Inhalt

Abbildungsverzeichnis

Tabellenverzeichnis

Glossar

This document is incomplete. The external file associated with the glossary ‘main’ (which should be called `main.gls`) hasn’t been created.

Check the contents of the file `main.glo`. If it’s empty, that means you haven’t indexed any of your entries in this glossary (using commands like `\gls` or `\glsadd`) so this list can’t be generated. If the file isn’t empty, the document build process hasn’t been completed.

You may need to rerun LaTeX. If you already have, it may be that TeX’s shell escape doesn’t allow you to run makeindex. Check the transcript file `main.log`. If the shell escape is disabled, try one of the following:

- Run the external (Lua) application:

  ```
  makeglossaries-lite "main"
  ```

- Run the external (Perl) application:

  ```
  makeglossaries "main"
  ```

Then rerun LaTeX on this document.

This message will be removed once the problem has been fixed.

Akronyme

This document is incomplete. The external file associated with the glossary ‘acronym’ (which should be called `main.acr`) hasn’t been created.

This has probably happened because there are no entries defined in this glossary. Did you forget to use `type=acronym` when you defined your entries? If you tried to load entries into this glossary with `\loadglsentries` did you remember to use `[acronym]` as the optional argument? If you did, check that the definitions in the file you loaded all had the type set to `\glsdefaulttype`.

This message will be removed once the problem has been fixed.

Intro text

Applikation Dienst

- Ist ein Container Manager

- Kann isolierte Laufzeitumgebungen mit Hilfe von Name Spaces und Pivot Root Methoden

- Kann diese Laufzeitumgebungen verwalten: Erstellen, Löschen, Starten, Stoppen

- Im Gegensatz zu üblichen Containerdiensten ist die Funktionalität auf das Laden und Ausführen von Binaries beschränkt

- Applikationsdiesnt kann über JSON Befehle gesteuert werden

Motivation

- Herausforderungen bei Verteiltem Rechnen auf Fahrzeugen:
  - Resourcenmanagement
  - Netzwerk Konnektivität
  - Sicherheit
  - Privatsphäre
  - Standardisierung
  - Kompatibilität

Stand der Technik

- Cloud Computing
  - : Definition
  - IT resourcen werden flexibel nach Bedarf zur Verfügung bereitgestellt
  - Realisiert durch Rechenzentren, Kunden können Resourcen mieten anstatt eigene Server betreiben
  - Resourcen sind von verschiednen Endgeräten erreichbar [**Sunyaev2020**]
  - Cloud kann öffentlich oder privat sein, privat für sensible Daten [**Ibrahim2021**]
  - Relevante Anwendungsfälle:
- Edge Computing
  - Daten entstehen in Endgeräten, Applikationen, die die Daten verarbeiten sind zunehmend ebenfalls in Endgeräten
  - Beispiel Flugzeuge oder autonome Fahrzeuge, generieren Daten in Größe von mehreren Gb pro Senkunde [**Liu2019**]
  - Traditionell werden Daten in der Cloud ausgewertet und wieder an den Benutzer zur Verfügung gestellt, diese ist aber mit zusätzlichem Resourcenaufwand verbunden wegen lange Übertragungsstrecken. [**Perez2022**]
  - Übertragung und Verarbeitung in der Cloud langsam/unmöglich wegen Bandbreite und Latenz

- Edge Computing ist das Konzept, dass anstatt zentrale Cloud Server, Daten zunehmend auf Endgeräten verarbeitet werden [**Shi2016**]
- Edge bezeichnet Geräte zwischen Datenquellen und cloud server
- “a form of distributed computing in which processing and storage takes place on a set of networked machines which are near the edge, where the nearness is defined by the system’s requirements” (ISO/IEC: Tr 30164:2020 - internet of things (iot) -edge computing. Tech. rep., ISO/IEC (2020))
- Motivation: Latenzreduzierung, unbenutzte Resourcen verwenden
- Anwendungsfälle:
  * Cloud Berechnungen auslagern
  * Smart Home, Daten lokal auswerten statt alles in die Cloud laden
  * Smart City
- Cloudlets
- Mobile Edge Computing
- Edge Computing Gateway
- Fog Computing
  * Fog Computing: Eine Form von Edge Computing, zusätzliche Schicht zwischen Endgeräte und Cloud
  * Edge Geräte kommunizieren mit Fog nodes, die wiederrum für die Kommunikation zwishen Edge und Cloud zuständig sind
  * Fog nodes übernehmen Datenverarbeitung lokal, für nur Lokal benötigte Daten, Leiten nur relevante Daten an Cloud weiter
  * Fog nodes können PC-s, raspberry PI-s, nano-server, micro-datenzentren sein. [**Mahmud2020**]
  * Beispiele:
    · Iot for All
    · FogFlow
- Mist Computing

