Disk Cache-Aware Task Scheduling
For Data-Intensive and Many-Task Workflow

Masahiro Tanaka and Osamu Tatebe

University of Tsukuba, JST/CREST

Outline

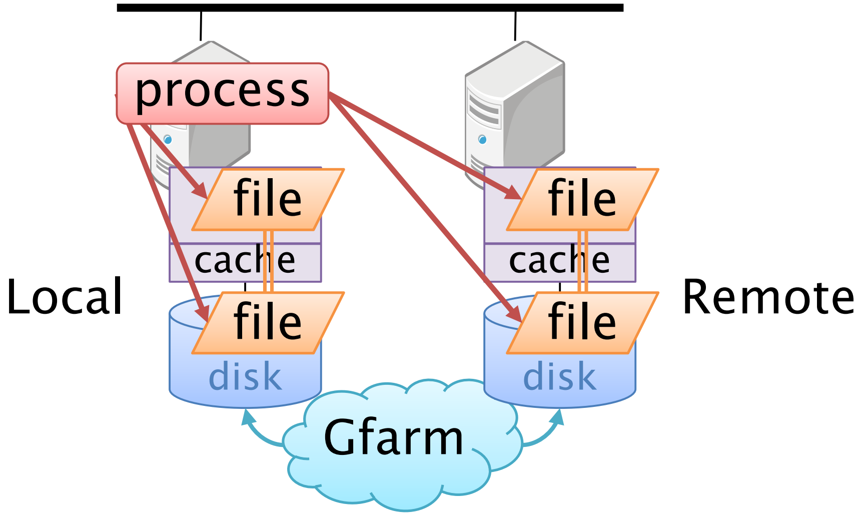
- Introduction
  - Workflow Scheduling for Data-Intensive and Many-Task Computing
- Disk Cache-aware Task Scheduling
- Proposed Method
  - (1) LIFO + HRF
  - (2) Rank Equalization + HRF
- Evaluation
  - (1) I/O-only workflow
  - (2) Montage astronomy workflow
- Related Work
- Conclusion

Background

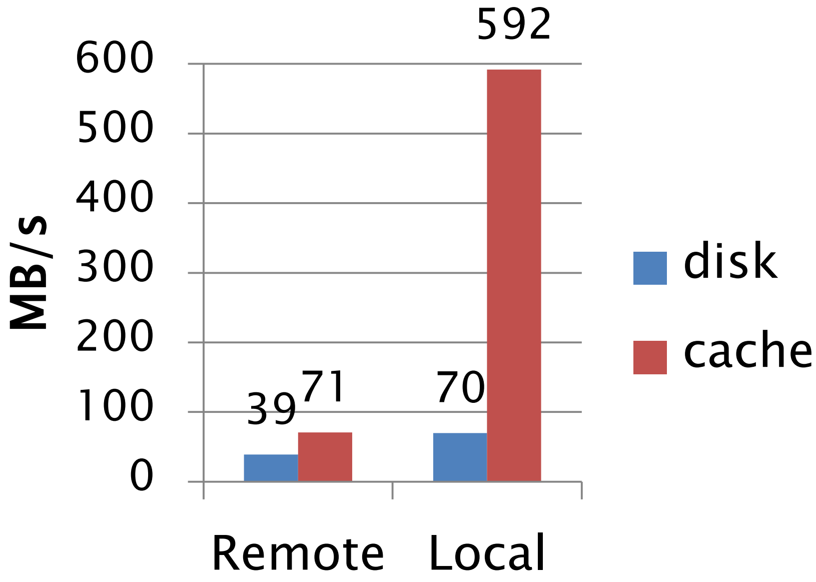
- Many Task Computing (MTC)
  - Raicu et al. (MTAGS 2008)
  - Task throughput for 10^3 – 10^6 tasks
- *Pwrake :* Parallel Workflow extension to ***Rake***
  - Rake = Ruby version of make
  - Target: Data-intensive and Many-task Workflows
- *Gfarm* distributed file system
  - Scalable I/O performance
  - Use local storage of compute nodes

Objective: I/O-aware Task Scheduling

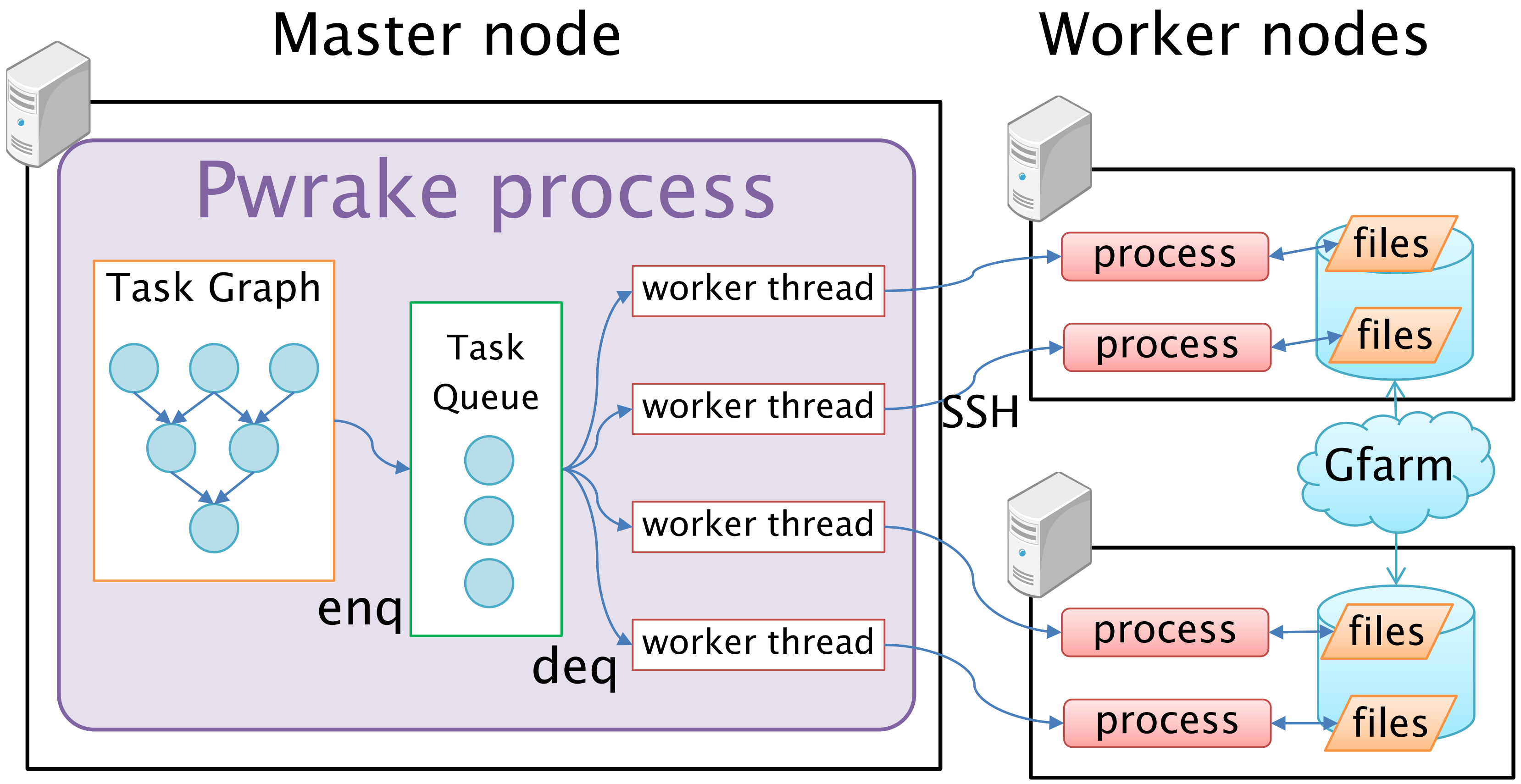
- Issues:
  - File Locality
    - our previous work
  - Disk cache (buffer/page cache)
    - this work



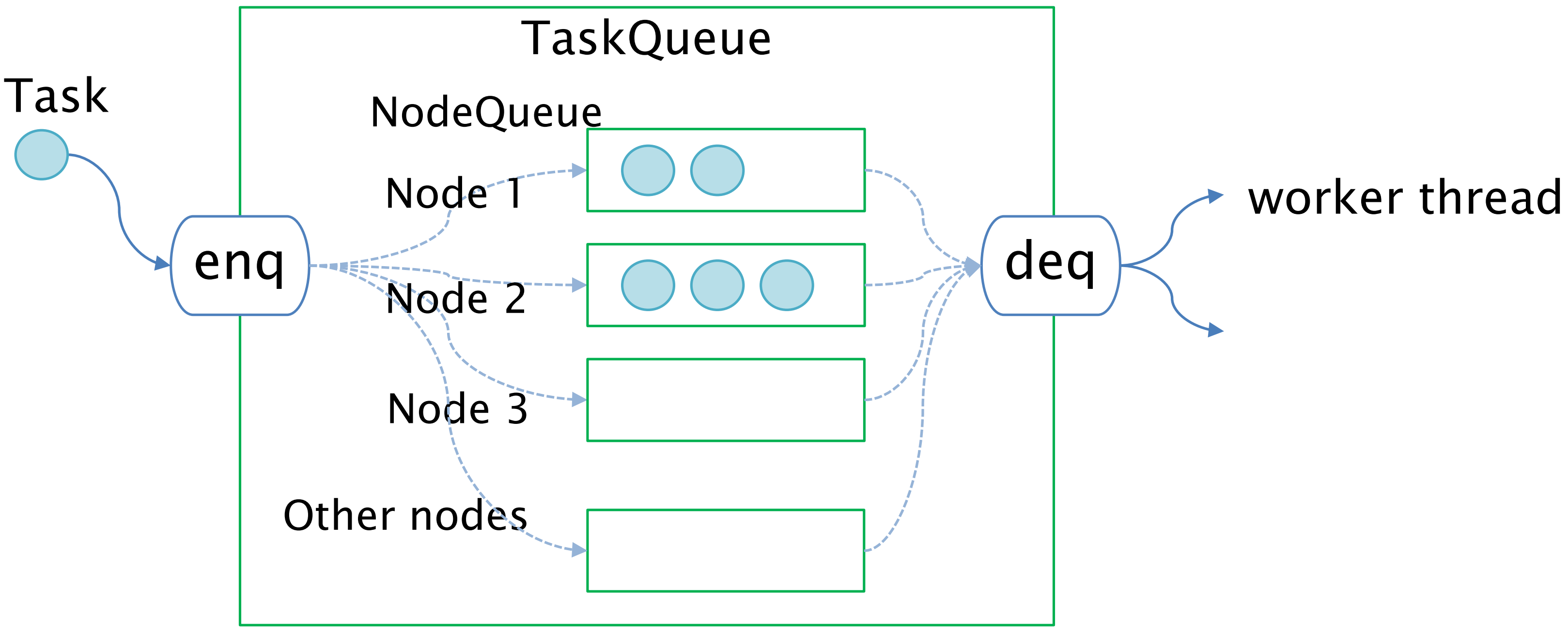
Read performance of Gfarm file (HDD, GbE)



Architecture of Pwrake



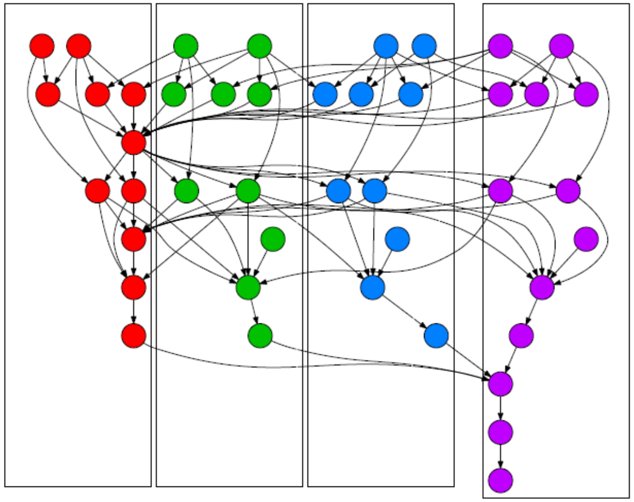
Task Queue of Pwrake



Method to define Candidate Nodes

1. Based on the location of input file
   - 47 % local access

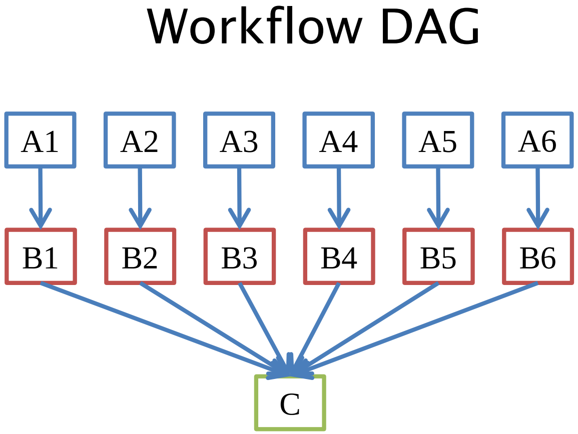
2. MCGP (Multi-Constraint Graph Partitioning)
   - Our previous work (CCGrid 2012)
   - use METIS library
   - 88 % local access

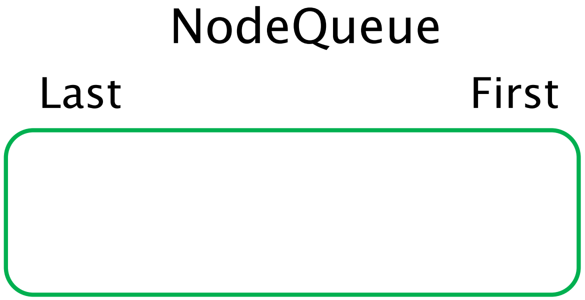


Disk Cache-aware Task Scheduling

- Later-saved file has high probability that the file is cached.
- The order of task execution relates to Disk Cache hit rate.
- In the following slides, I show the behavior of FIFO and LIFO queues.

