Institut für Betriebssysteme
und Rechnerverbund

KaaS: Kernel as a Service

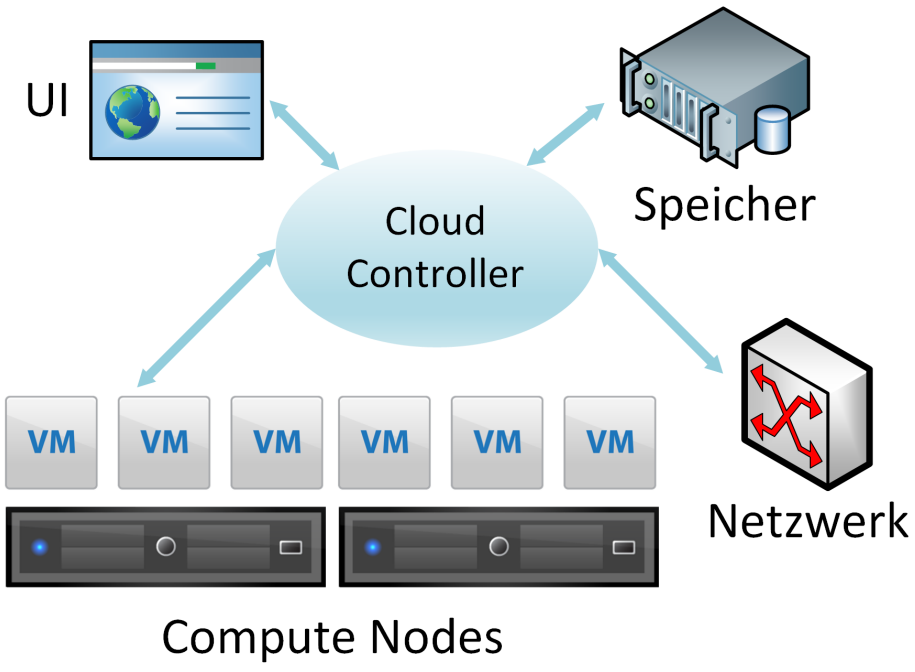
Maßschneiderung von Kernel für Infrastruktur-Clouds

Nico Weichbrodt, 21.11.2013

Technische Universität Braunschweig, IBR

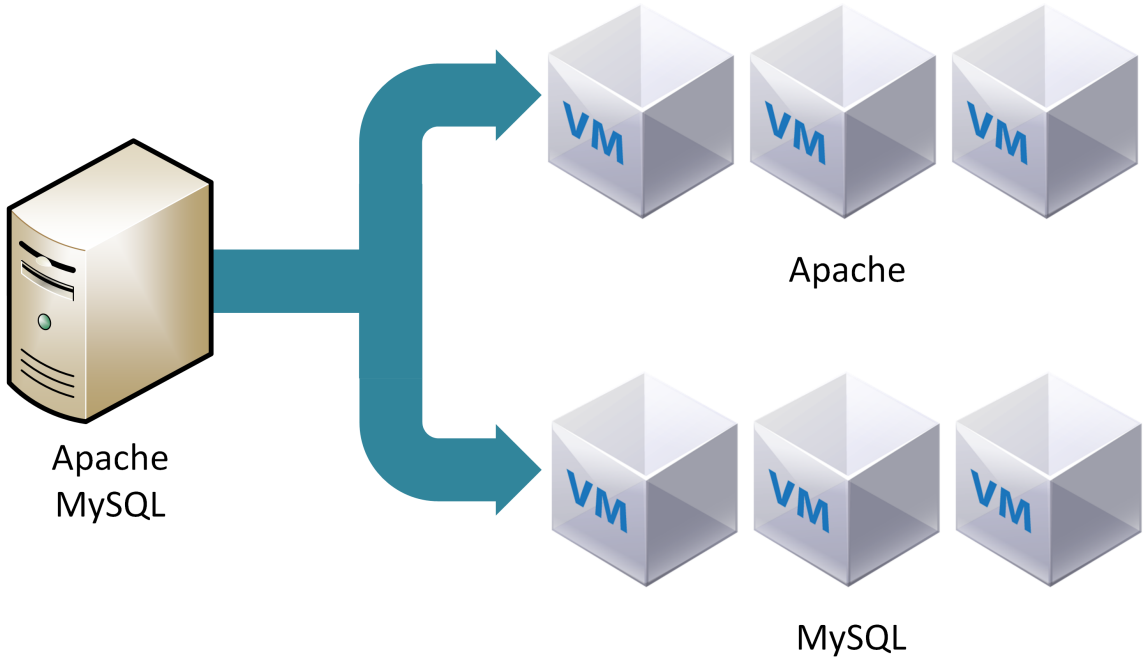
Motivation

- Stand: IaaS Clouds sind heutzutage nicht mehr wegzudenken
- Immer mehr Dienste werden in die Cloud verlagert



Motivation

- Dienste in Cloud einfacher skalierbar
- aktueller Trend: spezialisierte VMs die nur einen Dienst anbieten
  - Cache, Webserver, Datenbank, ...

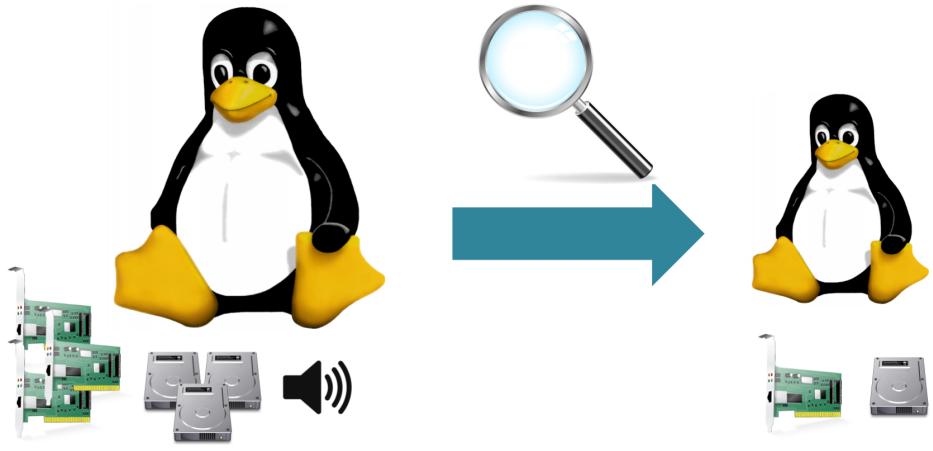


Motivation

- Standard OS-Images werden gerne genommen
- OS-Images/Kernel möglichst mit allem kompatibel
  - viele Gerätetreiber, Soundausgabe, ...
- Folge: Großteil der Kernelfeatures werden überhaupt nicht genutzt!
- unbenutzte Kernelfeatures können Sicherheitslücken beinhalten

Idee

- Idee: alle unbenutzten Features entfernen
- Vermessen des Kernels, Reduktion aufs nötige
- Folge: viel kleinerer, spezialisierter Kernel
- Tool: undertaker
- Reduktion der Angriffsfläche von 50% bis 85%[1]



[1] Attack Surface Metrics and Automated Compile-Time OS Kernel Tailoring, Kurmus et al., NDSS 2013

