China LUG 2018

Dynamic Forwarding Resource Allocation for High-end Supercomputers

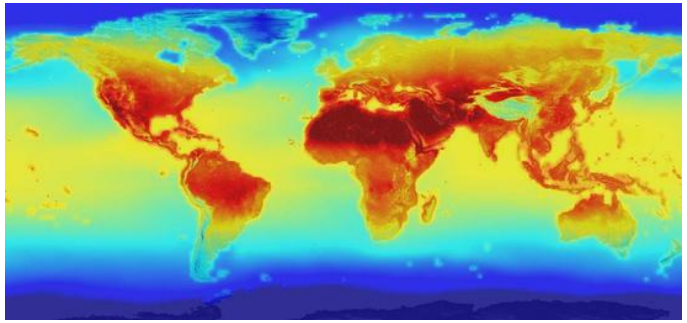
薛巍
清华大学、国家超级计算无锡中心
2018-10-23

Outline

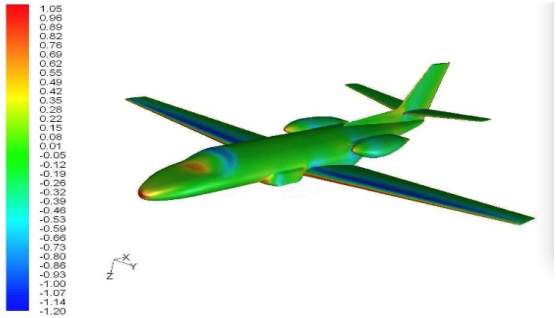
- Motivation
- Architecture
- Evaluation
- Conclusion

