2012. 05. 21

Conflict Avoidance Scheduling using Grouping List for Transactional Memory

Dongmin Choi*, Seung Hun Kim**, and Won W. Ro*

*Mobile Communication Division
Samsung Electronics Co., Ltd.
Gumi, Republic of Korea

**School of Electrical and Electronic Engineering
Yonsei University
Seoul, Republic of Korea

The 17th International Workshop in High-Level Parallel Programming Models and Supportive Environments

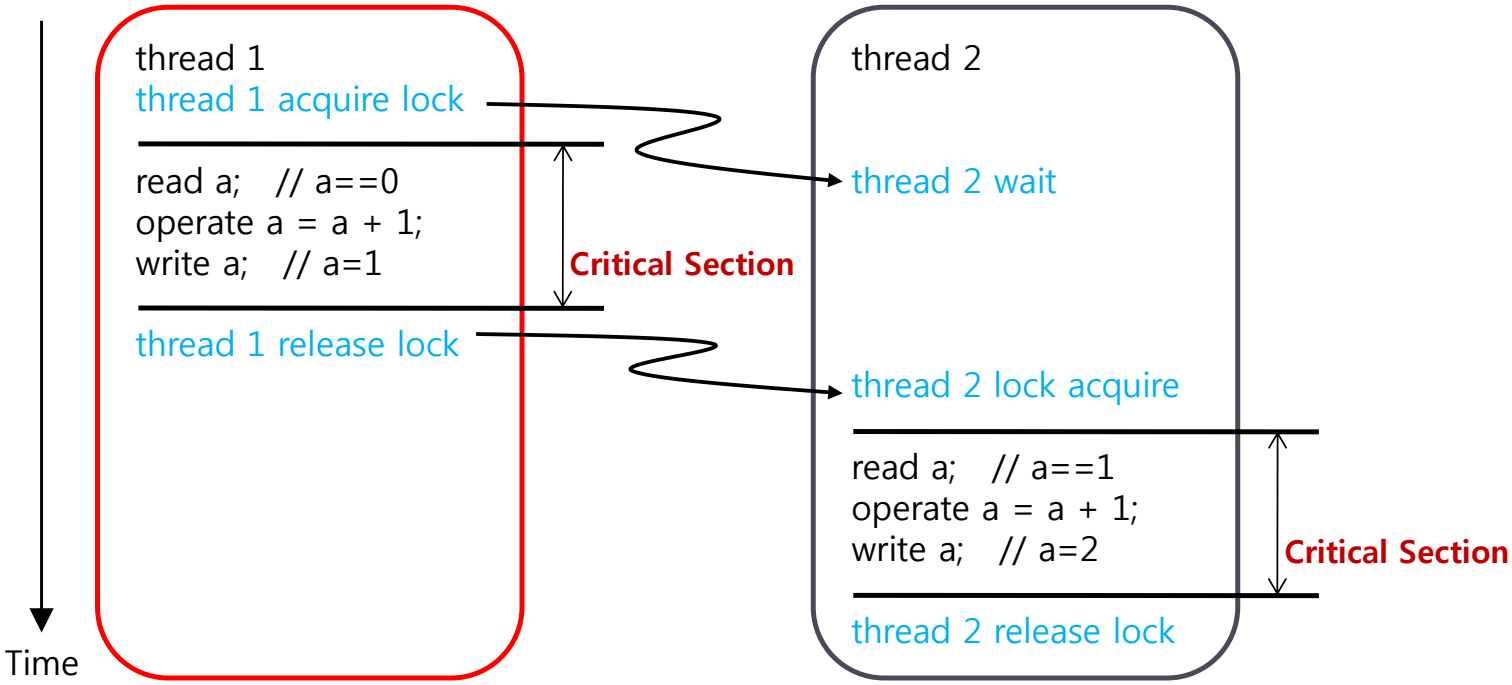
esCaL
Embedded Systems and Computer Architecture Lab.

Overview

➢ Transactional Memory systems

- A promising synchronization method for a parallel processing
- Overcoming the *limitations of lock-based* synchronization

  → dead-lock, live-lock, priority inversion, convoying, less parallelism …
- Lock based synchronization



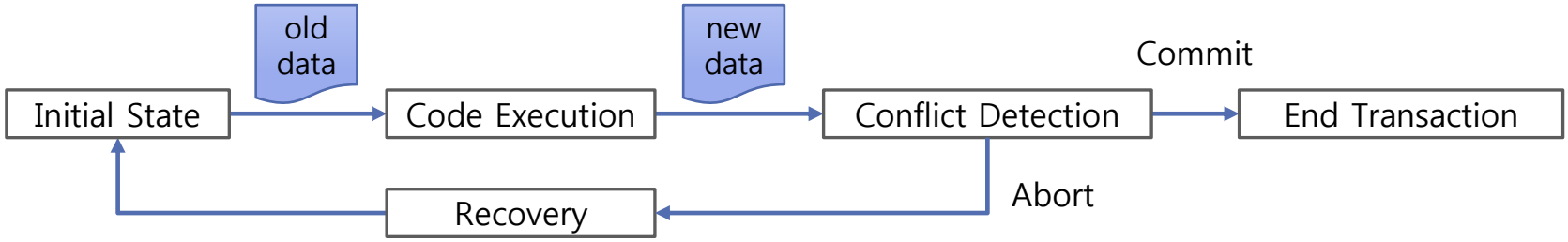
Overview

➢ Transactional Memory systems

- A promising synchronization method for a parallel processing
- Overcoming the *limitations of lock-based* synchronization

  → dead-lock, live-lock, priority inversion, convoying, less parallelism …

- Flow of transactions



  → eliminating mutual exclusion

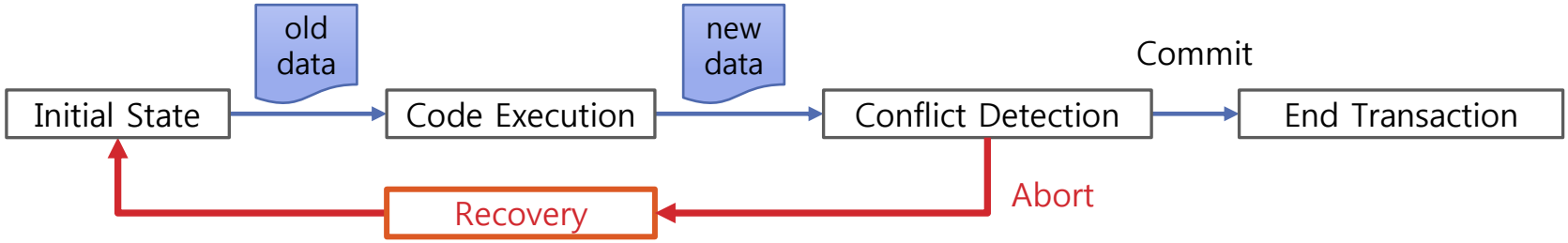
  → a hazard for a correct synchronization = conflict

  → rollback to initial state

Overview

➢ Transactional Memory systems

- A promising synchronization method for a parallel processing
- Overcoming the *limitations of lock-based* synchronization
  → dead-lock, live-lock, priority inversion, convoying, less parallelism …
- Flow of transactions



➢ Performance degradation factor

- Frequent rollback operation !
  → especially, in the case of high contention with large-sized transactions
- Proposed solution: **Conflict Avoidance Scheduling** (CAS) for transactions
  → 23% of performance improvement (avg.) over the original system

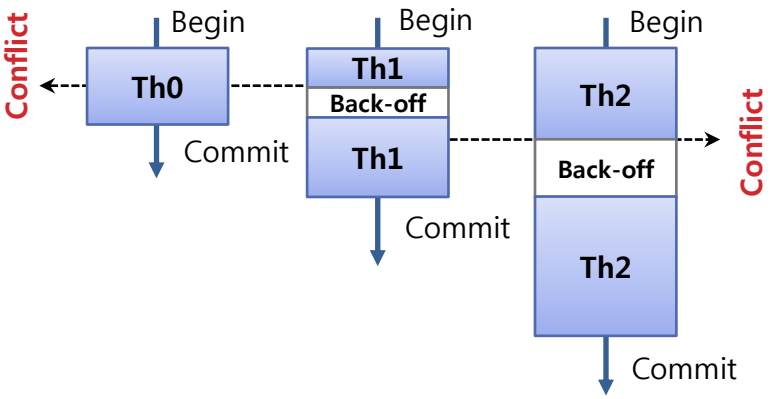
Outline

- **Related Work**
  - Contention management
  - Prior work
- **Conflict Avoidance Scheduling Mechanism**
  - Scheduling operation
  - Implementation
- **Results and Analysis**
  - Execution time analysis
  - Contention analysis
- **Conclusion**

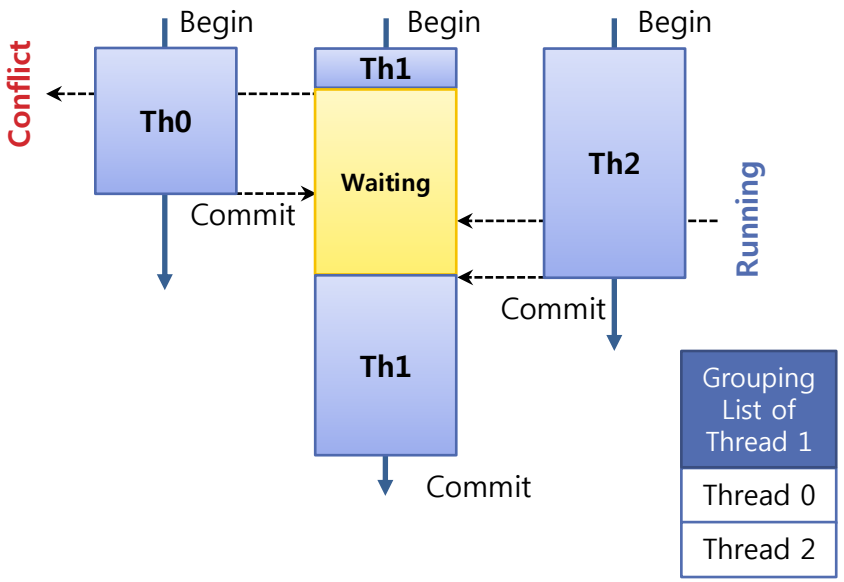
Contention Management

➢ Reducing abort penalty

- Conflicted transactions control



(a) Conventional Back-off based Contention Management



(b) Concept of Conflict Avoidance Scheduling

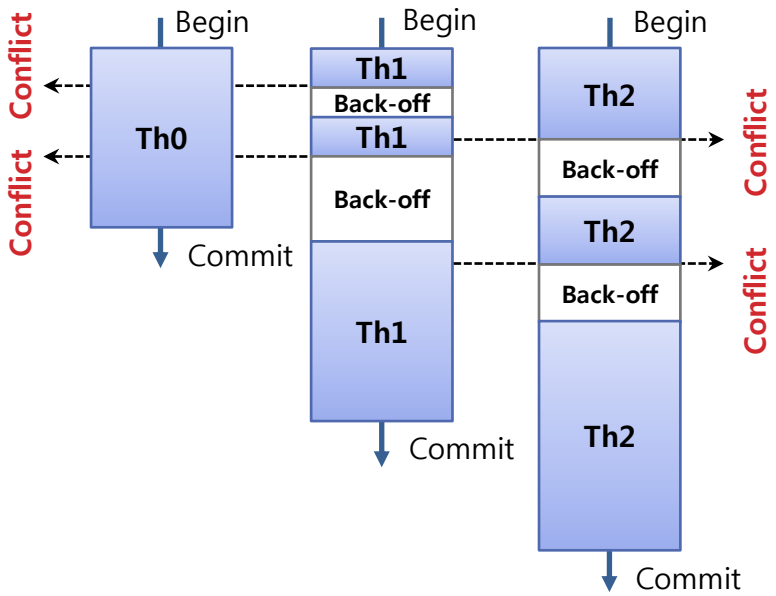
➢ Scheduling using grouping list

- Grouping list: history of the conflicted threads
- Conflict avoidance using the relationship among the threads

