Workflow Scheduling to Minimize Data Movement Using Multi-Constraint Graph Partitioning

Masahiro Tanaka and Osamu Tatebe

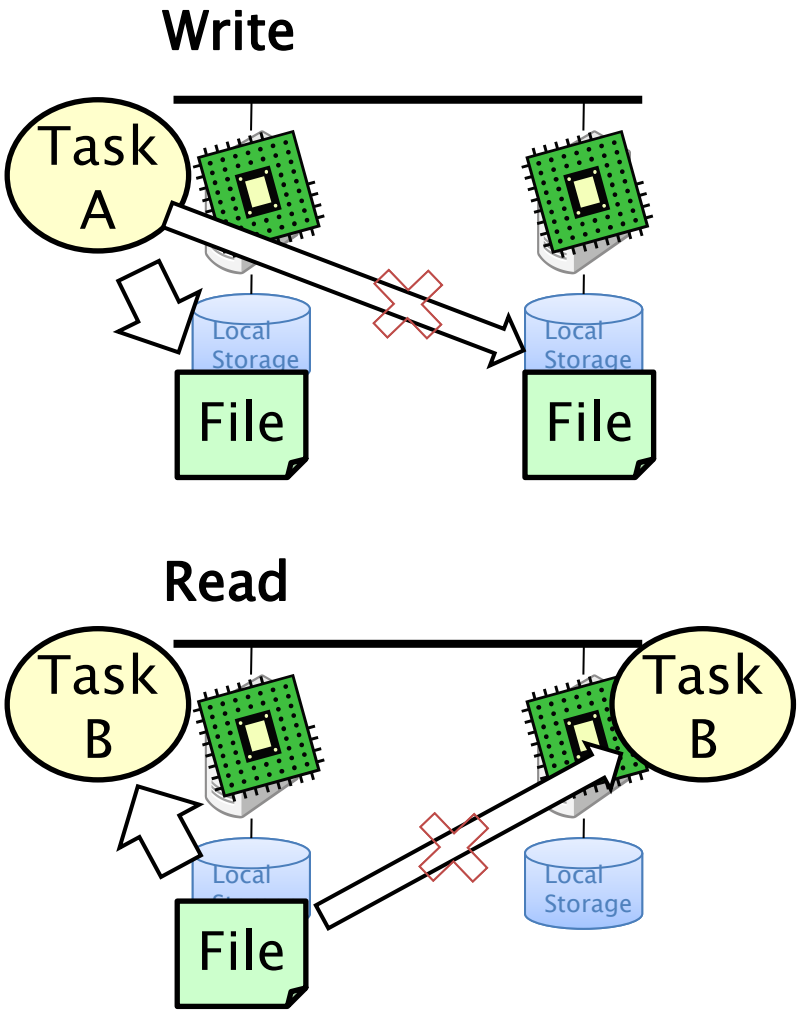
University of Tsukuba, JST/CREST

Outline

- Introduction
  - Workflow Scheduling for Data-Intensive Science
- Proposed Method
  - Workflow Scheduling using MCGP
- Evaluation
- Related Work
- Conclusion
- Future Work

Workflow for Data-Intensive Science

- Large Scientific Data
  - File I/O is a bottleneck
- **Data Locality** is a key
  - **Write a File**
    - **Select local storage**
    - to write output file
    - e.g. Gfarm File System
  - **Read a File**
    - **Assign a task to the node** where input file exits
    - Workflow System

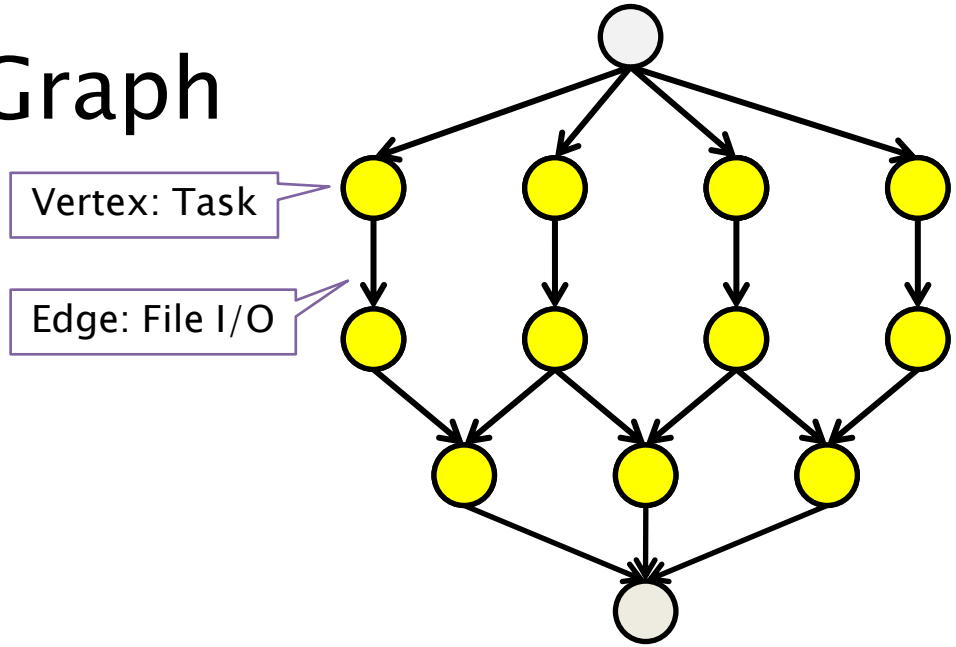


Workflow Scheduling

- Common method: minimize ***Makespan***
  - Makespan = completion time of workflow
  - Many works: e.g. HEFT, Minmin, …
- Our strategy:minimize ***Data Movement***
  - Maximize **Data Locality**
  - Based on Workflow DAG

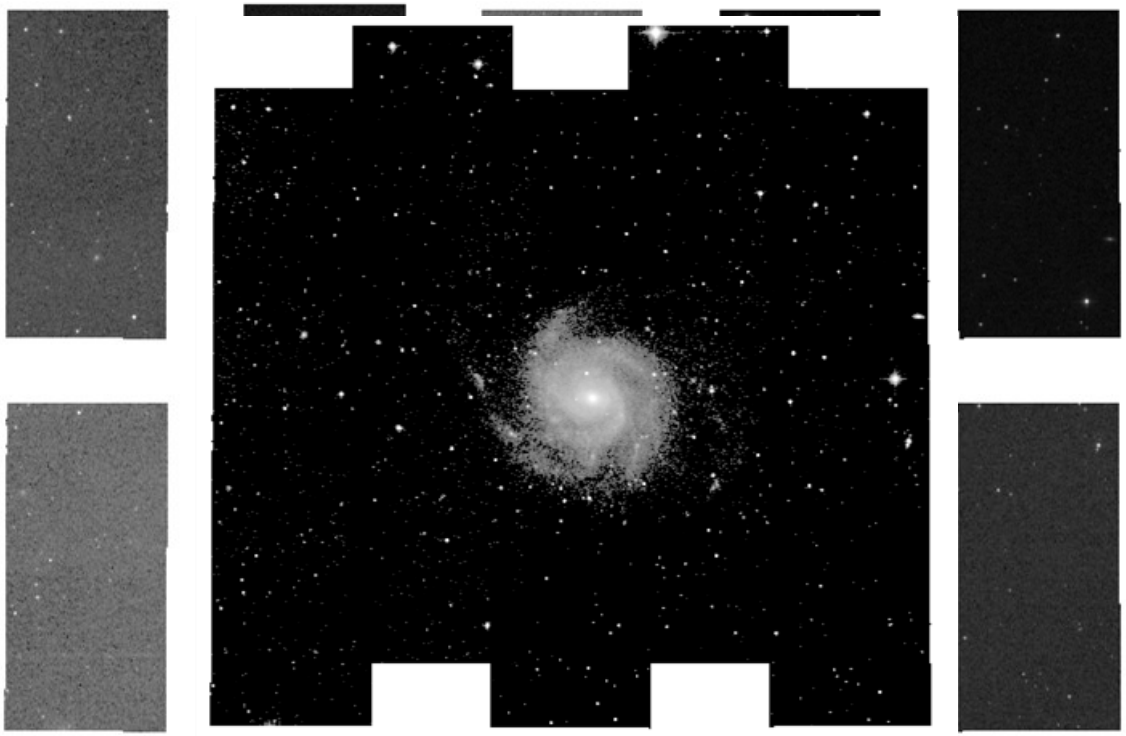
Workflow DAG

- Directed Acyclic Graph
  - Vertex : Task
  - Edge :
    - Dependency,
    - Data (File) I/O
- Scheduling
  - Assign Compute node
  - Order of Execution