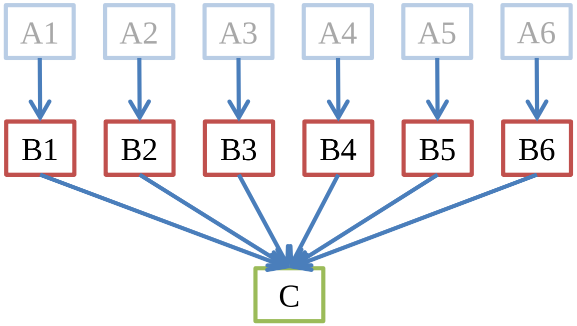
FIFO behavior



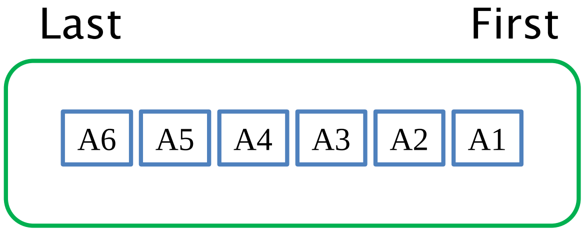


FIFO behavior

Workflow DAG

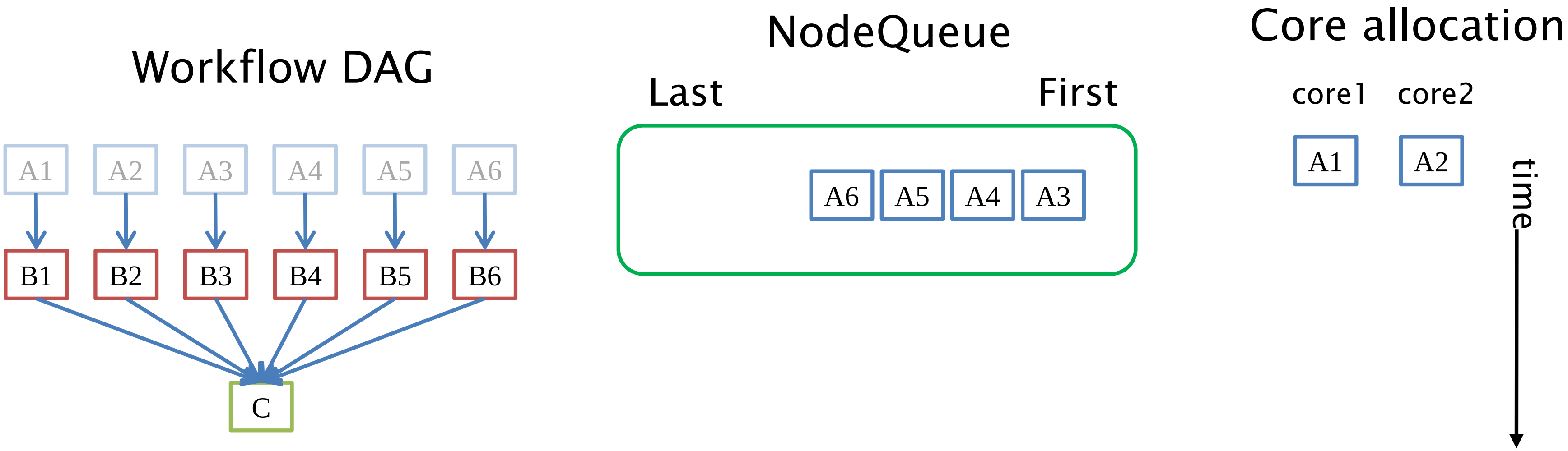


NodeQueue



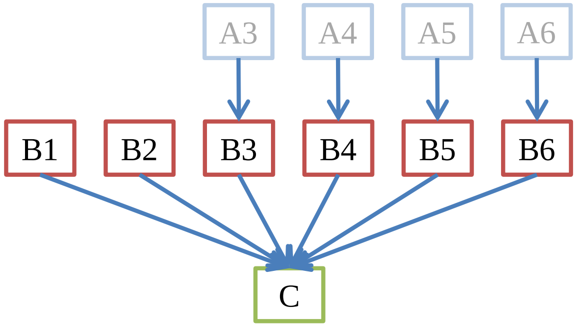
A6 A5 A4 A3 A2 A1

FIFO behavior

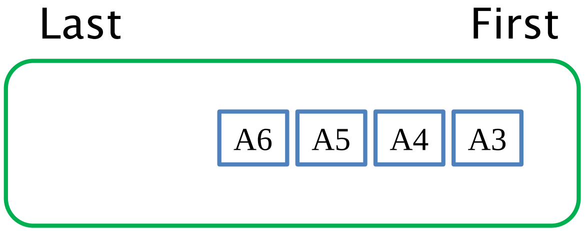


FIFO behavior

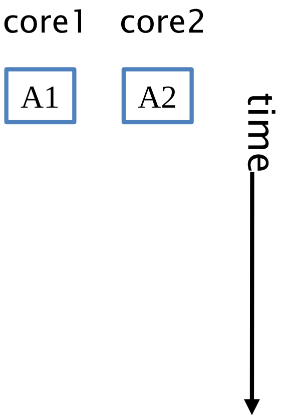
Workflow DAG



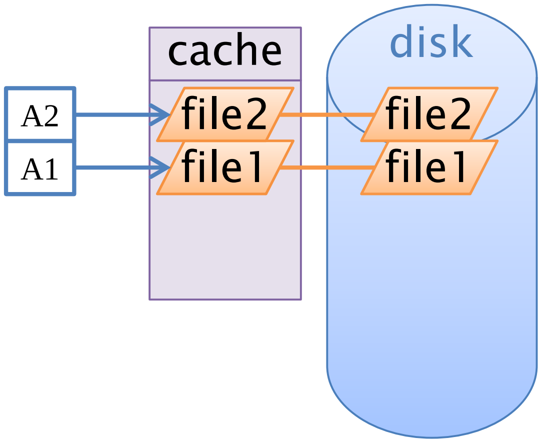
NodeQueue



Core allocation

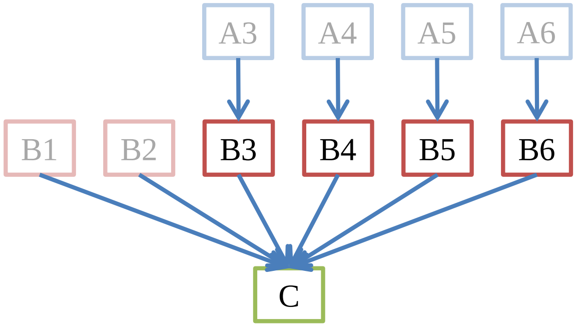


Storage of compute node

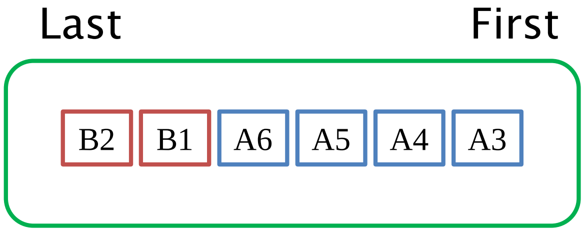


FIFO behavior

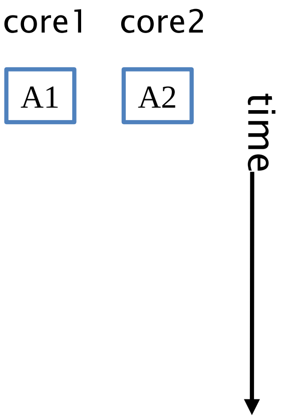
Workflow DAG

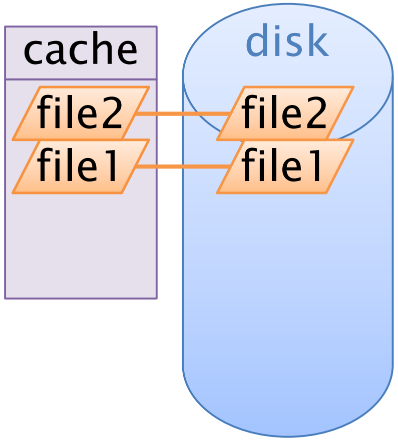


NodeQueue



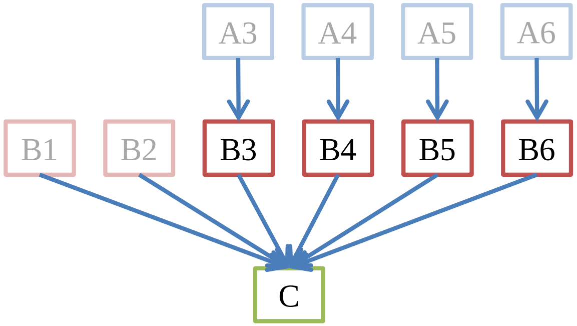
Core allocation



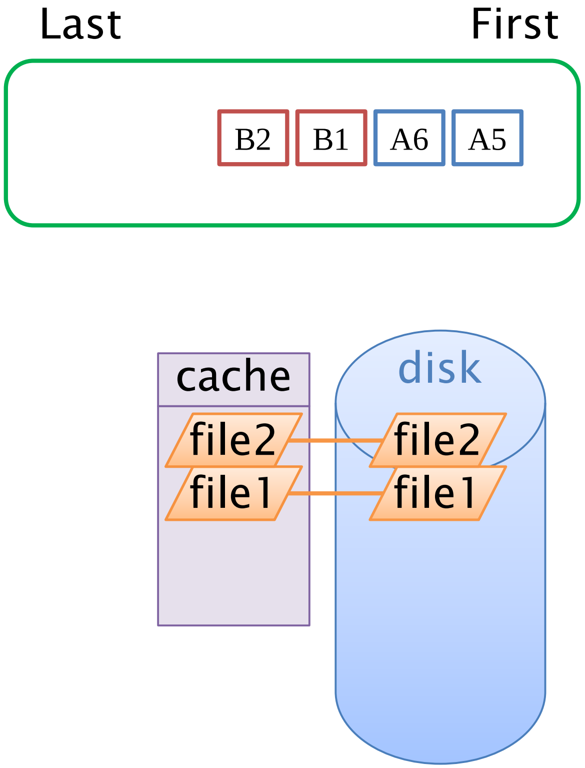


FIFO behavior

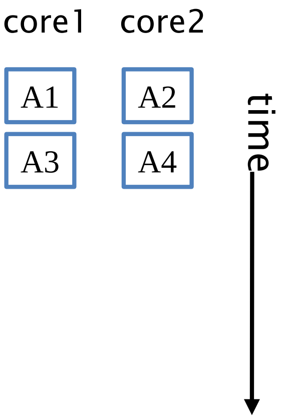
Workflow DAG



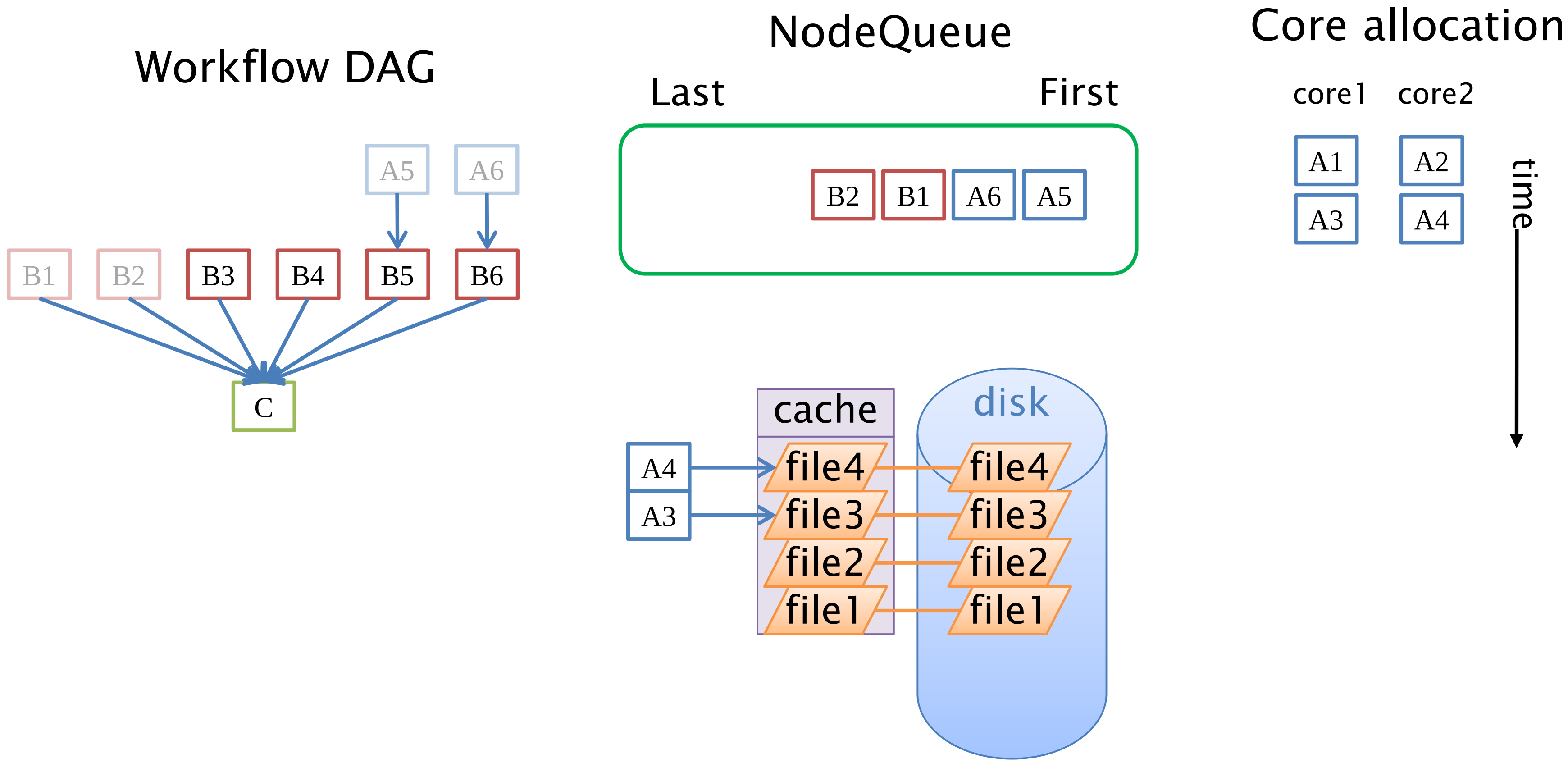
NodeQueue



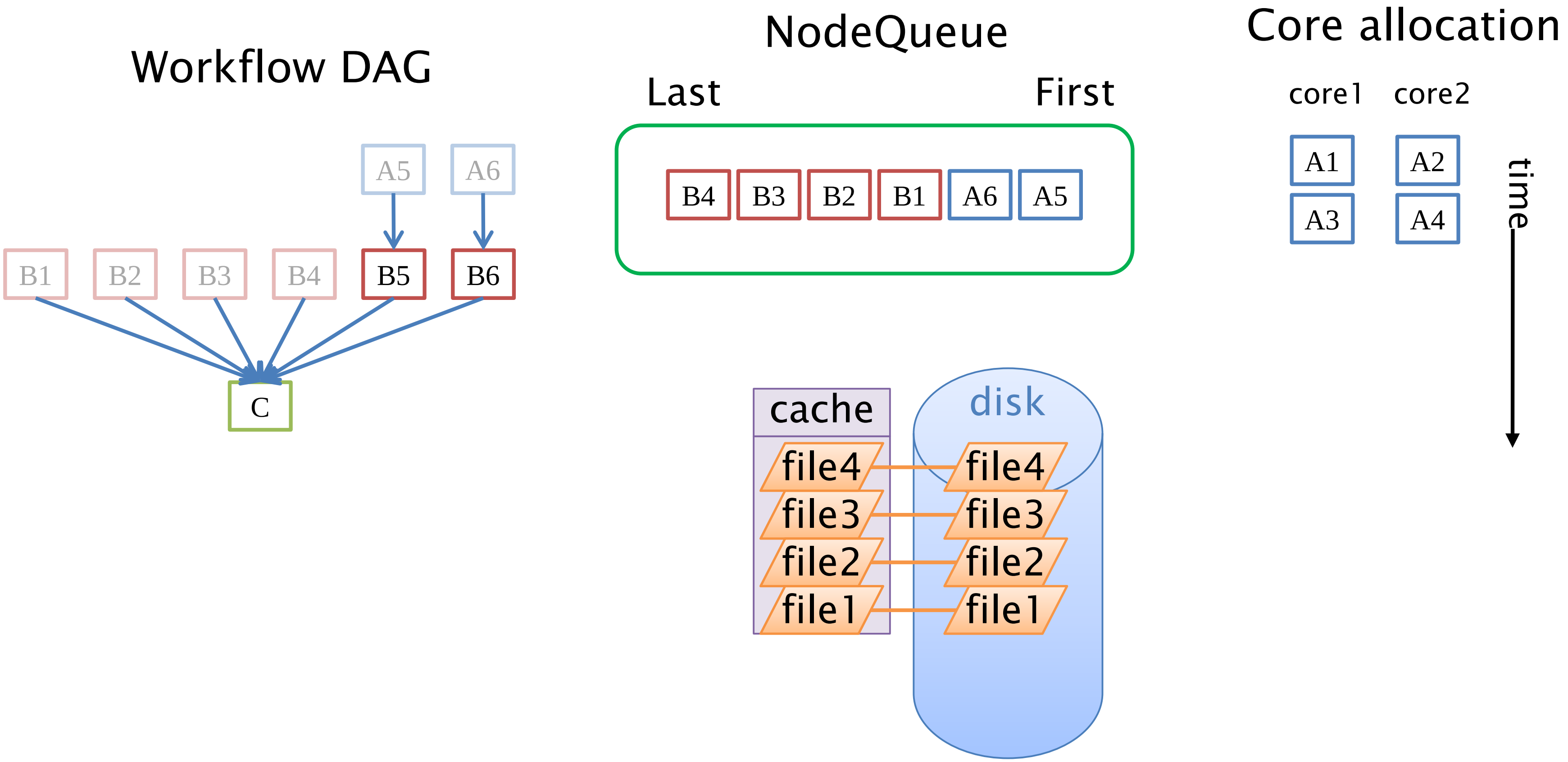
Core allocation



FIFO behavior



FIFO behavior



Workflow DAG

A5 A6

B1 B2 B3 B4 B5 B6

C

NodeQueue

Last First

B4 B3 B2 B1 A6 A5

Core allocation

core1 core2

A1 A2

A3 A4

time

cache

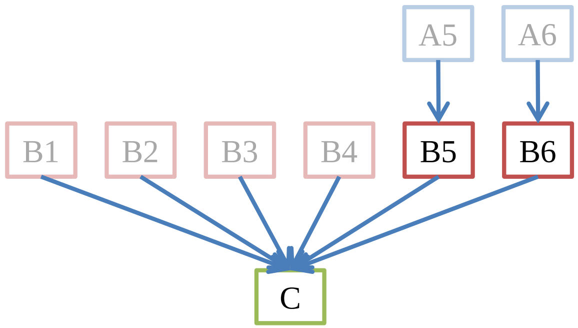
file4
file3
file2
file1

disk

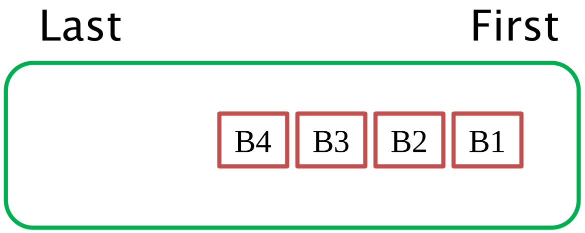
file4
file3
file2
file1

FIFO behavior

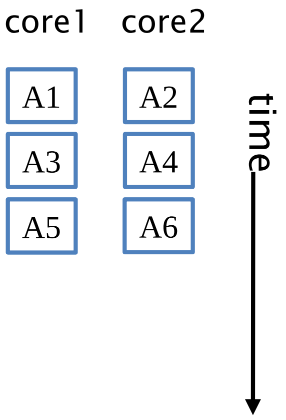
Workflow DAG



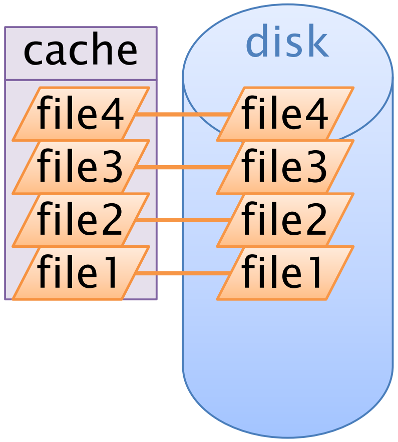
NodeQueue



Core allocation

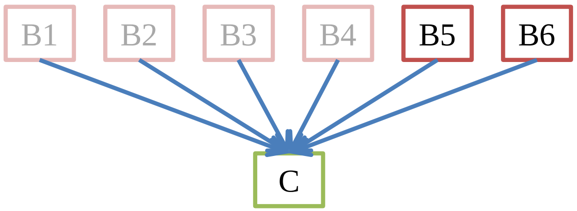


