NUMA Time Warp

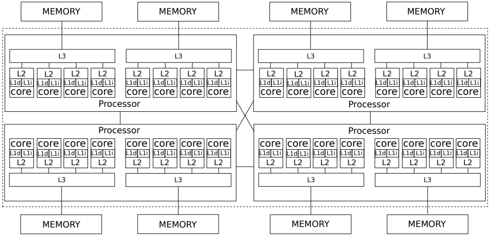
Alessandro Pellegrini
Francesco Quaglia

High Performance and Dependable
Computing Systems Group
Sapienza, University of Rome

SAPIENZA
UNIVERSITÀ DI ROMA

PADS 2015

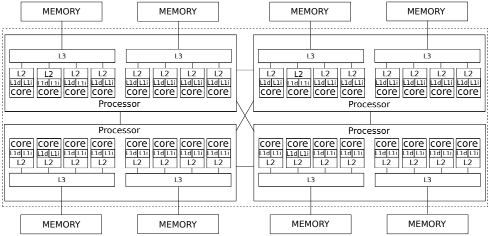
The NUMA Architecture



AMD Opteron 6128

- Memory divided into different banks
- The same core sees some banks *closer*, other *farther*
- This has an effect on access latency

The NUMA Architecture



AMD Opteron 6128

- Memory divided into different banks
- The same core sees some banks *closer*, other *farther*
- This has an effect on access latency
- Time-Warp systems are highly demanding for memory

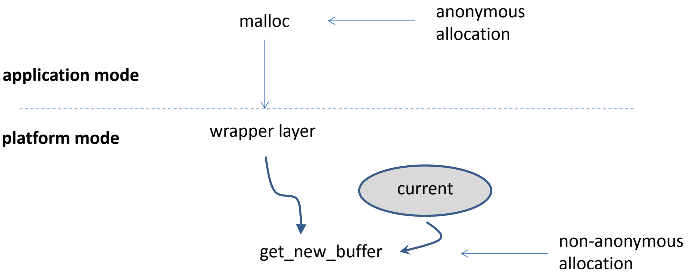
Reference Time Warp Architectural Context

- Optimistic PDES systems based on the multi-thread paradigm
  - highly suited for shared memory platforms
  - data exchange can be optimized
  - computing power can be well balanced
- Temporarily binding of simulation objects to worker threads
  - no concurrent access on recoverability data, and input/output queues of a simulation object
- Permanent binding of worker-threads to CPU cores
- Dual-mode execution scheme: *application* versus *platform* modes
- Worker threads schedule only one simulation object at a time (the *current* simulation object)

Goal: Optimizing Latency on NUMA Architectures

- NUMA-oriented memory manager
  - per-simulation-object management of memory segments made up by disjoint sets of pages
  - both *static* and *dynamic* binding of memory pages to specific NUMA nodes
- Page migration
  - to cope with worker-thread binding of simulation objects
  - based on Linux services
- Manage at the same time:
  - simulation states' memory pages
  - recoverability data
  - event buffers
- Fully transparent to the application-level code

The execution scenario



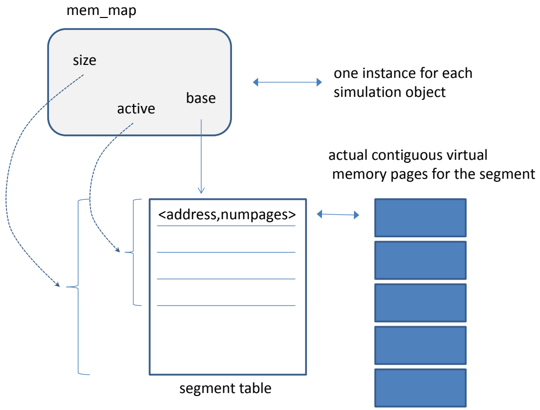
- `malloc` library calls are intercepted
- The simulation platform transforms anonymous allocations into non-anonymous allocations