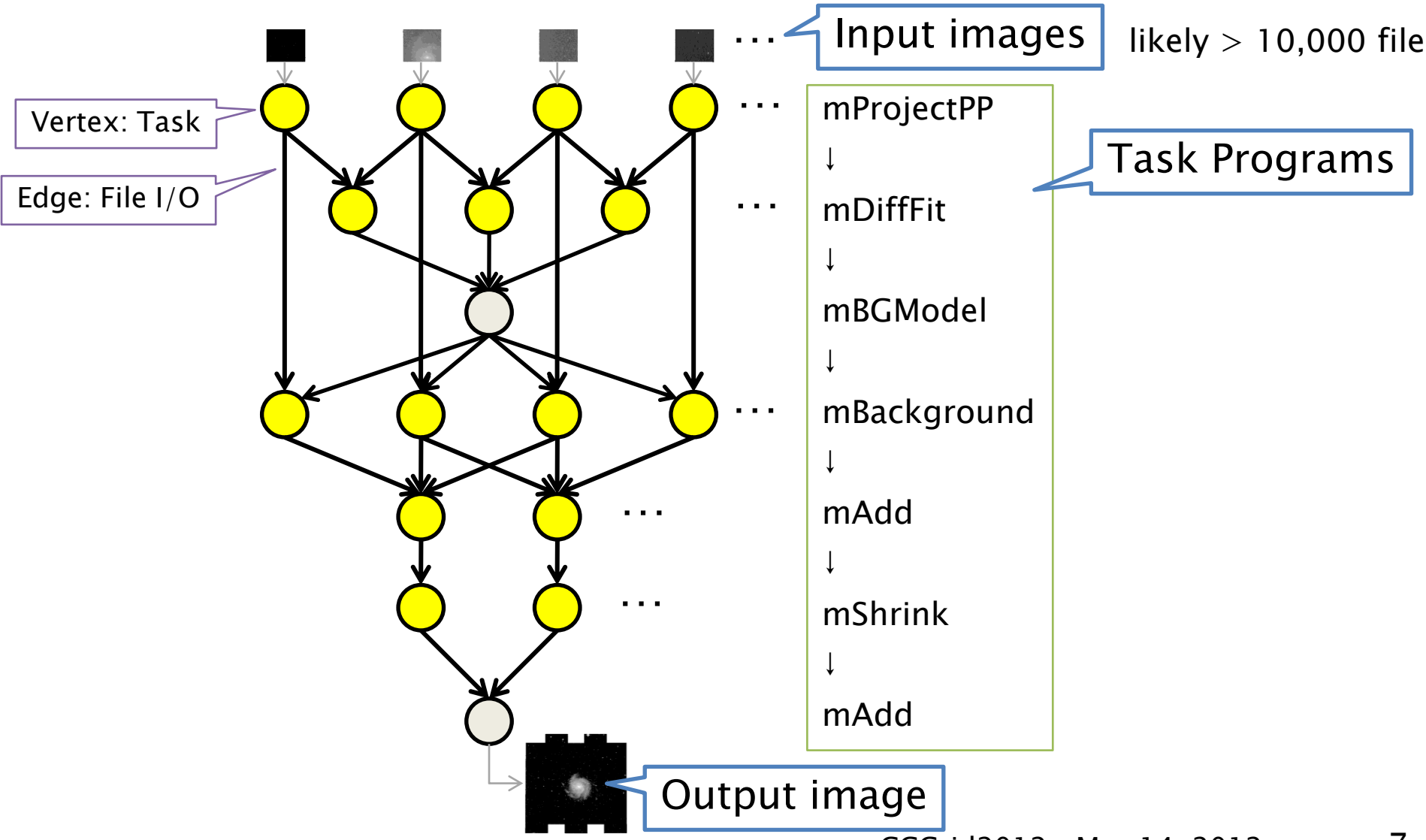


Workflow Example: Montage

- Combine multiple-shots of Astronomical Images and produce a custom Mosaic Image

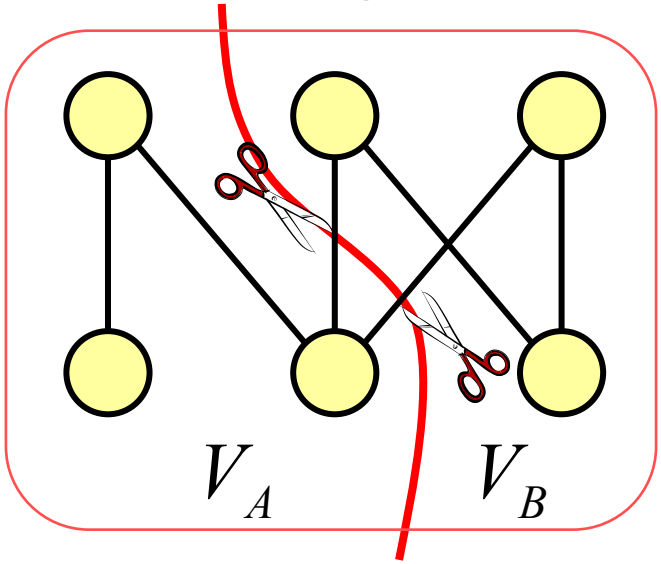


DAG of Montage workflow



Standard Graph Partitioning for Undirected Graph G=(V,E)

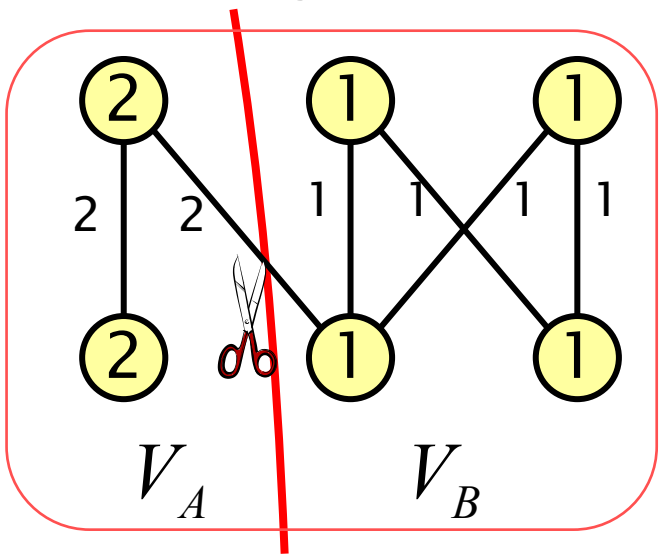
No weight



N of vertices: $|V_A| = |V_B| = 3$

N of edge-cut: $|E_{cut}| = 2$

Weighted



Vertex weight: $\sum_{i \in V_A} w^i = \sum_{j \in V_B} w^j = 4$

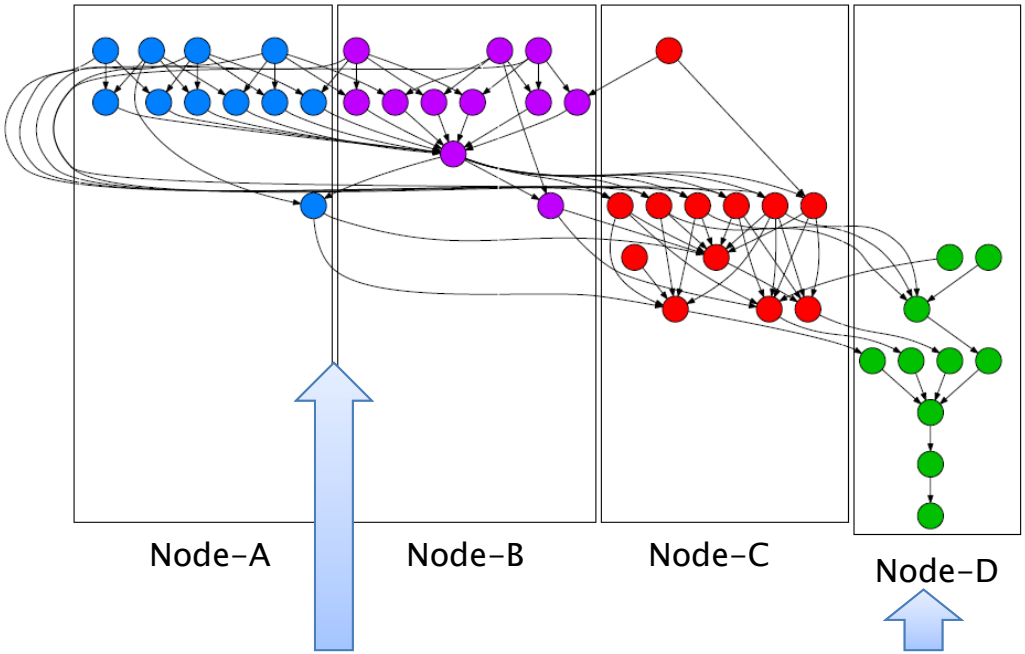
Edge weight: $\sum_{k \in E_{cut}} w^k = 2$

