GMF: A Framework for Module Management on the Grid

project overview

Peter Praxmarer

praxmarer@gup.jku.at

GUP Linz

Johannes Kepler Universität Linz

Austria

Agenda

1. Introduction
2. General Overview of Grid Management Framework
3. Module Overview
4. Current Status
5. Conclusion and Future Work

Introduction

- Utilization of grid environments requires parallel and distributed programming to solve single, but large-scale problems.
- Workload of different modules is distributed over various heterogeneous grid resources, which are interconnected as pipeline or graph structure.
- The **G**rid **M**anagement **F**ramework provides a basic framework to encapsulate common tasks necessary to **create** and **control** a module graph.

General Overview of GMF (1)

- Uses the Globus Toolkit `http://www.globus.org`
- Provides an object-oriented interface to parts of the Globus Toolkit:
  - GlobusCommon
  - GlobusIO
  - GlobusFTP-Client
  - GlobusGram-Client
- Performs error-handling on any Globus-Function
- Default error-handling provided by GMF can be overwritten on a per operation basis

General Overview of GMF (2)

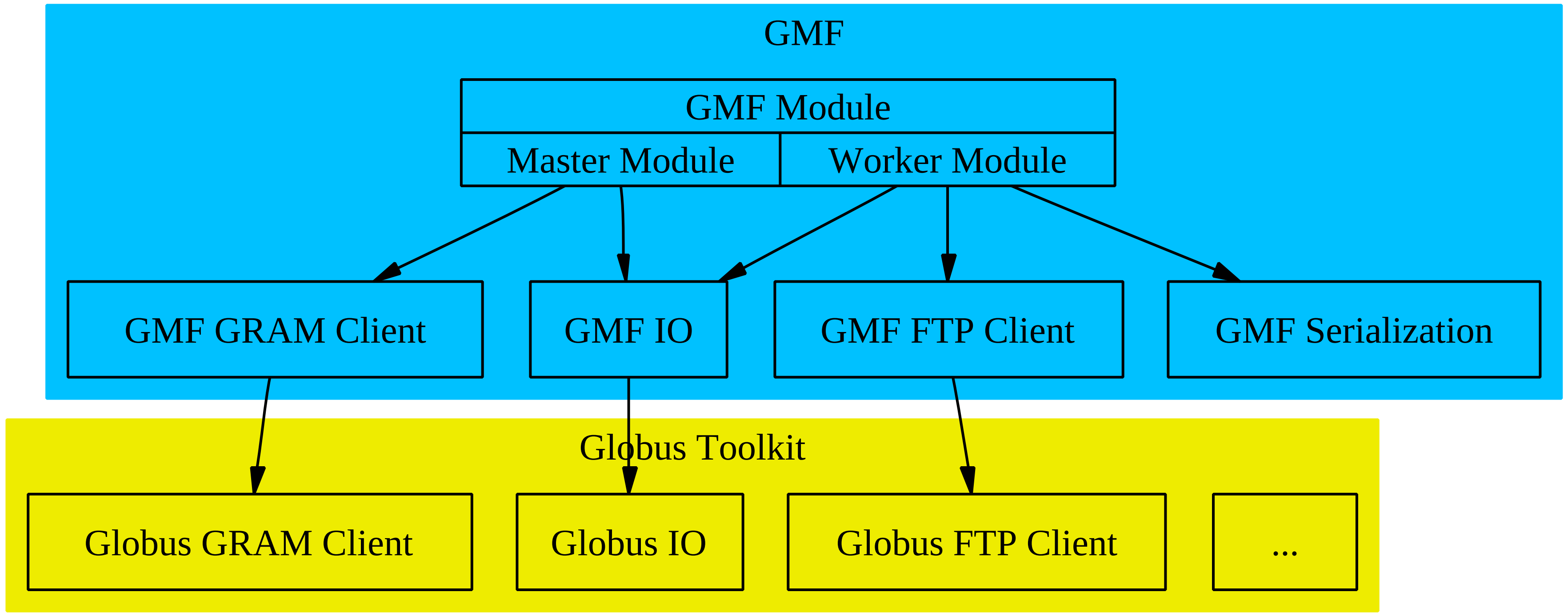


Figure 1: GMF structure

Enhancements in GMF IO

- Provide a simplified interface to GlobusIO without losing its flexibility.
- Enhances GlobusIO with

