Institute of Computer Science
Department of Mathematics and Computer Science

Freie Universität Berlin

Algorithms and Programming IV

Design and Implementation of Parallel Applications

Summer Term 2023 | 12.06.2023

Barry Linnert

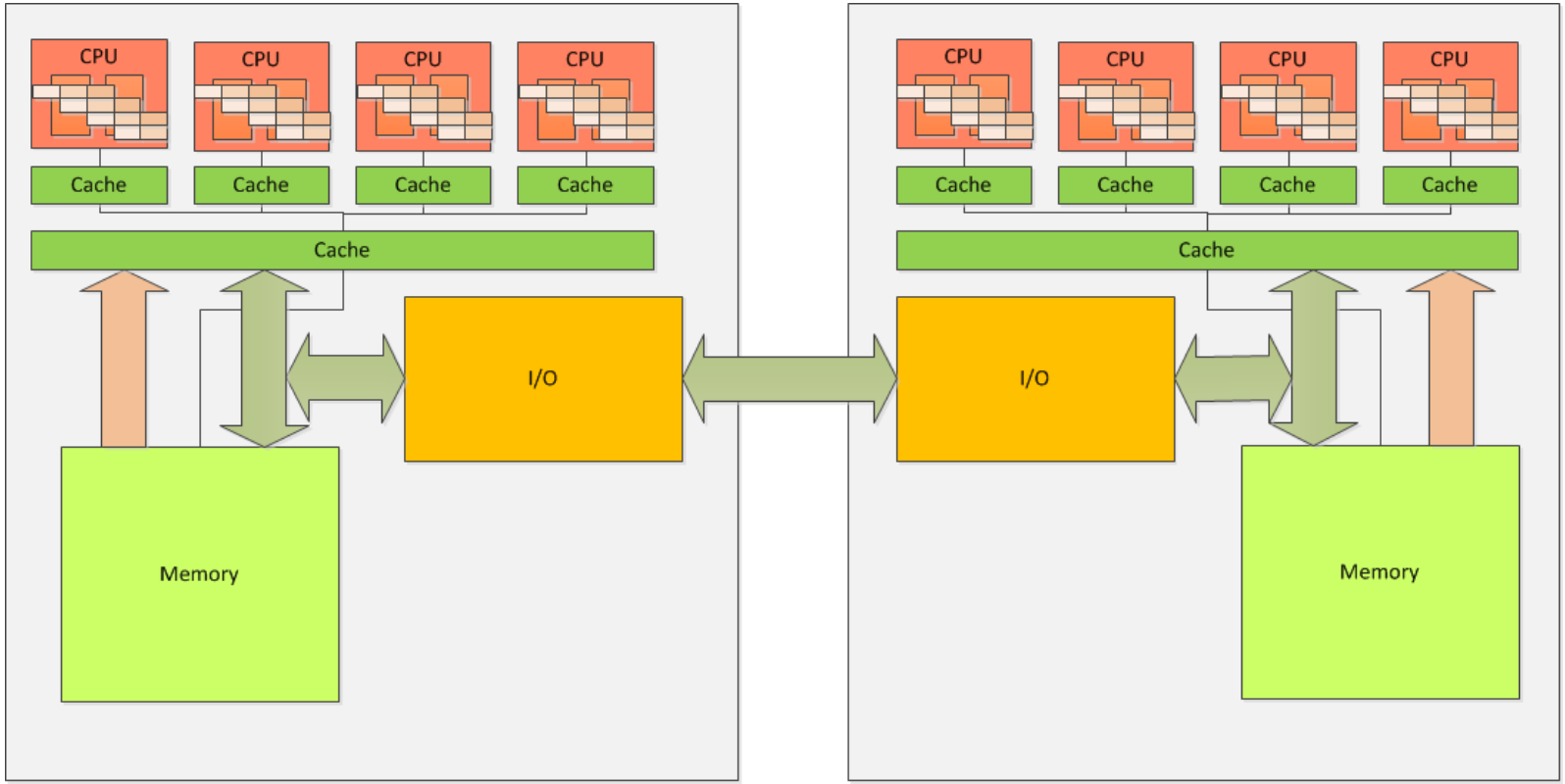
Objectives of Today‘s Lecture

- Document classification
- Parallel discrete event simulation

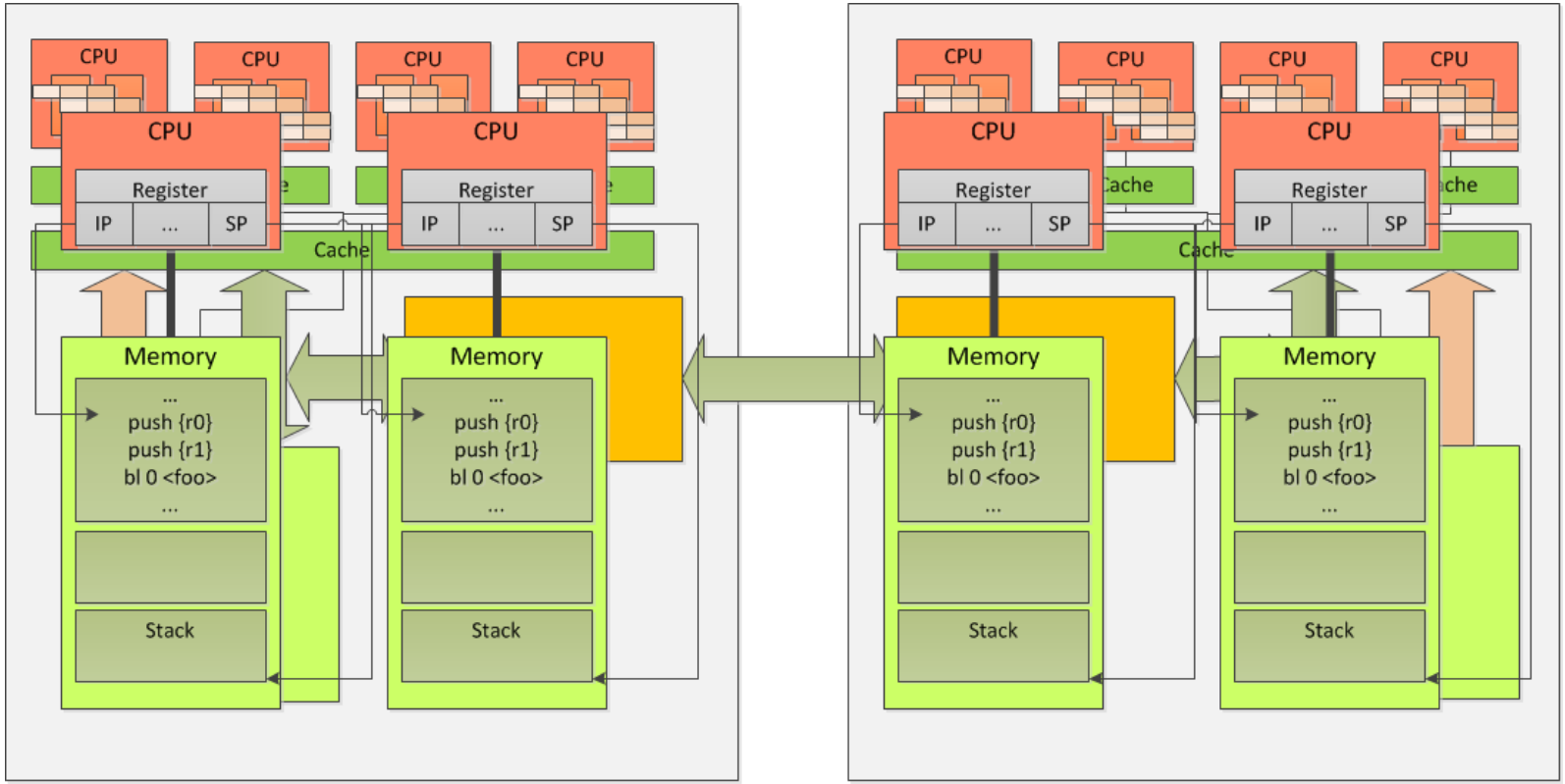
Concepts of Non-sequential and Distributed Programming