Non-Anonymous Memory Allocator

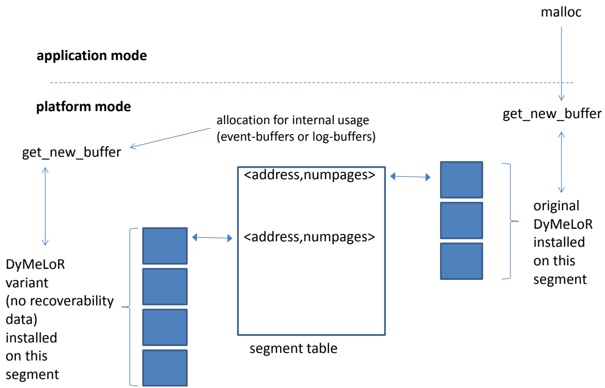
- Mid-level memory manager: DyMeLoR (any other can do the job)
  - traditional version to serve model requests
  - we have a new version with no recoverability data for platform usage
- Low-level NUMA memory manager:
  - memory is pre-reserved for the mid-level memory manager
  - pre-reserving done using `mmap`
  - for each simulation object, the following meta-data are kept:

```
void   *base;
size_t size;
int    active;
```

Managing the Memory Map



Allocations from Application- and Platform-level



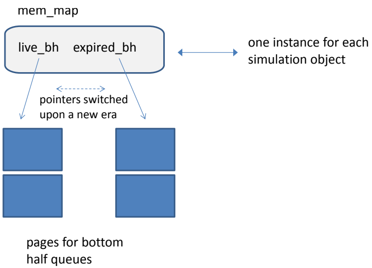
By using `set_mempolicy` we force the Linux kernel to materialize the pages on the NUMA node closest to the worker thread

Data Exchange Management

- Not all data accesses are "private": what about event exchange?

Data Exchange Management

- Not all data accesses are “private”: what about event exchange?
- NUMA-oriented implementation of the bottom half-based message-exchange scheme, using additional meta-data:

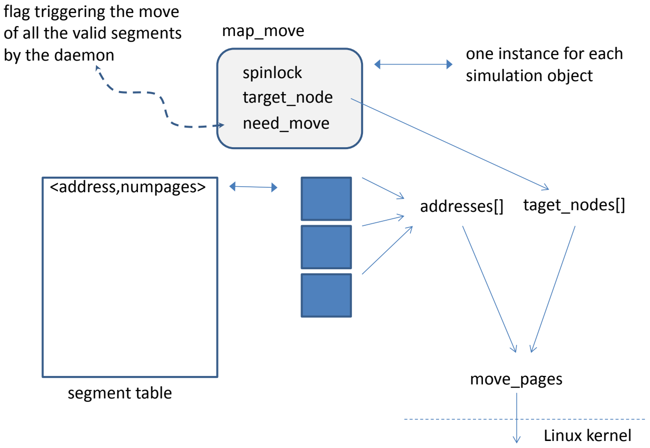


Data Exchange Management

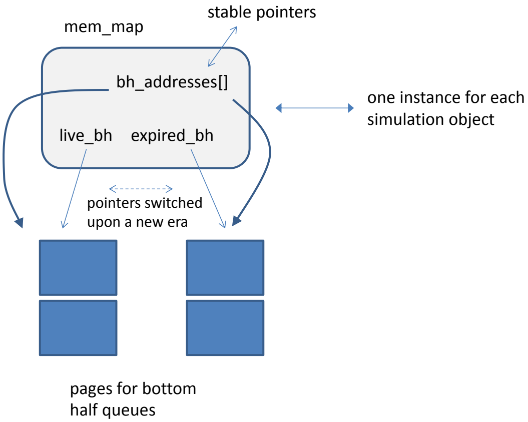
- Not all data accesses are “private”: what about event exchange?
- The worker thread managing the destination simulation object accesses it more frequently ⇒ keep pages close to it
- Yet the pages are *not* guaranteed to be located on the node closest to the CPU running this worker thread!
    - remember `set_mempolicy`?

