Static Scheduling of Parallel Tasks

- A Quick Review -

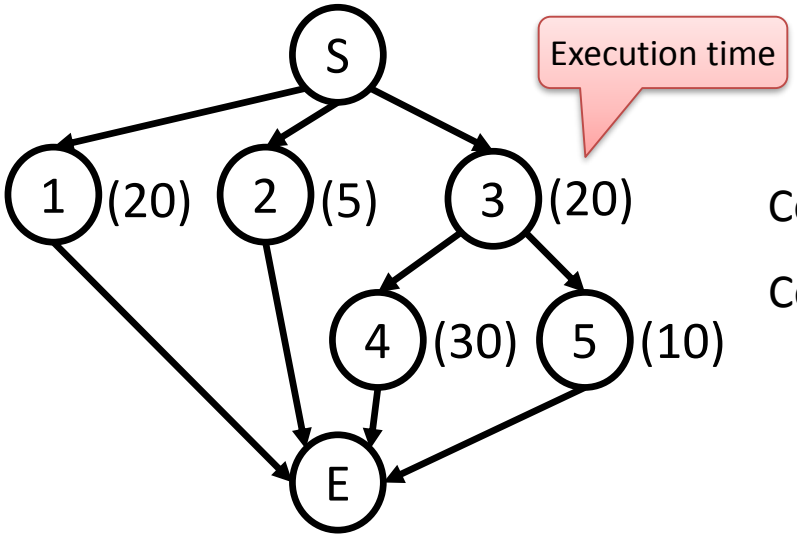
Hiroyuki Tomiyama

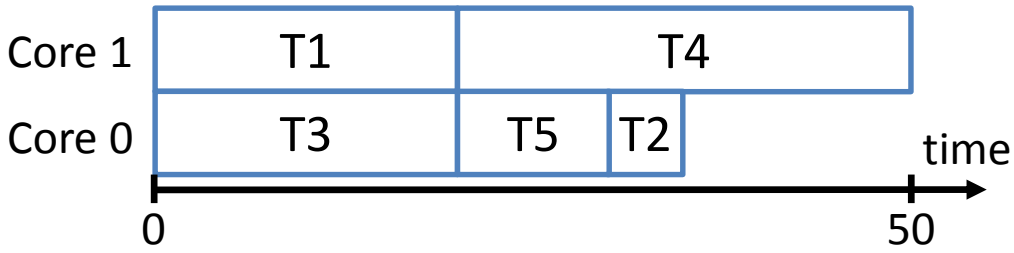
Ritsumeikan University
http://hiroyuki.tomiyama-lab.org/

MPSoC 2019

Classic Task Scheduling on Multicores

- Problem
  - Given: task graph, # cores
  - Goal: minimization of schedule length
- NP-hard complexity
- Many algorithms developed
  - List scheduling, SA, GA, B&B, ILP, etc.
- Heuristic algorithms try to execute as many tasks as possible simultaneously on different cores



