Multiplicity Computing

Engineering Software for
Reliability, Performance, and Security

Alexander Wolf

Department of Computing

Imperial College London

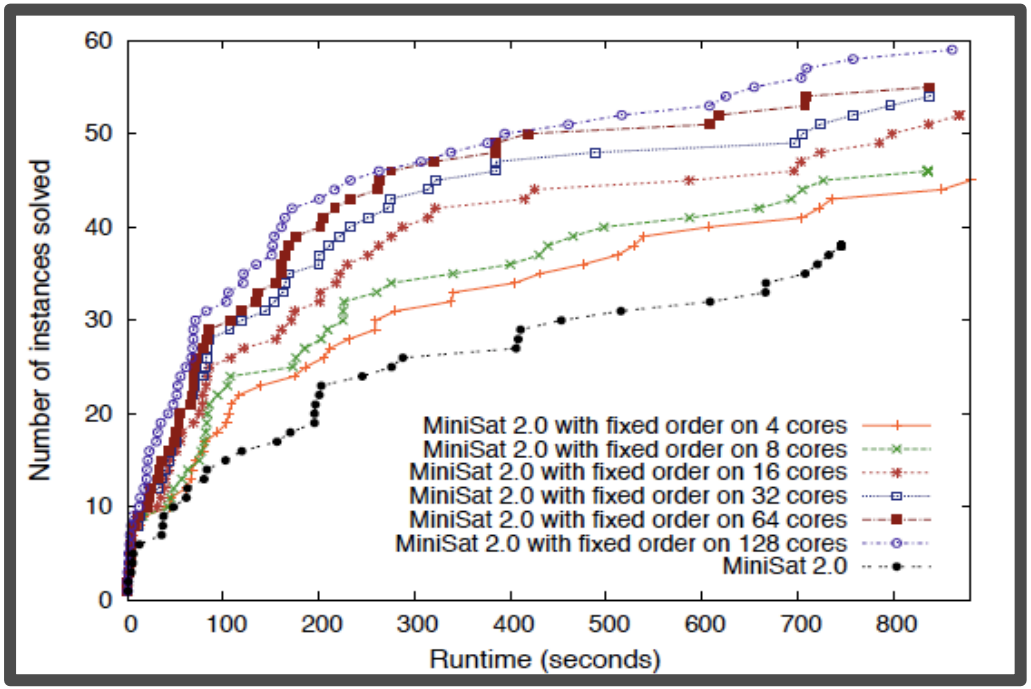
in collaboration with Cristian Cadar, Paolo Costa, and Peter Pietzuch

An inspiring example

- *Solving "hard" computational problems*
- *Idea:* run multiple instances of a SAT solver, each using a different parameter setting

  MiniSAT [Bordeaux et al., 2009]; ManySAT [Hamadi et al., 2009]

- *Result:* 128-core MiniSAT solves 55% more problems
- *Result:* ManySAT wins parallel SAT track at SAT-Race 2008 and SAT-Competition 2009



