CPU Utilization Control in Distributed Real-Time Systems

Chenyang Lu
CSE 5205

Highlight

- Common class of computing problems
  - MIMO: multi-input (knobs), multi-output (objectives)
  - Coupling between objectives.
  - Constraints on knobs.

- Model Predictive Control
  - Optimization + Prediction + Feedback

Why CPU Utilization Control?

- Overload protection
  - CPU over-utilization $\rightarrow$ system crash

- Meet response time requirement
  - CPU utilization $<$ bound $\rightarrow$ meet deadlines

Challenge: Uncertainties

- Execution times?
  - Unknown sensor data or user input

- Request arrival rate?
  - Aperiodic events
  - Bursty service requests

- Disturbance?
  - Denial of Service attacks

Control-theoretic approach

$\Rightarrow$ Robust utilization control in face of workload uncertainty

End-to-End Tasks

Distributed Real-Time Systems

- Periodic task $T_i$ = sequence of subtasks {$T_{ij}$} on different processors
  - All the subtasks of a task run at a same rate

- Task rate can be adjusted
  - Within a range
  - Higher rate $\rightarrow$ higher utility

Problem Formulation

- $B_i$: Utilization set point of processor $P_i$, $1 \leq i \leq n$
- $u_i(k)$: Utilization of $P_i$ in the $k^{th}$ sampling period
- $r_j(k)$: Rate of task $T_j$, $1 \leq j \leq m$ in the $k^{th}$ sampling period

$\min_{\{r_j(k) \mid 1 \leq j \leq m\}} \sum_{k=1}^{n} (B_i - u_i(k))^2$

subject to rate constraint:

$r_{\min} \leq r_j(k) \leq r_{\max}, 1 \leq j \leq m$
**Single-Input-Single-Output (SISO) Control**

**Single Processor**

- Set point: U_{i} = 89%
- Task Rates: \( R_{1}: (1, 5) \text{ Hz} \) \( R_{2}: (10, 20) \text{ Hz} \)

---

**New in Distributed Systems**

- Need to control utilization of multiple processors
- Utilization of different processors are coupled with each other due to end-to-end tasks
  - Replicating a SISO controller on all processors does not work!
- Constraints on task rates

---

**Control Theoretic Methodology**

1. Derive a dynamic model of the system
2. Design a controller
3. Analyze stability

---

**Dynamic Model: Single Processor**

\[
u(k) = u(k-1) + g \sum_{T_{ij} \in S} c_{ij} \Delta r_{j}(k-1)
\]

- \( S \): set of subtasks on \( P_{i} \)
- \( c_{ij} \): estimated execution time of \( T_{ij} \)
- \( g \): utilization gain of \( P_{i} \)
  - ratio between actual and estimated change in utilization
  - models uncertainty in execution times

---

**Dynamic Model: Multiple Processors**

\[
u(k) = u(k-1) + GF\Delta r(k-1)
\]

- \( G \): diagonal matrix of utilization gains
- \( F \): subtask allocation matrix
  - models the coupling among processors
  - \( f_{ij} = c_{ij} \) if task \( T_{j} \) has a subtask \( T_{ij} \) on processor \( P_{i} \)
  - \( f_{ij} = 0 \) if \( T_{j} \) has no subtask on \( P_{i} \)

---

**EUCON: Multi-input-Multi-output Control**

**Distributed System (m tasks, n processors)**

- \( B_{m \times n} \): diagonal matrix of utilization gains
- \( F_{n \times m} \): subtask allocation matrix
  - models the coupling among processors
  - \( f_{ij} = c_{ij} \) if task \( T_{j} \) has a subtask \( T_{ij} \) on processor \( P_{i} \)
  - \( f_{ij} = 0 \) if \( T_{j} \) has no subtask on \( P_{i} \)
Model Predictive Control

- Suitable for coupled MIMO control problems with constraints.
- Compute input to minimize cost over a future interval.
  - Cost function: tracking error and control cost.
  - Predict cost based on a system model and feedback.
- Compute input subject to constraints.
- Optimization + Prediction + Feedback

Cost Function

- Cost

\[ V(k) = \sum_{i=0}^{M} \left[ \| e(k+i) - \text{ref}(k+i) \| + \sum_{i=0}^{M-1} \| \Delta u(k+i) - \Delta u(k+i-1) \| \right] \]

Tracking Error
Control Cost

- Reference trajectory: exponential convergence to B

\[ \text{ref}(k+i) = B - e^{-\frac{k+i}{T_{ref}}} (B - u(k)) \]

Model Predictive Controller

At the end of each sampling period
- Compute inputs in future sampling periods \( \Delta u(k), \Delta u(k+1), \ldots, \Delta u(k+M-1) \) to minimize the cost function
- Cost is predicted using
  - (1) feedback \( u(k-1) \)
  - (2) approximate dynamic model
- Apply \( \Delta u(k) \) to the system

At the end of the next sampling period
- Shift time window and re-compute \( \Delta u(k+1), \Delta u(k+2), \ldots, \Delta u(k+M) \) based on feedback

EUCON Controller

Stability Analysis

- Stability: utilization of all processors converge to set points
- Derive stability condition \( \rightarrow \) range of G
  - Tolerable variation of execution times
- Provides analytical assurance despite uncertainty

Stable System

- CPU utilization
- Time (sampling period)
- P1 – P2 – – – Set Point
- execution time factor = 0.5
  (actual execution times = 0.5 of estimates)
Unstable System

Stability
- Stability condition → tolerable range of execution times
- Analytical assurance on utilizations despite uncertainty

Overestimation of execution times prevents oscillation

Predicted bound for stability

FC-ORB Middleware

End-to-End Object Request Broker
- Release guard for end-to-end tasks
- Priority management
  - Rate adaptation → continuous priority changes
  - Thread-per-priority → high overhead
  - Thread-per-subtask: Change priority only when the order of task rate changes

End-to-End Utilization Control Service
- Implements EUCON (End-to-end Utilization Control)
- Provides functional and performance portability
Task Migration
- Fault model: permanent processor failure
- Subtasks have backups on different processors
- Utilization control + fault-tolerance
  - Automatic controller reconfiguration
  - Handle overload caused by task migration

Experimental Setup
- 12 tasks (25 subtasks) and 4 Pentium IV processors
- KURT Linux 2.4.22
- Rate Monotonic Scheduling
- Subtasks on Norbert have backups on other processors

Goal 1: Robust Utilization Control
- Execution times change at runtime
- Disturbance from external resource contention

Goal 2: Performance Portability
- Same utilization – portable performance
  - Even on different systems with different computing capacity

Goal 3: Fault Tolerance
- 1. Norbert fails.
- 2. move its tasks to other processors.
- 3. reconfigure controller
- 4. control utilization by adjusting task rates
Summary: Model Predictive Control

- Robust utilization control for distributed systems
- Handle coupling among processors
- Enforce constraints on task rates
- Analyze tolerable range of execution times

References

- Project page: http://www.cse.wustl.edu/~lu/control.html