core1	core2
A1	A2
A3	A4
A5	A6

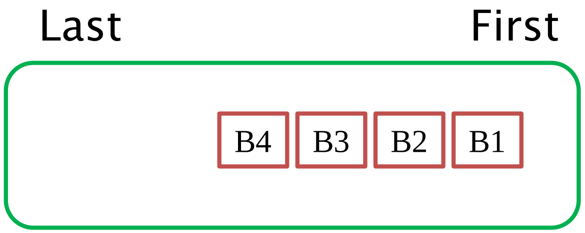


FIFO behavior

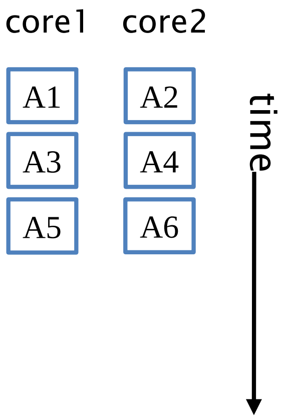
Workflow DAG

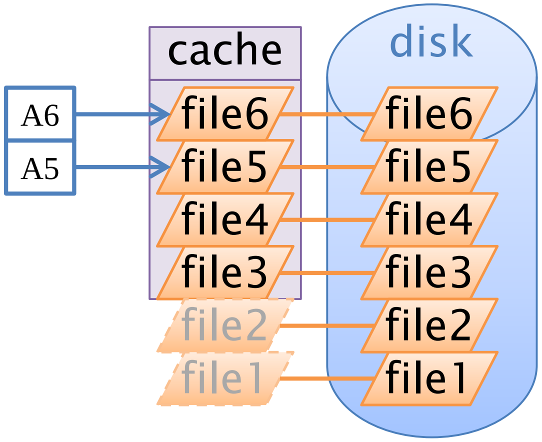


NodeQueue



Core allocation

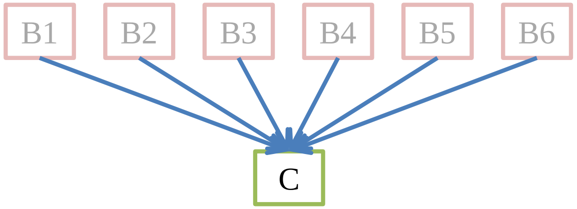




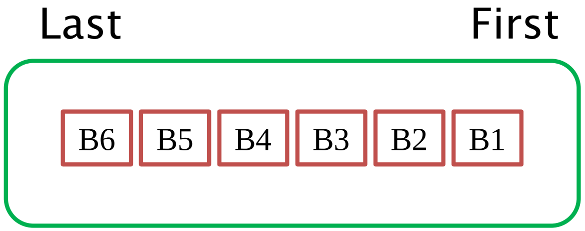
Evicted

FIFO behavior

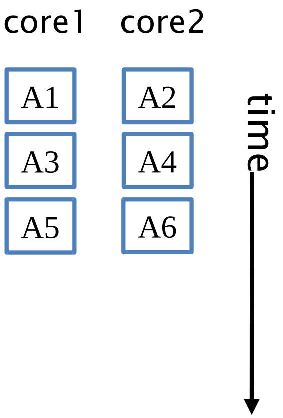
Workflow DAG

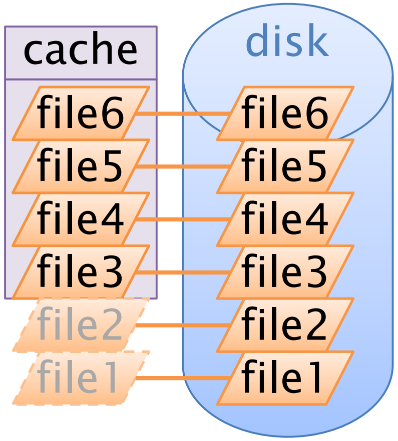


NodeQueue

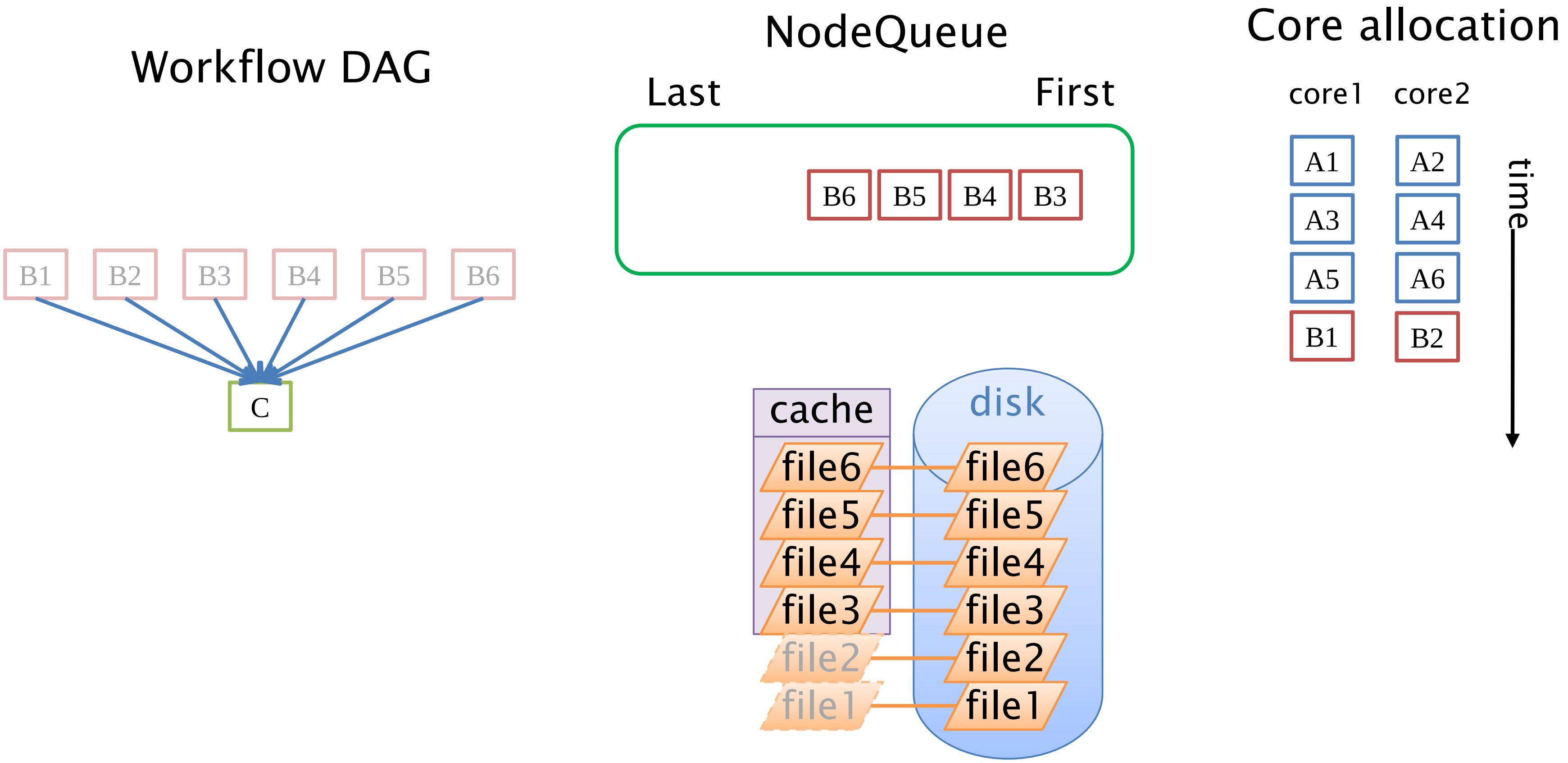


Core allocation

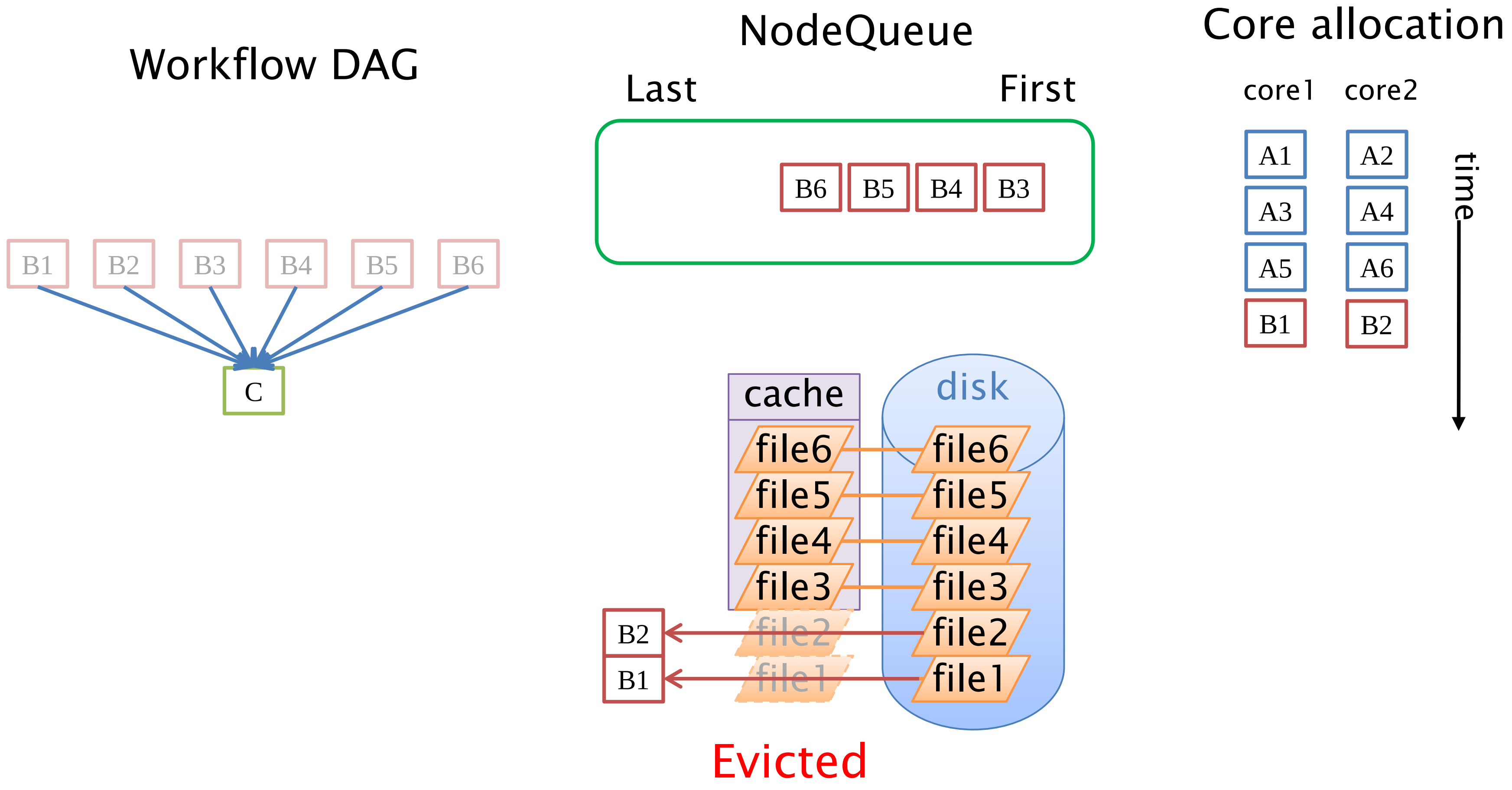




FIFO behavior

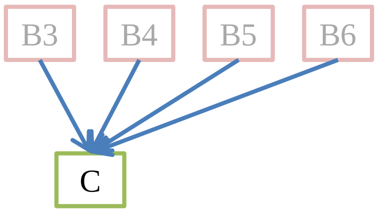


FIFO behavior

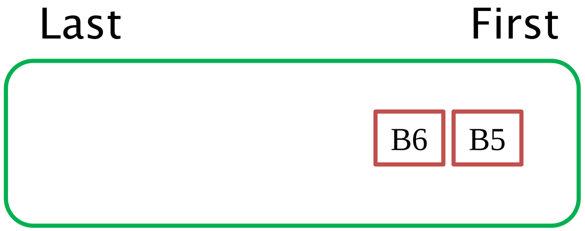


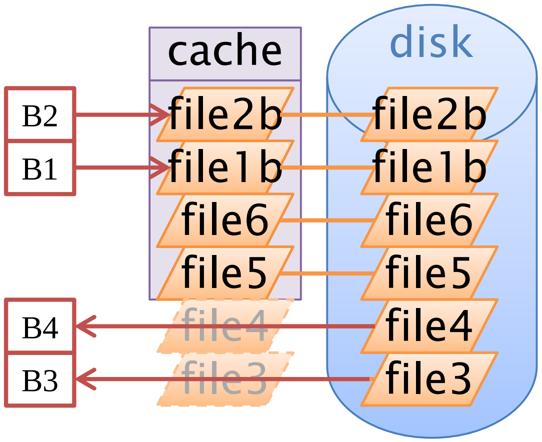
FIFO behavior

Workflow DAG



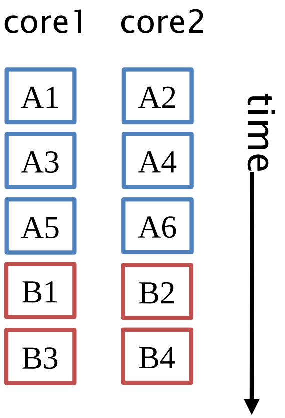
NodeQueue



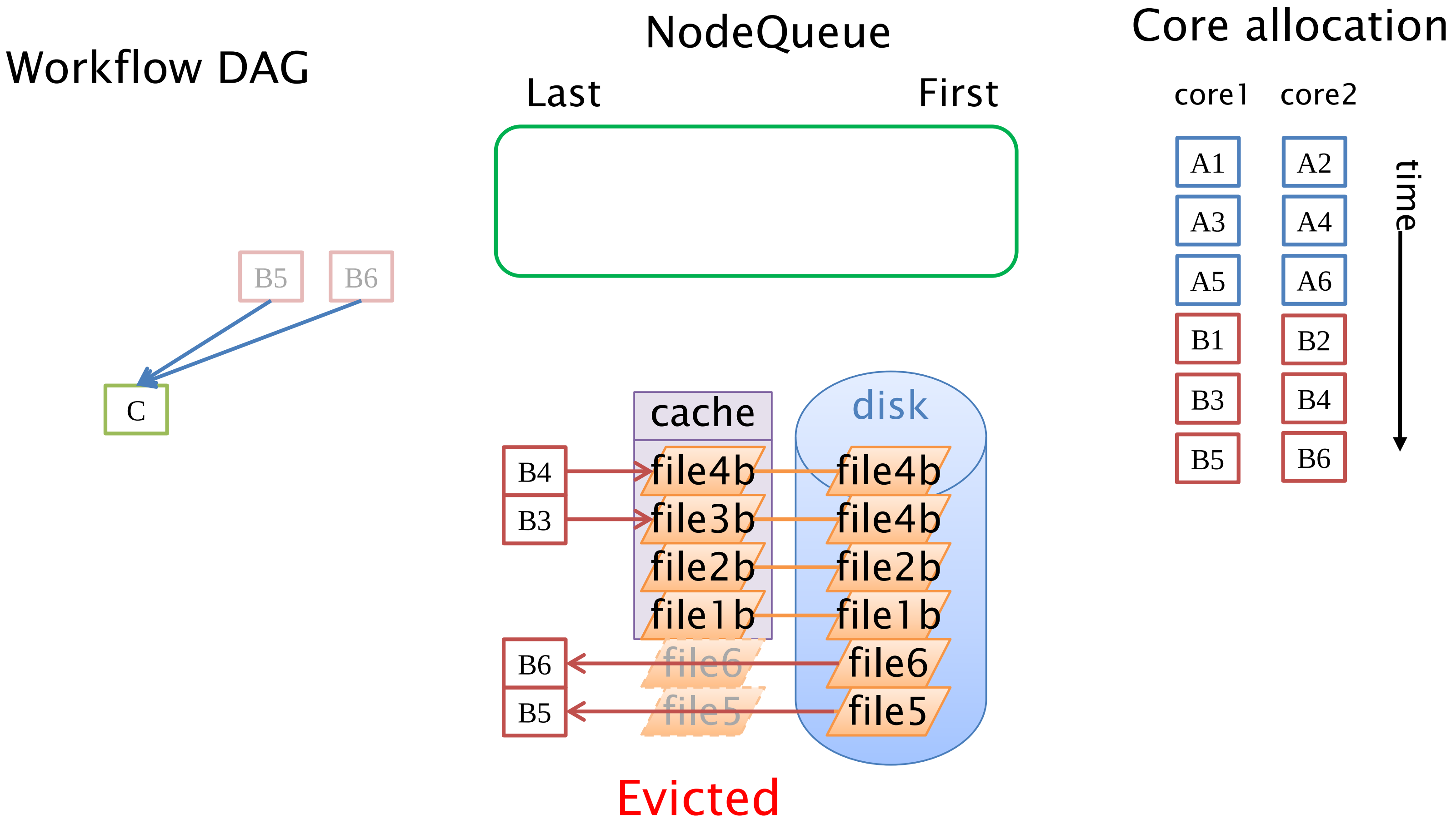


Evicted

Core allocation



FIFO behavior



Workflow DAG

B5 B6

C

NodeQueue

Last First

cache

disk

B4 → file4b

B3 → file3b

file2b

file1b

B6 ← file6

B5 ← file5

file4b

file4b

file2b

file1b

file6

file5

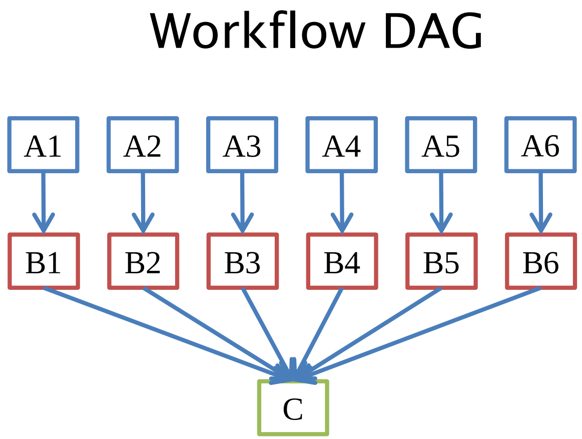
Evicted

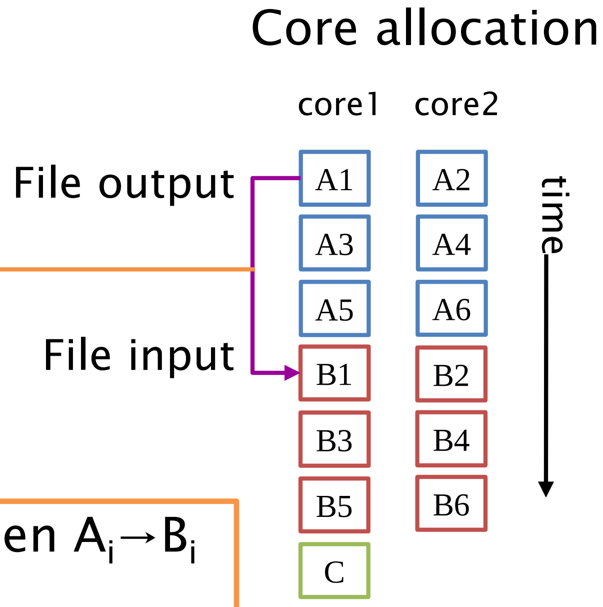
Core allocation

core1	core2
A1	A2
A3	A4
A5	A6
B1	B2
B3	B4
B5	B6

time

FIFO behavior



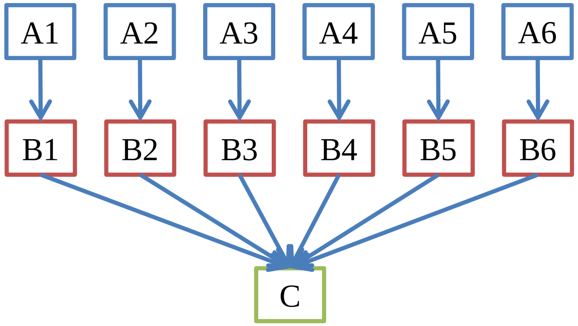


Average difference between $A_i \rightarrow B_i$

$$\sim \frac{\text{\# of A tasks}}{\text{\# of cores}} \text{ tasks}$$

LIFO behavior

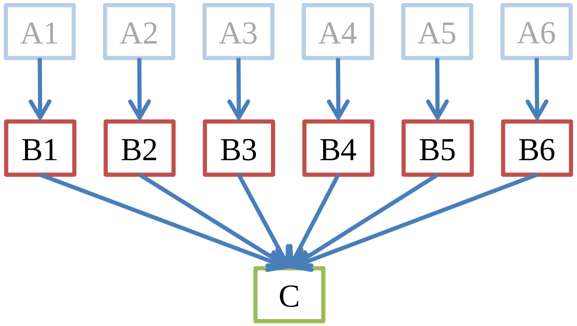