MultiplexedConnection Data is split into chunks and sent over multiple TCP-Connections.
$\rightarrow$ Aims at increasing throughput

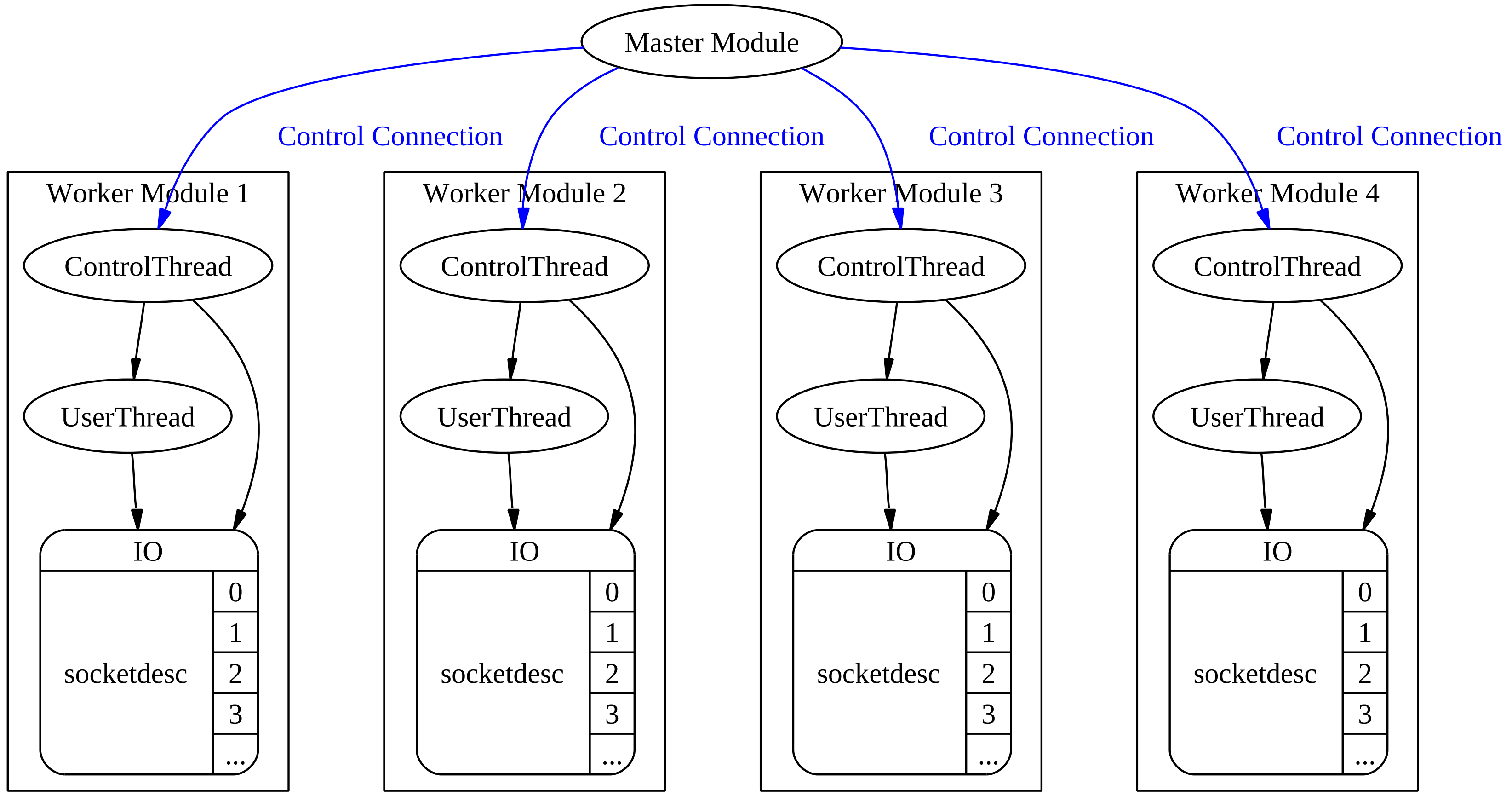
BufferedIO Mode Send/Receive data in a separate thread.
$\rightarrow$ Calculation overlaps Communication