Supercomputing is Important

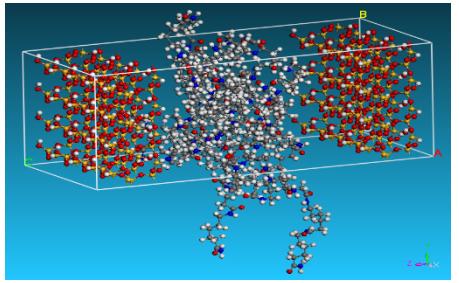
Climate simulation



Fluid dynamics



Molecular dynamics

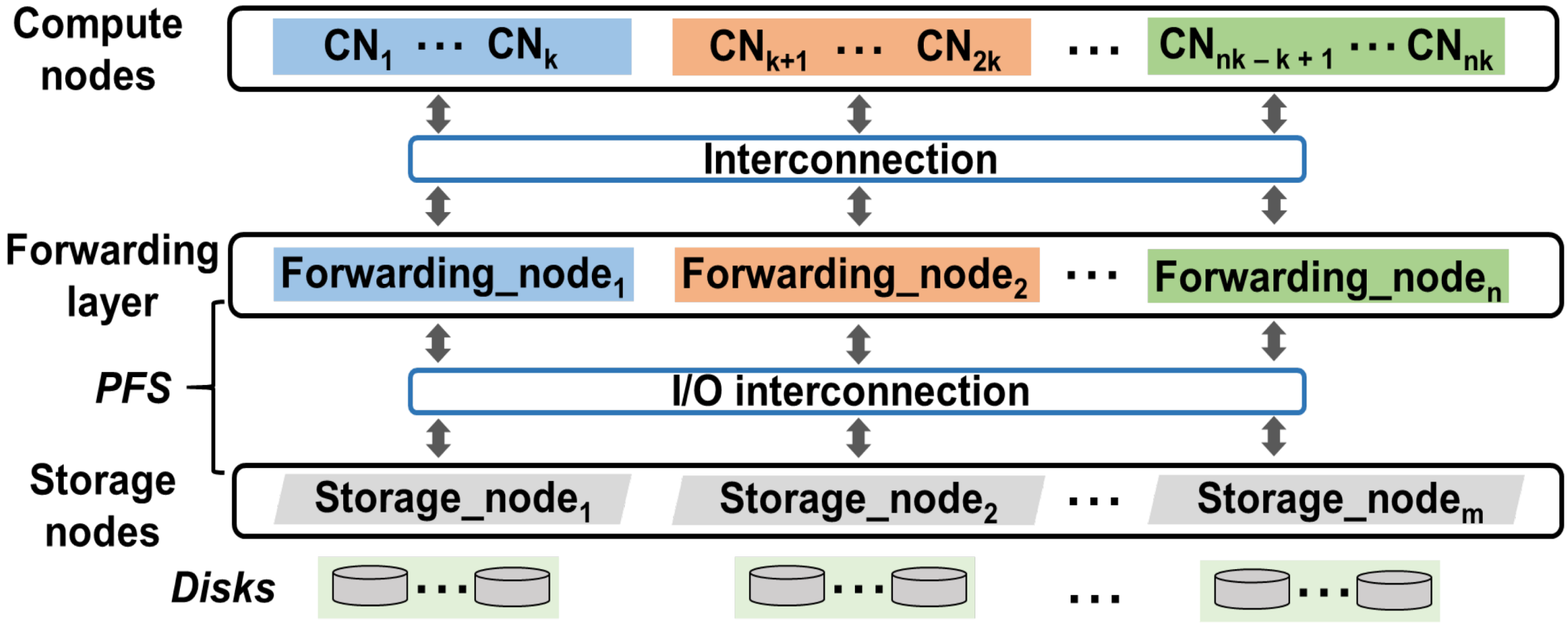


Sunway TaihuLight

Summit

Storage system plays a crucial role in supercomputers

I/O Forwarding Layer



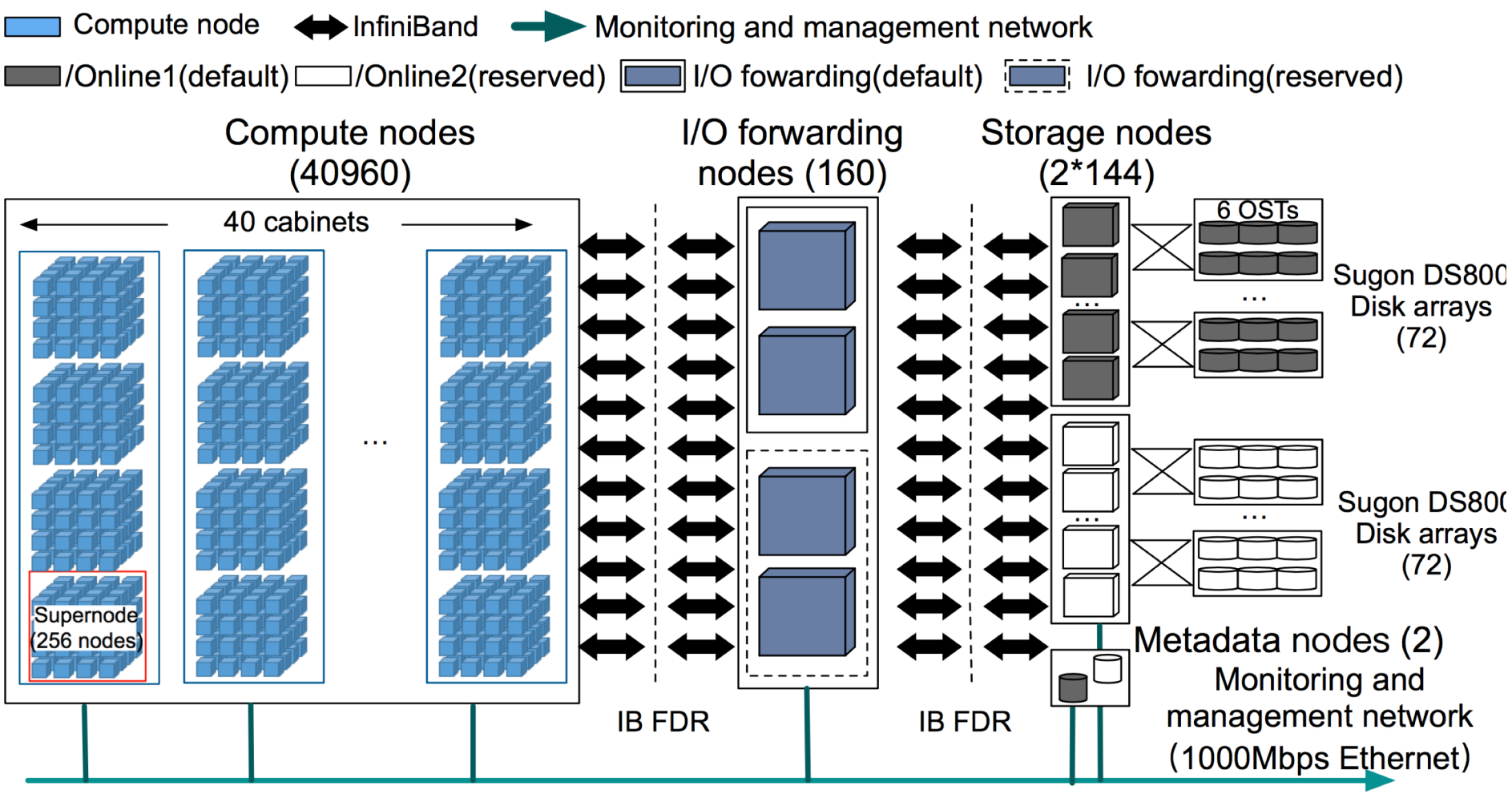
I/O Forwarding Layer

I/O forwarding layer is widely adopted by state-of-art supercomputers

-9 out of top 20 supercomputers adopt I/O forwarding (Jun 2018)

Rank	Machine	Type	File system
2	Taihulight [17]	Sunway	Lustre
4	Tianhe-2A [79]	Tianhe	Lustre+H2FS
6	Piz Daint [13]	Cray	Lustre+GPFS
7	Titan [18]	Cray	Lustre
8	Sequoia [15]	IBM BlueGene	Lustre
9	Trinity [19]	Cray	Lustre
10	Cori [1]	Cray	Lustre+GPFS
16	K computer [6]	Fujitsu	Lustre
17	Mira [9]	IBM BlueGene	GPFS

A Case Study: Sunway TaihuLight (TOP500 No.2)



I/O Forwarding Layer

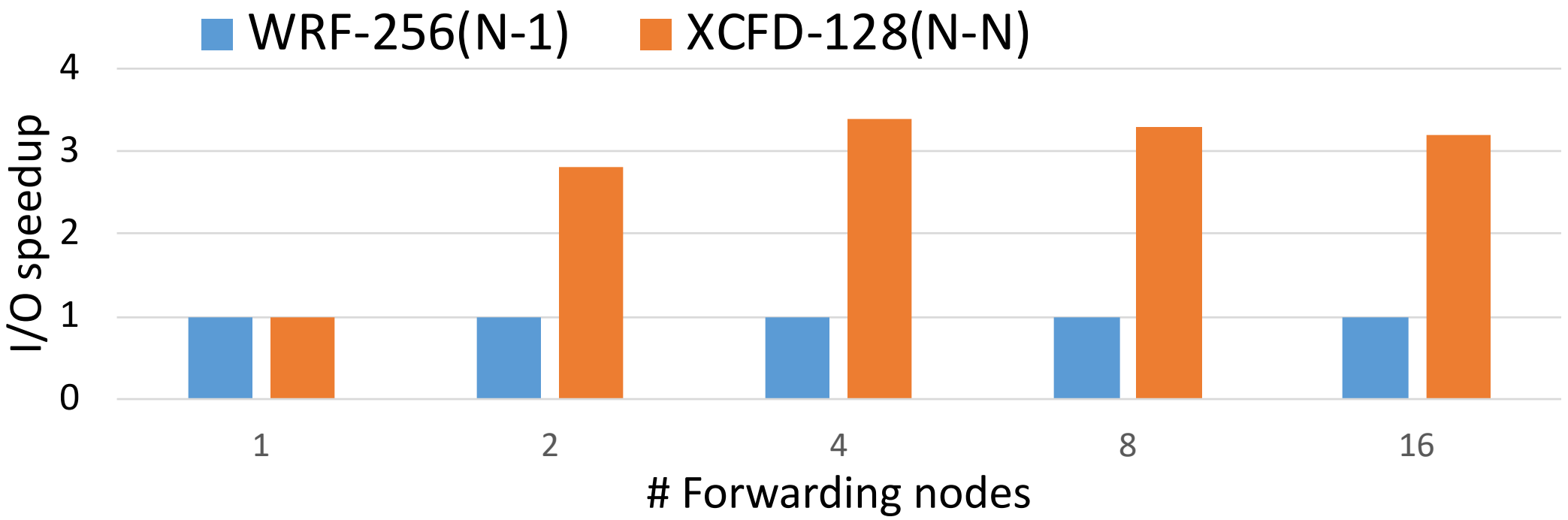
Advantage

- Reduce network connections –> *improve stability*
- More pure HPC OS -> ***better*** *computation performance*
- Additional prefetching/caching -> ***better*** *I/O performance*

Is I/O forwarding layer good enough?

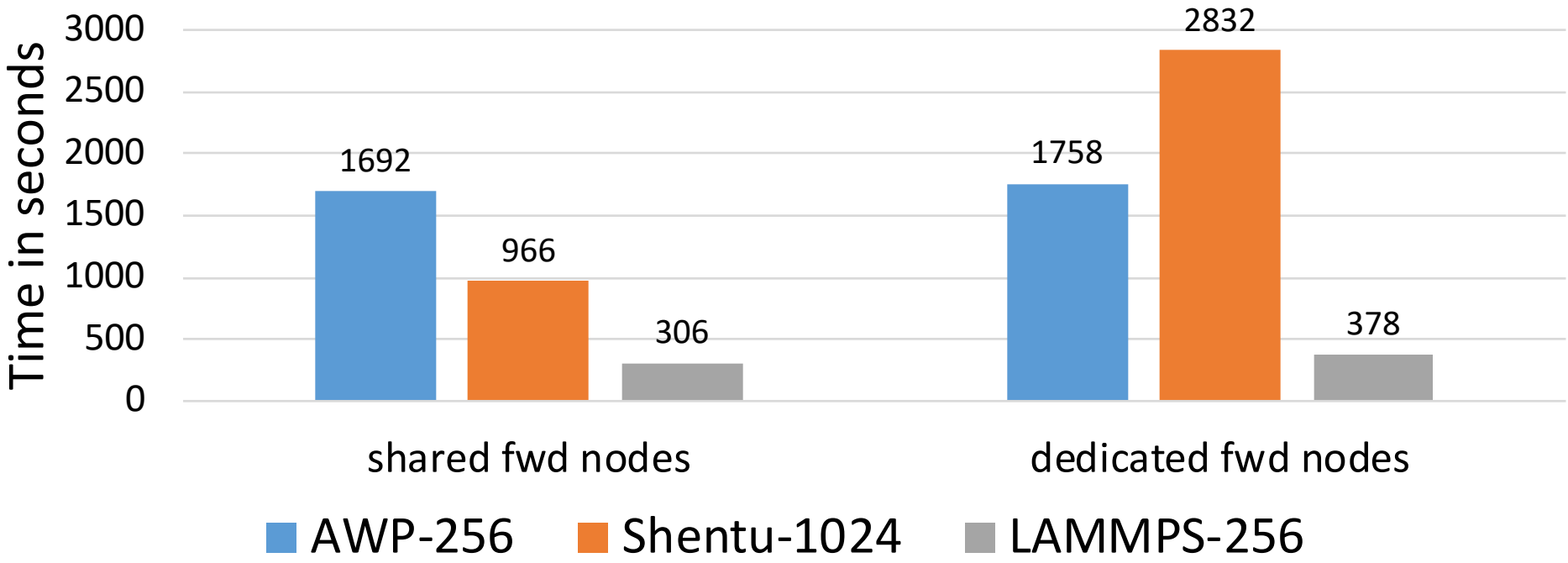
- Unclear requirement at first: How many forwarding nodes?
  More forwarding nodes, more cost – we are facing budget limitation
- From our operation: Forwarding layer always meet with *performance issues*

