Clousot today

- Runs on a single core on the developer box
- Input: assemblies + contracts
- The analysis
  - Order the methods according to the call-order
  - Analyze bottom up
  - Infer postconditions, necessary preconditions, and object invariants
  - Report warnings and verified code fixes
- Analysis of methods is sequential
The Goal

• Bring Clousot into the Cloud

Why?

• Exploit the massive number of processors and memory

• Faster and more precise analysis
  • Faster? Because methods are analyzed in parallel
  • More precise? See demo!
Let’s Demo!
Parallelization

• What we parallelize?
  • The analysis of a single method?
    • Past experience: no gain in performance
    • Too much time lost in synchronization/abstractions/…

• My internship: **Parallelize the analysis of the methods**

• Re-architecture of Clousot to make it distributed
  • Workers (analyzing methods) can be local or remote
  • Provide a structure (services) that can be easily ported to Windows Azure
High level algorithm for Cloudot

• Slice the input assembly into smaller assemblies
  • The minimum analyzable units
• Slices are pushed into a queue
• Workers pop slices from the queue and analyze them (as usual)
• Results are written into a shared database
  • Including the new inferred contracts
• All the dependent methods are re-added to the queue
Theories: Chaotic Asynchronous Iterations

- We do **not** compute an order on methods anymore
- Method analyses are chaotic and asynchronous
  - No synchronization!
- Compute a global greatest fixpoint (gfp)
  - Method contracts are (Top, Top) at the beginning and then refined
- Is it really the gfp?
- Th. [Cousot78] If the analyses are monotonic then it is the gfp
- Are our analyses monotonic?
Theory: Monotonicity…

• No, we do not have it. Two reasons:
  • Widening
  • Absence of best abstraction
    • Issue also for finite domains

• Have examples where inlining is less precise than modular analysis!
• Problem can be remediated by forcing monotonicity
• We can formalize all of this with nice Greek letters ;-)}
Back to practice: Services

• Same interface for all consumers
• Based on WCF (magic for Francesco)
• Can be hosted in:
  • A console, for debugging
  • A Windows service, for the Visual Studio extension
    • Speeds up small analyses by saving on Clousot initialization time (4 sec)
    • Our regression tests went 2x faster, without parallelization
  • A Web service, for RiSE for fun
  • A cluster or a cloud, for larger analyses
Slicer

• 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 smaller analyzable assembly containing:
  • The methods M
  • Fake versions of types/methods/properties(fields visible from M
  • Their contracts, object invariants, contract classes
  • Debugging information (pdb file)
Queue

• When we want to add a slice to the queue
  • Compute a snapshot: basically a hash of the inferred contracts found by prior analyses on the methods it depends on
  • If not already computed -> add it to the todo list
  • Remove all previous versions of this slice from the todo list

• Prioritize the queue as you want: FIFO, smallest, fastest slices first, etc.
Conclusions & Future

• Cloudot: a distributed version of Clousot
• Port it in the cloud: Azure? Amazon EC2? ...?
  • How difficult to port a .NET service into Azure?
• Balance slices using timing information from fixpoint computation
• Use inferred pre-state and post-state, object invariants (my prev. internship)
• Experience on large assemblies (Dynamics)
• ...
• Write a paper
Thank you!

In particular to Mike Barnett for all the hard work on the slicer!