GMF Module Tasks

- **Master Module**
  - Instantiate Worker Modules
  - Interconnect them
  - Start them
  - Migrate them (if module supports this operation)
  - Stop them
- **Worker Module**
  - Is instantiated by a Master Module
  - Performs the application task
  - Performs checkpointing (not mandatory)

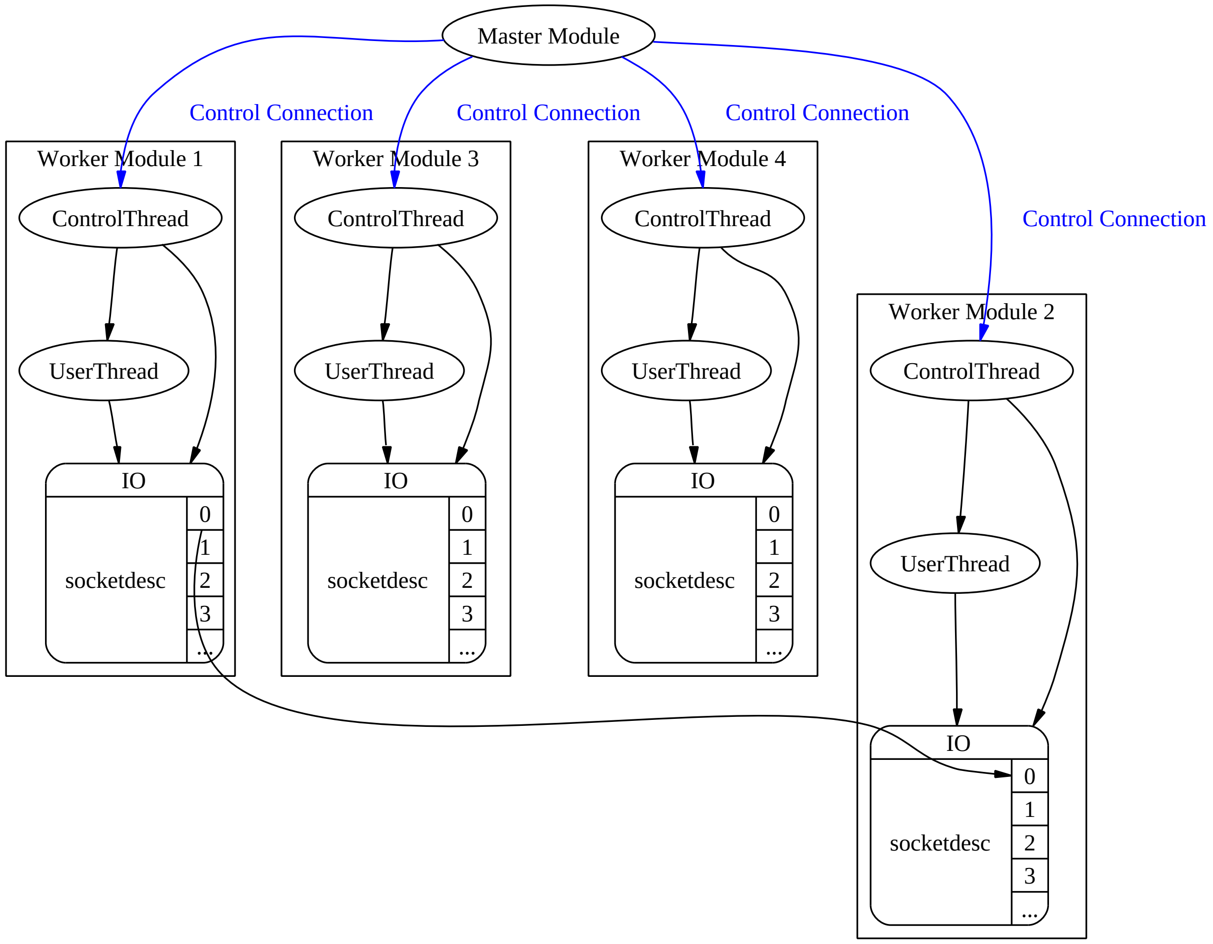
Scenario: Create a Module Graph (1)

