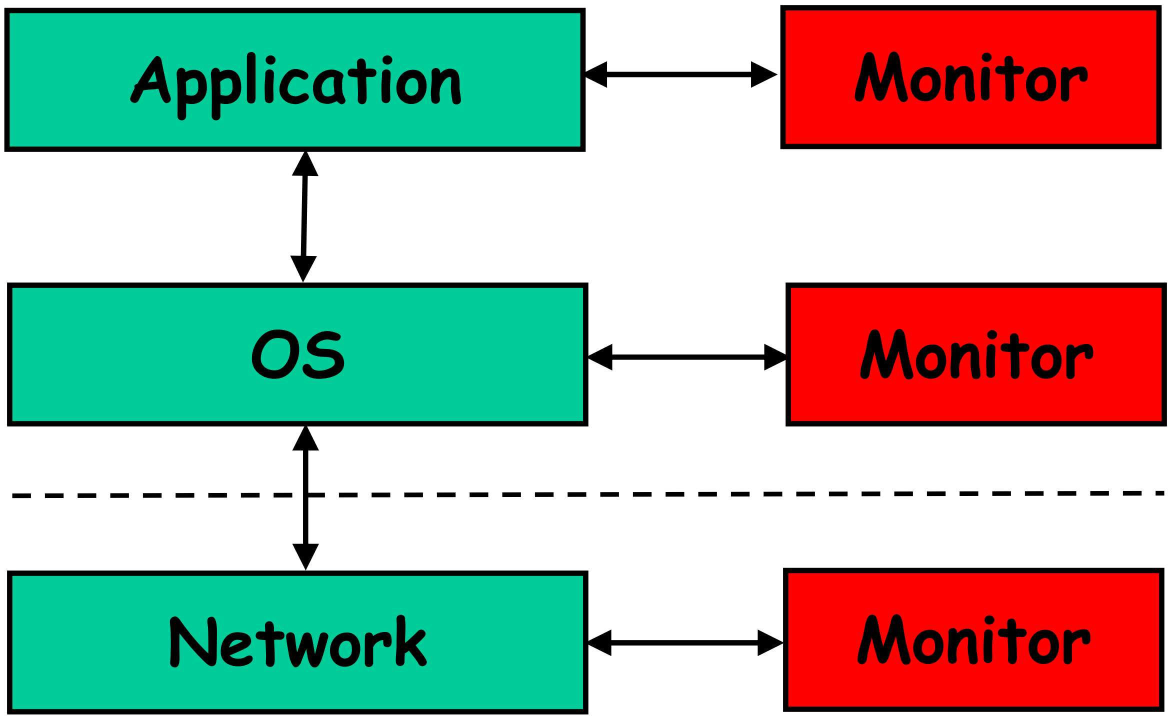
Model Predictive Control for Memory Profiling

Sean Callanan, Radu Grosu, Justin Seyster,
Scott A. Smolka, Scott D. Stoller, and Erez Zadok
Department of Computer Science
Stony Brook University
http://www.fsl.cs.sunysb.edu/hcos

The HCOS Project: High-Confidence Systems Software



Theme

Control Theory
+
Runtime Verification:
high-accuracy monitoring with controllable overhead

NSF CNS-0509230, *Runtime Monitoring & Model Checking for High-Confidence System Software*