Graph Partitioning ⇔ Task Scheduling

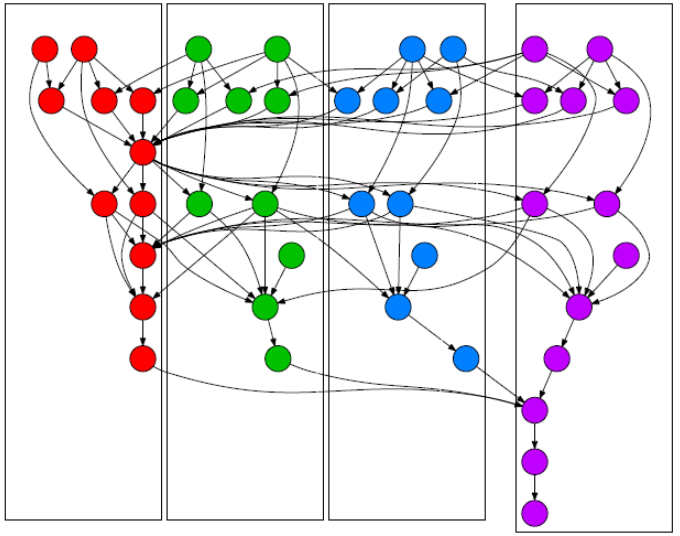
- Vertex-weight ⇔ Computation cost
- Edge-weight ⇔ Communication cost
- Minimize
  - Edge-cut ⇔ Data movement
- Graph Partitioning is used for
  - represent Geometrical relationship
- Q:
  - Graph Partitioning also applicable to DAG?

Graph Partitioning on DAG

Standard Graph Partitioning



Ideal Partitioning for Scheduling



Standard GP is not aware of parallelizable tasks

Proposed Method

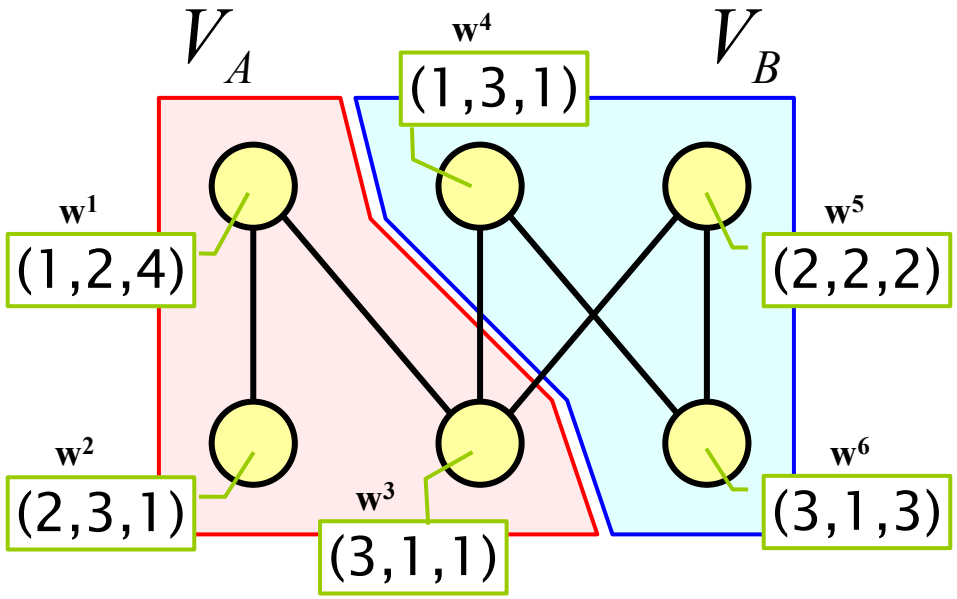
- Apply Multi-Constraint Graph Partitioning (**MCGP**) for workflow scheduling.
- **MCGP**:
  - Studied and Implemented in METIS library
    - Karypis & Kumar (SC98)
  - No report on Workflow DAG
- Our Contribution:
  - Proposal to apply MCGP to Workflow DAG
  - Implementation and Evaluation on real workflow

Outline

- Introduction
  - Workflow Scheduling for Data-Intensive Science
- Proposed Method
  - Workflow Scheduling using MCGP
- Evaluation
- Related Work
- Conclusion
- Future Work

Multi-Constraint Graph Partitioning (MCGP)

Vertex Weight Vectors $\mathbf{w}^i = \left(w_1^i, w_2^i, w_3^i\right)$



V_A V_B

1st dim: $\sum_{i \in V_A} w_1^i = \sum_{j \in V_B} w_1^j = 6$

2nd dim: $\sum_{i \in V_A} w_2^i = \sum_{j \in V_B} w_2^j = 6$

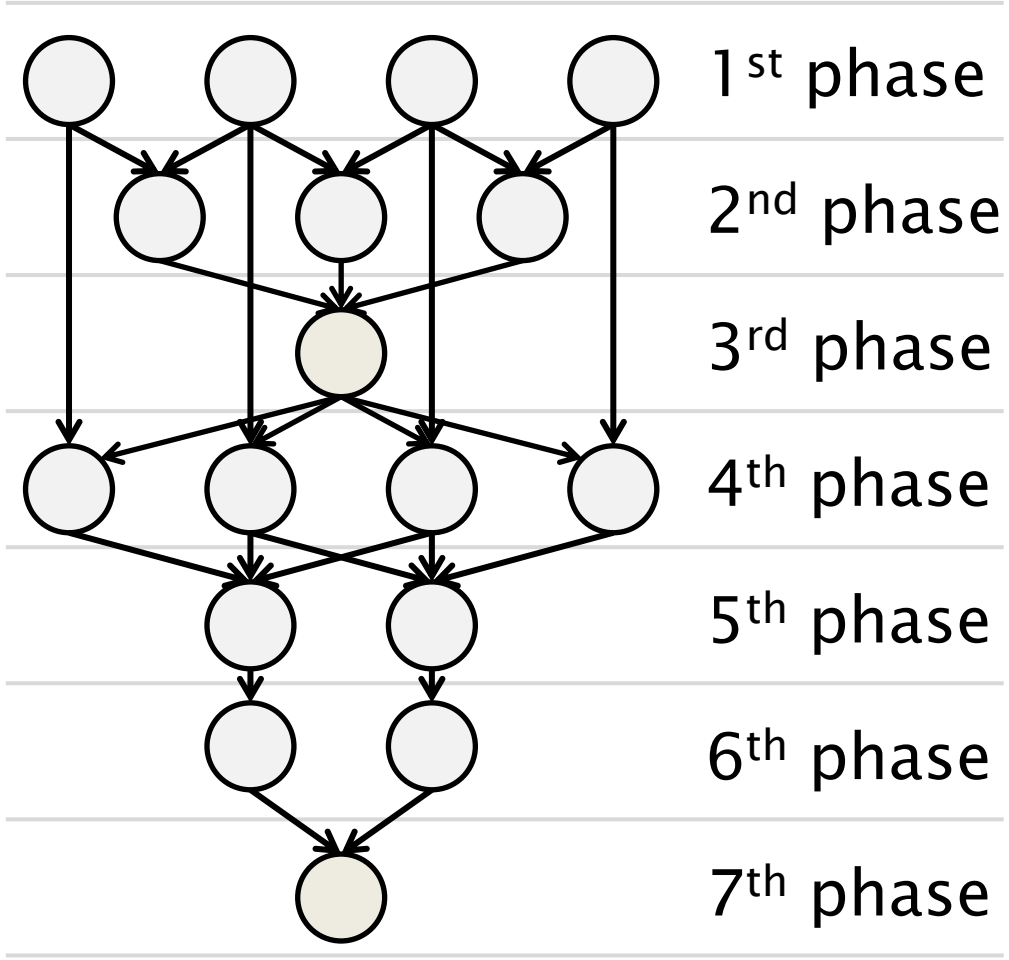
3rd dim: $\sum_{i \in V_A} w_3^i = \sum_{j \in V_B} w_3^j = 6$

