Sakuli

End2End-Monitoring

Simon Meggle

21.05.2014

ConSol

Wir unternehmen IT.

AGENDA

- **Begriff "End2End-Monitoring"**
- **die Situation und ihre Herausforderung**
  - Funktionsweise von Sahi/Sikuli
  - Die Tools und ihre Stärken im Vergleich
- **Aus zwei mach eins: Sakuli = Sahi + Sikuli**
- **Architektur von Sakuli**
- **Live-Demos**
- **Ausblick**
  - geplante Features
  - Queue-basierte Architektur (Release 2)
- **Download**

Begriff "End2End-Monitoring"

- **Business-kritische Applikationen bestehen oft aus einer Vielzahl an Komponenten**
- **Applikations-Qualität nur am Ende der Funktionskette messbar**

=> Monitoring aus Sicht des Anwenders durch User-Simulation

die Situation und ihre Herausforderung

- **viele frei verfügbare E2E-Tools**
- **Grenzen**
  - zu speziell
  - Betriebssystem-spezifisch
  - zu generisch
  - zu wenig an der Realität
- **Bisher: Sahi2OMD (vorgestellt auf der OSMC 2013)**

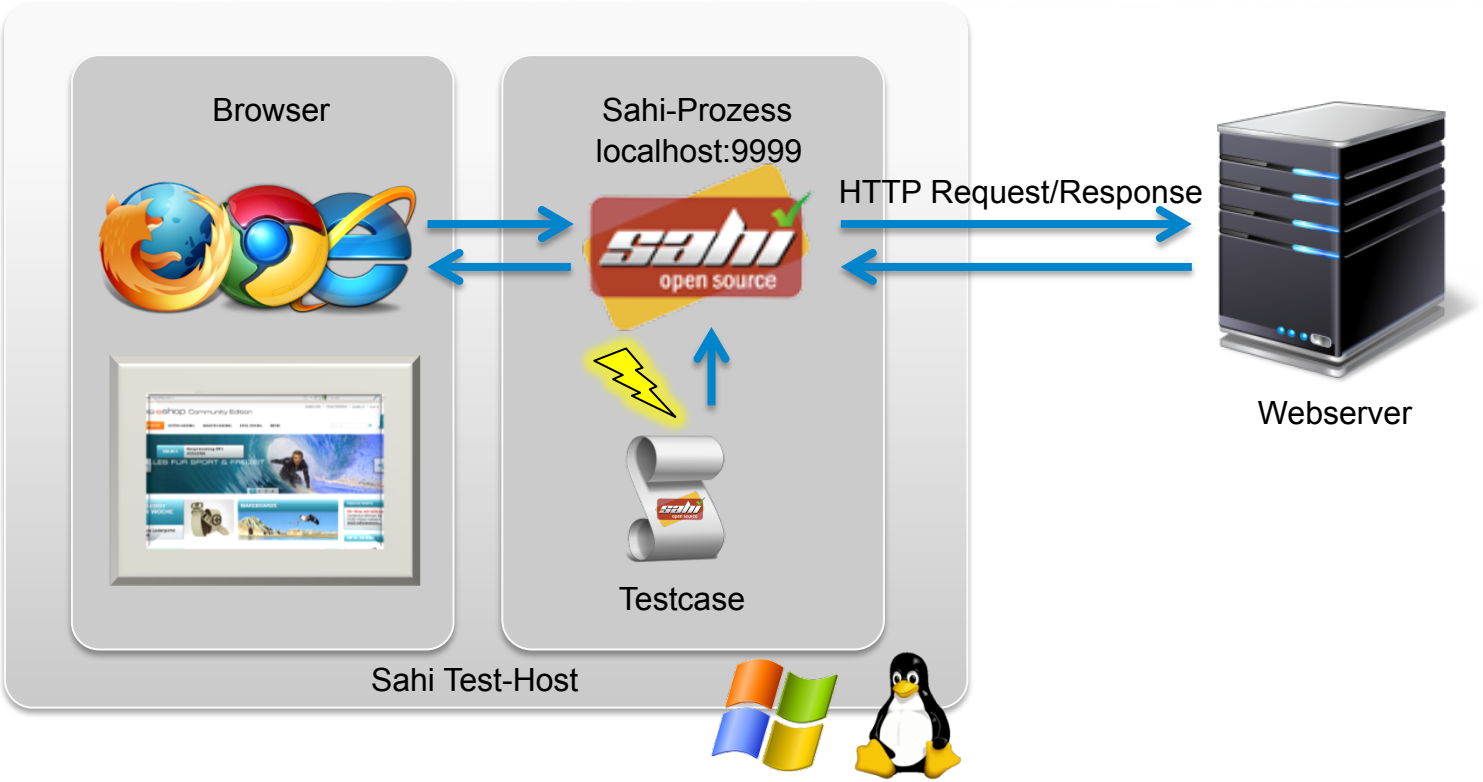
Funktionsweise von Sahi

- **Was ist/kann Sahi?**
  - ✓ entwickelt von Narayan Raman (Indien)
  - ✓ „Web Automation and Testing Tool“
    -> Tool zur Simulation von User-Aktion im Webbrowser
  - ✓ AJAX, Frames, iFrames, Up/Downloads, Browser alerts, NTLM Auth, HTTPS…
