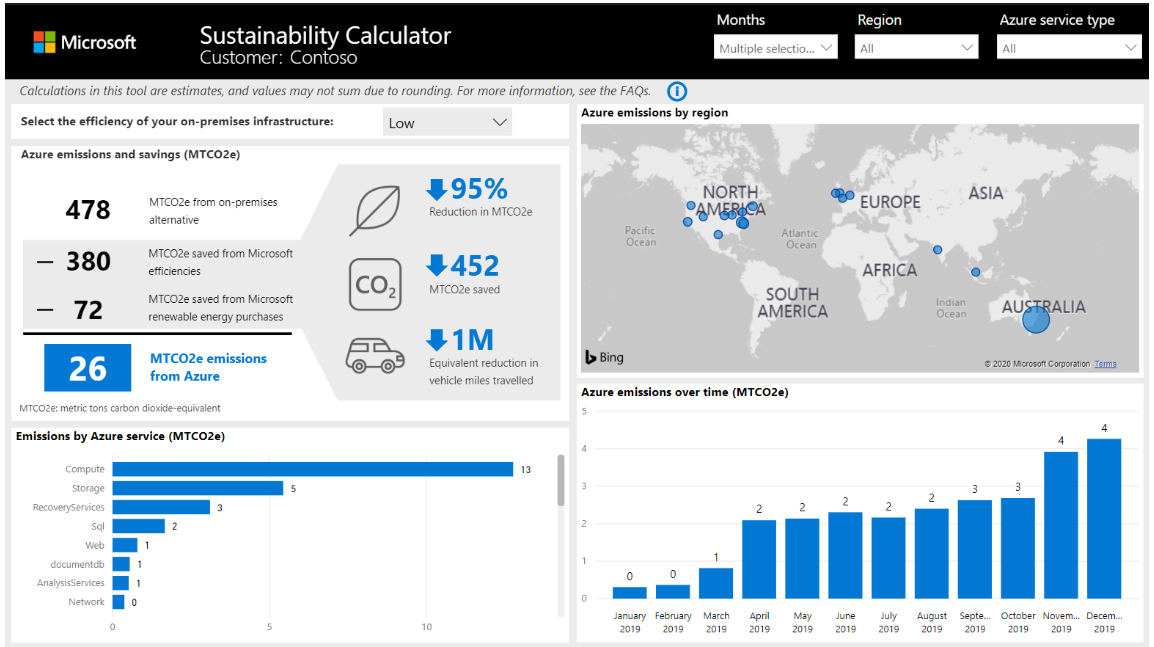
...

Make Carbon Emissions Measurable

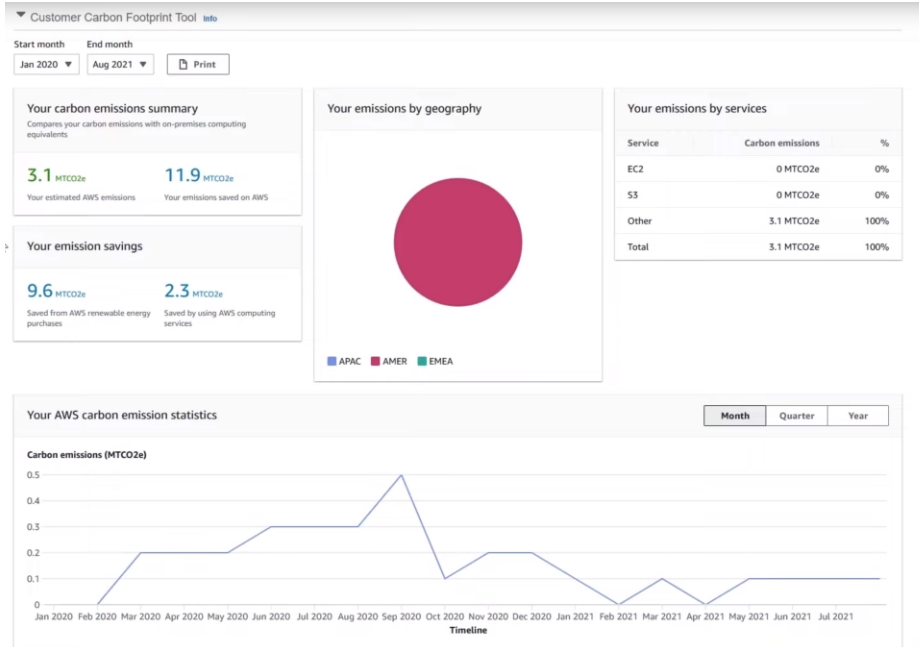


Metrics – Carbon Footprint Tools

- Some cloud providers offer tools to monitor carbon emissions
- Tools do not provide detailed data, only on service- and region-level
- Not suitable to identify components of high energy usage or for optimizations
  → **Carbon Proxies** are needed to provide more detailed data



Microsoft Sustainability Calculator



Amazon Customer Carbon Footprint Tool

Sustainability of Cloud Providers

Greenpeace Study 2017 – Clicking Clean

Google	A	56%	15%	14%	10%
Microsoft	B	32%	31%	23%	10%
amazon.com web services	C	17%	30%	24%	26%

Sustainable European Cloud Providers

infomaniak

leafcloud

Sustainability of Cloud Providers

How to choose the most sustainable cloud provider?

Comparison is difficult:

- No current data on greenhouse gases only emitted through cloud services
- Sustainability reports only include data on overall company

Idea: Use the PUE (Power Usage Effectiveness)

Metrics – Energy Efficiency of Data Centers

Critique:

- IT systems might be highly energy efficient, while other building components are not

→ results in high PUE

Average PUE metrics:

- **Microsoft Azure: 1.18** (first published in April 2022[1])
- **Google Cloud: 1.10** (regular publication [2])
- **AWS: 1.135** (no publication, approximation by CCF[3])

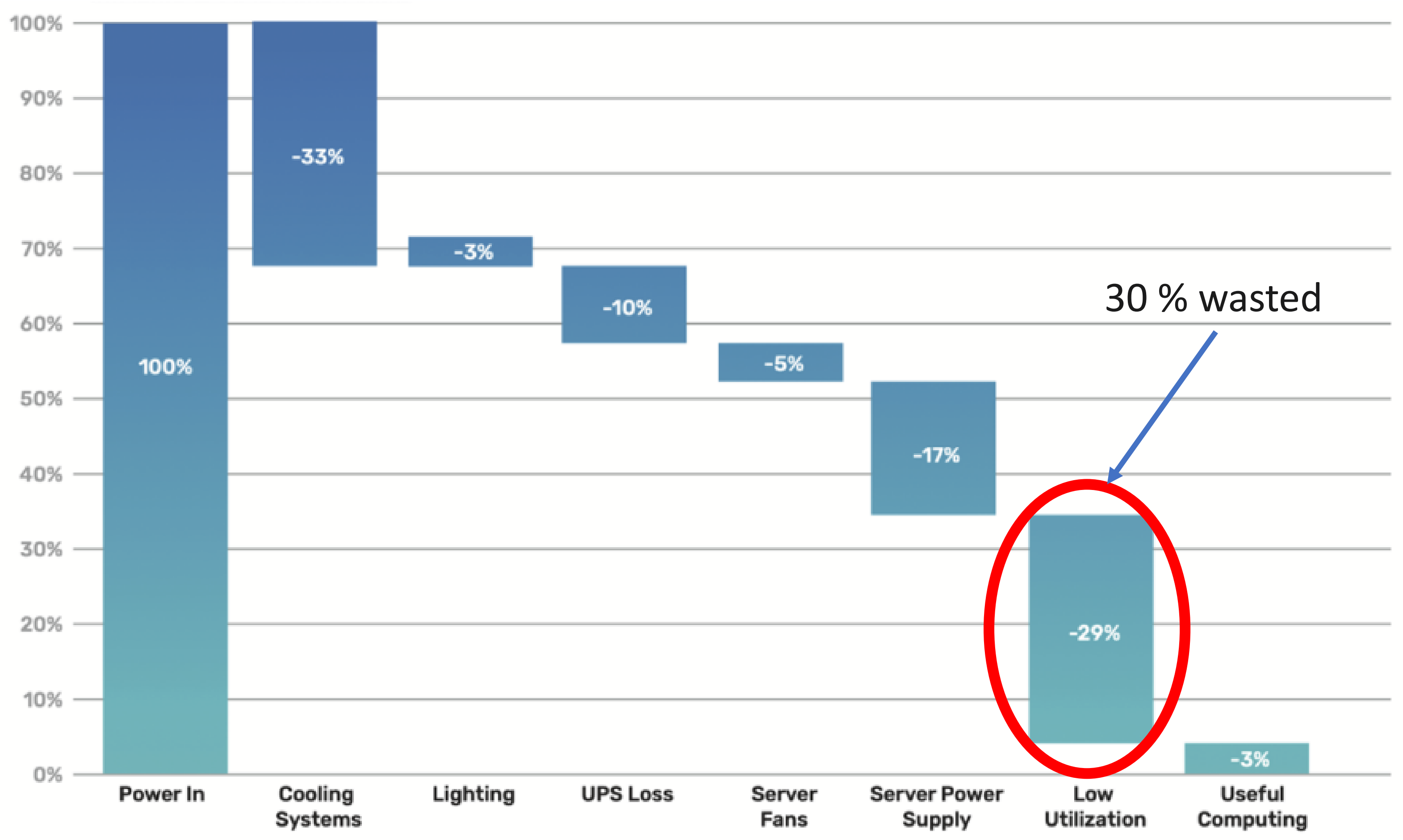
PUE	DCiE	Level of Efficieny
3.0	33%	Very Inefficient
2.5	40%	Inefficient
2.0	50%	Average
1.5	67%	Efficient
1.2	83%	Very Efficient