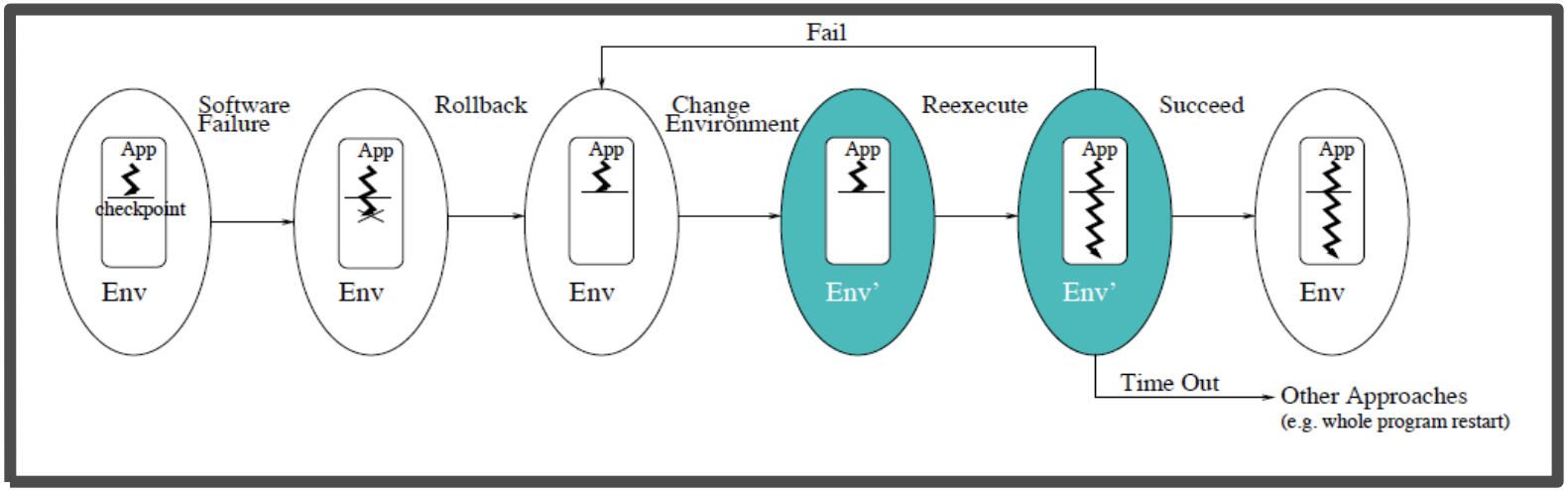
from Bordeaux et al., 2009

Another inspiring example

- *Improving reliability*
- *Idea:* "speculate" execution in different environments

  roll back on error; rerun in modified environment

  Rx [Qin et al., 2007]
- *Result:* certain hidden faults avoided in MySQL, Apache, CVS, ...



