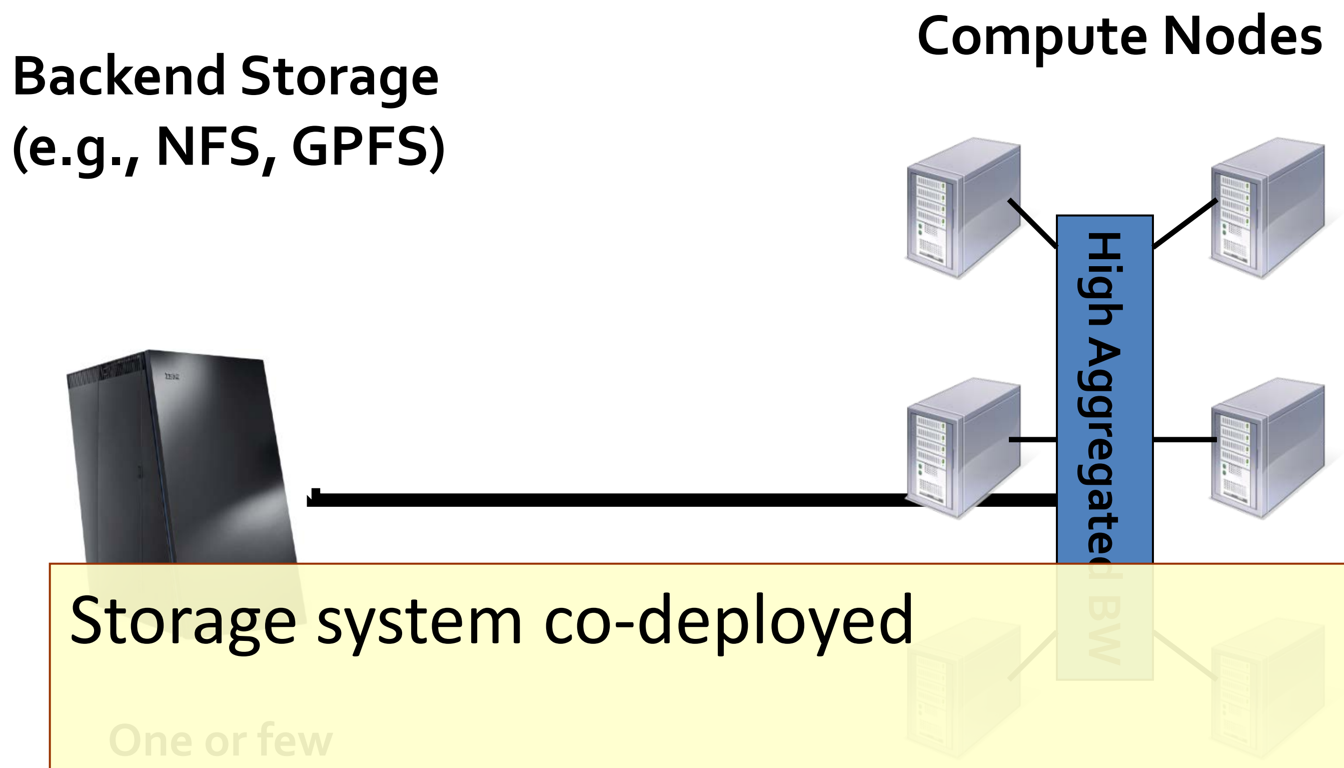
Predicting Intermediate Storage Performance for Workflow Applications

Lauro B. Costa, Samer Al-Kiswany,
Abmar Barros*, Hao Yang, and Matei Ripeanu

University of British Columbia
*UFCG, Brazil

PSDW'13 Nov, 18th co-located with SC'13

Storage System



Storage system co-deployed

Avoid backend storage as bottleneck

Opportunity to configure per application

Storage System Configuration

Different storage parameters

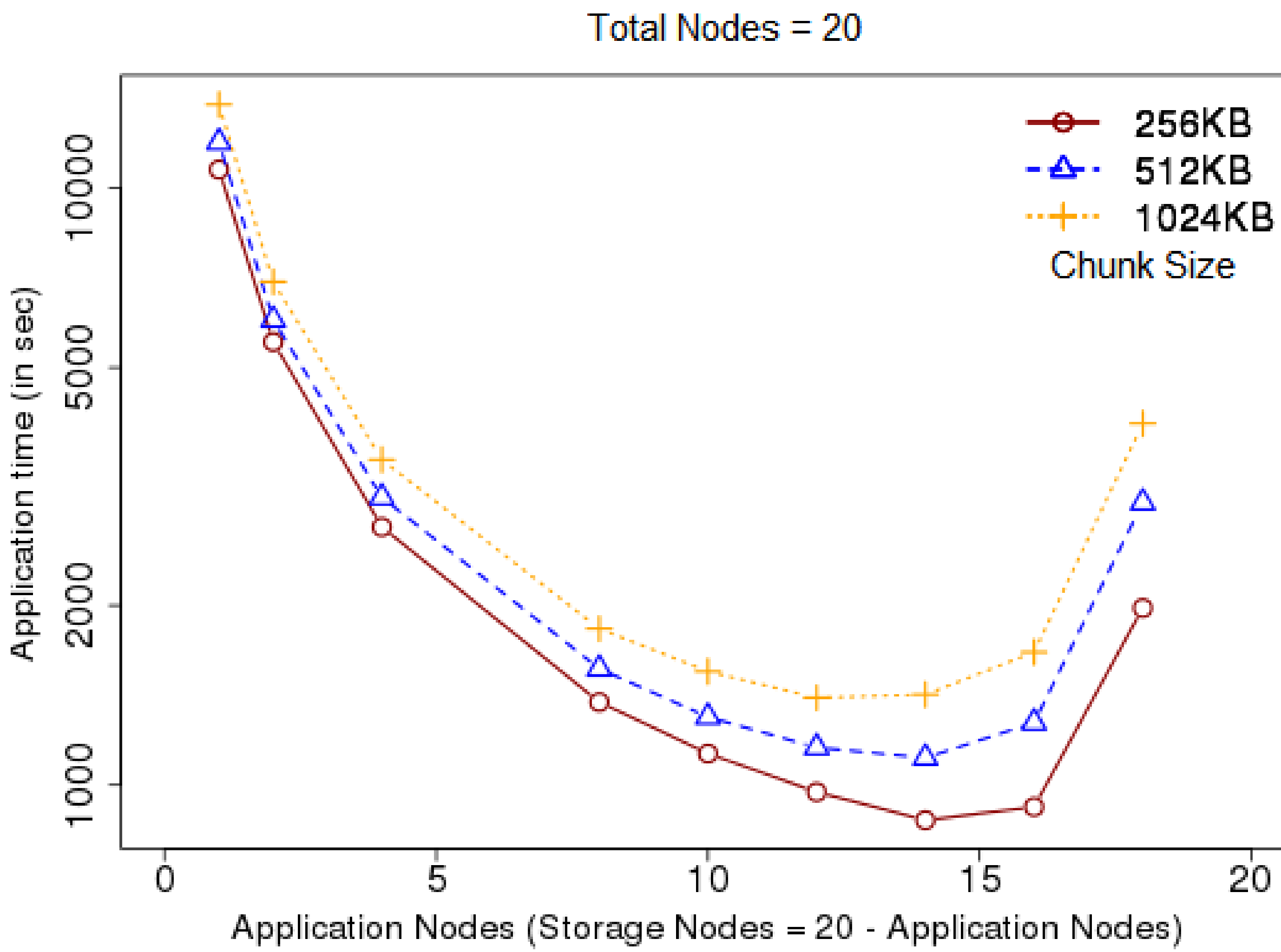
e.g., data placement, #nodes, chunk size

