3rd UCAAT User Conference on Advanced Automated Testing

ETSI World Class Standards

Sophia Antipolis, French Riviera
20-22 October 2015

Estimating performance characteristics using PBT

Tobias Gedell - QuviQ

QuviQ

WHAT ABOUT PERFORMANCE ?

- Performance is important but difficult to test for.
  - Non-functional requirement.
  - A system might be performant enough for a certain amount of data but might not handle a big increase.
    - Difficult to predict what data will cause a system to become non-performant.

- We perform black box testing of a system to automatically estimate its ***algorithmic complexity***.

COMPLEXITY

- Consider a phone book. How long does it take to look up a specific person?

- Depends on a number of factors:
  - The number of entries in the phone book.
  - What person we are looking up.
  - How we go about finding the person.

COMPLEXITY

- We can go through all names from the beginning to the end.

- If there are 100 names, we will on average have to check 50 names. If there are 50,000 names, we will have to check 25,000.
  - In the general case with N names we will have to check N/2.

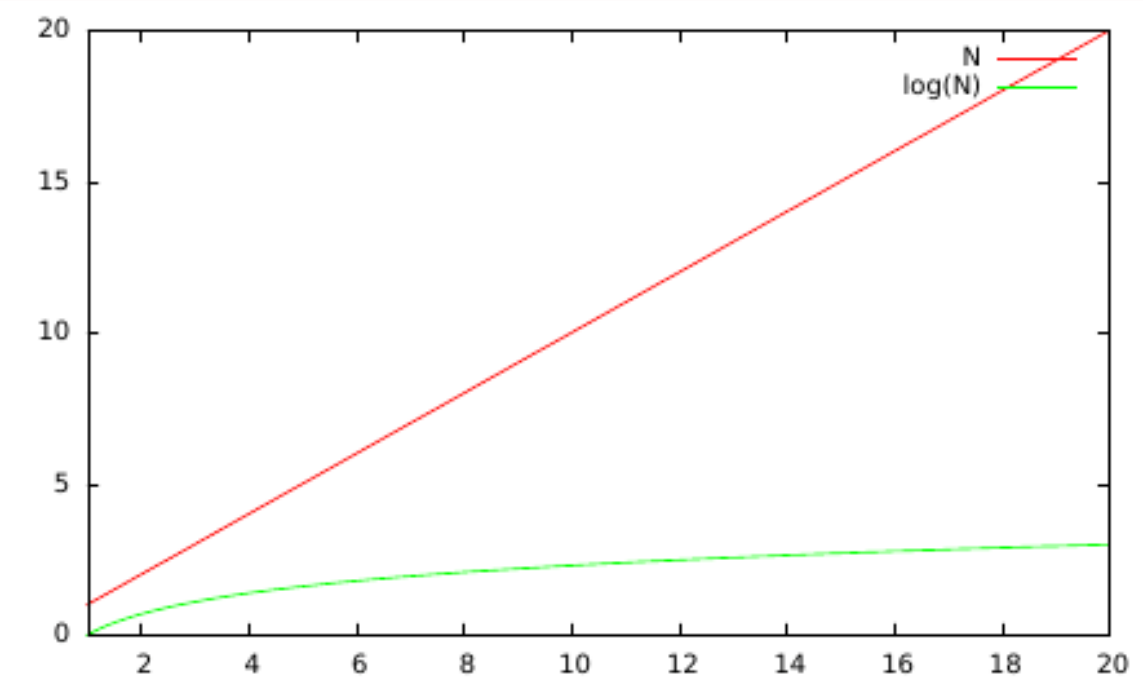
COMPLEXITY

- We can also make use of the fact that the names are sorted and open the book in the middle. We then know which half to continue looking in and do the same again until we have found the name.

- If there are 100 names, we will on average have to check 7 names. If there are 50,000 names, we will have to check 16.
  - In the general case with N names we will have to check $\log_2(N)$.

COMPLEXITY

We are interested in the trend, not individual data points.