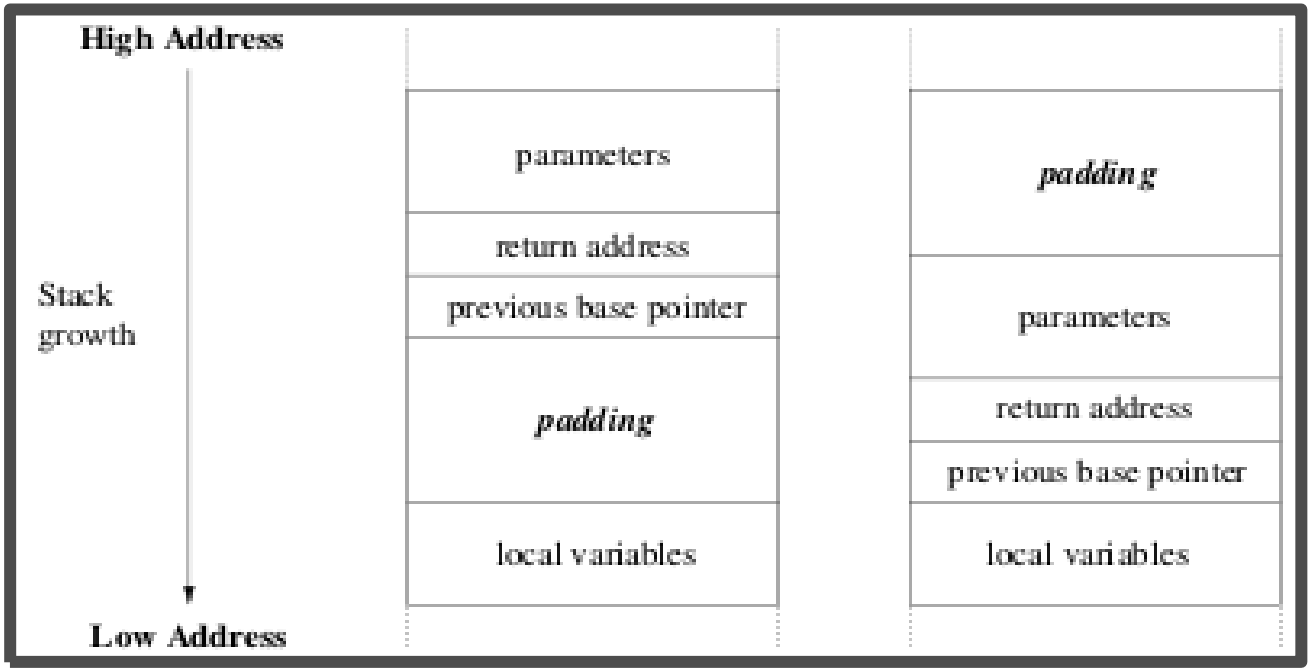
from Qin et al., 2007

Yet another inspiring example

- *Avoiding memory exploits*
- *Idea:* obfuscate addresses by randomizing memory layout in execution stack

  [Bhatkar et al., 2003]

- *Result:* success in defending against many kinds of buffer overflow attacks



from Bhatkar et al., 2003

What do we learn from these examples?

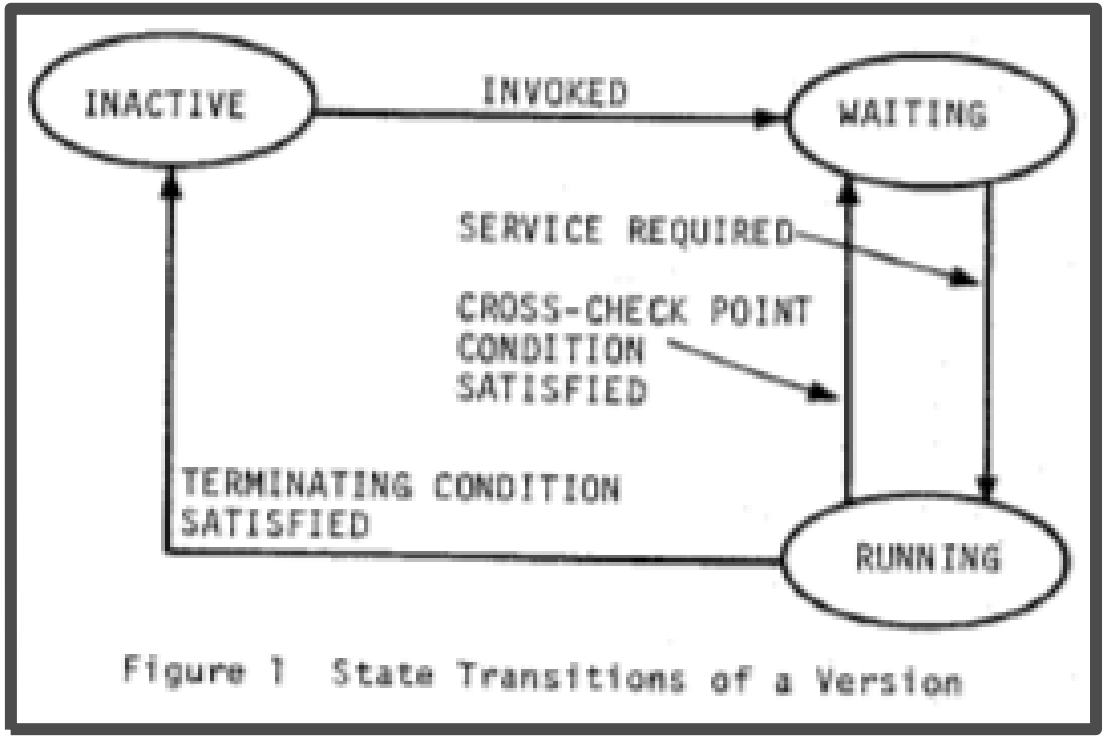
- Improvement achieved through *diversity*
- Good behaviors can emerge from *randomness*
- *Run-time techniques* sometimes better than design-time techniques
- Good choices are *situated* in context of use
- Sometimes easier to *detect* good/bad behaviors than to predict them
- Opportunity to exploit *parallelism* for non-parallel applications

