Run-time test configurations for load testing

Gábor Ziegler,
Ericsson Hungary Ltd.

Contents

- Introduction
  - What is TITANSim
  - Motivation for TITANSim
  - Functional description of the parts of TITANSim
    - CLL, Application Libraries and Control Logic
  - Three different perspectives of TITANSim:
    - HW, SW, Run-time configurations
- Comparison of the possible test configurations
  - “local scheduling” vs. "central scheduler PTC"
- Conclusions and questions
Introduction

About TITANSim

- TITANSim is aimed at performance testing
  - With TTCN-3 as the specification language
  - With Ericsson’s internal TTCN-3 Executor and Compiler tool

- Performance testing needs highly optimized test suite code

- Two contradicting requirements:
  - A framework shall be general $\rightarrow$ generalization
  - Optimization is task specific $\rightarrow$ specialization
Motivation for TITANSim

- TTCN-3 and TITAN is widely used for functional testing throughout Ericsson
- TITANSim aims to achieve cost savings via
  - reuse of the function test code base
  - reuse of testers' competence
  - reuse of existing, in-house tools
  - reuse of the new performance test solution by different projects throughout the company

Introduction

Functionalities and applications of TITANSim
Functionality of TITANSim CLL

- Run-time interaction with the test suite
  - A dynamically configurable run-time GUI and...
  - ...Parameters
  - ...Statistics
- Generic support for common programming tasks
  - Memory management support: “resource” pools
  - Scheduling support
- A logging framework
- Generic support for distributed scheduling:
  - EventQueue data type + support functions
- Concrete support for central scheduling
  - Ready-made scheduler component for central scheduling
  - Load balancing, regulated load, external execution control, traffic-mixer and a graphical console for all these
- Other useful data types and algorithms:
  - Linked lists (FreeBusyQueue), hash tables, binary search tree (Red-Black trees)

Functionality of Application Libraries and Control logic

- Application libraries: simulated entity specific tasks
  - Protocol message handling
  - Inbound message routing in case of multiple generator PTC
  - Protocol specific TITANSim parameters and TITANSim statistics
  - Building blocks and state-machine support for Control Logic
- Control logic: realization of particular traffic cases
Introduction

The 3 perspectives of TITANSim

Views of a TTCN-3 load test library

- At least 3 perspectives have to be considered:
  1. HW perspective: which hardware to use?
  2. SW perspective: how to modularize your code?
  3. Run-time perspective: what is the best run-time test-configuration?

This talk’s main topic
Run-time test configuration for load testing

1. The HW perspective

- SW and HW are separated
- One SW – many HW
- To expand load capacity only HW units shall be added

2. The SW perspective

- Three levels approach:
  - “Core Load Library” (CLL)
    - provides the generalization
  - “Application specific framework libraries” (AppLibs)
    - provides the "specialization"
    - code provided in-cooperation with project experts, relies on core library code
  - “Control logic”: can provided by non-experts, as well
    - builds on both application specific and core libraries code
3 Run-time configuration perspective

- Careful trade-off must be made between
  - Efficiency
  - Resulted code complexity

- Load testing means concurrency handling:
  - Many(!) parallel traffic flows…
  - …over some shared resource pools!

- TTCN-3 has a special concurrency model
  - PTC-s are run concurrently
  - A PTC is a "single CPU system"
    - No concurrency support below PTC level by the language
  - PTC-s are run isolated from each other
    - no shared memory

- Our dilemma: on which level do we handle the concurrency?

Alternatives for run time configurations

- The "simple" approach is follow the "usual" TTCN-3 semantics
  - Concurrency is to be handled on PTC-level
  - A single PTC is responsible for a single transaction

- The "advanced" approach is to let a single PTC handle multiple concurrent transaction
  - Concurrency is to be handled below PTC-level
  - A single PTC is responsible for multiple transactions
TITANSim Run-time test configurations

Central scheduling
Central scheduling

**Pros**
- It is the most user-friendly
- It requires no coding-paradigm change with respect to function tests
- Suitable for ad-hoc load testing projects
- It can be used without an application library
  - Can use ready-made scheduling functions that are totally independent of load generator component-type

**Cons**
- Less efficient
- Wrong scalability
  - Each traffic initiation requires internal communication — with a single central entity: extra overheads and delays
  - Sharing data of a run-time database across traffic cases / entities is difficult and inefficient

TITANSim Run-time test configurations

Local (distributed) scheduling
Local (distributed) scheduling

- **Pros**
  - It is the most-efficient
  - Less dependent on OS-scheduler
  - Sharing data of a run-time database across traffic cases of the same PTC can be easy and efficient
  - Load generator PTC-s schedule on their own
    - no internal communication overhead needed for load generation
    - They can run autonomously → scalability!

- **Cons**:
  - It requires some sort of an application library: Explicit concurrency handling shall be set up
  - It requires coding-paradigm change with respect to FT: event-based logic
  - Writing and using reusable ready-made scheduling algorithms requires dirty-tricks w.r.t. TTCN-3 language
Thank you for your attention!
Questions?

ERICSSON
TAKING YOU FORWARD