Balance the sum of Vertex Weights at each dimension

Workflow Scheduling using MCGP

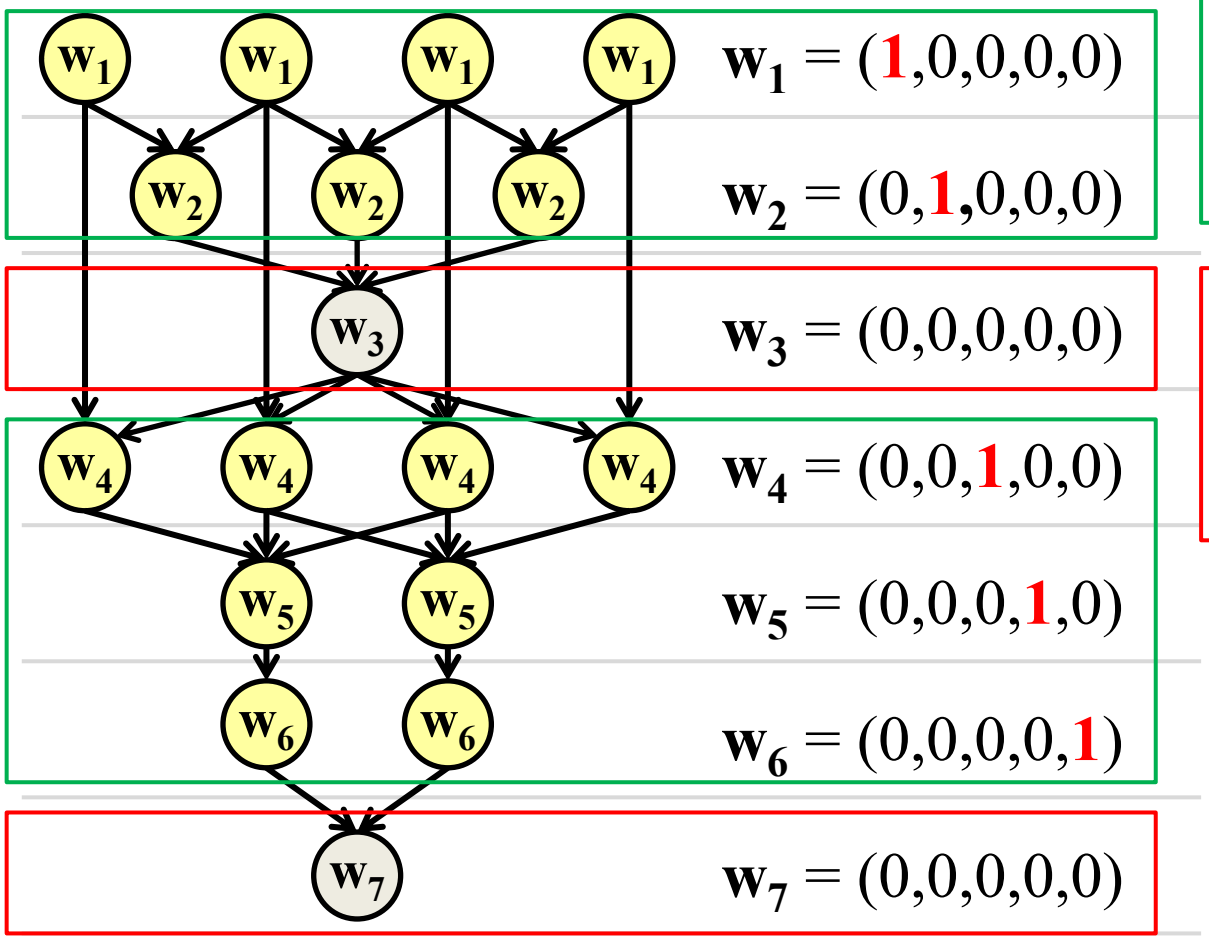
1. Define Workflow Phases
2. Define Weight Vector
3. Perform MCGP

