A Case for Static Analyzers in the Cloud

Mehdi Bouaziz
École normale supérieure, Paris, France

joint work with

Michael Barnett, Manuel Fähndrich, and Francesco Logozzo
Microsoft Research, Redmond, WA, USA

Eighth Workshop on
Bytecode Semantics, Verification, Analysis, and Transformation

March 23, 2013 – Rome, Italy

This talk

Why and how to bring a static analyzer to the cloud?

This talk

Why and how to bring a static analyzer to the cloud?

Outline:

- Static analyzers today
- Why bringing them to the cloud
- Architecture of a cloud-based static analyzer
- Bonuses, issues

Examplified with our ongoing work on moving Clousot, the .Net Code Contracts static checker, to the cloud.

Code Contracts

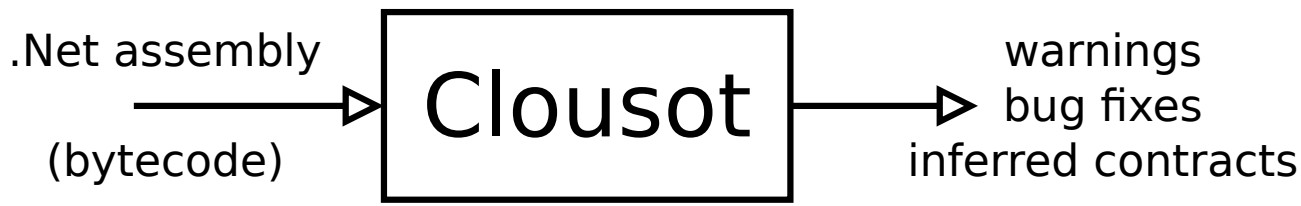
Contracts (preconditions, postconditions, object invariants) for .Net languages.

```
Class PositiveArray {
  int[] arr;

  void ObjectInvariant()
  {
    Contract.Invariant(this.arr != null);
    Contract.Invariant(Contract.ForAll(this.arr, x => x > 0));
  }

  int Max()
  {
    Contract.Requires(this.arr.Length > 0);
    Contract.Ensures(Contract.ForAll(this.arr, x => x <= Contract.Result<int>()));
    Contract.Ensures(Contract.Exists(this.arr, x => x == Contract.Result<int>()));
    ...
  }
}
```

- Contracts can be dynamically checked at runtime
- or statically checked, with Clousot

Clousot today

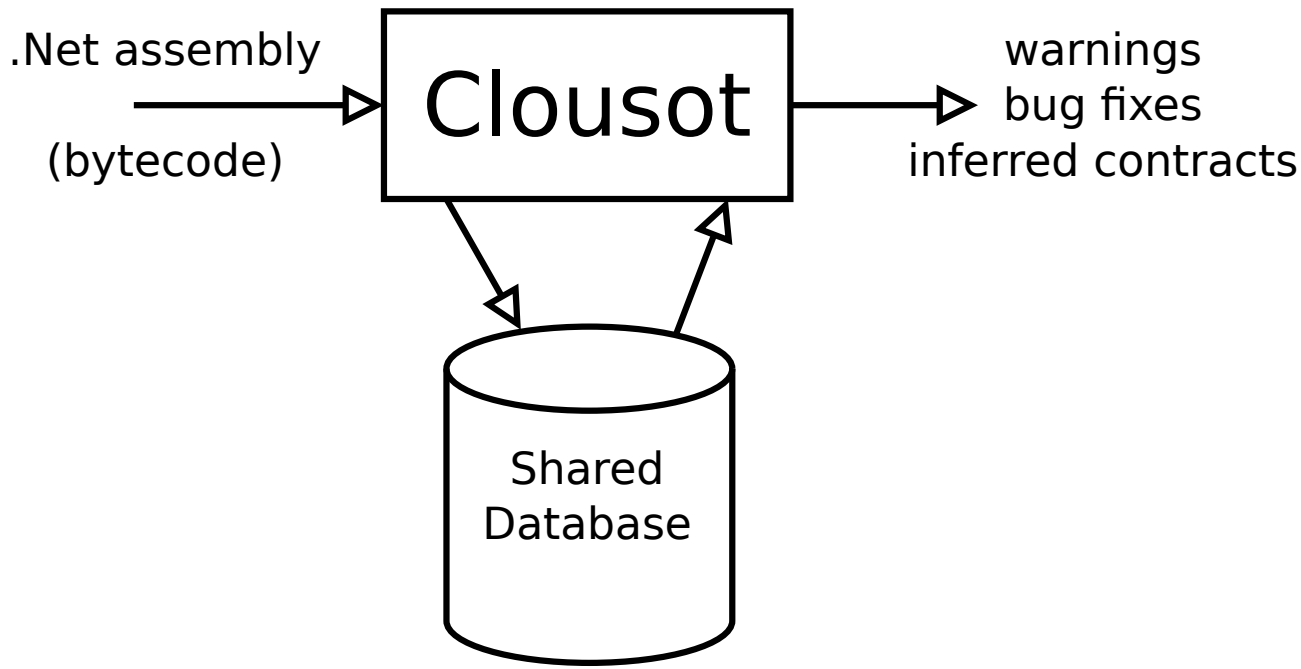
Abstract interpretation-based static checker for Code Contracts.