    * Datenverarbeitung direkt im Sensor.
    * Erlaubt z.b. einfache Monitoringfunktionen direkt im Sensor
    * Reduktion von benörigte Bandbreite und Rechenleistung in den übergeordneten Geräten
  - Distributed Cloud
- Internet of Things Internet of Things (IoT)
  - Internet of Things (Internet der Dinge) bezeichnet eindeutig identifizierbare Objekte und deren virtuelle repräsentation in interent-ähnliche Strukturen[**Jie2013**]
  - IoT findet mittlerweile in fast allen Bereichen Anwendung
  - Industrie: Produktionsanlagen, die durch vernetzte Systeme flexibel angepasst werden können (Beispiel: Produktvarianten)
  - Smart Home: Sensoren und Aktoren im Haus vernetzen [**Wortmann2015**]
- Security: [**Mahmud2020**]
  - Container: [**Costa2022**]
    * Abstraktionsebene zwischen Prozesse und OS
    * Packt die Prozesse und deren Abhängigkeiten zusammen so dass sie einfach auf andere Systeme portiert werden können
    * definiert schnittstelle
    * Läuft als Teil des Betriebssystems
    * Software kann auch ”bare metalöder in hypervisor virtuelle machine ausgeführt werden
  - Hypervisor:
    * Virtuelle Machine manager
    * Jede Anwendung läuft im eigenen virtuellen OS (guest)
    * Anwendung hat ggf. keine Information darüber dass es in virtuellen Umgebung läuft
    * jede Anwendung läuft getrennt in eigenem Runtime Umgebung

- Kommunikation
  - Punkt zu Punkt
  - Client-Server
  - Remote Procedure Call
  - Message oriented Middleware
  - Direct Data Access
  - Peer to Peer
  - Publish/Subscribe:
    * Modell für Nachrichtenbasierte Kommunikation [**MadeWirawan2018**]
    * Ereignisbasierte Kommunikation
    * Teilnehmer kommunizieren indem sie Nachrichten mit bestimmte Themen veröffentlichen (Publisher), Empfänger können Themen abonnieren (Subscriber) und bekommen nur die abonnierte Nachrichten
    * Kommunikation ist anonym
    * Komponenten: Publisher, Broker, Subscriber
    * Broker stellt Verbindungen zwischen Publisher und Subscriber her
    * Broker speichert die Subscriptions (Abos)
    * Weit verbreitete Implementierung: MQTT
    * RTPS:
      · Real Time Publish Subscribe
      · Kommunikation erfolgt ebenfalls über Themen
      · Kein Broker, Teilnehmeridentifizierung erfolgt zur Laufzeit dezentralisiert
      · Quality of Service parameter: Reliable writer speichert Sequenznummer der Nachrichten und kann erneut versenden bei Übertraugungsfehler, Best effort Writer hat diese Funktionalität nicht
      · Heartbeat message: writer gibt die verfügbaren nachrichten ID-s and

        - AckNack message: kommunikation von empfangenen und nicht erhaltenen Nachrichten

- Resourcenmanagement:
    - t
- Softwarearchitektur
-
- Kommunikation
    - Zertifikate (Public/Private Key)
    - identitätsverschlüsselung
    - Belohnung für bereitgestellte Rechenleistung
- Resourcenverteilung
    - Bestimmung der verfügbaren Rechenleistung
    - Optimierungsalgorythmen
    - Stackelberg Model
- Publish Subscribe
    -
    - Optimierungsalgorythmen
    - Stackelberg Model
- Softwarearchitektur in Fahrzeugen
    - RTOS
    - Middle Layer (ROS, keine automotive alternative stand 2019)
    - Cloud

- https://www.eprosima.com/index.php?option=com$_a rsview = browseslayout = normal$
    - Stand der Technik
        - Anwendungsfälle für Fahrzeuge:

    - Unbenutzte Rechenleistung von Hardware nutzen, autonome Fahrzeugen haben leistungsfähige Steuergeräte
    - Auslagerung von Rechenaufgaben auf Fahrzeuge in der Umgebung mit freier Rechenkapazitäten
    - Ausnutzung von Rechenleistung parkende Fahrzeuge
    - Möglichkeit, die unbenutzte Rechenleistung zu Vermieten (wie cloud service)

- Konzept UNICARagil:
  - Modulare Systemarchitektur in allen Domänen
  - Fahrzeuge können flexibel auf verschiedene Anwendungen angepasst werden: Taxi, Privatfahrzeug, Shuttle, Cargo
  - gemeinsamer mechanischer Platform
  - Sensoren und Aktoren mit definierte Schnittstellen
  - Alle Fahrzeuge vernetzt: untereinander, zur Cloud, Infobiene, Benutzer
- Motivation:
  - Zunehmende digitalisierung in Arbeits- und Privatumfeld
  - Cloud Dienste bereits sehr verbreitet
  - steigende Zahl an vernetzte Endgeräte
  - Daten entstehen an Endgeräten, verarbeitete Daten werden ebenfalls an Endgeräten benötigt
  - Cloud basierte ansätze erfordern immer höhere Bandbreite und zentralisierte Rechenleistung
  - Endgeräte haben immer höhere Rechenleistung, die oft unbenutzt bleibt
  - So auch in Fahrzeugen, die wegen autonome Fahrfunktionen deutlich mehr rechenleistung bekommen
  - Ungenutze Rechenleistung kann für externe Anwendungen zur verfügung gestellt werden
  - Dezentralisierte Rechenleistung verringert physikalische Distanz zwischen Entstehung und Verarbeitung der Daten

- Einsparung an Bandbreite und zusätzliche Server Rechenleistung

- Konzept: [**Mahmud2020**]
  - Applikation Architektur:
    - monolithische Architektur
      - Applikation beinhaltet alle operationen
      - die gleiche Applikation kann mehrfach ausgeführt werden für parallele berechnungen
    - Verteilt:
      - Modulbasiert:
      - Applikation auf Module aufgeteilt, abhängig voneinander
      - Bedienen Daten von einer Quelle
      - partitionierung von Applikationen [**Ashraf2022**]
      - Micro-Services:
      - Applikation setzt sich aus unabhängigen Prozessen zusammen
      - Ein Microservice erfüllt nur eine Aufgabe
  - Applikation Platzierung: [**Mahmud2020**]
    - Bestimmung verfügbare Rechenresourcen:
      - Profilierung: Applikation wird auf jedem Node ausgeführt, Rechenleistung wird aus Ausführungszeit bestimmt
      - Prädiktiv: Rechenleistung wird aus vergangenen Berechnungen ermittelt
      - Nach Bedarf: Je nach Erwartungen der Benutzer und Anforderungen an die Ausführungszeit wird nach Profil oder Prädiktiv verteilt
    - Offloading Methoden:
      - Bottom-Up: Edge Geräte verlagern Applikationen in die Node Server
      - Top-Down: Applikationen aus der Cloud oder Fog Node werden in Edge Geräte ausgelagert

- Hybrid: Top-Down, Bottom-Up, Edge geräte können Applikationen direkt untereinander austauschen

* Resourcenorientierung -Netzwerkstruktur
  - Hierarchie: Fog node server sind in Hierarchieebenen eingeteilt. Anzahl der Nodes auf einer Ebene nimmt nach unten hin zu.
  - Cluster: Fog nodes sind alle untereinander verbunden. Edge geräte werden in jeweilige Cluster gruppiert. Bessere horizontale Skalierung als bei Hierarchie.
  - Client-Server: Einige Fog Nodes funktionieren als Server, die anderen als client. Client nodes leiten ihre daten an den server weiter.
  - Master-Slave: Master node verteilt daten an slave nodes. Verwaltet die slave nodes. Master node kommuniziert die Ergebnisse.
* Mapping:
  - Prioritätsbasiert: Priorisiert app platzierung auf bestimmte fog nodes, meistens heuristische methoden (best fit, first fit).
  - Optimierung: Kostenfunktion für die optimale Aufteilung erstellen und optimieren
  - multi-objective trade-off: nach mehreren Anforderungen gleichzeitig optimieren wie Energieverbrauch, verfügbarkeit, kosten
* Platzierung
  - Statisch: Applikation wird einmal für jede Applikation platziert
  - dynamischen: Mehrere Instanzen einer Applikation können ausgeführt werden oder Applikation wird nur für eine Berechnung ausgeführt
  - Ereignisbasiert: Applikation wird zur Laufzeit umgezogen
* Ausführungsumgebung:
  - Bare metal
  - Virtuelle machine
  - container: [**Costa2022**]

