Safer in the Clouds

GianLuigi Ferrari, Chiara Bodei and Vietdung Dinh

University of Pisa

Jun 10, 2010
Content

1. Cloud computing
2. Differences between SOC and Cloud Computing
3. The framework
4. An example
5. Research Issues
Cloud computing

We consider the following definition borrowed from Buyya\textsuperscript{1}:
“A Cloud is a type of parallel and distributed system consisting of
a collection of inter-connected and virtualized computers that are
dynamically provisioned and presented as one or more unified
computing resource(s) based on service level agreements
established through negotiation between the service provider and
consumers.”

\textsuperscript{1}Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg and
Ivona Brandic. Cloud computing and emerging IT platforms: Vision, hype, and
reality for delivering computing as the 5th utility (2009)
Cloud computing - our main challenges

- **Semantics**: specify a programming model for designing and implementing Cloud systems.
- **Resource awareness**: provides suitable mechanism to express and enforce policies governing usages of resources available in the clouds.
- **Reasoning techniques**: to statically check the behavior of programs and ensure their safety execution at runtime.
Cloud computing

Could we apply Service-Oriented Computing (SOC) technologies? Are “services” enough for clouds?
Cloud computing

Could we apply Service-Oriented Computing (SOC) technologies?
Are “services” enough for clouds?
Original focuses:

▶ In SOC: system integration
▶ In Cloud Computing: outsource all layers of computing stack
Cloud computing

Could we apply Service-Oriented Computing (SOC) technologies? Are “services” enough for clouds?

- Take 1: the resource view
  - Service provides an abstract layer for resources.
  - Cloud service needs more control over resources.

- Take 2: the logical view
  - Service models business logic
  - Cloud service models business + operation logic

---

2Andrew Gordon - Get operations logic right: Types, Service-Orientation, and Static Analysis

Safer in the Clouds
Cloud computing

Could we apply Service-Oriented Computing (SOC) technologies? Are “services” enough for clouds?

- Take 1: the resource view
  - Service provides an abstract layer for resources.
  - Cloud service needs more control over resources.

- Take 2: the logical view
  - Service models business logic
  - Cloud service models business + operation logic

... take advantages of techniques for SOC.

---

\(^2\)Andrew Gordon - Get operations logic right: Types, Service-Orientation, and Static Analysis

Safer in the Clouds

University of Pisa
Our approach

- Services as functions with side effects: $\tau \xrightarrow{e} \tau'$.
- Side effect reflects the change of service state
  - Side effects may also reflect virtualization issues
- Usage policies based on the abstract behavior.
  - The abstract behavior takes form of sequences of resource accesses $^3$.
  - Policies are defined as Usage Automata $^4$.
- Asynchronous Communication

---

$^3$ Access Control based on Execution History - M. Abadi 2003
$^4$ Massimo Bartolleti. Usage Automata. 2009
The Model

A concurrent $\lambda$-calculus with

- parallel composition
- resources + access events
- usage policies
- service provision.
Operational Semantics

- Client interactions are modelled via asynchronous communication
- The configuration of cloud server: $\langle \eta \triangleright e \triangleleft \sigma \rangle$
  - $\eta \in H$ - a history: abstraction of resource usage and behavior
  - $e$ - abstract "code"
  - $\sigma$ - a service store: the directory of the available services.
These services are identified by their names
Example: An database service

\[ E_{\text{form}} = \lambda q. \ E_{\text{proc}} \ q \ \text{where} \ E_{\text{proc}} = \lambda y. \ \text{open}(db); (query \ db \ y); \text{close} \]

The cloud server:

\[ (0 \triangleright 0 \triangleleft \sigma[Q \rightarrow E_{\text{form}}]) \]
Example: An database service

\[
E_{form} q \\
\xrightarrow{\tau} E_{proc} q \\
\xrightarrow{\tau} open(db); ((query db q); close(db)) \\
open(db) \xrightarrow{\tau} ((query db q); close(db)) \\
\tau \ast dbcmd \xrightarrow{\tau} \ast close(db) \\
\tau \ast close(db) \xrightarrow{\tau} 0
\]

The normal possible traces: \(open(db).\tau^* .dbcmd.\tau^*.close(db)\)
Example: A tainted query: \texttt{syscmd}; q

\[
\begin{align*}
\text{syscmd} & \quad \tau^* \\
\text{dbcmd} & \quad \tau^* \quad \text{close}(db)
\end{align*}
\]

The trace under sql injection attack:
\texttt{open}(db).\texttt{syscmd}.\tau^*.\texttt{dbcmd}.\tau^*.\texttt{close}(db)

\textit{Solution:} $\phi[e\text{process}]$, where $\phi$ does not allow any $\texttt{syscmd}$ to be executed, will prevent that trace.
Example: A tainted query: $\texttt{syscmd}; q$

\[ \vdots \]

\[
\begin{align*}
\text{open}(\text{db}) & \quad \rightarrow ((\text{query } \text{db} \texttt{syscmd}; q); \text{close}(\text{db})) \\
\texttt{syscmd} & \quad \rightarrow^\tau^* \texttt{dbcmd} \\
& \quad \rightarrow^\tau^* \text{close}(\text{db}) \\
\vdots
\end{align*}
\]

The trace under sql injection attack:

$\text{open}(\text{db}). \texttt{syscmd}. \tau^* . \texttt{dbcmd}. \tau^* . \text{close}(\text{db})$

Solution: $\varphi[e_{\text{process}}]$, where $\varphi$ does not allow any $\texttt{syscmd}$ to be executed, will prevent that trace.
Abstract Semantic

- bisimulation semantics: to reason about the Clouds

\[(\eta_1 △ e_1 ◊ \sigma_1) \sim (\eta_2 △ e_2 ◊ \sigma_2)\]

- Instrumenting a program with a policy framing does not add new behavior to the program. \(\varphi[e] \lesssim e\)
Research Issues

1. difficult to automate verification
   • symbolic analysis
   • abstract interpretation

2. preliminary result shows that side effect as PA-processes with policy framing
   • correctness of PA-processes against usage policy via model checking
Thank you!