DESIGN AND IMPLEMENTATION OF PARALLEL APPLICATIONS

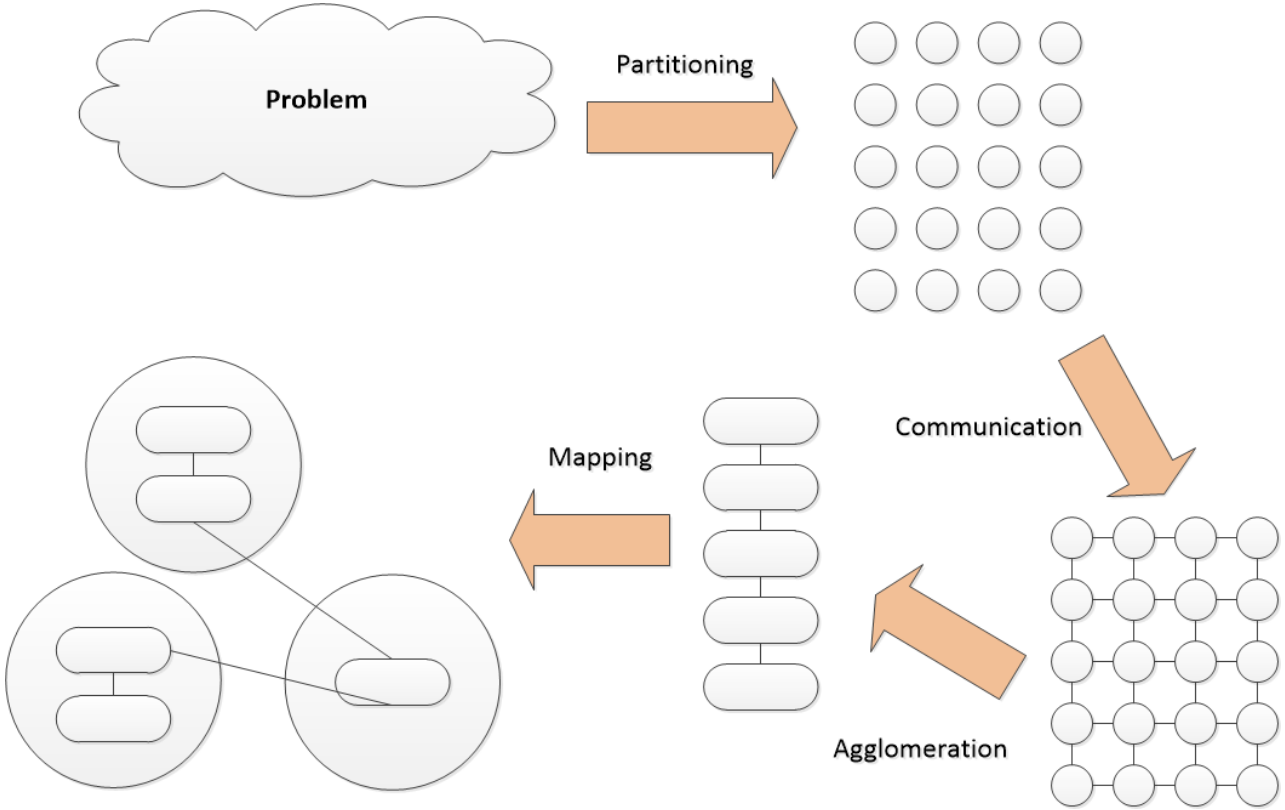
Machine Model



Machine and Execution Model



Foster's Design Methodology



Task/Channel (Programming) Model

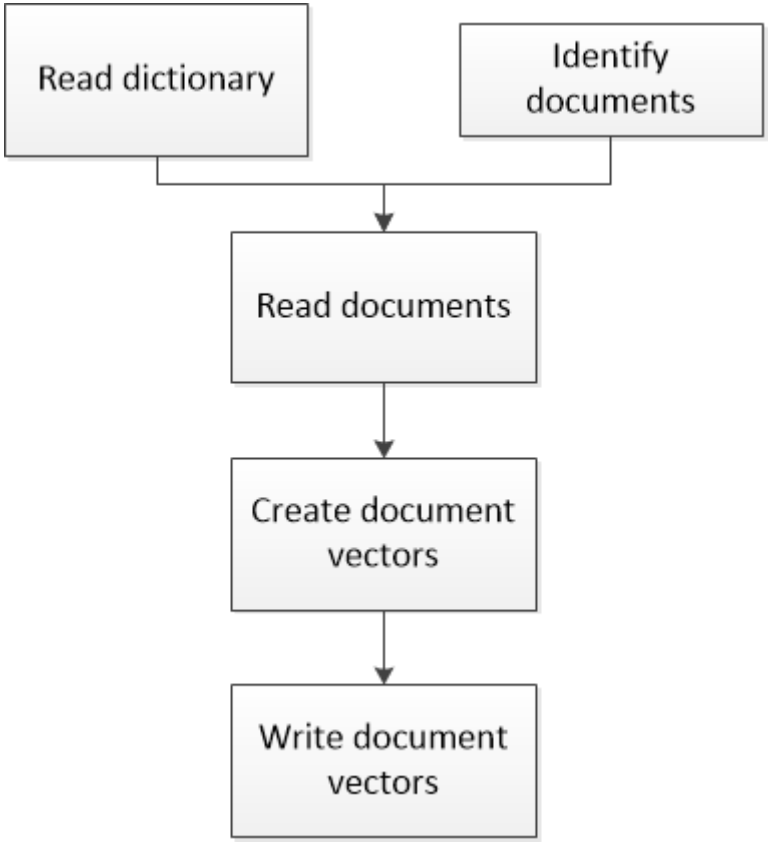
- The Task/Channel (Programming) Model serves as foundation for Forster‘s design methodology.
- A Task of the task/channel model is a part of the application with its own address space (process).
- Tasks can exchange data via messages using channels.
- A channel is a message queue connecting two specific tasks.
- If a task wants to receive a message, the task waits until the message is received (is blocked).
- Messages are sent immediately by the sender. The sender does not wait until the message is received.
- Thus, the task/channel model implements synchronous receive and asynchronous sending.

