SCALABLE COMPUTING
SOFTWARE LABORATORY

ILLINOIS INSTITUTE
OF TECHNOLOGY

TABIOS

TAsk-Based I/O System

Anthony Kougkas
akougkas@hawk.iit.edu

PDSW-DISCS'18
Dallas, TX

Monday
November 12th, 2018

Background

Resource Utilization

- Periodic I/O creates idleness between phases
- Over-provisioning due to ignorance or malicious intent
- Exclusive access reservations
- 24/7 always on

Under - Over resource utilization

Interface/API

- Tightly-coupled to certain APIs
- Bound to comply with deployed underlying storage
- Vendor-specific solutions
- Expensive connectors

Isolation, less flexibility and lower productivity.

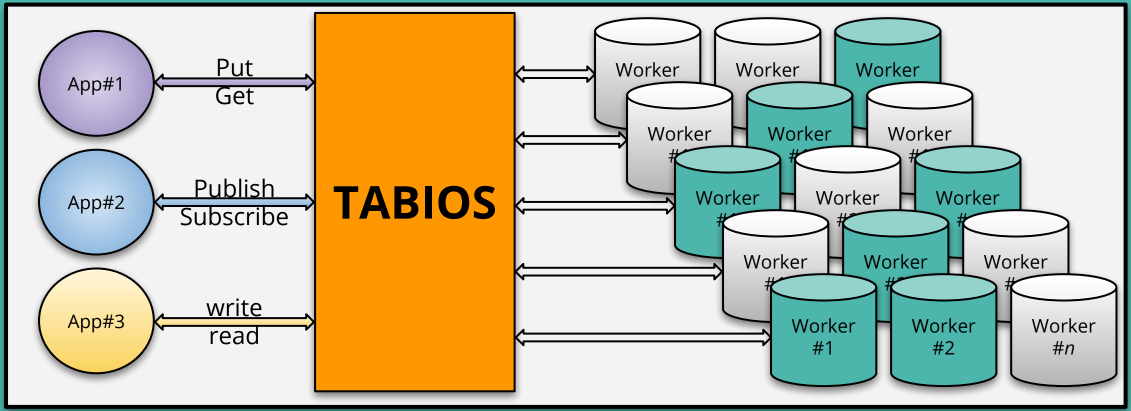
Performance/Energy

- Predefined static storage resources
- No support for power-cap I/O
- No support for tunable performance features

No adaptive concurrency
Resource heterogeneity

Motivation

Challenge 1: Resource Utilization	Challenge 2: Interface/API	Challenge 3: Performance/Energy
How to efficiently utilize I/O resources?	**How to support a wide range of I/O interfaces?**	**How to balance energy - performance?**
• Performance boost	• Productivity	• Energy savings
• System efficiency	• Flexibility	• Superior scalability
• Monetary benefits	• Compatibility	• Increased control



Introducing

TABIOS

An Adaptive, Elastic, Energy-Aware, Distributed Task-based I/O System

- Applications create I/O tasks: **DataTasks**
- A **DataTask** is a placeholder of an I/O job: {operation + pointer to data}
- DataTasks are pushed in a distributed queue
- Workers execute DataTasks independently

