
The green cloud(ish) triangle
Divya_Mohan02

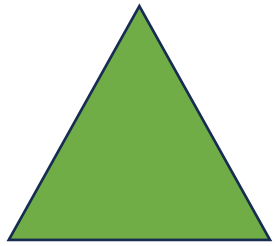
Hi there, I’m Divya!

- Senior Technical Evangelist @ SUSE
- Kubernetes documentation maintainer
- CNCF Ambassador

Divya_Mohan02

What are we going to learn today?

Green computing & why it matters!



The Green Cloud Triangle

Translating it into the cloud-native context

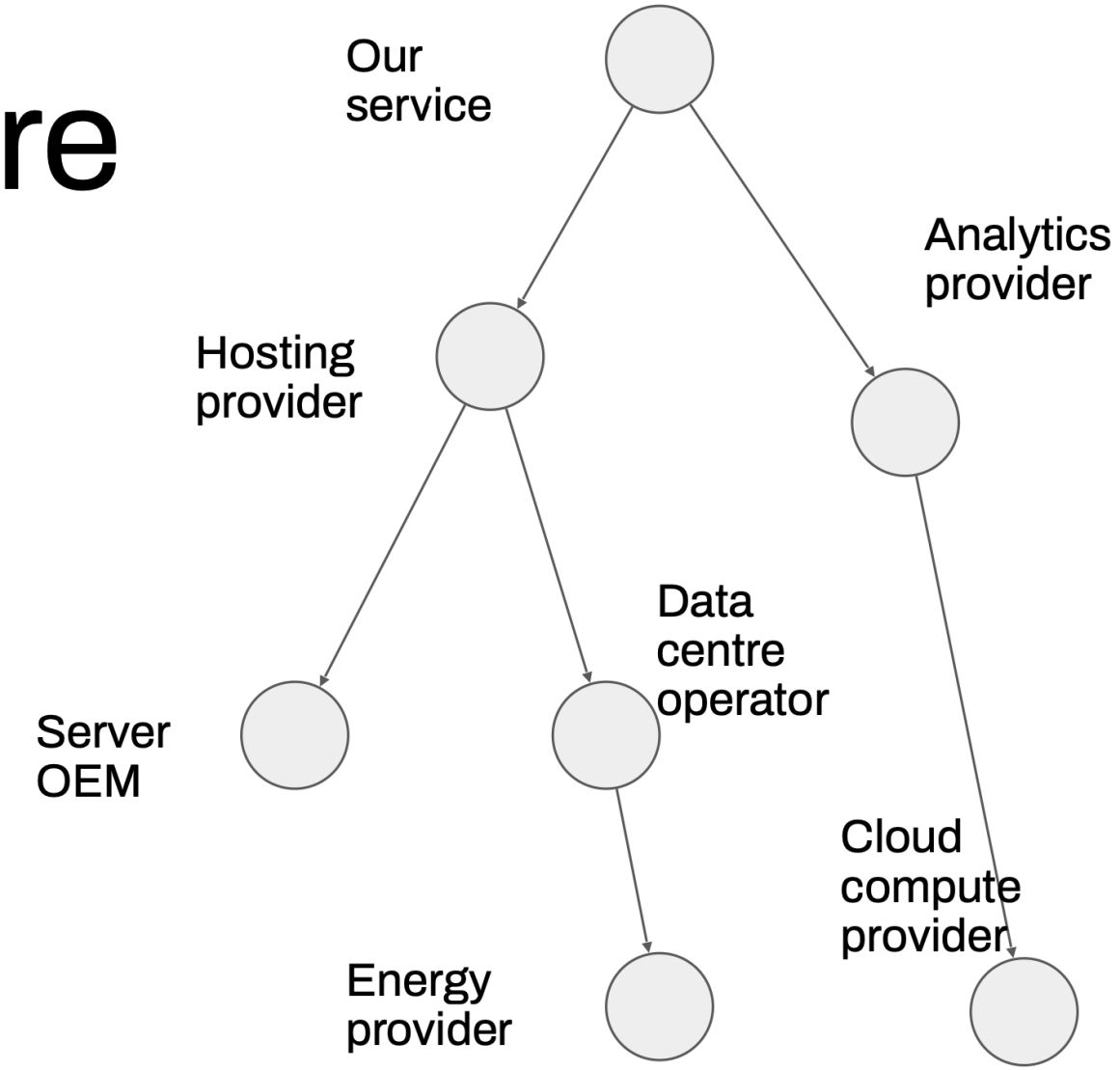
How are we doing so far?