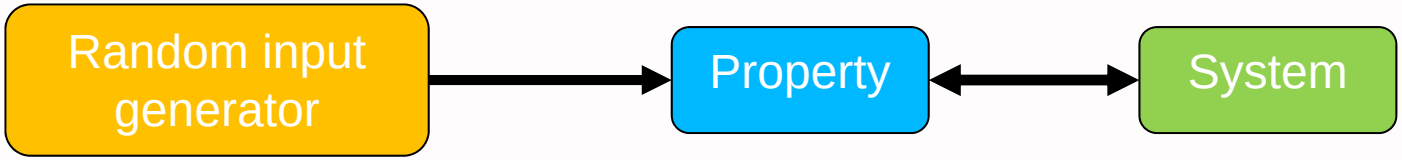
COMPLEXITY

- The complexity of a system tells us how it scales with an increase in its input.

- Not straightforward to find by normal testing.
  - A test case with few entries in the phone book is unlikely to have displayed any performance problems.
  - A test case with a name occurring in the beginning would also not have displayed any problems.

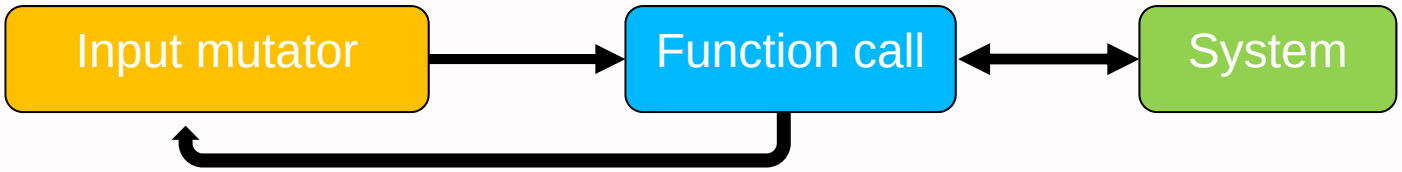
QUICKCHECK

- We make use of QuickCheck, which is a tool for performing property based testing.



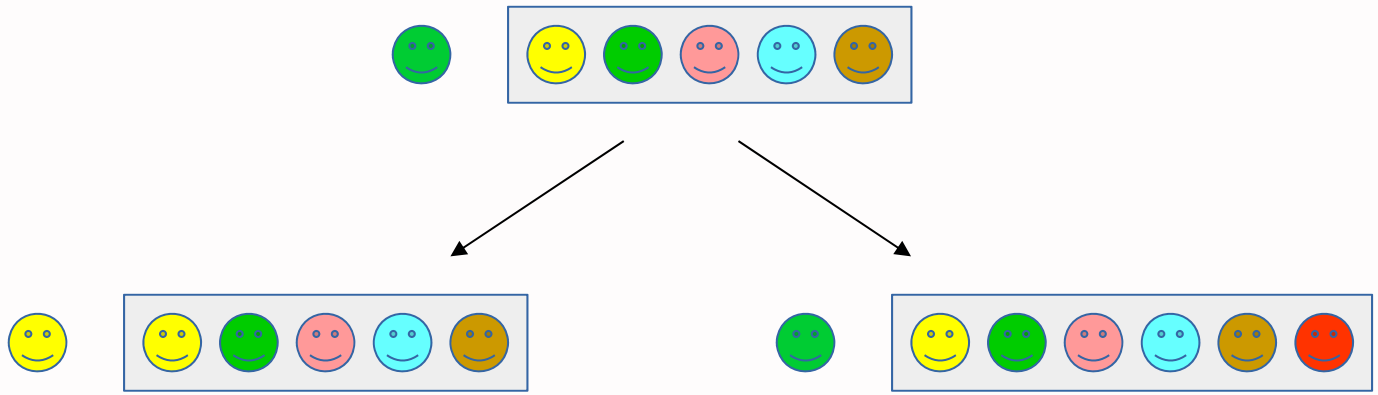
QUICKCHECK

- To automatically discover the worst case input for a system, we steer the random generators by iteratively mutating the input, while increasing its size, and keeping the worst candidates.



