A Case for the Accountable Cloud

Andreas Haeberlen
MPI-SWS

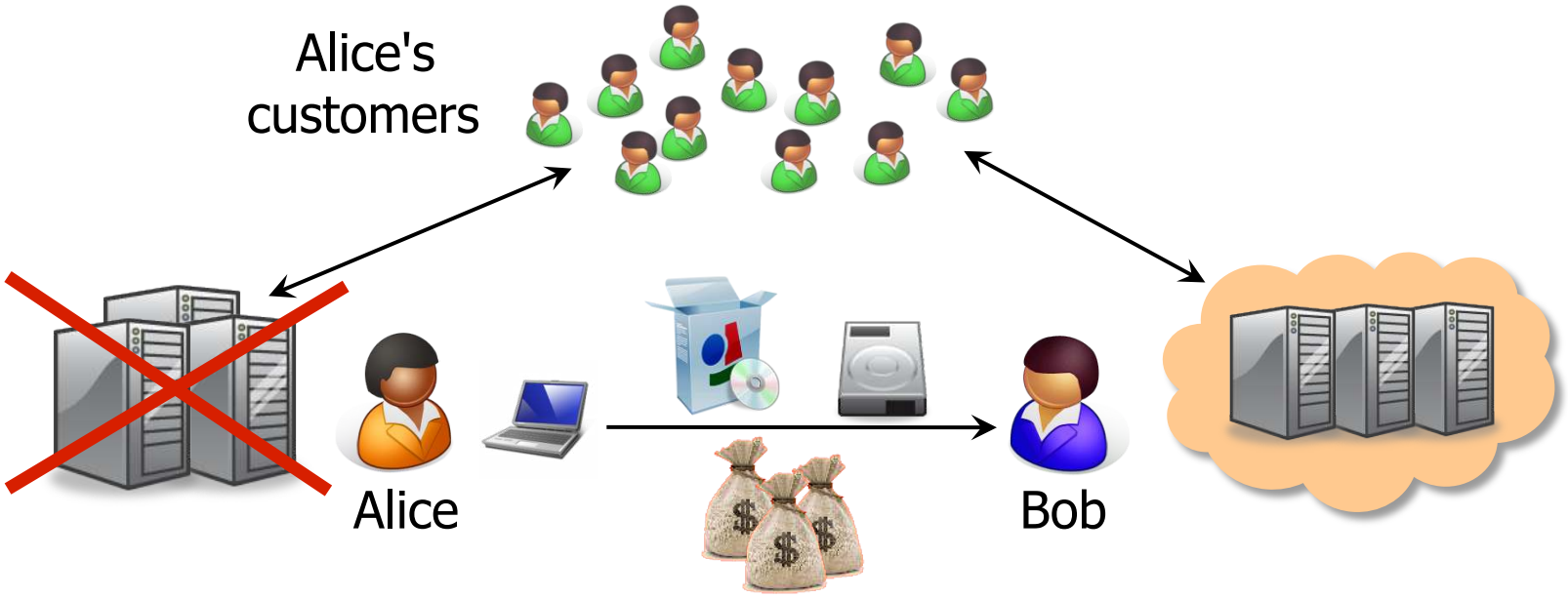
Outline

Problem

Solution

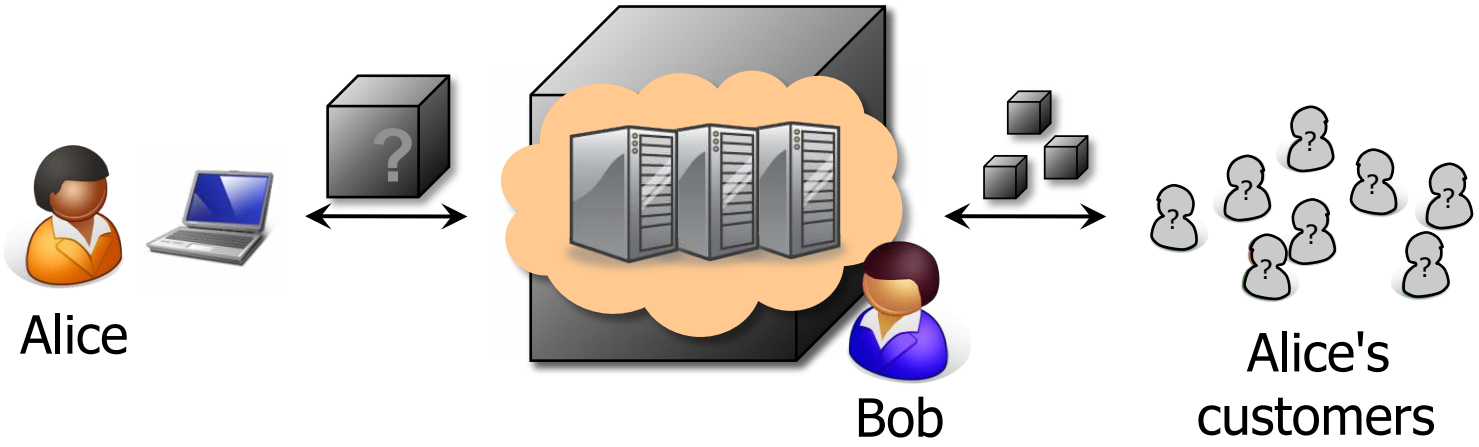
Call for action

The benefits of cloud computing



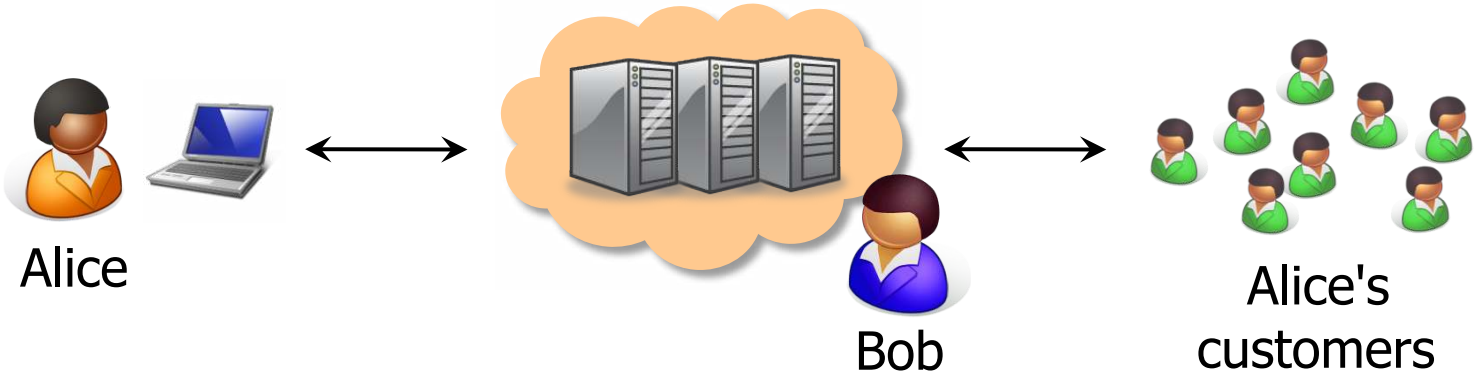
- The cloud enables Alice to:
  - obtain resources on demand
  - pay only for what she actually uses
  - benefit from economies of scale