Kernel as a Service

- Immer mehr Dienste laufen in IaaS Clouds
- Dienstanbieter nehmen bereitgestellte Standard Images
- Kernel selberbauen möglich, aber mühsehlig
  - >10000 Konfigurationsoptionen
  - tieferes Verständnis des Kernels notwendig
  - selbst mit undertaker aufwendig

Kernel as a Service

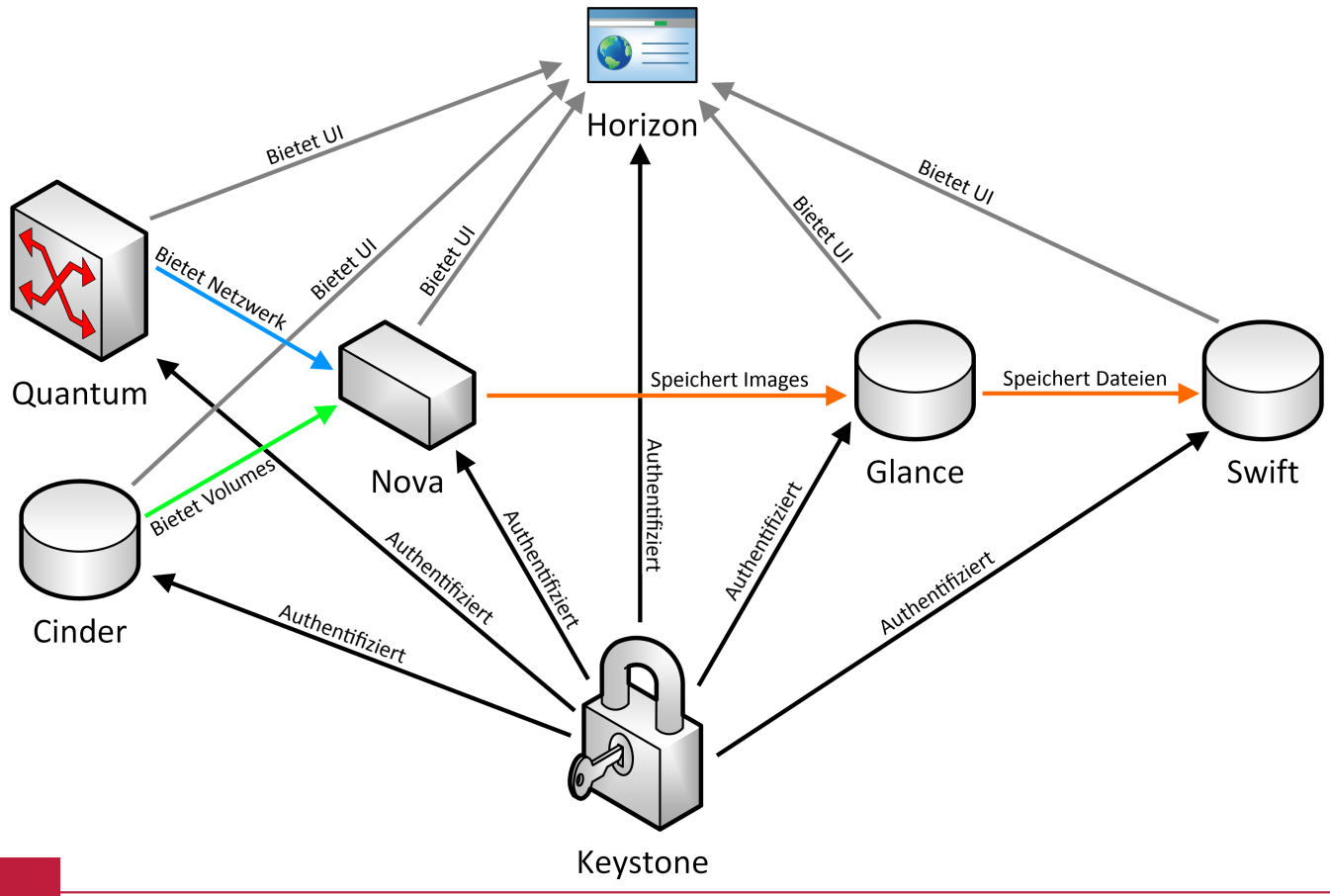
- Lösungsidee: Kernel as a Service (KaaS) mit One-Click Kernelbau
- Vereinfachung des Bauprozesses, für jedermann nutzbar

Inhalt

- **Basistechnologien**
  - OpenStack
  - Undertaker
- **KaaS**
  - Aufbau
  - Workflow
  - Demo
- **Evaluation**
  - memcached-Server
  - Herausforderungen
- **Zusammenfassung**

