Virtual Gang Scheduling of Parallel Real-Time Tasks

Waqar Ali*, Rodolfo Pellizzoni+ and Heechul Yun*
Many emerging workloads are parallel real-time tasks
Can benefit from parallelization on multicore platforms
Execution time of a task depends on co-running tasks

Due to interference in shared hardware resources

• A **gang scheduling** approach to avoid interference

• Schedule one gang task at a time

---

• Not all tasks are fully parallelizable
• Low utilization

Our Approach: Virtual Gang Scheduling

• Virtual Gang
  • A statically paired group of real-time tasks
  • Scheduled as a single gang task

• Benefits
  • Can achieve higher utilization
  • Co-running tasks never change
  • Can tightly bound interference

• Q: How to form the virtual gangs?

Real-Time Tasks: $t_1, t_2, t_3, t_4$
Virtual Gang ($t_g$): $t_1 + t_2 + t_3$
$prio(t_g) > prio(t_4)$
Considerations

• Interference
  • Co-running tasks influence execution timing

• Precedence constraints
  • Common in real applications
  • Limit feasible task pairings

System Model

• A multicore platform with \( m \) unit-speed cores
• \( n \) periodic, rigid real-time gang tasks \( \Gamma = \{ \tau_1, \tau_2, \ldots, \tau_n \} \)

• A real-time gang task: \( \tau_i = (c_i, h_i, r_i, T_i) \)
  • \( c_i \): WCET in isolation
  • \( h_i \): #of cores required to run (\( 1 \leq h_i \leq m \))
  • \( r_i \): Resource demand \( r_i \in [0,1] \)
  • \( T_i \): Period of \( \tau_i \)

• Precedence constraints described by a set of DAGs
A candidate-set: \( \Delta_T = \{ \forall \tau_i \in \Gamma \mid T_i = T \} \)
- Tasks that share a common period \( T \)
- A small number of candidate-sets exists in \( \Gamma \)

A virtual gang: \( w_l = (C_l, H_l, R_l, T_l) \)
- A subset of tasks within a candidate set \( \Delta_T \)
- \( C_l \): WCET of the virtual gang
- \( H_l \): collective core demand \( (1 \leq H_i \leq m) \)
- \( R_l \): collective resource demand \( (R_i \geq 0) \)
- \( T_l \): common shared period

System Model
A virtual gang $w_i$’s WCET: $C_l = \max_{\forall \tau_k \in w_l} \{c_k\} \times \max(R_l, 1)$

- $R_l < 1$: suffers no interference until the resource is over-utilized
- $R_l > 1$: applies a linear scaling
For a given candidate set of $N$ real-time tasks with the period $T$
Form a set of virtual gang tasks that minimize completion time
Subject to a given set of precedence constraints
Based on Satisfiability Modulo Theories (SMT)

- **Step-1:** Identify the parameters of the optimization problem and the optimization variable
- **Step-2:** Write the constraints that must be satisfied by the parameters in a feasible solution

See the paper for the details.
Greedy Heuristic Algorithm

• High-level idea
  • Group tasks with similar WCET values while minimizing combined resource utilization
  → higher core utilization, less slowdown

• Algorithm
  • Sort tasks by WCET
  • Pick the longest
  • Find feasible co-runners
  • Rank the feasible co-runners considering resource utilization
  • Form a virtual gang and repeat

• Not optimal but fast and effective

```
Algorithm 1: Virtual Gang Formation Heuristic

1: Input: Candidate Set (\(\Delta_T\)), Number of Cores (m)
2: Output: Taskset comprising virtual gangs
3: function gang_formation(\(\Delta_T\), m)
4:    \(pq\) = sort_tasks_by_wcet(\(\Delta_T\))
5:    virtualGangs = ()
6:    while not_empty(pq) do
7:        \(\tau_i\) = pq.pop()
8:        \(f_i\) = family(\(\tau_i\))
9:        partners = ()
10:       for \(\tau_j\) \in pq do
11:           if \(\tau_i.h + \tau_j.h \leq m \land \tau_j \notin f_i\) then
12:               partners \leftarrow partners \cup \{\tau_j\}
13:               \(pq_{ij}\) = score_partners(partners)
14:       while not_empty(pq_{ij}) do
15:           \(\tau_p\) = pq_{ij}.pop()
16:           \(\tau_i = merge(\tau_i, \tau_p)\)
17:           pq.remove(\(\tau_p\))
18:           update_partners(\(\tau_i, pq_{ij}\))
19:       virtualGangs \leftarrow virtualGangs \cup \{\tau_i\}
20:    return virtualGangs
```
For a taskset of virtual gangs \( \{w_1, w_2, \ldots, w_l\} \) and under the rate-monotonic priority assignment, the response time \( R_i \) of gang \( w_i \) can be iteratively computed using:

\[
R_i^{k+1} = C_i + \sum_{\forall w_j \in hp(w_i)} \left\lfloor \frac{R_i^k}{T_j} \right\rfloor C_j
\]

- \( C_i \) is the WCET of \( w_i \) + the WCET of all \( w_j \) (with the same period \( T \) as \( w_i \)) that come before \( w_i \) in the linear execution order.

First work which enables schedulability analysis of real-time gang tasks with precedence constraints!
Evaluation

• Simulation study with randomly generated synthetic tasksets
  • Lightly parallel \( (h=[1, \lceil 0.3 \times m \rceil]) \)
  • Mixed \( (h=[1, m]) \)
  • Heavily parallel \( (h=[\lceil 0.3 \times m \rceil, m]) \)
  • See the paper for other parameters (period, tasks, resource demand, etc.)

• Compared scheduling schemes
  • RT-Gang: one gang task at a time \(^1\)
  • Virtual Gang (SMT/Greedy): our approach
  • Gang-FTP: fixed-priority gang scheduling \(^3\)
  • Threaded: vanilla Linux (global FP scheduling) \(^4\)

---

\(^4\) J. Fonseca et al., “Improved Response Time Analysis of Sporadic DAG Tasks for Global FP Scheduling,” In RTNS (2017)
Results (with Precedence Constraints)

- Heuristic performs very close to the optimal algorithm!
- Schedulability is highest for virtual-gang (SMT)
Results (w/o Precedence Constraints)

When interference is considered:

- Virtual Gang scheme outperforms every other analysis for all taskset types
Runtime of Gang Formation Algorithms

SMT can be very slow!

Even for \( N = 50 \), the heuristic gives a solution in less than a second!
Summary

• Proposed a virtual-gang scheduling scheme
  • Achieves high utilization
  • Considers interference and precedence constraints
  • Enables effective and tight timing analysis

• Future work
  • Virtual gang formation on heterogeneous platforms (w/ accelerators)
  • Empirical evaluation with real-world workloads

https://github.com/CSL-KU/VirtualGang-Simulator