



university of
 groningen

Actionable observability into the emissions of cloud-based software

Vasilios Andrikopoulos

vandriko.github.io



Takeaways

01

$E(\text{software})$: non-trivial

02

$E(\text{software})$: hard(er) mode

03

$\mathcal{O}^+(E(\text{software}))$: essential



Software energy consumption

Establishing the relation between a software system & its consumed energy is non-trivial



Emissions and energy

Software Carbon Intensity (SCI) [ISO/IEC 21031:2024]

Carbon emitted per kWh
 of energy, gCO₂/kWh

Carbon emitted through
 the hardware that the
 software is running on

$$\text{SCI} = ((\text{E} * \text{I}) + \text{M}) \text{ per R}$$

Energy consumed by
 software in kWh

Functional Unit; this is how
 software scales, for example
 per user or per device



Consumed energy as the optimization target

Out of
(direct) control

Carbon emitted per kWh
of energy, gCO2/kWh

Out of
(direct) control

Carbon emitted through
the hardware that the
software is running on

$$SCI = ((E * I) + M) \text{ per } R$$

Partial
control

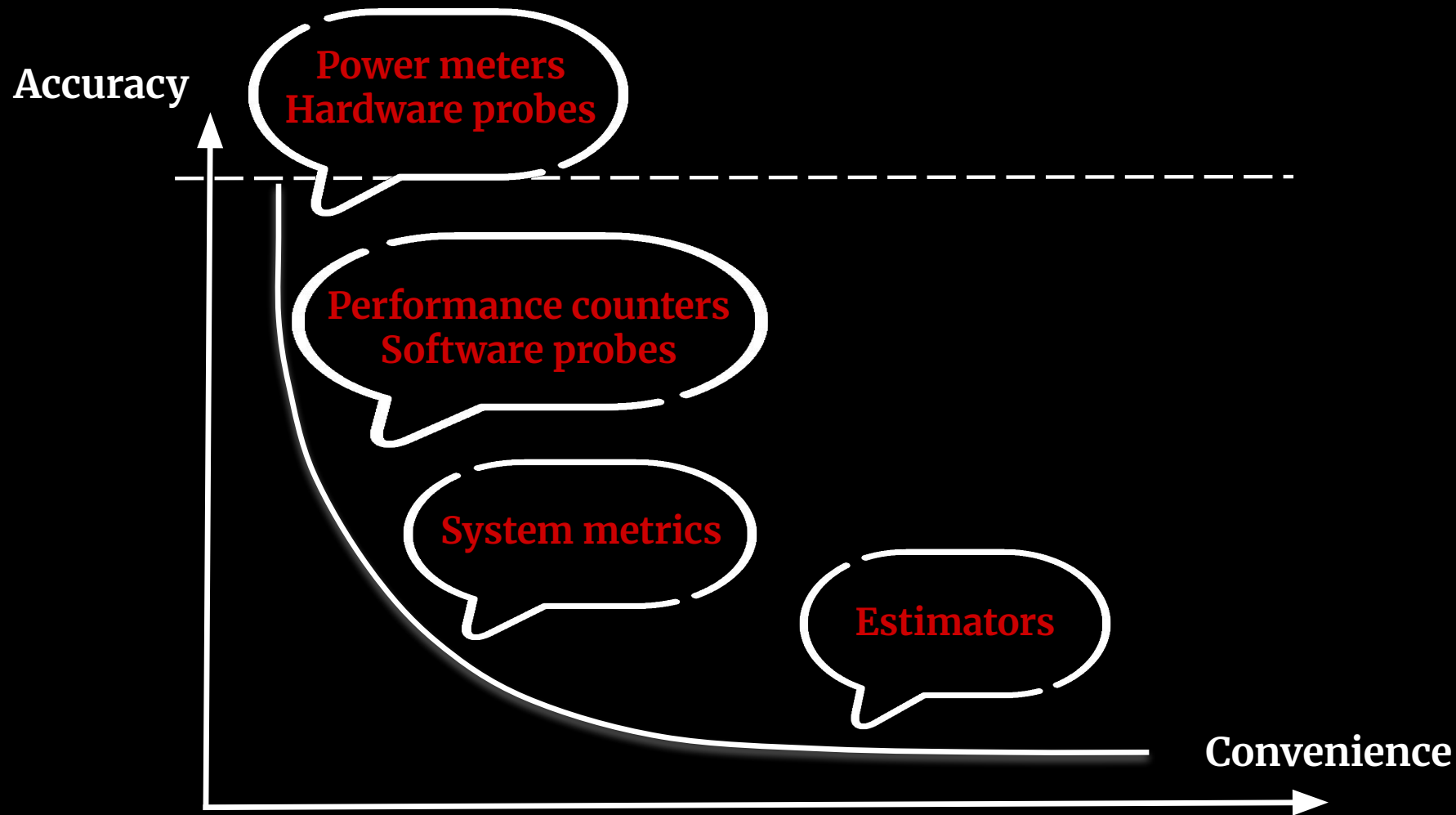
Energy consumed by
software in kWh

Functional Unit; this is how
software scales, for example
per user or per device

In
control



Measuring energy consumption





Empirical knowledge

We identified three recurring problems in the answers in our base group of questions: (i) **misconceptions** about software energy consumption and how it can be reduced, (ii) solutions that are **applicable in certain contexts** being presented as universal, and (iii) **lack of tools**, in particular, measurement tools.

Mining Questions about Software Energy Consumption

Gustavo Pinto[†], Fernando Castor[†], Yu David Liu[‡]

[†]Federal University of Pernambuco
Recife, PE, Brazil
{ghlp, castor}@cin.ufpe.br

[‡]SUNY Binghamton
Binghamton, NY 13902, USA
davidL@cs.binghamton.edu

ABSTRACT

A growing number of software solutions have been proposed to address application-level energy consumption problems in the last few years. However, little is known about how much software developers are concerned about energy consumption, what aspects of energy consumption they consider important, and what solutions they have in mind for improving energy efficiency. In this paper we present the first empirical study on understanding the views of application programmers on software energy consumption problems. Using STACKOVERFLOW as our primary data source, we analyze a carefully curated sample of more than 300 questions and 550 answers from more than 800 users. With this data, we observed a number of interesting findings. Our study shows that practitioners are aware of the energy consumption problems: the questions they ask are not only *diverse* – we found 5 main themes of questions – but also often more *interesting* and *challenging* when compared to the control question set. Even though energy consumption-related questions are popular when considering a number of different popularity measures, the same cannot be said about the quality of their answers. In addition, we observed that some of these answers are often flawed or vague. We contrast the advice provided by these answers with the state-of-the-art research on energy consumption. Our summary of software energy consumption problems may help researchers focus on what matters the most to software developers and end users.

1. INTRODUCTION

Nowadays, thanks to the rapid proliferation of mobile phones, tablets, and unwired devices in general, energy efficiency is becoming a key software design consideration where the energy consumption is closely related to battery lifetime. It is also of increasing interest in the non-mobile arena, such as data centers and desktop environments. Energy-efficient solutions are highly sought after across the compute stack, with more established results through innovations in hardware/architecture [2, 14, 28], operating systems [10, 19, 24], and runtime systems [8, 25, 31]. In recent years, there is a growing interest in studying energy consumption from higher layers of the compute stack and most of these studies focus on application software [13, 15, 18, 23, 26, 29]. These approaches complement prior hardware/OS-centric solutions, so that improvements at the hardware/OS level are not cancelled out at the application level, *e.g.*, due to misuses of language/library/application features.

We believe a critical dimension to further improve energy efficiency of software systems is to understand *how software developers think*. The needs of developers and the challenges they face may help energy-efficiency researchers stay focused on the real-world problems. The collective wisdom shared by developers may serve as a practical guide for future energy-aware and energy-efficient software development. The conceptually incorrect views they hold may inspire educators to develop more state-of-the-art curricula.



Empirical knowledge **redux**

[...] most practitioners struggle with **fundamental** energy concepts due to complex API documentation and practical experience required;

Energy-Efficient Software Development: A Multi-dimensional Empirical Analysis of Stack Overflow

Bihui Jin
 University of Waterloo
 Waterloo, Canada
 bihui.jin@uwaterloo.ca

Heng Li
 Polytechnique Montréal
 Montréal, Canada
 heng.li@polymtl.ca

Pengyu Nie
 University of Waterloo
 Waterloo, Canada
 pynie@uwaterloo.ca

Ying Zou
 Queen's University
 Kingston, Canada
 ying.zou@queensu.ca

Abstract

Energy consumption of software applications has emerged as a critical concern for developers to contemplate in their daily development processes. Previous studies have surveyed a limited number of developers to understand their viewpoints on energy consumption. We complement these studies by analyzing a meticulously curated dataset of 1,193 Stack Overflow (SO) questions concerning energy consumption. These questions capture real-world energy-related challenges in practice. To understand practitioners' perceptions, we investigate the intentions behind these questions, semantic topics, and associated technologies (e.g., programming languages). Our results reveal that: (i) the most prevalent energy consumption topic is about balancing *Positioning* usage; (ii) efficiently handling data is particularly challenging, with these questions having the longest response times; (iii) practitioners primarily ask questions to understand a concept or API related to energy consumption; and (iv) practitioners are concerned about energy consumption across multiple levels—hardware, operating systems, and programming languages—during energy efficient software development. Our findings raise awareness about energy consumption's impact on software development. We also derive actionable implications for energy optimization at different levels (e.g., optimizing API usage or hardware accesses) during energy-aware software development.

ACM Reference Format:

Bihui Jin, Heng Li, Pengyu Nie, and Ying Zou. 2026. Energy-Efficient Software Development: A Multi-dimensional Empirical Analysis of Stack Overflow. In *2026 IEEE/ACM 48th International Conference on Software Engineering (ICSE '26)*, April 12–18, 2026, Rio de Janeiro, Brazil. ACM, New York, NY, USA, 13 pages. <https://doi.org/10.1145/3744916.3773177>

including design, implementation, testing, and maintenance [29]. Energy-efficient software can save energy, extend battery life, and enhance user experience [80], helping mitigate the growing trend of global electricity consumption. Practitioners now view energy efficiency as a key software property and have demonstrated willingness to learn about energy issues in software development [64].

A few empirical studies have been conducted to understand practitioners' perceptions of energy consumption. For instance, prior studies [64, 80] conduct online surveys with 122 and 464 practitioners, respectively, who self-identify as experienced developers and testers across various application domains. The studies find that 65% of practitioners on Reddit recognize energy usage as a crucial factor in software quality [80], and many are willing to sacrifice other requirements to reduce energy consumption [64]. However, these studies only survey opinions rather than the concrete barriers practitioners face when developing energy-efficient software. The most similar work to ours is by Pinto et al. [84], who study 325 Stack Overflow (SO) questions from 2008 to 2013 and identify five primary themes of energy-related questions (e.g., measurements and code design). Given rapid technological changes (e.g., IoT devices), practitioners likely face different challenges today than a decade ago, making the earlier studies less relevant to current practitioners.

We follow the same methodology outlined in Pinto et al. [84] to collect energy-related posts. Through keyword search followed by manual verifications, we curated a dataset of 1,193 energy-related SO questions, which contains **larger and more recent** questions compared to the work by Pinto et al. (325 questions by 2013). Inspired by recent work on analyzing SO posts [16, 91, 109, 112], we perform **LDA topic modeling instead of thematic analysis** to



Tactics

All Tags

AWS
ai
algorithm-design
architecture
browser
cloud
cloud-efficiency
cloud-principles
cost-reduction
data-centric
data-compression
data-processing
deployment
design
documentation
edge-computing
email-sharing
energy-efficiency
energy-footprint
enterprise-optimization
green-ai
hardware

<- Back to all tags

Tactic(s) tagged with "energy-footprint"

- Select nearby regions with better renewable energy rates (AT)
- Avoid use of byte-code (SP)
- Batch I/O (SP)
- Code migration (SP)
- Compiler optimization (SP)
- Decrease algorithmic complexity (SP)
- Efficient GUI (SP)
- Free or unmap unneeded memory (SP)
- Keep 3rd party software up-to-date (SP)
- Lazy loading (SP)
- Less frequent or avoiding polling (SP)
- Put application to sleep (SP)
- Reduce data redundancy (SP)
- Reduce memory leaks (SP)
- Reduce QoS dynamically (SP)
- Reduce transparency and abstractions (SP)
- Static GUI (SP)
- Use asynchronous I/O (SP)
- Use efficient queries (SP)
- Use JIT compiler (SP)
- Use low-level programming (SP)
- Choose an energy efficient drift detection algorithm (SP)
- Static detection of flaky tests (SP)
- Mock highly consuming functions (SP)
- Use energy aware test suite prioritization (SP)
- Follow-the-sun test scheduling (SP)
- Rethink Digital Communication and Meetings (SP)
- Limit Ensemble Size (AT)
- RAG Context Caching (AT)
- RAG Context Filtering and Compression (AT)



Patterns

<https://patterns.greensoftware.foundation/>

Green Software Patterns

Green Software Patterns

Guide >

Catalog >

Artificial Intelligence (AI) >

Cloud >

Web >

Avoid chaining critical requests

Avoid an excessive DOM size

Avoid tracking unnecessary data

Defer offscreen images

Deprecate GIFs for animated content

Enable text compression

Keep request counts low

Minify web assets

Minimize main thread work

Optimize image size

Remove unused CSS definitions

Serve images in modern formats

Use server-side rendering for high-traffic pages

Tags

Home > Catalog > Web > Use server-side rendering for high-traffic pages

Use server-side rendering for high-traffic pages

Description

Modern web applications feature a lot of interface elements that are stored in templates on the server. The template is downloaded with the application code and then rendered with requested data from the server. This can dramatically increase the SCI in different scenarios: When the client device doesn't have access to green energy, when the page usage is short (e.g. accidental clicks, pages with short dwell time like landing pages) or if a lot of server round-trips are needed to gather data.