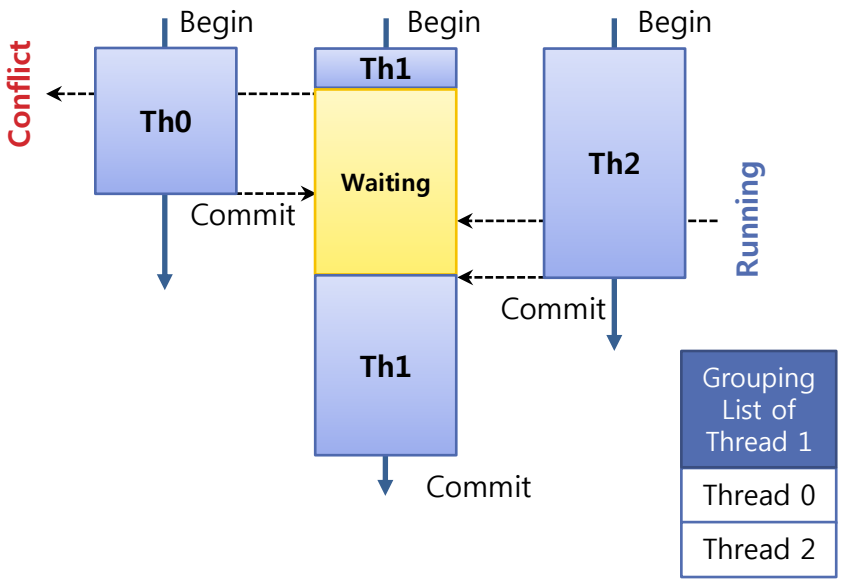
Contention Management

➢ Reducing abort penalty

- Conflicted transactions control



(a) Conventional Back-off based Contention Management



(b) Concept of Conflict Avoidance Scheduling

➢ Scheduling using grouping list

- Grouping list: history of the conflicted threads
- Conflict avoidance using the relationship among the threads

Prior Work

➢ Design dimensions of TM systems

- Conflict detection, version management, and contention management
  - → diverse approaches: hardware, software, or their hybrid
  - → various designs for the performance improvement: FlexTM, EazyHTM, FASTM …
- Transaction scheduling
  - → not oriented to reactive contention management

➢ Adaptive Transaction Scheduling (ATS), Yoo et al., 2008

- Measuring "contention intensity"
- Serialized execution of all transaction in a single queue
  - → over serialization

➢ Proactive Transaction Scheduling (PTS), Blake et al., 2009

- Threads swapping based on the prediction of conflicts
- Software and hardware overhead

Outline

- Related Work
  - Contention management
  - Prior work
- **Conflict Avoidance Scheduling Mechanism**
  - Scheduling operation
  - Implementation
- **Results and Analysis**
  - Execution time analysis
  - Contention analysis
- **Conclusion**

Scheduling Approach

➢ Approach

- Concurrent execution control
- Reducing the amount of contention

➢ Grouping list

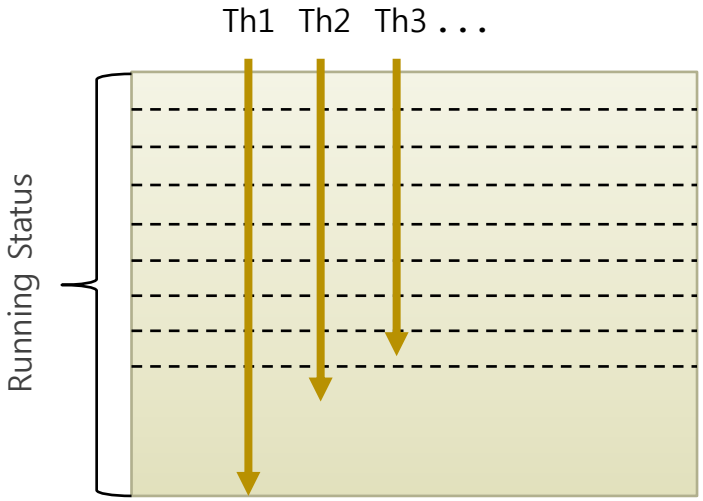
- Same group: threads that have high possibility of the conflict
- Dedicated grouping list for each threads
- List information
  → the ID of the threads in the same group and the running status of the threads
- List makeup
  → measuring **the number of repeated conflicts**
  → compared to the predefined threshold
- Repeatedly conflicts
  → additional conflicts in the future

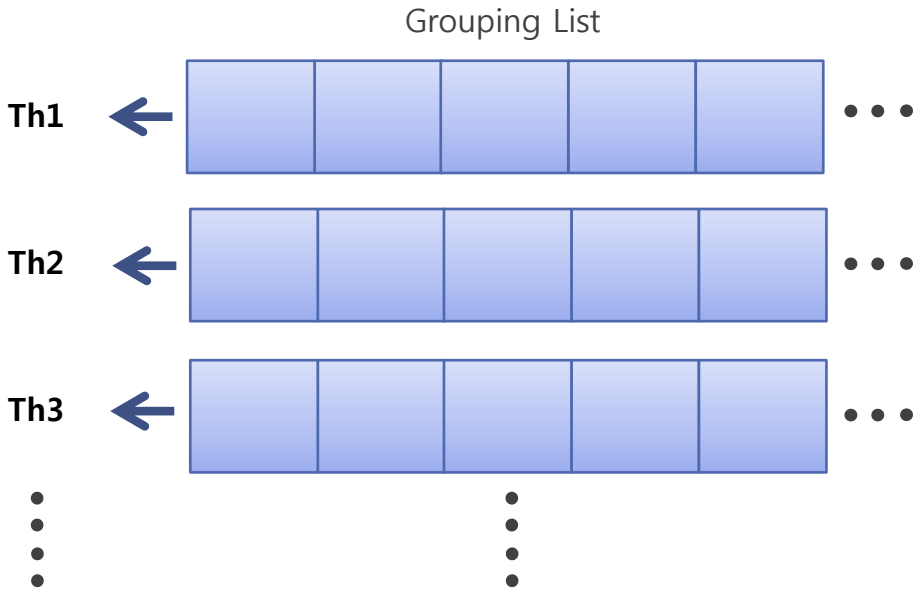
Conflict Avoidance Scheduling

➢ Scheduling operation

- Access on the grouping list for every transactional begin, restart, abort, and commit

Initial status



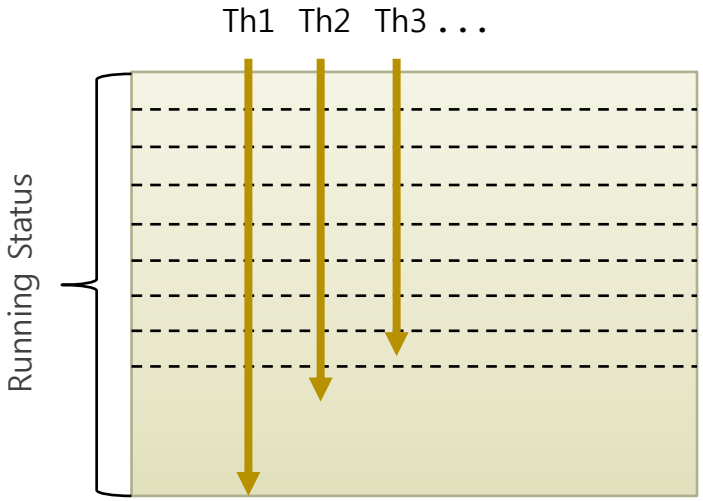


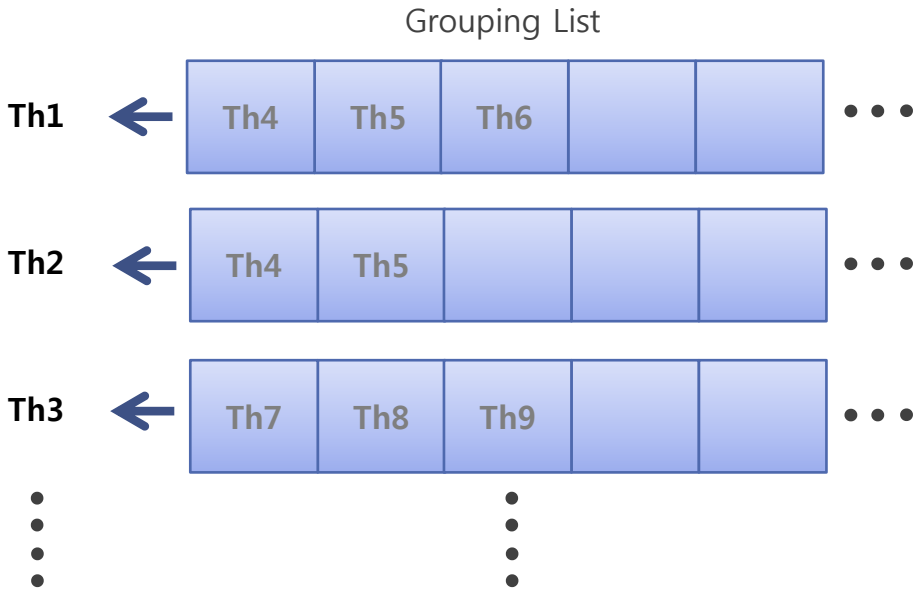
Conflict Avoidance Scheduling

➢ Scheduling operation

- Access on the grouping list for every transactional begin, restart, abort, and commit

