Verteilte Sensornetzwerke

Mit Datenaggregation und Sensorfusion

PD Stefan Bosse

Universität Bremen - FB Mathematik und Informatik

Überblick

Inhalte in diesem Kurs

1. Einführung des Sensormodells als Datenquelle
2. Erweiterung des Sensormodells auf Sensornetzwerke, IoT, Edge- und Cloud Computing
3. Virtualisierung und Netzwerke aus Virtuellen Maschinen (VM)
4. Kommunikation und Datenaggregation in Sensornetzwerken
5. Architekturen und Topologien von Sensornetzwerken, Simulation von Netzwerken mit zellulären Automaten und Diskreter Ereignissimulation
6. Algorithmen und Architekturen der Datenaggregation und Datenfusion
7. Parallel zu 1-7: Programmierung eines verteilten Sensornetzwerks mit Smartphones, Notebooks, und dem Raspberry Pi, und der Lua Programmiersprache
8. Formale Grundlagen von verteilten Systemen; Metriken; Effizienz; Skalierung

Begleitet von Übungen um obige Techniken konkret anzuwenden

Vorlesung

2 SWS mit Grundlagen und Live Programming
→ Präsenzunterricht/Synchroner Livestream + Chat
→ Tutorial Videos

Übung

1 SWS mit Programmierung und angewandter Vertiefung → LuaOS und LuaOS WEB mit Online Hilfe Funktion und Einreichungssystem

Voraussetzungen

Grundlegende Programmierfähigkeiten, Grundkenntnisse in Rechnerarchitektur und Netzwerken

Zielgruppen des Kurses

- Informatiker, Wirtschaftsinformatiker
- Systemingenieure (Systems Engineering)
- Produktionstechniker und Logistiker
- Elektrotechniker

[tiridifilm/istockphoto.com]

Ziele/Kompetenzen

Die Studenten erwerben/gewinnen/lernen

1. Verständnis der Grundprinzipien und Architekturen verteilter Sensornetzwerken und Fähigkeit zum Transfer auf technische Systeme, vor allem mit Fokus auf VMs
2. Verständnis und Fähigkeit der Programmierung mit Synchronisation und Kommunikation einfacher Sensorapplikationen
3. Verständnis der Probleme und dem Betrieb von verteilten Systemen im Vergleich klassischen Serversysteme (Effizienz, Blockierung, Skalierung, Ressourcenbedarf)
4. Praktische Kenntnisse der Programmierung und Programmierübungen mit Lua, der PLVM, und LuaOS
5. Erkenntnisse von Grenzen und Möglichkeiten der Verteilung und die Fähigkeit effiziente Systeme zu entwickeln
6. Studenten sind am Ende des Kurses in der Lage mit Lua einfache verteilte Sensorapplikation z.B. mit Smartphones zu programmieren und zu analysieren.

Materialien

1. Die Vorlesungsinhalte (Skript, Folien) werden auf **https://edu-9.de** unter der Rubrik Lehre zusammengestellt und angeboten. Achtung: Kürzel **dsn2k** beachten!

2. Weitere Materialien (Tutorials, Übungen, Software) werden ebenfalls auf **https://edu-9.de** bereitgestellt

3. Die Videos sind über **https://edu-9.de** verfügbar (Video Stream Server)

4. Interaktion der Teilnehmer findet über einen Wiki statt! (*dokuwiki*). Dieser ist über https://ag-0.de erreichbar und in den jeweiligen Veranstaltungszeiten auf **https://edu-9.de** verlinkt.

5. Es wird noch einen online Chat geben.

6. Alle weiteren Hinweise und Einführungen (z.B. in Software) auf dem Wiki und StudIP.

Leistungen

Folgende Möglichkeiten einer Prüfungsleistung stehen zur Auswahl:

1. Mündliche Prüfung
2. Schriftliche Ausarbeitung zu einer Fragestellung zu dem Thema (Review/Survey)
3. Die Bearbeitung einer experimentellen Arbeit (Lua) mit kleiner schriftlicher Arbeit (Dokumentation)

Literatur

Vorlesungsskript und Folien

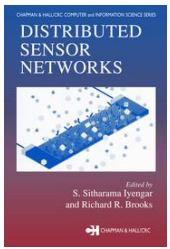
Die Inhalte der Vorlesung werden sukzessive bereitgestellt, das Vorlesungsskript steht im HTML und ebook Format zur Verfügung!