- Create 4 WorkerModules



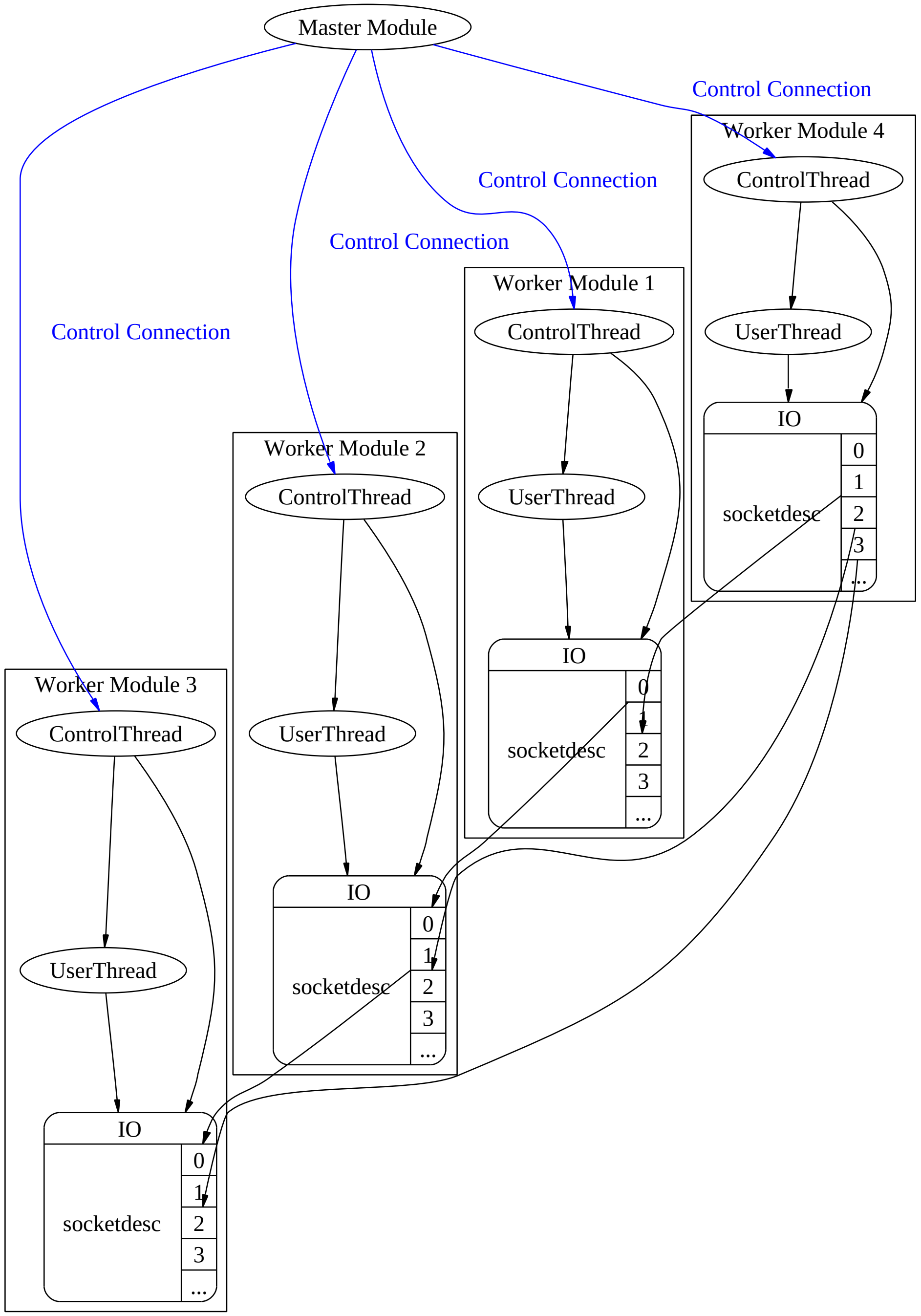
Scenario: Create a Module Graph (2)

- Connect *Worker 1, socketdesc 0* with *Worker 2, socketdesc 0*



Scenario: Create a Module Graph (3-6)

- Finally



Module Migration: Idea

- Input data is delivered in independent records (e.g. simulation data).
- Module performs an operation and passes it on.
- Repeated for each record.