Conflicts

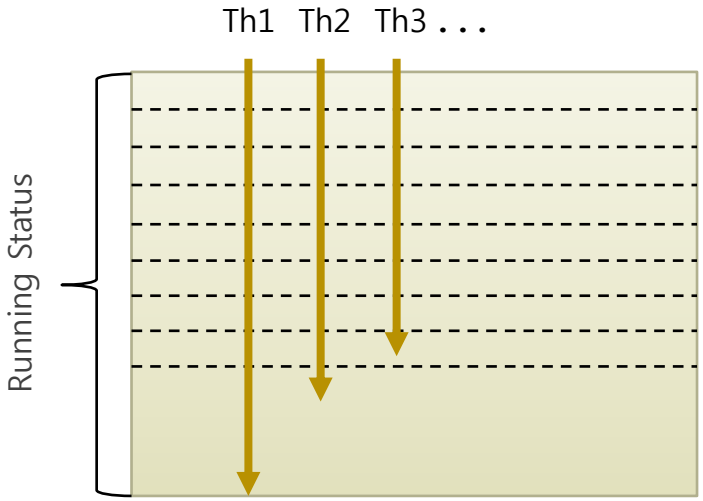


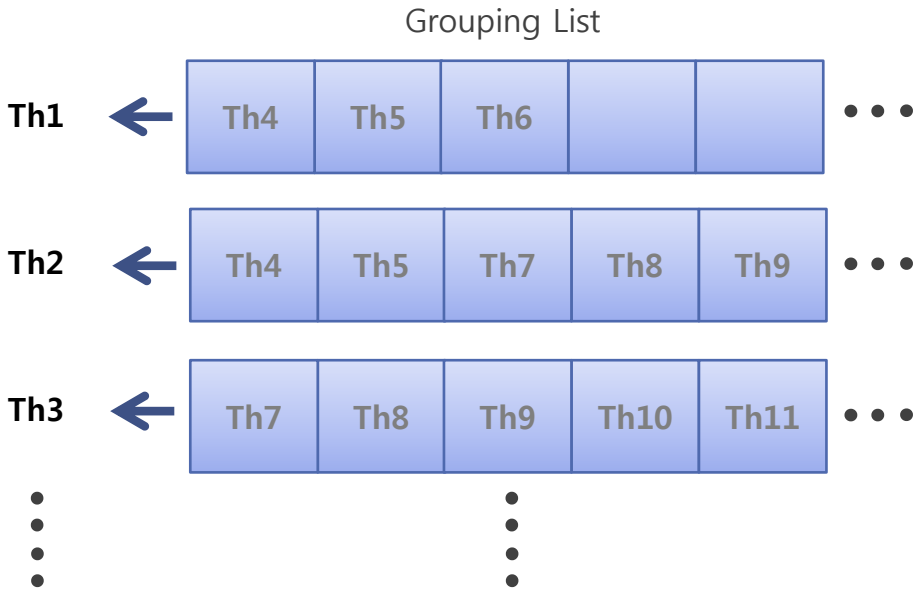


Conflict Avoidance Scheduling

➢ Scheduling operation

- Access on the grouping list for every transactional begin, restart, abort, and commit



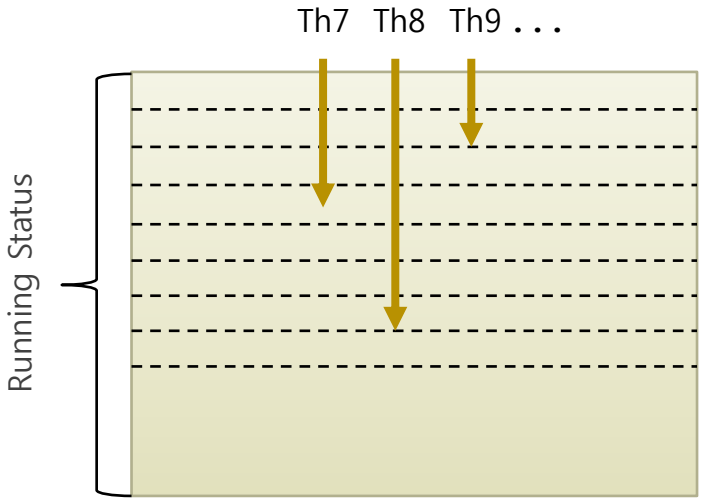


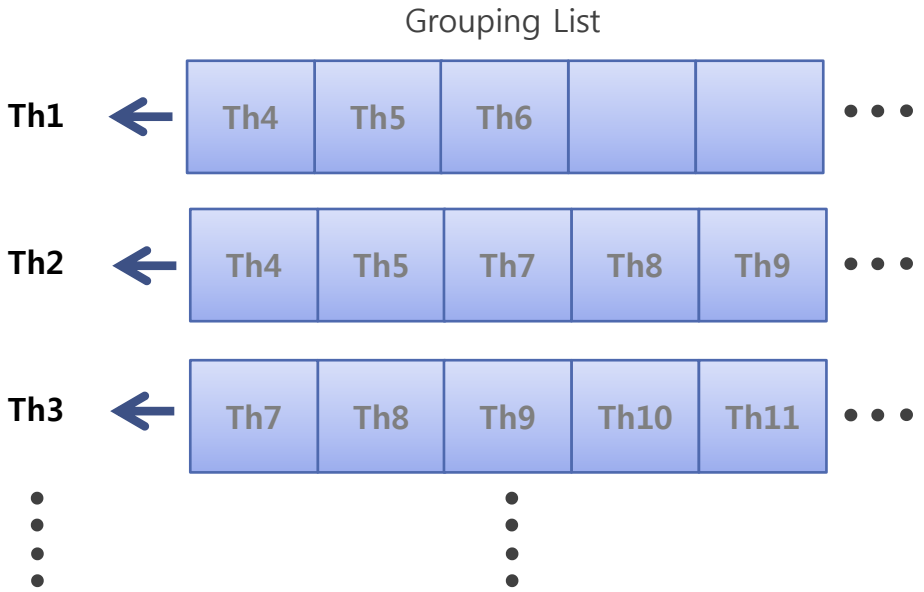
Conflict Avoidance Scheduling

➢ Scheduling operation

- Access on the grouping list for every transactional begin, restart, abort, and commit

Completion of thread 1 to 3



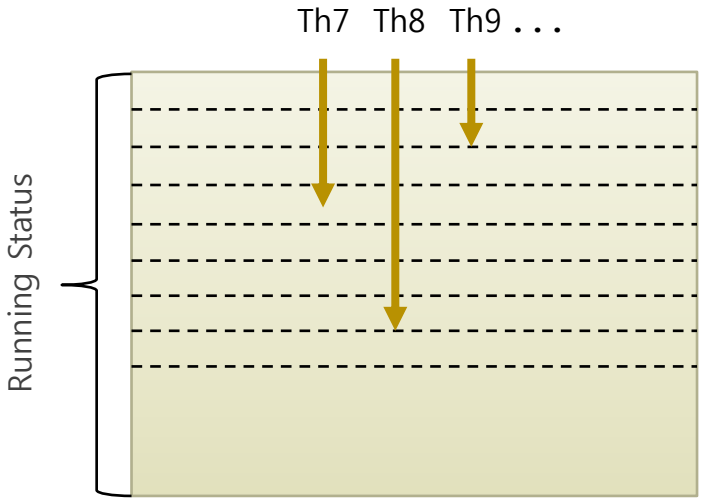


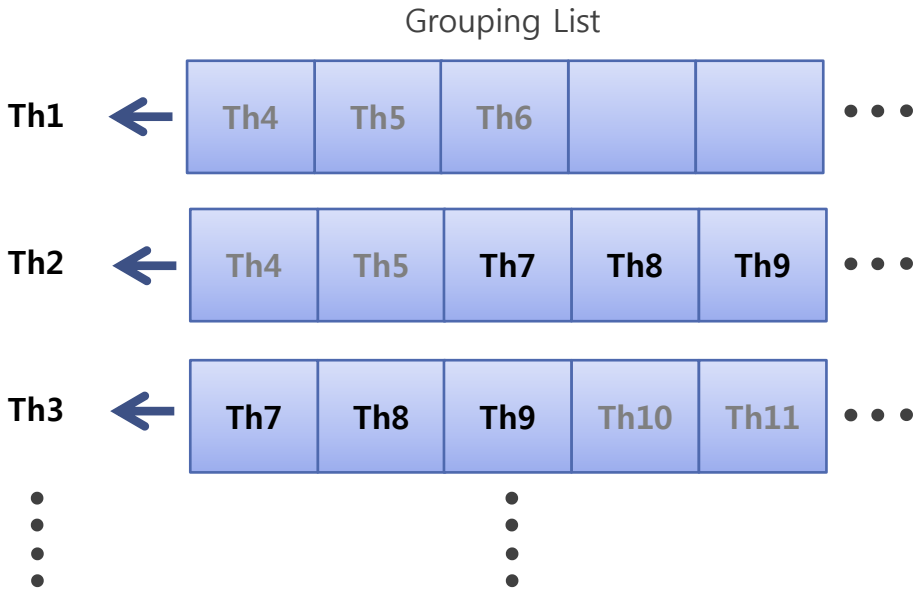
Conflict Avoidance Scheduling

➢ Scheduling operation

- Access on the grouping list for every transactional begin, restart, abort, and commit

Status change



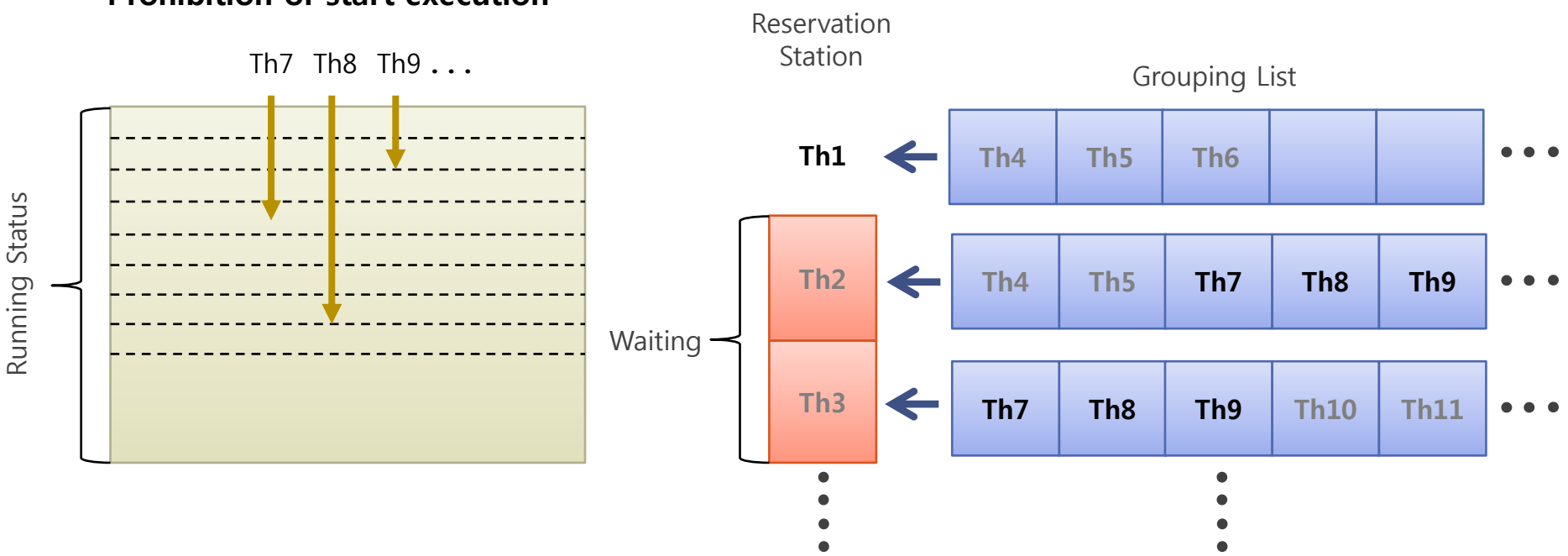


Conflict Avoidance Scheduling

➢ Scheduling operation

- Access on the grouping list for every transactional begin, restart, abort, and commit

