FORUM STIC Paris-Saclay 2014

A. Blanchard – N. Kosmatov – M. Lemerre – F. Loulergue

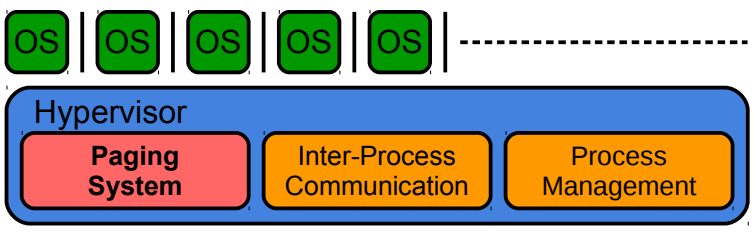
N° 1.8

Paging System of the Anaxagoros Hypervisor Formally Verified with Frama-C: A Case Study

Cloud Hypervisors

Hypervisors are used in clouds to virtualize OS, allowing to **share** resources between users.

They are composed of three main modules that have to ensure the correct isolation of client tasks. Their **formal verification** can increase our confidence in security and reliability of the cloud.

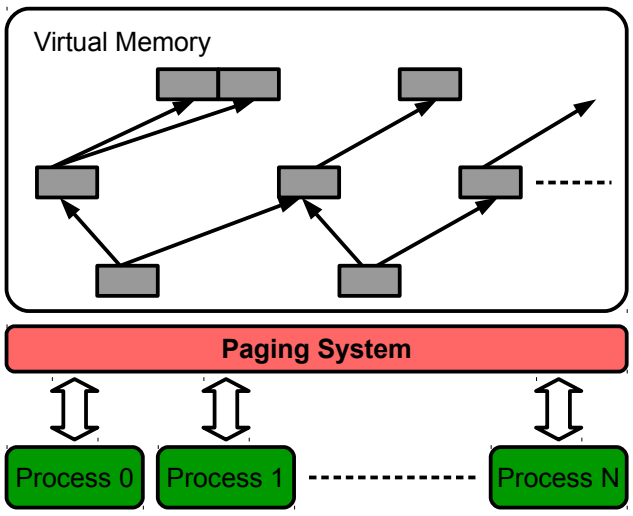


The Paging System of Anaxagoros

The **paging system** manages the **virtual memory**, ensuring that processes only perform legal operations on it. We analyse the algorithm responsible for modifying the memory tree.

It is part of Anaxagoros, an hypervisor developped at CEA LIST, which is designed to accept **non-blocking simultaneous** kernel calls from many processes.

We prove the system invariant by creating a **simulation** of **parallel executions** of this algorithm.



Simplified Source Code

```
#define NOF 2048//nb of frames
#define MAX 256 //max number of mappings
uint mappings[NOF];

int set_entry(int fn, int idx, int new){
  int c_n = mappings[new];
  if(c_n >= MAX) return 1;
  if(!CAS(&mappings[new], c_n, c_n+1))
    return 1;

  page_t p = get_frame(fn);
  uint old = exchange(&p[idx], new);

  if(!old) return 0;
  fetch_and_sub(&mappings[old], 1);

  return 0;
}
```

Simulation of Concurrent Executions

Atomic instructions are modeled by functions parameterized by the id of the thread we make advance:

```
void read_map_new(uint th){
  c_n[th] = mappings[new[th]];
  pct[th] = 2;
}
```

```
void FAS_map_old(uint th){
  mappings[old[th]]--;
  //@ghost ref[th] = 0;
  pct[th] = 0;
}
```

Execution context is modeled by arrays associating to each thread id the state of its local variables and its program counter:

```
#define THD 512 //max nb thread

uint fn [THD];
uint idx[THD];
uint new[THD];
uint c_n[THD];
uint old[THD];

uint pct[THD];
```

Finally, we model concurrent execution by an infinite loop that executes a step of a random thread, and start again:

```
while(true){
  int th = choose_a_thread();

  switch(pct[th]){
  case 0 : gen_args(th); break;
  case 1 : read_map_new(th); break;
  case 2 : test_map_new(th); break;
  case 3 : CAS_map_new(th); break;
  case 4 : EXCH_entry(th); break;
  case 5 : test_map_old(th); break;
  case 6 : FAS_map_old(th); break;
  }
}
```

Specification and Proof of the Simulation

We equip the simulation loop and every atomic instruction function with the **global invariant** we want to verify by adding ANSI C Specification Language (ACSL) annotations in the source code.

We use the **Frama-C** framework with its (**WP**) Weakest Precondition plugin to generate proof obligations that are mostly discharged by SMT solvers.

As some of them need complex induction proofs, we state them in auxiliary lemmas that we prove them with the **Coq** proof assitant.

frama C Software Analyzers

Summary

We prove that the algorithm is valid in a concurrent context. The simulation technique has a lack of scalability and modularity, but:

- ✔ allows analysis of parallel code with a **tool that does not natively support it**,
- ✔ is inexpensive compared to the effort of specification,
- ✔ It is mostly **automatic**,
- ✔ It does the job !

Reference

Blanchard, A., Kosmatov, N., Lemerre, M., Loulergue, F. : Paging System of Anaxagoros Hypervisor Formally Verified with Frama-C: A Case Study. Submitted.

Contacts :
- Allan BLANCHARD, CEA LIST/LIFO, allan.blanchard@cea.fr
- Nikolai KOSMATOV, CEA LIST, nikolai.kosmatov@cea.fr
- Matthieu LEMERRE, CEA LIST, matthieu.lemerre@cea.fr
- Frédéric LOULERGUE, LIFO, frederic.loulergue@univ-orleans.fr