  - ✓ leistungsfähige Funktionen, wie z.B.:
    - ✓ intelligente Objekt-Adressierung auch über Frames/iFrames hinweg (kein XPath)
    - ✓ Funktionen für File/DB-Zugriff
  - ✓ Batch Mode & Multi-Threaded Playback
  - ✓ Sahi erkennt selbst, wenn Seiten/AJAX-Requests fertig geladen sind.
  - ✓ Java-API
  - ✓ Sahi läuft auf jedem OS mit Java (z.B. auch iOS, Android…) in jedem Browser (ohne Plugin)
  - ✓ Open Source
- **Was ist/kann Sahi nicht?**
  - alles, was außerhalb des DOMs liegt (z.B. Flash-Inhalte, Java-Applets)
  - Tests außerhalb des Browserinhaltes

http://sahi.co.in/

Funktionsweise von Sahi

- **Sahi wird als Proxy zwischen Browser und Server geschaltet**
  - **Aufzeichnung** aller weitergeleiteten Requests in *SahiScript*
  - **Wiedergabe** durch Injizieren der SahiScripts in den Browser



Situation und Herausforderung

- **viele frei verfügbare E2E-Tools**
- **Grenzen**
  - zu speziell (Bsp. PIN-Abfrage in Webbrowser)
  - abhängig vom Betriebssystem
  - zu generisch
  - zu wenig an der Realität (Bsp. Test auf Protokoll-Ebene)

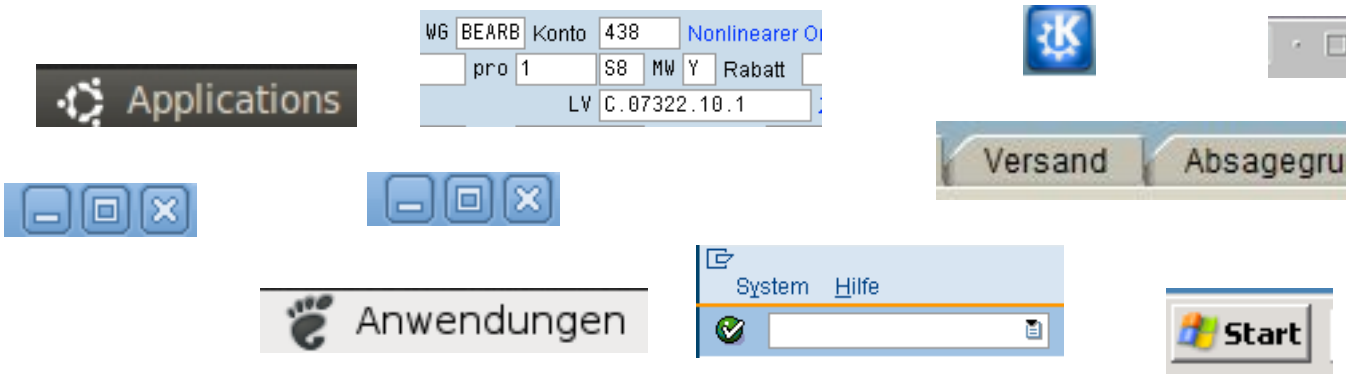
- **Bisher: Sahi2OMD (OSMC 2013)**
  - End2End Web-Tests
  - Hürden
    - Browserdialoge
    - Java-Applets, Flash, ...
    - Kundenwunsch: GUI-Monitoring