$\rightarrow$ If the state of the module can be serialized then it is possible to migrate the module before it starts the next operation.

GMF Serialization

- Builds a DataGraph
- (De-)Serializes the Graph
- Cares for cycles
- Copes with dynamic data structures
- Architecture independent
- **Drawback:** Requires user input (cf. Java Serialization)

GMF Serialzation: Requirements

Non-intrusive Allows serialization of objects without need for code change in existing classes.
$\rightarrow$ Use the C++ - template mechanism, since it allows parametric programming.

Architecture independent Provides serialization of objects across heterogeneous platforms.

Module Migration (1)

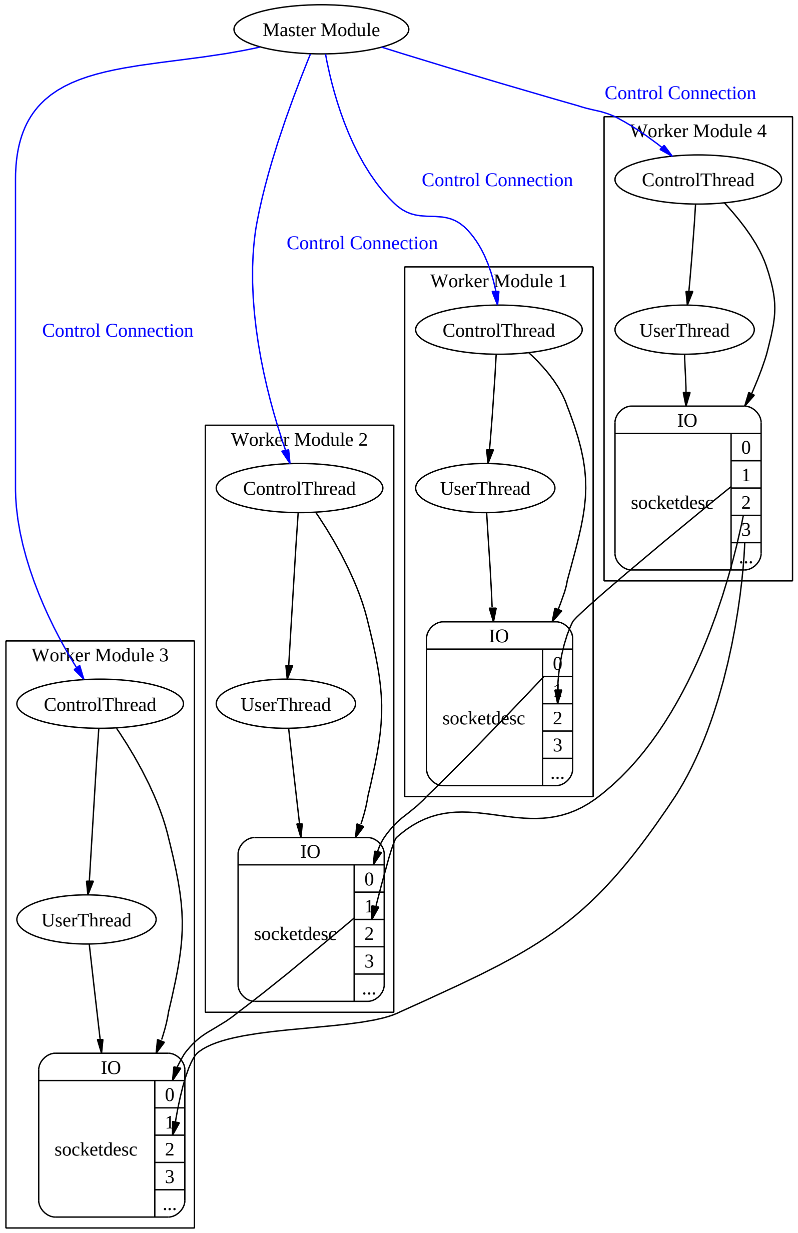
1. User implements application as a function object ('functor' in C++) that is called at least once for each record.
2. Provides a description on how the functor is serialized.
3. GMF applies the functor to each record until a migration is requested.
4. If the worker module is requested to migrate it suspends operation.

