Reaching into space
TOGETHER

MULTI-SCHEDULER AND MULTI-THREAD: POSSIBLE WITH SMP2 ?

Charles Lumet - Nadie Rousse - Pierre Verhoyen

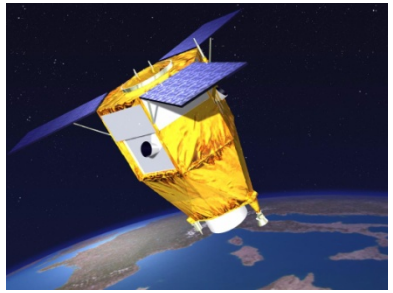
cnes

SPACEBEL

Challenges

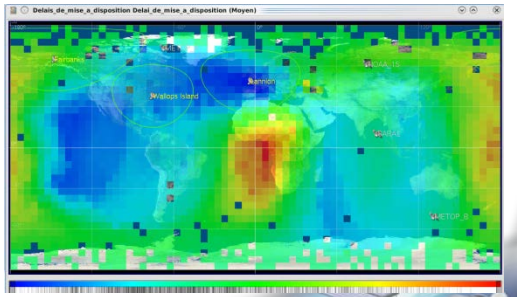
- **Satellites simulators**: simultaneous emulation of several processors and sophisticated modeling

  Example: Pléiades satellites have 5 processors to be dealt with in parallel during a simulation



- **Study simulators**: several hundreds of thousands of events per second

  Example: Argos simulator (worldwide beacon-based tracking and environmental monitoring system)

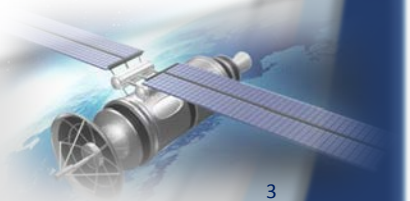


Overview

1. **BASILES**: a short overview of the CNES simulation plaftorm

2. **Multi-scheduler (MS)**: using multiple schedulers of different types in a simulation

3. **Multi-thread (MT)**: managing several threads within a simulation



BASILES – Model specificities

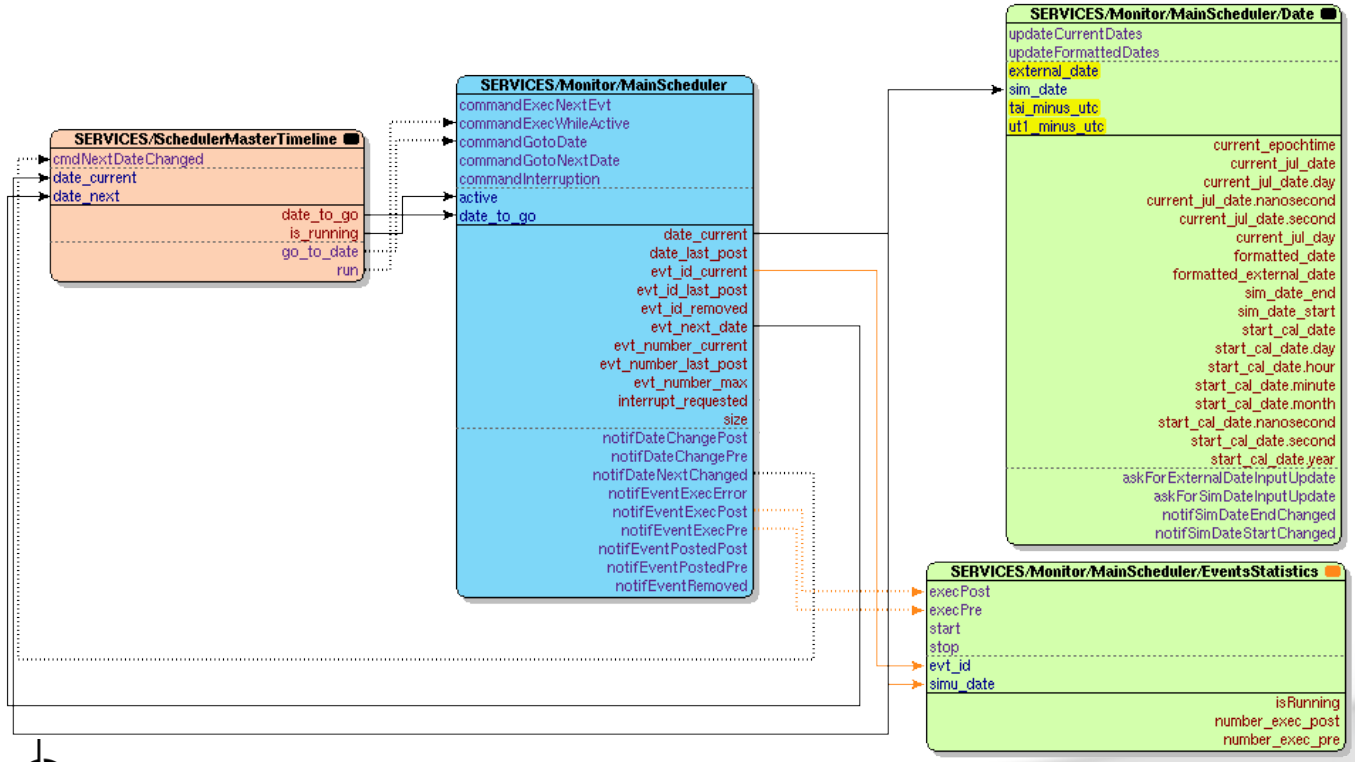
Main differences between BASILES and SMP2 models:

- **Data propagation**:
  In BASILES, no specific data propagation is required.
  ➔ High performance, easy introspection

- **Event management**:
  In BASILES, event publication is a mandatory step and allows the storage of contextual data (*event description*)
  ➔Frequency, drift ; priority, associated Scheduler

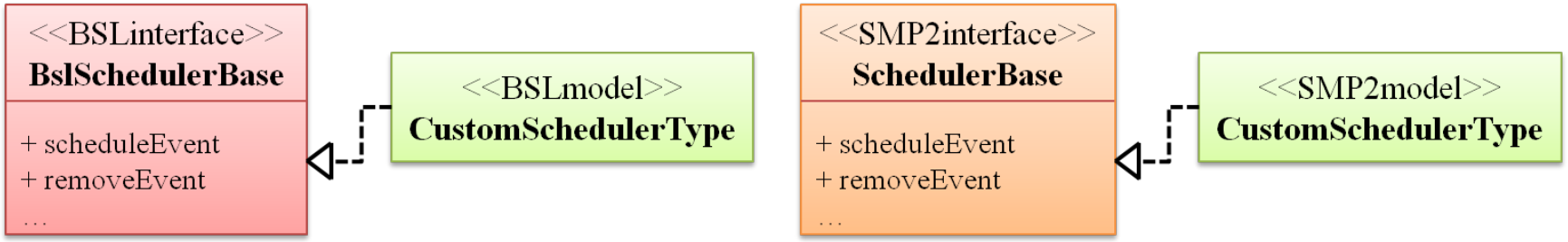
BASILES – Simulator kernel

- Configurated through a **simulation script**
- Built **with BASILES models**



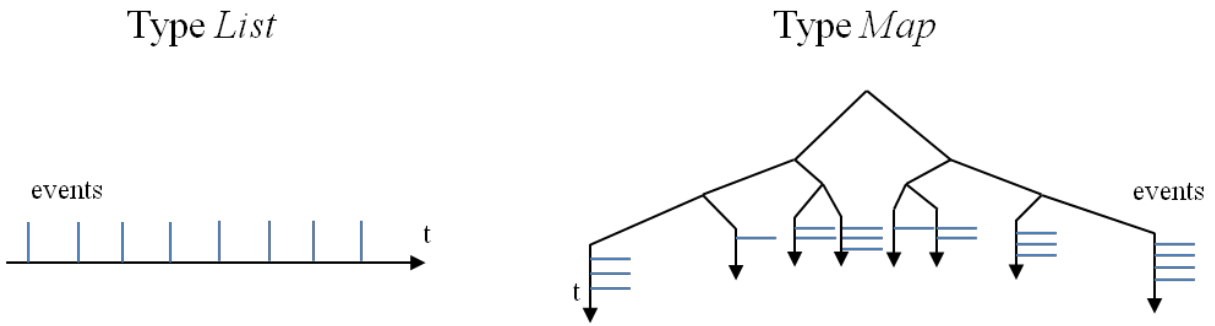
MS – Scheduler types

- **Several scheduler types** can be used in BASILES
- Simple **interface-based design**
  - With BASILES models
  - With SMP2 models