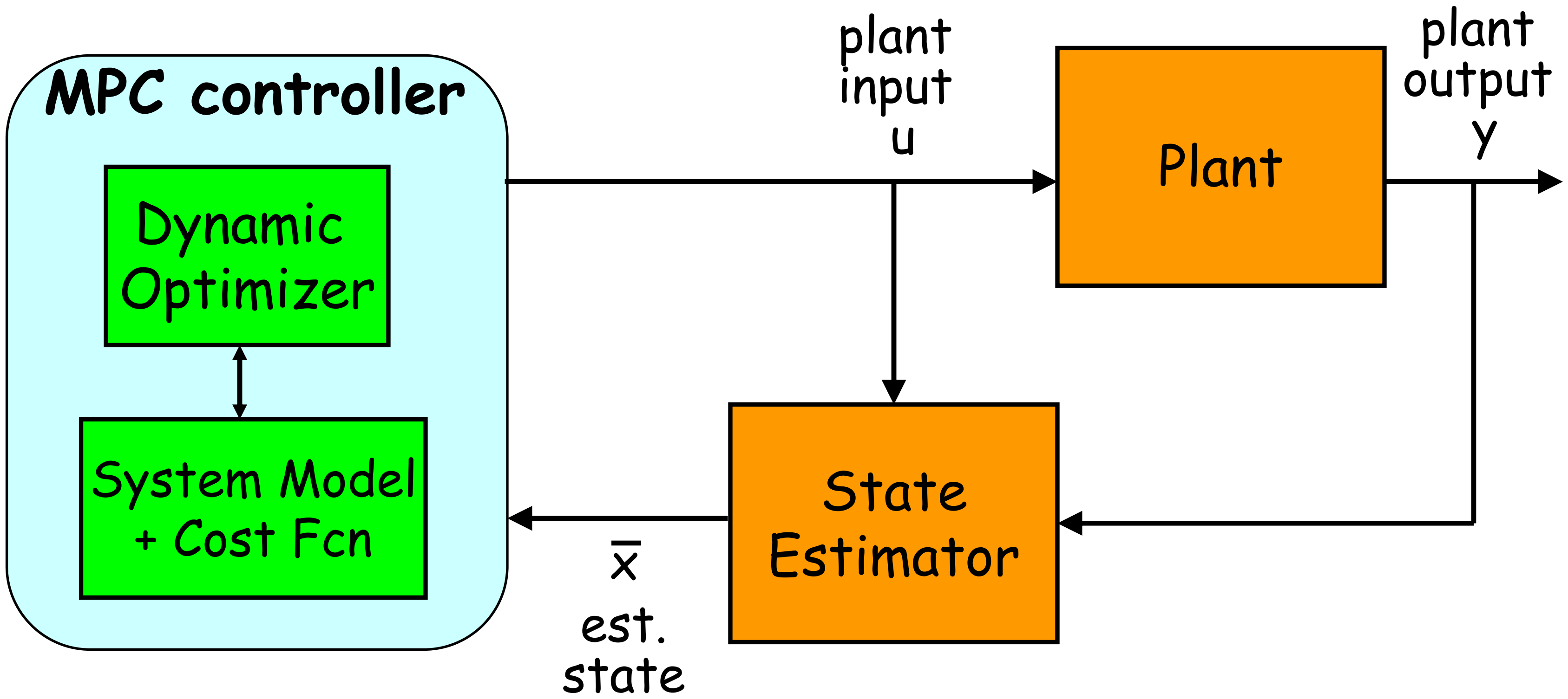
The HCOS Project: Tools and Techniques

- ***Compiler-Based Instrumentation:*** Plug-In architecture for GCC 4
- GCC Plug-Ins for:
  - Bounds checking
  - Monte Carlo software model checking
  - Kernel **`refcount`** usage checking
  - GIMPLE simulation and visualization
- **`Memcov`** memory profiler & leak detector

