A Model-Based Algorithm for Optimizing I/O Intensive Applications in Clouds using VM-Based Migration

Kento Sato† Hitoshi Sato† Satoshi Matsuoka†‡

†: Tokyo Institute of Technology
‡: National Institute of Informatics

TOKYO TECH
Pursuing Excellence

Outline

- Introduction
- Target cloud model
- Proposal
  - DAG algorithm
  - Markov model
  - Performance model
- Evaluation
- Conclusion

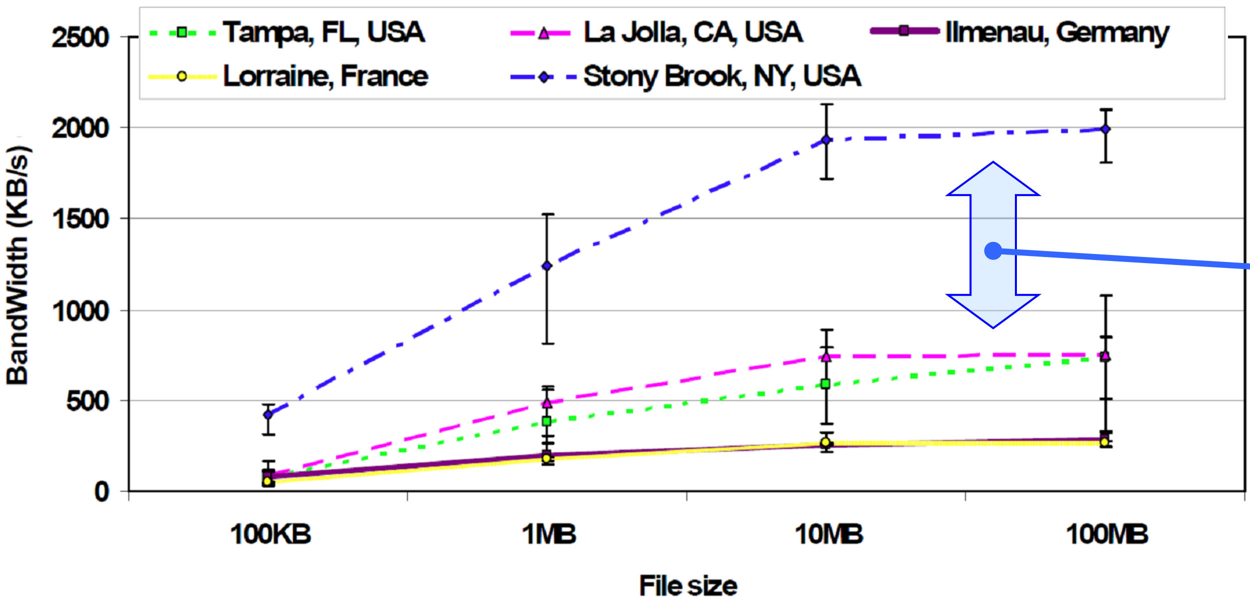
Background

- Large-scale distributed file system
  - Providing much larger amounts of storage resources than those of typical single-site
  - Giving a common view of all files stored independent from which node access the data

- Amazon S3(simple storage service)
  - virtually infinite storage spaces with high availability
  - cost-effective pay-as-you-go model

What's the problem ?

- Data Transfer Cost
  - Causing I/O performance degradation of data intensive applications



the location of the client impacts on the observed data access performance in Amazon S3

- Previous approach: File migration
  - File replication & File caching

Graph: Quoted from *Amazon S3 for Science Grids: a Viable Solution ?* in 4th USENIX Symposium on Networked Systems Design & Implementation (NSDI'07), 2007. M. Palankar, A. Onibokun, et al.

Our Approach

- VM-Based Approach : VM migration
  - Being in practical use
  - Migrating VMs onto the locations that hold target files
    - Increasing the performance of file accesses
    - Causing also VM migration cost

  ⇒ Difficult to determine when and where to migration VMs

Represent VM's file access patterns as a DAG, and determine the best location for file access

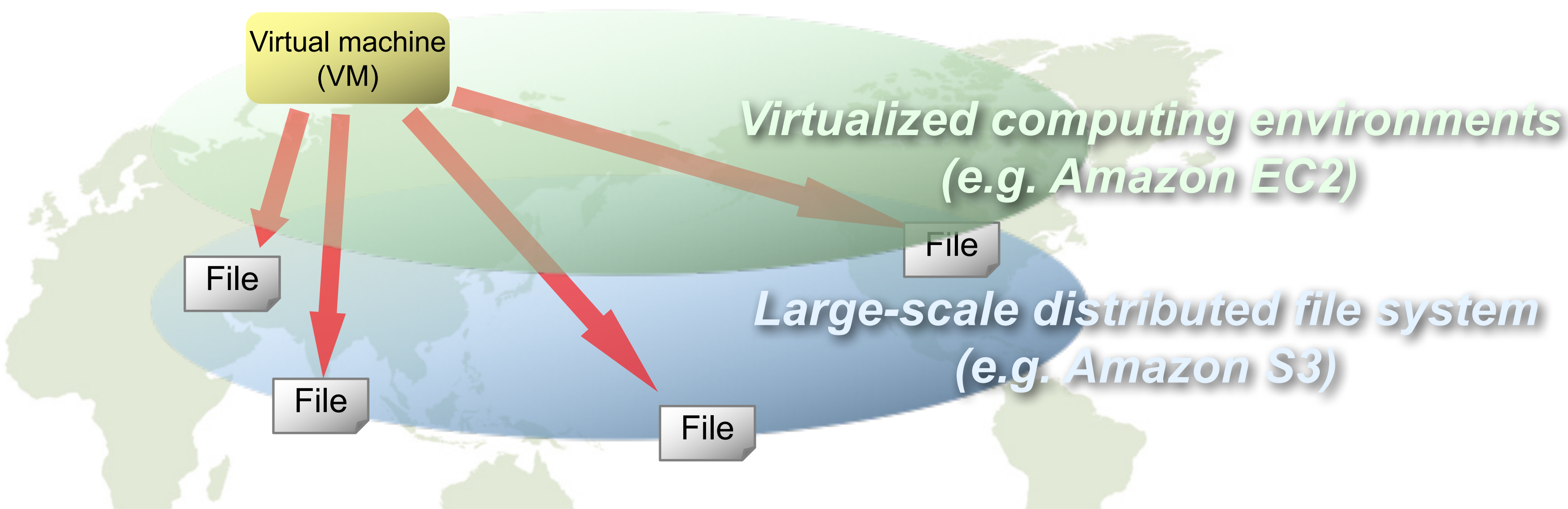
Goal and Achievement

- Goal
  - Optimization of I/O intensive application in Cloud using VM-based migration
- Achievement
  - Proposed a model-based I/O performance optimization algorithm for data-intensive application
  - Our algorithm can achieve higher I/O performance than simple techniques
    - Never migrating VM：38%
    - Always migrating VM：47%

TOKYO TECH Pursuing Excellence

Our Target Cloud Model

- Virtualized computing environments on distributed file system
- Target jobs feature: data-intensive application that accesses distributed multi-files
  - write-once, read-mostly applications



Optimizing the jobs by improving read performance

Previous Approach

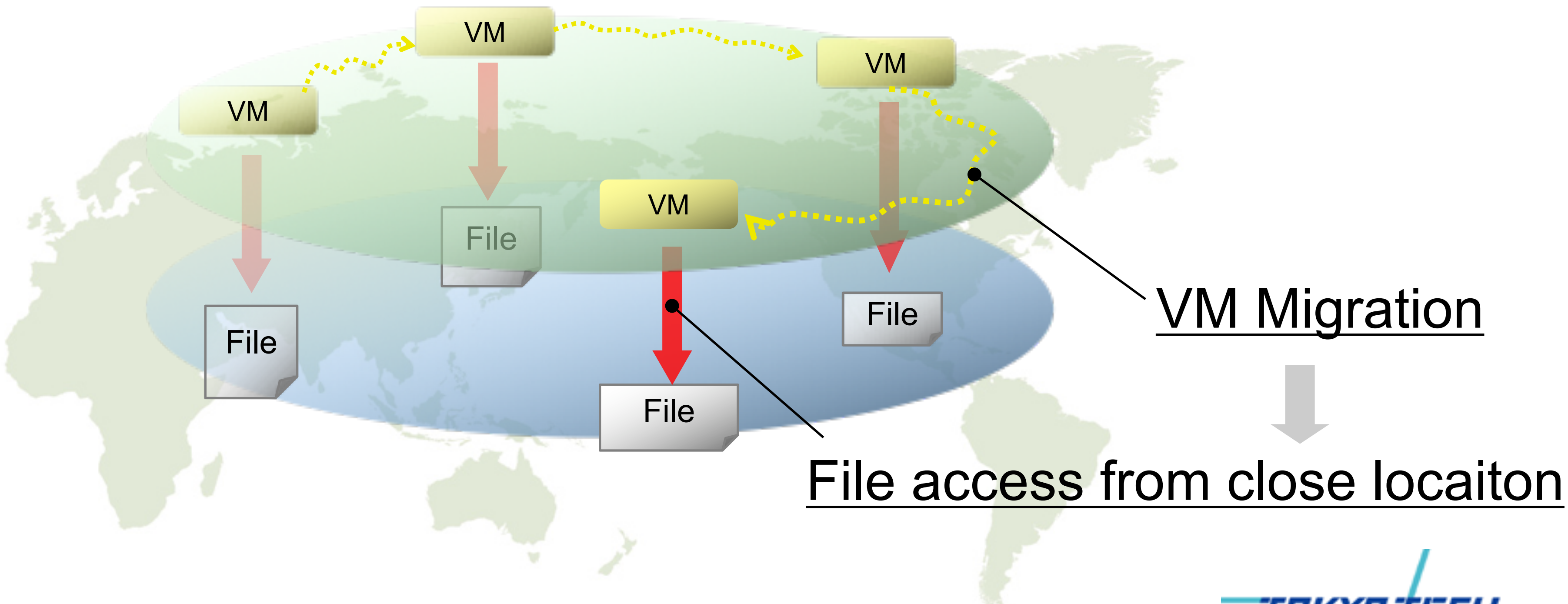
- File replication & caching [Venugopal et al. '06]
  - 😊 Minimizing remote file accesses by creating multiple copies and caching frequently-accessed hot file
  - ☹ Introducing a large amount of file transfer and storage consumption

- File-location-aware job scheduling [Shankar et al. '07]
  - 😊 Submitting jobs to sites where target files are located to avoid remote file access
  - ☹ Still causing remote file access, in case a job accesses to geographically distributed files