Solution

Consider using server-side rendering as a method to reduce energy consumption on the client side. Frameworks like Angular or React already provide solutions on how to implement this efficiently. Server side rendering can improve site loading times (including rendering time on slower hardware), reduce network requirements (less round trips needed), and utilize caching (pre-rendered pages can be stored on the client device).

SCI Impact

$$SCI = (E * I) + M \text{ per } R$$

Software Carbon Intensity Spec

Server side rendering will impact SCI as follows:

- E : Pre-rendering once can dramatically reduce energy consumption; pre-rendering on every request can reduce energy consumption if the server is more energy efficient and has access to green energy

Description

Solution

SCI Impact

Assumptions

Considerations

References





Smells

Environmental Code Smells

Name	Detailed Description
<i>Optimized API</i>	
Fused Location	The fused location provider is one of the location APIs in Google Play services which combines signals from GPS, Wi-Fi, and cell networks, as well as accelerometer, gyroscope, magnetometer and other sensors. It is officially recommended to maximize battery life. Thus, developer has to set up Google Play Service in her gradle file with a dependency to <code>com.google.android.gms:play-services-location:x.y.z</code> , and then to import from <code>com.google.android.gms.location</code> instead of the <code>android.location</code> package of the SDK.
Bluetooth Low-Energy	In contrast to classic Bluetooth, Bluetooth Low Energy (BLE) is designed to provide significantly lower power consumption. Its purpose is to save energy on both paired devices but very few developers are aware of this alternative API. From the Android client side, it means append <code>android.bluetooth.le.*</code> imports to <code>android.bluetooth.*</code> imports in order to benefit from low-energy features.
Lazy Loading	When displaying scrollable data on screen, the new Jetpack Compose API introduced lazy views instead of <code>ListView</code> , <code>GridView</code> and even <code>RecyclerView</code> . These components use the technique of lazy loading, which consists of loading data only when it arrives at the display area. Import <code>androidx.compose.foundation.lazy.*</code> to benefit from objects like <code>LazyColumn</code> , <code>LazyRow</code> , <code>LazyVerticalGrid</code> or <code>LazyHorizontalGrid</code> .
<i>Leakage</i>	
Media Leak	Creation of a Media Recorder object with <code>new MediaRecorder()</code> is used to record audio and video, while creation of a Media Player object with <code>new MediaPlayer()</code> can be used to control playback of audio/video files and streams. Both classes own a <code>release()</code> method. In addition to unnecessary resources (such as memory and instances of codecs) being held, failure to call this method immediately if a media object is no longer needed may also lead to continuous battery consumption for mobile devices.

<https://github.com/cnumr/best-practices-mobile>



Energy consumption in the cloud

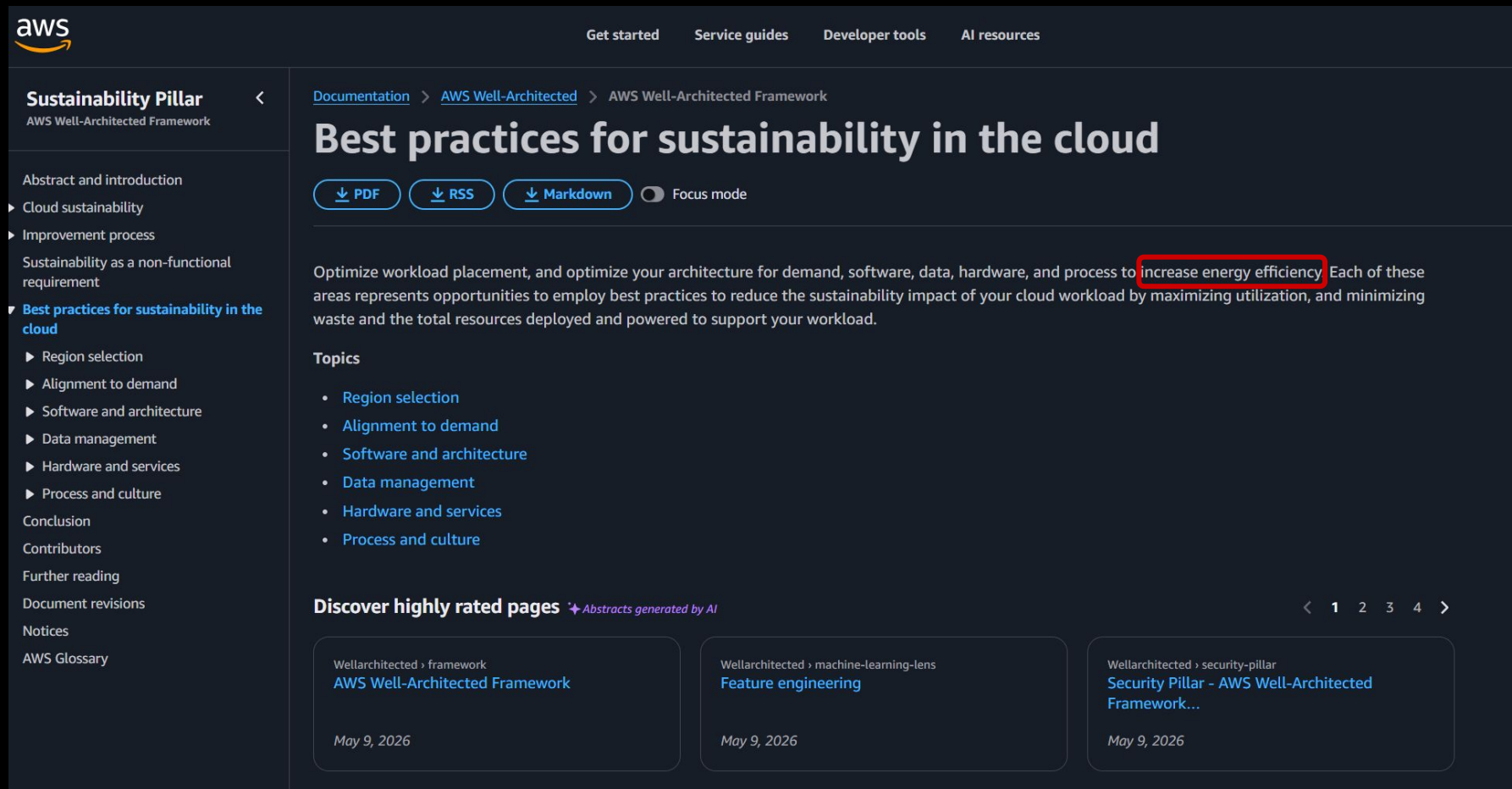
If the software is living in the cloud, the problem is in hard(er) mode



Software energy optimization in the cloud

The screenshot shows the AWS Well-Architected Framework documentation page. The left sidebar contains a navigation menu with the following items: AWS Well-Architected Framework, Abstract and introduction, The pillars of the framework (expanded), Operational excellence, Security, Reliability, Performance efficiency, Cost optimization, Sustainability, The review process, Conclusion, Contributors, Further reading, Document revisions, Appendix: Questions and best practices, Notices, and AWS Glossary. The main content area features the title 'The pillars of the framework' with options to download PDF, RSS, and Markdown, and a Focus mode toggle. The introductory text states: 'Creating a software system is a lot like constructing a building. If the foundation is not solid, structural problems can undermine the integrity and function of the building. When architecting technology solutions, if you neglect the six pillars of operational excellence, security, reliability, performance efficiency, cost optimization, and sustainability, it can become challenging to build a system that delivers on your expectations and requirements. Incorporating these pillars into your architecture will help you produce stable and efficient systems. This will allow you to focus on the other aspects of design, such as functional requirements.' Below this is a section titled 'Pillars' with a bulleted list: Operational excellence, Security, Reliability, Performance efficiency, Cost optimization, and Sustainability.

Software energy optimization in the cloud



The screenshot shows the AWS Well-Architected Framework documentation page for 'Best practices for sustainability in the cloud'. The page is part of the 'Sustainability Pillar' and includes a navigation sidebar, a breadcrumb trail, and a main content area with a table of contents and a list of topics.

Navigation: Get started | Service guides | Developer tools | AI resources

Breadcrumbs: Documentation > AWS Well-Architected > AWS Well-Architected Framework

Best practices for sustainability in the cloud

Optimize workload placement, and optimize your architecture for demand, software, data, hardware, and process to **increase energy efficiency**. Each of these areas represents opportunities to employ best practices to reduce the sustainability impact of your cloud workload by maximizing utilization, and minimizing waste and the total resources deployed and powered to support your workload.

Topics:

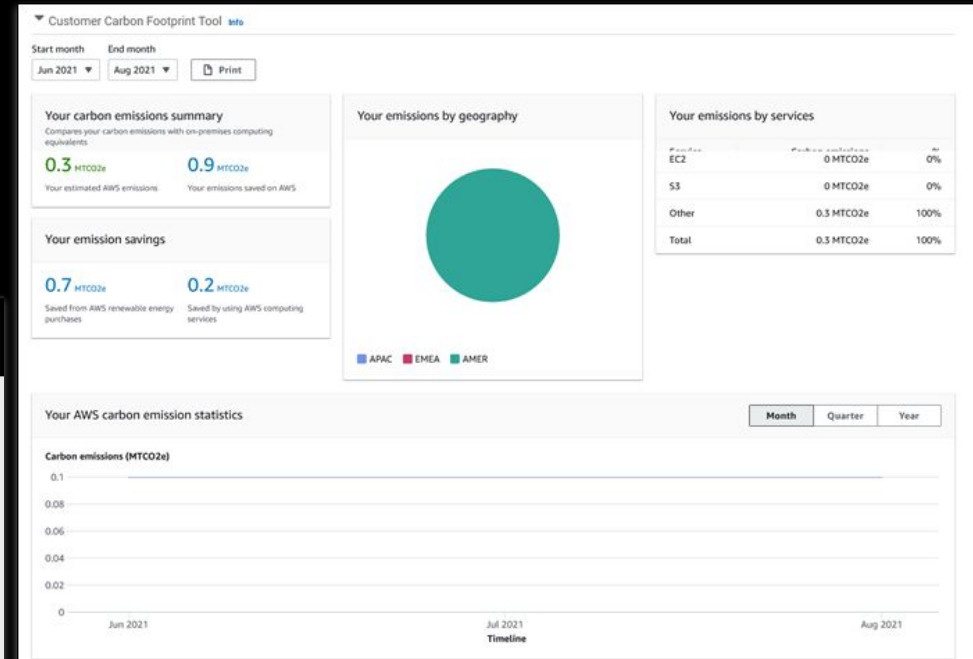
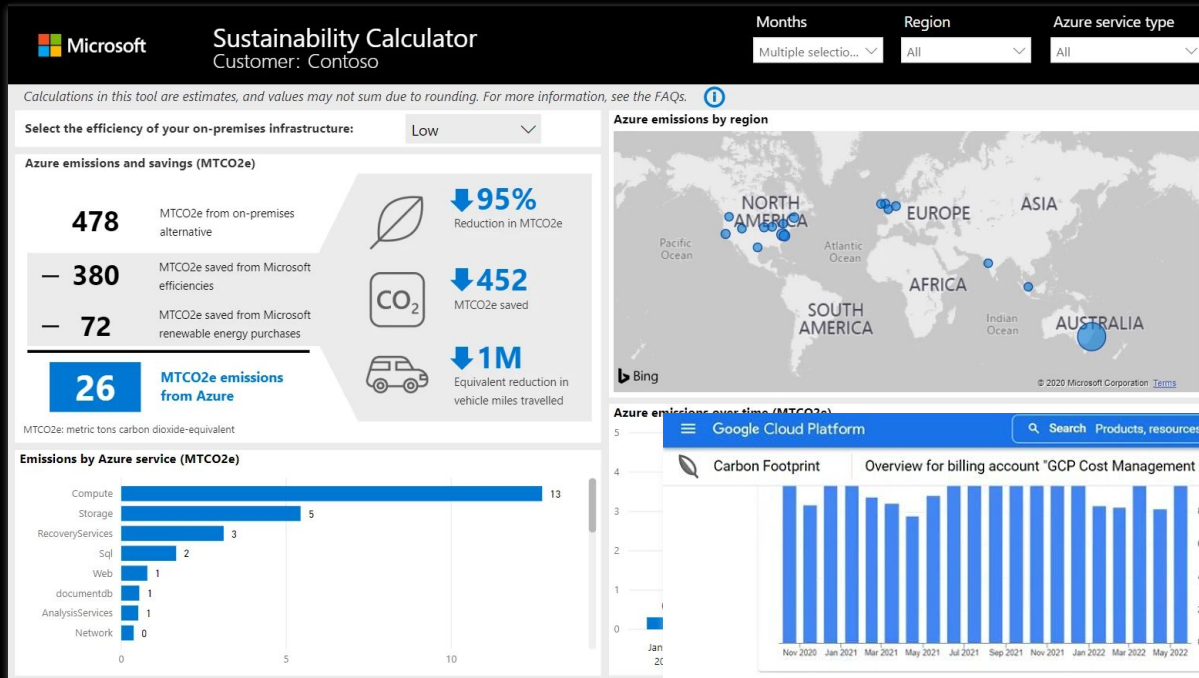
- Region selection
- Alignment to demand
- Software and architecture
- Data management
- Hardware and services
- Process and culture