Herausforderung:

Kompensation der Schwachstellen von Sahi durch zweites Tool

Funktionsweise von Sikuli

- **Sikuli kann alles sehen und steuern, was der User sieht und mit Maus und Tastatur steuern kann**
- Kern-Komponenten:
  - **java.awt.Robot** zur Steuerung von Maus/Tastaturevents
  - **OpenCV** Engine zur Erkennung von Bildmustern
- Screenshots werden auf dem Bildschirm lokalisiert => Region
- Region-Objekte erlauben Aktionen wie *click(), type(),* etc...
- Texterkennung (OCR), experimental
- auf jeder Plattform verfügbar
- Open Source (MIT)



Die Tools und ihre Stärken im Vergleich

100% Web	1)	
Web & "off-web"	3)	2)
GUI-Tests (off-Browser content)		4)

1) theoretisch möglich; im Vergleich zu Sahi jedoch viel zu aufwändig (und Unsinn!)

2) nicht möglich; Java, Flash, PIN-Dialoge etc. sind für Sahi nicht "sichtbar", da keine Web-Inhalte

3) theoretisch möglich (siehe 1); bestenfalls jedoch "Notlösung"

4) Sahi kennt nur den Content des Browser-Canvas (den aber gut...)

Die Tools und ihre Stärken im Vergleich

Sakuli

100% Web	1)		
Web & "off-web"	3)	2)	
GUI-Tests (off-Browser content)		4)	

1) theoretisch möglich; im Vergleich zu Sahi jedoch viel zu aufwändig (und Unsinn!)

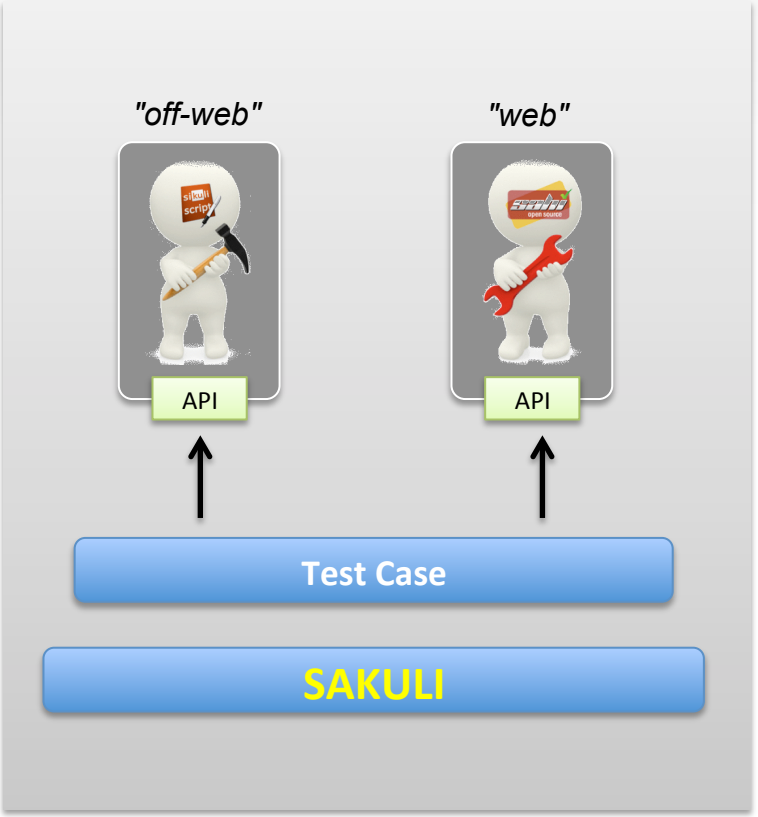
2) nicht möglich; Java, Flash, PIN-Dialoge etc. sind für Sahi nicht "sichtbar", da keine Web-Inhalte

3) theoretisch möglich (siehe 1); bestenfalls jedoch "Notlösung"

4) Sahi kennt nur den Content des Browser-Canvas (den aber gut...)