Workflow DAG



NodeQueue

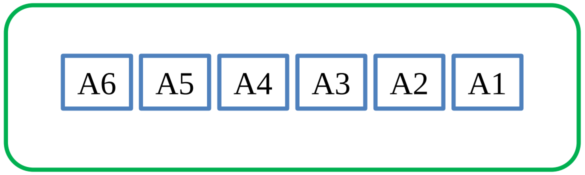
LIFO behavior

Workflow DAG

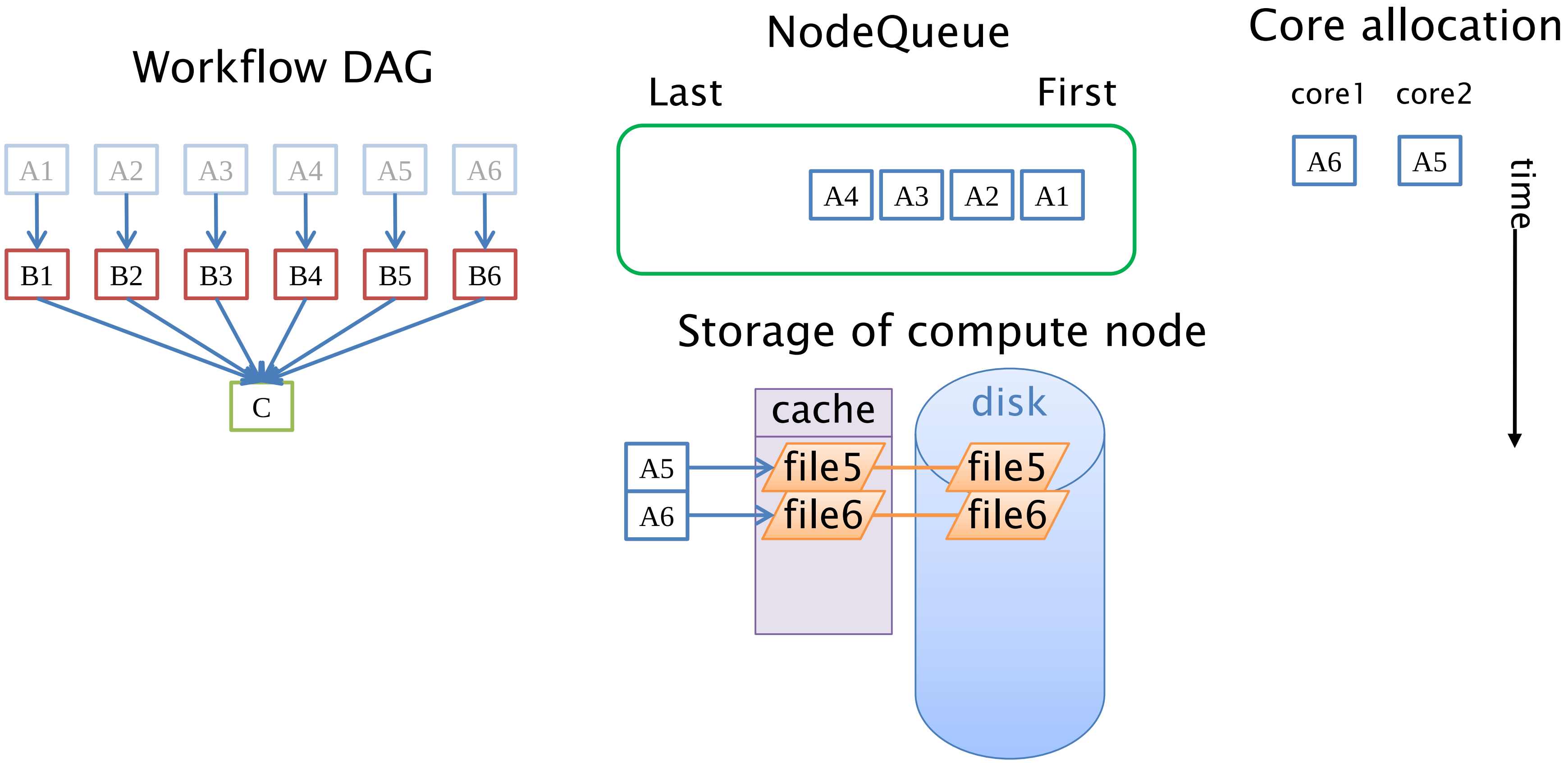


NodeQueue

Last First



LIFO behavior



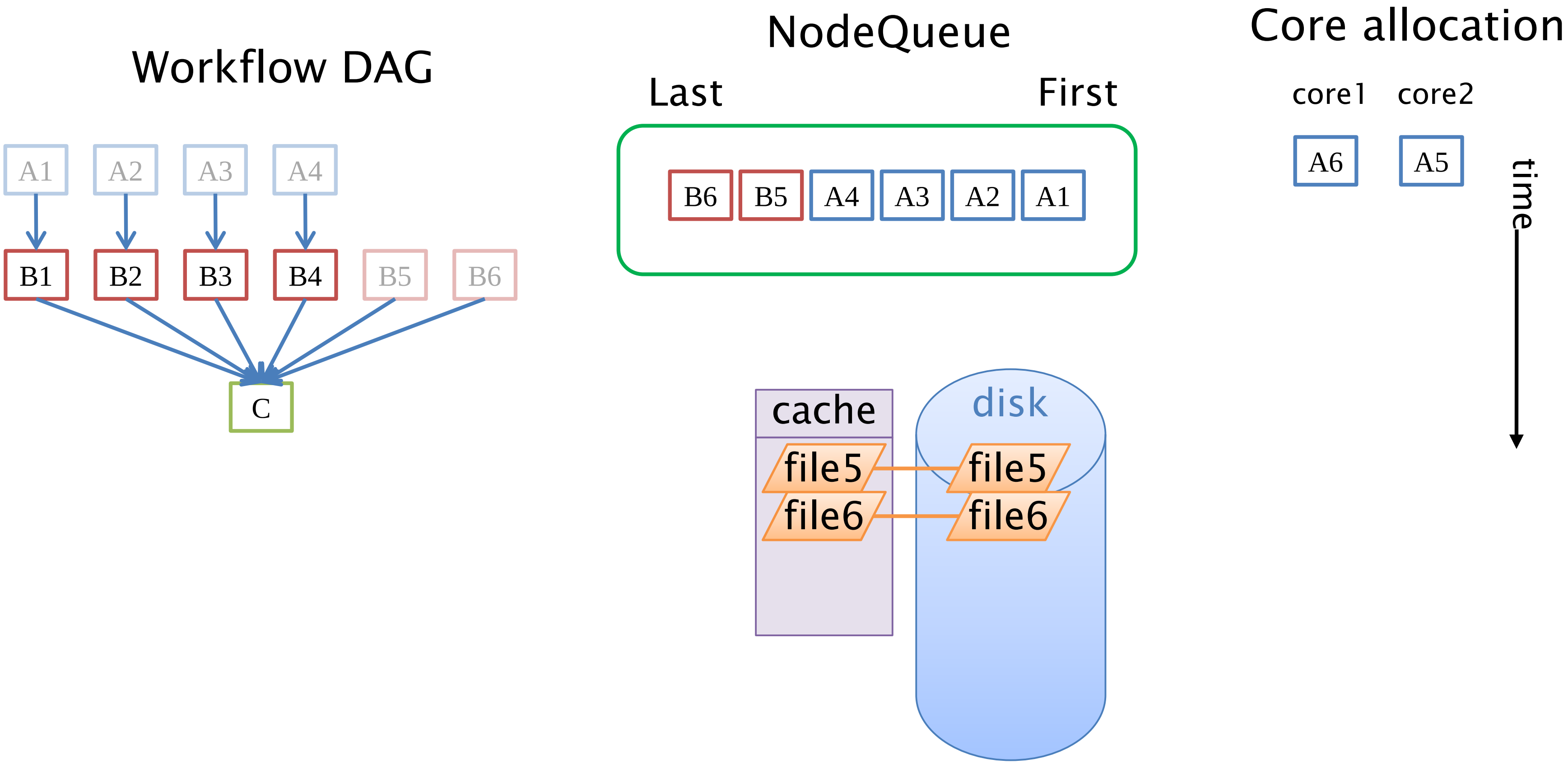
Workflow DAG

NodeQueue

Core allocation

Storage of compute node

LIFO behavior



Workflow DAG

A1 A2 A3 A4

B1 B2 B3 B4 B5 B6

C

NodeQueue

Last First

B6 B5 A4 A3 A2 A1

Core allocation

core1 core2

A6 A5

time

cache

file5

file6

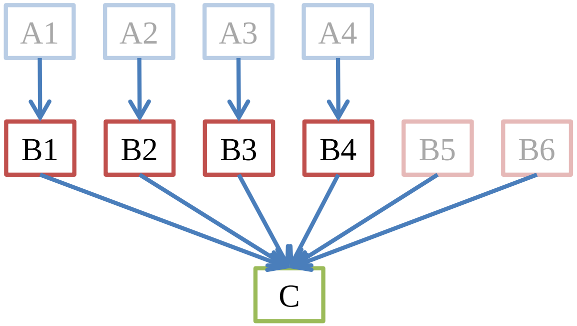
disk

file5

file6

LIFO behavior

Workflow DAG

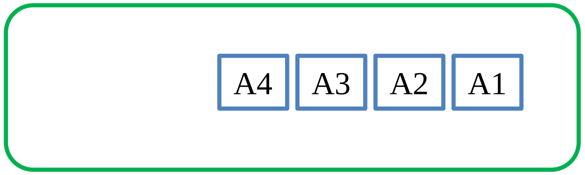

A1 A2 A3 A4

B1 B2 B3 B4 B5 B6

C

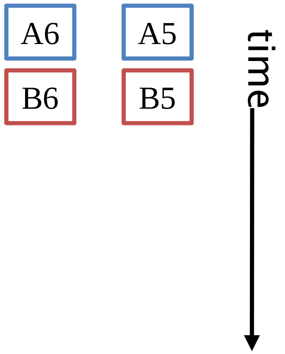
NodeQueue

Last First


A4 A3 A2 A1

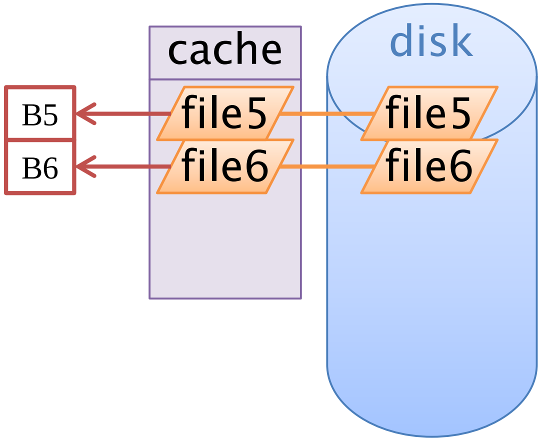
Core allocation

core1 core2


A6 A5

B6 B5

time

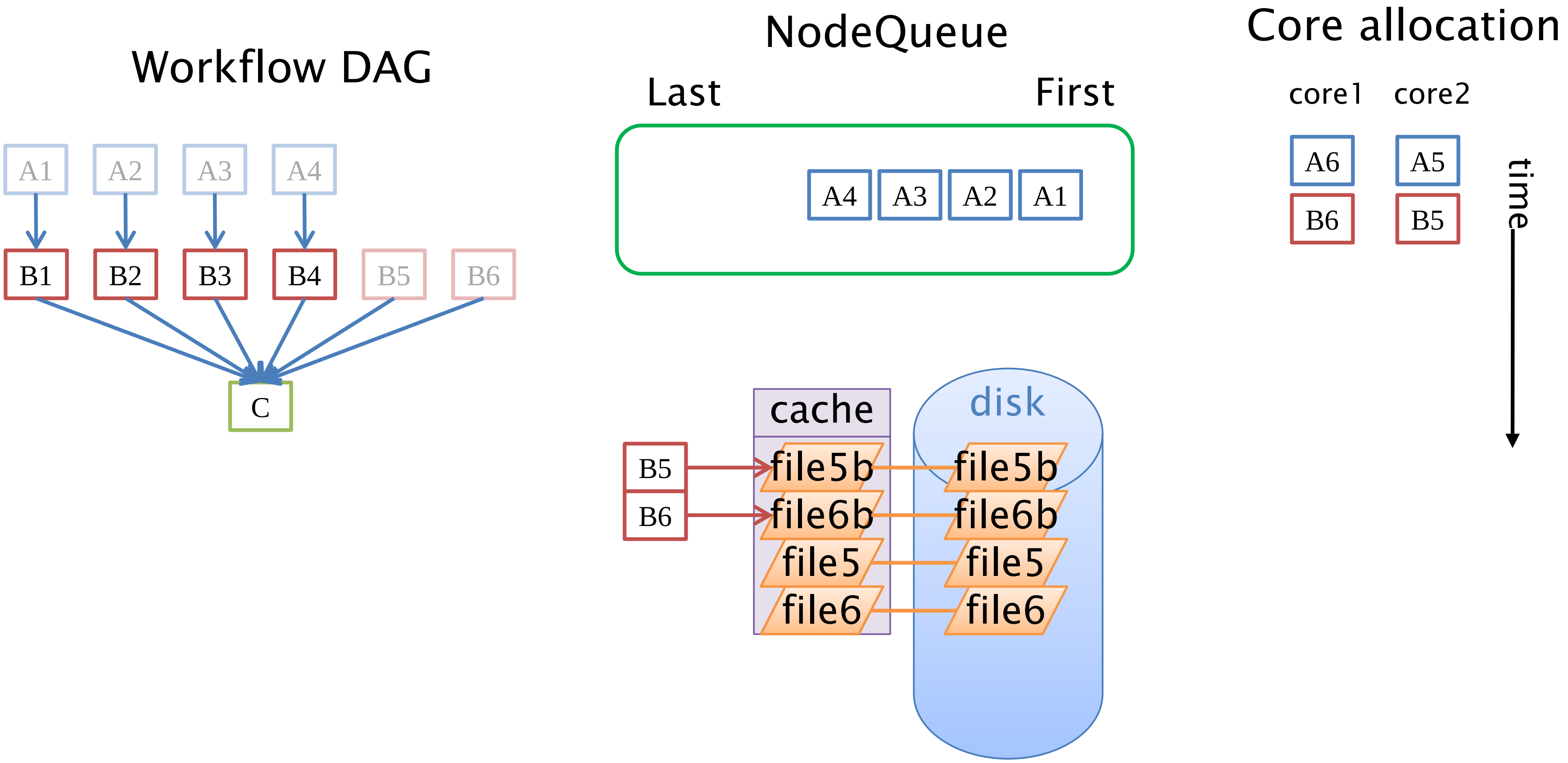

cache

disk

B5 file5 file5

B6 file6 file6

LIFO behavior



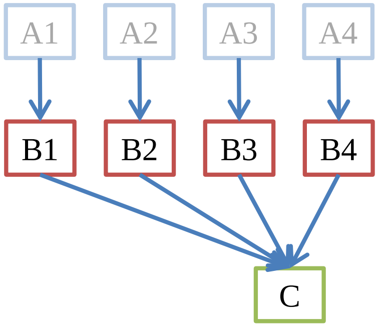
Workflow DAG

NodeQueue

Core allocation

LIFO behavior

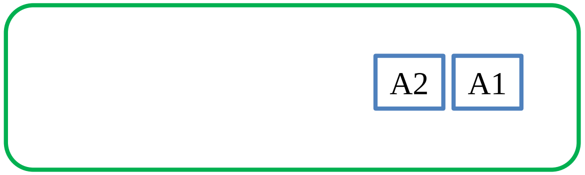
Workflow DAG



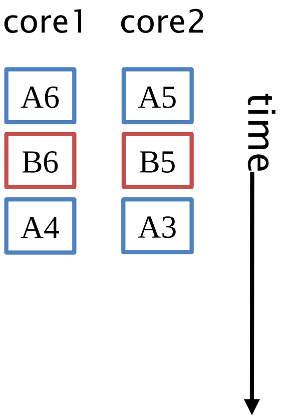
NodeQueue

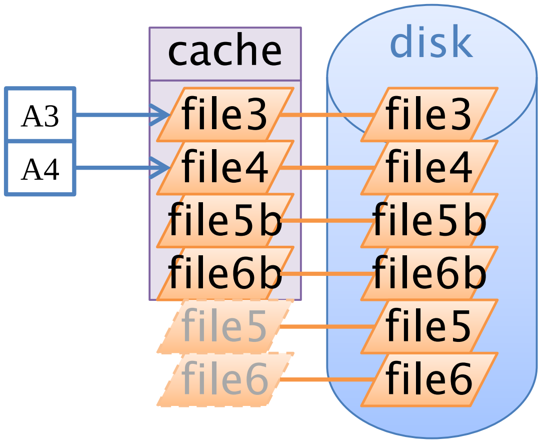
Last

First



Core allocation

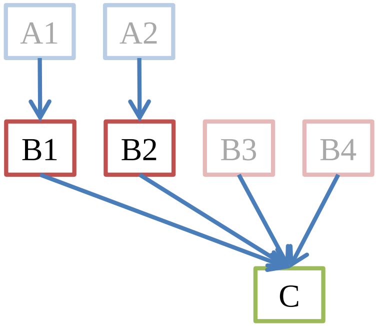




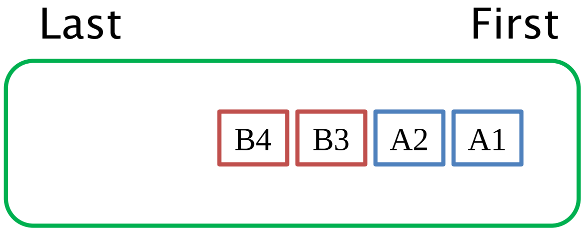
Evicted

LIFO behavior

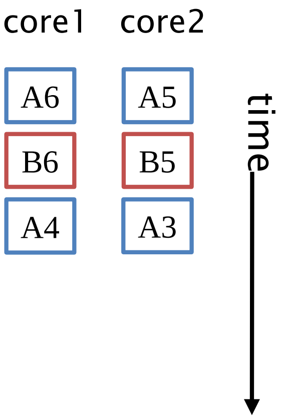
Workflow DAG



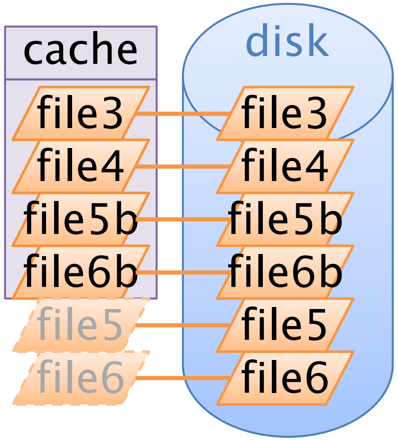
NodeQueue



Core allocation