- But...

Problem: Split administrative domain



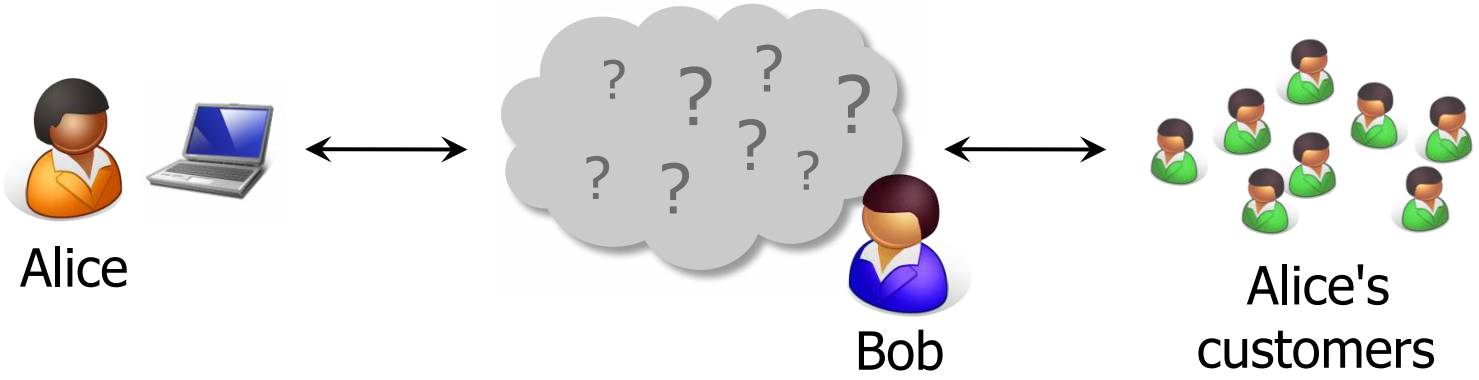
- Control and information about Alice's service are now split between Alice and Bob
  - Alice cannot control cloud machines or observe their status
    → Alice must have a lot of trust in Bob
  - Bob does not understand the details of Alice's software
    → Difficult to perform many administrative tasks