Issue 1: Resource Misallocation



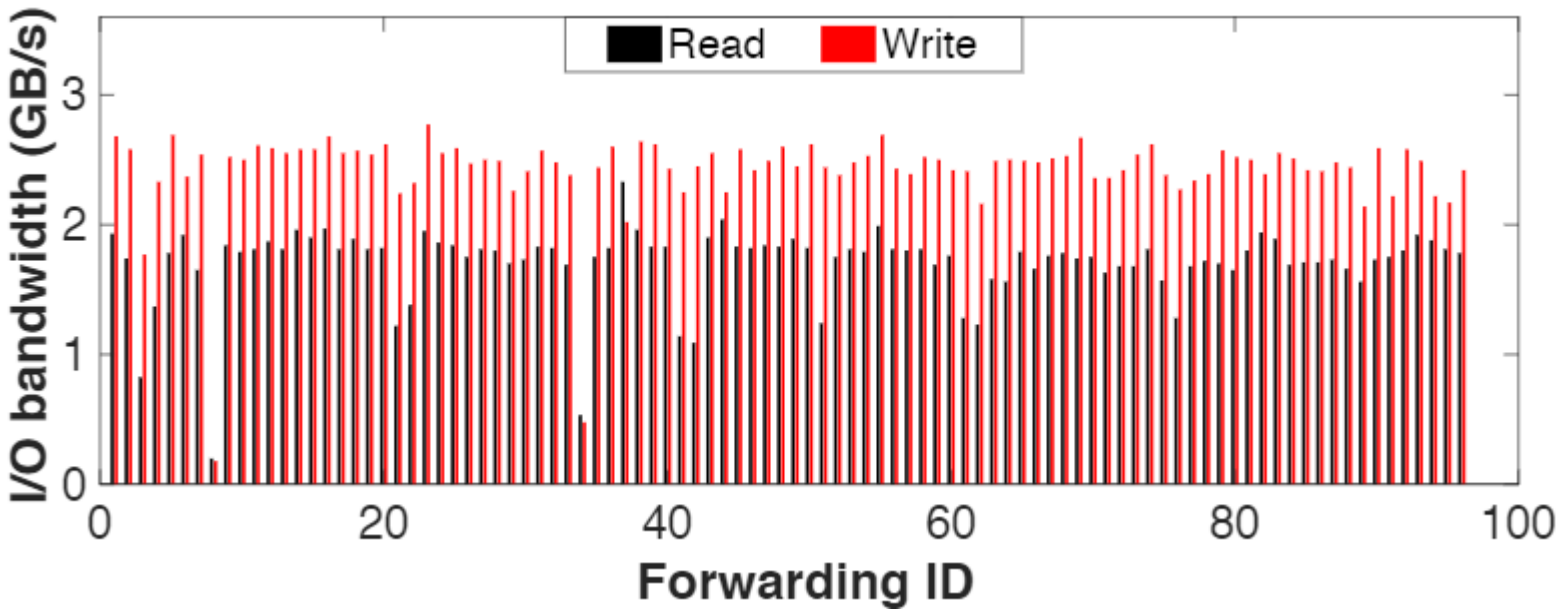
Different Apps have different I/O forwarding demands!

Issue 2: Interference



Applications may suffer from I/O interference at I/O forwarding layer!

Issue 3: Performance anomaly



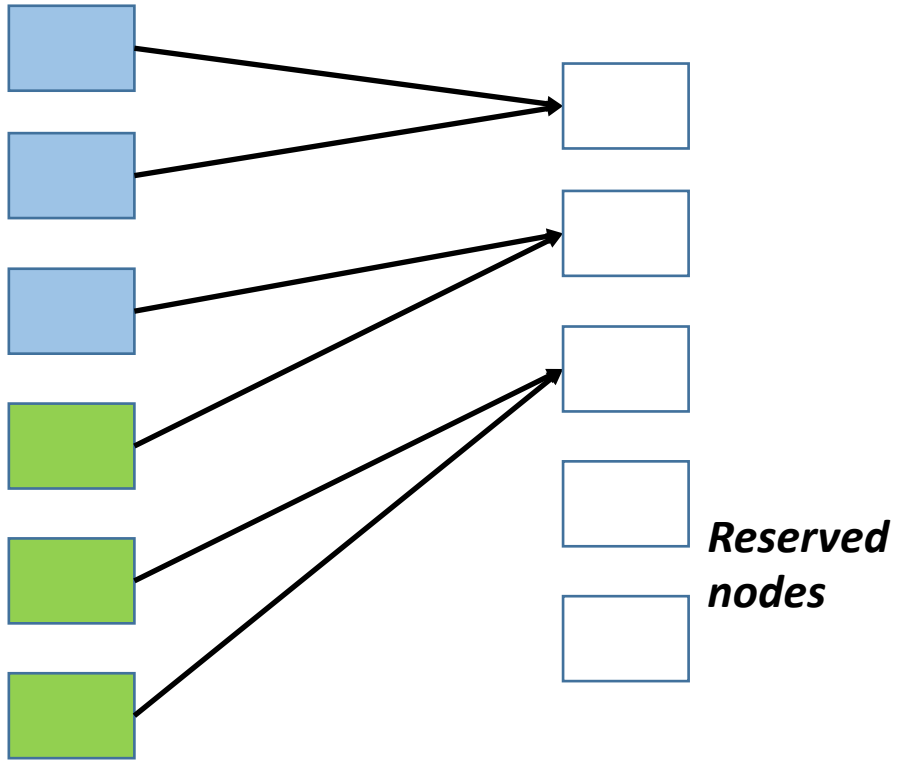
Forwarding nodes with IDs 8 and 34 are abnormal nodes, only delivering about 10% performance comparing to normal ones

Solution

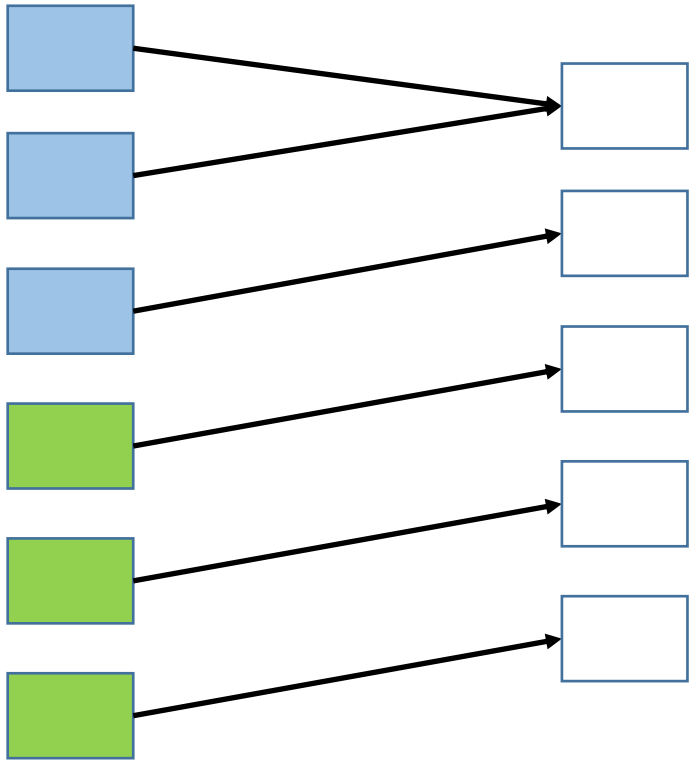
Dynamic forwarding resource allocation (DFRA)

