Cloud & Cyber-Physical Applications (Machines, IoT, Robots)

Josef Spillner <josef.spillner@zhaw.ch>
Service Prototyping Lab (blog.zhaiw.ch/icclab)

August 10, 2017 | Parque Tecnológico Itaipu
Background

Compound lecture based on previous work:
- workshop paper „Autonomous Participation in Cloud Services“ (2012)
- new workshop paper submission „Distributed SLAM and Autonomous Exploration on PaaS“ (2017)

But also new material (unreviewed!)
When Cyber (Digital) is not enough

Cyber-Physical Application (CPA): A composite application executed in parallel across physical and virtual spaces.
# CPA-Related Terms and Trends

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Software</th>
<th>Hybrid (CPS)</th>
<th>Hierarchical (CPSoS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robots</td>
<td>Devices</td>
<td>Containers</td>
<td>Functions</td>
</tr>
<tr>
<td>Sensors</td>
<td>Actors</td>
<td>Logic</td>
<td>Control Loop</td>
</tr>
<tr>
<td>Adaptronic Materials</td>
<td>Reconfigur. Hardware</td>
<td>Virtualisation</td>
<td>Programmable Platforms</td>
</tr>
</tbody>
</table>

- **Simple interfaces** ("services", "applications")
- **Complex systems** ("compositions")
- **Physical entities** ("components")
- **Building blocks** ("objects")
- **Physical foundation** ("code")
Example: Body Informatics

- Local code
- Analytics
- Hosted code
- Storage + Compute Clouds
  - Storage + Compute Clouds
  - RPC
  - HTTP
  - map-reduce
- Stealth layer
- Bluetooth
  - Bluetooth BLE
  - ANT+
- RPC
- HTTP
- dispersion
- encryption
Example: Body Informatics

- **Simple interfaces**
  - (“services“, “applications“)

- **Complex systems**
  - (“compositions“)

- **Physical entities**
  - (“components“)

- **Building blocks**
  - (“objects“)

- **Physical foundation**
  - (“code“)

- **Body Informatics**
  - **Hardware**
  - **Cloud Software Analytics**
  - **Hybrid (CPS)**
  - **Devices**
  - **Functions**
  - **Weight Sensors**
  - **Aggregation Logic**
  - **Programmable Platforms**
Example: Cloud Robotics Application
Example: Cloud Robotics Application

SLAM: Simultaneous Localization and Mapping (SLAM)
Sensoric capability: Mounted camera
Logic capability: Photo (feature) catalogue linked to object model
Example: Cloud Robotics Application

- Simple interfaces ("services", "applications")
- Complex systems ("compositions")
- Physical entities ("components")
- Building blocks ("objects")
- Physical foundation ("code")
Lecture Outline

1. Resource abstraction
   Practice: Using platforms and registries

2. Software encapsulation
   Practice: Using containers

3. Autonomous service interaction
   Practice: Using automated signup

4. Controlled resource use
   Practice: Using stealth databases

5. Towards Platforms
Resource Abstraction / 1

Architecture approach

Traditional CPS

Cloud-centric IoT

Stealthified CPSoS Edge
Resource Abstraction / 2

Support for emergent behaviour
Concrete architecture for federated robots

- Deploy application

- CLI access
- Future PaaS
- HTTP access (restricted)

- Robot Operating System
- Roboreg Module
- ROS Module
- Other Module
- Salt Minion(s)
- Salt API
- Salt Master
- Found applications & services:
  - ROS node: camera
  - Container: compression
  - ROS node: wheel
  - Container: map web app

Compared to general CPS architecture
- analytics blocks implemented by cloud-hosted containers
- data flow implemented by web and messaging services
- selective messaging through system properties (multicast)
Using platforms and registries

Prerequisites:
- Salt Stack (infrastructure management)
  - https://github.com/saltstack
- Roboreg (registry for robots with discoverable capabilities)
  - https://github.com/serviceprototypinglab/roboreg.git

Instructions:
- follow Roboreg README (step-by-step guide)
- understand mix of HTTP calls, MQ operations and runners
- think of executing ROS commands on actual robots
- think of PaaS access to this functionality
Software Encapsulation / 1

ROS: Robot Operating System

Nodes
• communicating executables
• connected by graphs
• advantages: fault tolerance, complexity reduction

Messaging
• simple types and arrays/structs
• transport via UDP and TCP, negotiated at connection time
• relies on 'Master' registry
Software Encapsulation / 2

ROS nodes + cloud
Software Encapsulation / 3

Portable software deployment
- across systems, architectures and platforms
- most suitable for CPS
- underpinning of service offerings for SaaS

Formats
- language-specific packages (e.g. JARs, eggs, gems, modules)
- system-level packages (e.g. DEB, RPM)
- executable container images (e.g. Docker)

Distribution and discovery
- repositories: versioning, search, download
- e.g. Docker Hub for Docker images
Software Encapsulation / 4

Container characteristics

- isolated namespaces for processes and files
  - ... provided by container engine
- option to containerise applications without OS
  - ... called application container
- boot from a virtual disk
  - ... called image

Limited isolation

Slightly reduced performance

Limitation: OS requirements often not fulfilled on older hardware
Container image creation with Dockerfile (specific to Docker)
Layered modifications
• e.g. base image with cloud/hardware-specific additions

FROM ubuntu
MAINTAINER xxxx@zhaw.ch

RUN apt-get install -y emacs
RUN apt-get install -y apache

EXPOSE 80 10001

ADD files/micro-inetd /opt/

CMD ["/opt/micro-inetd", "10001", "/usr/games/cowsay", "hello"]
Using containers

Prerequisites:
- Docker engine (container runtime)
  - https://docker.com

Instructions:
- Pull image with ROS components (may take a long time...)
  - on Docker Hub: gtoff/rosmaster
- Run image in interactive mode
- Launch ROS master node
Autonomous Service Interaction / 1

How to connect autonomous systems with service-oriented environments?