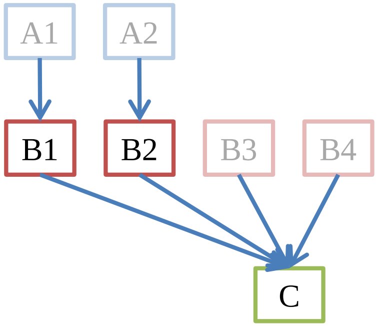


core1	core2
A6	A5
B6	B5
A4	A3



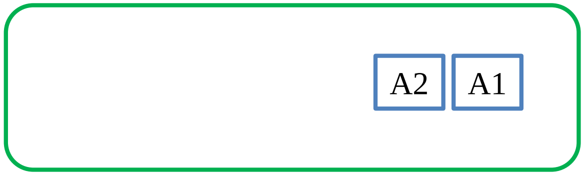
LIFO behavior

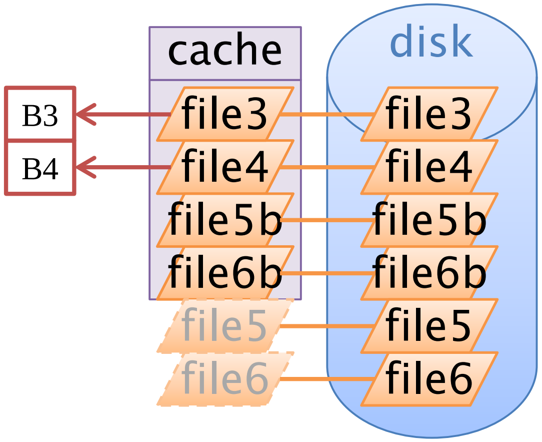
Workflow DAG



NodeQueue

Last First





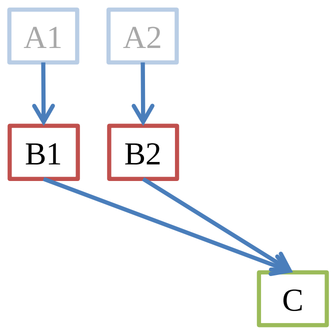
Core allocation

core1	core2
A6	A5
B6	B5
A4	A3
B4	B3

time

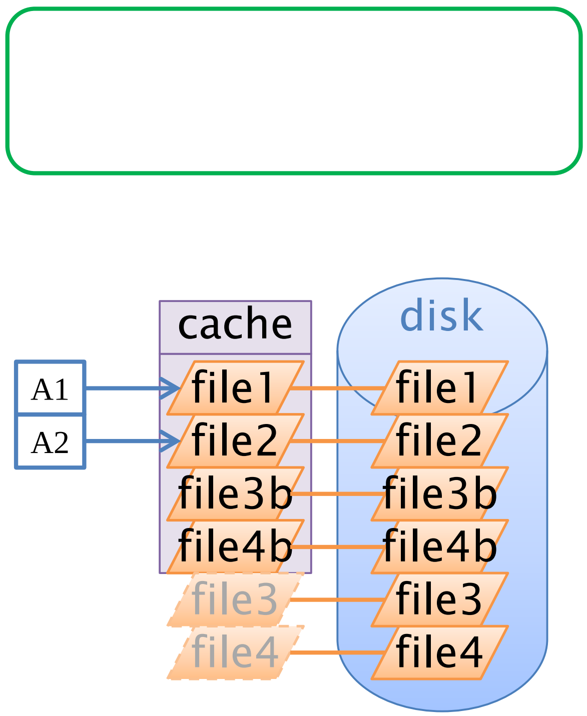
LIFO behavior

Workflow DAG

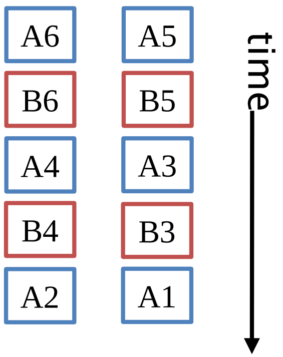


NodeQueue

Last First



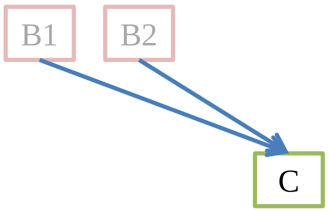
Core allocation



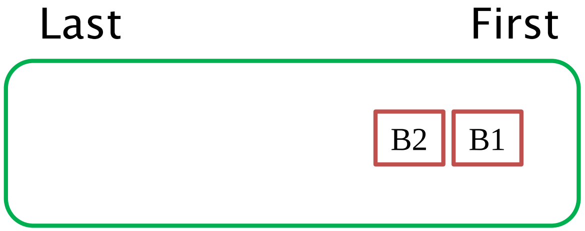
core1	core2
A6	A5
B6	B5
A4	A3
B4	B3
A2	A1

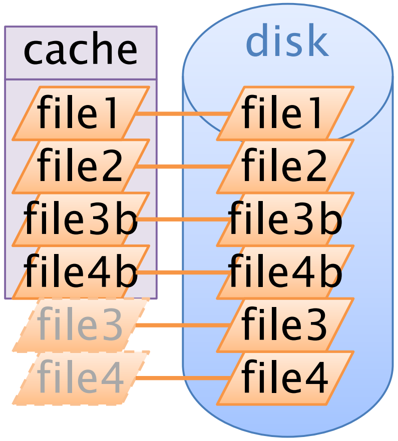
LIFO behavior

Workflow DAG

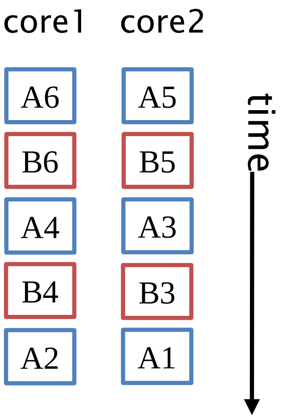


NodeQueue





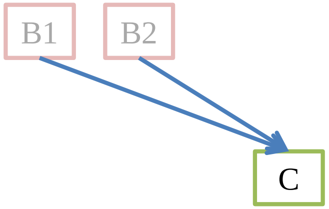
Core allocation



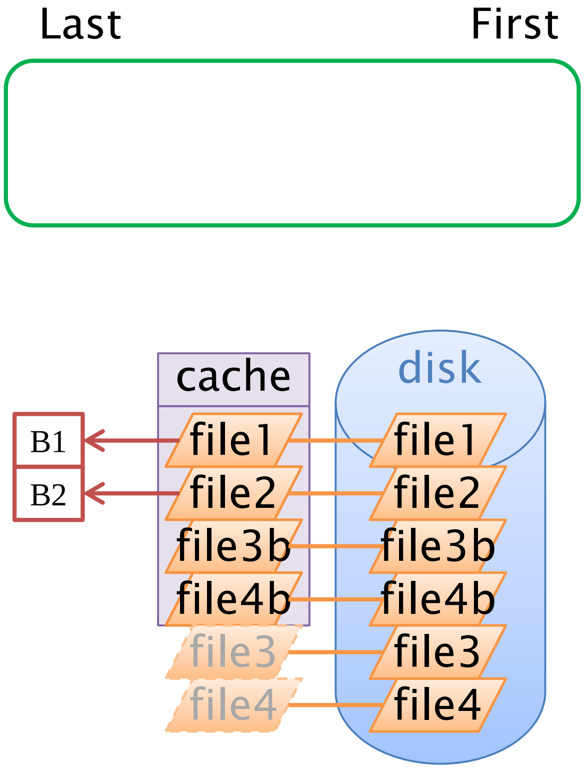
core1	core2
A6	A5
B6	B5
A4	A3
B4	B3
A2	A1

LIFO behavior

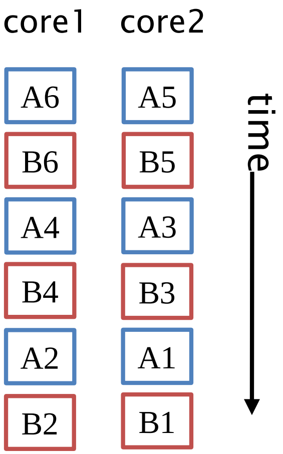
Workflow DAG



NodeQueue

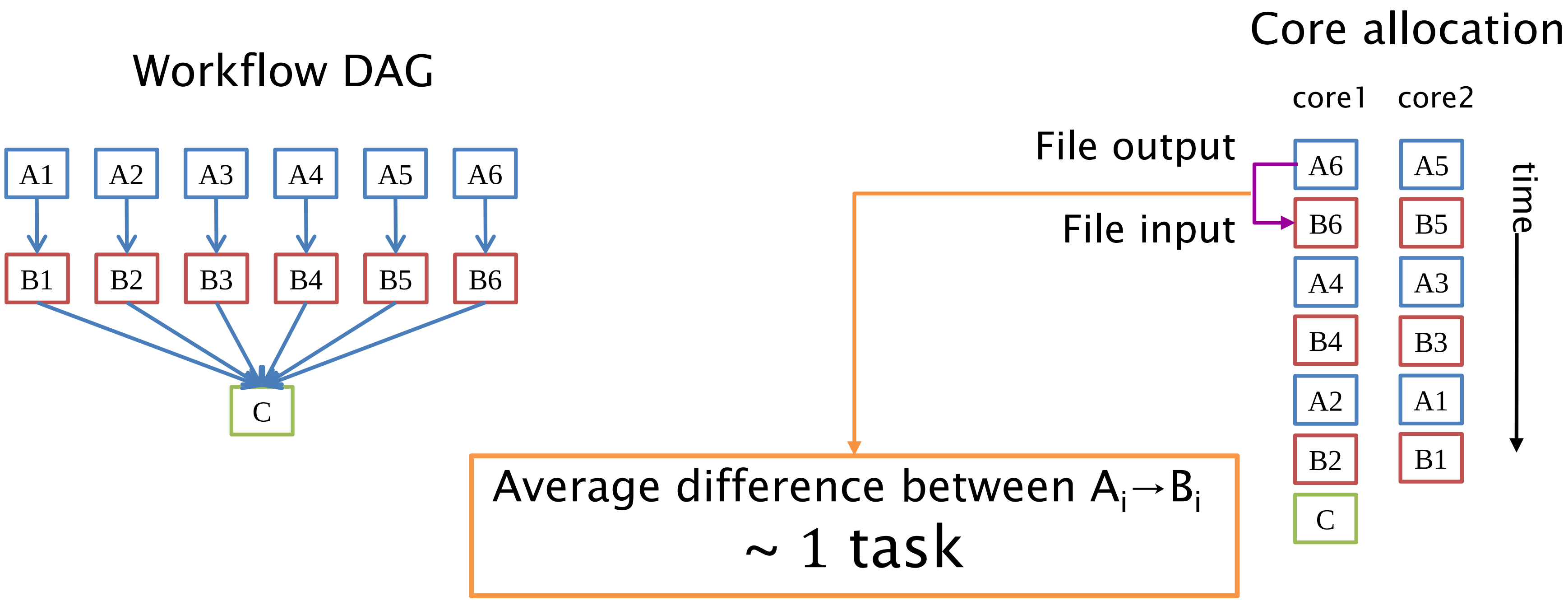


Core allocation



core1	core2
A6	A5
B6	B5
A4	A3
B4	B3
A2	A1
B2	B1

LIFO behavior

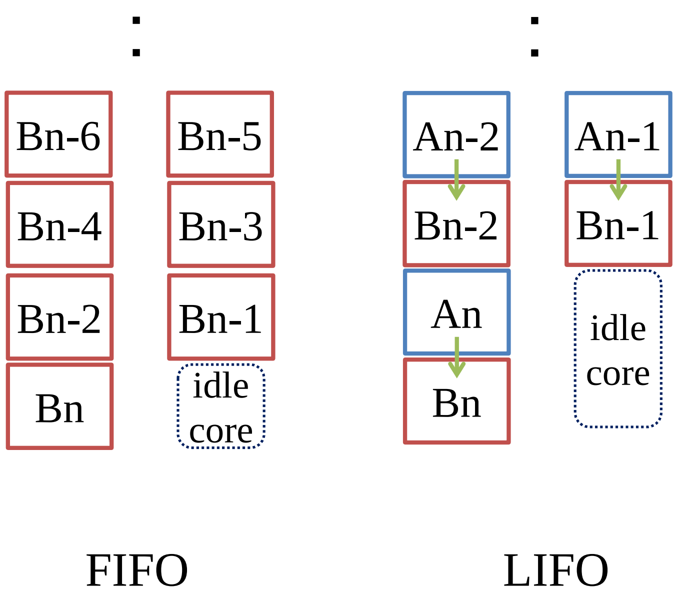


Average difference between $A_i \rightarrow B_i$
~ 1 task

Trailing Task Problem

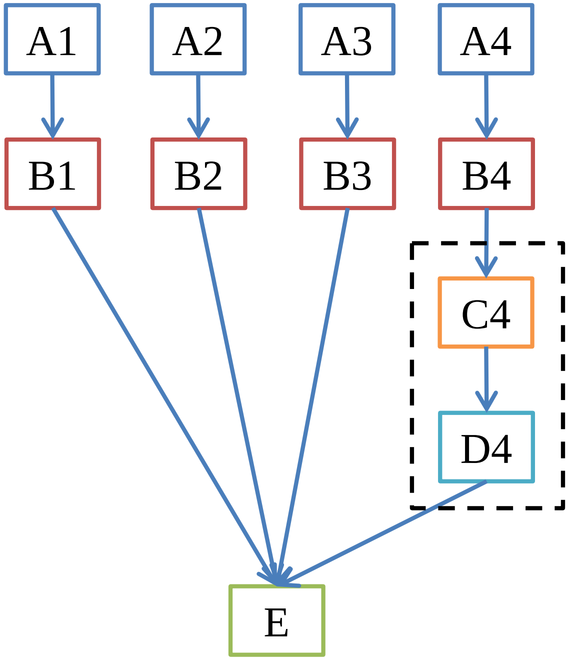
Armstrong et al. MTAGS 2010

- Idle cores in final phase
  - FIFO: max Task B
  - LIFO: max Task A+B
- FIFO has advantage in trailing task problem



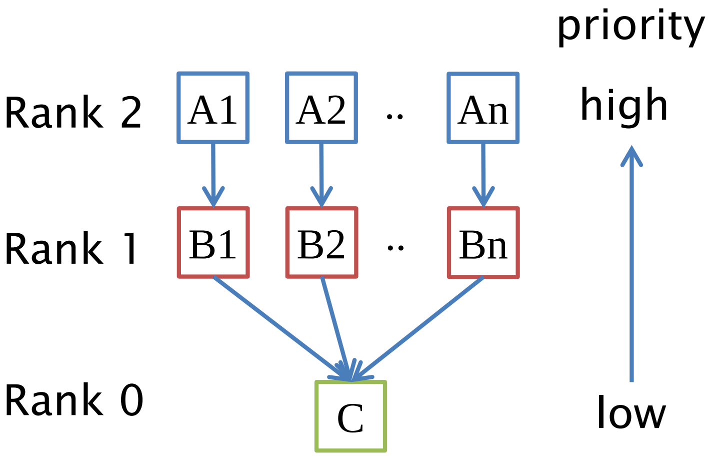
FIFO for Trailing Task Problem?