App A *App B*

FFM(Fixed forwarding mapping)



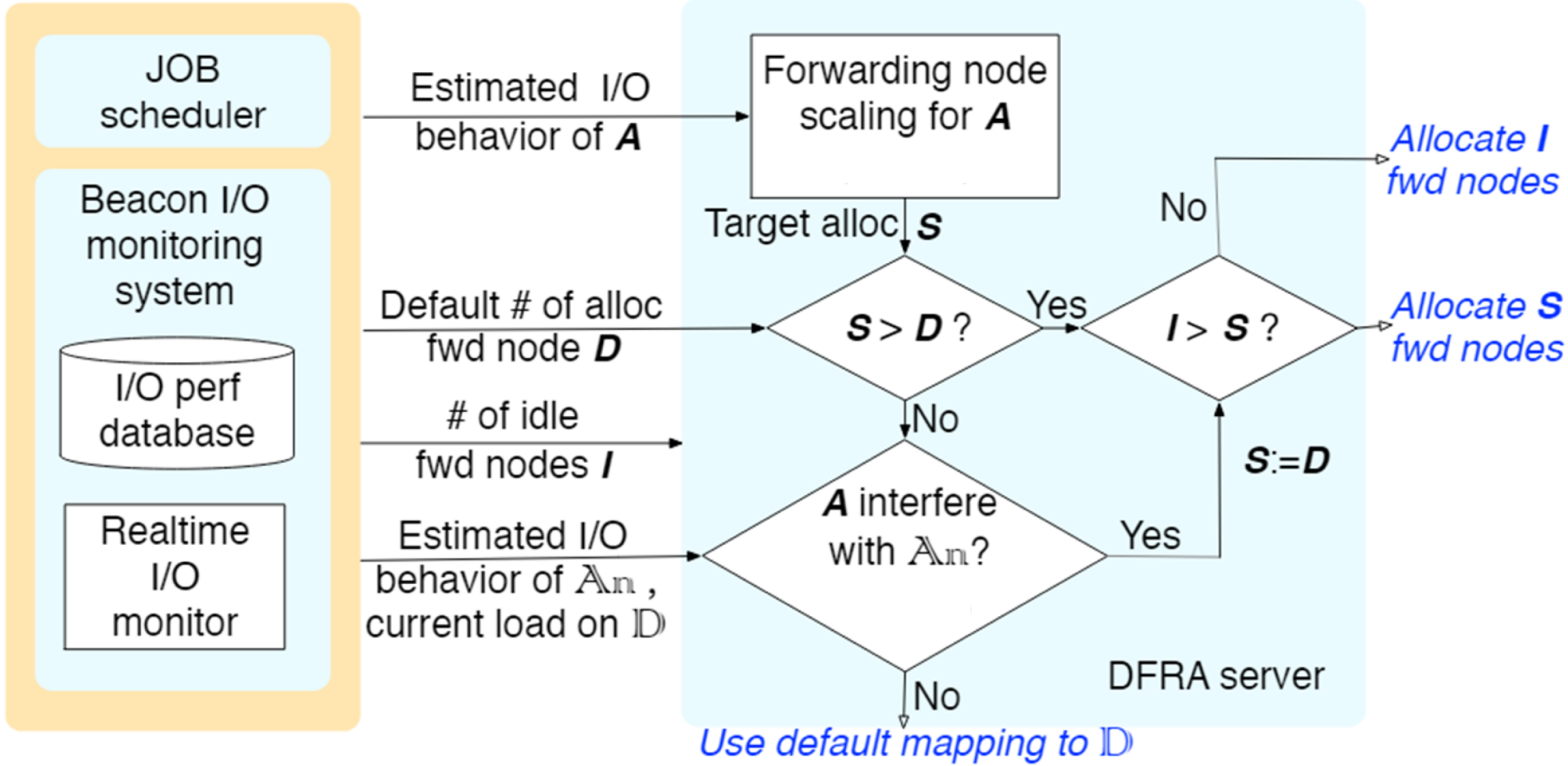
DFRA (with help of app I/O history)



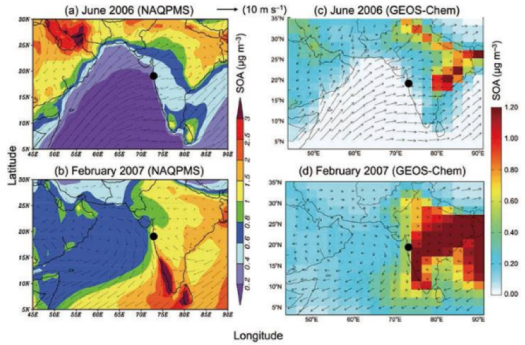
DFRA workflow

How to detect I/O interference?

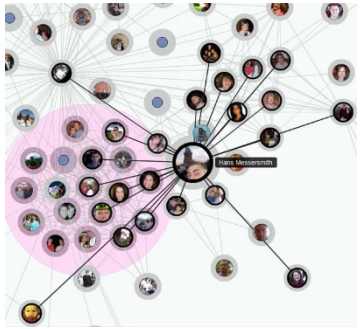
How to find I/O-intensive workloads? And How many forwarding nodes?



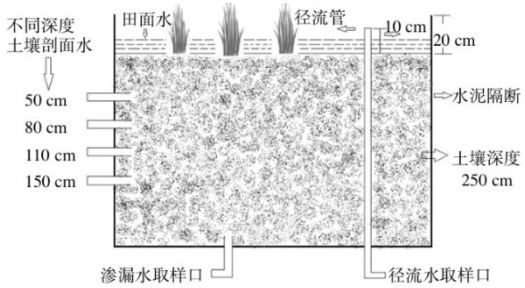
Applications



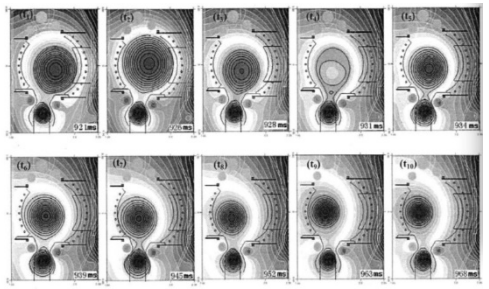
CAM (2017 GB finalist) atmospheric model



