BALLISTA

The Dimensionality Model for Characterizing Software Robustness

Jiantao Pan
ECE Department
Carnegie Mellon University
jpan@cmu.edu

Institute for Complex Engineered Systems

ECE

Carnegie Mellon

DARPA

Introduction

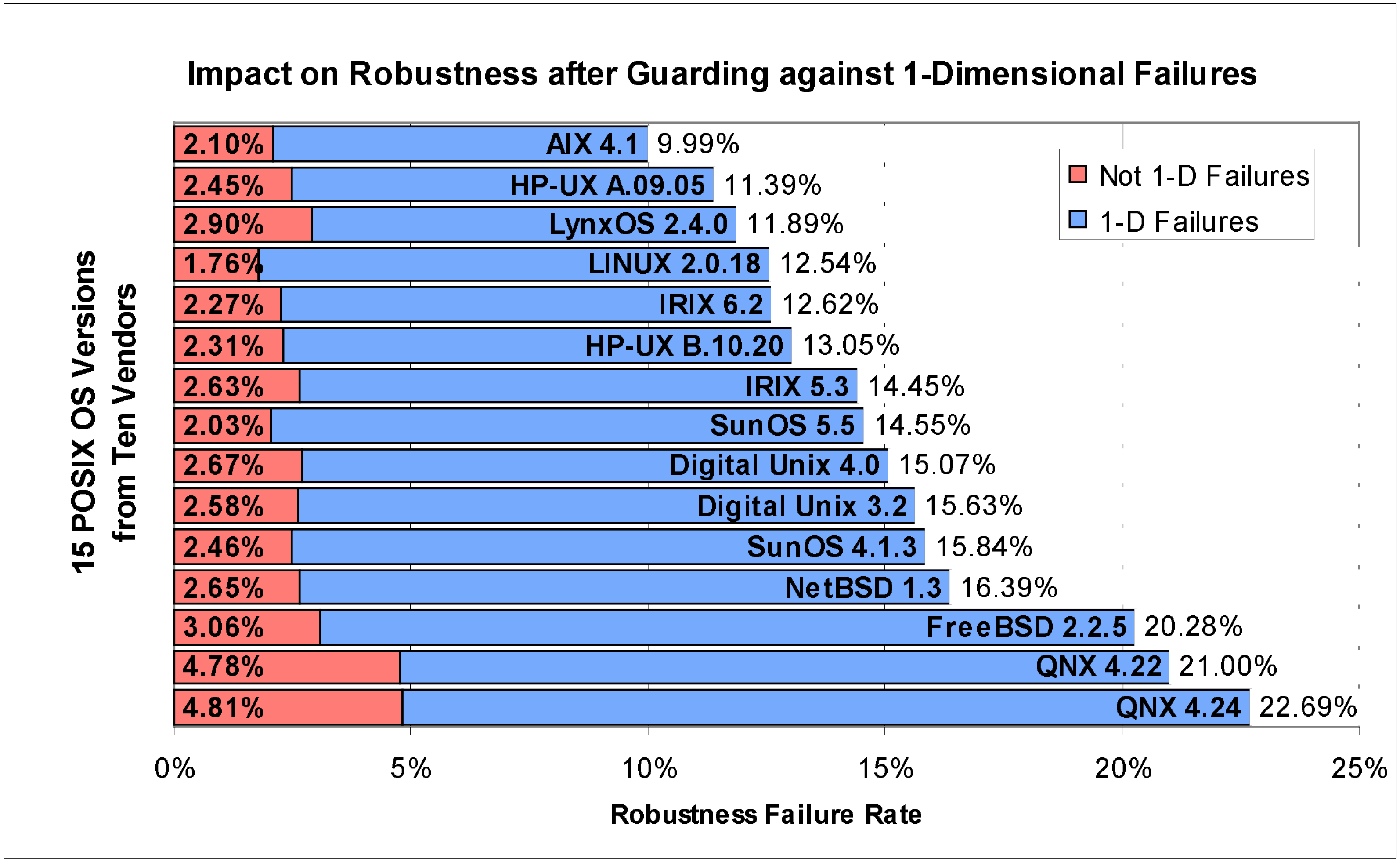
- **Software robustness matters**
- **The Ballista project**
  - Testing & hardening COTS/legacy software modules
  - 1.1 million data points on 15 POSIX OSes
  - The Dimensionality Model: for finding failure patterns
- **Potential uses of the model:**
  - Guiding robustness testing
  - Guiding robustness failure protection