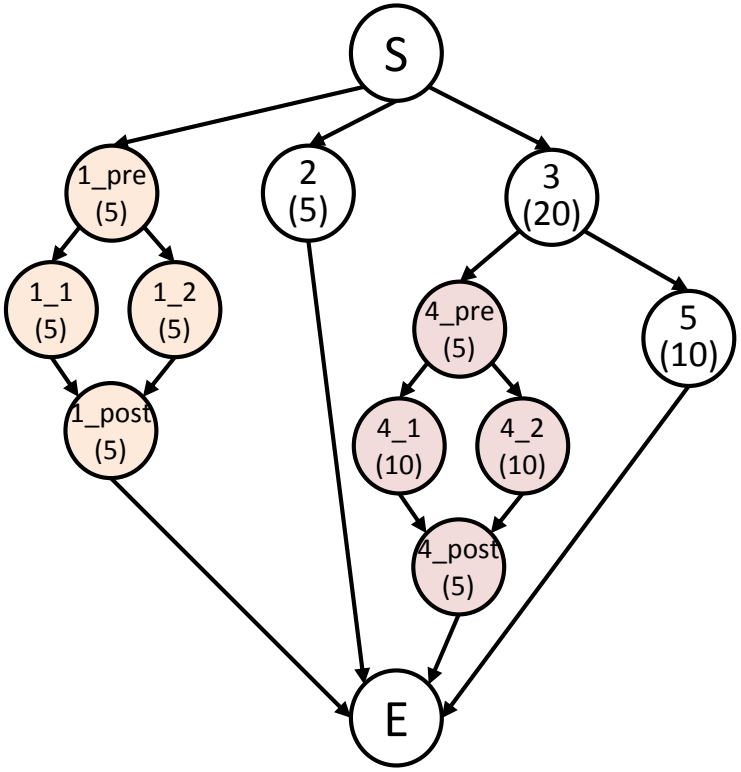


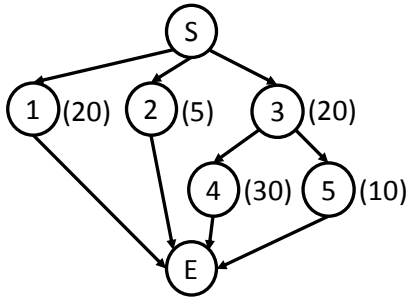
Lots of Extensions

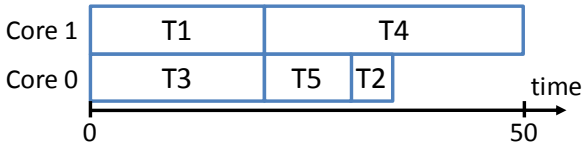
- Inter-task communication
  - Buses, NoCs, etc.
- Heterogeneous cores
- Dynamic power management
- Dynamic voltage and frequency scheduling
- Probabilistic execution times of tasks
- Resource conflicts among running tasks
  - memory, buses, I/O, etc.
- Conditional task graphs
- Pipelined scheduling
- Deadline constraints for individual tasks
- Multiple task graphs with different execution rates
- Intra-task data parallelism
  - Individual tasks may run on multiple cores
- Much more

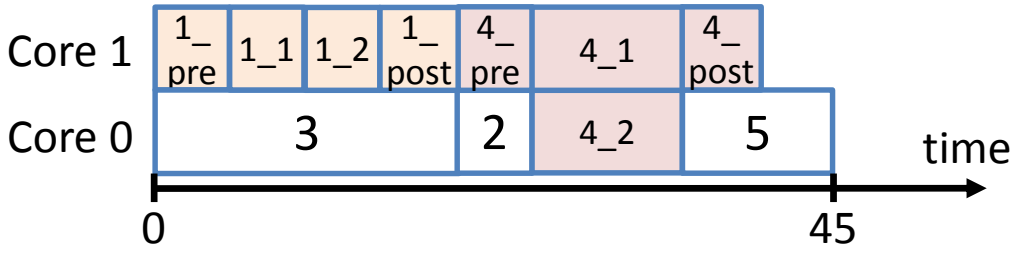
Fork-Join Parallel Tasks

- In many application domains such as multimedia, individual tasks have inherent data parallelism
- A task can be split into multiple threads (sub-tasks) to allow data parallel execution in a fork-join manner
- No extension is necessary in scheduling algorithms