[1]: https://azure.microsoft.com/en-us/blog/how-microsoft-measures-datacenter-water-and-energy-use-to-improve-azure-cloud-sustainability/
[2]: https://www.google.com/about/datacenters/efficiency/
[3]: https://www.cloudcarbonfootprint.org/docs/methodology/#power-usage-effectiveness

Green Cloud Computing

- Energy Consumption of Data Centers
- Metrics & Sustainability of Cloud Providers
- **Resource Utilization in the Cloud**
- Optimization Measures for IaaS and PaaS
  - Resource Selection – CPU & Location
  - Scaling Strategies
- Cloud Native Software Development
- Rebound Effects

Problem: Low Utilization

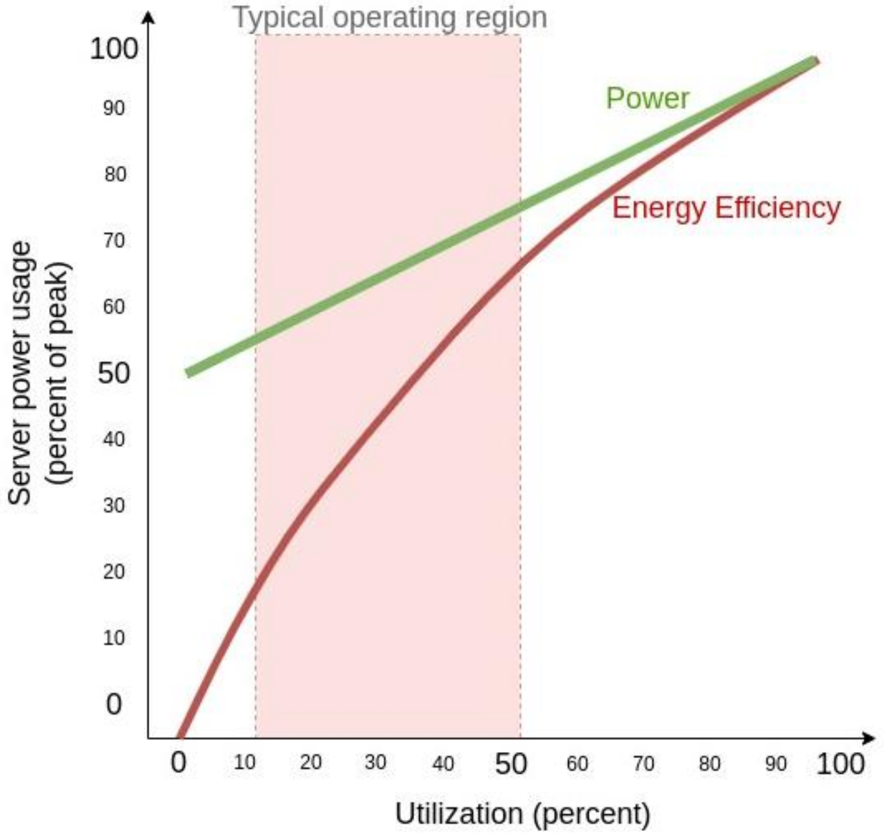


Quelle : „Improving Energy And Power Efficiency In The Data Center", SemiconductorEngineering

High Resource Utilizations

Why should servers be utilized as much as possible? An unused server doesn't consume any electricity, does it?

- Depending on the server, already 50% of power are used without any workload
- Energy efficiency increases with increasing utilization of the server



L. A. Barroso and U. Hölzle, "The Case for Energy-Proportional Computing," in Computer, vol. 40, no. 12, pp. 33-37, Dec. 2007, doi: 10.1109/MC.2007.443.

envite

High Resource Utilizations

Why should virtual machines also be utilized as much as possible?
A virtual unit itself does not consume any power, does it?

- VMs consume very little power, depending on the size of the server and the hypervisor
- Resources can be reserved for potential VMs by the hypervisor
- Poor efficiency when few VMs are provisioned on a hypervisor

→ Cloud providers recommend **stopping unused VMs** to be able to use the resources on the same hypervisor for VMs of other customers

Service Models – Overview

Energy Efficiency

Energy Efficiency

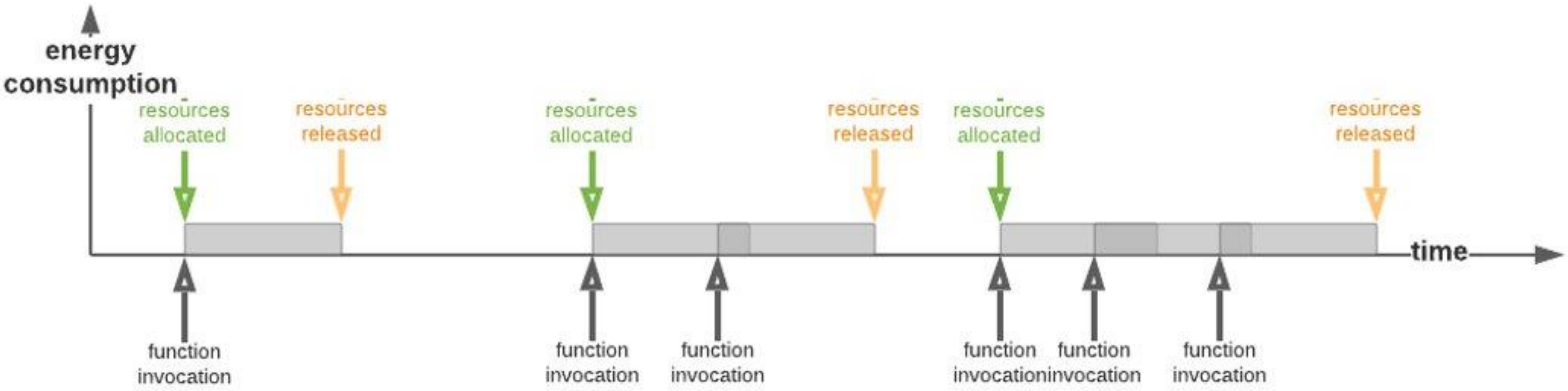
		IaaS	CaaS	PaaS	FaaS	DBaaS	SaaS
server-orient	Unmanaged	Google Compute Engine; Amazon EC2; Azure Virtual Machines	Amazon EKS; Amazon EC2; Azure Kubernetes Service; Azure Virtual Machines; Google Kubernetes Engine	Managed: Elastic Beanstalk; Azure App Services; Google App Engine		Amazon Relational Database Service; Cloud SQL; Microsoft SQL Azure	
task-oriented	Managed		Amazon EKS; AWS Fargate; Azure Kubernetes Service; Azure Container Instances; Azure Container Instances	DigitalOcean App Platform; HEROKU			
	Serverless		Cloud Run Serverless Containers	Cloud Run Serverless Containers; Cloud Build; Buildpacks.io	AWS Lambda; Azure Functions; Google Cloud Functions; DigitalOcean Functions	Cloud Firestore; Amazon DynamoDB; Azure Cosmos DB	Cloud9; salesforce; Office 365; Google Cloud Code
	UT	Weeks / Days	Days / Hours	Day / Hours	Minutes / Seconds		

Service Models – Functions as a Service (FaaS)

- Uses the **serverless** operating model
- Deployment of **functions** in the cloud, which are executed on demand

Aspects of energy efficiency:

- Resources are used to fit; no overprovisioning or underprovisioning
- Scale-to-Zero



Green Cloud Computing

- Energy Consumption of Data Centers
- Metrics & Sustainability of Cloud Providers
- Resource Utilization in Cloud
- **Optimization measures for IaaS and PaaS**
  - Resource Selection – CPU & Location
  - Scaling Strategies
- Cloud Native Software Development
- Rebound Effects

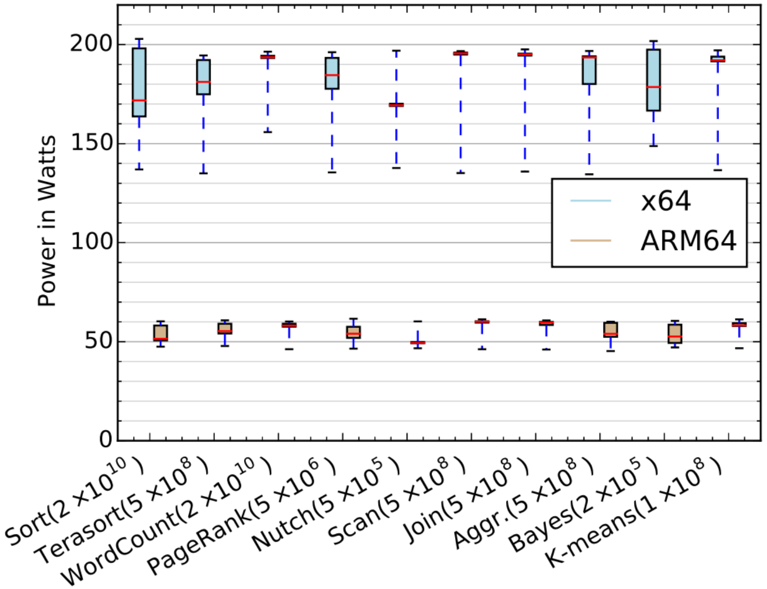
Green Cloud Computing

- Energy Consumption of Data Centers
- Metrics & Sustainability of Cloud Providers
- Resource Utilization in Cloud
- **Optimization measures for IaaS and PaaS**
  - **Resource Selection – CPU & Location**
  - Scaling Strategies
- Cloud Native Software Development
- Rebound Effects