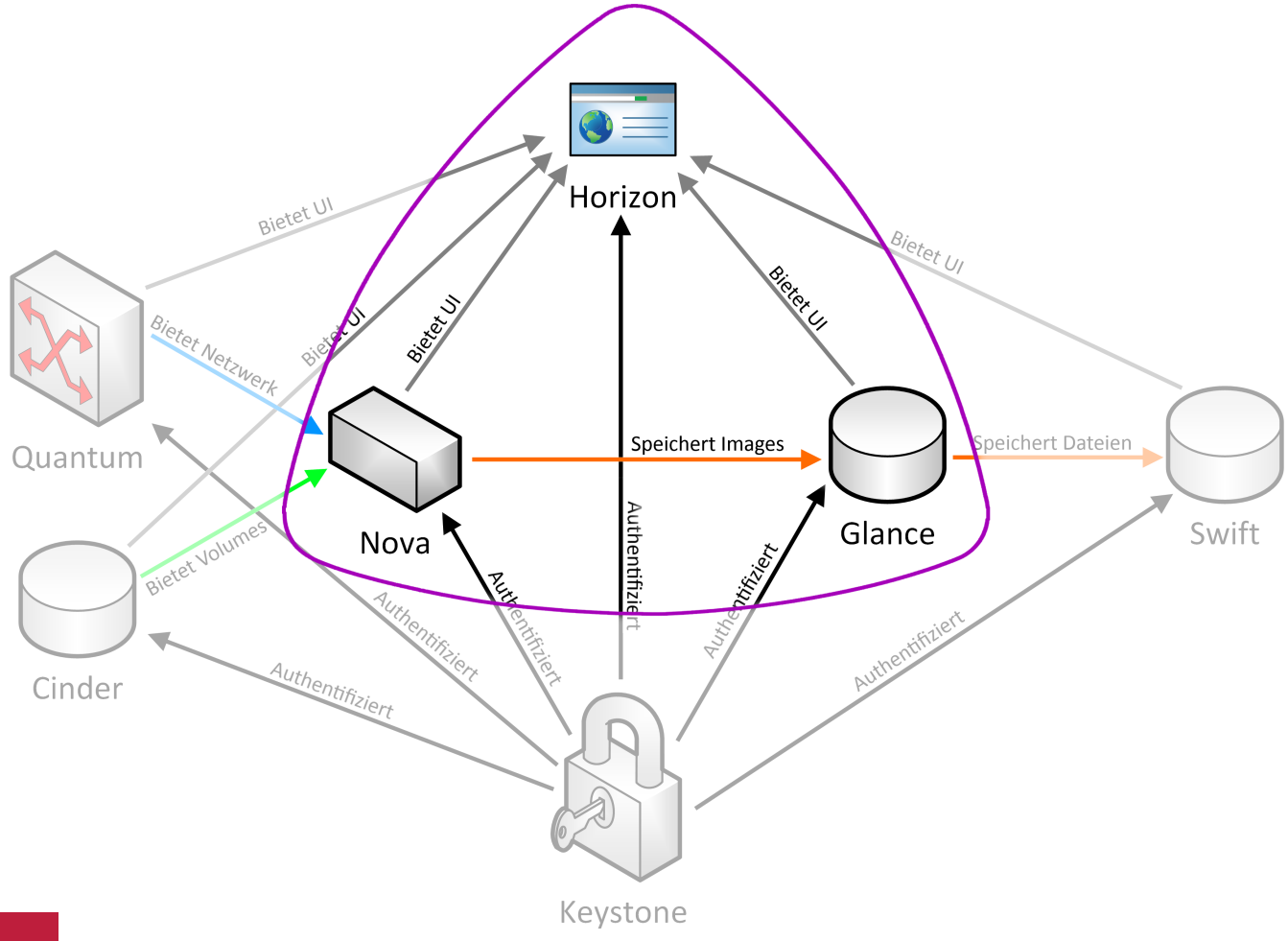
OpenStack

- Opensource IaaS Cloud Software



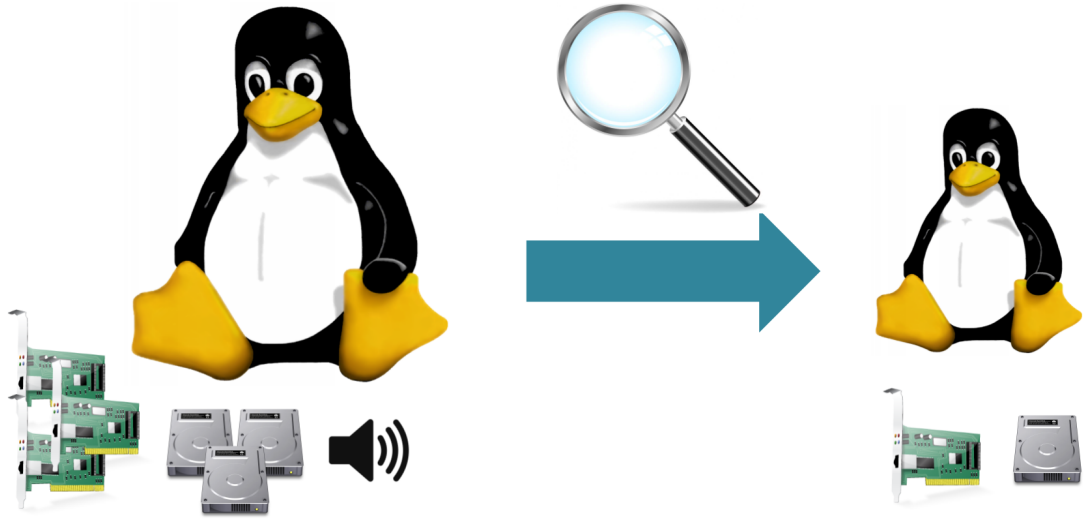
OpenStack

- KaaS: Nova, Glance, Horizon



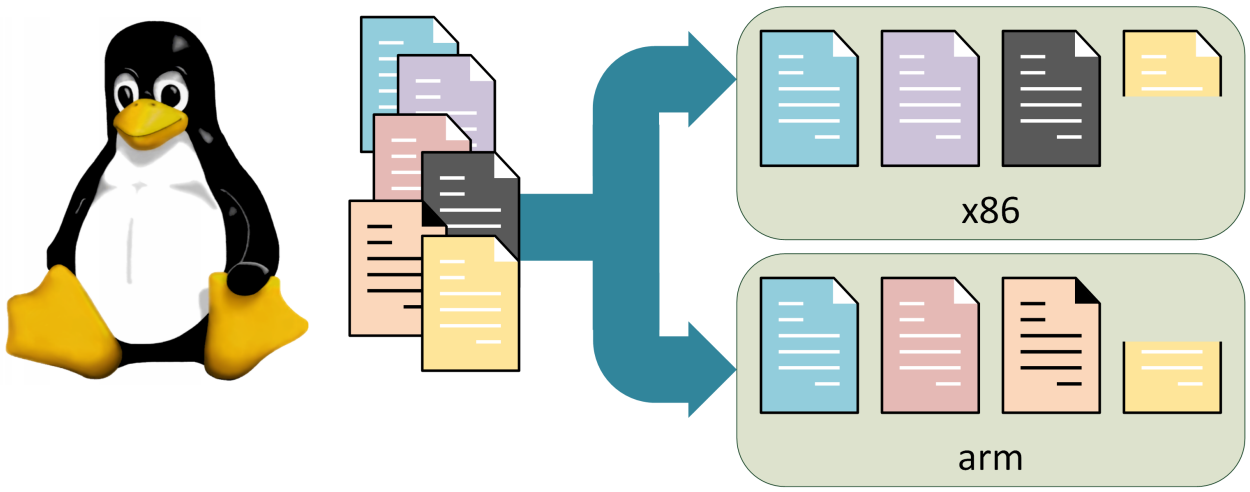
Undertaker

- Programmsammlung zum maßschneidern von Kerneln
- Vermessung des Linux Kernels mit `ftrace`
- Erstellt eine minimale Kernelkonfiguration



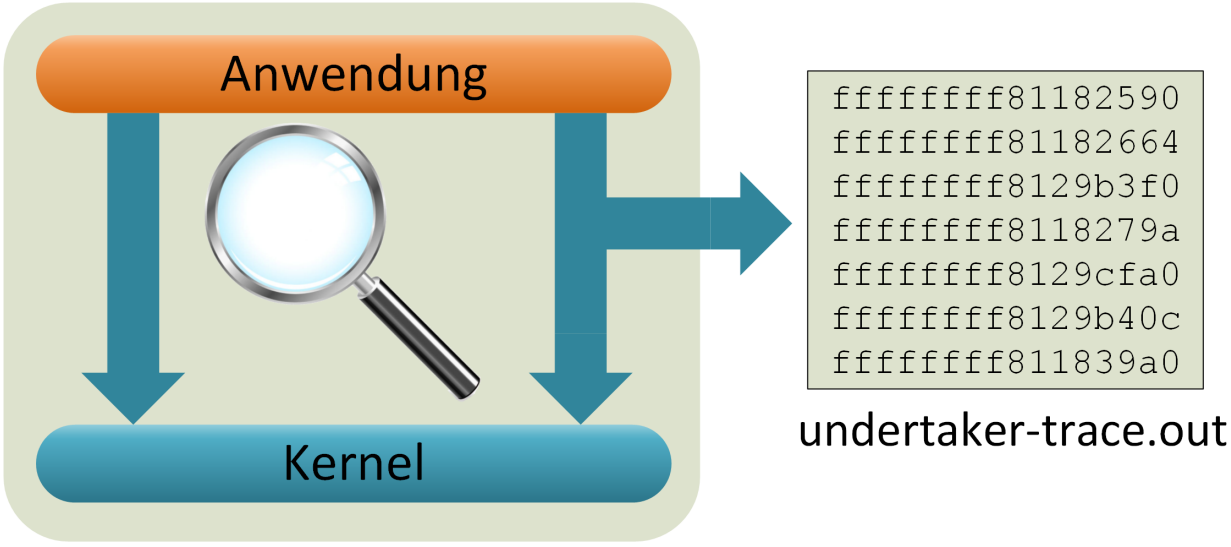
undertaker-kconfigdump

- Erstellt aus den Kernelsources ein Model für eine Architektur
  - x86, ia64, arm, ...
- Model enthält Referenzen zu allen vorhandenen Funktionen
- Model ist Kernelsource abhängig, nicht traceabhängig