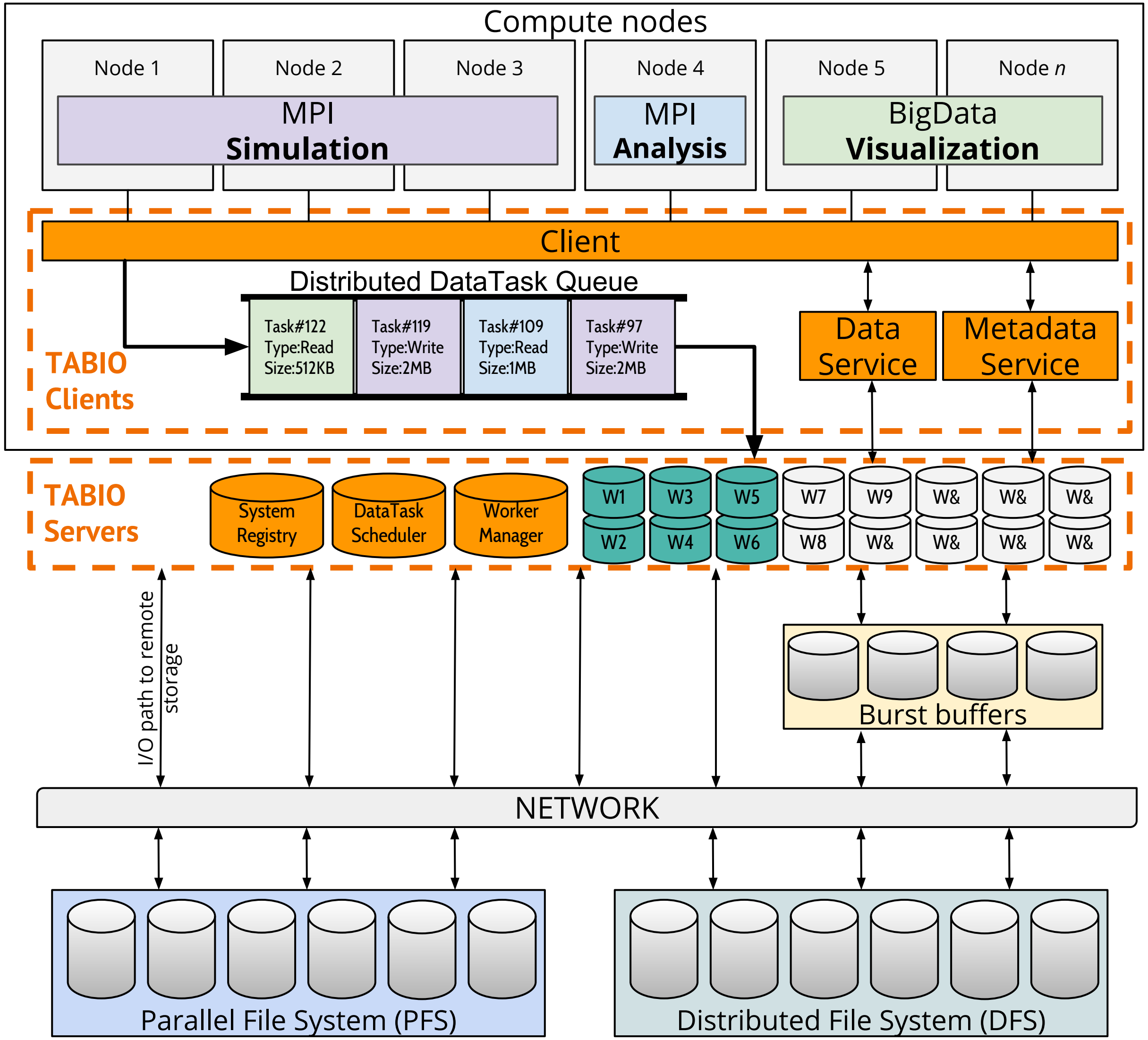
TABIOS

Objectives

- **Storage malleability,**
  - resources can grow/shrink based on the workload.
- **Asynchronous I/O,**
  - operating with mixed media and various configurable storage options.
- **Resource heterogeneity,**
  - supporting a variety of storage resources under the same platform.
- **Data provisioning,**
  - enabling in-situ data analytics and process-to-process data migration.
- **Storage consolidation,**
  - supporting a diverse set of conflicting I/O workloads on a single platform.

TABIOS Architecture

- **Agile**
  - Adaptive to the environment
  - Fully decoupled architecture
- **Software Defined Storage**
  - Offloading computation to servers
  - Data-centric architecture
- **Energy-aware**
  - Power-cap I/O
  - Elastic I/O resources
- **Reactive**
  - Tunable I/O performance - Concurrency control
  - Guaranteed Storage QoS based on job size
- **Flexible**
  - POSIX, MPI-IO, HDF5, REST/Swift, Hadoop
  - Lustre, HDFS,Object Stores