Our Approach

Migrates VM to onto close locations to target files

- Expected to improve the I/O performance



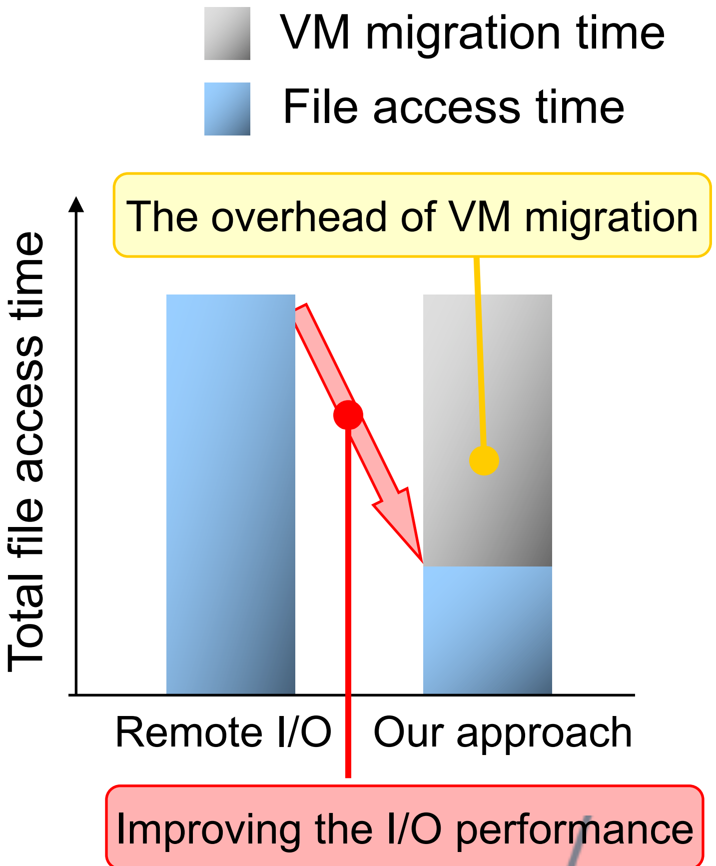
TOKYO TECH
Pursuing Excellence

Difficulty of VM relocation algorithm

Considering the overhead of VM migration

- Not good to migrate VM to target files every times
- File access time and VM migration time depends on runtime environments
  - e.g.) Network throughputs, access file size, VM memory size etc

⇒We have to determine the optimal migration strategy from the runtime environments

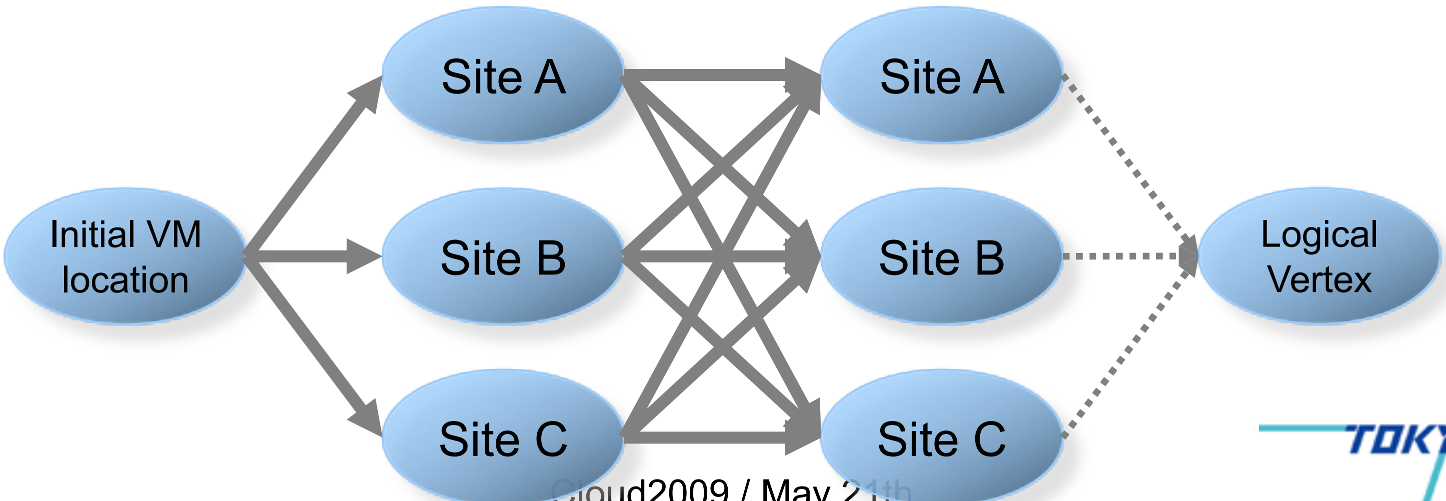


Optimal VM relocation techniques

- Determine VM migration strategies
  - i.e. When VM should be migrated to which sites
  - Minimizing file access time including VM migration time
- Collection of Information to be used
  - Cloud Information:
    - inter-site throughputs, local file system throughputs within each site
  - File Information:
    - size, location, dependency
  - VM Information:
    - memory size, location
- Output a optimal location for requested file

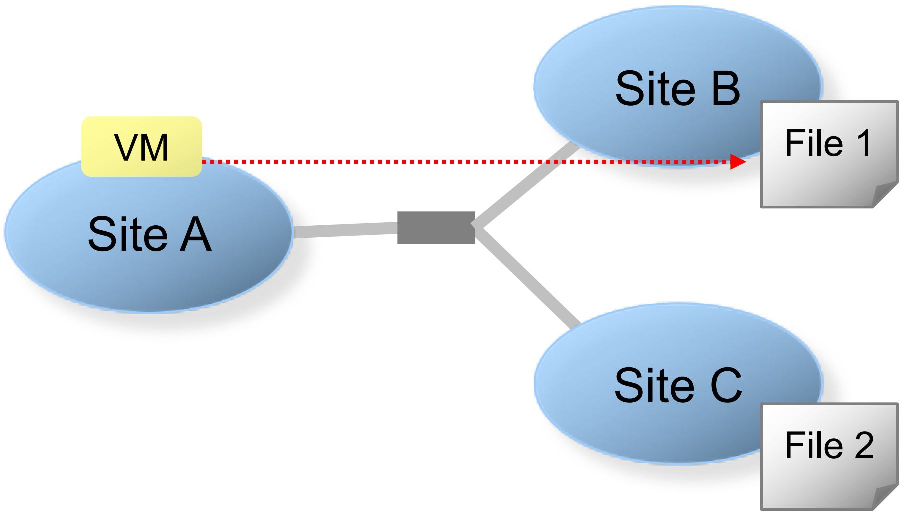
Overview of our algorithm

- Representing possible VM location as a DAG
  - Vertex: File access location
  - Edge: VM migration
- Calculating shortest path of the DAG
  - Vertex weights: Expected file access time
  - Edge weights: VM migration time



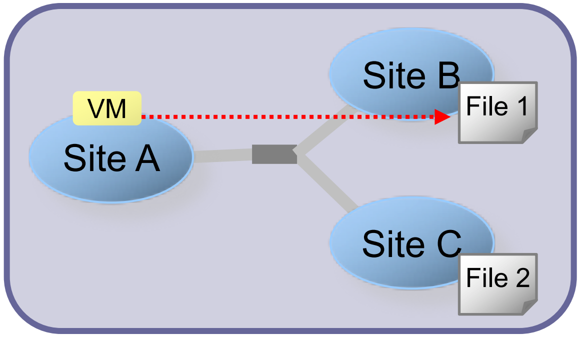
Example

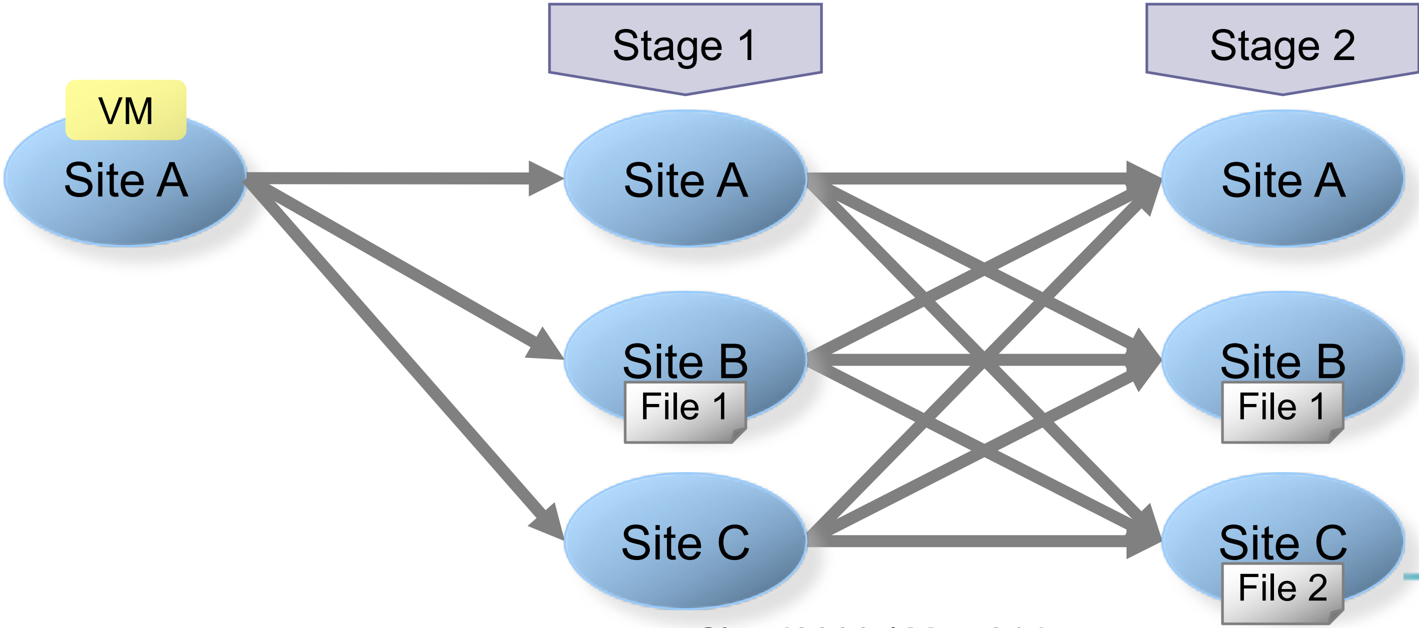
- Consider a simple situation
  - File location: File 1 (Site B), File 2 (Site C)
  - VM location: site A
- Explain how to determine a optimal location of VM that access to File 1



Possible migration strategies

- Representing possible VM location as a DAG
  - Vertex: File access location
  - Edge: VM migration