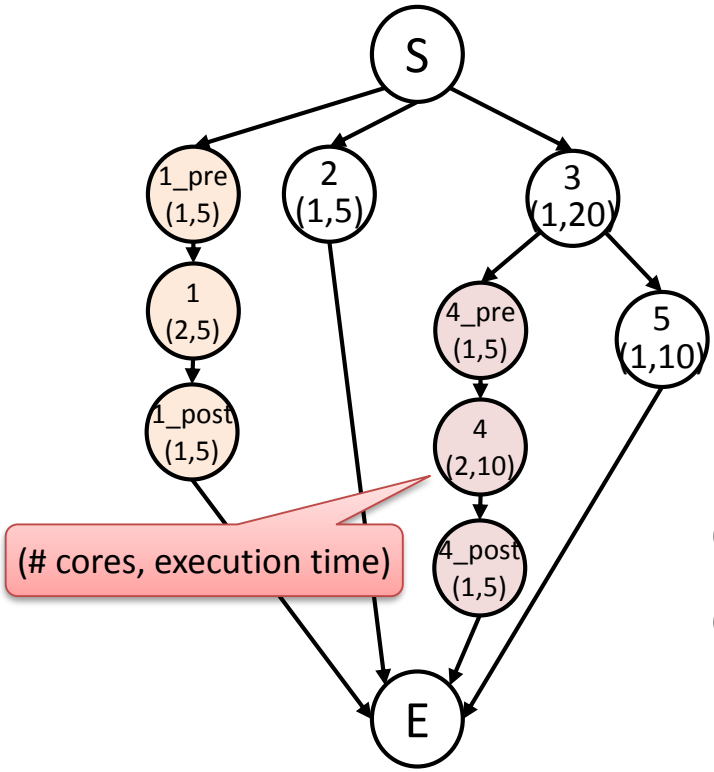


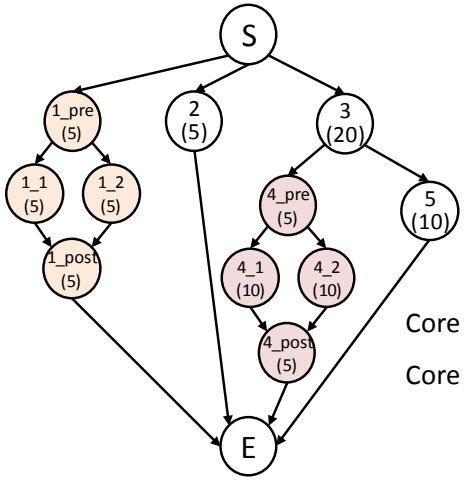


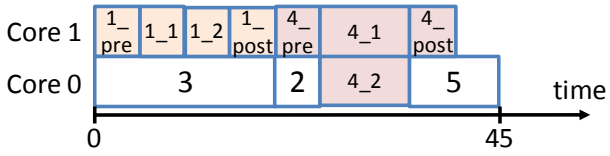


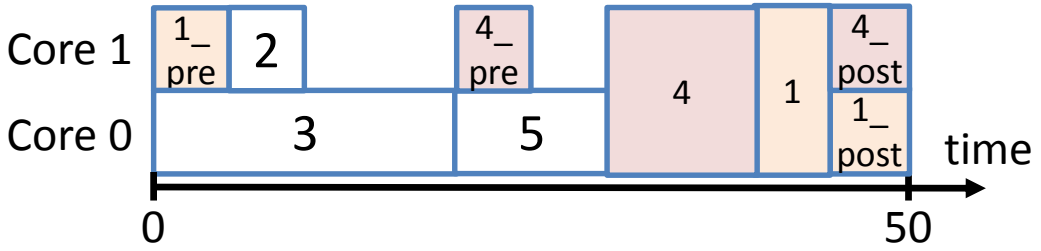
Synchronous Parallel Tasks

- Sub-tasks may communicate and synchronize with each other very frequently
- Such sub-tasks need to be executed in parallel at the same time
- Several algorithms extended
  - List scheduling, B&B, GA, etc.
  - Yang Liu, *Scheduling Algorithms for Data-Parallel Tasks on Multicore Architectures*, Ph.D. thesis, Ritsumeikan University, 2018.