Concepts of Non-sequential and Distributed Programming

EXAMPLE: DOCUMENT CLASSIFICATION

Document Classification

- The application is aimed to generate vectors that classify documents according to a given dictionary.



Quinn: Par. Prg. in C with MPI + OMP

Document Classification – Partitioning

- Which (preferably independent) tasks (elements of work) can be identified?

- The processing of the individual documents is independent of each other.
- Additionally, the following steps to handle a document can be identified:
  - Reading the dictionary
  - Determining the document to be classified
  - (Receiving and) reading the document
  - Generation of the vector
  - (Send and) write the vector to a central vector database.

Document Classification – Communication

- What is the data that has to be transferred (from which task to which other task)?
- Which transfer of data depends on which other activities (calculation, communication)?

- First of all the documents have to be read and the vectors are to be stored. Using a parallel computer, reading and writing of data is performed by communication with file servers.
- The dictionary is to be read just at the start of the program and transmitted to all tasks.

- There is no communication between the tasks to calculate the vector for a specific document.

Document Classification – Agglomeration

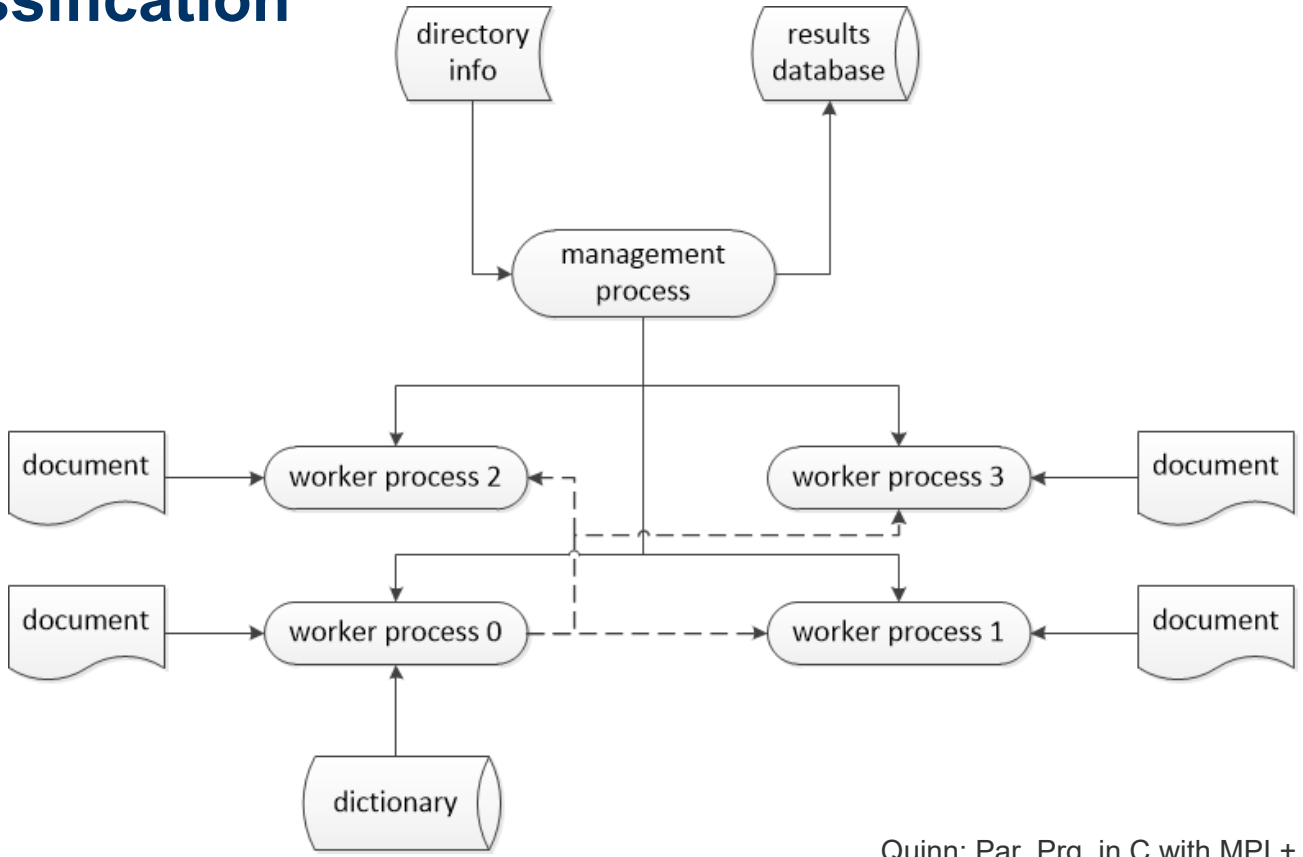
- Which tasks can be combined as long as the number of available processors is (much) smaller than the number of combined tasks (now processes)?
- How can the aggregation of the tasks be used to reduce communication between the processes?