- FIFO is better for trailing task problem than LIFO.
- FIFO's weak point
  - C4 and D4 can be trailing tasks.
  - A4 and B4 should have higher priority than A1-3, B1-3.



Highest Rank First (HRF)

- Rank:
  - Distance from the last task
  - Same as "upward rank" in HEFT algorithm except uniform task cost
- Highest Rank First (HRF)
  - Priority to higher-rank tasks
  - (FIFO: "downward rank")
  - Solution for Trailing Tasks Problem
  - Bad for Disk Cache



Proposed Methods

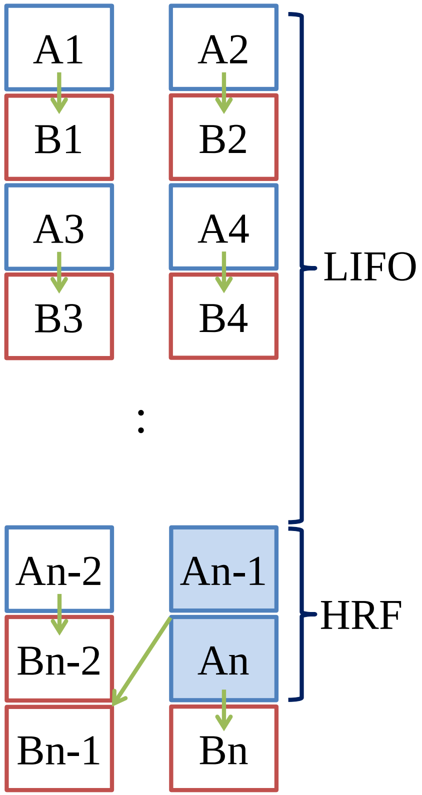
- (1) LIFO + HRF
- (2) Rank Equilization + HRF

Proposed method (1) LIFO+HRF

Nc : # cores/node

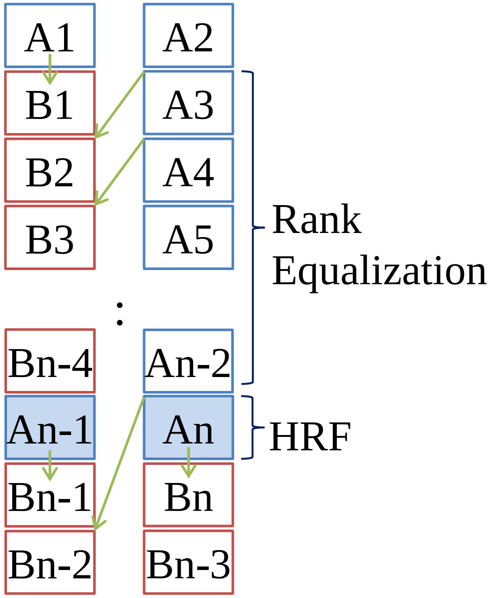
Nr : # tasks in the highest rank

- Algorithm:
  - LIFO if Nr > Nc
  - HRF if Nr ≦ Nc



Proposed method (2)
Rank Equalization+HRF

- Purpose：
  - Overlap compute and I/O
  - Task A: Compute intensive
  - Task B: I/O intensive

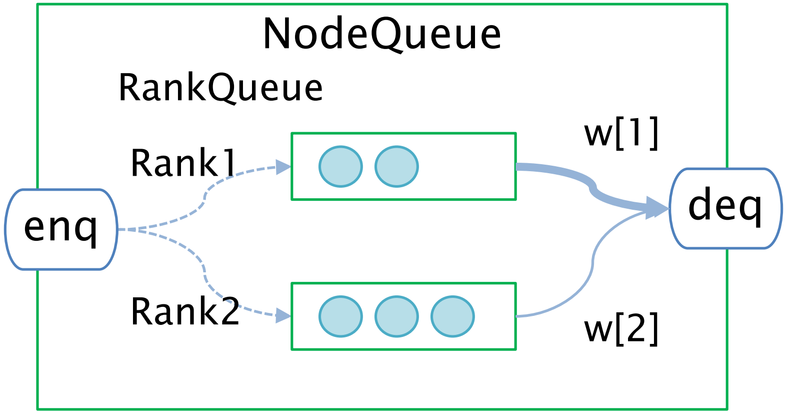


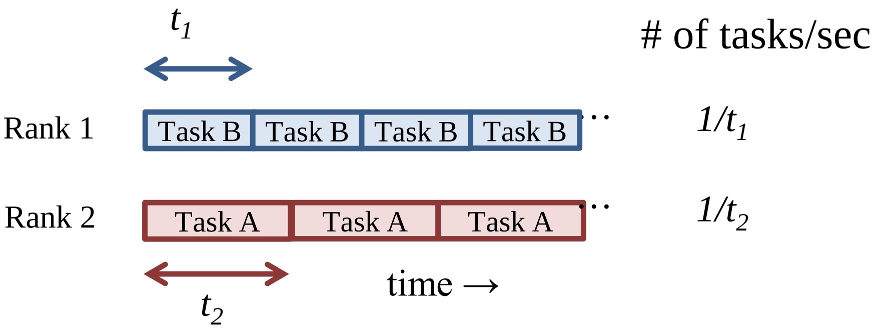
Rank Equalization

- Nr > Nc:
  - Enq:
    - Store tasks to RankQueue
  - Deq:
    - Select RankQueue with different frequency:

$$w[r] = 1/(\text{average execution time}) \propto \text{task invocation frequency}$$

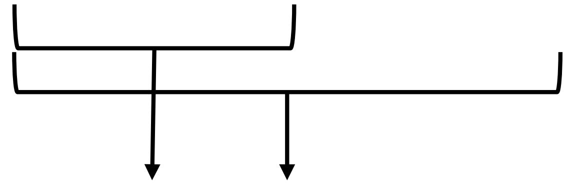
    - Deq a task in LIFO policy.
- Nr ≦ Nc
  - HRF





Summary of Scheduling methods

	FIFO	HRF	LIFO	Rank Eq. (w/ LIFO)
Disk Cache	×	×	◎	○
Trailing Task	○	◎	×	×
Rank Overlap	×	×	×	○



LIFO+HRF　　Rank Eq.+HRF

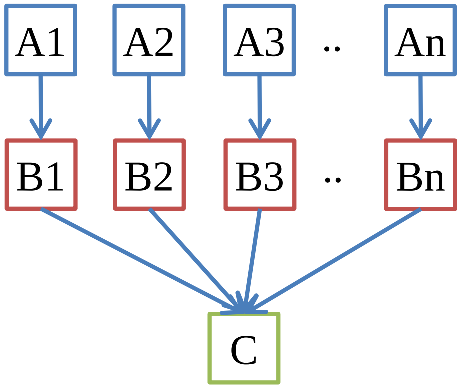
Performance Evaluation

Evaluation Environment

Cluster	InTrigger Tohoku site
CPU	Intel Xeon E5410 2.33GHz
Main Memory	32 GiB
# of cores / node	8
Max # of compute node	12
Network	1Gb Ethernet
OS	Debian 5.0.4
Gfarm	ver. 2.5.8.6
Ruby	ver. 2.1.1
Pwrake	ver. 0.9.9.1

Evaluation-1: “Copyfile” workload

- “Copyfile”: I/O intensive workload
  - Load an input file to main memory and write it to an output file.
- Workflow DAG →
  - Task A,B = “Copyfile” program

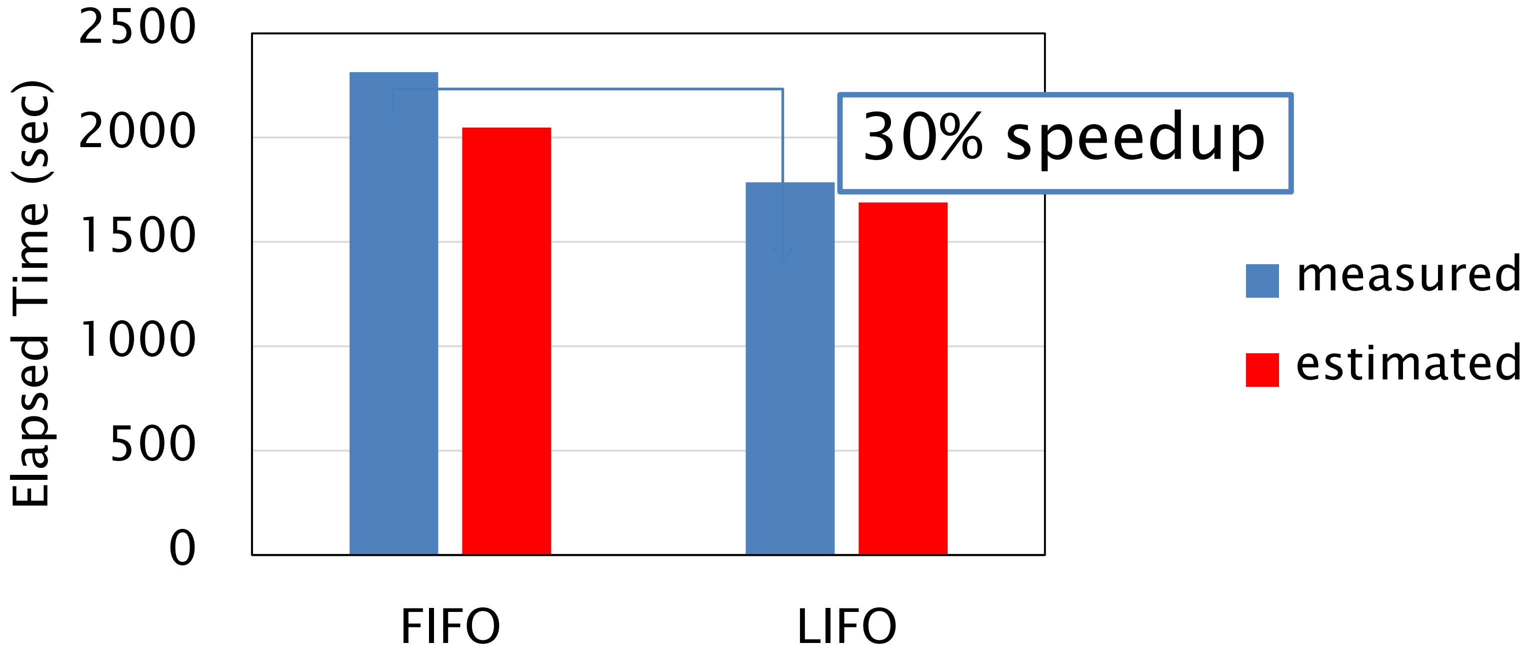


- Scheduling
  - FIFO, LIFO
  - (no +HRF because of 1 core experiment)

Input file		Mem
number	n=100	
one file size	3 GiB	< 32 GiB
total size	300 GiB	> 32 GiB

Used nodes	
10 nodes	1 core/node

Elapsed Time of Copyfile workflow



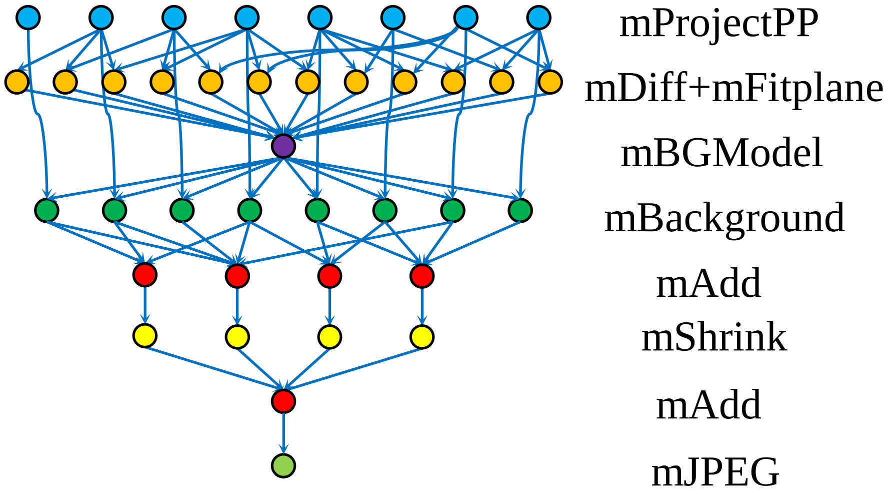
- Estimated time = I/R + O/W
  - I, O = Input, Output filesize
  - R, W = Read, Write bandwidth (depends on access target)

Evaluation-2: Montage Wowkflow

- Astronomy image processing
- Number of cores : 96

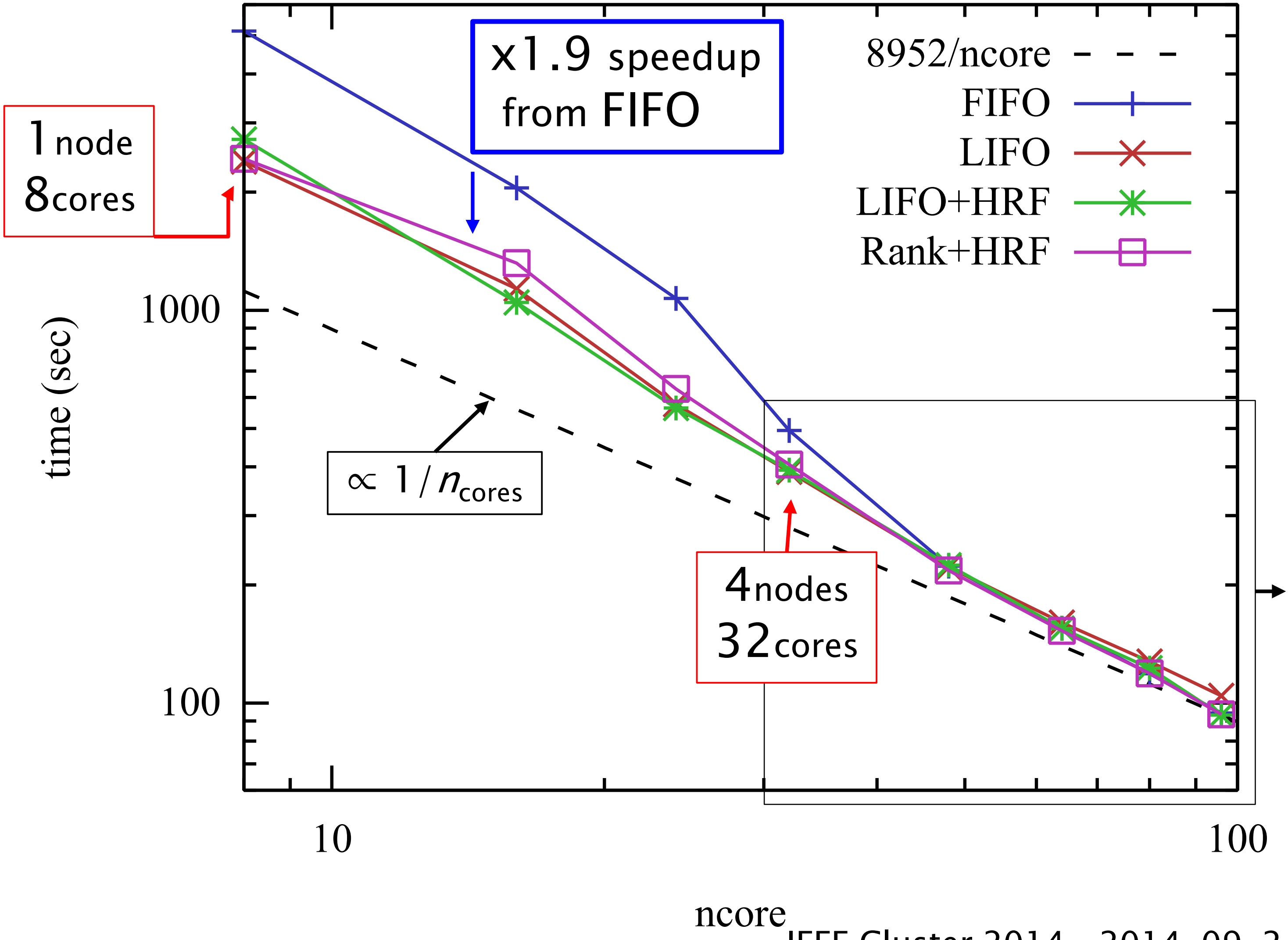
  (12 nodes x 8 cores)