Aus zwei mach eins: Sakuli = Sahi + Sikuli

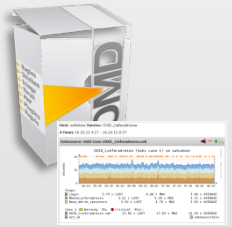
- entstanden 2013 aus "**sahi2OMD**" (-> OSMC 2012)
  => Gewinnung von **SIEMENS** als POC-Sponsor
- Java-Architektur: Steuern von Sahi und Sikuli über deren Java-API
  - Kapselung der Funktionsaufrufe beider Tools in JavaScript
    - einheitliche Handhabung
    - Erweiterung bestehender Funktionen
    - seamless integration: Sahi- und Sikuli-Funktionen sind im Test beliebig mischbar
  - Sakuli stellt alle Backend-Funktionen bereit (Initialisierung, Zugriff auf die Result-DB, Erstellen von Screenshots bei Exceptions, Logging, ...)
- Test-Cases in JavaScript-Syntax: **Keine Java-Kenntnisse** zur Test-Erstellung erforderlich



Architektur von Sakuli

"off-web"

"web"



MySQL Result DB

OMD

Test Case

SAKULI

Sakuli Client

Live-Demo

- **Aufbau von Sakuli-Scripts**
- **Testcase: eShop: Bestellung + Bestellbestätigung drucken**
  - Zusammenspiel von Sahi und Sikuli
  - check_mysql_health – Custom "Sakuli" Mode
  - Performancedaten – Custom PNP4Nagios Template
  - wenns kracht...
- **Sakuli "from scratch"**

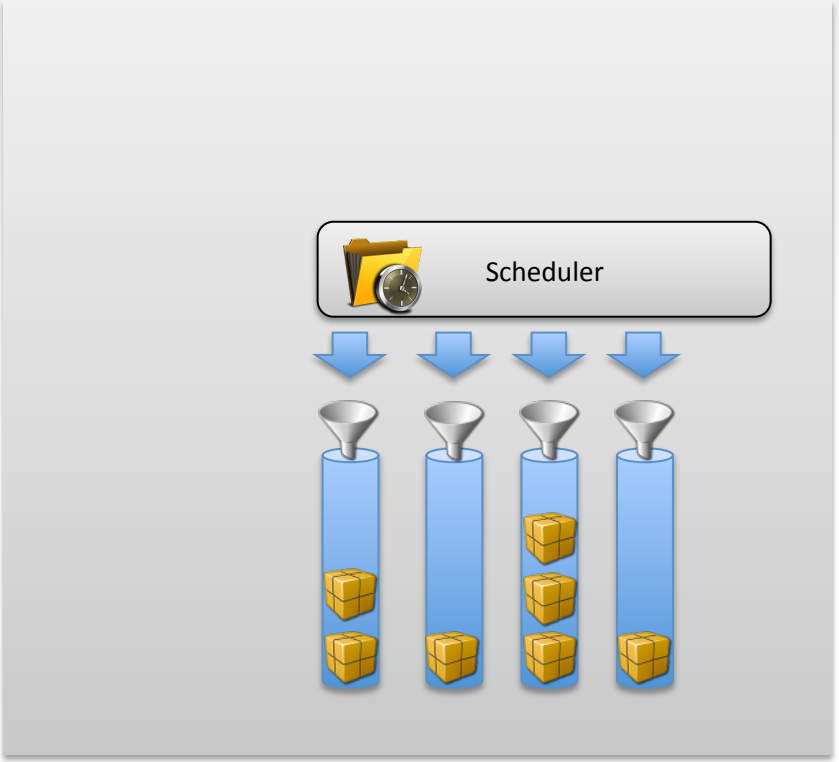
Ausblick: weitere geplante Features

- **Headless Execution**
  - Linux: xvfb
  - Windows: ?
- **Aufzeichnung der Testabläufe als Film**
  - Fehlerdokumentation
  - Debugging
- **Multi-Wait**
- **Sakuli als Integration Test Tool**
- **[Dein Vorschlag]**

Ausblick: Queue-basierte Architektur (Release 2)