Benefit different workloads

e.g., data sharing, I/O intensive, read/write size

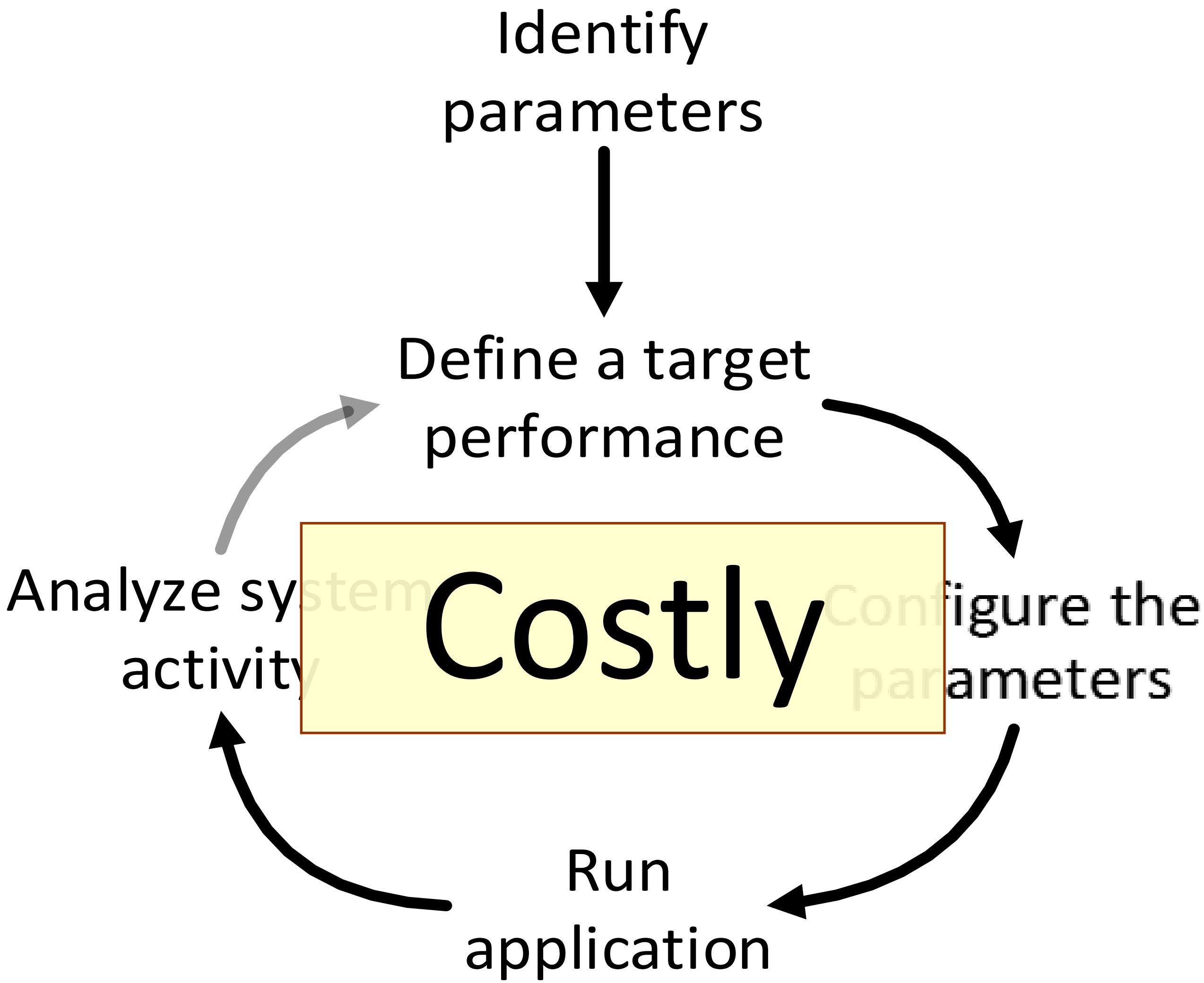
Proper choice of parameters depend on the workload