DAG



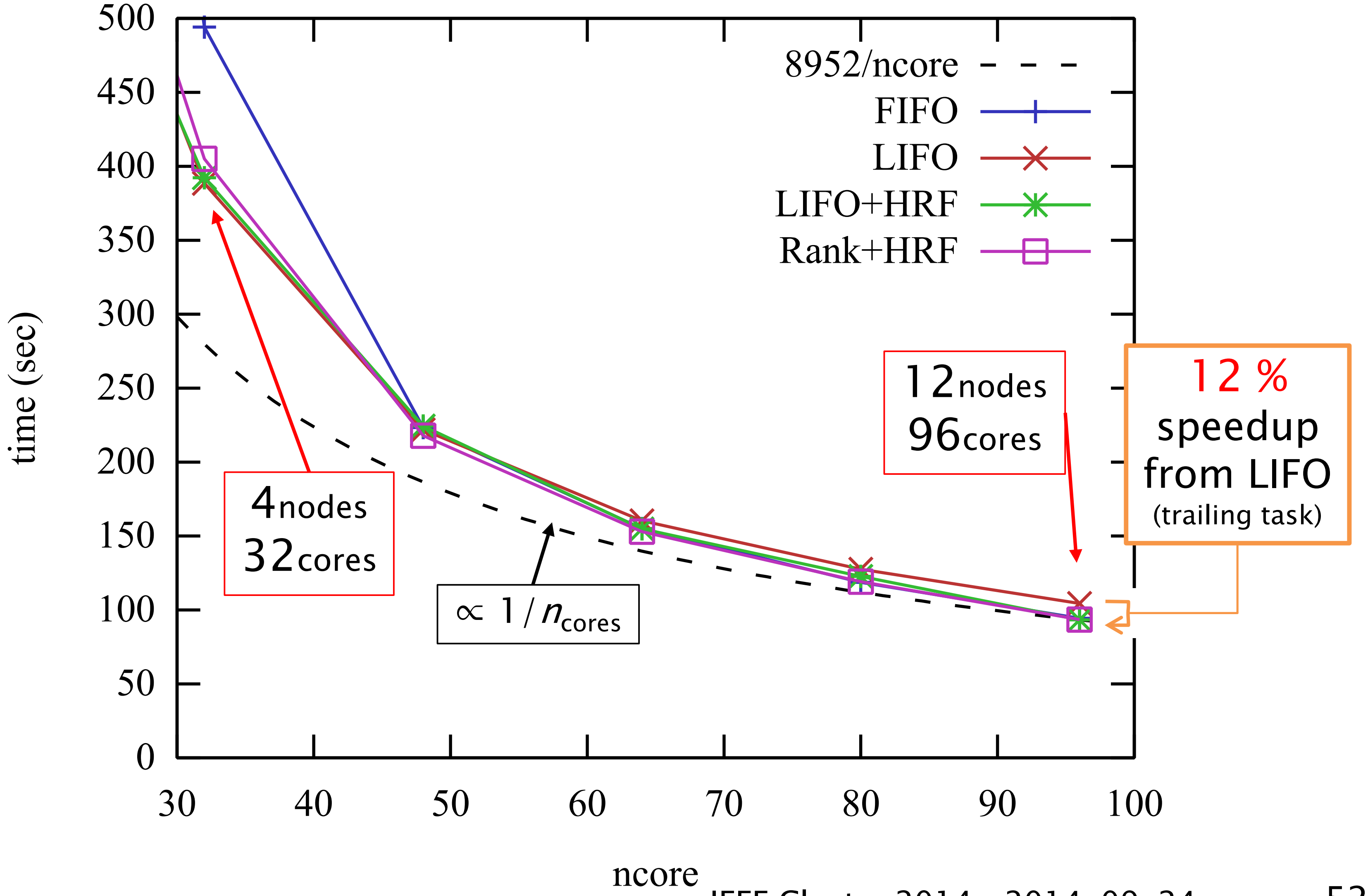
Input file	SDSS DR7
# of input files	421
Size of one input file	2.52 MB
Size of total input files	1061 MB
# of intermediary files	4720
Size of intermediary files	63.5 GB
# of tasks	2707

Measurement of Strong Scaling
(1–12 nodes, Logarithmic)



→ Next Slide

Measurement of Strong Scaling
(4–12 nodes, Linear)



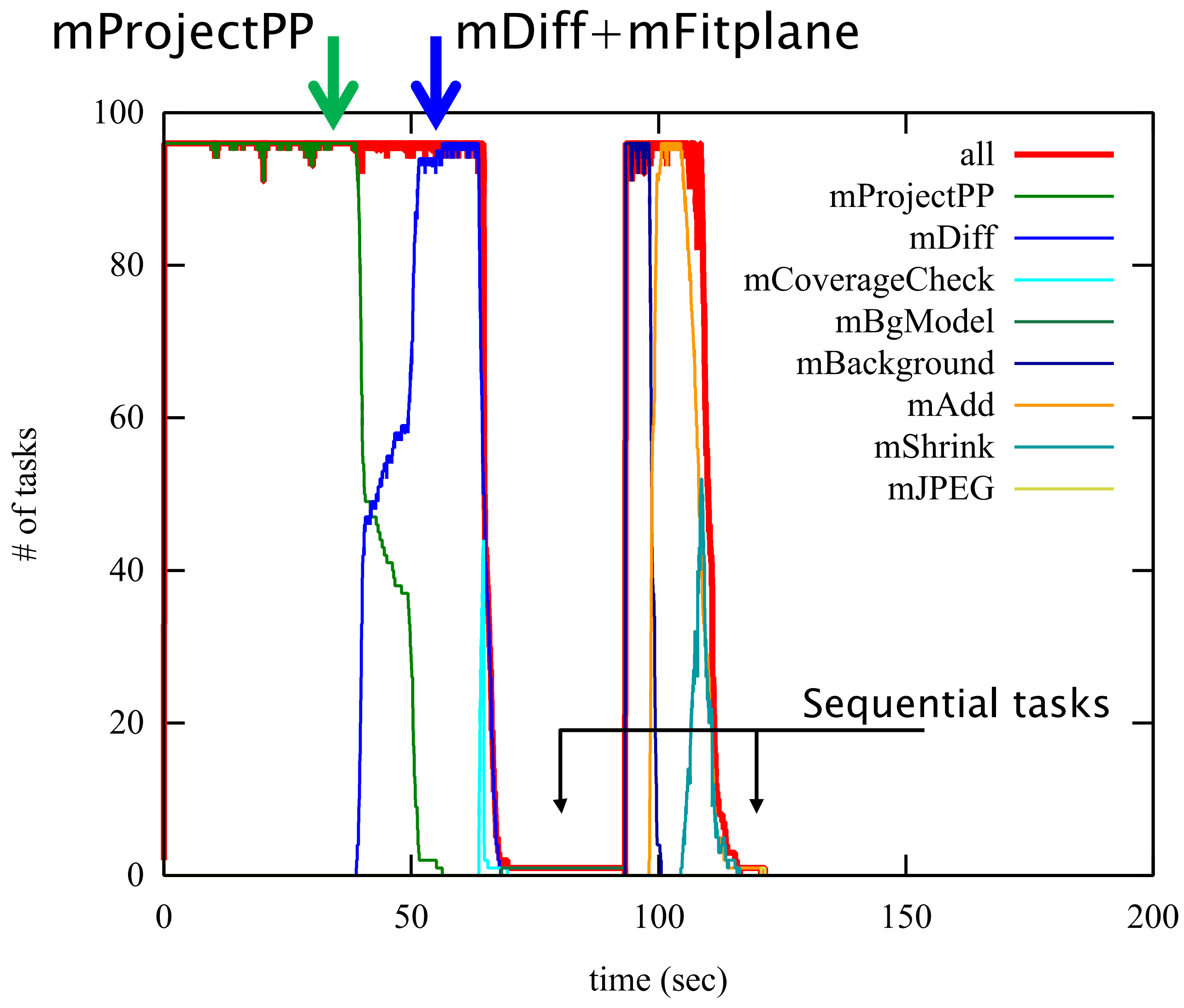
4nodes 32cores

12nodes 96cores

$\propto 1 / n_{\mathrm{cores}}$

12 % speedup from LIFO (trailing task)