Discover highly rated pages * Abstracts generated by AI

Wellarchitected > framework	Wellarchitected > machine-learning-lens	Wellarchitected > security-pillar
AWS Well-Architected Framework	Feature engineering	Security Pillar - AWS Well-Architected Framework...
May 9, 2026	May 9, 2026	May 9, 2026



Carbon dashboards

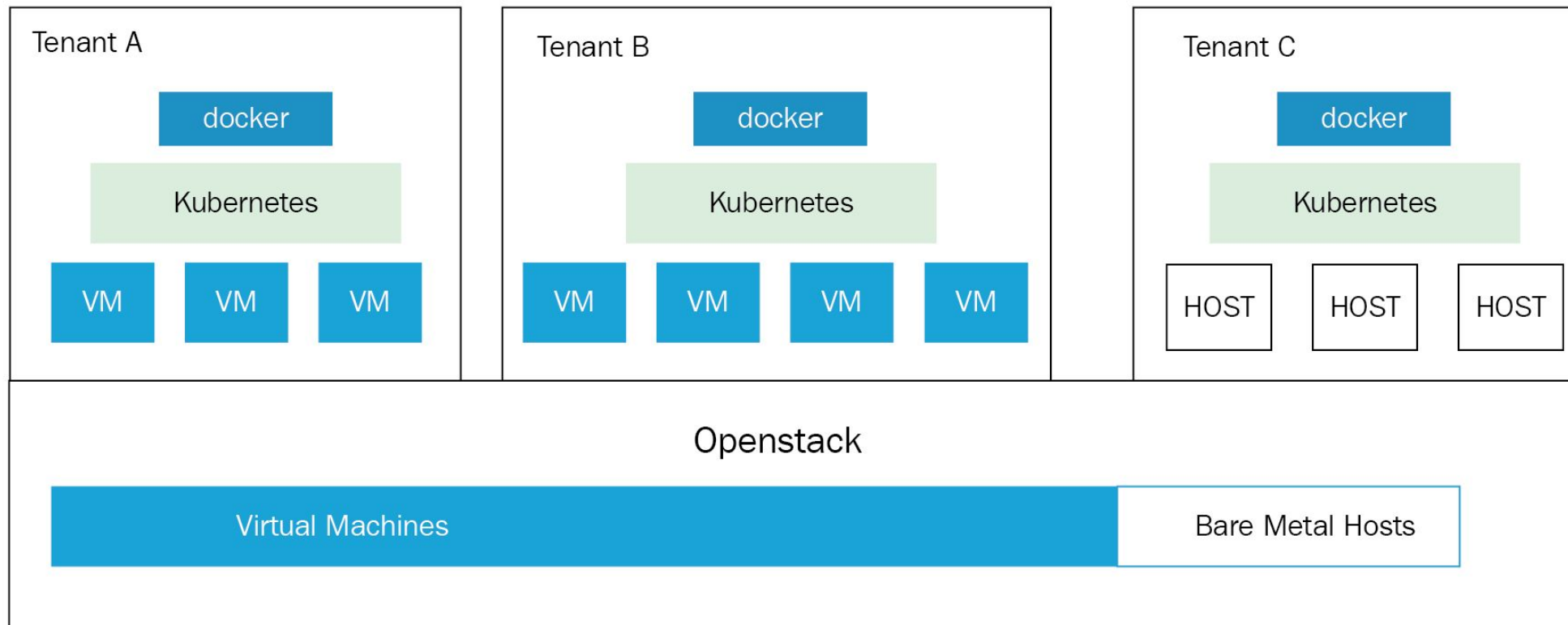
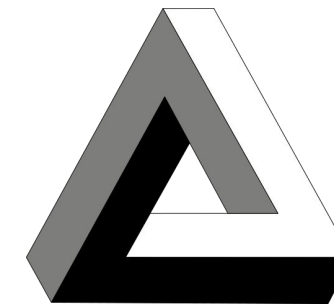


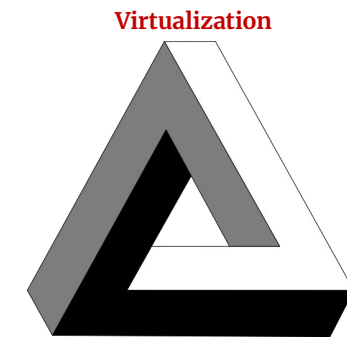


The Devil's triangle in the cloud

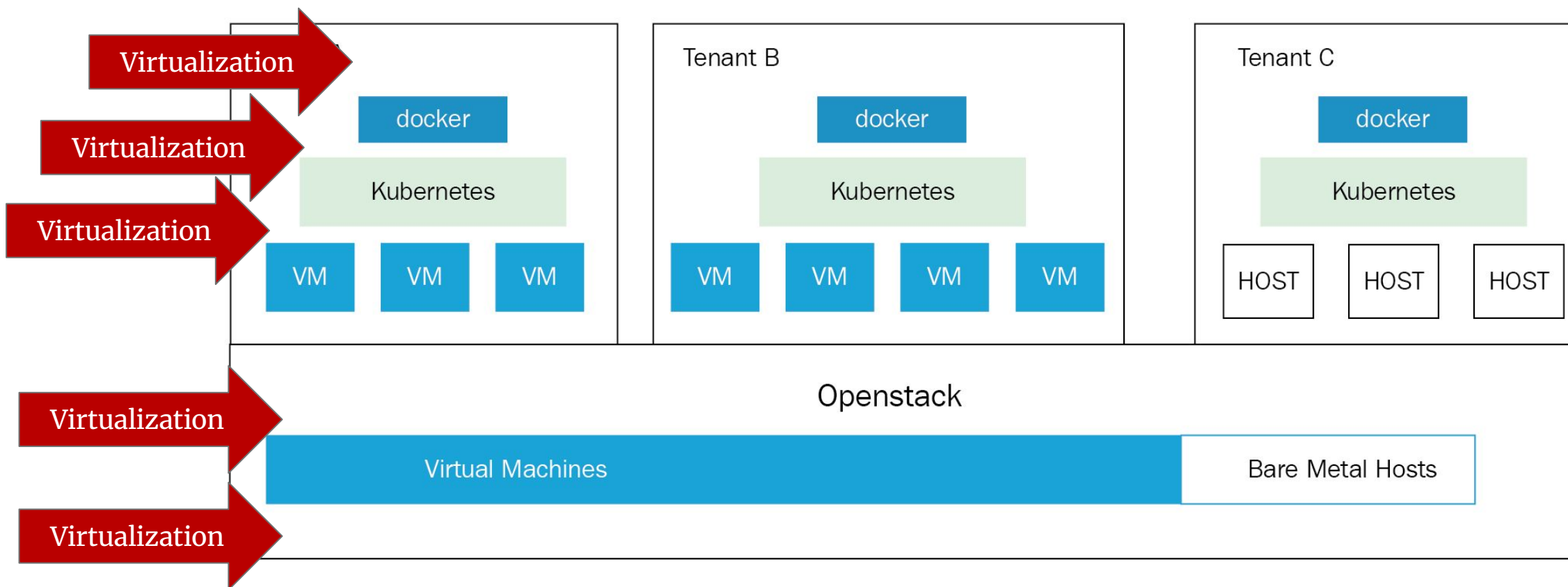


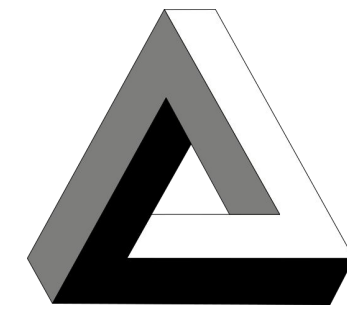
Virtualization layers





Virtualization layers





Monitoring energy in a cloud-native application

<https://doi.org/10.1145/3676151.3719371>

Application

OpenTelemetry (demo app)