- The tasks to be performed for every documents can be combined.
- But the reading of the dictionary at the start of the program may become a bottleneck.
- If the number of documents is greater than the number of processes to be created, several documents must be assigned to a process.
  - What is the load of the individual processes in this case?
  - What does this mean for the response time of the program?

Document Classification – Agglomeration II

- The reading of the dictionary should be designed as a one-time task with sending to all (working) processes if possible using broadcast communication.
- All steps to be processed for a single document are to be combined.
- Processes will process more than one document.
- To consider load distribution the work will provided by a management task. The process completed processing one document will ask for a new document.
- The collection of the vectors can be combined with work retrieval.

Document Classification



Quinn: Par. Prg. in C with MPI + OMP

Document Classification – Mapping

- Which combined tasks (processes) should be assigned to which processor?

- The process responsible to distribute the dictionary should be located on a node with short distances to the nodes holding the working processes.
- The requirement to imply short distances holds for the management process, too.

- No (further) communication between the work processes has to be considered.

Concepts of Non-sequential and Distributed Programming

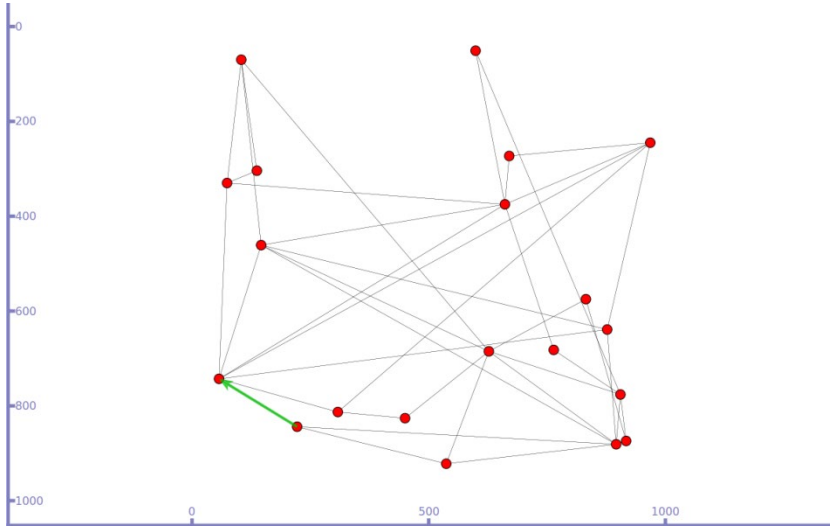
EXAMPLE: PARALLEL DISCRETE EVENT SIMULATION

