BigBlueButton Konferenzen mit wenig Latenz streamen

Tobias Gall
Daniel Schreiber

Chemnitzer Linux-Tage 2021

14.03.2021

Motivation

Motivation

- Chemnitzer Linux-Tage sollen digital stattfinden!
- Welche Möglichkeiten haben wir zur Umsetzung?
- Wie haben das andere Konferenzen gemacht?

Lösungsskizze

Wie ist der Plan?

- Vorträge mit BigBlueButton
  - Infrastruktur und Erfahrung ist vorhanden.
  - Wir betreuen seit März 2020 die BBB-Instanz der TU Chemnitz.

Aber:

- BigBlueButton ~~ist~~war nicht für viele Teilnehmer geeignet.

→ wir brauchen was skalierbares: Streaming

BigBlueButton

- OpenSource Videokonferenzlösung
- Setzt Focus auf Onlineunterricht
- Bietet vielfältige Interaktionswerkzeuge
  - Chat
  - Geteilte Notizen
  - Whiteboard
  - Umfragen
- ausschließlich Webbrowser als Client

Grundlagen

Streamingtechnologien

HLS/DASH:

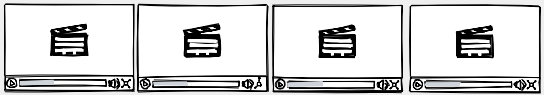
- Video in Segmente zerlegen
- mehrere Auflösungen bereithalten
- herunterladen, zusammensetzen, abspielen
- HTTP/TCP

WebRTC

- Videoframe kodieren

→ sofort senden

- RTP/UDP

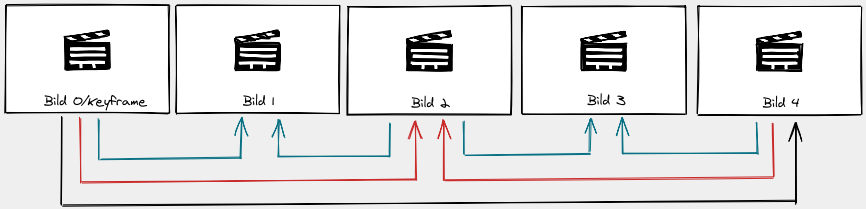


	HLS/DASH	WebRTC
Zielgruppe	VoD, Live-Stream	Echtzeitvideo
Datentransport	HTTPS (TCP)	SRTP (UDP)
Videocodecs	H.264/H.265/VP9/AV1	VP8/VP9/H.264
Audiocodecs	AAC	Opus
typische Latenz	10s-40s	< 1s
Skalierbarkeit	einfach	aufwändig
CPU-Bedarf Server	niedrig	höher
bei Netzstörungen	Stream steht	Frames droppen
Bendbreitensteuerung	Client	Server

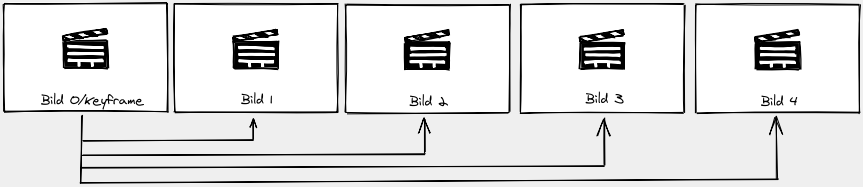
Tabelle: Vergleich von Web Streamingtechnologien

Video Encoding

Non Live Encoding



Live Encoding



Streaminglösungen für BBB

HLS/DASH (RTMP):

- aau-zid/BigBlueButton-liveStreaming (GitHub)
- spoutbreeze/spoutbreeze (GitHub)
- Neu: lukas2511/bbb-streaming (GitHub)
- Kommerzielle Streaminglösungen

WebRTC:

- **Grillenzirpen**
- Unsere eigene Lösung

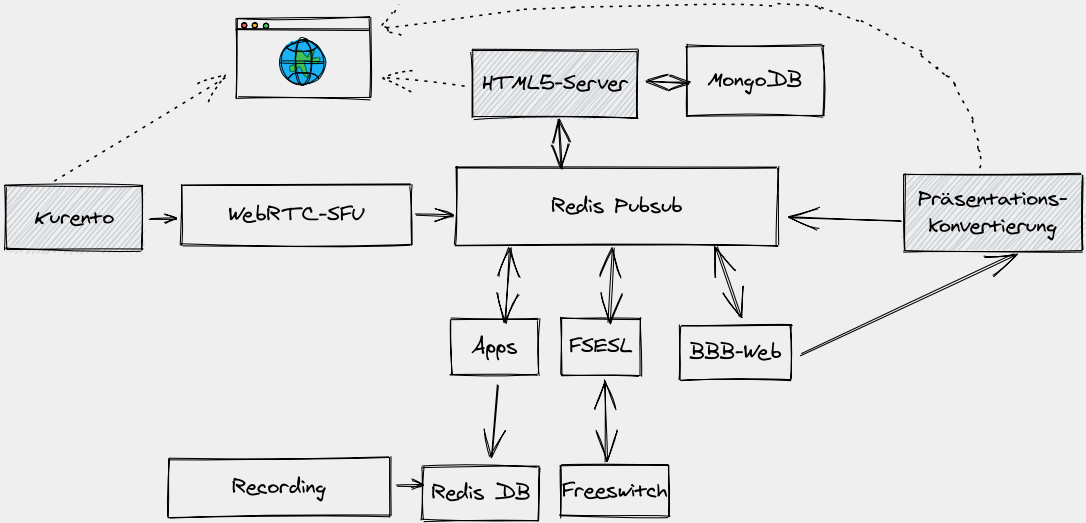
Designziele

Designziele

- Alle CLT-Gäste sollen Vorträge sehen können
- Interaktion ermöglichen $\rightarrow$ kurze Latenz
- Lösung soll auf Infrastuktur der TU Chemnitz laufen können
- Lösung soll wiederverwendbar sein
- wenig Leistungsbedarf auf dem Client

Umsetzung

Architektur von BBB

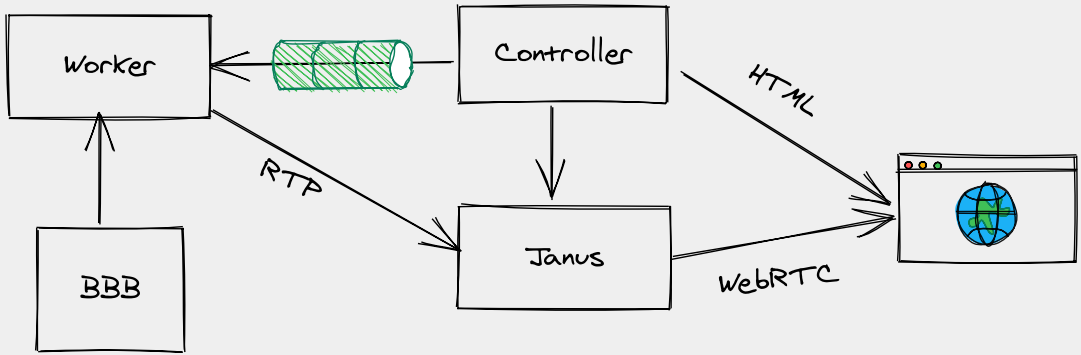


Warum BBB-Streaming schwierig ist

- BBB enthält keinen Bildmischer
- Layout dynamisch
- Whiteboard

→ Man muss einen Browser nachimplementieren

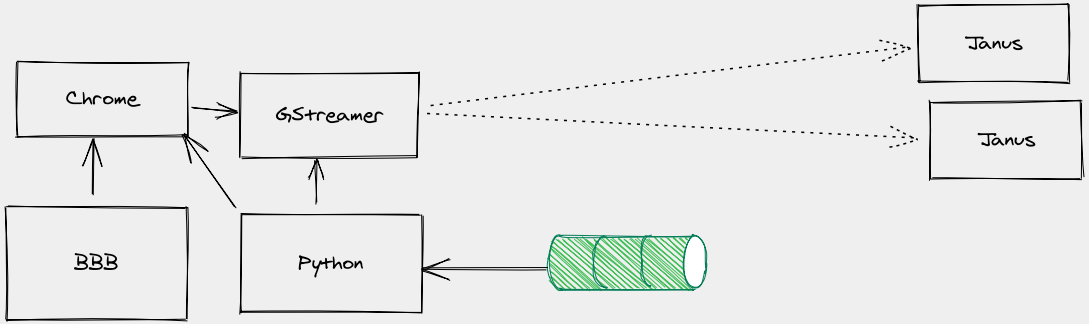
Architektur der Streaminglösung



Worker

- Python-Script (Als Docker-Image)
- Startet Chrome und nimmt an BBB-Konferenz teil
- Media-Pipline mit GStreamer
  - Screen-Capture mit xvfb
  - Pulseaudio loopback
  - Kodieren in VP8 und drei Auflösungen (Video-Simulscast)
  - Medien via RTP an Janus schicken

