

Implementacja i testy skalowalności systemu wideokonferencyjnego

2025-11-29

**Jakub Jędrzejczyk Sebastian Kwaśniak Anna
Berkowska**

Skalowanie jvb

Co zostało dodane do values.yaml?

- Aby przeskalać jvb, następujące wartości zostały dodane:

jvb:

```
## Set JVB instance count:  
replicaCount: 2  
## Expose JVB interface port to the outside world  
# only on nodes that actually have it:  
useHostPort: true  
## Make every JVB pod announce its Node's external  
# IP address and nothing more:  
useNodeIP: true
```

octo:

```
## Enable OCTO support for both JVB and Jicofo:  
enabled: true
```

OCTO - co to?

- OCTO to mechanizm multi-bridge routing, który pozwala rozproszyć uczestników jednego spotkania między wiele instancji Jitsi Videobridge (jvb).
- Bez OCTO każde spotkanie musi być obsługiwane przez jeden jvb; gdy osiągnie limit CPU/bitrate - jakość spada.
- Z OCTO spotkanie może używać kilku jvb jednocześnie.

Przed skalowaniem jvb

```
Context: kubernetes-admin@kubernetes [RW]
Cluster: kubernetes
User: kubernetes-admin
K9s Rev: v0.50.16
K8s Rev: v1.32.9
CPU: n/a
MEM: n/a

          <0> all      <a>     Attach
          <1> monitoring <ctrl-d> Delete
          <2> default    <d>     Describe
          <e>     Edit
          <?>     Help
          <shift-j> Jump Owner
```



pods(default)[4]							
NAME	PF	READY	STATUS	RESTARTS	IP	NODE	AGE
myjitsi-jitsi-meet-jicofo-6fff848dfc-hf5d8	●	1/1	Running	0	10.2.161.95	jitsi-2	5d21h
myjitsi-jitsi-meet-jvb-5f599db5c8-74t8z	●	2/2	Running	0	10.2.161.86	jitsi-2	5d21h
myjitsi-jitsi-meet-web-86c6d94bf-bxmr4	●	1/1	Running	0	10.2.161.69	jitsi-2	5d21h
myjitsi-prosody-0	●	1/1	Running	0	10.2.161.107	jitsi-2	2d16h

Po skalowaniu jvb

```
Context: kubernetes-admin@kubernetes [RW]
Cluster: kubernetes
User:   kubernetes-admin
K9s Rev: v0.50.16
K8s Rev: v1.32.9
CPU:    n/a
MEM:    n/a
```

pods (default) [6]							
Name	PF	Ready	Status	Restarts	IP	Node	Age
myjitsi-jitsi-meet-jicofo-d4978ffdb-w2nkn	●	1/1	Running	0	10.2.161.105	jitsi-2	22h
myjitsi-jitsi-meet-jvb-778dfb5469-c74sx	●	2/2	Running	0	10.2.161.75	jitsi-2	22h
myjitsi-jitsi-meet-jvb-778dfb5469-ghklw	●	2/2	Running	0	10.2.63.188	jitsi-3	22h
myjitsi-jitsi-meet-web-86c6d94bf-82fsp	●	1/1	Running	0	10.2.161.68	jitsi-2	22h
myjitsi-prosody-0	●	1/1	Running	0	10.2.63.190	jitsi-3	22h
tailscale-proxy	●	1/1	Running	0	10.2.161.79	jitsi-2	22h

Jak to wygląda w logach jicofo

Przetestowanie skalowania

Szybkie przypomnienie

- Master Node - maszyna której rolą jest zarządzanie Kubernetes
- Worker Node - maszyna której rolą jest trzymać aplikacje (w podach) użytkownika

Kubernetes

Co się okazuje:

- Master nie obsługuje ruchu użytkowników.
- Master nie uruchamia Podów aplikacyjnych.
- Master nie streamuje danych, nie przetwarza requestów HTTP, nie robi downloadów/uploadów.

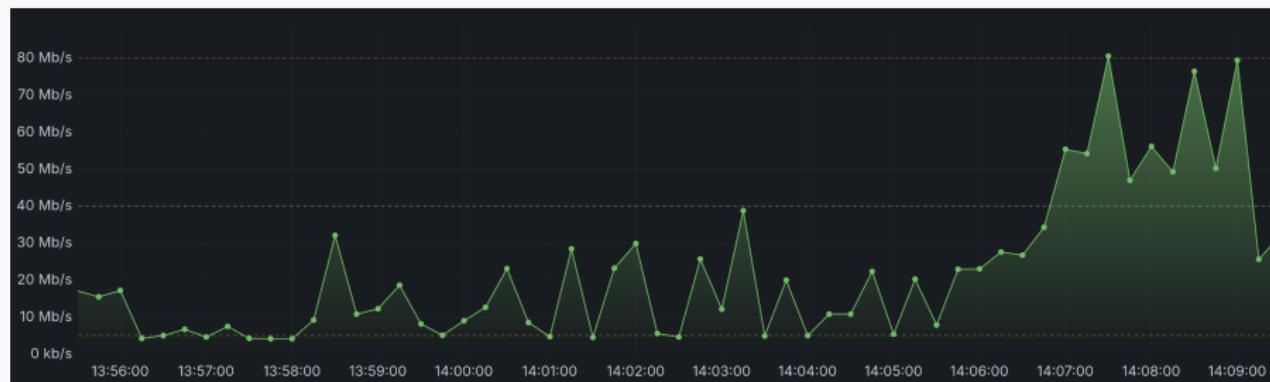
Kubernetes 2

Nawet jeśli master ma wolny internet (np. 1 Mbps), nie wpływa to na:

- prędkość odpowiedzi aplikacji,
- throughput,
- czas obsługi zapytań HTTP/Websocket/GRPC,
- szybkość pobierania danych przez aplikację.

Kubernetes 3

To w trakcie stress testowania jitsi, gdy master ma 128kbit/s:



Czemu nie użyliśmy narzędzi zewnętrznych

Webrtcperf dla jitsi jest całkowicie zepsute.

No to teraz jak przetestować jitsi...

Jitsi bardzo dobrze się skaluje. Według oficjalnych pomiarów:

- Dla 1056 strumieni wideo z bitrate 550mbit/s zużycie CPU to tylko 20% przy czterordzeniowym procesorze.
- Dla 1056 strumieni wideo Zużycie RAMu nie przekroczyło 1.5GB

Czemu tak się dzieje?

Jitsi Videobridge jest tylko przekaźnikiem, bez żadnego transkodowania.
Nie tworzy skomplikowanych reguł, ani nie weryfikuje nic.

Działa trochę jak router.

Demo

Dziękujemy za uwagę