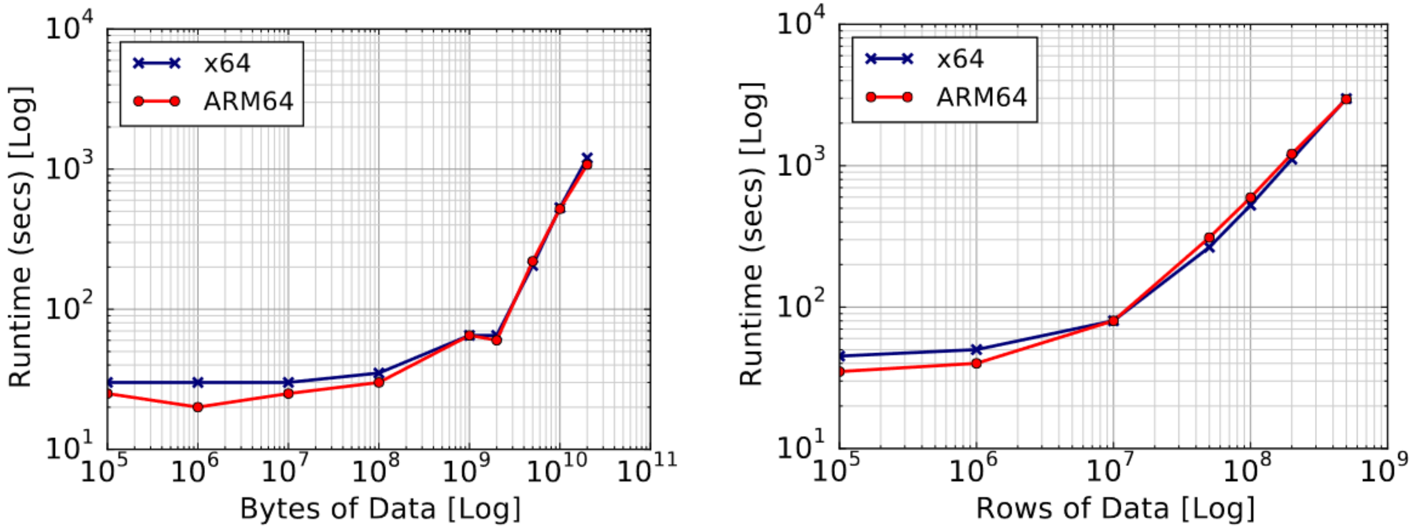
Resource Selection – CPU

Selecting a CPU:

- Architecture of the CPU is an important factor
- **ARM** based CPUs often much more energy efficient than **x86/64** alternatives
- Performance for many application areas comparable
- Recompilation might be necessary as many applications were developed on x64



Power consumption x64 vs. ARM64



Runtime of workloads x64 vs ARM64

J. Kalyanasundaram and Y. Simmhan, "ARM Wrestling with Big Data: A Study of Commodity ARM64 Server for Big Data Workloads," 2017 IEEE 24th International Conference on High Performance Computing (HiPC), Jaipur, India, 2017, pp. 203-212, doi: 10.1109/HiPC.2017.00032.

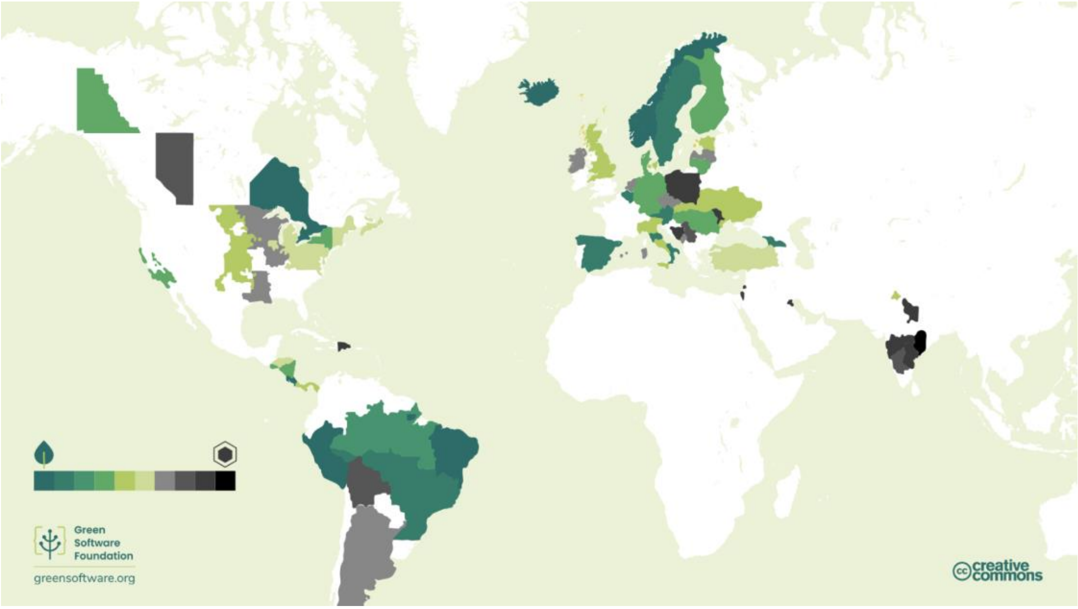
envite

Reduce Carbon Emissions

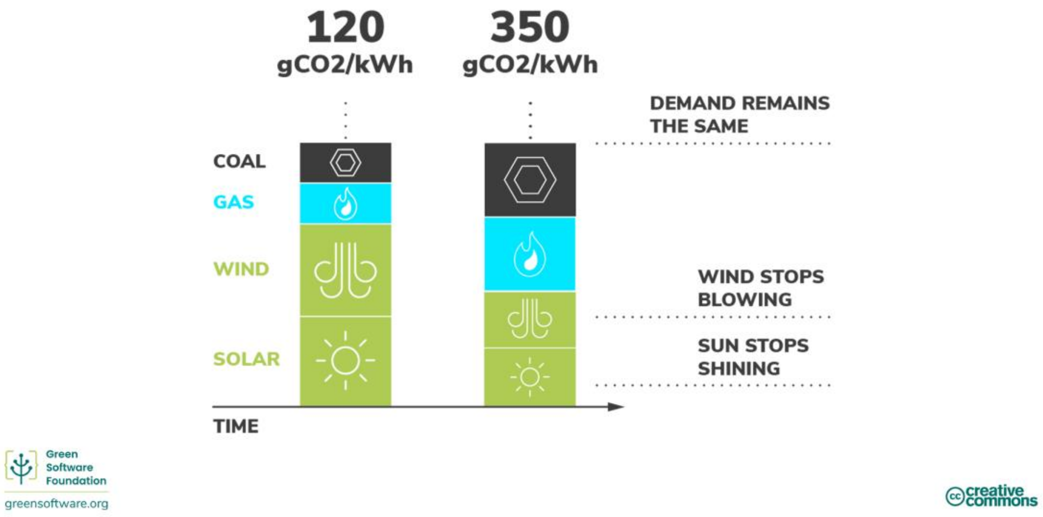
Carbon Intensity

Carbon intensity measures how much CO_2e is emitted per KWh of electricity.