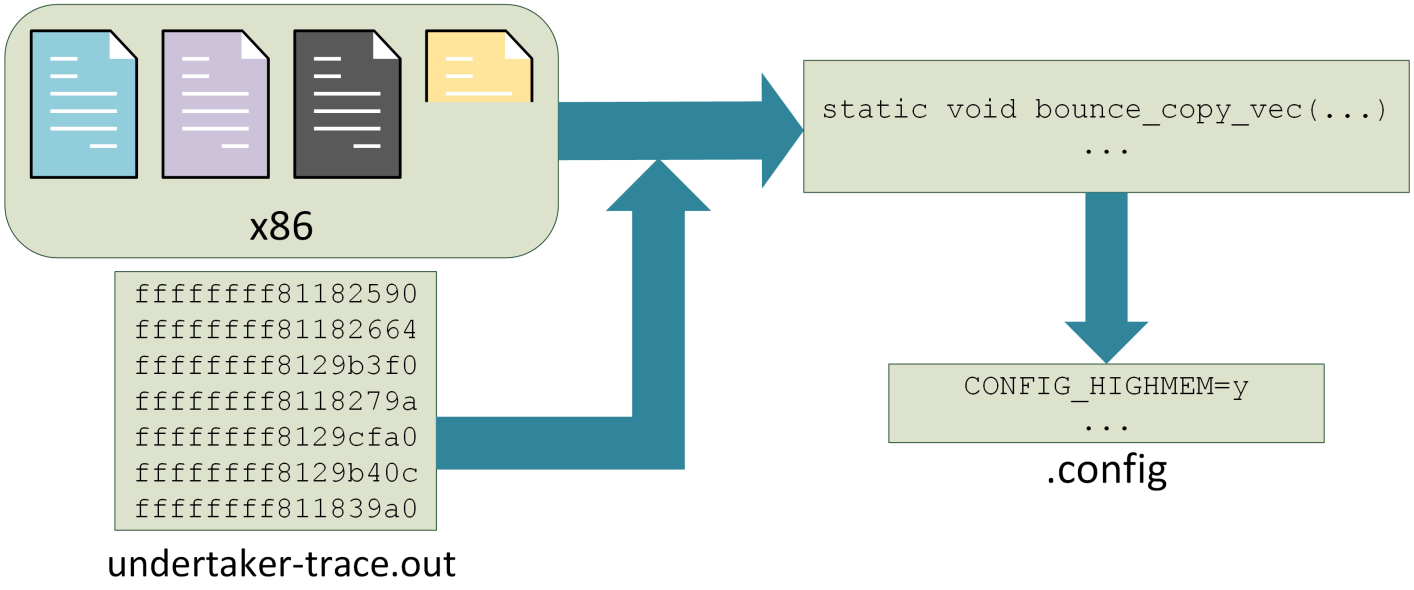
undertaker-tracecontrol

- Startet oder stoppt den Kerneltracer (benutzt ftrace)
- Dient auch zum Erstellen eines modifizierten initrd
  - startet dann den Tracer so früh wie möglich beim Bootvorgang
- undertaker-trace.out mit allen aufgerufenen Kernelfunktionen



undertaker-tailor

- Erstellt aus undertaker-trace.out und Model eine Kernelconfig
- Jede Funktion im Model suchen und einer Sourcefile zuordnen
- Kconfig Optionen setzen, um diese Funktion einzubauen
- Mit der Kconfig kann jetzt ein neuer Kernel gebaut werden



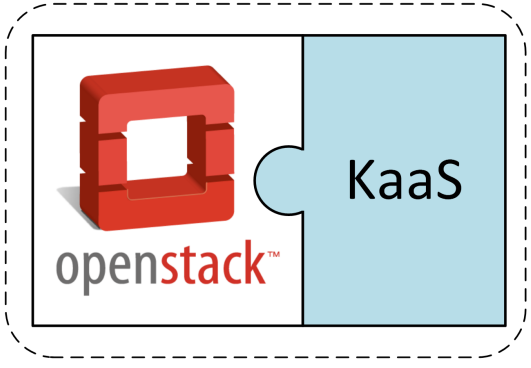
Inhalt

- Basistechnologien
  - OpenStack
  - Undertaker
- **KaaS**
  - Aufbau
  - Workflow
  - Demo
- Evaluation
  - memcached-Server
  - Herausforderungen
- Zusammenfassung

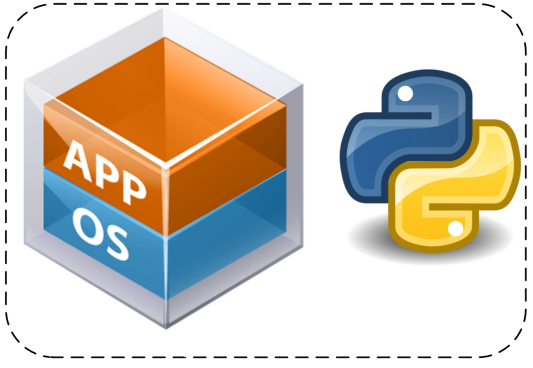
Aufbau

KaaS besteht aus drei Teilen

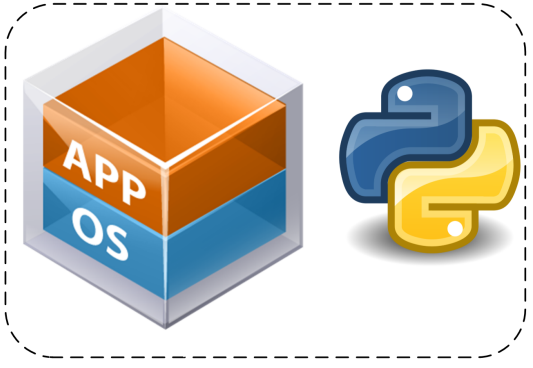
- Horizon Modul/Plugin
- Kunden VM-Image
- Buildserver VM-Image und VMs