Prohibition of start execution

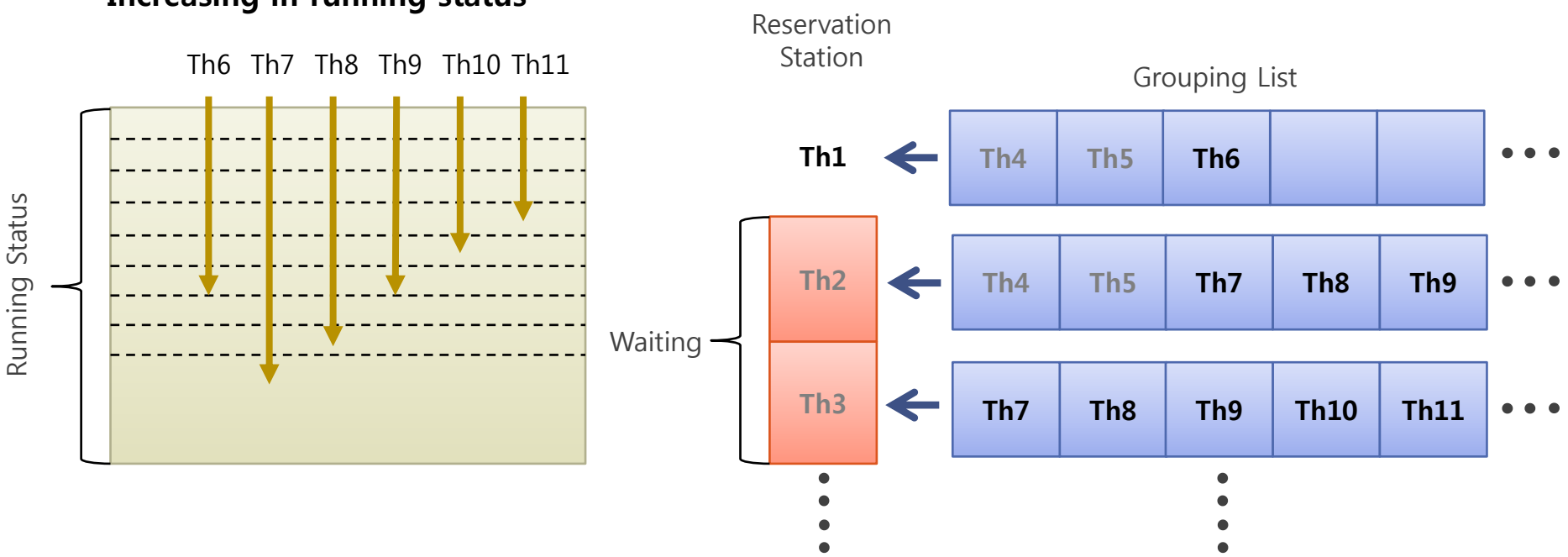


Conflict Avoidance Scheduling

➢ Scheduling operation

- Access on the grouping list for every transactional begin, restart, abort, and commit

Increasing in running status

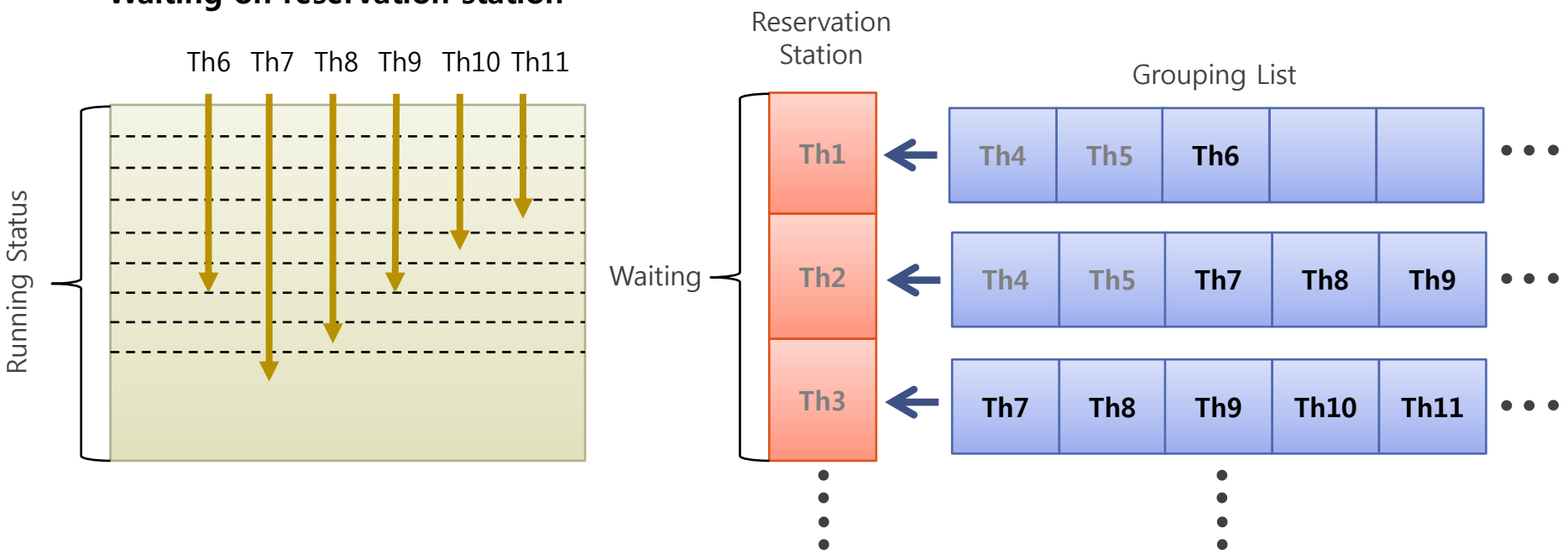


Conflict Avoidance Scheduling

➢ Scheduling operation

- Access on the grouping list for every transactional begin, restart, abort, and commit

Waiting on reservation station

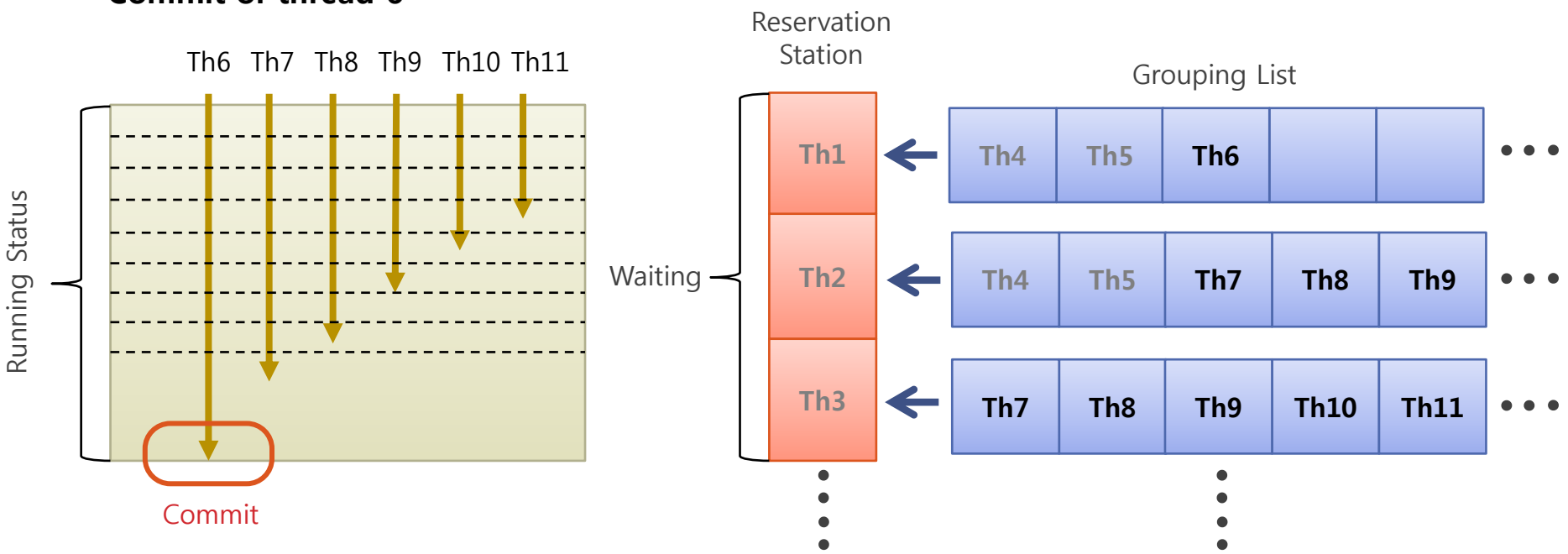


Conflict Avoidance Scheduling

➢ Scheduling operation

- Access on the grouping list for every transactional begin, restart, abort, and commit

