

SAFE INSPECTION AND CUSTOMIZATION OF LIVE VIRTUAL MACHINES

Sahil Suneja, Cristiana Amza, Eyal de Lara
University of Toronto

Ricardo Koller, Canturk Isci
IBM T.J. Watson Research Center

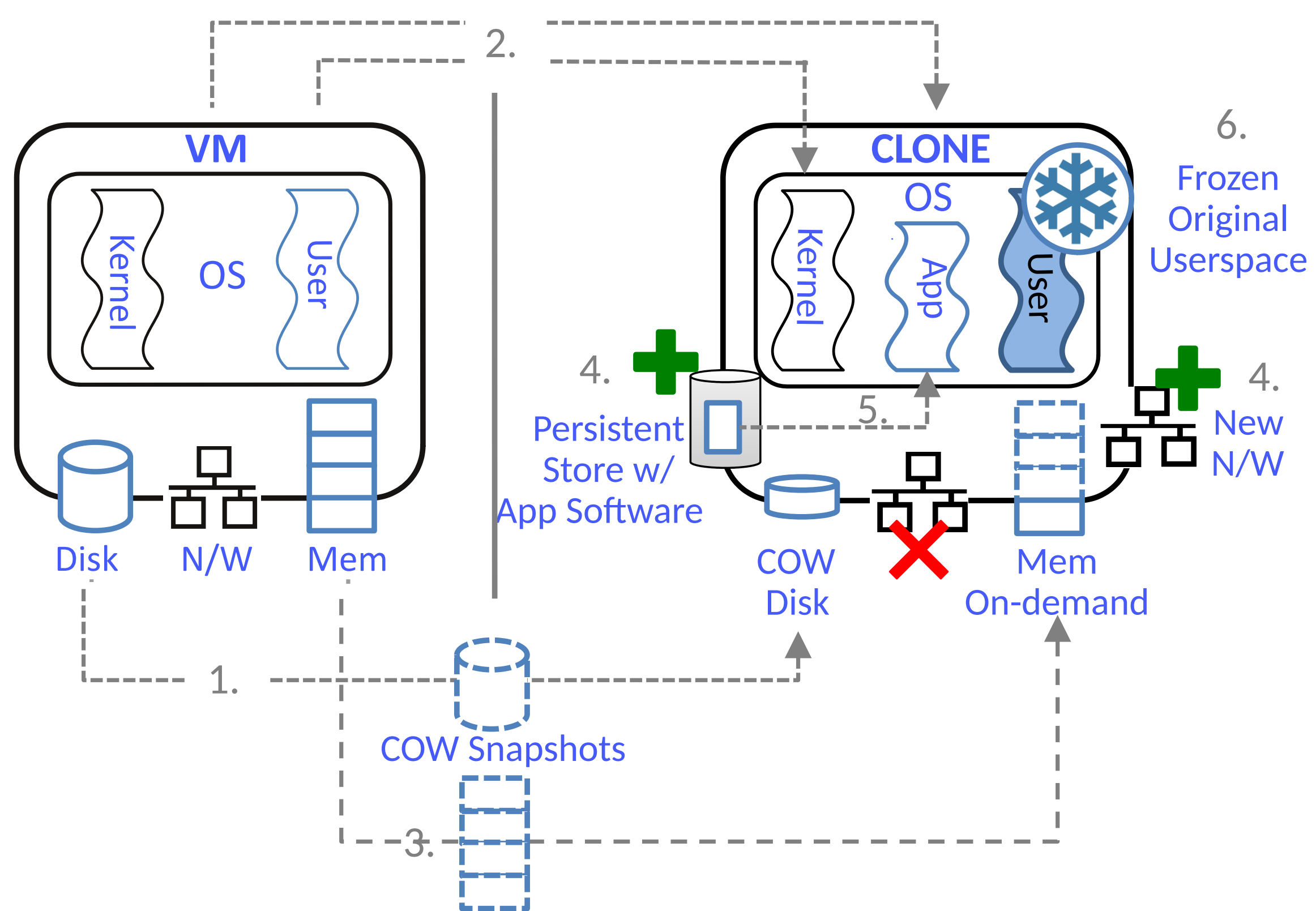
MOTIVATION: LIVE CUSTOMIZATION

- **Production VM = Hands-off entity**
 - restrictions during inspection and customization
- **Analyze, tune and add new functionality**
 - without fear of breaking production systems
 - dependent on runtime state, lacking during testing/staging
 - ex. system monitoring, tuning config params to live load, diagnosing perf problems

