Lightweight task migration in multi-tiled architectures – Communication consistency

Frédéric Rousseau

TIMA lab – University of Grenoble Alpes

A joint work with Ashraf El-Antably & Nicolas Fournel

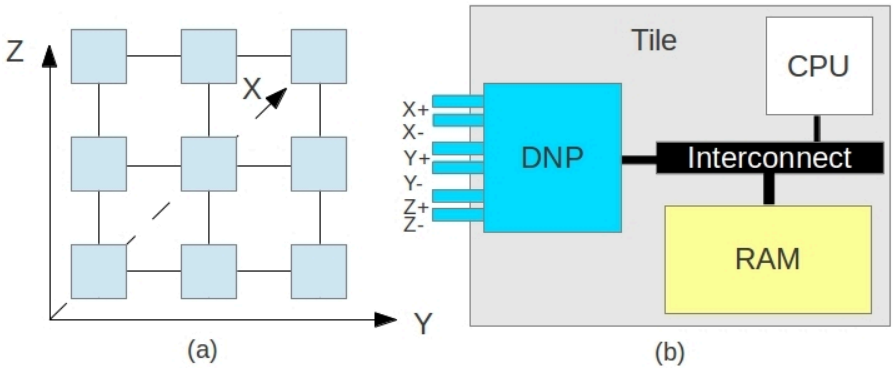
Agenda

1. Definition, motivation, and challenges
2. Communication inconsistency and task migration methodology
3. Implementation, simulation and results
4. Conclusion and future work

Definitions

* A **multi-tiled architecture** is a highly parallel processing platform, designed with several **tiles** connected with a NoC
* A **tile** contains at least one processing unit and peripherals (I/O, NoC routers, private memory, …)

  ➔NO SHARED MEMORY BETWEEN TILES



* In multi-tiled architecture, **task migration** is the transfer of task (process) context from source tile to destination tile

Motivations

Task migration is a suitable solution which circumvents several challenges of MPSoC:

1. **Reliability:** load balancing reduces peak temperature, avoids excessive use of certain cores
2. **System adaptivity:** with more complex applications, not all scenarios can be determined statically, task migration eases dynamic remapping
3. **Power consumption:** moving away tasks to enter power down mode

Challenges

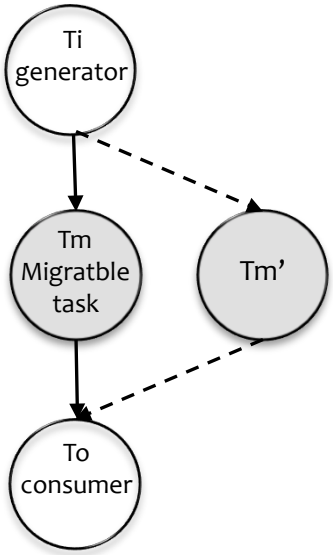
* **Task migration is a well-known process in Symmetric Multi-Processors (SMP) architectures**
  * Thanks to a shared memory

* **More challenges in multi-tiled architecture**
  * Transfer of the *task context: Where to stop the code ?*
    * Stack, heap, registers
  * What about the *task code: replication vs recreation ?*
    * Transfer to or duplication in the source tile ?
  * How to ensure *communication consistency* ?
    * Resuming communication after migration ?
    * How to manage unprocessed tokens ?

Communication inconsistency

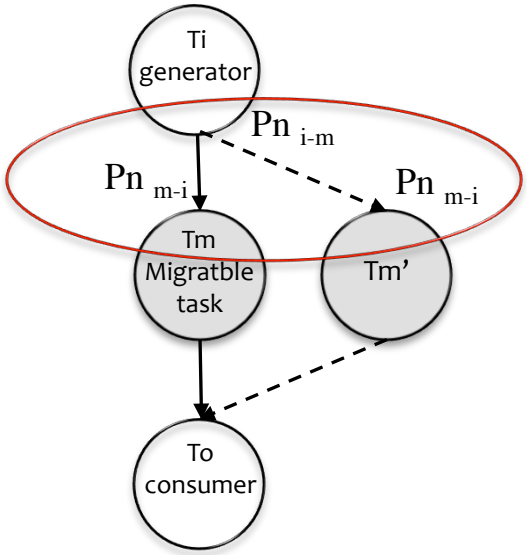
* Hypothesis
  * Inter-task communication is undergone through FIFO buffers
  * Reading and writing are blocking

* Reasons that may cause inconsistency
  * Change of the location where a task is running
    ➔ Modification of the channel after migration
    * to keep sending and receiving data
    ➔ Forwarding unprocessed data left in FIFO
    * Re-sent of data to the right corresponding FIFO



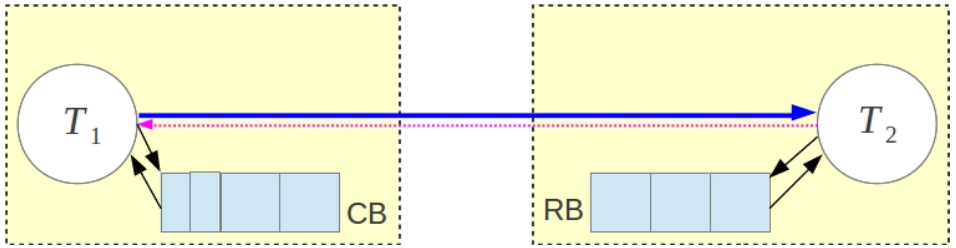
Communication inconsistency solution

* **Configurable channels**
  * Each channel is configurable
    * Second branch connected to the replica
    * Only one branch is active at a time
  * Same port number (what is seen by the application designer)
    * No change in the task code