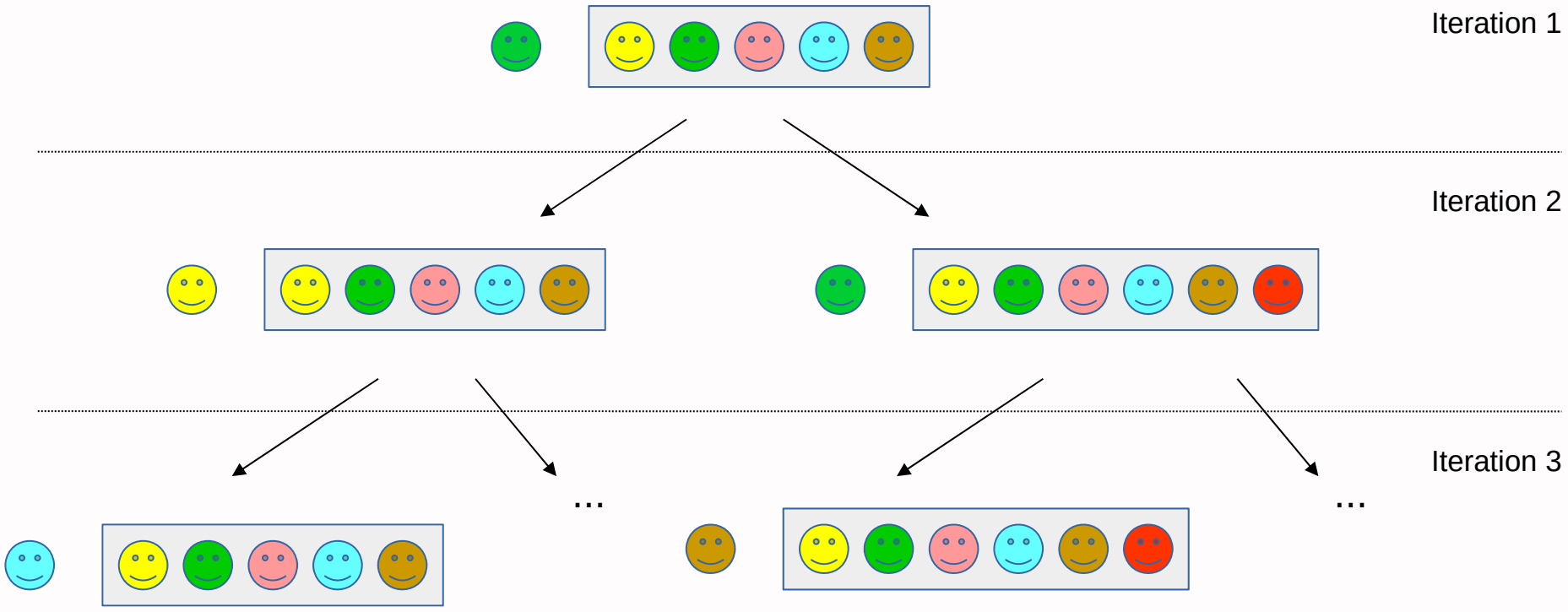
QUICKCHECK

- In the phone example there are two ways of mutating the input data:
  - Change the person we are looking for.
  - Insert another person in the phone book.



QUICKCHECK

This allows us to build a mutation tree.