Carbon intensity varies by location



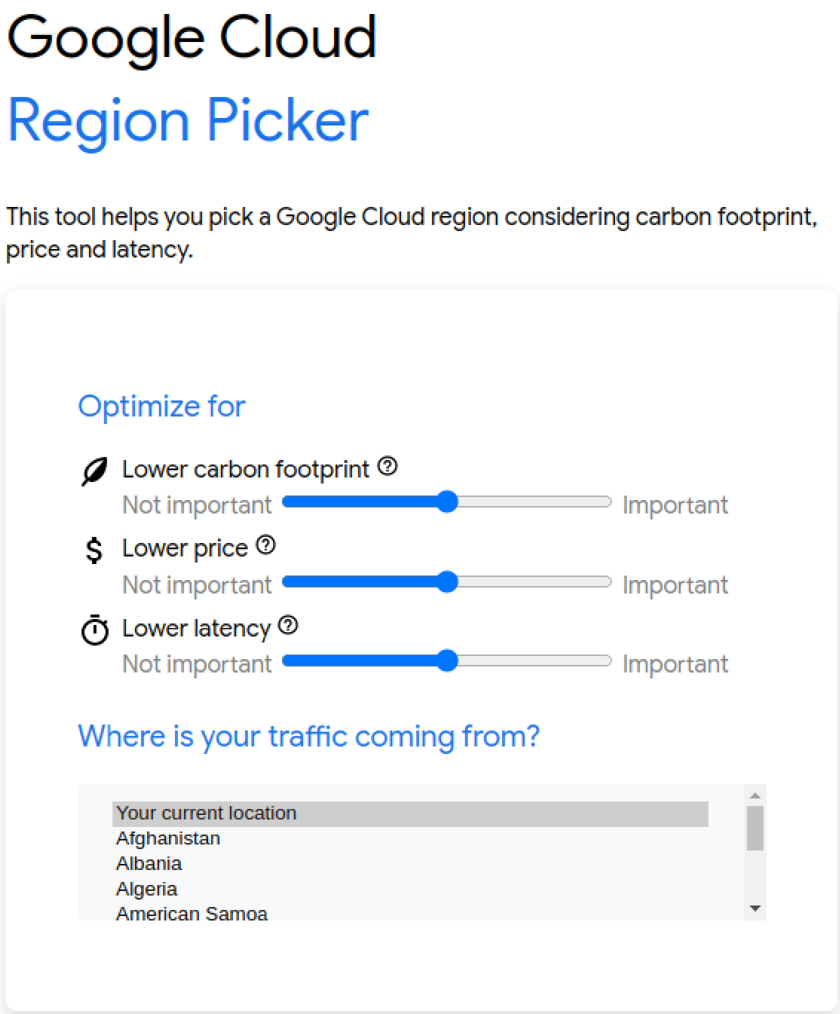
Carbon intensity changes over time



envite

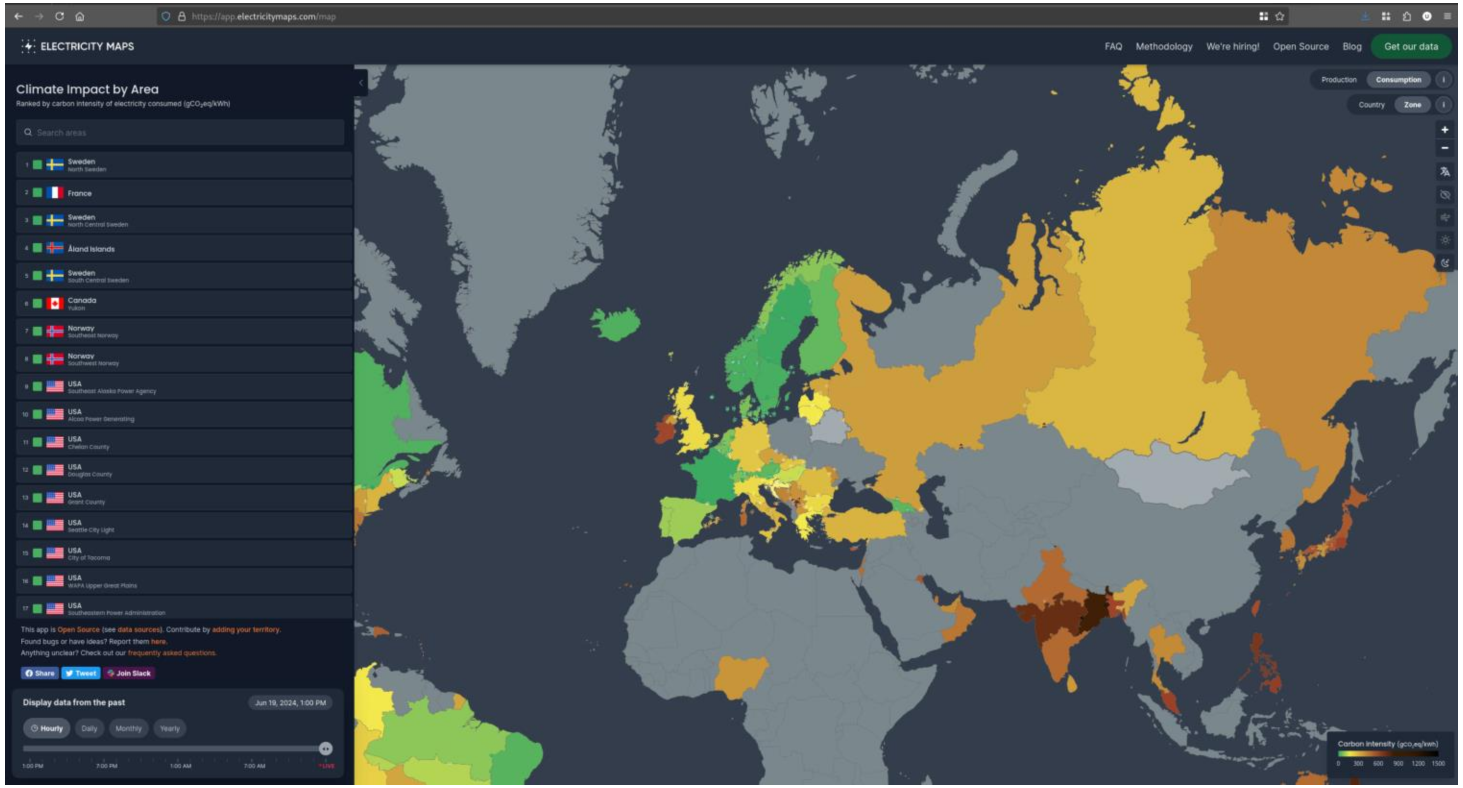
Resource Selection – Locality

- Cloud regions vary significantly in terms of carbon emissions
- Google offers the **Region Picker** to take into account carbon footprint, price, and latency
- Region Picker does not take energy mix into account
  → Nuclear power plants are considered "low carbon“



Resource Selection – Locality

- **Electricity Maps** as an alternative to Google Region Picker
- **WattTime** provides data on power plant emissions by using **measurements from space**



https://app.electricitymaps.com/map

Green Cloud Computing

- Energy Consumption of Data Centers
- Metrics & Sustainability of Cloud Providers
- Resource Utilization in Cloud
- **Optimization measures for IaaS and PaaS**
  - Resource Selection – CPU & Location
  - **Scaling Strategies**
- Cloud Native Software Development
- Rebound Effects

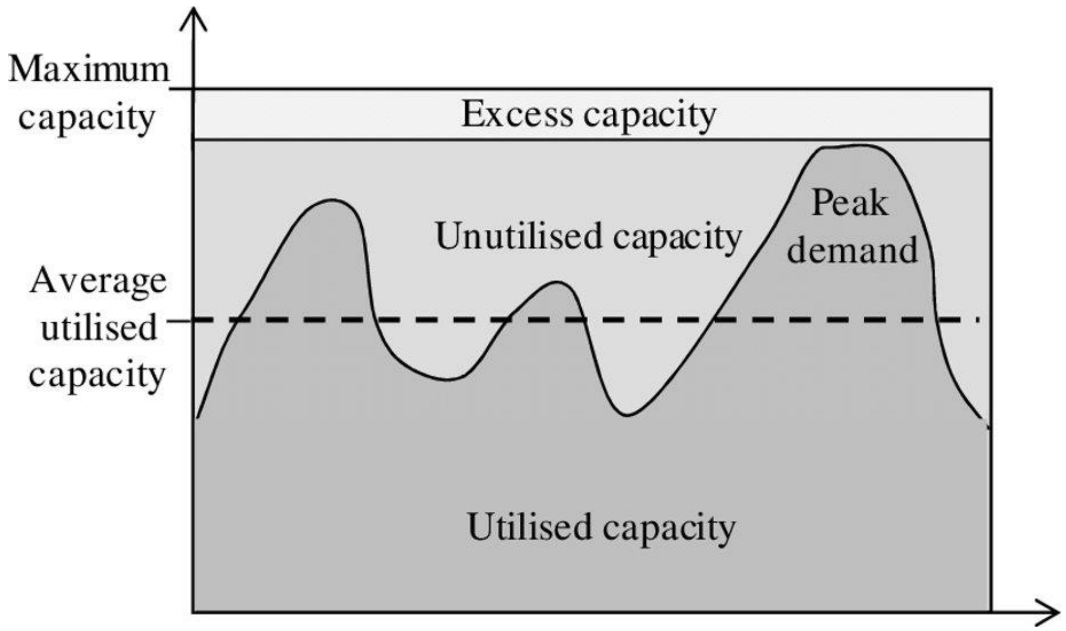
Overprovisioning

Anti-pattern when operating in the Cloud: **Overprovisioning**

- Permanent allocation of resources in order to be able to serve **peak loads**
- On average, resource allocation exceeds actual demand

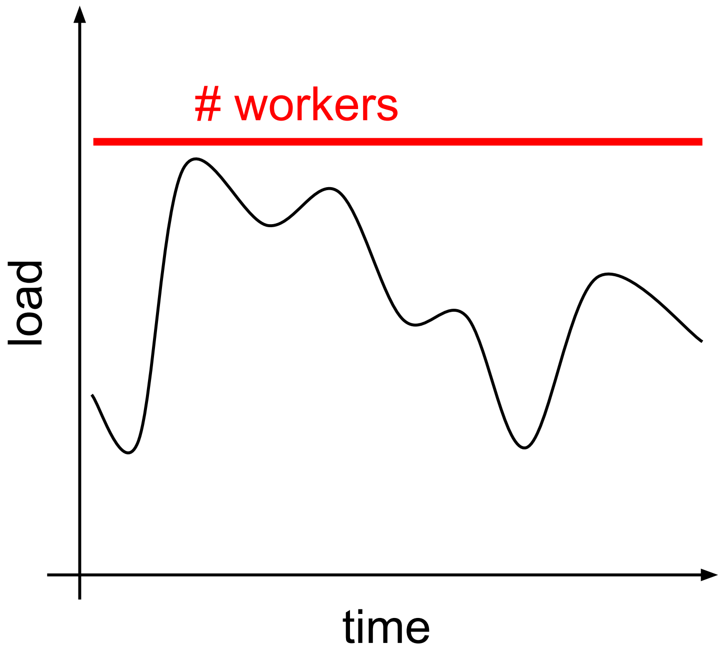
Examples:

- Provision online shop for peak loads in Christmas season
- Provision for execution of scheduled jobs