	2015	2021	Change
Internet users	3 billion	4.9 billion	+ 60 %
Internet traffic	0.6 ZB	3.4 ZB	+ 440 %
Data centre workloads	180 million	650million	+ 260 %
Data centre energy use (excluding crypto)	200 TWh	220 - 320 TWh	+ 10-60%
Crypto mining energy use	4 TWh	100 - 140 TWh	+ 2,300 - 3,300%
Data transmission network energy use	220 TWh	260 - 340 TWh	+ 20 - 60%

Divya_Mohan02

Source: https://www.thegreenwebfoundation.org/wp-content/uploads/FOSDEM-2023-Responsible-Clouds-and-the-Green-Web-Triangle.pdf

Digital services are increasingly built by assembling supply chains of other services



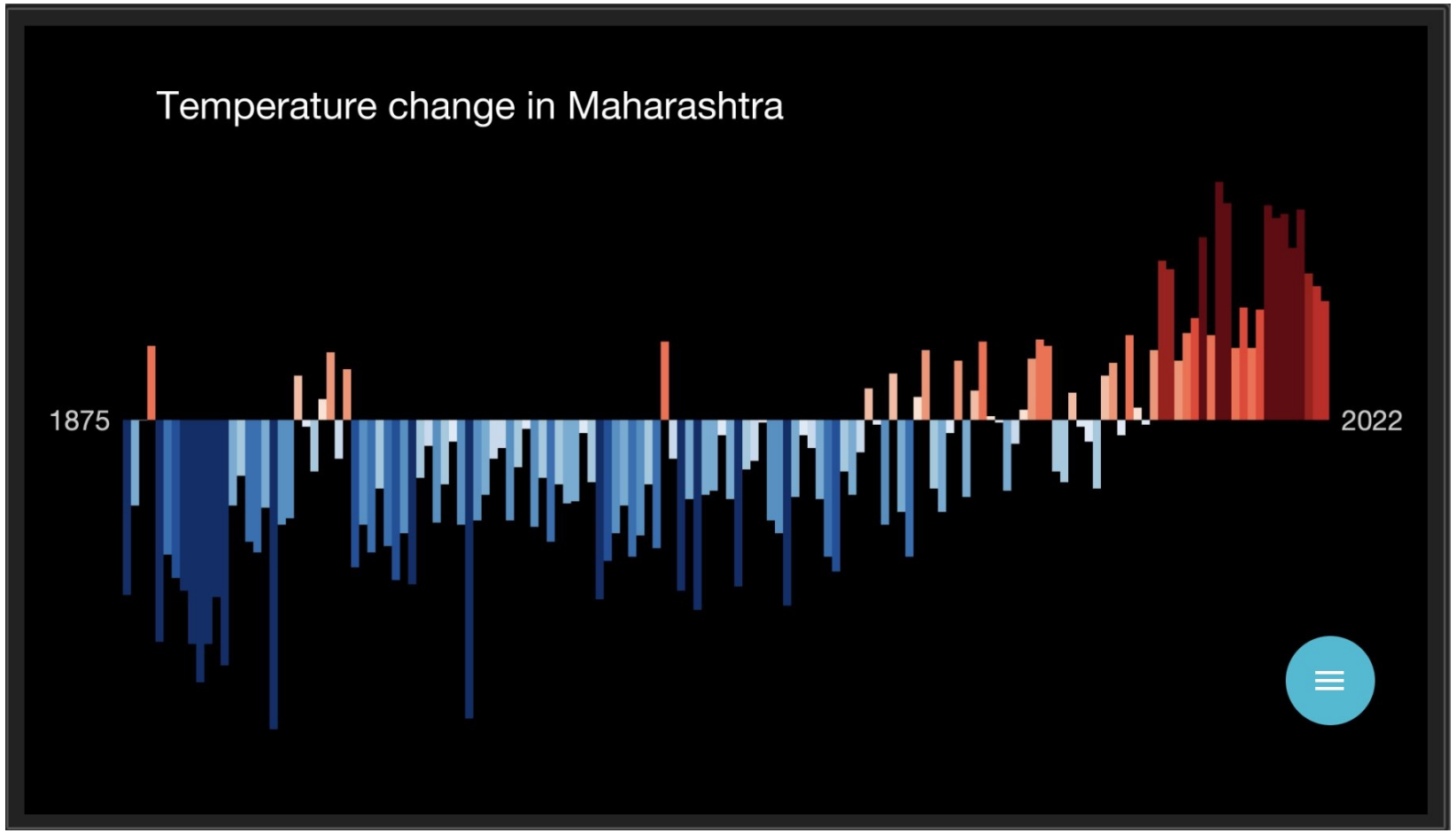
Divya_Mohan02

Source: https://www.thegreenwebfoundation.org/wp-content/uploads/FOSDEM-2023-Responsible-Clouds-and-the-Green-Web-Triangle.pdf

“Currently, ICTs account for between 5% and 9% of total electricity consumption”

www.enerdata.net. August 8, 2018.