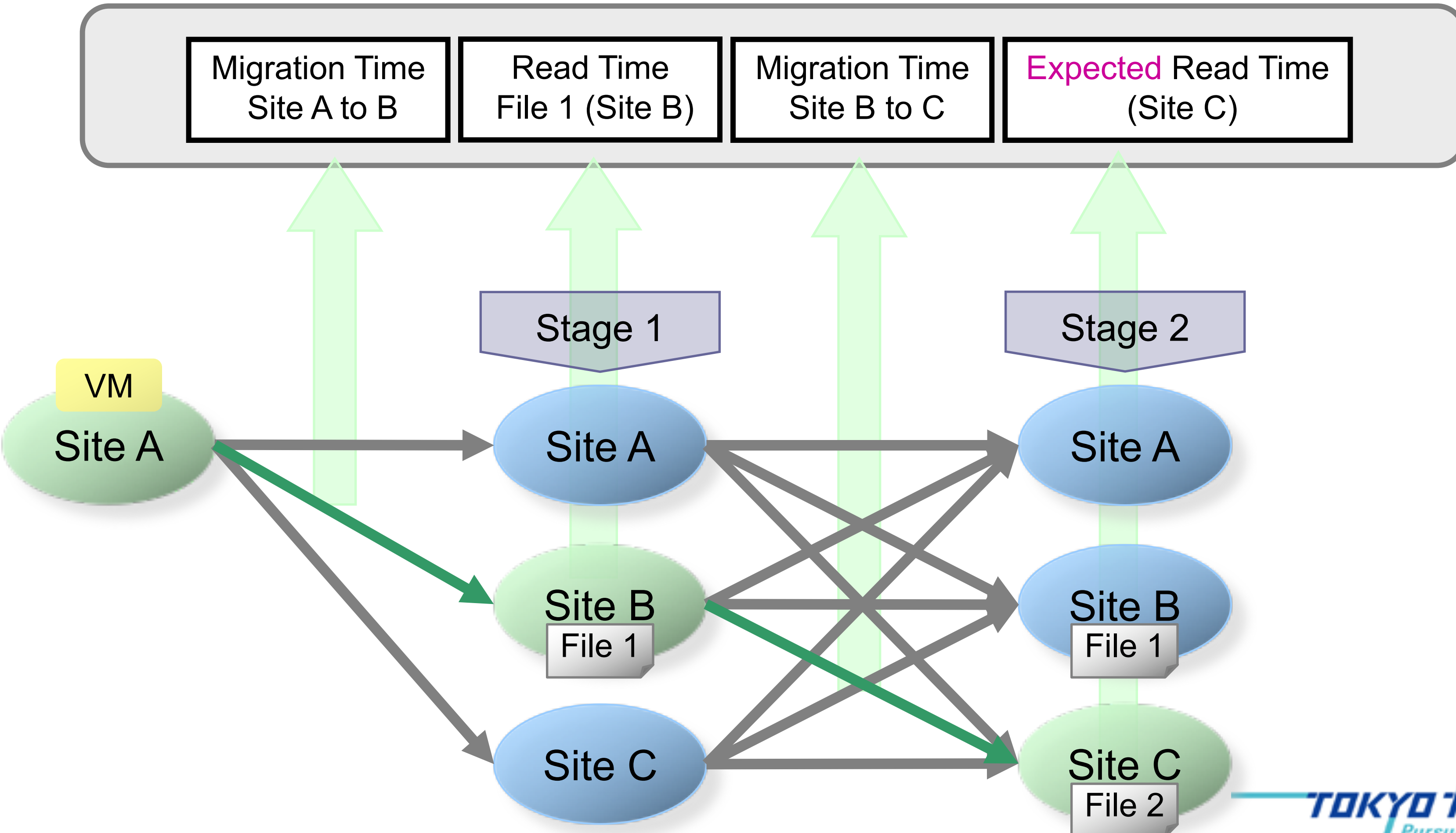




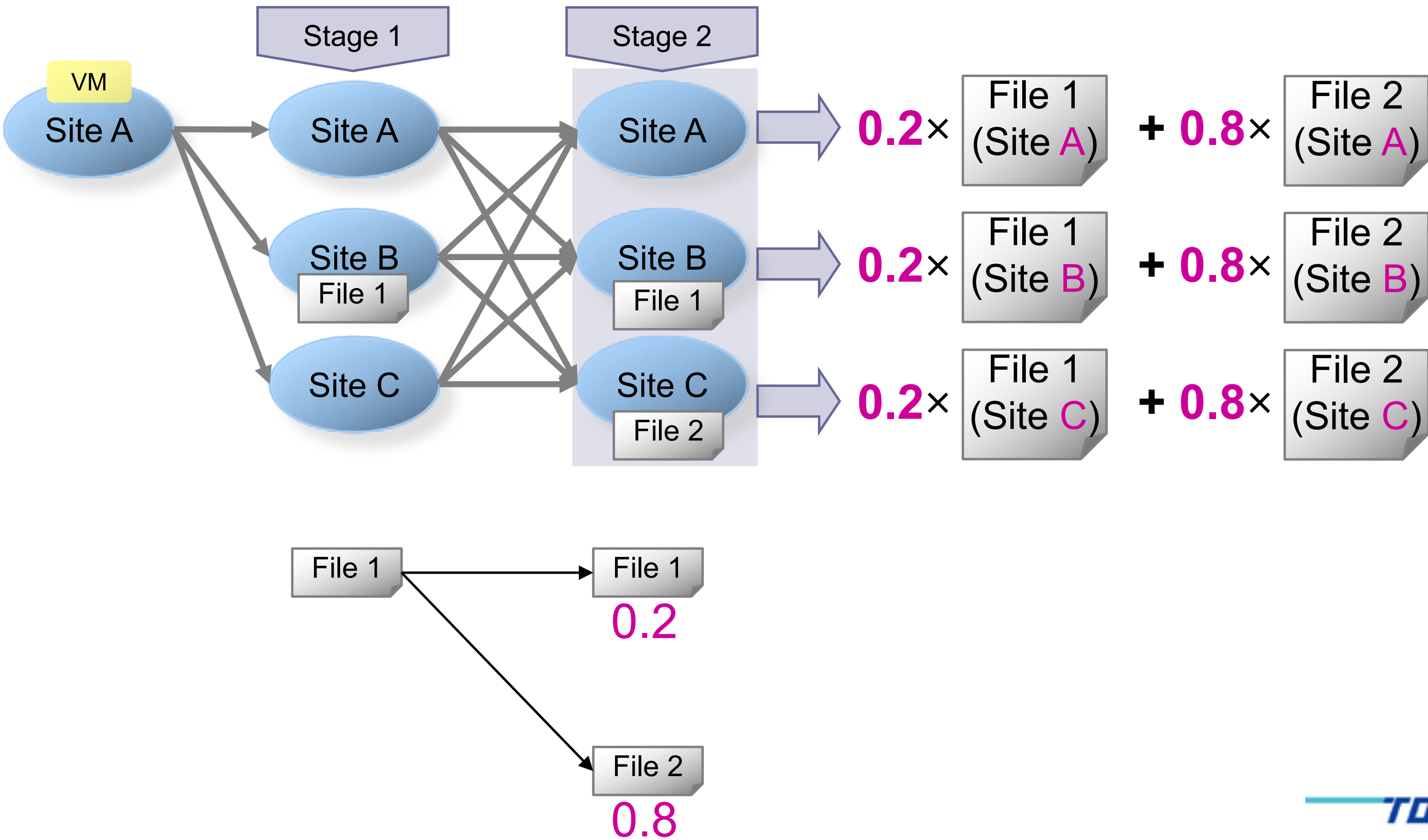
TOKYO TECH Pursuing Excellence

How to calculate the total access time ?

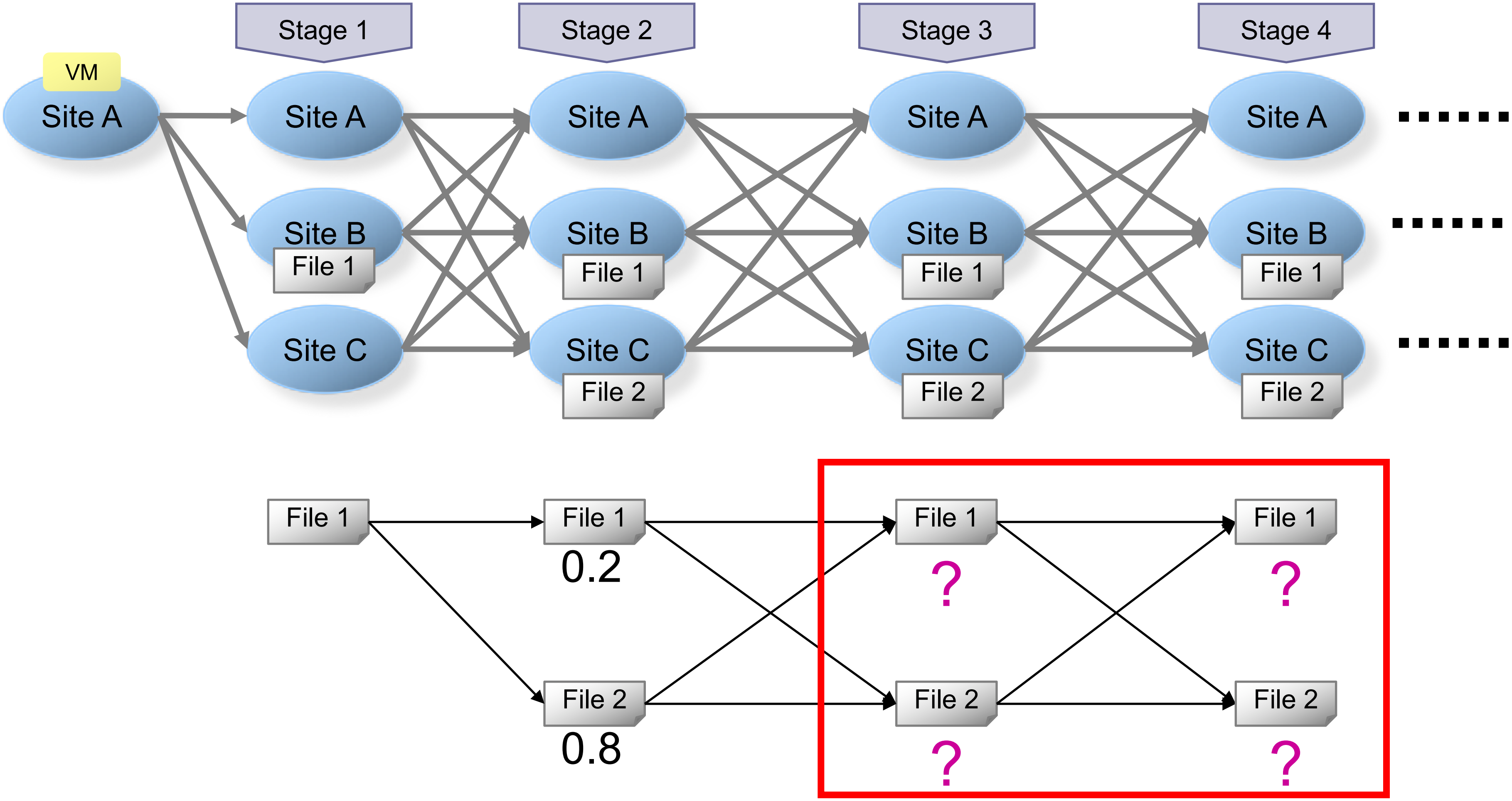
The total access time is the summation of following times



How to calculate the expected file access time ?