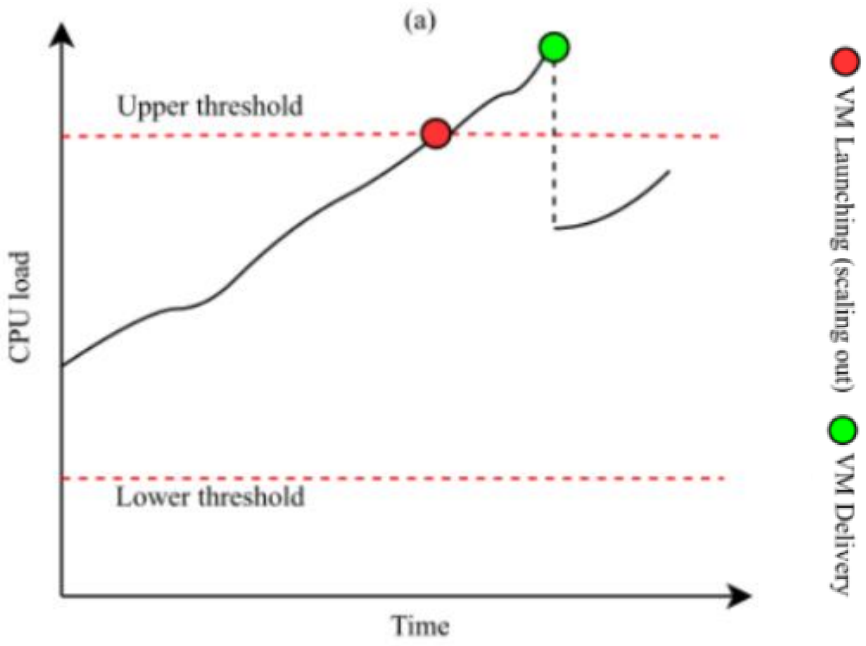
Overprovisioning

- Overprovisioning often occurs when **static or reactive scaling** is used



Static Scaling

No elasticity → overprovisioning needed to handle peak loads



Reactive Scaling (On-Demand)

Resources are not provided fast enough → overprovisioning needed

envite

Scaling Strategies

- Different scaling strategies to avoid overprovisioning and to save resources

Pro-Active /
On-Prediction

Random /
On-Coincidence

Demand Shifting /
On-Availability

Demand Shaping /
On-Availability

Scaling Strategies – Pro-Active

- Pro-active provisioning of resources
- Demand-driven scaling **before** actual demand is present ("On-Prediction")
- Counteracts overprovisioning that occurs due to excessive startup times of VMs or other instances

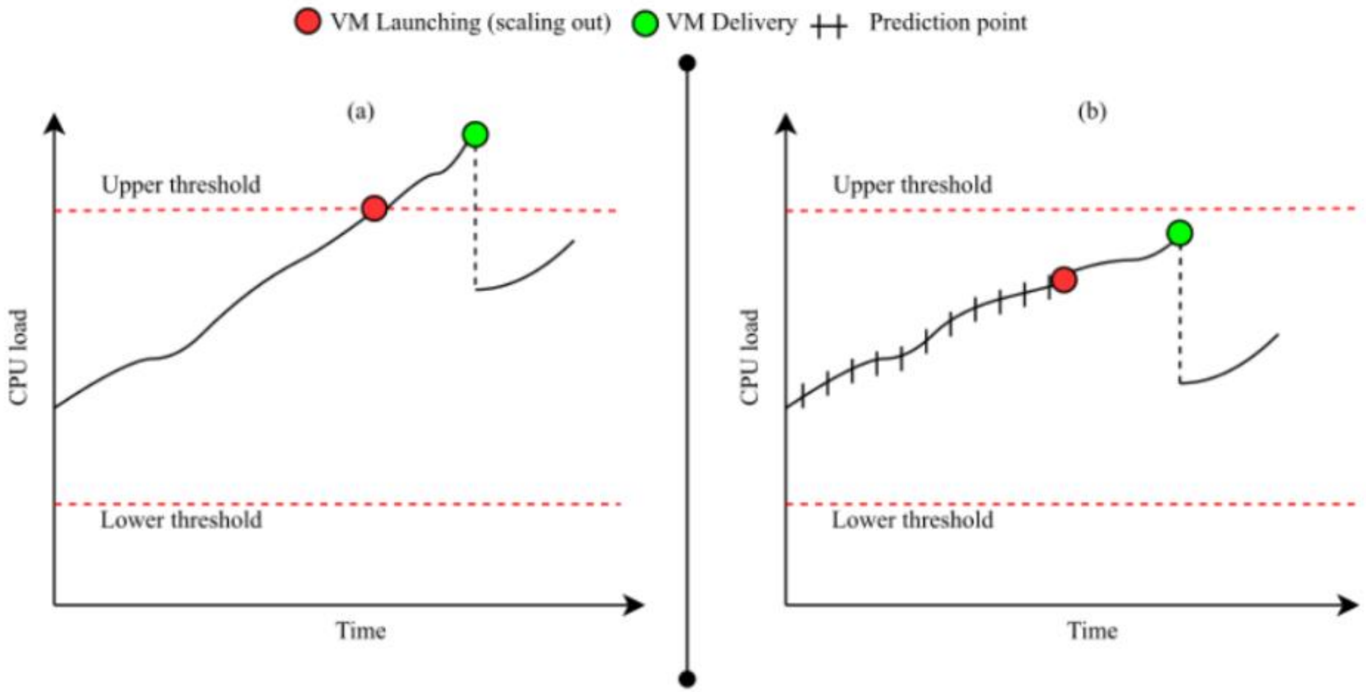


Fig. 1. Elasticity approaches: (a) reactive; (b) proactive.

Scaling Strategies – Random

- **Random provisioning** of resources to equalize peak loads
- Regular workloads will be started at random times to better distribute the load on the system
- Only possible for workloads without user interaction

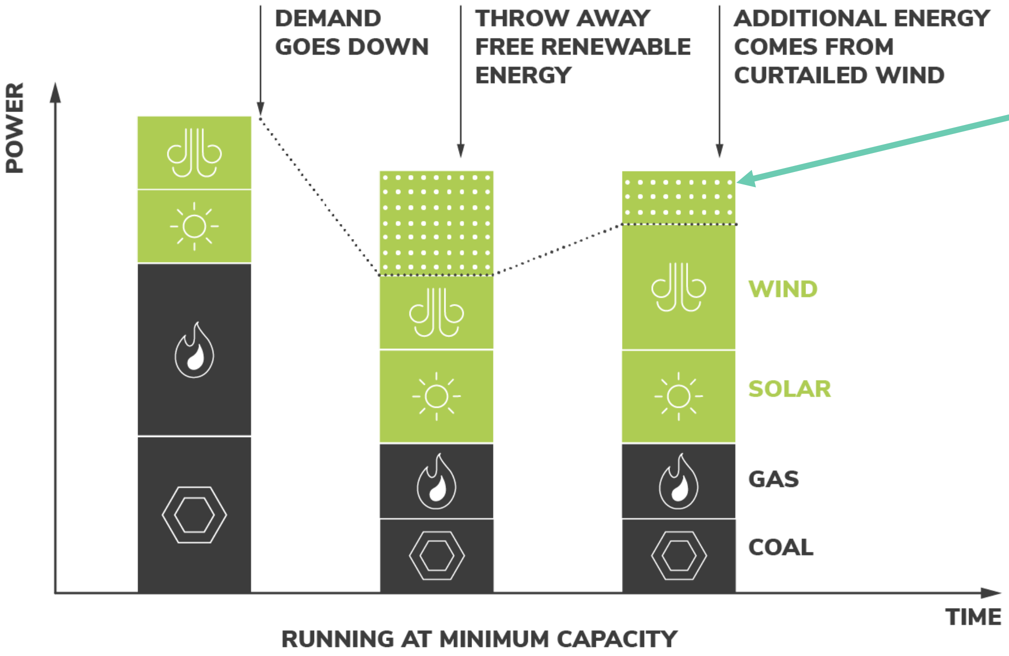
Examples:

- Event-Streaming applications
- Creating a database backup that would otherwise always run at 12 a.m.

Reduce Carbon Emissions

Marginal Carbon Intensity

Electricity supply and demand must always be in balance. **Marginal carbon intensity is the carbon intensity of the power plant that would have to be employed to meet new demand.**

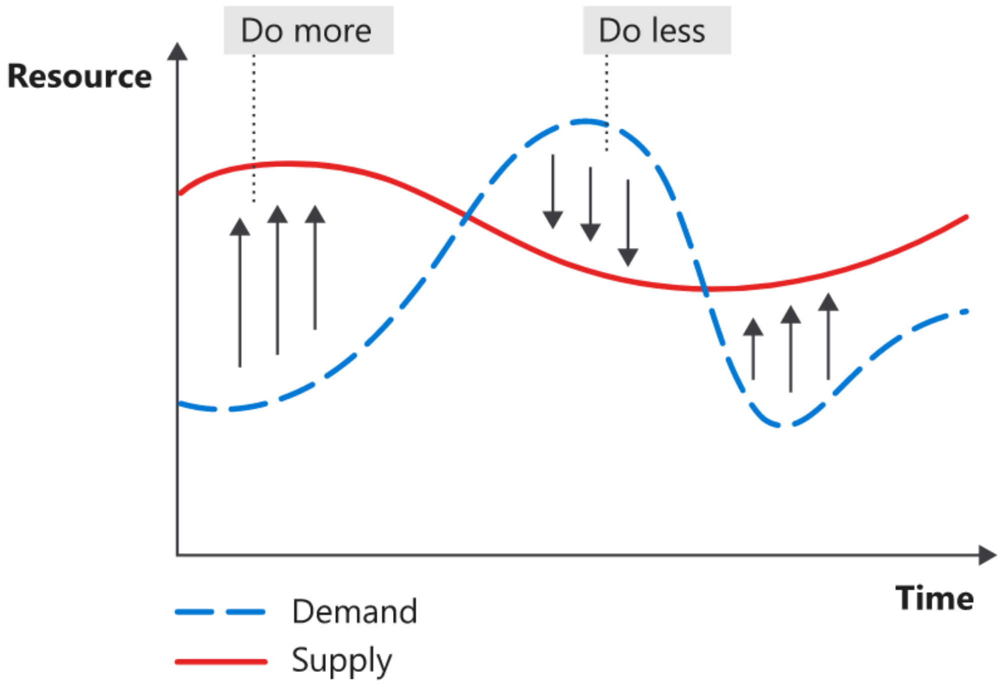


marginal carbon intensity of 0 gCO_2eq/kWh

Carbon Awareness:
Do more when the electricity is cleaner and do less when the electricity is dirtier.
https://learn.greensoftware.foundation/carbon-awareness/