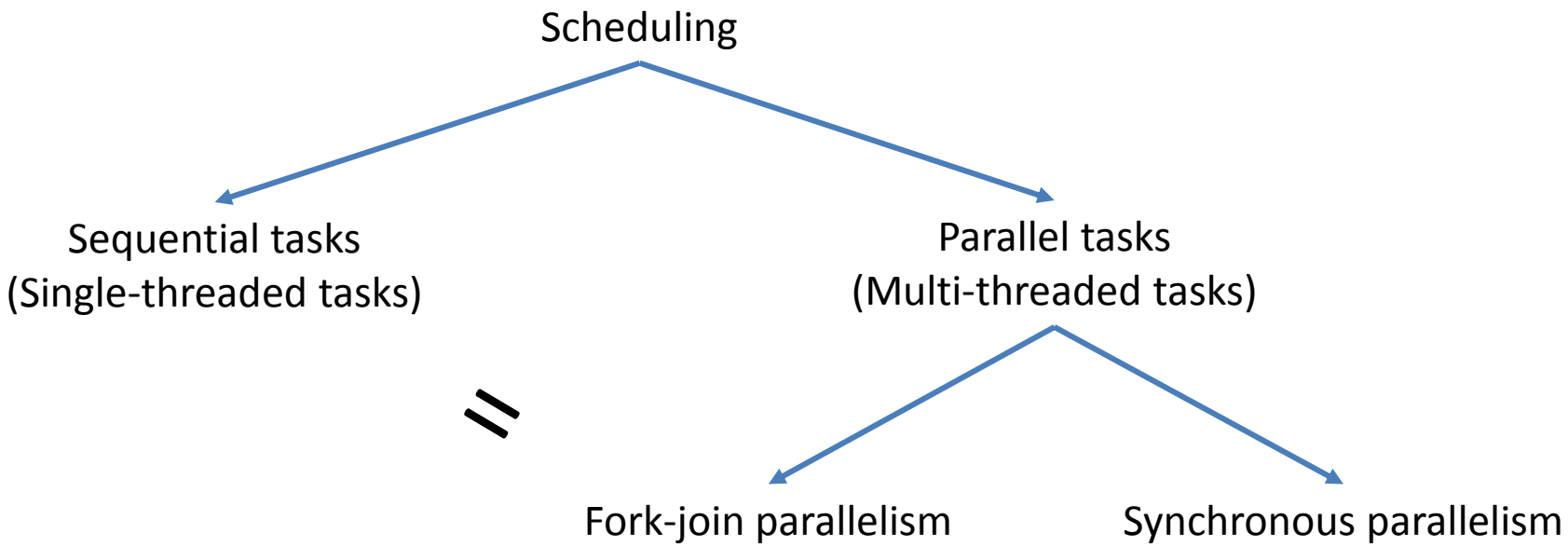






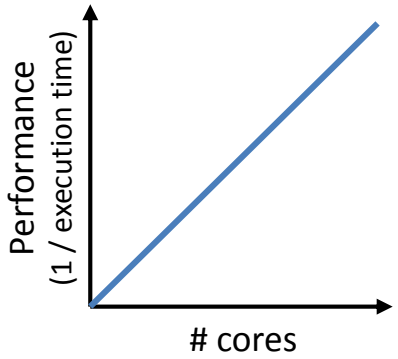


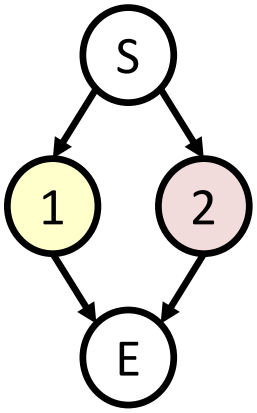
Classification of Multicore Task Scheduling



Question

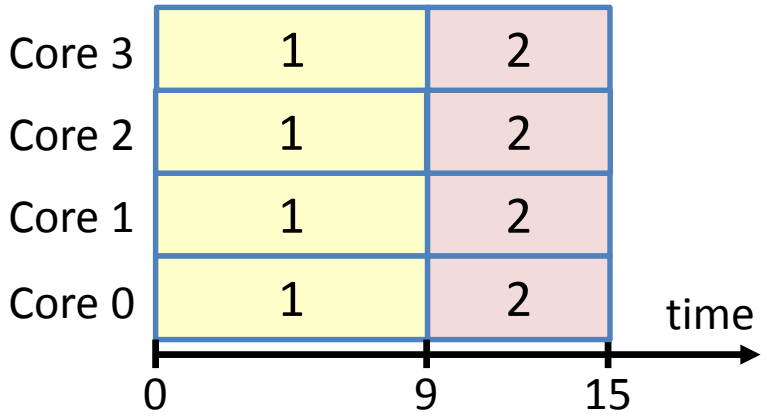
- For each task, who decides the number of sub-tasks, and how?
- Ideal case
  - All tasks are parallelizable and scalable
  - Best schedule is
    - Assign all cores to the tasks
    - Schedule the tasks sequentially





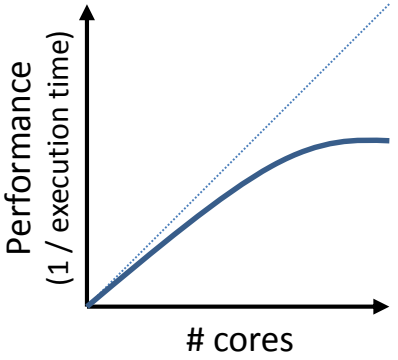
Execution time

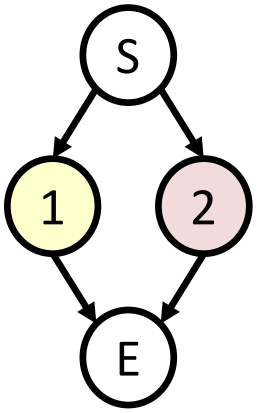
# cores	Task 1	Task 2
1	36	24
2	18	12
3	12	8
4	9	6