FIFO order Task Scheduling



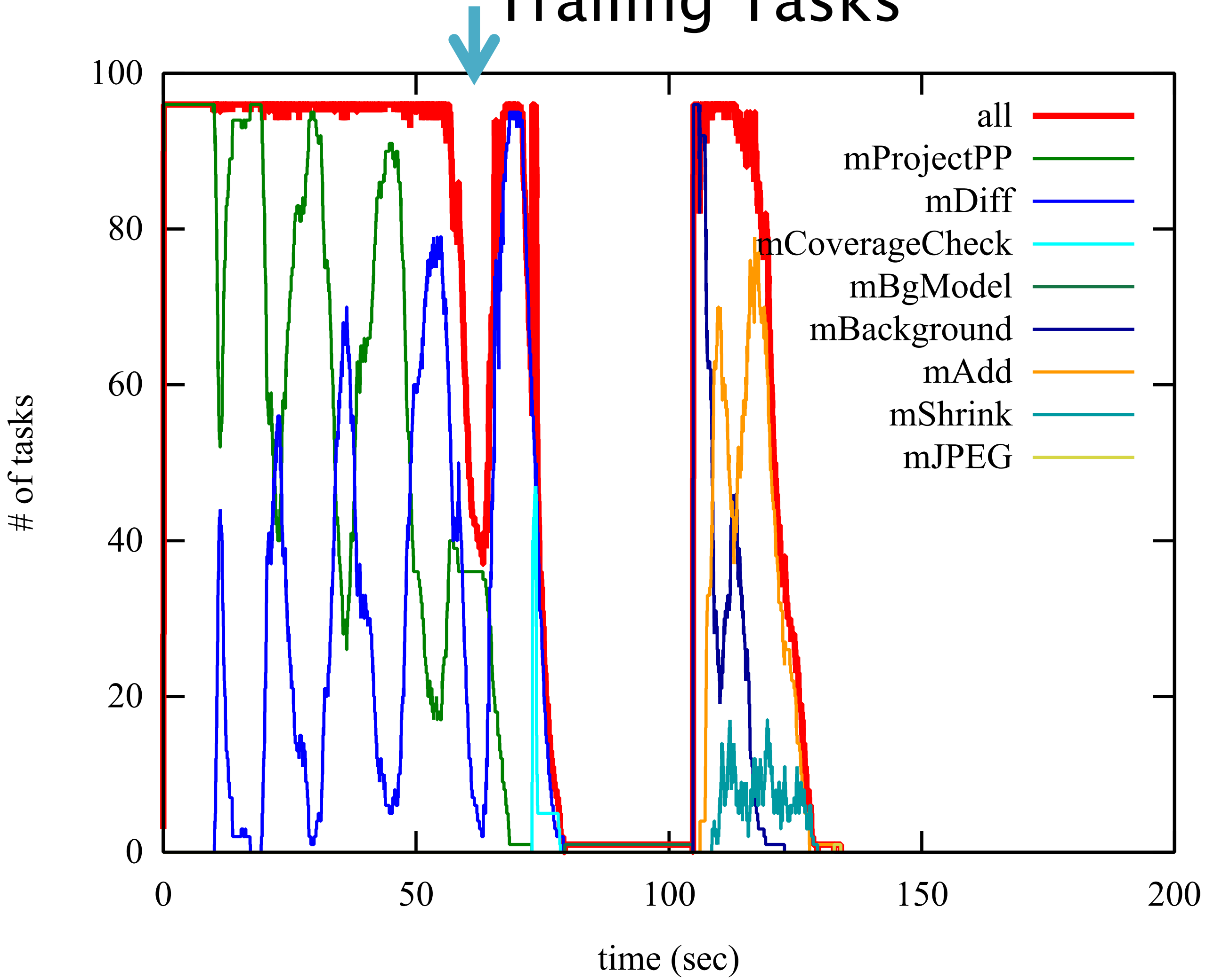
mProjectPP

mDiff+mFitplane

Sequential tasks

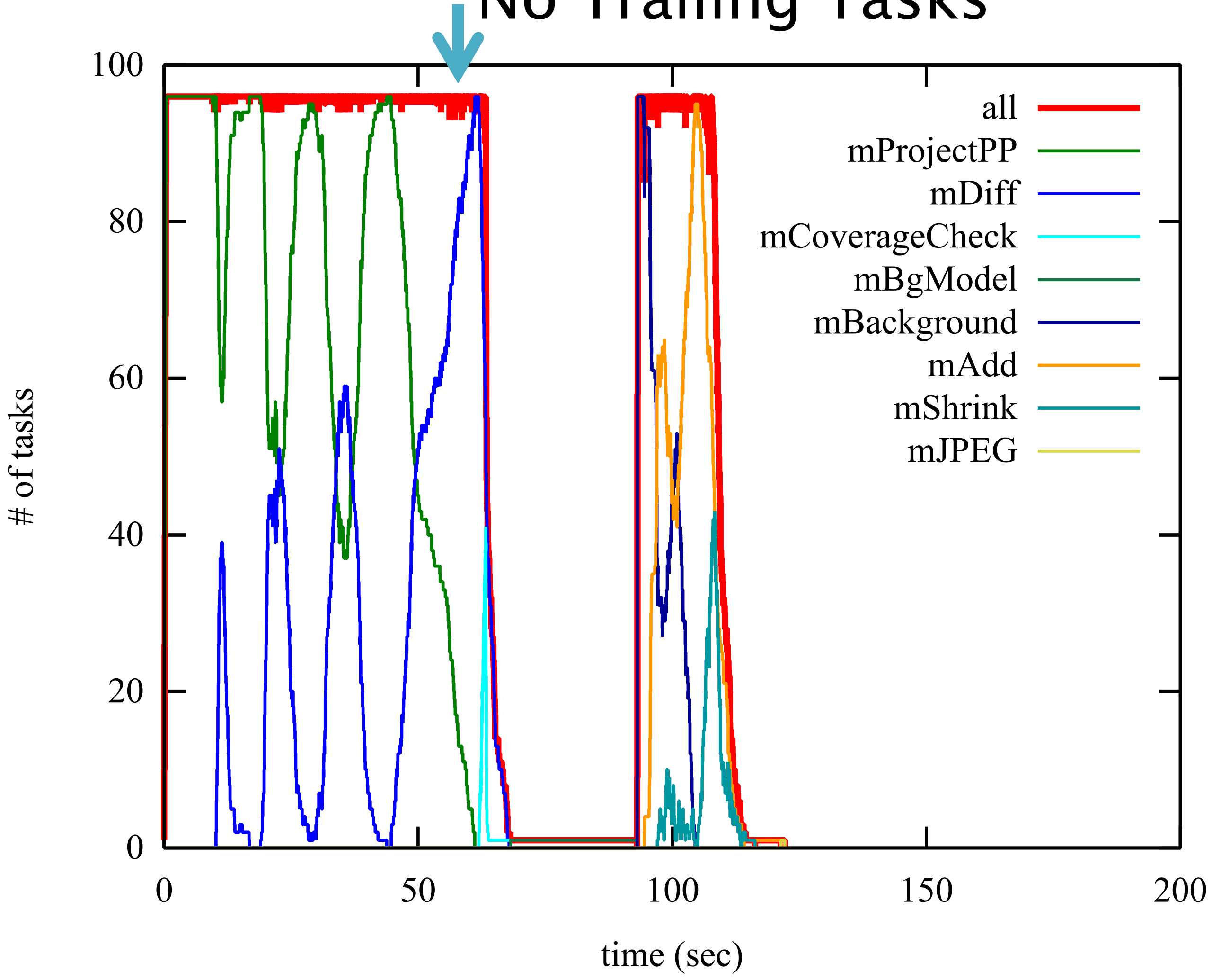
LIFO order Task Scheduling

Trailing Tasks

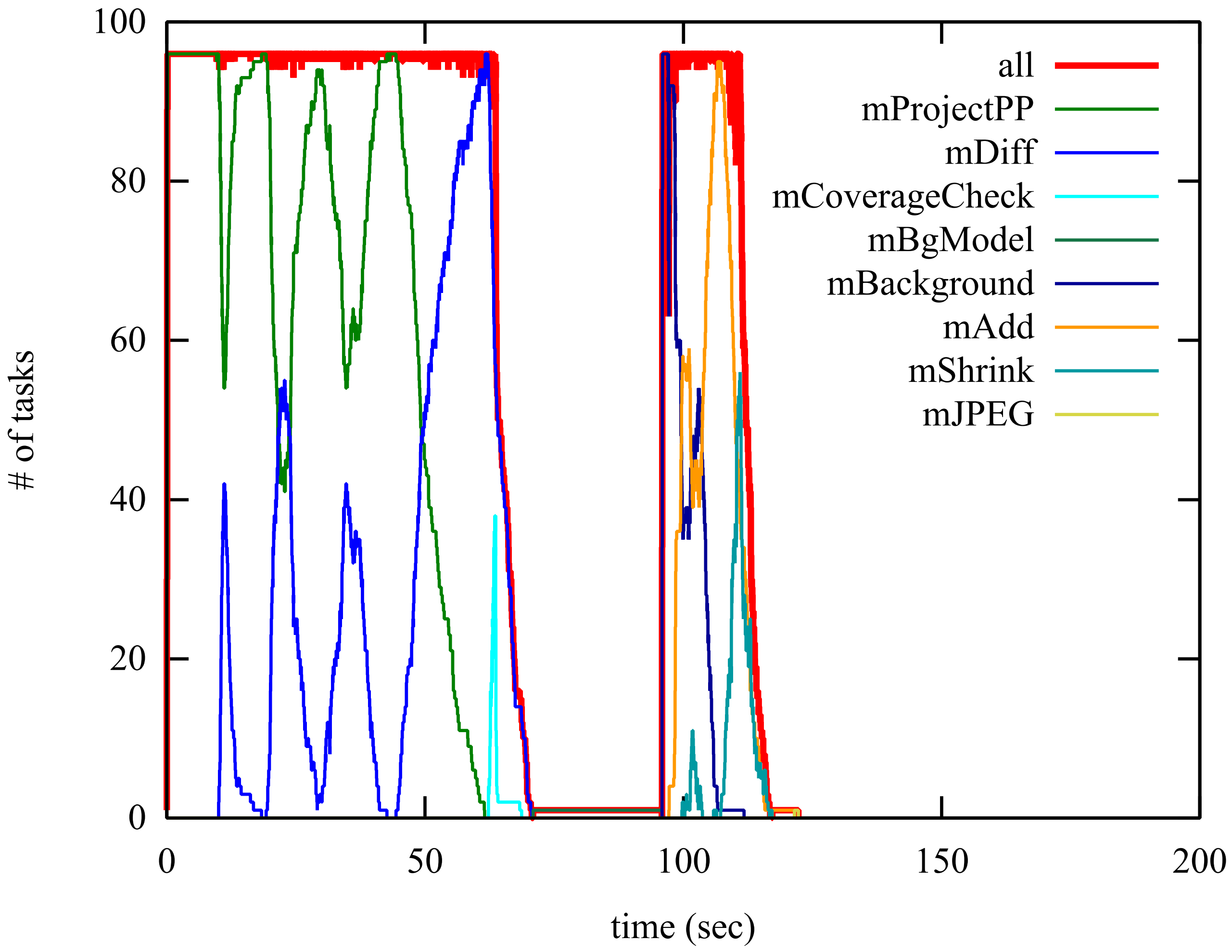


LIFO+HRF

No Trailing Tasks



LIFO+Rank Equalization

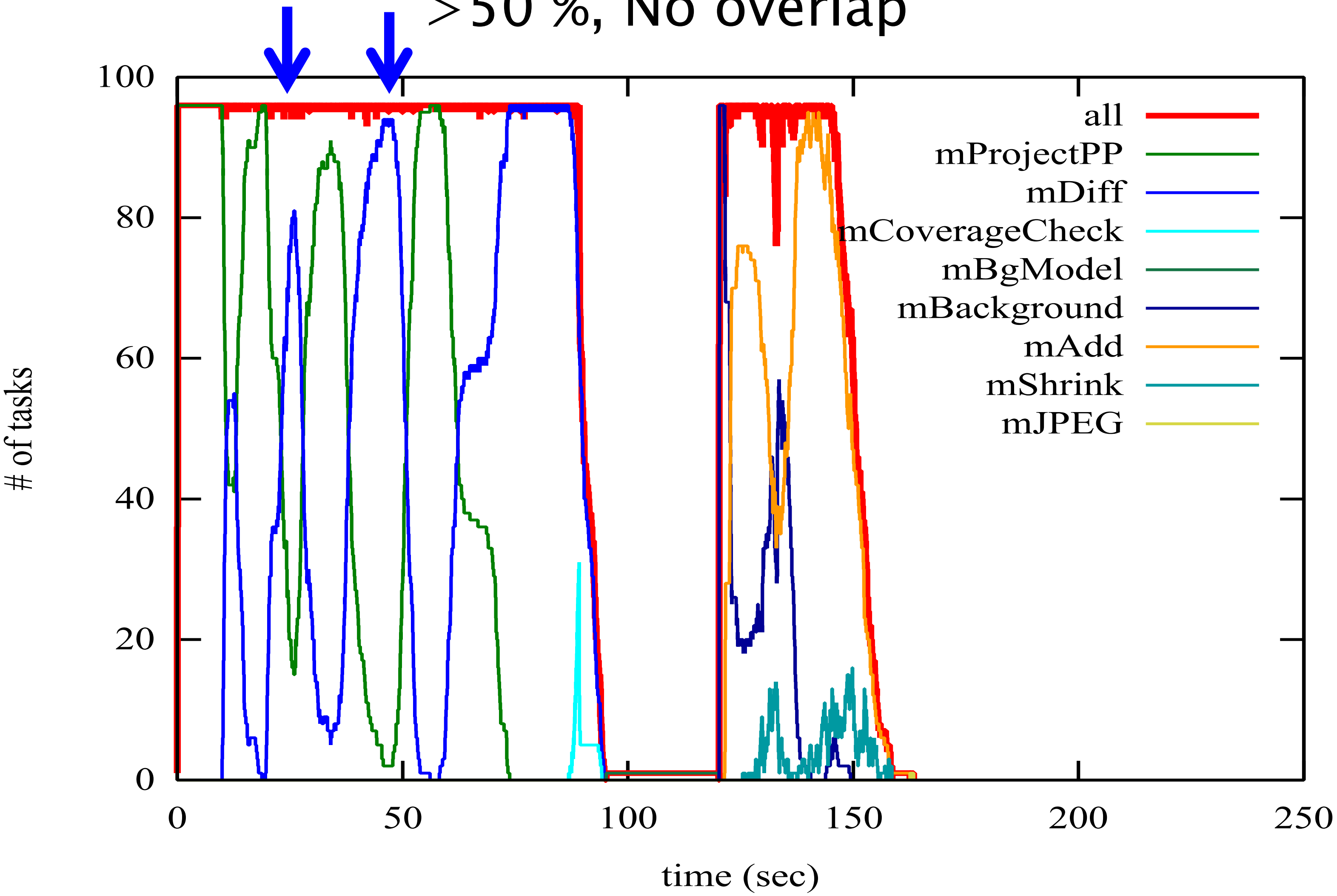


Result of 12-node experiment

- LIFO is the worst
  - due to trailing task problem
- Other methods are comparable.
- Why FIFO is good?
  - Data size per node < Cache size
    - Benefit of cache even in FIFO case.
- Why Rank Equalization is not better?
  - In LIFO+HRF case, mDiff is always less than 50% core utilization.
    - mDiff always overlaps with mProjectPP.

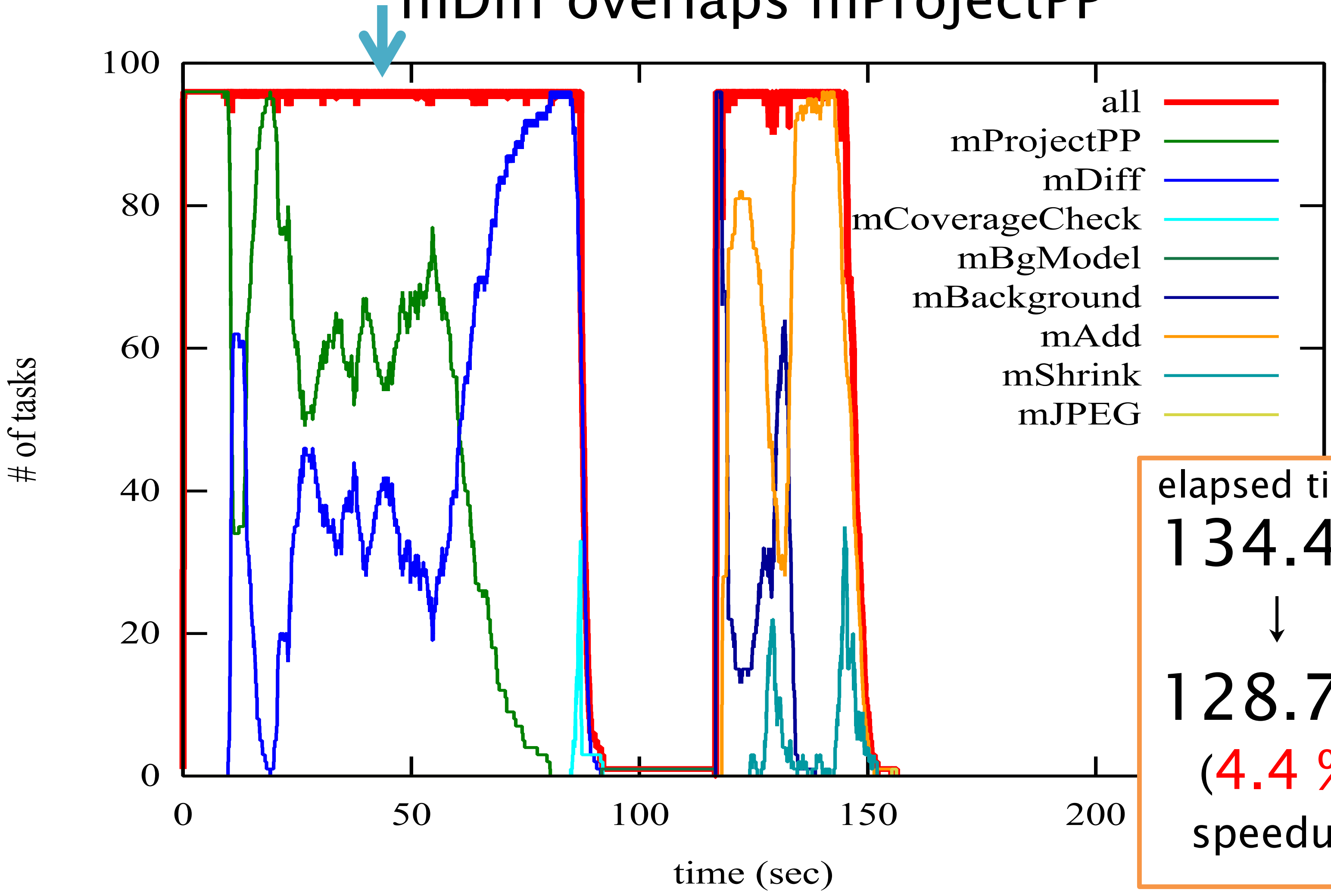
LIFO+HRF (non MCGP)

>50 %, No overlap



LIFO+Rank Equalization (non-MCGP)

mDiff overlaps mProjectPP



elapsed time:
134.4 s
↓
128.7 s
(4.4 %
speedup)

Related Work 1: Workflow System

- Swift + Falkon + data diffusion
  - Swift (Wilde et al. 2011)
    - Workflow language
  - Falkon (Raicu et al. 2007)
    - Task throughput (1500 tasks/sec)
  - data diffusion (Raicu et al. 2008)
    - Staging file management + Task scheduling
- GXP make (Taura et al. 2013)
  - Workflow system based on GNU make
  - Dispatches tasks invoked by GNU make

Related Work 2: Workflow Scheduling

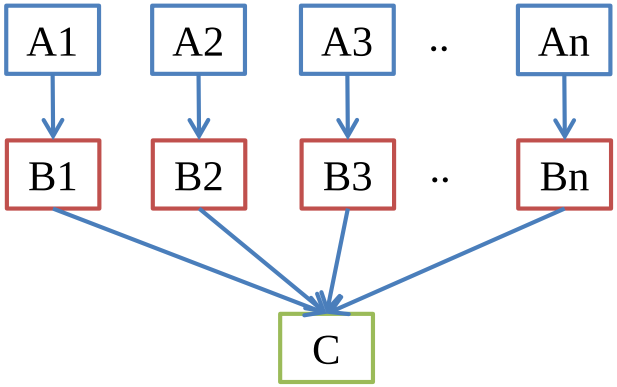
- Shankar and DeWitt (HPDC 2007)
  - studied DAG-based data-aware workflow scheduling for the Condor system.
  - focused on cached data on a local disk in order to avoid data movement.
- Armstrong et al. (MTAGS 2010)
  - discussed trailing task problem.
  - proposed tail-chopping approach.

Conclusion

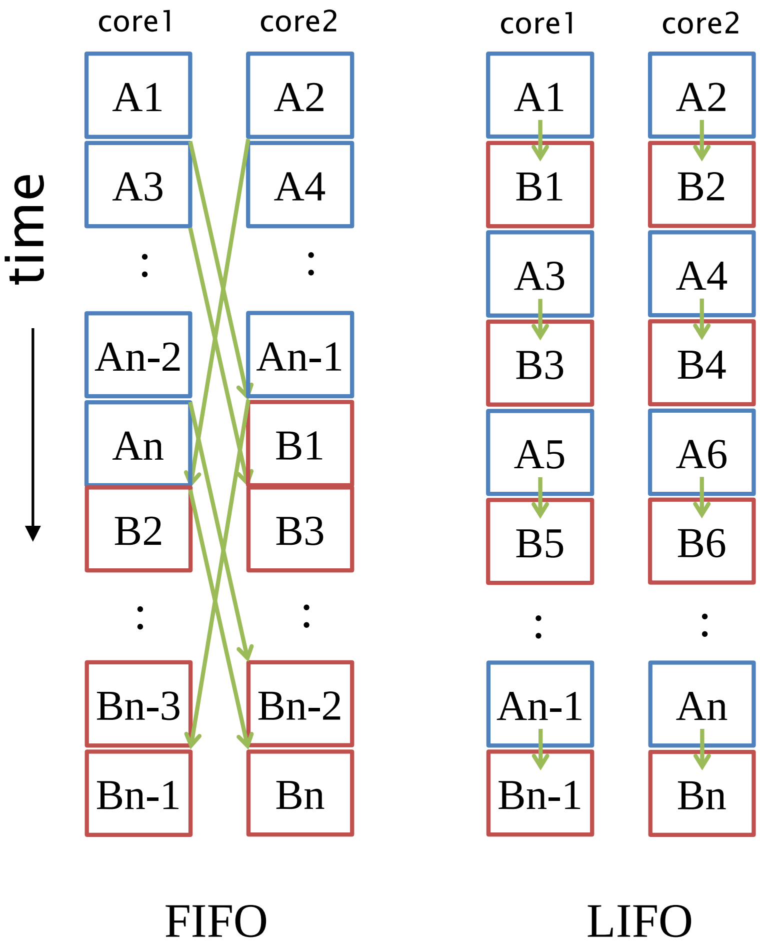
- Objective :
  - Disk cache aware task scheduling for Data-intensive and Many-task workflow.
- Proposed methods :
  - LIFO+HRF:
  - Rank Equalization+HRF:
- Evaluation :
  - Copyfile workflow:
    - LIFO: ~30% speedup
  - Montage Workflow:
    - LIFO: 1.9x speedup
    - HRF: ~12% speedup
    - Rank Equalization: ~4% speedup

Backup slides

Cache FIFO and LIFO

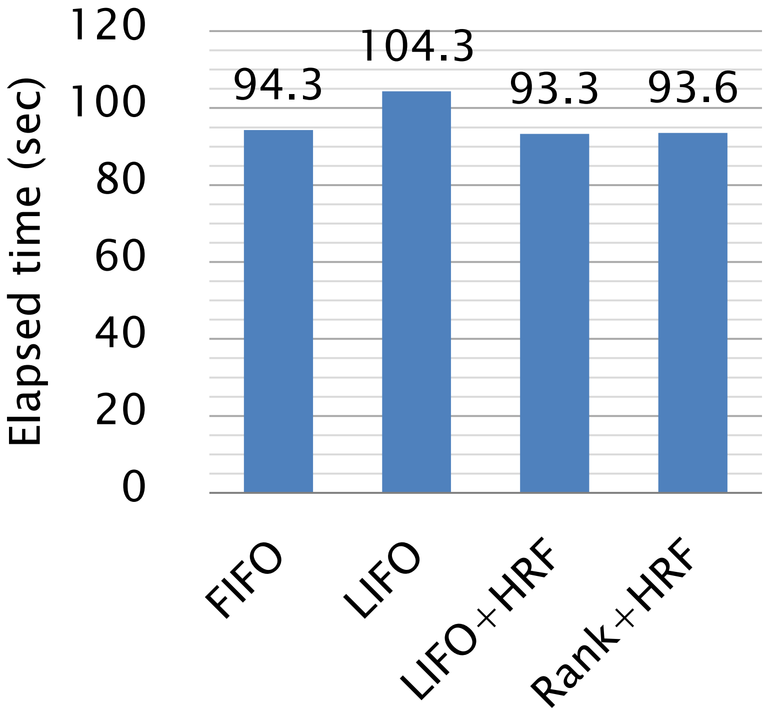


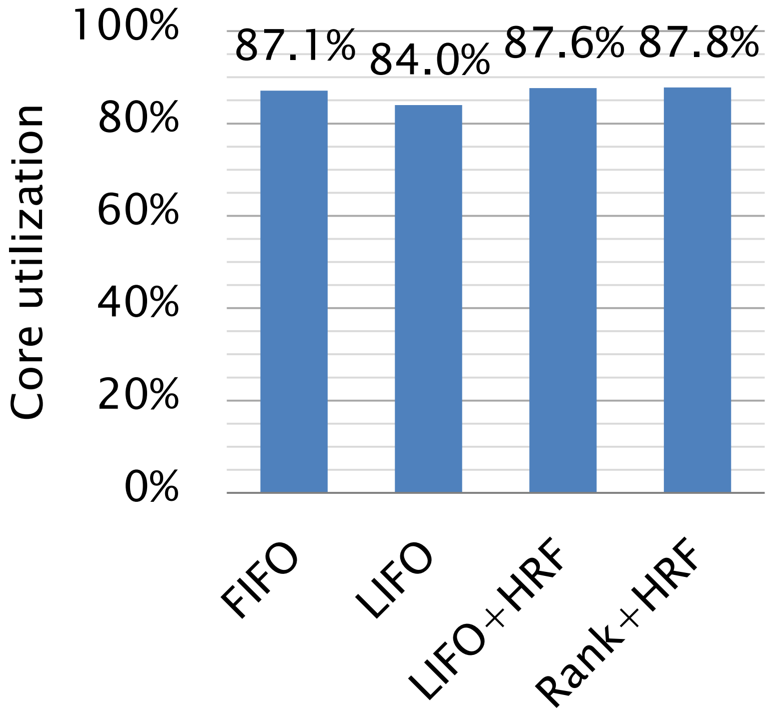
- A_i→B_i difference (average)
  - FIFO： n/2 tasks
  - LIFO： 1 tasks
- LIFO is good for Cache utilization



Elapsed time and Core utilization (12 nodes)

- t_{elap} = Elapsed time of workflow
- t_{cum} = Summated time of all the tasks
- Core utilization = $t_{\mathrm{cum}}/(t_{\mathrm{elap}} n_{\mathrm{cores}})$


Elapsed time (sec)
FIFO
LIFO
LIFO+HRF
Rank+HRF


Core utilization
FIFO
LIFO
LIFO+HRF
Rank+HRF