Scaling Strategies – Demand Shaping

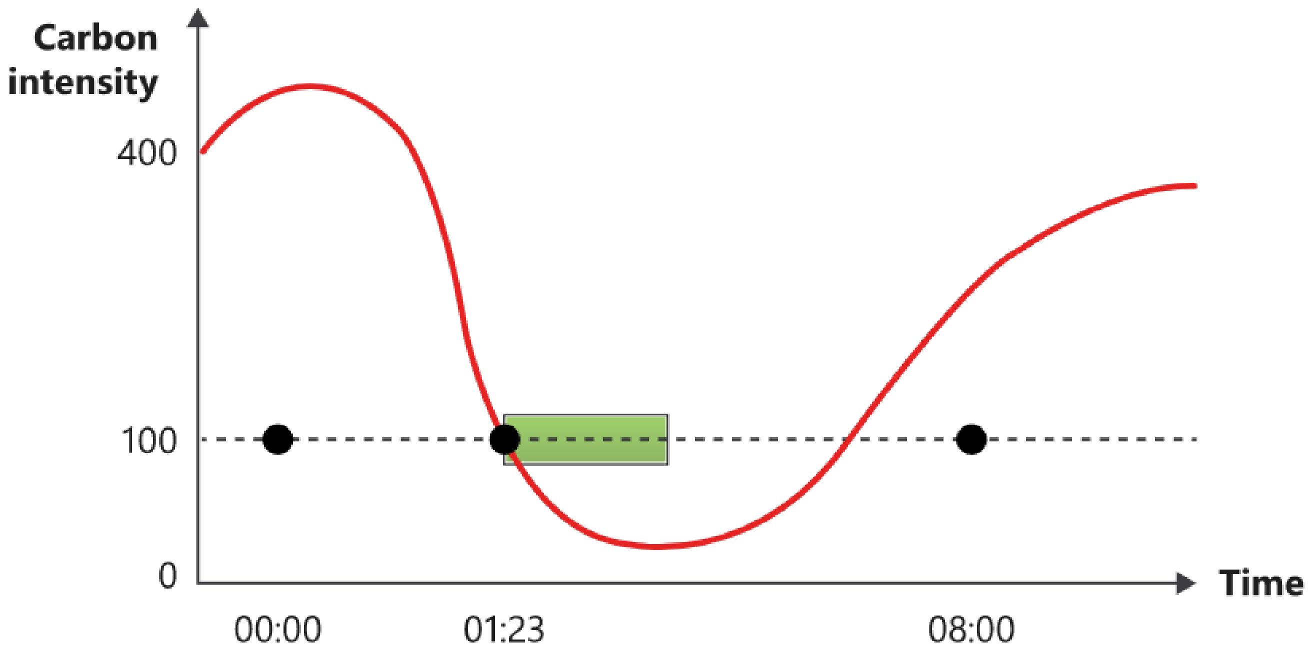
- "On-availability" scaling shapes demand according to **available supply**
- Strategy for flexible workloads without time constraints
- Adjusts the provisioned resources to available resources
  - Examples for constrained resources: **CPU capacity, renewable energies …**
- Workloads can be matched to free capacities of the cloud provider with **on-demand, spot and preemptible instances**



envite

Scaling Strategies – Demand Shifting

- Time-flexible workloads are shifted to **times or regions** where they can be executed with lower carbon emissions



- Alternatively, execution is shifted according to other criteria
  - Times or regions where unused cloud provider resources are available
  - Times or regions where own unused resources are available

Green Cloud Computing

- Energy Consumption of Data Centers
- Metrics & Sustainability of Cloud Providers
- Resource Utilization in the Cloud
- Optimization Measures for IaaS and PaaS
    - Resource Selection – CPU & Location
    - Scaling Strategies
- **Cloud Native Software Development**
- Rebound Effects

Cloud Native Software Development

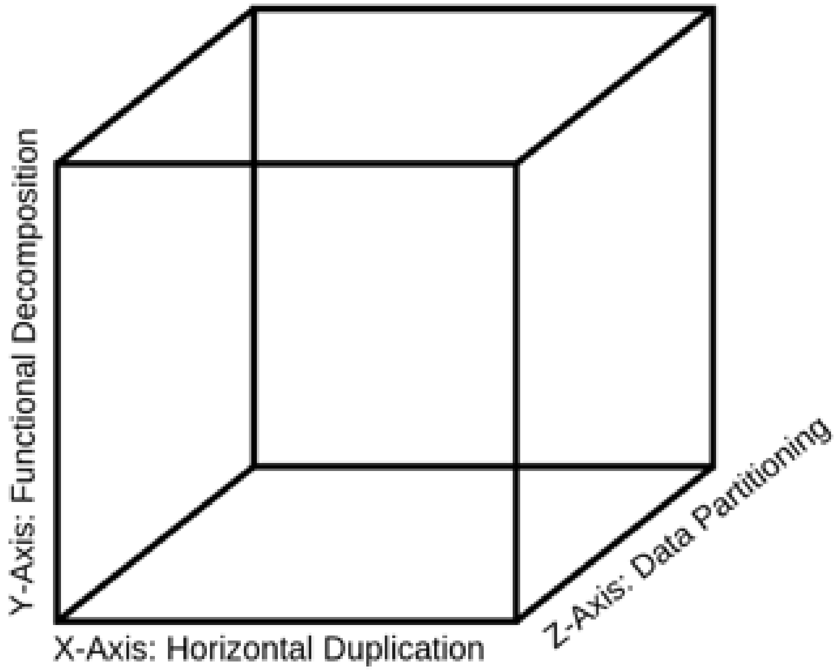
- Cloud-native software development deals with the specialized development of applications in the cloud
- Applications are designed to be…
  - highly scalable,
  - resilient,
  - flexible
- Flexibility of the cloud infrastructure is important
- Cloud development offers many advantages over solutions on-premises in terms of energy efficiency

Advantages can only be used if applications are designed to be operated in the cloud!

Cloud Native Software Development

Prerequisites for taking advantage of all the benefits in terms of energy efficiency:

- **Fast startup times** for flexible scalability
- **Fast "graceful shutdowns"** to be able to shut down applications without data corruption
- **Available failover strategy** to get back online quickly
- Should be **stateless**
- Have good **scalability** according to the scale cube model



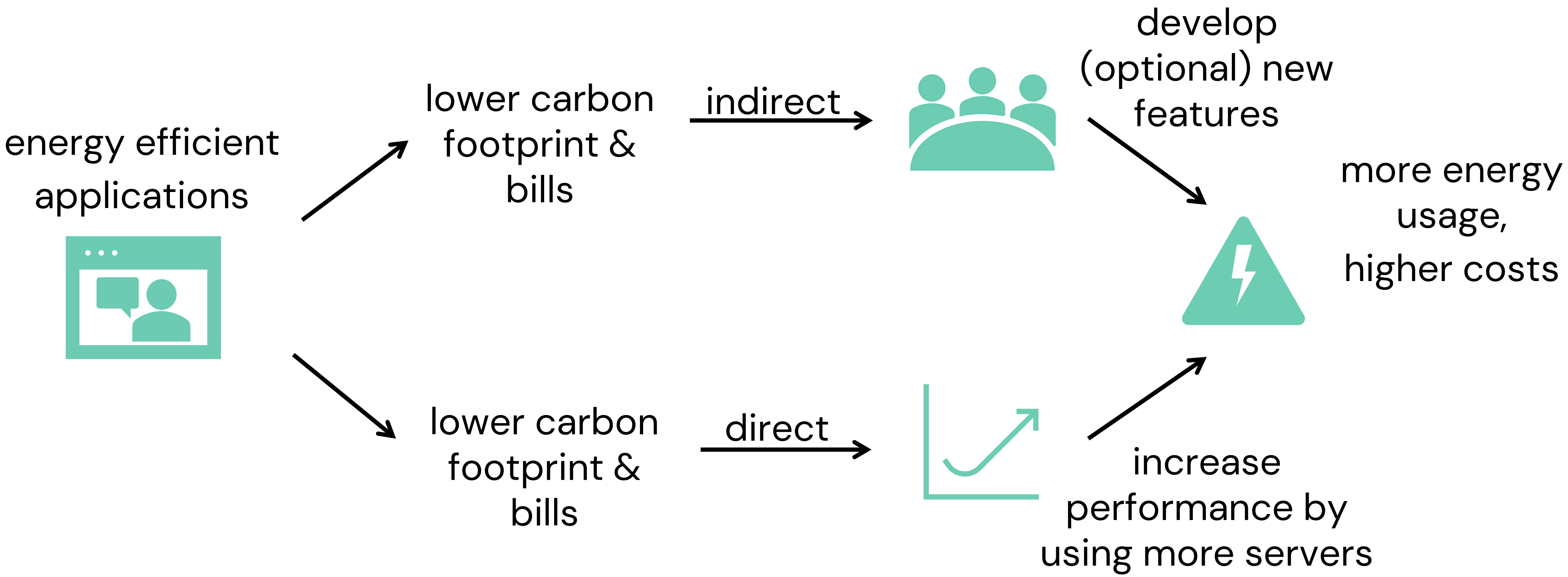
Applications should follow the **12 Factor Method** (https://12factor.net/)

Green Cloud Computing

- Energy Consumption of Data Centers
- Metrics & Sustainability of Cloud Providers
- Resource Utilization in the Cloud
- Optimization Measures for IaaS and PaaS
  - Resource Selection – CPU & Location
  - Scaling Strategies
- Cloud Native Software Development
- **Rebound Effects**

Rebound Effects

- Optimizations lead to energy and costs savings
- **Risk**: savings encourage changed behavior and lead to increased energy usage



Questions?