The Dimensionality Model

- **Definitions:**
  - Parameter dimensionality: number of arguments accepted by a software module
  - Robustness failure dimensionality: number of parameters contributing to the failure
- **Examples**
  - **3-D parameter dimensionality:**
    - `read(file_descriptor, buffer, bytes_to_read)`
  - **1-D failure:** `read(NULL, —, —)`
    - NULL `file_descriptor`
  - **2-D failure:** `read(—, 16K, 64K)`
    - `buffer` smaller than `bytes_to_read`

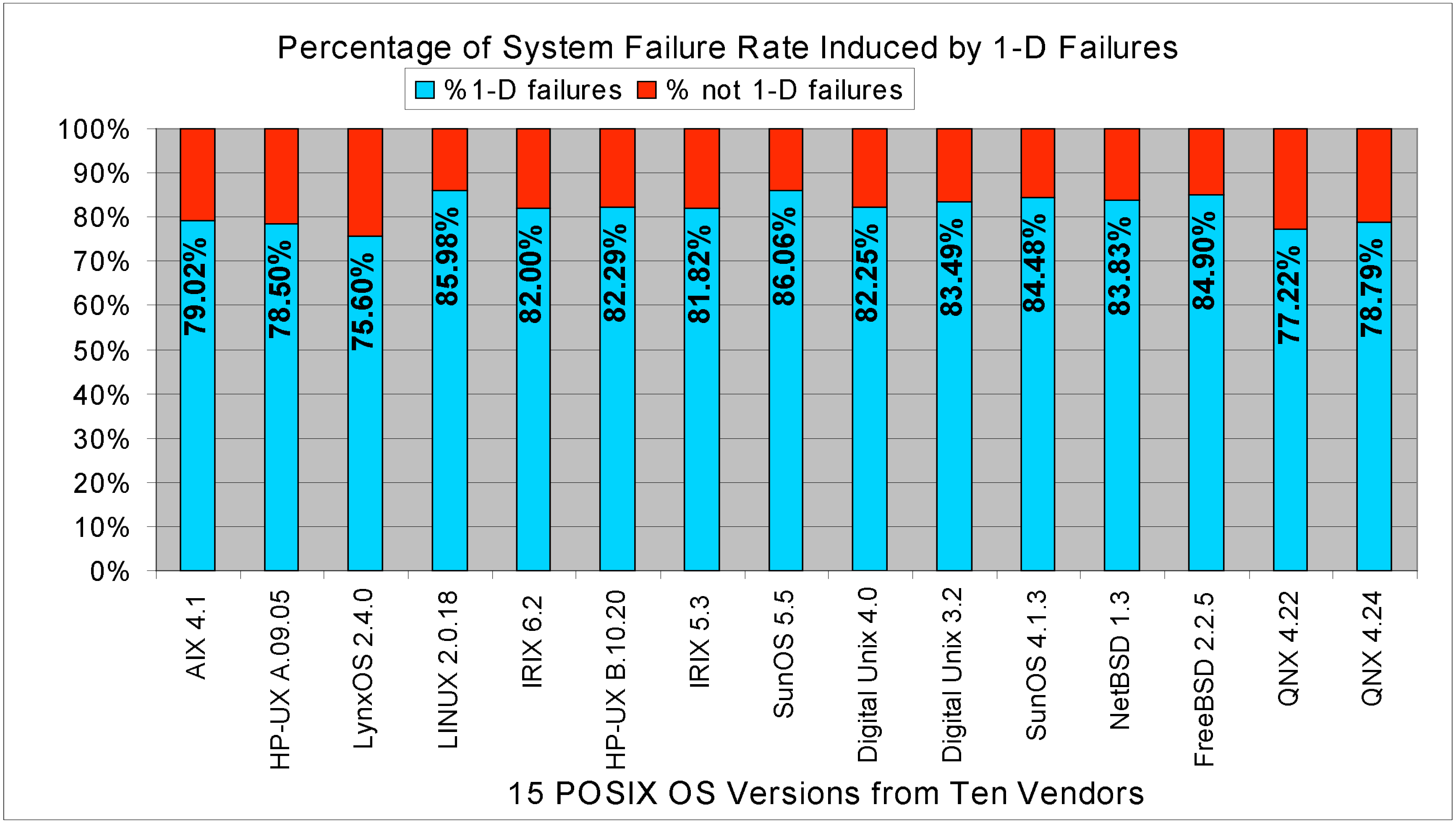
Low Dimensionality Failures Prevail

- **If we can eliminate 1-D failures, average failure rate drops from:**
  - **15.2 %** ↘ 2.8%