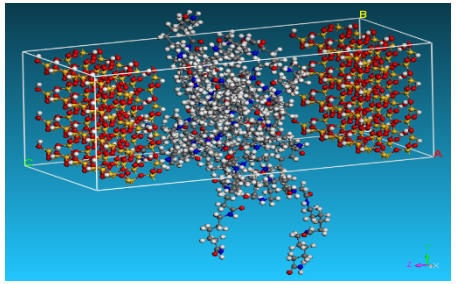
Shentu (2018 GB finalist) graph engine



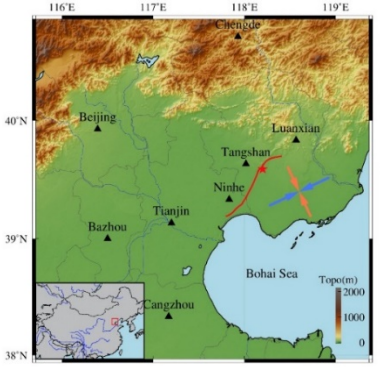
DNDC agro- ecosystems



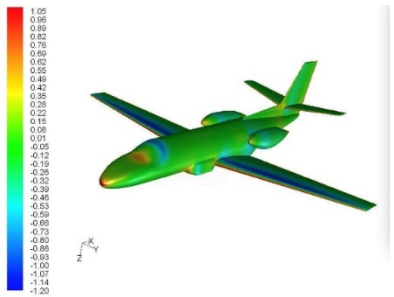
APT particle dynamics



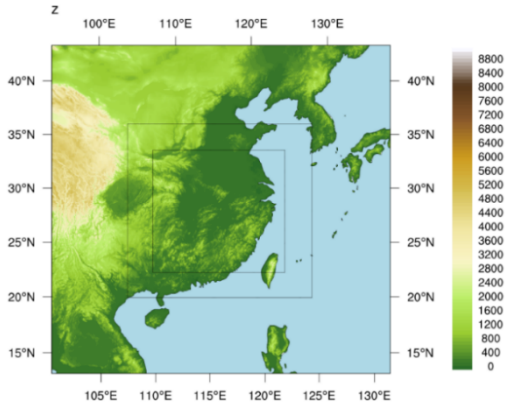
LAMMPS Molecular dynamics



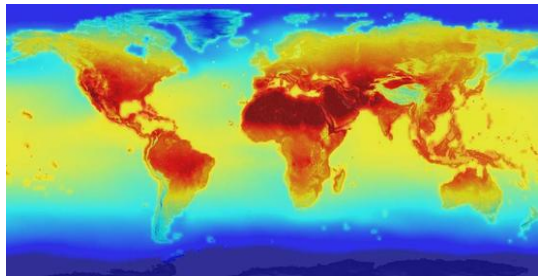
AWP (2017 GB prize) **Mrcdrp** earthquake simulation



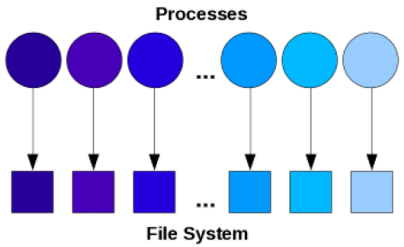
XCFD fluid dynamics



WRF weather forecasting



CESM climate simulator

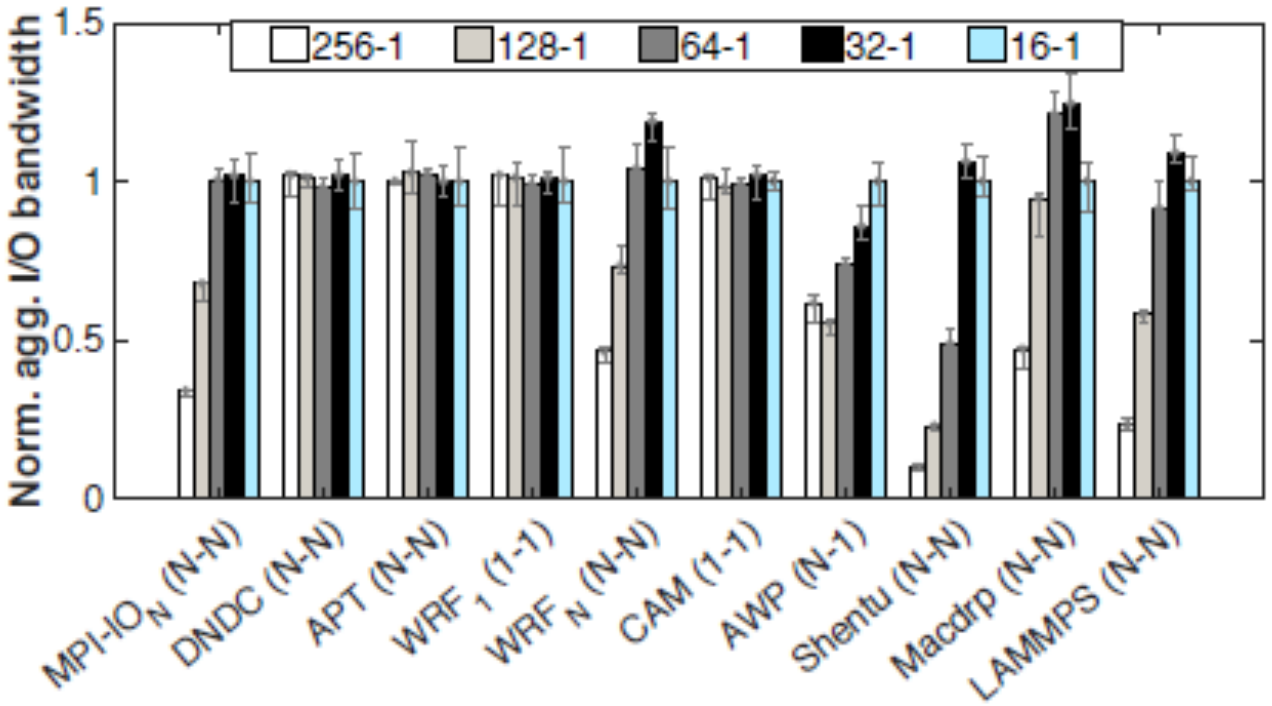


MPI-IO benchmark

Application IO Characteristics

App	Throughput	IOPS	Metadata	I/O Lib	I/O Mode
MPI-IO$_N$	High	Low	Low	MPI-IO	N-N
DNDC	Low	Low	High	POSIX	N-N
APT	Low	High	Low	HDF5	N-N
WRF$_1$	Low	Low	Low	NetCDF	1-1
WRF$_N$	High	High	Low	NetCDF	N-N
CAM	Low	Low	Low	NetCDF	1-1
AWP	Low	Low	Low	MPI-IO	N-1
Shentu	High	High	Low	POSIX	N-N
Macdrp	High	Low	Low	POSIX	N-N
LAMMPS	High	Low	Low	MPI-IO	N-N
XCFD	High	Low	Low	POSIX	N-N
CESM	High	Low	Low	NetCDF	N-N
swDNN	Low	Low	Low	HDF5	N-N

Automatic Forwarding Node Scaling

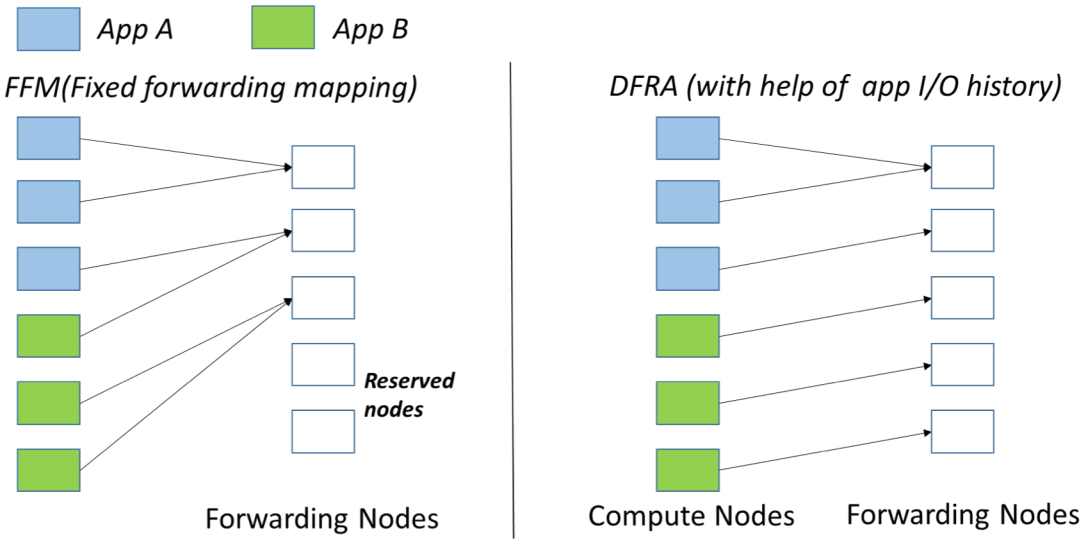


- 1-1 jobs don’t scale
- **DNDC** doesn’t scale due to metadata bound
- **APT** doesn’t scale due to few I/O nodes

