



deterlab
based on emulab

Towards a RAMP-based Workload Generator

Archana Ganapathi
Anthony D. Joseph

UC Berkeley



deterlab
based on emulab

DETER Users Need High-Performance Workload Generation

- Workload generation as a tool for:
 - System stress-testing
 - Debugging
 - Identifying system bottlenecks
 - Evaluating performance relative to other systems
- Desired features for workload generators
 - Scalability: Generate Google/Amazon-scale load
 - Configurability: Create new workload generators
 - Response-driven and response-ignorant workloads



State of the Art Workload Generators

- **Hardware vs. software based**
 - Hammer, Optixia vs SURGE, SLAMd
- **Tunability vs Automation**
 - SPECweb, TPC-W, Harpoon vs Optixia, SLAMd
- **Realistic vs Synthetic**
 - SURGE, SLAMd, Harpoon vs TPC-W
- **Generic vs App-Specific**
 - SLAMd, Harpoon vs TPC-W, Hammer
- **Open vs Closed**
 - Most are closed



deterlab
based on emulab

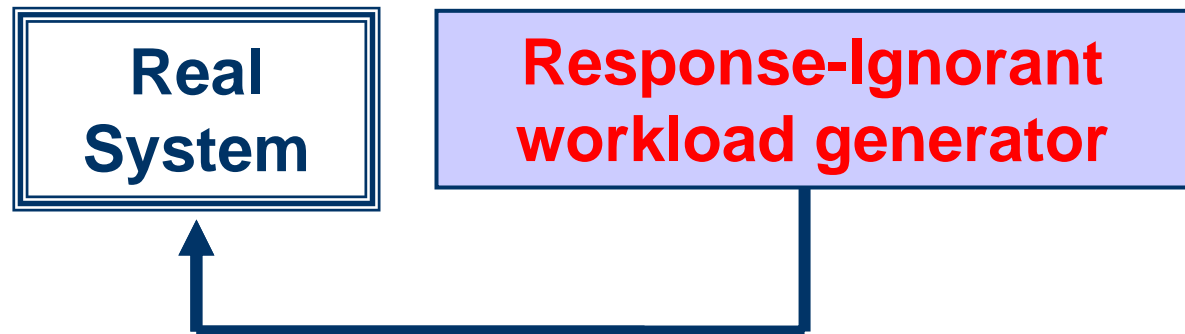


Research Accelerator for MultiProcessing (RAMP)

- Bee2 FPGA-based board
 - 5 Xilinx Virtex 4 FPGAs with 20 10GE, 4x2GB DRAM
 - O(100) simple CPUs & network, O(\$100) per CPU
- Alternate use – HW-based traffic generation
 - Limits: DRAM BW used for state mgmt, and pkts sent on 20 10GE ports
 - Peak: 40-100M web reqs/sec (R-I), ~3GByte/s
 - Google: ~200M search reqs/day, ~2000 reqs/sec



Building Block Layer 0: Response-Ignorant



- RAMP as an Integrated Emulation + Workload Generation System
 - Workload generation engine built on top of RAMP
- Designed this spring as a graduate student project