THE 'CIVIC' APPROACH

- **CIVIC: Cloning and Injection based VM Inspection and Customization**
- Live clone source VM; clone VM acts as proxy
 - Access to runtime state
 - Intrusion / impact isolated inside clone sandbox
- Add customization functionality via code injection
 - Guest saved from modification and cooperation

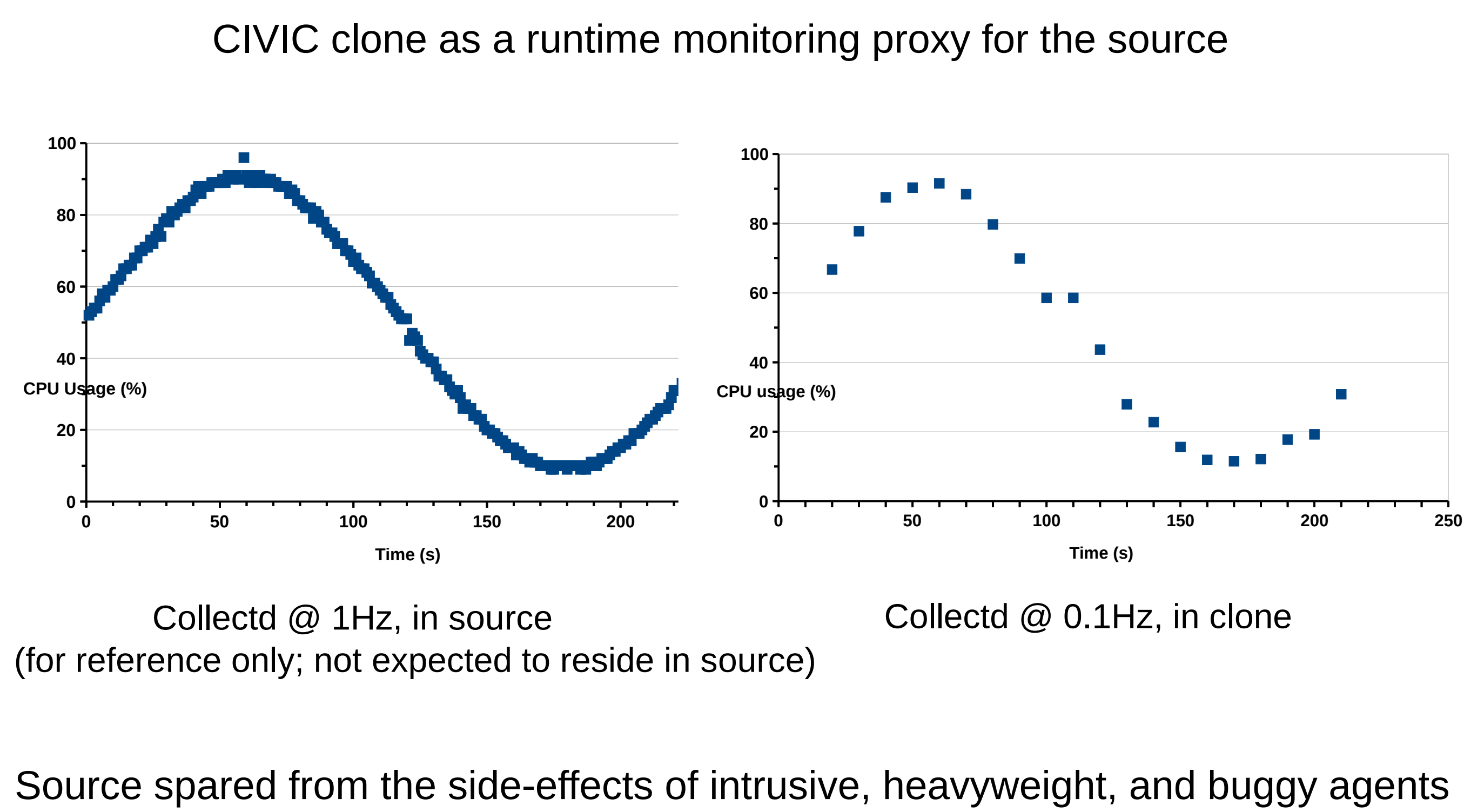
CIVIC's OPERATIONS



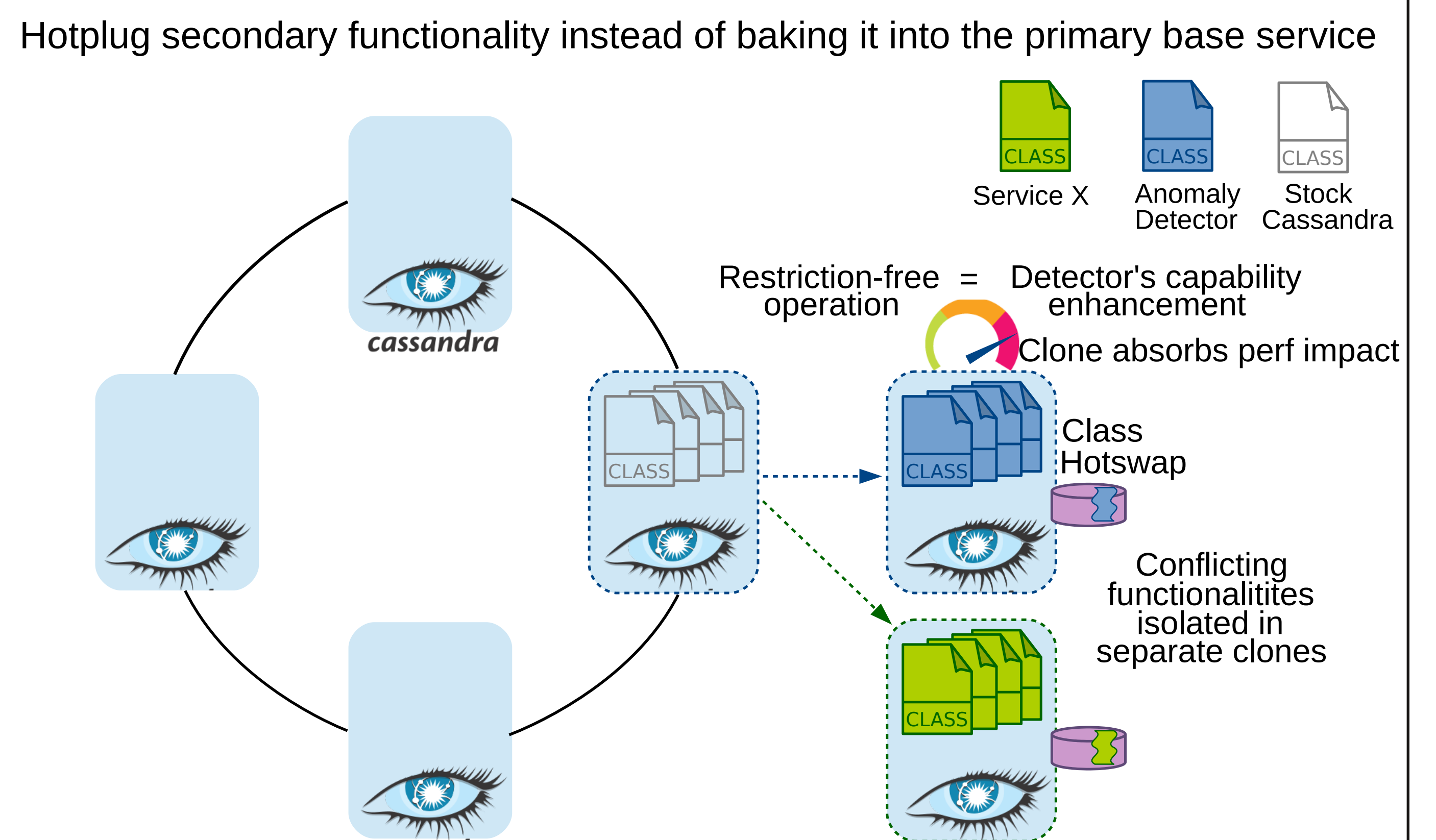
1. Disk COW: Consistency for independent clone existence
2. Live Migration: Cloning via post-copy migration
3. Memory COW: Consistency
4. Disk and NIC Hotplugging: State persistence
5. Code Injection: Into clone's kernel to run app loader script
6. App Loader Script: Setup clone, run desired app

Lightweight clones with 30MB transfer size & 80MB RSS for monitoring a 4G VM. Ready in at most 6.5s.