Problem: Split administrative domain



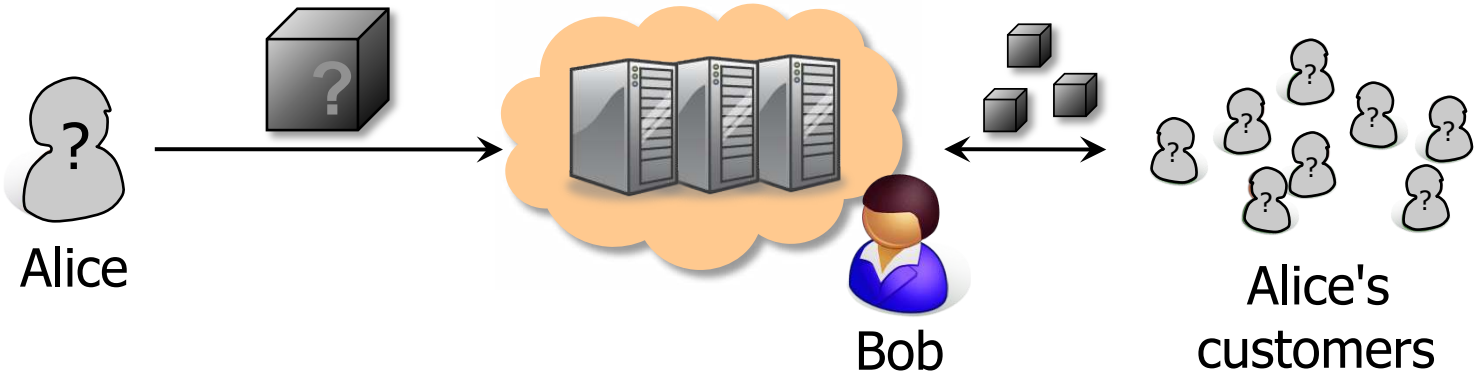
- What if there is a problem with the cloud?
  - Misconfiguration
  - Insufficient allocation of resources
  - Hacker attack
  - Data loss or unavailability
  - Hardware malfunction
  - ...

Handling problems: Alice's perspective



- If something is wrong, how will I know?
- How can I tell if it's my software or the cloud?
- If it's the cloud, how can I convince Bob?

Handling problems: Bob's perspective



- If something is wrong, how will I know?
- How can I tell if it's my software or the cloud?
- If it's the cloud, how can I convince Bob?

- If something is wrong, how will I know?
- How can I tell if it's the cloud or Alice's software?
- If it's Alice's software, how can I convince Alice?