`Memcov`: Memory Profiling + Leak Detection

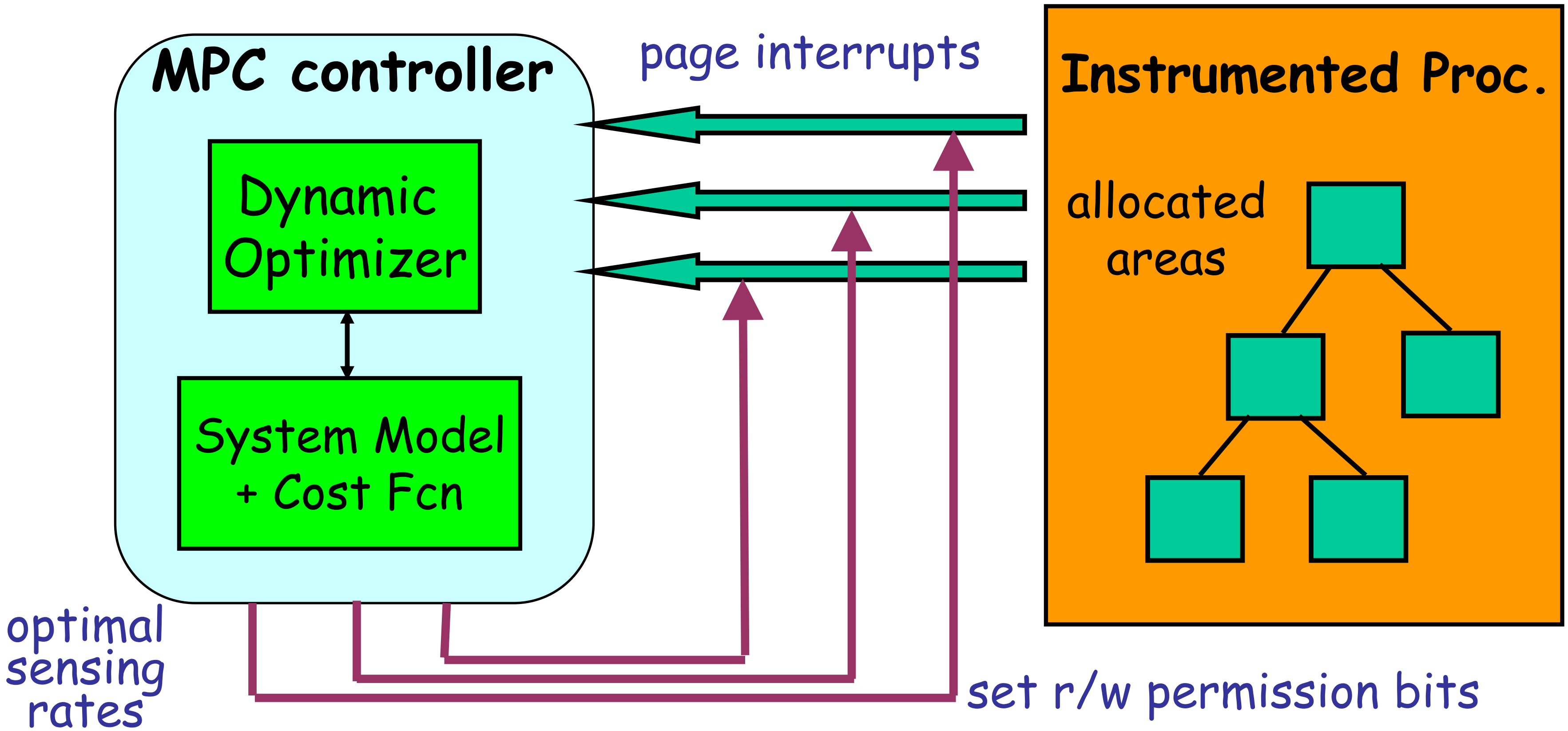
- **Goal:** detect ***memory leaks*** (or infrequently accessed areas) while regulating runtime monitoring overhead
- ***`Memcov`*** **samples memory accesses** to determine access frequencies and patterns, not just check for **`free()`**s and GC-style reachability
- **Model Predictive Control** (MPC) for *adaptive sampling*: maintain low constant overhead + high accuracy

Model Predictive Control



Goal: compute an optimal input by minimizing given **cost function** over a certain **prediction horizon** using **model of the system**