Sensornetzwerke in Theorie und Praxis

Ansgar Meroth, Petre Sora, Springer 2018, ISBN 9783658183868

Distributed Sensor Networks

S. S. Iyengar and R. R. Brooks, CRC Press, 2005



Literatur

Sensor Technologies - Healthcare, Wellness, and Environmental Applications

Michael J. McGrath, Cliodhna Ní Scanaill, Apress Open, 2014

Lua Scripting Language

Tutorial Points, K. K. Panigrahi, 2016.

Software

lvm

edu-9.de

- LuaJit VM mit Multithreading
- Ausführung von der Kommandozeile
- Wird auch für Live Programming und mit Digitalen Notebooks genutzt
- Einsatz auf verschiedenen Hostplattformen : PC, Smartphone, Raspberry Pi, Server, ..
- Einsatz auf Betriebssystemen: Windows, Linux, Solaris, MacOS, Android

```
> lvm parfib.lua
Thread [fe5af458:4] released
Thread [fe5afa00:5] released
{
  1 = 9227465,
  2 = 24157817,
  3 = 63245986,
  4 = 165580141,
  5 = 14930352,
  6 = 39088169,
  7 = 102334155,
  8 = 267914296,
}
686478381
Time 6235 ms
```

Ausführung Kommandozeile

Konzept

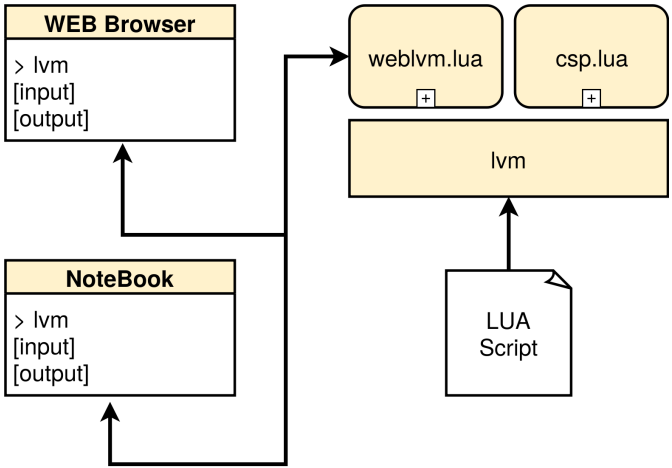


Abb. 1. LVM kann über einen WEB Proxy direkt vom WEB Browser aus benutzt werden

Konzept

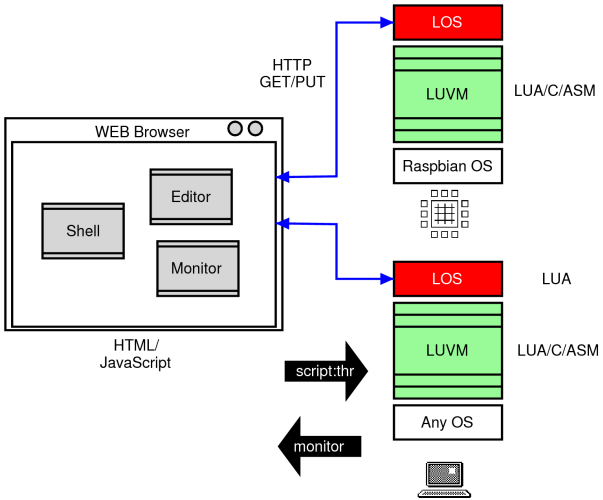


Abb. 2. Das LuaOS bietet die vernetzung von lvm Knoten und die Anbindung an den WEB Browser über die LuaOS WEB IDE

Konzept

Lokale Datenverarbeitung, Globale Fusion

- Es sollen verteilte Systeme betrachtet werden
- Typisch in verteilten Systemen sind drei Phasen:

1. Verteilungsphase von Daten und Aufgaben
2. Ausführung der Aufgaben
3. Zusammenführung der Ergebnisse

Konzept

In verteilten Sensornetzwerken sind die Daten meistens schon inhärent verteilt.