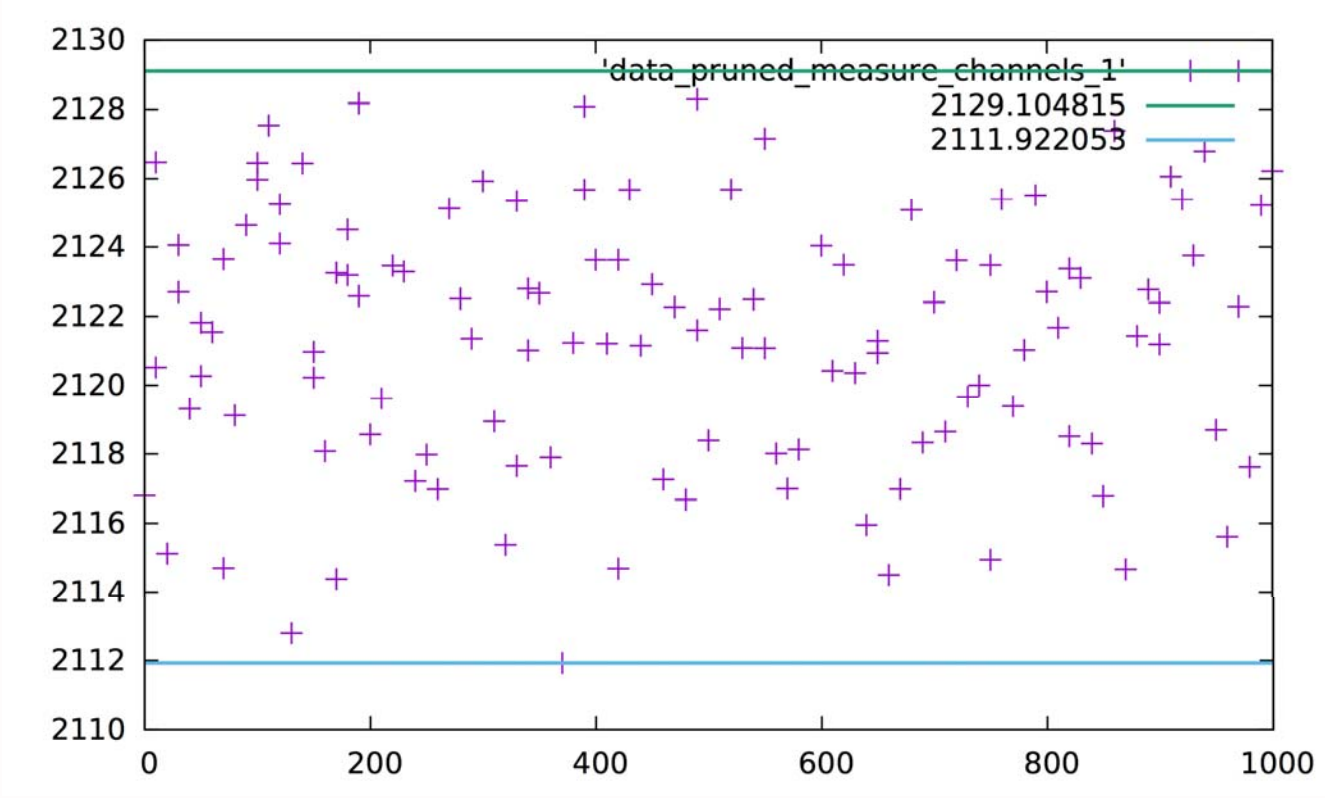
VoDKATV

- We have evaluated our method on VoDKATV which is a commercial video on demand platform developed by Interoud.

- We focused on a number of performance critical operations.
  - Get the available TV channels, user preferences and EPG from three different databases.
  - Combine and process the data.
  - Generate the JSON that is returned to the user.