Outline

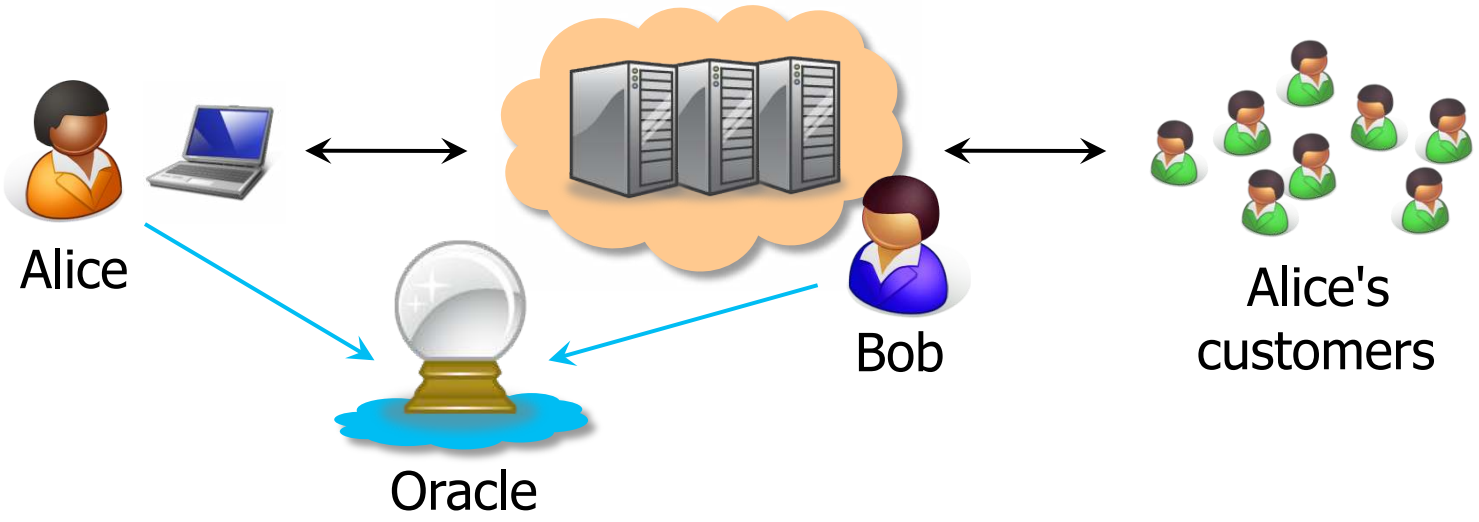
Problem

Split administrative domain

Solution

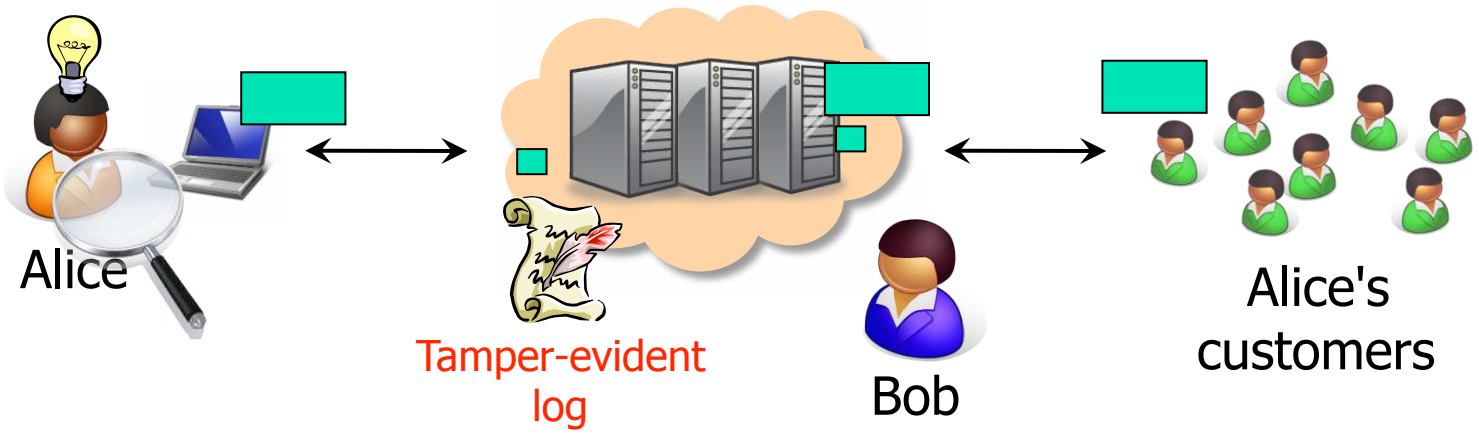
Call for action

An idealized solution



- What if we had an oracle that Alice and Bob could ask about cloud problems?
  - Completeness: If the cloud is faulty, the oracle will say so
  - Accuracy: If the cloud is not faulty, the oracle will say so
  - Verifiability: The oracle produces evidence that would convince a disinterested third party

The accountable cloud



- Idea: Make cloud accountable to Alice+Bob
  - Cloud records its actions in a tamper-evident log
  - Alice and Bob can audit the log and check for faults
  - Use log to construct evidence that a fault does (not) exist
- Provides completeness, accuracy, verifiability
  - Provable guarantees even if Alice and/or Bob are malicious!