Horizon Plugin



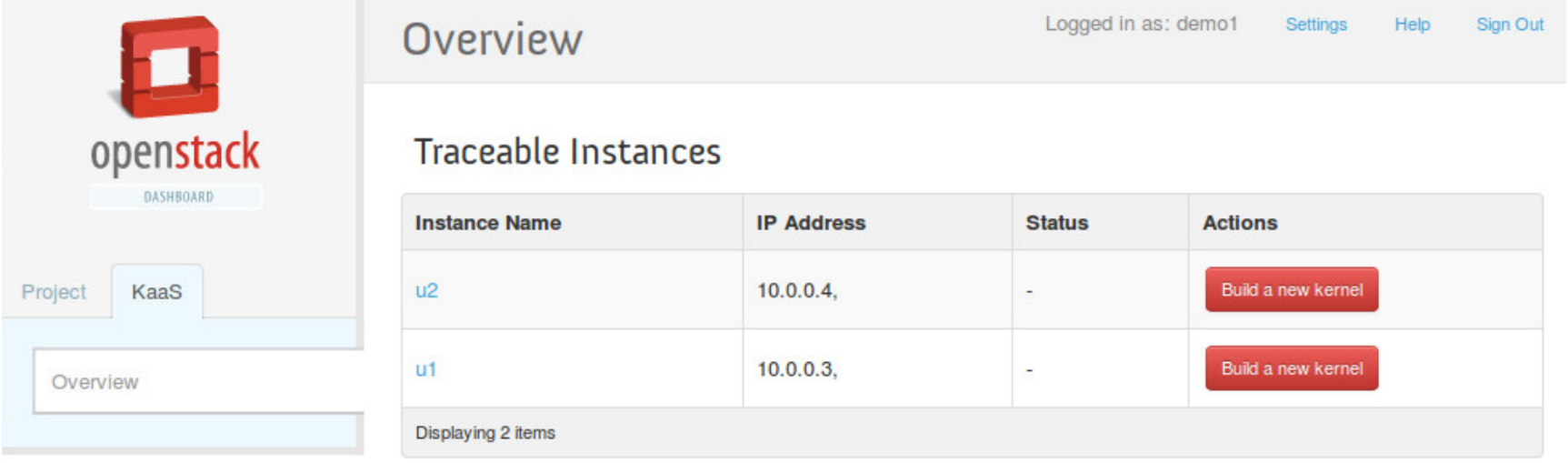
Buildserver VM



Kunden VM

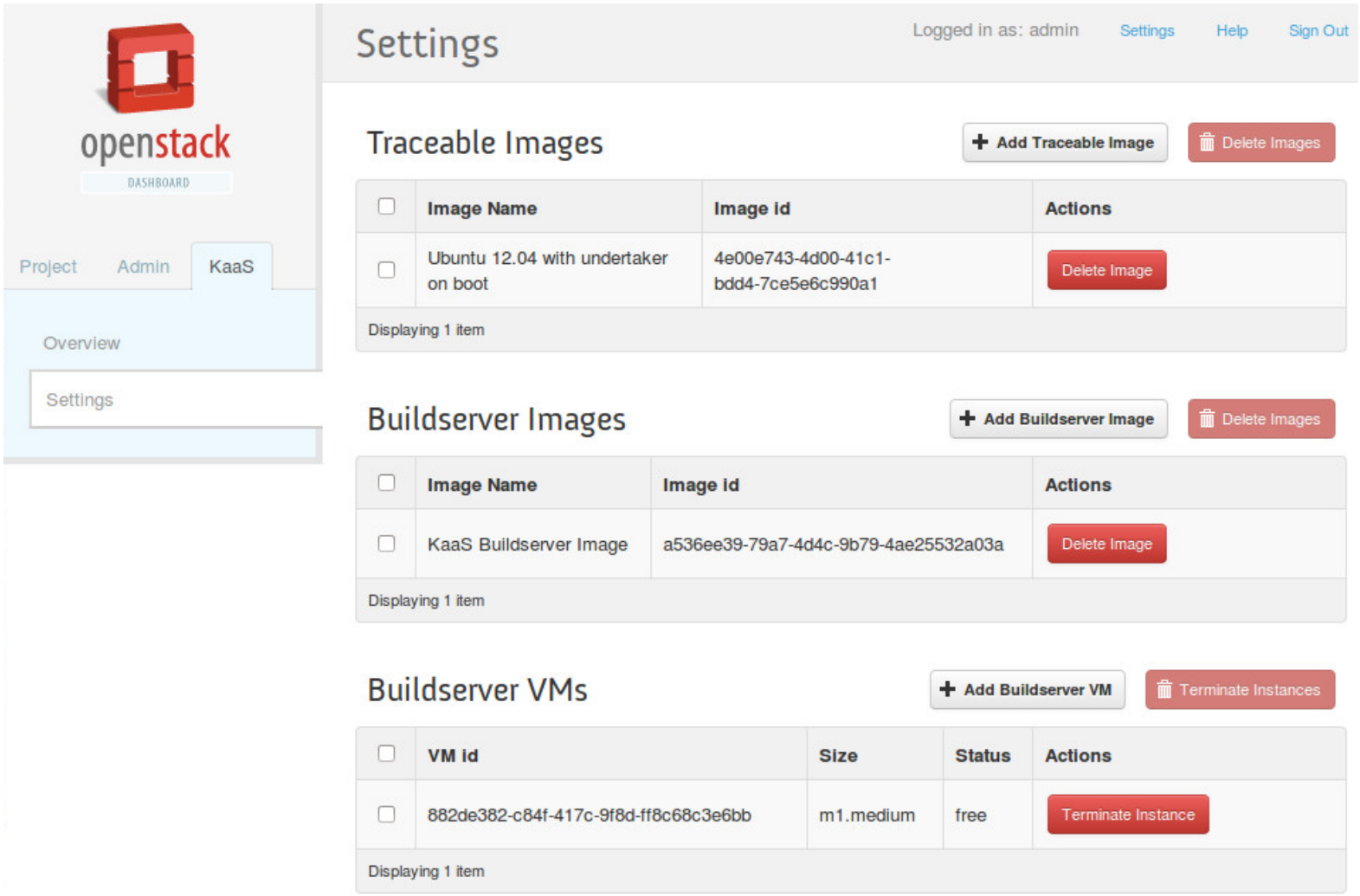
Horizon Modul I

- Einfaches Interface in Horizon um Kernel zu bauen
- Für User: One-Click Kernel bauen inkl. Status



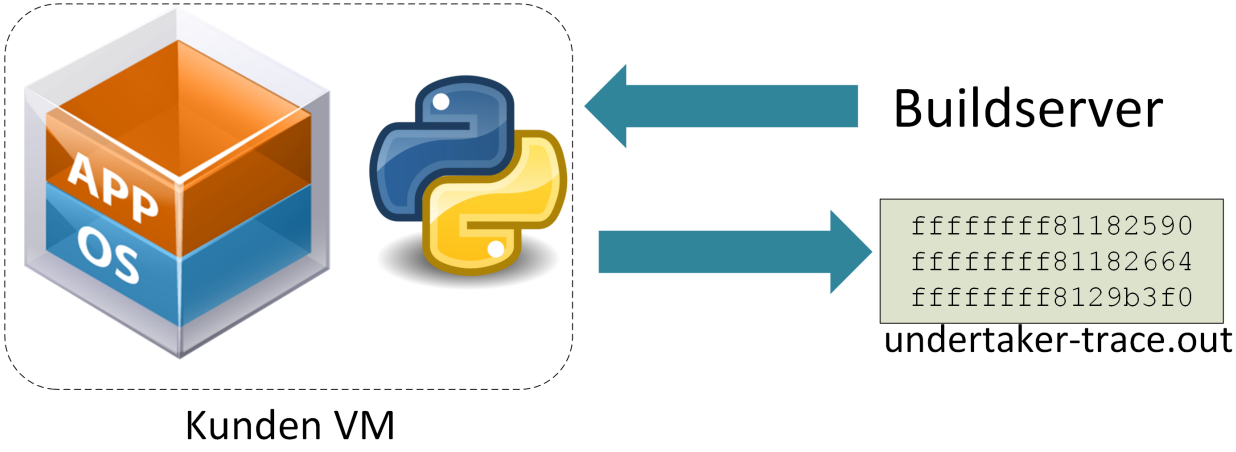
Horizon Modul II

- Für Administratoren: Verwalten von tracebaren Images, Buildserverimages und VMs