Parallel Discrete Event Simulation

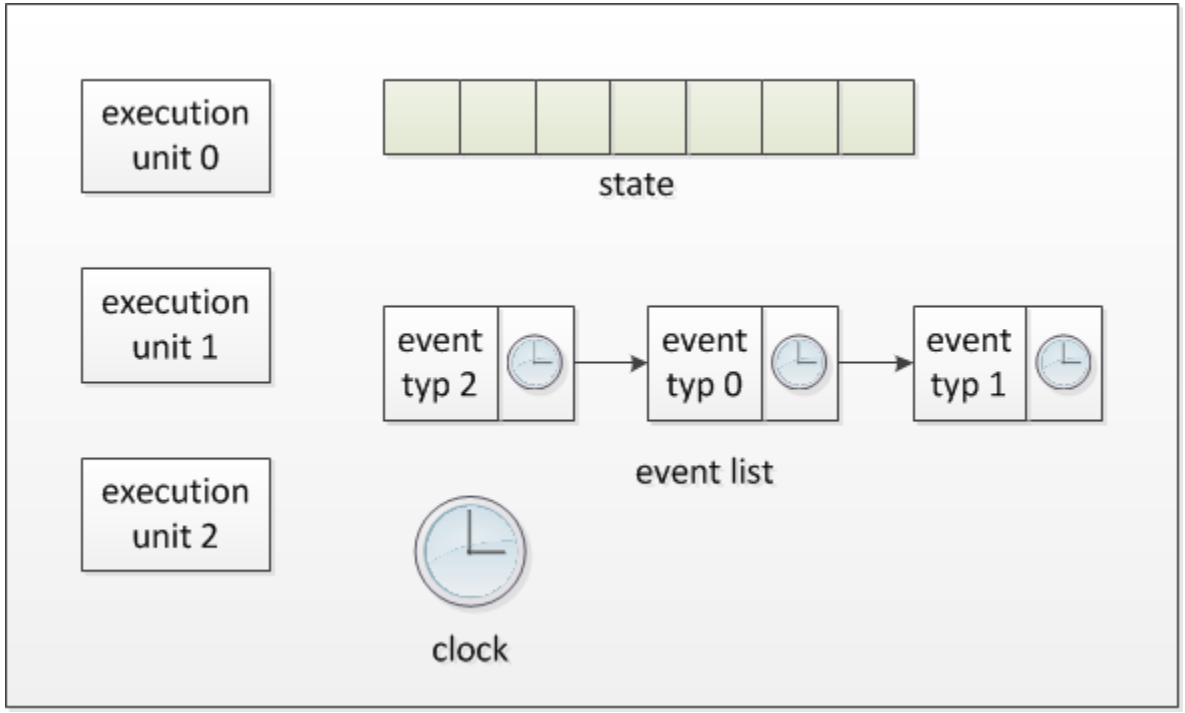
- Discrete event simulation are used, especially to simulate technical systems.
- The state of the system to be simulated and the changes of the states results from events and their processing.
- Each processing of an event again creates one or more events.
- Thus, new events are generated by processing events generated before. After handling an event it is removed from the system.
- Every events has a time at which it is to be processed by the simulation.
- The simulation time progresses according to the time of the events and their processing. The simulation time is not linked to the real time. Thus, the simulation time may run slower or faster than the real time.
- Each event takes place at a discrete point in time and may has a processing time.

Parallel Discrete Event Simulation II

- Examples:
  - Network simulators
  - ns-2
  - ns-3 - https://www.nsnam.org/
    - Simulation of cable-based and wireless networks.
    - Support for different network layers and
    - different network components.
  - Grid and cluster management systems
    - VRM



Parallel Discrete Event Simulation III



Parallel Discrete Event Simulation – Partitioning

- Which (preferably independent) tasks (elements of work) can be identified?

- The processing of events in parallel seams to be promising.
- But, the processing of the single event depends on the event itself as well as on the status of the simulation, which depends on the processing of other events, too.

- The processing of an single event depends on the steps to be performed to handle the specific event.
- This approach depends on the functions to be used to handle the single event. This may differ from type of events to type. For many events the amount of work to be done is limited.
- Additionally, a pool of tasks is to be used or the tasks has to be create a new process on demand. This comes with some overhead.