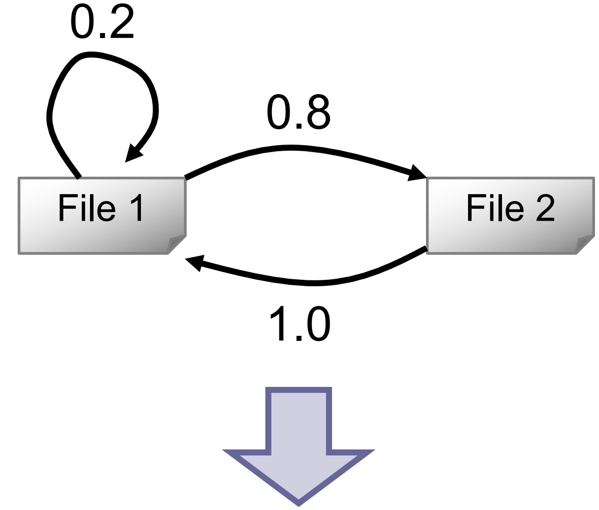
How to calculate expected file access time on the other stages ?



File Access Markov Model

Calculating expected file access time from markov model

- Markov model
  - representing the probability of access transitions from one file to another from monitored trace



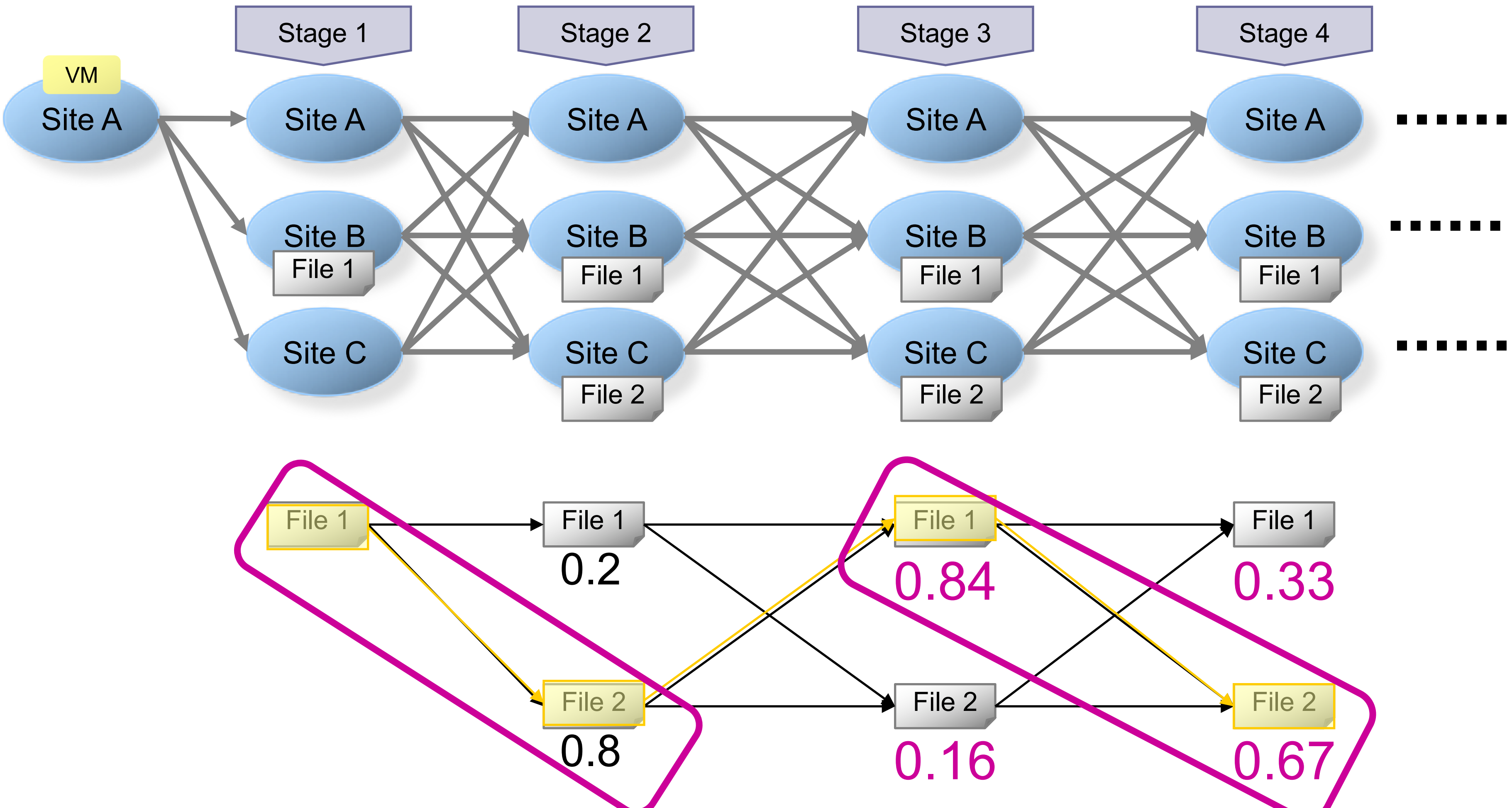
- Stochastic matrix
  - Describing a markov model as a matrix
  - P^k : the possibility of file access transitoins from one file to another with *k*-step

	File 1	File 2
File 1	0.2	0.8
File 2	1.0	0.0

$$P = \begin{pmatrix} 0.2 & 0.8 \\ 1.0 & 0.0 \end{pmatrix}$$

How to calculate expected file access time on the other stages ?

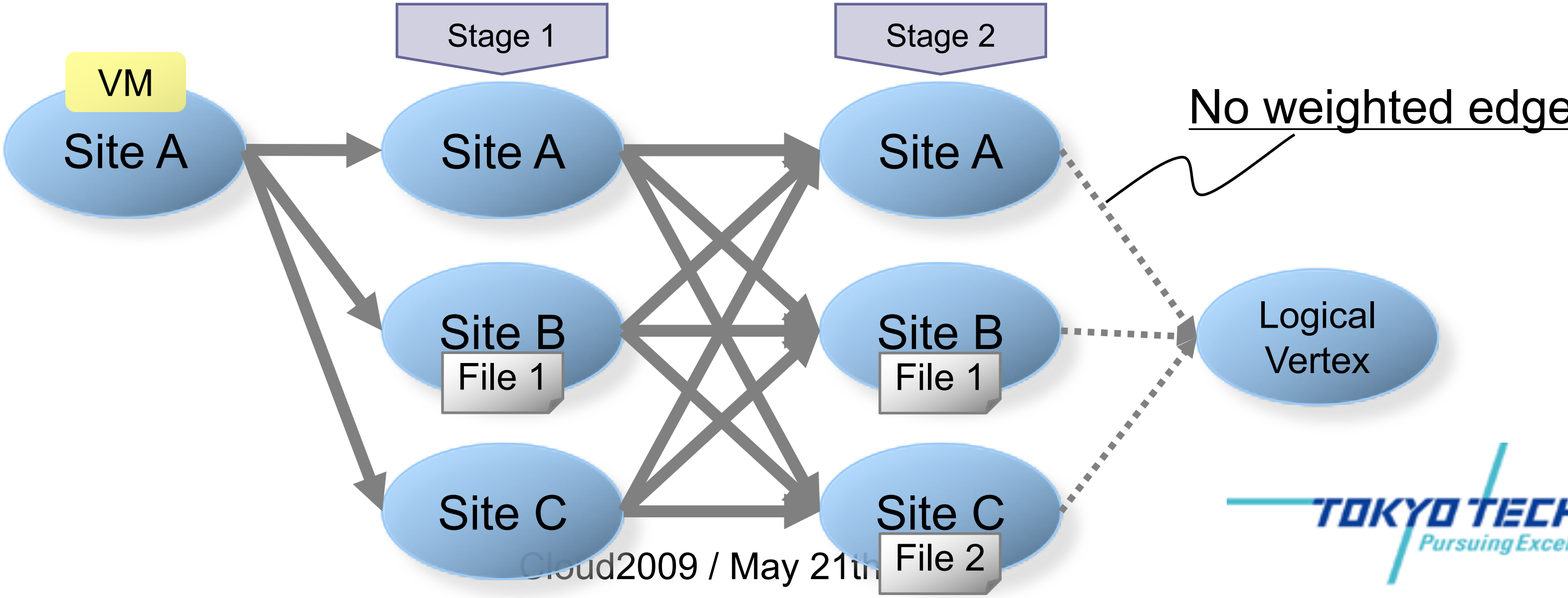
$$P = \begin{pmatrix} 0.2 & 0.8 \\ 1.0 & 0.0 \end{pmatrix} \quad P^2 = \begin{pmatrix} 0.84 & 0.16 \\ 0.2 & 0.8 \end{pmatrix} \quad P^3 = \begin{pmatrix} 0.33 & 0.67 \\ 0.84 & 0.16 \end{pmatrix}$$



How to determine a optimal location for File 1 ?