If you don’t believe climate change is real…



Source: showyourstripes.info

Recognize this place?

“Green computing, green IT, or ICT sustainability, is the study and practice of environmentally sustainable computing or IT. ”

- Wikipedia

Divya_Mohan02

The Green Cloud Triangle

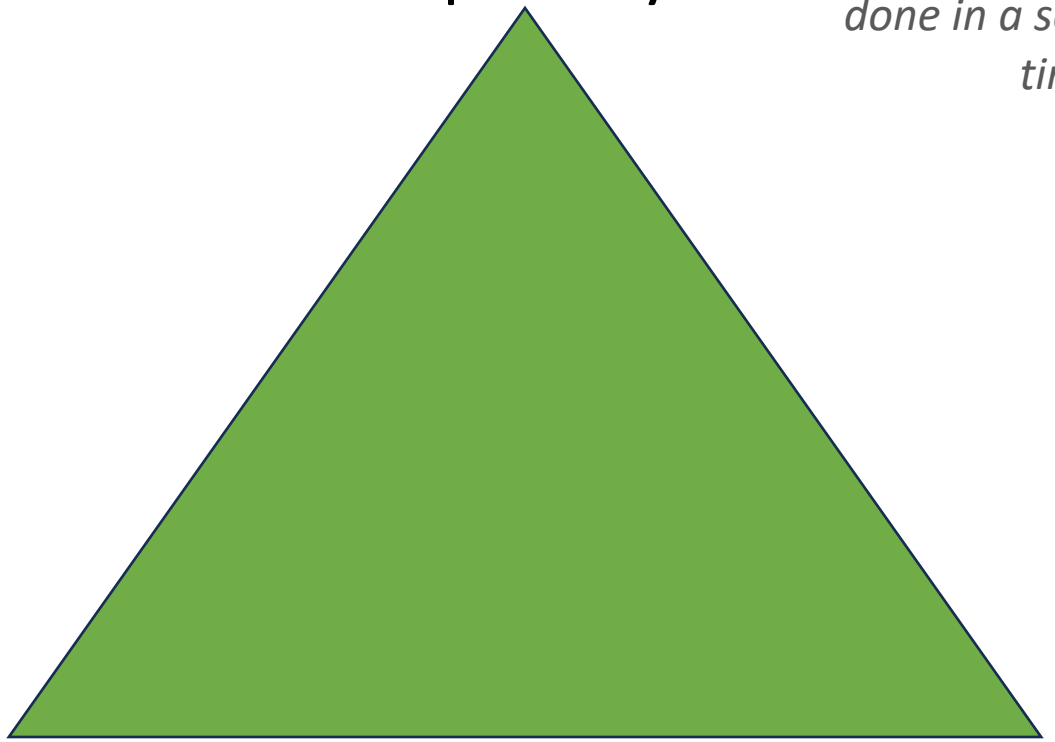
(part of the The Green Web Foundation Open Green Web Syllabus project)

Divya_Mohan02

The Green Cloud Triangle

Compute Cycles

How much work can be done in a set amount of time?



Response Times

What is the latency for the request?

Costs

What is the cost of emitting carbon for the workload at a sensible price?