Workflow Phases



Define **Task Phase** according to the dependencies

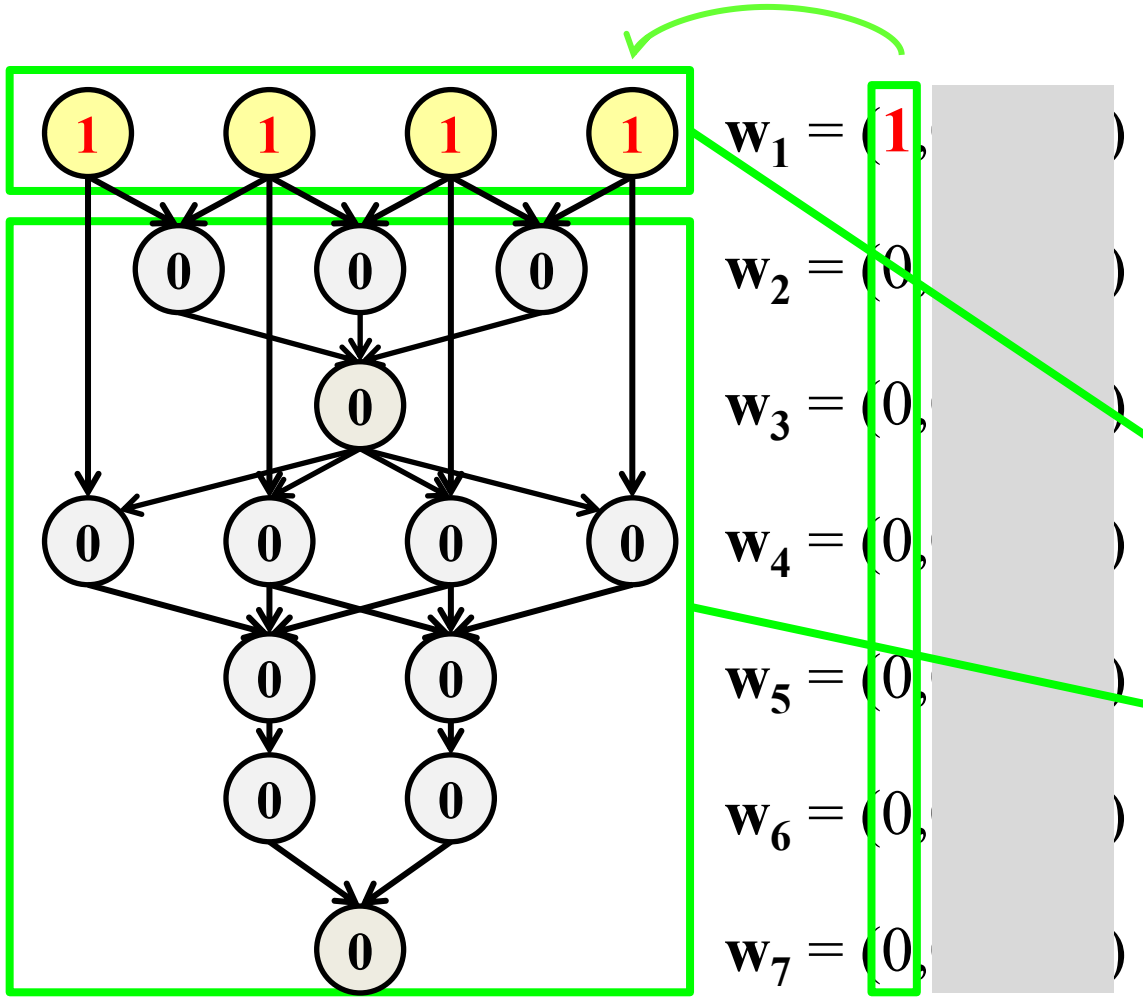
Weight Vector



$N_{task} \geqq N_{group}$:
- set 1 at i^{th} dim
- set 0 at others

$N_{task} < N_{group}$:
- set 0 to all dims

MCGP : the First dimension

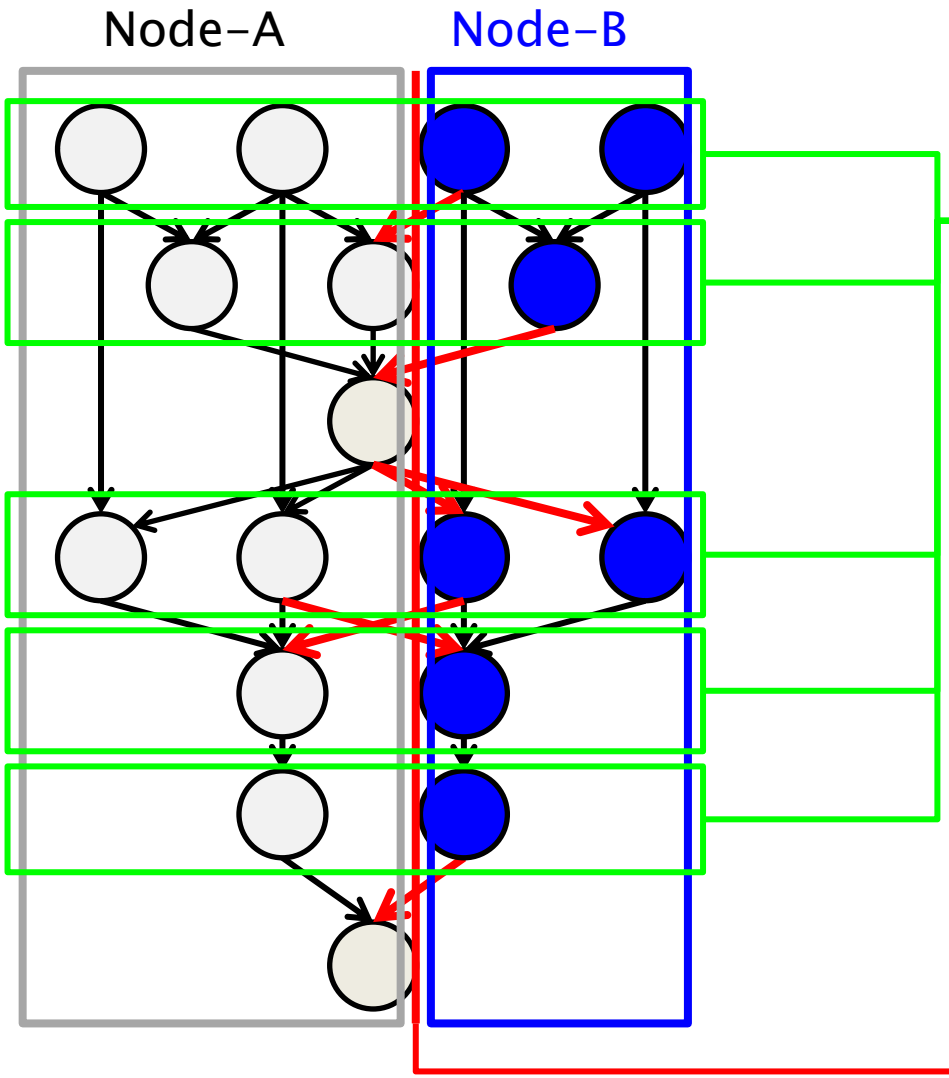


Weight:
- 1st phase … 1
- others … 0

Weight Balance:
1st phase only

Irrelevant to
Weight Balance

Result of MCGP



Uniform Distribution in each phase

Minimize Edge-cut applied to Entire Graph

Why Weight = 1 works ?

- Weight balancing in the same dim.
  - No influence on Task balancing between different phases
- Simple problem settings as a start
  - In the same phase
    - Computation costs are equal
  - Homogeneous computer cluster

Implementation

- Graph Partitioning: **METIS**
  - Schloegel et al. 2000
- Distributed Filesystem: **Gfarm**
  - Tatebe et al. 2010
  - Selects Storage node of Output Files
- Workflow System: **Pwrake**
  - Tanaka&Tatebe 2010
  - Based on Rake, Ruby version of *make*
  - Selects Compute node of Tasks

Outline

- Introduction
  - Workflow Scheduling for Data-Intensive Science
- Proposed Method
  - Workflow Scheduling using MCGP
- Evaluation
- Related Work
- Conclusion
- Future Work

Platform for Evaluation

- InTrigger Kobe (Saito 2007)

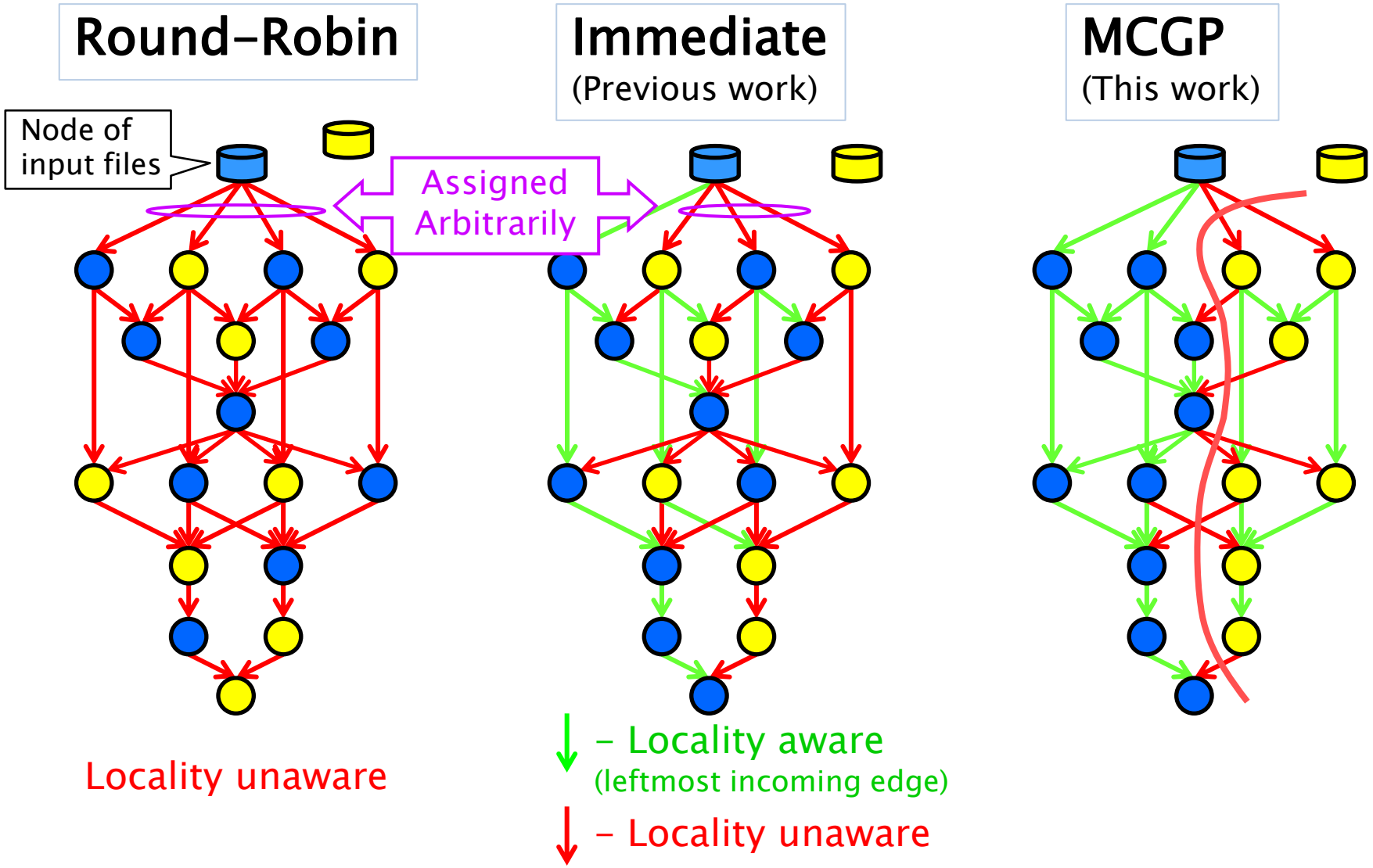
CPU	Xeon E5410 (2.3GHz)
Main Memory	16 GB
Network	GbE
# of Nodes	8
Total # of Cores	32