Question

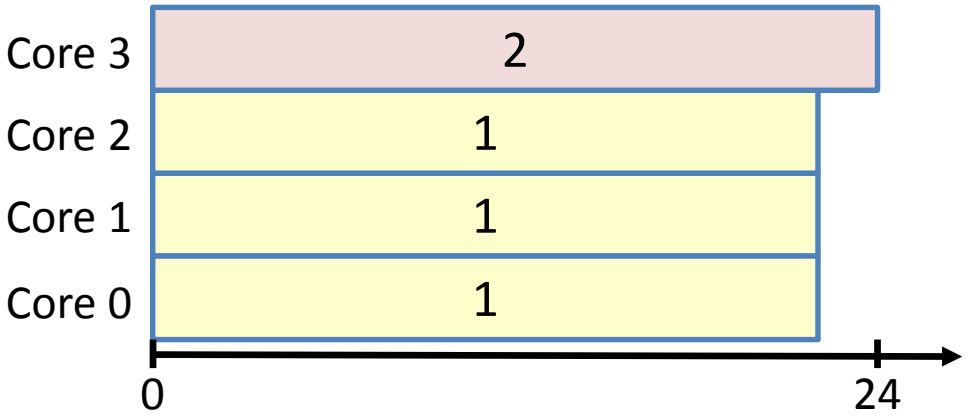
- For each task, who decides the number of sub-tasks, and how?
- Reality
  - Performance of parallel processing is rarely proportional to the number of cores
- The number of sub-tasks should be determined at the same time as task scheduling
  - Malleable (or moldable) task scheduling