An uninspiring example

- *N-version programming*
- *Idea:* programmers independently design/build to same spec, thereby avoiding common-mode failures

  [Chen and Avizienis, 1977]
- *Result:* shown to work in some special cases, but in general fails to achieve statistical independence

  [Knight and Levenson, 1986]



from Chen and Avizienis, 1977 (reprinted 1995)

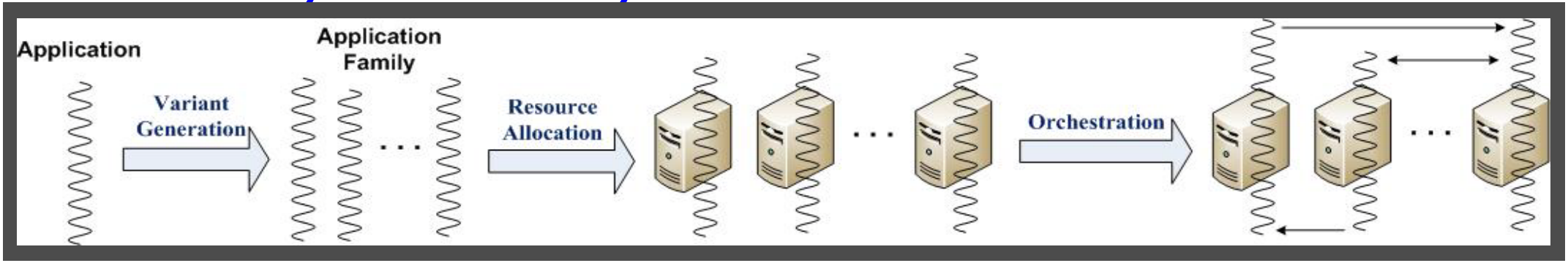
What do we learn from this example?

- Need much *richer sources* of diversity
- Need to *automate creation* of variants
- Need good understanding of *statistical properties* of variants
- Need some sort of *specification* or other source for constructing oracles

Multiplicity computing

Cadar, Pietzuch, and Wolf, 2010

- Tools, techniques, architectures, and languages for exploiting and managing diversity-based system execution