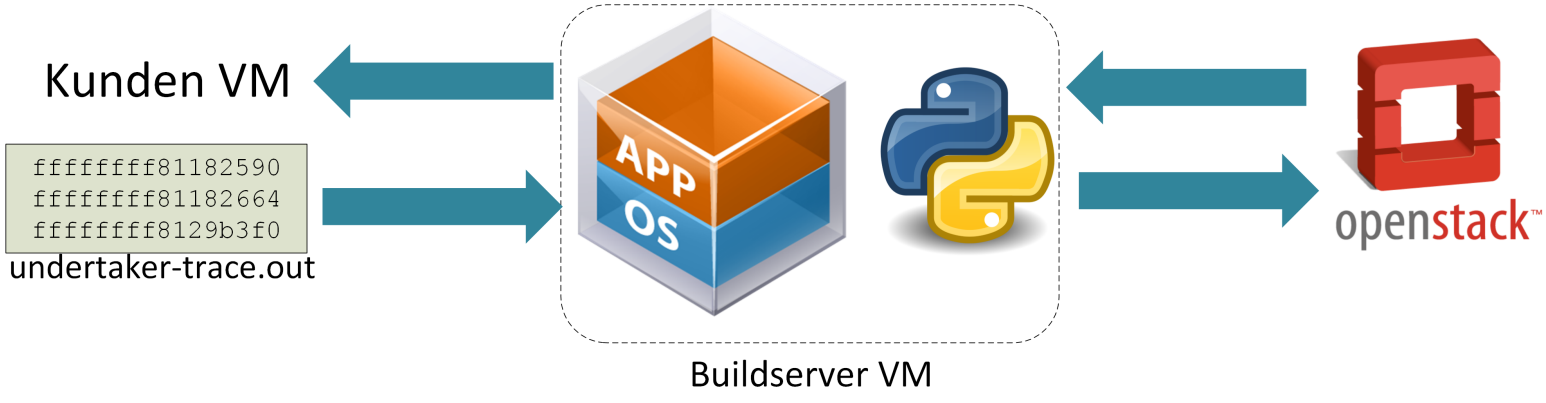
Kunden VM-Image

- Basiert auf einem Ubuntu Cloud Image
- Veränderungen: installierter undertaker, KaaS Python Trace Server
- Tracer startet automatisch beim Boot
- Python Server wartet auf Befehle
  - Stoppt Trace und sendet Tracefile zurück

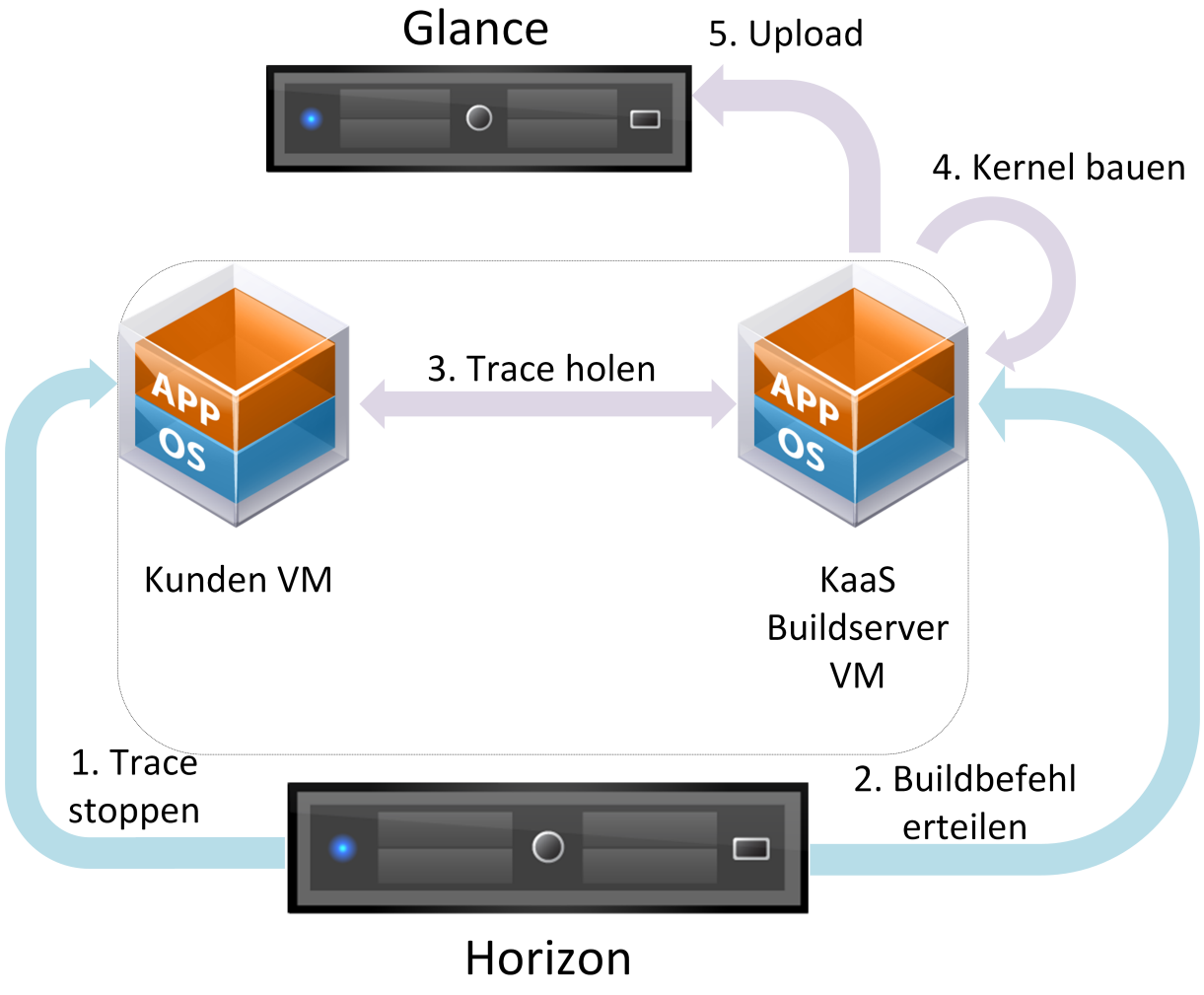


Buildserver VM-Image

- Basiert auf einem Ubuntu Cloud Image
- Veränderungen: installierter undertaker, KaaS Python Build Server
- Enthält auch ein komplettes Ubuntu Image
- Python Server wartet auf Befehle
  - Empfängt Informationen über Kernelbauanfragen