Parallel Discrete Event Simulation – Partitioning I

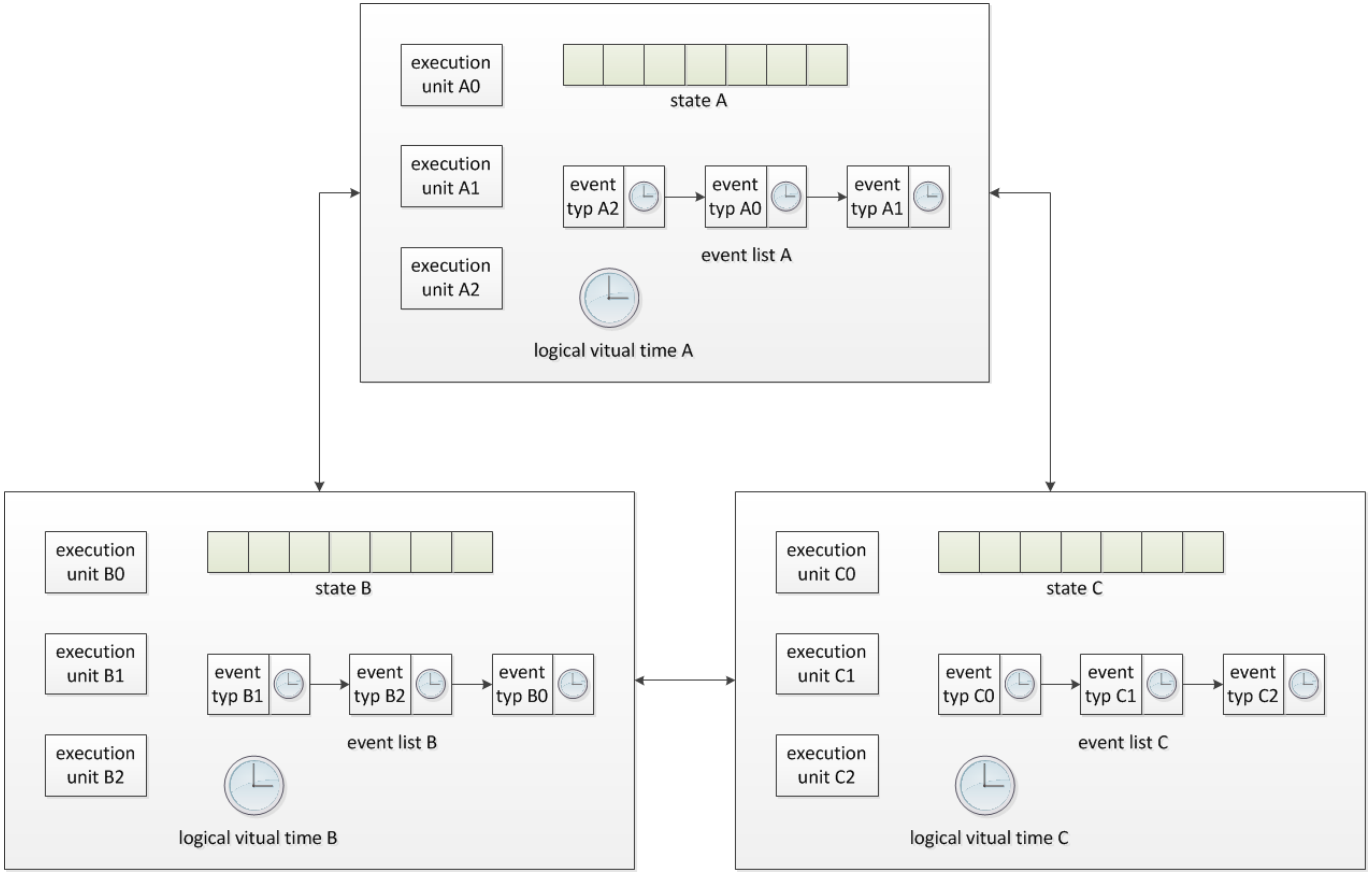
- In case several simulation runs are necessary the single run can be performed by a separated task (as a sequential program).
- The workflow from generation of various input data, start the simulation runs and collection of the results is to be managed.

- This approach is also called Monte Carlo simulation or Monte Carlo method.

Parallel Discrete Event Simulation – Partitioning II

- Regardless the challenges approaches for parallelization of discrete event simulations rely on the distribution of the event processing.
- Event processing has (hopefully) influence on a part of the simulation state only.
- The event processing is complex enough to justify the effort.