BLAST Example

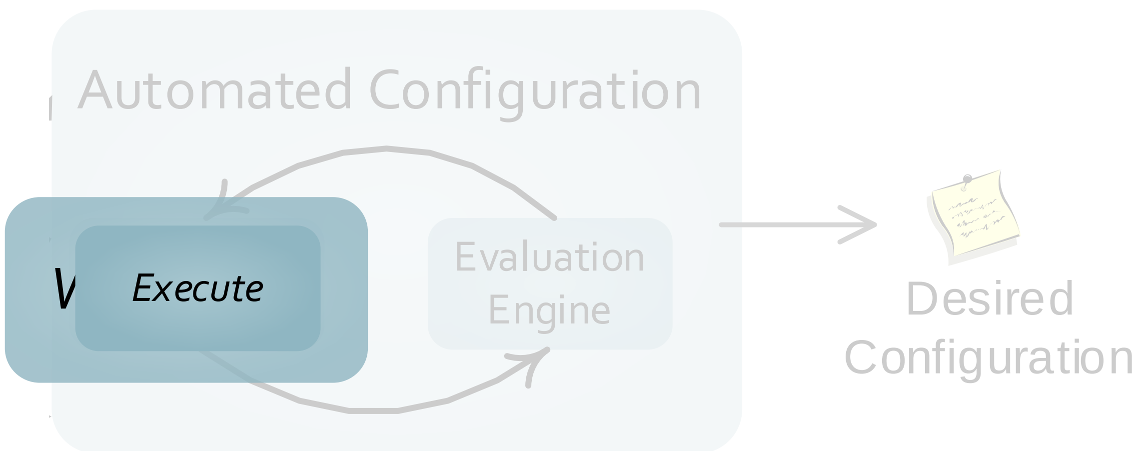


How to **support** the intermediate storage **configuration?**

Configuration Loop



Automating the Configuration Loop



Predictor Requirements

Accuracy

Response Time/Resource Usage

Usability

What…If…

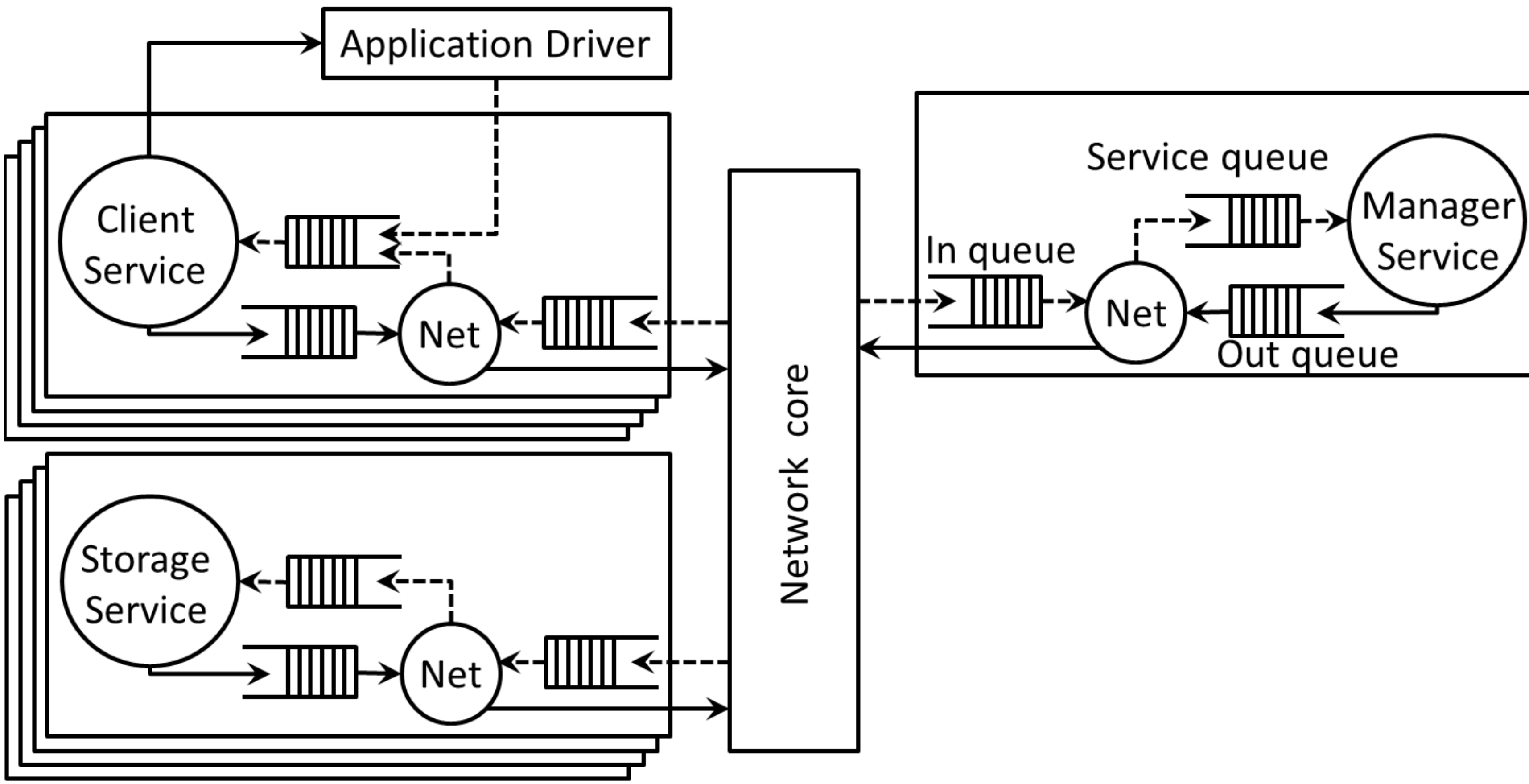
Storage System Model

Focus at high level

- Manager, storage nodes, clients
- No details (e.g., CPU)