Commit of thread 6

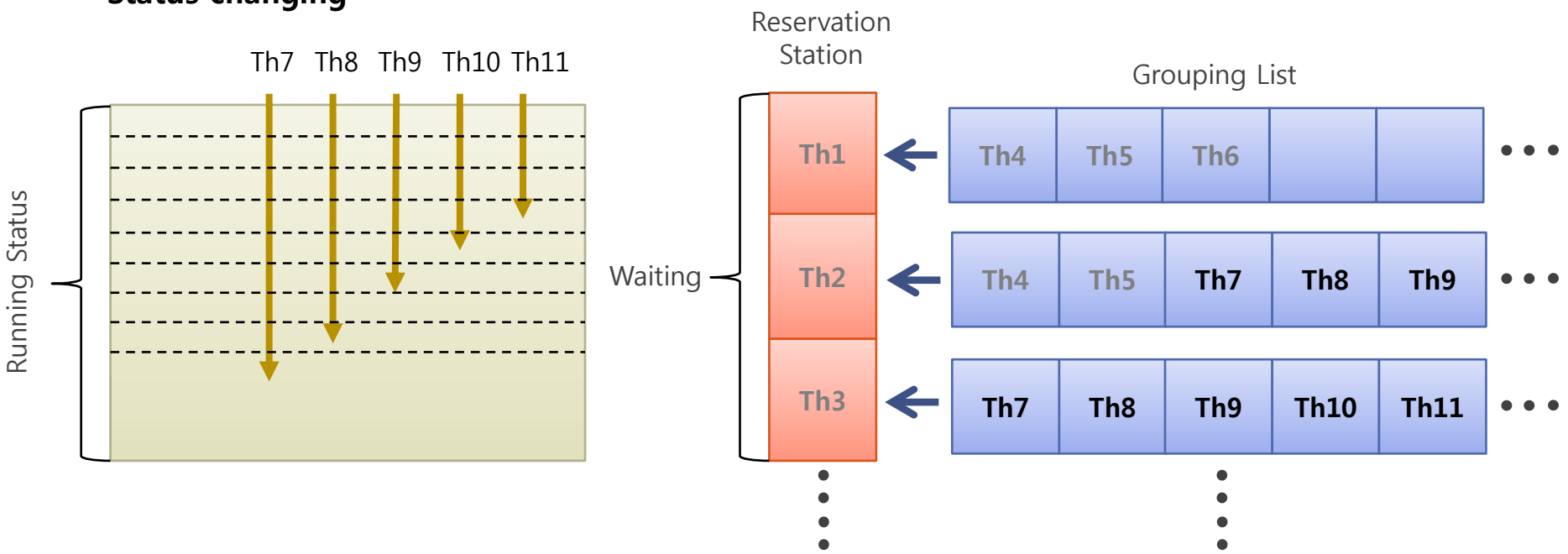


Conflict Avoidance Scheduling

➢ Scheduling operation

- Access on the grouping list for every transactional begin, restart, abort, and commit

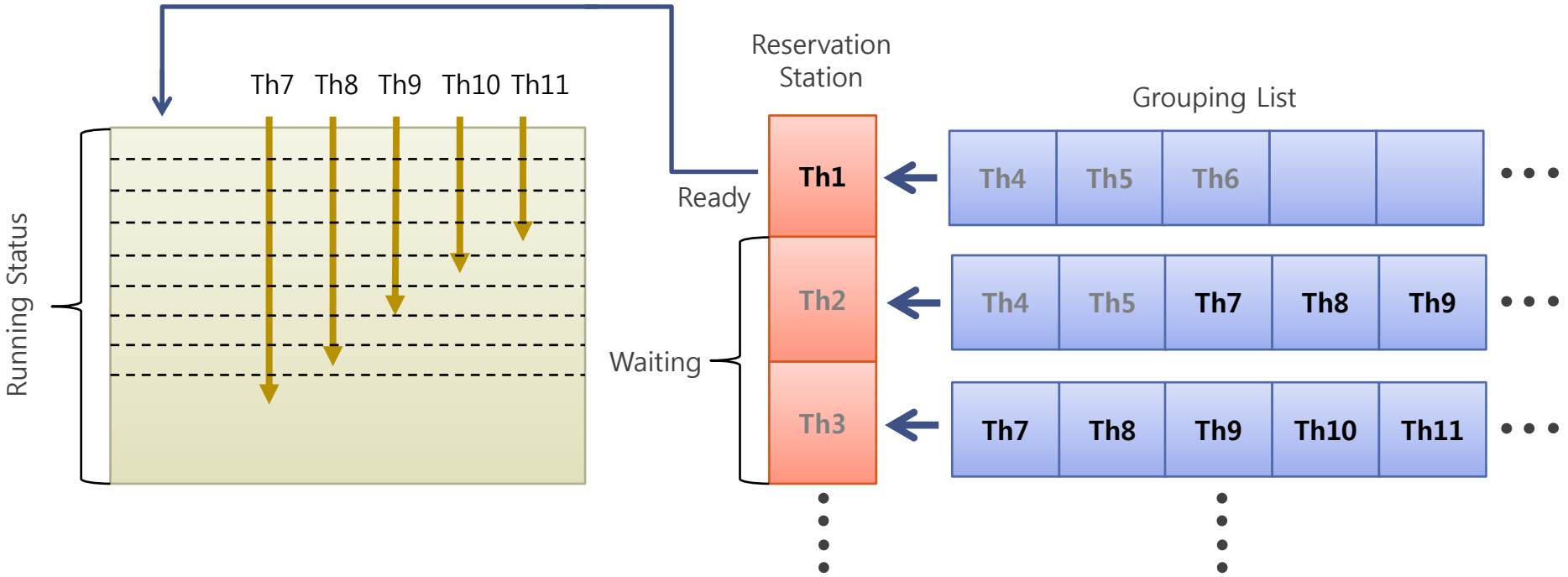
Status changing



Conflict Avoidance Scheduling

➢ Scheduling operation

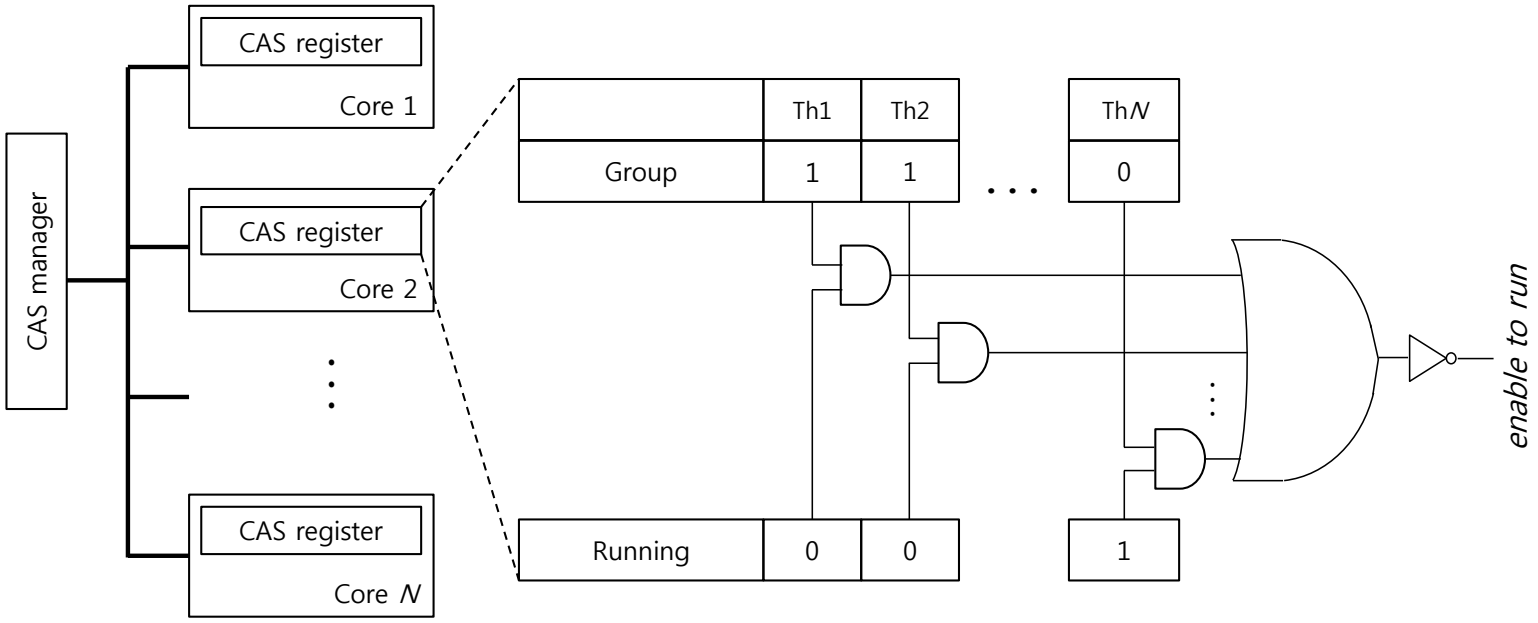
- Access on the grouping list for every transactional begin, restart, abort, and commit



Implementation

➢ CAS hardware architecture

- Centralized CAS manager and distributed CAS registers
- CAS register → grouping list
- Management of the list → CAS manager



- Set 1 *enable to run* signal → 0 for all threads in the same group

Outline

- Related Work
  - Contention management
  - Prior work
- Conflict Avoidance Scheduling Mechanism
  - Scheduling operation
  - Implementation
- **Results and Analysis**
  - Performance improvement
  - Execution time and contention analysis
- **Conclusion**

Simulation Environment

➢ Simulator and TM system

- **Simics**, a full system simulation platform (set as 16-core CMP)
- **GEMS** (General Execution-driven Multiprocessor Simulator) from Wisconsin's Multifacet Project

  → The baseline TM system: **LogTM-SE** of GEMS

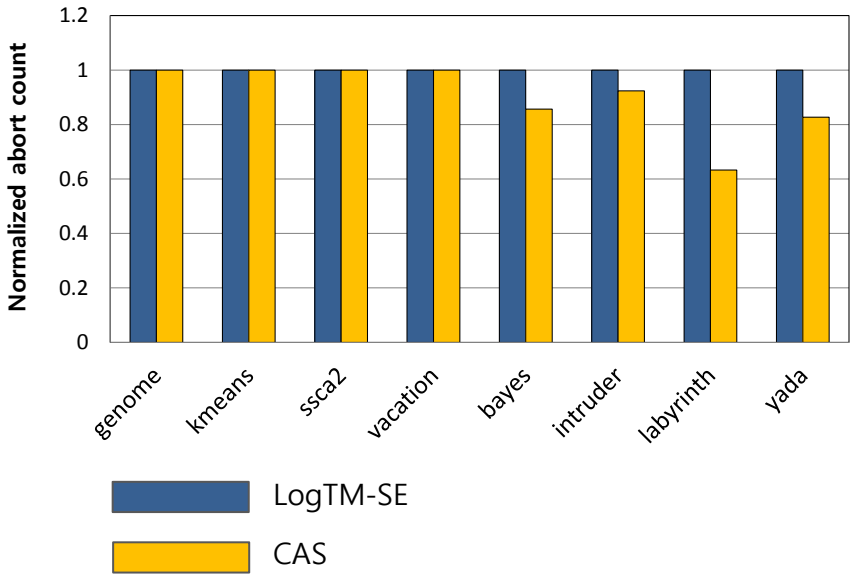
➢ Benchmark suite

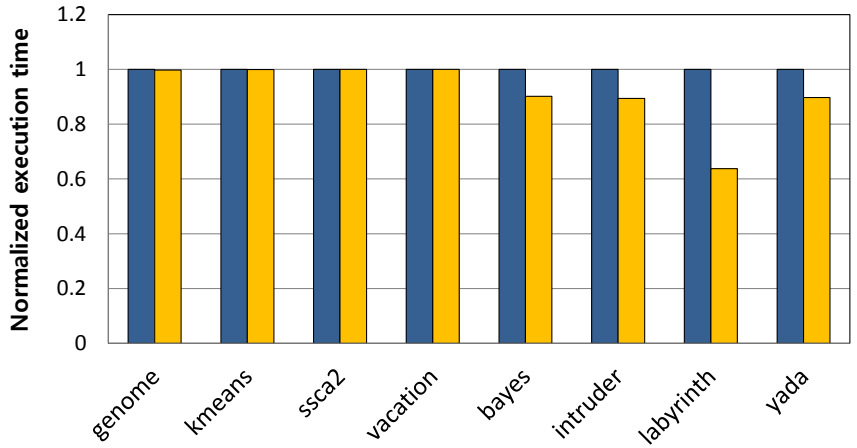
- **STAMP** (Stanford Transactional Applications for Multi-Processing)
- Large transactions with high contention than traditional programs