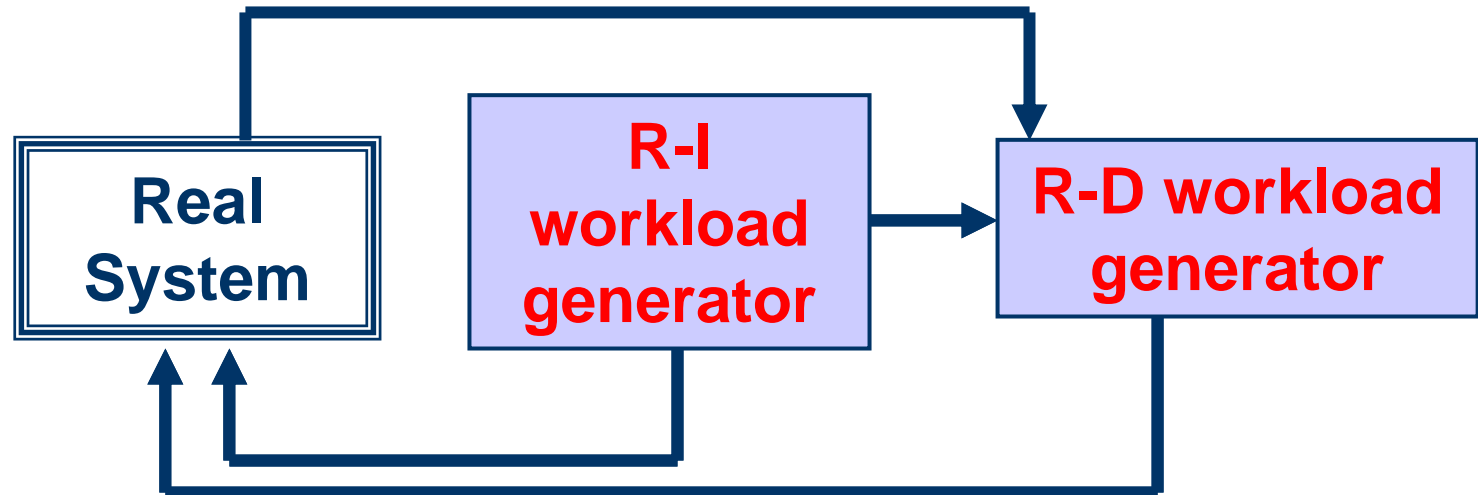
Version 0 – Response-Ignorant

(Spring 2006 CS252 class project by Lorenzo Orecchia and Madhur Tulsiani)

- Generated the dataset using analytical models and sampled from it in hardware
 - Server file size distribution
 - Request size distribution
 - Relative file popularity
- Derived URL connectivity graph and loaded it into memory
- Used circuit logic to perform random walk on graph
- *Achieves scalability within HW constraints*



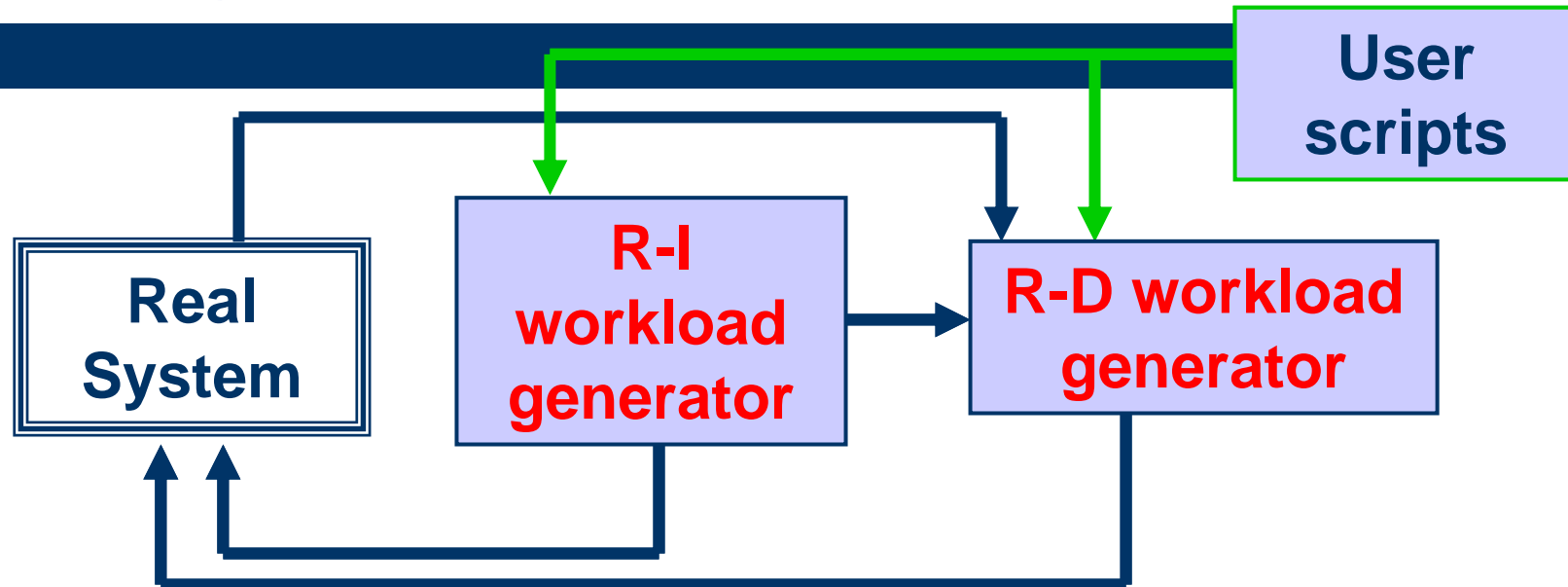
Building Block Layer 1: Response-Driven



- Handle server responses & generate follow-up request
 - Include “server response” states in logic
 - HW parser determines current state
- Gradient of what to parse (errors, congestion, timing, ...)
 - User think-time distribution + server response time



Building Block Layer 2: WDL



- Create *Workload Description Language* to specify primitives to compile onto an FPGA
 - Request distributions
 - Think-time distributions



Some Open Questions

- Limits on types of workloads?
- Workload trace sources?
 - PREDICT, existing traffic generators, ...?
- Role of Response-Ignorant trace generation?
 - UDP, error/congestion-free TCP, ...?
- Required level of fidelity for Response-Driven trace generation?
 - How much of TCP FSM to model?

Archana Ganapathi (archanag AT cs.berkeley.edu)

Towards a RAMP-based Workload Generator

Archana Ganapathi
Anthony D. Joseph

UC Berkeley