Clousot runs on a single core of the developer machine.
The bottom-up analysis of methods is sequential.
Analyses start from scratch each time, but analyses are expensive.

How to share results between developers?

Use a common database to share analysis results.

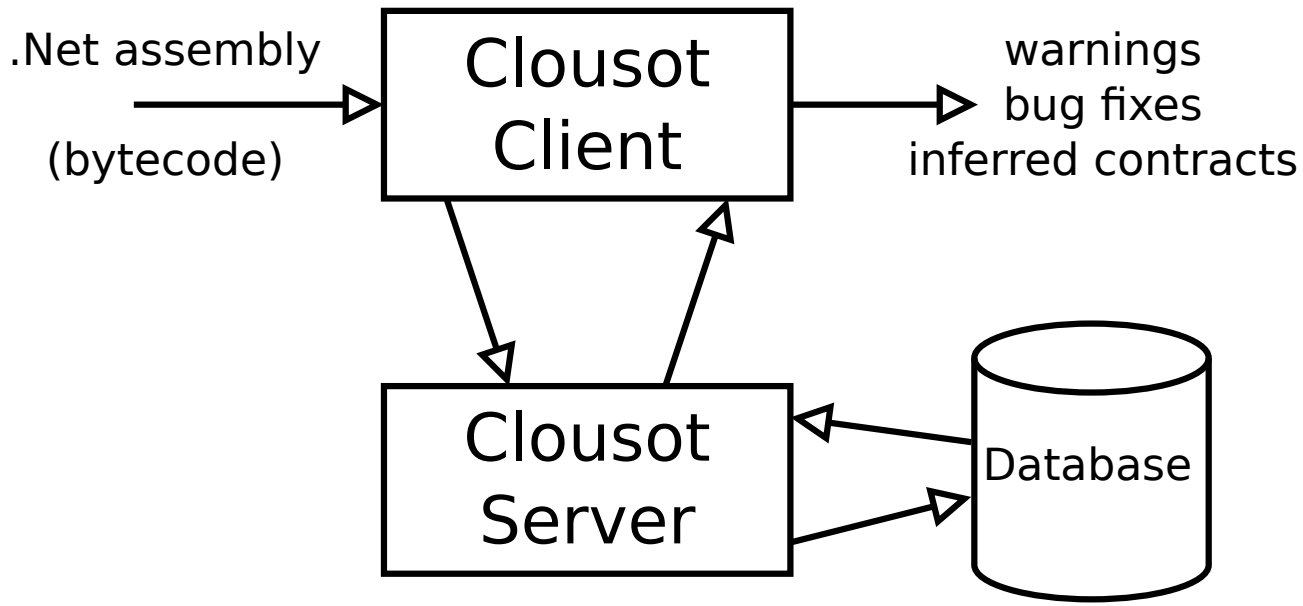


Analyses are still expensive

Use more CPUs and more memory!

Analyses are still expensive

Use more CPUs and more memory, i.e., use a centralized server.



A centralized server, does it scale up?