Search a Shortest Path of the DAG

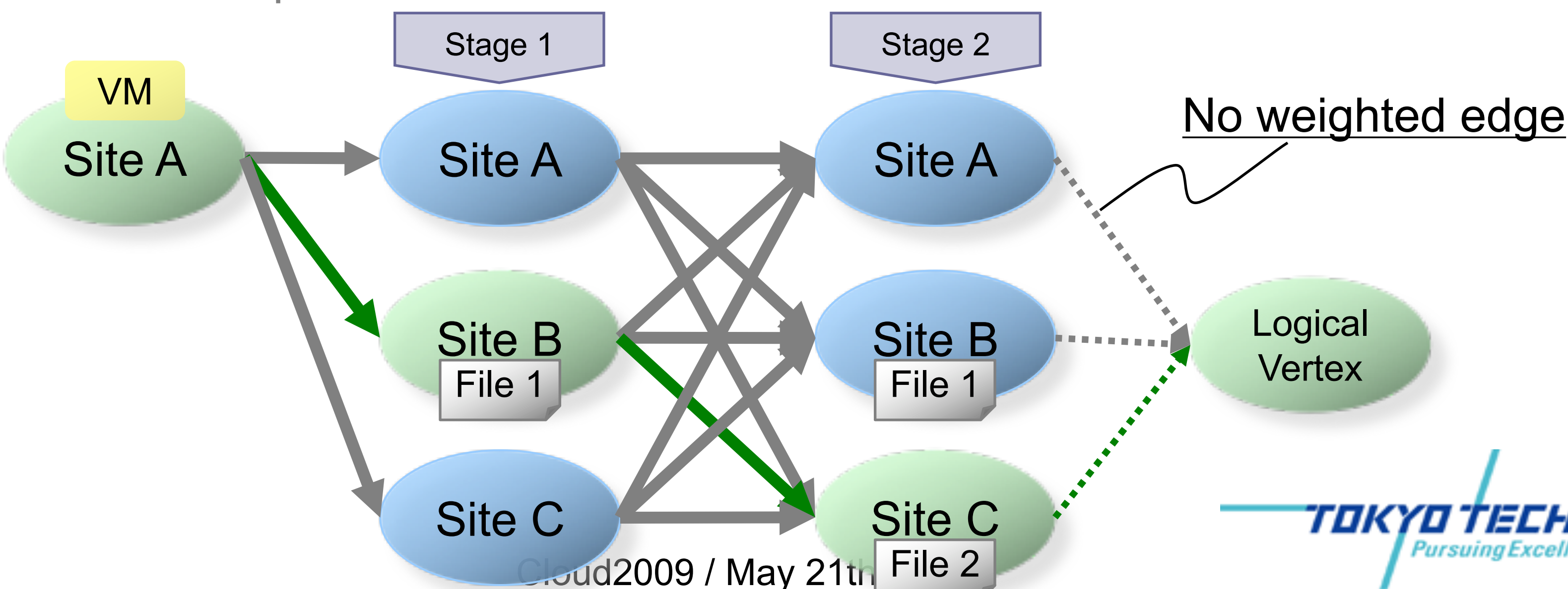
- Adding a logical vertex connected with no weighted edges at the end of DAG
- Solving a shortest path between each ends
  - Vertex weights: Expected file access time
  - Edge weights: VM migration time
- If following path is shortest one …
  - Site B is optimal location for File 1 and successive files



How to determine a optimal location for File 1 ?

Search a Shortest Path of the DAG

- Adding a logical vertex connected with no weighted edges at the end of DAG
- Solving a shortest path between each ends
  - Vertex weights: Expected file access time
  - Edge weights: VM migration time
- If following path is shortest one …
  - Site B is optimal location for File 1 and successive files



Performance Models

File access time model	$\frac{io\_size}{\min(network,\ local)}$
VM migration time model	$\frac{vm}{network} + c\ (const)$

io_size : Access File size (MB)

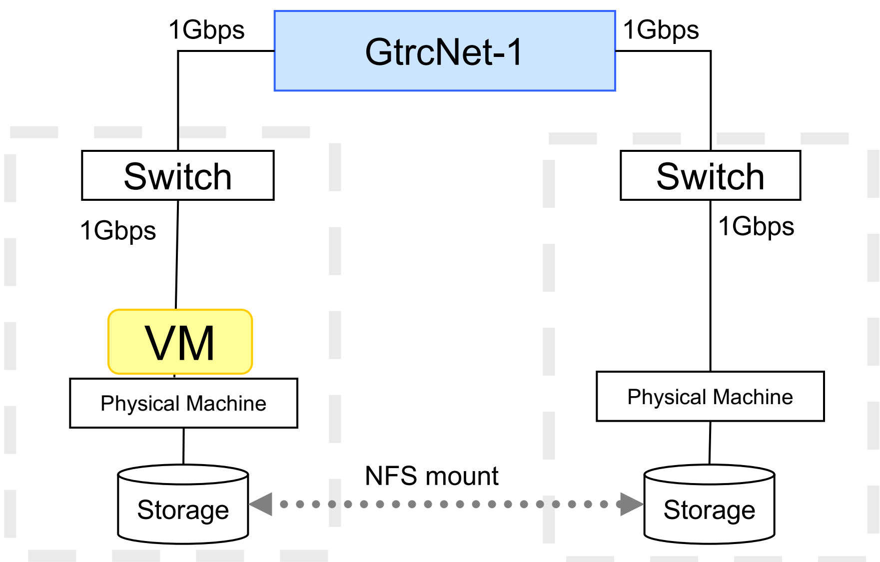
$network$: Network throughput (MB/s)

$local$: Local I/O throughput (MB/s)

vm : Allocated VM Memory size (MB)

Experimental Environment for Performance modeling

- Connect 2 machines via network emulator GtrcNet-1[Kodama et al '04]
  - PrestoIII cluster at Tokyo Tech
- Virtual machine monitor: Xen



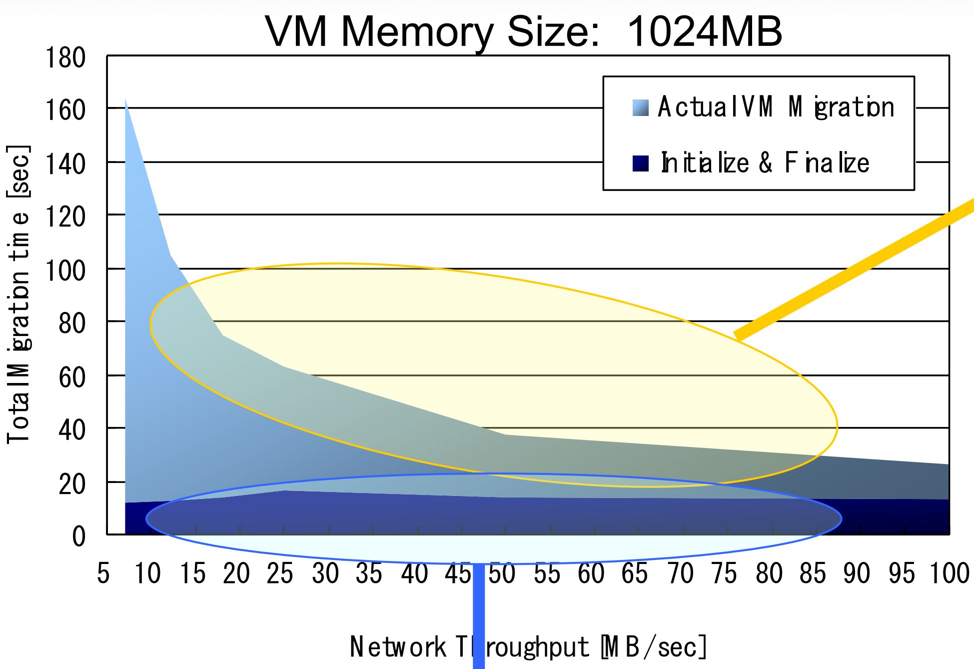
- Machines Configurations -

OS	Debian/Linux (kernel: 2.6.18-xen)
CPU	Opteron250 (2.4GHz) * 2
Memory	2GB
NIC	NetXtreme BCM5704
Xen	Xen 3.1.0

Experiments setting for creating Performance Models

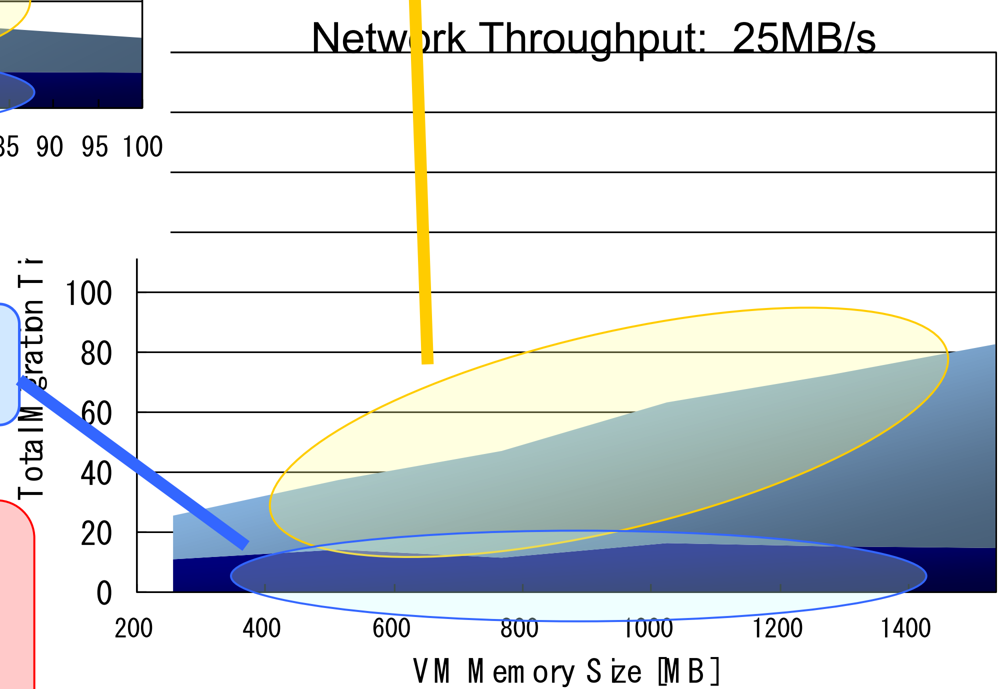
- VM Migration Time Model
  - Migrate a VM running an application between two machines
  - Application: BLAST, no application (idle)
  - Network throughputs: 5 – 100 [Mbps]
  - VM memory size: 256, 512, 768, … , 1536 [MB]
  - Target VM Migration: Stop-and-Copy Way

VM Migration time while BLAST exec



1. Memory Image transp time：
   - Inverse proportion to VM memory size
   - Proportion to Network throughput

2. initialization & finalization time is const



3. We got same results in case of migrating VM running no application

Performance Model

File access time model	$\dfrac{io\_size}{\min(network,\ local)}$
VM migration time model	$\dfrac{vm}{network} + c\ (const)$

io_size : Access file size (MB)

$network$: Network throughput (MB/s)

$local$: Local I/O throughput (MB/s)