- Daher **(semi-)autonome lokale Datenverarbeitung** mit Reduktion der Sensordaten auf relevante Informationen
- Kommunikation und Zusammenführung der reduzierten Informationen

Virtuelle Maschinen

- Die Verwendung von virtuellen Maschinen nimmt in der Datenverarbeitung immer mehr zu (**Skriptverarbeitung und Abstraktion**)
- Ein Schwerpunkt liegt in der Verteilung und Parallelisierung von Datenverarbeitung in und mit virtuellen Maschinen
- Vor allem kommunizierende Systeme sollen betrachtet werden!

```
local R = rpc:new()
local ips = {
  "192.168.8.141",
  "192.168.8.142",
  "192.168.8.143",
}
local sensors={}
for i=1,5 do
  local err,reply=R:trans(ips[i],12345,{cmd="get"})
  if (err==nil) then
    sensors[i]=reply.sensor
  end
end
```

Virtuelle Maschinen und Interpreter

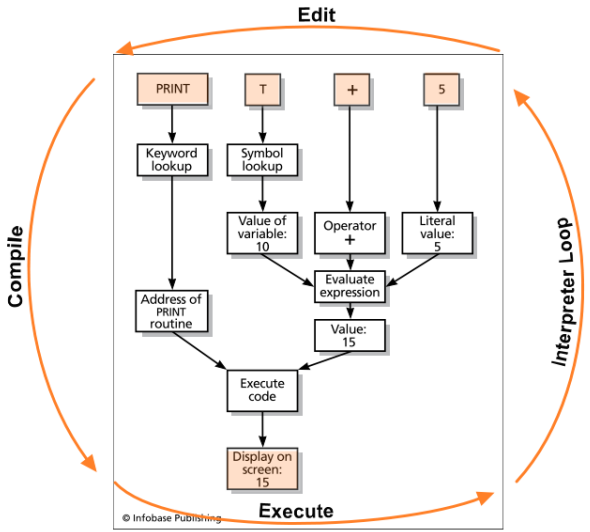


Abb. 3. Interpreter Zyklus: Editieren → Übersetzen → Ausführen

Just-in-Time Compiler

Interpreter können im wesentlichen auf drei Arten (Architekturklassen) implementiert werden:

1. Direkte Ausführung des Quelltextes (die Nutzereingabe und bereits geschriebene Skripte) (*Parse* → *Execute*)

2. Virtuelle Maschine und Übersetzung des Quelltextes in eine Zwischenrepräsentation die von einer virtuellen Maschine ausgeführt werden kann → Bytecode

3. Virtuelle Maschine mit Bytecode Übersetzung, Ausführung des Bytecodes, und ausgewählter Übersetzung des Bytecodes in nativen Maschinencode → JIT Compiler!

Virtuelle Maschine: Virtualisierung ≡ Abstraktion und Automatisches Speichermanagement

Beispiele

- **BASIC**: Klasse 1
- **Python**: Klasse 2 (Bytecode)
- **JavaScript**: Klasse 2 (Spidermonkey, WEB Browser) und Klasse 3 (Google Chrome/V8, nodejs)
- **OCaML**: Klasse 2 (und native Codeerzeugung mit Compiler)
- **Lua**: Klasse 2 (Lua) und Klasse 3 (LuaJit/lvm)

Parallelisierung der Datenverarbeitung von VM schwierig, Verteilung hingegen ist prinzipiell möglich.

Abstraktion

Was kann durch eine VM abstrahiert oder virtualisert werden?

Programmiersprachen

Welche Programmiersprachen werden häufig verwendet?

Welche parallelen Programmiersprachen sind bekannt?

Parallele und Verteilte Programmierung mit Lua

- In diesem Kurs soll die Programmierung mit der Skriptsprache **Lua** erfolgen und mit der virtuellen Maschine *lvm* oder der *Fengari Lua VM* ausgeführt werden;
- Der Lua Quelltext wird durch einen Übersetzer in Bytecode übersetzt der von *lvm* direkt ausgeführt wird.
  - Besonderheit: Der Bytecode wird direkt während des Parservorgangs erzeugt (kein AST-IR)
- Die LuaJit VM (lvm) unterstützt parallele sowie netzwerkbasierte Datenverarbeitung und das Konzept der Prozessblockierung
  - Prozesse
  - Threads
  - Coroutinen (Fibers)