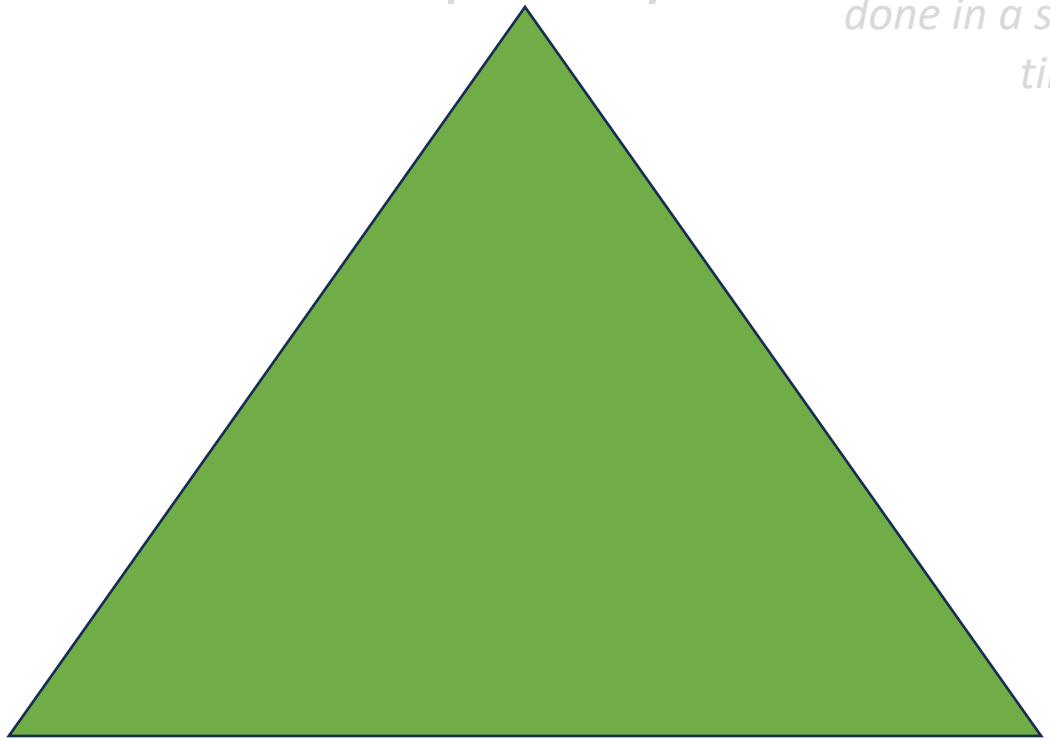
Divya_Mohan02

The Green Cloud Triangle

Keep:
- Low costs
- Low response times

Sacrifice:
- Compute Cycles

What you can do?
- High efficiency for the low latency requests that are serviced
- Use pre-computation and caching



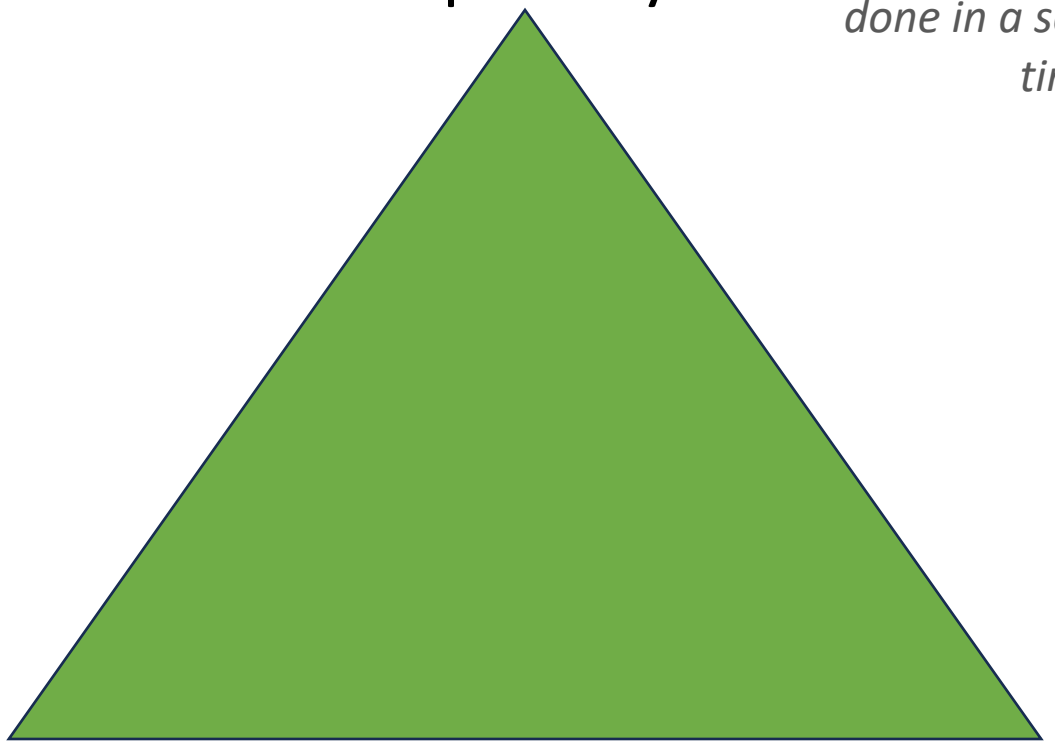
Divya_Mohan02

The Green Cloud Triangle

Keep:
- Low costs
- Compute Cycles

Sacrifice:
- Low response times

What you can do?
- Set expectations around UX
- Accept variable latency
- Parallelize and move work thro space/time