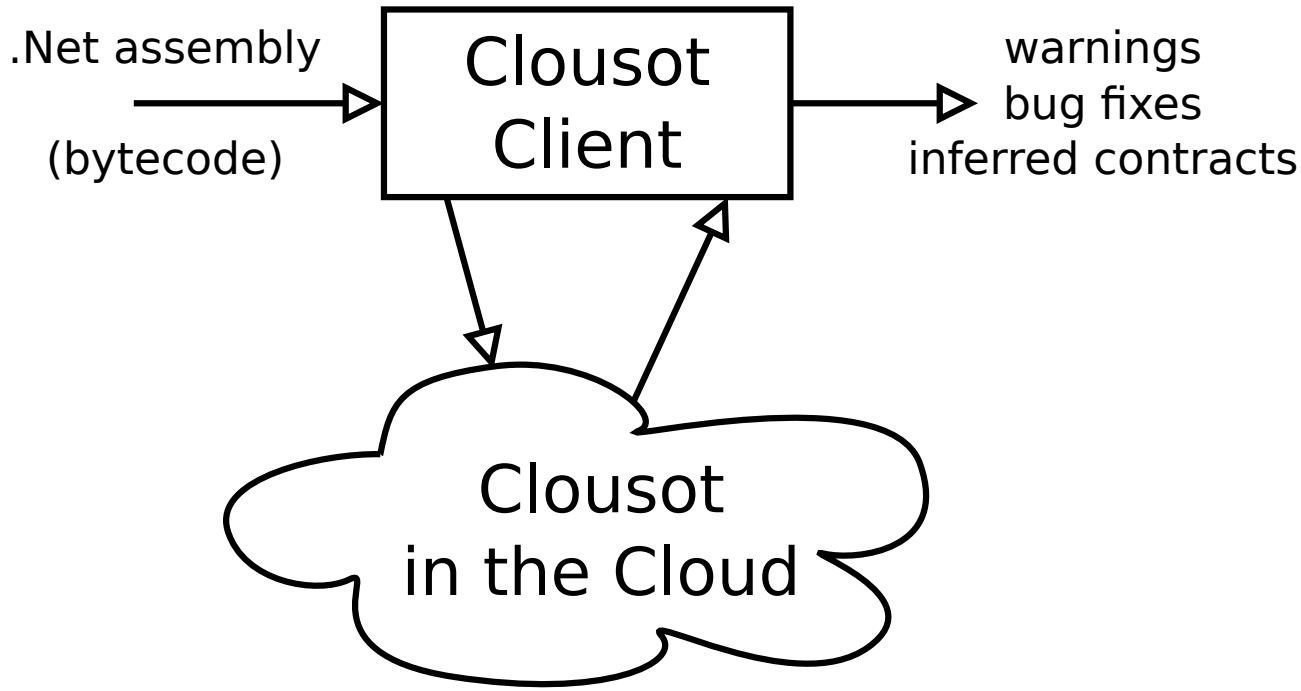
A centralized server, does it scale up?

- No elasticity in resource allocation
- Hardware/software maintenance issues

A centralized server, does it scale up?

- No elasticity in resource allocation
- Hardware/software maintenance issues

Use the cloud!



Static analyzer as a Cloud service

- Same as a centralized server but on a Cloud infrastructure
- Remote maintenance: one version for everybody
  - Clients do not need to update the tool
  - Faster deployment of bug fixes
- Results shared by everybody using the tool

Static analyzer as a Cloud service

- Same as a centralized server but on a Cloud infrastructure
- Remote maintenance: one version for everybody
  - Clients do not need to update the tool
  - Faster deployment of bug fixes
- Results shared by everybody using the tool
- Faster, more precise analyses
- Data collections on the usage of the tool enable:
  - Better understanding of how the tool is used
  - Identify the weaknesses, refine or design new domains
  - Reduction of false alarms
  - Semantics-guided warning suppression
  - Version-based metrics

The client part

- very small (call the service, show the results)
  - used anywhere: smartphone, tablet [TouchDevelop]

The client part

- very small (call the service, show the results)
  - used anywhere: smartphone, tablet [TouchDevelop]
- can issue parallel analyses on the same program, faster less precise results will come first, more precise ones later