Low Dimensionality is Common

- **All operating systems tested exhibit similar phenomena**
  - Average 82% (standard deviation 3.24%) failure rate is attributed to 1-D



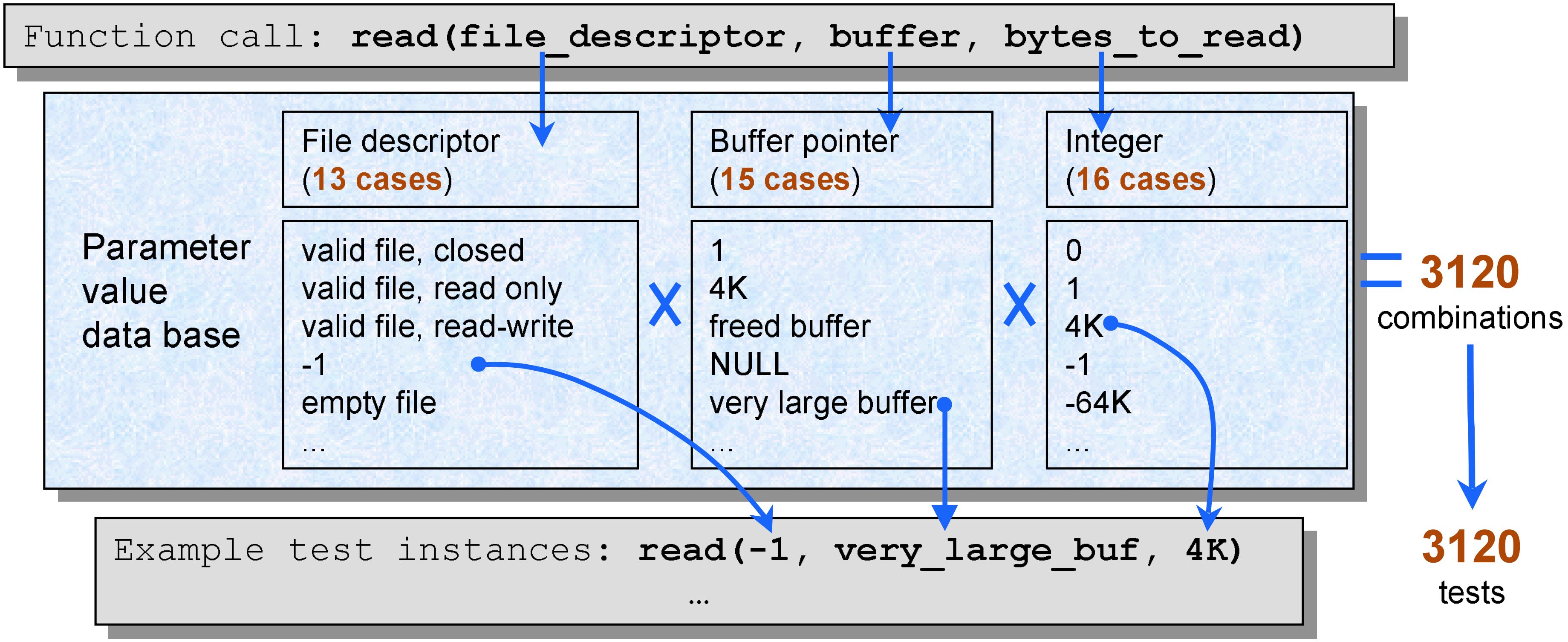
Conclusions

- **Most failures we have seen are 1-Dimensional**
  - Prevalent across a wide range of POSIX OS APIs
  - Confirms hypotheses of testers (AETG, *etc.*)