Simple seeding

Storage System Model



Seeding the Model

No monitoring changes to the system

- Use coarse level measurements
- Infers services' time

Small deployment

- One instance of each component

Evaluation

Metrics

– Accuracy
– Response time

Workload

– Synthetic benchmark
– An application

Testbed: cluster of 20 machines

An Application

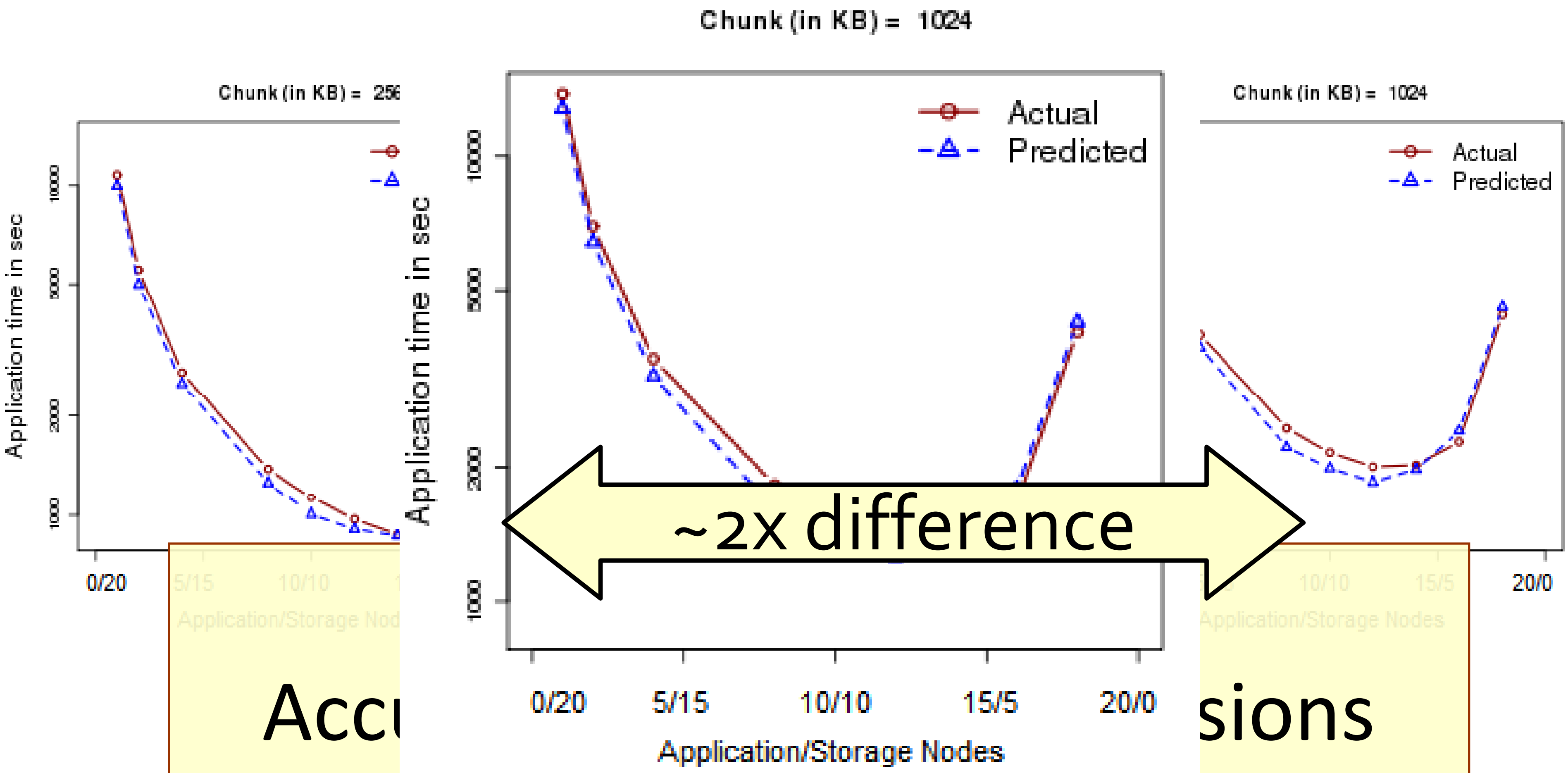
BLAST

- DNA database file
- Several queries (tasks) over the file

Evaluate different parameters

- # of storage nodes, # of clients
- chunk size

BLAST Results



Accu … sions

~3000x less resources

Concluding Remarks

Non-intrusive seeding process/system identification

Low-runtime

Accuracy allows good decision

Predictor can support development

Future Work

Automate parameter exploration

- Prune space by preprocessing input
- Induce placement based on task dependency

Add applications

Increase Scale

Add metrics

- Cost
- Energy is challenging
- Data transferred is accurate

Concluding Remarks

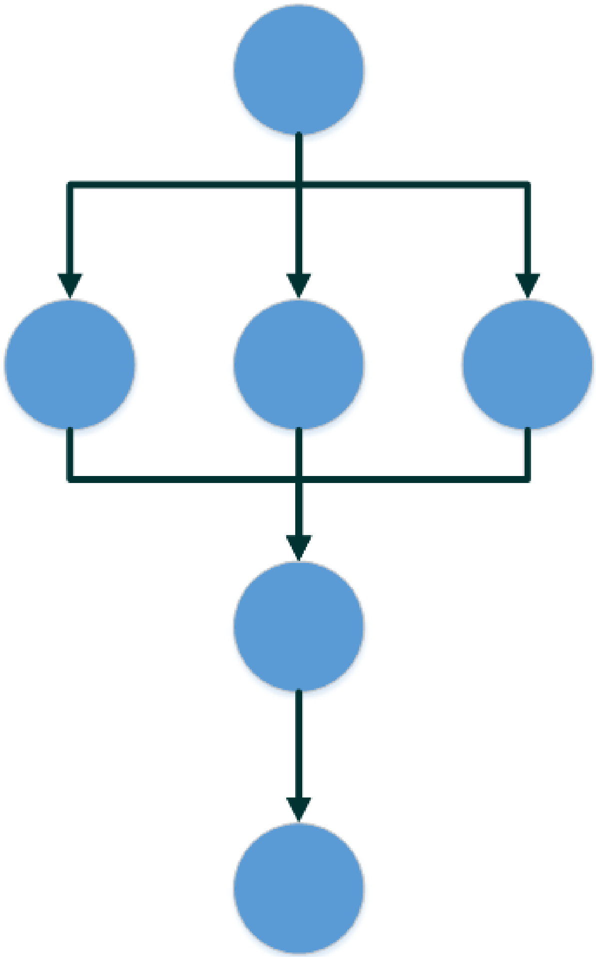
Non-intrusive seeding process/system identification