Model Predictive Control for Memory Profiling

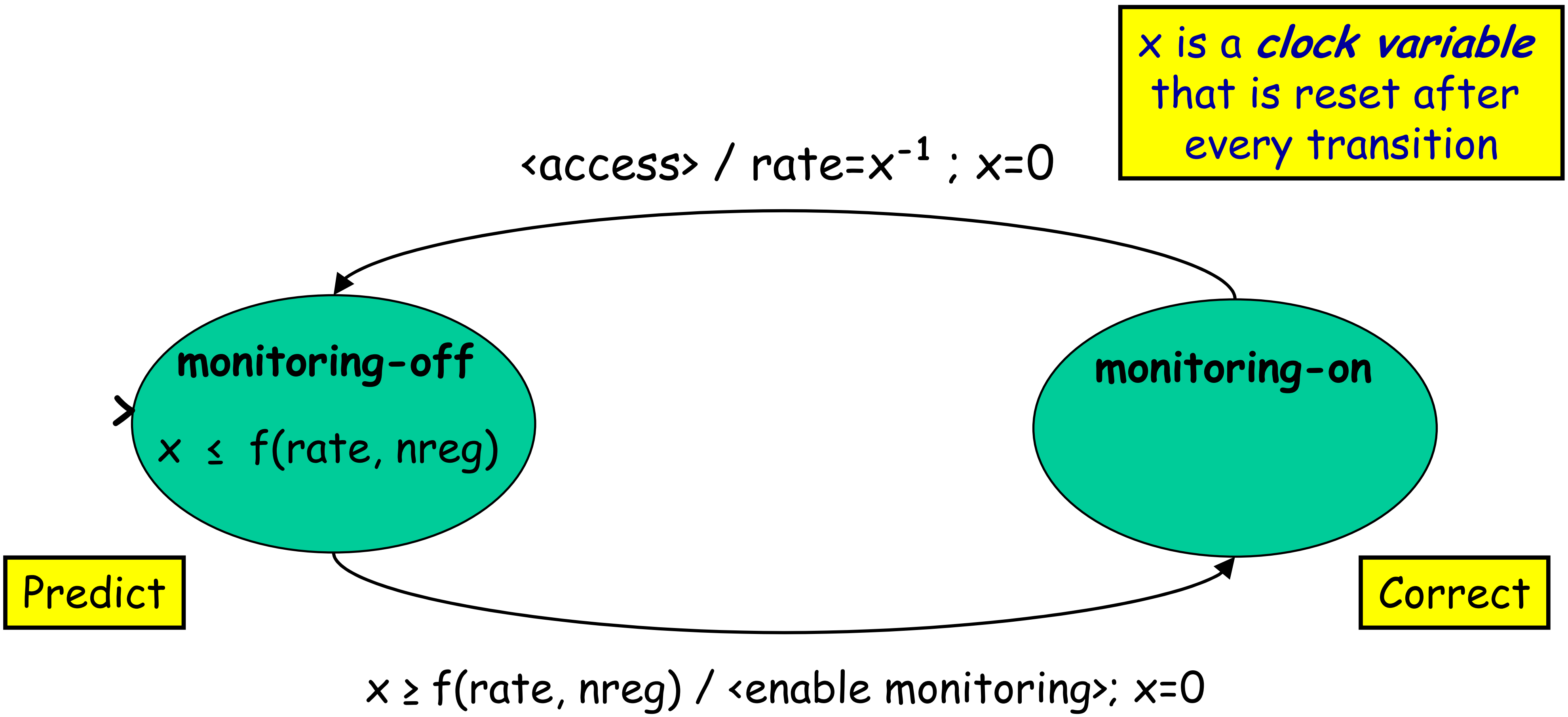


Instrumentation requires no recompilation; **dynamic loading** only

MPC for Memory Profiling (cont.)

