Do The Math: Autoscaling Applications with Kubernetes

Antoine Hamon, Nephely

About me

KubeCon CloudNativeCon Europe 2020 Virtual

Antoine Hamon

Freelance, Cloud Architect (AWS, OpenStack), SRE/DevOps
Nephely

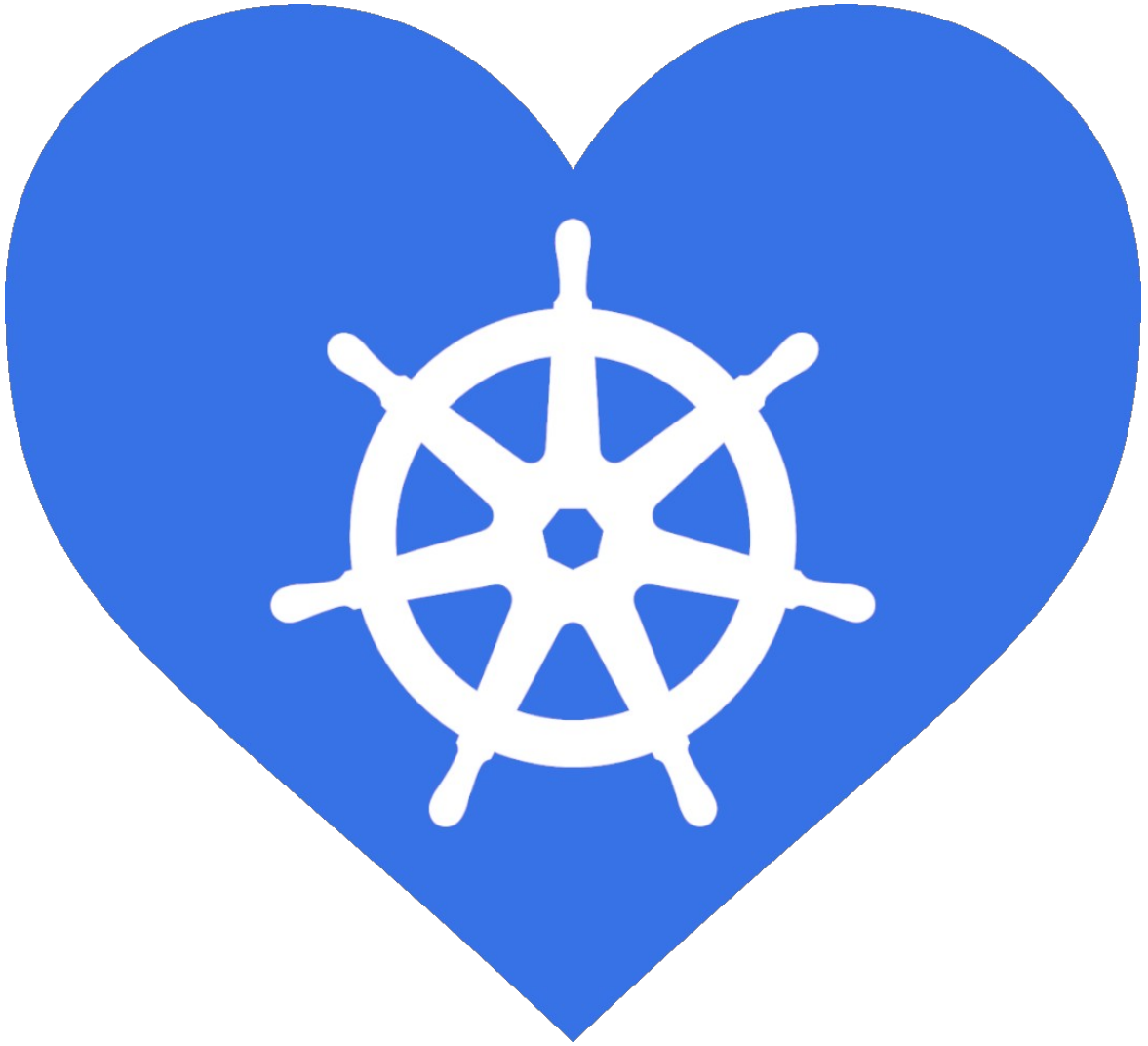
DevOps Lead @ PypeStream

 @AntoineKanshi

 https://medium.com/nephely/do-the-math-auto-scaling-microservices-applications-with-orchestrators-d15c78c0b12a