Low-runtime

Accuracy allows good decision

Predictor can support development

Workflow Applications



DAG represents task-dependency

Scheduler controls dependency and task execution on a cluster

Tasks communicate via files

Synthetic Benchmarks

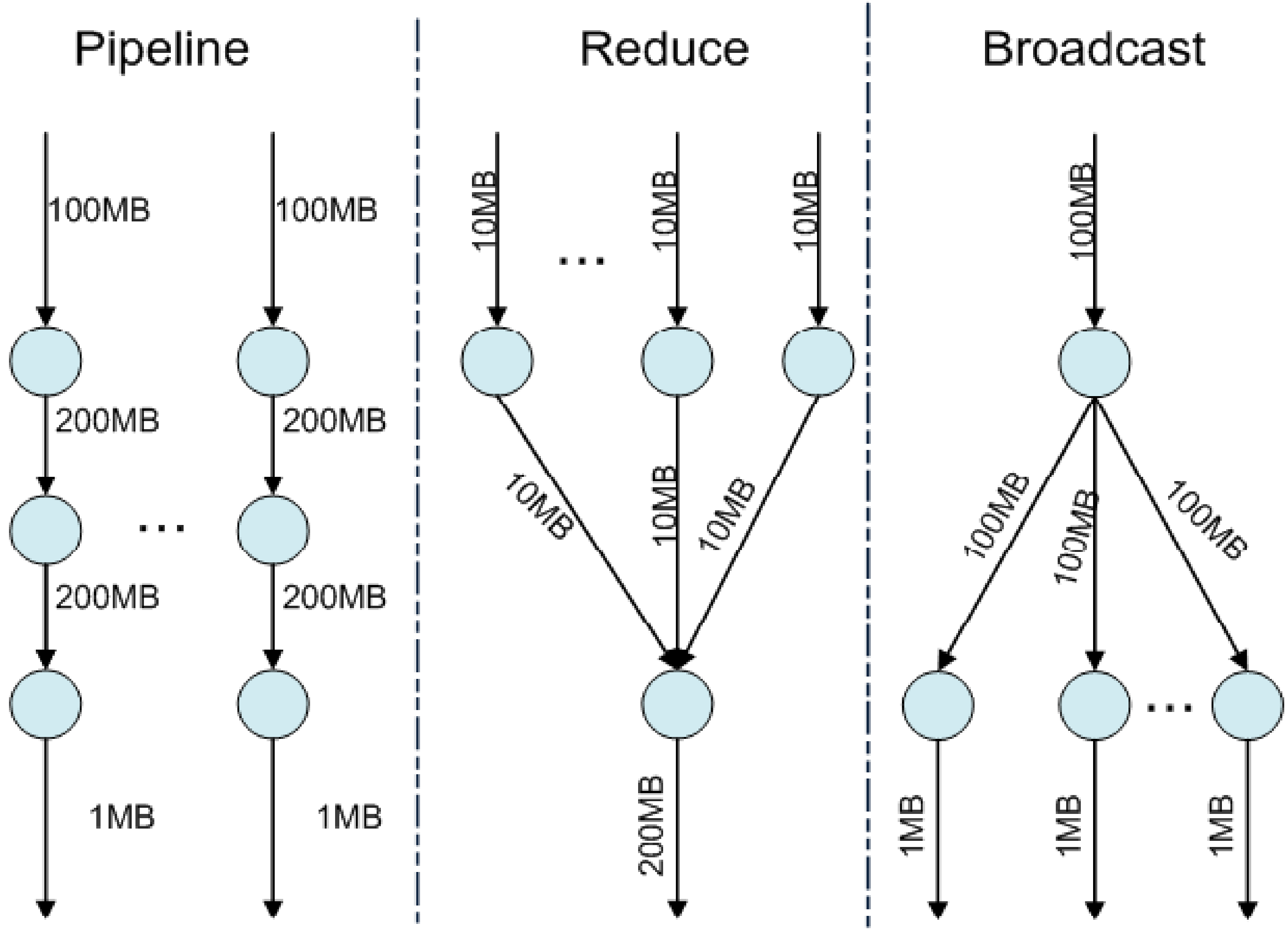
Stress the system

– I/O only, tend to create contention

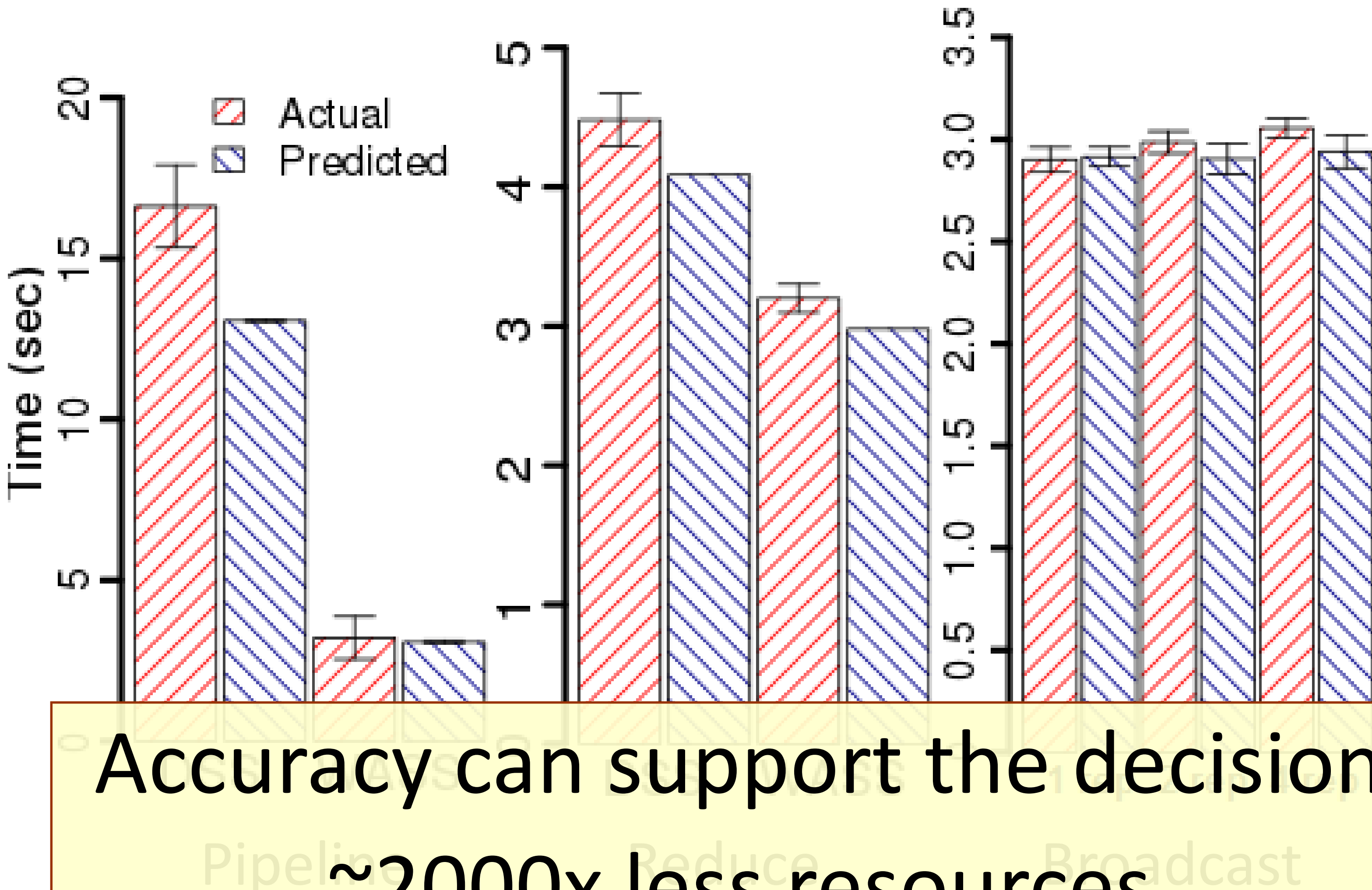
Based workflow patterns

– Evaluate different data placements

Workflow Patterns



Synthetic Benchmarks



Accuracy can support the decision
~2000x less resources

Related Work

- Storage enclosure focused
- Detailed model and seeding (monitoring changes)
- Lack of prediction on the total execution time for workflow applications
- Machine Learning