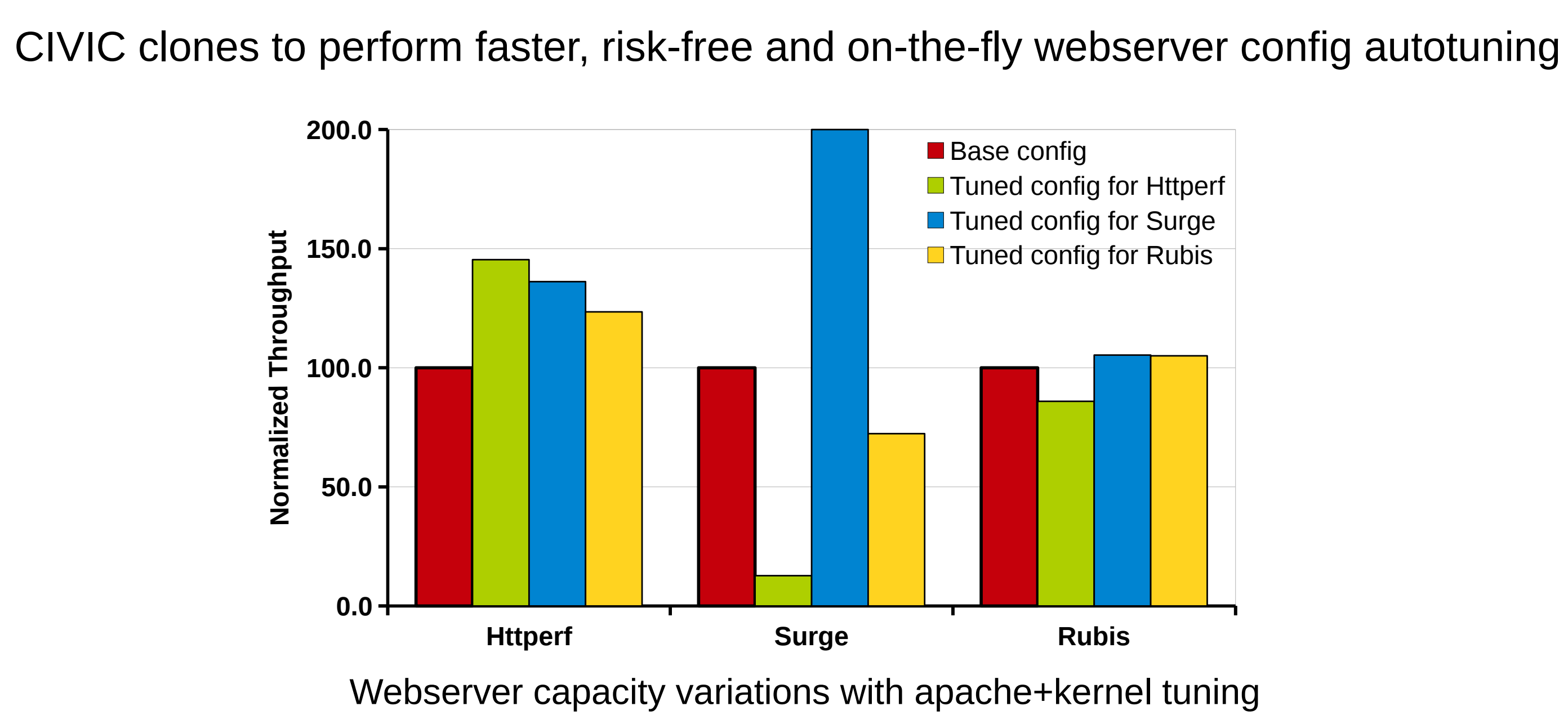
USE #1: SAFE AGENT REUSE



USE #2: FUNCTIONALITY HOTPLUGGING



USE #3: AUTOTUNING-AS-A-SERVICE



SUMMARY

- VM internal behavior reconfiguration at runtime
 - ~ existing VM-external *allocation, placement* reconfig
- Safe inspection and customization via live state cloning
 - Execution–customization decoupling
 - Intrusive / heavyweight / speculative ops
- On-the-fly functionality introduction via code injection
 - No guest modification or enforced cooperation