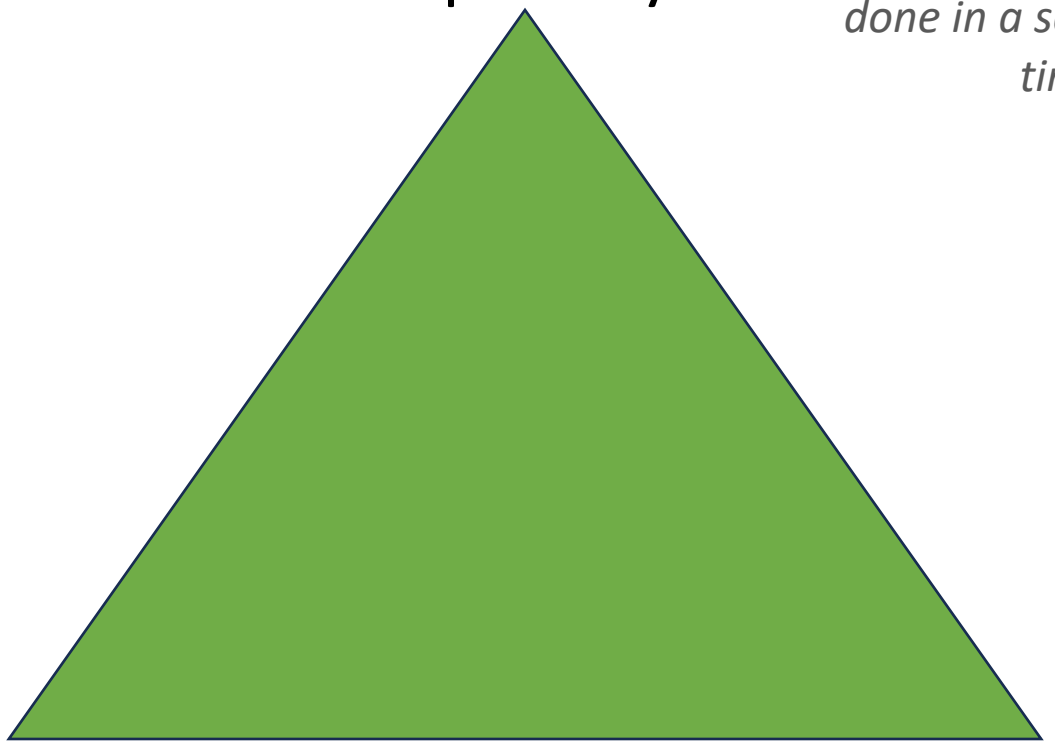
Divya_Mohan02

The Green Cloud Triangle

Keep:
- Low costs
- Compute Cycles

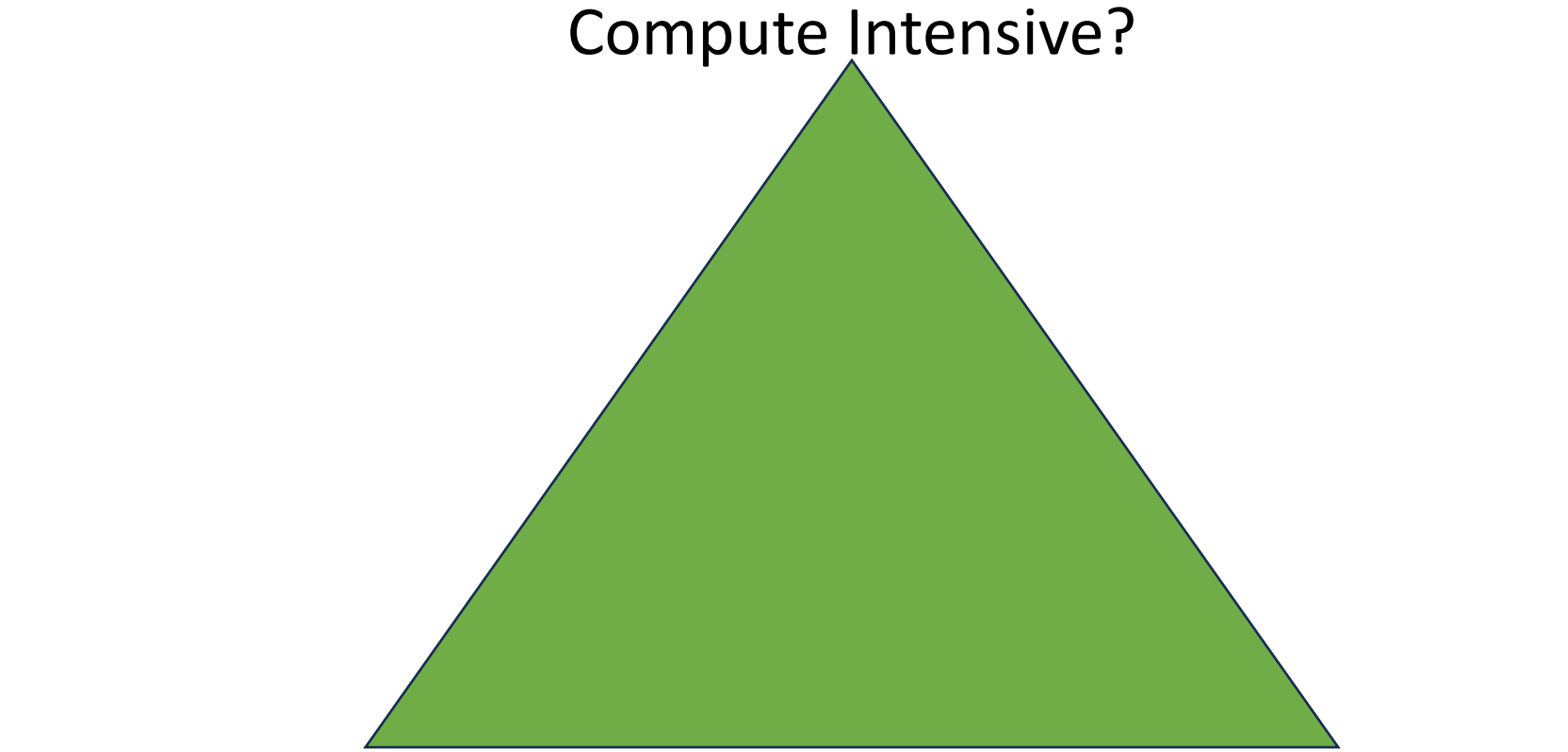
Sacrifice:
- Costs

What you can do?
- Invest in efficiency across the board
- Timeshift green energy to when you need it
- Split into different services



Divya_Mohan02

What if we rephrased this to?



Divya_Mohan02

The cloud native context

Divya_Mohan02

The science

- Generally, Kubernetes workloads have higher latencies
  - k3s & k0s were found to have the least latency **i.e. 5.6 s with similar throughput**

Kjorveziroski and Filiposka [16] conducted a series of experiments investigating the performance of serverless applications on K8s, k3s, and MicroK8s. They adapted the FunctionBench serverless benchmark [14] and used 14 tests to stress CPU (e.g., matrix multiplications), disk (e.g., GZip compression), and network (e.g., large file download). Using OpenFaaS as serverless platform, they measured cold-start latencies in a cluster of one controller and five worker nodes running Ubuntu 20.04 with 8 GB RAM. The authors considered the latency differences between k3s (5.6 sec) and K8s (6.4 sec) as significant. For the benchmark applications throughput was also almost equal across K8s distributions.Also average response times did not vary more than 3 percent in most cases. These tests

Source: https://programming-group.com/assets/pdf/papers/2023_Lightweight-Kubernetes-Distributions.pdf

Why is this important?

- Latency is directly proportional to response times
  - We want our applications to respond as quickly as possible.
  - Especially in the case of serverless applications with cold starts factored in.

Kjorveziroski and Filiposka [16] conducted a series of experiments investigating the performance of serverless applications on K8s, k3s, and MicroK8s. They adapted the FunctionBench serverless benchmark [14] and used 14 tests to stress CPU (e.g., matrix multiplications), disk (e.g., GZip compression), and network (e.g., large file download). Using OpenFaaS as serverless platform, they measured cold-start latencies in a cluster of one controller and five worker nodes running Ubuntu 20.04 with 8 GB RAM. The authors considered the latency differences between k3s (5.6 sec) and K8s (6.4 sec) as significant. For the benchmark applications throughput was also almost equal across K8s distributions.Also average response times did not vary more than 3 percent in most cases. These tests

Source: https://programming-group.com/assets/pdf/papers/2023_Lightweight-Kubernetes-Distributions.pdf

The science

- Resource utilization for containerized workloads when idling is **high**
  - For control plane & worker nodes
  - Even for lightweight distributions!