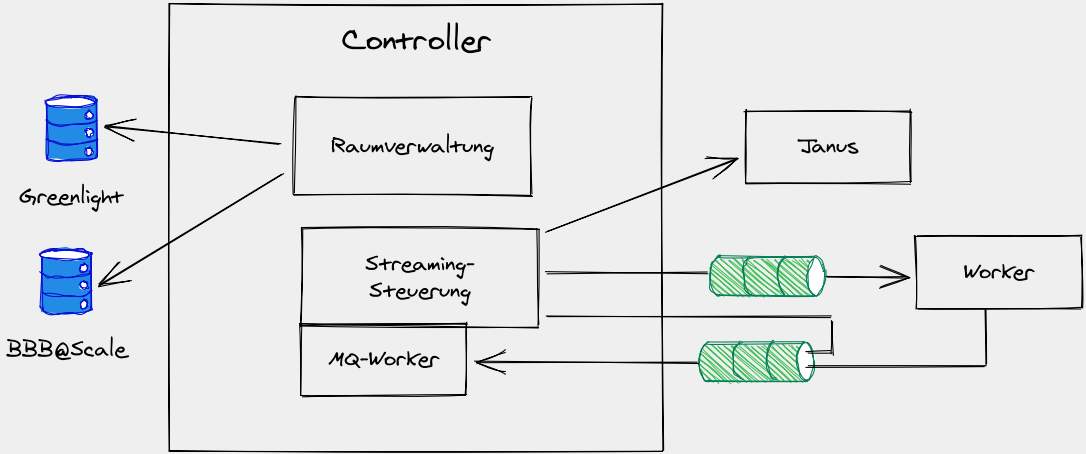
Worker



Controller

- Django-App
- Greift auf eine Raumdatenbank zu
  - Abstraktes Interface
  - Raume und Berechtigungen z.B. aus Greenlight
- Steuert den Janus und die Worker

Controller


Controller
Raumverwaltung
Streaming-
Steuerung
MQ-Worker
Janus
Worker
Greenlight
BBB@Scale

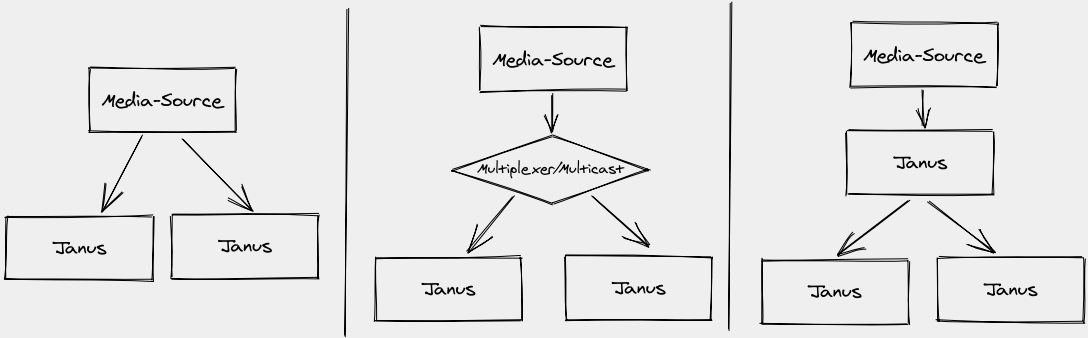
Janus

- WebRTC Mediagateway
- Pluginarchitektur

→ Streaming-Plugin

- Kann Streams ohne Unterbrechung wechseln
- Unterstützt Video-Simulcast, SVC (Scaleable-Video-Coding)
  - → 3 Auflösungen in 1 Stream
- Skalierbar
  - → RTP-Multiplexing oder Multicast
  - → Mehrstufige Janus-Architektur (VideoRoom-Plugin)

Scaling Janus



Latenz

- 30 FPS $\rightarrow$ alle 33ms 1 Bild
- Video Encoder
- GStreamer Queue
- Netzübertragung Worker $\rightarrow$ Janus
- Verarbeitung Janus
- Netzübertragung Janus $\rightarrow$ Publikum
- $\sum$ ca. 0.5s

Ressourcenbedarf

- Janus
  - pro 500 Streams in 1080p 8 CPU Kerne zu 60-80% ausgelastet
  - 500 Streams 2.5 GBit/s peak
  - skaliert linear mit Anzahl Streams
- Worker
  - 4-Core Haswell schafft das mit VP8, 3 Auflösungen
  - Epyc 7301 benötigt dafür 6 Kerne
  - Chrome benötigt davon die Hälfte

What the f**k!?!

Effekt

- Auf dem Desktop ist alles OK, im Container gibt es Störungen
- im Container ist Audio/Video asynchron

Lösung

GStreamer sucht sich eine geeignete Clock. Beim Desktop ist das Pulseaudio. Im Container muss man explizit die Systemclock setzen

2. WTF!?!

Effekt

- Streaming funktioniert überall. Außer an Hybridanschlüssen der Telekom
- Video geht manchmal ein paar Sekunden
- Video geht nicht, wenn Hybrid LTE zuschaltet

Lösung

Bei Hybrid hat LTE eine andere MTU als DSL. GStreamer verwendet MTU=1400 als Voreinstellung

3. WTF!?!

Effekt

- Ab und zu gibt es Audioaussetzer.
- Video in 540p und 720p läuft flüssig. 1080p hat manchmal komische Aussetzer
- Wenn die Folie sehr viel Schrift hat, gibt es mehr Videoaussetzer

Lösung

UDP Buffer im Kernel sind übergelaufen. Immer dann, wenn Keyframes geschickt wurden.

Apple!!!1elf

- Safari ist der Internet Explorer der Neuzeit
- auf iOS gibt es nur Safari als Browser
- Kein VP9, kein Scalable Video Coding

Fazit

Ob das wohl in der Praxis funktioniert? :-/

Referenzen

- https://gitlab.com/bbb-streaming
- https://janus.conf.meetecho.com