Example: robots

Requirement: identities & dependencies
Autonomous Service Interaction / 2

AdAPtS Architecture: Advanced Autonomous Participation Scheme
AdAPtS Implementation

Autonomous Participation
- Identity Generator
- CyberBrain AI (Python w/ plugins)
  - analyse & plan
  - Requirements (WSML goal)
  - Identity (vCard)
  - Agent Tool (Perl SOAP::Lite)
  - perform: selection
  - trigger
- User and Device Context
- Configuration, SLA Conditions

Service Marketplace / Broker
- Agent Code (Perl Mechanize, Tesseract OCR)
- Service Registry (ConQo)
  - Agent Description (XML)
  - Cloud Service Description (WSML)
  - perform: sign-up, use

Service Hosting
- Online/Cloud Service Provider
  - invoke
  - agent-less query

Client Application
- Client-specific Configuration
- Watcher
  - sense

Automation
Autonomous Service Interaction / Practice

Using automated signup

Prerequisites:
• OSST (Online Service Signup Tool)
  • https://github.com/cloudmaster/osst
  • A bunch of Perl modules

Instructions:
• study OSST README file for background information
• generate a vCard with a custom identity
• understand concepts of accounts, identities, service descriptions, agents and web automation
In rugged environments, data should be processed with care.

Key concerns: resource capacity, price, confidentiality, performance
Controlled Resource Use / 2

Example: Secret-sharing-aware bit expansion algorithm

- **Valid variants**
  - Variant 1
  - Variant 2
  - Variant 3

- **Invalid variants**
  - Variant 4
  - Variant 5
  - Variant 6

**Error cases:**
1. One bit too few set, Hamming weight wrongly determined
2. One bit too much set, secret-sharing rule not honoured
3. Non-decidable variant and fragment without information
### Data coding overview

<table>
<thead>
<tr>
<th>Coding technique</th>
<th>Minimal fragment number $\tilde{k}$</th>
<th>Tolerated failures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block creation/Chunking</td>
<td>$= k$</td>
<td>0</td>
</tr>
<tr>
<td>Erasure Coding</td>
<td>$= k$</td>
<td>$m$</td>
</tr>
<tr>
<td>Replication</td>
<td>$= 1$</td>
<td>$m$</td>
</tr>
<tr>
<td>Secret Sharing</td>
<td>$\geq k$</td>
<td>$0 / m$</td>
</tr>
<tr>
<td>Interpolation</td>
<td>$\leq k$</td>
<td>$k - 1$</td>
</tr>
<tr>
<td>Bitsplitting</td>
<td>$= k$</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Coding technique</th>
<th>Structure preservation</th>
<th>Recursion</th>
<th>Processing</th>
<th>Redundancy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block creation/Chunking</td>
<td>non-deterministic</td>
<td>no</td>
<td>limited</td>
<td>0%</td>
</tr>
<tr>
<td>Erasure Coding</td>
<td>non-deterministic</td>
<td>no</td>
<td>limited</td>
<td>0-100%,...</td>
</tr>
<tr>
<td>Replication</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>100%,200%...</td>
</tr>
<tr>
<td>Secret Sharing</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>0% / &gt;0%</td>
</tr>
<tr>
<td>Interpolation</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>0%</td>
</tr>
<tr>
<td>Bitsplitting</td>
<td>partial</td>
<td>yes</td>
<td>partial</td>
<td>1x (50%,...)</td>
</tr>
</tbody>
</table>
Using stealth databases

Prerequisites:
• StealthDB
  • http://lab.nubisave.org/stealthdb/

Instructions:
• Clone “dispersedalgorithms“ repository
• Run db/stealthdb
• Explore the unique commands:
  • USE CLOUDS and ALTER TABLE ... USE CLOUDS
  • SELECT ... OPTIMIZE FOR
  • SELECT ... FOREVER
• Create a dispersed in-memory table and perform some CRUD on it
Summary

Cyber-Physical Applications based on diverse Cyber-Physical Systems (cloud-connected sensors, robots, machines) remain a challenging research topic.

This talk has emphasised on:
- hardware abstraction / fleet management
- software encapsulation and portability
- redundancy exploitation and controlled traded-offs
- autonomous and autonomic participation management
- possible future platforms
Towards Platforms

Coordination Platform
- RoboEarth

Development Platform
- Gazebo Model Editor
- Pyro Trace
- Subsumption

Runtime Platform
- Lego Mindstorms
- ROS
- NAOqi

Hardware Platform
- ROP Turtle
- ROP Sergio
- Harvard Kilobot
- Bluefrog Buddy
Towards Platforms
Publications