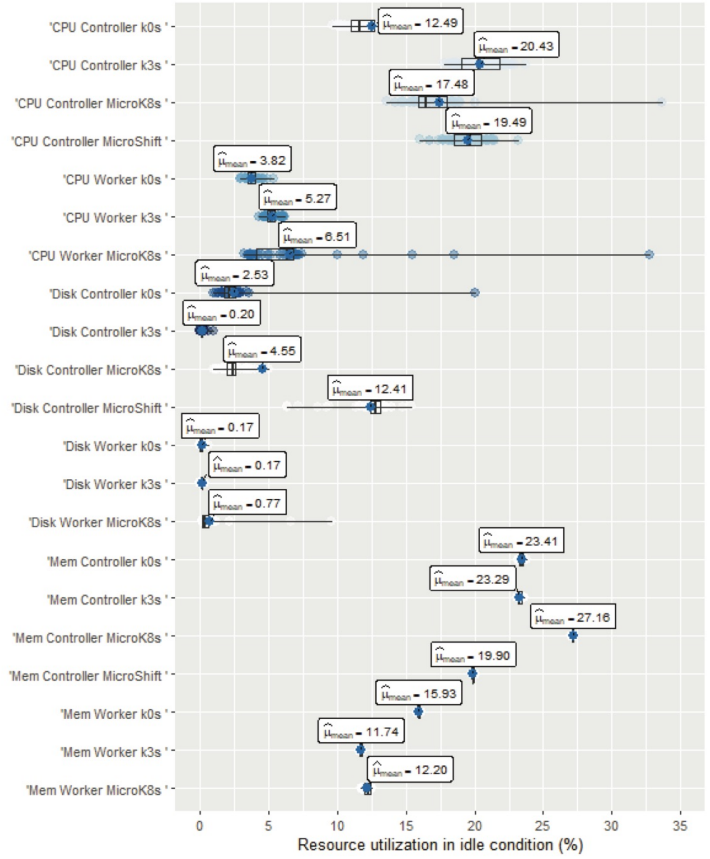


Figure 3: Resource utilizations of controllers and workers when idling. CPU is continuously utilized, none of the K8s distributions is sticking out.

Divya_Mohan02

Source: https://programming-group.com/assets/pdf/papers/2023_Lightweight-Kubernetes-Distributions.pdf

Why is this important?

- Resource utilization directly proportional to carbon cost
- Annual CO2 emissions per cloud server using 100% green electricity = 160 KG CO2e
  - On-premise server = 320 KG CO2e

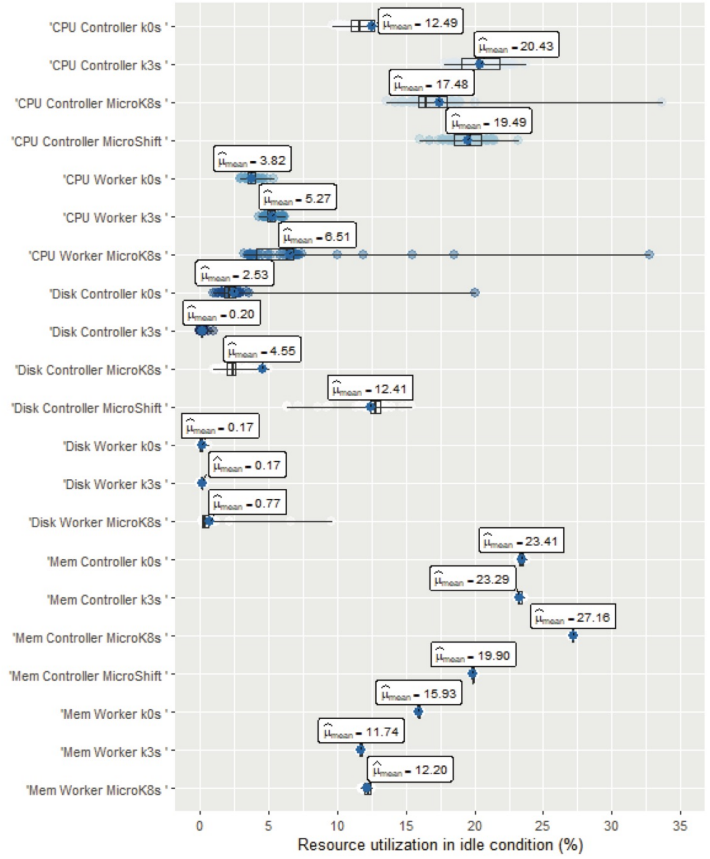
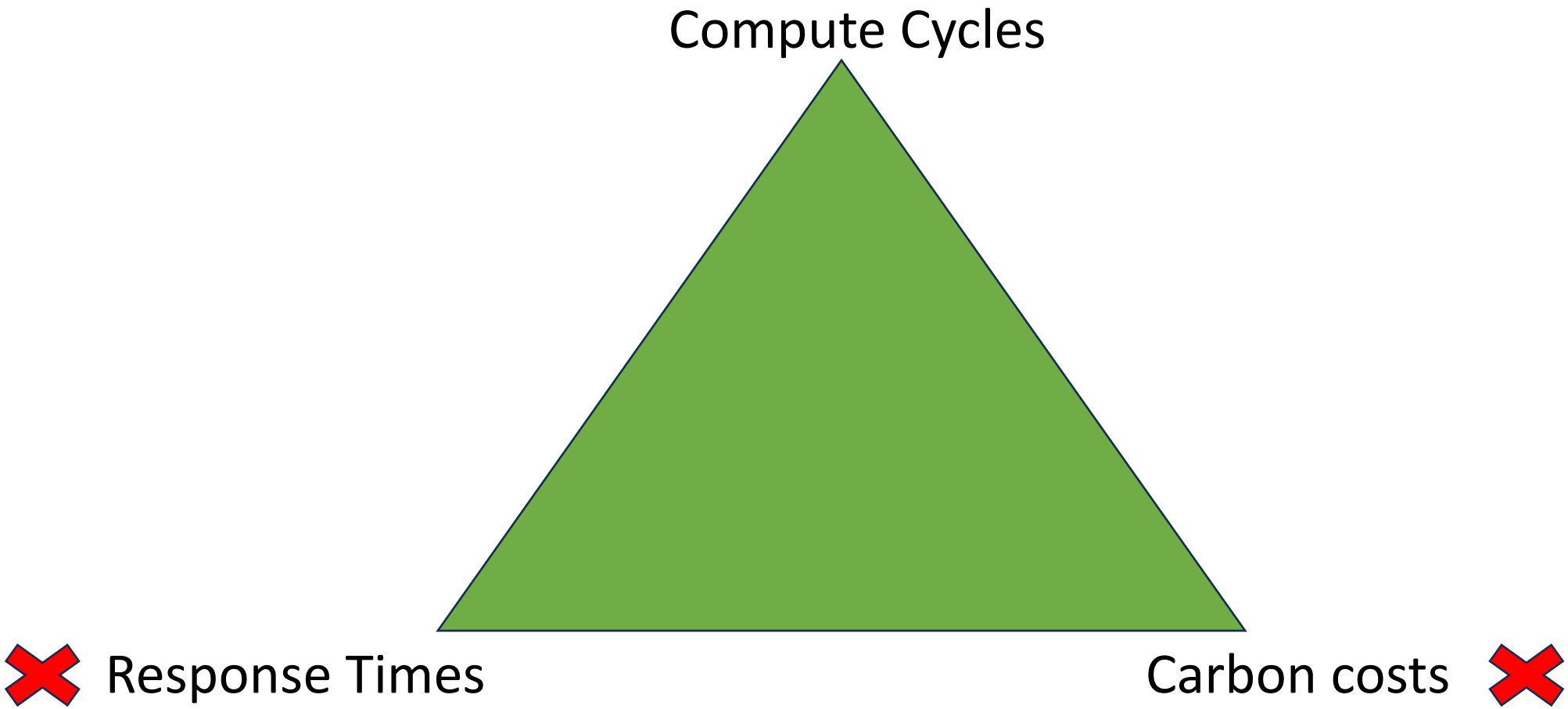