Server/Scheduler

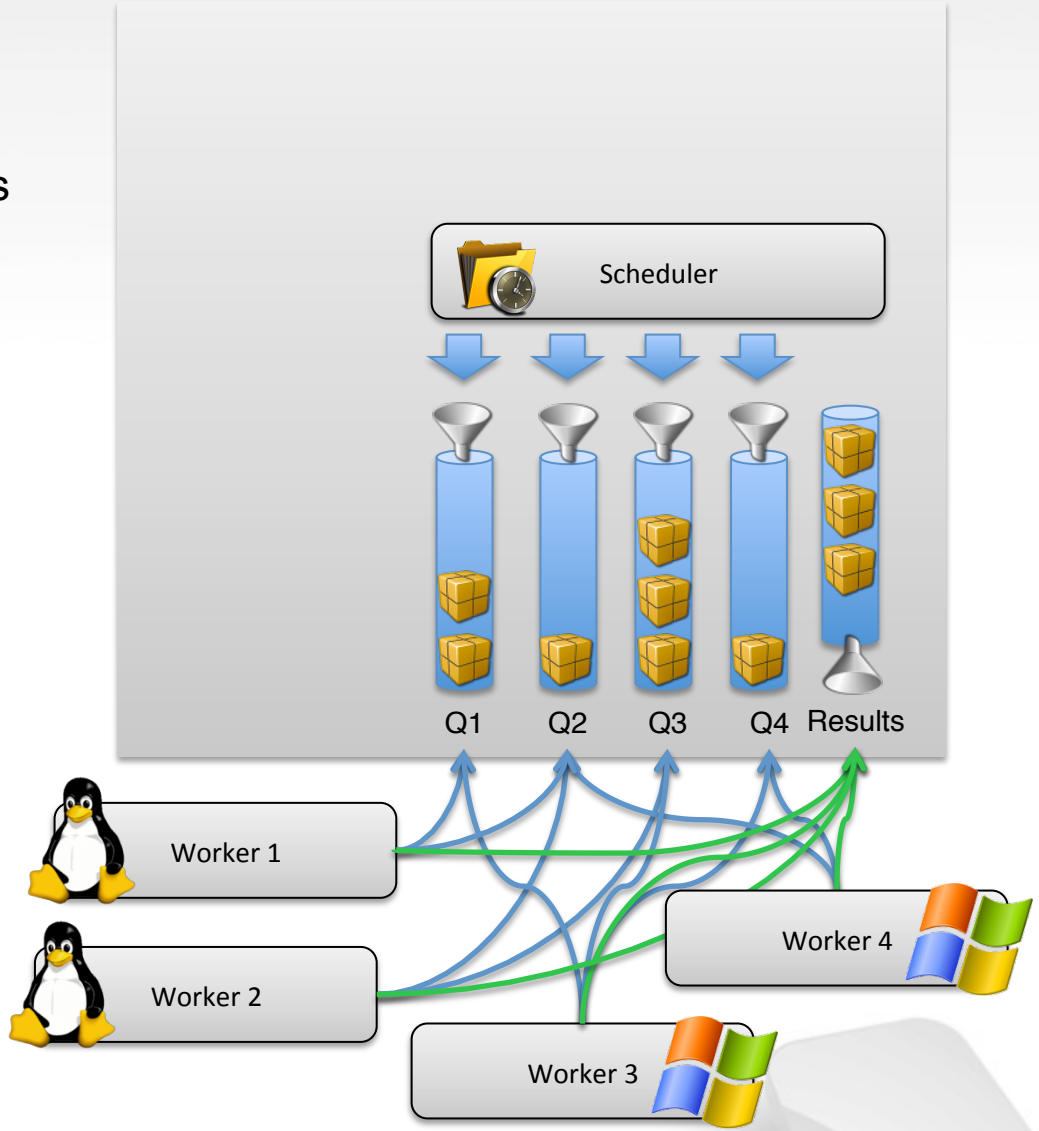
- mit Web-Oberfläche zur Verwaltung incl. JavaScript-Editor
- stellt Queues (gruppiert nach OS, Browser o.ä.)
- legt Test-Jobs in die entspr. Queues
- Test-Jobs enthalten alle zur Ausführung notwendigen Dateien



Ausblick: Queue-basierte Architektur (Release 2)

Worker

- ist in einer/mehreren Queue(s) registriert
- entnimmt Job aus der Queue und führt ihn aus
- speichert das Ergebnis in einer Result-Queue