Automatic Forwarding Node Scaling

Application is forwarding node scaling **(*App B*)** when it

- Has enough I/O volume (I/O volume > V_{min}) &
- Has enough I/O nodes (I/O nodes > N_{min}) &
- Is ***not*** metadata bound (Metadata queue length < $W_{metadata}$)



I/O Interference Study

Apps	MPI-IO$_N$	APT	DNDC	WRF$_1$	WRF$_N$	Shentu	CAM	AWP
MPI-IO$_N$	*(2.1,2.1)	**(1.1,9.3)**	**(4.8,1.1)**	(1.0,1.0)	*(2.1,2.0)	**(1.3,4.5)**	(1.0,1.0)	**(3.3,1.1)**
APT	-	*(2.0,2.1)	**(33.3,1.0)**	(1.0,1.0)	**(4.3,1.4)**	**(6.3,1.3)**	(1.0,1.0)	**(50.0,1.1)**
DNDC	-	-	*(2.0,2.0)	**(1.0,25.0)**	**(1.0,11.1)**	**(1.1,16.7)**	**(1.0,33.3)**	*(2.2,2.4)
WRF$_1$	-	-	-	(1.0,1.0)	(1.0,1.0)	(1.0,1.0)	(1.0,1.0)	**(50.0,1.0)**
WRF$_N$	-	-	-	-	*(2.1,2.1)	*(2.0,2.3)	(1.0,1.0)	**(12.5,1.3)**
Shentu	-	-	-	-	-	*(2.0,2.0)	(1.0,1.0)	**(12.5,1.1)**
CAM	-	-	-	-	-	-	(1.0,1.0)	**(100.0,1.0)**
AWP	-	-	-	-	-	-	-	*(2.0,2.0)

I/O slowdown factor pairs

- App exhibits ~2X slowdown when co-running with itself
- Apps (L-Throughput, L-IOPS, L-Metadata) do not suffer, but exception with AWP(N-1)

I/O Interference Study

Apps	$MPI\text{-}IO_N$	APT	DNDC	WRF_1	WRF_N	Shentu	CAM	AWP
$MPI\text{-}IO_N$	*(2.1,2.1)	**(1.1,9.3)**	**(4.8,1.1)**	(1.0,1.0)	*(2.1,2.0)	**(1.3,4.5)**	(1.0,1.0)	**(3.3,1.1)**
APT	-	*(2.0,2.1)	**(33.3,1.0)**	(1.0,1.0)	**(4.3,1.4)**	**(6.3,1.3)**	(1.0,1.0)	**(50.0,1.1)**
DNDC	-	-	*(2.0,2.0)	**(1.0,25.0)**	**(1.0,11.1)**	**(1.1,16.7)**	**(1.0,33.3)**	*(2.2,2.4)
WRF_1	-	-	-	(1.0,1.0)	(1.0,1.0)	(1.0,1.0)	(1.0,1.0)	**(50.0,1.0)**
WRF_N	-	-	-	-	*(2.1,2.1)	*(2.0,2.3)	(1.0,1.0)	**(12.5,1.3)**
Shentu	-	-	-	-	-	*(2.0,2.0)	(1.0,1.0)	**(12.5,1.1)**
CAM	-	-	-	-	-	-	(1.0,1.0)	**(100.0,1.0)**
AWP	-	-	-	-	-	-	-	*(2.0,2.0)

- N-1 is notoriously slow and brings high disturbance

- Forwarding node maintains fixed thread pool, N-1 generates lots of requests and flood thread pool
- Requests are further prolonged by slow Lustre backend processing

I/O Interference Study

Apps	MPI-IO$_N$	APT	DNDC	WRF$_1$	WRF$_N$	Shentu	CAM	AWP
MPI-IO$_N$	*(2.1,2.1)	**(1.1,9.3)**	**(4.8,1.1)**	(1.0,1.0)	*(2.1,2.0)	**(1.3,4.5)**	(1.0,1.0)	**(3.3,1.1)**
APT	-	*(2.0,2.1)	**(33.3,1.0)**	(1.0,1.0)	**(4.3,1.4)**	**(6.3,1.3)**	(1.0,1.0)	**(50.0,1.1)**
DNDC	-	-	*(2.0,2.0)	**(1.0,25.0)**	**(1.0,11.1)**	**(1.1,16.7)**	**(1.0,33.3)**	*(2.2,2.4)
WRF$_1$	-	-	-	(1.0,1.0)	(1.0,1.0)	(1.0,1.0)	(1.0,1.0)	**(50.0,1.0)**
WRF$_N$	-	-	-	-	*(2.1,2.1)	*(2.0,2.3)	(1.0,1.0)	**(12.5,1.3)**
Shentu	-	-	-	-	-	*(2.0,2.0)	(1.0,1.0)	**(12.5,1.1)**
CAM	-	-	-	-	-	-	(1.0,1.0)	**(100.0,1.0)**
AWP	-	-	-	-	-	-	-	*(2.0,2.0)

- H-metadata applications (eg, DNDC) generate heavy interference to concurrent workloads