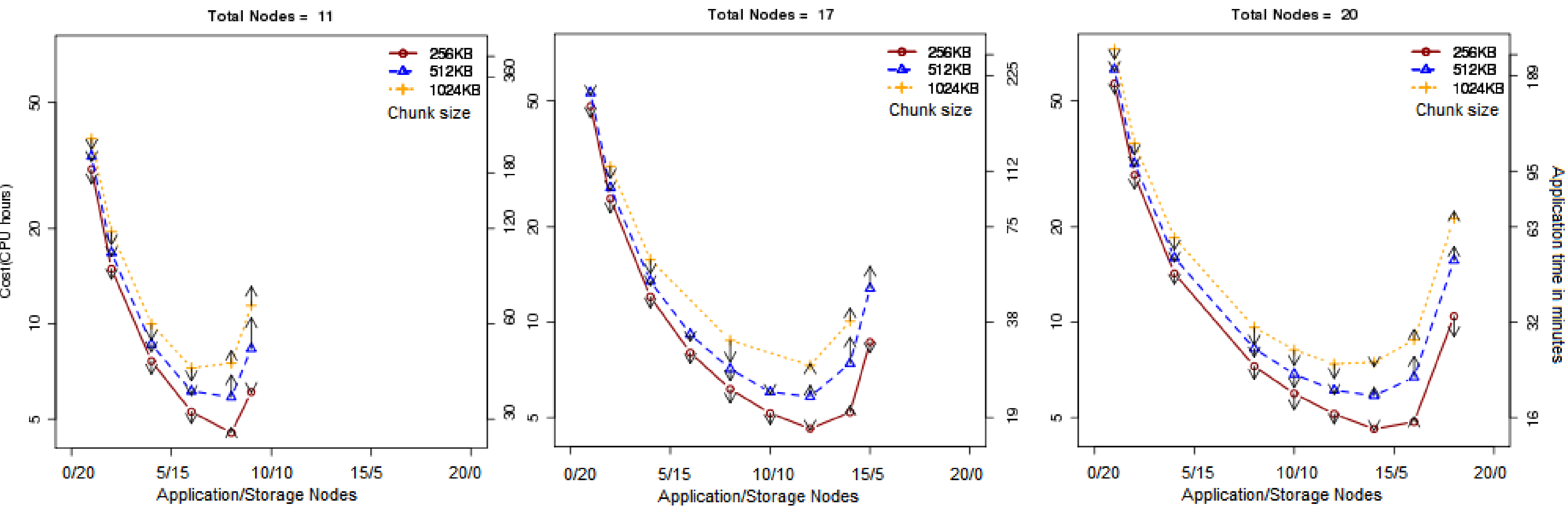
Workload Description

I/O trace per task

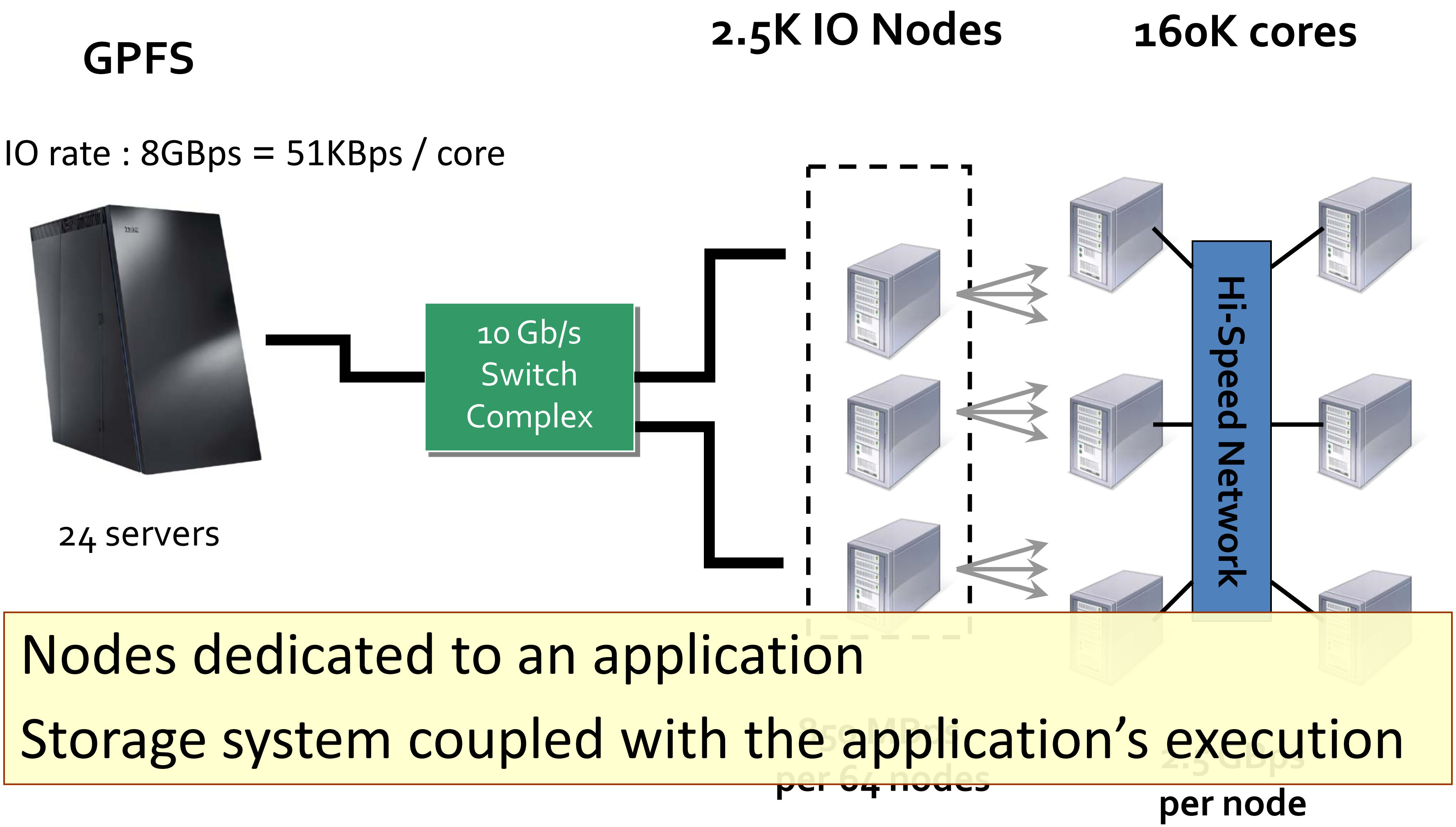
- read, write
- size, offset

Task dependency graph

BLAST: CPU hours



Platform Example – Argonne BlueGene/P