Application	Input Parameters	Contention
genome	-t15 -g256 -s16 -n16384	Low
kmeans	-p15 -m15 -n15 -t0.05 -i random-n2048-d16-c16	Low
ssca2	-t15 -s13 -i1.0 -u1.0 -l3 -p3	Low
vacation	-c15 -n8 -q10 -u80 -r65536 -t4096	Medium
bayes	-t15 -v32 -r1024 -n2 -p20 -i2 -e2	High
intruder	-t15 -a10 -l16 -n4096 -s1	High
labyrinth	-t15 -i random-x16-y16-z3-n16	High
yada	-t15 -a20 -i 633.2	Medium

Performance Improvement

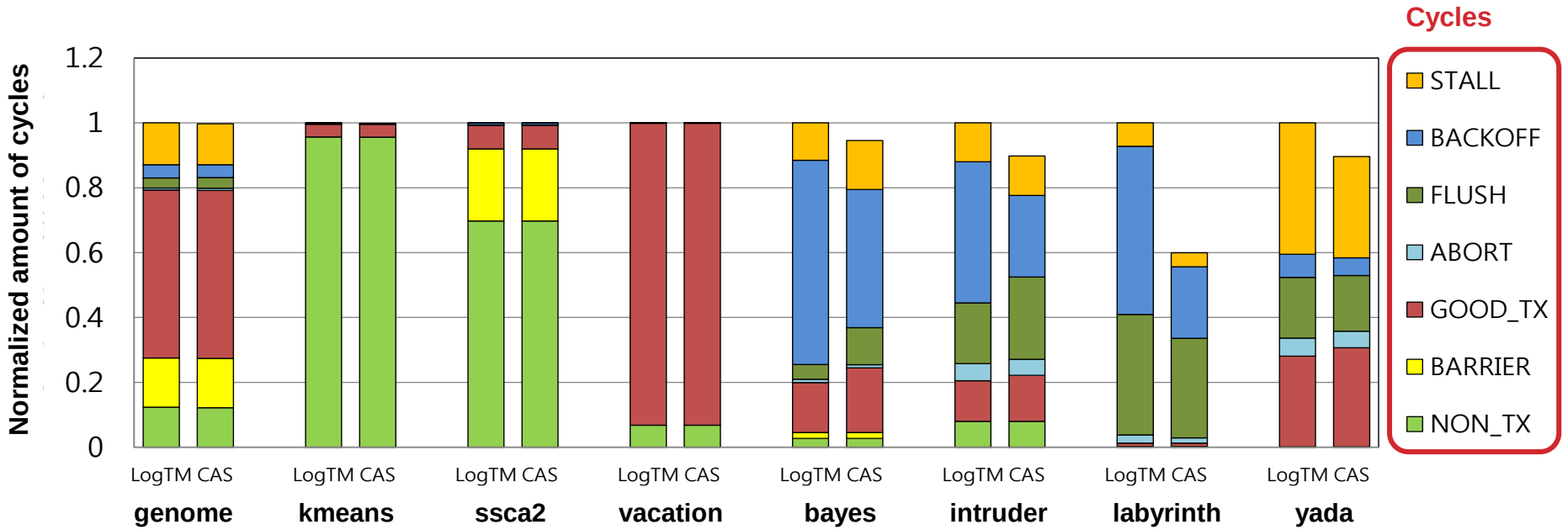
- **Reduced number of aborts and execution time**
  - Compared to the baseline system
  - Effectiveness of the proposed method
    - → in the contention medium/high applications
  - No harm in contention-low applications
  - 23% of average improvement for eight applications





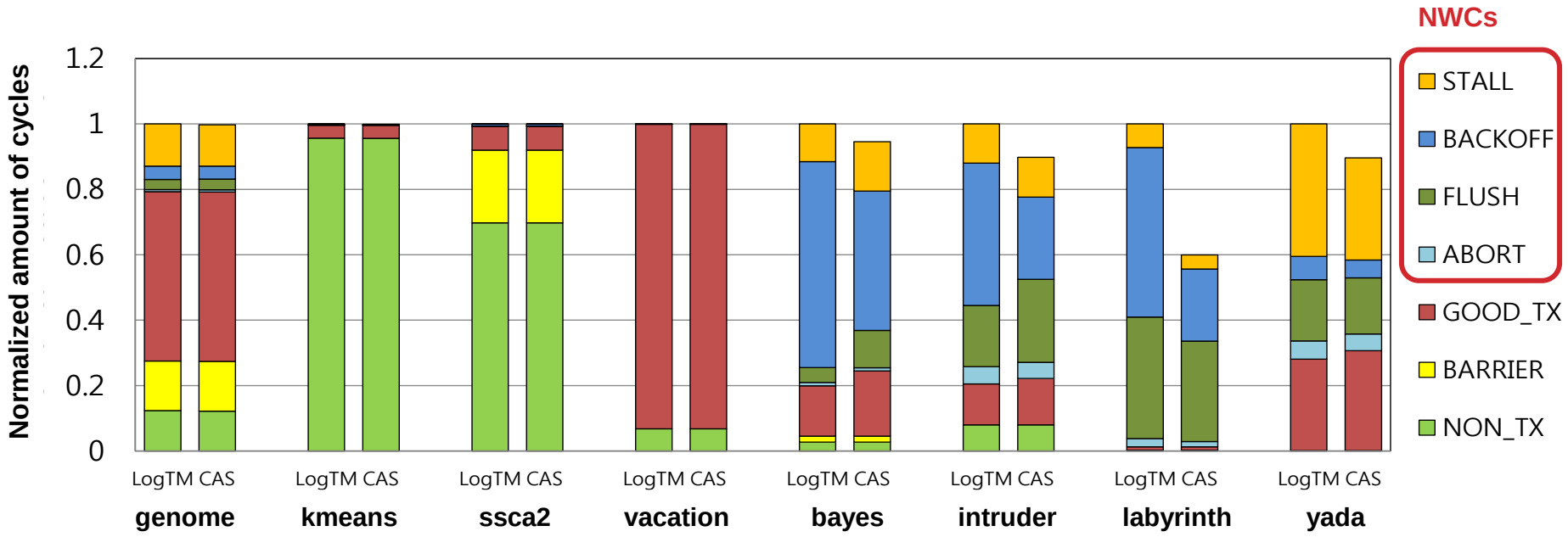
Execution Cycles Analysis

- **Cycle analysis during the parallel phase of each program**
  - Metric: seven different cycles
    - → Parallel processing using TM systems



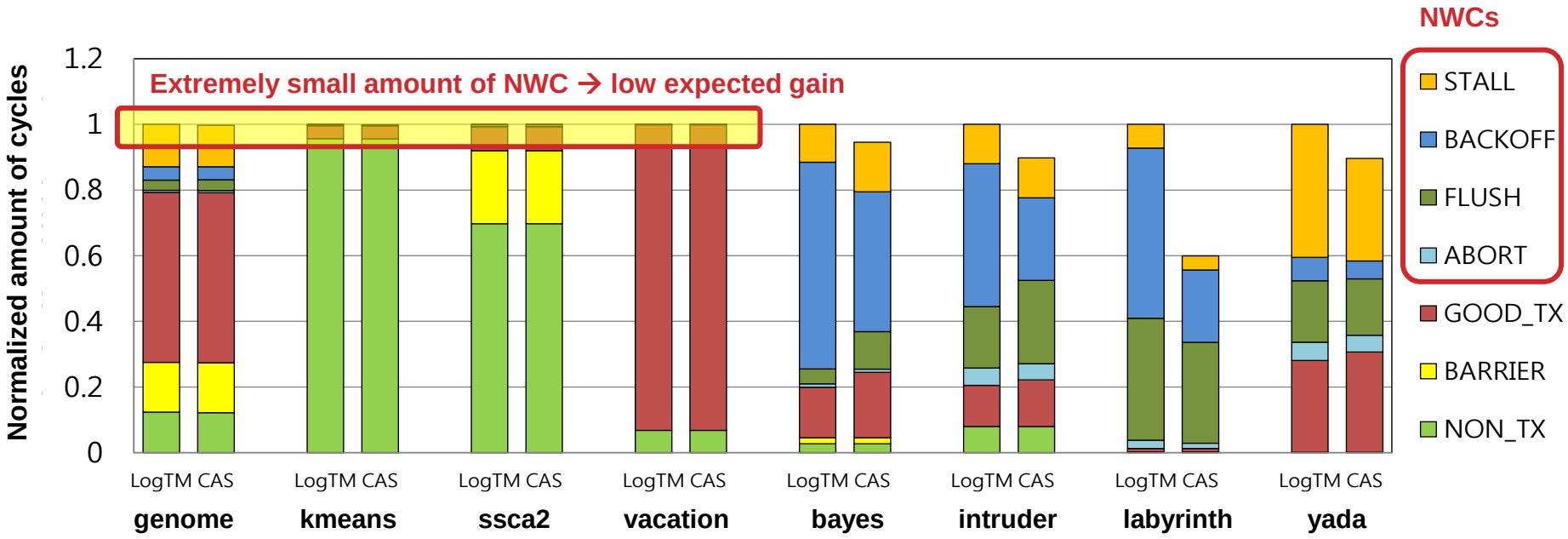
Execution Cycles Analysis

- **Cycle analysis during the parallel phase of each program**
  - Necessary Wasted Cycles (NWCs)
    - → sum of the cycles used for stall, back-off, flush, and abort
  - Factors of improving the performance
    - → the amount of NWCs, contention