Discussion

- Isn't this too pessimistic? Bob isn't malicious!
  - Hacker attacks, software bugs, disgruntled employees, operator error, ..., can have the same effect
  - Difficult to come up with a more restrictive fault model
  - Alice (or some other customer) could be malicious
- Shouldn't Bob use fault tolerance instead?
  - Bob certainly should mask faults whenever possible
  - But: Masking is never perfect; Alice still needs to check
- Why would a provider want to deploy this?
  - Attractive to prospective customers
  - Helps with handling angry support calls

Discussion: Guarantees

- Are these the right guarantees?
  - Completeness: "No false negatives"
    - Could be relaxed: e.g., probabilistic completeness
  - Accuracy: "No false positives"
    - Cannot be relaxed safely if the detection of a fault can have serious legal/financial consequences for Bob
  - Verifiability: "Produce enough evidence to convince a third party"
    - Could be relaxed: e.g., evidence only needs to convince a specific third party

Outline

Problem

Split administrative domain

Solution

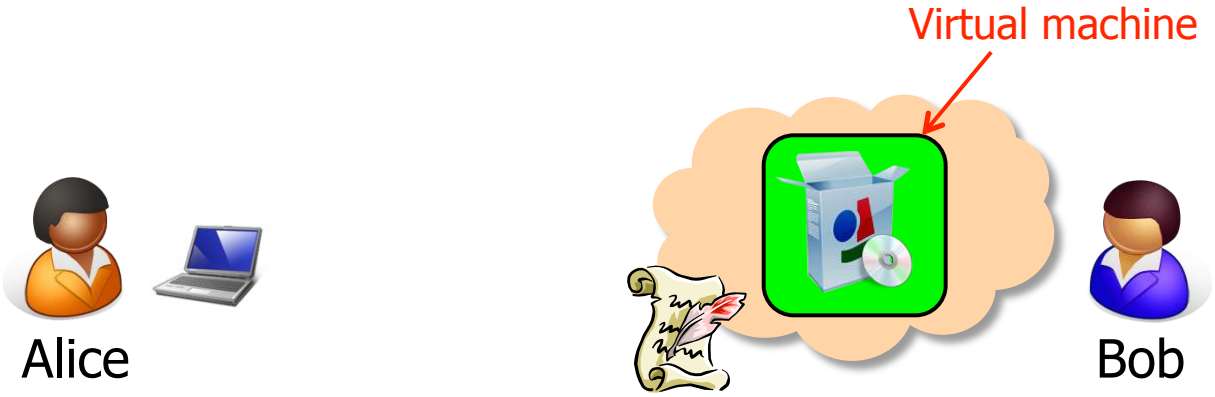
Make the cloud accountable

Call for action

Is the technology ready?

- Cloud accountability should:
  - Deliver provable guarantees ✓
  - Work for most cloud applications ✓
  - Require no changes to application code ?
  - Cover a wide spectrum of properties ?
  - Have a low overhead ?
- Can existing techniques deliver this?
  - ~~CATS~~, ~~Repeat&Compare~~, ~~AIP~~, ~~PeerReview~~, ~~NetReview~~, ~~AudIt~~, ...
- More research is needed!

Work in progress: AVM



- Goal: Provide accountability for arbitrary unmodified software
- Idea: Accountable virtual machine (AVM)
  - Cloud records enough data to enable determinstic replay
  - Alice can replay log with a known-good copy of the software
  - Can audit any part of the original execution

Summary

- Problem: Current cloud designs carry risks for both customers and providers
  - Customer loses control over his computation and data
  - Split administration → Difficult to detect+resolve problems
- Proposed solution: The accountable cloud
  - Can verify correct operation, produce evidence
  - Provable guarantees → solid foundation for both sides
  - Discussion: Guarantees, fault model, incentives, ...
- Lots of research opportunities

Questions?