- Input File: 2MASS Image

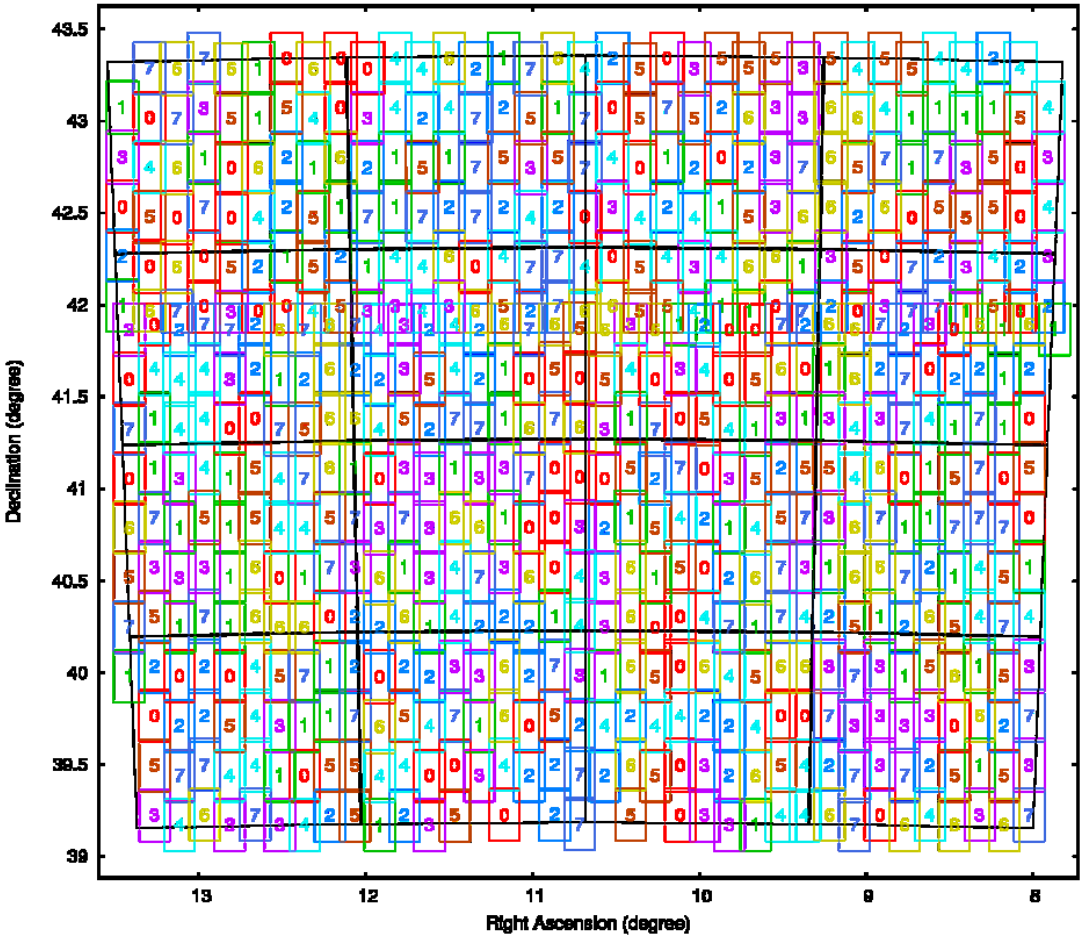
Data size of each File	2.1 MB or 1.7 MB
# of Input Files	607
Total Data size of Input Files	1270 MB
Data I/O size during Workflow	~24 GB
Total # of Tasks = # of Vertices	3090

At first, All the Input files are stored at a single node.