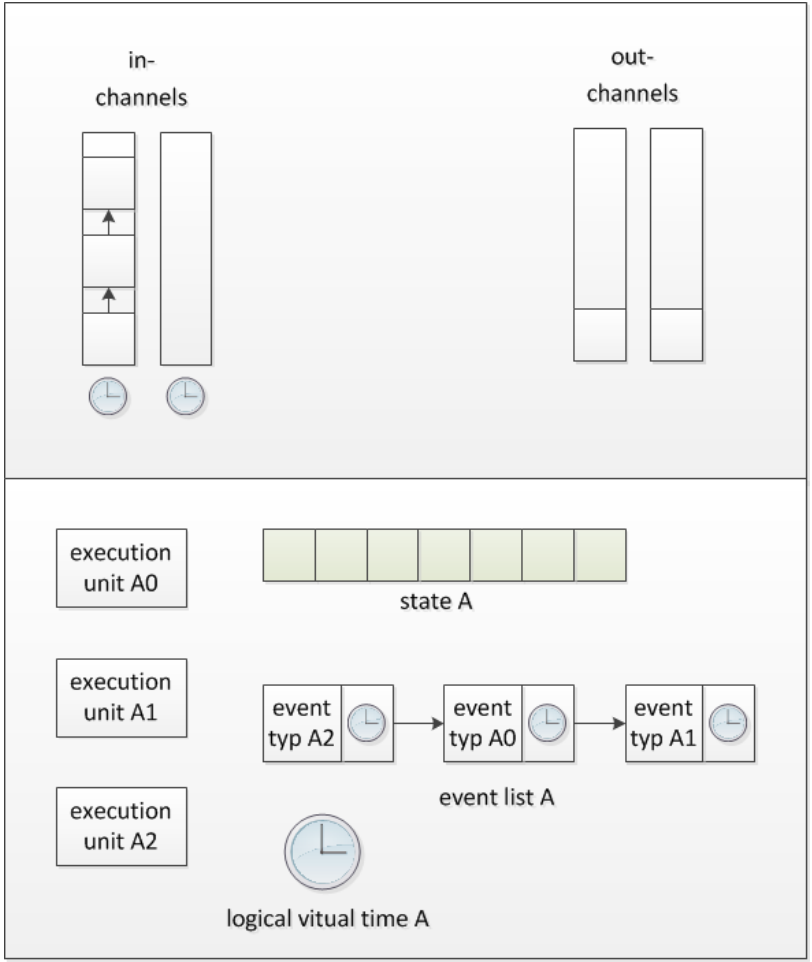
- But (at least) the simulation time must be coordinated.
- Thus the concept of "logical processes" (LP) is introduced.


execution unit A0
state A
execution unit A1
event typ A2
event typ A0
event typ A1
event list A
execution unit A2
logical vitual time A
execution unit B0
state B
execution unit B1
event typ B1
event typ B2
event typ B0
event list B
execution unit B2
logical vitual time B
execution unit C0
state C
execution unit C1
event typ C0
event typ C1
event typ C2
event list C
execution unit C2
logical vitual time C

Parallel Discrete Event Simulation – Communication

- What is the data that has to be transferred (from which task to which other task)?
- Which transfer of data depends on which other activities (calculation, communication)?

- In case an event created by one logical process is to be processed by another logical process, this event is to be transferred to the other logical process.
- The current local simulation time (logical virtual time) is to be promoted to all other logical processes.
- All clocks are synchronized for each event. All clocks always have the same time (global simulation time).
- The synchronization of the clocks can be implemented using barriers. This ensures the correct handling of local simulation time, but leads to a sequencing of the event processing, too.



- In order to handle the transfer of the event external events are received and stored in so called in-channels.
- There is a dedicated in-channel for every other logical process.
- Events to be sent to other LP are stored in out-channels.
- All events are sent and received between the LPs in sequence.
- The logical virtual time relies on the event time of the first events of all of the in-channels and the local event list.

Parallel Discrete Event Simulation – Approach I

- Based on the first steps of Foster‘s design methodology different approaches to handle the distribution and the processing of the events can be developed.
- The first approach is called the conservative approach.
- Following the conservative approach the events are processed according the event time strictly. Thus, the chronological order of event processing is preserved.
- The lowest time of events stored in all of the in-channels determines the logical virtual time of the logical process.
- If there are no events in the in-channels, the corresponding clock stops at the time of the last event.
- This may lead to a deadlock. To prevent deadlocks the logical processes send “zero messages” containing no event, but the local virtual time of the logical process.

Parallel Discrete Event Simulation – Approach II

- Another approach is called optimistic approach or timewarp approach.
- Following the timewarp approach the events are processed according to the chronological order even if in-channels are empty.
- Thus, local events can be processed without waiting for all of the other logical processes sending events.
- But the check for the correct local virtual time cannot be performed.
- Events from other logical processes may arrive after the later events of the local process are handled.
- If events "from the past" arrive with a lower time stamp, all processing steps must be rolled back.
- If events for other LPs are affected, these must be rolled back, too. To send a signal to roll back other LPs anti-messages are sent.

Parallel Discrete Event Simulation – Summary

- To find approaches to develop parallel discrete event simulations the steps of Foster‘s design methodology have to be combined.
- Based on the combination of partitioning and communication the different approaches – the conservative approach or the optimistic approach or timewarp approach – can be developed.

- The agglomeration is based on the design of the logical processes and combines the processing units that belong together (state of simulation) if possible.

- Finally the mapping should consider the communication between the logical processes. This is an all-to-all communication. Thus, all the overall distance should be small.

Concepts of Non-sequential and Distributed Programming

NEXT LECTURE

Institute of Computer Science
Department of Mathematics and Computer Science

Freie Universität Berlin

Introducing to the Concepts of Distributed Programming

APL IV: Concepts of Non-sequential and Distributed Programming (Summer Term 2023)