Kubernetes

cAdvisor + Kepler

Virtual Machine

Scaphandre

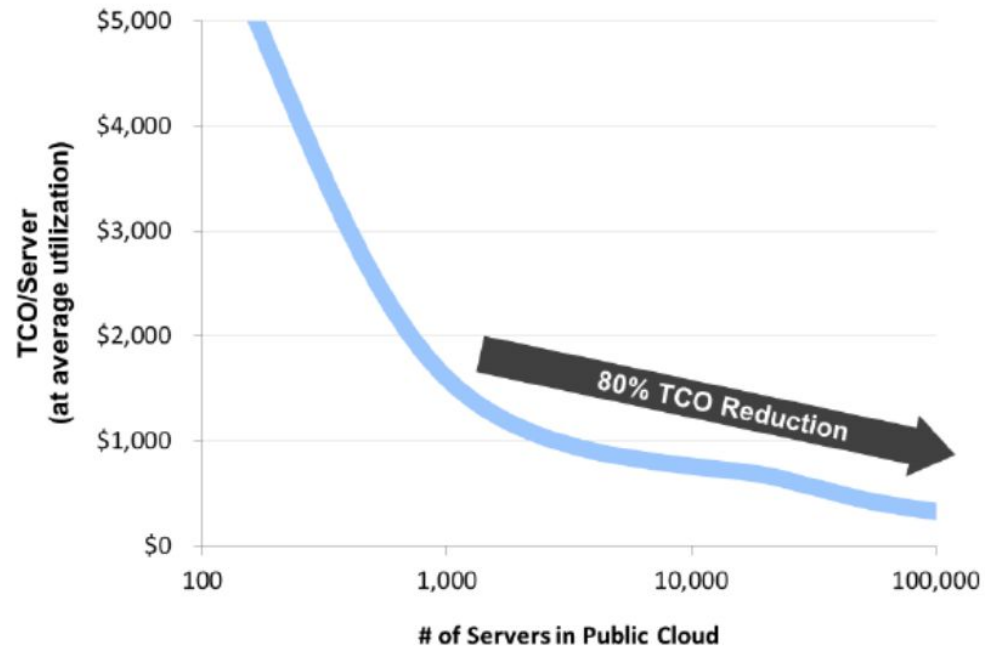
Bare metal

node_exporter + RAPL

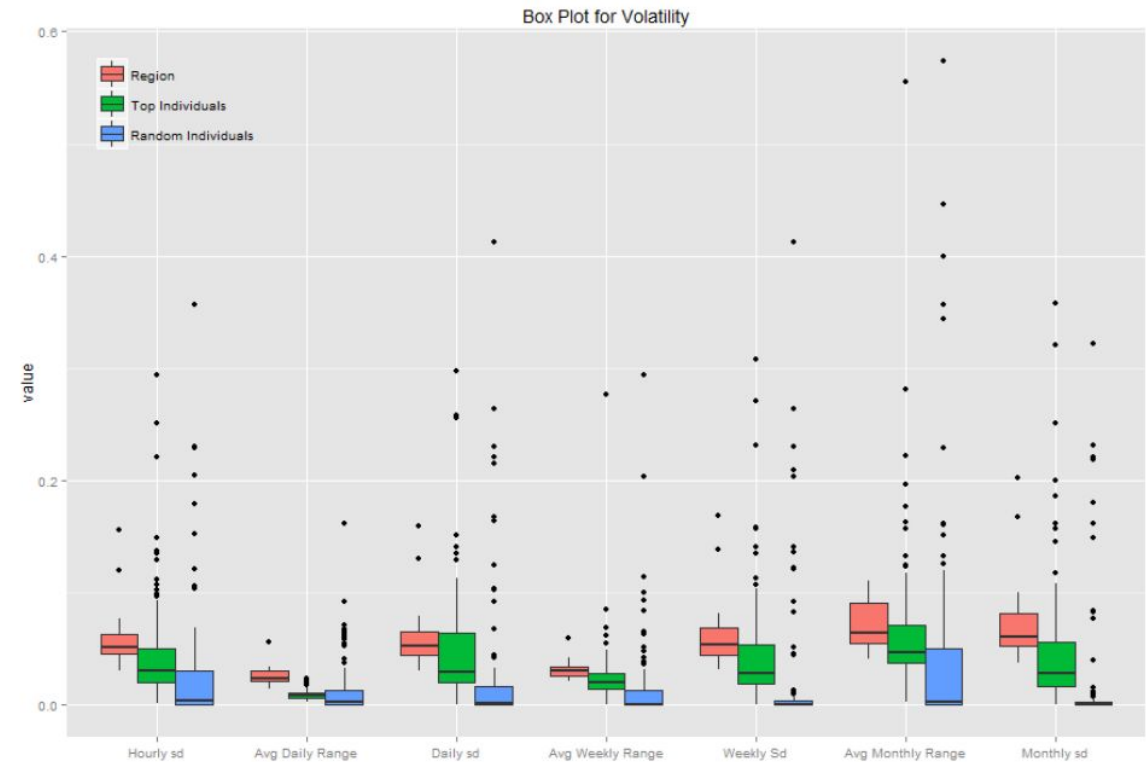
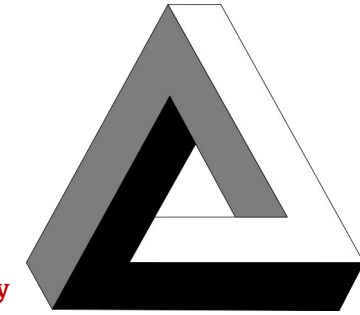
Physical

PowerPlug

Economies of Scale



Multitenancy



<https://doi.org/10.1145/3038912.3052707>

Everything as a Service




Servitization

Featured | AI + machine learning | App development | Compute | Databases + analytics | Hybrid + multcloud

Cloud databases

Find the right databases for your needs—including relational, NoSQL, and caching.


[Explore all databases](#)



Azure Arc

Unify on-premises, hybrid, and cross-cloud infrastructure.


[Explore the product](#)



Azure Copilot

Get help to design, operate, optimize, and troubleshoot apps and infrastructure from cloud to edge.


[Explore the product](#)



Azure Migrate

Simplify and accelerate cloud transformation with the new Azure Copilot migration agent.


[Explore the product](#)



GitHub Copilot

Increase software development velocity and inspire continuous innovation.


[Explore the product](#)



Azure Local

Build apps across on-premises, cloud, and the edge.

[Explore the product](#)



Azure Database for PostgreSQL

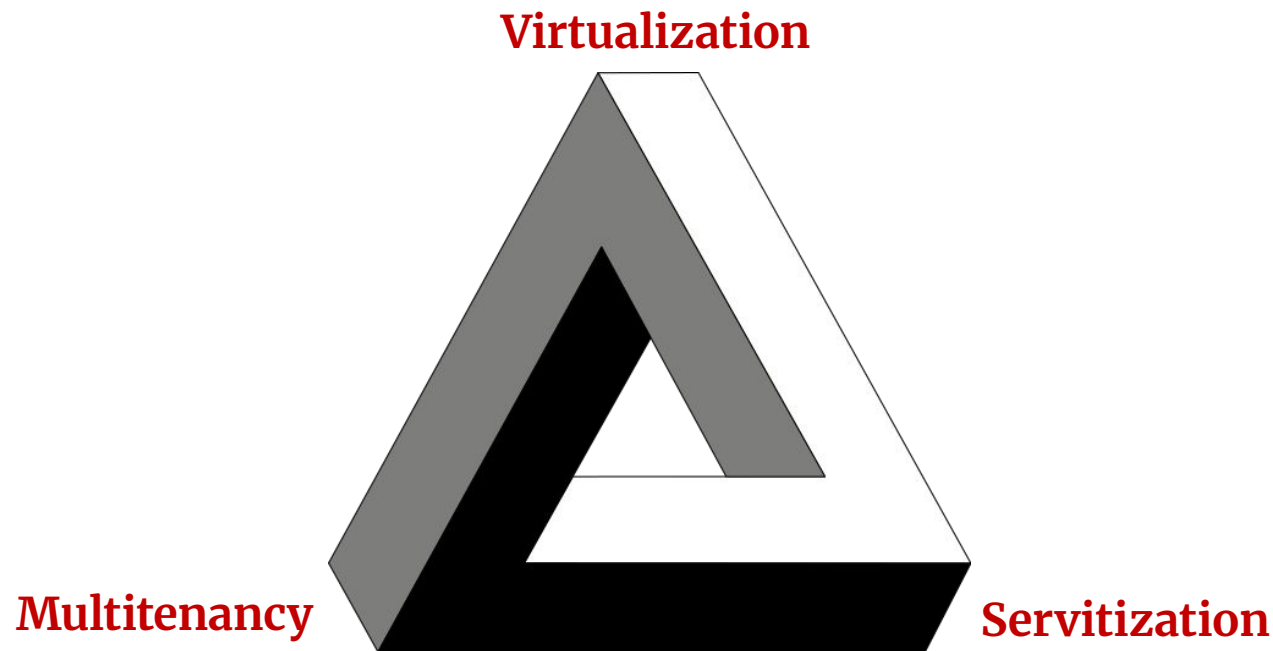
Innovate with a fully managed, AI-ready PostgreSQL database.

[Explore the product](#)

[> See all products \(200+\)](#)

Implications of the **D**evil's triangle

- ⇒ Existing tooling fails in at least one of three points
- ⇒ Need to treat the system as a (semi-)black box





Observability of energy consumption

Making sense using available data



Observability

Control theory: *a measure of how well internal states of a system can be inferred from knowledge of its external outputs (Kalman, 1960)*

$$\mathcal{O} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}$$



Observability

Control theory: *a measure of how well internal states of a system can be inferred from knowledge of its external outputs*

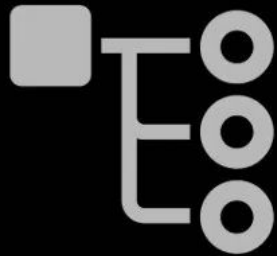
$$\mathcal{O} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}$$

Software systems: *how well a system's state can be understood from the obtained telemetry*

Pillars of observability



Metrics

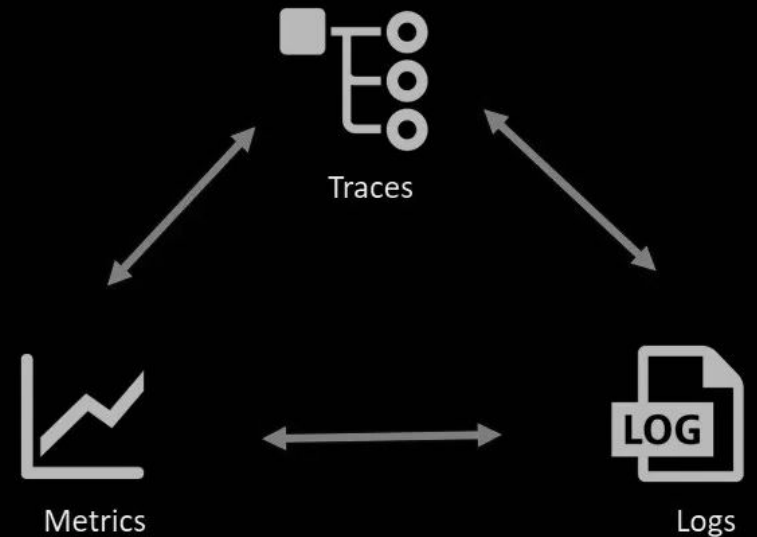


Traces



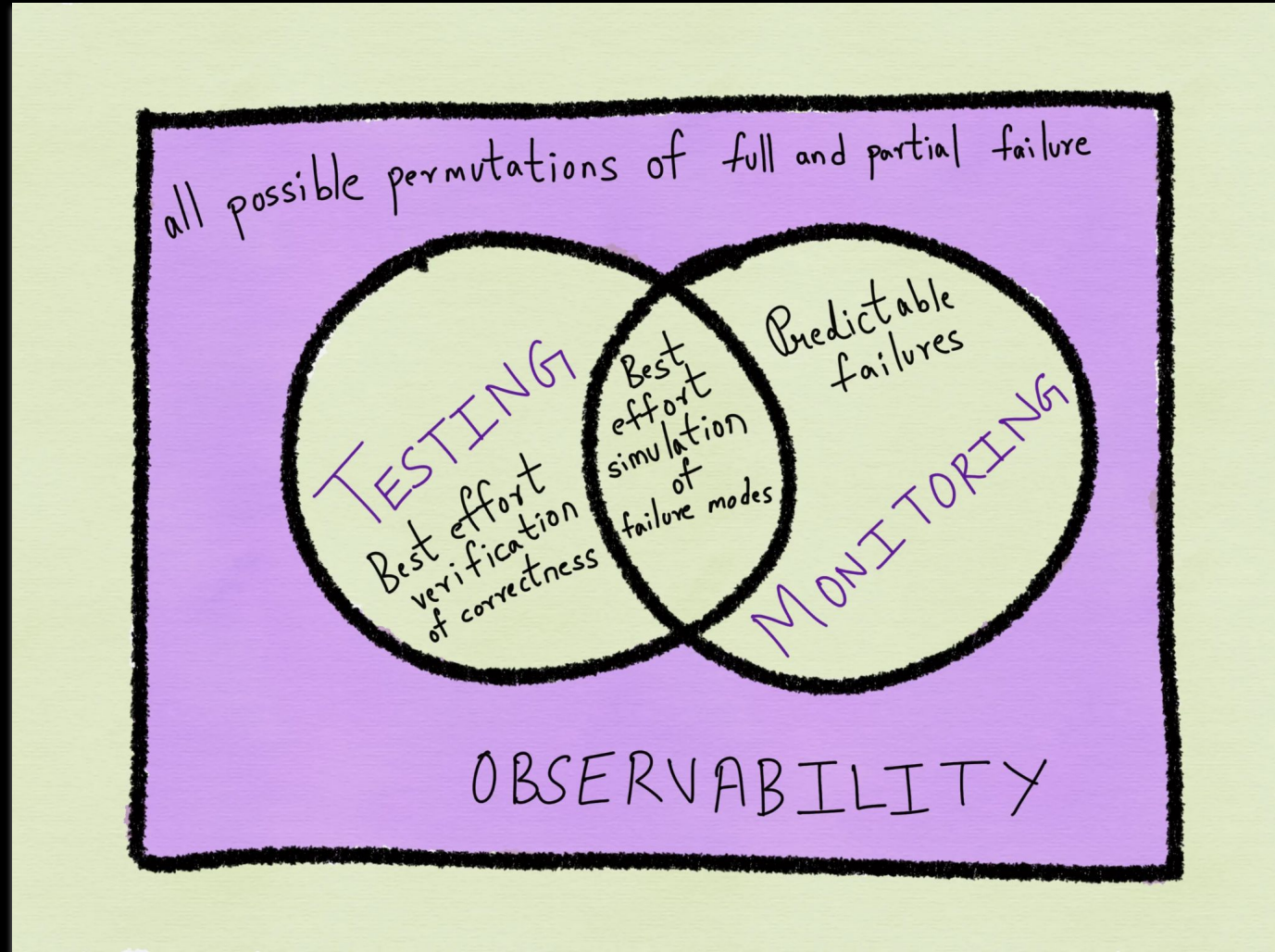
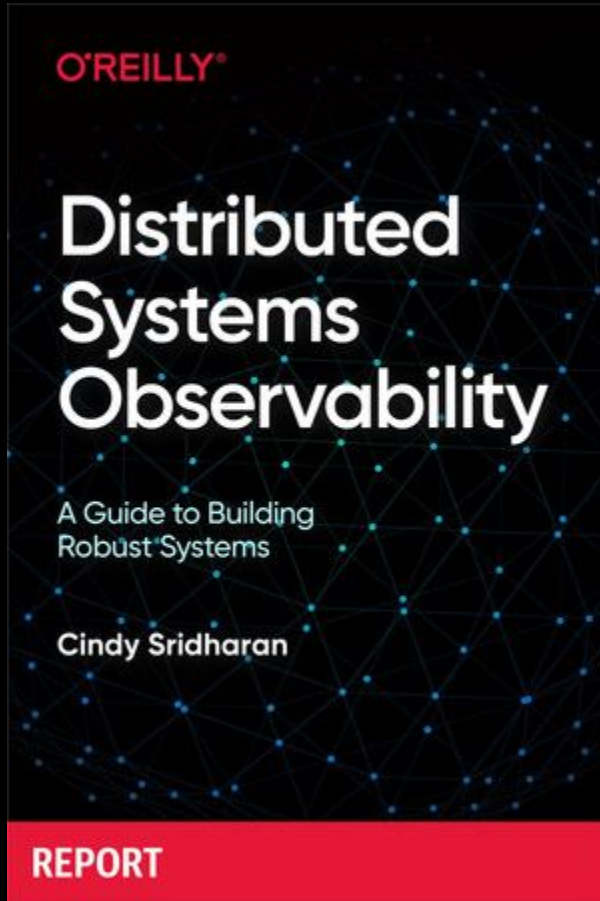
Logs