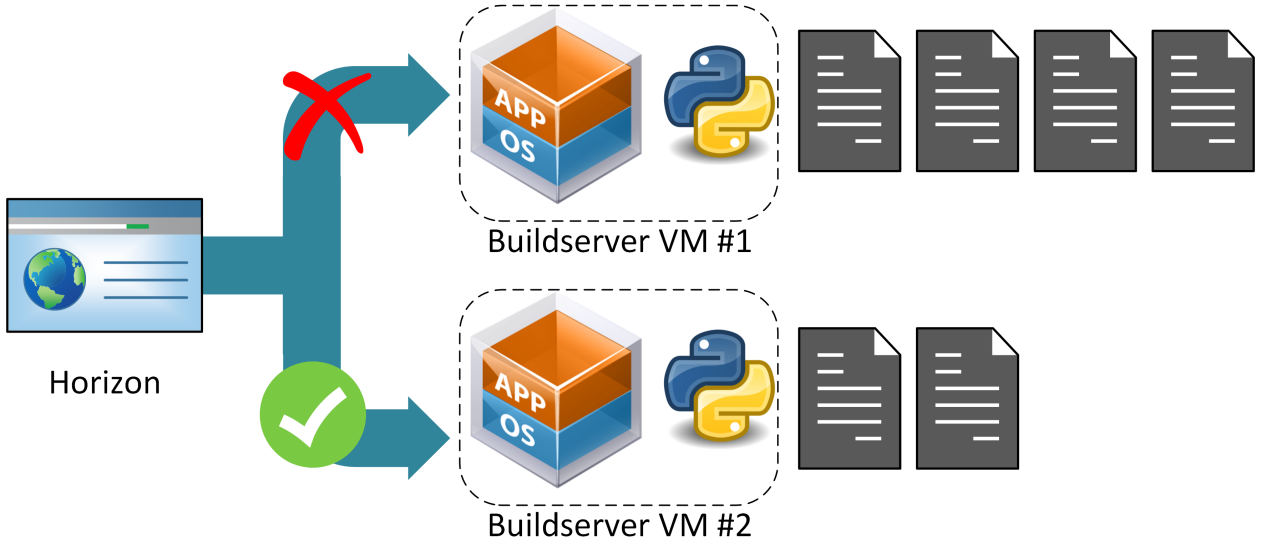


Hinter den Kulissen



Buildserver VMs

- Fester Pool an Buildserver VMs
- Admins können Buildserver hinzufügen oder entfernen
- Für neue Kernelbuilds immer den Buildserver mit kleinster Auslastung wählen
- Buildserver kann beliebig viele Kernelbuilds in seine Warteschlange aufnehmen



Workflow I

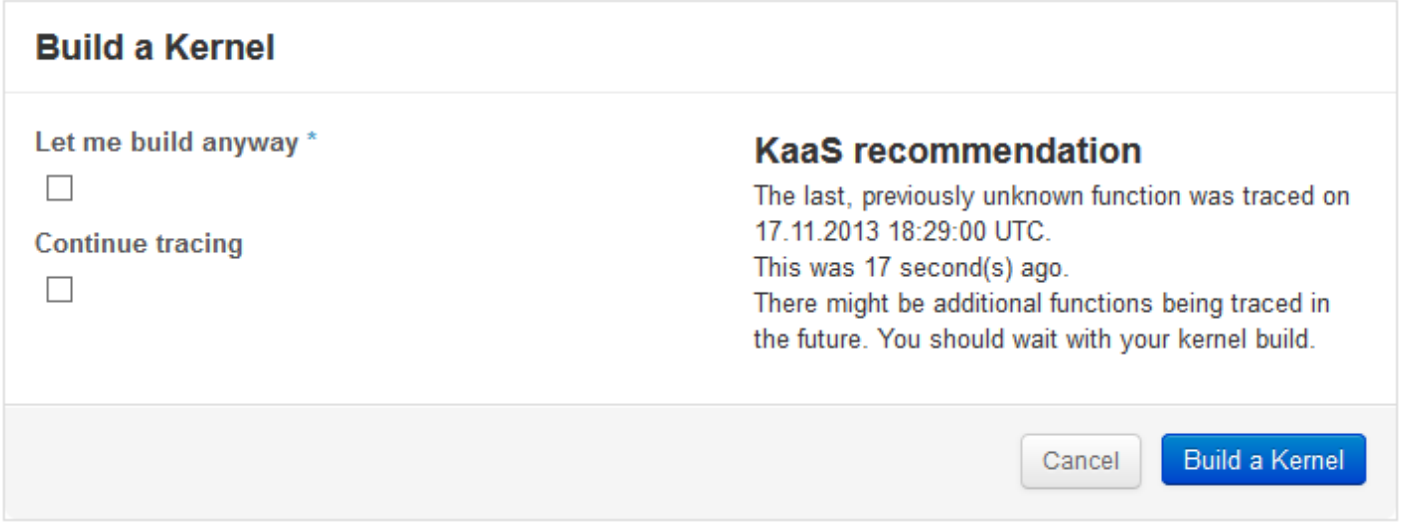
Für Benutzer heißt das also:

- In Horizon einloggen
- Ein tracebares Ubuntu Image starten
- Anwendungen installieren/konfigurieren
- Anwendung unter Last setzen
- Nach Wartezeit kann ein Kernel gebaut werden

Workflow II

Wie lange warten?

- Je länger, desto mehr seltene Funktionsaufrufe werden erfasst
- Generelle Empfehlung: mindestens einen Tag
- KaaS gibt dem Benutzer eine Empfehlung, ob er bauen oder warten sollte



Demo

Demo!