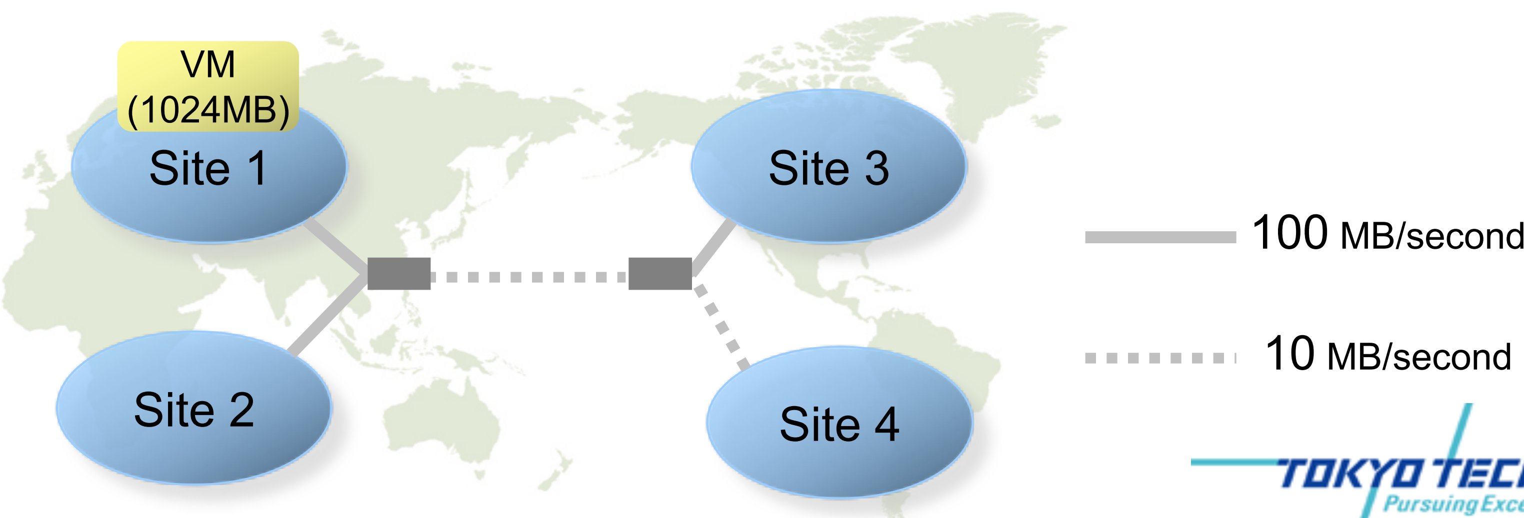
vm : Allocated VM memory size (MB)

Outline

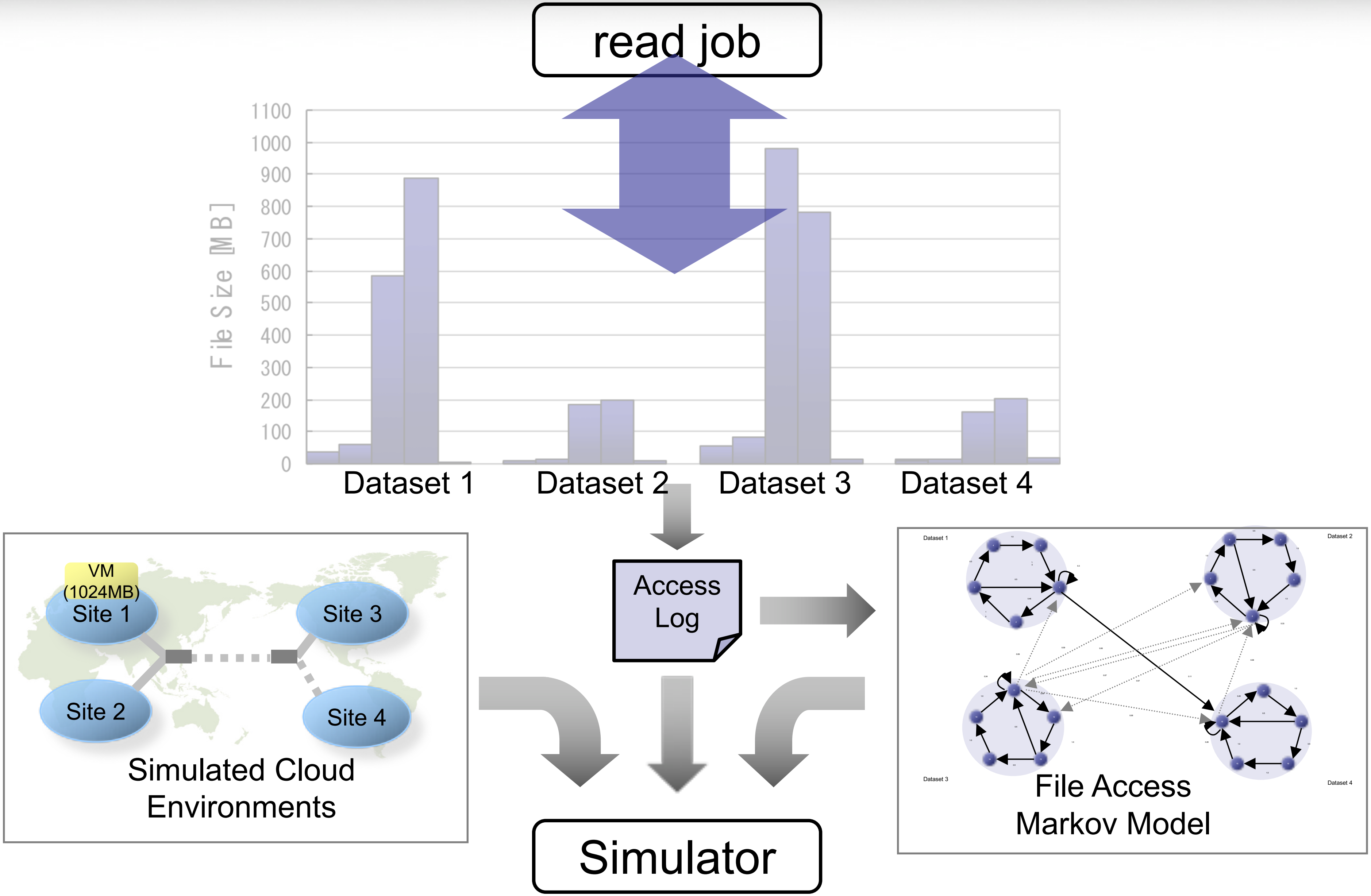
- Introduction
- Target cloud model
- Proposal
  - DAG algorithm
  - Markov model
  - Performance model
- Evaluation
- Conclusion

Experiments settings

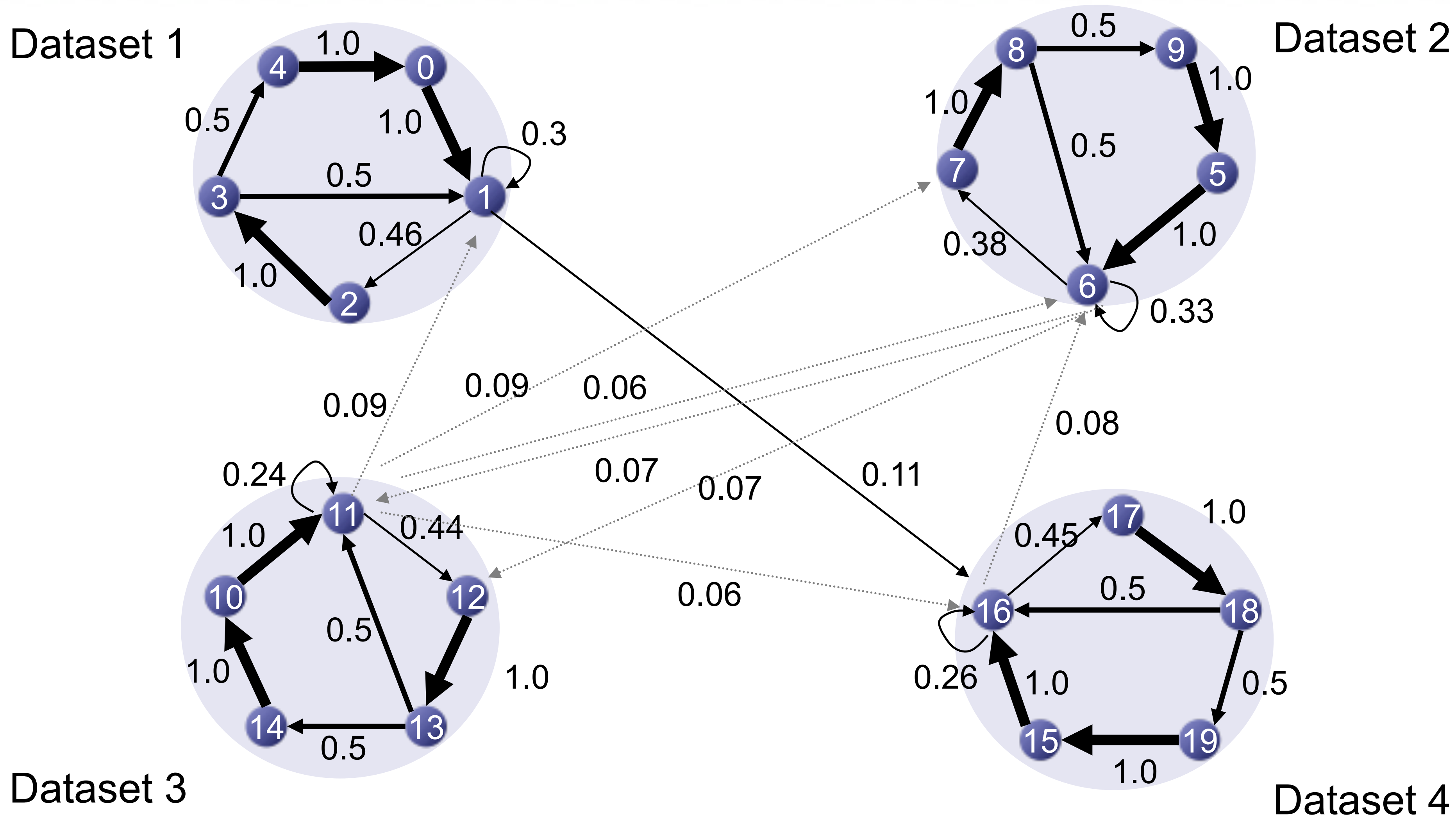
- Cloud settings
  - Network: 10 or 100[MB/second]
  - Local I/O Throughputs: 60[MB] on each site
- VM settings
  - Memory size: 1024[MB]
  - Initial Location: Site 1