- **The Dimensionality Model**
  - Analysis method for API level robustness failures
    - Generic analysis method for other applications?
  - Might be used to guide automated testing
    - Potentially cost-effective
  - Good for robustness hardening?
    - Automated robustness hardening guided by dimensionality analysis

Testing Methodology [backup]

- **Feed combinations of valid and invalid inputs to POSIX calls**
  - Assume no access to source code (black box)
  - Single call per test for simplicity, ignore interactions and timing
  - Testing method intended to work on other Commercial Off-The-Shelf(COTS) software



Function call: **read(file_descriptor, buffer, bytes_to_read)**

Parameter value data base

File descriptor (**13 cases**)

valid file, closed
valid file, read only
valid file, read-write
-1
empty file
…

X

Buffer pointer (**15 cases**)

1
4K
freed buffer
NULL
very large buffer
…

X

Integer (**16 cases**)

0
1
4K
-1
-64K
…

= **3120** combinations

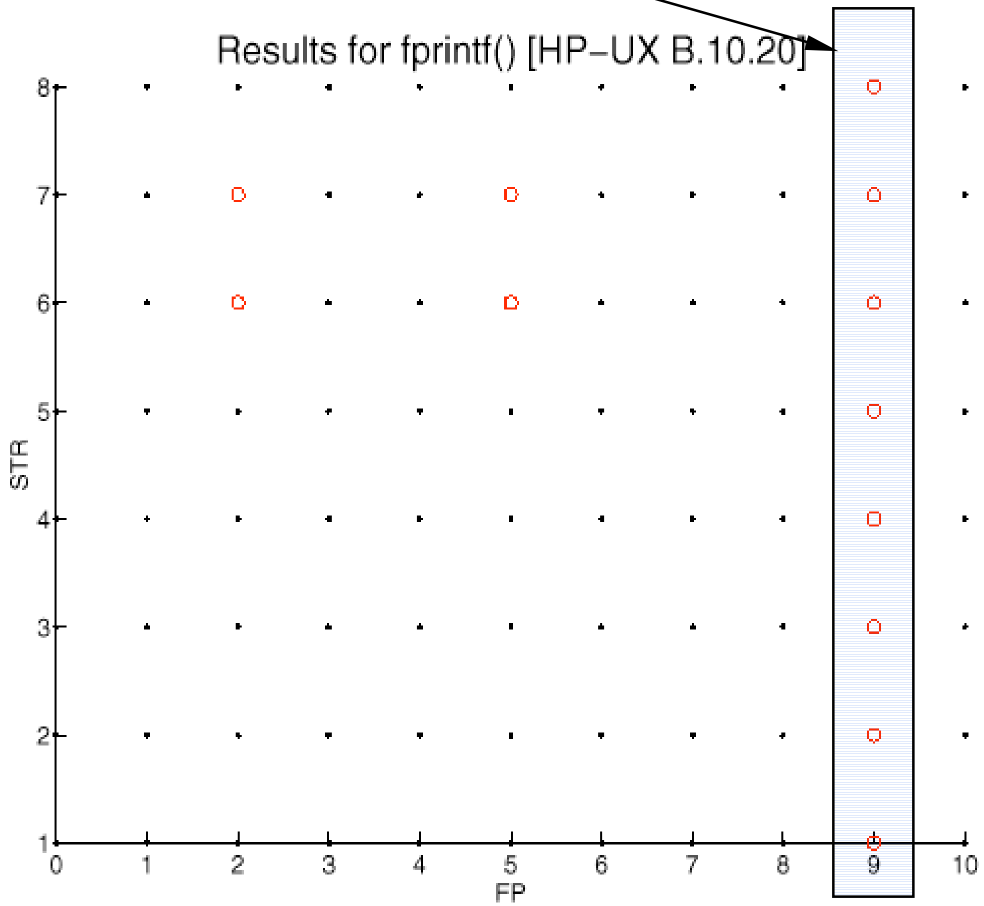
3120 tests

Example test instances: **read(-1, very_large_buf, 4K)**
…

Patterns of Testing Result [backup]

- **`fprintf(File_Pointer, STRing)` in HP-UX**
- **1-D failures:**
  - They form a line in a 2-D function (function that parameter dimensionality=2)
  - They form a hyperplane in a n-D function

All 1-D failures this line



● Pass (Success, Error code)

○ Robustness Failure (Catastrophic, Restart, Abort)