Open/close requests pile up and block those from other apps

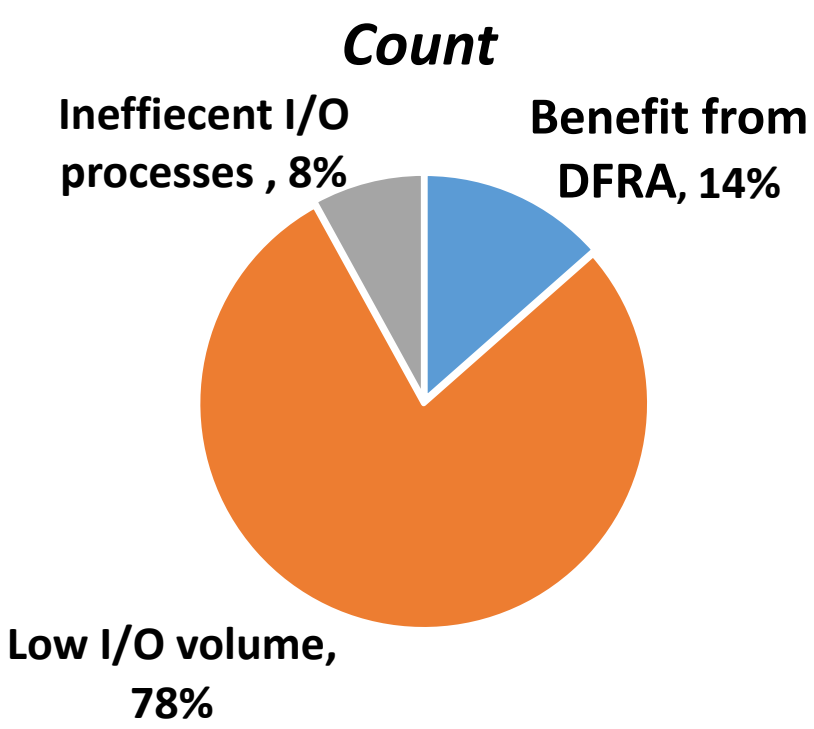
I/O Interference Study

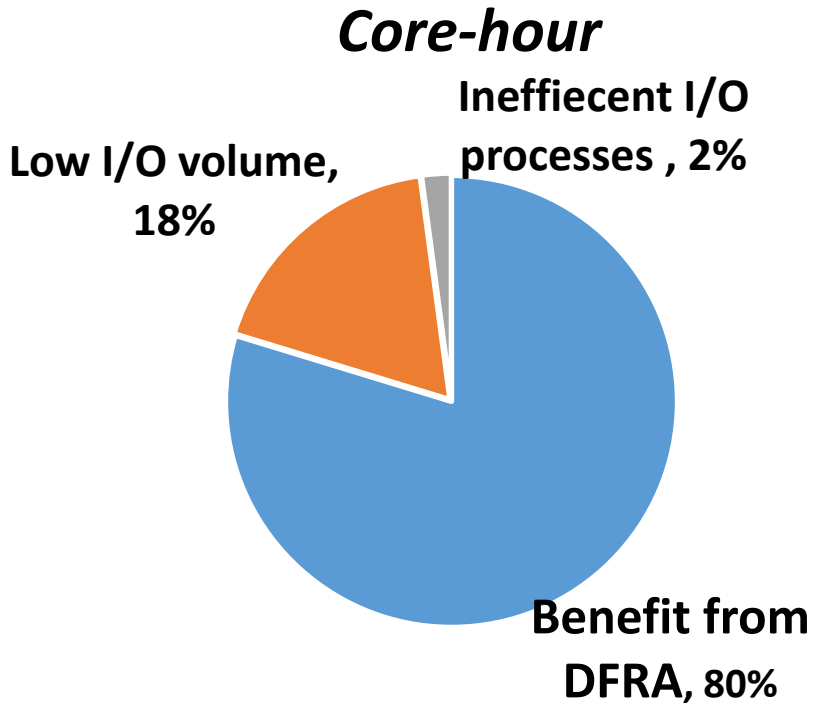
Detect I/O interference by checking whether ***App A*** and ***its neighbors*** is

- N-1 I/O mode; or
- Metadata-heavy; or
- High-bandwidth or High-IOPS

Evaluation

18 months I/O history analysis (Apr 2017 – Sep 2018)

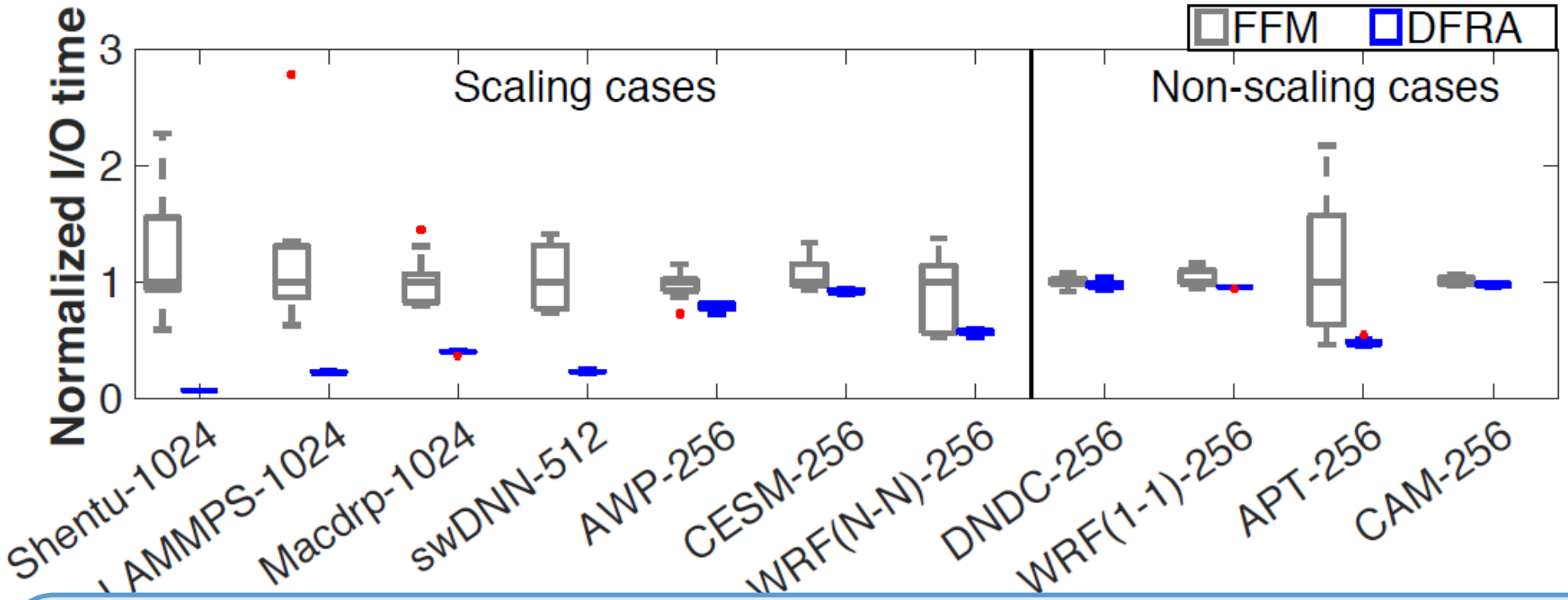




Few jobs consuming considerable corehours are benefiting from DFRA

Evaluation

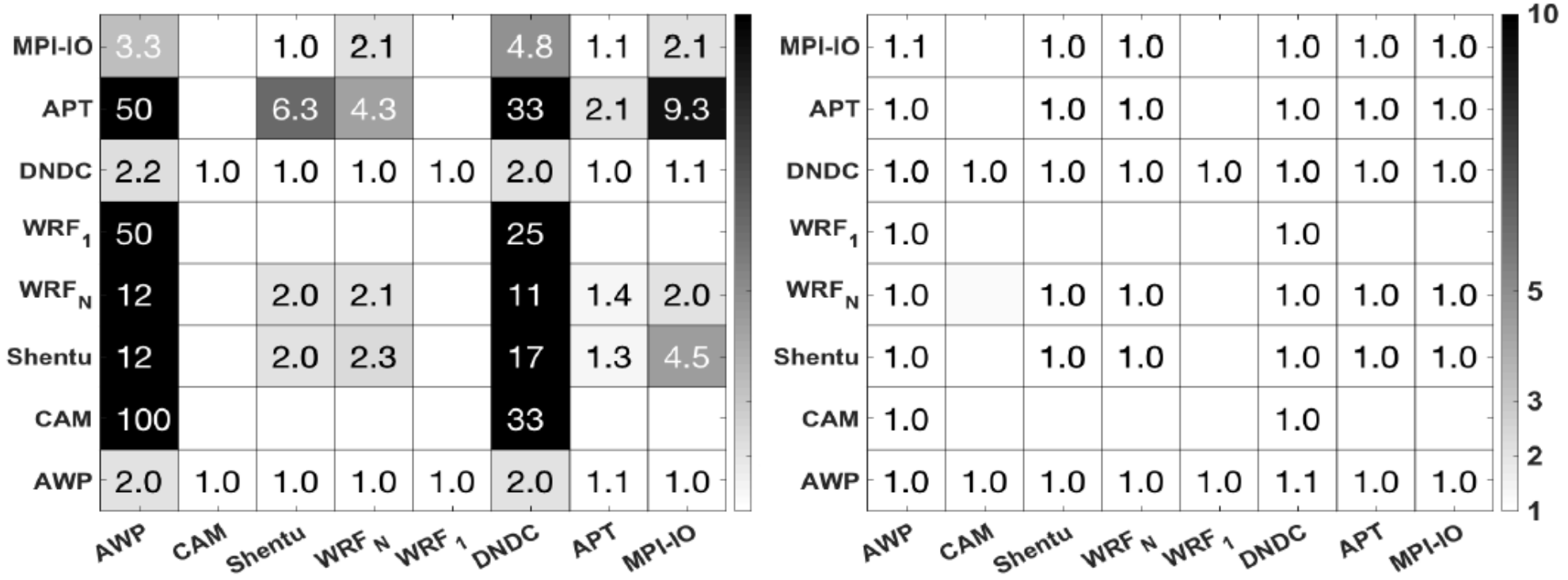
Overall performance from 10 runs (each box) in production environment



- ✓ The scaling group receives higher speedup (average at 4.5 and up to 16.0), while nonscaling applications benefit more from reduced performance variability
- ✓ An average I/O speedup of 3.3 across all 11 applications, from 1.03(CAM) to 16.0(Shentu).