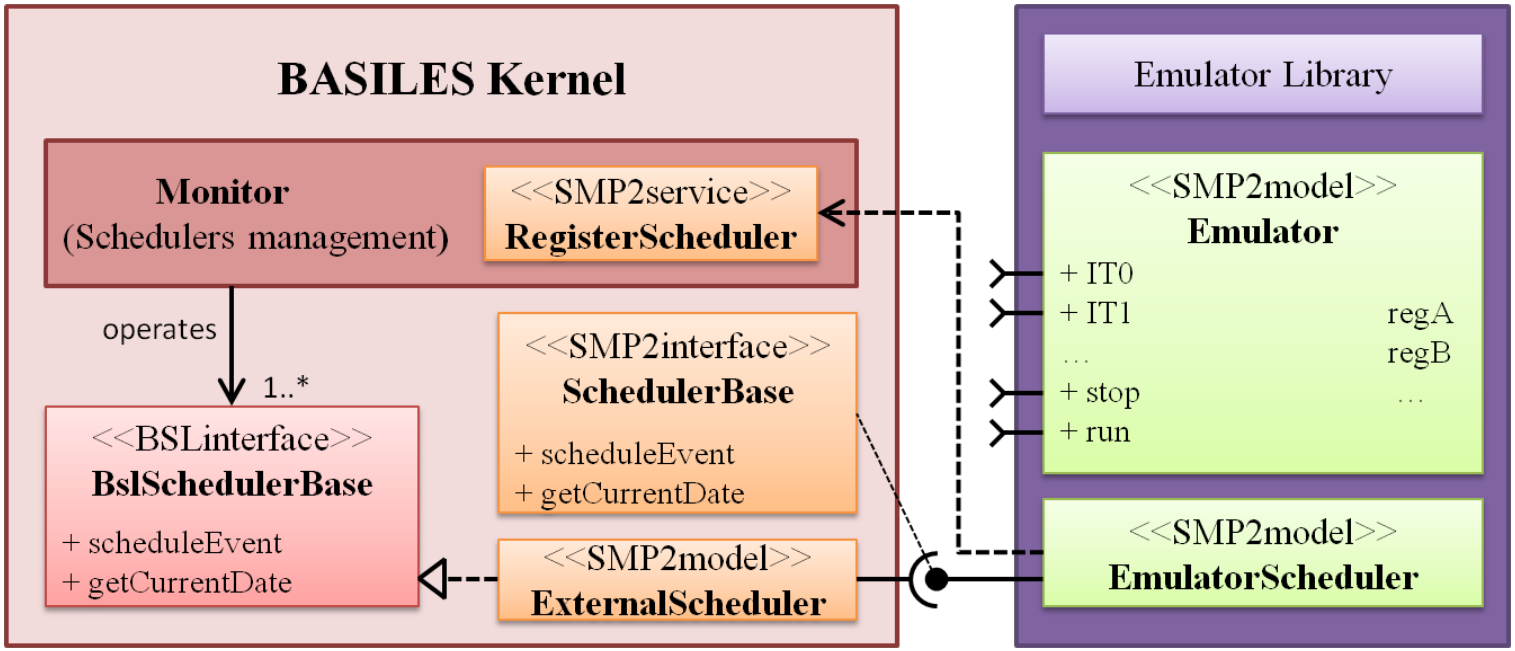
MS – Native scheduler types

- Two native scheduler types in BASILES : *list* and *map*
  → **different event internal storage mechanisms**