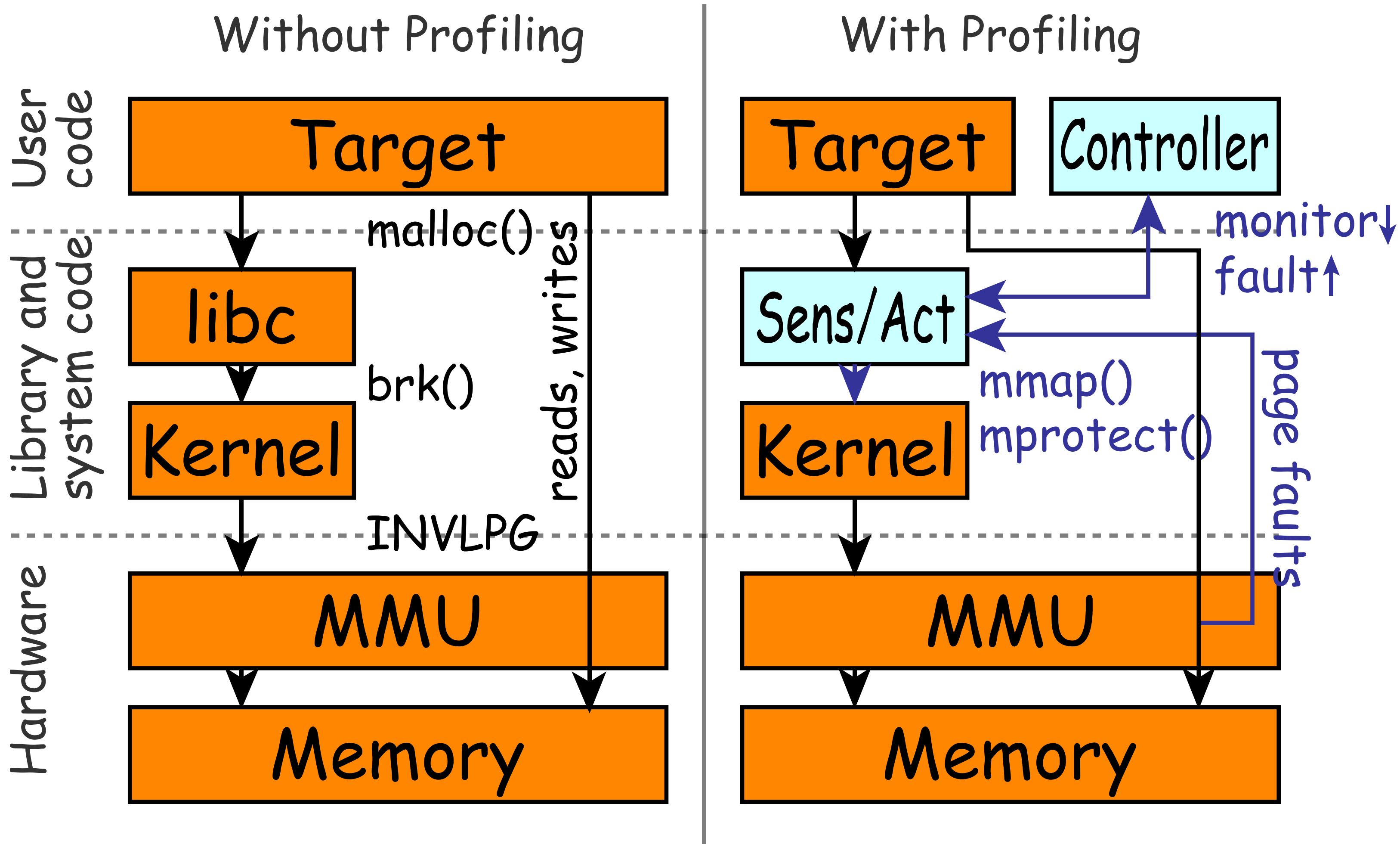
- **Sensor** = MMU + signal handler
- **Actuator** = an allocated area's memory protection bits
- Taking a **sample** = setting area's protection bits
- **Controller** adjusts sampling rate to be inversely proportional to area's access rate
- Thus, infrequently accessed areas (i.e. potential leaks) considered highly ***critical*** and monitored more frequently

Control Loop as a "Timed Automaton"