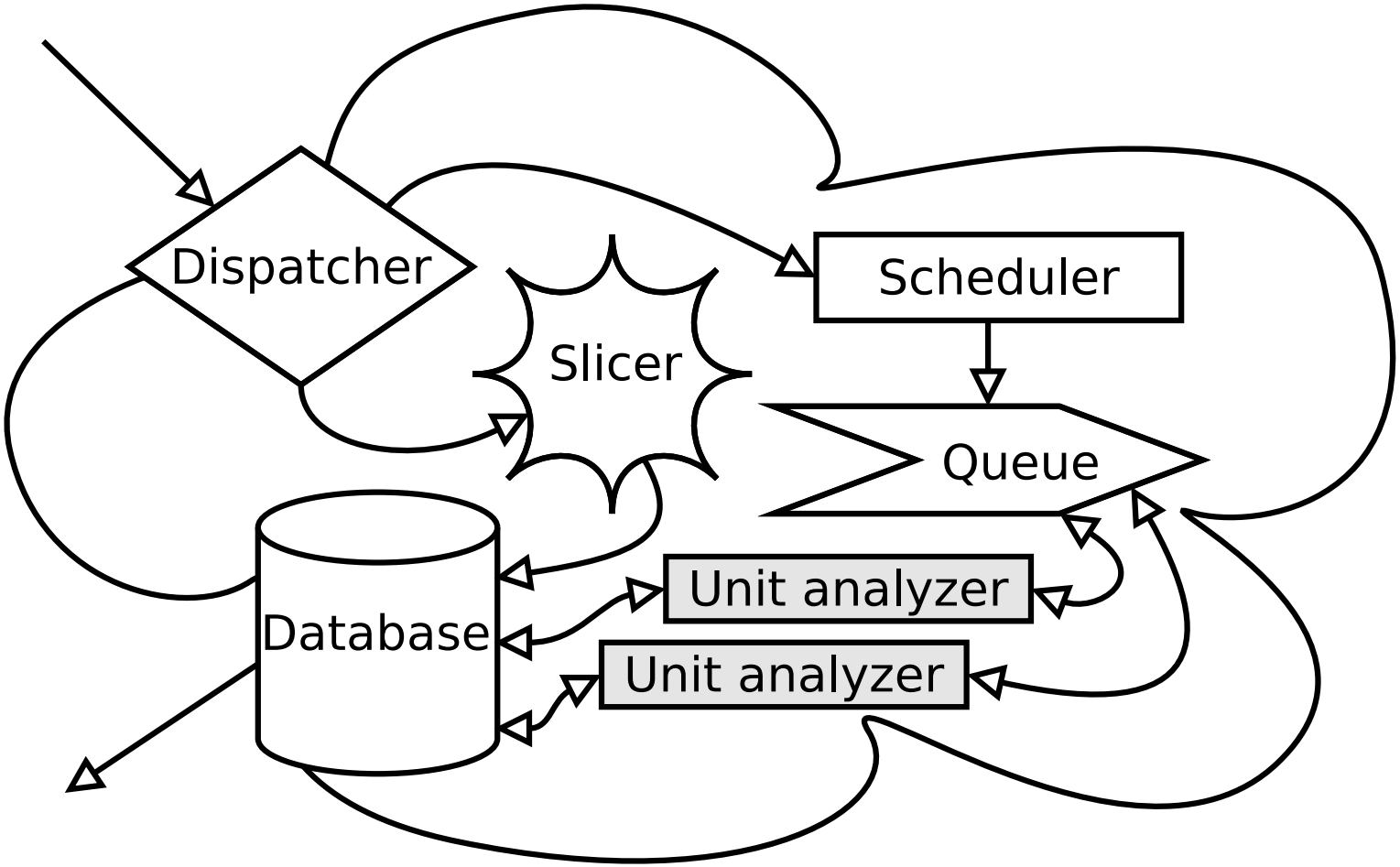
The client part

- very small (call the service, show the results)
  - used anywhere: smartphone, tablet [TouchDevelop]
- can issue parallel analyses on the same program, faster less precise results will come first, more precise ones later

Or a hybrid client:

- analyze "visible" code the developer machine for fast results
- analyze the rest on the Cloud and load results as the developer scrolls

What is in the cloud?



Parallelization

- Past experience: no gain in performance when parallelizing the analysis of a single method. Too much time lost in synchronization, ...
- Atomic computation: analysis of a method

Slicing

- Past experience: no gain in performance when parallelizing the analysis of a single method. Too much time lost in synchronization, ...
- Atomic computation: analysis of a method

- Goal: do not ship a big dll file to analyze just a part of it
- Given a .Net assembly and a set of methods M, generate a minimal analyzable unit (MAU) containing:
  - the methods M
  - fake versions of types/methods/properties/fields visible from M
  - their contracts, object invariants, contract classes
  - debugging information (pdb file)

Going further

- Goal: compute a global fixpoint over the analyses of all the methods
- Methods are not ordered anymore
- Method analyses are chaotic and asynchronous
- Asynchronous iterations [Cousot 77] converge to the greatest fixpoint with no synchronization!

Going further

- Goal: compute a global fixpoint over the analyses of all the methods
- Methods are not ordered anymore
- Method analyses are chaotic and asynchronous
- Asynchronous iterations [Cousot 77] converge to the greatest fixpoint with no synchronization!
- In the case of monotone operators, only!
- We do not have monotonicity
  - widenings, absence of best abstraction
- Problem can be remediated by forcing monotonicity

Summary: from desktop to the cloud

- make it parallelizable on a single machine (get rid of static variables, etc.)
- make it a service, even if the interface is very simple, i.e., an everlasting process waiting for queries
- optionally, use a centralized database for results and caching
- build the cloud service machinery: service workers, waiting queues, job schedulers
- find an axis of parallelization, with a medium granularity, e.g., some kind of slicing, or independent analyses
- depending on the analysis, global iterations may be needed to compute fixpoints

Issues?

- Trust

Issues?

- Trust

  Not so important since we analyze bytecode

Issues?

- Trust

  Not so important since we analyze bytecode

- Hardly predictable cost

Conclusion

- We (they) are working on Cloudot: a cloud-based version of Clousot, the static analyzer for .Net Code Contracts.
- The cloud enables:
  - fast and precise analyses of large programs thanks to the elasticity of the resources
  - sharing of computation and results
  - easier feedback on the usage of the tool
  - easier deployment of new versions of the tool and contracts for standard libraries

Try Clousot online: `http://rise4fun.com/CodeContracts`

Or download it (90,000+ downloads):
`http://research.microsoft.com/contracts`