Module Migration (2)

5. Master brings all connections of the worker down (care for in-transit messages).
6. Master creates a new module, brings all connections up again and transfers the state to the new module.
7. The new module resumes, the old one is discarded.

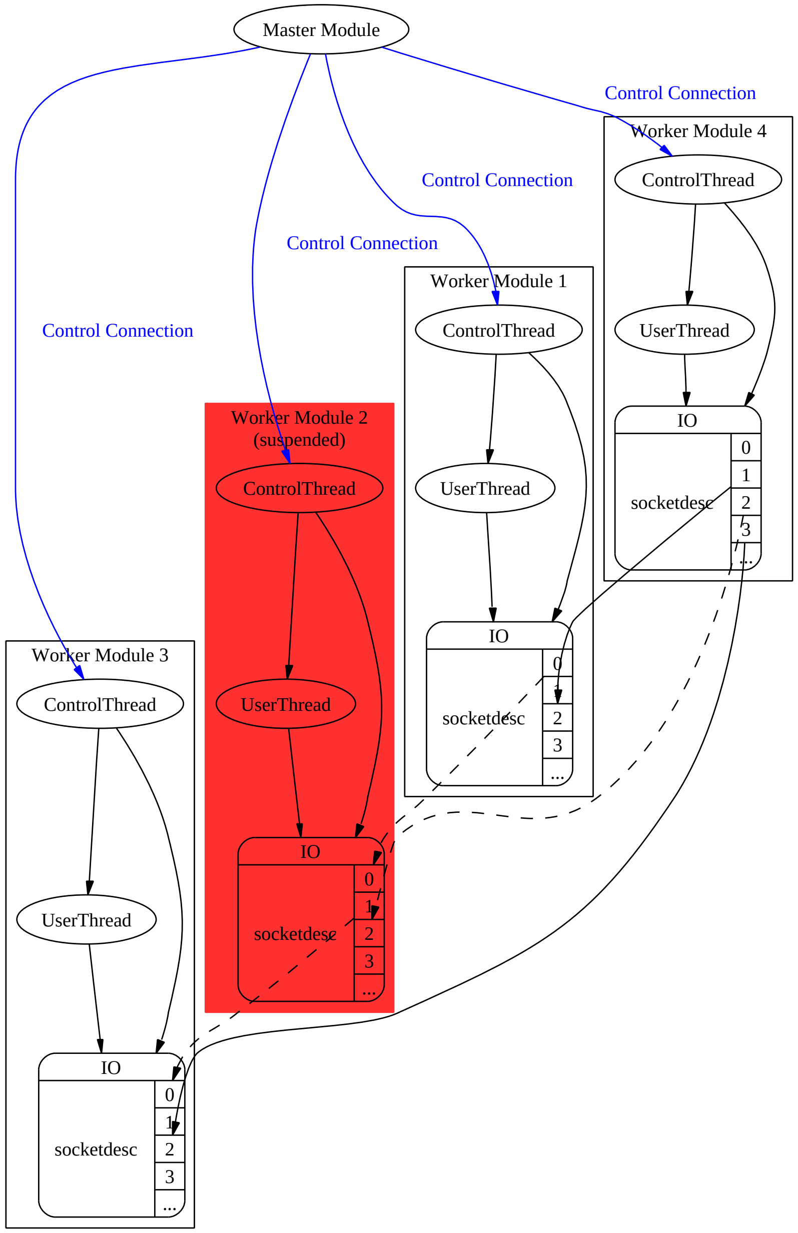
Scenario: Module Migration (1)

- Initial state: all modules are up and running.