Nodes dedicated to an application

Storage system coupled with the application's execution

Tuning is Hard

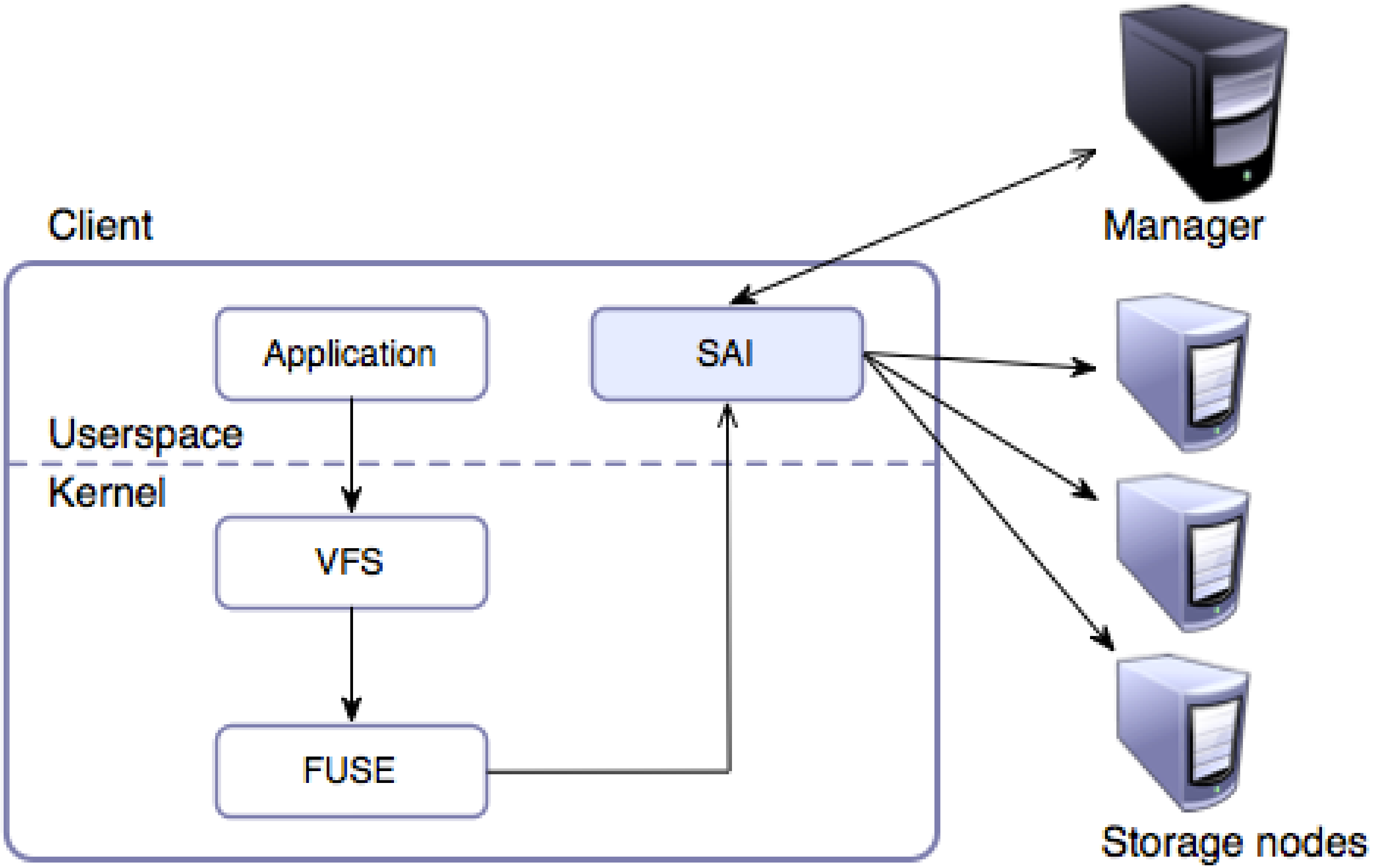
Defining target values can be hard

Understanding distributed systems, application or application’s workloads is complex