Scheduling Policies

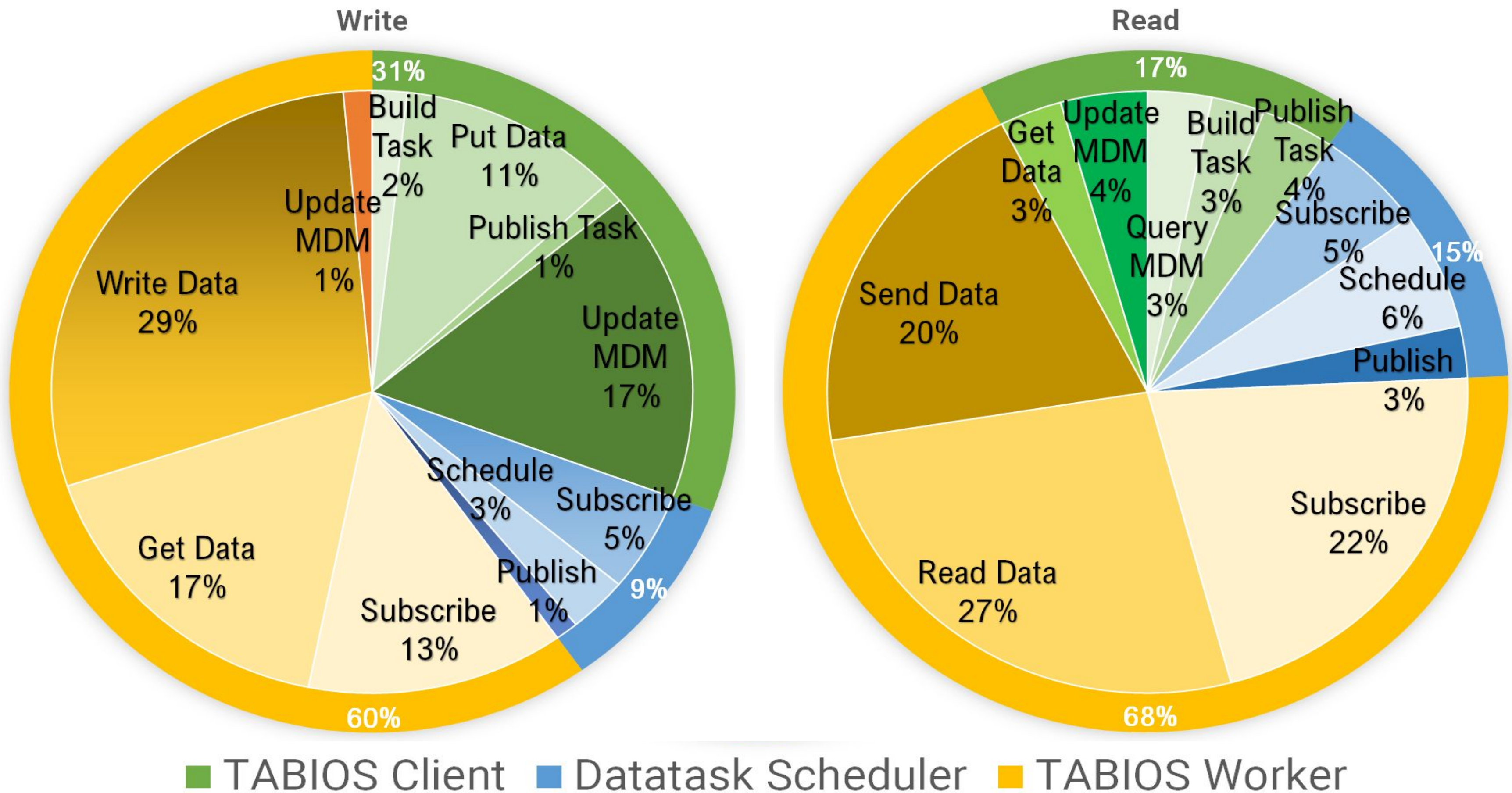
TABIOS

1. Random selection of worker
2. Round Robin
3. Load (queue size)
4. Capacity (size in GB)
5. Latency (access in ms)
6. Locality-aware (file location)

API Example

```
#include <tabios.hpp>
...
Client client = InitClient(ip, port, connConfig);
std::string path = "pvfs2:/data/integers.dat";
std::size_t pos = 0, size = 200MB;
DataTaskSrc src = new DataTaskSrc(path, pos, size);
DataTaskType type = SDS_IN_SITU; //complex type
DataTaskFlag flags = CACHED | MPI_IO; //keep in cache
std::function<int(vector<int>)> fn = FindMedian;
DataTask datatask = client.CreateDataTask(type,src,fn,flags);
Status status =client.IPublishDataTask(datatask);
...            //perform other computations
client.WaitDataTask(&status);
int median = std::static_cast<int>(status.data);
```

Anatomy of Operations



SCALABLE COMPUTING
SOFTWARE LABORATORY

ILLINOIS INSTITUTE
OF TECHNOLOGY

TABIOS

Thank you

Anthony Kougkas
akougkas@hawk.iit.edu
www.akougkas.com