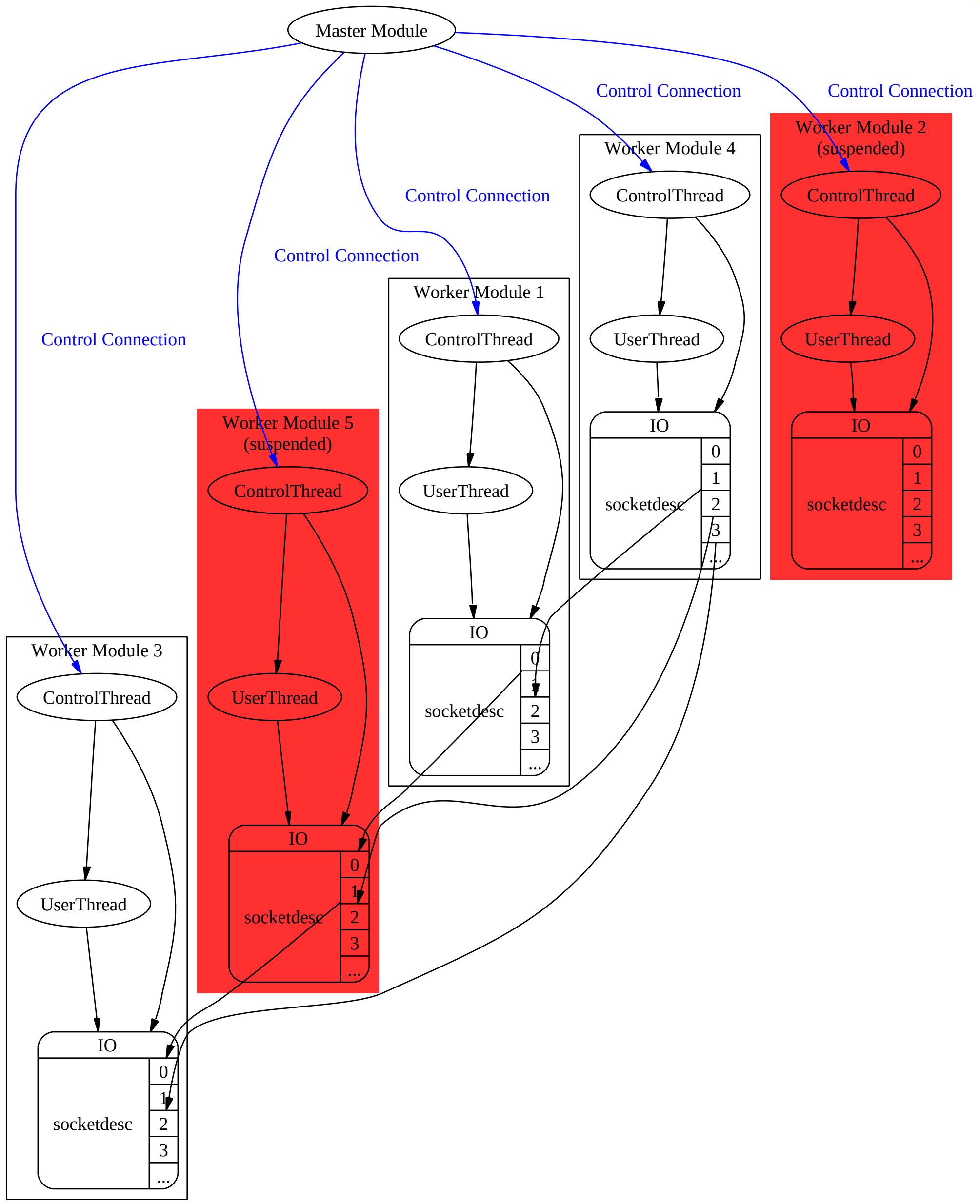
Scenario: Module Migration (2)