        - weniger overhead als virtuelle maschine
        - verschiedene runtimes bereits verfügbar: Docker
        - Orchestrierungsmöglichkeiten verfügbar: Kubernetes, Docker Swarm, Nomad, Marathon, Mesos
        - Kontainerorchestrator ermöglicht Lebenszyklus, Skalierung, selbsthailung, migrierung,
        - Docker als verbreitete Implmenetierung
    * Platzierungskriterien:
        - Quality of Service
        - Service level Agreement
        - Zeit und deadline
        - Profit
        - User-erfahrung
        - Kosten
        - externe Vorgaben
        - Energie
        - verfügbarkeit
        - Mobilität
- Applikationsverwaltung: [**Mahmud2020**]
    * Fog Netzwerkverwaltung: [**Costa2022**]
        - zentralisiert: fog orchestrator an zentraler Stelle. Hat gesamtüberblick über das Netzwerk
        - dezentralisiert:
        - Verteilt:
    * Safety:
        - Datenintegrität
        - Verschlüsselung

        - Authentifizierung
    * Leistungsüberwachung
        - Implikationbasiert:
        - Grenzwertbasiert:
    * Finanzielle Unterstützung
        - Kompensation: Bei Serviceausfall werden benutzer kompensiert
        - Anreize: Verfügbare resourcen werden bei Bedarf zur verfügung gestellt, besitzer der Resourcen werden vergütet
        - Reservierung: Verfügbare rechenleistung kann reserviert werden damit Verfügbarkeit garantiert wird
    * resillienz:
        - Backup-Restore
        - Verfielfachung
        - Operator Migration

- Anforderungen:
    - Verfügbare Rechenleistung für externe Kunden zugänglich machen
    - Kommunikationsmöglichkeit bieten, auch für parallele Berechnungen
    - Resourcenmanagement
    - Netwerküberwachung
    - Quality of Service sicherstellen
    - Lebenszyklus verwalten
    - Sicherheitsmaßnahmen vorsehen
    - Buchhaltung über Angefragte und gelieferte Rechenleistung
    - Verwaltungsdienst:
        * Kommunikationsschnittstelle für den Kunden
        * Erkennung von Teilnehmern (edge devices)

        * Informationen über die Fähigkeiten der Teilnehmer sammeln (Architektur, Rechenleistung, Verfügbarkeit, Latenz)
        * Ermittung einer optimalen Verteilung von Kundenaufgaben an Teilnehmer
        * Verteilung von Kundenapplikationen an geeignete Teilnehmer
        * Kommunikationsschnittstelle zu den verteilten Kundenapps
        * Kommunikationsschnittstelle für den Kunden
        * Datenbank über Angeforderte und geleistete Berechnungen
        * Vergütungsausschüttung nach Leistung
    - Loader Applikation:
        * Meldet verfügbarkeit an Verwaltungsdienst
        * Meldet Architektur, Rechenleistung, Verfügbarkeit an Verwaltungsdienst
        * Bietet Kommunikationsschnittstelle für die Übertraung von komppilierten Kundenapps
        * Bietet Kommunikationsschnittstelle für die Kommunikation mit Kunden-applikationen
        * Bietet Kommunikationsschnittstelle für die Verwaltung durch Verwaltungs-dienst
        * Richtet Laufzeitumgebung für Kundenapplikation ein
        * Zugriffsrechte der Kundenapplikation werden eingeschränkt
        * Kann Kundenapplikation starten und beenden
        * Kommuniziert Applikationsstatus an Verwaltung
- Netzwerkstruktur:
    - Autotechagil Struktur gegeben: Cloud: Leitwarte, Edge device: Fahrzeuge, Infobiene
    - Zentraler Struktur bietet sich an: Leitwarte auch Fog Orchestrator
    -
- Container Interface (OCI kompatibilität):

  - Create
  - Start
  - State
  - Kill
  - Delete
  - JSON Varianten:
  -
  -

- Container Parameter:
  - Stack size
  - Distroless
  -