Three data silo



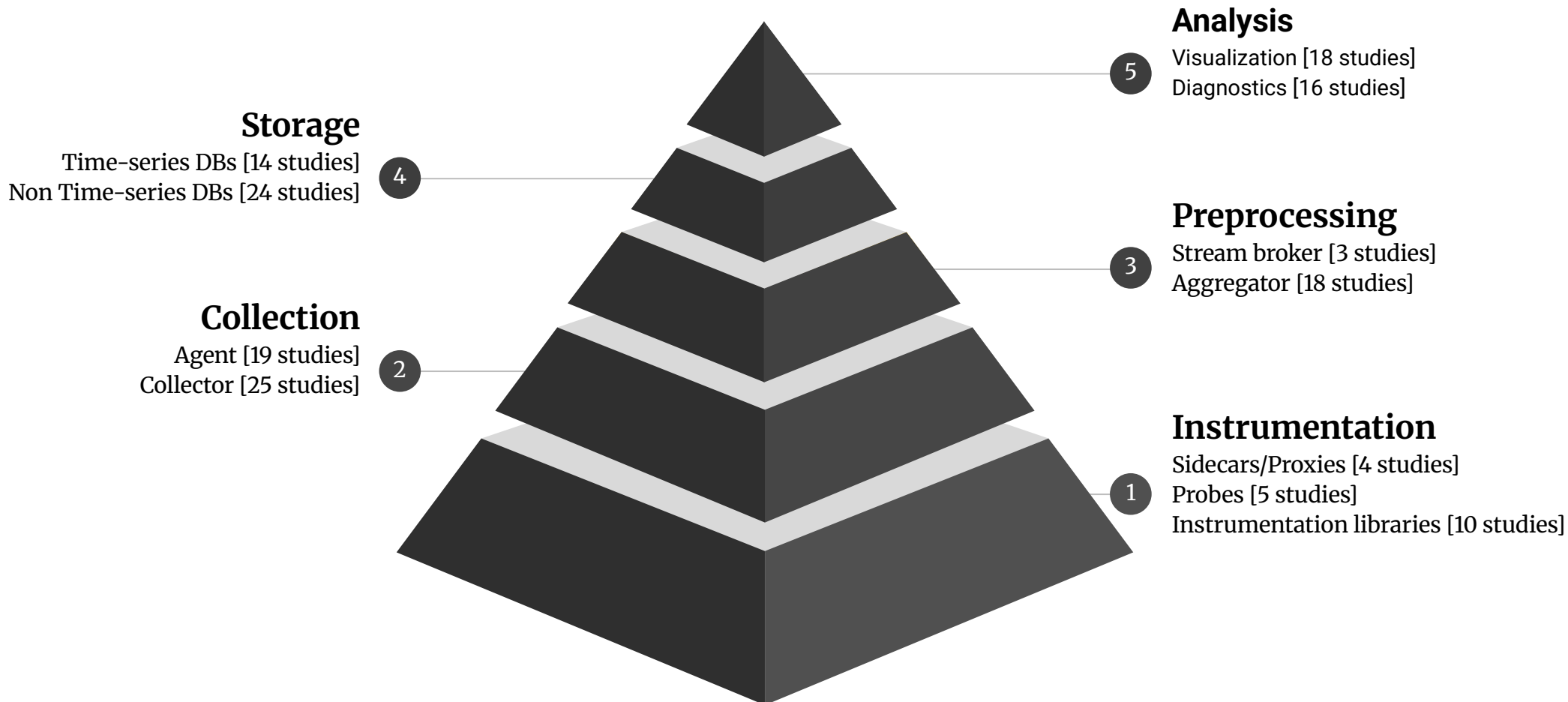
Three pillars of observability, in context

Source:  dynatrace

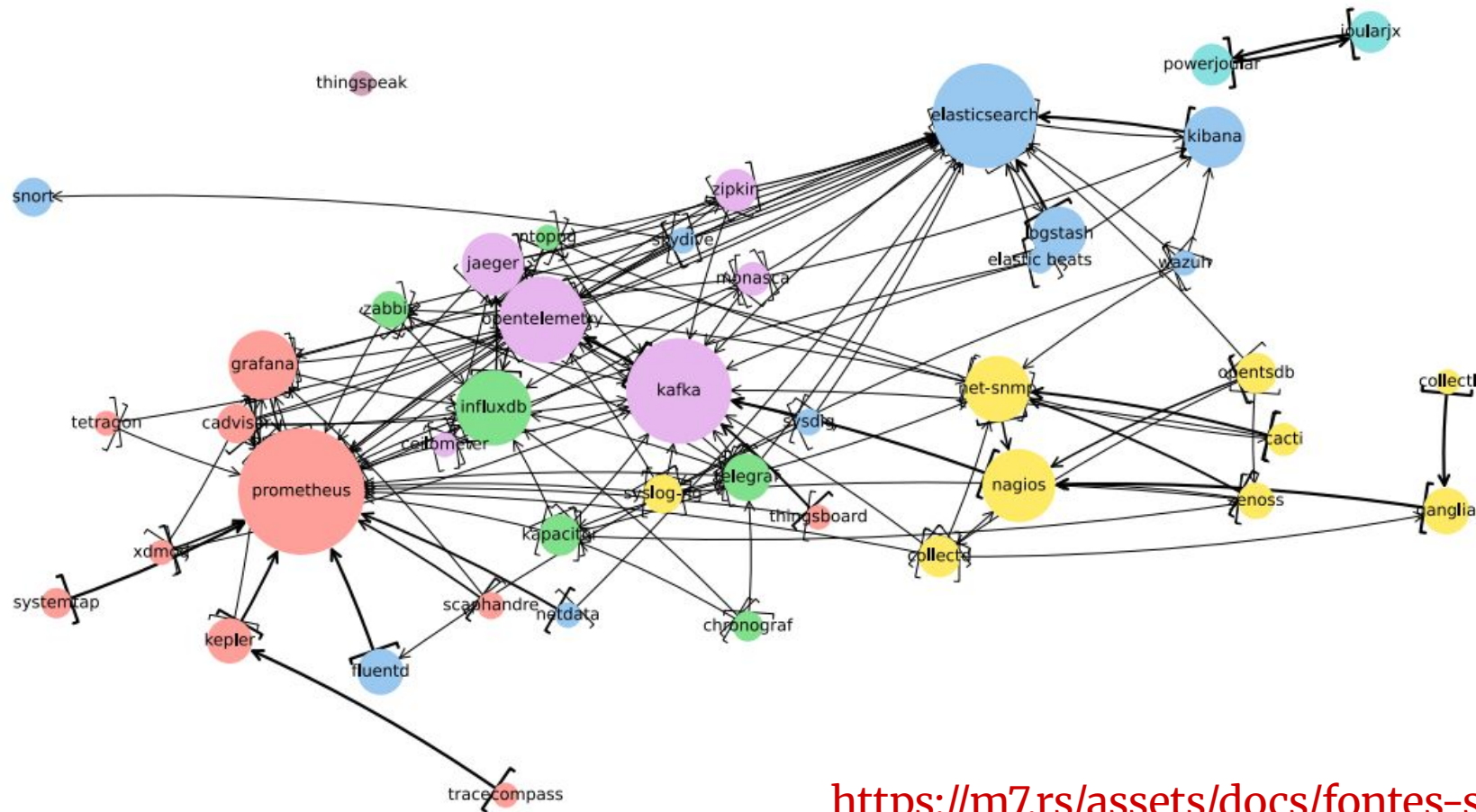




Observability solutions: literature landscape



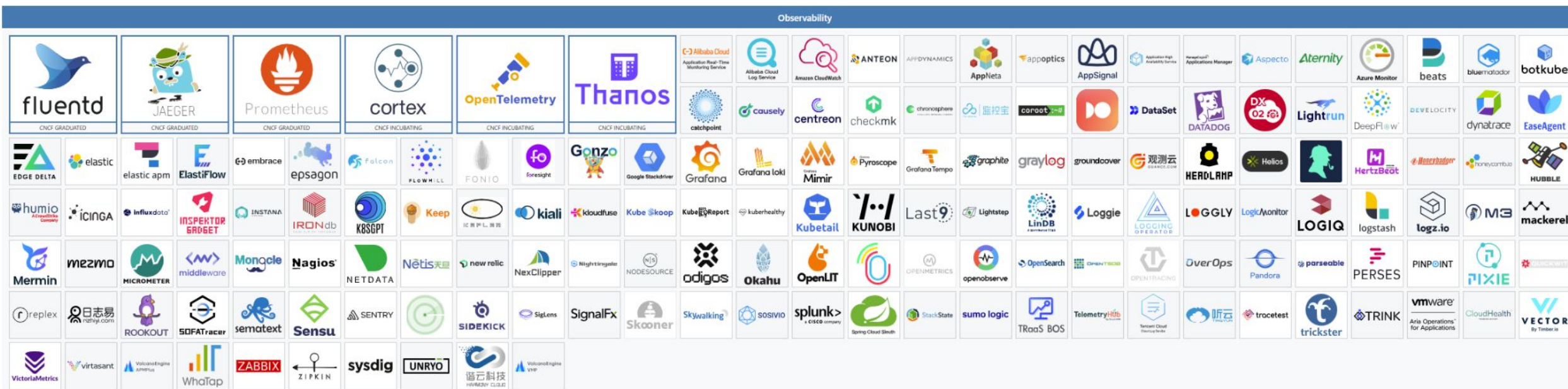
Observability solutions: open source landscape



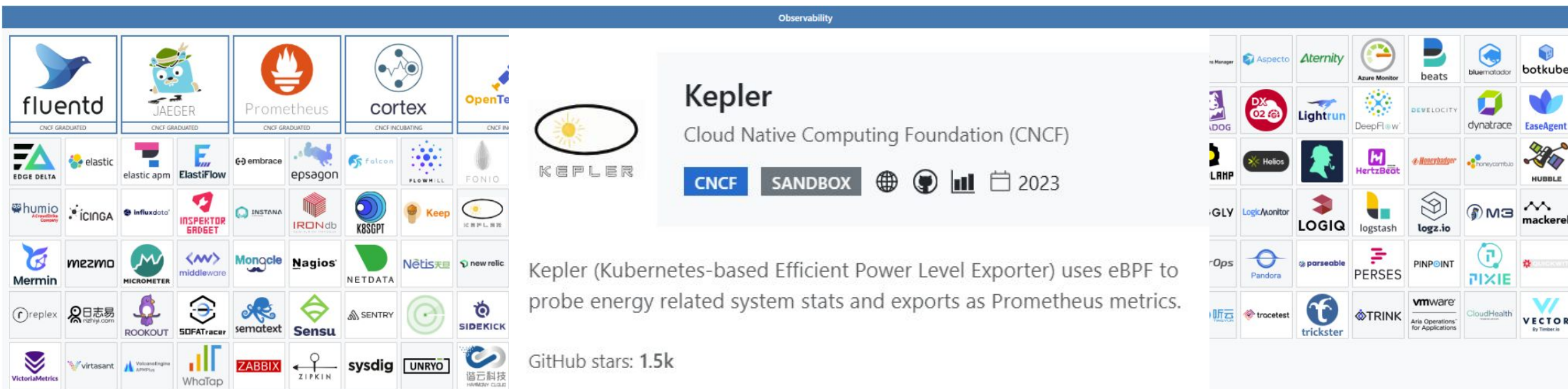
<https://m7.rs/assets/docs/fontes-sesos-2026.pdf>



Observability solutions: CNCF landscape



Observability solutions: CNCF's Kepler



The screenshot shows the CNCF Observability marketplace. On the left, a grid of tool cards is displayed, categorized by status: CNCF GRADUATED, CNCF GRADUATING, CNCF INCUBATING, and CNCF INCUBATING. Tools include fluentd, Prometheus, cortex, OpenTelemetry, elastic, ElasticFlow, epsagon, Falcon, FLOWMILL, FONIO, humio, iCINGA, influxdata, INSPEKTOR GRABET, INSTANA, IRONdb, K8SGPT, Keep, Mermin, mezmio, MICROMETER, middleware, Mongocle, Nagios, NETDATA, Nētis, new relic, replex, 日志易, ROCKOUT, SDFATracer, sematext, Sensu, SENTRY, SIDEKICK, VictoriaMetrics, virtasant, Whatap, ZABBIX, ZIPKIN, sysdig, UNRYO, and 行云科技.