DFRA reduces I/O variation and improves I/O performance at I/O forwarding layer

Evaluation

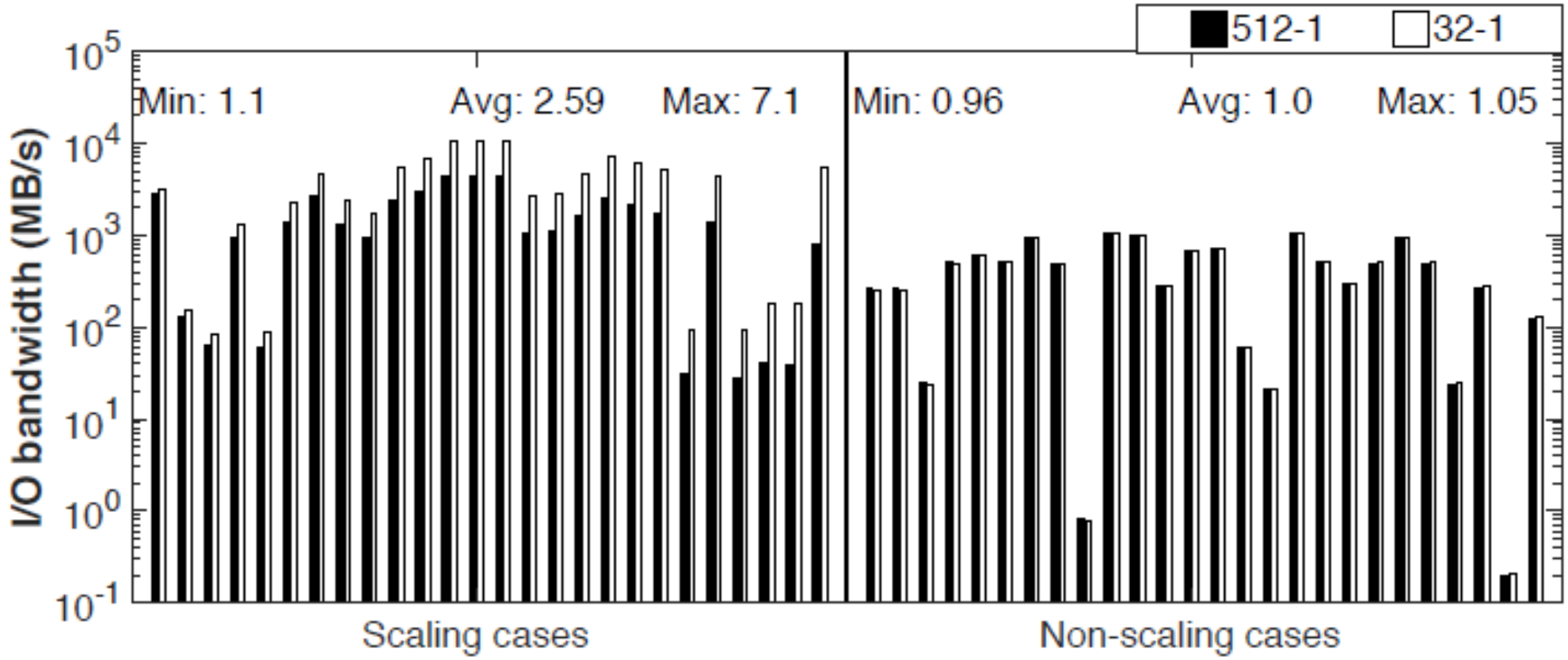


Before DFRA

After DFRA

DFRA reduce the variation and improve the I/O performance at I/O forwarding layer

Evaluation

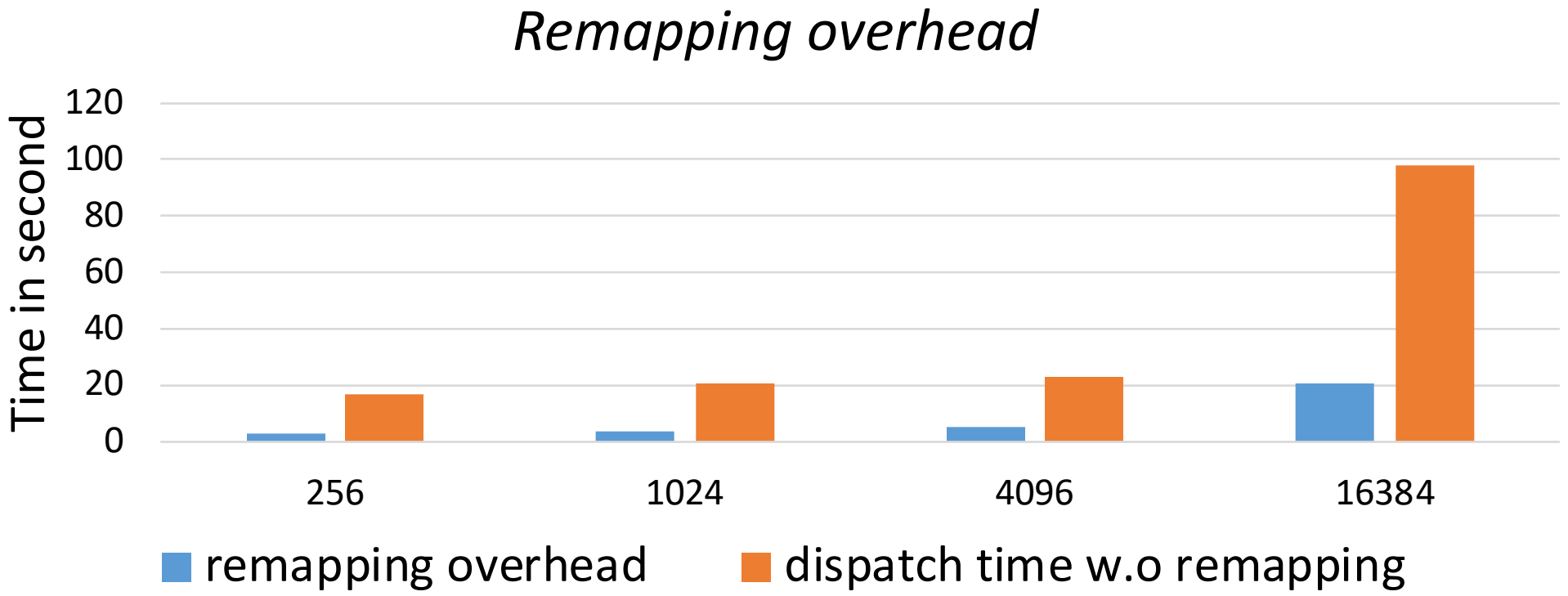


Scaling decision validation using MPI-IO benchmark

Instances (different IO mode, IO parallelism, IO request sizes, metadata ops), which run with 2 different compute-to-forwarding mapping ratios: 512-1 and 32-1.

The final results for both cases are consistent with the estimations projected by DFRA.

Evaluation



Remapping overhead is acceptable comparing to total core-hour saving (200 million core-hours in past 8 months)

Thanks for your time

Q&A