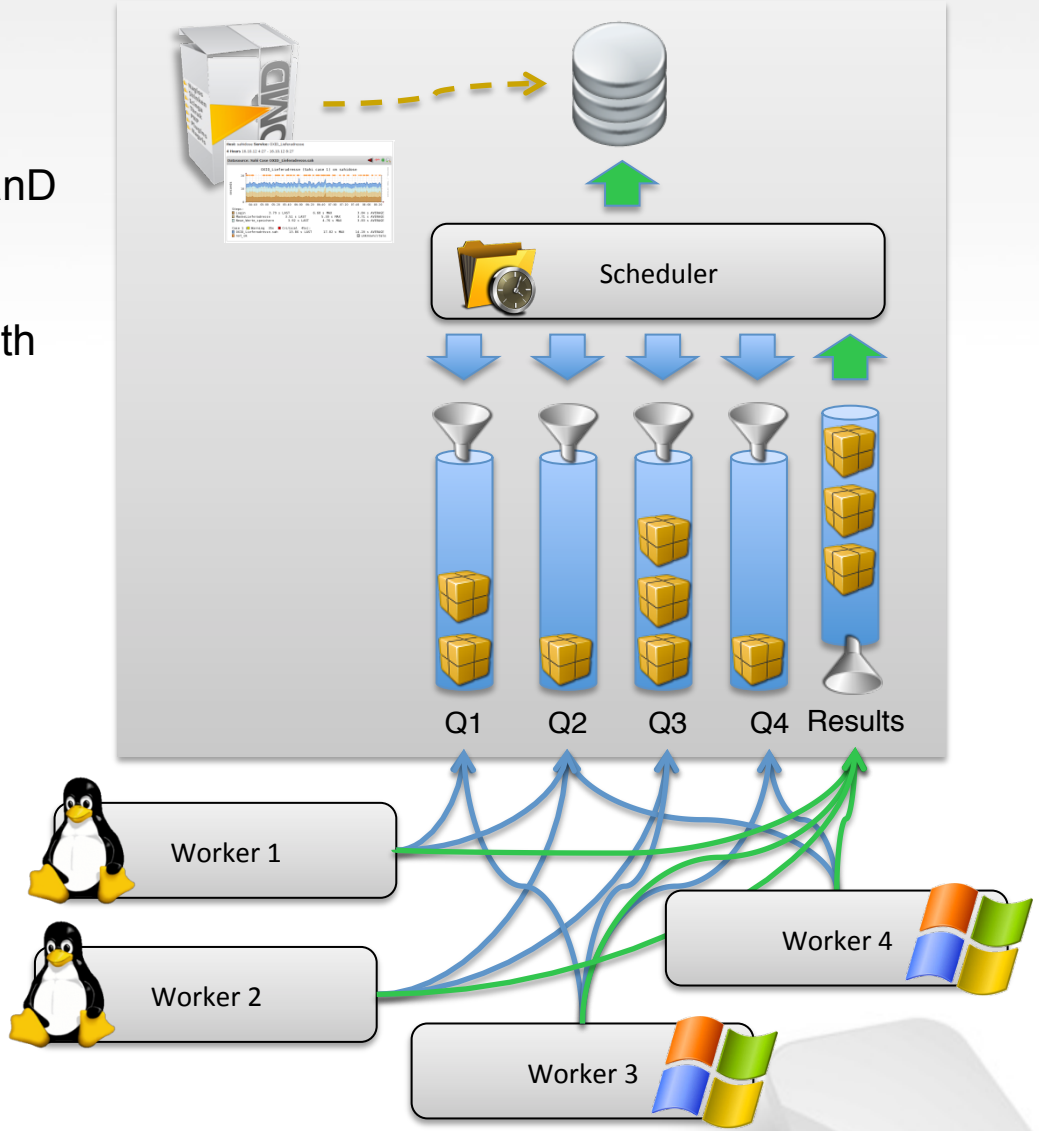
Ausblick: Queue-basierte Architektur (Release 2)

Server/Scheduler

- liest die Result-Queue und speichert die Ergebnisse in der DB / sendet sie per GearmanD