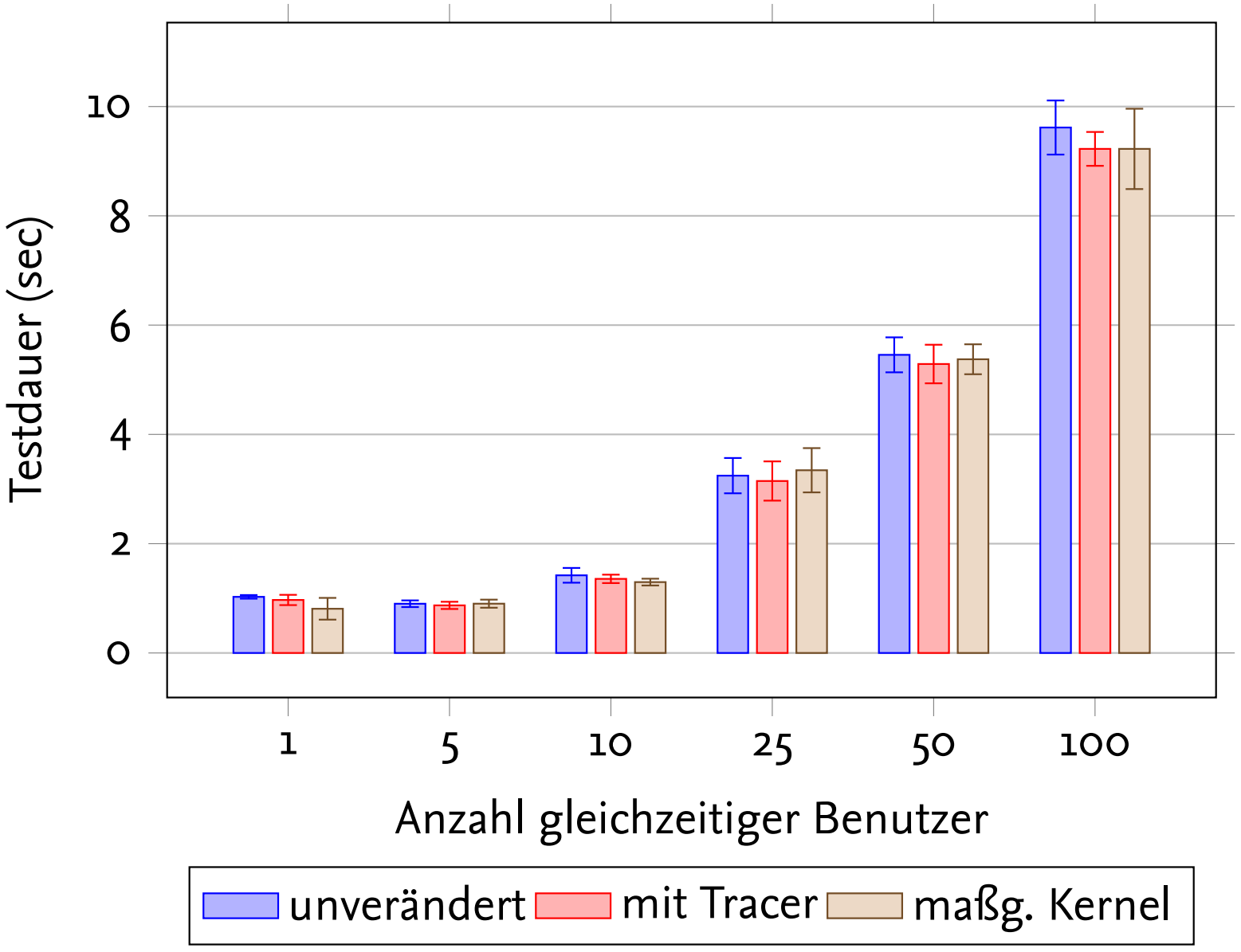
Inhalt

- Basistechnologien
  - OpenStack
  - Undertaker
- KaaS
  - Aufbau
  - Workflow
  - Demo
- **Evaluation**
  - memcached-Server
  - Herausforderungen
- Zusammenfassung

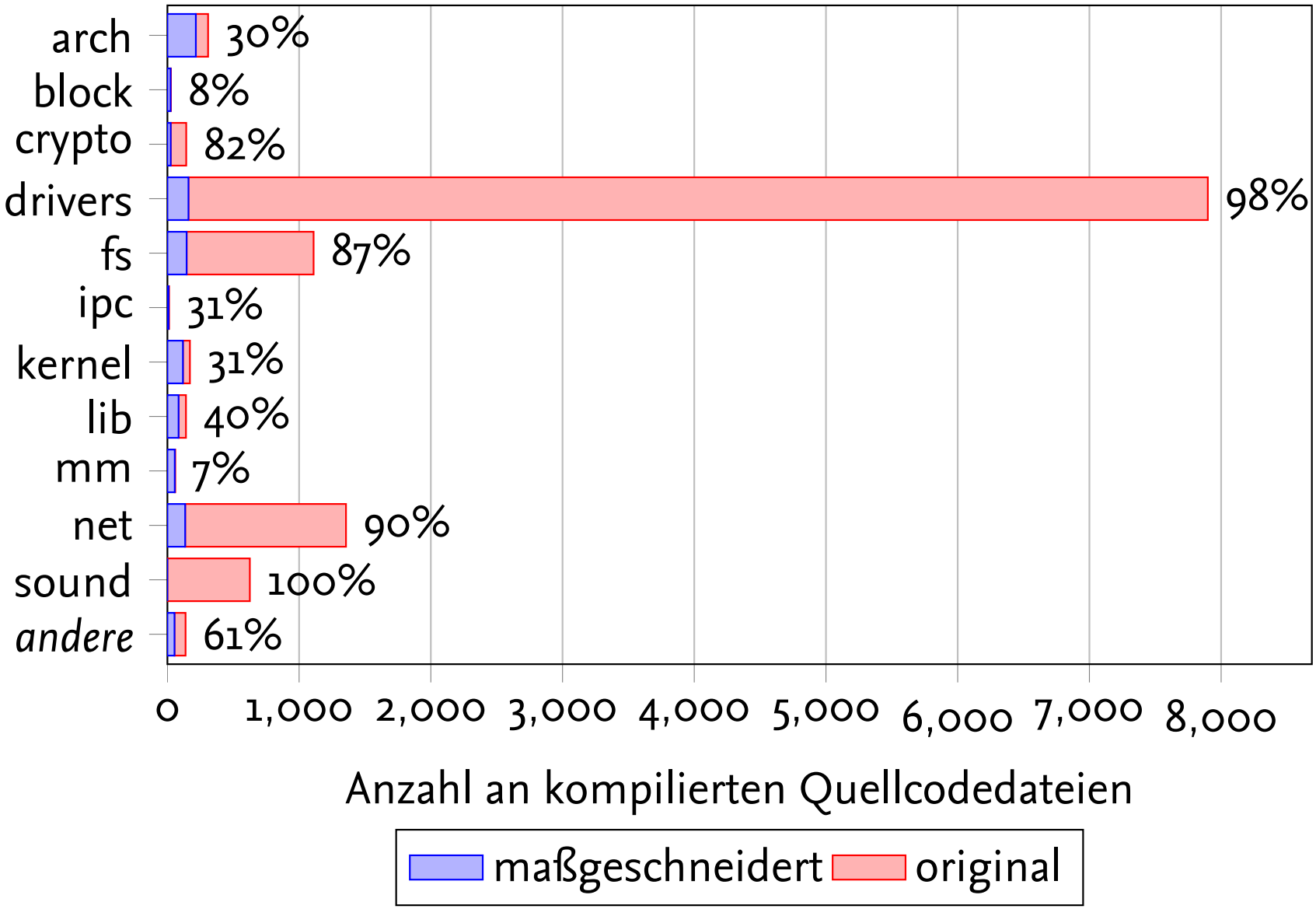
Evaluation

- Kein Leistungsunterschied zwischen gen. und maßg. Kernel
- generischer vs. maßgeschneiderter Kernel
- Anwendung: `memcached`-Server
- VMs: original Ubuntu, Ubuntu mit Tracer, maßg. Ubuntu
- 1 vCPU, 2GB RAM, 20GB Disk (m1.small)
- OpenStack: Ein Host, i7-3770, 16GB RAM, 256GB SSD; DevStack
- Bechmark: `memcslap`, 6 Versuche á 10 Durchläufe

Ergebnis


Testdauer (sec)
0
2
4
6
8
10
1
5
10
25
50
100
Anzahl gleichzeitiger Benutzer
unverändert
mit Tracer
maßg. Kernel

Ergebnis


arch
block
crypto
drivers
fs
ipc
kernel
lib
mm
net
sound
andere
30%
8%
82%
98%
87%
31%
31%
40%
7%
90%
100%
61%
0
1,000
2,000
3,000
4,000
5,000
6,000
7,000
8,000
Anzahl an kompilierten Quellcodedateien
maßgeschneidert
original

Herausforderungen

- Die Python API Doku von OpenStack ist sehr unvollständig
- Trotz Ubuntu 12.04 LTS, sehr alte Versionen der OpenStack Tools
- Falsche Version der Kernelsources führen zu Fehlern in undertaker
- Kernelinstallation in `chroot` nicht einfach möglich

Zusammenfassung

- Standard Kernel enthalten viele Features mit Sicherheitslücken
- Kernel selberbauen umständlich / schwer
- KaaS basiert auf OpenStack Horizon und nutzt undertaker
- Drei Teile: Horizon Modul, Buildserver und Kunden Image
- Einfache Möglichkeit für Endnutzer angepasste Kernel zu bauen
- 50% bis 85% verringerte Angriffsfläche bei gleicher Performance