REAL-TIME VIRTUAL RESOURCES IN THE CLOUD

ALOYSIUS K. MOK
UNIVERSITY OF TEXAS AT AUSTIN
SEPT 15, 2018
Synopsis

- Yesteryear - B777 AIMS
- Real-Time Virtual Resource
- Application Demo
- Challenges
AIMS runs on a multiprocessor system connected by a SAFEbus ultra-reliable bus. The objective is to develop a cyclic schedule for each processor and one schedule for the SAFEbus ultra-reliable bus.

AIMS contains 155 application processes and 910 communication processes.

Each process is instantiated to multiple instances within the major frame. The total number of application processes instances within one major frame is 252 and the number of communication processes is 778.

Total number of constraints is 8626.
AIMS Architecture

From <process> ( <rate> <duration> ) to <process>
  length <xfer duration> latency <latency bound>

From 5:16:2 (5Hz 1.126ms) to 7:16:2
  length 12.467us latency 200000us
From 5:16:2 (5Hz 1.126ms) to 7:16:3
  length 21.633us latency 93918us
From 7:16:2 (5Hz 1.126ms) to 5:16:2
  length 12.467us latency 200000us

...
Design Objective

The design objective is to construct a cyclic executive (a major frame) for each processor and one schedule for the SAFEbus backplane bus. The length of a major frame is the LCM (least common multiple) of the periodic tasks.

Features of the scheduling problem:

- Periodic tasks with jitter constraints
- Communication latency
- Asynchronous communication
- Cyclic Dependency
Traditional Solution: Cyclic Executive

This is a massive constraint satisfaction problem to meet deadlines.

Tools exist to synthesize cyclic executives automatically.
One of the Bibles

John A. Stankovic
Cyber physical systems (CPSs) share resources in the open system environment

- No global scheduler to schedule all tasks
- Each application schedules its own tasks to meet dynamic QoS demand and independent of other applications
Open System Environment

COTS Platform

Temporal and Spatial Partitioning

Application 1 Task Group Scheduler

Application 2 Task Group Scheduler

... Application n Task Group Scheduler
Resource Allocation is Commitment to an Application (Application Task Group)

**Goals:**
- Temporal firewall between application task groups: no detectable timing interference
- No change to application level (task group) scheduler
- No global schedulability analysis
- No interaction between application level scheduler and resource level scheduler needed after admitting application
- Full utilization of resource
Idealized Real-Time Resource Sharing

Resource Level Scheduler

Per Application (Task Group) Scheduler

fraction $\alpha$, Infinite time slicing
Liu & Layland Task Groups

Resource Level Scheduler

request times, deadlines

fraction $\alpha$

Per Task Group Scheduler

Liu & Layland, JACM 1973
A conceptually clean resource sharing strategy is to adopt weighted infinite time slicing.

Infinite time slicing is impractical 😞.

Issues with practical time partitioning schemes:

• How often must we switch partitions?
• What information needs to be exchanged between schedulers on different levels?
• When should information exchange occur?
• How is admission/schedulability analysis handled?
Open System Environment Revisited

- Each application is implemented by a task group.
- No global scheduler at the task level.
- Application task groups must not conflict with one another in task execution.
- Each application is responsible for scheduling its own task group on a “dedicated” processor of specific speed which is known to the application group.
- Resources may be multiplexed and shared between applications.
In reality, applications do not run on dedicated physical processors.

**Observation:**

From the application programmer’s point of view, time partitioning a CPU is as if program executes on a slower CPU that runs at a varying speed.

**Question:**

What is a good way to bound the speed variation (evenness of service provision)?

\[
\text{Partition } \Pi = \{(1, 4, 5, 7) \, 8\}^* 
\]
Real Time Virtual Resource

Delay Bound (also called jitter bound)

Maximum delay $\Delta$ that a partition must wait to get its share $\alpha$ of the resource for any time interval starting at any point in time

Example: for a partition with delay bound = 0.1 second, 10% of a CPU that executes $10^8$ instructions per second will provide $10^7$ instructions in any 1.1 second
Real Time Virtual Resources

Physical Resource(s)

RTVR 1 \((\alpha_1, \Delta_1)\)

RTVR 2 \((\alpha_2, \Delta_2)\)

RTVR n \((\alpha_n, \Delta_n)\)

Task Group 1 Scheduler

Task Group 2 Scheduler

Task Group n Scheduler

Task 1

Task 2

Task n

.....

.....
Task Level Scheduling

CPU partitioning (for $k$-regular partitions) is almost transparent to task scheduling

Rate Monotonic:
$$\sum \left( \frac{C_i}{(p_i-k)} \right) \leq m(2^{1/m} - 1) \alpha(\Pi)$$

Earliest Deadline First (Shigero, Takashi & Kei):
$$\sum \left( \frac{C_i}{(p_i-k)} \right) \leq \alpha(\Pi)$$

Note: $k$ is in basic time units (each time slice is one time unit)
Decoupling Resource Sharing in Timeliness Verification

How do we verify timing properties of programs running on a Real Time Virtual Resource with delay bound $= \Delta$?

- Add $\Delta$ to minimum allowable separation (delay) and subtract $\Delta$ from maximum allowable separation (deadline) between event pairs of interest.

- Think of it as verifying timing properties in systems where there is a jitter as big as $\Delta$ in the spacing between any two events.
Challenge of Managing Multiple Resources

- Each application in an open system environment has its own dedicated (virtual) resources that are required for running the application.

The notion of degrees of regularity captures the temporal “evenness” of resource provisioning.

CPS Application 1 + Performance Constraints

CPS Application 2 + Performance Constraints
A composite resource partition $C$ is a pair $(\mathbb{P}, E)$

- $\mathbb{P} = \{P_1, P_2, \ldots, P_n\}$ is a set of resource partitions
- $E$ is a binary relation on the set of all physical resources denoting the access order

Extend Regularity-based Resource Partition (RRP) model to multiple resources
The model car has two sub-components in Car controller component:
- Vision Controller, Speed and Steering Controller

Both controllers collaborate with each other and may change operating modes that may have different resource requirements.
Cyber-physical systems are a combination of information technology and physical services, e.g., robotic manufacturing.

Sensor input may be analyzed in the cloud and the results relayed back to the physical system components to take action, e.g., robot on Mars, undersea.

Tradeoffs between the quality of analysis results and system response time may be critical. How to do it best?
Future Challenges

- Characterization of resource usage for real-time load management in the *heterogeneous system environment* where there are many device types: sensors, actuators, computing platforms.

- Resource allocation and scheduling policies for heterogeneous systems to guarantee QoS requirements.

- Utility-Based Virtualization?