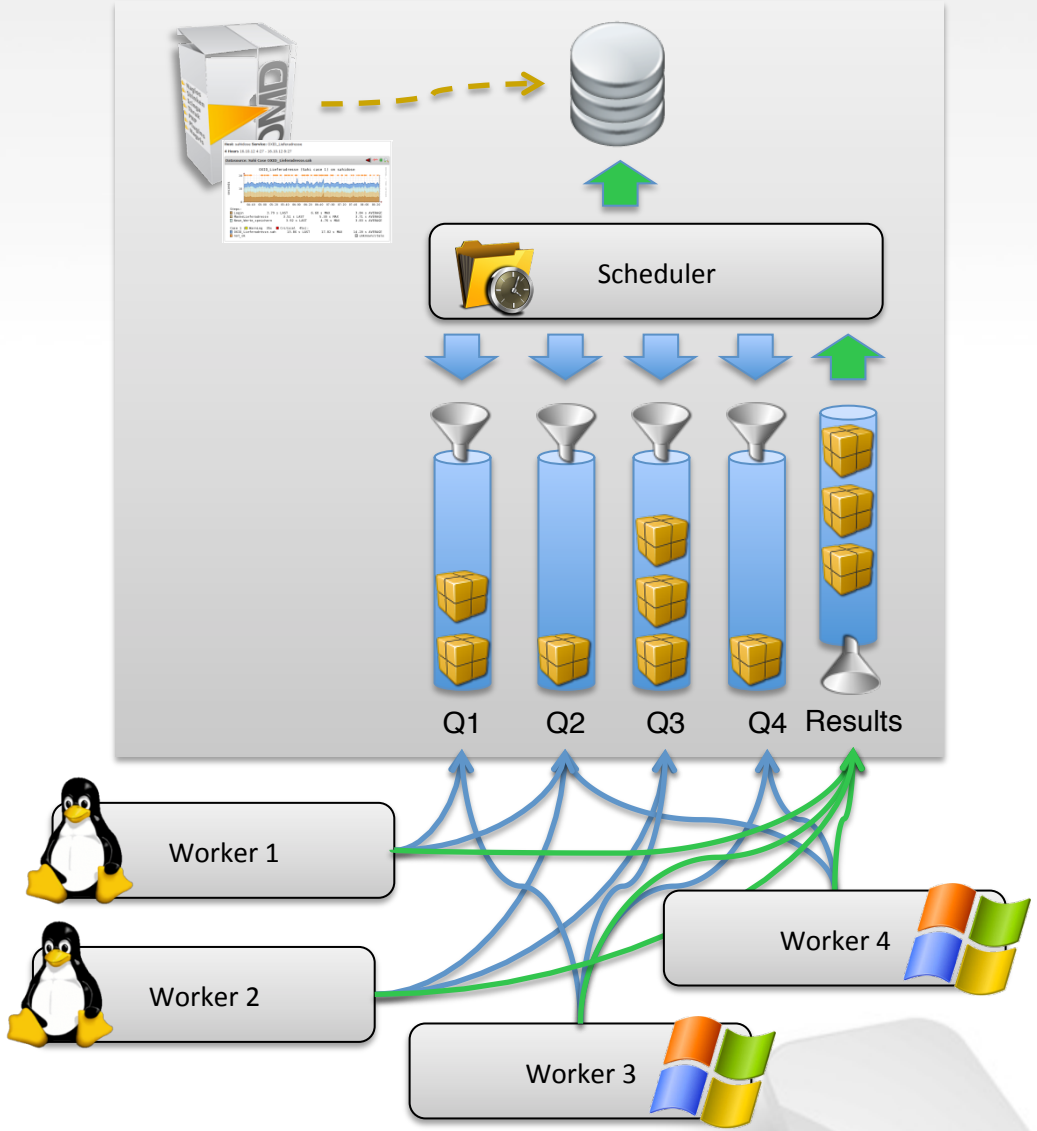
OMD

- prüft Result-Datenbank mit check_mysql_health (Sakuli-Mode)



Ausblick: Queue-basierte Architektur (Release 2)

- ✓ **Lastverteilung**
- ✓ Erhöhung der **Ausfallsicherheit**
- ✓ System "balanciert" sich selbst aus
- ✓ Clients können zur Wartung inaktiv geschalten werden,
  die Messungen verteilen sich auf andere Nodes
- ✓ **einfache Skalierung** durch Hinzufügen von Workern
- ✓ Integration des Sakuli-Schedulers in **OMD**



Download

https://github.com/ConSol/sakuli

Vielen Dank für die Aufmerksamkeit!

Q&A