The main focus is on the Kepler tool card, which features the Kepler logo (a sun in an oval) and the text:

 Kepler

 Cloud Native Computing Foundation (CNCF)

 Labels: CNCF, SANDBOX, 2023

 Description: Kepler (Kubernetes-based Efficient Power Level Exporter) uses eBPF to probe energy related system stats and exports as Prometheus metrics.

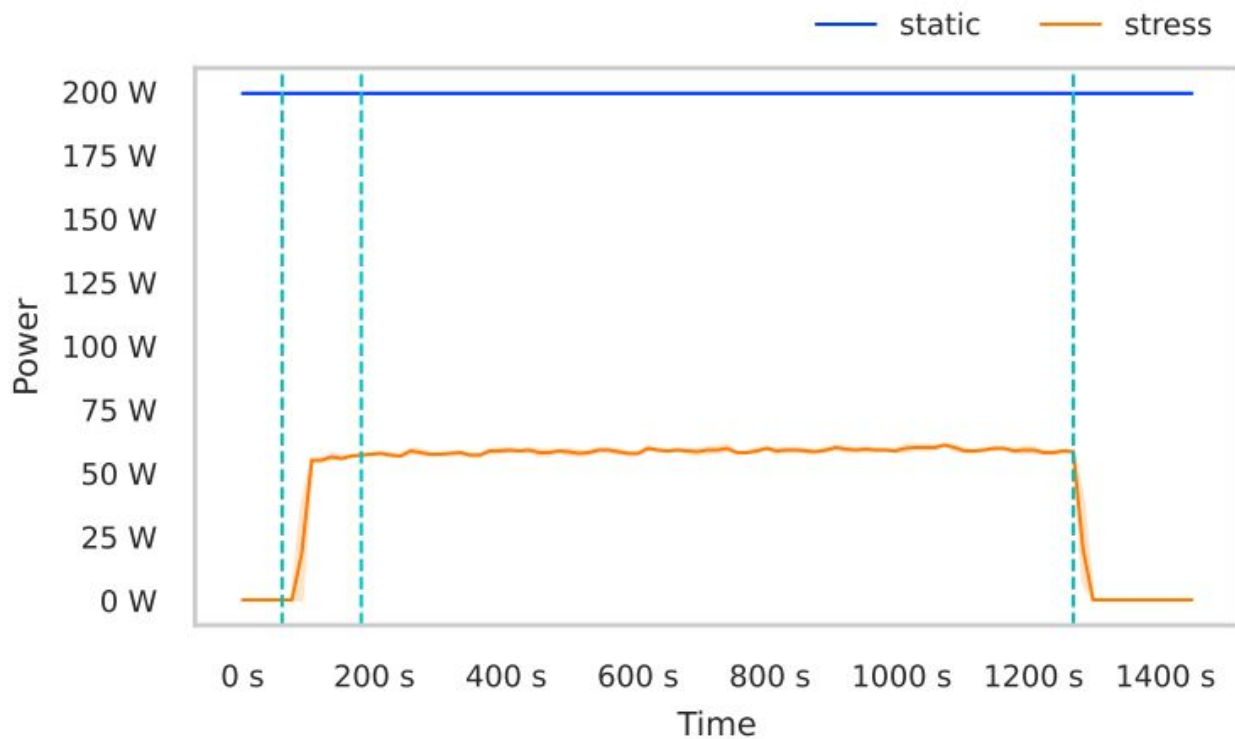
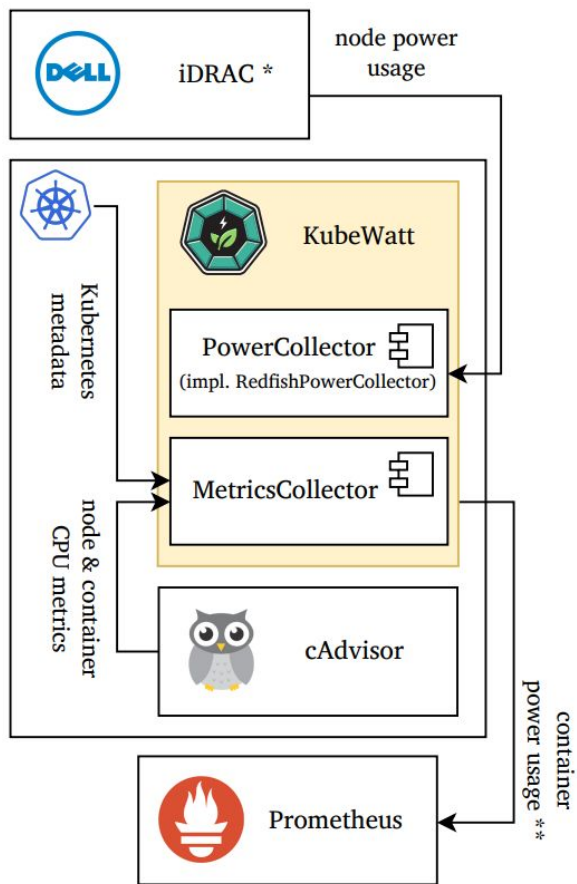
 GitHub stars: 1.5k

On the right side of the marketplace, a grid of other observability tools is visible, including Aspecto, Aternity, Azure Monitor, beats, blumatorator, botkube, Lighthouse, Lightrun, DeepFlow, BEVELOCITY, dynatrace, EaseAgent, LAMP, Helios, HertzBeet, Honeycomb, HUBBLE, LOGGLY, LogicMonitor, LOGIQ, logstash, logz.io, mackerel, Ops, Pandora, parseable, PERSES, PINPOINT, PIXIE, trocetest, trickster, TRINK, VMware Aria Operations for Applications, CloudHealth, and VECTOR.

<https://sustainable-computing.io/>

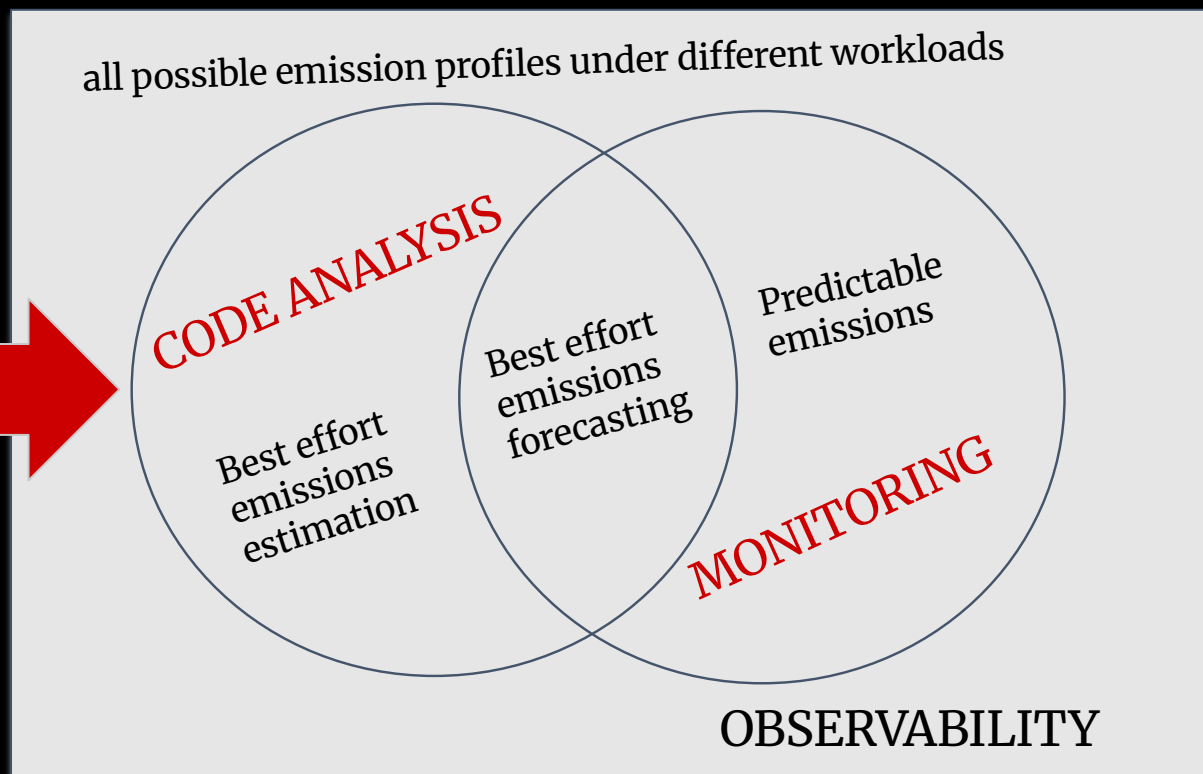
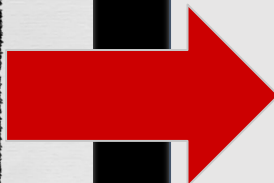
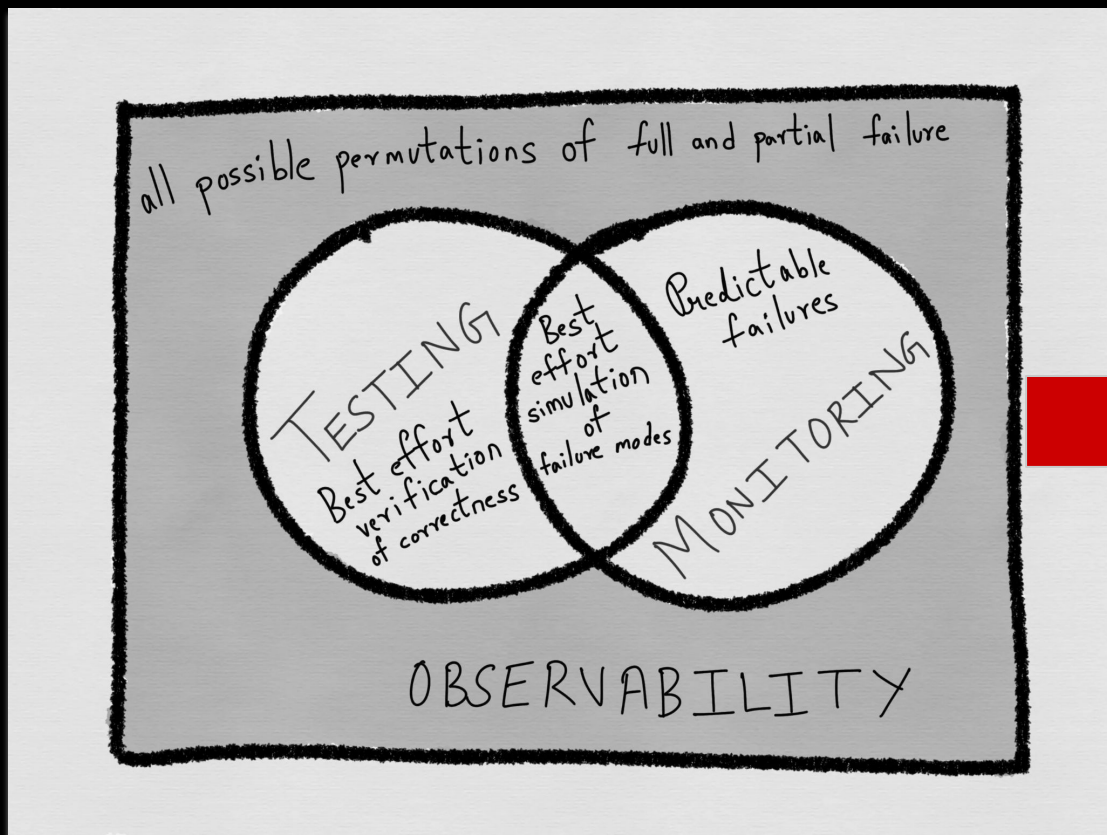


KubeWatt



<https://doi.org/10.1109/ICT4S68164.2025.00016>

Energy & carbon observability





Beyond observability

Is observability *enough* to understand how to reduce software-generated emissions?



Beyond observability

Is observability enough to understand how to reduce software-generated emissions?

Does establishing observability solve reliability problems?



Beyond observability

Is observability enough to understand how to reduce software-generated emissions?

Does establishing observability solve reliability problems?

Do we need something *more* than simply understanding system state?



