Event-Driven Multithreading Execution Platform for Real-Time On-Board Software Systems

Zain A. H. Hammadeh, Tobias Franz, Olaf Maibaum, Andreas Gerndt, Daniel Lüdtke
Sensor Data Fusion in BIRD (TET, BIROS) AOCS

Bispectral Infra-Red Detection

- Single-core
- Many sensors
- On-board data processing
- Main developer
- Launch: 2001
Sensor Data Fusion in BIRD (TET, BIROS) AOCS

Sensor A

Sensor C

Sensor B

Sensor D

Msg. A

1

F(A)

f(A)

1

E = F(A,B,C)

Msg. E

0, final

1

F(B)

f(B)

1

1

F(D,E)

f(D,E)

1

f(A)

f(B)

f(D,E)
Introduction: ASAP scheduling

Conservative scheduling in one code block:

Tasks with ASAP scheduling:
Outline

• Tasking Framework
  • API
  • Activation model
  • Call semantics
• Execution model
  • Scheduling and priority handling
• Tasking Modeling Language (TML)
• Outlook & Conclusion
Tasking Framework: Elements

TaskChannel

Sensor A
Msg. A
1
F(A)
f(A)

TaskEvent

Sensor C
C
5
E=F(A,B,C)
Msg. E

Sensor B
Msg. B
1
F(B)
f(B)

Sensor D
Msg. D
1
F(D,E)
f(D,E)

TaskInput

0, final
Tasking Framework: API

- **InputImpl**
- **Input**
- **Channel**
- **TaskImpl**
- **Task**
- **TaskProvider**
- **TaskArray**
- **Event**
- **Scheduler**

- Language: C++
Tasking Framework: Sequence

Sensor A
- Msg. A
  - push()
  - TaskChannel
  - TaskInput
  - Task
- F(A)
  - F(A, B, C)
  - E
  - f(A)

Sensor B
- Msg. B
  - TaskChannel
  - TaskInput
  - Task
- F(B)
  - F(D, E)
  - f(B)

Sensor C
- Msg. C
  - TaskChannel
  - TaskInput
  - Task
- E = F(A, B, C)
  - f

Sensor D
- Msg. D
  - TaskChannel
  - TaskInput
  - Task
- F(D, E)
  - f(D, E)
Tasking Framework: Sequence cont’d

- **TaskChannel**
  - push
  - notifyInput
  - synchronizeStart
  - synchronizeEnd
  - reset

- **Input**
  - activate
  - isActivated*

- **Task**
  - perform
  - synchronizeStart
  - synchronizeEnd
  - reset

- **Scheduler**
  - perform
  - synchronizeStart
  - execute
  - synchronizeEnd
  - reset

- **Executor**
  - perform
  - synchronizeStart
Activation model

1. \( \text{in}_0 \rightarrow \text{Task} \)
2. \( \text{in}_0, \text{final} \rightarrow \text{Task} \)
3. Period Relative-time

Relative-time

SpW

Task \( \tau_1 \)

Task \( \tau_2 \)
Call semantics

1. Asynchronous (Default)

2. Synchronous (Tasking::Group)
Outline

• Tasking Framework
  • API
  • Activation model
  • Call semantics

• Execution model
  • Scheduling and priority handling

• Tasking Modeling Language (TML)
• Outlook & Conclusion
Execution Platform

Platform Specific

SchedulerExecutionModel

Mutex

Signaler

ClockExecutionModel
Execution model

Scheduler

Task_0
Task_1
... 
Task_n

Scheduling policy

Queue

Executors (threads)

- Work-conserving scheduling
- The load is balanced on the available executors

Scheduling policies:

FIFO
LIFO
SPNP

Thread’s life cycle:

Create
Running
Signal
Waiting
Join
Wait(&cond_var)
Scheduling and priority handling

Application

Scheduler1
- Task0
- Task1
- Taskn1

Scheduling policy
Queue

Scheduler2
- Task0
- Task1
- Taskn2

Scheduling policy
Queue

Scheduler3
- Task0
- Task1
- Taskn3

Scheduling policy
Queue

Scheduler4
- Task0
- Task1
- Taskn4

Scheduling policy
Queue

Partitioned scheduling

Global scheduling
Outline

- Tasking Framework
  - API
  - Activation model
  - Call semantics
- Execution model
  - Scheduling and priority handling
- Tasking Modeling Language (TML)
- Outlook & Conclusion
**Tasking Modelling Language (TML)**

- Advanced editor with modeling support + Integration into Eclipse IDE
- Modeling Workflow
  1. Data types
  2. Component types
  3. Component connection
- Artifact generation

![Diagram showing the modeling workflow and artifact generation](image-url)
Outline

• Tasking Framework
  • API
  • Activation model
  • Call semantics
• Execution model
  • Scheduling and priority handling
• Tasking Modeling Language (TML)

• Outlook & Conclusion
Outlook

- Work stealing
- Open source:
  - introducing releases
  - contribution rules
  - legal regulations
- Bare-metal
Conclusion

• Tasking Framework
  • software development library
  • event-driven multithreading execution platform
  • C++
  • compatible with the POSIX-based RTOS
  • dedicated for data-driven on-board software systems

• Interested?

Website: https://www.dlr.de/sc/srv

ResearchGate: OSS