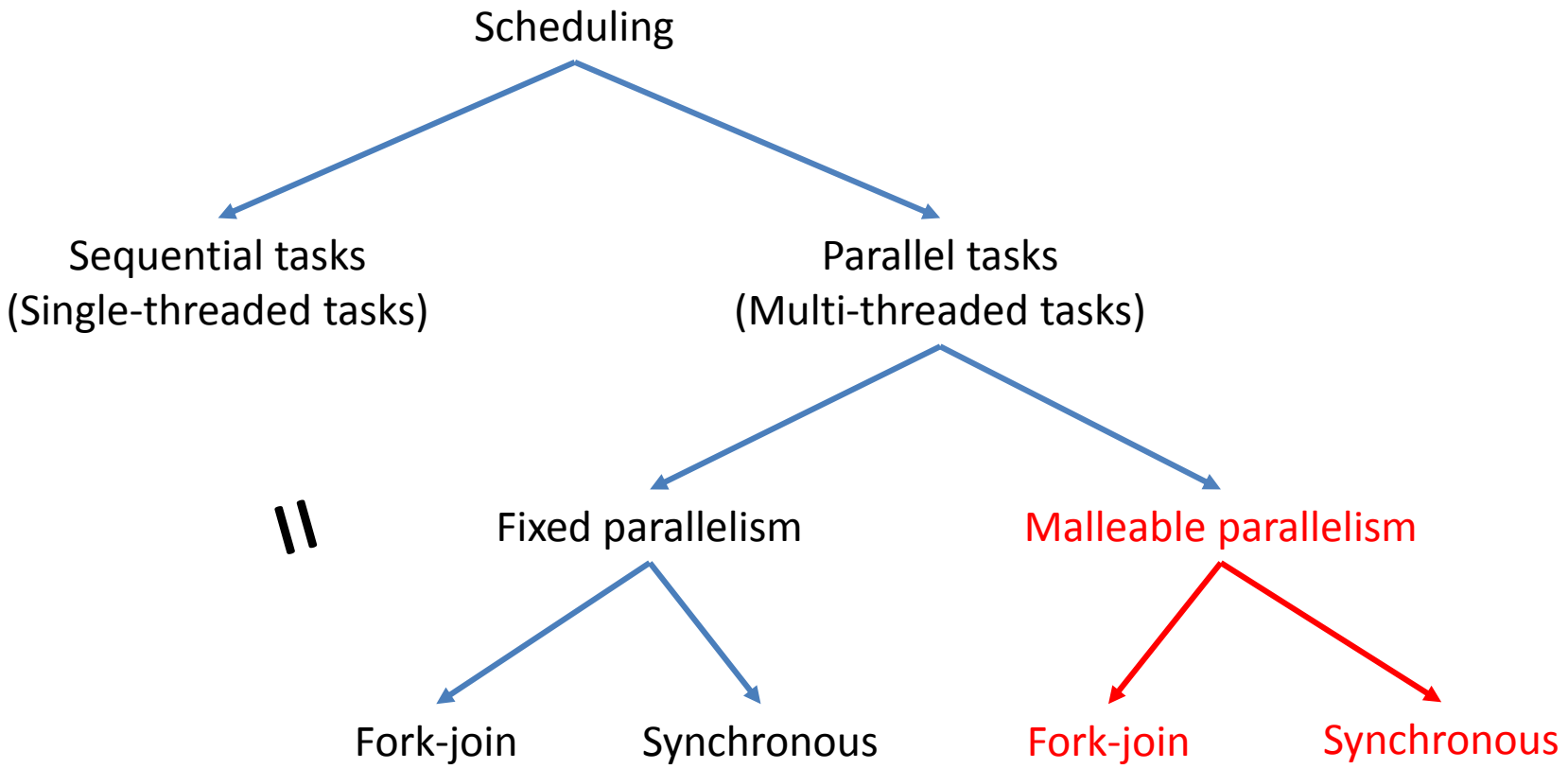


Execution time

# cores	Task 1	Task 2
1	36	24
2	28	18
3	22	14
4	18	12



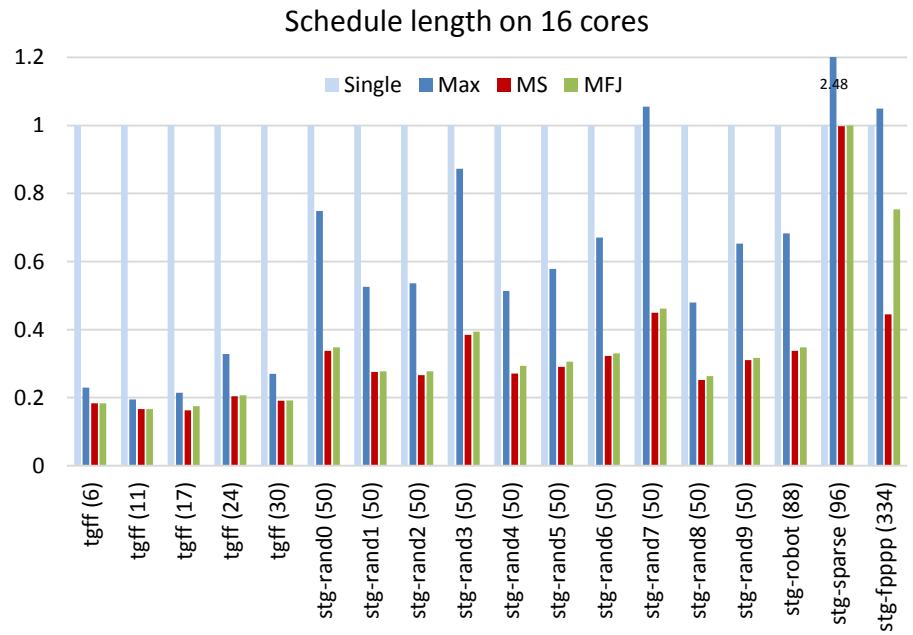
Classification of Multicore Task Scheduling