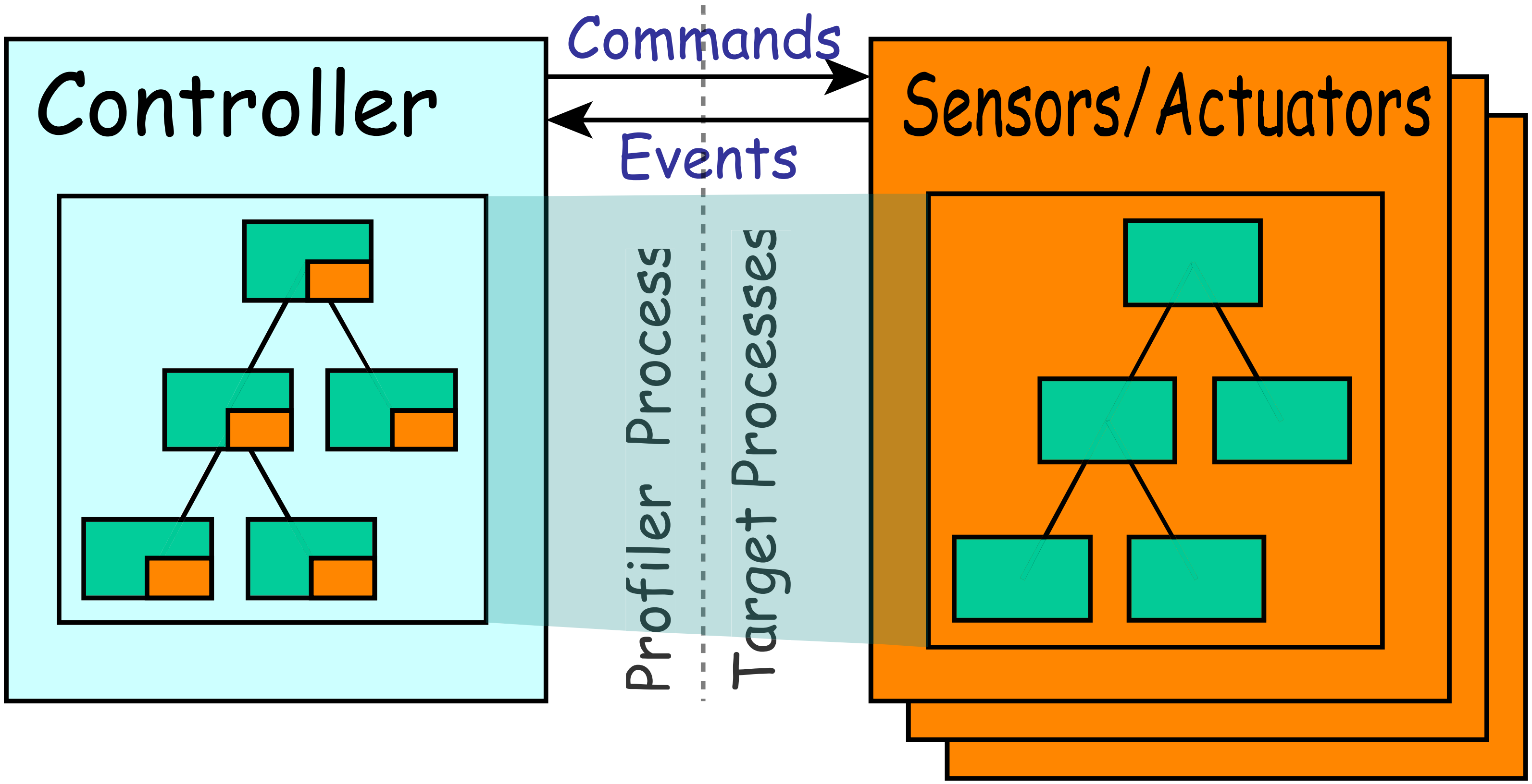


$$f\left(\text{rate},\text{nreg}\right)=\frac{p}{o_{global}/\text{nreg}}-\frac{1}{\text{rate}}-p$$