Parallele und Verteilte Programmierung mit Lua

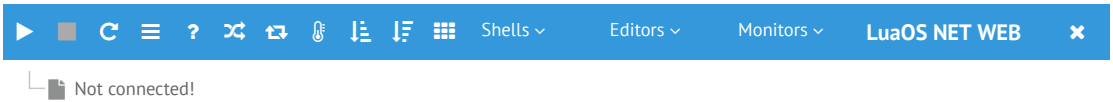
- Kontrollpfadparallelität benötigt i.A. Kommunikation und das Konzept der Blockierung!
- Formales Ausführungsmodell: **Communicating Sequential Processes** (CSP) nach Hoare
- **Programmfluß = Kontrollfluß + Datenfluß**
- Parallele und Verteilte Datenverarbeitung: *Übergang vom Shared Memory (SM) zum Distributed (Shared) Memory (DSM) Modell!* und von speicheroriertier zu nachrichtenbasierter Kommunikation

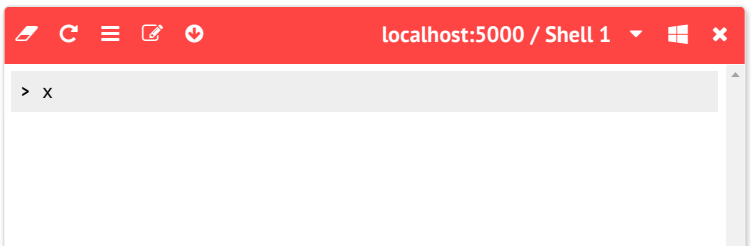
Verteilte Programmierung mit Lua

Parallel LuaJit Virtual Machine (LVM)

CLEAR GET RESET RESTART EXIT rpc server client

Verteilte Programmierung mit LuaOS und LuaWEB





Labor

- Das Labor soll den praktischen Einstieg in verteilte Sensornetzwerke bieten und **findet im Kurs und @home statt**!
  - Sensorknoten können Notebooks, Smartphones, und eingebettete Rechner (Raspberry) sein!
  - Die Sensorknoten werden über das Internet vernetzt
  - Die Sensoren werden über Schnittstellen angeschlossen (I2C)
- Programmierung mit Lua und LVM + LuaOS

Labor

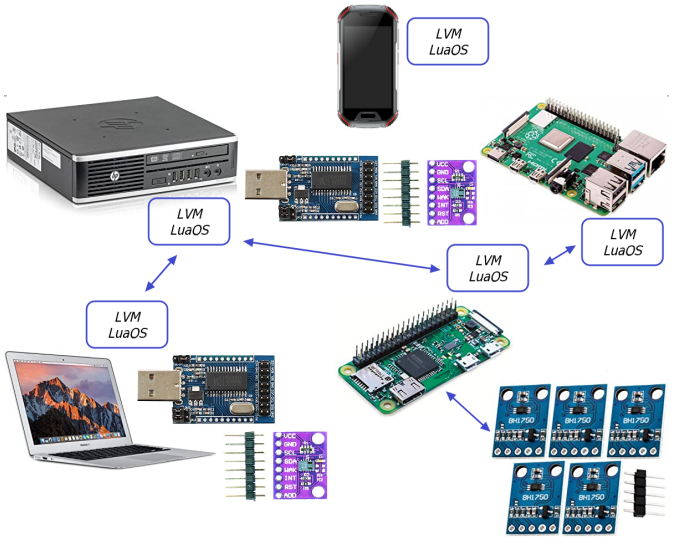


Abb. 4. Vernetzung von Rechnern über lvm/LuaOS und Aufbau eines verteilten Sensornetzwerkes

Labor

Folgende Hardwarekomponenten werden eingesetzt:

1. Der eigene Rechner (x86 singl/multi-core Prozessor)
2. Raspberry PI (Zero, ARM Cortex single-core Prozessor)
3. ESP32 (Transilica Dual Core Prozessor)
4. Sensormodule via I^2C angebunden (beim PC über Adapter)

Folgende Softwarekomponenten werden eingesetzt:

1. Luajit Parallel and Multitasking Virtual Machine *lvm* (Terminal, PC, Raspberry PI)
2. Parallel and Multitasking Fengari Lua Web VM mit Lua Web IDE (Web Browser, PC)
3. Parallel and Multitasking Fengari Lua Web VM mit Lua VNetOS (Web Browser, PC)
4. Multi-tasking Lua VM (ESP32)