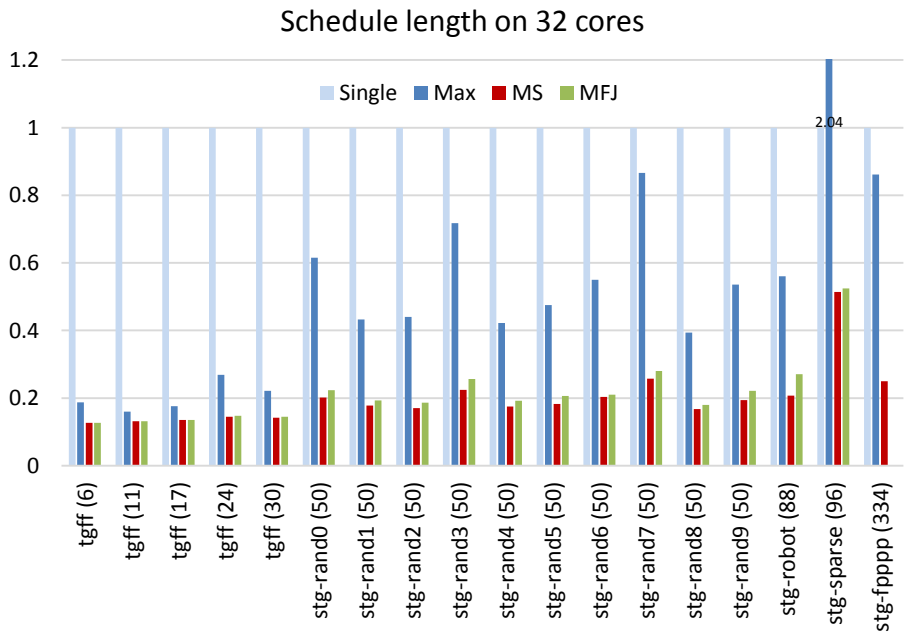


Experiments on Malleable Task Scheduling

- Our approach
  - Constraint programming
  - IBM CP Optimizer
    - CPU time limited to 10 hours
- Benchmark task graphs
  - TGFF (task graph for free)
  - STG (standard task graphs) from Waseda University

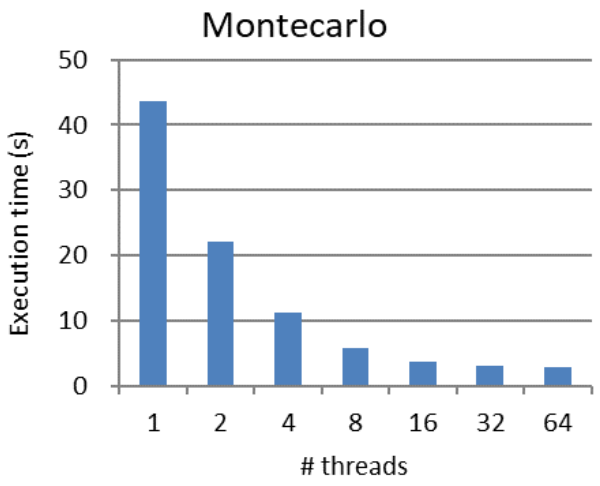
- Compared methods
  - Single
    - Single core for each task
  - Max
    - All cores for each task, sequential order
  - MS
    - Malleable synchronous scheduling
  - MFJ
    - Malleable fork-join scheduling

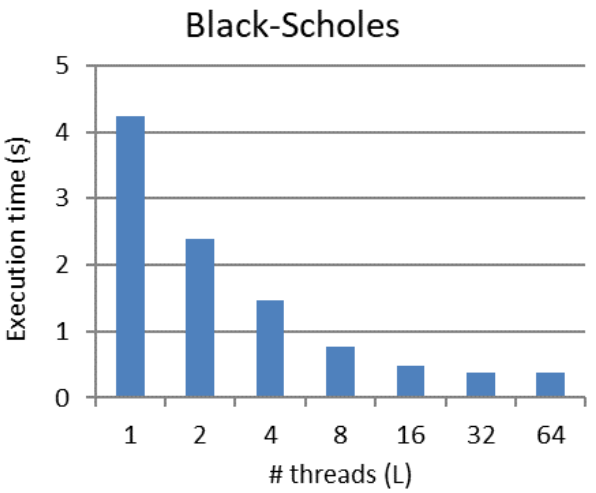




Our OpenCL Framework for Multicores

- In OpenCL, data is split into *work-items*
- A work-item is the minimum unit of parallel execution
  - On GPU, a work-item corresponds to a thread
- Our OpenCL framework aggregates work-items to form the user-specified number of threads
  - A thread has a *for*-loop to iteratively process the work-items
- We are now working on automatic optimization of the number of threads





Summary

- Task scheduling should take account of both inter-task parallelism and intra-task parallelism in order to take advantage of manycore architecture
- The degree of intra-task parallelism (the number of cores for each task) should be determined at the same time as task scheduling
- Future work
  - Lots of extensions
    - Communication, heterogeneous cores, DPM, DVFS, probabilistic execution times, resource conflicts, deadline constraints, and more
  - Online task scheduling with online learning