Simulation Method



Markov model of file dependency



Edge wight(<0.05) is omitted

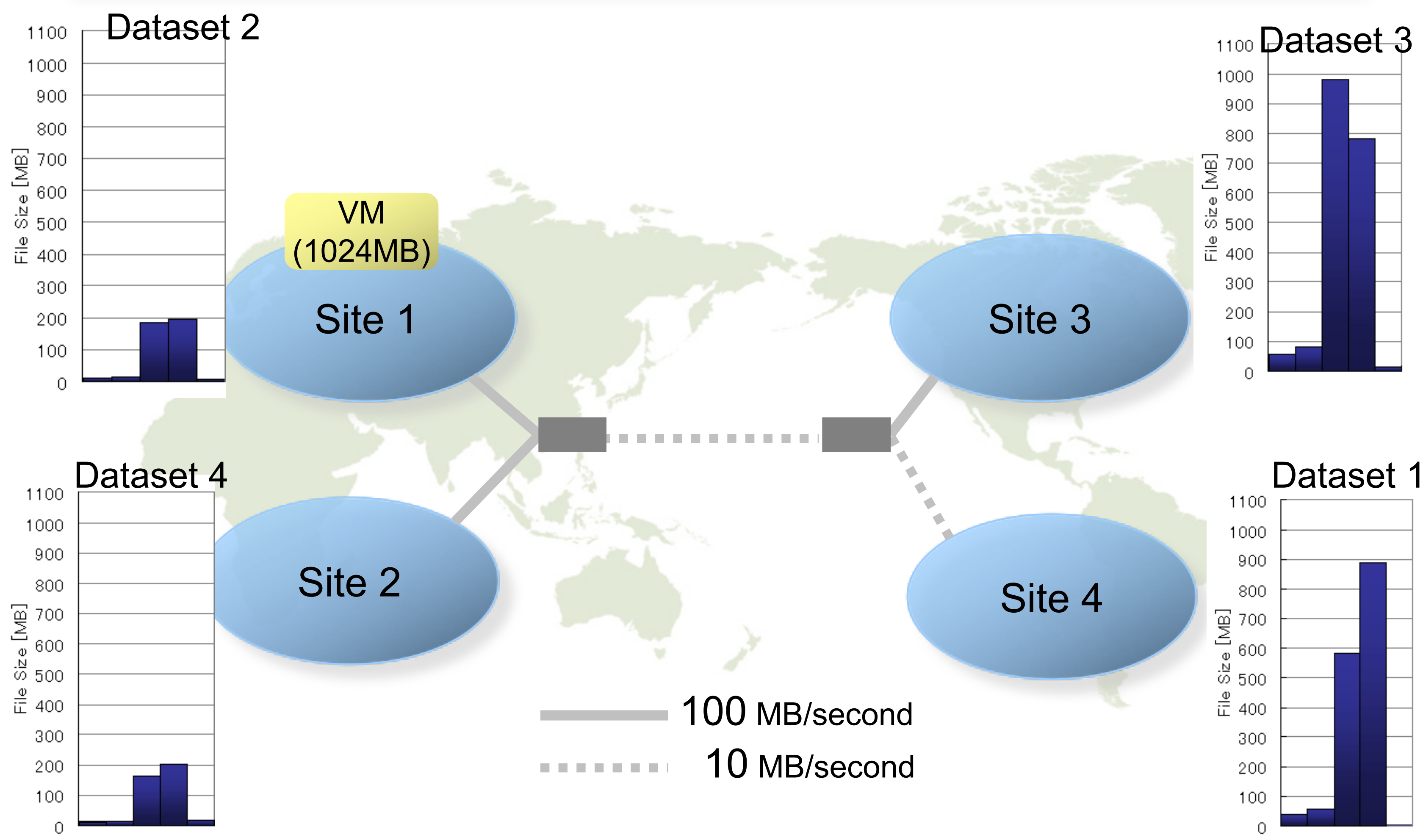
Experiment targets

Comparing a Total File Access time with following strategies

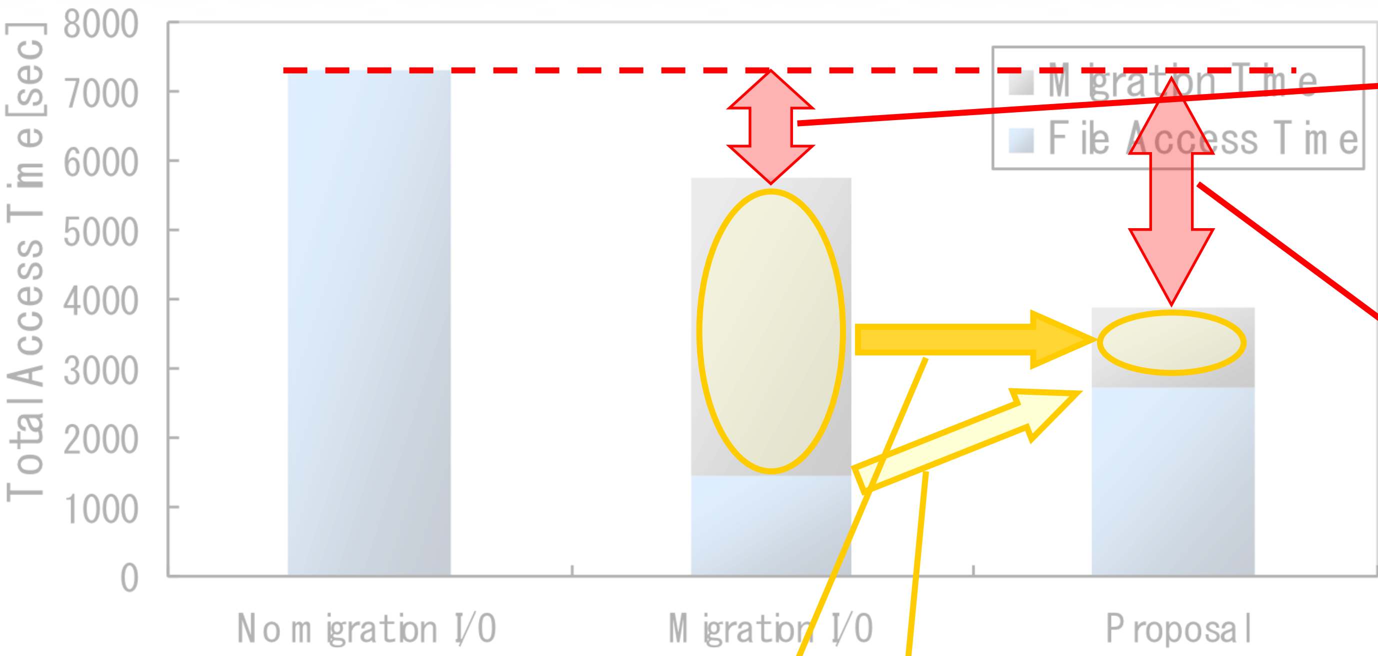
- No Migration I/O
  - Always accesses from the initial location (Site 1)
- Migration I/O
  - Always migrates VM onto sites that hold target file
- Proposal
  - Determine the VM migration strategy from our proposed algorithm

Experiment 1 : File size & location settings

large size dataset is located far from initial location



Experiment 1: Total file access time

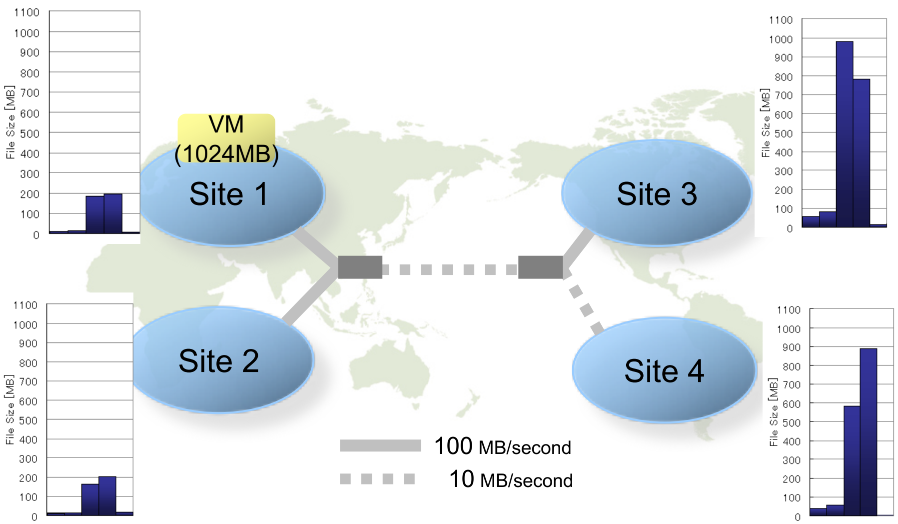


Reducing File access time by 21 %

Reducing File access time by 47 %

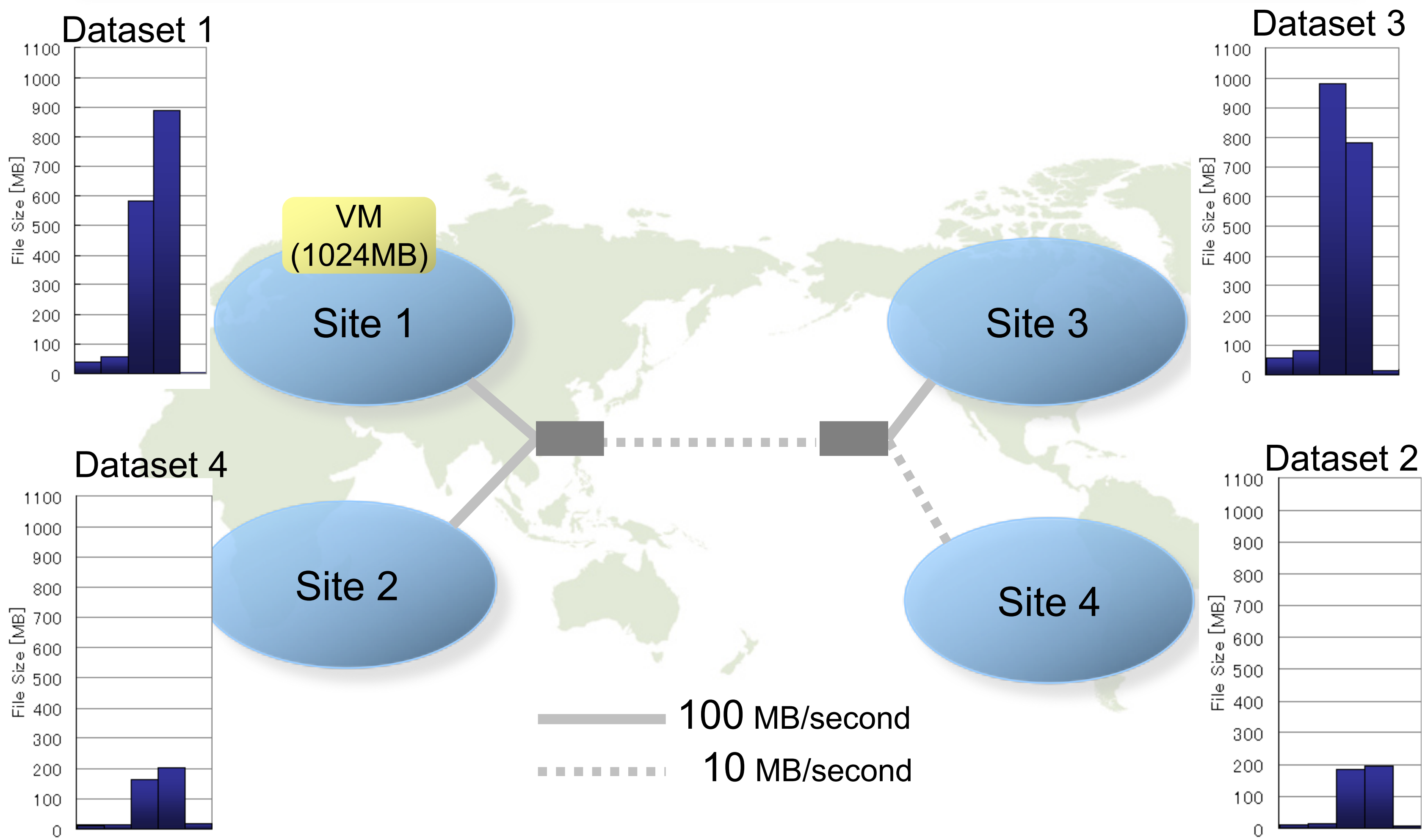
Reducing the total file access time more by avoiding unnecessary VM migration in WANs