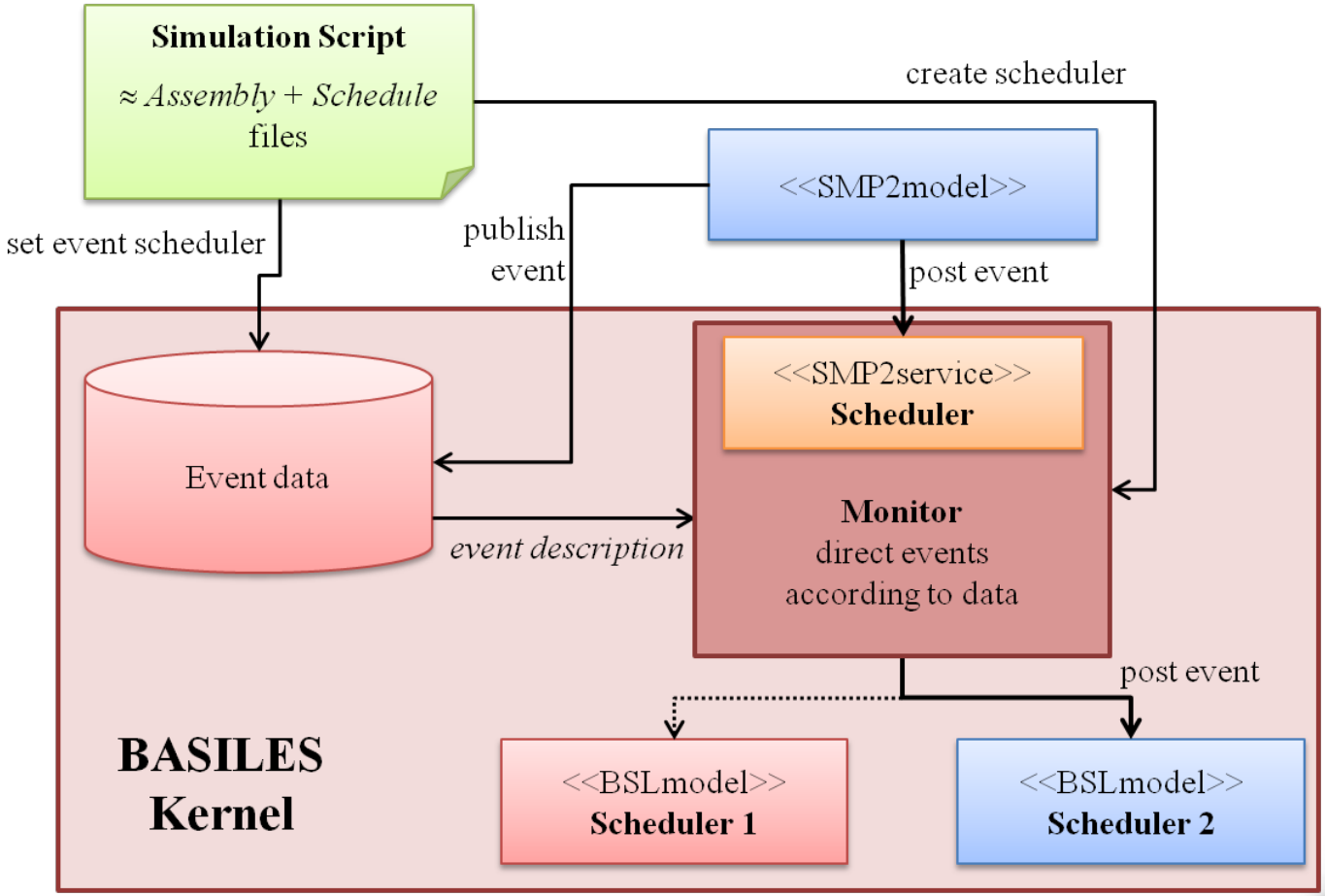
- **Performance improvement**
  Example: ARGOS simulator
  50 000 events in the scheduler, map *is 60 times faster*

MS – Integrating flight software



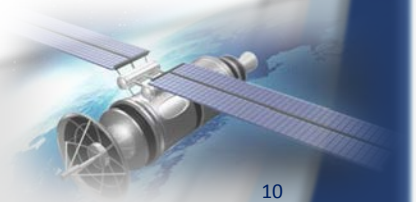
- Scheduler registration
- Scheduler use through interface link

MS – Multi-schedulers in action

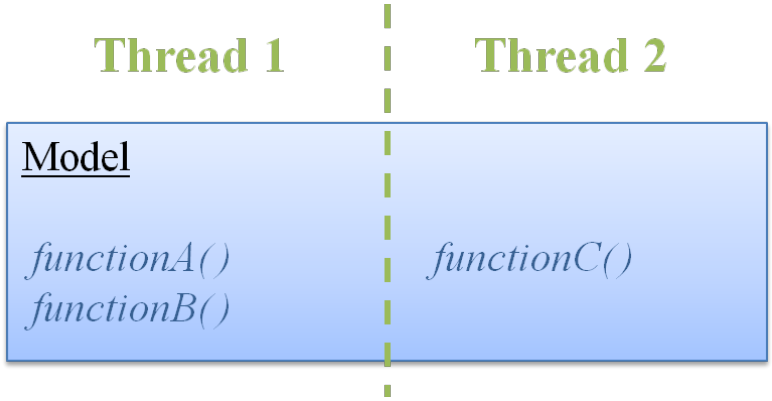


MT – Introduction

- Why?
  - **Performance** requirements (real time execution, …)
  - **External devices** integration
- How?
  - Different **levels of parallelisation**
    - *Function-level* multithreading
    - *Event-level* multithreading
    - *Scheduler-level* multithreading