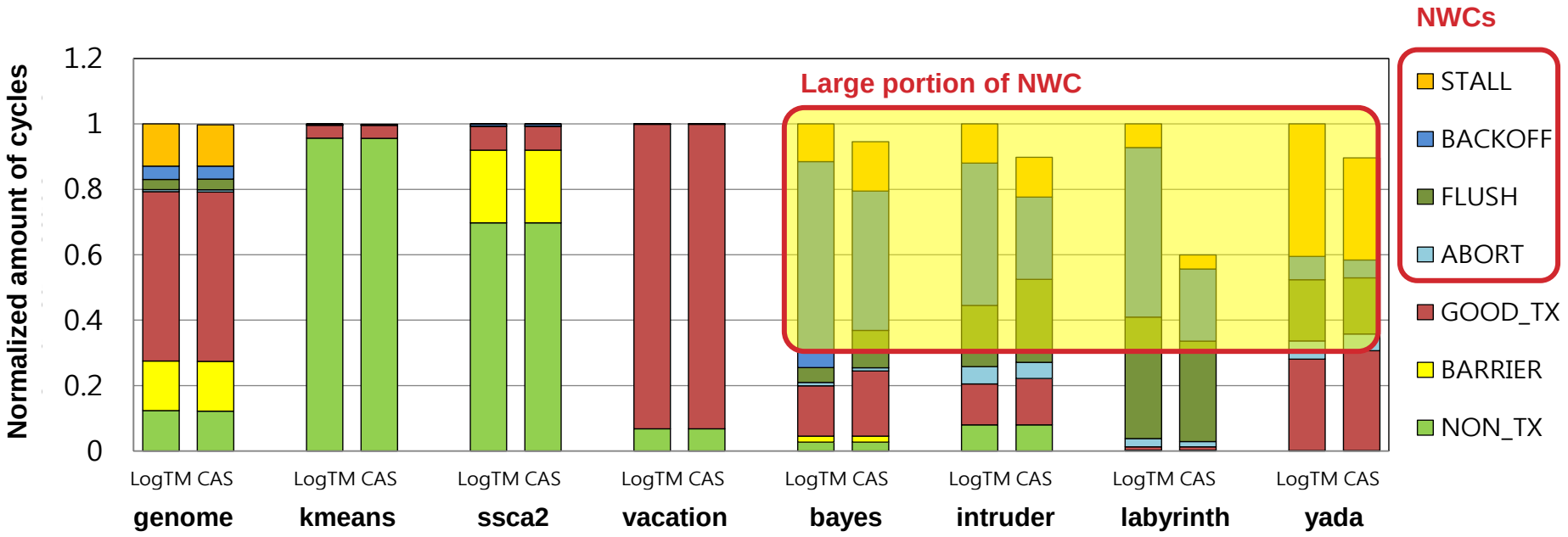
Execution Cycles Analysis

- **Cycle analysis during the parallel phase of each program**
  - Necessary Wasted Cycles (NWCs)
    - → sum of the cycles used for stall, back-off, flush, and abort
  - Factors of improving the performance
    - → the amount of NWCs, contention



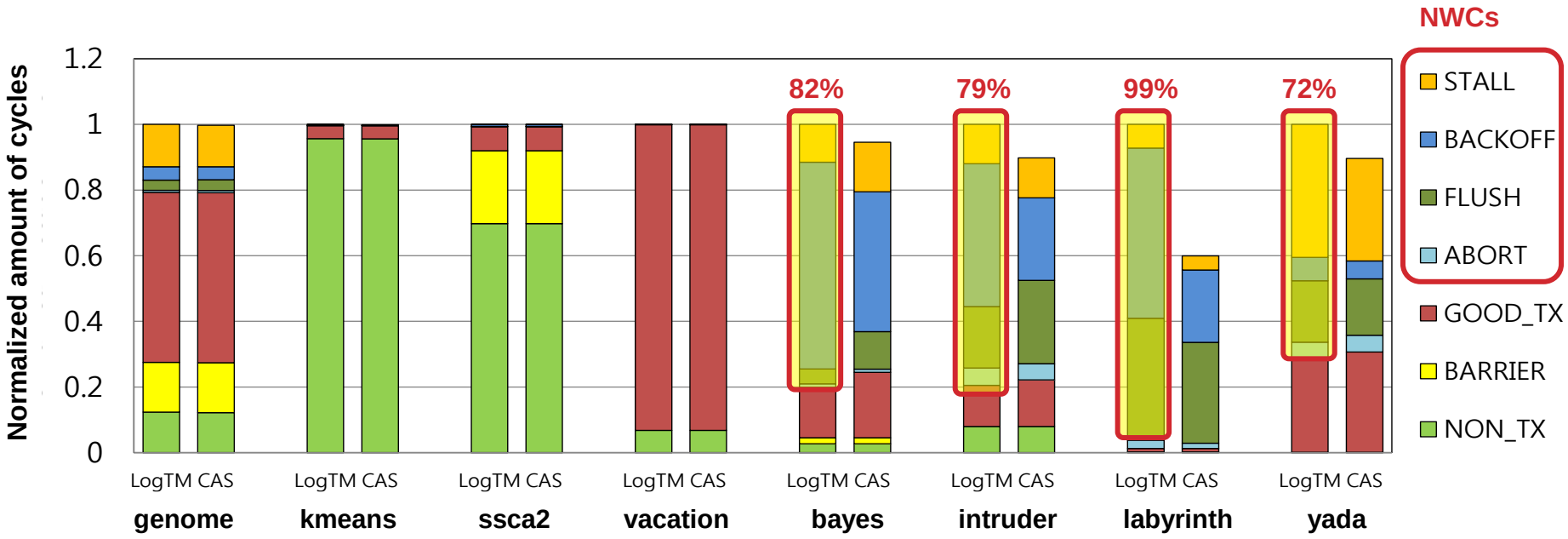
Execution Cycles Analysis

- **Cycle analysis during the parallel phase of each program**
  - Necessary Wasted Cycles (NWCs)
    - → sum of the cycles used for stall, back-off, flush, and abort
  - Factors of improving the performance
    - → the amount of NWCs, contention



Execution Cycles Analysis

- **Cycle analysis during the parallel phase of each program**
  - Necessary Wasted Cycles (NWCs)
    - → sum of the cycles used for stall, back-off, flush, and abort
  - Factors of improving the performance
    - → the amount of NWCs, contention



Contention Analysis

➢ **Prohibiting of concurrent threads execution of the same group**

	bayes		intruder		labyrinth		yada	
	rate	results	rate	results	rate	results	rate	results
Th1	Th2		Th15		Th15		Th7	
	20.73%	0.647	9.00%	0.767	51.61%	0.344	15.44%	1
Th2	Th6		Th15		Th15		Th12	
	24.28%	0.235	8.70%	0.84	49.18%	0.367	14.59%	0.439
Th3	Th10		Th15		Th15		Th12	
	41.67%	0.55	8.28%	0.709	45.20%	0.333	14.33%	0.425
Th4	Th3		Th15		Th15		Th2	
	33.33%	0.5	9.61%	0.682	56.41%	0.5	16.30%	0.307
Th5	Th2		Th15		Th15		Th11	
	27.16%	0.545	8.89%	0.766	50.00%	0.379	13.88%	0.667
Th6	Th10		Th15		Th15		Th13	
	21.86%	0.285	8.65%	0.764	48.78%	0.55	12.13%	0.482
Th7	Th9		Th15		Th15		Th14	
	25.30%	1.142	8.73%	0.769	43.63%	0.458	13.78%	0.436
Th8	Th2		Th15		Th15		Th15	
	25.84%	0.521	8.62%	0.765	45.83%	0.5	66.66%	2
Th9	Th2		Th15		Th15		Th7	
	21.26%	0.481	8.82%	0.674	47.05%	0.458	15.59%	0.382
Th10	Th9		Th15		Th15		Th2	
	46.37%	0.375	9.03%	0.777	76.92%	0.55	13.97%	0.579
Th11	Th2		Th15		Th15		Th2	
	25.97%	0.65	9.55%	0.745	50.74%	0.324	12.33%	0.324
Th12	Th10		Th15		Th15		Th2	
	27.96%	0.091	9.03%	0.723	100.00%	0.526	14.47%	0.295
Th13	Th2		Th15		Th15		Th4	
	38.18%	0.619	9.54%	0.686	52.27%	0.478	14.78%	0.262
Th14	Th2		Th8		Th15		Th2	
	52.23%	0.257	7.86%	0.922	62.50%	0.55	18.45%	0.186
Th15	Th10		Th3		-		Th12	
	39.76%	0.307	8.66%	0.705	-	-	19.71%	0.214

Contention Analysis

- **Prohibiting of concurrent threads execution of the same group**

	bayes		intruder		labyrinth		yada	
	rate	results	rate	results	rate	results	rate	results
Th1	Th2		Th15		Th15		Th7	
	20.73%	0.647	9.00%	0.767	51.61%	0.344	15.44%	1
Th2	Th6		Th15		Th15		Th12	
	24.28%	0.235	8.70%	0.84	49.18%	0.367	14.59%	0.439
Th3	Th10		Th15		Th15		Th12	
	41.67%		8.28%	0.709	45.20%	0.333	14.33%	0.425
Th4	Th3		Th15		Th15		Th2	
	33.33%	0.5	9.61%	0.682	56.41%	0.5	16.30%	0.307
Th5	Th2		Th15		Th15		Th11	
	27.16%	0.545	8.89%	0.766	50.00%	0.379	13.88%	0.667
Th6	Th10						Th13	
	21.86%	0.					12.13%	0.482
Th7	Th9						Th14	
	25.30%	1.					13.78%	0.436
Th8	Th2						Th15	
	25.84%	0.					66.66%	2
Th9	Th2		Th15		Th15		Th7	
	21.26%	0.481	8.82%	0.674	47.05%	0.458	15.59%	0.382
Th10	Th9		Th15		Th15		Th2	
	46.37%	0.375	9.03%	0.777	76.92%	0.55	13.97%	0.579
Th11	Th2		Th15		Th15		Th2	
	25.97%	0.65	9.55%	0.745	50.74%	0.324	12.33%	0.324
Th12	Th10		Th15		Th15		Th2	
	27.96%	0.091	9.03%	0.723	100.00%	0.526	14.47%	0.295
Th13	Th2		Th15		Th15		Th4	
	38.18%	0.619	9.54%	0.686	52.27%	0.478	14.78%	0.262
Th14	Th2		Th8		Th15		Th2	
	52.23%	0.257	7.86%	0.922	62.50%	0.55	18.45%	0.186
Th15	Th10		Th3		-		Th12	
	39.76%	0.307	8.66%	0.705	-	-	19.71%	0.214

