Experiences combining malleability and I/O control mechanisms

David E. Singh and Jesús Carretero
University Carlos III of Madrid (Spain)
I/O challenges

- Scientific applications (climate, genomics, high energy physics, astronomy etc.) ingest, generate, and process increasingly larger data sets
- Future high scale supercomputers need to deal efficiently with huge amounts of data
- Current I/O software stack needs to evolve in order to meet the oncoming scalability challenges

CPU challenges

- Malleable applications can leverage unused computational resources
Data staging challenges

- Concurrent parallel data flows
  - Lack of data staging coordination
    - Among applications
    - Between applications and the system
  - Lack of standards for dynamic monitoring of large scale infrastructures
- Need of coupled control and data mechanisms
- Lack of coordination with the job scheduler
Goals

- Integration of CLARISSE and FlexMPI into a framework
  - New coordination techniques between the applications and the scheduler
  - Application monitoring

- Use of application malleability to enhance the I/O performance:
  - Coordinated use of parallel I/O scheduling and malleability for reducing number of I/O interferences

  **I/O interference**: two or more I/O operations that occur partially or totally at the same time competing for the I/O resources

  - I/O-aware scheduling policies
Novel mechanisms for global data staging coordination to improve:

- Load balance, resilience, parallel I/O scheduling, locality exploitation

Decouple the data and control planes

- Data plane
- Control plane
- Policy

Facilitate the flow of control and data across the I/O stack

<table>
<thead>
<tr>
<th>Policies</th>
<th>Elastic collective I/O, parallel I/O scheduling, resilience, load balancing, etc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control plane</td>
<td>Publish/Subscribe API</td>
</tr>
<tr>
<td>Data plane</td>
<td>Collective I/O, Independent I/O (MPI-IO, put/get APIs), buffering</td>
</tr>
</tbody>
</table>
Global controller

1. subscribe(START_IO)
2. subscribe(FINISH_IO)

Application 1

3. subscribe(GRANT_IO)

4. publish(START_IO, 1)
5. wait(GRANT_IO)

6. publish(START_IO, 2)
7. wait(GRANT_IO)

8. publish(GRANT_IO, 1)

9. publish(FINISH_IO, 1)

10. publish(GRANT_IO, 2)

11. publish(FINISH_IO, 2)

Application 2

3. subscribe(GRANT_IO)

6. publish(START_IO, 2)
7. wait(GRANT_IO)

8. publish(GRANT_IO, 1)

Collective I/O

Collective I/O

Servers

Shared PFS

shuttle
I/O

shuttle
I/O

I/O
I/O scheduling versus no scheduling

Write timeline for two parallel clients with 3840 processes each - No scheduling

Write timeline for two parallel clients with 3840 processes each - FCFS scheduling
FLEX-MPI provides performance-aware malleability capabilities for MPI applications.

Goals:
- Dynamic application reconfiguration
- Automatic load balancing
- Monitoring capabilities
FlexMPI example

![Graph showing the number of processes and time over number of iterations for different classes.]

- **Class C8**
- **Class C6**
- **Class C7**
- **Class C1**
- **Execution time**

Number of processes vs. Number of iteration.

Time (Secs.) vs. Number of iteration.
FlexMPI example

![Chart showing process numbers for different classes over iterations](image)

- Class C8
- Class C6
- Class C7
- Class C1
- Available

Number of processes
- Class C8: 20
- Class C6: 18
- Class C7: 16
- Class C1: 14
- Available: 12

Number of iterations
- 0 to 500: 8
- 500 to 1000: 6
- 1000 to 1500: 4
- 1500 to 2000: 2
- 2000 to 2500: 0
- CLARISSE and FlexMPI integrated at application-level
- CLARISSE and FlexMPI use separate external controllers
- New control logic coordinates both runtimes
Combined framework

Parallel application

Application rank 1

MPI I/O (1) wrapped calls

Clarisse Control Point Logic

(2) MPI calls

FlexMPI Control Point Logic

(3) MPI non-I/O wrapped calls

Application rank 0

MPI I/O (1) wrapped calls

Clarisse Control Point Logic

(2) MPI calls

FlexMPI Control Point Logic

(3) MPI non-I/O wrapped calls

Clarisse Application Controller

FlexMPI Application Controller

MPI library

Clarisse monitoring and coordination

FlexMPI monitoring and coordination

(4) Clarisse application communication

(5) FlexMPI application communication

(6) Clarisse monitoring and coordination

(7) FlexMPI monitoring and coordination
Combined framework

- Application 1, rank 0
  - Clarisse Application Controller
  - FlexMPI Application Controller

- Application 2, rank 0
  - Clarisse Application Controller
  - FlexMPI Application Controller

- Performance modeller
  - Monitor
  - App. performance predictor
  - Offline
  - Runtime

- I/O interference controller
  - I/O interference predictor
  - I/O interference scheduler

- Performance aggregator
  - Monitor

- Resource Manager
  - Global performance predictor
  - Global performance scheduler

- FlexMPI and Clarisse Controllers
  - Clarisse central controller
  - FlexMPI central controller

Flow:
1. Application 1, rank 0
2. Application 2, rank 0
3a. Performance modeller
3b. Performance aggregator
4a. I/O interference controller
4b. Global performance predictor
5a. I/O interference scheduler
5b. Global performance scheduler
6. Global performance controller
7a. FlexMPI and Clarisse Controllers
7b. FlexMPI and Clarisse Controllers
8. Resource Manager
I/O interference

- **Time**
- **Partial I/O conflict**
- **Application 1**
- **Application 2**
- **Total I/O Conflict**
- **Period 1**
- **Period 2**
- **Inria Bordeaux Sud-Ouest**
Solutions:

- **I/O scheduling**: blocks one I/O operation using publish-subscribe support.
Avoiding I/O conflicts with Clarisse + FlexMPI

- Leverage malleability for changing the I/O time stamp
- Prediction of the I/O interference
- **Phase shifting**
  - Leverage malleability for changing the I/O access time (phase)
  - Temporary use of computational resources

---

**Diagram:**

- **Application 1**
  - Reconfiguration 1: increase $\Delta P$ processes
  - Reconfiguration 2: decrease $\Delta P$ processes
  - Shift $\Delta t$ seconds

- **Application 2**

---

**References:**

- Inria Bordeaux Sud-Ouest
Phase coupling

- Leverage malleability for changing the I/O period
- Long-term use of computational resources

**I/O interference avoidance**

**Application 1**

<table>
<thead>
<tr>
<th>T=0</th>
<th>Time</th>
</tr>
</thead>
</table>

**Application 2**

<table>
<thead>
<tr>
<th>T=0</th>
<th>Time</th>
</tr>
</thead>
</table>

Reconfiguration 1: increase $\Delta P$ processes

New period
Results

- Bebop cluster
- Two applications
- 150 and 128 processes
Results

- Bebop cluster
- Two applications
- 150 and 128 processes
Results

- Bebop cluster
- Two applications
- 150 and 128 processes
Results

- Two identical applications executed at the same time.
- 64 processes
Results
- Two different applications executed at the same time.
- 64 and 50 processes

I/O time

Total delay
I/O-aware scheduling
We only consider running applications

Assign the available processors to the running applications

Two baseline schedulers:
- Fair
- Speedup-based
- I/O-aware
I/O-aware scheduling
I/O-aware scheduling

![Graph showing comparison of delay times for different scheduling methods across different number of processes.](image-url)
Future work

- Extended Clarisse and FlexMPI coordination
- Automated learning techniques for I/O scheduling
- Integration with a system-wide monitoring tool
- Application modelling