Scheduling Schemes

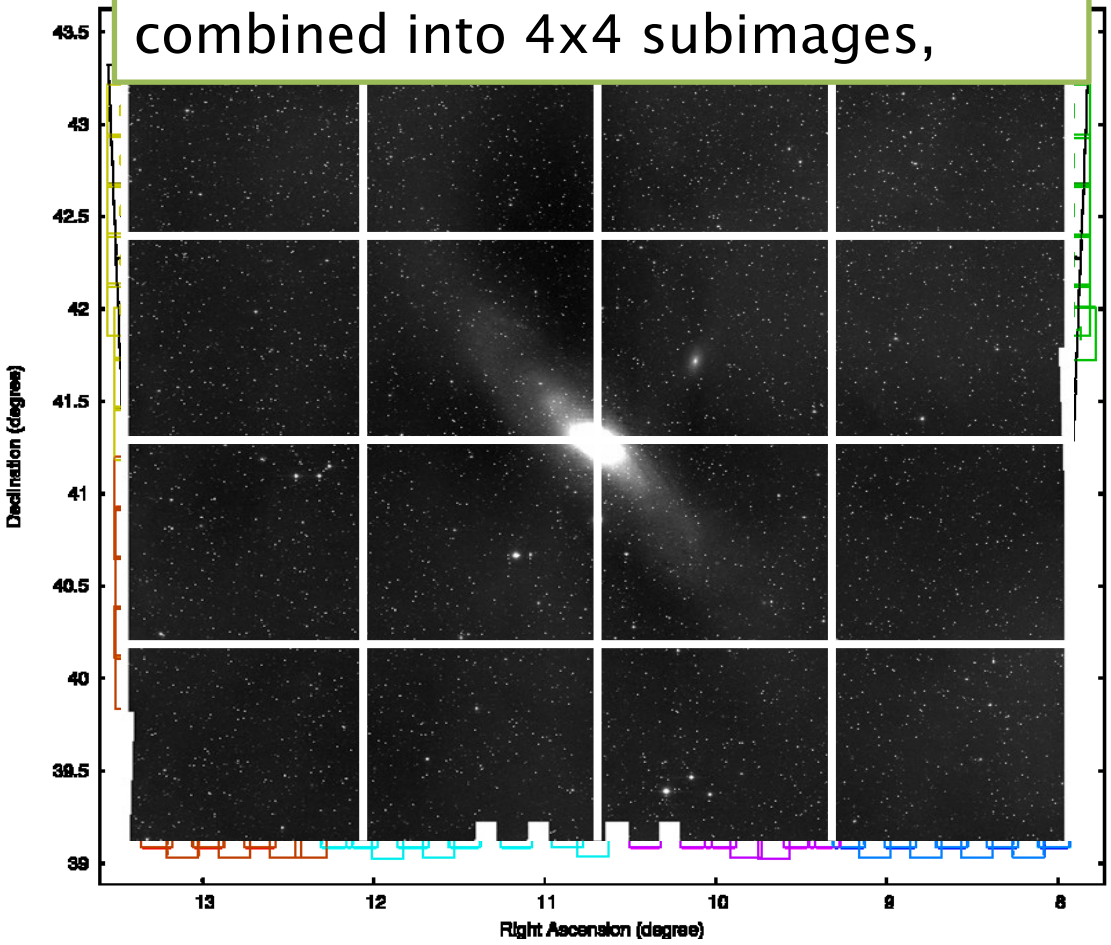


Task Assignment and Image Position: R-R and Immediate Schemes

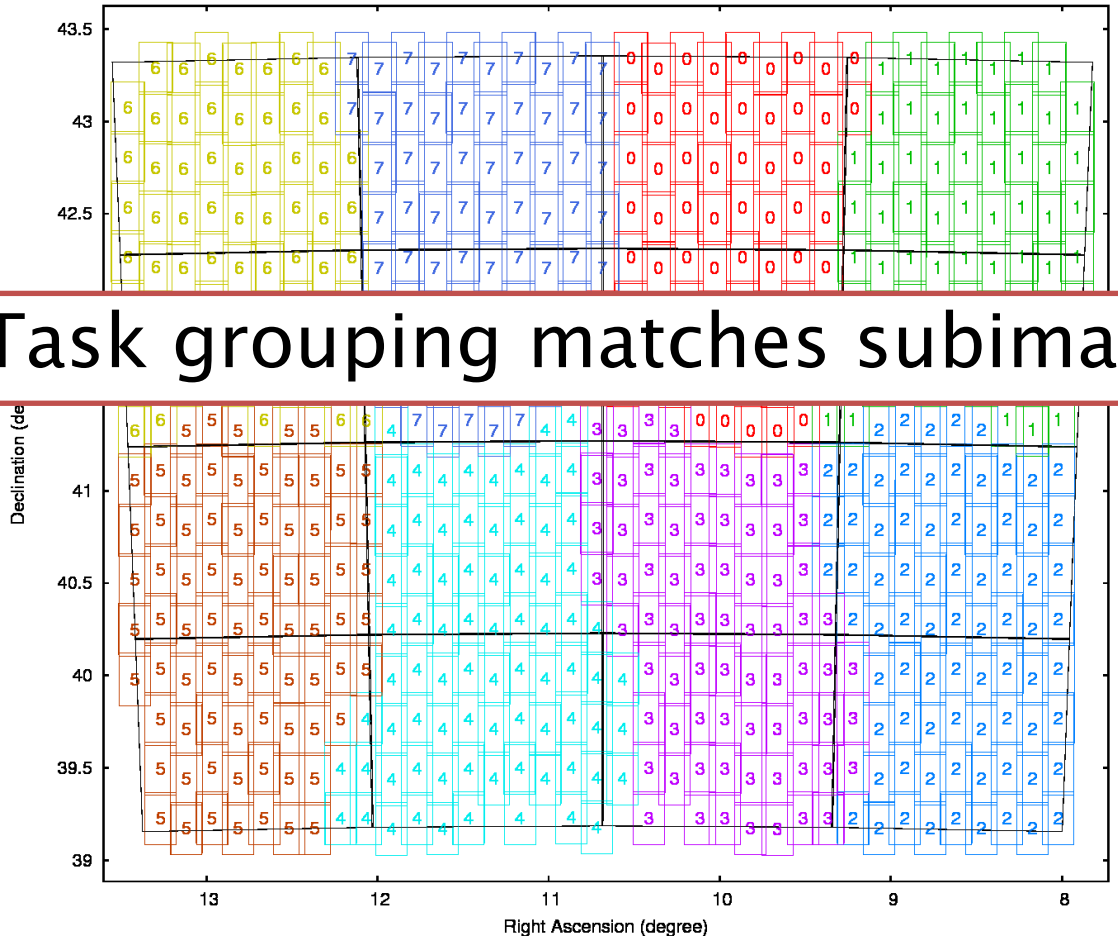


Task Assignment and Image Position: MCGP, 4x4 tiles

In the last phase of the workflow, combined into 4x4 subimages,

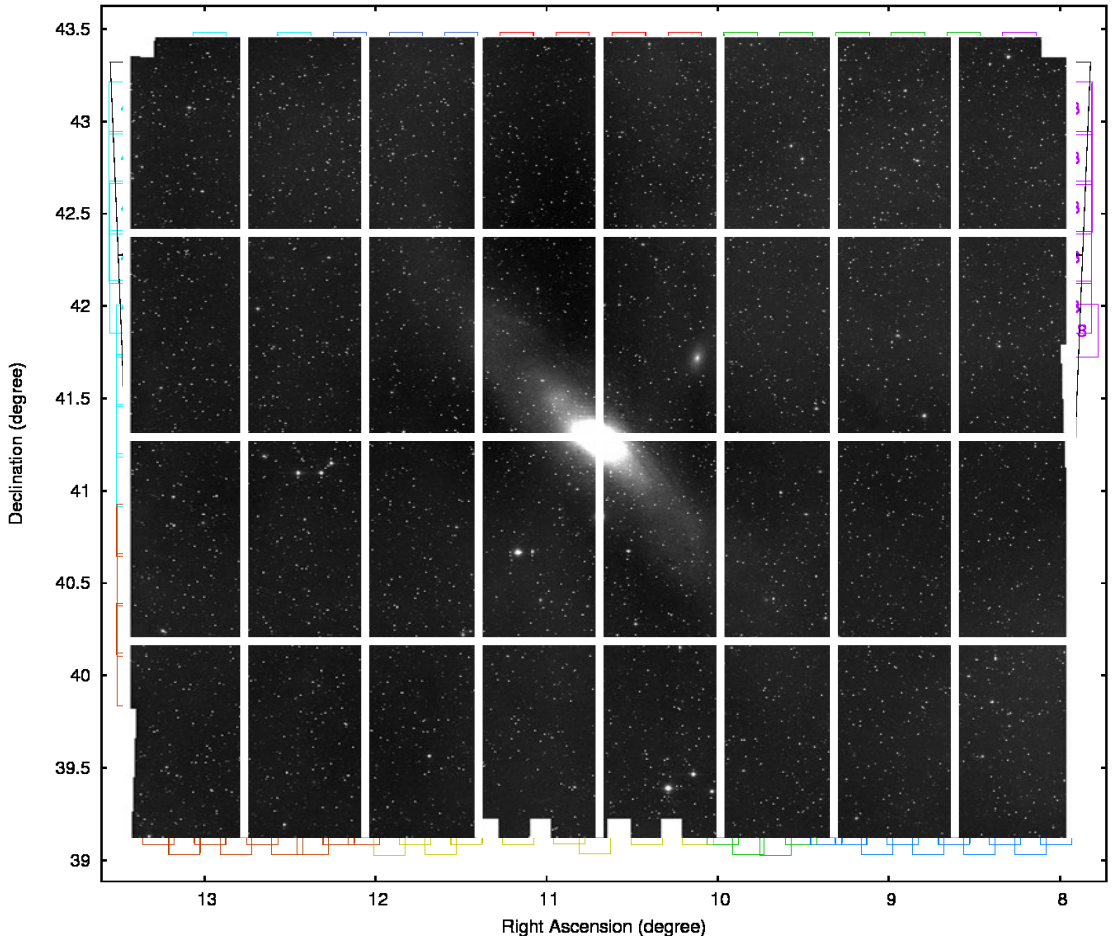


Task Assignment and Image Position: MCGP, 4x4 tiles

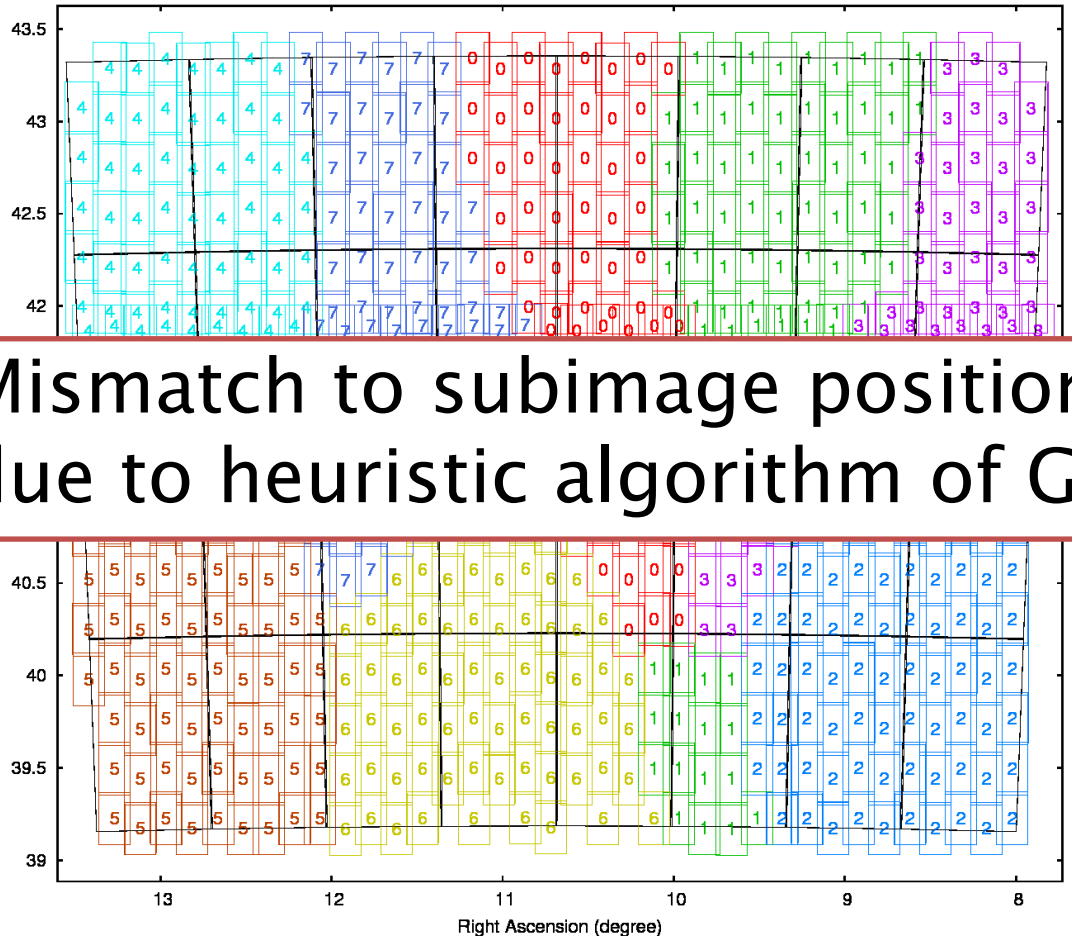


Task grouping matches subimage

Task Assignment and Image Position: MCGP, 8x4 tiles

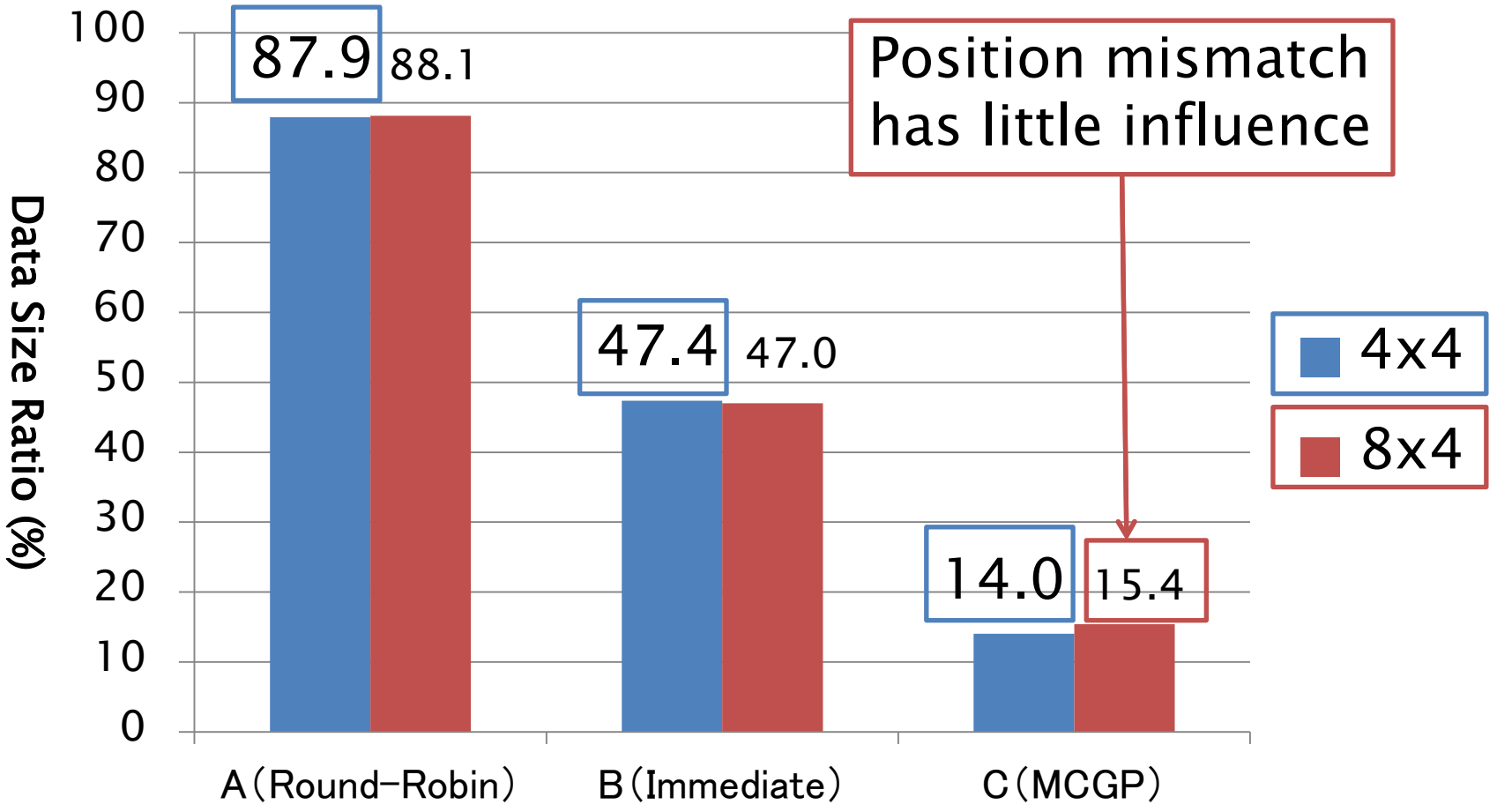


Task Assignment and Image Position: MCGP, 8x4 tiles

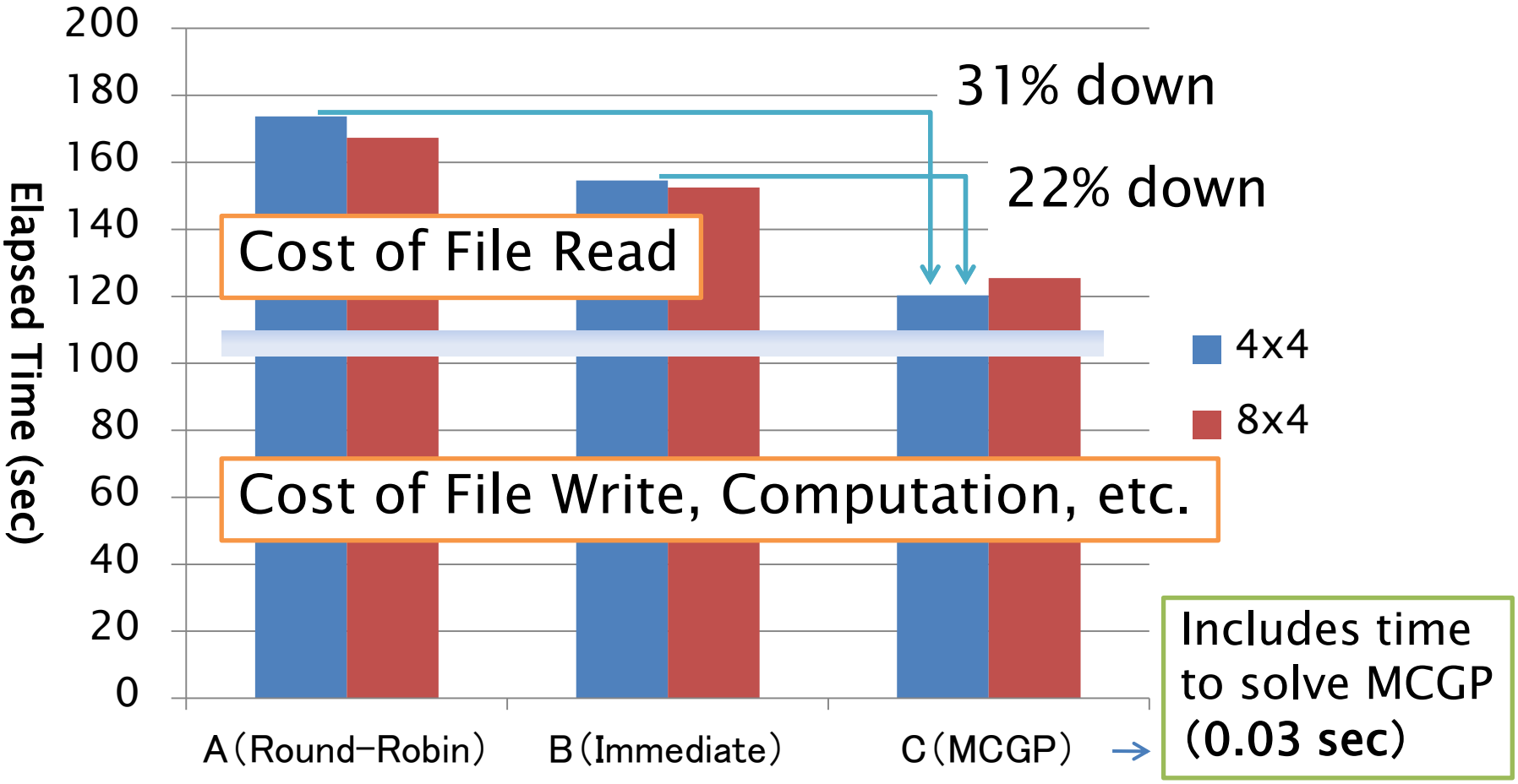


Mismatch to subimage position due to heuristic algorithm of GP

Inter-node Data Movement



Workflow Execution Time



Related Work

- Graph partitioning on workflow DAG
  - Dong et al. (2007), Kalayci et al., Sonmez et al. (2010)
  - not MCGP
- WF clustering using spatial information
  - Meyer et al. (2006)
  - only applicable to Astronomy
- MCGP for partitioning multi-phase tasks
  - Hendrickson & Kolda (2000)
  - not Workflow DAG

Conclusion

- Data-Intensive Science needs workflow scheduling to minimize data Movement
- We proposed a workflow scheduling method using MCGP
- Reduce the ratio of remote file access from 88% to 14%
- Decrease workflow execution time by 31%
- Time for MCGP is small (0.03 sec)

Future work

- Evaluation of workflows with uneven files sizes and computation costs
- Heterogeneous clusters
- Multi-level partitioning
  - Platform where processors are connected by networks with different throughput