SensApp

A reference platform to support Cloud experiments: from the internet of things to the internet of services

Sébastien Mosser, Franck Fleurey, Brice Morin, Franck Chauvel, Arnor Solberg
*SINTEF IKT, NSS Department, MOD Group, Oslo, Norway*

Iokanaan Goutier
*SINTEF IKT & Polytech’Lille, Lille, France*
Motivations

- Cloud computing relies on fuzzy definitions and buzzwords
- A fixed point is needed to compare solutions and assess results
- The AOM community faced exactly the same problem
- They produce the CCCMS in 2009
- 32 papers use this case study!
Requirements for a reference Cloud application

• Reality-driven
  • A toy is not enough (e.g., Petshop for J2EE)

• Reference implementation availability
  • “Theory guides, Experiments decide.”

• Positioned w.r.t. identified cloud-computing challenges
  • The application must be a “prototypical” cloud application
Have you met ...
Internet of Things example: the « e-Bike »
SensApp: collects data from IoT sensors
SensApp: supports IoT sensors exploitation

Internet of Services ← Internet of Things
Example: Sensor accuracy comparison
SensApp fulfils the two first requirements

- Reality-driven
  - SensApp is used in several places to collect data
  - SINTEF living lab, Car & bike experiments, I3S collaboration
- Reference implementation availability
  - Scalable platform implemented in Scala/MongoDB
  - Open-source, available on GitHub and CloudBees

- What about the “common cloud challenges” (3rd requirement)?
  - Identified in the Berkeley seminal cloud report?
A. Availability of Services

• Description:
  • Provider outage can last several hours
  • SLA become critical for “legal issue”
  • Multi-tenant cloud federation minimize the risks

• SensApp:
  • Can be easily deployed on multiple instances
  • Outage simulation == shutting down a SensApp node
  • Reference datasets available to control experimental setup
B. Data lock-in

• Description:
  • Providers offer proprietary data interface
  • Migration from one cloud to another is tricky
  • Providers refuse to offer a standard interface (political)

• SensApp:
  • Reify sensor data using the SenML standard (Cisco et al.)
  • SenML is small enough to be easily implemented (24 pages)
    • Available datasets contains up to 250,000 data/sensors
C. Data confidentiality and Auditability

• Description:
  • Data confidentiality is an issue for companies
  • Hybrid clouds are defined to keep some data internally
  • Some companies cannot put data abroad (legally)

• SensApp:
  • Data confidentiality == Sensor confidentiality
    • Air quality is public, personal location is private
  • MongoDB can be distributed on several nodes
D. Data transfer bottleneck

• Description:
  • Cloud application are data-intensive, consuming bandwidth
  • Bandwidth follows a “pay-as-you-go” scheme
  • “Smart deployment” == “money saved”

• SensApp:
  • Relies on SenML to model data as a fixed point
  • Bandwidth reduction techniques can be applied on SenML
  • Large reference datasets available for experiments
E. Performance Unpredictability

- **Description:**
  - Several virtual machines share the same concrete host
  - Even if “isolated”, performance can be affected by others
  - Difficult to re-play experiments

- **SensApp:**
  - Offers reference data sets, collected in real-life situations
  - VM isolation can be tested using these real-life data
  - Data are time-stamped, re-play == follow the timestamps
F. Scalable Storage

• Description:
  • Cloud application handles large amount of data
  • Storage must scale
  • Query interface must scale too

• SensApp:
  • 1 sensor, 1 value/s = 86,400 value/day = 9MB (3GB/year)
  • SenML data are exploited by external application (query)
  • Easy to change the internal data representation
G. Bugs in Large distributed Systems

- Description:
  - Clouds trigger a new family of bugs
  - Generated by cloud specificities (latency, faults)
  - Impossible to replicate outside of the clouds

- SensApp:
  - Reference implementation is open-source
    - Static analysis or code mutation are possible
  - JVM based web-server supports remote debugging
H. Scaling Quickly

• Description:
  • Clouds define “elasticity” as fundamental
  • Vertical: increasing the VM “power”
  • Horizontal: increasing the number of available VMs

• SensApp:
  • Build on top of a REST architecture, plain HTTP.
  • Intrinsically ready for load-balancing and horizontal scaling.
  • Simulated sensors are easy to implement (for stress testing)
I. Reputation fate sharing

- Description:
  - In the clouds, VM are “rented”, as well as IP addresses
  - A “nasty” VM might have is IP blacklisted
  - A “pretty” VM can be blacklisted when re-using such an IP!

- SensApp:
  - Implements asynchronous notification mechanisms
  - Source code is small enough to be easily modified
    - A reputation protocol can be implemented
J. Software Licensing

• Description:
  • Clouds want to provide anything “as a service”
  • Service == Business. Business == Money
  • Cost models must be carefully designed

• SensApp:
  • Reference data sets reify “normal” usage
    • Can feed cost model simulation
  • Multi-criteria analysis can be used to select “the” provider
Example of “Cloud Experiment”: Scalability

- Benchmarks to verify:
  - A. Availability of services
  - D. Data transfer bottleneck
  - F. Scalable Storage
  - G. Scaling Quickly
- Defined for SensApp
  - Reused by anybody!

```scala
class SensorPushSimulation extends Simulation {

  val numberOfData: Int = 200
  val maxDelayBetweenPush: Int = 400
  val url = "http://..."

  def apply = List(sensorPush.configure.users(10).ramp(10))

  val headers = Map("Content-Type" -> "application/json",
                    "Accept" -> "text/plain,application/json")

  val sensorPush =
    scenario("Sensor pushing Data")
      .exec{
        http("Alive?").get(url).check(status is 200)
      }.pause(100, 200, MILLISECONDS)
    .exec {
      http("Sensor Creation").post(url)
        .headers(headers).body(genSensor())
    }.pause(100, 200, MILLISECONDS)
    .loop{ chain
      .exec {
        http("Random data").put(url)
          .headers(headers).body(genRandomData())
      }.pause(100, maxDelayBetweenPush, MILLISECONDS)
    }.times(numberOfData)
  .exec {
    http("Sensor Deletion").delete(url)
  }
}
```

Gatling stress scenario
Gatling stress test output
Conclusions

• Clouds need a reference system to support research approaches comparisons and assessments

• Such an application must be « prototypical »

• We propose SensApp, an IoT platform
  • Used in several places,
  • Open-source reference implementation available
  • Can support research for all the identified cloud challenges
Technology for a better society