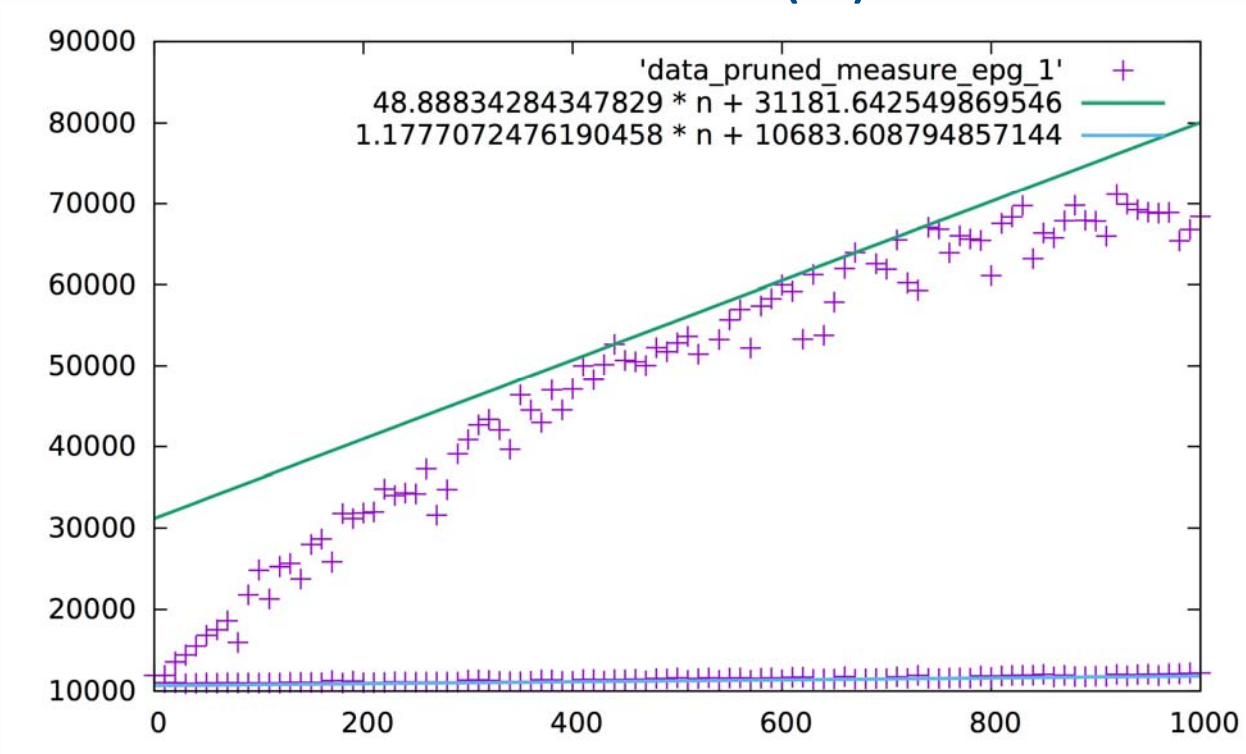
VoDKATV

Getting the TV channels from the database – O(1).



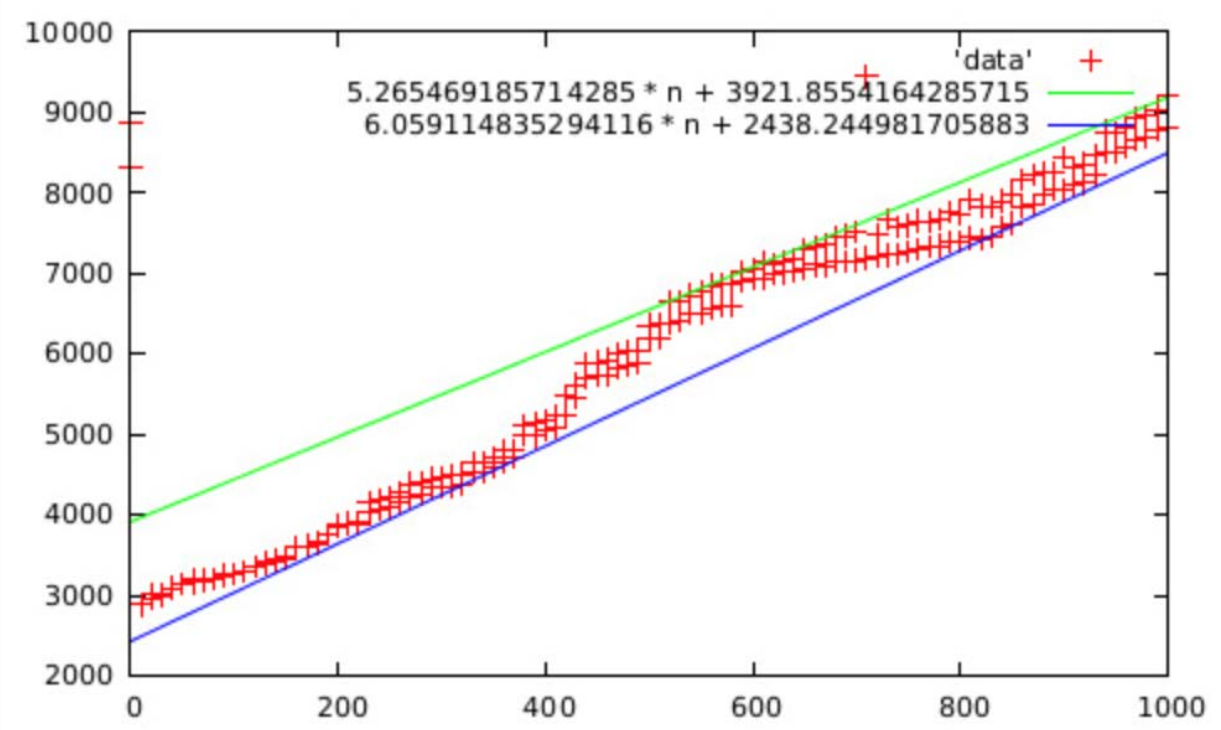
VoDKATV

Combine the TV channels, the preferences and the EPG data – O(N).