Thank you!

Jan Kirchner

jan.kirchner@envite.de

Vincent Rossknecht

Vincent.rossknecht@envite.de

envite consulting GmbH

www.envite.de

Serverless with Java

Best Practices for Serverless with Java

envite

Exercise

https://github.com/envite-consulting/showcase-graalvm

Book Demo App

```
+------------------+      +---------------------+
|                  |      |                     |
| graalvm-demo-book |<------| graalvm-demo-utilizer |
|                  |      |                     |
|    POST /books   |      |    POST /load-test  |
|    insert books  |      |    - configure url, |
|                  |      |      endpoint       |
+--------|--------+      |    - set numBooks,  |
         |                |      numRequests,   |
         |                |      numUsers       |
         v                |                     |
 +----------------+       +---------------------+
 |                |
 |    MongoDB     |
 |   Store books  |
 |                |
 +----------------+
```

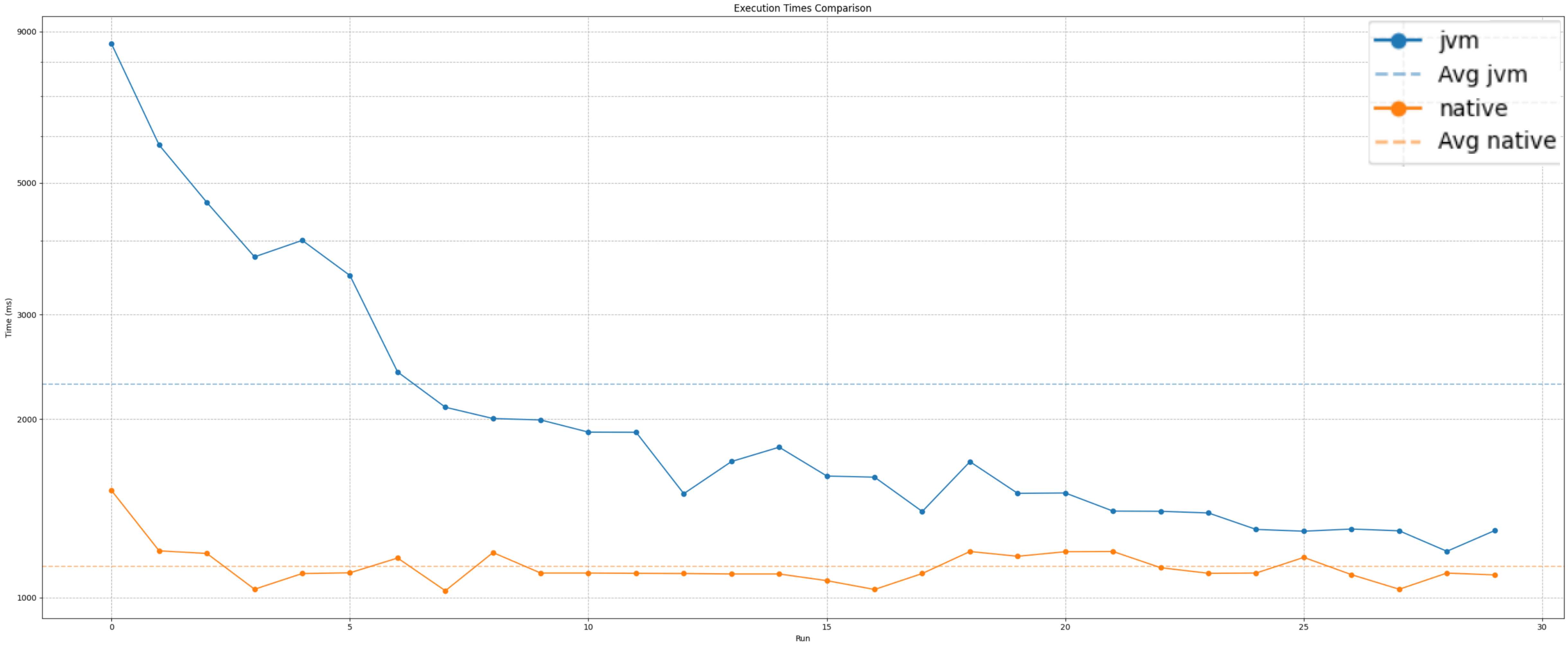
Startup Time

0.5 CPU and 512MB Memory

Java	10s
GraalVM	0.2s

Load Test (30 runs a‘ 600 requests)

0.5 CPU and 512MB Memory



envite

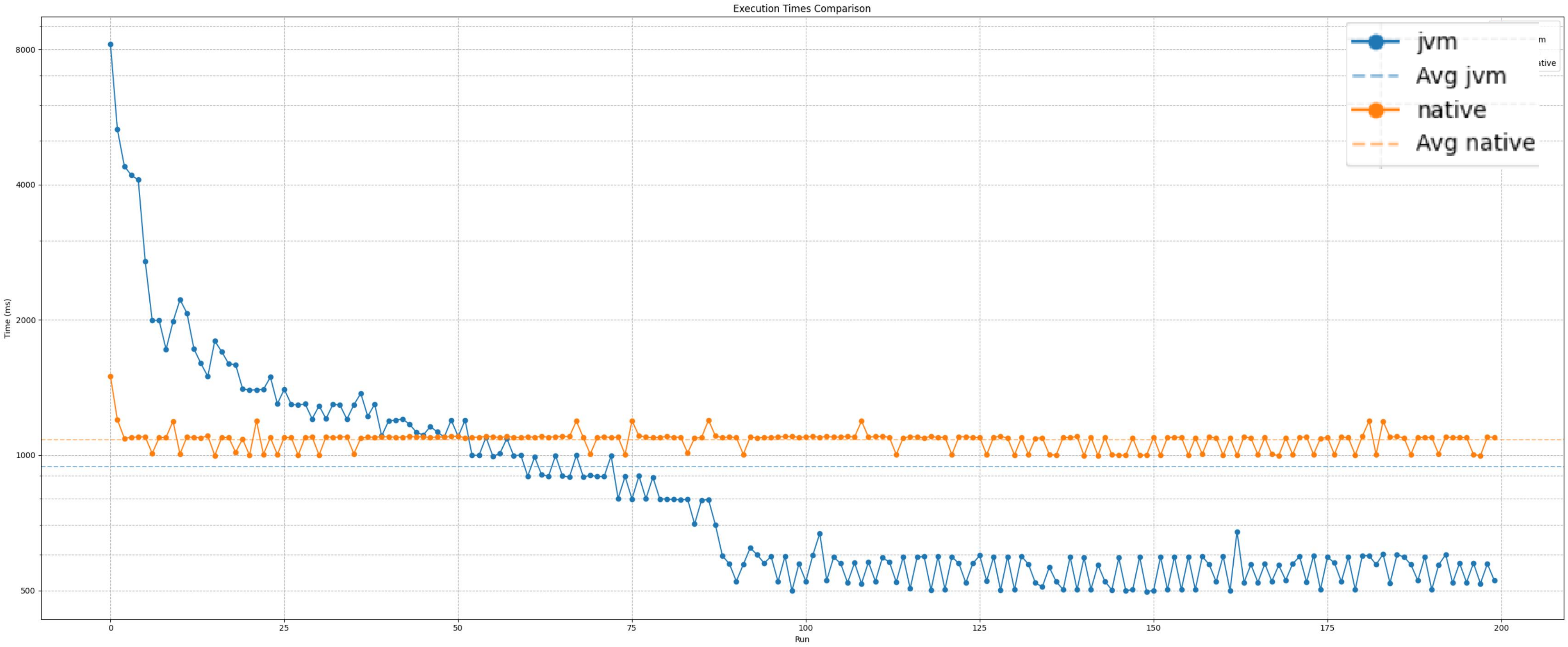
Load Test (30 runs a‘ 600 requests)

0.5 CPU and 512MB Memory

envite

Load Test (200 runs a‘ 600 requests)

0.5 CPU and 512MB Memory



Load Test (200 runs a‘ 600 requests)

0.5 CPU and 512MB Memory

envite

Load Test (200 runs a‘ 600 requests)

0.5 CPU and 512MB Memory

Load Test (150 runs a‘ 600 requests)