 https://github.com/nephely-io/app-autoscaling-calculator

Abstract



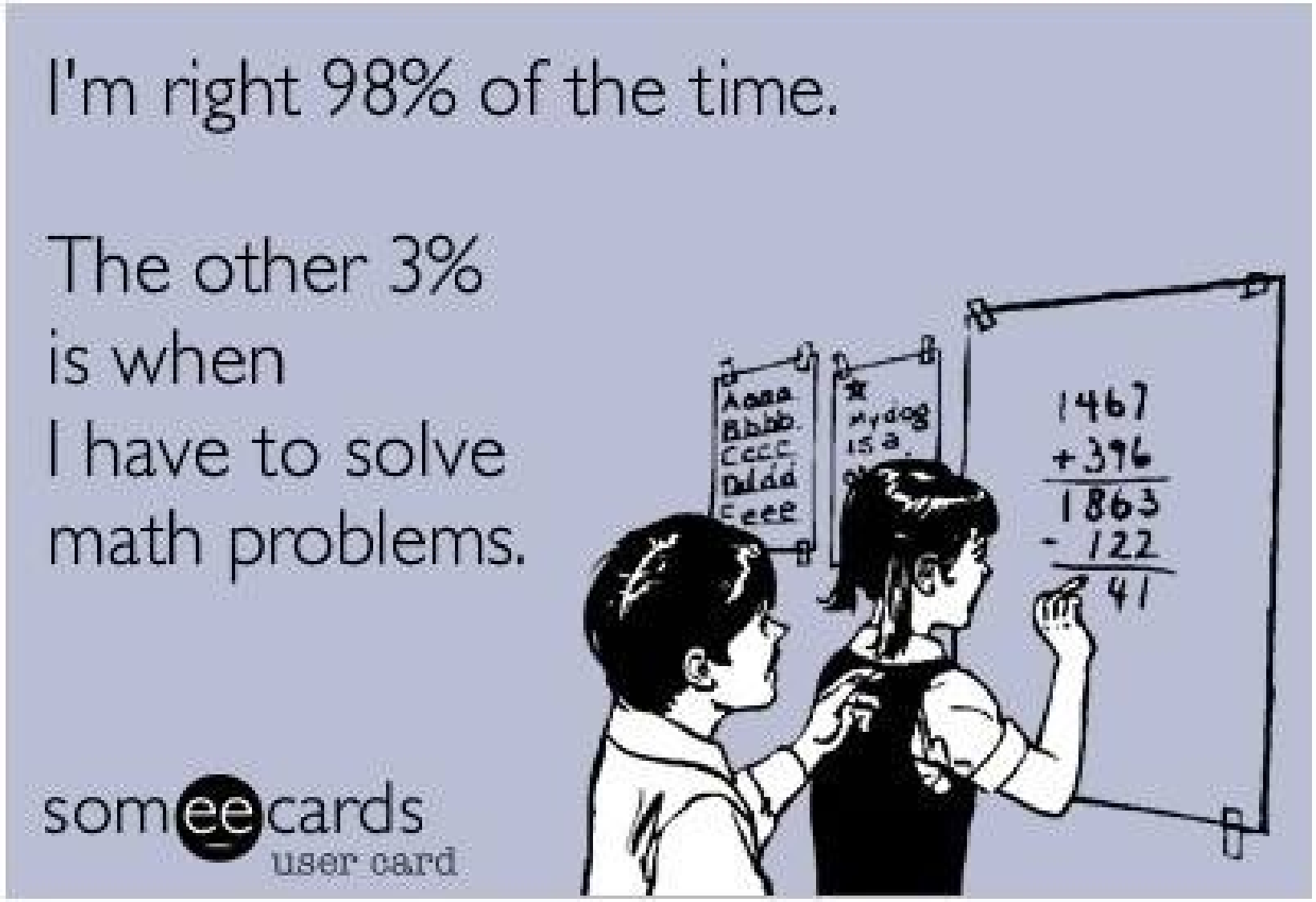
Auto-scaling

Goals

- Scale up is *fast-enough* so users do not face any error (UX)
- Run the minial number of replicas needed to handle the load ($)
- Work for every given load variation

⇒ Fine tune the upscale & downscale thresholds
(Kubernetes has a single parameter)

Google Math Translate



Variables

Introducing some variables:

- **N_u:** the number of users
- **T_{tot}:** The '*short period of time*'
- **$L_u(t)$:** the load generated by a single user on the system (t=0 points to the moment when the user starts the scenario)
- **$L_{tot}(t)$:** the total load of the system

Hypotheses

- Load is evenly distributed across all replicas
- Restful/stateless application
- Requests timings must be shorter that the Kubernetes load check interval
- Dealing with a large number of users