- Suspend Module 2
- Shutdown all connections from/to Module 2



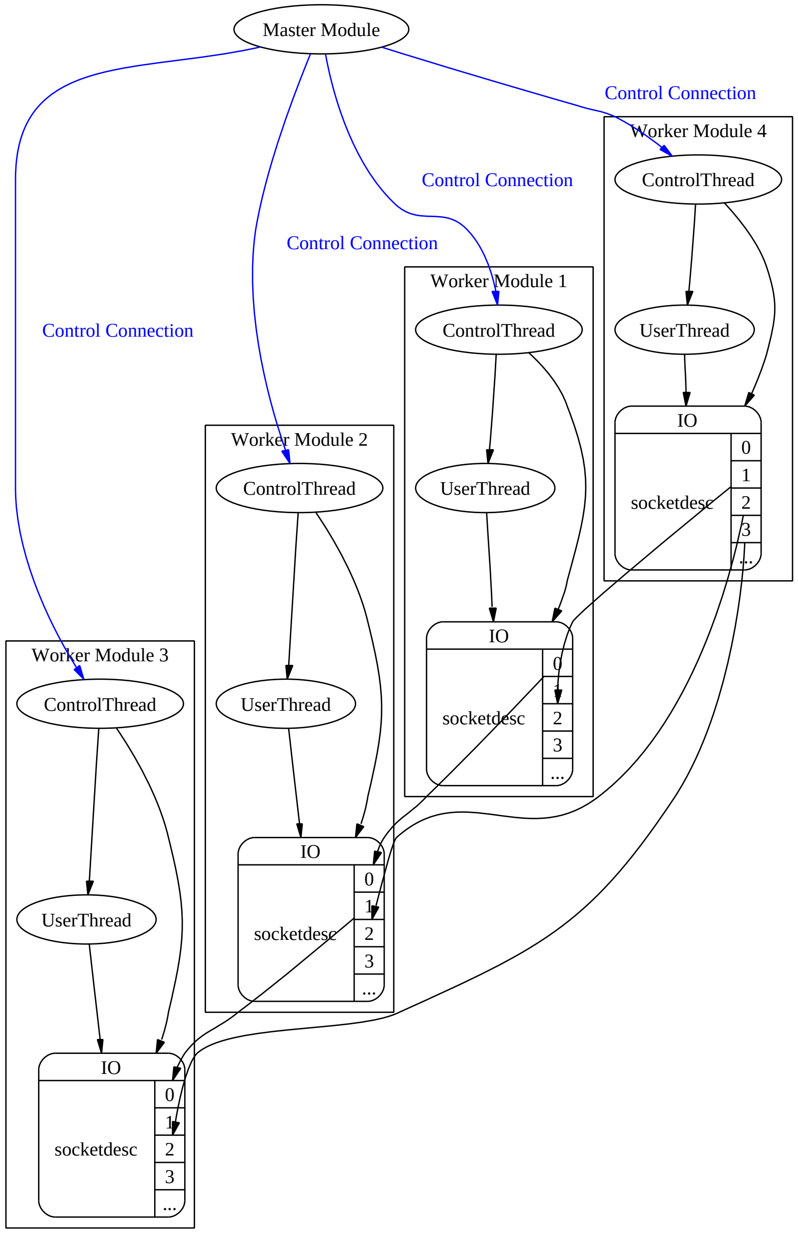
Scenario: Module Migration (3)

- Create the new Worker
- Restore previous connections



Scenario: Module Migration (4)

- Transfer state
- Resume the new module, drop the old one.



Current Status

- Major parts of GMF are implemented including
  - GMF IO, GMF FTP, GMF GRAM
  - GMF Module
- Work in Progress
  - Serialization FW
  - Testing
  - Documentation

Conclusion and Future Work

- GMF provides ...
  - an abstraction to parts of the Globus Toolkit
  - a Module framework that cares for common tasks to setup and reconfigure a module graph.
  - a Serialization FW that allows serialization of objects across multiple platforms.
- Future Work
  - Integral part of the **G**rid **V**isualization **K**ernel `http://www.gup.uni-linz.ac.at/gvk/`

GMF Serialization: Simple Example (1)

```
#include <vector>

struct TestSerialization {
  int a;
  char ch;
  int *ptr;
  int **pptr;
  int ***ppptr;

  TestSerialization() : a(1), ch(’a’),
     ptr(&a), pptr(&ptr), ppptr(&pptr) {}

  // methods
  // ...
};
```

GMF Serialization: Simple Example (2)

```
namespace GMF {
  // provide serialization template
  template<>
  class Node<TestSerialization> :
        public AbstractNode {
    TestSerialization *val;
  public:
    void staticMembers(DataTreeBuilder*
                             builder) {
      builder->add(val->a);
      builder->add(val->ch);
      builder->add(val->ptr);
      builder->add(val->pptr);
      builder->add(val->ppptr);
    }
  };
```