Memcov Instrumentation Architecture

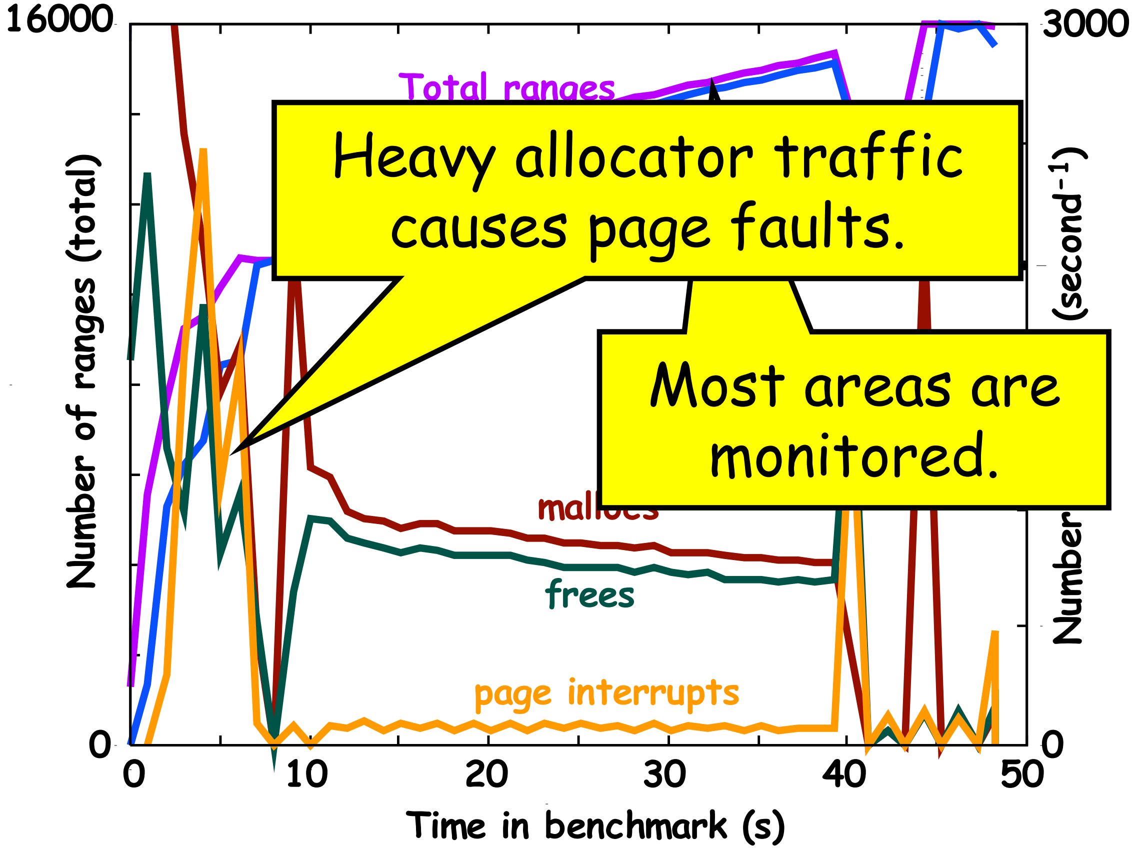


Memcov Multi-Process Architecture



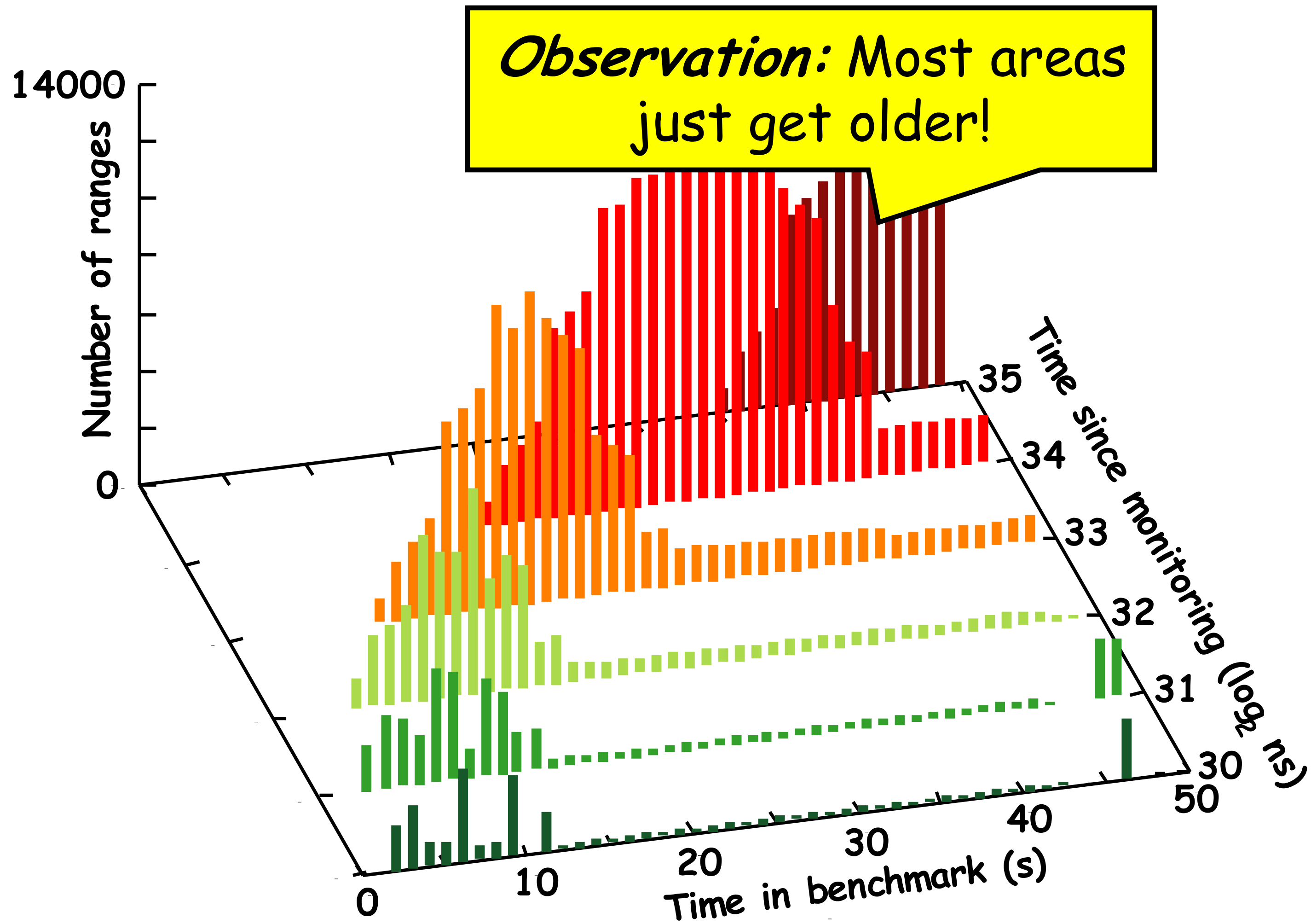
- **Commands:** Activate region
- **Events:** malloc, free, realloc, page fault

Benchmark Results: vim



- We wrote benchmark for vim that creates a file, populates it, saves it, then deletes all its data.

Memory efficiency for vim



Conclusions

- ***Model Predictive Control*** helps achieve accurate monitoring with low constant overhead
- Current and Future Research
  - Applying ***memcov*** to Firefox and Apache
  - MPC-based techniques for IDS ***packet sampling***, detecting kernel **refcount** mis-uses, bounds checking, etc.

Questions?

- ***Thank you!***
- For more information, please visit:

  http://www.fsl.cs.sunysb.edu/~hcos

- ***Bullet:*** Apply control theory to software sensors to control Heisenberg effect!