largest amount of conflict

0.647 times decrease

Th1	Th2	
	20.73%	0.647

Contention Analysis

➢ **Prohibiting of concurrent threads execution of the same group**

	bayes		intruder		labyrinth		yada	
	rate	results	rate	results	rate	results	rate	results
Th1	Th2		Th15		Th15		Th7	
	20.73%	0.647	9.00%	0.767	51.61%	0.344	15.44%	1
Th2	Th6		Th15		Th15		Th12	
	24.28%	0.235	8.70%	0.84	49.18%	0.367	14.59%	0.439
Th3	Th10		Th15		Th15		Th12	
	41.67%	0.55	8.28%	0.709	45.20%	0.333	14.33%	0.425
Th4	Th3		Th15		Th15		Th2	
	33.33%	0.5	9.61%	0.682	56.41%	0.5	16.30%	0.307
Th5	Th2		Th15		Th15		Th11	
	27.16%	0.545	8.89%	0.766	50.00%	0.379	13.88%	0.667
Th6	Th10		Th15		Th15		Th13	
	21.86%	0.285	8.65%	0.764	48.78%	0.55	12.13%	0.482
Th7	Th9		Th15		Th15		Th14	
	25.30%	1.142	8.73%	0.769	43.63%	0.458	13.78%	0.436
Th8	Th2		Th15		Th15		Th15	
	25.84%	0.521	8.62%	0.765	45.83%	0.5	66.66%	2
Th9	Th2							
	21.26%	0.4	8.					0.382
Th10	Th9							
	46.37%	0.375	9.					0.579
Th11	Th2							
	25.97%	0.65	9.55%	0.745	50.74%	0.524	12.55%	0.324
Th12	Th10		Th15		Th15		Th2	
	27.96%	0.091	9.03%	0.723	100.00%	0.526	14.47%	0.295
Th13	Th2		Th15		Th15		Th4	
	38.18%	0.619	9.54%	0.686	52.27%	0.478	14.78%	0.262
Th14	Th2		Th8		Th15		Th2	
	52.23%	0.257	7.86%	0.922	62.50%	0.55	18.45%	0.186
Th15	Th10		Th3		-		Th12	
	39.76%	0.307	8.66%	0.705	-	-	19.71%	0.214

Th7	Th9	
	25.30%	1.142

Contention Analysis

➢ **Prohibiting of concurrent threads execution of the same group**

	bayes		intruder		labyrinth		yada	
	rate	results	rate	results	rate	results	rate	results
Th1	Th2		Th15		Th15		Th7	
	20.73%	0.647	9.00%	0.767	51.61%	0.344	15.44%	1
Th2	Th6		Th15		Th15		Th12	
	24.28%	0.235	8.70%	0.84	49.18%	0.367	14.59%	0.439
Th3	Th10		Th15		Th15		Th12	
	41.67%	0.55	8.28%	0.709	45.20%	0.333	14.33%	0.425
Th4	Th3		Th15		Th15		Th2	
	33.33%	0.5	9.61%	0.682	56.41%	0.5	16.30%	0.307
Th5	Th2		Th15		Th15		Th11	
	27.16%	0.545	8.89%	0.766	50.00%	0.379	13.88%	0.667
Th6	Th10		Th15		Th15		Th13	
	21.86%	0.285	8.65%	0.764	48.78%	0.55	12.13%	0.482
Th7	Th9		Th15		Th15		Th14	
	25.30%	1.142	8.73%	0.769	43.63%	0.458	13.78%	0.436
Th8	Th2		Th15		Th15		Th15	
	25.84%	0.521	8.62%	0.765	45.83%	0.5	66.66%	2
Th9	Th2		Th15		Th15		Th7	
	21.26%	0.481	8.82%	0.674	47.05%	0.458	15.59%	0.382
Th10	Th9		Th15		Th15		Th2	
	46.37%	0.375	9.03%	0.777	76.92%	0.55	13.97%	0.579
Th11	Th2		Th15		Th15		Th2	
	25.97%	0.65	9.55%	0.745	50.74%	0.324	12.33%	0.324
Th12	Th10		Th15		Th15		Th2	
	27.96%	0.091	9.03%	0.723	100.00%	0.526	14.47%	0.295
Th13	Th2		Th15		Th15		Th4	
	38.18%	0.619	9.54%	0.686	52.27%	0.478	14.78%	0.262
Th14	Th2		Th8		Th15		Th2	
	52.23%	0.257	7.86%	0.922	62.50%	0.55	18.45%	0.186
Th15	Th10		Th3		-		Th12	
	39.76%	0.307	8.66%	0.705	-	-	19.71%	0.214

Outline

- Related Work
  - Contention management
  - Prior work
- Conflict Avoidance Scheduling Mechanism
  - Scheduling operation
  - Implementation
- Results and Analysis
  - Execution time analysis
  - Contention analysis
- **Conclusion**

Conclusion

➢ A new contention management policy

- Providing fast program execution by reducing the abort penalty
  → 23% of average performance improvement
- Proposal of the grouping list for the contention management
  → analysis based on the thread relationship

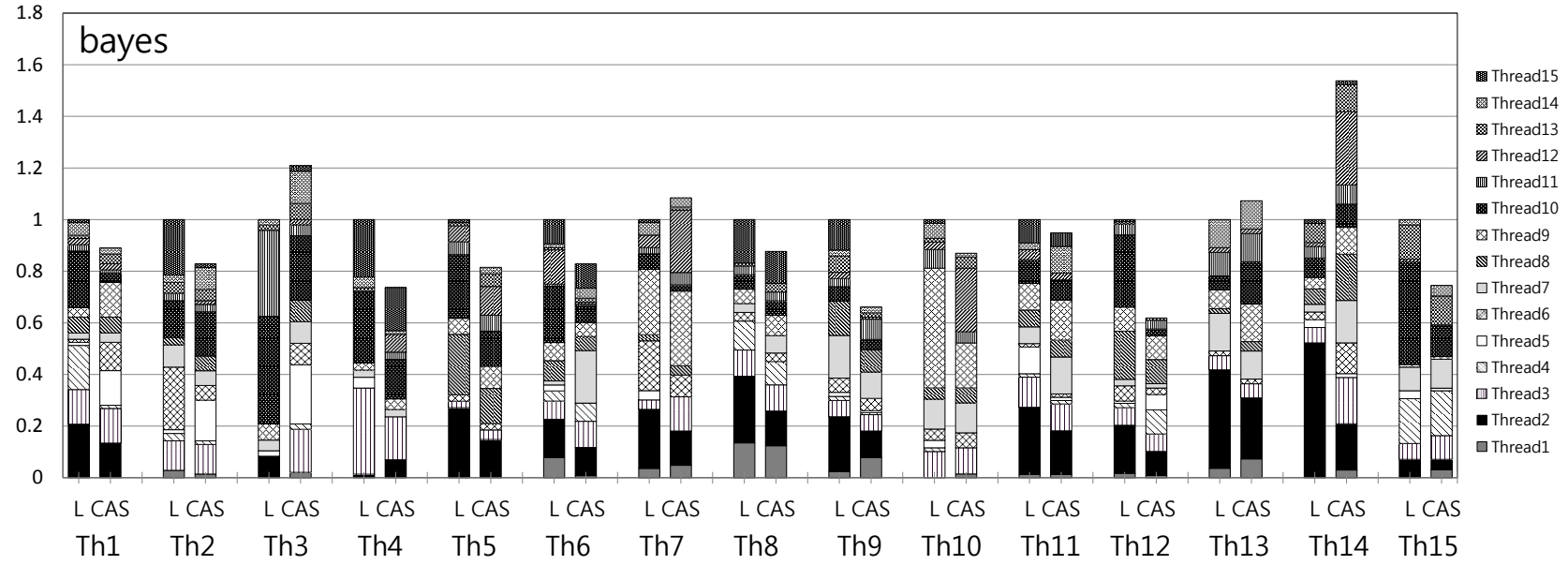
➢ Future work

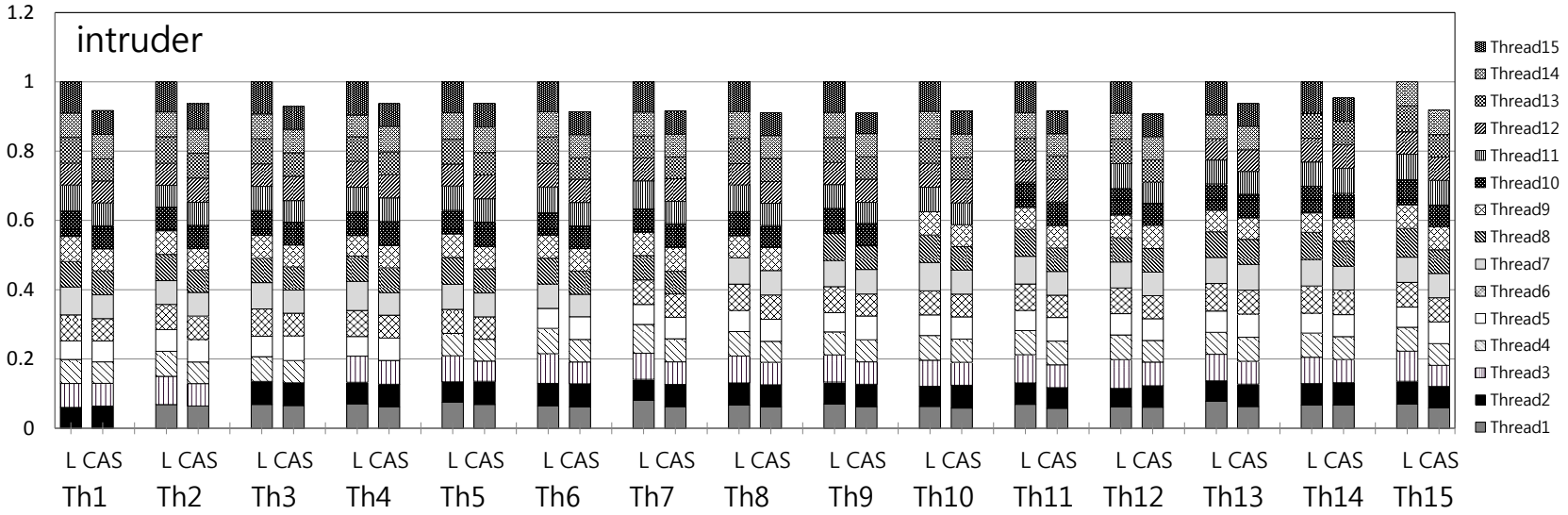
- Measuring overhead of the scheduling

Thank you!

- **Questions, comments:**
  - kseunghun@gmail.com
  - http://escal.yonsei.ac.kr

Supplement-1





Supplement-2