Longer file access time than Migration I/O strategy

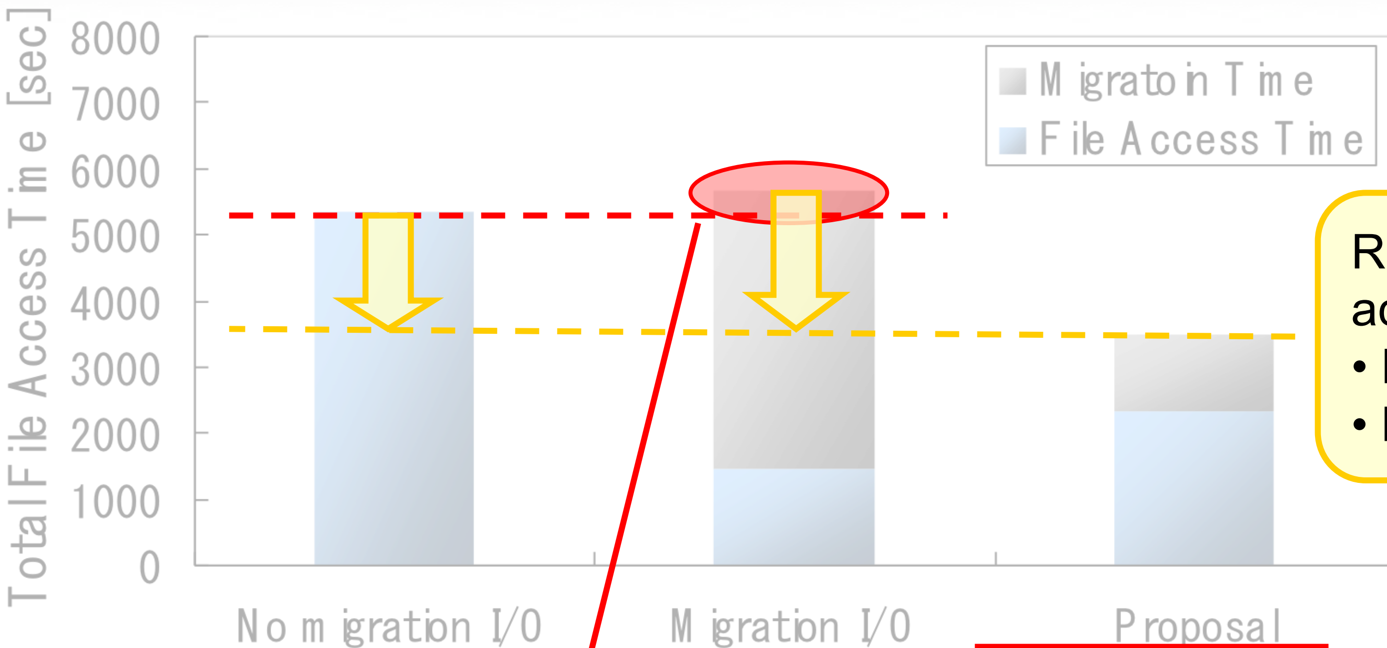


Experiment 2 : File size & location settings

One of large size dataset is located in initial location



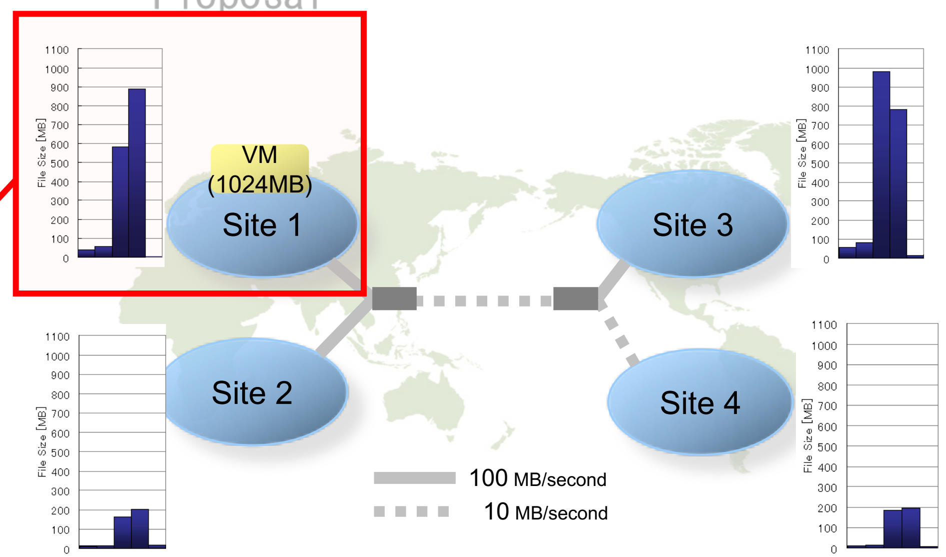
Experiment 2: Total file access time



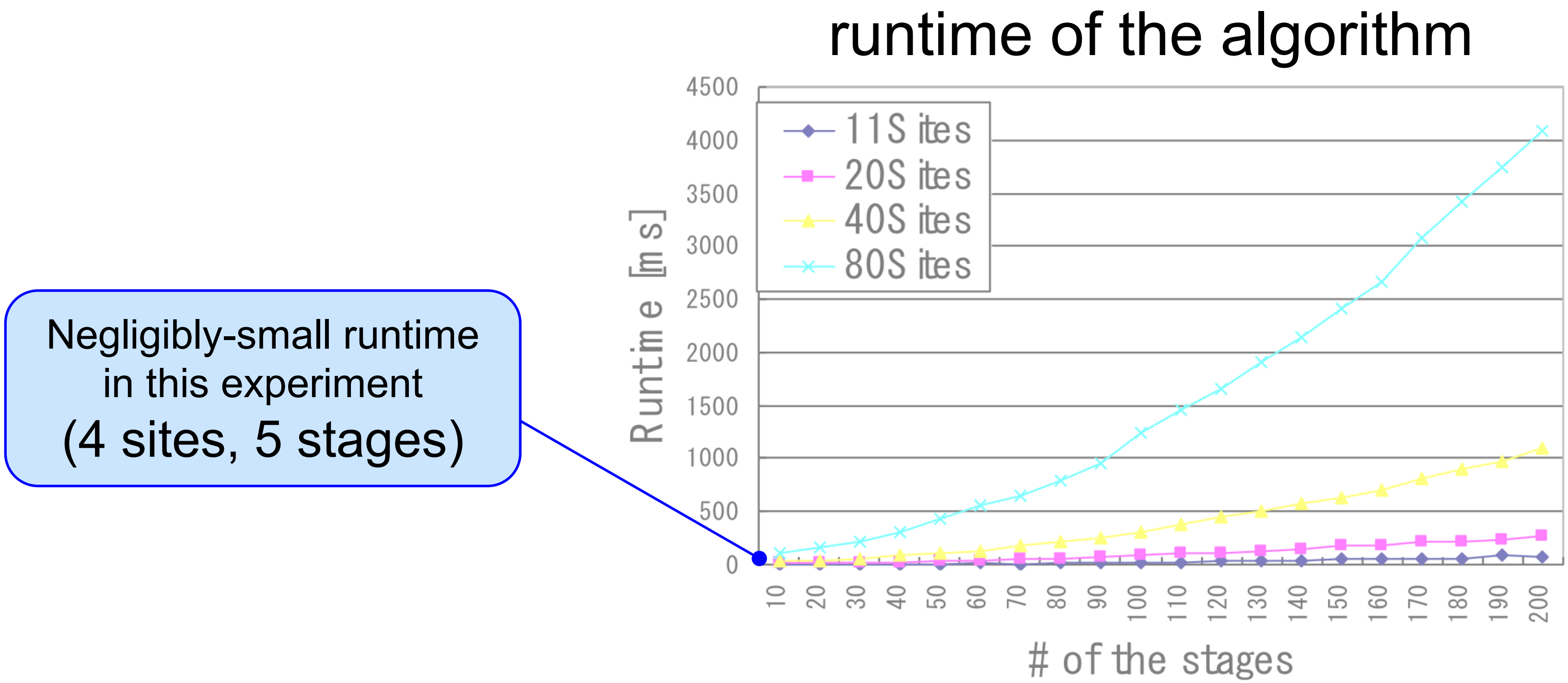
Reducing the total file access time
- No migration I/O: 34%
- Migration I/O: 38 %

Longer than the total access time of the No migration strategy

large size files are located on the same site to the initial location of the VM



Scalability



- For scalability, we can …
  - Set the maximum # of the stages to control the runtime
  - Reuse the results of the shortest path search
  - Solve the shortest path problem previously

Conclusion

- Created Performance model
  - File access time and VM migration time
- Proposed optimizing algorithm for I/O intensive application
  - Representing the access dependency between files as a markov model
  - Determining VM migration strategy
- Achieved higher performance than simple techniques
  - No migration: 38%
  - Always migration: 47%
- Our proposed algorithm is expected to be more effective for applications accessing TB-sized files and larger

Future work

- For the performance model
  - Considering CPU and memory usages for heterogeneous environments
- For the optimizing algorithm
  - Considering other VM placements
    - Load balancing
  - Considering a VM migration algorithm in conjunction with file migration

Thank you,

Any Questions ?

TOKYO TECH
Pursuing Excellence
東京工業大学　佐藤賢斗