Actionable Observability

*To provide tangible outcomes we need to connect what we observe
with what we build*



Actionable Observability \mathcal{O}^+

How well the need for specific interventions on an existing software system can be inferred from obtained telemetry



Actionable Observability

- ⇒ Dealing with *engineered* and not *natural* systems
- ⇒ Closing the loop between *intended* and *observed* behavior
- ⇒ Bridging *design-* and *run-time* to avoid further debt

see also <https://doi.org/10.1145/3643657.3643909>



Actionable Observability roadmap

Signals

What we
observe

Elements

Where we
need to
intervene

Knowledge

How we
intervene

Action

How we use
it in practice



Actionable Observability roadmap

Signals

Elements

Knowledge

Action

- ⇒ Define *correct* granularity of information
- ⇒ Connect signals to KPIs/NFRs/QAs
- ⇒ Avoid data overload



Data overload

Data availability paradox:
too much data leading to
paralysis

Can We Ever Escape from Data Overload? A Cognitive Systems Diagnosis

D. D. Woods¹, E. S. Patterson¹ and E. M. Roth²

¹Institute for Ergonomics, Ohio State University, Columbus, Ohio, USA; ²Roth Cognitive Engineering, Inc., Brookline, Massachusetts, USA

Abstract: Data overload is a generic and tremendously difficult problem that has only grown with each new wave of technological capabilities. As a generic and persistent problem, three observations are in need of explanation: Why is data overload so difficult to address? Why has each wave of technology exacerbated, rather than resolved, data overload? How are people, as adaptive responsible agents in context, able to cope with the challenge of data overload? In this paper, first we examine three different characterisations that have been offered to capture the nature of the data overload problem and how they lead to different proposed solutions. As a result, we propose that (a) data overload is difficult because of the context sensitivity problem – meaning lies, not in data, but in relationships of data to interests and expectations and (b) new waves of technology exacerbate data overload when they ignore or try to finesse context sensitivity. The paper then summarises the mechanisms of human perception and cognition that enable people to focus on the relevant subset of the available data despite the fact that what is interesting depends on context. By focusing attention on the root issues that make data overload a difficult problem and on people's fundamental competence, we have identified a set of constraints that all potential solutions must meet. Notable among these constraints is the idea that organisation precedes selectivity. These constraints point toward regions of the solution space that have been little explored. In order to place data in context, designers need to display data in a conceptual space that depicts the relationships, events and contrasts that are informative in a field of practice.

Keywords: Agent; Alarm; Context; Data overload; Information Visualisation; Workload

1. DATA OVERLOAD IS A GENERIC, DIFFICULT PROBLEM

Information is not a scarce resource. Attention is. (Herbert Simon)

Each round of technical advances, whether in artificial intelligence, computer graphics or electronic connectivity, promises to help people better understand and manage a whole host of activities, from financial analysis to monitoring data from space missions to controlling the national air space. Certainly, this ubiquitous computerisation of the modern world has tremendously advanced our ability to collect, transmit and transform data, producing unprecedented levels of access to data.

However, our ability to interpret this avalanche of data,

i.e., to extract meaning from artificial fields of data, has expanded much more slowly, if at all. In studies across multiple settings, we find that practitioners are bombarded with computer-processed data, especially when anomalies occur. We find users lost in massive networks of computer-based displays, options and modes. For example, one can find a version of the following statement in most accident investigation reports:

although all of the necessary data was physically available, it was not operationally effective. No one could assemble the separate bits of data to see what was going on. (Joyce and Lapinski 1983)

The challenge has become finding what is informative given our interests and needs in a very large field of available data.



Observability

Observability as the desired
property *beyond* data
availability

Can We Ever Escape from Data Overload?

4. *Observability is more than mere data availability.*

The greatest value of a picture is when it *forces* us to notice what we never expected to see. (Tukey 1977, p. vi)

There are significant differences between the available data and the meaning or information that a person extracts from that data. *Observability* is the technical term that refers to the cognitive work needed to extract meaning from available data (Rasmussen 1985). This term captures the relationship among data, observer and context of observation that is fundamental to effective feedback.

Observability is distinct from data availability, which refers to the mere presence of data in some form in some location. For human perception, 'it is not sufficient to have something in front of your eyes to see it' (O'Regan 1992, p. 475). Recent studies on visual perception emphasise the role of attention in the perception of suprathreshold stimuli (Resnick et al 1997; Simons and Levin 1997) – 'there seems to be no conscious perception without attention' (Mack and Rock 1998, p. ix).

Data overload

⇒ Context-sensitive treatment
 of data collection

- Fundamental models as *conceptual spaces*
- Artificial intelligence to support observers

Can We Ever Escape from Data Overload? A Cognitive Systems Diagnosis

D. D. Woods¹, E. S. Patterson¹ and E. M. Roth²

¹Institute for Ergonomics, Ohio State University, Columbus, Ohio, USA; ²Roth Cognitive Engineering, Inc., Brookline, Massachusetts, USA

Abstract: Data overload is a generic and tremendously difficult problem that has only grown with each new wave of technological capabilities. As a generic and persistent problem, three observations are in need of explanation: Why is data overload so difficult to address? Why has each wave of technology exacerbated, rather than resolved, data overload? How are people, as adaptive responsible agents in context, able to cope with the challenge of data overload? In this paper, first we examine three different characterisations that have been offered to capture the nature of the data overload problem and how they lead to different proposed solutions. As a result, we propose that (a) data overload is difficult because of the context sensitivity problem – meaning lies, not in data, but in relationships of data to interests and expectations and (b) new waves of technology exacerbate data overload when they ignore or try to finesse context sensitivity. The paper then summarises the mechanisms of human perception and cognition that enable people to focus on the relevant subset of the available data despite the fact that what is interesting depends on context. By focusing attention on the root issues that make data overload a difficult problem and on people's fundamental competence, we have identified a set of constraints that all potential solutions must meet. Notable among these constraints is the idea that organisation precedes selectivity. These constraints point toward regions of the solution space that have been little explored. In order to place data in context, designers need to display data in a conceptual space that depicts the relationships, events and contrasts that are informative in a field of practice.

Keywords: Agent; Alarm; Context; Data overload; Information Visualisation; Workload

1. DATA OVERLOAD IS A GENERIC, DIFFICULT PROBLEM

Information is not a scarce resource. Attention is. (Herbert Simon)

Each round of technical advances, whether in artificial intelligence, computer graphics or electronic connectivity, promises to help people better understand and manage a whole host of activities, from financial analysis to monitoring data from space missions to controlling the national air space. Certainly, this ubiquitous computerisation of the modern world has tremendously advanced our ability to collect, transmit and transform data, producing unprecedented levels of access to data.

However, our ability to interpret this avalanche of data,

i.e., to extract meaning from artificial fields of data, has expanded much more slowly, if at all. In studies across multiple settings, we find that practitioners are bombarded with computer-processed data, especially when anomalies occur. We find users lost in massive networks of computer-based displays, options and modes. For example, one can find a version of the following statement in most accident investigation reports:

although all of the necessary data was physically available, it was not operationally effective. No one could assemble the separate bits of data to see what was going on. (Joyce and Lapinski 1983)

The challenge has become finding what is informative given our interests and needs in a very large field of available data.

Actionable Observability roadmap

Signals

Elements

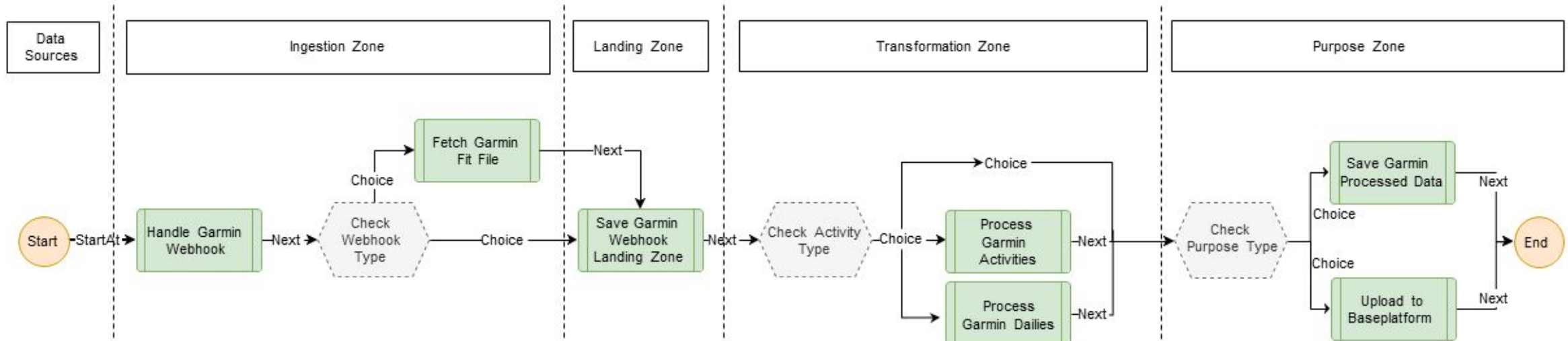
Knowledge

Action

- ⇒ Identify pertinent architectural elements at *right* level of abstraction
- ⇒ Attribute observed information to elements
- ⇒ Identify hotspots as optimization/refactoring targets