High Performance Dependable Computing Systems

Page Migration: the pagemigd daemon



Migrating Bottom Halves



Experimental Evaluation: Test-bed Platform

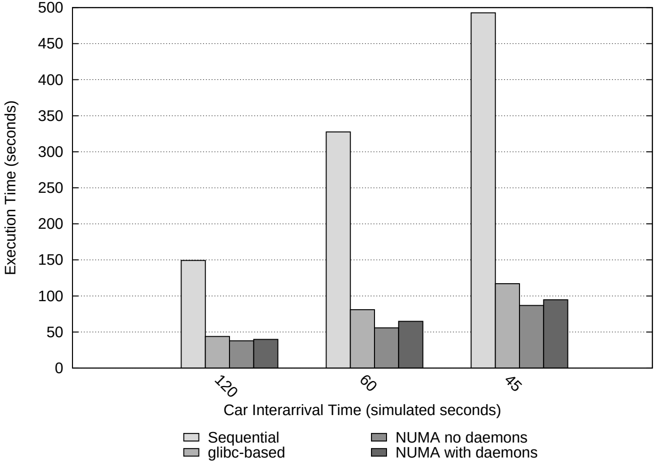
- Hardware configuration:
  - HP ProLiant server equipped with 64GB of RAM
  - 4 8-cores CPU (32 cores total)
  - 8 NUMA nodes, close to 4 cores, distant to all the others

- Software configuration:
  - ROOT-Sim Optimistic Simulation Kernel, using 32 symmetric worker threads
  - Debian 6
  - 2.6.32-5-amd64 Linux kernel

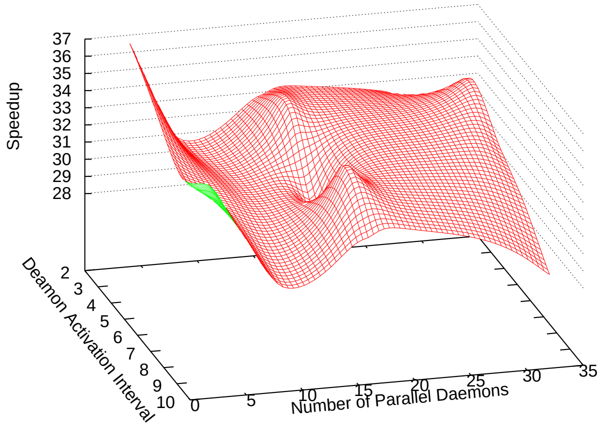
Benchmark Application: *Traffic*

- Balanced Scenario:
  - 137 simulation objects
  - Accident probability close to zero
  - Even workload (no rebinding)
  - When active, `pagemigd` daemons are very aggressive
- Unbalanced Scenario:
  - 1024 simulation objects
  - Number of active daemons in $[4, 32]$
  - Activation interval in $[2, 10]$

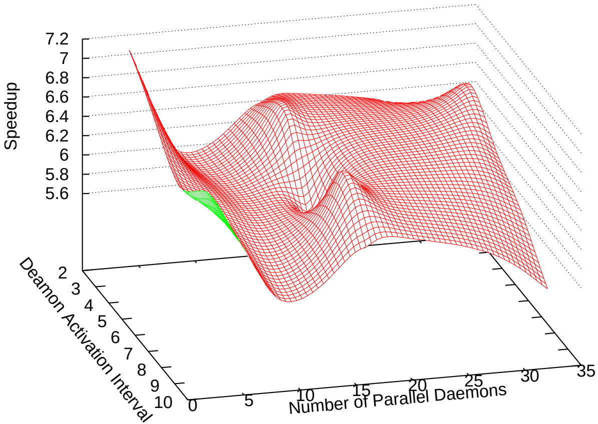
Balanced Configuration: Execution Time



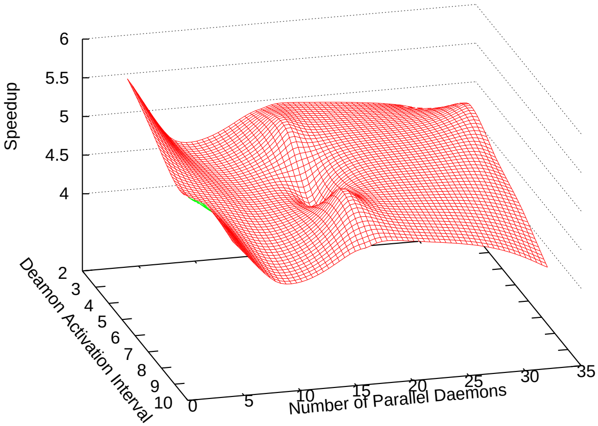
Unbalanced Configuration: vs sequential



Unbalanced Configuration: vs `glibc`



Unbalanced Configuration: with or without daemons



Thanks for your attention

Questions?

pellegrini@dis.uniroma1.it
http://www.dis.uniroma1.it/∼pellegrini
http://www.github.com/HPDCS/ROOT-Sim