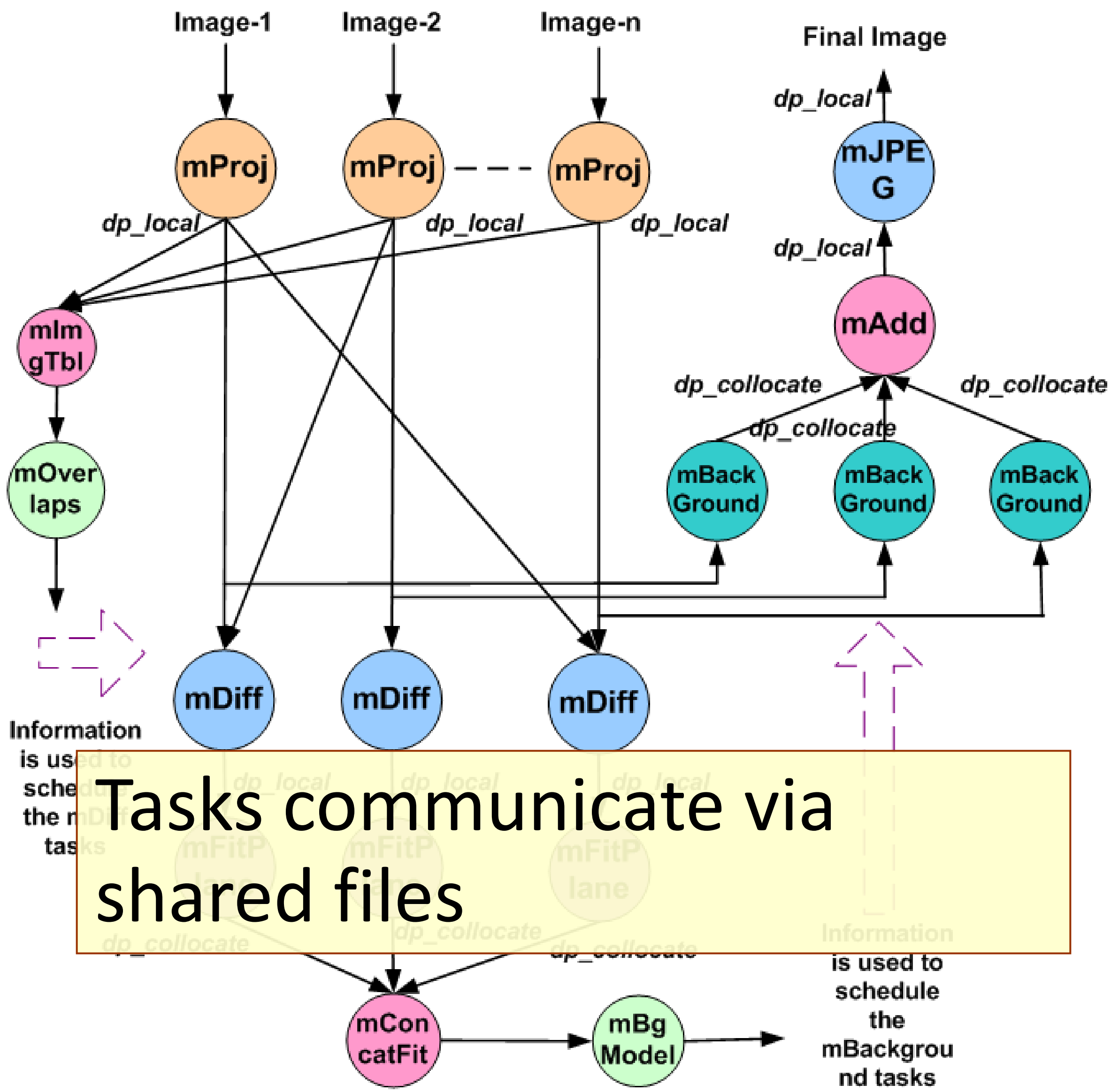
Workload or infrastructure can change

Tuning is time-consuming

Storage System



Montage Example



Tasks communicate via shared files

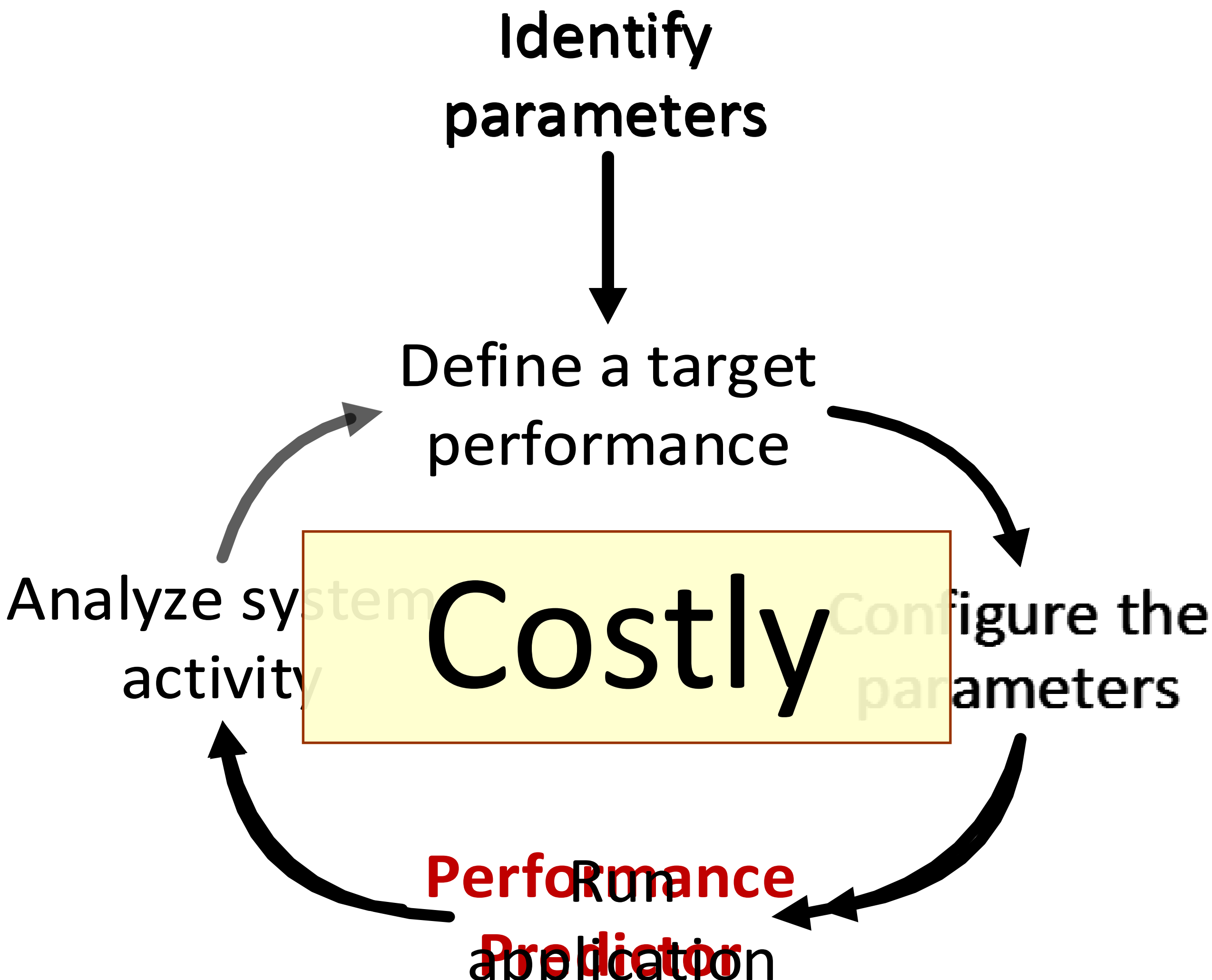
Storage System

Meta-data manager

Storage module

Client module

Configuration Loop



Intermediate Storage System

Storage system co-deployed

Avoid backend storage as bottleneck

Opportunity to configure per application