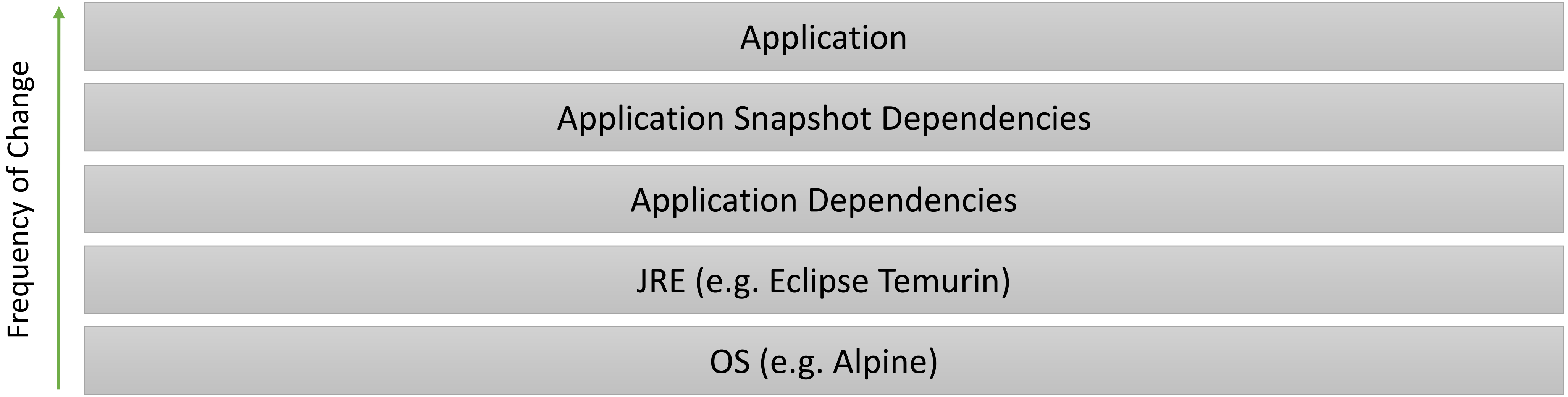
0.5 CPU and 512MB Memory

Container Image Layers

Best Practices for Small Image Footprint

envite

Java Container Image Layers



envite

Spring Boot Layered Container Image (Example)

Frequency of Change

0.1 MB	Application
0.4 MB	Spring Boot Loader
0.0 MB	Application Snapshot Dependencies
31.0 MB	Application Dependencies

docker.io/eclipse-temurin:21.0.3_9-jre-alpine

182.0 MB	JRE (Eclipse Temurin 21)
7.4 MB	OS (Alpine 3.19)

Spring Boot Layered Container Image (Example)

Dockerfile (Multi-Stage build)

```
##############################
# Application Builder
##############################
FROM eclipse-temurin:21.0.3_9-jdk-alpine as builder

ARG BUILDER_WORKDIR
WORKDIR ${BUILDER_WORKDIR}
# Download dependencies
COPY pom.xml mvnw ./
COPY .mvn .mvn
RUN ./mvnw verify --fail-never
# Build layered application jar
COPY src src
RUN ./mvnw package

ARG BUILDER_DIST_DIR
# Extract layers from application jar file
RUN java -Djarmode=tools -jar target/demo-book-*.jar extract \
      --layers --launcher --destination "${BUILDER_DIST_DIR}"
```

```
##############################
# Build Layered Image
##############################
FROM eclipse-temurin:21.0.3_9-jre-alpine

RUN adduser -D -s /bin/false -u 1000 appuser

WORKDIR /opt/dist
ARG BUILDER_DIST_DIR
COPY --from=builder ${BUILDER_DIST_DIR}/dependencies/ ./
COPY --from=builder ${BUILDER_DIST_DIR}/snapshot-dependencies/ ./
COPY --from=builder ${BUILDER_DIST_DIR}/spring-boot-loader/ ./
COPY --from=builder ${BUILDER_DIST_DIR}/application/ ./

CMD exec java ${JAVA_OPTS} -server \
      "org.springframework.boot.loader.launch.JarLauncher"

EXPOSE 8080

USER appuser
```

pom.xml (enable layered jar)

```xml
<plugin>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</artifactId>
    <configuration>
        <layers>
            <enabled>true</enabled>
        </layers>
    </configuration>
</plugin>
```

Spring Boot Layered Container Image (Example)

- (+) Changes to the application code does not require re-build and re-distribution of dependencies
- (+) Unpacking jar also reduces startup time *(1)*

Recommendations

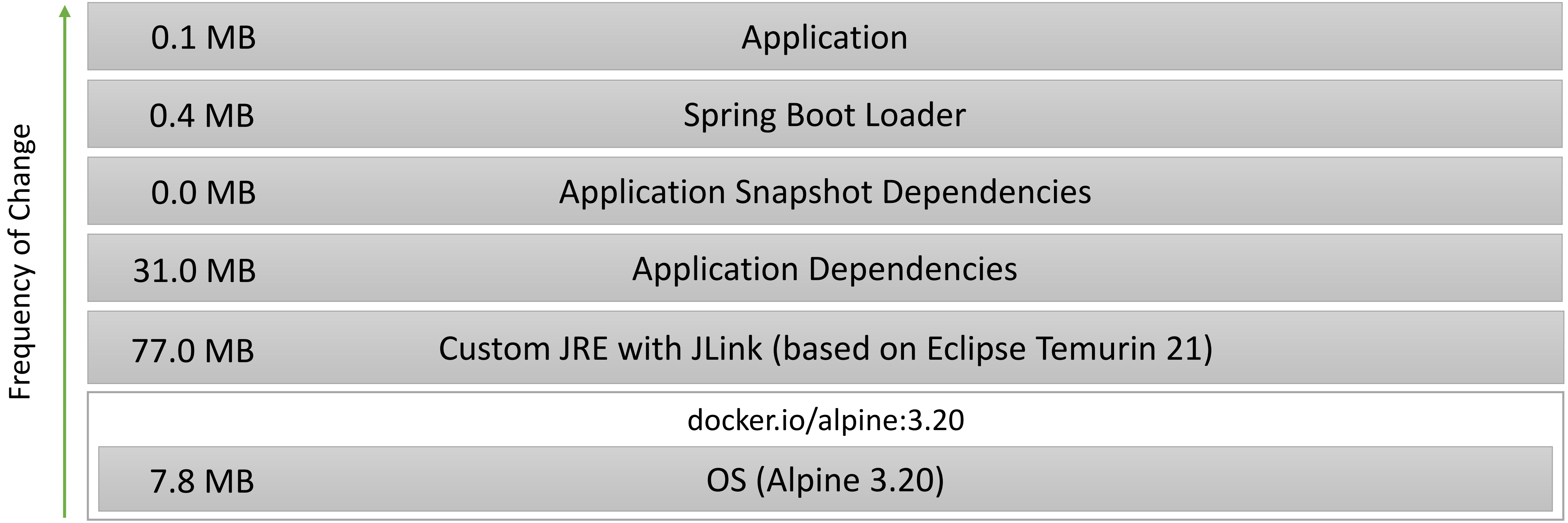
- Use minimalistic base image (Alpine + JRE), to keep image size small
- Use same base image in your applications to increase re-use of layers on target hosts

Important

- In-order to avoid re-build of layers without change you need to do additional steps on CI/CD Pipeliens to re-use the image cache

(1) https://docs.spring.io/spring-boot/reference/deployment/efficient.html#deployment.efficient.unpacking

Java Container Image with JLink (Example)



Java Container Image with JLink (Example)

Dockerfile (Multi-Stage build)

```
#############################
# Application Builder
#############################
FROM eclipse-temurin:21.0.3_9-jdk-alpine as builder

ARG BUILDER_WORKDIR
WORKDIR ${BUILDER_WORKDIR}
# Download dependencies
COPY pom.xml mvnw ./
COPY .mvn .mvn
RUN ./mvnw verify --fail-never
# Build layered application jar
COPY src src
RUN ./mvnw package

ARG BUILDER_DIST_DIR
# Extract layers from application jar file
RUN java -Djarmode=tools -jar target/demo-book-*.jar extract \
      --layers --launcher --destination "${BUILDER_DIST_DIR}"

# Detect JRE modules required by your application
RUN jdeps --print-module-deps --ignore-missing-deps \
      --multi-release 21 --recursive \
      --class-path "${BUILDER_DIST_DIR}/dependencies/BOOT-INF/lib/*" \
      "${BUILDER_DIST_DIR}" > target/deps.info
ARG BUILDER_JRE_DIR
# Build custom JRE which only contains required modules
RUN jlink --strip-debug --compress zip-6 --no-header-files --no-man-pages \
      --add-modules "$(cat target/deps.info),jdk.naming.dns,jdk.crypto.ec" \
      --output "${BUILDER_JRE_DIR}"
# Generate CDS archive for custom JRE
RUN "${BUILDER_JRE_DIR}/bin/java" -Xshare:dump
```

```
#############################
# Build Layered Image
#############################
FROM alpine:3.20

RUN adduser -D -s /bin/false -u 1000 appuser

ENV JAVA_HOME="/opt/java/openjdk"
ENV PATH=$JAVA_HOME/bin:$PATH
ARG BUILDER_JRE_DIR
COPY --from=builder ${BUILDER_JRE_DIR} $JAVA_HOME

WORKDIR /opt/dist
ARG BUILDER_DIST_DIR
COPY --from=builder ${BUILDER_DIST_DIR}/dependencies/ ./
COPY --from=builder ${BUILDER_DIST_DIR}/snapshot-dependencies/ ./
COPY --from=builder ${BUILDER_DIST_DIR}/spring-boot-loader/ ./
COPY --from=builder ${BUILDER_DIST_DIR}/application/ ./

CMD exec java ${JAVA_OPTS} -server \
      "org.springframework.boot.loader.launch.JarLauncher"

EXPOSE 8080

USER appuser
```

Spring Boot Layered Container Image (Example)

- (+) Reduced image size (in this example 105MB smaller)
- (+) Reduced memory consumption during runtime
- (-) If every application uses its own custom JRE, the JRE layer cannot be shared anymore
- (-) If code change adds requirement for an additional JRE module, the JRE needs to be re-build and re-distributed

Open Issues

- JRE and OS security patches may require re-build and re-distribution of entire image
  - Redcuce probability that you are affected
    - With custom build JRE, you only need to upgrade if a security issue affects a module which you are using
    - With very small OS base image, probability is reduced that there is a security issue
    - You can further reduce size of OS by build your custom OS image e.g. with apko, which only includes shared libraries you actually need