Figure 3: Resource utilizations of controllers and workers when idling. CPU is continuously utilized, none of the K8s distributions is sticking out.

Source: https://programming-group.com/assets/pdf/papers/2023_Lightweight-Kubernetes-Distributions.pdf

Divya_Mohan02

What does this mean?



Containerization isn’t going away!

How can we make this better?

- Usage of WebAssembly-based runtimes where possible
  - Smaller resource footprint
  - Millisecond cold start times

For even smaller embedded devices, the use of Linux containers and container orchestration may however be challenging. Wind River has announced an OCI-compliant container runtime for its RTOS VxWorks, which reportedly has a memory footprint of less than 100 KByte, but still preserves the RTOS deterministic execution and certification[17]. An alternative to using full-blown containers could be using WebAssembly outside of the browser[18], which is also sandboxed and portable. Krustlet is a Kubelet that schedules workloads in a WebAssembly runtime[19]. WebAssemblies can be very small, sometimes in the order of bytes. However, unlike containers they require re-compilation of the source code to WebAssembly.

How can we make this better?

- Use lightweight Kubernetes distributions, wherever possible
  - Ideal for **local & resource-constrained workloads**
  - Smaller binary size
  - Smaller resource footprint
- Visualize, measure, and act!
  - Form metrics around resource consumption and measure.
  - Take action!
    - **Most** dev/preview workloads do not need to be serviceable during non-working hours.
    - Green hosting


Demo time!

What we’re going to do

1

Install and run workloads on the following Wasm runtimes in a k3d cluster

- Spin
- Lunatic
- WWS
- Slight

2

Sending requests & viewing user response

3

Visualise the results via Kepler’s Grafana Dashboard

4

Demonstrate the scheduled shut down of a workload using the kube-green operator

Divya_Mohan02

Why did we do this

- Practically implementing the results of what we learned before
  - Lightweight Kubernetes
    - We’re using a k3d cluster i.e. a lightweight docker wrapper for the k3s distribution
  - Wasm runtimes
    - We’re using multiple Wasm runtimes – spin, lunatic, slight, wws
      - Comparatively smaller footprints & startup times
  - Visualise, measure, and act
    - Visualised that memory consumption for the wws workload was higher for a particular timeframe
      - This was done via Kepler’s Grafana dashboard
    - Scheduled a shutdown for the wws workload based on the above observation

Divya_Mohan02

What did we learn?

- It is **possible** to make our cloud native workloads greener!
- There are several **open source projects** already doing the work:
  - Kepler
  - Kube-green
  - KEDA
  - WebAssembly
    - Via the component model spec

Resources

GitHub link for slides

Divya_Mohan02

Resources

GitHub link for demo

Divya_Mohan02

 Thank you 

Divya_Mohan02