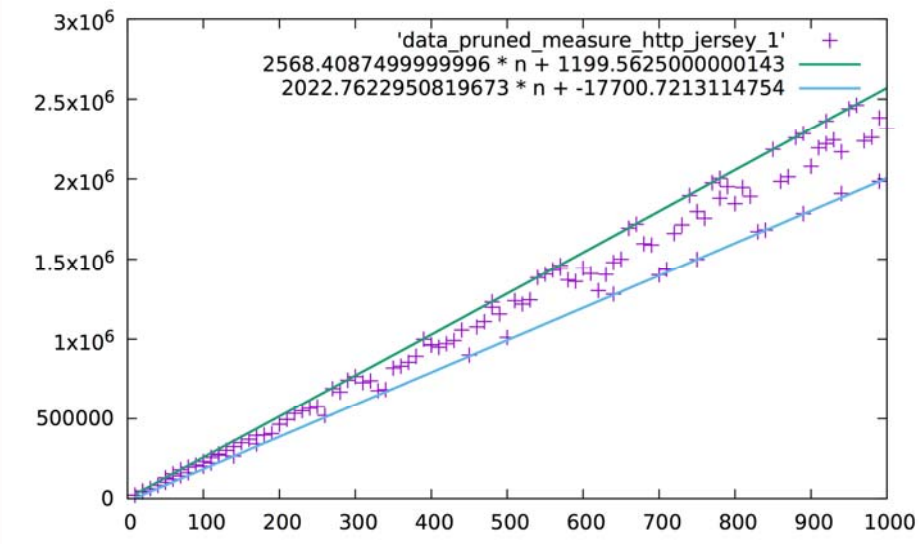
VoDKATV

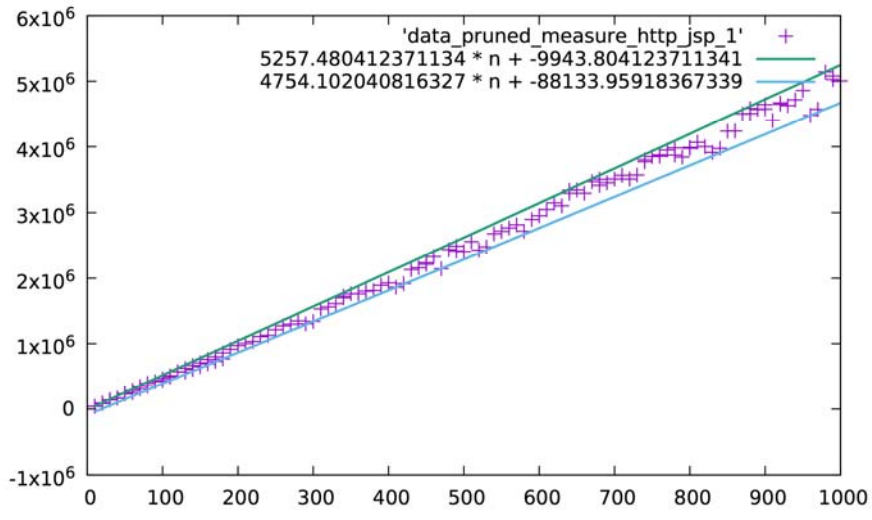
Caching the EPG data – O(N)!



VoDKATV

Comparing two Java JSON libraries: Jersey and JSP





Conclusions

- The results are promising.
  - Faster and more accurate than trying to infer the complexity manually.
  - Works on black box systems.
  - Useful to both find performance bugs, like the unexpected complexity of the EPG cache, and to make implementation decisions, like choosing which JSON library to use.