Communication inconsistency solution

* How to manage unconsumed data ?
  * Our solution: **A copy buffer-based protocol**
    * Copy buffer to sender and receiver
    * When a data has been consumed by the consumer, an ACK is sent back to the sender, and the data is removed from the sender buffer



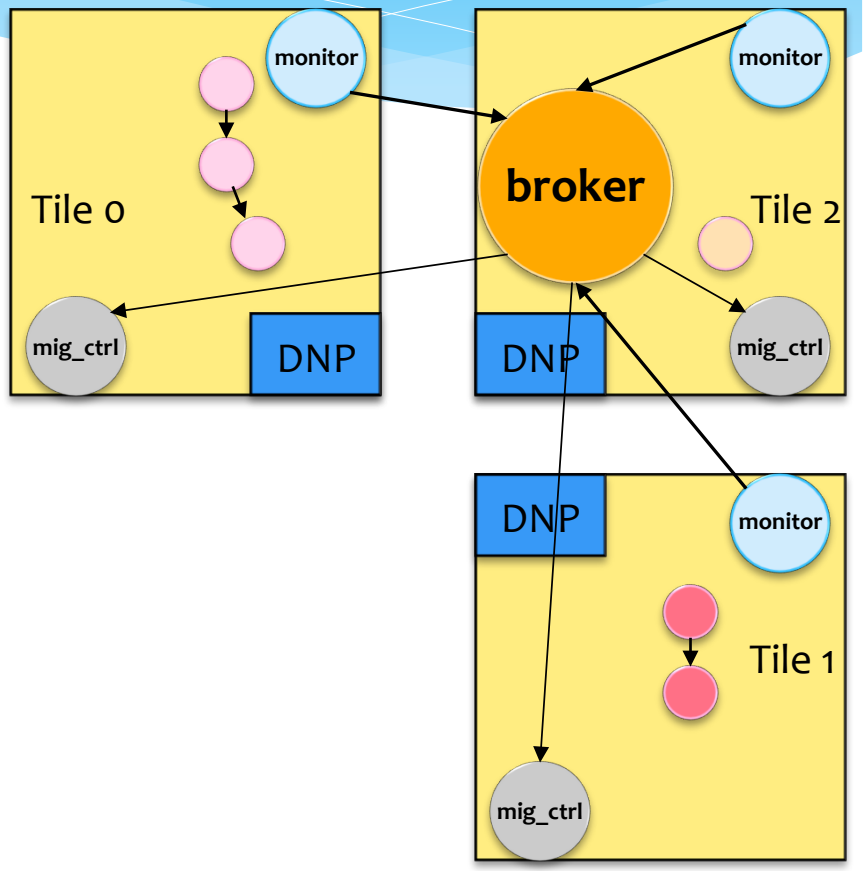
Migration decision

- ***Semi-distributed* topology is used**
  - A system has a number of ***clusters***
  - A cluster has a number of ***tiles***

- **Detection of troubles: DNP & monitor**
  - DNP for link integrity, temperature status
  - *One* monitor process per tile checking DNP

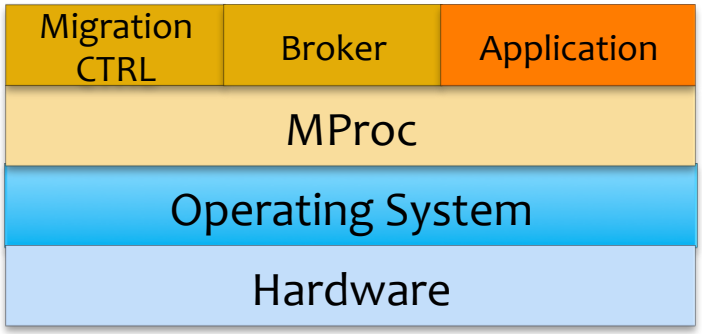
- **Decision: Broker process**
  - Responsible for migration decision
  - 1 broker per cluster

- **Implementation: mig_ctrl process**
  - Responsible for implementing migration process
  - *One* mig_ctrl per tile



Implementation

* The overall algorithm is quite complex
  * Several processes are needed to manage migration
  * Several tables to keep a view of
    * Location of each task
    * List of adjacent tasks
  * Creation of new channels to manage migration control
* Definition of a specific layer
  * APIs for creating, starting, pausing, resuming and migrating processes along with all APIs responsible for communication



Reaching migration point

* Reached only if the iteration is able to be completed
  * IO and computations are finished
* All neighbours are paused in reverse token dependency
  * Avoid deadlocks
* One reason not to reach the migration point: a channel is full
  * The migration management detects such a blockage and increases the FIFO size.

```
/* Allocate state variables */
  p->init(p);
  /* until it is canceled or migrated */
  while (!CANCELED()) {
     /* Normal execution */
    if (p->cmd == RUN)
      p->fire(p);
    else /* migration point */
      if (p->cmd == MIG_REQUEST){
        Init_migration();
      }
```

Simulation and results

* A multi-tiled simulation platform is built (SystemC TLM – ARM based QEMU emulator) – Talk of Prof. Frédéric Pétrot on Friday

* In terms of memory footprint
  * An increase of less than 4% of the memory footprint
    * Mainly from the .text section (MigCtrl tasks and replica)

* In terms of performances
  * Overall execution time static/dynamic migration request
    * 14,8% of overhead

Conclusion and perspectives

* Conclusion
  * Lightweight task migration methodology
    * No modification in the OS (no MMU, no dynamic loading) or drivers
  * Light overhead
  * Communication inconsistency solved

* Perspectives
  * Automatic generation
  * Port on a real hardware multi-tile platform
    * With our partner INFN (Roma): QUONG architecture

* This work has been funded by the EC (FP7 EURETILE)