The Researchable case study

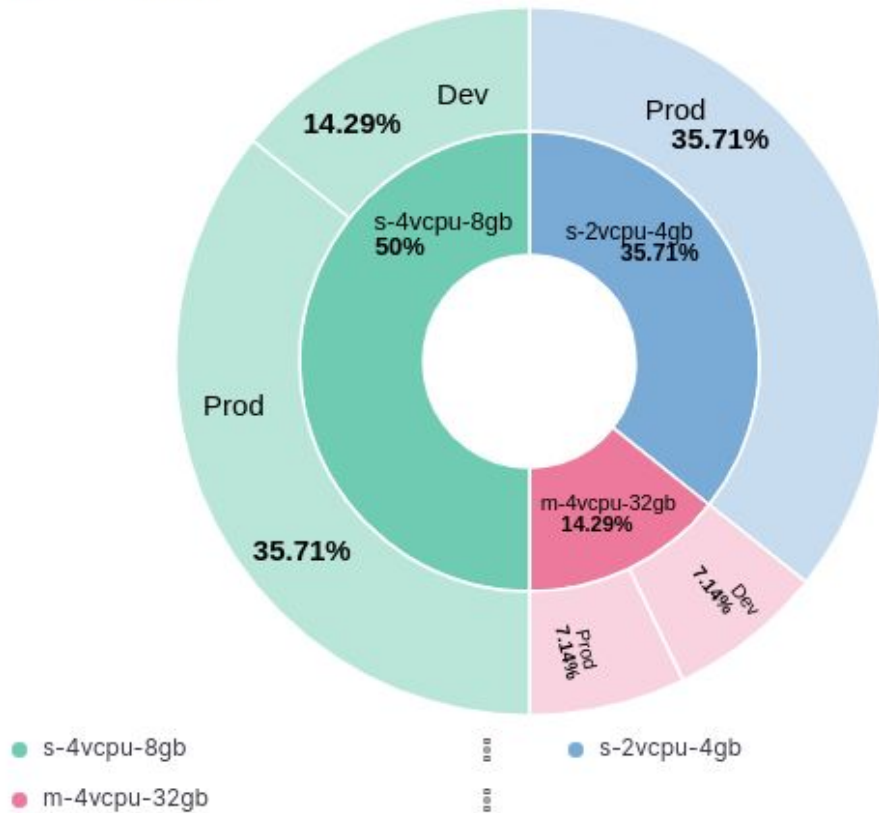


<https://doi.org/10.3390/fi15010037>

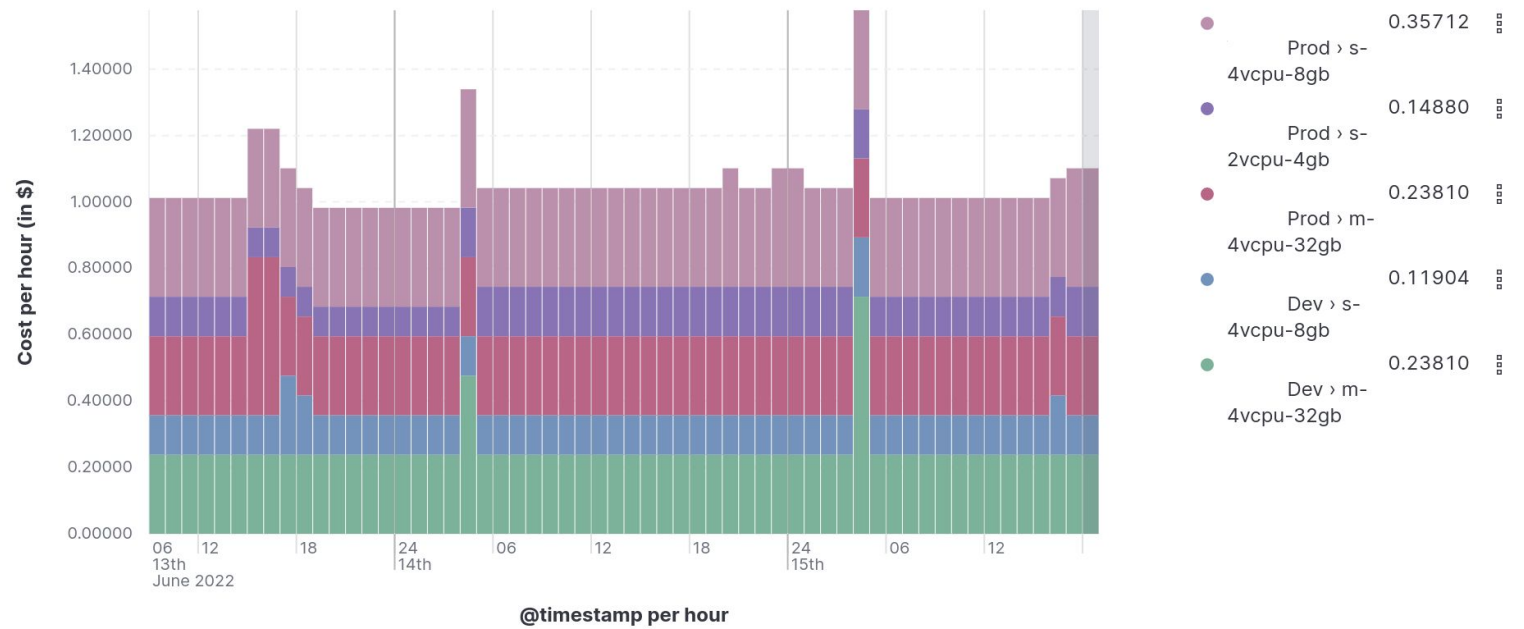


Cost observability dashboard

Instance distribution



Hourly price breakdown

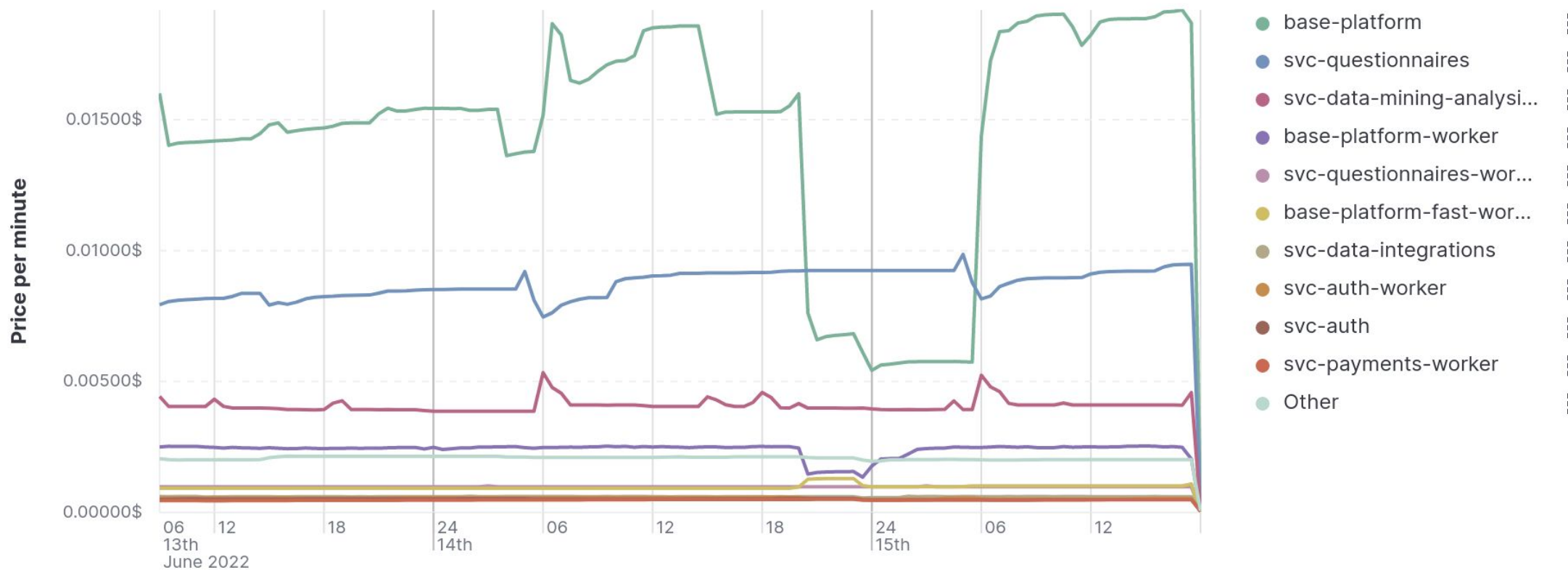


<https://doi.org/10.3390/fi15010037>



Cost allocation per microservice

Cost per deployment



@timestamp per 30 minutes

<https://doi.org/10.3390/fi15010037>



Actionable Observability roadmap

Signals

Elements

Knowledge

Action

- ⇒ Decide which best practices/tactics/patterns to apply
- ⇒ Decide when, where, and how to apply them
- ⇒ Deal with heterogeneity of infrastructure over time and space

Actionable Observability roadmap

Signals

Elements

Knowledge

Action

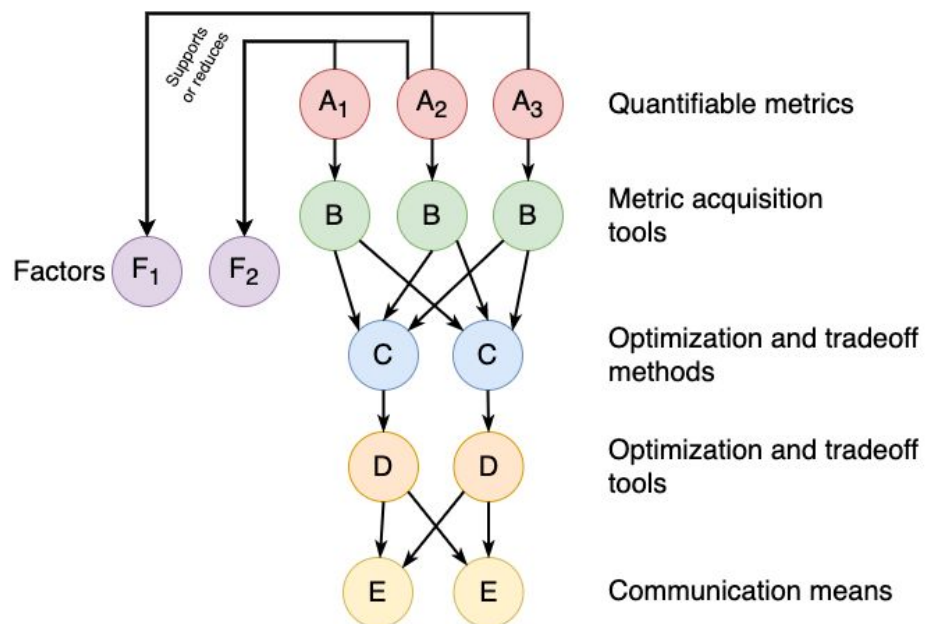
- ⇒ Rationalize & automate refactoring
- ⇒ Integrate into development pipelines & practices
- ⇒ Manage tradeoffs between affected QAs
- ⇒ Enable sustainability across dimensions



MAST ITEA4 project

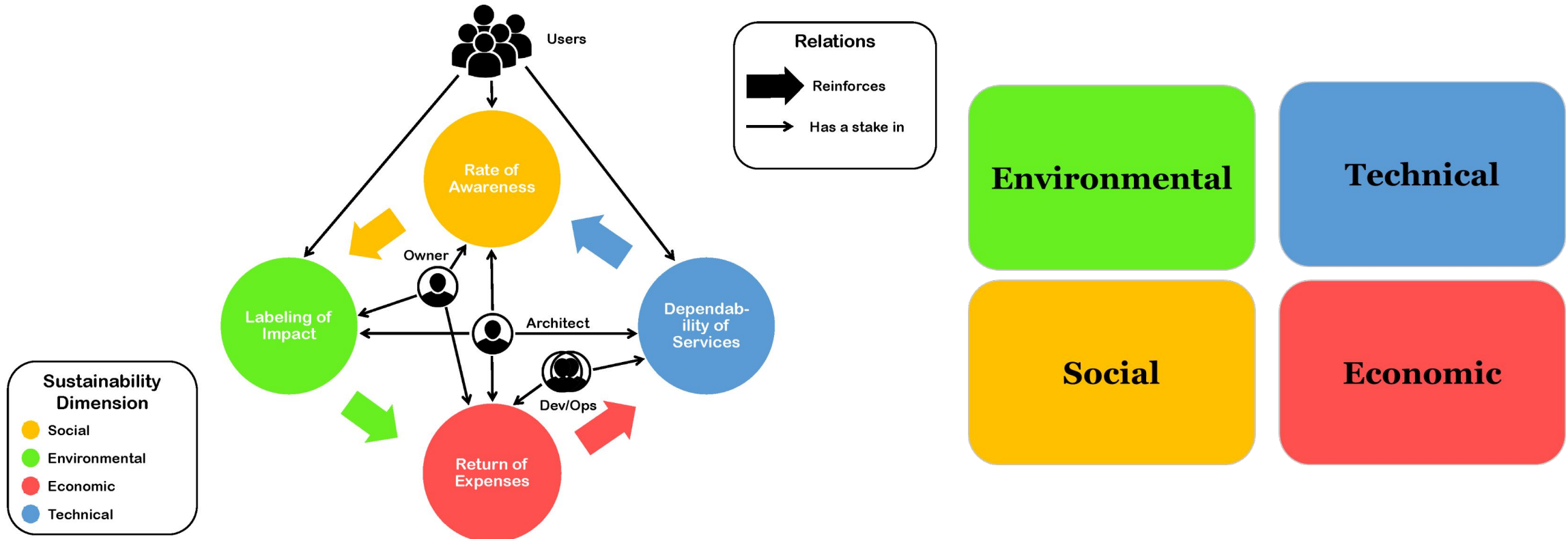
Environmental

Technical



<http://itea-mast.org/>

SustainableCloud



<https://s2group.cs.vu.nl/sustainablecloud/>

Coda

In music, a passage that brings a piece (or a movement) to an end



Takeaways (revisited)

01

$E(\text{software})$

Be aware and beware of the tools

02

$E(\text{software})$

*Requires reverting a few decades
worth of design decisions*

03

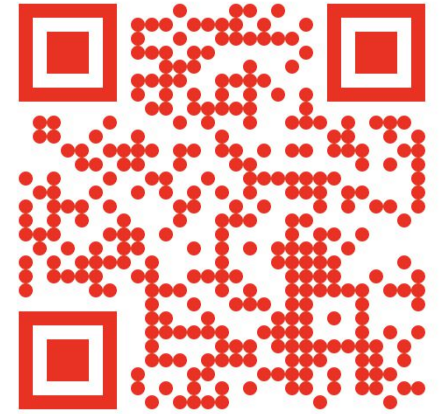
$\mathcal{O}^+(E(\text{software}))$

*Actionable observability,
not just observability*

Colophon

In publishing, a brief statement containing information about the publication

It takes a village

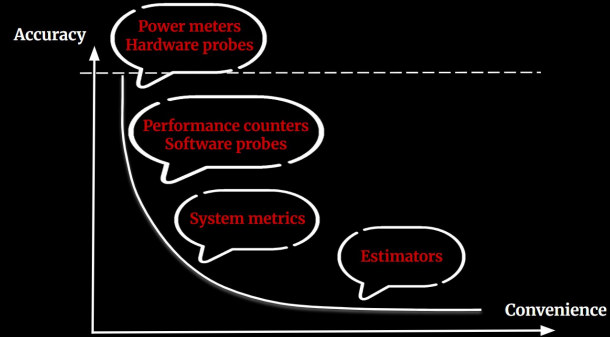


<https://vandriko.github.io/>

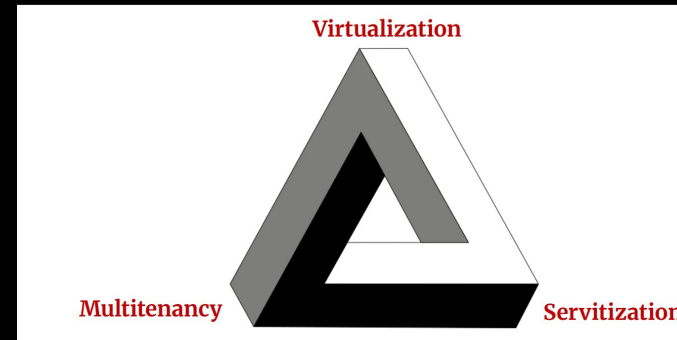


Thank you

Measuring energy consumption



The Devil's triangle in the cloud



Actionable Observability roadmap