MT – Function-level mutithreading

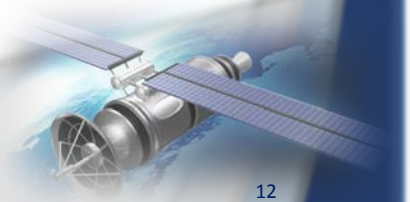


- **Inside a model**, by the model designer
- Usually used to process **heavy computations**
- Possible use of dedicated programming languages (Cilk, …)

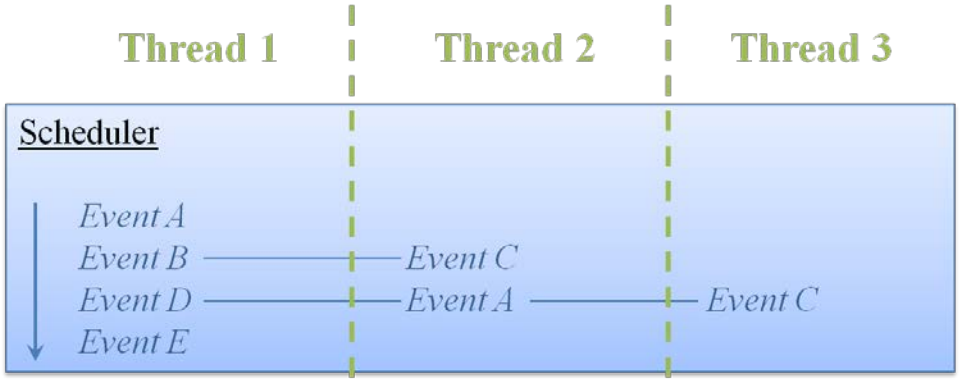
MT – Function-level mutithreading

- **This is low-level multithreading**
  - \+ Does not need any information outside of the model
  - \+ Stay invisible from outside the model
  - \- Does not rely on or take advantage of discrete event simulation mechanisms

➔ We do not study this approach



MT – Event-level mutithreading



- Principle: events with **same date and priority** are executed in parallel.
- A **new scheduler type** is designed, which manages threads.
- Highly efficient for events with a **long execution time**

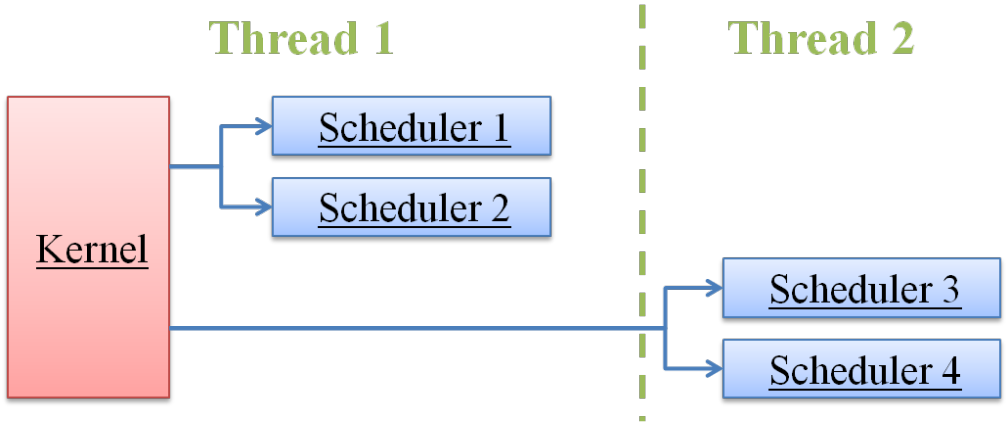
MT – Event-level mutithreading

- **This is intermediate-level multithreading**

  + No prior knowledge needed
  + Does not depend on model implementation

  - Only takes advantage on basic event data
  - High-level separability is hidden to this approach

MT – Scheduler-level mutithreading



- Direct extension of the multi-scheduler approach
- Principle: **each scheduler can be affected to a thread** (configuration)
- All events in the scheduler will be executed in the scheduler thread

MT – Scheduler-level mutithreading

- **This is high-level multithreading**
  - + Models are (multi-)thread-agnostic
  - + Allows for a fine tuning of the parallelisation, using high-level information (model separability, …)

- Comes with high-level challenges
  - **Causality** and thread **synchronization**
  - **Temporal coherence** of variables

Conclusion

- Multi-scheduler provides:
  - Performance improvement with **scheduler types customization**
  - A direct and flexible way to **integrate external software** (emulators)
  - All through **straightforward kernel configuration**

- Multi-thread provides:
  - Performance improvement through **parallelisation**
  - **Several levels of parallelisation** depending on the goal
  - **Use of multi-scheduler capabilities** to finely configure the parallelisation

Thank you!

Any questions?