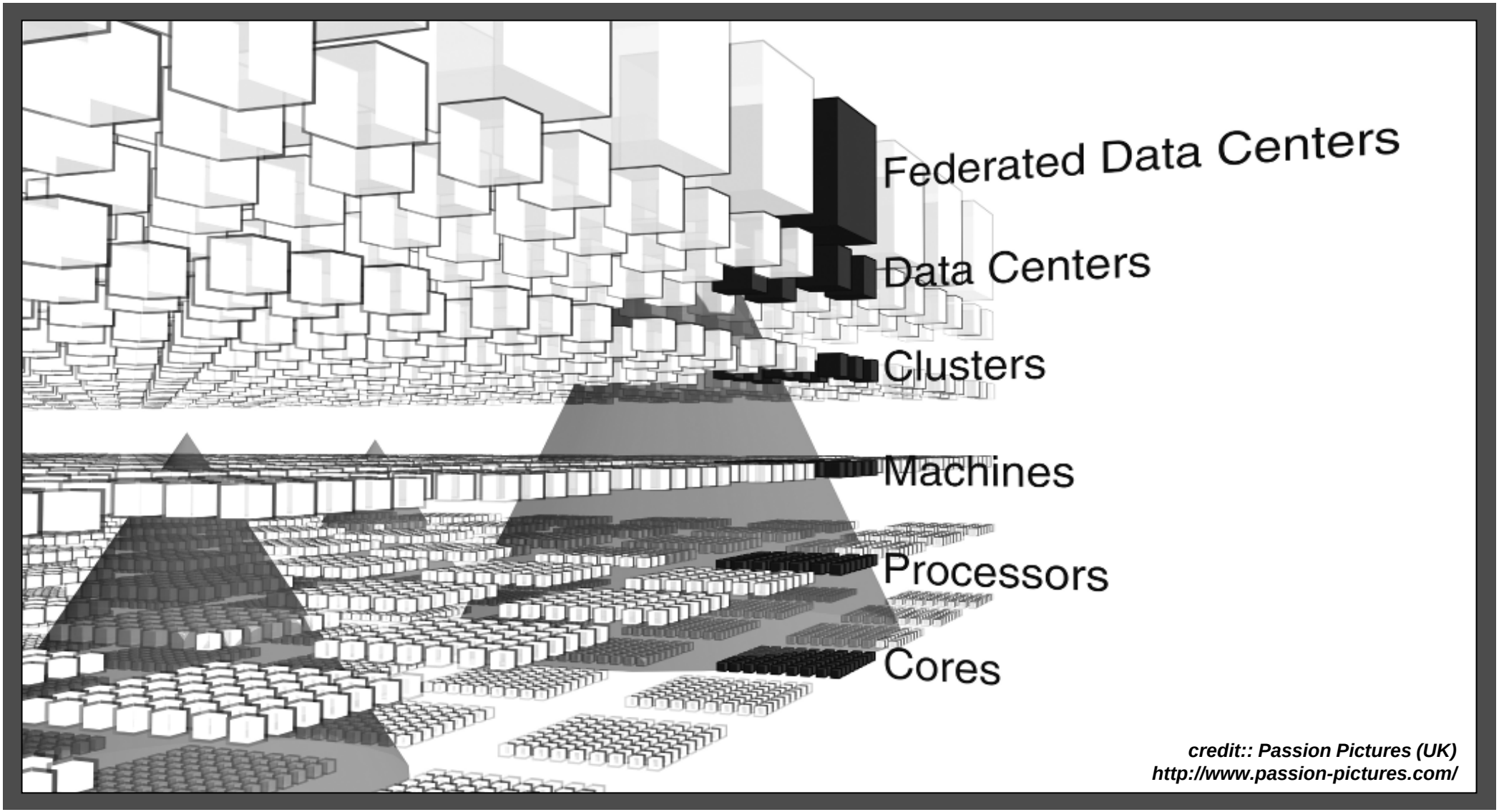
- design issues
  - finding exploitable diversity
  - automated creation of variants
  - design methods and architectures
- execution issues
  - run-time infrastructure for managing and executing variants
  - platform for managing resources

Many potential sources of diversity

Code	***Environment***
Code mutations	Configuration parameters
Data structure choice	Communication topologies
Library choice	Scheduling
Peep-hole transformations	Memory layout
GA transformations	Garbage collection
Symbolic execution	Message re-orderings
Computational precision	Time delays

Resources are not the issue

multiplicity of computation, communication, and storage



So what are the issues?

- Automatically generating variants
- Understanding statistical properties
- Managing lifetimes
- Managing resources
- Managing state, side effects, and interactions
- Developing reliability, performance, and security oracles
- Giving illusion of a single instance, even when multiple variants are executed (a “virtual app”)

What are some applications?

- Staged deployment
- In vivo (in situ?) experimentation/testing
- Optimal (parameter) tuning
- Patch selection/validation
- High-level speculative execution

Summing up

- Multiplicity computing continues shift of software engineering from a development-time activity to a run-time activity
- From absolute properties to “propabilities”
- Parallel execution for the other 99%
- Requires contributions from many disciplines
  - traditional and search-based software engineering, distributed middleware and operating systems, programming languages and run-time systems, evolutionary programming, ...