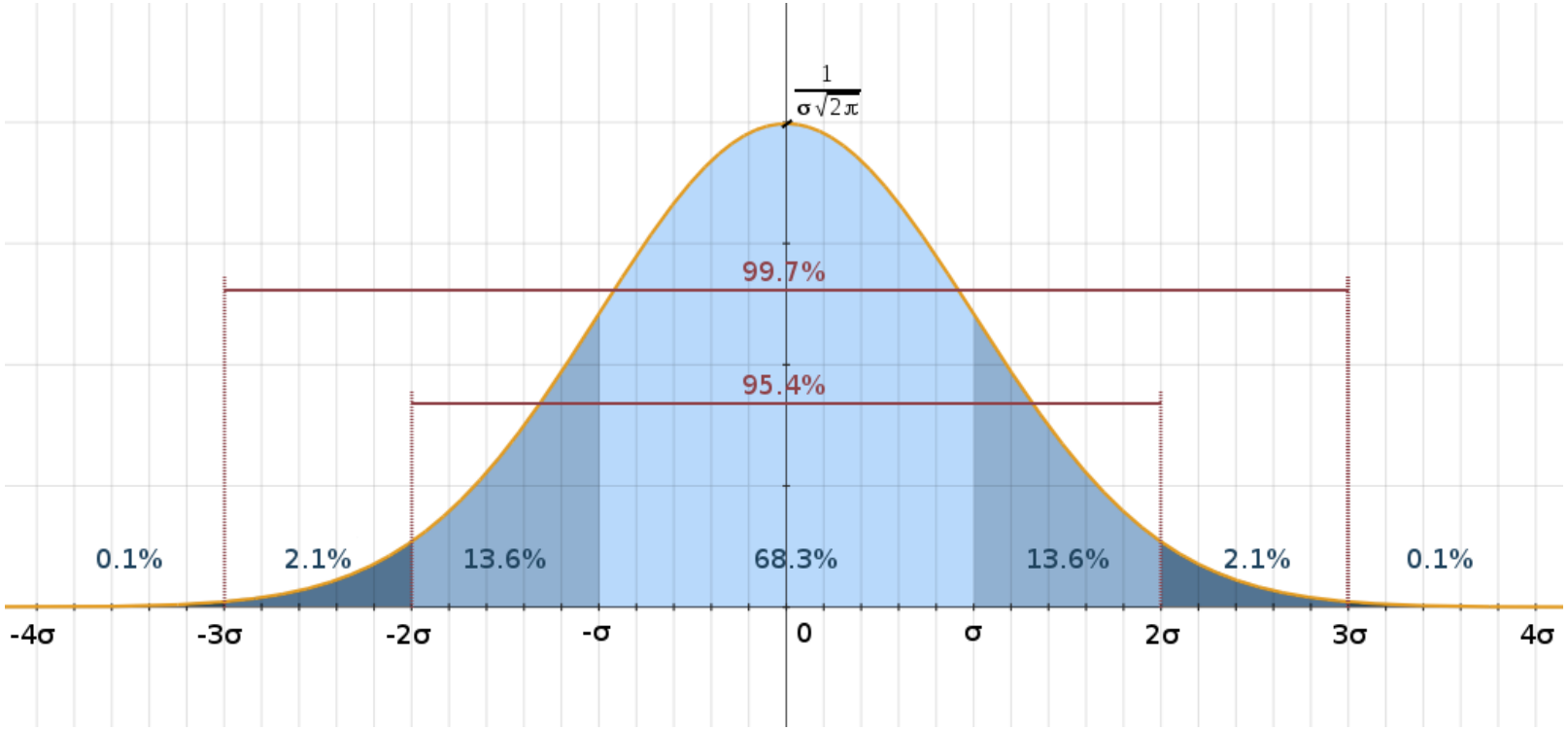
Getting All Gaussian

Gaussian (or normal) distribution:

$$G(t) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(t-\mu)^2}{2\sigma^2}}$$

μ: is the expected value

σ: is the standard deviation



Getting All Gaussian

$$\sigma = T_{tot} / 4 \quad \& \quad \mu = 3 / 4 \times T_{tot}$$

⇒ ***The load $L_{tot}(t)$ generated by 99.7% of N_u, each user performing a consuming operation $L_u(t)$ and where 95.4% of them are doing it within a duration T_{tot}.***

$$G(t) = \frac{4 N_u}{T_{tot} \sqrt{2\pi}} e^{\frac{-(4t - 3T_{tot})^2}{T_{tot}^2}}$$

Introducing Reimann

Reimann sum:

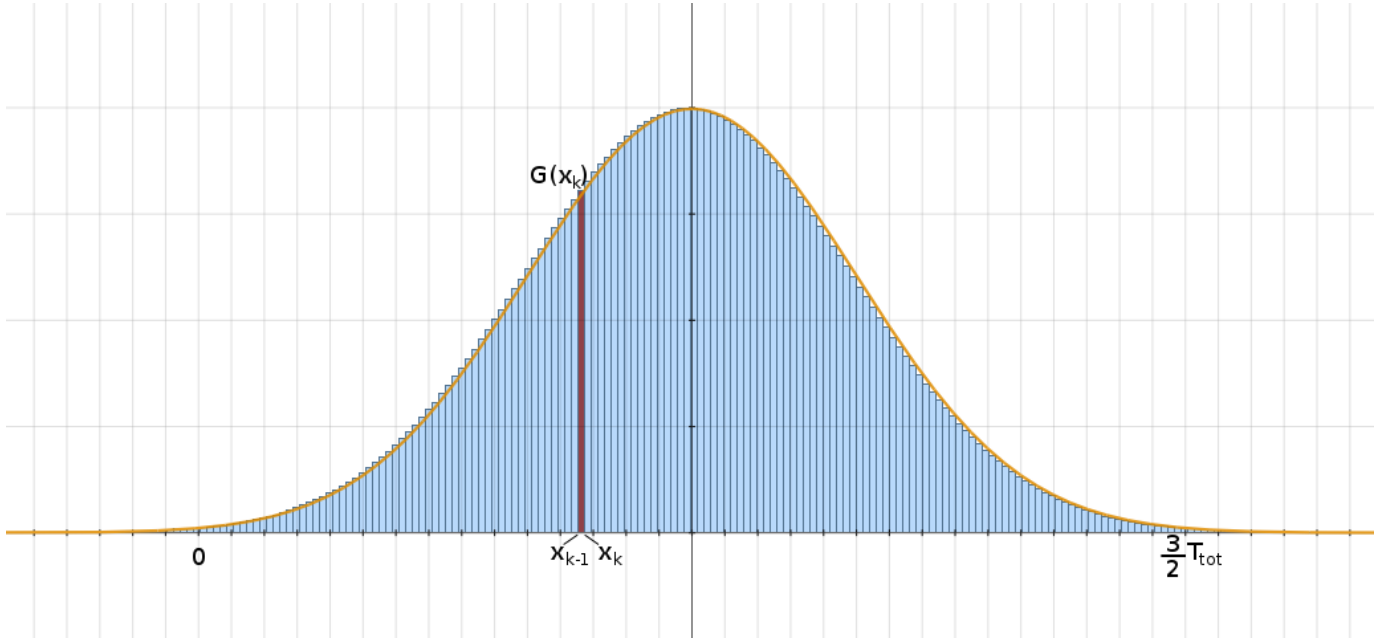
$$\int_a^b f(x)\,dx = \lim_{n\to\infty} \sum_{k=1}^{n} (x_k - x_{k-1}) f(x_k)$$

f being the function to approximate
(the Gaussian in our case)

with $\mathbf{x_k}$ defined as follow:

$$x_k = a + k\frac{b-a}{n}, 0 \leqslant k \leqslant n$$

n: is the number of subdivisions
a: is the lower bound (0)
b: is higher bound (3 / 2 x T_{tot})



Introducing Reimann

The load formula at the given time ***t*** is the sum of every subdivision's number of users multiplied by the user load function at their corresponding time:

$$L_{tot}(t) = \lim_{n \to \infty} \sum_{k=1}^{n} (x_k - x_{k-1}) G(x_k) L_u(t - x_k)$$

After replacing variables and having simplified the formula, this becomes:

$$L_{tot}(t) = \frac{6 N_u}{\sqrt{2\pi}} \lim_{n \to \infty} \sum_{k=1}^{n} \left(\frac{1}{n}\right) e^{-\left(\frac{6k}{n} - 3\right)^2} L_u\left(t - k \frac{3 T_{tot}}{2n}\right)$$

To finish, we will use the dichotomy algorithm to find the upscale threshold.

User Load Function

⚠ Impossible to have a proper mathematical representation

1. Single instance, single action. Increase the number of users until resources limit is reached (or when timings start to exeed expectations).
2. Devide resources per the number of users. Calculate also the average time of the action.
3. Create a user scenario out of previous load test figures.

Demo Time!

Best practices

- Application’ startup time should be fast
- Play with both resources request & resources limit
- Increasing the minimum number of replicas will also increase the upscale threshold (since the highest load increase per replica is at the beginning of the simulation)
- Increasing resources limitations will also increase the upscale threshold


KubeCon
CloudNativeCon
Europe 2020
Virtual
HELM
KEEP CLOUD NATIVE
CONNECTED